Q-LEARNING WITH WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Off-policy reinforcement learning (RL) has become increasingly sample-efficient, enabling applications such as RL fine-tuning of Vision-Language-Action models into reliable, high-performing policies. World models offer a further lever for sample efficiency, as they predict state changes rather than actions alone, but their success has largely been confined to supervised policy learning. Prior model-based RL methods often optimize the policy or value function directly on imagined rollouts, which is prone to compounding bias and struggles to scale to large, high-dimensional problems such as real-world robotics, a problem that worsens with task horizon and visual complexity. In this work, we instead ask whether we can leverage world models directly on top of standard Q-learning to improve performance, while remaining trained and grounded in the real, online setting. We propose QWM, a framework that leverages world models to perform test-time search over imagined trajectories on top of Q-learning to select high-value actions during both online rollouts and evaluation. Since the policy and value function are trained only on real transitions, QWM avoids compounding model bias while still gaining the sample-efficiency benefits of predictive search. On challenging manipulation benchmarks Robomimic and LIBERO, QWM significantly outperforms strong prior state-of-the-art methods on both sample efficiency and performance.

1 INTRODUCTION

Recent advances in off-policy reinforcement learning (RL) have substantially improved its sample efficiency, enabling applications such as RL fine-tuning of Vision-Language-Action (VLA) models into highly reliable, high-performing policies (Intelligence et al., 2025; Dong et al., 2026a). As RL is applied to increasingly complex scenarios, further gains in sample efficiency become increasingly valuable, if not essential. World models offer a natural lever for closing this gap, as they have the ability to predict changes in state rather than actions alone. So far, however, their success has been largely confined to supervised policy learning. This raises a natural question: is there a simple way to leverage world models to improve the performance of online RL, without resorting to traditional model-based RL approaches that are prone to compounding the world model's bias into the training process?

Prior work on model-based RL has explored learning a dynamics model and optimizing a policy or value function directly within it (Hansen et al., 2024; Hafner et al., 2024), but scaling this approach to large, high-dimensional problems such as real-world robotics remains difficult: policies trained primarily on rollouts inside an imperfect world model inherit and compound its biases, and this problem worsens as task horizon and visual complexity increase. Rather than using the world model as a substitute for real interaction, we ask whether it can instead sharpen an RL agent that remains trained and grounded in the real, online setting. In this work, we study Q-learning with a world model, where the model serves not as a replacement for environment interaction but as a way for improving performance.

Our key insight is that a learned world model can be used on top of standard Q-learning to enable test-time search over actions by imagining future trajectories, and that this improved action selection alone directly translates into higher performance for Q-learning methods. We propose Q-Learning with World Models (QWM), a framework that leverages world models for test-time search directly on top of standard Q-learning. Instead of using the state value function to conduct search as in typically done in model-based RL, the learned Q-function also enables more powerful search to evaluate candidate actions and select the highest-value one before it is executed. This test-time search over

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

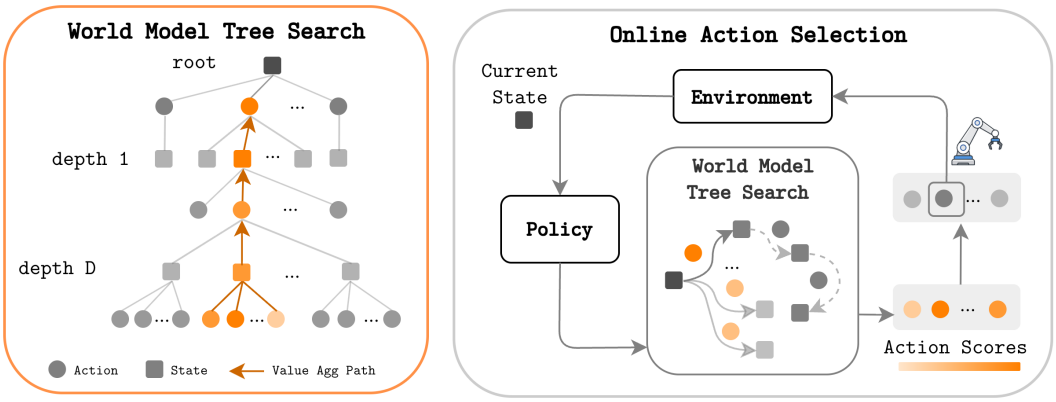


Figure 1: **Overview of QWM. Left: World-model tree search.** From the current state, the policy proposes candidate actions and the world model imagines future states. The value of each initial action is computed by aggregating value of intermediate and leaf nodes. **Right: Online action selection.** At each environment step, QWM uses tree-search scores to select among policy-proposed actions before executing the selected action in the environment. Tree search is used for both online sampling and evaluation, while the policy and critic are trained only on real environment transitions.

actions is applied both during online rollouts, to obtain better data for online RL, and at evaluation time, to act more effectively with the learned policy. Rather than sampling a single action from the policy or sampling multiple actions and selecting the one with the highest Q-value, QWM leverages the imagination capability of world models to select the best action for downstream performance, while avoiding the compounding bias and instability of policies trained inside a learned model.

Our main contribution is QWM, a simple and general framework for leveraging learned world models to improve Q-learning through test-time search. Because the underlying policy and value function are still trained online against real environment transitions, QWM avoids the bias accumulation that afflicts methods relying primarily on imagined rollouts, while still reaping the sample-efficiency benefits such a predictive model can offer at decision time. We evaluate QWM on challenging manipulation benchmarks, Robomimic and LIBERO, across both state-based and visual observation settings, and find that it significantly outperforms strong prior methods on both sample efficiency and performance.

2 Related Work

Reinforcement learning with prior data. To improve performance and sample efficiency of online RL, prior works have studied the problem of using an offline dataset to accelerate online learning. Popular approaches involve balancing exploration and exploitation (Yang et al., 2023; Zhang et al., 2023; Mark et al., 2023), calibrating value estimates (Nakamoto et al., 2024; Dong et al., 2026c), or retaining offline data alongside newly collected online data during fine-tuning (Vecerik et al., 2018; Nair et al., 2021; Ball et al., 2023; Dong et al., 2025a;b). Offline RL algorithms have also been applied directly to online fine-tuning (Fujimoto et al., 2019; Fujimoto & Gu, 2021; Hansen-Estruch et al., 2023; Park et al., 2025; Dong et al., 2025c; 2026b;e), where the prior data is first trained with offline RL before obtaining new samples online. Our work differs from these lines of research in that we focus on leveraging prior data through world models to improve online RL, rather than on how prior data is incorporated into the RL objective itself; this makes our approach applicable to any RL fine-tuning algorithm.

Model-based RL. Model-based RL methods learn a dynamics model of the environment and use it for policy or value learning. One family of methods generates additional synthetic transitions by rolling the learned model forward and treats them as if they were real experience (Sutton, 1991; Gu et al., 2016; Kalweit & Boedecker, 2017; Kurutach et al., 2018; Kaiser et al., 2024), or otherwise optimizes the policy directly against imagined trajectories (Zhang et al., 2019; Hafner et al., 2020; 2022; 2024; 2025) or through short branched rollouts mixed with real data (Feinberg et al., 2018; Janner et al., 2021). A second family instead uses the learned model only to plan over imagined rollouts with trajectory optimizers such as the cross-entropy method or MPPI (Nagabandi et al., 2018;

Ebert et al., 2018; Chua et al., 2018; Hansen et al., 2022; 2024); some of these methods further learn a terminal value function to bound the planning horizon (Hansen et al., 2022; 2024; Kim et al., 2026). A third family combines a learned (or known) model with Monte Carlo tree search guided by a learned value function, mainly for discrete-action, densely-simulated domains (Silver et al., 2017; Schrittwieser et al., 2020; Ye et al., 2021; Schrittwieser et al., 2021), with recent works extending it to continuous control (Hubert et al., 2021; Wang et al., 2024). These methods rely on some combination of having a known dynamics model, discrete actions, or bootstrapping policy and value targets from search statistics computed over model-imagined rollouts. Furthermore, many of these works rely on abundant simulated data, where model error and sample budget are far less of a concern than in real-world, sparse-reward robotic manipulation. Across all of these approaches, the policy or value function trained substantially on model-generated rollouts inherits and compounds the model's own biases directly into training, an issue that grows with task horizon and visual complexity (Janner et al., 2021; Chua et al., 2018; Kaiser et al., 2024; Hafner et al., 2020). QWM sidesteps this issue by leveraging the model purely at test-time rather than a source of additional training data on top of standard Q-learning, and train the policy and critic on real data.

World models for VLAs. A closely related line of work applies world models to VLA policies. VLAW (Guo et al., 2026) and World-VLA-Loop (Liu et al., 2026) alternate between fine-tuning world models on real rollouts and using it to generate synthetic training data for policy learning. WMPO (Zhu et al., 2025) and World4RL (Jiang et al., 2026a) run on-policy RL directly on imagined rollouts, while WoVR (Jiang et al., 2026b) targets the resulting reward hallucination and OOD drift via keyframe-initialized rollouts and policy-aligned co-evolution. RISE (Yang et al., 2026), World-Gymnast (Sharma et al., 2026), and GigaBrain-0.5M (Team et al., 2026) score imagined rollouts with a learned progress/value model or VLM and use the resulting rewards for policy-gradient updates. All of these use the world model primarily as a training data generator, which is prone to inheriting compounding bias. SAILOR (Jain et al., 2025) learns a world model and reward model to guide imitation and recover from out-of-support states. QWM can be applied to any RL fine-tuning methods for VLAs with a Q-function, and instead of using the world model to generate data or guide imitation learning, we leverage world models to improve Q-learning at test time.

Test-time scaling and best-of-N inference. A substantial body of work has demonstrated that test-time scaling, for example via best-of-N sampling, can significantly improve the performance of generative models across domains, including autoregressive language models (Wang et al., 2022; Snell et al., 2024; Chow et al., 2024), diffusion models (Ma et al., 2025), and policy learning (Dong et al., 2025b; Hansen-Estruch et al., 2023; Dong et al., 2026d; Chen et al., 2023). In most of these settings, candidates are scored independently by a verifier or value function, and only the single best sample is retained, without modeling how a candidate's near-term consequences unfold. QWM can instead be viewed as a form of test-time scaling in which search evaluates candidates on their predicted downstream value, yielding a substantially better estimate of which actions are best.

3 Preliminaries

We consider a Markov decision process (MDP) specified by the tuple $\{\mathcal{S}, \mathcal{A}, \rho, r, \gamma, T\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\rho(s)$ is the distribution over initial states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $T(s'|s, a)$ is the transition probabilities governing the environment. The goal of RL is to find a policy π that maximizes the expected discounted return $\mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$.

In this paper, we study how a learned world model can be used to improve online RL fine-tuning at decision time. Concretely, we assume access to a dataset $\mathcal{D}_{\text{offline}} = \{(s, a, r, s')\}$ of environment transitions – either collected offline or accumulated online – which is used to pretrain a world model $M_\psi(s'|s, a)$ that predicts the next state given the current state and action. The world model is used at decision time. During online training, the agent collects tuples (s_t, a_t, r_t, s_{t+1}) through environment interaction; these are appended to a replay buffer $\mathcal{D}$ and used to update the policy and Q-function toward higher returns. Our central question is how to leverage the world model to improve the performance of Q-learning.

We focus on off-policy RL, where the Q-function estimates the discounted return of the policy given a state and action, and is trained with TD learning:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[\left(r_t + \gamma Q_{\phi'}(s_{t+1}, \tilde{a}^*_{t+1}) - Q_\phi(s_t, a_t) \right)^2 \right], \tag{1}$$

where $Q_{\phi'}$ is a target network and $\tilde{a}^*_{t+1}$ is the next action selected by the RL policy. We implement QWM on top of two state-of-the-art off-policy Q-learning algorithms.

EXPO. The first is EXPO (Dong et al., 2025b), a high-performing online RL method that couples a base policy with a lightweight edit policy. EXPO maintains two parameterized policies: a base flow policy—in our case, the base policy π_{base}, trained with a supervised loss—and an edit policy π_{edit}, trained to maximize the Q-function:

$$\mathcal{L}(\pi_{\text{edit}}) = -\mathbb{E}_{(s_t, a_t) \sim \mathcal{D},\ \hat{a}_t \sim \pi_{\text{edit}}} \left[Q_\phi(s_t,\ a_t + \hat{a}_t) - \alpha \log \pi_{\text{edit}}(\hat{a}_t \mid s_t, a_t) \right]. \tag{2}$$

The edit policy predicts an edit $\hat{a}_t$, clipped to $[-\beta, \beta]$, that is added to the base action a_t to produce an edited action $\tilde{a}_t = a_t + \hat{a}_t$. This design avoids backpropagating gradients through the base policy while still grounding TD updates in near-optimal actions. The final inference and TD-backup policy is an on-the-fly (OTF) policy that selects the value-maximizing candidate among N sampled base and edited actions:

$$\tilde{a}^* = \underset{a \in \bigcup_{i=1}^{N} \{a_i, \tilde{a}_i\}}{\arg\max} \; Q_\phi(s, a), \tag{3}$$

where Q_ϕ is trained with a standard TD objective.

RLPD. The second is RLPD (Ball et al., 2023), a sample-efficient off-policy RL algorithm that combines offline and online replay data with high update-to-data (UTD) ratios, an ensemble of critics, and a Gaussian actor. RLPD trains the actor π_θ with a SAC-style (Haarnoja et al., 2018) entropy-regularized objective,

$$\mathcal{L}(\pi_\theta) = -\mathbb{E}_{s_t \sim \mathcal{D},\ a_t \sim \pi_\theta} \left[Q_\phi(s_t, a_t) - \alpha \log \pi_\theta(a_t \mid s_t) \right], \tag{4}$$

and stabilizes the high-UTD critic updates using an ensemble of M critics with random subsets used for the target computation, together with layer normalization. At each update, transitions are drawn via symmetric sampling, with equal proportions from the offline buffer and the online replay buffer.

4 Method

In this section, we present the key component of QWM to leverage world models to perform test-time search for better performance for Q-Learning. We instantiate QWM as a tree search over actions on top of standard Q-learning We first describe the approach for tree search with world models, split into three parts: constructing the search tree (Section 4.1), aggregating the value of expanded nodes into a value for each proposed action (Section 4.2), and searching with the Q-function (Section 4.3); we then discuss implementation details (Section 4.4).

4.1 Tree Search With the World Model

Our goal is to construct a tree such that searching over it yields good actions and adequate coverage of possible futures, and is general enough to be compatible with different problem settings. We construct the tree governed by three quantities: the number of actions sampled at each state N, the number of next states sampled per action K, and the search depth D.

Concretely, at each decision step, we set the root of the tree to the current state s_0 where depth 0 represents the root of the tree. Nodes alternate between state layers and action layers, with one state layer followed by one action layer constituting a single level of depth. At a state node at level d, the tree is expanded by drawing N candidate actions from the policy, $\{a_d^n\}_{n=1}^N \sim \pi_\theta(\cdot \mid s_d)$. At each action node, the tree is expanded and form depth $d+1$ by querying the world model K times to obtain K predicted future states, $\{s_{d+1}^k\}_{k=1}^K \sim M_\psi(s_d, a_d^n)$.

We repeat this expansion recursively to depth D (Figure 2). At the final layer, N_{leaf} actions are sampled from the policy. The tree is intended to give a short-horizon prediction of the consequences of an action, not to plan exhaustively – this keeps rollout lengths short enough to prevent the world model's own prediction error to accumulate.

4.2 AGGREGATING THE VALUE

Rather than scoring only the leaves and propagating value to the initial action, we observe that the value of every state-action pair in the tree is useful for action selection. The value of a node can either be expressed as a function of state-action value Q_ϕ or a function of state value V_ϕ, each with complementary trade-offs. As such, we express the value of a node as a combination of these two estimators of the same underlying quantity.

State-action value estimator. At depth d, we can score each of the N sampled actions $a_d^n \sim \pi_\theta(\cdot \mid s_d)$ directly with the critic, which already estimates the return of the full remaining trajectory:

$$V_Q\big(d \mid s_d\big) \;=\; \operatorname*{agg}_{n\in[N]} \; Q_\phi\big(s_d,\, a_d^n\big). \tag{5}$$

This estimator is low-variance and independent of the remaining search depth and error from the model, but relies entirely on Q_ϕ being learned accurately.

State value estimator. We can alternatively estimate using the world model's predicted per-step reward $r_\psi(s_d, a_d^n)$ and a discounted value of the next state:

$$V_r\big(d \mid s_d\big) \;=\; \operatorname*{agg}_{n\in[N]} \left[\operatorname*{agg}_{k\in[K]} \left[r_\psi\big(s_d, a_d^n\big) \;+\; \lambda\, V\big(d+1 \mid s_{d+1}^{n,k}\big) \right] \right], \tag{6}$$

where $s_{d+1}^{n,k} = M_\psi\big(s_d, a_d^n\big)$ and $r_\psi(s_d, a_d^n)$ is the learned reward model. At the final layer, N_{leaf} actions are sampled from the policy and there is no rollout left to recurse into, so the leaf value reduces to state value which can be computed as an aggregate over Q-values for actions in that state,

$$V\big(D \mid s_D\big) \;=\; \operatorname*{agg}_{n\in[N_{\text{leaf}}]} \; Q_\phi\big(s_D,\, a_D^n\big). \tag{7}$$

The state value estimator exploits the full imagined rollout but compounds world-model error with depth.

Combined node value. The two estimators have complementary failure modes: V_Q is depth-independent but blind to the rollout, while V_r uses the rollout but accumulates simulation error. We take advantage of both estimators to get a combined node value:

$$V\big(d \mid s_d\big) \;=\; \tfrac{1}{2}\Big(V_Q\big(d \mid s_d\big) + V_r\big(d \mid s_d\big)\Big) \tag{8}$$

which can be extended to a weighted average $\alpha V_Q + (1-\alpha)V_r$, where $\alpha \in [0, 1]$.

Root value. The value of each originally sampled root action a_0^n follows the same computation at $d = 0$:

$$Q_{ts}(s_0, a_0^n) \;=\; \tfrac{1}{2}\left(Q_\phi(s_0, a_0^n) \;+\; \left[r_\psi(s_0, a_0^n) + \operatorname*{agg}_{k\in[K]} \lambda\, V\big(1 \mid s_1^k\big) \right] \right), \qquad s_1^k = M_\psi(s_0, a_0^n). \tag{9}$$

The action ultimately selected is a max or softmax over the original sampled actions, weighted by their tree-search values $Q_{ts}(s_0, a_0^n)$. We use this search procedure in two places: to select actions during training, and to act at evaluation time.

4.3 SEARCHING WITH THE Q-FUNCTION.

Because the tree grows very large very quickly with depth, due to computational constraints we in practice design a heuristic to select only a subset of nodes to expand and prune the rest using the Q-function as a search policy. Let J be the number of paths that will be expanded. Let $\mathcal{B}_d$ denote the surviving set of J partial paths after pruning at depth d (with $\mathcal{B}_0 = \{a_0^n\}_{n=1}^N$, the single root path under consideration). Each surviving path $b \in \mathcal{B}_d$ carries an accumulated discounted score

$$\Sigma(b, d) \;=\; \sum_{d'=1}^{d} \lambda^{d'}\, Q_\phi\big(s_{d'}^b, a_{d'}^b\big), \tag{10}$$

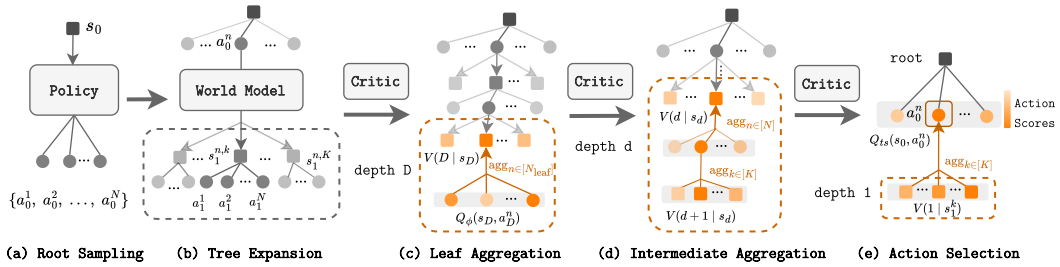


Figure 2: **Tree Search and Value Aggregation in QWM.** From the current state s_0, the policy proposes N candidate actions. The world model predicts the corresponding future states, and this expansion repeats to depth D. Rather than evaluating only the leaf nodes, values are aggregated from intermediate nodes, where deeper imagined branches are discounted according to the tree-search discount λ.

where $\left(s^b_{d'}, a^b_{d'}\right)$ is the state-action pair visited by path b at depth d'.

At depth $d+1$, every surviving path $b \in \mathcal{B}_d$ is expanded into N candidate children $\{a^{b,n}_{d+1}\}_{n=1}^{N} \sim \pi_\theta(\cdot \mid s^b_{d+1})$, each scored by its prospective cumulative sum:

$$\tilde{\Sigma}(b, n, d+1) \;=\; \Sigma(b, d) \;+\; \lambda^{d+1}\, Q_\phi\big(s^{b,n}_{d+1}, a^{b,n}_{d+1}\big). \tag{11}$$

The surviving set at depth $d+1$ is the top J over *all* candidates, ranked by Equation (11):

$$\mathcal{B}_{d+1} \;=\; \text{top-}J\Big\{\tilde{\Sigma}(b, n, d+1) \;:\; b \in \mathcal{B}_d,\; n \in [N]\Big\}. \tag{12}$$

At depth D, each surviving path $b \in \mathcal{B}_{D-1}$ samples N_{leaf} final actions, whose values are aggregated per-beam:

$$L(b) \;=\; \operatorname*{agg}_{n \in [N_{\text{leaf}}]}\; Q_\phi\big(s^b_D, a^{b,n}_D\big). \tag{13}$$

The root's tree-search value combines all surviving beams. Intuitively, Equation (12) selects survivors by the *additive discounted sum* $\tilde{\Sigma}$. When J equals the max number of nodes, no candidates are discarded at Equation (12) ($\mathcal{B}_d$ always contains all N^d paths). When J is less than the max number of nodes, Equation (12) discards candidates at every intermediate depth based on Equation (11).

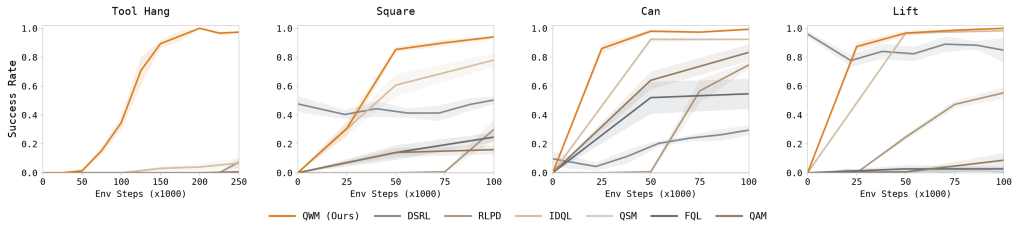
4.4 Practical Implementation

We instantiate the action-conditioned world model in two forms depending on the observation modality. In practice, because we operate in sparse reward settings where rewards are zero except for the terminal state, we do not learn a reward model on top of the world model. For low-dimensional state observations, we use a deterministic residual dynamics model $M_\psi(s_t, a_t) = s_t + \Delta_\psi(s_t, a_t)$ implemented as a three-layer MLP with hidden dimension 256. The model takes the Robomimic low-dimensional state representation, including end-effector pose, gripper states, and object features, together with the 7-DoF action as input. It is pretrained offline on demonstration transitions using an MSE objective. For pixel observations, we adapt Wan2.2-TI2V-5B (Wan et al., 2025) into an action-conditioned video world model by introducing an MLP-based action encoder. Specifically, each robot action is mapped into action tokens through a three-layer MLP, which are then concatenated with the text conditioning tokens and provided to the diffusion transformer as additional conditioning inputs. The action encoder and diffusion transformer are jointly fine-tuned while keeping the VAE and text encoder frozen. The model is trained on demonstration video clips with aligned action sequences using the standard Wan2.2 flow-matching objective. During inference, the world model predicts short-horizon future observations conditioned on candidate actions. For LIBERO, we generate 5-frame 128×128 video clips, and use the predicted next frame as the subsequent observation for iterative rollout and tree search.

5 Experiments

The goal of our experiments is to answer the following core questions:

- **(Q1)** How does QWM perform compared to state-of-the-art model-free and model-based RL methods?



Compared with model-free baselines. Success rates of QWM and model-free baselines in the online settings. QWM outperforms strong model-free baselines in sample efficiency.

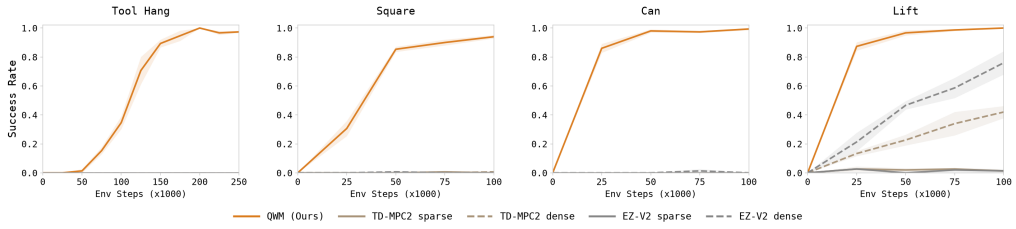


Figure 3: **Compared with model-based RL methods.** Success rates of QWM, TD-MPC2, and EZ-V2 across Lift, Can, Square, and Tool Hang under sparse and dense reward settings. QWM achieves strong performance across challenging manipulation tasks.

(Q2) How does QWM perform compared to the base method it is implemented on top of?

(Q3) Does QWM scale to a pixel-based setting?

(Q4) What components of QWM are most important for performance?

Environments. We evaluate QWM on challenging robotic manipulation tasks from Robomimic (Mandlekar et al., 2021) and LIBERO (Liu et al., 2023), where a 7-DoF robot arm is required to complete diverse manipulation behaviors under sparse task-completion rewards. For Robomimic, we consider four tasks: Lift, Can, Square, and Tool Hang. Lift requires the robot to grasp and lift an object; Can requires the robot to grasp a cylindrical object and place it at a target location; Square involves precisely inserting a square nut onto a peg; and Tool Hang is a long-horizon multi-stage assembly task where the robot constructs a stand and hangs a tool. For LIBERO, we evaluate on five tasks following the experimental protocol of prior work (Dong et al., 2026d).

Baselines. We compare against both state-of-the-art model-free algorithms and model-based algorithms. We refer to Section B.4 for details on algorithmic comparisons.

5.1 How Does QWM Perform Compared to State-of-the-Art Model-Free and Model-Based RL Methods?

Model-free. We compare against strong model-free RL baselines in Section 5.1, all of which leverage online interaction to learn Q-functions. QWM achieves the strongest performance across all tasks. RLPD, DSRL, QSM, QAM, and FQL each sample a single action from the policy for execution in the environment. While this action is intended to approximately maximize the Q-function, optimization delays and errors prevent it from being truly optimal. QWM instead leverages the world model's ability to predict future outcomes, performing test-time search over candidate actions and selecting the ones with the best predicted future returns. While IDQL selects actions by sampling multiple candidates and selecting the one with the highest Q-value, it considers only the immediate next action rather than a sequence of future actions, the latter carrying substantially more signal about long-term performance. These results demonstrate that the test-time search in QWM provides benefits complementary to Q-learning, enabling better action selection without substituting online interactions.

Model-based. We further compare QWM with state-of-the-art model-based RL methods TD-MPC2 and EfficientZero V2. We include both sparse and dense reward variants for the model-based algorithms, as many of these methods were designed for dense rewards, and struggles to learn as well in sparse reward settings. We present the results in Figure 3. QWM achieves consistently stronger performance across the evaluated manipulation tasks, while TD-MPC2 and EfficientZero V2 only obtain non-zero success on Lift for the number of training steps reported. Different from conventional

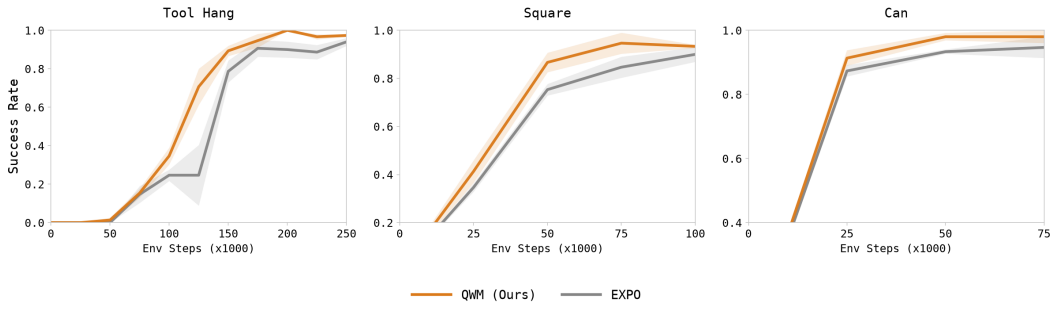


Figure 4: **Effect of world-model tree search on EXPO.** Online success rates of EXPO and QWM on Tool Hang, Square, and Can. Tree search consistently improves sample efficiency, with larger gains on Tool Hang and Square.

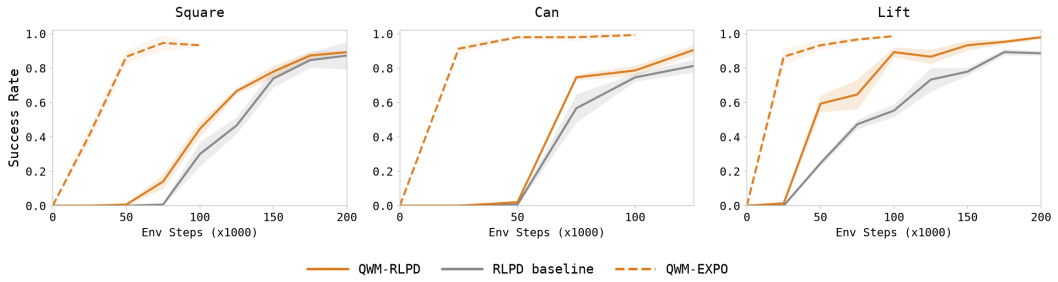


Figure 5: **Effect of world-model tree search on RLPD.** Online success rates of RLPD, RLPD with tree search, and QWM on Square, Can, and Lift. Tree search improves the sample efficiency of RLPD across all three tasks, while QWM built on EXPO remains more sample-efficient overall.

model-based RL methods that incorporate model predictions into policy optimization or learning targets, QWM uses the world model only at test-time search while keeping policy and value learning grounded on real environment transitions. This design allows QWM to leverage future predictions for improved action selection without relying on model-generated trajectories during learning, providing an effective way to combine the predictive capability of world models with online Q-learning while avoiding compounding bias from the world model.

5.2 How Does QWM Perform Compared to the Base Method It Is Implemented on Top of?

In this section, we evaluate the performance of QWM compared to the base algorithm it is implemented on. We implement QWM on top of EXPO and RLPD for the paper as representative sample efficient Q-learning algorithms, but QWM can be implemented on other model-free algorithms as well. We start by comparing QWM against EXPO in Figure 4. QWM consistently improves learning efficiency across Tool Hang, Square, and Can, with particularly pronounced gains on harder tasks such as Tool Hang. Comparing with RLPD as the base algorithm, as shown in Figure 6, using QWM with RLPD consistently improves learning compared to RLPD alone with particularly clear gains in sample efficiency. These findings show that QWM provides consistent benefits across different underlying RL algorithms. At the same time, QWM will benefit from a more sample efficient base algorithm as shown by the results of QWM on top of EXPO compared to QWM on top of RLPD.

5.3 Does QWM Scale to a Pixel-Based Setting?

We further evaluate whether the performance gains of QWM extend to high-dimensional visual observations. Due to computational constraints, we use the world model only for sampling during online RL data collection, not for evaluation. As shown in Figure 6, QWM achieves a clear overall improvement over EXPO. The gains are most evident on Tasks 60, 79, and 29, where QWM learns faster and reaches stronger late-stage performance. On Task 28, both methods eventually attain near-perfect success, but QWM reaches high performance earlier, demonstrating a clear improvement in sample efficiency. As shown in the ablations in Figure 4, restricting the world model to sampling-only (without evaluation) reduces performance, suggesting the visual results can be even stronger given sufficient compute. Nonetheless, even in this sampling-only setting, QWM consistently improves

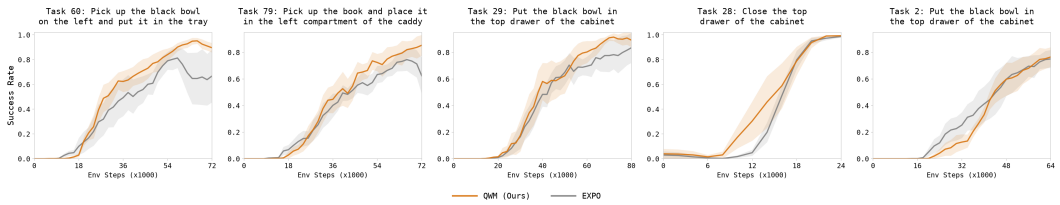


Figure 6: **Pixel-based evaluation on LIBERO.** Online success rates of QWM and EXPO across five LIBERO tasks. QWM achieves clear gains on Tasks 60 and 79, improves learning speed on Task 28, and reaches comparable or stronger final performance across the remaining tasks.

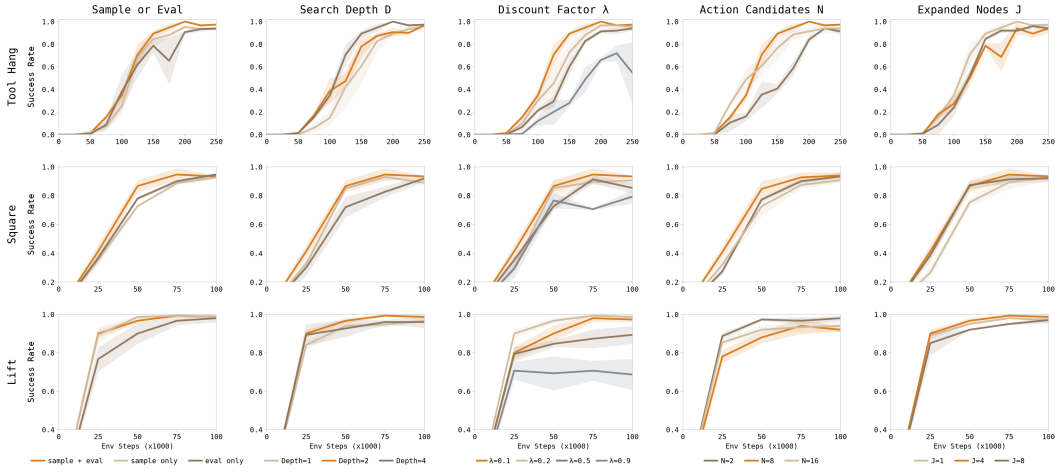


Figure 7: **Ablation of world-model lookahead on state-based tasks.** We compare applying world-model-guided action selection during online sampling, evaluation, or both stages. The *sampling + evaluation* variant achieves the most consistent improvement across tasks, showing the complementary benefits of improving online experience collection and test-time action selection.

over the base algorithm, showing that its benefits extend naturally to high-dimensional visual inputs, which are substantially harder to learn a dynamics model from.

5.4 What Components of QWM Are Most Important for Performance?

To better understand the significance of different pieces of QWM, we ablate over three key components: (1) the importance of using QWM during online RL sampling, evaluation, or both, (2) the hyperparameters for constructing the tree, and (3) the number of expanded nodes during search to constrain the tree size. We present the results in Figure 7. Overall, we find that using QWM in both sampling and evaluation, and choosing the value discount to matter most for performance. We present additional analysis on which component of QWM are most important and additional experiments comparing against performing search with only a state value function in Section A.

6 Discussion

In this paper, we present QWM, a framework for leveraging world models via test-time search over actions on top of Q-learning to improve performance. Rather than using the world model to optimize the policy directly, as in conventional model-based RL, we use it purely at test time both during online rollouts and at evaluation. We show that QWM outperforms strong model-free baselines, while avoiding the compounding model bias that afflicts methods which optimize policies primarily inside a learned world model. Despite these results, QWM has limitations. The tree search introduces non-trivial computational overhead at rollout and evaluation time relative to a standard policy forward pass, which may be prohibitive under tight latency requirements. Additionally, QWM depends on learning a world model, which is an expensive and often challenging to train; while grounding the policy and Q-function in real data prevents bias from the model from compounding, reducing search overhead and obtaining high quality world models remain important future directions.

AI USE STATEMENT

We did not use generative AI tools for developing the core research ideas, formulating the Q-Learning with World Models (QWM) framework, designing the world-model-guided tree search algorithm, interpreting experimental results, or making scientific claims. More generally, we did not use AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We used generative AI tools for language proofreading, improving writing clarity, and assisting with code development and debugging tasks, such as generating and refining code components and improving code readability. All AI-assisted content and code were carefully reviewed and verified by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work investigates reinforcement learning methods for robotic manipulation and does not involve human subjects, personally identifiable information, or private human data. The experiments are conducted on publicly available robotic manipulation benchmarks, including Robomimic and LIBERO, following their respective usage guidelines.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed QWM framework, training procedures, and experimental settings in the main paper and supplementary material. The algorithmic details of QWM are presented in Section 4, while the experimental protocols, benchmark descriptions, hyperparameters, and implementation details are provided in Section 5 and Appendix B. We evaluate our method on publicly available robotic manipulation benchmarks, including Robomimic and LIBERO.

REFERENCES

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling, 2023. URL https://arxiv.org/abs/2209.14548.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-N sampling in large language models. *CoRR*, abs/2412.15287, 2024. doi: 10.48550/arXiv.2412.15287. URL https://arxiv.org/abs/2412.15287.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Perry Dong, Alec M. Lessing, Annie S. Chen, and Chelsea Finn. Reinforcement learning via implicit imitation guidance, 2025a. URL https://arxiv.org/abs/2506.07505.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies, 2025b. URL https://arxiv.org/abs/2507.07986.

Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics?, 2025c. URL https://arxiv.org/abs/2505.08078.

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models, 2026a. URL https://arxiv.org/abs/2605.25477.

Perry Dong, Kuo-Han Hung, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. Tql: Scaling q-functions with transformers by preventing attention collapse, 2026b. URL https://arxiv.org/abs/2602.01439.

Perry Dong, Ron Polonsky, Dorsa Sadigh, and Chelsea Finn. Do you really need to pretrain q-functions for online rl fine-tuning?, 2026c. URL https://arxiv.org/abs/2607.27203.

Perry Dong, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. Faster: Value-guided sampling for fast rl, 2026d. URL https://arxiv.org/abs/2604.19730.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows, 2026e. URL https://arxiv.org/abs/2510.07650.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control, 2018. URL https://arxiv.org/abs/1812.00568.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018. URL https://arxiv.org/abs/1803.00101.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning, 2021. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2019. URL https://arxiv.org/abs/1812.02900.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration, 2016. URL https://arxiv.org/abs/1603.00748.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model, 2026. URL https://arxiv.org/abs/2602.12063.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022. URL https://arxiv.org/abs/2010.02193.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models, 2025. URL https://arxiv.org/abs/2509.24527.

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

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control, 2022. URL https://arxiv.org/abs/2203.04955.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023. URL https://arxiv.org/abs/2304.10573.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces, 2021. URL https://arxiv.org/abs/2104.06303.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled sailor: Robust imitation via learning to search, 2025. URL https://arxiv.org/abs/2506.05294.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2021. URL https://arxiv.org/abs/1906.08253.

Zhennan Jiang, Kai Liu, Yuxin Qin, Shuai Tian, Yupeng Zheng, Mingcai Zhou, Chao Yu, Haoran Li, and Dongbin Zhao. World4rl: Diffusion world models for policy refinement with reinforcement learning for robotic manipulation, 2026a. URL https://arxiv.org/abs/2509.19080.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, Yu Wang, Haoran Li, Chao Yu, and Dongbin Zhao. Wovr: World models as reliable simulators for post-training vla policies with rl, 2026b. URL https://arxiv.org/abs/2602.13977.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari, 2024. URL https://arxiv.org/abs/1903.00374.

Gabriel Kalweit and Joschka Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 195–206. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/kalweit17a.html.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning, 2026. URL https://arxiv.org/abs/2601.16163.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization, 2018. URL https://arxiv.org/abs/1802.10592.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-vla-loop: Closed-loop learning of video world model and vla policy, 2026. URL https://arxiv.org/abs/2602.06508.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps. *CoRR*, abs/2501.09732, 2025. doi: 10.48550/arXiv.2501.09732. URL https://arxiv.org/abs/2501.09732.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation, 2021. URL https://arxiv.org/abs/2108.03298.

Max Sobol Mark, Archit Sharma, Fahim Tajwar, Rafael Rafailov, Sergey Levine, and Chelsea Finn. Offline retraining for online rl: Decoupled policy learning to mitigate exploration bias, 2023. URL https://arxiv.org/abs/2310.08558.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey 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)*, pp. 7559–7566, 2018. doi: 10.1109/ICRA.2018.8463189.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021. URL https://arxiv.org/abs/2006.09359.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning, 2025. URL https://arxiv.org/abs/2502.02538.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019. URL https://arxiv.org/abs/1910.00177.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41163–41182. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/psenka24a.html.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL http://dx.doi.org/10.1038/s41586-020-03051-4.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model, 2021. URL https://arxiv.org/abs/2104.06294.

Ansh Kumar Sharma, Yixiang Sun, Ninghao Lu, Yunzhe Zhang, Jiarao Liu, and Sherry Yang. World-gymnast: Training robots with reinforcement learning in a world model, 2026. URL https://arxiv.org/abs/2602.02454.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/arXiv.2408.03314. URL https://arxiv.org/abs/2408.03314.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

GigaBrain Team, Boyuan Wang, Bohan Li, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, Lv Feng, Mingming Yu, Peng Li, Qiuping Deng, Tianze Liu, Xinyu Zhou, Xinze Chen, Xiaofeng Wang, Yang Wang, Yifan Li, Yifei Nie, Yilong Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. Gigabrain-0.5m*: a vla that learns from world model-based reinforcement learning, 2026. URL https://arxiv.org/abs/2602.12099.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2018. URL https://arxiv.org/abs/1707.08817.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. EfficientZero v2: Mastering discrete and continuous control with limited data. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51041–51062. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wang24at.html.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171, 2022. doi: 10.48550/arXiv.2203.11171. URL https://arxiv.org/abs/2203.11171.

Hanlin Yang, Chao Yu, Peng Sun, and Siji Chen. Hybrid policy optimization from imperfect demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 4653–4663, 2023.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, Ping Luo, Xiangyu Yue, and Hongyang Li. Rise: Self-improving robot policy with compositional world model, 2026. URL https://arxiv.org/abs/2602.11075.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data, 2021. URL https://arxiv.org/abs/2111.00210.

Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning, 2023. URL https://arxiv.org/abs/2302.00935.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning, 2019. URL https://arxiv.org/abs/1808.09105.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models, 2025. URL https://arxiv.org/abs/2511.09515.

A Additional Experiments

A.1 Search with state versus state and action

To isolate the benefit of performing test-time search with a Q-function, we compare against a variant that learns only a state value function V and searches with it. Here we fit V, form an advantage estimate, and optimize the policy by advantage-weighted regression (Peng et al., 2019). We evaluate this variant both as the edit policy within EXPO and as a standalone method. Results are reported in Figure 8 and Figure 9. Searching with V substantially underperforms searching on top of Q-learning across all settings. We attribute this to two factors: the Q-function supplies an additional action-conditioned estimator that V alone cannot provide, which yields a stronger estimator overall, and the underlying Q-learning base algorithm is itself stronger. These results motivate leveraging world models for Q-learning rather than performing search with V as typically done with model-based RL.

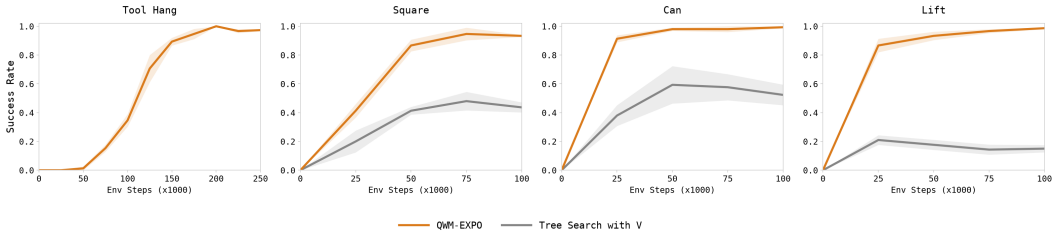


Figure 8: **Search with V vs Q on top of EXPO.** Online success rates of QWM-EXPO against a search with V variant where the edit policy is learned via a state value function V. Search with Q does substantially better than the V counterpart.

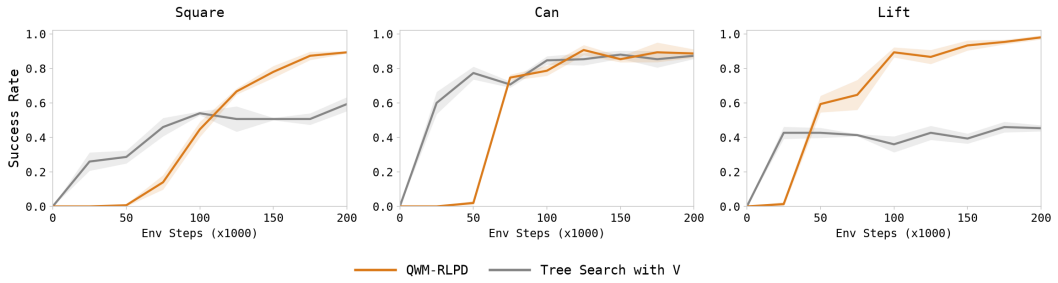


Figure 9: **Search with V vs Q as a standalone method.** Online success rates of QWM-RLPD against search with V variant with a Gaussian policy learned from V in the style of AWR. Search with Q does substantially better than the V counterpart.

A.2 Analysis on which components are most important for performance

Sampling versus evaluation. We first ablate whether QWM is applied during online RL data collection, at evaluation time, or in both. The *sampling only* variant performs test-time search during online data collection but disables it at evaluation, while the *evaluation only* variant applies tree search only at evaluation time. The full *sampling + evaluation* variant uses test-time search in both. As shown in Figure 7, applying test-time search at both stages provides the strongest and most consistent learning efficiency. The full variant improves faster and more smoothly and reaches high success rates earlier. Sampling-time search affects which transitions are collected and added to the replay buffer, allowing subsequent policy and critic updates to benefit from higher-value online experience. Evaluation-time search, in contrast, directly improves action selection without modifying the learned policy or training data. Performing test-time search at both stages therefore combines improved online experience collection with stronger execution-time action selection, which gives more consistent advantage over either single-stage variant.

Tree-search hyperparameters. We next ablate key hyperparameters for constructing the search tree: search depth D, recursive discount factor λ, and the number of action candidates N. In Figure 7,

we see that search depth has a clearer effect on learning. Depth 2 generally provides the strongest performance gain in our experiments, whereas depth 1 improves more slowly as a shallow search may not capture enough future consequence to reliably distinguish candidate actions. For tasks requiring longer term planning, higher depths can be beneficial, though making the depth too high could expose the search to additional world-model prediction error and uncertainty. Consequently, a moderate search depth provides a favorable trade-off between incorporating future information and maintaining reliable predictions.

The value aggregation discount factor λ, which controls how strongly future tree values contribute to the recursive value aggregation in Equation (8) is important for performance and setting λ too large or too small leads to lower performance, and relatively small to moderate values, particularly $\lambda = 0.2$ for our settings, provide the strongest learning efficiency. A larger λ places greater weight on values propagated from deeper imagined states, making the value aggregation more sensitive to accumulated world-model and value-estimation errors, while an overly small λ limits the contribution of a deeper search. Overall, the results favor large weighting of short-term estimates with a moderately small future-value weighting.

We next examine the effect of the number of action candidates N considered at each node of the search tree. Increasing N allows the search to evaluate a broader set of possible actions before committing to the one that maximizes the recursive value estimate, which in principle should improve the quality of the selected action by reducing the chance that a good candidate is overlooked. Empirically, we find that the number of actions N that performs the best is dependent on the environment. In general, performance improves as N increases from very small values, since too few candidates can fail to include actions that meaningfully diverge from one another, limiting the benefit of search altogether. However, the gains from increasing N diminish beyond a moderate number of candidates, and excessively large N can even hurt performance, likely because evaluating many candidates amplifies the influence of world-model and value-estimation errors across a larger number of imagined rollouts, while also increasing computational cost with limited additional benefit. We find that a moderate number of action candidates offers the best trade-off between exploration of the action space and robustness to compounding prediction errors, and we adopt this setting as default in our main experiments.

Number of expanded nodes. Lastly, we ablate over the number of expanded nodes in our search heuristic. In Figure 7, we observe that performance is relatively insensitive to the number of expanded nodes J, as the heuristic accounts for which paths have the largest value to decide which nodes to keep expanding and which to prune. While this greedy approach does not guarantee finding the highest value paths of the full tree, it provides a good approximation. This suggests that retaining a small number of high-value branches is already sufficient to capture useful candidate futures.

B Experiment Details

B.1 Hyperparameters

For the state-based Robomimic experiments, QWM and all model-free baselines follow the same online RL protocol, including identical demonstration sources, online interaction budgets, and optimization settings. Unless otherwise specified, all methods start online training after 5,000 environment steps, use an offline data ratio of 0.5, do not perform additional offline pretraining, and use an update-to-data (UTD) ratio of 20 with batch size 256. For Q-learning-based methods, we use a critic TD discount $\gamma = 0.99$ and a target network update coefficient $\tau = 0.005$. Importantly, γ is used only in the Bellman TD backup of the critic and is distinct from the tree-search discount λ, which controls the contribution of deeper future values during search. For EfficientZero V2, we follow the official training protocol provided by the authors and use its recommended UTD ratio of 0.4.

QWM-specific hyperparameters. We instantiate QWM on top of two base RL algorithms, EXPO and RLPD. For both variants, the original policy and critic optimization procedures are kept unchanged, while world-model-guided tree search is additionally used for action selection during online interaction and evaluation.

Table 1: Shared optimization hyperparameters for QWM and model-free baselines on Robomimic.

Hyperparameter	Value
Start training steps	5,000
Offline data ratio	0.5
Offline pretraining steps	0
UTD ratio	20
Batch size	256
Critic TD discount γ	0.99
Target update coefficient τ	0.005

At each state node, the policy proposes N candidate actions, and each action is evaluated through K world-model predictions. To control the size of the search tree, we retain at most J partial paths during intermediate expansion, following the pruning procedure described in Section 4.4. We use max aggregation for both intermediate-node values and leaf values. Specifically, intermediate-node aggregation combines the values of candidate actions and their predicted future states in the recursive value computation in Equation (8), while leaf aggregation combines the N_{leaf} Q-values sampled at the final search depth in Equation (7). The same max aggregation is used when combining world-model branches in the root action score. The tree-search settings are summarized in Table 2. Here, λ is used only within the tree-search value recursion and is independent of the critic TD discount γ. We note that this λ removes the factor of $\frac{1}{2}$ in Equation (9) and Equation (8), and to exactly match the equations the λ in the table should be multiplied by $\frac{1}{2}$.

Table 2: Shared tree-search settings for state-based Robomimic experiments.

Hyperparameter	Value
Candidate actions N	8
World-model samples per action K	8
Intermediate-node aggregation	Max
Leaf-value aggregation	Mean
Search during online sampling	Yes
Search during evaluation	Yes
Nodes expanded J	1
Number of leaf actions N_{leaf}	8
Depth D	4
Value aggregation discount λ	0.1

B.2 Environments

We evaluate QWM on two widely used robotic manipulation benchmarks: Robomimic (Mandlekar et al., 2021) and LIBERO (Liu et al., 2023). These benchmarks provide diverse manipulation tasks with human-collected demonstrations and are commonly used to evaluate offline-to-online robot learning algorithms.

Robomimic. For state-based experiments, we evaluate QWM on four manipulation tasks from Robomimic: Lift, Can, Square, and Tool Hang. These tasks cover diverse manipulation capabilities, including object grasping and lifting, object relocation, precise insertion, and long-horizon assembly. We use the low-dimensional state observations provided by Robomimic, consisting of robot proprioception and object-related features, including end-effector pose, gripper states, and object states. The action space is a 7-DoF operational space control (OSC) command.

Following prior work, we use different demonstration settings for different tasks. Lift uses a 10-episode subset of the original dataset to create a more challenging evaluation setting, Can uses the multi-human (MH) dataset, and Square and Tool Hang use the standard proficient-human (PH) split. The PH dataset contains demonstrations collected by a single proficient teleoperator, while the MH dataset contains demonstrations collected by multiple teleoperators with varying levels of expertise. All experiments follow the standard Robomimic evaluation protocol and report online success rates.

LIBERO. For pixel-based experiments, we evaluate QWM on five tasks from the LIBERO benchmark. LIBERO is a language-conditioned lifelong robot learning benchmark designed to evaluate knowledge transfer and generalization in robotic manipulation. It contains diverse manipulation tasks with different object configurations, spatial arrangements, and task semantics.

We follow the task selection and evaluation protocol used in prior work and evaluate on five representative tasks (Task 60, Task 79, Task 29, Task 28, and Task 2). The agent receives RGB observations from the robot cameras and generates actions conditioned on visual observations and task instructions. We use the standard LIBERO simulation setup and report online success rates averaged over evaluation episodes.

B.3 Additional World Model Details and Prediction Quality

We provide additional training and inference details for the world models used in our experiments. For the state-based experiments, the dynamics model is pretrained offline on demonstration transitions (s, a, s') using an MSE objective with Adam, a learning rate of 3×10^{-4}, batch size 256, and 100k gradient steps, and is kept fixed throughout subsequent policy and critic optimization. For the vision-based experiments, we fine-tune the action-conditioned Wan2.2-TI2V-5B world model on demonstration videos paired with aligned raw action sequences. Before training, video clips are encoded once by the frozen Wan2.2 VAE and text prompts are encoded by the frozen umT5-XXL encoder, allowing both representations to be cached and removed from the training loop. We then jointly fine-tune the diffusion transformer and action encoder using the standard Wan2.2 flow-matching objective, while keeping the VAE and text encoder frozen. Training uses AdamW with a learning rate of 1×10^{-5}, a per-GPU batch size of 1 on 8 GPUs, bfloat16 precision, gradient checkpointing, and 200k gradient steps. For LIBERO, we train on 5-frame 128×128 clips and model the agent-view and wrist-camera observations as two video streams. During tree search, however, each world-model query advances the rollout by only one step: we use the second frame of the generated clip as the predicted next observation, and then condition a new generation on this prediction for subsequent tree expansion. At inference time, we use 1 denoising step. To examine the next-step generation quality of the resulting world model, Figure 10 compares the generated next observation with the corresponding ground-truth next observation conditioned on the same current observation and action. The predictions capture the task-relevant changes in robot configuration and object motion at the next step, supporting their use as predicted future observations during tree search.

B.4 Comparisons

We evaluate QWM against both state-of-the-art model-free algorithms and model-based algorithms, as while QWM uses a learned model, it does not use data from the model for training and is trained on top of standard Q-learning.

Model-free baselines. The first set of comparisons is against state-of-the-art model-free algorithms. As QWM learns with only data in the real world instead of from the model, we compare against other such methods to evaluate effectiveness.

EXPO (Dong et al., 2025b). EXPO jointly learns an expressive base policy and a lightweight edit policy, where the edit policy transforms actions sampled from the base policy toward higher-value distributions.

IDQL (Hansen-Estruch et al., 2023). IDQL trains a diffusion policy through behavior modeling and performs implicit policy extraction using best-of-N sampling, selecting the candidate action with the highest Q-value.

RLPD (Ball et al., 2023). RLPD is a sample-efficient off-policy RL method that leverages replay data with high update-to-data ratios, critic ensembles, and a Gaussian policy.

QSM (Psenka et al., 2024). QSM incorporates gradients from the learned Q-function into the diffusion training objective, guiding the denoising process toward high-value actions through score matching.

DSRL (Wagenmaker et al., 2025). DSRL adapts a frozen diffusion-based behavior cloning policy by performing reinforcement learning over the initial noise used for action generation, learning a policy that predicts the noise seed.



Figure 10: **Qualitative comparison of world-model predictions on LIBERO.** We compare the current observation, ground-truth future observations, and action-conditioned predictions over short rollout horizons. The predicted observations capture the major robot and object changes relevant to tree-search action evaluation, while prediction errors become more visible at later rollout steps.

QAM (Li & Levine, 2026). QAM uses adjoint matching to propagate value gradients through the diffusion trajectory and constructs policy optimization objectives based on the learned Q-function.

FQL (Park et al., 2025). FQL trains a one-step flow policy to maximize Q-values learned with a temporal-difference objective, while regularizing the policy toward a behavior-cloned flow policy.

Model-based baselines. The second category of comparison methods consists of model-based RL algorithms, which similarly learn a dynamics model but, unlike our approach, use this model directly for training.

TD-MPC2 (Hansen et al., 2024). TD-MPC2 learns an implicit latent world model and performs short-horizon MPPI planning in latent space. It evaluates sampled action sequences using predicted rewards and bootstrapped terminal values, with a learned maximum-entropy policy prior guiding the sampling process.

EfficientZero V2 (Wang et al., 2024). EfficientZero V2 learns a MuZero-style latent world model and performs Sampling-based Gumbel tree search for both discrete and continuous control. The search provides bootstrapped policy and value targets that are reused for policy and value training.