VETO: REWARD-SUPERVISED WORLD MODELS AS ACTION VERIFIERS FOR SAMPLE-EFFICIENT RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning pre-trained manipulation policies with reinforcement learning (RL) is a promising route to real-world adaptation, but its sample efficiency remains the limiting factor on real robots. A key source of wasted interactions comes from *credit-assignment lag*, i.e., RL policies repeatedly reattempting failure modes they have already observed because observed outcomes must be propagated through a slowly converging temporal-difference (TD) critic, and then through the policy, before behavior changes. Model-based RL methods are often used to improve sample efficiency since they rely on supervised learning, but they can introduce model-error bias into policy gradients. Our key insight is that action ranking need not rely on TD learning: a latent world model trained with *supervised reward prediction* can provide useful rankings with fewer transitions than TD-trained Q-functions. We introduce **VETO** (**V**erifying **E**xploratory **T**rajectories **O**nline), which uses a continually learning, lightweight world model as an online action verifier. The policy proposes candidate action chunks, and the verifier scores them by rolling out latent dynamics and predicting rewards to veto poor action candidates. Because policy and critic targets use real transitions rather than imagined rollouts, world-model errors do not enter the Bellman backup, allowing for asymptotic optimality. Across 7 tasks in 2 simulation suites and on real-world YAM and Franka platforms, **VETO** adapts in fewer environment interactions than off-policy and model-based baselines and improves pre-trained policies in the real world.

1 INTRODUCTION

As pre-trained manipulation policies are deployed in the real world and face generalization challenges during zero-shot deployment (Wang et al., 2025; Gao et al., 2026), improving their performance via autonomous experience is becoming increasingly important. Fine-tuning of pre-trained policies with reinforcement learning (RL) presents a promising solution (Wagenmaker et al., 2025a; Intelligence et al., 2025; Zhang et al., 2024b; Luo et al., 2025; Zhang et al., 2024a; Yin et al., 2025; Yang et al., 2024; Hong et al., 2026), but *sample efficiency* remains a limiting factor on real robots, even for reportedly efficient off-policy RL algorithms (Mirchandani et al., 2024). A surprising observation is that much of that sample cost is not spent acquiring new information; instead, RL policies repeatedly reattempt failure modes they have already observed because their outcomes require many updates to reach the policy. We denote this suboptimal behavior as *credit-assignment lag*.

The lag stems from the way off-policy RL algorithms are trained: failures first propagate through the critic via slow-converging temporal-difference (TD) learning (Baird, 1995; Tsitsiklis & Van Roy, 1997; Kumar et al., 2022), then via high-variance policy gradient updates to change the actor's behavior. While TD learning is slow due to value bootstrapping, it is asymptotically performant: given enough data and updates, it should learn the correct values of poor actions. On the other hand, model-based RL methods (Sutton, 1990) can learn faster by using *supervised learning* to fit dynamics models to online interaction data and then using them to generate synthetic trajectories for policy training (Hafner et al., 2019a; Janner et al., 2019; Guo et al., 2026a; Yin et al., 2026; Jain et al., 2025). While these models learn quickly, their predictions carry model error, biasing policy updates and leading to potentially suboptimal behavior at convergence. To develop a practical solution for real-world robotic adaptation, we aim to reduce credit assignment lag without sacrificing asymptotic optimality.

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.

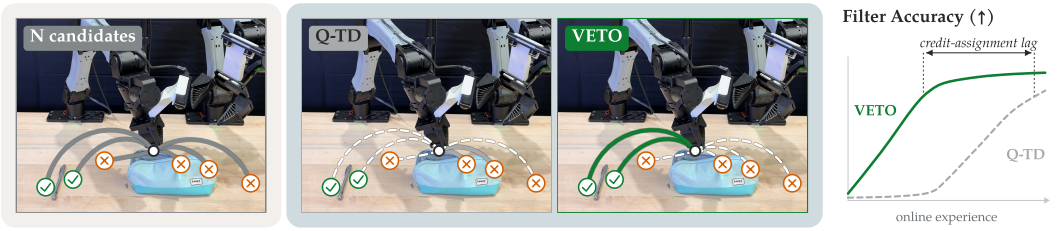


Figure 1: **VETO** trains with supervised learning, filters actions correctly faster than verifiers trained with TD-learning by guiding online action selection to higher-reward regions.

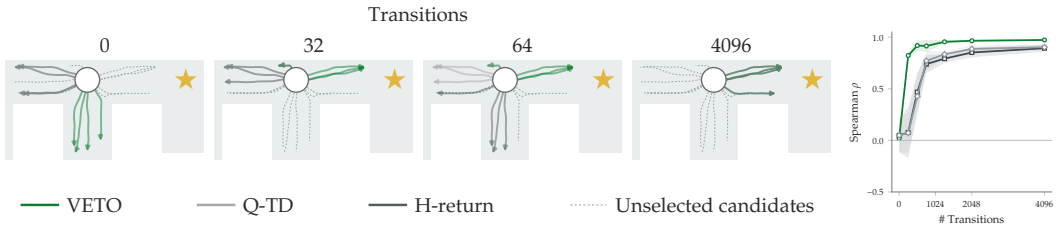


Figure 2: **Reward-supervised world model ranks actions correctly with less data than TD.** We train **WM-Reward**, **Q-TD**, and **H-return** (directly regress against H-step return in the replay buffer) on growing PointMaze transition subsets (Park et al., 2025) to rank shared candidate chunks (—). **Left:** Selections; ★ is the goal. **Right:** Spearman correlation with ground-truth chunk-return ranking.

We propose training a world model to act solely as an *action verifier* during data collection, filtering out low-reward action chunks before execution (Figure 1). This model can be trained with supervised learning on rewards, improving sample efficiency by easing the credit assignment lag. Meanwhile, the underlying off-policy RL algorithm avoids model-synthesized data and trains exclusively on real interaction data in order to achieve the same asymptotic performance. For example, we demonstrate in Figure 2 that a world model that predicts *rewards* (`WM-Reward`) under supervised learning ranks candidate action chunks correctly with far fewer transitions than Q-functions trained with off-policy RL. Using this model to filter policy actions during exploration feeds the RL algorithm *higher-reward* data earlier than relying solely on TD learning, avoiding credit assignment lag.

Turning this insight into an algorithm for sample-efficient RL, we introduce **VETO** (**V**erifying **E**xploratory **T**rajectories **O**nline). **VETO** uses a lightweight latent world model as an online *action verifier*: during online data collection, an RL policy proposes N action chunk candidates, the verifier scores them by rolling out latent dynamics and reward prediction, and one chunk is sampled uniformly from the top K. This verification shifts execution toward actions the verifier predicts will yield greater task progress while preserving some diversity for exploration. The collected experience then updates the actor and critic through off-policy RL; because the verifier is independent of the policy class, **VETO** pairs with most off-policy RL algorithms. When task rewards are unavailable, a general-purpose video reward model labels the collected trajectories. These labels supervise reward prediction, while observed transitions supervise dynamics prediction, allowing **VETO**'s world model to be continuously updated with online experience.

Empirically, we demonstrate across 2 simulation suites that **VETO**, integrated with diffusion steering off-policy RL methods, adapts in fewer interactions than off-policy and model-based baselines. Furthermore, across a bimanual YAM platform and a Franka 3, **VETO** demonstrates a 2.5× improvement in success rate over iterative supervised behavioral cloning adaptation of pre-trained π_0 vision-language-action and Flex-π world-action-model policies.

2 Related Works

Adaptation with World Models. Dynamics models, now often termed world models, have long been used to support planning, control, and policy learning (Sutton, 1990; Nagabandi et al., 2018; Kamthe & Deisenroth, 2018; Chua et al., 2018; Ha & Schmidhuber, 2018; Hafner et al., 2019a;b;

Zhang et al., 2020; Hafner et al., 2020; Hansen et al., 2022; 2024; Hafner et al., 2025; Hansen et al., 2026; Wang et al., 2026a). These approaches have enabled sample-efficient RL on real hardware (Nagabandi et al., 2020; Han et al., 2025; 2026; Wu et al., 2023; Lancaster et al., 2024), often demonstrating sample-efficiency gains over model-free approaches.

Building on these advances, recent approaches use pretrained video-generation models (Guo et al., 2026b; Wan et al., 2025; Liao et al., 2026; NVIDIA, 2025) to improve generalist robot policies by training on model-generated trajectories (Liang et al., 2025; Guo et al., 2026a; Sharma et al., 2026; Liu et al., 2026; Yin et al., 2026; Yang et al., 2026; Wang et al., 2026b). Generating these rollouts can be computationally expensive, and model errors can propagate into policy updates. Instead, **VETO** uses its world model exclusively as an action verifier to prevent biasing policy updates.

Steering policies with verifiers. Another line of work that is relevant to ours is that of using *verifiers* in the loop to steer pretrained policies to better perform a downstream task. These methods train a generalist verifier, such as a vision-language-input scoring function (Nakamoto et al., 2024; Kwok et al., 2025; 2026; Frans et al., 2026; Jain et al., 2026; Peng et al., 2026), a success classifier (Attarian et al., 2026), or a dynamics model (Wu et al., 2026; Neary et al., 2025; Du & Song, 2025), and perform best-of-N sampling or inject explicit action sampling guidance to determine which action to execute. However, due to model size or unavailability of training signals, these works do not *improve* their verifiers during online adaptation—**VETO** does, continually updating a lightweight latent dynamics model with supervised reward and next state targets.

Finally, a line of work directly uses the Q-function learned during RL as a verifier. With these methods, the Q-function improves with online experience to further improve verification. It's used at inference time either directly via best-of-N sampling (Mark et al., 2024; Sun & Song, 2026; Patil et al., 2026; Giridhar et al., 2026), or through tree search in imagined states from a frozen image-generation world model (Dong et al., 2026). Because the verifier here is the TD-learned critic itself, its ranking quality is still limited by the slow convergence of TD-learning.

3 VETO: REWARD-SUPERVISED WORLD MODEL ACTION VERIFICATION

We now introduce **VETO**, a framework that performs latent world model-based verification to guide data collection for sample-efficient RL. In this section, we describe the problem setting, how **VETO** interacts with the RL policy during online adaptation, and the architecture and training objectives of **VETO**'s underlying world model verifier. See Figure 3 for an overview.

3.1 PRELIMINARIES

Problem setting. We aim to enable RL improvement of a policy $\pi_{\text{base}}(\mathbf{a}|\mathbf{s})$ pre-trained on offline data $\mathcal{D}_{\text{off}}$ taking (chunked) actions $\mathbf{a}$ given state information $\mathbf{s} = \{\mathbf{s}_{\text{lang}}, \mathbf{s}_{\text{img}}, \mathbf{s}_{\text{proprio}}\}$ consisting of language, image, and proprioception. We focus on sample efficiency, so the goal is to maximize environment return with minimal environmental interaction, i.e., minimize cumulative regret.

RL on pre-trained policies. We focus on off-policy RL methods that aim to be sample-efficient by reusing previously collected data to improve π_{base}. DSRL (Wagenmaker et al., 2025b) and TMRL (Hong et al., 2026) improve pre-trained diffusion or flow policies that map latent noise $\omega \in N(0, I)$ to actions, giving $\pi_{\text{base}}(\mathbf{a} \mid \mathbf{s}, \omega)$. These methods train a noise policy $\pi_\theta(\omega \mid \mathbf{s})$ to predict the latent noise of π_{base} with off-policy RL (Haarnoja et al., 2018). The steered policy induces the following action distribution during RL: $\pi_{\text{RL}}(\mathbf{a} \mid \mathbf{s}) := \int \pi_{\text{base}}(\mathbf{a} \mid \mathbf{s}, \omega)\, \pi_\theta(\omega \mid \mathbf{s})\, d\omega$. **VETO** is agnostic to the underlying off-policy algorithm; our experiments use DSRL, TMRL, and iterative behavioral cloning on top of pre-trained flow policies.

Latent world model. **VETO** enables sample-efficient RL by training a world model to act as an action verifier for π_{RL}, selecting higher-reward action chunks during data collection. For fast training and verification, we train a lightweight, *latent* dynamics model $\texttt{WM}_\phi$ structured similarly to latent models from prior work (Hansen et al., 2024; Zhou et al., 2024; Hansen et al., 2026):

$$
\begin{array}{lll}
\text{Language encoder} & \mathbf{g} = \texttt{TextEncoder}(\mathbf{s}_{\text{lang}}), & \triangleright \text{ Encodes natural language instruction} \\
\text{Image encoder} & \mathbf{z} = \texttt{VisEncoder}(\mathbf{s}_{\text{img}}), & \triangleright \text{ Encodes RGB image observation} \\
\text{Latent dynamics} & \mathbf{z}' = f_\phi(\mathbf{z}, \mathbf{a}), & \triangleright \text{ Predicts latent forward dynamics} \\
\text{Reward} & \hat{r} = R_\phi(\mathbf{z}, \mathbf{g}). & \triangleright \text{ Predicts reward } r \text{ of a transition}
\end{array} \tag{1}
$$

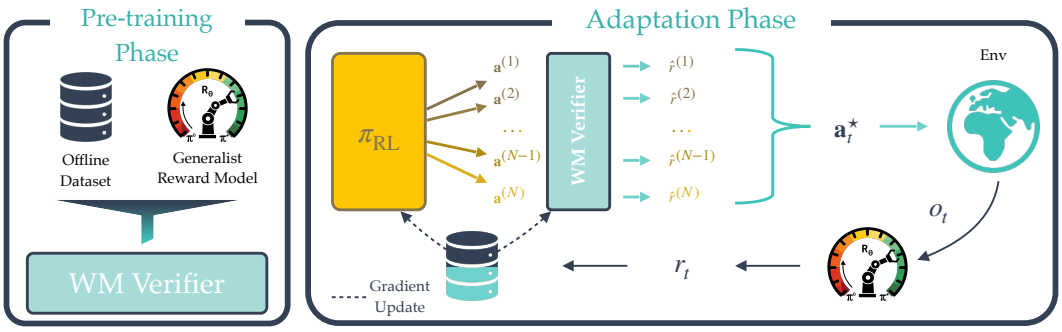


Figure 3: **VETO: (Pretraining)** We train latent dynamics via next-latent prediction and the reward head via supervised regression on task-reward labels. **(Adaptation)** We sample N candidate action chunks from π_{RL}. The verifier rolls out latent dynamics over the chunk, accumulates predicted rewards, and selects uniformly from the top K chunks. $\texttt{WM}_\phi$ adapts with the RL policy: $\texttt{WM}_\phi$ continues training on new transitions, and the policy is updated via any off-policy RL algorithm.

Here, $\mathbf{z}$ represents the latent space, $\mathbf{g}$ the latent goal, and $\hat{r}$ the predicted reward, used for scoring actions during online RL adaptation to filter out actions with low rewards.

3.2 Adaptation with VETO

The central design goal of **VETO** is an action verifier that *filters out low-reward actions* during RL exploration while *adapting quickly to new experience*. **VETO** thus trains its world-model verifier solely on supervised objectives, without Q bootstrapping (eq. (1)). As a result, it learns useful candidate rankings with fewer transitions than a TD critic (Figure 2). Due to its compact size, it scores candidates with minimal overhead and can be re-fit on new transitions alongside RL updates, enabling *continual* steering toward better actions sooner than TD updates change π_{RL}'s behavior.

Overview. **VETO** starts with an optional pre-training phase for the world model $\texttt{WM}_\phi$. When offline pre-training data $\mathcal{D}_{\text{off}}$ is available, we first pre-train $\texttt{WM}_\phi$ on a supervised objective: reward prediction and reconstruction of frozen, future image embeddings. We detail the training objective and the world model architecture in Section 3.3. After pre-training $\texttt{WM}_\phi$, we proceed to online adaptation, using $\texttt{WM}_\phi$ as an action verifier on samples from π_{RL}. We next detail how we sample and filter actions using $\texttt{WM}_\phi$, and how we update both $\texttt{WM}_\phi$ and the RL algorithm on collected data.

Sampling and Verification. During online data collection, we draw N independent action-chunk samples $\{\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)}\}$ from $\pi_{\text{RL}}(\cdot \mid \mathbf{s})$. These candidates form the set ranked by $\texttt{WM}_\phi$. For each candidate $\mathbf{a}^{(i)}$ with chunk size H, we roll out the latent dynamics model over the chunk horizon:

$$\mathbf{z}_0 = \texttt{VisEncoder}(\mathbf{s}_{\text{img}}), \quad \mathbf{z}_{t+1} = d(\mathbf{z}_t^{(i)}, \mathbf{a}_t^{(i)}), \quad \hat{R}^{(i)} := \sum_{t=1}^{H} \gamma^{t-1} R\left(\mathbf{z}_t^{(i)}, \mathbf{g} = \texttt{TextEncoder}(\mathbf{s}_{\text{lang}})\right). \quad (2)$$

We rank all the N candidates by $\hat{R}^{(i)}$ and retain the top $K \ll N$ action chunks. To enable exploration among actions and prevent potential model errors from dominating action selection, we uniformly sample one action chunk from the top K to execute.[1]

Adaptation via RL with filtered data. All collected transitions $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$, where r comes from environment or vision-language-input reward model rewards, are stored in a replay buffer $\mathcal{B}$. π_{RL} is updated via any RL algorithm that can learn from the resulting data; for example, our experiments use diffusion steering methods such as DSRL (Wagenmaker et al., 2025b), trained with SAC (Haarnoja et al., 2018) on $\mathcal{B}$ via standard TD updates. Crucially, the RL algorithm is trained only on real, collected transitions so world modeling error cannot directly bias the policy or TD gradients. **We continue training $\texttt{WM}_\phi$ on collected data**, enabling continuous refinement of reward and dynamics prediction in the target environment. See Figure 3 for an overview of RL adaptation.

3.3 World Model Architecture and Training Objective

Following the latent world model overview in Equation (1), we now describe **VETO**'s specific instantiation and training objective. The key requirement is that the verifier be fast enough to score

[1] During the RL "warmup" phase we sample π_{RL} without **VETO** filtering to preserve initial action diversity.

Algorithm 1 VETO: one adaptation iteration and training rollout

Require: π_{RL}, world model $\texttt{WM}_\phi$, replay buffer $\mathcal{B}$, N, K, chunk len H

1: **for** each decision step every $t = 0, H-1, 2H-1, \ldots$ in episode **do**
2: Sample $\{\mathbf{a}^{(i)}\}_{i=1}^{N} \sim \pi_{\text{RL}}(\cdot \mid \mathbf{s}_t)$ and score $\hat{R}^{(i)}$ via latent rollout ▷ Equation (2)
3: Execute $\mathbf{a}^\star \sim \text{Uniform}(\text{top-}K \text{ by } \hat{R}^{(i)})$;
4: Relabel new transitions with reward model (Robometer); store transitions in $\mathcal{B}$
5: Update π_{RL} via off-policy RL and train $\texttt{WM}_\phi$ via supervised regression on $\mathcal{B}$ ▷ Equation (3)

every candidate action chunk online and update continually from online experience. Without continual adaptation, the verifier *cannot correct for model error* induced from π_{RL} collecting unseen transitions as it improves. This requirement directly motivates $\texttt{WM}_\phi$'s architecture, described below.

Architecture. We adopt a model architecture inspired by Newt (Hansen et al., 2026) (i.e., larger TD-MPC2; Hansen et al., 2024), but we drop Q-value prediction and use an improved language encoder to enable more effective pre-training on robotics tasks. Specifically, we use frozen `DINO-v2` (Oquab et al., 2023) `CLS` image and `Qwen3-Embedding-0.6B` (Zhang et al., 2025) language embeddings so that online updates are restricted only to small dynamics and reward MLPs (f_ϕ and R_ϕ). As a result, depending on the environment, only ∼4.5–12M parameters are trainable, making continual re-fitting practical without excessive wall-clock cost or training instability.

Training objective. Given a trajectory $\tau = (\mathbf{s_0}, \mathbf{a_0}, r_0, \mathbf{s_1}, \mathbf{a_1}, \ldots)$, we train $R_\phi(\mathbf{z}, \mathbf{g})$ and $f_\phi(\mathbf{z}, \mathbf{a})$ with ground truth reward and future observation embeddings. Both components of the world model are learned entirely through supervised objectives—no TD bootstrapping. The overall loss is:

$$\mathcal{L}(\phi) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{g})_{0:H-1}} \left[\sum_{t=0}^{H-1} \lambda^t \left(\underbrace{\| f_\phi(\mathbf{z}_t, \mathbf{a}_t) - \text{sg}(\mathbf{z_{t+1}}) \|_2^2}_{\text{Supervised latent dynamics loss}} + \underbrace{\left(R_\phi(\mathbf{z}_t, \mathbf{g}) - r_t\right)^2}_{\text{Supervised reward loss}} \right) \right], \quad (3)$$

where λ controls the per-timestep loss and sg denotes stop-grad. See Section D for hyperparameters.

Summary. VETO trains a world model to guide online action selection, filtering out low-reward actions while continuously adapting to newly collected experience. Its small size and fast-converging, supervised learning objectives enable it to quickly improve with minimal samples or wall-clock time. Finally, because it's updated with the underlying RL algorithm on newly collected data, it can filter out poor actions sooner than the underlying RL algorithm can adapt, thereby accelerating RL improvement. Algorithm 1 summarizes adaptation; the complete adaptation loop, including off-policy RL updates and online world-model refinement, is in Algorithm 2. Next, we theoretically analyze VETO's verifier to analyze conditions on *when* using VETO's verifier improves actions.

3.4 Theoretical Analysis: When the Verifier Improves Action Selection

In Appendix Proposition 1, we provide a condition for when VETO improves upon the base policy's selected actions during data collection based on regret against an *oracle* top-K action set. The overall takeaway is that **VETO does not need to exactly rank top-K candidates the same as the oracle**. Instead, disagreements can have low *selection regret* when incorrect candidates have similar values, still resulting in better action selection.

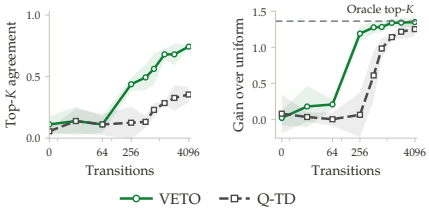


Figure 4: VETO selects higher-return chunks than Q-TD with less data. Left: exact top-K agreement. Right: 8-step return gain over uniform selection.

Empirical check. On OGBench PointMaze (Park et al., 2025) (N=16, K=4), Figure 4 compares exact agreement and H-step ($H = 8$) return gain over uniform selection from the N candidates. At 256 transitions, VETO matches the true top-K only 44% of the time yet achieves 87% of the oracle gain; Q-TD obtains 4.7%. This result empirically illustrates Proposition 1: return gain can remain high despite imperfect agreement with oracle top-K values.

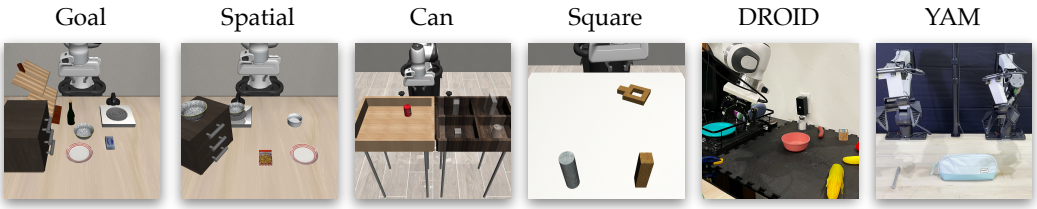


Figure 5: We evaluate **VETO** on manipulation tasks spanning simulation and the real-world.

4 EXPERIMENTS

Our experiments thoroughly investigate the following questions:

(Q1) Does **VETO** select better actions than TD-trained value functions early in adaptation?

(Q2) Does **VETO** improve the sample efficiency across different policy improvement algorithms?

(Q3) Can **VETO** enable effective real-robot adaptation under limited interaction budgets?

(Q4) What design choices are critical to **VETO** and why does **VETO** work?

Setup. We evaluate **VETO** on three Robomimic (Mandlekar et al., 2021) tasks, four LIBERO (Liu et al., 2023) tasks, and two real-world tasks, as shown in Figure 5. For Robomimic, we use the simulator's dense shaped rewards. For LIBERO and real-world tasks, where shaped rewards are unavailable, we derive dense rewards from task-progress estimates produced by Robometer (Liang et al., 2026) and DNA (Anonymous, 2026), reflecting the real-world setting in which task rewards are not directly available. Our real-world evaluation uses two platforms: a Franka robot in a DROID (Khazatsky et al., 2025) setup and a bimanual YAM. Across all tasks, policies and verifiers receive image observations and robot proprioception. Our pretrained policies use flow matching (Lipman et al., 2023) to generate action chunks conditioned on the current observation and sampled latent noise. In our experiments, **VETO** uses action verification only during data collection. We evaluate the learned policies without a verifier to measure policy improvement from the collected experience.

RL Baselines. We apply **VETO** on top of two noise-space steering methods, both of which train a noise policy with SAC over a frozen pre-trained flow policy (see Section 3.1). **TMRL** (Hong et al., 2026) in particular steers the flow policy pre-trained on noisy inputs, broadening the policy's action distribution on unseen tasks and giving the RL policy an extra knob to increase action diversity. We include both because we found **DSRL** to perform poorly in some tasks, where **TMRL** performs better on due to increased action diversity. We compare against:

- **DSRL** (Wagenmaker et al., 2025b) and **TMRL** (Hong et al., 2026): the base off-policy RL algorithms without any verifier, isolating the effect of **VETO**'s action filtering.
- **Q-BoN**: the same top-K filtering as **VETO**, but ranking candidates with the base algorithm's TD critic, similar to Q-based best-of-N methods (Mark et al., 2024; Sun & Song, 2026; Patil et al., 2026; Giridhar et al., 2026; Dong et al., 2026). We run this on top of **DSRL**.
- **Newt** (Hansen et al., 2026): a model-based method based on TD-MPC2 (Hansen et al., 2024) that trains a latent world model and policy on the offline data and fine-tunes both online, selecting actions via MPC search through latent Q-predictions.
- **FOWM** (Feng et al., 2023): extends TD-MPC2 with a Q-ensemble uncertainty penalty for more conservative offline-to-online fine-tuning and MPC Q-based action selection.
- **DYNA** (Sutton, 1990; Janner et al., 2019): uses **VETO**'s world model to generate synthetic chunk-level transitions from real buffer states for critic training, rather than to filter actions. It runs on top of DSRL (Robomimic) and TMRL (LIBERO), is implemented using MBPO (Janner et al., 2019), and isolates whether the world model should generate trajectories or guide data collection.

Additional details in Section D.

Improvement with Iterative BC. To demonstrate the applicability of **VETO** to other types of RL adaptation algorithms, we also conduct iterative behavioral cloning (BC) experiments in which a

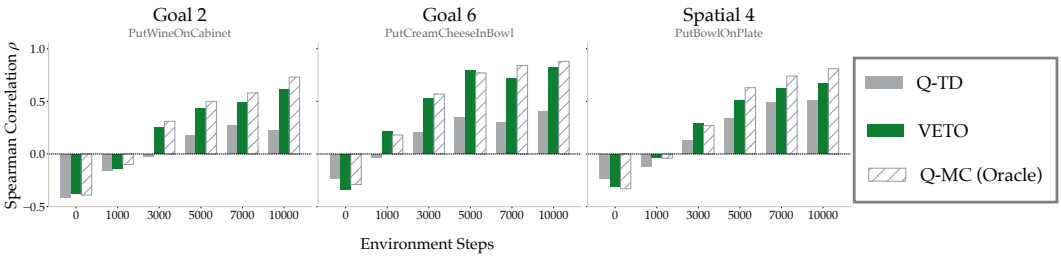


Figure 6: **Ranking accuracy on LIBERO.** Spearman rank correlation between each model's scores and MC return estimates (50 simulator-reset rollouts per candidate) for 5 action chunks, averaged over held-out states. VETO ranks candidates more accurately than Q-TD from 3000 environment steps onward and comparably to the oracle Q-MC.

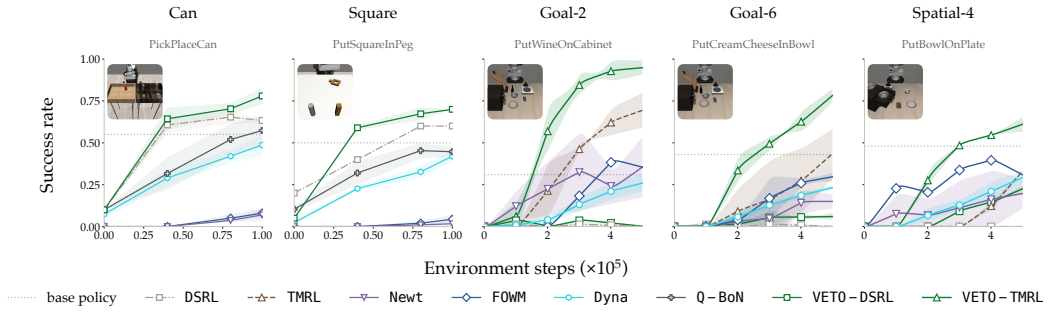


Figure 7: **Online RL experiments.** Across the board, VETO improves the sample-efficiency of off-policy RL algorithms (DSRL and TMRL) in Robosuite and LIBERO tasks. Using VETO's latent world model to rank actions outperforms TD-learned Q-BoN, and VETO also outperforms the model-based baselines (Newt, FOWM). We do not run TMRL in Robosuite since DSRL already performs well there. Shaded region denotes standard deviation across 5 seeds with 200 evaluations.

fixed, frozen policy is used for multiple episodes of data collection before being fine-tuned on successful trajectories. We then repeat this step for all future iterations. We found this approach more stable to deploy on real hardware; it represents a special case of off-policy RL: reward-weighted regression (RWR; Peters & Schaal, 2007) with weights corresponding to binary episode success.

4.1 (Q1): Ranking fidelity of VETO

We first compare how well VETO *ranks* candidate actions during off-policy RL against Q-TD, the underlying algorithm's TD-trained Q-function, and Q-MC, an oracle regressed onto Monte-Carlo returns from simulator resets at every buffer state-action pair, i.e., the best supervised regression can do given the buffer's coverage. We run TMRL on 3 LIBERO tasks; after an initial warmup, we sample 5 policy chunks at held-out states and report the Spearman ρ between each model's ranking and ground-truth Monte-Carlo returns from simulator resets (Figure 6). VETO improves much faster than Q-TD, with 3× higher ρ at just 3000 steps, and closely tracks Q-MC throughout. Even Q-MC stays below $\rho = 1$, since every model must generalize to unseen states and chunks (details in Section D.1).

4.2 (Q2): VETO improves sample efficiency across RL algorithms

We next evaluate whether VETO improves sample efficiency in off-policy RL and iterative BC. For both Robomimic and LIBERO, the world model is trained on a broader and more diverse data distribution than the base policy, which is trained only on expert demonstrations. We provide the exact dataset composition and augmentation details in Appendix D.

Off-policy RL. Our off-policy simulation results across both Robosuite and LIBERO are shown in Figure 7. In Robosuite, VETO-DSRL outperforms DSRL. Q-BoN, implemented on top of DSRL, results in worse sample efficiency than DSRL itself, signifying that over-optimizing against a TD-learned Q-

function can hurt performance, especially early during adaptation when its action ranking accuracy is poor (shown in Section 4.1). The model-based baselines perform poorly in both Robosuite tasks.

In LIBERO, we find that **VETO**-TMRL achieves significantly better performance than the base **TMRL** policy, especially early during adaptation. **VETO**-DSRL outperforms DSRL, but both perform poorly due to the general inability of DSRL to sample sufficiently diverse action chunks beyond the base policy in these LIBERO tasks overall. Q-BoN again performs worse than DSRL on its own due to a combination of early critic overoptimization and compared to model-based methods such as **Newt** and **FOWM**, **FOWM** sometimes performs well initially due to offline-to-online conservatism, but it does not improve further compared to **VETO**-TMRL. DYNA, which trains the critic on synthetic transitions from the same world model, underperforms both **VETO** and its base algorithm (DSRL or **TMRL**). Since the two share the world model, base algorithm, and pretrained policy, this isolates the cost of injecting model error into the Bellman backup.

Iterative BC. Next, we perform iterative BC in LIBERO, with results in Figure 8. We specifically choose a task where the pre-trained policy succeeds in very few initial verifier-free evaluations, leaving substantial room to test whether better collection improves the learned policy. All methods initialize from the same flow policy pretrained with input noising (Hong et al., 2026). Each round starts with collecting warmup episodes (shaded region) after which the policy is fine-tuned from the base policy on all successful episodes so far. Then, all subsequent iterations collect 50 additional episodes before repeating. We compare three methods: **Unguided**, **VETO**, and Q-BoN, which trains a critic with TD learning on the same data and is used only for ranking. At the plotted endpoints, **VETO** has the highest mean success on `Object-9`, but also reaches that success rate far earlier in training. For more details, refer to Section D.2.

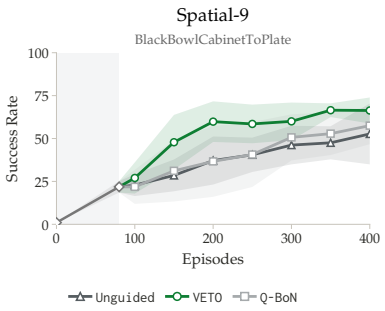


Figure 8: **Iterative BC.** **VETO** outperforms Q-BoN and **Unguided** significantly earlier during adaptation.

Across six tasks in two simulation suites, **VETO** improves sample efficiency with both off-policy RL and iterative BC. It does so while outperforming baselines that use TD-learned Q-functions to rank actions, while also outperforming a trajectory-generation baseline. Next, we demonstrate **VETO** improving adaptation on real hardware.

4.3 (Q3): Real-world Adaptation

In the real world we adapt with iterative BC on two platforms: a DROID (Khazatsky et al., 2025) Franka 3 single-arm setup and a bimanual YAM setup. Because the pre-trained models are large (3B for DROID, 6B for YAM), rather than fine-tuning them we perform BC on the flow *noise* ω that generated each executed action chunk in successful trajectories (DSBC; Tang et al., 2026), training a policy $\pi(\omega \mid \mathbf{s})$ that is fast to train and evaluate between iterations.

Bimanual YAM. Starting from a pre-trained Flex-π policy (Yan et al., 2026), we evaluate a dexterous, long-horizon task: `unzip deformable pencil-case, put pen inside, and zip the case` (Figure 14). After 10 warmup episodes, we collect 60 more with and without **VETO**, using Robometer rewards (Liang et al., 2026) fine-tuned on the released Flex-π YAM training data.

DROID. Starting from π_0 (Black et al., 2024), the task is to pick up a bowl from a dish rack and place it on the table. Due to the rack placement, π_0 often grasps nearby objects or the rack itself instead of the bowl, achieving a 20% base success rate. After 10 warmup episodes with the base policy, we collect 20 more with and without **VETO** filtering, using rewards from an in-submission reward model (Anonymous, 2026) built on `GPT-6-Luna`.

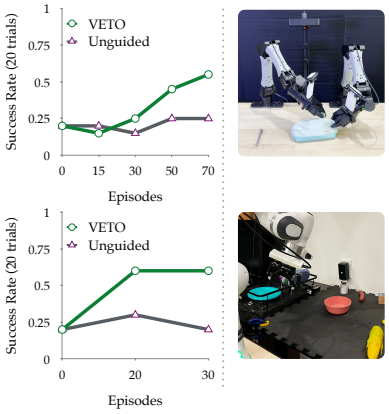


Figure 9: **VETO on real-robot tasks.** **VETO** with iterative DSBC on π_0 and Flexi-π improves against no action verification.

In Figure 9, **VETO** drastically improves success in both experiments over the base Flex-π and π_0 policy success rates. Notably, collecting data with **VETO** yields an average $\sim 2.5\times$ success rate increase over iterative BC without **VETO** (unguided); without action verification, fewer successful trajectories are gathered for BC to train with.

4.4 (Q4): Analysis and Ablations

Finally, we study which design choices are critical to **VETO** by ablating the dynamics modeling objective and investigating how K in top-K sampling affects learning stability. We also analyze how incorporating Q-values into **VETO** in certain ways can improve or degrade its performance. We perform additional analysis in Section E.

Does explicit dynamics prediction improve action ranking? To test whether return prediction alone explains **VETO**'s advantage over `Q-TD`, we compare it on the PointMaze setup of Figure 4 with two dynamics-free baselines: a regressor onto the same discounted H-step return, and a model predicting each reward in the chunk directly from the initial state and action chunk. After 256 transitions, **VETO** reaches a Spearman correlation of 0.82, versus 0.07 and -0.03 for the two baselines (Figure 10). Predicting the same return, or even its individual rewards, does not reproduce **VETO**'s early ranking accuracy. Instead, explicit dynamics modeling is necessary.

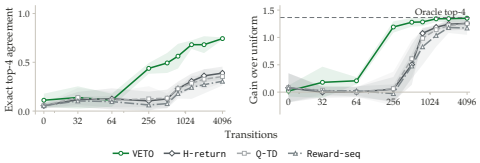
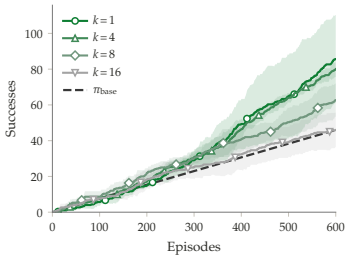


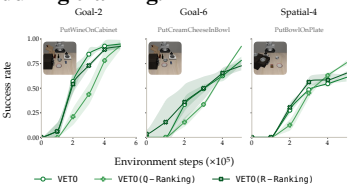
Figure 10: **Explicit dynamics improves early action ranking in PointMaze.** Left: agreement with simulator-measured chunk returns. Right: return gain from selecting the predicted top four of sixteen candidates.

How does the choice of K affect VETO collection? We study K, the number of top-ranked chunks retained from $N = 16$ candidates (Figure 11a). To isolate when filtering helps, we freeze the base policy and train only the verifier; $K = N = 16$ retains all candidates and serves as the unfiltered baseline. Early on, all K collect successes at similar rates. As the verifier trains, $K = 4$ and $K = 8$ separate from the baseline. $K = 1$ also performs well but with much higher across-seed variance (24.6 vs. 4.6 for $K = 4$): with no exploration among top candidates, verifier errors can impact the executed chunk.



(a) **Effect of top-K in VETO as measured by cumulative successes during training.**

Does including Q-values in VETO improve performance? We evaluate two variants that incorporate Q-values into action-chunk ranking (Figure 11b): (a) `R-ranking`, which uses the reward model to select the top-K chunks and the Q-function to choose the final action, and (b) `Q-ranking`, which reverses this order. Early in adaptation, **VETO** is the most sample-efficient. As the critic improves, `Q-ranking` catches up and eventually surpasses **VETO**, while `R-ranking` performs similarly to **VETO**.



(b) **Effect of using $Q-$values in VETO ranking.**

5 Conclusion and Limitations

We proposed **VETO**, a policy-agnostic method to improve the sample-efficiency of off-policy RL. Our key insight is that supervised learning propagates training signal far faster than TD learning, enabling us to train a supervised latent world model to filter out poor actions before the underlying RL algorithm can change the policy's behavior. This helps us reduce *credit-assignment lag* that plagues standard RL. However, **VETO** still carries limitations. While world model inference and training compute is minimal, **VETO** trades off additional policy sampling cost for better sample-efficiency. Furthermore, **VETO** relies on access to rewards; recent progress in general-purpose reward models has made this assumption more practical, but improvements in VLM reward capabilities are likely needed for **VETO** to improve on extremely long-horizon tasks.

AI USE STATEMENT

In this work, we used generative AI tools to aid paper writing by polishing, research on theory, proving the theory, implementing iterative BC, and our Dyna baseline. We have not used generative AI tools for drafting entire sections, and generating synthetic datasets is not applicable. We have reviewed all AI-assisted work by verifying the code generated for correctness by at least 1 author, verifying the proof by at least 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made an effort to ensure results are reproducible. We list the full algorithm in Algorithm 2, implementation details in Section A, hyperparameters in Table 2 and Table 1. We also clearly explain assumptions for theoretical analysis in Section 3.4, along with providing empirical evidence for this analysis in Figure 4. Simulated experiments are run across 5 seeds and upon acceptance, we plan to open-source our code. For real-world experiments, we use open-weight, pre-trained base policies that we do not fine-tune ourselves and we reference the datasets and works these models and their data come from.

REFERENCES

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025. 22

Anonymous. Title omitted for double-blind review. *Manuscript submitted for publication*, 2026. Blinded for peer review. 6, 8, 17, 21

Maria Attarian, Ian Vyse, Jasper Gerigk, Evgenii Opryshko, Yifan Ruan, Anas Almasri, Sumeet Singh, Yilun Du, Igor Gilitschenski, and Claas Voelcker. Update-free on-policy steering via verifiers. *arXiv preprint arXiv:2603.10282*, 2026. 3

Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, 1995. 1

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 8, 21

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, 2018. 2

Perry Dong, Yueru Jia, Chelsea Finn, and Dorsa Sadigh. Q-learning with world models. *arXiv preprint arXiv:2608.17163*, 2026. 3, 6, 18

Maximilian Du and Shuran Song. Dynaguide: Steering diffusion policies with active dynamic guidance. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025. 3

Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Fine-tuning offline world models in the real world. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, 2023. 6, 18

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=PwaUsAk4i6. 3

Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot manipulation policies. *IEEE Robotics and Automation Letters (RA-L)*, 2026. 1

Varun Giridhar, Anant Khandelwal, Jeremy A Collins, Ignat Georgiev, and Animesh Garg. Beyond imitation: Self-improving robot policies via off-policy q-planning. *arXiv preprint arXiv:2608.21204*, 2026. 3, 6, 18

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. *arXiv preprint arXiv:2602.12063*, 2026a. 1, 3

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation, 2026b. URL https://arxiv.org/abs/2510.10125. 3

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018. 2

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 3, 4, 17

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a. 1, 2

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b. 2

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. 3

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025. 3

Tyler Han, Yanda Bao, Bhaumik Mehta, Gabriel Guo, Anubhav Vishwakarma, Emily Kang, Sanghun Jung, Rosario Scalise, Jason Zhou, Bryan Xu, et al. Model predictive adversarial imitation learning for planning from observation. *arXiv preprint arXiv:2507.21533*, 2025. 3

Tyler Han, Bat Nemekhbold, Siyang Shen, Rohan Baijal, Richard Ebock, Harine Ravichandiran, Sanghun Jung, Kevin Huang, and Byron Boots. Online world modeling enables real-world inverse reinforcement learning from observation. In *Proceedings of the 10th Conference on Robot Learning (CoRL)*, 2026. 3

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022. 3

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024. 3, 5, 6, 18

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control, 2026. 3, 5, 6, 18

Matthew M. Hong, Jesse Zhang, Anusha Nagabandi, and Abhishek Gupta. Tmrl: Diffusion timestep-modulated pre-training enables exploration for efficient policy fine-tuning. In *Robotics: Science and Systems (RSS)*, 2026. 1, 3, 6, 8, 17

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. *arXiv:2511.14759*, 2025. 1

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=qN5hmLkBtC. 1

Arnav Kumar Jain, Yilin Wu, Jesse Farebrother, Gokul Swamy, and Andrea Bajcsy. Weaver, better, faster, longer: An effective world model for robotic manipulation. *arXiv preprint arXiv:2606.13672*, 2026. 3

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019. 1, 6, 18

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 2018. 2

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945. 6, 8, 21

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. DR3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations*, 2022. 1

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, Proceedings of Machine Learning Research, pp. 3200–3217, 2025. 3

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling verification can be more effective than scaling policy learning for vision-language-action alignment. *arXiv preprint arXiv:2602.12281*, 2026. 3

Patrick Lancaster, Nicklas Hansen, Aravind Rajeswaran, and Vikash Kumar. Modem-v2: Visuomotor world models for real-world robot manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2024. 3

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu,

Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. In *Robotics: Science and Systems 2026*, 2026. 6, 8, 20, 22

Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025. 3

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng YAN, Maoqing Yao, and Guanghui Ren. Genie envisioner: A unified world foundation platform for robotic manipulation. In *International Conference on Learning Representations*, 2026. 3

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747. 6

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 2023. 6

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. *arXiv preprint arXiv:2602.06508*, 2026. 3

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025. 1

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. In *arXiv preprint arXiv:2108.03298*, 2021. 6

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024. 3, 6, 18

Suvir Mirchandani, Suneel Belkhale, Joey Hejna, Evelyn Choi, Md Sazzad Islam, and Dorsa Sadigh. So you think you can scale up autonomous robot data collection? In *8th Annual Conference on Robot Learning*, 2024. 1

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)*, 2018. 2

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Proceedings of the Conference on Robot Learning*, 2020. 3

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *Conference on Robot Learning (CoRL)*, 2024. 3

Cyrus Neary, Omar G Younis, Artur Kuramshin, Ozgur Aslan, and Glen Berseth. Improving pre-trained vision-language-action policies with model-based search. *arXiv preprint arXiv:2508.12211*, 2025. 3

NVIDIA. Cosmos-predict2: World simulation model for physical ai, 2025. URL https://github.com/nvidia-cosmos/cosmos-predict2. 3

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 5

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025. 2, 5

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, Zeynep Temel, Oliver Kroemer, Sergey Levine, Abhishek Gupta, Hongkai Dai, Paarth Shah, and Max Simchowitz. OGPO: Sample efficient full-finetuning of generative control policies. In *International Conference on Machine Learning*, 2026. 3, 6, 18

Quanquan Peng, Yutong Liang, Rui Yan, Nicklas Hansen, and Xiaolong Wang. Fact: Failure-aware causal training for world-action models. *arXiv preprint arXiv:2608.10232*, 2026. 3

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007. 7

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. *arXiv preprint arXiv:2602.02454*, 2026. 3

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive rl finetuning. In *International Conference on Machine Learning*, 2026. 3, 6, 18

Richard S Sutton. Integrated modeling and control based on reinforcement learning and dynamic programming. In R.P. Lippmann, J. Moody, and D. Touretzky (eds.), *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann, 1990. 1, 2, 6, 18

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering. 2026. 8, 21

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997. doi: 10.1109/9.580874. 1

Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning. *arXiv preprint arXiv:2512.16911*, 2025a. 1

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. *Conference on Robot Learning*, 2025b. 3, 4, 6, 17

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. 3, 22

Jie Wang, Matthew Leonard, Kostas Daniilidis, Dinesh Jayaraman, and Edward Hu. Evaluating pi0 in the wild: Strengths, problems, and the future of generalist robot policies, 2025. URL https://penn-pal-lab.github.io/Pi0-Experiment-in-the-Wild/. 1

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026a. 3

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. In *Robotics: Science and Systems (RSS)*, 2026b. 3

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pp. 2226–2240. PMLR, 2023. 3

Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia Pérez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. *2026 IEEE International Conference on Robotics Automation (ICRA)*, 2026. 3

Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π: A multi-stream world-action model with compute flexibility. *arXiv preprint arXiv:2608.10860*, 2026. 8, 22

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. In *Robotics: Science and Systems (RSS)*, 2026. 3

Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2024. 1

Patrick Yin, Tyler Westenbroek, Simran Bagaria, Kevin Huang, Ching-An Cheng, Andrey Kolobov, and Abhishek Gupta. Rapidly adapting policies to the real-world via simulation-guided fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2025. 1

Tenny Yin, Zhiting Mei, Zhonghe Zheng, Miyu Yamane, David Wang, Jade Sceats, Samuel M Bateman, Lihan Zha, Apurva Badithela, Ola Shorinwa, et al. Playworld: Learning robot world models from autonomous play. *arXiv preprint arXiv:2603.09030*, 2026. 1, 3

Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. Cautious adaptation for reinforcement learning in safety-critical settings. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 3

Jesse Zhang, Minho Heo, Zuxin Liu, Erdem Biyik, Joseph J Lim, Yao Liu, and Rasool Fakoor. EXTRACT: Efficient policy learning by extracting transferrable robot skills from offline data. In *Conference on Robot Learning*, 2024a. 1

Jesse Zhang, Karl Pertsch, Jiahui Zhang, and Joseph J. Lim. Sprint: Scalable policy pre-training via language instruction relabeling. In *International Conference on Robotics and Automation (ICRA)*, 2024b. 1

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. 5

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024. 3

Algorithm 2 VETO: complete adaptation loop

Require: Base policy π_{base}, current RL policy π_{RL}, pretrained world model $\texttt{WM}_\phi = (f_\phi, R_\phi)$, replay buffer $\mathcal{B}$, candidate count N, retained count K, chunk horizon H, warmup length T_{warm}, adaptation rollouts M

for rollout $m = 1, \dots, M$ **do**
  Reset environment and observe initial state $\mathbf{s}_0$
  for each decision time $t = 0, H, 2H, \dots$ until episode ends **do**
    if $t < T_{\text{warm}}$ **then**
      Sample $\mathbf{a}^\star \sim \pi_{\text{RL}}(\cdot \mid \mathbf{s}_t)$ ▷ unfiltered warmup
    else
      $\mathbf{z}_0 \leftarrow \texttt{VisionEncoder}(\mathbf{s}_t^{\text{img}})$; $\mathbf{g} \leftarrow \texttt{TextEncoder}(\mathbf{s}^{\text{lang}})$
      Sample N chunks $\{\mathbf{a}^{(i)}\}_{i=1}^{N} \sim \pi_{\text{RL}}(\cdot \mid \mathbf{s}_t)$
      for $i = 1, \dots, N$ **do**
        $\mathbf{z}_0^{(i)} \leftarrow \mathbf{z}_0$
        for $h = 0, \dots, H-1$ **do**
          $\mathbf{z}_{h+1}^{(i)} \leftarrow f(\mathbf{z}_h^{(i)}, \mathbf{a}_h^{(i)})$
        $\hat{R}^{(i)} \leftarrow \sum_{h=1}^{H} \gamma^{h-1} R(\mathbf{z}_h^{(i)}, \mathbf{g})$ ▷ predicted return
      $\mathcal{I}_K \leftarrow \text{topK}_{i \in \{1,\dots,N\}} \hat{R}^{(i)}$; sample $I \sim \text{Uniform}(\mathcal{I}_K)$
      $\mathbf{a}^\star \leftarrow \mathbf{a}^{(I)}$
    Execute $\mathbf{a}^\star$ in the environment; store resulting real transitions in $\mathcal{B}$
  Obtain reward labels r for new real data from environment or video-language-input reward model and add labels to $\mathcal{B}$
  Update π_{RL} using the chosen off-policy improvement algorithm and real data in $\mathcal{B}$
  Fine-tune $\texttt{WM}_\phi$ on labeled real data in $\mathcal{B}$ using Equation (3)

A Implementation Details

A.1 VETO

Decoupling the verifier from policy learning. A central design principle of VETO is that the world model and the RL learner are trained independently and interact only at action selection. We enforce this separation along three axes.

- **Objectives.** $\texttt{WM}_\phi$ is trained exclusively with the supervised latent-dynamics and reward losses in Eq. equation 3. It has no Q-head and no TD term, and RL gradients never flow into it. Conversely, world-model predictions never enter the actor or critic losses: no synthetic transitions are added to the replay buffer, and critic targets are computed only from real rewards and real next states.
- **Data.** During pre-training, the world model and the base policy use different data. The base policy is trained only on expert demonstrations. The world model is trained on a broader distribution: the multi-human (`mh`) datasets for Robomimic, and all LIBERO-{`goal`, `spatial`, `10`} tasks augmented with 20% noise-perturbed failure trajectories for LIBERO. Online, both learners consume the same replay buffer $\mathcal{B}$ of real transitions, labeled with environment rewards (Robomimic) or Robometer-derived progress rewards (LIBERO, real world).
- **Optimization.** The world model and the RL learner use separate networks, optimizers, and update schedules. Image and language encoders (DINO-v2 CLS and Qwen3-Embedding-0.6B) are frozen, so only the dynamics MLP f_ϕ and reward MLP R_ϕ are updated online ($\sim$4.5–12M trainable parameters depending on the environment).

Action selection. At each decision step, VETO draws N candidate chunks from π_{RL}, rolls f_ϕ over each full proposed chunk from the encoded current observation, and scores it by the discounted sum of predicted rewards. One chunk is sampled uniformly from the top K. When only a prefix of each chunk is executed before replanning, we still score the full proposed chunk. During an initial

warmup of T_{warm} steps, actions are sampled from π_{RL} without filtering to preserve action diversity and to collect initial data for the verifier.

Hyperparameters. Table 1 lists VETO's hyperparameters. Hyperparameters of the underlying RL algorithms (DSRL, TMRL) follow their original implementations unless stated otherwise.

Table 1: VETO hyperparameters.

	Robomimic	LIBERO (RL)	DROID	YAM
Verification				
Candidates N	16	16	16	16
Retained K	4	4	4	4
Scored chunk length H	10	10	32	32
Executed actions per chunk	4	5	10	32
Discount γ	0.99	0.99	0.99	0.99
Warmup	10 eps.	10 eps.	20 / 100 eps.	10 eps.
World model				
Image encoder (frozen)	DINO-v2 CLS ([768/1024]-d)			
Language encoder (frozen)	Qwen3-Embedding-0.6B			
Latent dim	768	768	1024	1024
Trainable parameters	4.5M	4.5M	12M	12M
Loss horizon	3	3	3	3
Per-step loss weight λ	1	1	1	1
Optimizer / learning rate	Adam, 3e-4			
Batch size	256	256	1024	1024
Reward source	Env. (dense)	Robometer	(Anonymous, 2026)	Robometer

A.2 Baselines

In this section, we describe the off-policy RL baselines used in Section 4. Unless otherwise stated, all baselines use the same pre-trained flow-matching base policy π_{base}, the same observation space (image observations and robot proprioception), the same reward source (simulator dense rewards for Robomimic; Robometer-derived progress rewards for LIBERO), and the same action-chunk length $H = 4$ (for Robomimic) and $H = 5$ (for LIBERO) as VETO. Where a baseline requires offline pre-training, it is pre-trained on the same data as VETO's world model (Appendix D). All methods are run for 5 seeds and evaluated over 200 episodes at every 10k environment steps. Hyperparameters for all baselines are listed in Table 2.

DSRL (Wagenmaker et al., 2025b). DSRL freezes the pre-trained flow policy $\pi_{\text{base}}(\mathbf{a} \mid \mathbf{s}, \omega)$ and trains a noise policy $\pi_\theta(\omega \mid \mathbf{s})$ using SAC (Haarnoja et al., 2018). Given the current state $\mathbf{s}$, the noise policy predicts the initial latent noise ω, which is then decoded by π_{base} into an action chunk. Both the actor and critic operate on observation features, such as frozen DINOv2 embeddings concatenated with proprioceptive inputs, while the critic $Q(\mathbf{s}, \omega)$ is defined directly over the latent noise space.
Training begins with 2,000 warmup environment steps consisting of unfiltered rollouts from the base policy, after which RL updates are enabled. We use an update-to-data ratio of 20 and a critic ensemble of five target networks. For target computation, two critics are sampled uniformly at random from the ensemble and their minimum value is used. DSRL serves as the base algorithm for both VETO-DSRL and Q-BoN; all hyperparameters shared across the three methods are kept identical.

TMRL (Hong et al., 2026). TMRL follows the same noise-space steering procedure as DSRL, but steers a flow policy pre-trained with diffusion-timestep-modulated input noising. This pre-training broadens the policy's action distribution and gives the RL policy an additional noise-level control, which increases action diversity on tasks where DSRL fails to explore beyond the base policy. We use $[0, 1]$ noise-level range, with action noise perturbation set to $[-0.05, 0.05]$, and otherwise match the DSRL SAC hyperparameters. TMRL is the base algorithm for VETO-TMRL. We do not run TMRL on Robomimic since DSRL already performs well there.

Q-BoN. Q-BoN isolates the effect of the ranking model by replacing VETO's world-model verifier with the base algorithm's TD-trained critic, following Q-based best-of-N methods (Mark et al., 2024; Sun & Song, 2026; Patil et al., 2026; Giridhar et al., 2026; Dong et al., 2026). At each decision step, we sample $N = 16$ candidate chunks from π_{RL}, score each with the DSRL critic $Q(\mathbf{s}, \omega^{(i)})$, and execute one chunk sampled uniformly from the top $K = 5$. The same warmup period without filtering is used as in VETO. The critic is trained exactly as in DSRL; no additional ranking-specific training is performed.

Newt (Hansen et al., 2026). Newt is a model-based method built on TD-MPC2 (Hansen et al., 2024) that jointly learns a latent world model (encoder, latent dynamics, reward head), a TD-learned Q-function, and a policy prior. For a fair comparison, we pre-train the world model and the policy head (with behavior cloning) with the same set of data that we use to train WM_ϕ and π_{base}. However, we do not train the Q-functions of newt since we found that due to significant distributional shift, from offline-to-online RL, Q-value trained from scratch performed better. For the remaining hyperparameters we refer the reader to the original paper (Hansen et al., 2026).

FOWM (Feng et al., 2023). FOWM extends TD-MPC2 for offline-to-online fine-tuning by penalizing planning objectives with the epistemic uncertainty of a Q-ensemble, making action selection more conservative on out-of-distribution actions. We use an ensemble of 5 Q-functions and an uncertainty penalty coefficient of λ, and otherwise use the same pre-training data, encoders, planner settings, and online fine-tuning schedule as Newt. We experimented with $\lambda = \{1, 10\}$ and consistently found $\lambda = 1$ to perform better, hence we report our final set of results with $\lambda = 1$.

DYNA We implement an MBPO-style Dyna baseline (Sutton, 1990; Janner et al., 2019) that uses the same world model WM_ϕ as **VETO**, with identical architecture, pretraining data, and online refitting schedule. The model generates training data instead of filtering actions. After each environment step, we sample 256 real states from the replay buffer $\mathcal{B}$, encode them with the frozen DINO-v2 encoder, and sample noise $\omega \sim \pi_\theta(\cdot \mid \mathbf{z})$ from the current noise policy. Each ω is decoded through the frozen base policy into an action chunk, and we roll f_ϕ over that chunk, summing the discounted predicted rewards $\hat{R} = \sum_{t=1}^{H} \gamma^{t-1} R_\phi(\mathbf{z}_t, \mathbf{g})$. This yields a synthetic chunk-level transition $(\mathbf{z}, \omega, \hat{R}, \mathbf{z}')$, which is stored in a FIFO model buffer $\mathcal{B}_{\text{model}}$ so that only rollouts from recent policies are kept. Rollouts are limited to a single chunk to limit compounding model error, and synthetic transitions are treated as non-terminal. Each SAC batch draws a fraction 0.95 of its samples from $\mathcal{B}_{\text{model}}$ and the remainder from $\mathcal{B}$ (following (Janner et al., 2019)). Actor and critic inputs use the same DINO-v2 latent as the world model, so that imagined next states can be consumed directly. Actions are executed from π_{RL} without filtering, and the update-to-data ratio and total gradient steps match **VETO**'s. We run **DYNA** on top of DSRL for Robomimic and **TMRL** for LIBERO.

Table 2: Hyperparameters for the off-policy RL baselines. Shared values are listed once.

Hyperparameter	DSRL	TMRL	Q-BoN	Newt	FOWM
Actor learning rate	3e-4	3e-4	3e-4	3e-4	3e-4
Critic learning rate	3e-4	3e-4	3e-4	3e-4	3e-4
Batch size	256	256	256	1024	1024
Discount γ	0.99	0.99	0.99	$[0.95, 0.995]$	$[0.95, 0.995]$
Update-to-data ratio	20	20	20	1	1
Target update rate τ	0.005	0.005	0.005	0.01	0.01
Critic ensemble size	5	5	5	5	5
Warmup steps	2000	2000	2000	5000	5000
Candidates N / top-K	–	–	16/4	–	–
Planning horizon	–	–	–	3	3
MPPI samples / elites / iters	–	–	–	512 / 64 / 6	512 / 64 / 6
Uncertainty penalty	–	–	–	–	1.0
Offline pre-training steps	–	–	–	100000	100000

B THEORETICAL ANALYSIS

Gain and regret. For brevity, write $\pi = \pi_{\mathrm{RL}}(\cdot \mid \mathbf{s})$ and define the H-step truncated return $q(\mathbf{a}) = Q_H^{\pi}(\mathbf{s}, \mathbf{a}) := \mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r_t \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}]$. Draw N candidates $\mathbf{a}^{(i)} \sim \pi$ independently. Let S^* index the true top-K and S be $\mathtt{WM}_\phi$'s selected set, assuming that **VETO** selects uniformly from the top-K. Define the oracle gain over the candidate mean, $G := \frac{1}{K}\sum_{i\in S^*} q(\mathbf{a}^{(i)}) - \frac{1}{N}\sum_{i=1}^{N} q(\mathbf{a}^{(i)})$, and the verifier's selection regret, $R := \frac{1}{K}\sum_{i\in S^*} q(\mathbf{a}^{(i)}) - \frac{1}{K}\sum_{i\in S} q(\mathbf{a}^{(i)})$. Let $\mathbf{a}^\star$ be a chosen action drawn uniformly from **VETO**'s selected set S.

Proposition 1 (Local value decomposition)**.** *If $q(\mathbf{a})$ is integrable, then*

$$\mathbb{E}[q(\mathbf{a}^\star)] = \mathbb{E}_{\mathbf{a}\sim\pi}[q(\mathbf{a})] + \mathbb{E}[G] - \mathbb{E}[R],$$

where expectations include candidate draws, verifier randomness, and the final uniform selection. In expectation, ***filtering improves H-step action return when oracle gain exceeds selection regret****.*

Proof sketch. Conditioned on the candidates and selected set, uniform execution has mean $K^{-1}\sum_{i\in S} q(\mathbf{a}^{(i)}) = N^{-1}\sum_i q(\mathbf{a}^{(i)}) + G - R$. Taking expectations and using i.i.d. proposals gives the identity; the true top-K maximizes the value sum, so $G, R \geq 0$. Proof in Section C.1. □

Worst-case guarantee. The gain–regret identity compares the values with regret. We also provide a pessimistic worst-case return bound:

Suppose $q(\mathbf{a}) \in [q_{\min}, q_{\max}]$ for all proposed chunks, and let $R_H := q_{\max} - q_{\min}$. Assume that, with probability at least $1-\delta$ over candidate draws and verifier randomness, the verifier's top-K set agrees with the true top-K set under q, using consistent tie-breaking.

Proposition 2 (Local return guarantee)**.** *Under this ranking-fidelity condition,*

$$\mathbb{E}[q(\mathbf{a}^\star)] \geq \mathbb{E}_{\mathbf{a}\sim\pi}[q(\mathbf{a})] - \delta R_H.$$

See proof in Section C.2.

C PROOFS FOR THE THEORETICAL ANALYSIS

C.1 PROOF OF THE LOCAL VALUE DECOMPOSITION

Proof of Proposition 1. Condition on the candidate draw $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)}$ and the verifier's selected set S. Since execution is uniform over S,

$$\mathbb{E}\Big[q(\mathbf{a}^\star) \mid \mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(N)}, S\Big] = \frac{1}{K}\sum_{i\in S} q(\mathbf{a}^{(i)}).$$

By the definitions of G and R, this is exactly

$$\frac{1}{N}\sum_{i=1}^{N} q(\mathbf{a}^{(i)}) + G - R.$$

Taking expectations gives the claimed identity because the candidates are i.i.d. from π, and hence

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} q(\mathbf{a}^{(i)})\right] = \mathbb{E}_{\mathbf{a}\sim\pi}[q(\mathbf{a})].$$

The true top-K set maximizes the sum of q over all subsets of size K. Therefore its mean value is at least both the mean over all N candidates and the mean over the verifier's selected set, proving $G \geq 0$ and $R \geq 0$. □

C.2 Conservative local return guarantee

Proof of Proposition 2. Let F be the event that the verifier selects the true top-K set, and let $\bar{q}_S$ and $\bar{q}_N$ denote, respectively, the mean true value in the selected set and in all N candidates. On F, the verifier's set is the oracle set, so $\bar{q}_S \geq \bar{q}_N$. On F^c, both means lie in $[q_{\min}, q_{\max}]$, and thus $\bar{q}_S - \bar{q}_N \geq -R_H$. Since $\Pr(F^c) \leq \delta$, averaging these inequalities yields

$$\mathbb{E}[\bar{q}_S - \bar{q}_N] \geq -\delta R_H.$$

Uniform execution from the selected set gives $\mathbb{E}[q(\mathbf{a}^\star)] = \mathbb{E}[\bar{q}_S]$. Also, the candidates are i.i.d. from π, so $\mathbb{E}[\bar{q}_N] = \mathbb{E}_{\mathbf{a}\sim\pi}[q(\mathbf{a})]$. Substitution proves the stated bound. □

D Experiment Details

In this section we describe the general details of our experimental setup. For Robomimic, the world model is trained on the combined suboptimal multi-human (`mh`) datasets from the `Can`, `Lift`, and `Square` tasks. The base policy, by contrast, is trained on expert demonstrations for the specific task being evaluated.

For our LIBERO experiments, we pre-train a multi-task world model on all tasks from the `LIBERO-{goal, spatial, 10}` datasets. We append an augmented dataset with 20% consisting of synthetically generated failure trajectories, obtained by injecting Gaussian noise into the actions following the data augmentation procedure of (Liang et al., 2026). The flow-matching base policies are trained only on expert demonstrations from the `LIBERO-{goal, spatial}` datasets. We intentionally use different data sources for the world model and policy, avoiding any assumption of matched training distributions and better reflecting realistic adaptation settings.

D.1 Ranking

This experiment evaluates **VETO**'s ability to rank action chunks against `Q-TD`, which ranks actions using a TD-learned Q-function, and `Q-MC`, an oracle trained on Monte Carlo estimates of Q^π. We evaluate on **TMRL** runs from three LIBERO tasks. To obtain training targets for `Q-MC`, for each (s, a) pair in the replay buffer, we execute a from s and roll out 50 trajectories under the current policy π until termination. Their average return serves as the ground-truth estimate of $Q^\pi(s, a)$. We evaluate each of the three models on held out transitions collected by rollout out policy at the current timestep.

D.2 Iterative LIBERO

We evaluate on LIBERO `Spatial-9` which consists of the task: `pick up the black bowl on the wooden cabinet and place it on the plate`. All methods start from the same pre-trained policy and share 80 unguided episodes that serve as the buffer dataset for the first round of offline training. We use the version of **VETO** without optional offline world-model pretraining. Its world model and the **Q-BoN** critic are initially fit using the same 80 warmup episodes before either verifier guides collection. We fix the context-noise timestep at its minimum and sample action noise to generate proposals. `Unguided` executes one proposal, while **VETO** and **Q-BoN** sample $N = 16$ chunks and execute one uniformly from the top $K = 4$ according to their respective verifiers. Each proposal contains 32 actions; the robot executes the first 10 before replanning with a maximum episode length of 200 environment steps.

After warmup, each method collects its own data. The **VETO** world model and the **Q-BoN** critic continue training on their respective collected trajectories, including failures, with Robometer rewards. At each checkpoint, we restart policy fine-tuning from the same initial pretrained weights and train on all successful episodes accumulated by that method. Each fit uses 8,000 flow-matching updates at learning rate 3×10^{-5}; its targets are the stored selected chunks, including their unexecuted suffixes. The first post-warmup block brings collection to 100 episodes, and subsequent blocks add 50 episodes each.

We evaluate each fine-tuned policy without a verifier, using zero action noise on 200 paired initial placements per seed. Figure 8 reports the mean and standard deviation over the same three seeds at every checkpoint through 400 collection episodes.

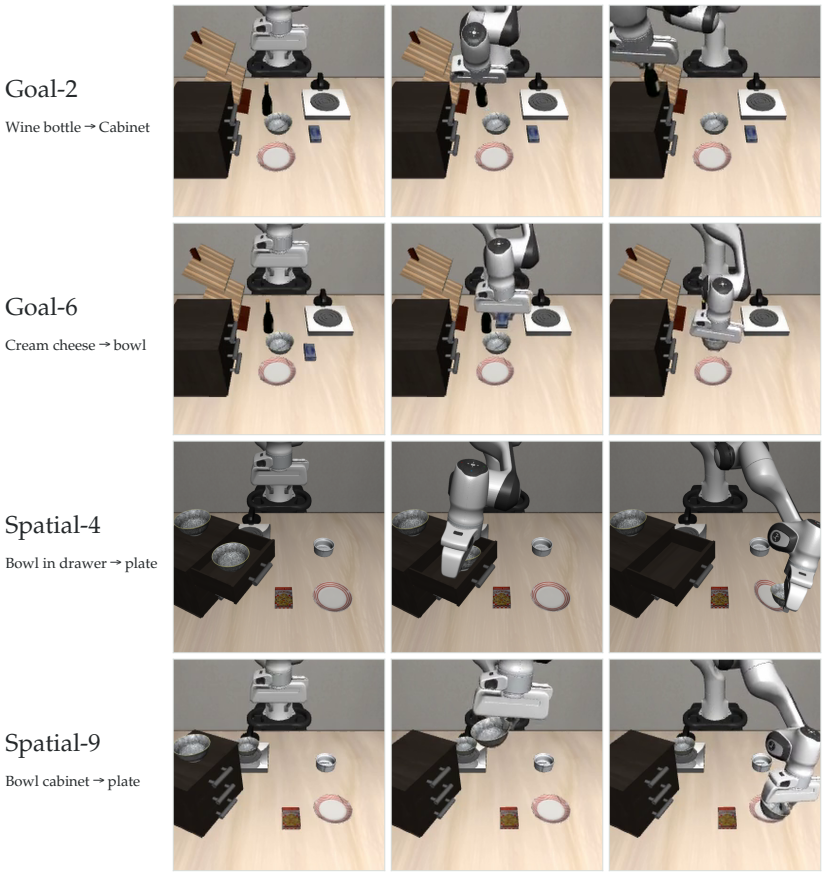


Figure 12: **LIBERO tasks.** Each row shows three frames from successful rollout, from left to right.

D.3 Real-World DROID

We evaluate on a DROID (Khazatsky et al., 2025) setup with a single Franka arm and one external camera: a wrist camera and a third-person camera. The task is to `put the bowl from the dish rack onto the table`. An episode counts as a success only if the bowl is placed upright on the table and released within 350 low-level action steps. The base policy is π_0 (Black et al., 2024), a flow-matching vision-language-action model; we use the publicly released DROID fine-tuned checkpoint [`pi0_droid`]. Each call outputs a 15-step action chunk, of which the first 8 actions are executed open-loop. We keep π_0 frozen throughout. We improve π_0 only by changing the initial noise of its flow-matching sampler via iterative DSBC (Tang et al., 2026).

We first collect 10 warmup episodes with the base policy, recording the noise used for each chunk and the episode outcome, and train the **VETO** verifier on this data using progress rewards from an in-submission (anonymized) reward model (Anonymous, 2026). Each strategy then collects 10 additional episodes. **VETO** samples $N = 16$ noise vectors per decision step and executes one of the top $K = 4$ chunks uniformly at random, updating the verifier after each trajectory. The unguided baseline executes the base policy without verification. For each strategy, we fit a noise DSBC policy on all successful episodes. At test time we decode the DSBC policy through π_0 without any verifier. We refit and evaluate after 20 and 30 total episodes (10 in each round + warmup episodes), using 10 rollouts per evaluation.

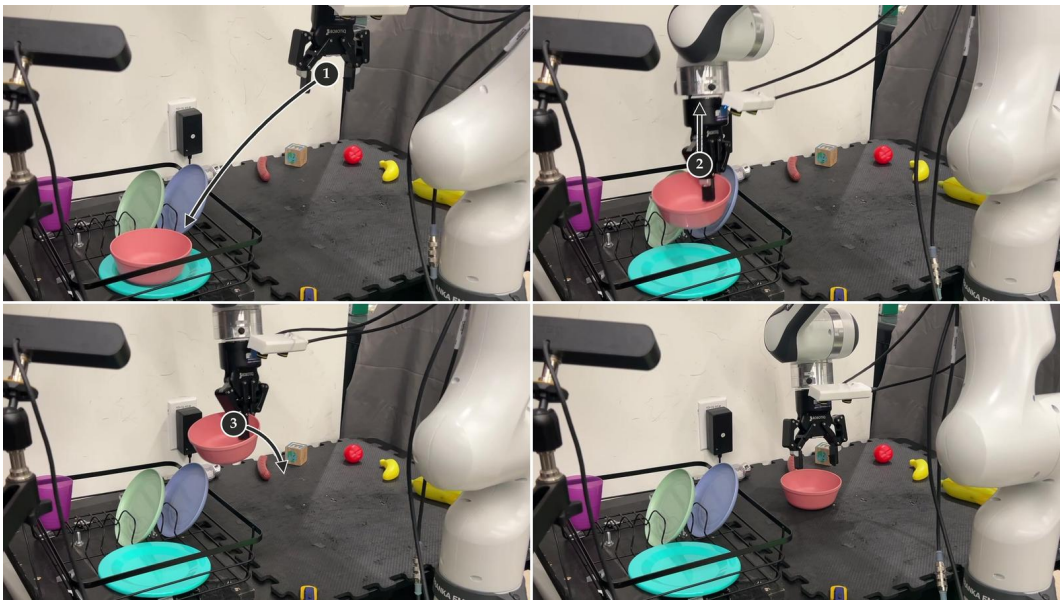


Figure 13: **Real-world DROID task.** The robot (1) reaches to the bowl in the dish rack, grasps it by the rim, (2) lifts it off the plate, and (3) sets it down on the mat before releasing.

D.4 Real-World YAMs

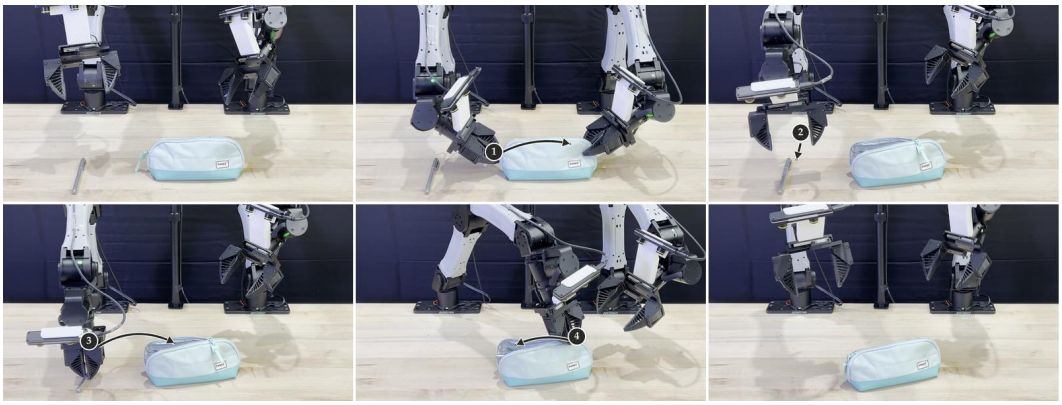


Figure 14: **Real-world YAMs task:** The bag starts zipped. The right arm holds the bag firmly while the left arm (1) grasps the zipper and slides it open, (2) lowers to the pen and (3) places it into the bag, then (4) slides the zipper closed.

We evaluate on a bimanual YAM setup with three ZED cameras (one overhead, one per wrist). The task is to unzip a fabric pouch, place a pen inside, and zip it closed, and an episode counts as a success only if the majority of the pen is fully inside the pouch and the zipper almost fully closed. The base policy and its training data come from Yan et al. (2026). Flex-π is a 6B world-action model built on the Wan-2.2 video model (Wan et al., 2025), pre-trained on about 500 hours of AgiBot World data (AgiBot-World-Contributors et al., 2025) and fine-tuned on 534 teleoperated soft-bag demonstrations (9.7 hours). Each call outputs a 32-step action chunk that runs open-loop at 30 Hz; each episode overall is at most 3000 timesteps (100 seconds of robot interaction time). We keep it frozen and use it in "action-only" generation mode due to faster inference speed.

We first collect 10 episodes with the base policy, recording the noise used for each action chunk and whether the episode succeeded, and train the **VETO** verifier on this data using Robometer (Liang et al., 2026) rewards. Each strategy then collects 60 more episodes: **VETO** samples $N = 16$ noise

vectors per step and executes one of its top 4 chunks at random, updating after every episode, while the unguided baseline runs without verification. For each iteration, we train a DSBC policy on all successful data collected so far and collect the next iteration with the updated DSBC policy. We evaluate after 15, 30, 50, and 70 total episodes, using 20 trials per evaluation.

E Additional Analyses and Ablations

E.1 Filtering Fidelity

We further examine what a verifier needs to improve collection. Using the same held-out PointMaze candidate sets, we compare how often each verifier identifies the single best action chunk with how effectively it removes low-return action chunks. We also measure the return from sampling uniformly among the retained candidates and vary K to assess the effect of filtering strength. Figure 15 shows that useful filtering does not require identifying the single best chunk. After 256 training transitions, VETO identifies a highest-return chunk in 53.5% of held-out states, yet its top-four set retains only 0.7% of the true bottom-quartile candidates, compared with 25% under uniform filtering. Sampling from VETO's retained set gains 1.191 in true H-step return over sampling from all 16 proposals, recovering 87% of the gain available to the oracle top four. At the same budget, the other scorers retain roughly 29% of the bottom quartile and have mean gains near zero. With 4,096 transitions, those scorers also exclude nearly all bottom-quartile chunks, although they identify the single best chunk in only 32–34% of states.

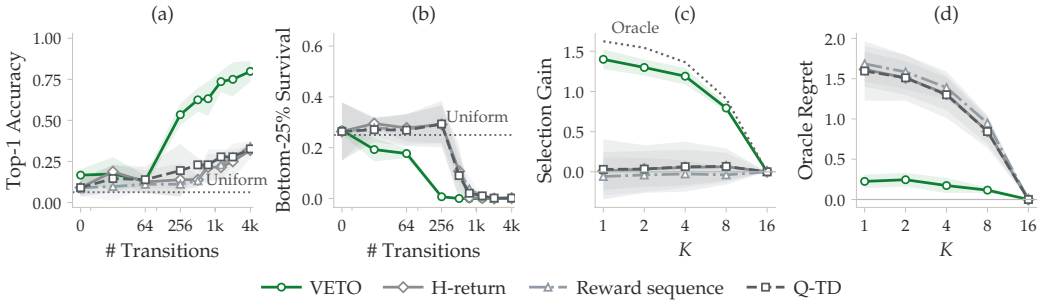


Figure 15: **Filtering fidelity on PointMaze.** Each method scores the same $N = 16$ candidate chunks of horizon $H = 8$. (a) How often its highest-ranked chunk has the highest simulator-measured return. (b) The fraction of the true bottom quartile retained among its top $K = 4$; uniform filtering retains 25% in expectation. Panels (c) and (d) vary K after 256 training transitions: they show the true-return gain over uniform choice from all 16 candidates and the regret relative to the true top-K, respectively. The dotted curve in (c) is the gain available to an oracle using the same candidates.