Model Predictive Actor-Critic: Accelerating Robot Skill Acquisition with Deep Reinforcement Learning

Andrew S. Morgan[1]*, Daljeet Nandha[2]*, Georgia Chalvatzaki[2], Carlo D'Eramo[2], Aaron M. Dollar[1], and Jan Peters[2]

***Abstract*— Substantial advancements to model-based reinforcement learning algorithms have been impeded by the model-bias induced by the collected data, which generally hurts performance. Meanwhile, their inherent sample efficiency warrants utility for most robot applications, limiting potential damage to the robot and its environment during training. Inspired by information theoretic model predictive control and advances in deep reinforcement learning, we introduce Model Predictive Actor-Critic (MoPAC)[†], a hybrid model-based/model-free method that combines model predictive rollouts with policy optimization as to mitigate model bias. MoPAC leverages optimal trajectories to guide policy learning, but explores via its model-free method, allowing the algorithm to learn more expressive dynamics models. This combination guarantees optimal skill learning up to an approximation error and reduces necessary physical interaction with the environment, making it suitable for real-robot training. We provide extensive results showcasing how our proposed method generally outperforms current state-of-the-art and conclude by evaluating MoPAC for learning on a physical robotic hand performing valve rotation and finger gaiting–a task that requires grasping, manipulation, and then regrasping of an object.**

I. Introduction

Robotic systems are expected to operate in increasingly unstructured and dynamical environments. Aside from the difficulties evident in perception, decision making, and planning, precisely controlling robots still remains difficult, as they must operate under non-linear, contact-rich conditions. Traditional approaches to this problem include methods of optimal control [1], [2], but require that a model of the robot and its environment is known *a priori*, which generally cannot be guaranteed.

In this direction, model-based reinforcement learning (MBRL) approximates iteratively the dynamics model of the environment while planning actions through trajectory optimization [3]–[8], commonly using Model Predictive Control (MPC) [9]–[11]. Though sample-efficient, MBRL has been heavily impeded by the bias in the learned model–as optimal control methods will generally continue to exploit known regions of the learned dynamics without exploring outwards to new, unrevealed states [12]–[14]. Model-free reinforcement learning (MFRL), on the other hand, has achieved impressive results in learning complex skills through deep neural networks [15]–[19], offering high performance. However, unlike their MB counterpart, the lack of an internal model makes MF algorithms data-hungry, suffering from poor use of samples; as the complexity of the task increases, so does the number of samples required to learn the optimal policy [20]–[22]. This disadvantage makes modern deep MFRL methods inappropriate for learning tasks on physical robots [23]–[25], as the increased amount of necessary interactions is likely to damage the system.

We believe that the effective combination of these RL paradigms will enable the learning of complex skills on real

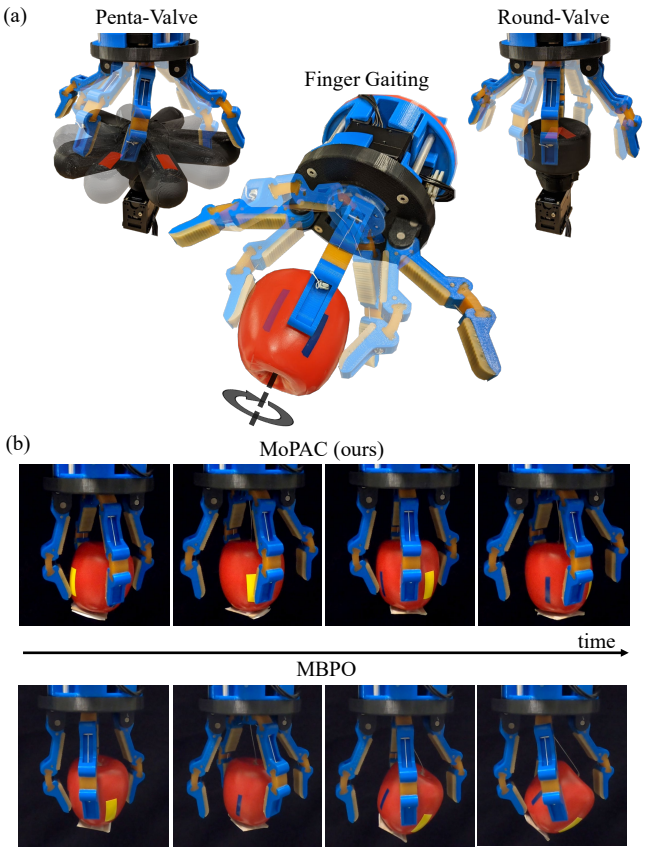

Fig. 1. (a) Complex robot skills, like in-hand manipulation, are difficult to acquire and perform with conventional methods. (b) MoPAC is suitable for training an underactuated hand to efficiently learn such tasks, like finger gaiting, compared to other state-of-the-art techniques.

Computations performed on the Lichtenberg cluster of TU Darmstadt and the Yale HPC Grace cluster. This work has been partially funded by the RoboTrust, Skill4Robots projects and NSF Grants IIS-1752134 & IIS-1900681. Dr. Chalvatzaki is funded by the DFG EN Program (CH 2676/1-1).

[1]Department of Mechanical Engineering & Materials Science, Yale University, USA. ({andrew.morgan, aaron.dollar}@yale.edu).

[2]Intelligent Autonomous Systems, Technische Universität Darmstadt, Germany (daljeet.nandha@stud.tu-darmstadt.de, {georgia, carlo}@robot-learning.de, mail@jan-peters.net).

*Authors contributed equally.

[†]Code available: https://github.com/dnandha/mopac.

robots [26], [27]. In this paper, we address this challenge by investigating optimal methods of guiding policy learning, while also learning the model dynamics. Intuitively, having interweaving MB-MF components can potentially allow scheduling of separate solutions according to different phases of learning, i.e. MB when action planning is required, and MF when additional exploration is needed.

Various works try to mix the benefits of MB and MF methods. For example, [28] proposes the use of trajectory optimization for guiding the policy search by exploring high-reward regions. In [29], the authors propose a MB method which learns an ensemble of models and optimizes a meta-policy objective over all models. [30] proposes the initialization of MFRL algorithms from learned model dynamics, for combining the sample efficiency of MB methods with the task-specific performance of MF approaches. Many works study the use of MB methods for accelerating the learning of MFRL. The latter can be extended into MB methods by sampling from a model (or model ensemble to increase stochasticity) [11], [31], [32]. MBPO [14] uses branched rollouts in an actor-critic setting to exploit the learned dynamics for model-based policy optimization. [33] proposes an extension to MBPO, by performing model rollouts of specific horizons, while optimizing the policy objective with back-propagation through time.

In this work, we introduce Model Predictive Actor-Critic (MoPAC)–an algorithm that seamlessly combines the sample efficiency of MBRL, with MF actor-critic methods for improved exploration. To ensure exploitation of the learned policy, we propose the use of model predictive rollouts, a method that inherits properties of model predictive path integrals [10], capitalizing on the free-energy of the learned system and leveraging the information theoretic constraint in the MB simulations. Our novel method is theoretically sound, as we provide a bound on the performance of trajectory optimization through MPC, when approximating the dynamics model and the value-function, that potentially allows planning for longer time-horizons. Moreover, the maximum entropy objective [24] in the policy optimization counterbalances the exploitation of model predictive rollouts with exploration on the real environment, enabling the learning of more expressive model dynamics, together with approximating the optimal policy. Our empirical results, both in simulated control tasks and on a physical robotic hand that performs in-hand manipulation, showcase the accelerated learning that MoPAC offers against representative baselines, evincing its effectiveness for learning complex skills on real-robot platforms.

II. Preliminaries

We consider *Reinforcement Learning* [34] for solving control problems modeled as finite-horizon Markov Decision Processes (MDPs) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, and $\gamma \in [0, 1)$ is the discount factor. We define a policy $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the probability distribution of the event of executing an action a in a state s. A policy π induces a value function (VF) corresponding to the expected cumulative discounted reward collected by the agent when executing action a in state s, and following the policy π thereafter: $Q^\pi(s, a) \triangleq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{i+k+1} | s_i = s, a_i = a, \pi\right]$, where r_{i+1} is the reward obtained after the i-th transition. Solving an MDP consists of finding the optimal policy π^*, i.e. the one maximizing the expected cumulative discounted reward.

Trajectory Optimization designs a trajectory that minimizes some measure of performance. MPC is such a technique that optimizes a cost function over a finite time-horizon, while taking into account the system dynamics. The cost function is optimized w.r.t. a control variable, yielding the optimal control value for a given state with consideration of the predicted future states. In essence, MPC provides a locally optimal policy or sequence of actions (up to horizon H), based on the following optimization problem:

$$\pi_{MPC}(s) = \arg\max_{\pi_{0:H-1}} \mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) + \gamma^H r_f(s_H)] \quad a_t = \pi_t(s_t), s_0 = s \tag{1}$$

where the states evolve according to the transition dynamics of the MDP, i.e. $s_{t+1} = f(s_t, a_t)$. From each optimized sequence resulting from the optimization process of MPC (of length equal to horizon H), the first action is applied to the agent, and the procedure is repeated again at the next time step. The term $r_f(s_H)$ denotes the terminal reward.

In the context of MBRL, [35] combines MB trajectory optimization with VF estimation. [36] uses model predictive path integrals [37] in a Q-learning setting, and [38] shows that combining MPC with VF approximation yields optimal policies, however considering the true dynamics.

III. Performance bound under imperfect model

Coupling MPC with a VF, that propagates global information, optimizes action sequences for longer time horizons, while being less prominent to approximation errors than greedy action selection [38]. In this work, we design an actor-critic algorithm that will benefit from the integration of trajectory optimization for acquiring optimal control policies, together with the learning of stable VF approximations provided by modern deep actor-critic algorithms, e.g. soft actor-critic (SAC) [24]. To do so, we extend the bound of [38] by incorporating the approximation error when learning the dynamics model.

Theorem 1. *Let the approximation error of the dynamics model be $\epsilon_f = |\hat{f}(s, a) - f^*(s, a)|$, the approximation error of the VF be $\epsilon_V = max_s |\hat{V}(s) - V^*(s)|$, and the terminal reward of (1) be $r_f(s_H) = \hat{V}(s_H)$, then the performance of the MPC policy in (1) with the learned dynamics model is bounded by:*

$$J(\pi^*) - J(\pi^{MPC}) \leq \frac{2\gamma^H \epsilon_V}{1 - \gamma^H} + r_{max} \frac{1 - \gamma^H}{1 - \gamma} \epsilon_f. \tag{2}$$

Proof is provided in the Appendix. As the contribution of the model error increases with the horizon, $H \to \infty$ leads to the upper bound

$$J(\pi^*) - J(\pi^{MPC}) \leq \frac{r_{max} \epsilon_f}{1 - \gamma}. \tag{3}$$

Algorithm 1 Model Predictive Actor-Critic

Initialize parameters θ, ρ, ψ, $\bar{\psi}$, ϕ, $\bar{\phi}$
Initialize D, D_{env}, D_{model} ▷ initialize experience buffers
$\bar{\phi} \leftarrow \phi$, $\bar{\psi} \leftarrow \psi$ ▷ initialize target parameters
for each iteration **do**
  $D_{env} \leftarrow D_{env} \cup \{s_{t+1}, a_t, s_t\}, a_t \sim \pi_\theta(s_t)$
  for N epochs **do**
    Train model f_ρ on D_{env}: $\rho \leftarrow \rho - \lambda_f \nabla_\rho J_{f_\rho}$
  end for
  for M model predictive rollouts **do**
    Sample s_t uniformly from D_{env}
    Perform MPR (Alg. 2) from s_t
    Add transitions to D_{model}
  end for
  for G gradient steps **do**
    Update parameters using data $D \leftarrow D_{env} \cup D_{model}$
    $\psi \leftarrow \psi - \lambda_\psi \nabla_\psi J_{V_\psi}$ ▷ VF update
    $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi J_{Q_\phi}$ ▷ Q update
    $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta J_{\pi_\theta}$ ▷ policy update
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\psi$ ▷ target VF update
    $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\phi$ ▷ target Q update
  end for
end for

As expected, the performance error between the optimal policy and the MPC policy is affected by the model approximation error ϵ_f, given a prediction horizon H. As we will show in the following, the maximum entropy exploration of SAC [24] can acquire more expressive dynamics models by visiting unmodeled transitions in the environment; together with the approximation of the VF, we can leverage the bound of (2) to acquire near-optimal trajectories for policy learning.

Model-based monotonic improvement, as proven by [14], can be achieved when learning the dynamics model together with the policy. Namely, the authors give an upper-bound in the performance gain obtained when applying the learned policy to the learned dynamics model, compared to applying it on the real MDP (i.e. the true dynamics). Their finding is summarized in the following lemma.

Lemma 2. *Let the expected TV-distance error of the transition probability distributions be bounded by ε_f and the policy divergence be bounded by ε_π. Then the following bound holds:*

$$J(\pi) - \hat{J}(\pi) \geq -\left[\frac{2\gamma r_{max}(\varepsilon_f + 2\varepsilon_\pi)}{1-\gamma^2} + \frac{4 r_{max}\varepsilon_\pi}{1-\gamma}\right]. \quad (4)$$

This lemma is directly applicable to our proposed algorithm. Its combination with our Theorem 1 suggests that sufficiently low errors in model learning and policy approximation can yield near-optimal performance.

IV. Model Predictive Actor-Critic

We introduce Model Predictive Actor-Critic (MoPAC), an algorithm that leverages the theoretical guarantees provided by Theorem 1 and Lemma 2. MoPAC has three main interacting components: (i) model learning from environment transitions, (ii) model predictive rollouts for acquiring samples from optimal trajectories, and (iii) soft updates using a maximum entropy objective for policy learning over a mixture of model and environment data. Algorithm 1 summarizes our approach.

Algorithm 2 Model Predictive Rollouts (MPR)

Input: s_0, ρ, $\bar{\psi}$, π_θ, n, ζ, γ, λ ▷ start state, model parameters, target VF parameters, current policy, actuation noise, annealing parameter, discount, control hyperparameter
Output: a_{MPR}, s_{MPR} ▷ MPR trajectory
$H \leftarrow anneal(\zeta)$ ▷ horizon length
$\mathcal{R} \leftarrow zeros(\cdot)$ ▷ trajectory reward
for $t = 0, ..., H-1$ **do**
  Sample an initial action from policy $a_t \sim \pi_\theta(s_t)$
  Sample exploration noise $\{n_0, \ldots, n_{H-1}\} \sim n$
  $\mathcal{R} \leftarrow \mathcal{R} + \gamma^t r(s_t, a_t + n_t)$ ▷ rewards
  $s_{t+1} \leftarrow f_\rho(s_t, a_t + n_t)$ ▷ system dynamics
end for
$\mathcal{R} \leftarrow \mathcal{R} + \gamma^H V_{\bar{\psi}}(s_H)$ ▷ add terminal state reward
$\mathcal{C} = -\mathcal{R}$ ▷ convert to cost minimization
$\beta \leftarrow min[\mathcal{C}]$
$\eta \leftarrow \sum_{i=1}^{N} \exp(-\frac{1}{\lambda}[\mathcal{C}_i - \beta])$
$w(n) \leftarrow \frac{1}{\eta}\exp(-\frac{1}{\lambda}[\mathcal{C} - \beta])$ ▷ import. sampling weight
$a_{MPR} \leftarrow a + \sum_{i=1}^{N} w(n_i) n_i$ ▷ adjust action sequence
$s_{MPR} \leftarrow f_\rho(s_0, a_{MPR})$ ▷ state following optimal action

Model learning. We use an ensemble of N functions to approximate the model of the environment $\{f_{\rho_1}, ..., f_{\rho_N}\}$. Each of these functions is a probabilistic deep neural network whose purpose is to approximate the system dynamics, namely the next state of the agent given the current state and action. Probabilistic model ensembles have been studied as a way of realizing Bayesian neural networks, for capturing the epistemic uncertainty when learning complex dynamics, in order to mitigate overfitting when using a single model [11]. Specifically, an ensemble of models is randomly initialized within an observable space; hence, each model learns a different mapping of the dynamics. For generating predictions from the ensembles, we sample transitions from the elite networks, i.e. those with the lowest L2-loss in a validation set, during the model rollouts by sampling uniformly a model for every simulation.

Model predictive rollouts. Solving the MPC optimization problem of (1) is prohibitively expensive and hard to obtain online. This limitation is overcome by the information theoretic model predictive path integral (i-MPPI) method [10], a sampling-based algorithm that uses an approximation of the true dynamics, and is able to optimize both convex and non-convex cost criteria, thus being applicable to large classes of stochastic systems and representations. i-MPPI uses the free energy of the system and relative entropy (KL-divergence) for providing generalized path integral expressions. Crucially, it uses importance sampling for acquiring the optimal control paths. In MoPAC, we use a similar setting for generating model-based rollouts that will profit from the optimal performance guarantees of Sec. III. In our model predictive rollouts (MPR), we initialize the control sequence employing the learned policy π_θ to sample initial actions. Then, we design a similar setup as in i-MPPI for creating trajectory simulations in a specific time horizon H, starting from an initial true state (Alg. 2). We evaluate the rewards of the rollouts, using also a VF for the final state (2), and we perform importance sampling of the optimal transitions over all simulations, along with

an information-theoretic update of the actions' exploration noise. Finally, we collect the resulting optimal transitions in the model-based replay memory D_{model}.

When the approximation error of the learned model is low, the trajectory optimization on the learned model performed by MoPAC yields comparable performance as acting on the real environment (Lemma 2). This remark, combined with the use of the VF, allows us to obtain near-optimal performance–up to an approximation error (Theorem 1). Moreover, using the learned policy distribution for sampling the initial control sequence, together with the relative entropy objective, guarantees that we are simulating transitions in a reasonable area around the learned policy, providing good trajectories for exploitation, while the underlying actor-critic explores new transitions.

Soft policy optimization. As the underlying actor-critic algorithm in MoPAC, we adopt SAC [24] to benefit from the exploration induced by the soft policy updates based on the maximum entropy principle, counterbalancing the effect of the exploitation induced by MPRs. Nevertheless, MoPAC can be applied to any known off-policy actor-critic algorithm. In SAC, the training of the policy alternates between a soft policy evaluation step based on the soft Bellman backup operator [25], and a soft policy improvement step that minimizes the expected KL divergence: $J_\pi(\theta, D) = \mathbb{E}_{s_t \sim D}[D_{KL}(\pi || \exp\{Q^\pi - V^\pi\})]$, where Q^π, V^π are the soft Q-function and soft VF of policy π respectively.

V. Experimental results

A. Simulated tasks

We evaluate MoPAC in the simulated control tasks of MuJoCo [39] included in the OpenAI-gym library [40]: `HalfCheetah-v2`, `Ant-v2`, `Hopper-v2`, and `Walker2d-v2`. We compare the average return of MoPAC over 5 trials, consisting of $1,000$ true environment interactions per epoch, against the baselines SAC [25], MBPO [14], and MBRL [10]. We decided not to compare with [33] as the absence of code and a detailed algorithm, makes the reproduction of their results prohibitive. Here, SAC and MBPO are trained according to the hyperparameter settings provided by their respective works, while for MBRL we use the same settings as MoPAC, namely using horizons $5 - 15$ with linear annealing for all tasks. The number of environment interactions per episode is constant across all algorithms. A batch size of $10,000$ is chosen for the model rollouts in both MoPAC and MBPO.

Fig. 2 shows the average return of each algorithm per environment, plotted w.r.t. the number of epochs. In all environments MoPAC and MBPO outperform both SAC, evincing the advantage of using model rollouts, and MBRL, that strongly suffers from poor exploration in the use of the learned model. Notably, MoPAC learns *faster* than MBPO. Specifically, we observe a significant speed-up in the learning of `HalfCheetah-v2` and `Walker2d-v2`. The `Ant-v2` and `Hopper-v2` are more challenging tasks, as they require more interactions with the environment to learn their dynamics; in `Hopper-v2` we observe a speedup in learning and convergence, while in `Ant-v2` MoPAC learns faster in the first epochs and ends up with slightly better performance than MBPO. We expect that by further tuning the prediction horizons of our MPR, as now we use the same horizons across all tasks regardless of task-complexity, will result in increased performance.

B. Robotic tasks

We further underscore the efficacy of MoPAC by comparing our algorithm with SAC and MBPO on a Yale Openhand Model Q [41], [42] through two different manipulation tasks–valve rotation and finger gaiting. The Model Q is an open source underactuated hand with four two-link fingers and four total actuators (Dynamixel XM-430). Within the hand, a single motor actuates two opposing fingers that are coupled by a differential, allowing passive reconfigurability between the fingers. Two additional motors actuate the remaining two fingers individually. The fourth and final motor serves to rotate the palm of the hand, allowing the two coupled fingers to reorient perpendicular to the palm axis (Fig. 1). Being underactuated, the Model Q's joint configuration cannot be accurately determined, as with many soft, compliant, or underactuated hands, this hand is not equipped with joint encoders or tactile sensors [43]. This inherent compliance suits the spirit of our evaluations well, as the reconfigurability of the fingers presents added challenges in sufficiently learning the dynamics of the environmental interactions.

Valve Rotation. We develop two different valves, the penta-valve and the round-valve (resembling the geometry of a door knob), for evaluation (Fig. 1). Each of the valves were connected to a current-disabled Dynamixel actuator placed directly below the hand as to measure rotation via the encoder. Albeit disabled, valve rotation was resisted by the $353.5:1$ gearbox inside of the actuator, generally limiting rotation to only large actions from the hand (see twisting of flexure joints in Fig. 1). Policies were trained for both valves with each of the three aforementioned algorithms on the physical system. The goal of this task was to continually rotate the valve counterclockwise until episode completion (50 actions). We define a standardized reward function, $r(\cdot) = \theta_{s'} - \theta_s$, and run each of the algorithms with similar hyperparameters and epoch lengths (50 interactions per episode, 5 episodes per epoch, MPR horizons $2 - 5$ annealed).

Experimental results to these tasks, depicted in Fig. 2(e-g), illustrate the benefit of our MoPAC algorithm compared to baselines. Notably, the reward convergence of the round-valve was less than that of the penta-valve, generally due to the increased difficulty in estimating the task's dynamical hand-object nature, since the round-valve requires grasping before rotation. Due to the design of the hand's flexure joints, the motors are decreasingly able to provide rotational torque the more a finger is actuated, i.e. off-axis torsion limits the amount of force the finger-contacts can transmit to rotate the object. This further leads to a grasping and rotational action that, if enacted with rigid joints instead of compliant ones, would result in a reward from the valve rotation, but does not necessarily happen with flexure joints. Generally,

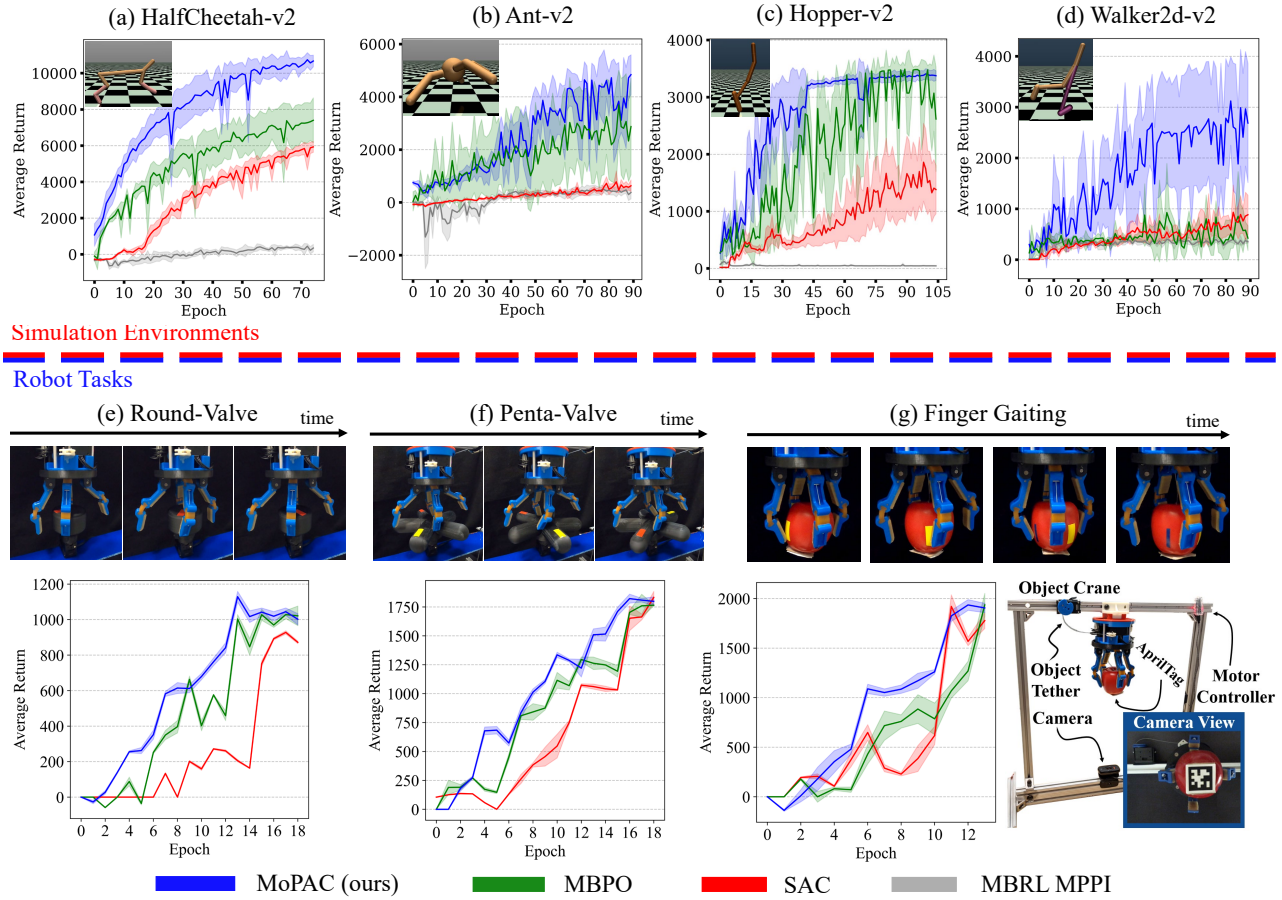

Fig. 2. Experimental results comparing MoPAC to other baselines in both MuJoCo environments (a-d) and robotic manipulation tasks (e-g). Results on MuJoCo tasks are averaged over 5 separate experiments, while for the robotic manipulation tasks we learn a single policy and average its performance on 5 evaluation runs. All plots show the 95% confidence intervals. The automated object reset system is depicted in the bottom right.

the function of mapping grasping force and palm orientation to valve rotation must first be inferred by the dynamics model in order to sufficiently learn the task.

Albeit in some ways easier to estimate, hand-object dynamics of the penta-valve is also not deterministic, as the fingers interact with both the fingerpad (rubber) and side of the finger (plastic), where slipping can occur. This task noted a higher converging reward comparatively, as it was not required that the valve be grasped for rotation, dismissing off-axis torsion constraints. Towards the beginning of training, the hand utilized the two independent, non-rotational fingers to achieve a greedy reward, i.e. small movement given the valve configuration, but soon thereafter noted that retreating the fingers into a non-actuated position generally benefited the cumulative reward as to limit interference with the valve.

Finger Gaiting. In addition to the valve rotation assessment, we further challenged our algorithm with the task of finger gaiting–a task requiring coordinated movement between the opposing finger pairs in the hand. Here, the hand is not only expected to maintain a stable grasp on the object during the entire episode, but to rotate the object about the palm axis of the hand (Fig. 1). We incorporated a similar reward function as in the previous section, $r(\cdot) = \theta_{s'_{palm}} - \theta_{s_{palm}}$, but now the object is not constrained to only reorient along the reward axis, being free to move in $SE(3)$ space. In this evaluation, we utilize the apple (Obj. #13) from the YCB Object and Model Set [44] as to maintain standardization of the task.

The episode starts with a stable grasp acquired by the two individually driven fingers, with the coupled fingers close to, but not touching, the object. We monitor the pose of the object during manipulation via an AprilTag [45] attached to the bottom of the apple, and detected by an external camera. Subsequently, the reward of an action can be calculated through this setup, in addition to detecting when the hand drops the object. When a drop is detected, the episode is ended and the object is systematically reset via an automated object reset system (Fig. 2). This system consists of an object crane that controls a tether routed through the center of the hand. During reset, the object lifts the object into the palm of the hand as to stabilize, and then slowly lowers the object into the starting position.

In this evaluation, we note similar curves to those in the valve rotation task (Fig. 2). Specifically, we see MoPAC outperform the two other algorithms, reaching convergence faster with fewer environment interactions. Intuitively, this quality is increasingly advantageous for tasks where system reset cannot be completed quickly or autonomously–although it was possible in this experiment, object reset required about

12 seconds and slowed the learning process. Training for the valve rotation tasks took ~ 2 hours for each valve and each learning algorithm, whereas training for finger gaiting took ~ 5 hours. Notably, if we were to stop training upon MoPAC convergence, significant time can be saved, which is especially desirable when increasing task complexities.

VI. Conclusions

Transferring the advances of deep RL into real-world robotic problems is challenging. Deep model-free RL (MFRL) methods, though able to learn complex skills, typically require an excessive amount of interactions with the environment, making their applicability prohibitive for robotics. On the other hand, model-based RL (MBRL) approaches learn the dynamics model and select actions through trajectory optimization techniques; albeit sample-efficient, they suffer from local optima. In this work, we proposed Model Predictive Actor-Critic (MoPAC), a method seeking to combine the advantages of deep MF actor-critic methods with the data-efficiency of MB approaches. In particular, MoPAC introduces model predictive rollouts, inspired by information-theoretic model predictive path integrals, based on the principles of the dynamics free-energy and on an information-theoretic constraint for collecting samples through trajectory optimization. MoPAC uses a MFRL actor-critic algorithm for policy improvement and model learning, and it uses the model for performing the model predictive rollouts to collect additional samples for guiding policy learning. The core advantage of MoPAC is that, though MB rollouts favor policy exploitation through planning using the model, the MFRL actor-critic encourages efficient exploration for policy optimization and model learning.

Our model predictive rollouts are backed up by a performance bound, which guarantees that sufficiently low errors in the value function, and model approximation, yields near-optimal performance. We empirically demonstrate the efficiency of MoPAC in providing accelerated policy learning for simulated control tasks against representative baselines. Furthermore, we showcase the applicability of MoPAC for learning challenging in-hand manipulation tasks with a four-fingered robotic hand. In the future, we will study ways of principally adjusting the mixing of the MB-MF samples across the training process, but also ways of scheduling the MB over the MF method according to the learning progress, and vice versa.

VII. Appendix

Proof of Theorem 1. Let the performance of applying policy $\hat{\pi}$ from MPC using the approximated model $\hat{f}$ be denoted as $\hat{V}$ and the performance gain of applying the optimal policy π^* on the perfect model f^* be V^* over a planning horizon H. The performance error for any given starting state s is

$$V^*(s) - \hat{V}(s) = \sum_{s\sim f^*}[\sum_{t=0}^{H-1} f^*(s_t,a_t)\gamma^t r_t + \gamma^H V^*(s_H)] - \sum_{s\sim \hat{f}}[\sum_{t=0}^{H-1} \hat{f}(s_t,a_t)\gamma^t r_t + \gamma^H \hat{V}(s_H)]. \quad (5)$$

Adding and subtracting $\sum_{s\sim\hat{f}}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t r_t + \gamma^H V^*(s_H)]$in (5) yields

$$V^*(s) - \hat{V}(s) = \gamma^H \sum_{s\sim\hat{f}}[V^*(s_H) - \hat{V}(s_H)] + \sum_{s\sim f^*}[\sum_{t=0}^{H-1} f^*(s_t,a_t)\gamma^t r_t + \gamma^H V^*(s_H)] - \sum_{s\sim\hat{f}}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t r_t + \gamma^H V^*(s_H)]. \quad (6)$$

Since the value function error is upper-bounded by $max_s|V^*(s) - \hat{V}(s)| = \epsilon_V$, we can bound the following relations as

$$\sum_{s\sim f^*}[\sum_{t=0}^{H-1} f^*(s_t,a_t)\gamma^t r_t + \gamma^H V^*(s_H)] \leq \sum_{s\sim f^*}[\sum_{t=0}^{H-1} f^*(s_t,a_t)\gamma^t r_t + \gamma^H \hat{V}(s_H)] + \gamma^H\epsilon_V \quad (7)$$

$$\sum_{s\sim\hat{f}}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t r_t + \gamma^H V^*(s_H)] \geq \sum_{s\sim\hat{f}}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t r_t + \gamma^H \hat{V}^(s_H) - \gamma^H\epsilon_V. \quad (8)$$

Substituting (7) into (6), we have

$$\begin{aligned}
V^*(s) - \hat{V}(s) &\leq \gamma^H \sum_{s\sim\hat{f}}[V^*(s_H) - \hat{V}(s_H)] + 2\gamma^H\epsilon_V \\
&+ \sum_{s\sim f^*}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t r_t - \sum_{s\sim\hat{f}}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t r_t \\
&\leq \gamma^H \sum_{s\sim\hat{f}}[V^*(s_H) - \hat{V}(s_H)] + 2\gamma^H\epsilon_V \\
&+ r_{max}\{\sum_{s\sim f^*}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t] - \sum_{s\sim\hat{f}}[\sum_{t=0}^{H-1}\hat{f}(s_t,a_t)\gamma^t]\} \\
&= \gamma^H \sum_{s\sim\hat{f}}[V^*(s_H) - \hat{V}(s_H)] + 2\gamma^H\epsilon_V + r_{max}\sum_{t=0}^{H-1}\gamma^t\epsilon_f \\
&\leq 2\gamma^H\epsilon_V(1 + \gamma^H + \gamma^{2H} + ...) + r_{max}\sum_{t=0}^{H-1}\gamma^t\epsilon_f \\
&\leq \frac{2\gamma^H\epsilon_V}{1-\gamma^H} + r_{max}\sum_{t=0}^{H-1}\gamma^t\epsilon_f = \frac{2\gamma^H\epsilon_V}{1-\gamma^H} + r_{max}\frac{1-\gamma^H}{1-\gamma}\epsilon_f \quad (9) \\
&\leq \frac{r_{max}\epsilon_f}{1-\gamma}. \quad (10)
\end{aligned}$$

For MPC with imperfect dynamics and prediction horizon H, the bound (9) holds, showcasing the inevitable error due to the model approximation. However, this relation can be upper-bounded by the quantity in (10) considering an infinite-horizon, $H \to \infty$, prediction. The bound of (9) shows that sufficiently low approximation errors in the VF and the model can yield near-optimal performance, which is related to the prediction horizon and the discounted factor.

REFERENCES

[1] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Advances in neural information processing systems*, 2008, pp. 1465–1472.

[2] C. G. Atkeson, "Using local trajectory optimizers to speed up global optimization in dynamic programming," in *Advances in neural information processing systems*, 1994, pp. 663–670.

[3] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[4] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 378–383.

[5] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees," *arXiv preprint arXiv:1807.03858*, 2018.

[6] G. Chalvatzaki, X. S. Papageorgiou, P. Maragos, and C. S. Tzafestas, "Learn to adapt to human walking: A model-based reinforcement learning approach for a robotic assistant rollator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3774–3781, 2019.

[7] T. Wang and J. Ba, "Exploring model-based planning with policy networks," *arXiv preprint arXiv:1906.08649*, 2019.

[8] A. S. Morgan, K. Hang, and A. M. Dollar, "Object-agnostic dexterous manipulation of partially constrained trajectories," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5494–5501, 2020.

[9] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[10] 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)*. IEEE, 2017, pp. 1714–1721.

[11] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, 2018, pp. 4754–4765.

[12] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," in *Advances in Neural Information Processing Systems*, 2018, pp. 8289–8300.

[13] M. Pereira, D. D. Fan, G. N. An, and E. Theodorou, "Mpc-inspired neural network policies for sequential decision making," *arXiv preprint arXiv:1802.05803*, 2018.

[14] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Advances in Neural Information Processing Systems*, 2019, pp. 12 519–12 530.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[22] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6244–6251.

[23] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[25] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[26] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," *arXiv preprint arXiv:1703.03078*, 2017.

[27] V. Pong, S. Gu, M. Dalal, and S. Levine, "Temporal difference models: Model-free deep rl for model-based control," *arXiv preprint arXiv:1802.09081*, 2018.

[28] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[29] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," *arXiv preprint arXiv:1809.05214*, 2018.

[30] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[31] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[32] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*, 2020, pp. 1101–1112.

[33] I. Clavera, V. Fu, and P. Abbeel, "Model-augmented actor-critic: Backpropagating through paths," *arXiv preprint arXiv:2005.08068*, 2020.

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

[35] F. Farshidian, D. Hoeller, and M. Hutter, "Deep value model predictive control," *arXiv preprint arXiv:1910.03358*, 2019.

[36] M. Bhardwaj, A. Handa, D. Fox, and B. Boots, "Information theoretic model predictive q-learning," in *Learning for Dynamics and Control*, 2020, pp. 840–850.

[37] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440.

[38] 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.

[39] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[40] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[41] R. R. Ma and A. M. Dollar, "An underactuated hand for efficient finger-gaiting-based dexterous manipulation," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014, pp. 2214–2219.

[42] "Yale openhand project." [Online]. Available: https://www.eng.yale.edu/grablab/openhand/

[43] A. S. Morgan, W. G. Bircher, and A. M. Dollar, "Towards generalized manipulation learning through grasp mechanics-based features and self-supervision," *IEEE Transactions on Robotics*, pp. 1–17, 2021.

[44] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[45] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4193–4198.