Finetuning Offline World Models in the Real World

Anonymous Author(s)
Affiliation
Address
email

Abstract: Reinforcement Learning (RL) is notoriously data-inefficient, which makes training on a real robot difficult. While model-based RL algorithms (*world models*) improve data-efficiency to some extent, they still require hours or days of interaction to learn skills. Recently, offline RL has been proposed as a framework for training RL policies on *pre-existing* datasets without any online interaction. However, constraining an algorithm to a fixed dataset induces a state-action distribution shift between training and inference, and limits its applicability to new tasks. In this work, we seek to get the best of both worlds: we consider the problem of pretraining a world model with offline data collected on a real robot, and then finetuning the model on online data collected by planning with the learned model. To mitigate extrapolation errors during online interaction, we propose to regularize the planner *at test-time* by balancing estimated returns and (epistemic) model uncertainty. We evaluate our method on a variety of visuo-motor control tasks in simulation and on a real robot, and find that our method enables few-shot finetuning to seen and unseen tasks even when offline data is limited. Videos are available at https://owmcorl.github.io.

Keywords: Model-Based Reinforcement Learning, Real-World Robotics

1 Introduction

Reinforcement Learning (RL) has the potential to train physical robots to perform complex tasks autonomously by interacting with the environment and receiving supervisory feedback in the form of rewards. However, RL algorithms are notoriously data-inefficient and require large amounts (often millions or even billions) of online environment interactions to learn skills due to limited supervision [1, 2, 3]. This makes training on a *real* robot difficult. To circumvent the issue, prior work commonly rely on custom-built simulators [4, 5, 1] or human teleoperation [6, 7] for behavior learning, both of which are difficult to scale due to the enormous cost and engineering involved. Additionally, these solutions each introduce additional technical challenges such as the simulation-to-real gap [8, 1, 9, 10] and the inability to improve over human operators [11, 12], respectively.

Recently, offline RL has been proposed as a framework for training RL policies from *pre-existing* interaction datasets without the need for online data collection [13, 14, 15, 16, 17, 18, 19]. Leveraging existing datasets alleviates the problem of data-inefficiency without suffering from the aforementioned limitations. However, any pre-existing dataset will invariably not cover the entire state-action space, which leads to (potentially severe) extrapolation errors, and consequently forces algorithms to learn overly conservative policies [20, 21, 22, 23]. We argue that extrapolation errors are less of an issue in an online RL setting, since the ability to collect new data provides an intrinsic self-calibration mechanism: by executing overestimated actions and receiving (comparably) negative feedback, value estimations can be adjusted accordingly.

In this work, we seek to get the best of both worlds. We consider the problem of pretraining an RL policy on pre-existing interaction data, and subsequently finetuning said policy on a limited amount of data collected by online interaction. Because we consider only a very limited amount of online

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

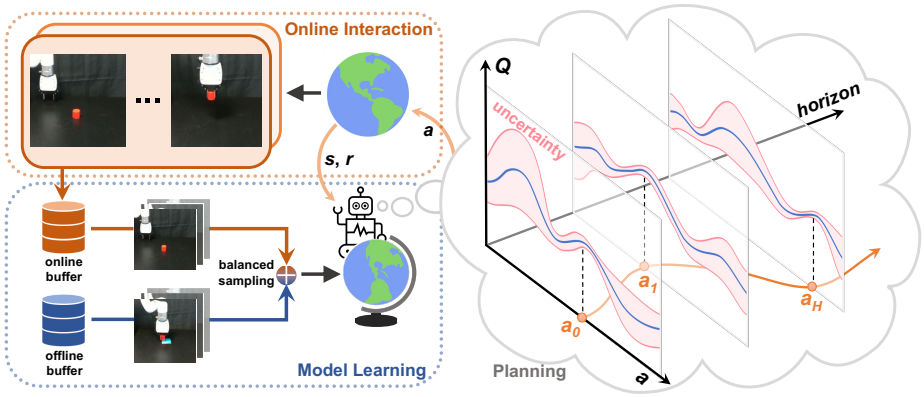

Figure 1. **Approach.** We propose a framework for offline pretraining and online finetuning of world models directly in the real world, without reliance on simulators or synthetic data. Our method iteratively collects new data by **planning with the learned model**, and **finetunes the model** on a combination of pre-existing data and newly collected data. Our method can be finetuned few-shot on unseen task variations in $\leq$**20 trials** by leveraging novel *test-time* regularization during planning.

interactions ($\leq$**20 trials**), we shift our attention to model-based RL (MBRL) [24] and the TD-MPC [25] algorithm in particular, due to its data-efficient learning. However, as our experiments will reveal, MBRL alone does not suffice in an offline-to-online finetuning setting when data is scarce. In particular, we find that planning suffers from extrapolation errors when queried on unseen state-action pairs, and consequently fails to converge. While this issue is reminiscent of overestimation errors in model-free algorithms, MBRL algorithms that leverage planning have an intriguing property: their (planning) policy is non-parametric and can optimize arbitrary objectives *without* any gradient updates. Motivated by this key insight, we propose a framework for offline-to-online finetuning of MBRL agents (*world models*) that mitigates extrapolation errors in planning via novel *test-time* behavior regularization based on (epistemic) model uncertainty. Notably, this regularizer can be applied to both purely offline world models *and* during finetuning.

We evaluate our method on a variety of continuous control tasks from the D4RL [26] and xArm [27] task suites in simulation, as well as visuo-motor control tasks on a real xArm robot. We find that our method outperforms state-of-the-art methods for offline and online RL across all tasks, and enables few-shot finetuning to *unseen* tasks and task variations even when offline data is limited. For example, our method improves the success rate of an offline world model **from 22% to 67% in just *20 trials*** for a real-world visual *pick* task with *unseen* distractors. We are, to the best of our knowledge, the first work to investigate offline-to-online finetuning with MBRL on real robots, and hope that our encouraging few-shot results will inspire further research in this direction.

2 Preliminaries: Reinforcement Learning and the TD-MPC Algorithm

We start by introducing our problem setting and MBRL algorithm of choice, TD-MPC, which together form the basis for the technical discussion of our approach in Section 3.

Reinforcement Learning We consider the problem of learning a visuo-motor control policy by interaction, formalized by the standard RL framework for infinite-horizon Partially Observable Markov Decision Processes (POMDPs) [28]. Concretely, we aim to learn a policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_+$ that outputs a conditional probability distribution over actions $\mathbf{a} \in \mathcal{A}$ conditioned on a state $\mathbf{s} \in \mathcal{S}$ that maximizes the expected return (cumulative reward) $\mathcal{R} = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where t is a discrete time step, r_t is the reward received by executing action $\mathbf{a}_t$ in state $\mathbf{s}_t$ at time t, and $\gamma \in [0, 1)$ is a discount factor. We leverage an MBRL algorithm in practice, which decomposes π_θ into multiple learnable components (a *world model*), and uses the learned model for planning. For brevity, we use subscript θ to symbolize learnable parameters throughout this work. In a POMDP, environment interactions obey an (unknown) transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, where states $\mathbf{s}$ themselves are assumed unobservable. However, we can define approximate environment states $\mathbf{s} \doteq (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_n)$ from sensory observations $\mathbf{o}_{1:n}$ obtained from *e.g.* cameras or robot proprioceptive information.

TD-MPC Our work extends TD-MPC [25], an MBRL algorithm that plans using Model Predictive Control (MPC) with a world model and terminal value function that is jointly learned via Temporal Difference (TD) learning. TD-MPC has two intriguing properties that make it particularly relevant to our setting: *(i)* it uses planning, which allows us to regularize action selection at test-time, and *(ii)* it is lightweight relative to other MBRL algorithms, which allows us to run it real-time. We summarize the architecture in Figure 2. Concretely, TD-MPC learns five components: *(1)* a representation $\mathbf{z} = h_\theta(\mathbf{s})$ that maps high-dimensional inputs $\mathbf{s}$ to a compact latent representation $\mathbf{z}$, *(2)* a latent dynamics model $\mathbf{z}' = d_\theta(\mathbf{z}, \mathbf{a})$ that obeys the transition function $\mathcal{T}$, and three prediction heads: *(3)* a reward predictor $\hat{r} = R_\theta(\mathbf{z}, \mathbf{a})$ that predicts the instantaneous reward, *(4)* a terminal value function $\hat{q} = Q_\theta(\mathbf{z}, \mathbf{a})$, and *(5)* a latent policy guide $\hat{\mathbf{a}} = \pi_\theta(\mathbf{z})$ that is used as a behavioral prior for planning. We use $\mathbf{z}', \mathbf{s}'$ to denote the successor (latent) states of $\mathbf{z}, \mathbf{s}$ in a subsequence, and use $\hat{\mathbf{a}}, \hat{r}, \hat{q}$ to differentiate predictions from observed (ground-truth) quantities $\mathbf{a}, r, q$. In its original formulation, TD-MPC is an online off-policy RL algorithm that maintains a replay buffer $\mathcal{B}$ of interactions, and jointly optimizes all components by minimizing the objective

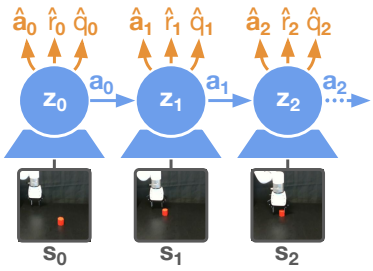

Figure 2. **Architecture.** Our world model encodes an observation $\mathbf{s}_0$ into its latent **representation** $\mathbf{z}_0$, and then recurrently predicts future latent states $\mathbf{z}_{1:h}$ as well as optimal **actions** $\hat{\mathbf{a}}_{0:h}$, **rewards** $\hat{r}_{0:h}$, and **values** $\hat{q}_{0:h}$. Future states $\mathbf{s}_{1:h}$ provide supervision for learning but are not required for planning.

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')_{0:h}\sim\mathcal{B}} \left[\sum_{t=0}^{h} \left(\underbrace{\|\mathbf{z}'_t - \mathrm{sg}(h_\phi(\mathbf{s}'_t))\|_2^2}_{\text{Latent dynamics}} + \underbrace{\|\hat{r}_t - r_t\|_2^2}_{\text{Reward}} + \underbrace{\|\hat{q}_t - q_t\|_2^2}_{\text{Value}} - \underbrace{Q_\theta(\mathbf{z}_t, \hat{\mathbf{a}}_t)}_{\text{Action}} \right) \right] \tag{1}$$

where $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')_{0:h}$ is a subsequence of length h sampled from the replay buffer, ϕ is an exponentially moving average of θ, sg is the `stop-grad` operator, $q_t = r_t + Q_\phi(\mathbf{z}'_t, \pi_\theta(\mathbf{z}'_t))$ is the TD-target, and gradients of the last term (action) are taken only wrt. the policy parameters. Constant coefficients balancing the losses are omitted. We refer to Hansen et al. [25] for additional implementation details, and instead focus our discussion on details pertaining to our contributions. During inference, TD-MPC plans actions using a sampling-based planner (MPPI) [29] that iteratively fits a time-dependent multivariate Gaussian with diagonal covariance over the space of action sequences such that return – as evaluated by simulating actions with the learned model – is maximized. For a (latent) state $\mathbf{z}_0 = h_\theta(\mathbf{s}_0)$ and sampled action sequence $\mathbf{a}_{0:h}$, the estimated return $\hat{\mathcal{R}}$ is given by

$$\hat{\mathcal{R}} = \gamma^h \underbrace{Q_\theta(\mathbf{z}_h, \mathbf{a}_h)}_{\text{Value}} + \sum_{t=0}^{h-1} \gamma^t \underbrace{R_\theta(\mathbf{z}_t, \mathbf{a}_t)}_{\text{Reward}}, \quad \mathbf{z}_{t+1} = \underbrace{d_\theta(\mathbf{z}_t, \mathbf{a}_t)}_{\text{Latent dynamics}}, \quad \mathbf{z}_0 = \underbrace{h_\theta(\mathbf{s}_0)}_{\text{Encoder}}. \tag{2}$$

To improve the rate of convergence in planning, a fraction of sampled action sequences are generated by the learned policy π_θ, effectively inducing a behavioral prior over possible action sequences. While π_θ is implemented as a deterministic policy, Gaussian action noise can be injected for stochasticity. TD-MPC has demonstrated excellent data-efficiency in an online RL setting, but suffers from extrapolation errors when naïvely applied to our problem setting, which we discuss in Section 3.

3 Approach: A *Test-Time* Regularized World Model and Planner

We propose a framework for offline-to-online finetuning of world models that mitigates extrapolation errors in the model via novel *test-time* regularization during planning. Our framework is summarized in Figure 1, and consists of two stages: *(1)* an *offline* stage where a world model is pretrained on pre-existing offline data, and *(2)* an *online* stage where the learned model is subsequently finetuned on a limited amount of online interaction data. While we use TD-MPC [25] as our backbone world model and planner, our approach is broadly applicable to any MBRL algorithm that uses planning. We start by outlining the key source of model extrapolation errors when used for offline

RL, then introduce our test-time regularizer, and conclude the section with additional techniques that we empirically find helpful for the few-shot finetuning of world models.

3.1 Extrapolation Errors in World Models Trained by Offline RL

All methods suffer from extrapolation errors when trained on offline data and evaluated on unseen data due to a state-action distribution shift between the two datasets. In this context, value overestimation in model-free Q-learning methods is the most well-understood type of error [14, 16, 17, 19]. However, MBRL algorithms like TD-MPC face unique challenges in an offline setting: state-action distribution shifts are present not only in value estimation, but also in (latent) dynamics and reward prediction when estimating the return of sampled trajectories as in Equation 2. We will first address value overestimation, and then jointly address other types of extrapolation errors in Section 3.2.

Inspired by Implicit Q-learning (IQL; [19]), we choose to mitigate the overestimation issue by applying TD-backups only on *in-sample* actions. Specifically, consider the value term in Equation 1 that computes a TD-target q by querying Q_ϕ on a latent state $\mathbf{z}'_t$ and potentially out-of-sample action $\pi_\theta(\mathbf{z}'_t)$. To avoid out-of-sample actions in the TD-target, we introduce a state-conditional value estimator V_θ and reformulate the TD-target as $q_t = r_t + \gamma V_\theta(\mathbf{z}'_t)$. This estimator can be optimized by an asymmetric ℓ_2-loss (expectile regression):

$$\mathcal{L}_V(\theta) = |\tau - \mathbb{1}_{\{Q_\phi(\mathbf{z}_t, \mathbf{a}_t) - V_\theta(\mathbf{z}_t) < 0\}}|(Q_\phi(\mathbf{z}_t, \mathbf{a}_t) - V_\theta(\mathbf{z}_t))^2, \tag{3}$$

where $\tau \in (0, 1)$ is a constant *expectile*. Intuitively, we approximate the maximization in $V_\theta(\mathbf{z}_t) = \max_{\mathbf{a}_t} Q_\phi(\mathbf{z}_t, \mathbf{a}_t)$ for $\tau \to 1$, and are increasingly conservative for smaller τ. Note that $\mathbf{a}_t$ is the action from the dataset (replay buffer), and thus no out-of-sample actions are needed. For the same purpose of avoiding out-of-sample actions, we replace the action term for policy learning in Equation 1 by an advantage weighted regression (AWR) [30, 31, 32] loss $\exp(\beta(Q_\phi(\mathbf{z}_t, \mathbf{a}_t) - V_\theta(\mathbf{z}_t))) \log \pi_\theta(\mathbf{a}_t|\mathbf{z}_t)$, where $\beta \geq 0$ is a temperature parameter.

3.2 Uncertainty Estimation as Test-Time Behavior Regularization

While only applying TD-backups on in-sample actions is effective at mitigating value overestimation during offline *training*, the world model (including dynamics, reward predictor, and value function) may still be queried on unseen state-action pairs during *planning*, *i.e.*, when estimating returns using Equation 2. This can result in severe extrapolation errors despite a cautiously learned value function. To address this additional source of errors, we propose a *test-time* behavior regularization technique that balances estimated returns and (epistemic) model uncertainty when evaluating Equation 2 for sampled action sequences. By regularizing estimated returns, we retain the expressiveness of planning with a world model despite imperfect state-action coverage. Avoiding actions for which the outcome is highly uncertain is strongly desired in an offline RL setting where no additional data can be collected, but conservative policies likewise limit exploration in an online RL setting. Unlike prior offline RL methods that predominantly learn an explicitly and consistently conservative value function and/or policy, regularizing planning based on model uncertainty has an intriguing property: as planning continues to cautiously explore and the model is finetuned on new data, the epistemic uncertainty naturally decreases. This makes test-time regularization based on model uncertainty highly suitable for both few-shot finetuning and continued finetuning over many trajectories.

To obtain a good proxy for epistemic model uncertainty with *minimal* computational overhead and architectural changes, we propose to utilize a small ensemble of value functions $Q_\theta^{(1)}, Q_\theta^{(2)}, \ldots, Q_\theta^{(N)}$ similar to Chen et al. [33]. We optimize the Q-functions with TD-targets discussed in Section 3.1, and use a random subset of (target) Q-networks to estimate the Q-values in Equation 3. Although ensembling all components of the world model may yield a better estimate of epistemic uncertainty, our choice of ensembling is lightweight enough to run on a real robot and is empirically a sufficiently good proxy for uncertainty. We argue that with a Q-ensemble, we can not only benefit from better Q-value estimation, but also leverage the standard deviation of the Q-values to measure the uncertainty of a state-action pair, i.e., whether it is out-of-distribution or not. This serves as a test-time regularizer that equips the planner with the ability to balance exploitation

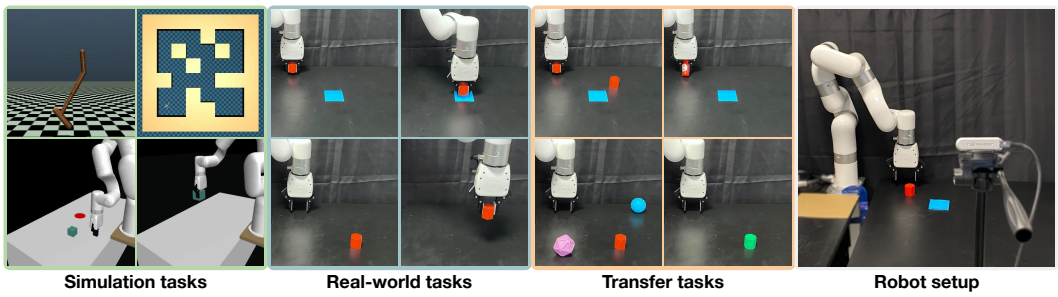

Figure 3. **Tasks.** We consider diverse tasks in simulation (D4RL [26], xArm [39]) and on a real robot. Our real-world tasks use raw pixels as input. Our method achieves high success rates in offline-to-online transfer to both seen and unseen tasks in just 20 online trials on a real robot.

Table 1. **Real-world offline-to-online results.** Success rate (%) as a function of online finetuning trials. Mean of 20 trials and 2 seeds.

		online	offline-to-online	
	Trials	TD-MPC	TD-MPC	**Ours**
Reach	0	0±0	50±18	**72±6**
	10	0±0	67±12	**94±6**
	20	0±0	**78±12**	**89±0**
Pick	0	0±0	0±0	0±0
	10	0±0	28±6	**33±0**
	20	0±0	28±6	**50±6**

Table 2. **Finetuning to unseen real-world tasks.** Success rate (%) of our method for each task variation shown in Figure 3. We include 3 successful transfers and 1 failure. See Appendix B for task descriptions. Mean of 20 trials and 2 seeds.

		online trials		
	Variation	0	10	20
Reach	distractor	22±0	22±11	**62±6**
	object shape	44±11	44±0	**78±11**
Pick	distractors	22±11	56±0	**67±11**
	object color	0±0	0±0	0±0

and exploration without explicitly introducing conservatism in training. By penalizing the actions leading to high uncertainty, we prioritize the actions that are more likely to achieve a reliably high return. Formally, we modify the estimated return in Equation 2 of an action sequence $\mathbf{a}_{0:h}$ to be

$$\hat{\mathcal{R}} = \gamma^h \left(Q_\theta(\mathbf{z}_h, \mathbf{a}_h) - \lambda u_h\right) + \sum_{t=0}^{h-1} \gamma^t \left(R_\theta(\mathbf{z}_t, \mathbf{a}_t) - \lambda u_t\right), \quad u_t = \mathrm{std}\left(\{Q_\theta^{(i)}(\mathbf{z}_t, \mathbf{a}_t)\}_{i=1}^N\right), \tag{4}$$

where the uncertainty regularizer u_t is highlighted in **red**. Here, std denotes the standard deviation operator, and λ is a constant coefficient that controls the regularization strength. We use the same value of λ for both the offline and online stages in practice, but it need not be equal.

To facilitate rapid propagation of information acquired during finetuning, we maintain *two* replay buffers, $\mathcal{B}_{\text{off}}$ and $\mathcal{B}_{\text{on}}$ for offline and online data, respectively, and optimize the objective in Equation 1 on mini-batches of data sampled in equal parts from $\mathcal{B}_{\text{off}}$ and $\mathcal{B}_{\text{on}}$, *i.e.*, online interaction data is heavily oversampled early in finetuning. Balanced sampling has been explored in various settings [34, 35, 36, 21, 37, 38], and we find that it consistently improves finetuning of world models as well.

4 Experiments & Discussion

We evaluate our method on diverse continuous control tasks from the D4RL [26] and xArm [39] task suites in simulation, as well as two visuo-motor control tasks on a real xArm robot. Our tasks are visualized in Figure 3. We design a series of experiments that aim to answer the following questions:

— **Q1:** *How does our approach compare to state-of-the-art methods for offline RL and online RL?*
— **Q2:** *Can our approach be used to finetune world models on unseen tasks and task variations?*
— **Q3:** *How do individual components of our approach contribute to its success?*

In the following, we first detail our experimental setup, and then proceed to address each of the above questions based on our experimental results. **Our code will be released to the public.**

Real robot setup Our setup is shown in Figure 3 *(right)*. The agent controls an xArm 7 robot with a jaw gripper using positional control, and 224×224 RGB image observations are captured

Table 3. **Offline-to-online results in simulation.** Success rate (xArm) and normalized return (D4RL) of methods **before** and **after** online finetuning. See Appendix B for task explanations. Mean of 5 seeds.

	online	offline-to-online		
Task	TD-MPC	TD-MPC	IQL	**Ours**
Push	**81.0**	**59.0** → 80.0	20.0 → 17.0	45.0 → 64.0
Pick	84.0	0.0 → **96.0**	0.0 → 0.0	0.0 → 88.0
Hopper (m)	2.8	0.8 → 11.0	**66.3** → 76.1	49.6 → **100.7**
Hopper (mr)	6.1	13.4 → 13.7	76.5 → **101.4**	**84.4** → 93.5
AntMaze (mp)	52.0	0.0 → 68.0	54.0 → 80.0	**58.0** → **96.0**
AntMaze (md)	72.0	0.0 → **91.0**	62.0 → 88.0	**75.0** → 89.0
Average	49.7	12.2 → 60.0	46.5 → 60.4	**52.0** → **88.5**

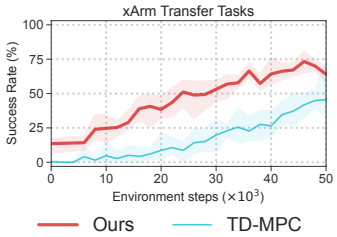

Figure 4. **Finetuning to unseen tasks.** Success rate (%) aggregated across 9 transfer tasks in simulation. Mean of 5 seeds.

by a static third-person Intel RealSense camera; the agent also has access to robot proprioceptive information. Our setup requires no further instrumentation. We consider two tasks, *reach* and *pick*, and several task variations derived from them. Our tasks are visualized in Figure 3. The goal in *reach* is to reach a target with the end-effector, and the goal in *pick* is to pick up and lift a target object above a height threshold. We use a manually designed success detector to determine task success and automatically provide sparse rewards (albeit noisy) for both offline and online RL. We use 120 offline trajectories for *reach* and 200 for *pick*; see Section 4.1 *(Q3.2)* for details and ablations.

Simulation tasks and datasets We consider a diverse set of tasks and datasets, including four tasks from the D4RL [26] benchmark (*Hopper (medium)*, *Hopper (medium-replay)*, *AntMaze (medium-play)* and *AntMaze (medium-diverse)*), and two visuo-motor control tasks from the xArm [27] benchmark (*push* and *pick*); the two xArm tasks are similar to our real-world tasks except using lower image resolution (84×84) and dense rewards. See Figure 3 *(left)* for task visualizations. For xArm tasks we use the first 40k transitions (800 trajectories) from the replay buffer of a TD-MPC agent trained from scratch as our offline datasets. See Appendix B for more details.

Baselines We compare our approach against strong online RL and offline RL methods: *(i)* **TD-MPC** [25] trained from scratch with online interaction only, *(ii)* **TD-MPC (+offline)** which naïvely pretrains on offline data and is then finetuned online, but without any of our additional contributions, and *(iii)* **IQL** [19], a *state-of-the-art* offline RL algorithm which has strong offline performance and also allows for policy improvement with online finetuning. Additionally, we also conduct an exhaustive ablation study to demonstrate the relative importance of each of our contributions. See Appendix C and D for extensive implementation details on our method and baselines, respectively.

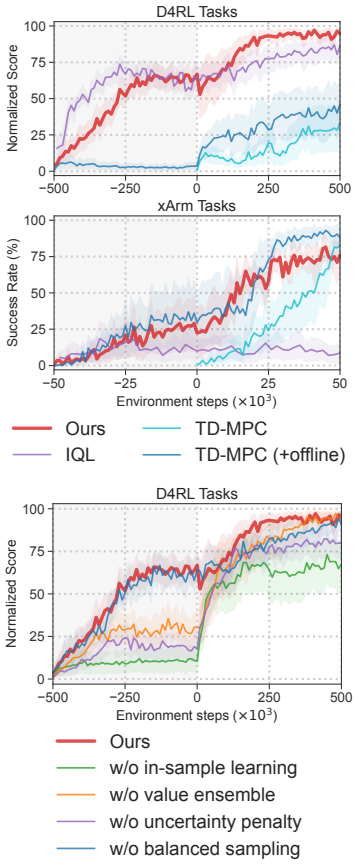

Figure 5. **Aggregate results.** *Top:* comparison to baselines. *Bottom:* ablations. Offline pretraining is shaded gray. 5 seeds.

4.1 Results

Q1: Offline-to-online RL We benchmark methods across all tasks considered; real robot results are shown in Table 1, and simulation results are shown in Table 3. We also provide aggregate curves in Figure 5 *(top)* and per-task curves in Appendix A. Our approach consistently achieves strong zero-shot and online finetuning performance across all tasks, outperforming offline-to-online TD-MPC and IQL **by a large margin in both simulation and real** in terms of asymptotic performance. Notably, performance of our method is more robust to variations in dataset and task than baselines, as evidenced by the aggregate results.

Q2: Finetuning to unseen tasks A key motivation for developing learning-based (and model-based in particular) methods for robotics is the potential for generalization and fast adaptation to unseen scenarios. To assess the versatility of our approach, we conduct additional offline-to-online finetuning experiments where the offline and online tasks are distinct, *e.g.*, transferring a *reach* policy to a *push* task, introducing distractors, or changing the target object. We design 4 real-world transfer tasks, and 9 in simulation. See Figure 3 *(center)* and Appendix B for task visualizations and descriptions. Our real robot results are shown in Table 2, and aggregated simulation results are shown in Figure 4. We also provide per-task results in Appendix A. Our method successfully adapts to unseen tasks and task variations – both in simulation and in real – and significantly outperforms TD-MPC trained from scratch on the target task. However, as evidenced in the *object color* experiment of Table 2, transfer might not succeed if the initial model does not achieve *any* reward.

Q3: Ablations To understand how individual components contribute to the success of our method, we conduct a series of ablation studies that exhaustively ablate each design choice. We highlight our key findings based on aggregate task results but note that per-task curves are available in Appendix A for *all* experiments and tasks.

Q3.1: Algorithmic components We ablate each component of our proposed method in Figure 5 *(bottom)*, averaged across all D4RL tasks. Specifically, we ablate *(1)* learning Q_θ with in-sample actions and expectile regression as described in Section 3.1, *(2)* using an ensemble of 5 value functions instead of 2 as in the original TD-MPC, *(3)* regularizing planning with our uncertainty penalty described in Section 3.2, and *(4)* using balanced sampling, *i.e.*, sampling offline and online data in equal parts within each mini-batch. The results highlight the effectiveness of our key contributions. Balanced sampling improves the data-efficiency of online finetuning, and all other components contribute to both offline and online performance.

Q3.2: Offline dataset Next, we investigate how the quantity and source of the offline dataset affect the success of online finetuning. We choose to conduct this ablation with real-world data to ensure that our conclusions generalize to realistic robot learning scenarios. We experiment with two data sources and two dataset sizes: *Base* that consists of 50/100 trajectories (depending on the dataset size) generated by a BC policy with added noise; and *Diverse* that consists of the same 50/100 trajectories as Base, but with an additional 20 exploratory trajectories from a suboptimal RL agent. In fact, the additional trajectories correspond to a 20-trial replay buffer from an experiment conducted in the early stages of the research project. Results for this experiment are shown in Table 4. We find that results improve with more data regardless of source, but that exploratory data holds far greater value: neither TD-MPC nor our method succeeds zero-shot when trained on the Base dataset – regardless of data quantity – whereas our method obtains 33% and 72% success rate with 70 and 120 trajectories, respectively, from the Diverse dataset. This result demonstrates that replay buffers from previously trained agents can be valuable data sources for future experiments.

Table 4. **Real-world ablation on offline data.** Success rate (%) as a function of online finetuning trials for two data sources and sizes from our real-world reach task: *Base* that consists of 50/100 trajectories generated by a noisy BC policy; and *Diverse* that consists of the **same** 50/100 trajectories as Base, but with an additional 20 exploratory trajectories. Exploratory data greatly improves zero-shot results. Mean of 20 trials and 2 seeds.

	TD-MPC		**Ours**			
	Base	Diverse	Base		Diverse	
Trials	100	120	50	100	70	120
0	0±0	50±18	0±0	0±0	33±0	**72±6**
10	44±12	67±12	50±0	61±6	61±6	**94±6**
20	**83±6**	78±12	61±6	**89±0**	**89±0**	**89±0**

Q3.3: Uncertainty regularization We seek to understand how the uncertainty regularization strength λ influences the test-time performance of our method. Results for two tasks are shown in Figure 6; see Appendix A for more results. While $\lambda > 0$ almost always outperforms $\lambda = 0$ in both offline and online RL, large values, *e.g.*, $\lambda = 20$, can be detrimental to online RL in some cases. On the other hand, increasing λ from 1 to 20 improves offline results (score) from 4.0 to 49.6

on *Hopper (medium)*. We use the same λ for offline and online stages in this work, but remark that our uncertainty regularizer is a *test-time* regularizer, which can be tuned at any time at no cost.

5 Related Work

Offline RL algorithms seek to learn RL policies solely from pre-existing datasets, which results in a state-action distribution shift between training and evaluation. This shift can be mitigated by applying explicit regularization techniques to conventional online RL algorithms, commonly the model-free Soft Actor-Critic (SAC; [40]) algorithm. Most prior works constrain the learned policy to be close to the policy that generated the data [13, 14, 15, 16], or regularize a learned value function to be conservative (*i.e.*, underestimating) [17, 18, 19]. While these strategies can be highly effective for offline RL, they have been shown to slow down convergence when finetuned with online interaction [20, 21, 23]. Lastly, Agarwal et al. [41] and Yarats et al. [22] both show that standard online RL algorithms are sufficient for offline RL when data is abundant and diverse.

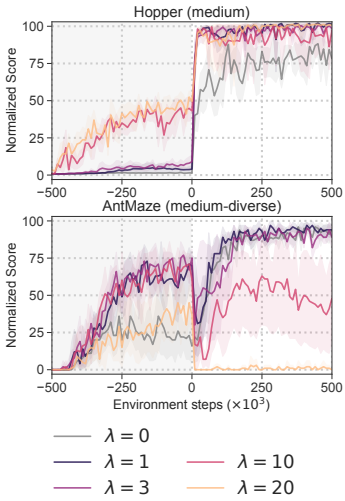

Figure 6. **Regularization (λ).** Score as a function of regularization strength for two tasks from D4RL. Gray shade indicates offline RL. Mean of 5 seeds.

Finetuning policies with RL Multiple works have considered finetuning policies obtained by, *e.g.*, imitation learning [42, 12, 37], self-supervised learning [43], or offline RL [20, 44, 21, 45, 23]. Notably, Rajeswaran et al. [42] learns a model-free policy and constrains it to be close to a set of demonstrations, Baker et al. [12] learns a policy for Minecraft by imitating a large dataset of human play and subsequently finetuning with an RL policy, and Lee et al. [21] shows that finetuning an ensemble of model-free actor-critic agents trained with conservative Q-learning on an offline dataset can improve sample-efficiency in simulated control tasks. By instead learning a *world model* on offline data, our method regularizes actions at *test-time* (via planning) based on model uncertainty, *without* explicit loss terms, which proves particularly beneficial for *few-shot* finetuning. While we do not compare to [21] in our experiments, we compare to IQL [19], a concurrent offline RL method that is conceptually closer to our method.

Real-world RL Existing work on real robot learning typically trains policies on large amounts of data in simulation, and transfers learned policies to real robots without additional training (simulation-to-real). This introduces a domain gap, for which an array of mitigation strategies have been proposed, including domain randomization [8, 4, 1], data augmentation [39], and system identification [46]. Additionally, building accurate simulation environments can be a daunting task. At present, only a limited number of studies have considered training RL policies in the real world without any reliance on simulators [47, 20, 48]. For example, researchers have proposed to accelerate online RL with human demonstrations [47] or offline datasets [20, 35] for model-free algorithms. Most recently, Wu et al. [48] demonstrates that MBRL can be data-efficient enough to learn diverse real robot tasks from scratch. Our work is conceptually similar to [20, 48] in that we leverage offline data and an MBRL algorithm on real hardware, respectively. However, our problem setting is not directly comparable, as we consider an order of magnitude less online data than prior work.

6 Limitations

We are excited about our strong real robot results. However, we also acknowledge that several open problems remain: offline data quality and quantity heavily impact few-shot learning, and we limit ourselves to sparse rewards since dense rewards are difficult to obtain in the real world. As discussed in Section 4.1 (Q3.3) we also find that the optimal value of λ can differ between tasks and between offline and online RL. We leave it as future work to automate this hyperparameter search, but note that doing so is relatively cheap since it can be adjusted at test-time without any overhead.

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39:20 – 3, 2018.

[2] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. W. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. *ArXiv*, abs/1912.06680, 2019.

[3] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. P. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588:604 – 609, 2019.

[4] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2017.

[5] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *ArXiv*, abs/1710.06542, 2017.

[6] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

[7] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. A. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *ArXiv*, abs/2212.06817, 2022.

[8] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[9] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.

[10] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations (ICLR)*, 2021.

[11] A. Kumar, J. Hong, A. Singh, and S. Levine. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AP1MKT37rJ.

[12] B. Baker, I. Akkaya, P. Zhokhov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *ArXiv*, abs/2206.11795, 2022.

[13] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2018.

[14] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

[15] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel : Model-based offline reinforcement learning. *ArXiv*, abs/2005.05951, 2020.

[16] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *ArXiv*, abs/2106.06860, 2021.

[17] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Neural Information Processing Systems*, 2019.

[18] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *ArXiv*, abs/2006.04779, 2020.

[19] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *ArXiv*, abs/2110.06169, 2021.

[20] A. Nair, M. Dalal, A. Gupta, and S. Levine. Accelerating online reinforcement learning with offline datasets. *ArXiv*, abs/2006.09359, 2020.

[21] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[22] D. Yarats, D. Brandfonbrener, H. Liu, M. Laskin, P. Abbeel, A. Lazaric, and L. Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *ArXiv*, abs/2201.13425, 2022.

[23] M. Nakamoto, Y. Zhai, A. Singh, M. S. Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *ArXiv*, abs/2303.05479, 2023.

[24] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018.

[25] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

[26] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[27] Y. Ze, N. Hansen, Y. Chen, M. Jain, and X. Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, pages 1–1, 2023.

[28] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[29] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control using covariance variable importance sampling. *ArXiv*, abs/1509.01149, 2015.

[30] J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[31] 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.

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

[33] X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

[34] R. C. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman. Efficient adaptation for end-to-end vision-based robotic manipulation. *ArXiv*, abs/2004.10190, 2020.

[35] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv*, abs/2104.08212, 2021.

[36] N. Hansen, H. Su, and X. Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[37] N. Hansen, Y. Lin, H. Su, X. Wang, V. Kumar, and A. Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. 2023.

[38] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. D. Edwards, N. M. O. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent. *Trans. Mach. Learn. Res.*, 2022, 2022.

[39] R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, pages 1–1, 2022. doi:10.1109/LRA.2022.3144512.

[40] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018.

[41] R. Agarwal, D. Schuurmans, and M. Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *ArXiv*, abs/1907.04543, 2019.

[42] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[43] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[44] C. Wang, X. Luo, K. W. Ross, and D. Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *ArXiv*, abs/2202.10324, 2022.

[45] Y. Xu, N. Hansen, Z. Wang, Y.-C. Chan, H. Su, and Z. Tu. On the feasibility of cross-task transfer with model-based reinforcement learning. 2023.

[46] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *ArXiv*, abs/2107.04034, 2021.

[47] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin. A framework for efficient robotic manipulation. *ArXiv*, abs/2012.07975, 2020.

[48] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, 2022.

[49] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[50] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.