IMWM: INTUITION MODELS COMPLEMENT WORLD MODELS FOR LATENT PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with a learned latent world model is a promising route to control from raw pixels, and when such a planner fails, the usual remedy is a better world model. We show experimentally that this is not always enough: even with a perfect world model—obtained by replacing the world model with the simulator itself, which can predict the outcome of any action sequence with zero error—a planner with a finite sampling budget still often fails to reach the goal. This can happen because planning involves two coupled operations: sampling candidate action sequences and evaluating them. A perfect world model removes prediction error from evaluation, but sampling successful action sequences can still be difficult. Motivated by this limitation, we propose IMWM (Intuition Model + World Model), which pairs the world model with an intuition model trained on demonstrations to recognize promising action sequences. Three lightweight components connect them: (i) Retrieval Initialization guides sampling by starting it from a retrieved demonstration; (ii) Hybrid Cost adds the intuition score to evaluation, alongside the world model's goal loss; (iii) Reliability Gate decides, before planning, whether to enable the intuition model. Experiments on six pixel-based goal-reaching tasks show that IMWM improves on or matches the world-model-only planner, with large gains on OGBench-Cube (93.2%, $+28.5$ percentage points) and Two-Room (99.3%, $+10.0$ points). On OGBench-Scene and DMC-Cheetah, two tasks added after all planner settings had been fixed, it gains 7.5 and 20.5 points, respectively.

1 INTRODUCTION

A learned world model gives an artificial agent a way to predict the consequences of actions and to plan from raw pixels (Ebert et al., 2018; Hafner et al., 2019; 2025; Hansen et al., 2022). Planning with a world model involves two coupled operations: sampling candidate action sequences and evaluating them. The sampling procedure proposes candidate action sequences, while the world model predicts their outcomes. A goal loss scores those predicted outcomes, and the scores guide subsequent sampling and the selection of the final action sequence to execute. When a planner with a latent world model (Maes et al., 2026b; Zhou et al., 2025; Assran et al., 2025) fails, the usual remedy is a better world model. We show that this remedy is not always enough, as illustrated in Figure 1 (left): a world model predicts what an action sequence would do but not which action sequences to try, and a planner can sample only a tiny fraction of a vast space of action sequences.

We conduct an experiment with a "perfect world model" to further illustrate this challenge. We obtain the perfect world model by replacing the world model with the simulator itself, which can predict the outcome of any action sequence with zero error. Yet we find that the planner still fails in many episodes.

Research in cognitive science and computational neuroscience has examined how humans and animals plan. They are thought to form cognitive maps of the relationships within an environment (Tolman, 1948; O'Keefe & Nadel, 1978; Behrens et al., 2018), and hippocampal activity can depict future paths to remembered goals (Pfeiffer & Foster, 2013). Models of motor control and planning pair a forward model, which predicts the consequences of actions, with an inverse model, which maps a desired outcome to actions (Wolpert & Kawato, 1998); with only a few candidate solutions, the Generative Cognitive Map Learner (GCML) of Lin et al. (2026) solves a compositional planning task this way substantially more often than a sampling-based model-predictive controller.

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.

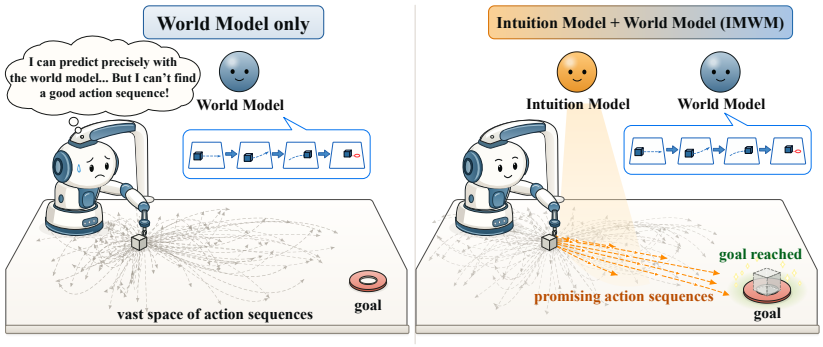


Figure 1: A world model alone is not enough. *Left:* guided only by its world model, a planner must find a good action sequence by trial. *Right:* IMWM adds an intuition model that recalls a demonstrated action sequence to start from and scores candidates alongside the world model.

Motivated by the challenge revealed by the perfect-world-model experiment and inspired by research in cognitive science and computational neuroscience, we propose IMWM (Intuition Model + World Model). IMWM introduces an intuition model trained on demonstrations and uses it alongside a world model for planning. Unlike the world model, which simulates an action sequence's outcome, the intuition model judges directly how promising it is for reaching the goal (Figure 1, right). The world model and the intuition model are analogous to GCML's forward and inverse models, respectively. Three lightweight components connect the two models: (i) *Retrieval Initialization* guides sampling by starting it from a retrieved demonstration; (ii) the *Hybrid Cost* adds the intuition score to evaluation, alongside the world model's goal loss; and (iii) the *Reliability Gate* decides, before planning, whether to enable the intuition model.

Experiments on six pixel-based goal-reaching tasks show that, with the same number of world-model rollouts, IMWM improves on or matches the world-model-only planner, with large gains on OGBench-Cube (93.2%, +28.5 percentage points) and Two-Room (99.3%, +10.0 points). On OGBench-Scene and DMC-Cheetah, two tasks added after all planner settings had been fixed, it gains 7.5 and 20.5 points, respectively. On OGBench-Cube and Two-Room, where we tested the perfect world model, IMWM with its learned world model even succeeds more often than the world-model-only planner with the perfect world model, or with its learned one and a far larger sampling budget.

Contributions. (1) A **diagnosis of latent planning with a perfect world model**, which leaves many failures in place and can even lower success. (2) The **IMWM method**, which joins an intuition model to the world model through three lightweight components. (3) **Empirical gains and analyses on six tasks**, which show where the gains come from and when the gate errs.

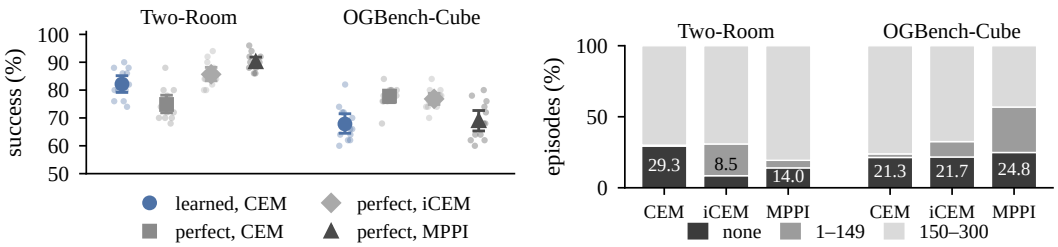
2 A PERFECT WORLD MODEL IS NOT ENOUGH

How much would exact predictions help a latent planner? We test this with a perfect world model.

2.1 PLANNING WITH A LATENT WORLD MODEL

The agent observes only images and must reach the state shown in a goal image o_g, as judged by a task-specific success criterion (Appendix A). A LeWM world model (Maes et al., 2026b), a joint-embedding predictive model trained on each task's demonstrations, maps an image to a latent state with its encoder E and predicts from a latent state and five actions the latent state five steps later.

The planner of LeWM, which we call WM-only, optimizes *action sequences* $\mathbf{a}$ of $H = 25$ actions (often called action chunks) with the cross-entropy method (CEM; Rubinstein, 1999). Actions are standardized with the demonstrations' mean and standard deviation, so the *zero action sequence* repeats the average demonstrated action. At each *planning step*, CEM starts from the proposal $\mathcal{N}(\mathbf{0}, I)$; each of $M = 30$ iterations scores $N = 300$ candidates, the proposal's mean and 299



(a) Success.

(b) Final candidates at the first planning step.

Figure 2: A perfect world model does not make the planner reliable (diagnosis episodes). (a) Success: means, 95% bootstrap CIs and seeds. (b) Perfect world model: episodes by how many of the 300 final candidates of the first planning step end in a success state (numbers: share with none).

samples, by the *goal loss* $\ell(\mathbf{a}) = \|\hat{\mathbf{z}}_{t+H}(\mathbf{a}) - E(o_g)\|^2$ of the predicted final latent state and refits the proposal to the $K = 30$ best. The agent executes all 25 actions of the final refit mean, which is itself never evaluated. An episode ends at the goal or after 50 steps, so it has at most two planning steps.

2.2 Using the simulator as a perfect world model

To separate prediction from the rest of planning, we give WM-only a *perfect world model*: for every candidate, we restore the simulator to the current state, execute the actions and encode the final image $o^\star$, so the goal loss becomes $\|E(o^\star) - E(o_g)\|^2$. This removes prediction error but keeps the latent goal loss, whose embedding distance may misjudge task success. We ran it on OGBench-Cube and Two-Room with 12 *seeds* each; a seed fixes 50 *diagnosis episodes*, drawn from trajectories in no retrieval bank (Appendix D), and the planner's random numbers. We report mean success and paired differences in percentage points (pp) with 95% bootstrap confidence intervals (CIs) over seeds, which omit training variability (one run per model).

The perfect world model does not make the planner reliable, and it does not even always help (Figure 2a). It raises success on OGBench-Cube from 67.8% to 77.8% ($+10.0$ pp, CI $[+7.7, +12.2]$) but lowers it on Two-Room from 82.2% to 74.8% (-7.3 pp, $[-10.2, -4.7]$), a drop we could not explain (Appendix D). With iCEM (Pinneri et al., 2021) or MPPI (Williams et al., 2017) instead of CEM, 10% to 31% of the episodes still fail. On the main evaluation episodes of Section 4, it changes success in the same directions, from 64.7% to 73.7% and from 89.3% to 84.3%.

2.3 Where planning fails

Nearly every failure of CEM with the perfect world model begins with a planning step whose final candidates all end outside a success state (Figure 2b). The simulator records, for each of the 300 candidates of the last iteration, whether it ends in a success state: in 127 of the 133 failures on OGBench-Cube and all 151 on Two-Room, none does at the first planning step, whereas when some final candidate succeeds, so does the episode in 98.7% and 100% of cases. A candidate that reaches the goal and then leaves it counts as a miss. The protocol builds in part of this association, since an episode ends at the goal and a failed first action sequence leaves only one more planning step; the association is weaker for iCEM and MPPI (Appendix D).

The records cannot tell which operation of planning is at fault. As only the last iteration is recorded, a successful action sequence may never have been sampled, or it may have been sampled and then lost because the goal loss favored unsuccessful candidates or the refit proposal moved away from it. A constructed worst case shows that a finite sampling budget can fail even with a cost that identifies success exactly (Theorem 1, Appendix B), although it does not explain the observed failures.

3 IMWM: Pairing an intuition model with a world model

IMWM guides both operations of planning with an intuition model that recalls and judges promising action sequences, which the world model then checks (Figure 3). The world model, CEM and the

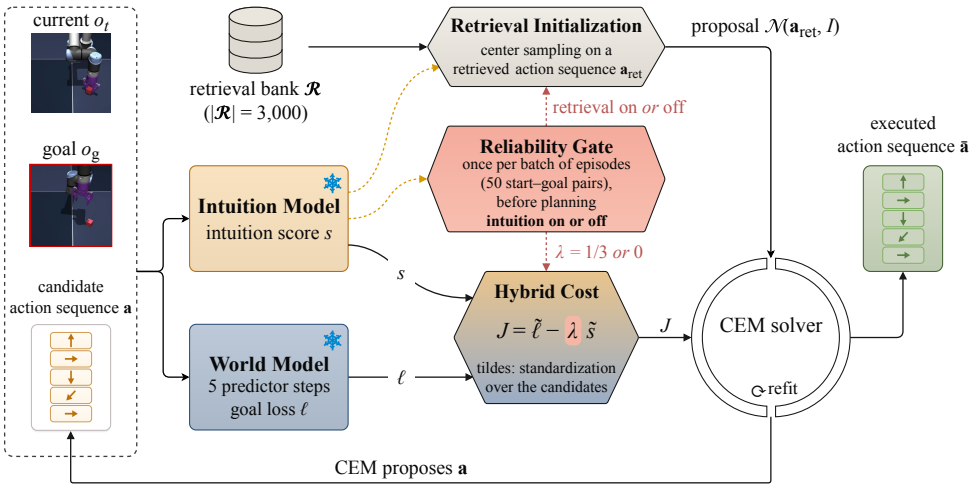


Figure 3: Overview of IMWM. All three components use the intuition model: the Hybrid Cost J combines its score s with the goal loss ℓ; Retrieval Initialization (dashed) retrieves with its encoder the action sequence $\mathbf{a}_{\text{ret}}$ around which CEM starts; the Reliability Gate (dashed) uses its scores to switch the other two on or off. Drawn action sequences have five actions; ours have 25.

rollout budget of WM-only stay unchanged: both planners roll out $N \times M = 9{,}000$ candidates per planning step, and IMWM adds about 7% to the median planning time (Appendix C).

3.1 The intuition model: scoring action sequences without simulation

The intuition model judges how promising an action sequence $\mathbf{a}$ is for reaching the goal observation o_g from the current observation o_t (Figure 4). An observation encoder φ and an MLP map o_t and o_g to a query $\mathbf{q}$, an action encoder maps $\mathbf{a}$ to a key $\mathbf{k}$, and the intuition score is $s(o_t, o_g, \mathbf{a}) = \mathbf{q}^\top \mathbf{k}/\tau$ with $\tau = \sqrt{192}$, for embeddings of dimension 192.

The intuition model is trained contrastively on demonstration windows $(o, o', \mathbf{a})$, each an observation, the observation h steps later and the h actions in between: in a batch of $B = 24$ windows from different trajectories, each start–goal pair should score its own action sequence above the others' (Figure 5), the InfoNCE objective (van den Oord et al., 2018)

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp s(o_i, o_i', \mathbf{a}_i)}{\sum_{j=1}^{B} \exp s(o_i, o_i', \mathbf{a}_j)}, \tag{1}$$

averaged over $h \in \{5, 10, 20, 25\}$. A high score thus means that, in the demonstrations, action sequences like $\mathbf{a}$ tend to lead from observations like o_t to observations like o_g. One intuition model is trained per task and kept fixed (Appendix C). Like GCML's inverse model, it complements the world model: it learns which action sequences connect a start to a goal, not how each action changes the state, and scores a whole action sequence without accumulating prediction errors.

3.2 Retrieval Initialization: starting sampling from a demonstration

Retrieval Initialization starts sampling from a demonstrated action sequence instead of WM-only's proposal $\mathcal{N}(\mathbf{0}, I)$, which carries no information about the goal. From a bank $\mathcal{R}$ of 3,000 demonstration windows $(o^i, o^i_g, \mathbf{a}^i)$ of 25 steps, which excludes the trajectories of the episodes evaluated on the same seed so that no episode finds its own demonstration, it retrieves at each planning step the window whose start and goal look most like the current observation and goal to the intuition model's encoder,

$$\mathbf{a}_{\text{ret}} = \mathbf{a}^{i^\star}, \qquad i^\star = \arg\max_i \cos\big([\varphi(o_t); \varphi(o_g)],\ [\varphi(o^i); \varphi(o^i_g)]\big), \tag{2}$$

and samples the first candidates from $\mathcal{N}(\mathbf{a}_{\text{ret}}, I)$, whose mean $\mathbf{a}_{\text{ret}}$ is itself one of them.

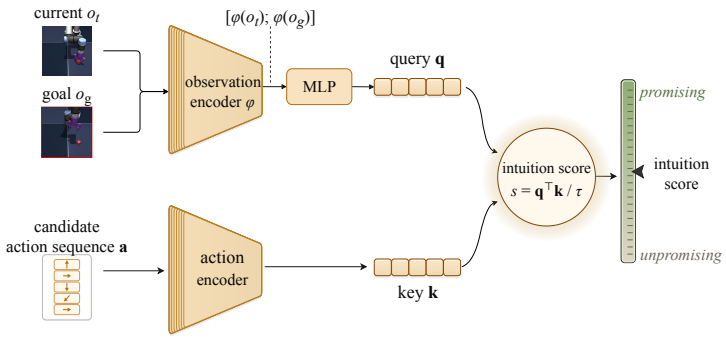


Figure 4: Scoring a candidate: the observation encoder φ and an MLP turn the current and goal observations into a query $\mathbf{q}$, an action encoder turns the action sequence into a key $\mathbf{k}$, and their scaled inner product is the intuition score s; the dotted branch serves retrieval (Section 3.2).

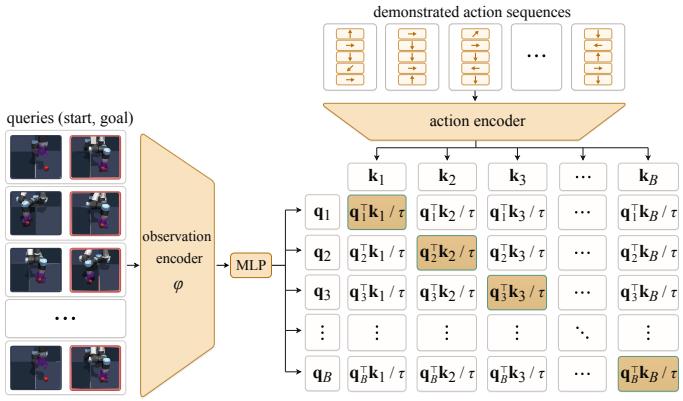


Figure 5: Contrastive training: in a batch of B demonstration windows, the score of each start–goal pair with its own action sequence (diagonal) should exceed its scores with the others.

3.3 Hybrid Cost: Combining the Goal Loss and the Intuition Score

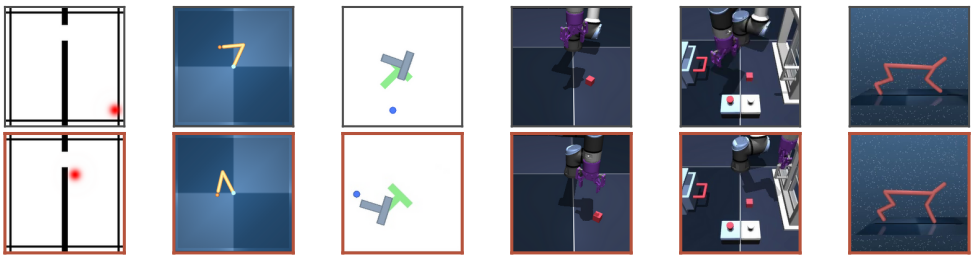
The Hybrid Cost rewards a high intuition score next to a low goal loss, $J(\mathbf{a}) = \tilde{\ell}(\mathbf{a}) - \lambda\, \tilde{s}(\mathbf{a})$, where the tilde denotes standardization over the N candidates of the current CEM iteration, so one weight suffices. We fixed $\lambda = 1/3$ before the main evaluation (Appendix H); $\lambda = 0$ gives the goal loss.

3.4 Reliability Gate: Deciding Whether Intuition Guides Planning

Because an intuition model can be wrong about a task, the Reliability Gate decides, once for each batch of episodes (the 50 start–goal pairs of a seed) and before any action, whether to use the intuition model (per-episode decisions did worse; Appendix I). It checks whether the intuition model scores its retrieved demonstration above most random action sequences: for each pair (o_0^j, o_g^j), it scores $\mathbf{a}_{\text{ret}}^j$ and 300 random action sequences from $\mathcal{N}(\mathbf{0}, I)$ and computes the *margin* m_j and the *reliability score* r,

$$m_j = \frac{s(o_0^j, o_g^j, \mathbf{a}_{\text{ret}}^j) - Q_{95}^j}{\text{MAD}^j + 10^{-6}}, \qquad r = \text{median}_{j=1,\ldots,50}\ m_j, \tag{3}$$

where Q_{95}^j and MAD^j are the 95th percentile and the median absolute deviation of the random scores. If $r > \eta = 0.05$, IMWM uses Retrieval Initialization and the Hybrid Cost; otherwise it plans exactly as WM-only. The gate uses no outcome when it decides, but its threshold was fitted on 24 runs of an earlier evaluation grid (six seeds per *development task*, Section 4.1) to reproduce a per-task assignment derived from development results (on for OGBench-Cube, Push-T and Two-Room, off for Reacher), so only the two tasks added later test it independently (Appendix C).



(a) Two-Room (b) Reacher (c) Push-T (d) OGBench-Cube (e) OGBench-Scene (f) DMC-Cheetah

Figure 6: One episode per task: start (top), goal (bottom, red frame); (a)–(d): development tasks.

4 EXPERIMENTS

4.1 TASKS AND PROTOCOL

We use six goal-reaching tasks from the stable-worldmodel platform (Maes et al., 2026a): navigation (Two-Room), pose matching (Reacher, DMC-Cheetah), block pushing (Push-T) and robot-arm manipulation (OGBench-Cube, OGBench-Scene; Figure 6, Appendix A). Each task's trajectories, collected by random policies for Reacher and DMC-Cheetah, are our *demonstrations*: every segment demonstrates how to reach the observation at its end. An episode starts from a demonstration state, with the image 25 steps later as its goal and 50 steps to reach it.

We designed IMWM on the four development tasks and then added two tasks with all planner and gate settings unchanged: OGBench-Scene after a favorable pilot, and DMC-Cheetah because we expected the intuition model to hurt there. Each task has 12 seeds of 50 episodes, shared by the compared methods (Table 1). As in the platform's protocol, the world model and, in part, the intuition model are trained on trajectories that include the evaluated ones; retrained without them, IMWM still improves on all three tasks tested (Appendix J).

4.2 MAIN RESULTS ON SIX TASKS

IMWM improves on or matches WM-only on all six tasks (Table 2, Figure 7). It gains 28.5 pp on OGBench-Cube and 10.0 pp on Two-Room, on every seed. On Push-T, the gain ($+0.8$ pp, CI $[-1.5, +3.2]$) is inconclusive, and 48 further seeds show equivalence within ± 2 pp (Appendix E). On Reacher, the gate switches the intuition model off on every seed, so the planners coincide. On the added tasks, IMWM gains 7.5 pp on OGBench-Scene, an exploratory result given its selection, and 20.5 pp on DMC-Cheetah, where we had predicted harm: random-policy demonstrations do not by themselves make the intuition model harmful, and whether the gate would switch off an intuition model that hurts on a new task remains untested (Appendix F).

On the main evaluation episodes, IMWM also outperforms the planner with the perfect world model (73.7% and 84.3%, Section 2.2): with its learned world model, IMWM exceeds it by 19.5 pp (CI $[+15.8, +22.8]$) on OGBench-Cube and 15.0 pp ($[+12.5, +17.7]$) on Two-Room and succeeds more often on every seed, in a comparison made after both runs. Both planners share the demonstration-trained encoder of the goal loss; IMWM also draws on demonstrations through its intuition model and

Table 1: Sets of evaluation episodes. Planner comparisons are paired within a set; the diagnosis episodes come from trajectories outside all retrieval banks.

Episode set	Seeds $\times$ episodes, tasks	Used in
main evaluation episodes	12×50, all six tasks	Sections 2.2, 4.2–4.5
diagnosis episodes	12×50, OGBench-Cube and Two-Room	Sections 2.2, 2.3, 4.3
disjoint-split episodes	12×50, three tasks	Appendix J
Push-T equivalence study	48×100, Push-T	Section 4.2, Appendix E
earlier runs	development and follow-up studies	Appendices C, F, I

Table 2: Success (%) over 12 seeds of 50 episodes. Δ: mean paired difference (95% bootstrap CI); bold: CI excludes zero; seeds +/=/−: IMWM better, equal, worse; gate on: seeds with intuition on.

Task	WM-only	IMWM	Δ (pp) [95% CI]	seeds +/=/−	gate on
Development tasks					
Two-Room	89.3	**99.3**	+10.0 [+8.2, +12.0]	12/0/0	12
Reacher	82.3	82.3	0.0 [0.0, 0.0]	0/12/0	0
Push-T	93.3	94.2	+0.8 [−1.5, +3.2]	5/4/3	12
OGBench-Cube	64.7	**93.2**	+28.5 [+25.8, +31.2]	12/0/0	12
Added tasks					
OGBench-Scene	71.7	**79.2**	+7.5 [+3.8, +11.0]	10/1/1	12
DMC-Cheetah	62.5	**83.0**	+20.5 [+16.5, +24.2]	12/0/0	12

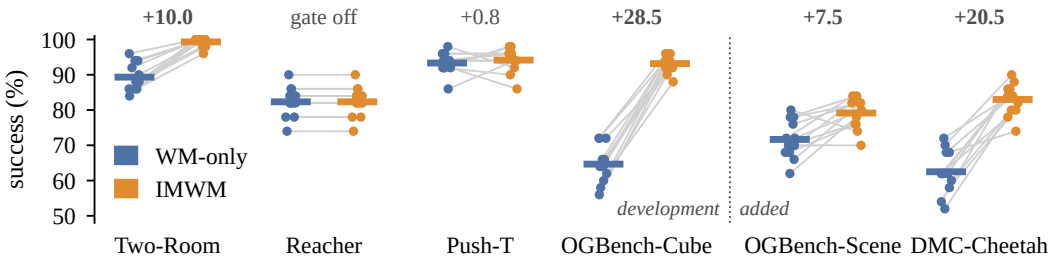


Figure 7: Success on each seed; lines connect the planners on a seed, bars mark means, numbers give mean gains (pp; bold: the 95% CI excludes zero).

bank. For this planner on these tasks, at the default budget, adding demonstration-guided sampling and evaluation improves success more than replacing learned dynamics with exact simulation.

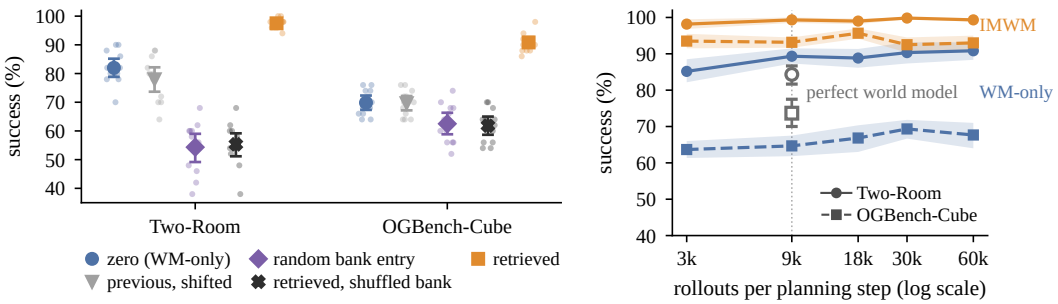
4.3 Retrieval Initialization: A Better Start or More Samples?

On OGBench-Cube and Two-Room, retrieval with the goal loss alone recovers most of IMWM's gain. Table 3 crosses where sampling starts with how candidates are evaluated, bypassing the gate so that the intuition model is used wherever a configuration includes it (*forced on*). With the goal loss alone, the retrieved start raises OGBench-Cube from 64.7% to 90.2% and helps on Two-Room and Push-T, but lowers Reacher by 16.5 pp.

Among the starting points we tried on the diagnosis episodes, only a demonstration that matches the current observation and goal helps (Figure 8a; Appendix G): neither the previous plan shifted by five actions, here reusing already executed actions, nor the action sequence of a random bank entry or one retrieved from a bank with shuffled action sequences beats the zero action sequence, and retrieval beats them by 20.3 to 35.8 pp on average over the two tasks. On these episodes, the retrieved start with the learned world model and goal loss alone even beats the planner with the perfect world model, by 13.0 pp (CI [+11.5, +14.7]) on OGBench-Cube and 22.7 pp ([+19.3, +25.5]) on Two-Room (post hoc).

Table 3: Starting point and evaluation (intuition model forced on): success (%), shaded by value, and difference to WM-only (first row; CIs in Table 9). †Weight study (Appendix H); —: not run.

Start	Evaluation	Two-Room	Reacher	Push-T	OGBench-Cube
zero	goal loss	89.3	82.3	93.3	64.7
zero	intuition score	96.7 (+7.3)	60.5 (−21.8)	49.3 (−44.0)	72.2 (+7.5)
zero	Hybrid Cost	97.3 (+8.0)	—	93.8 (+0.5)	75.8 (+11.2)
retrieved	goal loss	97.5 (+8.2)	65.8 (−16.5)	95.7 (+2.3)	90.2 (+25.5)
retrieved	intuition score	97.0 (+7.7)	53.8 (−28.5)	57.2 (−36.2)	82.5 (+17.8)
retrieved	Hybrid Cost	99.3 (+10.0)	70.0† (−12.3)	94.2 (+0.8)	93.2 (+28.5)



(a) Where sampling starts (goal loss only). (b) Larger sampling budgets.

Figure 8: Other starting points, larger budgets and the perfect world model. (a) Diagnosis episodes; the zero start is a separate run from Figure 2a with other random numbers (82.0% and 69.8% against 82.2% and 67.8% there). (b) Main evaluation episodes; dotted line: default budget; hollow markers: perfect world model. Means with 95% bootstrap CIs.

More samples do not close the gap (Figure 8b): with 3,000 to 60,000 rollouts per planning step (varying candidates and iterations), WM-only reaches at most 69.3% on OGBench-Cube and 90.8% on Two-Room, whereas IMWM stays above 92% and 98%, even with a third of its default budget.

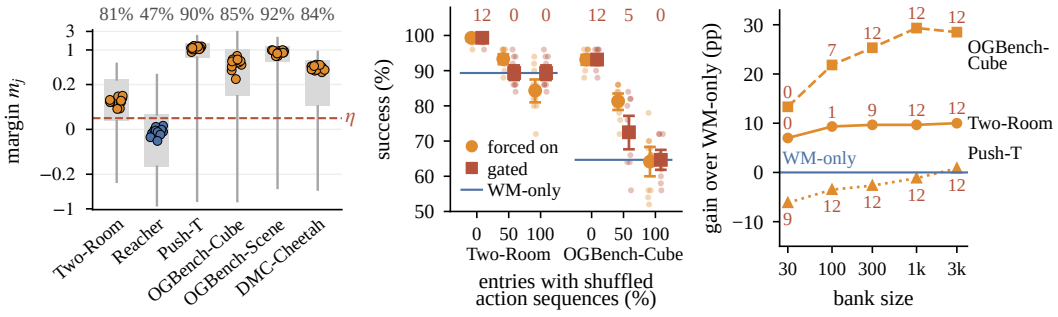
4.4 Hybrid Cost: How should candidates be evaluated?

The intuition score complements the goal loss in evaluation but does not replace it (Table 3). Without retrieval, the Hybrid Cost raises success over the goal loss by 11.2 pp on OGBench-Cube and 8.0 pp on Two-Room; with retrieval, it adds only 3.0 and 1.8 pp, so their gains do not add up. The intuition score alone, without the goal loss, is unreliable: it fails on Push-T and Reacher, and where it helps, it trails the Hybrid Cost. Varying λ from 0.1 to 3 changes the gains on OGBench-Cube and Two-Room by at most 1 pp, but on Push-T, success falls below WM-only's at $\lambda = 1$ and 3 (Appendix H).

4.5 Reliability Gate: When should the planner trust its intuition?

With the default banks, the gate switches the intuition model off only on Reacher, where every configuration that uses it lowers mean success, by 12.3 to 28.5 pp (Table 3): Reacher's reliability scores are -0.05 to 0.02, against 0.09 to 1.27 on the other tasks (Figure 9a), a separation partly built in, as the threshold was calibrated on the development tasks.

The gate also responds to the quality of the bank, but imperfectly (Appendix I). With every entry's action sequence shuffled, the forced intuition model hurts or does not help, and the gate switches it off on every seed; with half shuffled, it errs on the side of caution (Figure 9b). With smaller



(a) What the gate sees. (b) Shuffled banks. (c) Smaller banks (forced on).

Figure 9: The Reliability Gate. (a) Margins m_j of all start–goal pairs (bars: quartiles; whiskers: 5th and 95th percentiles) and seeds' reliability scores r (dots; blue: gate off); linear axis within ± 0.25; top: share of $m_j > 0$. (b), (c) Red: seeds (of 12) with the intuition model on.

banks, it errs in both directions (Figure 9c): with 30 entries, it switches off an intuition model that still gains 13.3 pp (OGBench-Cube) and 7.0 pp (Two-Room), whereas on Push-T, where the forced model loses 6.2 and 3.5 pp with 30 and 100 entries (CIs exclude zero), it stays on for 9 and 12 of 12 seeds. Passing the threshold thus neither guaranteed a benefit nor was necessary for one, echoing a finding about human judgment: how convincing an intuition feels is not a reliable indicator of its accuracy (Kahneman & Klein, 2009). A gate that also checks whether retrieved action sequences reach the goal might avoid both errors.

4.6 Ongoing: comparing planning, replay and learned policies

A pre-specified comparison with replaying the retrieved action sequence and with policies trained on the same demonstrations (goal-conditioned behavior cloning (Lynch et al., 2020), implicit Q-learning (Kostrikov et al., 2022) and, on Push-T, Diffusion Policy (Chi et al., 2023)) is in preparation (Appendix K); we will add its results when available.

5 Related work

World models and planning. Model-based agents plan or learn policies with learned dynamics (Ebert et al., 2018; Hafner et al., 2019; 2025; Chua et al., 2018; Amos et al., 2018; Hansen et al., 2022; 2024). Recent work develops latent representations (Balestriero & LeCun, 2025) and predictive world models (Maes et al., 2026b; Zhou et al., 2025; Assran et al., 2025), or trains models for control (Farahmand et al., 2017; Lambert et al., 2020). We instead change how a planner with a fixed world model samples and evaluates, and isolate prediction error with the true simulator, which PlaNet and iCEM also use (Hafner et al., 2019; Pinneri et al., 2021). Other planners improve sampling (Pinneri et al., 2021; Williams et al., 2017; Bharadhwaj et al., 2020) or guide search with learned priors (Silver et al., 2018; Schrittwieser et al., 2020; Wang & Ba, 2020; Argenson & Dulac-Arnold, 2021); TD-MPC, for example, samples part of its candidates from a learned policy (Hansen et al., 2022; 2024); IMWM instead starts from a retrieved demonstration, adds a contrastive score to evaluation, and can switch both off.

Forward and inverse models in the brain. GCML (Lin et al., 2026) combines cognitive maps, hippocampal sequences toward remembered goals (Pfeiffer & Foster, 2013; Mattar & Daw, 2018) and paired forward and inverse models (Wolpert & Kawato, 1998) and is a key conceptual precedent for IMWM: noisy sampling from its inverse model's estimated action utilities, with the forward model updating the imagined state, yields diverse goal-directed trajectories. IMWM adopts a related division of labor, but trains deep models offline, scores whole action sequences rather than building them action by action, refits CEM's proposal using the Hybrid Cost, and gates the intuition model.

Demonstrations and retrieval. Policies learned from trajectory data act without a world model (Lynch et al., 2020; Ghosh et al., 2021; Kostrikov et al., 2022; Chi et al., 2023; Florence et al., 2022; Rajeswaran et al., 2018); episodic control, retrieval-augmented agents and behavior priors reuse experience (Blundell et al., 2016; Pritzel et al., 2017; Goyal et al., 2022; Schmied et al., 2026; Pertsch et al., 2021; Tirumala et al., 2022; Janner et al., 2022); contrastive reinforcement learning uses a contrastive start–goal–action score as a value function (Eysenbach et al., 2022).

6 Conclusion

A better world model is the usual remedy for a failing latent planner, yet even a perfect one left many failures in place. IMWM adds an intuition model that guides sampling and evaluation unless a reliability check fails. With the same number of rollouts, it improves the planner on four tasks and matches it on two, and on OGBench-Cube and Two-Room it outperforms the planner with the perfect world model or a far larger budget: there, knowing which action sequences are worth trying mattered more than predicting their outcomes exactly. The intuition model worked best alongside the world model, echoing paired forward and inverse models (Wolpert & Kawato, 1998; Lin et al., 2026). As world models improve, which action sequences a planner considers, and when it can trust a learned judgment of them, may increasingly decide whether it succeeds.

AI USE STATEMENT

Generative AI tools were used in this work to help develop the conceptual framework and the formal statements and proofs of Appendix B; to formulate and refine hypotheses and pre-specified decision rules; to design and critique the experimental methodology; to implement the method and evaluation, aggregation and verification scripts; to clean and reformat result data; to draft and edit the text; to assist with figure design; and to find and format references.

No synthetic training or evaluation data were generated: every experiment ran on the benchmark platform. The tables and figures reporting experimental results are generated by scripts directly from the recorded result files, and the numbers quoted in the text were checked against the recorded results. Every pre-specified analysis was fixed before its decision-bearing results existed; pilots and amendments are disclosed in the appendices. The authors reviewed all AI-assisted code, analyses and text and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Section 4.1 and Appendix A describe the tasks and the evaluation protocol; Section 3 and Appendix C describe the method with all fixed settings; Appendices D–K describe the additional studies, including their pre-specified analyses. Theorem 1 is proved in Appendix B. The tables and figures reporting experimental results are generated by scripts from the recorded result files. These scripts, a map from every reported study to its source records, and the code, configurations, checkpoints, retrieval banks and per-seed evaluation logs will be released with the camera-ready version.

ETHICS STATEMENT

This work studies planning in simulated goal-reaching benchmarks with publicly available datasets and involves no human subjects, personal data or deployment in the physical world. We see no ethical concern beyond the general considerations that apply to research on autonomous control.

REFERENCES

Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems 31*, 2018.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

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

Timothy E. J. Behrens, Timothy H. Muller, James C. R. Whittington, Shirley Mark, Alon B. Baram, Kimberly L. Stachenfeld, and Zeb Kurth-Nelson. What is a cognitive map? Organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018. doi: 10.1016/j.neuron.2018.10.002.

Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 277–286. PMLR, 2020.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z. Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

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 XIX*, 2023.

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 31*, 2018.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems 35*, 2022.

Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1486–1494. PMLR, 2017.

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 *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168. PMLR, 2022.

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*, 2021.

Anirudh Goyal, Abram L. Friesen, Théophane Weber, Andrea Banino, Nan Rosemary Ke, Adrià Puigdomènech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, and Charles Blundell. Retrieval-augmented reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7740–7765. PMLR, 2022.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022.

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

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022.

Daniel Kahneman and Gary Klein. Conditions for intuitive expertise: A failure to disagree. *American Psychologist*, 64(6):515–526, 2009. doi: 10.1037/a0016755.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 761–770. PMLR, 2020.

Hui Lin, Yukun Yang, Rong Zhao, Giovanni Pezzulo, and Wolfgang Maass. Neural sampling from cognitive maps enables goal-directed imagination and planning. *Nature Machine Intelligence*, 8 (7):1045–1065, 2026. doi: 10.1038/s42256-026-01254-4.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2020.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation. *arXiv preprint arXiv:2605.21800*, 2026a.

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*, 2026b.

Marcelo G. Mattar and Nathaniel D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11):1609–1617, 2018. doi: 10.1038/s41593-018-0232-z.

John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Clarendon Press, Oxford, 1978.

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

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2021.

Brad E. Pfeiffer and David J. Foster. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74–79, 2013. doi: 10.1038/nature12112.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065. PMLR, 2021.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2827–2836. PMLR, 2017.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems XIV*, 2018.

Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143.

Thomas Schmied, Fabian Paischer, Vihang Prakash Patil, Markus Hofmarcher, Razvan Pascanu, and Sepp Hochreiter. Retrieval-augmented decision transformer: External memory for in-context RL. In *Proceedings of the 4th Conference on Lifelong Learning Agents*, volume 330 of *Proceedings of Machine Learning Research*, pp. 376–417. PMLR, 2026.

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

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

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.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning. *Journal of Machine Learning Research*, 23(221):1–68, 2022.

Edward C. Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948. doi: 10.1037/h0061626.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017. doi: 10.2514/1.G001921.

Daniel M. Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7–8):1317–1329, 1998. doi: 10.1016/S0893-6080(98)00066-5.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

A TASKS, DATASETS AND EVALUATION EPISODES

Table 4 lists the six tasks. All run on the stable-worldmodel platform (Maes et al., 2026a) with its evaluation harness (environments `TwoRoom-v1`, `ReacherDMControl-v0`, `PushT-v1`, `OGBCube-v0`, `OGBScene-v0` and `CheetahDMControl-v0`). Two-Room is the two-room navigation environment of DINO-WM (Zhou et al., 2025); Push-T is the task of Diffusion Policy (Chi et al., 2023), adapted from implicit behavioral cloning (Florence et al., 2022); Reacher and DMC-Cheetah come from the DeepMind Control Suite (Tassa et al., 2018); and OGBench-Cube and OGBench-Scene come from OGBench (Park et al., 2025). In Two-Room, a point agent moves between two rooms through a door; in Reacher, a two-joint arm must match a target pose; in Push-T, the agent pushes a T-shaped block into a target pose; in OGBench-Cube, a robot arm moves a cube to a target position; in OGBench-Scene, a robot arm must bring a cube, two buttons, a drawer and a window into target states; and in DMC-Cheetah, a planar cheetah must match a target pose.

Episodes. An episode starts from a state of a dataset trajectory; its goal is the observation of the same trajectory 25 environment steps later, and the agent has 50 environment steps. The environment checks the task's success criterion after every step, and an episode ends as soon as it is met. A valid start state is any state followed by at least 25 further states of its trajectory. The 50 main evaluation episodes of a seed are valid start states drawn uniformly without replacement, so longer trajectories

are drawn more often;[1] the diagnosis and disjoint-split episodes instead draw a trajectory and then a valid start state within it (Appendices D and J).

Seeds and statistics. A seed fixes the 50 evaluation episodes (trajectory and start index) and the planner's random numbers, which all methods of a study share on that seed; different studies on the same episodes use different random numbers. (The earlier evaluation grid of Appendices C and I set the episodes and the random numbers with separate data and solver seeds.) Each model is trained once per task, so intervals over seeds reflect the sampling of episodes and the planner's randomness, not variation between training runs. Unless stated otherwise, intervals are 95% percentile bootstrap intervals over seeds; pre-specified analyses use the intervals they specified (paired t, simultaneous Welch or two one-sided tests), as noted where they appear. Differences are computed before rounding, so they can differ by 0.1 pp from the difference of two rounded success rates.

Figure 6. The figure shows the first evaluation episode of seed 3 for each development task, rendered from the dataset states, and the first evaluation episode of seed 11 for each added task, taken from the first frame of that episode's recorded rollout video, whose panels show the current observation and the goal image, because no separately rendered start and goal images were stored for the added tasks.

Table 4: Tasks, datasets and success criteria. Two-Room and Push-T positions are in each environment's own coordinate units. The Reacher and DMC-Cheetah datasets were collected by random policies (DMC-Cheetah's by us); the others come with the platform or with OGBench (Park et al., 2025).

Task	Dataset	Success criterion
Two-Room	`tworoom`	agent within 16 units of the goal position
Reacher	`reacher_random`	every joint angle within 0.05 rad of the goal
Push-T	`pusht_expert_train`	agent and block positions within 20 units, block angle within 20°
OGBench-Cube	`cube_single_expert`	cube within 4 cm of its goal position
OGBench-Scene	`ogb_scene_single`	cube, both buttons, drawer and window in their goal states
DMC-Cheetah	`cheetah_qpos_random`	every joint position except the horizontal root within 0.1 of the goal

Two-Room layout. A goal would be unreachable if the dataset had been recorded with a different wall or door layout from the environment's. At the start of an evaluation episode, the Two-Room environment sets only the agent and target positions from the dataset; the wall and door keep the environment's fixed default layout (only `agent.position` and `target.position` are varied at reset). The default door appears in the goal images of all 600 main evaluation episodes and in the first frames of 40 randomly sampled demonstration trajectories, and the acting environment showed the same door as the goal image in all 600 episodes. On the disjoint-split evaluation episodes of Appendix J, replaying the recorded actions through the evaluation harness met the success criterion within 25 steps in all 600 Two-Room episodes. We did not replay the demonstrations on the diagnosis episodes.

B A WORST CASE FOR BLACK-BOX SAMPLING

This appendix formalizes the worst case mentioned in Section 2.3. Let A be the set of action sequences a planner may propose, flattened to vectors and rescaled so that $A = [0, 1]^D$ (actions are bounded, so this is an affine change of coordinates). Let $U_\varepsilon(x)$ be the Euclidean ball of radius ε around x, and V_D the volume of the unit ball in $\mathbb{R}^D$.

Setting. A task has a hidden target action sequence $a^\dagger \in A$; an action sequence succeeds if it lies in $S = U_\varepsilon(a^\dagger) \cap A$. The cost is exact but binary, and zero exactly on the successful set: $f(a) = 0$

[1] The sampler used for the main evaluation episodes could not draw the last valid start state of the whole dataset, a negligible restriction that applies to all methods alike.

if $a \in S$ and $f(a) = c > 0$ otherwise. A *black-box planner* chooses up to n action sequences $x_1, \ldots, x_n$ adaptively, observes $f(x_i)$ for each, and outputs an action sequence x_{out}; it may use any internal randomness but learns about f only through these evaluations. CEM with N candidates per iteration and M iterations is such a planner with $n = NM$: it uses the cost only through the values of the candidates it evaluates.

Theorem 1 (Finite-budget sampling bound)**.** *Let $a^\dagger$ be drawn uniformly from A, independently of the planner's randomness. For every black-box planner with at most n evaluations,*

$$\Pr\left[\exists\, i \in \{1, \ldots, n, \text{out}\} : x_i \in S\right] \le (n+1)\, V_D\, \varepsilon^D.$$

Consequently, for every such planner and every $\delta \in (0, 1)$, there is a task (a radius ε and a target $a^\dagger$, which may depend on the planner) on which the planner succeeds with probability at most δ.

Proof. Fix the planner's randomness ω. Until an evaluation returns 0, every evaluation returns c, so the planner's choices $x_1^\omega, \ldots, x_n^\omega, x_{\text{out}}^\omega$ along this constant transcript are fixed by ω. The planner succeeds only if $a^\dagger$ lies within ε of one of these $n+1$ points, which has probability at most $(n+1)\, V_D\, \varepsilon^D$ under the uniform $a^\dagger$. Averaging over ω gives the first claim. Choosing $\varepsilon < \left(\delta/((n+1)V_D)\right)^{1/D}$ and averaging over $a^\dagger$ shows that some fixed target meets the second claim. □

What the theorem shows. The bound uses two assumptions that the experiments do not share: every unsuccessful action sequence has the same cost, so failed evaluations carry no information about where S lies, and the target is placed adversarially (or uniformly at random). With a cost that decreases towards the goal, a planner can do much better. The theorem therefore shows only that a planner with a finite sampling budget can fail with an exact cost; it does not explain the failures observed in Section 2.

What retrieval could add. Conversely, retrieval could guarantee success under strong assumptions. Suppose a bank contains an action sequence $\mathbf{a}_{\text{ret}} \in A$ within $\varepsilon/2$ of $a^\dagger$ and retrieval selects it. (i) A planner that evaluates $\mathbf{a}_{\text{ret}}$ and returns the lowest-cost action sequence it has evaluated then succeeds with probability one; retaining a successful action sequence while returning a different, unevaluated one gives no such guarantee. (ii) Consider one iteration of elite-mean CEM in which one of the N candidates is $\mathbf{a}_{\text{ret}}$ itself and the other $N-1$ are independent draws from $\mathcal{N}(\mathbf{a}_{\text{ret}}, \sigma^2 I)$, each projected onto A (clipped to the cube). Projection onto the convex set A does not increase the distance to $a^\dagger \in A$, so every draw within $\varepsilon/2$ of $\mathbf{a}_{\text{ret}}$ lands, after projection, within ε of $a^\dagger$, that is, in S. Hence at least K candidates succeed with probability at least $\Pr[1 + \text{Binomial}(N-1, p) \ge K]$, where $p = \Pr\left[\chi^2_D \le \varepsilon^2/(4\sigma^2)\right]$. In that event all K elites have cost 0 and lie in the convex set $S = U_\varepsilon(a^\dagger) \cap A$, and so does their mean. These statements assume coverage (a bank entry within $\varepsilon/2$ of the target) and correct retrieval (retrieval selects that entry), which no experiment in this paper measures, and they hold for one iteration or with best-so-far retention. Statement (ii) also assumes that candidates are clipped to A, which the deployed CEM does not do during planning. We have no bound for the deployed planner, which runs 30 iterations and executes the final mean.

C Implementation details

World model. Each task's LeWM world model (Maes et al., 2026b), a joint-embedding predictive model learned end to end from pixels, is trained from scratch for 30 epochs with the platform's training pipeline on the task's demonstrations, and the final epoch is used. It encodes 224×224 images into latent states; the predictor maps a latent state and a block of five actions to the latent state five environment steps later. It is kept fixed during planning.

Intuition model. The observation encoder φ has the architecture of the world model's encoder. It is initialized from the LeWM weights released for the development tasks, which were trained on data that include the evaluation episodes, and from the task's own world model on the added tasks, and it is trained jointly with the rest of the intuition model; on the development tasks, the intuition model's encoder therefore starts from the released weights, whereas the world model used for planning is our own training run. This difference arose during development, and we did not test whether it matters. The query MLP is a two-layer perceptron (hidden width 256, GELU) that maps the concatenated

embeddings $[\varphi(o); \varphi(o')]$ to a 192-dimensional query. The action encoder applies a one-dimensional convolution (kernel 3) and a GRU to the individual actions of the action sequence, in the standardized coordinates in which CEM samples them, followed by layer normalization and a linear map to a 192-dimensional key. The score is $\mathbf{q}^\top \mathbf{k}/\sqrt{192}$.

Training of the intuition model. A training batch holds $B = 24$ windows from different trajectories. Each window's start observation is paired with the observations 5, 10, 20 and 25 steps later and with the corresponding first 5, 10, 20 and 25 actions, and the InfoNCE losses of Eq. (1) for the four lengths, each with in-batch negatives, are averaged; these lengths were fixed during development and not ablated. Windows come from the first 100 steps of 1,500 randomly chosen demonstration trajectories (a limit of the training code). Training uses AdamW (learning rate 10^{-4}, weight decay 10^{-4}) with a cosine learning-rate schedule for 12,000 steps; the final checkpoint is used and no checkpoint is selected on validation data. On Two-Room and the two added tasks, training adds a forward contrastive (CPC) loss on the embeddings of consecutive observations, weighted by gradient-norm balancing and used only to regularize the encoder; this difference in configuration was not ablated.

Overlap with evaluated trajectories. Because the intuition model's training trajectories are a random subset of all demonstrations, they include some evaluated source trajectories. An earlier audit of the training split counted, within each seed, the distinct source trajectories of the evaluated episodes and pooled these counts over seeds. On the main evaluation of the development tasks, 9.5–17.4% of them were training trajectories (57 of 599 on Push-T, 84 of 600 on Two-Room, and 104 of 599 on both OGBench-Cube and Reacher), close to the share of all trajectories used for training (15%, or 8% on Push-T). For the added tasks, the audit covered only an earlier six-seed grid (48 of 299 on each). Appendix J retrains the models without this overlap.

Retrieval bank. For each task and seed, the bank holds 3,000 windows of 25 environment steps drawn uniformly from the demonstrations, excluding the trajectories of all episodes evaluated on that seed. Each entry stores the normalized concatenation $[\varphi(o^i); \varphi(o^i_g)]/\|[\varphi(o^i); \varphi(o^i_g)]\|$ and the action sequence $\mathbf{a}^i$. Retrieval returns the entry with the highest cosine similarity to the current pair (Eq. 2), one nearest-neighbor lookup among 3,000 vectors per planning step. The smaller and shuffled banks of Appendix I are built the same way: each excludes the trajectories evaluated on its own seed, but may contain trajectories evaluated on other seeds.

CEM. Candidates are sampled in the space of standardized actions, flattened over the action sequence, from a Gaussian with diagonal covariance and initial standard deviation 1. At every iteration, one candidate slot holds the current mean; the $K = 30$ candidates with the lowest cost refit the mean and the per-dimension standard deviation. After $M = 30$ iterations, the mean is converted back to physical actions and executed; out-of-range actions are clipped by the environment. When the intuition score alone evaluates candidates (Table 3), CEM keeps the K candidates with the highest scores.

Reliability Gate. The gate runs once per batch of episodes (the 50 episodes of a seed), before planning, on their start and goal observations (Eq. 3): 300 random action sequences per pair from $\mathcal{N}(\mathbf{0}, I)$, the 95th percentile and median absolute deviation of their scores, and the median over pairs. The gate chooses between two configurations: intuition on, with $\lambda = 1/3$ and retrieval, or off.

Calibration of the gate. The threshold $\eta = 0.05$ was chosen on a separate calibration pool of 24 runs on the development tasks (six seeds per task from an earlier evaluation grid, whose seeds select episodes differently from the main evaluation), before the main evaluation, and was not changed afterwards. The threshold was set so that the gate's decisions on this pool matched a per-task assignment fixed beforehand: intuition on for OGBench-Cube, Push-T and Two-Room, off for Reacher. The weight was selected separately (Appendix H). During development, a comparison of fixed configurations on 48 runs of the four development tasks (twelve per task: data seeds 3, 5 and 7, which select the episodes, crossed with solver seeds 42, 1 and 2, which set the planner's random numbers, and data seeds 9, 11 and 13 with solver seed 42), which include the 24 calibration runs, led us to keep $\lambda = 1/3$ and to remove a third configuration, with $\lambda = 10$, from the gate's choices. That the threshold reproduces the assignment on this pool confirms only the calibration; it does not validate

the gate independently. The assignment itself was derived from development results on these tasks. Because the calibration pool and the main evaluation use the same four tasks, the development-task results are not independent of the gate's calibration; the added tasks are.

Timing. Planning-call latency was measured in a separate controlled study on the four development tasks, on one otherwise idle NVIDIA RTX 5880 Ada GPU with batch size 1. The medians were 0.364 s for WM-only, 0.369 s with retrieval only, 0.382 s with the Hybrid Cost only, and 0.388 s for IMWM; the planner with the intuition model switched off took the same time as WM-only. Retrieval alone adds 1.4%, the Hybrid Cost alone 4.9% and IMWM about 7% to the median time of WM-only. These times exclude training, bank construction and the gate's one-time pass.

D THE PERFECT-WORLD-MODEL TEST: IMPLEMENTATION AND RECORDS

Implementation. For every candidate action sequence, the simulator is restored to the state of the current planning step, the 25 actions are executed, and the rendered image is encoded with the world model's encoder. At the last CEM iteration, the simulator also reports whether each candidate's final state satisfies the task's success criterion; earlier iterations are not logged. On OGBench-Cube, one seed of 50 episodes took 4.3 to 7.4 hours of wall-clock time, depending on the sampling method (4.6 to 6.3 hours with CEM), with up to three runs sharing a GPU.

Episodes. The study uses 12 seeds of 50 episodes per task, allocated by a fixed procedure before any run: trajectories are drawn at random, excluding every trajectory that appears in any retrieval bank of the task, and each gets a start index drawn uniformly among its valid start states (Appendix A). Every episode runs in its own simulator, one at a time, and all arms of the study use the same episodes; the starting-point comparison of Section 4.3 uses them too, which is why they exclude every bank trajectory. On these episodes, WM-only succeeds less often on Two-Room than on the main evaluation episodes (82.2% instead of 89.3%); the two sets were drawn by different procedures, we did not investigate the difference, and we compare planners only on the same episodes. The planners with the learned and the perfect world model use common random-number streams for corresponding planning steps, but their candidates can differ after the first iteration, because each planner refits its proposal to its own lowest-goal-loss candidates. Push-T is excluded (below), and Reacher and the added tasks were not part of the study.

The main evaluation episodes. On the main evaluation episodes, the perfect world model changes success in the same directions as on the diagnosis episodes. It raises OGBench-Cube from 64.7% to 73.7% ($+9.0$ pp, CI $[+6.2, +11.8]$, higher on 11 of 12 seeds) and lowers Two-Room from 89.3% to 84.3% (-5.0 pp, $[-6.8, -3.0]$, lower on 9 seeds and higher on none). These results come from a rerun with corrected code on the 12 seeds of 50 main evaluation episodes of the WM-only runs in Table 2, paired seed by seed; every record carries its episode's index, and all 24 runs passed a check of these indices. The rerun's records of the final candidates, pooled over first and later planning steps, show no successful final candidate in 39.7% of the planning steps on OGBench-Cube and 28.6% on Two-Room; when at least one final candidate succeeds, the one with the lowest goal loss is successful in 94.5% and 99.0% of these planning steps.

The fault in the first run. Our first run on the main evaluation episodes had a fault in its later planning steps, and we do not report it. At a planning step, the platform passes only the episodes that still need a plan to the planner, and the perfect world model looked up each episode's simulator by its position in that shortened list instead of by its episode index. At the first planning step all 50 episodes are passed in order, so the lookup is correct; at the second, the candidates of an unfinished episode were simulated in the simulator of whichever episode held the same position. The fix passes each episode's index along with its observations, and a regression test that plans two steps for a non-contiguous set of unfinished episodes fails on the old code and passes on the fixed code. The diagnosis episodes were not affected, because each of them ran in its own simulator.

Post hoc comparisons. Section 4.2 compares IMWM with this rerun on the main evaluation episodes, and Section 4.3 compares the retrieved starting point with the perfect world model on the diagnosis episodes. We made both comparisons after the runs; they were not part of any pre-specified

analysis, and each pairs two arms that share their episodes and uses a descriptive 95% paired bootstrap interval over seeds.

Final-candidate records. Table 5 gives the records behind Section 2.3 for all three sampling methods. A candidate reaches the goal only if it ends in a success state; one that passes through a success state and leaves it counts as a miss, although executing it would end the episode successfully. The records do not show how often this happens. With CEM, in nearly all episodes whose first planning step has a successful final candidate, at least half of the candidates succeed (76.2% and 70.0% of all episodes on OGBench-Cube and Two-Room). iCEM and MPPI end the first planning step without a successful candidate in 21.7% and 24.8% of the OGBench-Cube episodes and in 8.5% and 14.0% of the Two-Room episodes, and their association with failure is weaker: with iCEM on Two-Room, for example, only 41 of 86 failures begin this way. Part of the association follows from the protocol: an episode whose first action sequence succeeds ends there, and a failed one gets a single further planning step, from the state reached by executing the planner's first action sequence. Later planning steps therefore start from states reached by the planner rather than from a demonstration's state. We did not replay the demonstrations on the diagnosis episodes, so reachability from every restored start remains unverified.

Table 5: Perfect world model on the diagnosis episodes (600 episodes per task and sampling method): success, and how many of the 300 last-iteration candidates end in a success state at the first planning step and at later planning steps. "Failures with none first" counts the failed episodes whose first planning step had no successful candidate; "steps" counts second planning steps (at most one per episode).

			first planning step (% of 600)			failures	later planning steps	
Task	method	success (%)	none	1–149	150–300	with none first	steps	none (%)
Two-Room	CEM	74.8	29.3	0.7	70.0	151 / 151	180	82.2
	iCEM	85.7	8.5	22.3	69.2	41 / 86	112	62.5
	MPPI	90.2	14.0	5.3	80.7	59 / 59	85	69.4
OGBench-Cube	CEM	77.8	21.3	2.5	76.2	127 / 133	137	97.1
	iCEM	76.8	21.7	10.8	67.5	111 / 139	202	56.4
	MPPI	69.0	24.8	32.0	43.2	143 / 186	191	94.2

Push-T. A dedicated test of Push-T's state restoration found that loading dataset start states through the evaluation adapter diverged from the reference trajectory in 7 of 10 replay cases and 5 of 10 candidate-rollout cases, so the evaluation adapter cannot restore Push-T's simulator state exactly. We therefore did not run the perfect world model on Push-T. For the same reason, the demonstrated actions of a Push-T episode need not reproduce its goal; this affects all planners in the main evaluation alike.

Other sampling methods. iCEM samples temporally correlated noise and carries elite candidates over to the next iteration (Pinneri et al., 2021); MPPI averages candidates with weights that decay exponentially with their cost instead of refitting a Gaussian to the best ones (Williams et al., 2017). Both use the same budget as CEM. We did not run them with the learned world model, so their numbers compare sampling methods under exact dynamics only. iCEM uses colored noise with exponent 2.0, keeps 5 elites across iterations, applies a momentum of 0.1 to the mean, returns the mean and clips candidates to the action range. The platform's MPPI keeps the 30 lowest-cost candidates and averages them with weights proportional to $\exp(-(\ell - \ell_{\min})/0.5)$, where ℓ is the goal loss and $\ell_{\min}$ its minimum over the candidates; it keeps the sampling standard deviation fixed and does not clip. Relative to CEM with the perfect world model on the same seeds, iCEM changes success by $+10.8$ pp (95% bootstrap CI $[+7.5, +13.8]$) on Two-Room and -1.0 pp ($[-3.3, +1.2]$) on OGBench-Cube, and MPPI by $+15.3$ pp ($[+12.2, +17.8]$) and -8.8 pp ($[-11.3, -6.2]$).

Executing the best final candidate. We do not know why CEM does worse on Two-Room with the perfect world model than with the learned one. One candidate explanation is that the elite candidates split between distinct solutions, so that their mean, which the agent executes, reaches neither. To test it on the diagnosis episodes, the planner executed the lowest-cost candidate of the

last iteration instead of the refit mean. With the perfect world model, this changed Two-Room success by $+1.33$ pp (pre-specified 95% paired t interval $[-0.13, +2.80]$; a bootstrap over seeds gives $[+0.0, +2.5]$) and OGBench-Cube success by -0.2 pp; with the learned world model, by -1.0 (Two-Room) and -1.7 pp (OGBench-Cube). The pre-specified criterion for confirming the explanation was a Two-Room recovery of at least 3.835 pp with an interval excluding zero; it was not met. The criterion was derived before the run from the Two-Room deficit measured in the faulty run, which we no longer report, so it has no validated basis; we keep it as it was specified.

E PUSH-T EQUIVALENCE STUDY

Because the 12-seed Push-T comparison is inconclusive, we ran a separate study with 48 pre-specified seeds of 100 paired episodes each, with the number of seeds and episodes chosen by a power analysis beforehand. The primary analysis is a paired t test over the 48 seed-level differences with an equivalence margin of ± 2 pp. The mean difference is $+0.125$ pp (standard deviation 2.51 pp). The 90% interval $[-0.48, +0.73]$ lies inside $(-2, +2)$, which establishes equivalence at level 0.05 by two one-sided tests, and the 95% interval $[-0.60, +0.85]$ contains zero, so superiority is not established. A percentile bootstrap ($[-0.46, +0.71]$ at 90%) and a restricted wild-cluster test agree. The result concerns these 48 seeds and does not replace the 12-seed comparison of Table 2.

F ADDED TASKS: SELECTION AND HISTORY

DMC-Cheetah. Cheetah's demonstrations come from a random policy, like Reacher's, and we selected it expecting the intuition model to hurt. Before any intuition model or bank existed for Cheetah (a world-model-only pilot and a smoke-test training run preceded it), we fixed four criteria: WM-only must beat a random policy by at least 10 pp; forcing the intuition model on must lose at least 5 pp with $p < 0.05$ (harm); the gated planner must then beat the forced one by at least 5 pp; and the gate must turn the intuition model off. On an earlier six-seed grid, WM-only beat the random policy by 52 pp, the forced intuition model gained $+19.3$ pp instead of losing, and the gate turned it on for every seed. The harm prediction therefore failed in the opposite direction. The main 12-seed evaluation reproduces both the direction and the gate's decisions. The Cheetah world model was trained with mixed precision and two numerical guards in its regularizer, which are exact no-ops for in-range values, after a first run diverged.

OGBench-Scene. Scene had no task-specific criteria fixed in advance. It was added after a single-seed pilot in which IMWM succeeded in 84% and WM-only in 80% of episodes, so its selection was favorable and its result is exploratory. A first six-seed evaluation revealed a caching defect that could score an environment with another environment's embeddings; after the fix, validated against recomputing every embedding, the affected runs were repeated and the gain grew from $+4.3$ to $+6.3$ pp. The defect was in the planner's cache of intuition-model embeddings, which every IMWM run uses, and every run reported in this paper was made with the fix. Five seeds of this 12-seed evaluation re-select episodes of that earlier grid, so it is a partly overlapping replication rather than an independent sample.

G STARTING POINTS AND SAMPLING BUDGETS

Other starting points. To test whether other starting points would do as well as a retrieved demonstration, we compare Retrieval Initialization with three other starting points on the diagnosis episodes, with the goal loss as the only cost (Figure 8a). The first is the previous planning step's action sequence shifted by one block of five actions, with a block of zeros appended; at an episode's first planning step it is the zero action sequence. This imitates the usual warm start of receding-horizon control, which reuses the unexecuted rest of the previous plan. Because our agent executes the whole action sequence, it here re-proposes actions that were already executed, and because an episode has at most two planning steps, it differs from the zero action sequence only at the second one. It tests whether this default warm start of receding-horizon planners could replace retrieval. The second is the action sequence of a random bank entry, and the third is the action sequence that retrieval returns from a bank whose action sequences were shuffled across entries, so that it belongs to a different start–goal pair. Averaged over OGBench-Cube and Two-Room, retrieval beats these starting

points by 20.3, 35.8 and 35.6 pp. None of them beats the zero action sequence: the shifted action sequence does 4.0 pp worse on Two-Room (CI $[-5.5, -2.5]$) and matches the zero action sequence on OGBench-Cube (-0.2 pp), and the other two do clearly worse on both tasks. Among the starting points we tested, only a demonstration that matches the current observation and goal helps; other learned proposals remain untested. The arm that starts from the zero action sequence is a separate run of WM-only on these episodes, with different random numbers from Figure 2a. With the learned world model and the goal loss alone, starting from the retrieved action sequence succeeds in 90.8% of the OGBench-Cube episodes and 97.5% of the Two-Room episodes, against 77.8% and 74.8% for the planner with the perfect world model, which starts from the zero action sequence; we made this comparison after both runs (Appendix D).

Larger sampling budgets. Figure 8b varies the budget of both planners on the main evaluation episodes from 3,000 to 60,000 rollouts per planning step: $N \times M$ = 100×30, 300×30 (default), 300×60, $1{,}000 \times 30$ and $1{,}000 \times 60$, with $K = N/10$ elites. WM-only improves slowly and not monotonically: on OGBench-Cube it ranges from 63.7% to 69.3% and reaches 67.7% at 60,000 rollouts, 6.7 times the default; on Two-Room it reaches 90.8% at 60,000 rollouts. IMWM varies little, between 92.5% and 95.7% on OGBench-Cube and between 98.2% and 99.8% on Two-Room. With 3,000 rollouts, a third of its default budget, IMWM exceeds WM-only with 60,000 rollouts by 25.8 pp on OGBench-Cube and 7.3 pp on Two-Room. In the tested range, a larger budget does not close the gap to IMWM, and neither does the perfect world model at the default budget (hollow markers); we did not test larger budgets with the perfect world model.

H WEIGHT OF THE INTUITION SCORE: ORIGIN AND SENSITIVITY

How the weight was chosen. IMWM uses $\lambda = 1/3$ on every task. The weight was carried over from development experiments with earlier planner variants, including sweeps of the relative weight between the intuition score and the goal loss of a separate LeWM world model. A later comparison of fixed configurations on 48 runs of an earlier evaluation grid retained $\lambda = 1/3$ and removed $\lambda = 10$ from the gate's choices (Appendix C). The resulting default was fixed before the main evaluation and not changed afterwards.

Sensitivity. After the main evaluation, we reran IMWM on its episodes with the intuition model forced on and $\lambda \in \{0.1, 1, 3\}$ on the four development tasks. Figure 10 and Table 6 add $\lambda = 0$, which is retrieval with the goal loss alone (Table 3). The gains on OGBench-Cube and Two-Room stay within 1 pp of their default values for λ between 0.1 and 3. Compared with the default, $\lambda = 0.1$ changes success by -0.7 pp on Two-Room (95% bootstrap CI $[-1.8, +0.5]$), by -0.5 pp on OGBench-Cube ($[-2.2, +1.0]$) and by $+1.8$ pp on Push-T ($[+0.3, +3.3]$), or by $+0.2$ pp on average over these three tasks, on which the gate turns the intuition model on. The best weight depends on the task: Two-Room gains up to $\lambda = 1$, OGBench-Cube is flat between 0.1 and 1 but loses 3.0 pp at $\lambda = 0$, and Push-T does best with little or no weight on the intuition score (95.7% at $\lambda = 0$ and 96.0% at 0.1, against 94.2% at the default). On Reacher, where the gate keeps the intuition model off, every weight loses between 9.7 and 16.5 pp.

We use this study to measure sensitivity, not to choose λ. Choosing the weight from it would tune λ on the evaluation episodes, and Push-T's gain at $\lambda = 0.1$ is one of twelve comparisons made after the fact (four tasks, three alternative weights). Re-selecting the weight properly would require separate tuning episodes and new runs of every IMWM experiment, including the added tasks, whose settings were fixed before they were run. A weight chosen per task, for example by the gate, is a possible extension.

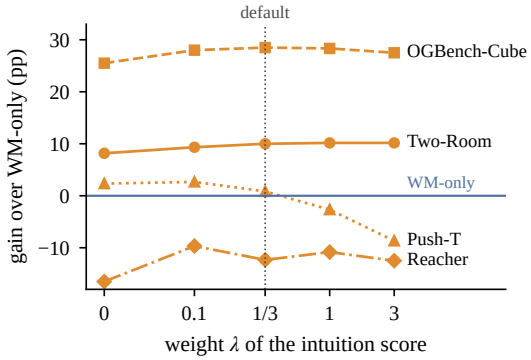


Figure 10: Weight λ of the intuition score in the Hybrid Cost: gain over WM-only with the intuition model forced on, on the development tasks (12 seeds of the main evaluation episodes). $\lambda = 0$ is retrieval with the goal loss alone (Table 3); the axis is linear between 0 and 0.1 and logarithmic beyond.

Table 6: Weight λ of the intuition score: success (%) with the intuition model forced on, over the main evaluation episodes. $\lambda = 0$ is retrieval with the goal loss alone, from Table 3; the default weight on Reacher comes from the weight study, as in Table 3.

Weight	Two-Room	Reacher	Push-T	OGBench-Cube
$\lambda = 0$	97.5	65.8	95.7	90.2
$\lambda = 0.1$	98.7	72.7	96.0	92.7
$\lambda = 1/3$ (default)	99.3	70.0	94.2	93.2
$\lambda = 1$	99.5	71.5	90.7	93.0
$\lambda = 3$	99.5	69.8	84.7	92.2
WM-only	89.3	82.3	93.3	64.7

I Reliability Gate: scores, stress tests and follow-up studies

Table 7: Reliability scores with the default banks, over 12 seeds per task, and the share of start–goal pairs whose retrieved action sequence scores above the 95th percentile of random action sequences ($m_j > 0$).

Task	range of r over seeds	pairs with $m_j > 0$	intuition on
Two-Room	$[0.091, 0.150]$	81%	12 / 12
Reacher	$[-0.051, 0.016]$	47%	0 / 12
Push-T	$[0.883, 1.266]$	90%	12 / 12
OGBench-Cube	$[0.224, 0.702]$	85%	12 / 12
OGBench-Scene	$[0.695, 0.990]$	92%	12 / 12
DMC-Cheetah	$[0.267, 0.418]$	84%	12 / 12

Separation of the reliability scores. With the default banks, the reliability scores of the five tasks where the gate turns the intuition model on range from 0.09 to 1.27 across seeds, and those of Reacher from -0.05 to 0.02 (Table 7). The gap to the threshold is smallest on Two-Room, whose scores (0.09 to 0.15) exceed it by 0.04 to 0.10. Any threshold between 0.017 and 0.091 would make the same decisions on all 72 seeds; this describes the observed separation, not the optimality of the threshold, and on the development tasks it is not independent of the calibration (Appendix C). On Reacher, the retrieved action sequence scores above the 95th percentile of random action sequences for 47% of the start–goal pairs, compared with 81–92% on the other tasks. The intuition model thus separates demonstrated from random action sequences on Reacher too, but not by the margin the gate requires.

What the score does and does not track. On Reacher, retrieval finds close matches in its embedding (on an earlier evaluation grid, the median cosine similarity between the retrieval query

$[\varphi(o_t); \varphi(o_g)]$ and the retrieved bank entry was 0.977), but the retrieved action sequence's score sits close to the 95th percentile of random action sequences. Three follow-up studies probed this. First, we checked whether random action sequences lie unusually close to Reacher's demonstrated action sequences. For each random action sequence, we computed its mean distance to the eight nearest bank action sequences, with every dimension scaled by the bank's standard deviation, and divided the median of these distances by the same median for the bank's own action sequences. The ratio is 1.000 on Reacher, below the other development tasks and OGBench-Scene (1.015 to 2.066) but almost the same as on DMC-Cheetah (0.997), which passes the gate, so this distance does not explain the difference between Reacher's and DMC-Cheetah's margins. Second, on three seeds (150 episodes) of the earlier grid, fine-tuning Reacher's intuition model with random action sequences as hard negatives raised its reliability score past the threshold (from 0.011 to 0.062), but its effect on planning with the intuition model forced on was not resolved: $+2.0$ pp against the unmodified model (95% CI $[-6.0, +10.7]$) and -7.3 pp against WM-only ($[-16.0, +1.3]$). Third, a relative version of the gate, which compares the margins of a batch with those of held-out demonstrations of the same task, reproduced the gate's decisions on only 30 of 36 runs of the earlier grid (six seeds on each of six tasks). All six Reacher runs flipped to on, because on Reacher even the held-out demonstrations' own action sequences score barely above random ones, so the retrieved action sequences look typical. The stress tests below give clearer evidence on what the threshold means.

Stress tests. Stress tests with rebuilt banks show when the gate errs (Table 8, the numbers behind Figure 9b and c). All runs use the main evaluation episodes. Like the default banks, the rebuilt and shuffled banks exclude the trajectories of the episodes evaluated on their own seed (most contain trajectories evaluated on other seeds), and the gate is recomputed on each bank.

Shuffled banks. Because the gate scores the action sequences that retrieval returns, it tests the intuition model and the bank together. To test it when retrieval is wrong, we shuffle the action sequences across bank entries, so that retrieval returns an action sequence that belongs to a different start–goal pair. With all entries shuffled and the intuition model forced on, success changes by -5.0 pp on Two-Room (CI $[-8.3, -1.8]$) and -0.5 pp on OGBench-Cube ($[-3.8, +3.2]$) against WM-only; the gate turns the intuition model off on all 12 seeds of both tasks and so recovers WM-only exactly. With half of the entries shuffled, the forced intuition model still gains 16.7 pp on OGBench-Cube ($[+13.8, +19.5]$) and 4.0 pp on Two-Room ($[+1.5, +6.3]$), but the gate turns it off on every Two-Room seed and on 7 of 12 OGBench-Cube seeds: it errs on the side of caution.

Smaller banks. With smaller banks, the gate errs in both directions. The forced intuition model gains 13.3 pp ($[+10.0, +16.7]$) on OGBench-Cube and 7.0 pp ($[+4.3, +9.7]$) on Two-Room with 30 entries, where the gate turns it off on every seed of both tasks. On Push-T, it lowers the mean success with every smaller bank: by 6.2 pp ($[-8.8, -3.3]$) with 30 entries, 3.5 pp ($[-5.8, -1.0]$) with 100, 2.7 pp ($[-5.7, +0.3]$) with 300 and 1.2 pp ($[-4.0, +1.5]$) with 1,000, while the gate keeps it on for 9 of 12 seeds with 30 entries and for every seed with larger banks. Across the tested bank conditions, passing the threshold is thus neither a guarantee of benefit nor a prerequisite for it.

Table 8: Stress tests of the gate: success (%) with the intuition model forced on, and the seeds (of 12) on which the gate turns it on. WM-only is 89.3 (Two-Room), 93.3 (Push-T) and 64.7 (OGBench-Cube).

	Two-Room		Push-T		OGBench-Cube	
Condition	forced	on	forced	on	forced	on
bank of 30	96.3	0	87.2	9	78.0	0
bank of 100	98.7	1	89.8	12	86.5	7
bank of 300	99.0	9	90.7	12	90.0	12
bank of 1,000	99.0	12	92.2	12	94.0	12
bank of 3,000 (default)	99.3	12	94.2	12	93.2	12
half of the action sequences shuffled	93.3	0	—	—	81.3	5
all action sequences shuffled	84.3	0	—	—	64.2	0

Other follow-up studies. After the main evaluation, we ran further studies on an earlier evaluation grid, whose seeds select episodes differently from the main evaluation; none of them changes the

Table 9: Differences to WM-only in Table 3 (pp) with 95% bootstrap CIs over the 12 seeds of the main evaluation episodes. The Reacher entry with retrieval and the Hybrid Cost comes from the weight study (Appendix H) at the default weight, on the same seeds; the Reacher entry without retrieval was not run.

Start	Evaluation	Two-Room	Reacher	Push-T	OGBench-Cube
zero	intuition score	$+7.3$ $[+5.8, +8.8]$	-21.8 $[-26.0, -17.7]$	-44.0 $[-50.2, -37.8]$	$+7.5$ $[+4.7, +10.0]$
zero	Hybrid Cost	$+8.0$ $[+6.7, +9.3]$	—	$+0.5$ $[-1.3, +2.3]$	$+11.2$ $[+8.5, +13.8]$
retrieved	goal loss	$+8.2$ $[+5.7, +10.7]$	-16.5 $[-21.3, -11.8]$	$+2.3$ $[+0.2, +4.5]$	$+25.5$ $[+22.8, +28.5]$
retrieved	intuition score	$+7.7$ $[+5.7, +9.7]$	-28.5 $[-32.7, -24.3]$	-36.2 $[-41.2, -31.7]$	$+17.8$ $[+14.5, +21.7]$
retrieved	Hybrid Cost	$+10.0$ $[+8.2, +12.0]$	-12.3 $[-16.0, -9.0]$	$+0.8$ $[-1.5, +3.2]$	$+28.5$ $[+25.8, +31.2]$

reported method. Keeping the top 30% of elites across CEM iterations improved IMWM by $+1.7$ pp over six tasks ($p = 0.005$), including $+4.2$ pp on Reacher, where the gate switches the intuition model off; it is not used in this paper. Re-injecting the retrieved action sequence at every iteration, refitting a Gaussian mixture seeded by retrieval, per-episode instead of per-batch trust decisions, and a learned verifier that accepts or rejects proposed action sequences all failed their pre-specified criteria; the per-episode decisions, for example, lost 5.2 pp because thresholds fitted on some tasks did not transfer to others.

J RETRAINING ON DISJOINT TRAJECTORIES

Following the platform's protocol, the world model is trained on all demonstrations, including the trajectories from which evaluation episodes are drawn, and the intuition model is trained on a random subset of the trajectories that includes some of them (Appendix C); only the retrieval bank excludes them. To measure performance without this overlap, we retrained the models of three tasks on trajectories disjoint from the evaluation.

Design. For OGBench-Cube, Two-Room and DMC-Cheetah, the three tasks with the largest gains in Table 2, a random allocation made before any run split each task's 10,000 trajectories into disjoint sets: 8,900 for training, 200 for validation (used only to check that the training losses stay finite), 300 for calibration and 600 for evaluation (12 seeds of 50 episodes, one per trajectory). The world model, the intuition model, the action normalization and the retrieval bank were rebuilt from the training trajectories only, with the same recipes and budgets as the deployed models, with one declared exception: on the development tasks, the intuition model's encoder was initialized from the retrained world model's encoder instead of the official LeWM weights, which were trained on data that include the evaluation episodes. Each seed was evaluated with WM-only; with IMWM, its gate computed on the batch of evaluation episodes as in the main experiments; with IMWM, its gate computed on 10 or 25 of the calibration episodes; and with a replay of the evaluation episode's own recorded actions, a reference that met the success criterion within its first 25 replayed actions in every episode of all three tasks.

Analysis. The quantity of interest is Δ_{gain}, the gain of IMWM over WM-only after disjoint retraining minus the gain in Table 2. For each task, a one-sided Welch test compared $H_0 : \Delta_{\text{gain}} \geq -2$ pp with $H_1 : \Delta_{\text{gain}} < -2$ pp, with Holm's correction over the three tasks. This design can detect only large reductions: with 12 seeds on each side, the reductions detectable with 80% power are about 7.4, 5.8 and 9.5 pp (OGBench-Cube, Two-Room and DMC-Cheetah).

Results. Table 10 gives the results. IMWM still improves over WM-only on all three tasks. No task shows a detected reduction beyond 2 pp (Holm-adjusted p = 0.20, 1.00 and 0.71, in the same order), but for OGBench-Cube and DMC-Cheetah the changes in gain, -5.5 and -3.2 pp, have simultaneous

Table 10: Retraining on trajectories disjoint from the evaluation. Gains are IMWM minus WM-only in pp; the disjoint gain has a 95% paired t interval over 12 seeds. The change in gain is the disjoint gain minus the gain in Table 2, with a simultaneous 95% Welch interval over the three tasks.

Task	WM-only / IMWM (%)	disjoint gain	original gain	change in gain
OGBench-Cube	70.7 / 93.7	$+23.0$ $[+19.3, +26.7]$	$+28.5$	-5.5 $[-11.3, +0.3]$
Two-Room	82.0 / 98.3	$+16.3$ $[+14.4, +18.3]$	$+10.0$	$+6.3$ $[+2.8, +9.9]$
DMC-Cheetah	64.8 / 82.2	$+17.3$ $[+12.2, +22.5]$	$+20.5$	-3.2 $[-11.2, +4.9]$

intervals that include both no change and reductions of about 11 pp, so whether removing the overlap reduces the gain is inconclusive there. On Two-Room, the gain is larger after retraining. The change in gain also compares gains measured on different episodes; on Two-Room, for example, WM-only succeeds in 82.0% of the disjoint evaluation episodes and in 89.3% of the main ones. Each task was retrained once, so the differences combine the removal of overlap, the initialization change on the development tasks and the variation between training runs. The gate computed on 10 or 25 calibration episodes made the same decisions as the gate computed on the evaluation episodes on every seed, so the three IMWM variants gave identical results. It turned the intuition model on for every seed, so this check does not cover a task on which the gate switches the model off.

K Ongoing: comparing planning, replay and learned policies

Design. On 12 new seeds per task, the comparison evaluates WM-only; IMWM; WM-only with Retrieval Initialization and the goal loss as the only cost; replay of the retrieved action sequence without planning; goal-conditioned behavior cloning of five-block action sequences (three training seeds); CEM with the world model started from the behavior-cloning policy's action sequence; goal-conditioned implicit Q-learning (three training seeds), included only if it passes a pre-specified check of its temporal semantics, data split, determinism and training health; and Diffusion Policy on Push-T (eight training seeds). The learned policies are trained on all demonstration trajectories except 200 validation trajectories used for checkpoint selection, so they see the trajectories of every evaluated episode.

Analysis. Six one-sided tests form one family with Holm's correction: IMWM against replay and against Retrieval Initialization alone on the four tasks where IMWM gains (a material-superiority margin of 2 pp), and IMWM against behavior cloning, against behavior-cloning-initialized CEM and against implicit Q-learning on all six tasks, and against Diffusion Policy on Push-T (superiority over 0).

Status. The design and its analysis were specified on 2026-08-13, before any run, and amended five times, each time without consulting any outcome. Three earlier runs were superseded: one stopped before the qualification check, one stopped at the evaluation stage after a software failure, and one completed but lost every OGBench-Scene evaluation to a data-reader incompatibility. Their records are archived and excluded, and no outcome from them has been read. A later launch halted at its preflight check before any training or evaluation. A corrected run is being prepared; its results are not part of this submission and will be added when available.