TD-M(PC)^2: Improving Temporal Difference MPC Through Policy Constraint

Haotian Lin vlin3@andrew.cmu.edu
Pengcheng Wang wangpc@berkeley.edu
Jeff Schneider jeff4@andrew.cmu.edu
Guanya Shi guanyas@andrew.cmu.edu

Abstract

Model-based reinforcement learning algorithms that combine model-based planning and learned value/policy prior have gained significant recognition for their high data efficiency and superior performance in continuous control. However, we discover that existing methods that rely on standard SAC-style policy iteration for value learning, directly using data generated by the planner, often result in *persistent value overestimation*. Through theoretical analysis and experiments, we argue that this issue is deeply rooted in the structural policy mismatch between the data generation policy that is always bootstrapped by the planner and the learned policy prior. To mitigate such a mismatch in a minimalist way, we propose a policy regularization term reducing out-of-distribution (OOD) queries, thereby improving value learning. Our method involves minimum changes on top of existing frameworks and requires no additional computation. Extensive experiments demonstrate that the proposed approach improves performance over baselines such as TD-MPC2 by large margins, particularly in 61-DoF humanoid tasks. View qualitative results at webpage.

1 Introduction

Model-based reinforcement learning (MBRL) is a promising approach that leverages a predictive world model to solve sequential decision-making problems [25].

MBRL, in general, learns a dynamics model that enables the simulation of future trajectories and leverages the model for either policy learning or online planning. Due to its ability to exploit the structure of the environment, MBRL is fundamentally more sampling efficient than model-free algorithms both theoretically [37, 17] and empirically in diverse domains such as robotics control [24] and autonomous driving [18].

Recent studies have drawn attention to the combination of value learning and planning for test-time optimization, which could be referred to as temporal difference learning for model predictive control (TD-MPC) [15]. Different from Dyna-like MBRL [37, 13] that treats the learned model as a simulator for model-free policy learning, this line of works [3, 35, 15] leverage sampling-based MPC for planning, and model-free RL to acquire policy or value prior. Such a combination significantly reduces the planning burden for performant short-horizon planning and enhances data efficiency by rapid state space coverage through trajectory space exploring [26].

Despite their impressive performance in continuous control benchmarks [14], we observed that standard policy iteration in TD-MPC implementation leads to *persistent value overestimation*. It is also empirically observed that the performance of TD-MPC2 is far from satisfactory at some high-dimensional locomotion tasks [34]. This phenomenon is closely connected to, yet distinct from, the well-known overestimation bias arising from function approximation errors and error accumulation in temporal difference learning [40, 38, 7]. More precisely, we identify the underlying issue as *policy mismatch*. The behavior policy generated by the MPC planner governs data collection, creating a buffered data distribution that does not directly align with the learned value or policy prior. Consequently, errors in the value function are not adequately represented by the behavior policy, making these errors unnoticed and uncorrected during training. Through theoretical analysis, we further show that this structural issue causes such mismatch to accumulate with approximation errors, which are subsequently amplified by the planner. Notably, this challenge resembles the distributional shift problem in offline RL [23], where the behavior policy never aligned with the current policy [8]. These findings highlight the necessity of conservative exploitation strategies when combining planners with policy improvement to address this structural deficiency.

Concretely, we propose a simple yet effective approach that allows a planning-based MBRL algorithm to better exploit data collected from online planning. The resulting algorithm, Temporal Difference Learning for Model Predictive Control with Policy Constraint, TD-M(PC)^2, acquires value and policy prior through distribution-constrained policy iteration. Such procedure extracts performant policy from the online buffer while reducing *out-of-distribution* query that leads to approximation error. TD-M(PC)^2 is easy to implement, and it only requires a minimalist modification

to the *state-of-the-art* TD-MPC2 framework with less than ten lines of code. Without introducing additional computational budget or need for environment-specific hyperparameter tuning, it seamlessly inherits desirable features in the previous pipeline and consistently improves its performance for high-dimensional continuous control problems on both DM control and HumanoidBench [34], especially in complex 61-DoF locomotion tasks.

2 Related Work

Model-based RL. The core of model-based reinforcement learning is how to leverage the world model to recover a performant policy. Dyna-Q [37] first introduced the idea of using simulated rollouts from a learned model to augment real-world experience for policy optimization. MBPO [17] further provides a theoretical guarantee of monotonic policy improvement and promotes short model-generated rollouts. Dreamer [19, 12, 13] optimizes policies entirely in imagination, leveraging latent world models for high-dimensional tasks like visual control. These methods are computationally efficient as they decouple model rollouts from online decisions, but they can suffer from model errors over long horizons.

Planning-based approaches use the world model for online decision-making by optimizing actions directly through simulated trajectories. PlaNet [11] employs a latent dynamics model with trajectory optimization in latent space, while PETS [5] utilizes an ensemble of probabilistic models and the Cross-Entropy Method (CEM) for sampling-based optimization. These methods are highly adaptive to online changes and precise for short-horizon tasks but face challenges in scaling to tasks with high-dimensional states or action spaces due to the computational cost of rollouts during execution.

Temporal-Difference Model Predictive Control. Recent advances aim to balance scalability and adaptability by integrating strengths from both paradigms. [3, 35, 15, 42] adopt a temporal-difference (TD) learning framework that combines with model predictive control, illustrating how a value-based learning signal can mitigate the need for handcrafted cost functions and long-horizon planning. Building upon this idea, TD-MPC2 [14] is able to learn scalable, robust world models tailored for continuous control tasks, effectively reducing compounding modeling errors and improving planning stability. These advances highlight how embedding temporal-difference learning within the MPC paradigm can significantly enhance control strategies' flexibility, sample efficiency, and robustness in high-dimensional continuous domains.

Off-policy Learning with Policy Constraint Distributional mismatch is a long-standing challenge in off-policy learning. Standard off-policy algorithms are highly sensitive to distributional shifts, as bootstrapping errors can compound over time, leading to instability and poor generalization [21]. Recent studies in offline RL have taken a huge leap in enabling policy learning from off-policy demonstrations. To enforce distributional constraints, [21, 6] incorporate policy regularization, while [8, 32] mitigate OOD queries through importance sampling. Alternatively, [20, 9] adopt in-sample learning techniques to implicitly recover a policy from observed data, bypassing direct constraints on action selection. The off-policy issue is also critical for planning-based MBRL that leverages a policy or value prior. [35] introduced an approach that marries off-policy learning with online planning by actor regularization control, introducing conservatism into the planner. In comparison, our method addresses such constraints on the policy prior without compromising the planner. [1] achieves a similar planning process by learning a behavior cloning (BC) policy and corresponding value function. However, due to its pure offline nature, all the components can be considered to originate from the distribution of the behavior policy.

3 Planning with Value and Policy Prior

3.1 Preliminaries

Continuous control problems can be defined as a Markov decision process (MDP) [38] represented by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \rho, \rho_0, r, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition of states $\rho(s'|s,a)$, initial state distribution ρ_0, reward function $r(s,a)$ and discount factor $\gamma \in (0,1]$. The objective of the agent is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes discounted cumulative reward:

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

where τ^π is a trajectory sampled by executing π.

Model-based RL leverages the internal structure of the MDP by learning the dynamics model and planning through it rather than completely relying on the value function estimator. A refined closed-loop control policy can be acquired through local trajectory optimization methods such as Model Predictive Path Integral (MPPI) [41]. Action sequences of length H are sampled and evaluated by rolling out *latent trajectories*. At each step, parameters μ^* and σ^* of a multivariate Gaussian are computed to maximize the expected return:

$$\begin{aligned} \mu^*, \sigma^* &= \arg\max_{\mu,\sigma} \mathbb{E}_{(a_t, a_{t+1}, \ldots, a_{t+H}) \sim \mathcal{N}(\mu, \sigma^2)}[G(s_t)] \\ G(s_t) &= \sum_{h=t}^{H-1} \gamma^h r(z_h, a_h) + \gamma^H \hat{V}(z_{t+H}) \\ \text{s.t.} \quad z_{t+1} &= d(z_t, a_t) \end{aligned} \tag{2}$$

where $\mu, \sigma \in \mathbb{R}^{H\times m}$. Note that only the first action $a_t \sim \mathcal{N}(\mu_t^*, {\sigma_t^*}^2)$ is executed, and another optimization problem is solved at time step $t+1$ (i.e., receding horizon). We denote such H-step lookahead policy, which solves (2) at every time step, as π_H. π_H leverages both the planner and the value function $\hat{V}$.

3.2 Basic Pipeline

Combining planning and temporal-difference learning has proven to be an effective way to reduce planning horizon and improve data efficiency. A widely recognized pipeline from TD-MPC2 is to jointly learn encoder $z = h(s, e)$, latent dynamics $z' = d(z, a, e)$, reward, $\hat{r} = R(z, a, e)$, nominal policy $\hat{a} = \pi(z, e)$, and action value function $\hat{q} = \hat{Q}(z, a, e) \approx Q^\pi(z, a, e)$. where $\mathbf{z}$ is the latent state representation and e is a learnable task embedding for training multitask world models.

Specifically, h, d, R, Q are jointly trained through the following loss:

$$\mathcal{L} \doteq \mathop{\mathbb{E}}_{(s,a,r,s')_{0:H}} \Big[\sum_{t=0}^{H} \gamma^t \Big(\|d(z_t, a_t, e) - sg(h(s'_t))\|_2^2 + CE(\hat{r}_t, r_t) + CE(\hat{q}_t, q_t) \Big) \Big], \tag{3}$$

where the targets q_t is generated by bootstrapping nominal policy π (refer to (9)), and sg is the stop-grad operator. π is a stochastic Tanh-Gaussian policy trained with the maximum Q objective in a model-free manner. During inference, π selects action at terminal state, resulting in value estimation as $\hat{V}(z_{t+H}) = \mathbb{E}_{a_{t+H}\sim\pi(z_{t+H},e)}[\hat{Q}(z_{t+H}, a_{t+H}, e)]$, which is appended to the end of sampled trajectories in (2).

3.3 Value Overestimation

Planning with a value and policy prior ideally requires the value function to be close to the global optimal V^*. In TD-MPC2, value estimation is derived from approximate policy iteration (API), like regular off-policy learning. Despite prior work suggesting that nominal policy acquired from SAC-style policy learning is sufficiently expressive for value training [15], we find that value approximation error could still be significant for complex high-dimensional locomotion tasks. Figure 1 provides a clear demonstration on value approximation error $\mathbb{E}_{\rho_0}[\hat{V} - V^\pi]$ in four distinct control tasks from DMControl [39] and HumanoidBench [34]: `Hopper-Stand` ($\mathcal{A} \in \mathbb{R}^4$, 15% error), `Dog-Trot` ($\mathcal{A} \in \mathbb{R}^{36}$, 231% error), `h1hand-run-v0` ($\mathcal{A} \in \mathbb{R}^{61}$, 2159% error), `h1hand-slide-v0` ($\mathcal{A} \in \mathbb{R}^{61}$, 746% error). While overestimation bias is within an acceptable range in low-dimensional tasks, it is incredibly large in high-dimensional tasks and does not tend to converge to ground truth. This persistent value overestimation is also reflected in performance. According to benchmarking results [34], TD-MPC2 failed to acquire performant policy in many high-dimensional humanoid locomotion tasks.

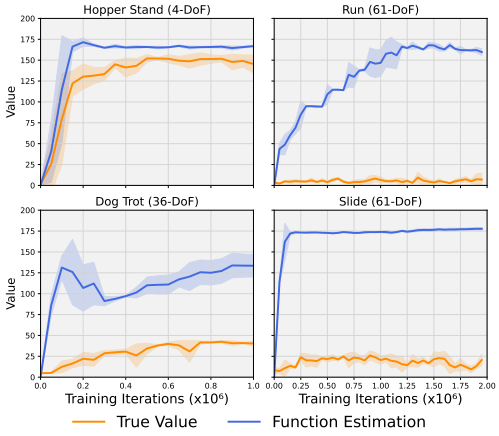


Figure 1. Value approximation error for TD-MPC2. The true value is estimated using the average discounted return over 100 episodes following the nominal policy π; Function estimation is obtained by $\hat{V} = \mathbb{E}_\pi[\hat{Q}]$. The results are averaged over three seeds for an unbiased assessment.

To understand how the approximation error influences the overall performance of the H-step look-ahead policy, we present Theorem 3.1 that is adopted from Theorem 1 in LOOP [35]. Detailed proof can be found in Appendix A.5.

Theorem 3.1 (H-step Policy Suboptimality)**.** *Assume the nominal policy π_k is acquired through approximation policy iteration (API) and the resulting planner policy at k-th iteration is $\pi_{H,k}$, given upper bound for value approximation error $\|\hat{V}_k - V^{\pi_k}\|_\infty \le \epsilon_k$. Also denote approximation error for dynamics model $\hat{\rho}$ as $\epsilon_m = \max_{s,a} D_{TV}(\rho(\cdot|s_t, a_t)\|\hat{\rho}(\cdot|s_t, a_t))$, planner suboptimality as ϵ_p. Also let the reward function r is bounded by $[0, R_{max}]$ and $\hat{V}$ be upper bounded by $V_{max} \le \frac{1}{1-\gamma} R_{max}$, then the following uniform bound of performance suboptimality holds:*

$$\begin{aligned} &\limsup_{k\to\infty} |V^* - V^{\pi_{H,k}}| \\ &\le \limsup_{k\to\infty} \frac{2}{1-\gamma^H} \Big[C(\epsilon_{m,k}, H, \gamma) + \frac{\epsilon_{p,k}}{2} \\ &\quad + \frac{\gamma^H(1+\gamma^2)}{(1-\gamma)^2} \epsilon_k \Big] \end{aligned} \tag{4}$$

for any policy μ, while C is defined as:

$$C(\epsilon_m, H, \gamma) = R_{max} \sum_{t=0}^{H-1} \gamma^t t \epsilon_m + \gamma^H H \epsilon_m V_{max} \tag{5}$$

This theorem demonstrates that errors in the model and value functions have a decoupled influence on planning performance. If assuming model error and planner suboptimality are insignificant, then converging value approximation error guarantees converging planning performance. Notably, the theorem also indicates that, under identical conditions, planning procedure allows π_H to mitigate its reliance on value accuracy by at least a factor of γ^{H-1} compared to a greedy policy[1]. This explains why TD-MPC2 achieves strong performance in certain tasks despite exhibiting significant overestimation bias. However, the policy learning framework does not guarantee reduced approximation error in practice. As task complexity increases—particularly in environments with high-dimensional action spaces such as `h1hand-run-v0`—value overestimation worsens, leading to inefficient learning and suboptimal performance. In the following chapter, we delve deeper into the root cause of this phenomenon and uncover a fundamental structural limitation in TD-MPC2.

4 Policy Mismatch in Planning-Based MBRL

In many cases, leveraging online planning for data collection is favorable, resulting in high-quality trajectories. However, Due to the planner used to interact with the environment, training data distribution corresponds with the planner policy π_H instead of the nominal policy π. Such distributional mismatch between π_H and π may incur severe generalization errors from the value estimation, undermining training stability. In this chapter, we provide intuitions and theoretical analysis of the value learning issue while distinguishing it from similar problems encountered in model-free RL or offline RL.

4.1 Policy Mismatch and Extrapolation Error

Extrapolation error that has been well articulated in offline RL studies [21, 32, 23]. Such errors appear during policy evaluation, where the value function is queried with *out-of-distribution* (OOD) state-action pair. Then, temporal difference methods propagate generalization errors iteratively, causing the value estimation to deviate further. Sequentially, value approximation error directly undermines performance by corrupting the return estimation of sampled trajectories as discussed in 3.3.

While the training dataset is fixed in offline RL, in online policy learning, we expect to fix such errors by exploring the over-estimated regions in state-action space. Thus, it is important to understand how the policy mismatch brought by the planner affects value learning and what distinguishes it from model-free off-policy learning. We demonstrate such influence by the following toy example.

[1]See proof in Appendix A.6

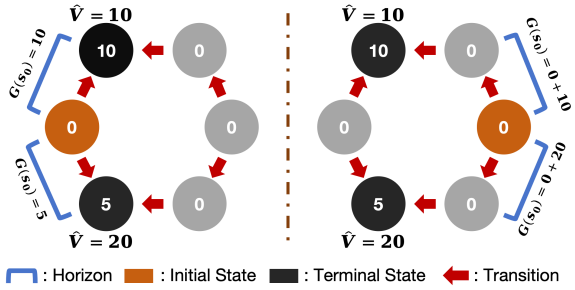


Figure 2. Toy Example

Figure 2 illustrates a simple oriented graph world with two terminal states among six states. Assume all rewards (labeled in each state) are known. Consider a 1-step lookahead policy π_1 acquired through planning process (2) and inaccurate value estimation $\hat{V}$ at terminal states. If initialized on the left, π_1 will always choose the optimal action that ends up getting reward 10. However, the value error remains since the agent does not explore the other terminal state. Thus, when initialized on the right, the agent will be misled to the suboptimal terminal state. This describes a latency on value estimation calibration. In contrast, a standard greedy policy will immediately fix the value error by directly visiting the overestimated state. In more general cases, the planner policy refined through the H-step planning process may never cover relatively low reward regions where under-trained nominal policy tends to go.

4.2 Error Accumulation in TD-MPC

While the policy mismatch in TD-MPC delays the correcting of value overestimation bias, one might expect that, given sufficient perturbations, the agent would eventually visit overestimated regions and rectify the errors. However, we argue that this self-correction is extremely difficult because value approximation errors not only propagate across states [7] but, more importantly, accumulate through policy iteration. As a result, the H-step lookahead policy π_H drifts further from the nominal policy π, exacerbating the mismatch rather than resolving it. This compounding effect makes policy mismatch in TD-MPC a far more critical issue than it might initially appear. We first quantify approximation error accumulation in the following theorem. We defer the complete proof to Appendix A.7.

Theorem 4.1 (TD-MPC Error Accumulation)**.** *Consider π_k is the nominal policy acquired through approximation policy iteration (API), and the resulting H-step lookahead policy is $\pi_{H,k}$. Assume $\pi_{H,k}$ outperforms π_k with performance gap $\delta_k = \|V^{\pi_{H,k}} - V^{\pi_k}\|_\infty$. Denote value approximation error $\epsilon_k = \|\hat{V}_k - V^{\pi_k}\|_\infty$, approximated dynamics holds*

model error $\epsilon_{m,k}$, planner sub-optimality is ϵ_p. Also let the reward function r is bounded by $[0, R_{max}]$, then the following uniform bound of performance gap holds:

$$\delta_k \leq \frac{1}{1-\gamma^H}\Big[2C(\epsilon_{m,k-1}, H, \gamma) + \epsilon_{p,k-1} + (1+\gamma^H)\delta_{k-1} + \frac{2\gamma(1+\gamma^{H-1})}{1-\gamma}\epsilon_{k-1}\Big] \tag{6}$$

where C is defined in equation (5)*.*

Note that the upper bound is quite loose due to the usage of infinite norm. Nonetheless, the direct takeaway of Theorem 4.1 is that we can always expect a relatively large performance gap between H-step lookahead policy and nominal policy due to the accumulating approximation error. We further discuss this theorem by comparing the value overestimation trend of TD-MPC2 with horizon H=1 in Appendix C. The following theorem further bridges the performance gap and policy mismatch between $\pi_{H,k}$ and π_k:

Theorem 4.2 (Policy divergence)**.** *Given policies $\pi, \pi' \in \Pi : S \to A$, suppose reward is upper bounded by R_{max}, then we have policy divergence lower bounded by performance gap as:*

$$\max_s D_{TV}\left(\pi'(a|s) \| \pi(a|s)\right) \geq \frac{(1-\gamma)^2}{2R_{max}}|J^\pi - J^{\pi'}| \tag{7}$$

Proof can be found in Appendix A.4.

Here, we connect error accumulation in the performance gap to policy mismatch. As approximation errors amplify the value gap shown in Theorem 4.1, they also induce large policy divergence, exacerbating distributional shifts. Consequently, the region corresponding to π is underrepresented in the buffer, preventing the correction of overestimation and perpetuating generalization errors during policy evaluation.

In conclusion, although in 3.3 the H-step lookahead policy is theoretically less sensitive to value approximation errors, a substantial of them are introduced and accumulate over training time due to policy mismatch. As a result, naively applying policy iteration appears to be flawed, failing to fully exploit the potential of combining model-based optimization and temporal-difference learning. The next chapter will discuss potential approaches to improve the current temporal difference MPC algorithms.

5 Improving Value Learning By A Minimalist Approach

In order to mitigate policy mismatch, constraints or regularization can be either addressed on the planner or policy iteration to align both policies. Unlike prior work, [35] that constrains the planner with the policy prior, we address policy mismatch during policy iteration. This is based on the intuition that constrained online searching the former applied inevitably harms exploration and data efficiency, as its performance does not match the original TD-MPC [15], which learns without additional regularization.

5.1 TD-MPC with Policy Constraint

While in principle, many training approaches in offline RL can be applied, some of them are overly convoluted and hard to tune due to the use of complex sampling-based estimation or the massive amounts of hyperparameters [6]. We favor a simple implementation framework that can be seamlessly integrated into the current planning-based MBRL pipeline.

To avoid *out-of-distribution* query, a constrained policy improvement can be described as solving a constrained optimization problem:

$$\begin{aligned} \pi_{k+1} \coloneqq \operatorname*{argmax}_{\pi} \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[Q_k(a,s)] \\ \text{s.t.} \mathbf{D_{KL}}(\pi_{k+1} \| \mu_k) \leq \epsilon \end{aligned} \tag{8}$$

$$\mathcal{T}^\pi Q \coloneqq r(s,a) + \gamma \mathbb{E}_{s'\sim\rho(\cdot|s,a)} \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q^\pi(s',a')] \tag{9}$$

Where μ_k denotes the behavior policy from the buffer at the k-th iteration. It can be represented as a weighted sum of H-step lookahead policies of past iteration $\mu_K(\cdot|s) = \Sigma_{k=0}^K \omega_k \pi_{H,k}(\cdot|s)$, where ω_k is the weight for the multivariant Gaussian policy $\pi_{H,k}$.

5.2 Implementation of A Minimalist Modification

Instead of directly tackling this constrained policy improvement, many practices approximately solve it by dealing with its Lagrangian version [32, 29]. We express the resulting implementation of policy training loss as follows:

$$\mathcal{L}_\pi = -\mathop{\mathbb{E}}_{a\sim\pi}\left[Q(s,a) - \alpha \log \pi(a|s) + \beta \log \mu(a|s)\right] \tag{10}$$

By decoupling the parameters α and β, we interpret this training objective as the lagrangian version of (8) with entropy regularization. It can also be seen as a combination of two key components: A maximum entropy objective [10], which promotes exploration and facilitates policy improvement, and a distribution regularization term that encourages the policy to select in-domain actions. To simplify the calculation, we can maximize $\mathbb{E}_{\mu'\sim\{\mu\}}[\log \mu']$ as the lower bound of $\log(\mu)$.

The proposed approach is a general modification compatible with most value-guided and planning-based MBRL algorithms, requiring only a simple adjustment to the policy improvement step. In the experiment section, we build our algorithm on **TD-MPC2**, leveraging **MPPI** to solve (2) in

Algorithm 1 TD-M(PC)^2

Require: π_θ, d_ψ, enc, $Q_\phi(s,a)$, $Q_{\phi'}(s,a)$, P, α, β, ρ
Initialize policy network π_θ, latent world model d_ψ, encoder enc, and value functions $Q_\phi, Q_{\phi'}(s,a)$ by pertaining on uniformly sampled data.
for each training step **do**
  if collect data **then**
    Planning: $a \sim \mu = P(\pi_\theta, Q_\phi, d, enc(s))$.
    Environment step: $r, s', done = env.step(a)$
    Add (s, a, μ, r, s') to buffer $\mathcal{B}$.
  end if
  Sample trajectories $\{(s_t, a_t, \mu_t, r_t, s_{t+1})_{0:H}\} \sim \mathcal{B}$.
  # Model Update
  Calculate TD target by bootstrapping π_θ
  Update d_ψ, enc, Q_ϕ by (3)
  # Constrained Policy Update
  Calculate policy loss:
  $\mathcal{L}_\pi = -\mathop{\mathbb{E}}_{a\sim\pi}[Q(s,a) - \alpha \log \pi(a|s) + \beta \log \mu(a|s)]$
  Polyak update $\phi'_i = \rho\phi'_i + (1-\rho)\phi_i$, $i = 1, 2$
end for

latent space. The resulting algorithm is demonstrated in Algorithm 1.

Balance Exploration and Exploitation. In addition, we notice that addressing policy constraints during the initial stage sometimes results in failure to reach out to the local minima. Thus, we maintain moving percentiles of the Q function as in [15, 14] to scale the training loss that does not enforce the regularization term until the percentile is greater than a small threshold.

6 Experiments

Our experiments aim to assess whether the proposed method enhances the performance of Temporal Difference Model Predictive Control (TD-MPC) and aligns with our theoretical analysis. Specifically, we seek to answer the following key questions:

- Does our method reduce value approximation error, leading to more accurate policy evaluation?
- Does our approach improve upon the *state-of-the-art* TD-MPC2 algorithm in high-dimensional continuous control tasks?
- Which design component is most critical to the observed performance improvements in complex, high-dimensional tasks?

Benchmarks. To ensure a rigorous evaluation, we benchmark our method across 21 high-dimensional continuous control tasks drawn from HumanoidBench [34] and the DeepMind Control Suite (DMControl) [39]. *Humanoid-Bench* is a standardized suite for humanoid control. We test all 14 locomotion tasks in `h1hand-v0`, requiring whole-body control of a humanoid robot with dexterous hands and 61-dimensional action space, making these tasks particularly challenging. Additionally, we benchmark our method on seven high-dimensional **DMControl** tasks, including `dog` (36-DoF) and `humanoid` (21-DoF) environments. While these tasks are less complex, they provide a broader evaluation of our method's generalization across diverse settings. By conducting experiments on these benchmark suites, we ensure a comprehensive and challenging evaluation of our method in both extreme high-dimensional and more regular continuous control settings.

Baselines. To evaluate the effectiveness of our method, we primarily compare it against TD-MPC2 [14]. Moreover, we benchmark against leading model-based and model-free RL methods, including DreamerV3 [13] and Soft Actor-Critic (SAC) [10], both of which have demonstrated strong performance across various high-dimensional control problems. For a fair comparison, we use the official TD-MPC2 implementation and DMControl scores[2], while reporting HumanoidBench results from author-provided scores[3].

6.1 Improved Value Learning

As illustrated in Figures 5 and 1, empirically, our method enables the learned value function to more closely align with actual returns, reducing structural bias in value estimation, which justifies our theoretical analysis. By mitigating policy mismatch through conservative policy updates, our approach effectively reduces extrapolation error, leading to more reliable value learning. This improvement is particularly crucial in high-dimensional control tasks, where inaccuracies in value estimation can compound over time. We will show that this improvement in value learning ultimately results in more effective decision-making and greater confidence in the resulting H-step lookahead policy.

6.2 Benchmark Performance

We adopt the same settings for shared hyperparameters as those reported in the TD-MPC2 paper to our algorithm without any task-specific tuning. This consistency allows us to directly assess the adaptability and robustness of our approach across different tasks and environments. We provide a comprehensive list of hyperparameter settings for reproducibility and transparency in B. Our experiments are conducted across three random seeds for DMControl and

[2] `https://github.com/nicklashansen/tdmpc2`
[3] `https://github.com/carlosferrazza/humanoid-bench`

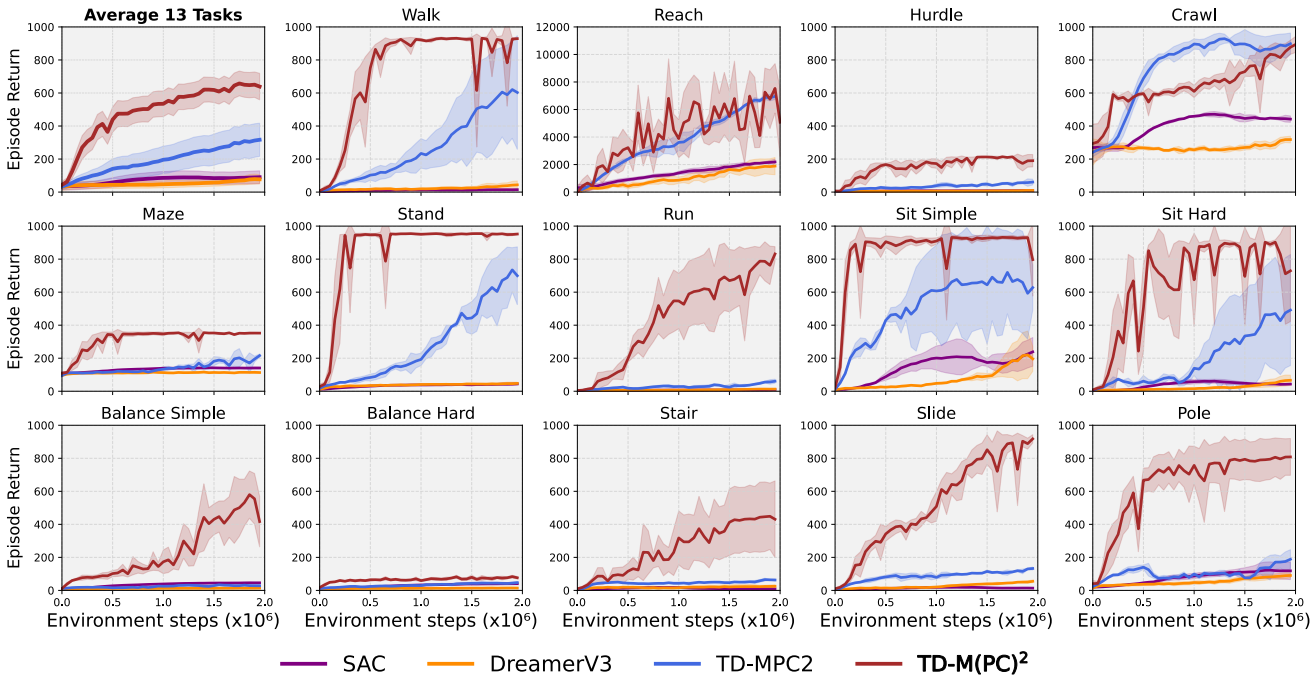


Figure 3. **Humanoid-Bench Locomotion Suite.** Average episode return of our method (TD-M(PC)^2) and baselines. We report mean performance and 95% CIs across 14 humanoid locomotion tasks. We do not include `Reach-v0` in the average result due to its distinct reward scale.

five random seeds for HumanoidBench, with a total of two million environment steps of online data collection.

For HumanoidBench, as demonstrated in Figure 3, our method consistently outperforms the baseline by a large margin for most tasks. In specific, we observe significant improvements in locomotion tasks, including `Run`, `Slide`, and `Pole`. In these tasks, the humanoid robot only needs to perform regular and consistent motion, such as walking and running, but under diverse scenarios and terrain. Intuitively, we expect these tasks to be easier to solve, but TD-MPC2 suffers from low data efficiency even compared to more complex goal-oriented locomotion tasks like `Reach`. We found TD-M(PC)^2 tends to perform cautiously, while the baseline demonstrates exaggerated motion corresponding to its overestimated value. In addition, we provide the average training curve for episode return. In terms of average performance by the end of the training, TD-M(PC)^2 improves TD-MPC2 by over 100%.

We also evaluate our approach on some of the most challenging tasks in the DMControl suite, with the training curves presented in Figure 4. On average, TD-M(PC)^2 slightly outperforms the baseline. Notably, significant improvements are observed in three `dog` tasks, whereas the performance on `Humanoid` tasks, which feature a slightly smaller action space, remains comparable to the baseline. We visualize trajectories generated by our method for some tasks in 9.

6.3 Ablation Study

To better understand the impact of policy regularization, we evaluate two variants of TD-M(PC)^2: a mildly regularized version with a regularization coefficient of $\beta = 0.05$ and an overly conservative variant that directly applies behavior cloning (BC) for policy updates. These variants are tested on two high-dimensional continuous control tasks; implementation details on the BC variant are provided in Appendix B.

As shown in Figure 6, all three variants achieve similar performance, suggesting that the choice of regularization strength has a limited effect. Notably, despite the lack of value-based policy learning, the behavior cloning variant performs nearly on par with the others. This highlights the dominant role of conservatism in high-dimensional tasks, suggesting that reducing out-of-distribution actions is a key factor in improving stability and performance.

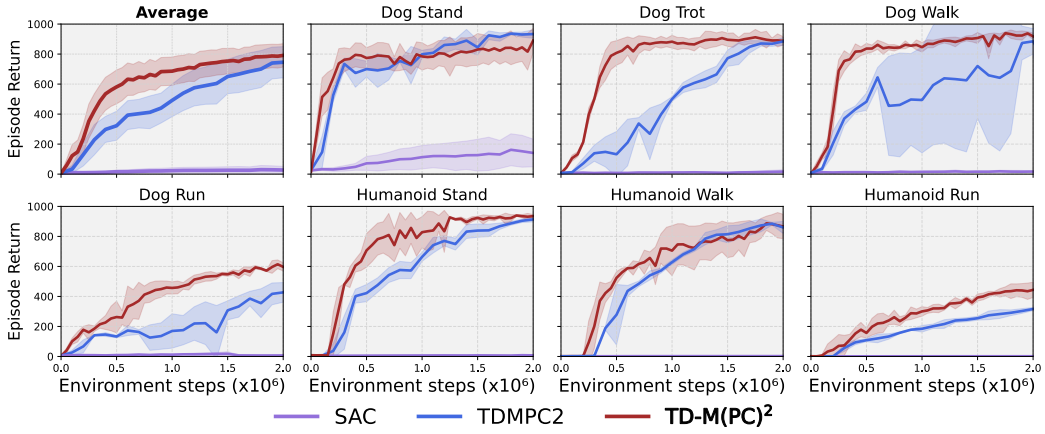


Figure 4. **DM Control Suite.** Average episode return of our method (TD-M(PC)^2) and baselines. We report mean performance and 95% CIs across 7 high-dimensional continuous control tasks. We also present the average performance on all algorithms.

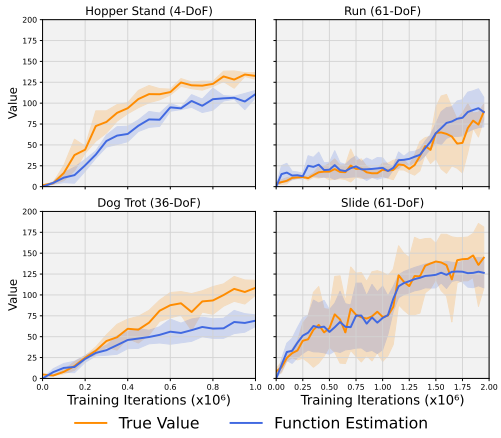


Figure 5. Value estimation of TD-M(PC)^2. The true value and function estimation are obtained with the same approach in Figure 1. The proposed significantly mitigates value overestimation for all four tasks.

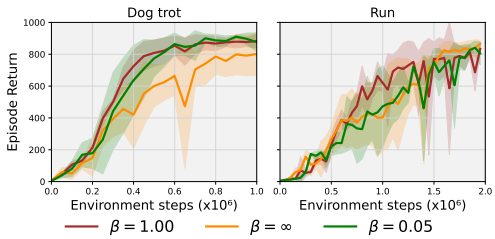


Figure 6. Ablation study on β. We evaluate all variants on two high-dimensional tasks from different domains: `dog-trot` and `h1hand-run-v0`. The results indicate that our method is not sensitive to β.

7 Conclusions

In this paper, we identify and address a fundamental limitation in existing model-based reinforcement learning algorithms—persistent value overestimation caused by structural policy mismatch. Through both theoretical and empirical analysis, we demonstrate that standard policy iteration leads to compounding errors in value estimation. To mitigate this issue, we introduce a simple yet effective policy regularization term that reduces out-of-distribution queries. Our approach seamlessly integrates into existing frameworks and significantly enhances performance, particularly in challenging high-dimensional tasks such as 61-DoF humanoid control. These findings underscore the importance of conservative exploitation of planner-generated data.

References

[1] A. Argenson and G. Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

[2] D. Bertsekas. Neuro-dynamic programming. *Athena Scientific*, 1996.

[3] M. Bhardwaj, A. Handa, D. Fox, and B. Boots. Information theoretic model predictive q-learning. In *Learning for Dynamics and Control*, pages 840–850. PMLR, 2020.

[4] A. Chan, H. Silva, S. Lim, T. Kozuno, A. R. Mahmood, and M. White. Greedification operators for policy optimization: Investigating forward and reverse kl divergences. *Journal of Machine Learning Research*, 23(253):1–79, 2022.

[5] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[6] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[7] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[8] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[9] D. Garg, J. Hejna, M. Geist, and S. Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[11] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[12] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[13] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv: 2301.04104*, 2023.

[14] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[15] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

[16] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[17] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[18] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[19] S. Katsigiannis and N. Ramzan. Dreamer: A database for emotion recognition through eeg and ecg signals from wireless low-cost off-the-shelf devices. *IEEE journal of biomedical and health informatics*, 22(1):98–107, 2017.

[20] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[21] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

[22] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[23] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[24] C. Li, A. Krause, and M. Hutter. Robotic world model: A neural network simulator for robust policy optimization in robotics. *arXiv preprint arXiv:2501.10100*, 2025.

[25] M. Littman and A. Moore. Reinforcement learning: A survey, journal of artificial intelligence research 4, 1996.

[26] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

[27] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, et al. Imitation is not enough: Robustifying imitation

with reinforcement learning for challenging driving scenarios. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7553–7560. IEEE, 2023.

[28] R. Munos. Performance bounds in l_p-norm for approximate value iteration. *SIAM journal on control and optimization*, 46(2):541–561, 2007.

[29] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[30] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36, 2024.

[31] S. Park, K. Frans, S. Levine, and A. Kumar. Is value learning really the main bottleneck in offline rl? *arXiv preprint arXiv:2406.09329*, 2024.

[32] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[33] J. Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[34] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

[35] H. Sikchi, W. Zhou, and D. Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pages 1622–1633. PMLR, 2022.

[36] S. P. Singh and R. C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16:227–233, 1994.

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

[38] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[39] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[40] S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pages 255–263. Psychology Press, 2014.

[41] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[42] G. Zhou, S. Swaminathan, R. V. Raju, J. S. Guntupalli, W. Lehrach, J. Ortiz, A. Dedieu, M. Lázaro-Gredilla, and K. Murphy. Diffusion model predictive control. *arXiv preprint arXiv:2410.05364*, 2024.

A Theory and Discussion

A.1 Useful Lemma

Lemma A.1. *[36] Suppose π_{k+1} is 1-step greedy policy of value function $\hat{V}_k$. Denote V^* as the optimal value function, if there exists ϵ such that $\|V^* - \hat{V}_k\|_\infty \leq \xi_k$, we can bound the value loss of π by:*

$$V^* - V^{\pi_{k+1}} \leq \frac{2\gamma\xi_k}{1-\gamma} \tag{11}$$

Lemma A.2. *[2] Suppose $\{\pi_k\}$ is policy sequence generated by approximate policy iteration (API), then the maximum norm of value loss can be bounded as:*

$$\limsup_{k\to\infty} \|V^* - V^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \|\hat{V}_k - V^{\pi_k}\|_\infty \tag{12}$$

Lemma A.3. *[35] Denote approximation error for dynamics model $\hat{\rho}$ as $\epsilon_m = \max_{s,a} D_{TV}(\rho(\cdot|s_t,a_t)\|\hat{\rho}(\cdot|s_t,a_t))$. Denote ϵ_p as suboptimality incurred in H-step lookahead optimization such that $J^* - \hat{J} \leq \epsilon_p$. Let $\hat{V}$ be an approximate value function such that $\|V^* - \hat{V}\|_\infty \leq \xi$. Also let the reward function r is bounded by $[0, R_{max}]$ and $\hat{V}$ be bounded by $[0, V_{max}]$. Then, the performance of the H-step lookahead policy can be bounded as:*

$$J^{\pi^*} - J^{\pi_H} \leq \frac{2}{1-\gamma^H}\Big[C(\epsilon_m, H, \gamma) + \frac{\epsilon_p}{2} + \gamma^H \xi\Big] \tag{13}$$

while C is defined as:

$$C(\epsilon_m, H, \gamma) = R_{max} \sum_{t=0}^{H-1} \gamma^t t \epsilon_m + \gamma^H H \epsilon_m V_{max} \tag{14}$$

A.2 Proof of Theorem

Theorem A.4 (Policy divergence)**.** *Given policies $\pi, \pi' \in \Pi : S \to A$, suppose reward is upper bounded by R_{max}, then we have policy divergence lower bounded by performance gap as:*

$$\max_s D_{TV}\left(\pi'(a|s)\|\pi(a|s)\right) \geq \frac{(1-\gamma)^2}{2R_{max}}|J^\pi - J^{\pi'}| \tag{15}$$

Proof. From the definition of expected return, we have the following inequality,

$$\begin{aligned}
|J^\pi - J^{\pi'}| &= |\Sigma_s(p^\pi(s,a) - p^{\pi'}(s,a))r(s,a)| \\
&= |\Sigma_t \Sigma_{s,a} \gamma^t (p^\pi(s,a) - p^{\pi'}(s,a))r(s,a)| \\
&\leq R_{\max} \Sigma_t \Sigma_{s,a} \gamma^t |p_t^\pi(s,a) - p_t^{\pi'}(s,a)| \\
&= 2R_{\max} \Sigma_t \gamma^t D_{TV}(p_t^\pi(s,a)\|p_t^{\pi'}(s,a))
\end{aligned}$$

Using Lemma B.1 and Lemma B.2 from [17] we can relate joint distribution TVD to policy TVD:

$$\begin{aligned}
D_{TV}(p_t^\pi(s,a)\|p_t^{\pi'}(s,a)) &\leq D_{TV}(p_t^\pi(s)\|p_t^{\pi'}(s)) + \max_s D_{TV}(p_t^\pi(a|s)\|p_t^{\pi'}(a|s)) \\
&\leq (t+1)\max_s D_{TV}\left(\pi(a|s)\|\pi'(a|s)\right)
\end{aligned}$$

Plug this inequality back we thus the main conclusion:

$$\begin{aligned}
|J^\pi - J^{\pi'}| &\leq 2R_{\max}\Sigma_t \gamma^t (1+t) \max_s D_{TV}\left(\pi(a|s)\|\pi'(a|s)\right) \\
&\leq \frac{2R_{\max}}{(1-\gamma)^2} \max_s D_{TV}\left(\pi(a|s)\|\pi'(a|s)\right)
\end{aligned}$$

□

Theorem A.5. *Assume the nominal policy π_k is acquired through approximation policy iteration (API) at k-th iteration and the resulting planner policy at k-th iteration is $\pi_{H,k}$, given upper bound for value approximation error $\|\hat{V}_k - V^{\pi_k}\|_\infty \leq \epsilon_k$. Also denote approximation error for dynamics model $\hat{\rho}$ as $\epsilon_m = \max_{s,a} D_{TV}(\rho(\cdot|s_t, a_t)\|\hat{\rho}(\cdot|s_t, a_t))$, planner sub-optimality as ϵ_p. Also let the reward function r is bounded by $[0, R_{max}]$ and and $\hat{V}$ be bounded by $[0, V_{max}]$, then the following uniform bound of performance suboptimality holds:*

$$\limsup_{k\to\infty} |V^* - V^{\pi_{H,k}}| \leq \limsup_{k\to\infty} \frac{2}{1-\gamma^H}\Big[C(\epsilon_{m,k}, H, \gamma) + \frac{\epsilon_{p,k}}{2} + \frac{\gamma^H(1+\gamma^2)}{(1-\gamma)^2}\epsilon_k\Big] \tag{16}$$

while C is defined as:

$$C(\epsilon_m, H, \gamma) = R_{max}\sum_{t=0}^{H-1}\gamma^t t\epsilon_m + \gamma^H H\epsilon_m V_{max} \tag{17}$$

Proof. Denote the planner policy (H-step look-ahead policy) as π_H, which is acquired through planning with terminal value $\hat{V}$. We define τ^* as a trajectory sampled by optimal policy π^*, and $\hat{\tau}$ as a trajectory sampled by π_H under the real dynamics. We do not consider approximation error for the reward function since knowledge of the reward function can be guaranteed in most training cases. Following deduction in Theorem 1 of LOOP[35], we can have the H-step policy suboptimality bound through the following key steps (We ignore k here for simplicity since the following deduction holds true for any $\hat{V}$). The complete proof can be found in Theorem 1 of the referenced paper.

$$\begin{aligned}
V^*(s_0) - V^{\pi_{H,k}}(s_0) &= \mathbb{E}_{\tau*}\left[\Sigma\gamma^t r(s_t, a_t) + \gamma^H V^*(s_H)\right] - \mathbb{E}_{\hat{\tau}}\left[\Sigma\gamma^t r(s_t, a_t) + \gamma^H \hat{V}_k(s_H)\right] \\
&\leq \mathbb{E}_{\tau^*}\left[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}_k(s_H)\right] - \mathbb{E}_{\hat{\tau}}\left[\sum \gamma^t r(s_t, a_t) + \gamma^H \hat{V}_k(s_H)\right] \\
&\quad + \gamma^H \mathbb{E}_{\tau^*}\left[V^*(s_H) - \hat{V}_k(s_H)\right] - \gamma^H \mathbb{E}_{\hat{\tau}}\left[V^*(s_H) - \hat{V}_k(s_H)\right] \\
&\quad + \gamma^H \mathbb{E}_{\hat{\tau}}\left[V^*(s_H) - V^{\pi_{H,k}}(s_H)\right] \\
&\leq \frac{2}{1-\gamma^H}\Big[C(\epsilon_{m,k}, H, \gamma) + \frac{\epsilon_{p,k}}{2} + \gamma^H\|V^* - \hat{V}_k\|_\infty\Big]
\end{aligned}$$

Due to the optimality, we always have the fact that $V^{\pi_H} \leq V^*$. Leveraging the value error bound for API given by [28], we further bound ϵ_v with approximation error us:

$$\begin{aligned}
\limsup_{k\to\infty} |V^* - V^{\pi_{H,k}}| &\leq \limsup_{k\to\infty} \frac{2}{1-\gamma^H}\Big[C(\epsilon_{m,k}, H, \gamma) + \frac{\epsilon_{p,}}{2} + \gamma^H\|V^* - V^{\pi_k}\|_\infty + \gamma^H\epsilon_k\Big] \\
&\leq \limsup_{k\to\infty} \frac{2}{1-\gamma^H}\Big[C(\epsilon_{m,k}, H, \gamma) + \frac{\epsilon_{p,k}}{2} + \frac{\gamma^H(1+\gamma^2)}{(1-\gamma)^2}\epsilon_k\Big]
\end{aligned}$$

This indicates that given identical conditions, knowing that the approximation error is small implies that V^{π_H} will be close to the optimal value V^* eventually. □

Similar to LOOP, we can compare performance bounds between H-step lookahead policy and 1-step greedy policy (Notice that in API, π_{k+1} iteration can be seen as 1-step greedy policy of $\hat{V}_k$ if we assume policy improvement step is optimal). With Lemma A.1, we show how do value approximation error influences greedy policy performance.

Theorem A.6. *Suppose π_{k+1} is 1-step greedy policy of value function $\hat{V}_k$. Denote V^* as the optimal value function, if there exists ϵ such that for any k, $\|\hat{V}_k - V^{\pi_k}\|_\infty \leq \epsilon$, we can bound the performance of $\pi_{\hat{V}}$ by:*

$$\limsup_{k\to\infty} \|V^* - V^{\pi_{k+1}}\|_\infty \leq \limsup_{k\to\infty} \frac{2\gamma(1+\gamma^2)\epsilon}{(1-\gamma)^3} \tag{18}$$

Proof. Directly combining A.1 and A.2.

$$\begin{aligned}
\|V^* - V^{\pi_{k+1}}\|_\infty &\leq \frac{2\gamma}{1-\gamma}\|V^* - \hat{V}_k\|_\infty \\
&\leq \frac{2\gamma}{1-\gamma}\|V^* - \hat{V}_k\|_\infty
\end{aligned}$$

$$\limsup_{k\to\infty}\|V^*-V^{\pi_{k+1}}\|_\infty \le \frac{2\gamma}{1-\gamma}(\frac{2\gamma}{(1-\gamma)^2}+1)\limsup_{k\to\infty}\|\hat{V}_k-V^{\pi_k}\|_\infty$$
$$\le \frac{2\gamma(1+\gamma^2)}{(1-\gamma)^3}\epsilon_k$$

□

Based on Theorem A.6 and Theorem A.5, we conclude that the test-time planning enables $\pi_{H,k}$ to reduce its performance dependency on value accuracy by at least a factor of γ^{H-1} compared to the greedy policy π_{k+1}, providing a general sense of superiority introduce by the planner. However, we point out this does not guarantee nominal policy π performs well under TD-MPC pipeline due to the possible persistent value over-estimation.

Theorem A.7 (TD-MPC Error Propagation)**.** *Consider π_k is the nominal policy acquired through approximation policy iteration (API), and the resulting H-step lookahead policy is $\pi_{H,k}$. Assume $\pi_{H,k}$ outperforms π_k with performance gap $\delta_k = \|V^{\pi_{H,k}} - V^{\pi_k}\|_\infty$. Denote value approximation error $\epsilon_k = \|\hat{V}_k - V^{\pi_k}\|_\infty$, approximated dynamics holds model error $\epsilon_{m,k}$, planner sub-optimality is ϵ_p. Also let the reward function r is bounded by $[0, R_{max}]$ and $\hat{V}_{k-1}$ be bounded by $[0, V_{max}]$, then the following uniform bound of performance gap holds:*

$$\delta_k \le \frac{1}{1-\gamma^H}\left[2C(\epsilon_{m,k-1}, H, \gamma) + \epsilon_{p,k-1} + (1+\gamma^H)\delta_{k-1} + \frac{2\gamma(1+\gamma^{H-1})}{1-\gamma}\epsilon_{k-1}\right] \tag{19}$$

where C is defined as (14)*.*

Proof. Denote the planner policy (H-step look-ahead policy) as π_H, which is acquired through planning with terminal value $\hat{V}$. We do not consider approximation error for the reward function since knowledge of the reward function can be guaranteed in most training cases. At k-th iter, denote $\hat{\tau}^k$ as trajectory sampled by planner policy; τ^k as trajectory sampled by optimal planner leveraging real dynamics model; τ^{π_k} as trajectory sampled by nominal policy π_k (all trajectories are sampled in the environment rather than under approximate dynamics model). For simplicity, Σ in this proof stands for $\Sigma_{t=0}^{H-1}$ if not specified.

$$\begin{aligned}
V^{\pi_{H,k}}(s_0) - V^{\pi_k}(s_0) &= \mathbb{E}_{\hat{\tau}^k}\left[\Sigma\gamma^t r(s_t,a_t) + \gamma^H V^{\pi_{H,k}}(s_H)\right] - V^{\pi_k}(s_0)\\
&= \mathbb{E}_{\hat{\tau}^k}\left[\Sigma\gamma^t r(s_t,a_t) + \gamma^H V^{\pi_k}(s_H)\right] - V^{\pi_k}(s_0)\\
&\quad + \gamma^H \mathbb{E}_{\hat{\tau}^k}\left[V^{\pi_{H,k}}(s_H) - V^{\pi_k}(s_H)\right]
\end{aligned}$$

Particularly, we have:

$$\begin{aligned}
&\mathbb{E}_{\hat{\tau}^k}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H V^{\pi_k}(s_H)] - V^{\pi_k}(s_0)\\
&= \mathbb{E}_{\hat{\tau}^k}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - \hat{V}_{k-1}(s_H)] - V^{\pi_k}(s_0)\\
&\le \mathbb{E}_{\tau^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] - \mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)]\\
&\quad + \mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - \hat{V}_{k-1}(s_H)] - V^{\pi_k}(s_0)\\
&\le \mathbb{E}_{\tau^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] - \mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)]\\
&\quad + \mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - \hat{V}_{k-1}(s_H)]\\
&\quad - V^{\pi_{k-1}}(s_0) + [V^{\pi_{k-1}}(s_0) - V^{\pi_k}(s_0)]
\end{aligned}$$

The second step is due to the definition of τ^{k-1}:

$$\tau^{k-1} = \arg\max_\tau \mathbb{E}_\tau[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)]$$

We use Theorem 1 in [35] for the first row to bind them by model error and planner sub-optimality.

$$\mathbb{E}_{\tau^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] - \mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] \le 2C(\epsilon_{m,k-1}, H, \gamma) + \epsilon_{p,k-1}$$

where $C(\epsilon_{m,k-1}, H, \gamma)$ is the same as defined in Theorem A.5.

For the second row, since we want to construct a bound related to $\|V^{\pi_{H,k-1}} - V^{\pi_{k-1}}\|$. Under this orientation, we first show that:

$$\begin{aligned}
&\mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H \hat{V}_{k-1}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - \hat{V}_{k-1}(s_H)] \\
=& \mathbb{E}_{\hat{\tau}^{k-1}}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H V^{\pi_{H,k-1}}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^{k-1}}[\hat{V}_{k-1}(s_H) - V^{\pi_{H,k-1}}(s_H)] \\
&+ \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - V^{\pi_{k-1}}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_{k-1}}(s_H) + \hat{V}_{k-1}(s_H)] \\
=& V^{\pi_{H,k-1}}(s_0) + \gamma^H \mathbb{E}_{\hat{\tau}^{k-1}}[\hat{V}_{k-1}(s_H) - V^{\pi_{k-1}}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^{k-1}}[V^{\pi_{k-1}}(s_H) - V^{\pi_{H,k-1}}(s_H)] \\
&+ \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - V^{\pi_{k-1}}(s_H)] + \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_{k-1}}(s_H) - \hat{V}_{k-1}(s_H)]
\end{aligned}$$

Sequentially, we combine the inequalities above:

$$\begin{aligned}
&\mathbb{E}_{\hat{\tau}^k}[\Sigma\gamma^t r(s_t,a_t) + \gamma^H V^{\pi_k}(s_H)] - V^{\pi_k}(s_0) \\
\leq& 2C(\epsilon_{m,k-1}, H, \gamma) + \epsilon_{p,k-1} + 2\gamma^H \epsilon_{k-1} \\
&+ V^{\pi_{H,k-1}}(s_0) - V^{\pi_{k-1}}(s_0) + \gamma^H \mathbb{E}_{\hat{\tau}^{k-1}}[V^{\pi_{k-1}}(s_H) - V^{\pi_{H,k-1}}(s_H)] \\
&+ \gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - V^{\pi_{k-1}}(s_H)] + [V^{\pi_{k-1}}(s_0) - V^{\pi_k}(s_0)]
\end{aligned}$$

We can have a rough bound on the second line by $(1+\gamma^H)\delta_{k-1}$. With Lemma 6.1 in [2], we can bound the final line as:

$$\gamma^H \mathbb{E}_{\hat{\tau}^k}[V^{\pi_k}(s_H) - V^{\pi_{k-1}}(s_H)] + [V^{\pi_{k-1}}(s_0) - V^{\pi_k}(s_0)] \leq (1+\gamma^H)\frac{2\gamma}{1-\gamma}\epsilon_{k-1}$$

Combining all the inequalities and leveraging the contraction property, we get the final bound for the performance gap:

$$V^{\pi_{H,k}}(s_0) - V^{\pi_k}(s_0) \leq \frac{1}{1-\gamma^H}\left[2C(\epsilon_{m,k-1}, H, \gamma) + \epsilon_{p,k-1} + (1+\gamma^H)\delta_{k-1} + \frac{2\gamma(1+\gamma^{H-1})}{1-\gamma}\epsilon_{k-1}\right]$$

Thus, we can easily get the final result. □

In addition, according to Lemma 6.1 in [2], which describes the policy improvement bound for a greedy policy, the dependence of policy improvement on value accuracy is increased by a factor of $\frac{1+\gamma^{H+1}}{1-\gamma^H} \geq 1$. This highlights the importance of accurate value estimation for TD-MPC. Given the same scale of improvement in value estimation, the H-step lookahead policy theoretically promises a greater policy improvement than the greedy policy.

B Algorithm Formulation and Implementation Details

Although our proposed method builds upon the TD-MPC framework and leverages the implementation of [14], we argue that our modifications are non-trivial and are carefully designed to address the key challenges mentioned in this paper. To further elaborate on this, we present the general formulation of constrained policy iteration, followed by a detailed discussion of its implementation and interaction with other critical components within TD-MPC.

Following [32], the constrained policy update step can be formulated as follows:

$$\pi_{k+1} \coloneqq \underset{\pi}{\operatorname{argmax}} \underset{a\sim\pi(\cdot|s)}{\mathbb{E}}[Q^{\pi_k}(a,s)], \quad s.t. \mathbf{D_{KL}}(\pi_{k+1}\|\mu_k) \leq \epsilon \quad , \tag{20}$$

while the policy evaluation operator follows the standard definition in policy iteration:

$$\mathcal{T}^\pi Q \coloneqq r(s,a) + \gamma \mathbb{E}_{s'\sim\rho(\cdot|s,a)} \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q^\pi(s',a')] \tag{21}$$

Given the behavior policy $\mu(\cdot|s) = \Sigma_{k=0}^K \omega_k \pi_{H,k}(\cdot|s)$, constraint policy improvement step can be considered as a trust region update [33] with the trust region near μ rather than π_{k-1}, the optimal solution of optimization problem (20) is a combination of behavior policy and Boltzmann distribution, with partition function $Z(s)$ and Lagrangian multiplier β:

$$\mu^*(a|s) \coloneqq \frac{1}{Z(s)}\mu(a|s)\exp(\frac{1}{\beta}Q(s,a)), \quad Z(s) \coloneqq \int_a \mu(a|s)\exp(\frac{1}{\beta}Q(s,a)) \tag{22}$$

In principle, the training objective can be formulated using either the reverse KL-divergence (RKL) $D_{KL}(\pi\|\mu^*)$, or forward KL-divergence (FKL) $D_{KL}(\mu^*\|\pi)$. While both have been explored in offline RL, FKL is significantly more prevalent, particularly in in-sample learning methods [32, 29, 20, 16]. In the online off-policy setting, however, beyond its well-known "zero-avoiding" behavior, prior study [4] have shown that FKL encourages exploration but does not guarantee policy improvement, often leading to degraded performance, especially under large entropy regularization. In C, we show that FKL could mitigate the value overestimation problem for high-dimensional tasks but may lead to training instability.

Based on these observations, we choose TD3-BC [6] style policy constraint. Instead of directly calculate the log-likelihood of μ, we maximize $\mathbb{E}_{\mu'\sim\{\mu_k\}}[\log \mu']$ as its lower bound. Such a surrogate greatly simplifies the calculation and keeps the modifications on top of TD-MPC2's code base simple.

In specific, the policy improvement step can be realized as:

$$\begin{aligned}&\pi_{k+1} \leftarrow \underset{\pi}{\operatorname{argmin}} \mathbb{E}_{(s,\mu)_{0:H}\sim\mathcal{B}}\Big[\sum_{t=0}^{H}\lambda^t \mathbb{E}_{a_t\sim\pi(\cdot|z_t)}\big[-Q^{\pi_k}(z_t,a_t)+\alpha\log(\pi(a_t|z_t))-\beta\log(\mu_t(a_t))\big]\Big],\\&z_0=h(s_0),\quad z_{t+1}=d(z_t,a_t)\end{aligned} \tag{23}$$

In addition, we notice that overly addressing policy constraints during the initial stage sometimes results in failure to reach out to the local minima. Thus, we maintain moving percentiles S_q for the Q function as in [15, 14]. This results in an adaptive curriculum on β:

$$\beta = \begin{cases} 0, & \text{if } S_q < s_{\text{threshold}} \\ \beta, & \text{otherwise} \end{cases}$$

For exploration-intensive tasks like `humanoid-run` in DMControl, this curriculum allows approximately 100k environment steps in the initial stage without constraints enforced on the nominal policy. This enables better exploration, helping the agent recover from the low-rewarded region. For most tasks with denser reward signals, its impact on performance is minimal.

We have the same training objective as TD-MPC2 when it comes to the dynamics model, reward model, and value function and encoder:

$$\mathcal{L} = \mathop{\mathbb{E}}_{(s,a,r,s')_{0:H}}\Big[\sum_{t=0}^{H}\gamma^t\Big(c_d\cdot\|d(z,a,e)-sg(h(s'_t))\|_2^2 + c_r\cdot CE(\hat{r}_t,r_t)+c_q\cdot CE(\hat{q}_t,q_t)\Big)\Big] \tag{24}$$

Where the reward function and value function's output are discretized and updated with cross-entropy loss given their targets.

Baseline Implementation In addition, we present detailed implementation for baseline variants used in section 6.3. We directly update the policy for the behavior cloning version ($\beta=\infty$) by maximizing the log-likelihood term. Following [15, 14], we also introduce a moving percentile S to scale the magnitude of the loss:

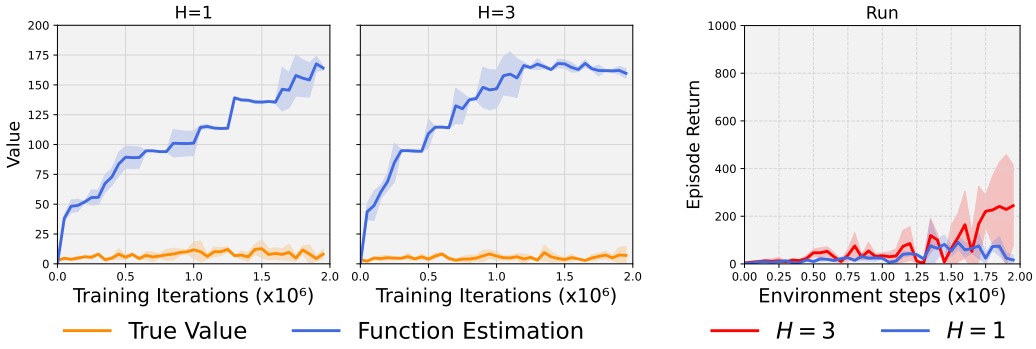
$$\mathcal{L}_\pi = \mathbb{E}_{s\sim\mathcal{B}}\mathbb{E}_{a\sim\pi(\cdot|s)}\log\mu(a\mid s)/\max(1,S) \tag{25}$$

C Discussion and Additional Results

Value Approximation Error. In Section 3.3, we illustrated value overestimation by comparing the true value estimate with the value function's estimate. The true value is approximated using Monte Carlo sampling as $\frac{1}{N}\Sigma_{n=1}^{N}[R(\tau_n^\pi)]$, where τ_n^π is trajectory following the nominal policy π. Unlike the approach to demonstrate overestimation in [7] that averages over states drawn i.i.d. from the buffer, we sample all trajectories starting from initial state $s_0\sim\rho_0$. Accordingly, the function estimation is calculated by averaging the action value following π at the initial state as $\mathbb{E}_{s\sim\rho_0,a\sim\pi(\cdot|s)}[\hat{Q}(s,a)]$. We argue that this approach more effectively illustrates the overestimation phenomenon. First, the data distribution in the replay buffer does not directly correspond to the current policy. Second, value approximation errors propagate through TD learning and accumulate at the initial state [38], making overestimation more pronounced and easier to observe.

In addition to Figure 1, we compare the approximation error between TD-MPC2 with a planning horizon of 1 and a horizon of 3 using `h1hand-run-v0` task. As shown in Figure 7, while both versions exhibit significant overestimation bias, the patterns of error growth differ. Over 2 million training steps, the error in the horizon-1 version grows nearly linearly, showing no clear trend of convergence. In contrast, although the horizon-3 version initially accumulates errors more rapidly, its error growth rate gradually decreases over time.

By combining Theorem A.1 and Theorem A.3, we know that π_H's dependency on value error is scaled by a factor of $\frac{\gamma^{H-1}(1-\gamma)}{(1-\gamma^H)}$ relative to the greedy policy's dependency on value error. Consequently, given the same $\hat{V}$, we expect the performance gap between the H-step lookahead policy $\pi_{H,k}$ and the greedy policy π_{k+1} to be smaller in the early stages of training, which aligns with the lower approximation error initially observed. However, according to Theorem 4.1, shorter horizons amplify the error accumulation term, resulting in a faster growth rate. Therefore, this empirical observation further supports our theoretical analysis.



(a) Value Estimation (b) Performance

Figure 7. TD-MPC2 ablation results of horizon at `h1hand-run-v0`. (a) Value estimation error with different horizons; (b) Episode return with different horizons. Variant with longer horizon shows convergent error growth pattern and better performance with more training steps.

Model Bias. In addition to direct OOD query, model bias[21] is also considered an essential source of extrapolation error for offline RL: Due to a limited number of transitions contained in the training dataset, TD-target does not strictly reflect an estimation of real transition. For an online off-policy problem, the problem is not critical since the buffer is, in fact, continuously updated.

Pessimistic Policy Training. Offline RL algorithms aim to stabilize the learning process and improve policy performance by carefully handling unseen data. However, we argue that not all offline RL methods are well-suited for the TD-MPC setting. Despite the theoretical equivalence, we empirically found that the FKL algorithm performs worse than bc-constrained RKL policy learning. In Figure 8, *AWAC-MPC* refers to the variant that employs AWAC [29] for constrained policy iteration. Its implementation is based on CORL. These findings are aligned with [31], which demonstrates advantage of bc-constrained policy learning over AWR/AWAC due to the encourage of mode-seeking.

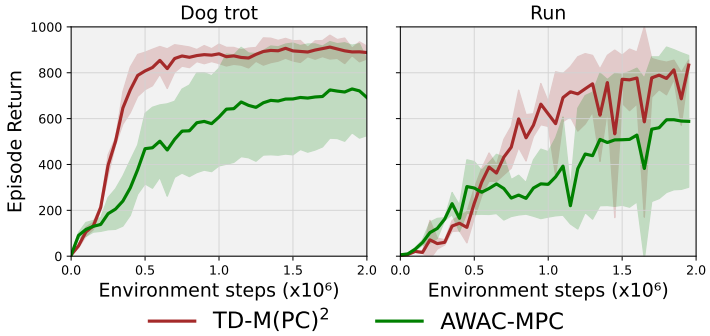


Figure 8. **Constrained policy update through AWAC** Mean and 95% CIs across 3 ramdom seeds on high-dimensional tasks.

Moreover, in scenarios where exploration is critical, intuitively we do not recommend employing conservative Q-learning methods [22]. Such methods are designed to penalize the Q-values of out-of-distribution (OOD) actions, ensuring that the agent remains within the boundaries of the training data. While this helps to prevent overestimation, it may introduce a significant drawback: a consistent underestimation of the overall Q-value function. This underestimation not only out-of-distribution data but also reduces the scale of the Q-values overall[30]. As a result, value-guided planning becomes excessively cautious, disincentivizing the selection of novel actions outside the buffer. This overly conservative behavior severely limits the agent's ability to explore, which, however, is a key aspect of online reinforcement learning. We favor TD3-BC[6] or BC-SAC[27] style algorithm for this particular problem setting.

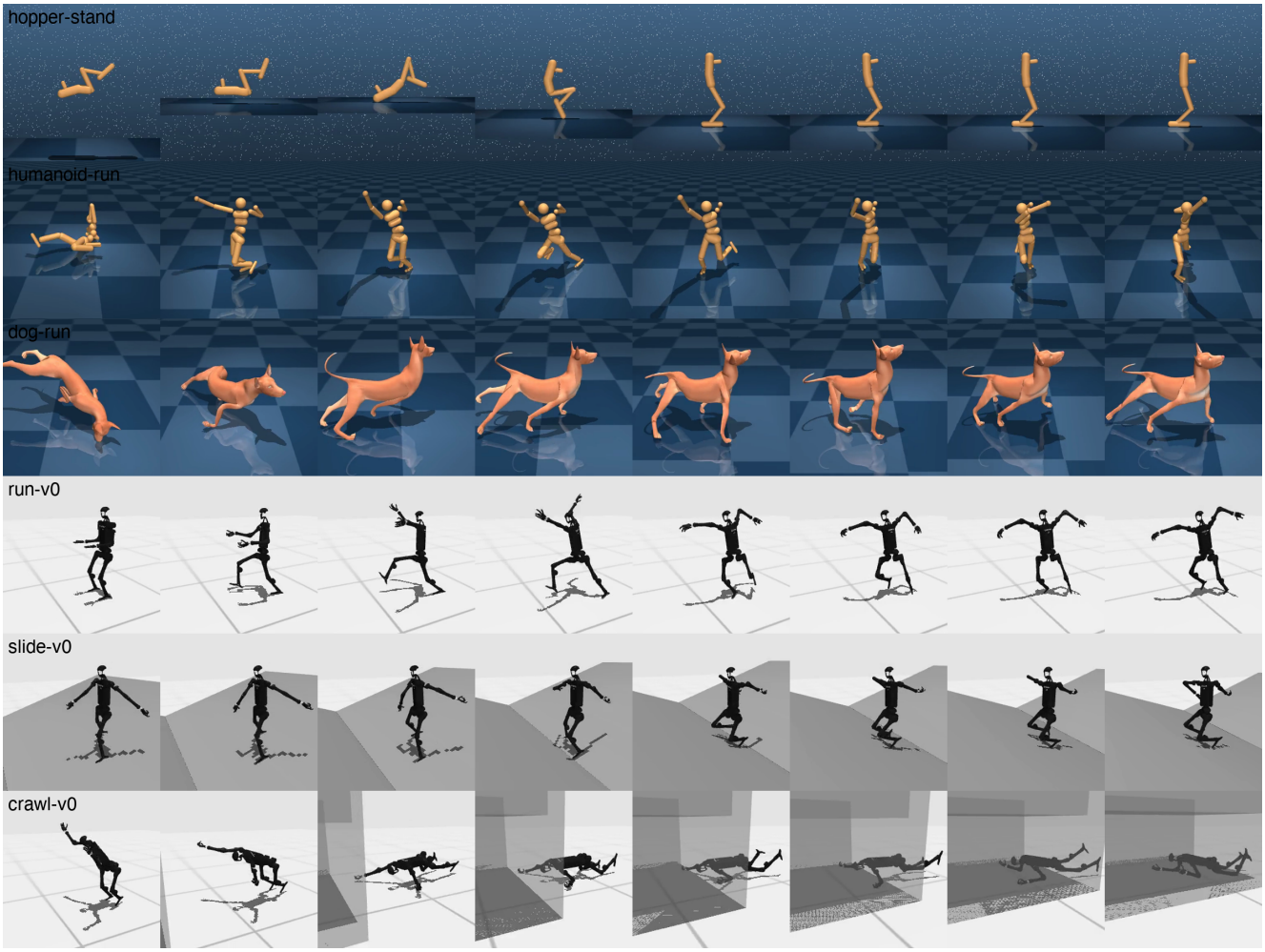


Figure 9. **Visualizations.** We demonstrate trajectories generated by our method 6 tasks across two benchmarks (DMControl and HumanoidBench) as qualitative results; tasks are listed as follows: `Hopper-stand` ($\mathcal{A} \in \mathbb{R}^4$), `Humanoid-run` ($\mathcal{A} \in \mathbb{R}^{21}$), `Dog-run` ($\mathcal{A} \in \mathbb{R}^{36}$), `h1hand-run-v0` ($\mathcal{A} \in \mathbb{R}^{61}$), `h1hand-slide-v0` ($\mathcal{A} \in \mathbb{R}^{61}$), `h1hand-crawl-v0` ($\mathcal{A} \in \mathbb{R}^{61}$).

Table 1. Hyperparameter settings. We directly apply settings in [14] for the shared hyperparameters without further tuning. We share the same setting across all tasks demonstrated before.

Hyperparameter	**Value**
Training	
Learning rate	3×10^{-4}
Batch size	256
Buffer size	1_000_000
Sampling	Uniform
Reward loss coefficient (c_r)	0.1
Value loss coefficient (c_q)	0.1
Consistency loss coefficient (c_d)	20
Discount factor (γ)	0.99
Target network update rate	0.5
Gradient Clipping Norm	20
Optimizer	Adam
Planner	
MPPI Iterations	6
Number of samples	512
Number of elites	64
Number policy rollouts	24
horizon	3
Minimum planner std	0.05
Maximum planner std	2
Actor	
Minimum policy log std	-10
Maximum policy log std	2
Entropy coefficient (α)	1×10^{-4}
Prior constraint coefficient (β)	1.0
Scale Threshold (s)	2.0
Critic	
Q functions Esemble	5
Number of bins	101
Minimum value	-10
Maximum value	10
Architecture(5M)	
Encoder layers	2
Encoder dimension	256
MLP hidden layer dimension	512
Latent space dimension	512
Task embedding dimension	96
Q function drop out rate	0.01
MLP activation	Mish
MLP Normalization	LayerNorm