EXTRACTING BEHAVIOR FROM FROZEN LATENT WORLD MODELS:

EXPERIENCE-ANCHORED PROPOSALS WITH OFFLINE VERIFIER ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning in frozen latent world models is usually performed with the cross-entropy method (CEM). CEM is memoryless: every replan starts a new Gaussian search that uses none of the experience the model was trained on. Planning is therefore slow, 2.1 s per replan in our setup, and uneven, from 67% to 91% success across four suites. We replace this search with a proposer that learns from the model's experience. A generative proposer is behavior-cloned from the corpus. At plan time it generates candidates and the model selects among them by imagined latent distance to the goal. We then improve the proposer by offline reinforcement learning with this imagined distance as the reward, and analytic relative-position constraints can be added at plan time without retraining. We recommend a rectified-flow proposer, the *LeWM Action Proposer* (LeAP), improved by *World-model Advantage-weighted Verifier Regression* (WAVR), which needs no trust-region term. With 256 candidates, a replan takes 24 ms and $1/35$ of CEM's model evaluations, about 90 times faster per planning call. On Cube, TwoRoom and PushT, whose corpora contain directed or expert behavior, the behavior-cloned proposer alone reaches 99.9%, 98.0% and 96.2%, against 67.0%, 88.8% and 91.4% for CEM. On Reacher, whose corpus is random behavior, WAVR raises success from 76.1% to 97.3%, against 84.3% for CEM. With a keep-out region added on Cube at plan time, the planner reaches the goal without entering it in 91.6% of episodes, against 55.9% for CEM.

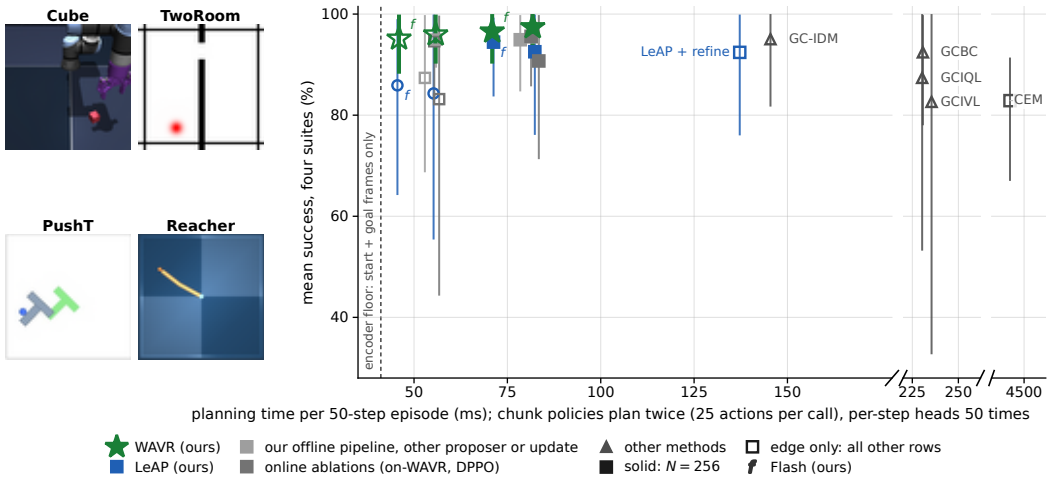

Figure 1: **Mean success against planning time.** Each marker is a row of Table 1: mean success over the four suites, whisker from its lowest to its highest suite. Solid markers are the $N{=}256$ rows of the sampled proposers; all other markers are hollow, including the $N{=}1$ rows of our own pipeline (LeAP, WAVR and their Flash variants). Planning time represents the planner's wall-clock time over one 50-step episode (one environment, one GPU), including the encoding of the start and goal frames, rather than the time of the planner alone. Dashed line: the harness floor, a solver that only encodes the start and goal frames and returns a zero chunk (mean over suites).

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.

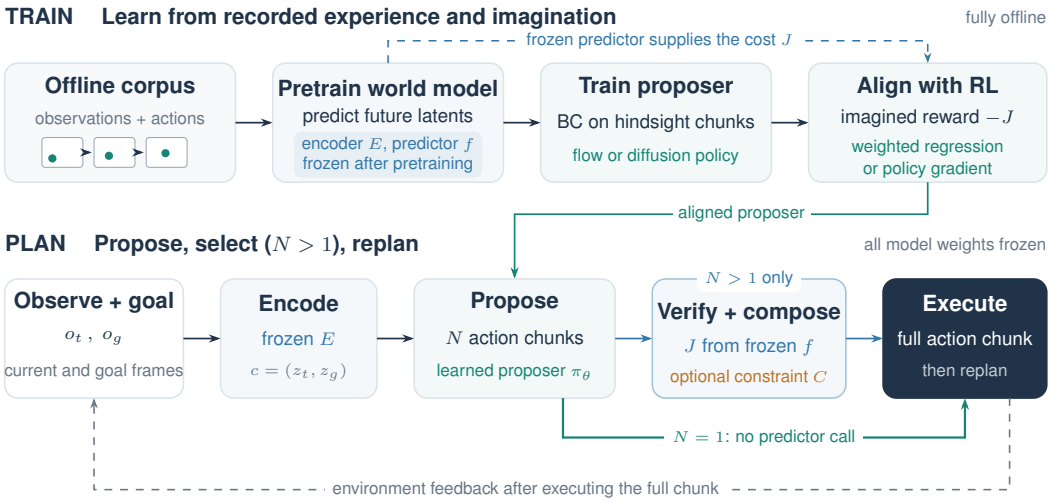

Figure 2: **Extracting a planner from a frozen latent world model.** A shared offline corpus is used to train the world model and initialize a generative proposer through behavior cloning. Offline RL then improves the proposer using rewards from the frozen predictor.

1 Introduction

A central goal in machine learning is to develop general algorithms that produce useful and reusable representations of the world. World models (Hafner et al., 2019; 2025a) represent a promising approach towards this goal by learning representations of observations and predicting future states conditioned on the agent's actions. They can be trained from entirely offline datasets (Hafner et al., 2025b; Maes et al., 2026), enabling the generation of imagined experiences and evaluation of counterfactual scenarios without further environment interaction.

Some world models (Hafner et al., 2025b) learn their representations through image reconstruction, which can devote significant computation to task-irrelevant visual details despite achieving excellent long-horizon results. DINO-WM (Zhou et al., 2025) avoids reconstruction by learning to predict future states in the feature space of a frozen, pretrained DINOv2 encoder (Oquab et al., 2024). LeWorldModel (LeWM) (Maes et al., 2026) instead uses an efficient end-to-end JEPA approach, jointly learning the encoder and an action-conditioned predictor.

Latent world models usually involve an encoder that maps observations to latents, an action-conditioned predictor that rolls them forward, and a planner that plans using the predictor's imagined rollouts (Hafner et al., 2019; Hansen et al., 2024; Zhou et al., 2025). While the world-model trunk varies by method (Chua et al., 2018; Hafner et al., 2019; Zhou et al., 2025; Hansen et al., 2024), planning often uses search methods such as CEM (Rubinstein & Kroese, 2004) or MPPI (Williams et al., 2017) (§2). At plan time, these methods sample action sequences from Gaussian distributions, score them with imagined rollouts, and refit toward the best scorers. This process can require thousands of predictor rollouts and also result in suboptimal latency, from under one second for the more efficient LeWM to tens of seconds for DINO-WM (Maes et al., 2026).

To overcome these limitations, we propose a generative proposer for LeWorldModel (Maes et al., 2026) that learns from the same corpus used to train the world model. Through behavior cloning on hindsight-relabeled sequences, the proposer gains an experience-based prior on navigating the latent space. In the RL stage, the world model predictor serves as a reward model grading generated proposals to align the proposer with goal-conditioned behavior. Our aim is to learn reusable behavior offline to amortize planning, while retaining the world model's ability to compare candidates and evaluate additional constraints at plan time.

Contributions.

1. A recipe for extracting an effective planner from a frozen latent world model through behavior cloning, offline RL, and optional test-time selection and composition.

2. A performant default implementation: a rectified-flow proposer, *LeAP*, aligned through *WAVR*. WAVR computes group-relative advantages from negative imagined costs and uses them for advantage-weighted regression without a trust-region term. Furthermore, the compute-optimized *LeAP-Flash* and *WAVR-Flash* use a 0.31M-parameter proposer and four sampling steps while achieving comparable planning performance. Proposing a chunk with a Flash variant takes around 6 ms and costs 2.5 MFLOP, 50 times less than LeAP's 130 MFLOP and 0.04% of the 6.8 GFLOP fixed cost of encoding the start and goal frames. *Compose* adds analytic constraints at plan time without retraining and is tested on Cube.
3. An empirical analysis of when the frozen model remains useful for planning and policy improvement (§6; Supp. E).
4. Replications over independent pretraining draws, three to six per suite. On Reacher, the WAVR draws agree within 1.0 points at $N{=}1$ and 0.4 at $N{=}256$ (Supp. F.1).

2 Related work

BC and RL in frozen world models. Dreamer 4 (Hafner et al., 2025b) learns long-horizon behavior from offline data by training a policy inside a generative video world model. We train a separate generative action proposer using negative goal costs from a frozen latent model as rewards. Offline model-based RL penalizes model rewards or values by uncertainty or conservatism (Yu et al., 2020; Kidambi et al., 2020; Yu et al., 2021). WAVR adds no penalty but queries the model only to rank samples from its own proposer. Section 6 measures when a frozen latent model's costs remain useful for ranking actions and improving the policy.

Generative proposers. Diffusion models generate trajectories for planning and action sequences for imitation learning (Janner et al., 2022; Ajay et al., 2023; Chi et al., 2023). LeAP uses a rectified flow (Liu et al., 2023; Lipman et al., 2023) with classifier-free guidance (Ho & Salimans, 2022), trained on hindsight pairs from the world model's recorded corpus (Andrychowicz et al., 2017; Ghosh et al., 2021; Lynch et al., 2019). Offline RL methods also sample from a generative behavior model and select or reweight samples with a learned critic (Chen et al., 2023; Hansen-Estruch et al., 2023). LeAP ranks samples with the frozen predictor and trains no critic.

Aligning samplers with rewards. WAVR uses advantage-weighted regression, as in AWR and AWAC (Peng et al., 2019; Nair et al., 2020), with group-relative advantages as in GRPO (Shao et al., 2024) (§4.2). DPPO and Flow-GRPO instead update generative policies using policy gradients (Ren et al., 2025; Liu et al., 2025). Best-of-N selection with learned verifiers is also used in language models (Cobbe et al., 2021; Lightman et al., 2024; Snell et al., 2025). ReST and RAFT train on high-reward generated outputs (Gulcehre et al., 2023; Dong et al., 2023). BOND trains a model to match the best-of-N distribution (Sessa et al., 2025). WAVR belongs to this family. As in reward-model overoptimization (Gao et al., 2023), improvements in imagined reward do not always produce better executed behavior.

Per-step policies on frozen latents. VPT (Baker et al., 2022) uses an inverse-dynamics model to label actions in video, then trains a policy on those labels. GC-IDM (Nguyen et al., 2026) predicts the next action from the current latent, goal latent, and remaining horizon, and replans at every step. The goal-conditioned BC and value heads of OGBench (Park et al., 2025), trained on the same frozen latents as in Maes et al. (2026), also act per step; Table 1 reports them beside our rows. LeAP samples complete action chunks and evaluates more candidates as N grows.

Planning with learned world models. World models are used for control by searching over action sequences or by training policies on imagined experience (Chua et al., 2018; Hafner et al., 2019; 2025a; Hansen et al., 2024; Ha & Schmidhuber, 2018; Hafner et al., 2020). LeWM (Maes et al., 2026) learns latent dynamics without reward labels or image reconstruction, following the JEPA approach (LeCun, 2022; Assran et al., 2023; Bardes et al., 2024). DINO-WM (Zhou et al., 2025) uses CEM to select action sequences by predicted distance to a goal in feature space. Our CEM baseline follows this approach using a frozen LeWM. Flow-JEPA (Huo et al., 2026) uses flow matching to predict future latent trajectories. Qantara (Rakhimov et al., 2026) jointly models actions and latents, allowing one checkpoint to support planning, behavior cloning, and inverse dynamics. Our flow model proposes action chunks while the world model stays frozen.

3 BACKGROUND: FROZEN WORLD MODELS AND GOAL COSTS

We train an unmodified LeWM (Maes et al., 2026) on each environment's offline observation–action dataset. Its ViT-tiny encoder E maps an observation o to a latent representation $z = E(o) \in \mathbb{R}^D$. The transformer predictor f predicts the next latent from the $H{=}3$ most recent latents and their action blocks, where each block u_k contains five low-level actions: $\hat{z}_{k+1} = f(z_{k-H+1:k}, u_{k-H+1:k})$. At plan time, the predictor is conditioned with one observed latent. Both LeAP and CEM plan over chunks of $T = 5$ such blocks (25 low-level actions).

The encoder and predictor are trained jointly with

$$\mathcal{L}_{\text{pre}}(E, f) \;=\; \mathbb{E}\left\|\hat{z}_{k+1} - E(o_{k+1})\right\|^2 \;+\; \lambda\,\text{SIGReg}(z), \tag{1}$$

where the first term (prediction loss) matches the predicted latent $\hat{z}_{k+1}$ to the encoded next latent $E(o_{k+1})$, one action block ahead. The second term, SIGReg (Balestriero & LeCun, 2025), discourages representation collapse by aligning the latent distribution to an isotropic Gaussian. Both E and f are frozen after pretraining.

To score an action chunk a, we roll it forward from the initial latent z_0 and compare the predicted latents $\hat{z}_t(a)$ with the encoded goal z_g:

$$J(a;\, z_0, z_g) \;=\; \text{Agg}_t\, \left\|\hat{z}_t(a) - z_g\right\|_2^2. \tag{2}$$

Here, Agg_t combines the distances across prediction steps. Our evaluations follow the default implementation of LeWM (further ablations in Supp. A.2). We call the frozen predictor the *verifier* when it is used to compute this cost.

4 METHOD

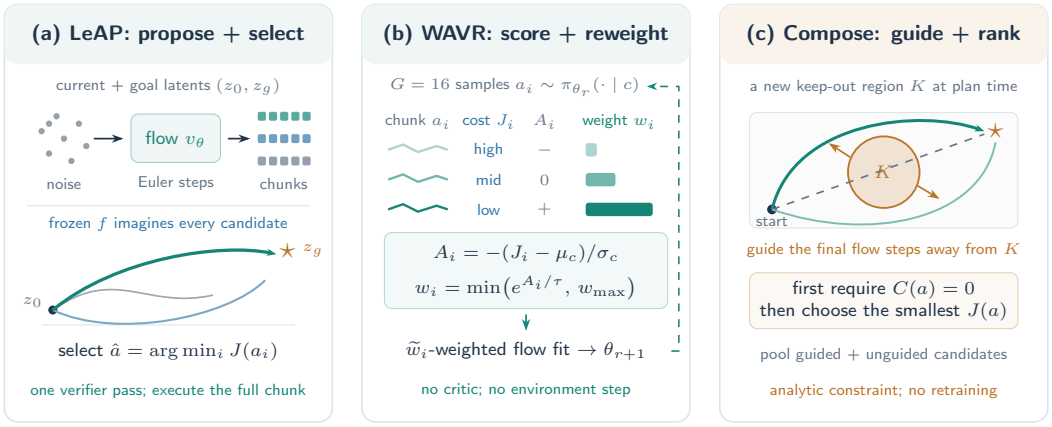

Figure 3: **Method.** (a) LeAP samples chunks and keeps the lowest imagined cost. (b) WAVR refits the proposer to its own samples with capped advantage weights. (c) Compose guides and ranks under an analytic constraint.

4.1 PLAN: A PROPOSER PRETRAINED ON EXPERIENCE (LEAP)

Training: flow matching on hindsight pairs. LeAP is a small rectified-flow proposer $\pi(a \mid c)$ over action chunks (Fig. 3a), trained on the pretraining corpus with evaluation episodes excluded (data sources and pool sizes in Supp. A.1). The training corpus is hindsight-relabeled (Andrychowicz et al., 2017; Ghosh et al., 2021): for an observation o_k, the goal is set as the observation o_{k+25} reached one chunk later on the same trajectory. The proposer is conditioned with $c = (z_0, z_g) = (E(o_k), E(o_{k+25}))$, and in training the proposer network is matched to a conditional velocity field $v_\theta(a_t, t \mid c)$ along the straight path from Gaussian noise a^0 to the recorded chunk a^*, the 25 actions executed between o_k and o_{k+25}. With $a^0 \sim \mathcal{N}(0, I)$ and $t \sim \mathcal{U}[0, 1]$,

$$\mathcal{L}(\theta) \;=\; \mathbb{E}\left\|v_\theta\big((1{-}t)\,a^0 + t\,a^*,\; t \mid c\big) \;-\; \big(a^* - a^0\big)\right\|^2. \tag{3}$$

Inference: sample, verify, select. To sample a candidate, we integrate $\dot{a}_t = v_\theta(a_t, t \mid c)$ from Gaussian noise at $t{=}0$ to $t{=}1$ with a fixed number of Euler steps (Fig. 3a). We draw N candidates in one batch and choose the one with the lowest imagined cost:

$$a_1, \ldots, a_N \sim \pi(\cdot \mid z_0, z_g), \qquad \hat{a} = \arg\min_{i \le N} J(a_i;\, z_0, z_g). \tag{4}$$

The verifier scores all N candidates in one batched rollout of T prediction steps. The selected chunk is executed in full and we replan only at chunk boundaries. At $N{=}1$, LeAP is a flow policy where no ranking is needed and the predictor is not run. *Propose-then-refine* (LeAP + refine) adds three rounds of elite refitting to the $N{=}256$ pool: each round fits a diagonal Gaussian to the 32 lowest-cost chunks scored so far, draws 128 new chunks from it and scores them. The chunk with the smallest imagined cost over all 640 scored chunks is then selected (Supp. B.2).

4.2 Teach: offline RL on the proposer (WAVR)

WAVR is our offline RL algorithm (Fig. 3b). It is on-policy in imagination: each round fits chunks sampled from the current proposer, and no environment step is taken. Here, the state is a context $c = (z_0, z_g)$, the action is an action chunk a, and the reward is $-J(a; z_0, z_g)$ with the minimum over the five prediction steps as the aggregation rule Agg_t; deployment ranking (§4.1) keeps LeWM's default (Supp. A.2). We cache a set of contexts from the corpus. In round r, the proposer π_{θ_r} samples a group of G chunks $a_1, \ldots, a_G$ per context. The verifier then rolls the chunks out autoregressively and returns their costs J_i, and we standardize the rewards within the group:

$$A_i = -\frac{J_i - \mu_c}{\max(\sigma_c, 10^{-6})}, \qquad w_i = \min\Big\{e^{A_i/\tau},\, w_{\max}\Big\}, \qquad \widetilde{w}_i = \frac{w_i}{\overline{w}}, \tag{5}$$

where μ_c and σ_c are the mean and standard deviation of the group's costs, $\overline{w}$ is the mean weight over the buffer $\mathcal{B}_r$, τ is a temperature and $w_{\max}$ is a weight cap. The next proposer minimizes the flow-matching loss (3) on this fixed buffer through weighted regression

$$\mathcal{L}_r(\theta) = \mathbb{E}_{(c,a_i)\sim\mathcal{B}_r}\, \widetilde{w}_i \left\| v_\theta\big((1{-}t)\, a^0 + t\, a_i,\; t \mid c\big) - \big(a_i - a^0\big) \right\|^2. \tag{6}$$

We conduct this weighted regression on the aforementioned buffer iteratively over multiple rounds (full settings in Supp. B.3). Note that since the weights increase with $-J$, each fit sharpens the sampler toward the chunks that best-of-N selection would pick, resulting in a powerful $N{=}1$ policy (§5.3).

4.3 Compose: constraints as test-time adaptation

Composition handles objectives that become available only at plan time (Fig. 3c). Each objective adds an analytic cost C_j in raw action space. Our example is a hard keep-out ball on the relative displacement path obtained by summing the chunk's actions. Its penetration term, $C(a)$, is differentiable almost everywhere and requires no world-model call. The goal cost J is then the only learned term that could be exploited, and we use it only to rank candidates.

A constraint can enter at selection or at sampling. At selection, we rank the prior's N samples under the composite cost. At sampling, we steer the frozen flow toward the feasible set with analytic guidance,

$$v \leftarrow v - \eta \|v\| \widehat{\nabla} C, \tag{7}$$

where $\widehat{\nabla} C$ is the unit gradient, η sets the step relative to the flow field, and guidance acts only in the final Euler steps (Supp. B.7). We pool unguided candidates with guided candidates at several η and select lexicographically:

$$\hat{a} = \arg\min_{a \in \mathcal{P}\,:\, C(a)=0} J(a). \tag{8}$$

If no candidate in $\mathcal{P}$ is feasible, we execute the one with minimum penetration and record the fallback (Supp. B.7).

5 EXPERIMENTS

5.1 ENVIRONMENTS AND BASELINES

Suites. Our evaluation contains four pixel-based environments spanning navigation, reaching, pushing, and manipulation: *Reacher* (DMC; Tassa et al., 2018), *PushT* (Florence et al., 2021; Chi et al., 2023) (inputs 224^2), *Cube* (OGBench cube-single pick-and-place; Park et al., 2025), and *TwoRoom* (shipped with `stable-worldmodel`). Goals are frames encoded by the frozen encoder, and success is determined by whether the world state comes within a tolerance zone centered on the goal state; complete environment and evaluation definitions are in Supp. A.1.

Evaluation. All methods use the chunked loop from §4. We report both $N{=}1$ and $N{=}256$ results. Complete evaluation and reporting details are in Supp. A and F.

Baselines. *CEM* iteratively refits Gaussian proposals toward chunks with low frozen-verifier cost (Supp. A.1). The LeAP rows use the behavior-cloned flow policy before alignment. GCBC, GCIVL and GCIQL are the OGBench methods (Park et al., 2025) as implemented in the LeWM release (Maes et al., 2026); unlike the published configuration, which used DINOv2 features, here they act on the LeWM latent. *Flow-GRPO+KL* (Liu et al., 2025) applies a policy-gradient update with a KL anchor to the same flow proposer and the same imagined cost (Supp. C.4). *Online-WAVR* shares WAVR's policy and update, but learns from new environment contexts scored by shaped environment rewards (Supp. C.3).

Diffusion Policy. Chi et al. (2023) learn a conditional denoising diffusion model over action sequences. Our baseline adapts it to frozen latent inputs $[z_0, z_g]$, using the DDPM (Ho et al., 2020) implementation of DPPO (Ren et al., 2025) (Supp. C.2).

5.2 PLANNING WITH LEAP

Table 1 and Figure 1 compare all methods on the four suites. LeAP outperforms CEM on Cube and TwoRoom. On Cube, selection at $N{=}256$ reaches 99.9 against CEM's 67.0. On PushT, LeAP reaches $96.2{\pm}1.8$ against CEM's $91.4{\pm}3.9$ with $1/35$ of its verifier evaluations. One replan for one environment takes 24 ms at $N{=}256$ and 10 ms at $N{=}1$ against 2.1 s for CEM (Supp. B.1). With selection, LeAP trails CEM only on Reacher ($76.1{\pm}1.3$ against 84.3), where the behavior-cloned proposer covers goal-directed reaching diffusely. This gap motivates the alignment stage. Table 1 also lists predictor-free policies and a supervised inverse model trained on the same frozen latents (Supp. C.1). The LeAP-Flash and WAVR-Flash rows run the same recipe with a 64×3 proposer (0.31M parameters, 8.4 times fewer) sampled in four Euler steps instead of 25 (Supp. B.6).

The limits of additional search. Additional search saturates: propose-then-refine (§4.1) stays within 0.7 points of LeAP at $N{=}256$ in Table 1 and does not close the Reacher gap (Supp. B.1).

5.3 OFFLINE ALIGNMENT WITH WAVR

Offline rounds from the behavior-cloned prior (16, or 32 on Reacher; settings in Supp. B.3) raise $N{=}1$ success on every suite with headroom (Table 1): Reacher from 55.4 to $94.0{\pm}1.6$, TwoRoom from 94.2 to 99.8 and PushT from $87.7{\pm}3.1$ to $90.2{\pm}0.6$, while Cube starts at 99.9 at both operating points and ends at 99.9 and 100.0. With selection at $N{=}256$, Reacher rises from 76.1 to $97.3{\pm}0.7$ and TwoRoom from 98.0 to 99.7, while PushT falls from 96.2 to 92.6 (§7). The three Reacher pretraining draws agree to 1.0 pp at $N{=}1$ and 0.4 pp at $N{=}256$ (Supp. F.1).

Comparison with online-WAVR. Online-WAVR starts from WAVR's prior and trains for the same number of rounds with environment interaction (Supp. C.3). WAVR and online-WAVR reach similar success at both $N{=}1$ and $N{=}256$. On Reacher, WAVR can also start from random weights and approach online-WAVR with no expert data at any stage (Supp. B.4).

Decline after many rounds. One original Reacher run peaks around rounds 40–48 and declines by round 64 (Supp. B.4). Group standardization can amplify small cost differences, but we have not

Table 1: Success (%) by representation and predictor use. Rows are our measurements on frozen LeWM encoders with the retained pool and one evaluator (n=200 and three evaluation seeds unless stated in Supp. B.1; mean±sd over the units of Supp. B.1, C.1), except the cited DINOv2 group. N: candidates scored by the predictor at test time; refine uses 640 predictor evaluations per replan, CEM 9,000. Environment interaction: 26.7M steps (DPPO), 12.8M (online-WAVR; Reacher 152.3M, Supp. C.3). Release-recipe rows: one training seed, three-frame history as trained. WAVR-zero: WAVR from a randomly initialized proposer (Supp. B.4).

	predictive latent	TD value	env. interaction	predictor in training	predictor at test time	Cube	TwoRoom	PushT	Reacher
Model-free on DINOv2 features (Maes et al., 2026; Oquab et al., 2024)									
GCBC (DINOv2)	✗	✗	✗	✗	✗	84	100	75	—
GCIVL (DINOv2)	✗	✓	✗	✗	✗	56	100	33	—
GCIQL (DINOv2)	✗	✓	✗	✗	✗	64	100	20	—
Behavior cloning									
GCBC, per step (release)	✓	✗	✗	✗	✗	94.2±1.3	99.8±0.3	78.0±3.9	97.7±1.2
GC-IDM, per step (horizon-cond.)	✓	✗	✗	✗	✗	99.1±0.3	100.0±0.0	81.7±1.4	99.3±0.0
Diffusion Policy, N=1	✓	✗	✗	✗	✗	99.8±0.3	96.5±1.0	84.5±3.5	68.7±5.0
LeAP, N=1	✓	✗	✗	✗	✗	99.9±0.1	94.2±1.2	87.7±3.1	55.4±1.5
LeAP-Flash, N=1	✓	✗	✗	✗	✗	99.0±0.3	96.6±0.5	83.8±0.4	64.2±3.5
Offline model-free RL									
GCIVL, per step (release)	✓	✓	✗	✗	✗	97.8±0.3	100.0±0.0	32.7±4.0	100.0±0.0
GCIQL, per step (release)	✓	✓	✗	✗	✗	96.5±0.5	100.0±0.0	53.2±6.4	99.8±0.3
Online RL									
DPPO, N=1	✓	✗	✓	✗	✗	96.5±0.5	99.7±0.3	44.3±1.2	92.2±1.0
online-WAVR, N=1	✓	✗	✓	✗	✗	99.8±0.3	99.7±0.3	83.3±2.5	95.7±0.7
Search									
CEM	✓	✗	✗	✗	✓	67.0±4.6	88.8±1.7	91.4±3.9	84.3±0.8
Offline model-based RL (our pipeline)									
WAVR, N=1 (ours)	✓	✗	✗	✓	✗	99.9±0.2	99.8±0.1	90.2±0.6	94.0±1.6
WAVR-Flash, N=1 (ours)	✓	✗	✗	✓	✗	97.3±0.4	99.9±0.1	88.2±0.3	95.2±1.9
WAVR-zero, N=1	✓	✗	✗	✓	✗	67.4±1.1	87.1±0.6	60.3±1.2	91.9±2.4
Flow-GRPO+KL, N=1	✓	✗	✗	✓	✗	99.0±0.2	98.7±0.3	92.7±0.9	89.4±1.8
With test-time search									
LeAP, N=256 (ours)	✓	✗	✗	✗	✓	99.9±0.1	98.0±0.7	96.2±1.8	76.1±1.3
LeAP-Flash, N=256 (ours)	✓	✗	✗	✗	✓	99.3±0.3	99.1±0.3	95.7±0.3	83.7±0.3
LeAP + refine (ours)	✓	✗	✗	✗	✓	99.9±0.2	98.3±0.3	95.5±1.0	76.0±3.6
WAVR, N=256 (ours)	✓	✗	✗	✓	✓	100.0±0.0	99.7±0.3	92.6±0.5	97.3±0.7
WAVR-Flash, N=256 (ours)	✓	✗	✗	✓	✓	98.7±0.1	100.0±0.0	90.2±0.8	97.3±1.3
WAVR-zero, N=256	✓	✗	✗	✓	✓	70.4±0.8	88.2±0.5	71.4±2.4	96.2±0.6
Diffusion Policy, N=256	✓	✗	✗	✗	✓	99.7±0.3	99.8±0.3	95.5±1.8	84.7±1.3
Flow-GRPO+KL, N=256	✓	✗	✗	✓	✓	98.8±0.5	98.5±0.5	94.8±0.3	89.9±1.0
DPPO, N=256	✓	✗	✓	✗	✓	97.4±0.8	99.8±0.2	71.3±0.9	94.4±1.5
online-WAVR, N=256	✓	✗	✓	✗	✓	99.8±0.3	100.0±0.0	85.7±1.8	96.9±0.6

established verifier noise as the cause. The training scripts use fixed round counts and no checkpoint selection.

Single-sample deployment. WAVR effectively distills best-of-N selection into the N=1 feedforward policy. On Reacher, WAVR at N=1 (94.0) exceeds the prior with selection at N=256 (76.1), and on TwoRoom it does as well (99.8 against 98.0). On PushT, WAVR at N=1 (90.2) stays 6.0 points below the prior with selection (96.2).

Policy-gradient baselines. DPPO (online, 26.7M environment steps) and Flow-GRPO+KL (offline, against the same frozen imagined cost with a static KL anchor), the two policy-gradient rows of Table 1, are compared with WAVR in Supp. C.4.

5.4 Plan-time composition

We evaluate plan-time composition with a hard keep-out region placed across the integrated action path on Cube; Table 2 isolates each part of the method. Ranking alone reaches 34.1% joint success, indicating limited feasible coverage under the newly introduced obstacle. Constrained CEM satisfies the action-path constraint but has approximately 36 points lower goal success than guided selection. Analytic guidance changes the candidate distribution while keeping prior weights fixed, and uses the verifier only to rank candidates. It satisfies the constraint with $1/9$ of constrained CEM's predictor evaluations (1,024 against 9,000) and requires no model update, prior update, or new data; all replications have zero violations (Supp. B.7).

Table 2: Keep-out composition on Cube, with the same episode-keyed keep-out zones for every method. Joint success requires goal success and zero penetration; values are means±sd over evaluation runs of n=200 episodes: four runs per LeAP-based row (one per pretraining draw) and twelve for constrained CEM (four draws × three evaluation seeds). Supp. B.7 gives the zone protocol and the run units.

	Joint success (%) ↑	Violation rate (%) ↓	Goal success (%) ↑
Unconstrained LeAP	12.2±0.9	87.8±0.9	99.9±0.2
+ ranking only	34.1±2.3	65.9±2.3	98.1±0.5
+ late guidance, mixed-η	**91.6±2.2**	0.0	91.6±2.2
Constrained CEM	55.9±3.6	0.0	55.9±3.6

5.5 Effect of proposal-stage data volume

We retrain the behavior-cloned proposer on nested subsets of each corpus, from the full training set down to $1/256$, with the pretrained world model of each suite fixed (Fig. 4; full results and controls in Supp. D.1). At N=1, Cube and TwoRoom lose about ten points down to $1/64$ of the corpus and then drop sharply at the next halving, whereas PushT declines across the whole ladder; selection with N=256 degrades more gradually on every suite, and on TwoRoom a proposer trained on only 31 episodes still reaches $95.6 \pm 1.4\%$ with 256 candidates (coverage analysis in §6). Reacher is absent because its corpus is random behavior; there the behavior-cloned prior adds only 2.1 and 1.1 points over a random initialization at N=1 and N=256 (Table 1; Supp. B.4). Aligning each reduced-data initialization with WAVR, with its context pool drawn from the same reduced corpus, brings N=1 success close to the initialization's N=256 success until the context pool becomes small (Fig. 4b).

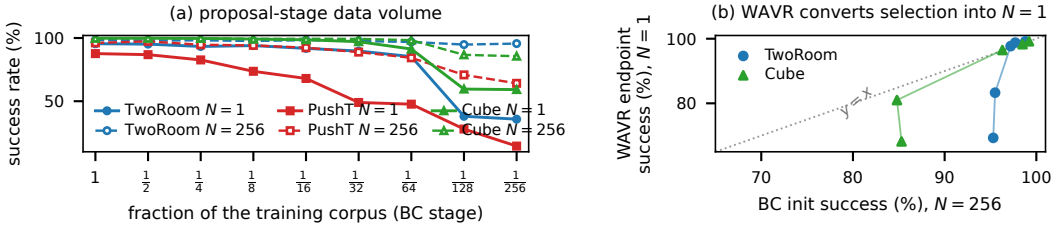

Figure 4: **Effect of proposal-stage data volume.** *(a)* LeAP success against the fraction of the training corpus; every PushT point and every point at $1/8$ or more uses draw 1, and TwoRoom and Cube points at $1/16$ and below pool three draws. *(b)* WAVR's N=1 success after alignment against the N=256 success of its reduced-data initialization, TwoRoom and Cube.

6 Pessimism through support: where the frozen model is reliable

To forestall reward hacking, model-based offline RL usually relies on mechanisms such as penalizing model uncertainty or KL-based trust regions to keep the policy and its imagination within the region where the model is reliable. WAVR has no such anchor. Instead, its pessimism comes from the information it asks of the model and from its update rule, weighted regression on its own proposals: WAVR only queries the frozen predictor to rank chunks sampled by the current policy and never optimizes against it. This section measures the robustness of this guard (full measurements in Supp. E).

Where the support is. Let $\mathcal{Z}_r$ be a reference set of 15,000 start latents drawn from the corpus and, for any latent z, let $d(z)$ be the mean Euclidean distance from z to its ten nearest members of $\mathcal{Z}_r$. Let $\mathcal{H}$ be a held-out set of 4,000 corpus windows (z_0, a), disjoint from $\mathcal{Z}_r$, each imagined under its own recorded chunk to give the predicted latents $\hat{z}_1(a), \ldots, \hat{z}_5(a)$ of Eq. (2). The latent threshold is the 99th percentile

$$\tau_z \;=\; Q_{0.99}\big\{\, d(\hat{z}_t(a)) \;:\; (z_0, a) \in \mathcal{H},\; t = 1, \ldots, 5 \,\big\}, \tag{9}$$

so that one percent of in-distribution imagined latents lie beyond it. A chunk a from context z_0 is *in-support* when $\max_t d(\hat{z}_t(a)) \le \tau_z$, and a context is off-support-driven when at least half of its sampled chunks are not. The action-space check repeats the construction with recorded chunks as

the reference set, in the proposer's normalized action coordinates, giving a threshold τ_a (Supp. E.1). These diagnostics locate where the model extrapolates; they do not certify the model's accuracy inside the region (metric calibration in Supp. E.2).

Ranking is reliable; optimizing against the model is not. We measured the two kinds of query, ranking sampled chunks and moving a chunk along the model's cost gradient, by executing candidates (Supp. E.3). For each of 128 contexts per suite with the 25-step hindsight goal, the Table 1 proposer samples 64 chunks, which the frozen model scores and the simulator executes. On these samples, imagined cost ranks executed distance with Spearman 0.36–0.48. On TwoRoom, PushT and Cube a random sample already reaches the goal within the chunk in 91–100% of contexts and the imagined argmin in 99–100%, so ranking has little left to add there, whereas on Reacher the one-frame verifier picks a reaching chunk in 20% of contexts and the history-seeded verifier in 46%, against 10% for a random sample and 91% for an oracle. We then optimized each sample against the model, moving it along the negative gradient of the imagined cost within a fixed path budget and keeping only the steps that lowered the cost. The imagined cost fell by 83–97% for the median candidate while the executed distance fell by only 18–30%, 22–33% of the candidates ended farther from the goal than they started, and among the optimized candidates the model's ranking collapsed to Spearman 0.07–0.21, against 0.36–0.49 after random moves of the same size. This is model exploitation in the offline-RL sense, and the model can no longer order what the optimization produced. Rank-only queries on the policy's own samples avoid it without a penalty, which is why LeAP selects before it refines and why WAVR never differentiates through the predictor.

The update rule decides which queries are made. Because WAVR reweights a fixed buffer sampled from the current policy, its queries stay on samples of a policy that began in the data, whereas a policy-gradient update on the same imagined cost with a KL anchor (Flow-GRPO+KL), our comparison, moves the policy with likelihood-ratio gradients. Measured with the thresholds of Eq. (9), both rules leave at most 2.7% of contexts off-support-driven, but 17–18% of the gradient policy's chunks exceed τ_a against 0.0% for WAVR, and on the policy's own samples the history-seeded Spearman falls from 0.76 to 0.35 over 20,000 gradient updates while it rises to 0.82–0.88 under WAVR (one training run per rule; Supp. E.3). The guard therefore lives in restricting the model's queries to samples it has been shown to rank, not in a penalty term.

Coverage bounds improvement. Offline RL can only improve behavior that the data cover, and here the data enter through the proposer: a chunk of probability p under it appears among N samples with probability $1-(1-p)^N$. Ranking can therefore recover rare behavior at test time and WAVR can make it likely in a single sample, but both act on the proposer's initial coverage, which is why ranking helps most on reduced corpora and why alignment declines once conditioning coverage becomes too small (§5.5). The model's answer also applies to the complete imagined chunk, so interrupting a chunk and replanning can change the outcome (Supp. E.4).

7 Conclusion and Limitations

We propose a framework for extracting behavior from a frozen latent world model. A generative proposer serves as the policy. It is fitted by behavior cloning on the model's corpus and improved by offline reinforcement learning against the model's imagined goal cost, with no environment interaction. On four suites this reaches robotics-level latency (10 ms per replan at $N{=}1$ and 24 ms at $N{=}256$, against 2.1 s for CEM) and success rates of 90.2 to 100.0 (Table 1), while keeping the option to conduct plan-time composition under analytic constraints.

Limitations. Per-step alternatives act on every frame and read 97.7–100.0 on Reacher (Table 1), against 97.3 for WAVR at $N{=}256$ and 94.0 at $N{=}1$; our proposer commits to 25 actions between planning calls. We read the gap as the value of per-step feedback on a suite with momentum. WAVR has no value function: its reward is the imagined cost at the end of one chunk, so credit is not assigned across chunks as it is in actor-critic training in imagination (Hafner et al., 2025b). As a result, TwoRoom success falls to 77.0–87.3 for goals 100 steps away, which is far beyond the one-chunk training horizon (Supp. B.4). On PushT, WAVR raises $N{=}1$ success from 87.7 to 90.2 but lowers $N{=}256$ success from 96.2 to 92.6.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations (ICLR)*, 2023.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research (TMLR)*, 2024.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research (TMLR)*, 2023.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning (CoRL)*, 2021.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, 2023.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

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 (ICML)*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025a. arXiv:2301.04104.

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

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. *arXiv preprint arXiv:2304.10573*, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yanchen Huo, Ziying Song, and Yadan Luo. Flow-JEPA: Flow matching for robust latent dynamics in JEPA world models. *arXiv preprint arXiv:2608.29029*, 2026.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning (CoRL)*, 2019.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models. *arXiv preprint arXiv:2605.08732*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

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

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Ruslan Rakhimov, George Bredis, Yuriy Maksyuta, and Daniil Gavrilov. Qantara: Bridge-flow training for multi-paradigm JEPA control. *arXiv preprint arXiv:2607.04978*, 2026.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer, 2004.

Pier Giuseppe Sessa, Robert Dadashi, Léonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Ramé, Bobak Shariari, Sarah Perrin, Abe Friesen, Geoffrey Cideron, et al. BOND: Aligning LLMs with best-of-N distillation. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning (ICML)*, 2025.

AI USE STATEMENT

In this work, we used generative AI tools (Claude, Anthropic) for implementing methods, designing and providing feedback on experimental protocols, interpreting results, proposing and refining hypotheses, and helping develop the conceptual framework of Section 6 and the supplement. We have not used generative AI tools to generate synthetic datasets or to write proofs; translation, dataset cleaning and reformatting, and qualitative or thematic analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, create and modify scientific figures, draft parts of the main paper and of the supplementary material from the authors' outlines and results, edit the paper for readability, identify and summarize relevant literature, and format references. We have reviewed all AI-assisted work: every experiment driver was reviewed by the authors before use, every reported number is produced by the released scripts and was re-derived from the raw evaluation records, AI-drafted text was revised by the authors, and AI-suggested related work was checked against the cited papers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.