INVESTIGATING PRIORS FOR INFORMED LATENT PLANNING IN WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models leveraging the Joint Embedding Predictive Architecture (JEPA) are trained to reach goals zero-shot through latent planning. Most work starts from an uninformed Gaussian prior for planning, while a few recent works show the advantage of learned priors. However, the effectiveness of different priors is not thoroughly studied. In this work, we investigate three priors learned by different policy heads, i.e., a deterministic head (Behavior Cloning), a Gaussian one, and a generative one (Flow Matching). We evaluate these priors on success rate together with planning latency across four world model backbones and six robot environments. Our results show that learned priors can improve the success rate by a large margin, e.g., by 40.5 percentage points in the Cube environment. Moreover, learned priors exhibit computational efficiency, reducing planning time per action sequence by a factor of 2 to 15 across all environments, highlighting their practical utility. We show that while no single prior dominates in the short-horizon setting, Flow Matching learns a more robust prior most capable of challenging long-horizon planning tasks. Based on our findings, we distill guidance on choosing between priors, and other algorithm settings such as replanning cadence, to inform the design of future latent planning systems for real-world applications.

1 INTRODUCTION

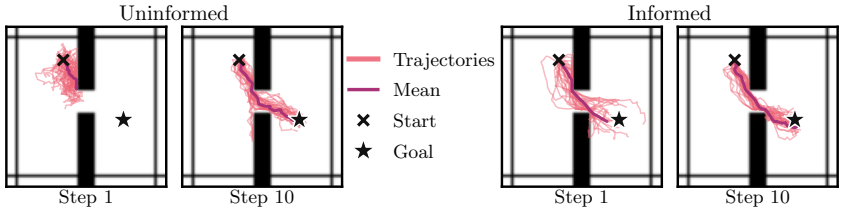


Figure 1: **Uninformed vs. informed latent planning.** Lines represent top 30 trajectories and their mean after 1 and 10 planning steps, using DINO-WM in the Wall environment. Uninformed planning starts undirected and passes through the doorway only after refinement, whereas informed planning routes through the doorway from Step 1.

Robot learning has long sought an agent that, trained once on unlabeled interaction, can pursue new goals at test time without demonstrations or rewards (Lynch et al., 2019; Park et al., 2023). Latent world models offer a natural route: Instead of reconstructing pixels or fitting a single reward, they learn how the environment evolves under actions in a compact latent space (LeCun, 2022), and behavior is *searched for* rather than trained (Hafner et al., 2019). Joint-embedding predictive architectures (JEPAs) (LeCun, 2022) are a widely adopted class of models that predicts future *embeddings* rather than frames, avoiding task-irrelevant pixel detail. Trained on reward-free offline data, they reach image-specified goals zero-shot (Sobal et al., 2025; Zhou et al., 2025; Maes et al., 2026b). In planning, such a model rolls out action sequences and scores how close the predicted terminal embeddings land to the goal. It does not, however, generate action sequences.

Such systems leave sampling action sequences to a planner with no prior over what a good action sequence looks like. At every decision the planner initializes N candidate chunks, rolls each one

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.

through the world model, scores the predicted embeddings against the goal, and then either executes the best candidate or optimizes its candidates and repeats. DINO-WM and LeWorldModel refine with the Cross-Entropy Method (CEM) (Rubinstein, 1997) and PLDM with MPPI (Williams et al., 2015), both sampling candidate chunks from a zero-mean isotropic Gaussian (Hafner et al., 2019), while gradient-based planners start from zeros. The agent executes part of the chunk before planning again, and how much it executes sets the replanning cadence. The sample size and the number of refinement steps together form the planning budget, but they are not equally expensive: The candidate chunks are rolled out in parallel, so increasing candidates costs little latency, whereas the refinement steps run in sequence and dominate planning time, and too few of them risks severe performance degradation (Pinneri et al., 2020). Without a prior, the planner spends its budget rediscovering action structure already present in the training data.

Very recent work closes this gap by learning a prior for planning. LeFlow (Huang et al., 2026) generates goal-conditioned latent paths with a flow model and decodes them into action chunks, and SAGE (Cheng et al., 2026) uses a Gaussian mixture model. PRISM (Wang et al., 2026b) seeds the planner's candidates from a goal-conditioned Gaussian prior, and GC-IDM (Nguyen et al., 2026) replaces planning altogether with a deterministic single action predictor. All report large gains over uninformed planning in success rate and latency. Figure 1 illustrates the advantage of informed planning with learned priors: Uninformed planning (CEM) scatters its initial candidate chunks (Step 1) around the start and needs refinement (Step 10) to find the doorway, whereas an informed planner concentrates its first action chunks on routes that already pass through it.

While these works show that priors help, each pairs a different prior with its own planner, sample size and replanning schedule, and one removes search altogether. It thus remains open which prior matters, and when and how it should be combined with search. In this work, we address these research questions systematically, *rethinking when and how an informed prior is needed in latent planning*. Concretely, we study three classes of learned priors: A deterministic policy (Behavior Cloning), a (unimodal goal-conditioned) Gaussian policy, and a generative policy (Flow Matching). All three generate an action chunk from the same start and goal embeddings and differ only in the distribution placed over that chunk, which orders them by expressivity: Behavior Cloning predicts the mean chunk, the Gaussian head adds a learned per-dimension variance, and the generative prior is not restricted to a parametric form and can represent several distinct routes to the same goal.

Our experiments reveal that the priors separate far less than their expressivity suggests, and none dominates in short-horizon planning, while Flow pulls ahead under frequent replanning and in long horizon planning. The benefit of a prior is determined by the quality of the offline data, and the planning setting around the prior has a larger effect on success than the choice of the prior itself, with the replanning cadence and the goal weighting together accounting for swings of up to 66 percentage points (pp). We therefore invite future work to report priors across these settings and to further explore prior design, particularly in multimodal settings.

In summary, the contributions of this work are as follows:

- We investigate different types of priors for informed latent planning, i.e., deterministic Behavior Cloning, a unimodal goal-conditioned Gaussian, and generative policy Flow Matching, in a comprehensive evaluation. To the best of our knowledge, we are the first to thoroughly study their advantage.
- Our results highlight that priors are effective and efficient, reaching 99.7% on Cube when the demonstrations are goal-directed, whereas uninformed CEM reaches 59.2% using seven times the planning time. With less informative data, priors still learn useful action structure that complements CEM, together achieving state-of-the-art performance.
- We provide a detailed discussion for choosing between priors and other algorithm settings such as the sample size, the replanning cadence and the weighting of the goal distance. For instance, the generative prior is the most robust choice under frequent replanning. Such guidance supports reliable latent planning in real-world applications.

2 Related Work

Latent world models. Latent world models learn environment dynamics and plan through them at test time (Ha & Schmidhuber, 2018; Hafner et al., 2019). Neither dominant training signal, how-

ever, suits a reusable, reward-free planner: Reward or value prediction presumes a task reward it lacks, while reconstruction ties representation capacity to pixel fidelity (Hafner et al., 2023; Hansen et al., 2024). Joint-embedding predictive architectures (JEPAs) (LeCun, 2022) instead predict the *embedding* of a future observation, avoiding pixel reconstruction but risking representation collapse, which methods prevent in different ways (Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025). Our backbones are three such models: DINO-WM (Zhou et al., 2025) freezes a pretrained DINOv2 encoder and learns a causal predictor over its patch tokens; PLDM (Sobal et al., 2025) trains encoder and predictor end-to-end with VICReg regularization (Bardes et al., 2022); and LeWorldModel (Maes et al., 2026b) regularizes with SIGReg (Balestriero & LeCun, 2025). The geometry of the latent space matters for planning, motivating work that smooths or straightens it so that embedding distance better tracks progress toward the goal (Terver et al., 2026; Wang et al., 2026a). Collectively, these models provide a *verifier grounded in learned dynamics*: A latent rollout whose terminal-embedding cost measures goal progress. We take this verifier as given and instead tackle the other half of the problem: The distribution the planner searches over.

Uninformed Latent Planning. Given a trained latent world model, a plan is obtained by optimizing a finite-horizon action sequence, executed either open-loop or re-optimized in a receding-horizon loop (García et al., 1989). Two families of optimizer dominate. Zeroth-order optimization methods sample candidate sequences and score them by the model: The cross-entropy method (Rubinstein, 1997) refits a Gaussian to a top-K elite set, while MPPI (Williams et al., 2015) reweights the samples by cost. These are widely used in many systems (Hafner et al., 2019; Zhou et al., 2025; Sobal et al., 2025; Maes et al., 2026b). First-order methods instead differentiate the rollout with respect to the actions (Srinivas et al., 2018; Howell et al., 2022), but rely on local optimization over a non-convex landscape and are sensitive to initialization. Both families are *uninformed at initialization*: CEM begins from a zero-mean isotropic Gaussian and gradient descent from zeros (Hafner et al., 2019; Maes et al., 2026a). Yet the distribution the candidates are drawn from matters as much as the refinement: Predictive Sampling, essentially "CEM with a non-adaptive distribution", stays competitive with more elaborate optimizers (Howell et al., 2022), and iCEM improves performance at small sample sizes by swapping CEM's white noise for temporally correlated noise (Pinneri et al., 2020). Our work focuses on learned priors that provide more informative candidate chunks. These priors are not only used to replace search algorithms, but also to give search algorithms a warm start.

Informed Latent Planning. A recent line of work equips the latent planner with a prior learned from the same reward-free demonstrations the world model is trained on. GC-IDM (Nguyen et al., 2026) removes the planner altogether, amortizing the search into a deterministic goal-conditioned mapping to single actions. PRISM (Wang et al., 2026b) retains the planner and replaces the initial candidate chunks of its search with samples from a learned goal-conditioned Gaussian. SAGE (Cheng et al., 2026) conditions a Gaussian mixture on predicted subgoals, and LeFlow (Huang et al., 2026) generates action chunks with a latent flow planner. Each of these works proposes a single prior, evaluated on one world model at a fixed replanning cadence, and their designs differ in architecture and conditioning as well as in distributional form. We systematically compare different types of priors within a unified framework across four world models and six environments. We focus on analyzing how planning settings such as sample size, replanning frequency, and goal-distance weighting shape their benefits.

3 METHOD

Uninformed latent planning scores rolled-out embeddings against a goal but does not propose potential action sequences: CEM samples from $\mathcal{N}(0, \sigma^2 I)$ and Gradient Descent starts near zero. In this work, we improve planning by replacing the uninformed initialization with samples drawn from a learned prior conditioned on the current and the goal embedding. We first introduce the latent planning problem in world models and the uninformed trajectory solution, followed by our designs for informed planning priors.

3.1 PRELIMINARIES

Control problem. From the current image observation and a goal image, produce actions that drive the environment to the goal. Observations are $o \in \mathcal{O}$ and raw actions $u \in \mathbb{R}^{d_u}$. As is standard

in visual world modeling, we plan at a coarser rate than the simulator: A block of c raw actions forms one *decision-step* action $a_t = (u_{ct}, \dots, u_{ct+c-1}) \in \mathbb{R}^{d_a}$ with $d_a = c\, d_u$, and observations are indexed at the same rate. The planner outputs a *plan* of horizon H, a chunk $A = (a_0, \dots, a_{H-1}) \in \mathcal{A} \triangleq \mathbb{R}^{H \times d_a}$, scored by the latent objective defined in Equation 2.

Latent world model. We require only two things: An encoder $E_\phi : \mathcal{O} \to \mathcal{Z}$, and latent dynamics f_θ conditioned on a bounded window of the l most recent latents and actions, applied autoregressively to give an H-step rollout operator,

$$\hat{z}_{t+1} = f_\theta\big(z_{t-l+1:t},\, a_{t-l+1:t}\big), \qquad \hat{z}_{1:H} = \Phi_\theta(z_0, A) \;:\; \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}^H. \tag{1}$$

Here Φ_θ denotes the H-step rollout operator that results from applying f_θ autoregressively, feeding each prediction back as the next input, so that it maps an initial latent and a whole action chunk to the H predicted latents. The lookback l bounds how far *back* the predictor sees, and the horizon H how far *forward* the planner commits, with $z_0 = E_\phi(o_0)$ being the encoded initial observation. Each world model is trained to predict one step ahead in latent space, using a backbone-specific mechanism to prevent representation collapse. We use pretrained and frozen world models.

Planning objective and execution. Goals are given as *images* o_g, encoded as $z_g = E_\phi(o_g)$. The planner minimizes the terminal latent distance

$$J\big(A \mid z_0, z_g\big) \;=\; \big\| \big[\Phi_\theta(z_0, A)\big]_H - z_g \big\|_2^2, \qquad A_{\text{opt}} = \arg\min_{A \in \mathcal{A}} J\big(A \mid z_0, z_g\big). \tag{2}$$

As the cost in Equation 2 is *terminal only*, intermediate latents are unconstrained. This lack of intermediate shaping makes optimizing through the nonlinear dynamics Φ_θ more difficult by withholding any signal for partial progress. For this reason, we also consider a weighted variant that scores intermediate latents,

$$J_b\big(A \mid z_0, z_g\big) \;=\; \sum_{i=1}^{H} w_i \, \big\| \big[\Phi_\theta(z_0, A)\big]_i - z_g \big\|_2^2, \qquad w_i \;=\; \frac{b^{\,i}}{\sum_{j=1}^{H} b^{\,j}}, \tag{3}$$

where the base $b > 0$ controls how the weight is distributed over the rollout and the normalization makes the weights sum to one, so that $b < 1$ emphasizes early steps, $b = 1$ weights all steps equally, $b > 1$ emphasizes late steps, and $b \to \infty$ recovers Equation 2. Crucially Φ_θ is differentiable in A, so both objectives admit zeroth- and first-order optimization. Planning runs in a model-predictive control loop (García et al., 1989): The current observation o_0 is encoded to z_0 and the goal image o_g to z_g, Equation 2 is solved for a chunk of H decision steps, the first $R \leq H$ of them are executed, and the loop repeats until the episode budget is exhausted. The goal embedding z_g is fixed for the whole episode, whereas z_0 is re-encoded from the observation reached after each execution. We refer to R as the replanning cadence. At $R = H$ a chunk is executed in full before any replanning, and the quality of chunks rather than the ability to recover through frequent replanning is measured.

Hindsight relabeling. We train on offline demonstrations $\mathcal{D} = \{(o^{(n)}_{0:T_n}, a^{(n)}_{0:T_n-1})\}$, where T_n denotes the length of the n-th trajectory, utilizing no reward and no task labels. We extract sliding windows by enumerating all valid start indices $\mathcal{W} = \{(n, t) : 0 \leq t < T_n - H\}$. Since $\mathcal{D}$ carries no goals, the goal of a window is assigned in hindsight (Andrychowicz et al., 2017; Lynch et al., 2019): The observation the demonstrator reached H decision steps later is declared to be the goal, which turns every window into a successful demonstration of reaching it,

$$z_0 = E_\phi\big(o^{(n)}_t\big), \quad z_g = E_\phi\big(o^{(n)}_{t+H}\big), \quad A^\star = \big(a^{(n)}_t, \dots, a^{(n)}_{t+H-1}\big), \tag{4}$$

Only the goal is relabeled; $A^\star$ is the chunk the demonstrator executed and is the regression target.

3.2 Priors

With a prior, the planner models a distribution $p_\psi(A \mid z_0, z_g)$ over the H-step action chunk conditioned on the encoded current and goal observation, and all are trained on the same hindsight-relabeled windows of Equation 4. We consider three such priors: A deterministic head (Behavior

Cloning), a Gaussian head, and a generative one (Flow Matching). They span a range of distributional expressiveness, from a point estimate, through a unimodal distribution with learned spread, to an arbitrary multimodal one. All three share the same architecture and differ only in the distribution they place on the chunk.

Behavior Cloning (BC). The least expressive prior places no distribution on the chunk and regresses the demonstrated one directly, minimizing $\|\mu_\psi(z_0, z_g) - A^\star\|_2^2$, where μ_ψ denotes the predicted mean chunk. Its optimum is the conditional *mean* chunk. CEM needs a spread of candidates to refine, so we apply a fixed amount of Gaussian jitter to allow sampling from this prior. We draw candidates as $\mu_\psi + \sigma_{\text{jit}}\,\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$, keep one candidate unperturbed, and fix the jitter scale to $\sigma_{\text{jit}} = 0.3$.

Gaussian head (Gaussian). This prior predicts a diagonal Gaussian over the flattened action chunk, $\mathcal{N}\big(\mu_\psi(z_0, z_g),\ \operatorname{diag}\sigma_\psi^2(z_0, z_g)\big)$. Here the spread is learned and goal-conditioned. The scale is emitted as $\sigma = \operatorname{softplus}(\cdot) + \sigma_{\min}$ with a floor $\sigma_{\min} = 0.05$ that prevents an overconfident prior from collapsing the planner's search, and the head is trained with the β-NLL objective (Seitzer et al., 2022), which interpolates between the Gaussian likelihood and plain regression and avoids the variance collapse the likelihood suffers on near-deterministic action dimensions. This prior models a *unimodal* distribution.

Flow Matching (Flow). The most expressive prior represents $p_\psi(A \mid z_0, z_g)$ implicitly, as the transport of a Gaussian onto the action distribution along a learned velocity field (Lipman et al., 2023). It retains probability on several distinct routes when a goal admits more than one, instead of averaging them into an interpolated compromise that may yield an erroneous trajectory. We train the field with the optimal-transport conditional flow-matching objective (Tong et al., 2024) and sample by integrating its ordinary differential equation (ODE) with 20 midpoint steps.

4 Experiments

4.1 Experimental Settings

Baselines & Environments. We perform latent planning with four world-model backbones: LeWorldModel (LWM) (Maes et al., 2026b), PLDM (Sobal et al., 2025), DINO-WM (Zhou et al., 2025), and DINO-WM with a straightened embedding space (Wang et al., 2026a). As uninformed planning baselines, we use CEM and GD. Both draw $N = 300$ samples: CEM refits on the 30 lowest-cost elites and executes their mean, while GD optimizes all samples in parallel and, like the priors, executes the lowest-cost sequence (Equation 2). We inherit the number of refinement steps of CEM from the respective codebases. GD utilizes five Adam steps and a constant learning rate of 0.1 as a default. We plan and execute five decision steps ($H = R = 5$) with each decision step consisting of five raw actions (frameskip of 5) for a total of 25 raw actions. We evaluate our planning methods on six environments: Two-Room and Wall (Zhou et al., 2025; Sobal et al., 2025), Push-T (Chi et al., 2023), OGBench-Cube (Park et al., 2025), as well as on PointMaze-UMaze and PointMaze-Medium (Fu et al., 2020). Three seeds and 200 evaluation episodes drawn from a dedicated test split are used to assess performance on each environment. More details regarding the underlying datasets and the baselines can be found in Appendix A.

Implementation Details. We train each prior for every (world model, environment) pairing using a training split of the original pretraining data. We precompute hindsight-relabeled tuples (z_0, z_g, A) with $H = 5$ from the training trajectories using each world model's encoder (Equation 4), and train on these demonstrations for 10 epochs. Training a prior is cheap, taking between less than a minute and at most eleven minutes on a single GPU. More training details can be found in Appendix A.4. At inference, we draw N candidate chunks $A^{(j)}$ from the prior p_ψ, score each one, and select the one with the lowest cost:

$$A_{\text{prior}} = \underset{j \le N}{\arg\min}\ J\big(A^{(j)} \mid z_0, z_g\big), \quad \text{where } A^{(j)} \sim p_\psi(\cdot \mid z_0, z_g). \tag{5}$$

Beyond using the prior alone, we also combine it with planners such as CEM and GD by replacing their initializations with samples from the prior. Algorithm details are given in Appendix A.3.

Table 1: Success rate (%) of the priors compared to two uninformed planners. Results are averaged over three independent runs. Bold denotes the best planner per environment.

Planner	Flow	Gaussian	BC	CEM	GD
LeWorldModel (Maes et al., 2026b)					
Two-Room	97.17±2.08	97.83±1.26	**100.00±0.00**	84.67±1.61	94.50±1.00
Push-T	81.33±1.89	76.67±3.33	79.33±2.89	**85.83±0.58**	73.00±5.57
OGBench-Cube	99.17±0.76	**99.50±0.50**	**99.50±0.50**	71.83±1.53	61.83±3.55
PLDM (Sobal et al., 2025)					
Two-Room	96.50±1.00	98.17±0.58	**100.00±0.00**	95.50±2.65	94.00±1.80
Push-T	**82.83±2.47**	80.33±2.02	78.67±3.51	77.83±6.03	70.83±3.01
OGBench-Cube	98.67±0.58	**99.67±0.29**	99.50±0.50	59.17±3.55	56.17±1.04
DINO-WM (Wang et al., 2026a)					
Wall	92.83±2.75	88.33±1.04	80.33±2.08	**94.00±1.50**	92.17±0.29
Push-T	**77.00±1.50**	70.33±3.75	72.00±4.92	74.17±2.08	39.17±1.53
PointMaze-UMaze	66.33±4.48	67.67±5.03	61.50±3.50	**83.50±4.09**	76.33±2.36
PointMaze-Medium	67.67±2.02	69.17±6.33	67.67±1.53	**88.83±2.02**	68.50±5.89
DINO-WM (Straightened) (Wang et al., 2026a)					
Wall	94.83±1.15	91.33±1.76	81.33±0.76	**98.50±0.50**	94.50±0.50
Push-T	75.83±1.76	71.83±2.93	70.00±4.27	**78.50±1.50**	43.50±5.07
PointMaze-UMaze	78.83±2.75	82.33±1.61	90.00±3.04	**93.50±1.50**	84.00±2.00
PointMaze-Medium	69.33±3.21	70.83±0.76	79.17±1.61	**85.00±2.50**	72.33±3.82

4.2 Priors vs. Uninformed Planning

Table 1 lists the success rates of three learned priors across six environments. From the results, we see that all priors beat uninformed search in the OGBench-Cube and Two-Room environments, e.g., BC improves the success rate compared to the uninformed planning CEM on OGBench-Cube by 27.7 and 40.3 pp with LWM and PLDM, and reaches 100% on Two-Room under both world models. However, on the environments PointMaze-UMaze/-Medium and Wall, the uninformed planner CEM is leading, by between 1.2 and 19.7 pp over the best prior. The reason is that the demonstrations on OGBench-Cube and Two-Room are collected with a scripted and a goal-directed policy, while the demonstrations on Wall and the two PointMazes come from a random policy. With random demonstrations, a prior does not learn goal-reaching behavior but only reproduces the undirected motion of the data. On PointMaze-UMaze, where the priors reach their lowest absolute success, straightening the latent space boosts them by a large margin, e.g., BC rises from 61.5% to 90.0% and the Flow from 66.3% to 78.8%. In a nutshell, informed planning is beneficial when learning from high-quality demonstrations, and the three priors perform comparably.

Given the disadvantage on the two PointMazes and Wall, and the narrow margins on Push-T, we further improve them by replacing CEM's uninformed initial candidate chunks with a prior, as given in the algorithms in Appendix A.3. Table 2 lists the results. We see that with learned priors, we achieve the state-of-the-art performance. E.g., Flow rises from 67.7% to 87.3% on PointMaze-Medium and from 66.3% to 85.0% on PointMaze-UMaze, and on Push-T with PLDM from 82.8% to 93.3%, against 78.0% for CEM alone. The prior provides initial candidate chunks concentrated on demonstration-consistent behavior, which CEM cannot obtain from its isotropic initialization, while CEM refines these chunks toward the goal using the world model. Combining the two therefore outperforms either alone.

4.3 Long-Horizon Planning

Whether a prior trained on short ($H{=}5$) chunks remains useful at a longer horizon is a natural concern, and Figure 2 compares two ways of obtaining long-horizon ($H{=}10$) chunks on Push-T: 1) Extending a short chunk with Gaussian-sampled actions, or 2) training the prior at $H{=}10$ directly. An extended prior used on its own is weak, since half its chunk is uninformed, and does not beat CEM by more than 1.0 pp under any world model. Refinement changes that, adding between +7.5

Table 2: Success rate (%) of the three priors combining with CEM on Wall, Push-T and PointMazes. Results are averaged over three independent runs. Bold denotes the best planner per environment.

Planner	Flow+CEM	Gaussian+CEM	BC+CEM	CEM
DINO-WM				
Wall	**98.17±0.76**	98.00±0.87	94.50±0.87	94.00±1.50
Push-T	**89.17±2.57**	87.83±3.21	85.50±3.97	74.17±2.08
PointMaze-UMaze	**85.00±3.28**	83.67±2.52	77.50±1.32	83.50±4.09
PointMaze-Medium	87.33±1.26	88.50±1.32	81.17±2.84	**88.83±2.02**
DINO-WM (Straightened)				
Wall	**98.67±0.58**	97.33±1.53	94.17±1.44	98.50±0.50
Push-T	**88.33±1.61**	**88.33±1.76**	85.00±3.50	78.50±1.50
PointMaze-UMaze	93.67±1.04	92.83±2.93	**96.83±0.58**	93.50±1.50
PointMaze-Medium	86.17±2.25	**88.00±2.60**	87.50±3.12	85.00±2.50

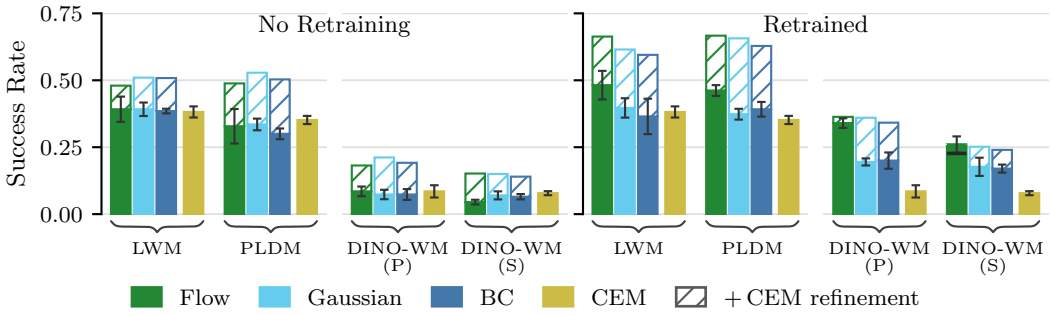


Figure 2: **Long-horizon planning on Push-T.** Without retraining, short-horizon (H=5) candidate chunks are extended to H=10 with Gaussian actions; retrained priors are trained directly at H=10.

pp (BC, straightened DINO-WM) and +20.3 pp (BC, PLDM). Training at H=10 improves the priors by between 9.0 and 25.5 pp over their extended counterparts. In retraining with larger H, Flow outperforms the others by between 7.0 pp (PLDM) and 14.0 pp (projected DINO-WM), as illustrated in the right subfigure of Figure 2. Refinement helps least where the candidate chunks are strongest: The retrained Flow prior shows the smallest improvements, and, using straightened DINO-WM, even loses 3.3 pp. We conclude that Flow captures the structure of long action chunks better than the two unimodal priors, and that training at a larger H is preferable to extending a prior.

An informed prior also reaches high success rates efficiently, and Figure 3 plots success rate against planning time on Push-T across optimization budgets. Across both short- and long-horizon settings, Flow+CEM achieves success rates matching the best uninformed planners in 3.6× to 10× less planning time. At the long horizon Flow+GD reaches the highest absolute success (67.3% under LWM and 76.0% under PLDM), while at the short horizon Flow+CEM is best. Notably, both informed planners improve most within the first five to ten optimization steps. The Flow with ten CEM steps reaches 88.7% and 91.5% on short-horizon Push-T, outperforming 30-step uninformed CEM within half of the time. Sampling candidate chunks from the Flow prior starts the search in promising regions and thus requires fewer iterations, while a GD step remains more expensive than a CEM refinement. We therefore recommend Flow+CEM with ten refinement steps as the default.

4.4 Hyperparameter Analysis

Impact of sample size. Figure 4 varies the sample size from 30 to 500 candidates on Push-T, holding the elite fraction at 10%. Uninformed CEM is the most sensitive to the sample size N, gaining between 14.7 pp (LWM) and 24.5 pp (straightened DINO-WM) from $N = 30$ to $N = 300$. The priors, by contrast, are less sensitive to the sample size, as the gain is only between 3.8 and 14.7 pp. All three priors beat CEM under all world models at $N = 30$, whereas at $N = 300$ CEM is leading under LWM and straightened DINO-WM. E.g., when using LWM, Flow outperforms CEM

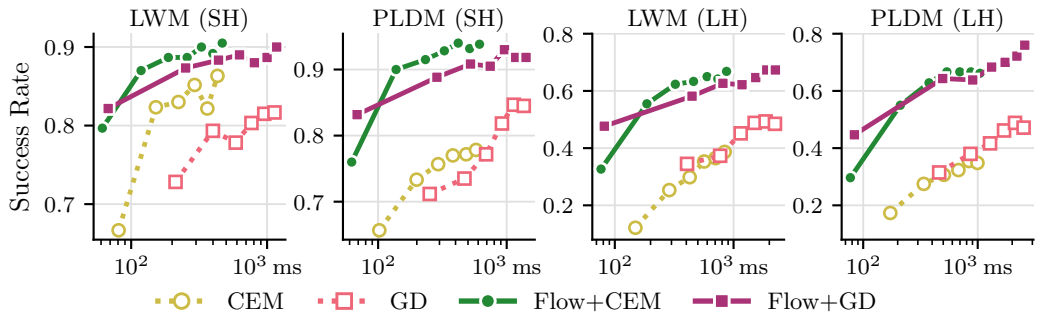


Figure 3: **Success rate (%) against per-sample planning time.** Results on short-horizon (SH) and long-horizon (LH) Push-T. All planners were evaluated at $5, 10, 15, 20, 25, 30$ steps. Additionally results of the first Flow+CEM step and the initial candidate chunks of Flow+GD are shown.

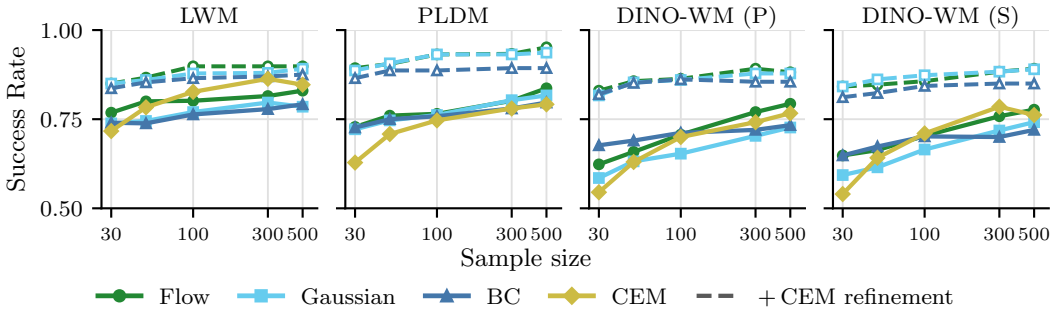


Figure 4: **Success rate (%) against sample size N.** Results on Push-T. Dashed lines indicate priors seeding CEM. CEM refits from $N/10$ elites at every budget.

by 5.2 pp at $N = 30$ but falls behind by 4.8 pp at $N = 300$. This demonstrates how a single pair of methods can yield contradictory conclusions depending on the sample size. We therefore conclude that evaluating planners at a single sample size is insufficient.

Replanning cadence and cost weighting. We study the effect of different combinations of replanning cadence R and cost weights b in Tables 3 and 13. When setting R to the horizon H ($R = H = 5$), a terminal embedding cost ($b = \infty$) achieves the highest success rates in almost all configurations, while finite weights, penalizing the distance of intermediate latent states, dominate for denser replanning ($R \in \{1, 2\}$) with the optimal b depending on the planner. The terminal embedding cost dominates at a full horizon because penalizing intermediate distances in a curved latent space is suboptimal for complete sequence rollouts, whereas shorter cadences rely on intermediate trajectory guidance. Furthermore, under optimal weightings, the Flow achieves robust performance for dense replanning (changing by between -6 and $+1$ pp from $R = 5$ to $R = 1$), while the other priors suffer substantial drops (8–26 pp for the Gaussian head, and 17–36 pp for BC). Flow provides more stable guidance, making it less sensitive to the replanning cadence than the other priors.

4.5 Qualitative Results

Figure 5 shows rollouts executed without replanning on the short-horizon Push-T task using the straightened DINO-WM. t refers to the executed raw environment steps. The first row in Figure 5 illustrates that CEM's executed action chunk barely moves the end-effector (the blue dot). But CEM is able to achieve the goal after ten refinements as shown in the second row. In contrast, Flow's initial lowest-cost action chunk already routes the end-effector to the block, pushes the block to the goal pose, and also routes the end-effector into its goal pose without any refinement. This shows that the Flow emits learned goal-reaching behavior rather than trying out random trajectories. Additional qualitative comparisons on Wall can be found in Appendix B.7.

Table 3: Success rate (%) using LWM on Push-T. Different combinations of replanning cadence R and cost weighting b are used. For each cadence and planner, the best cost weighting is in bold.

R	b	Flow	Gaussian	BC	CEM
5	0.5	70.00±2.18	69.00±3.50	69.67±4.62	64.33±1.61
5	1	80.50±3.50	75.00±2.60	76.67±2.36	72.33±1.76
5	2	78.83±0.76	**79.17±4.73**	77.50±3.77	82.33±1.89
5	∞	**86.00±3.04**	78.33±5.13	**77.67±2.02**	**86.33±1.04**
2	0.5	79.67±0.58	**73.50±2.65**	**61.00±2.18**	67.83±2.47
2	1	**82.67±2.25**	71.50±3.50	55.00±0.50	75.83±3.62
2	2	73.33±4.86	56.67±2.75	48.33±4.04	**82.17±4.37**
2	∞	21.67±2.36	33.33±4.37	32.50±3.50	38.17±2.57
1	0.5	83.33±4.04	**71.17±1.53**	**60.67±2.75**	67.50±2.18
1	1	**87.33±2.08**	64.00±2.29	50.17±3.21	78.50±1.00
1	2	73.50±4.27	43.17±2.36	37.00±4.36	**82.50±0.87**
1	∞	20.83±2.93	24.50±1.73	25.00±1.32	29.50±4.44

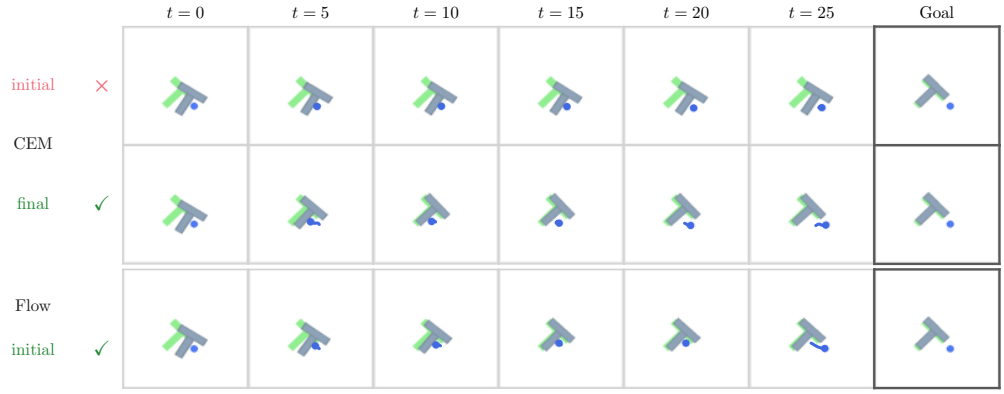


Figure 5: **Planning on Push-T** with straightened DINO-WM. *Initial* is the elite mean for CEM and the lowest-cost candidate chunk for Flow. *Final* is CEM's elite mean chunk after ten refinements.

5 Conclusion

In this work, we investigate when an informed prior helps world model latent planning. The three studied priors differ in the distribution they learn, i.e., a deterministic Behavior Cloning policy, a unimodal Gaussian head, and a generative Flow Matching policy. We compare them with uninformed CEM and GD planning across six environments and four world model configurations. Our results show that priors yield a gain in success rate compared to uninformed planning. This gain is significant when the demonstration data contains goal-directed trajectories. Otherwise, the prior can still learn useful action structure, which complements the CEM search and yields state-of-the-art performance. Furthermore, using priors can massively improve computational efficiency, especially when using Flow in long-horizon planning tasks. We further provide guidance on choosing the sample size, replanning cadence, and cost weighting in planning algorithms. Based on our discussion, we highlight the importance of evaluating priors across sample sizes and replanning cadences, and the value of using priors for effective and efficient latent planning.

Limitations. Despite their effectiveness, the priors can be further improved. First, priors are trained per world model and environment, preventing zero-shot transfer across encoders or tasks. Future work should learn priors in a shared latent space across tasks. Second, demonstration quality is essential in prior training but varies across environments rather than being controlled. Third, we compare priors with a single, matched size across four world model configurations and six simulated environments. Future work should explore prior sizes, more encoders, and real-world evaluations.

AI USE STATEMENT

In this work, we used generative AI tools to assist in improving the readability of the text. Concretely, we used generative AI tools to provide feedback on this manuscript and to assist in writing and polishing this manuscript by refining phrasing, clarity, and language. Moreover, we also used AI agents for partially executing and monitoring experiments, debugging, and improving code clarity. Specifically, we used agents to adapt scripts the author provided to new settings. Generative AI tools were NOT used to generate research ideas, design experiments, produce experimental data, or analyze results. All scientific claims, experimental settings, and analyses were written and validated by the authors. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We are committed to ensuring reproducibility of our findings. We provide a detailed description of our approach and experimental design in the Method and Experiments sections. The Appendix contains extensive information about the experiment environments, world models and planners used, as well as implementation details. To support the review process, we have provided an anonymous repository containing a codebase to reproduce key aspects of our results, available at `https://anonymous.4open.science/r/Investigating-Priors-0B18`. Upon acceptance of this manuscript, we commit to publicly releasing the source code to fully support community research.

REFERENCES

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5048–5058, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html`.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael G. Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 15619–15629. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01499. URL `https://doi.org/10.1109/CVPR52729.2023.01499`.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew J. 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, 2025. URL `https://doi.org/10.48550/arXiv.2506.09985`.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *CoRR*, abs/2511.08544, 2025. doi: 10.48550/ARXIV.2511.08544. URL `https://doi.org/10.48550/arXiv.2511.08544`.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=xm6YD62D1Ub`.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Trans. Mach. Learn. Res.*, 2024, 2024. URL `https://openreview.net/forum?id=QaCCuDfBk2`.

Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. Chapter 3 - the cross-entropy method for optimization. In C.R. Rao and Venu Govindaraju (eds.), *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pp. 35–59. Elsevier, 2013. doi: https://doi.org/10.1016/B978-0-444-53859-8.00003-5. URL https://www.sciencedirect.com/science/article/pii/B9780444538598000035.

Letian Cheng, Qi Zhang, and Yisen Wang. Sage: Subgoal-conditioned action generation for latent world model planning. *CoRR*, abs/2607.17973, 2026. URL https://arxiv.org/abs/2607.17973.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://doi.org/10.15607/RSS.2023.XIX.026.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning, 2020. URL https://arxiv.org/abs/2004.07219.

Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989. ISSN 0005-1098. doi: https://doi.org/10.1016/0005-1098(89)90002-2. URL https://www.sciencedirect.com/science/article/pii/0005109889900022.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models, 2023. URL https://doi.org/10.48550/arXiv.2301.04104.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive sampling: Real-time behaviour synthesis with mujoco, 2022. URL https://arxiv.org/abs/2212.00541.

Hsiang-Wei Huang, Jianxu Shangguan, Junbin Lu, and Jenq-Neng Hwang. Leflow: Generative latent flow planning for world models. *CoRR*, abs/2608.24855, 2026. URL https://arxiv.org/abs/2608.24855.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2019. URL http://proceedings.mlr.press/v100/lynch20a.html.

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, 2026a. URL https://arxiv.org/abs/2605.21800.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026b. URL https://doi.org/10.48550/arXiv.2603.19312.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models. *CoRR*, abs/2605.08732, 2026. doi: 10.48550/ARXIV.2605.08732. URL https://doi.org/10.48550/arXiv.2605.08732.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: offline goal-conditioned RL with latent states as actions. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/6d7c4a0727e089ed6cdd3151cbe8d8ba-Abstract-Conference.html.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3942–3951. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11671. URL https://doi.org/10.1609/aaai.v32i1.11671.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolínek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In Jens Kober, Fabio Ramos, and Claire J. Tomlin (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065. PMLR, 2020. URL https://proceedings.mlr.press/v155/pinneri21a.html.

Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(96)00385-2. URL https://www.sciencedirect.com/science/article/pii/S0377221796003852.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=aPOpXlnV1T.

Vlad Sobal, Wancong Zhang, Kynghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models, 2025. URL https://doi.org/10.48550/arXiv.2502.14819.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4739–4748. PMLR, 2018. URL http://proceedings.mlr.press/v80/srinivas18b.html.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=cHZn5Gdh8e.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=CD9Snc73AW.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning, 2026a. URL https://doi.org/10.48550/arXiv.2603.12231.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: prior-guided imagination sampling in world models. *CoRR*, abs/2606.07974, 2026b. doi: 10.48550/ARXIV.2606.07974. URL https://doi.org/10.48550/arXiv.2606.07974.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling, 2015. URL https://arxiv.org/abs/1509.01149.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: world models on pre-trained visual features enable zero-shot planning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A Extended Experimental Setup

A.1 Environments

Two-Room is a continuous 2D navigation task (Sobal et al., 2025) in which an agent must move between two rooms joined by a single doorway in a dividing wall. Because the only passage between the rooms is the door, reaching a target on the far side requires routing through it rather than travelling in a straight line. The dataset, collected by Maes et al. (2026b), consists of 10,000 trajectories of on average 92 steps, gathered with a noisy heuristic policy that first steers the agent toward the door and then toward the goal once it has entered the second room.

Push-T is a contact-rich 2D manipulation task, originally introduced by Chi et al. (2023), in which a circular pusher must arrange a T-shaped block into a target configuration through pushing contact alone. An episode succeeds when the pusher position, the block position and the block orientation all fall within fixed tolerances of the goal. Push-T appears in both of our codebases with the identical task, success criterion and demonstration data, namely the 18,685 trajectories of average length 125 collected by Zhou et al. (2025). Planning with LWM and PLDM uses image observations alone while planning with the DINO-WM variants additionally conditions on the pusher's proprioceptive state. This is in accordance with the DINO-WM and temporal-straightening codebases, whose planning objective combines a visual and a proprioceptive term (Zhou et al., 2025; Wang et al., 2026a).

OGBench-Cube is a continuous 3D manipulation task from the OGBench benchmark (Park et al., 2025), in which a robotic arm with an end-effector must pick up a cube and place it at a target location. We use the single-cube variant of the task. The dataset, collected by Maes et al. (2026b) with OGBench's scripted data-collection policy, consists of 10,000 trajectories of 200 steps each.

Wall is a 2D navigation environment (Zhou et al., 2025; Sobal et al., 2025) consisting of two rooms separated by a wall with a single narrow door, in which an agent must travel from a start position to a target position specified through images. Its layout mirrors the Two-Room environment described above, with the single door as the only passage between rooms. The agent is purely kinematic, with actions given as 2D displacements in the plane. The dataset consists of 1,920 undirected random-walk trajectories of 50 steps each, released by Zhou et al. (2025).

PointMaze-UMaze is a 2D maze navigation task built on the MuJoCo physics engine (Fu et al., 2020), in which a force-actuated point mass must be driven from a start position to a target position. In contrast to Wall, the agent obeys realistic physics, so its motion is governed by velocity, acceleration, and inertia, and actions are forces applied along the x and y axes. The UMaze layout is a single U-shaped corridor. The dataset consists of 2,000 fully random trajectories of 100 steps each, released by Zhou et al. (2025).

PointMaze-Medium shares the physics and action space of the UMaze variant but uses a larger maze with more branching corridors, which lengthens the routes between start and target and makes navigation more demanding. The dataset consists of 4,000 fully random trajectories of 100 steps each, released by Zhou et al. (2025).

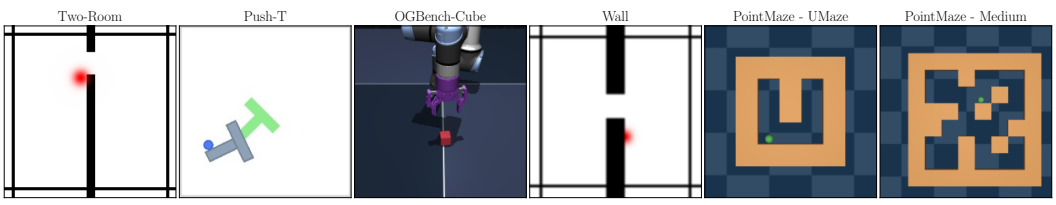


Figure 6: The six evaluation environments. The first three (Two-Room, Push-T and OGBench-Cube) are used with LeWorldModel and PLDM and the remaining three (Wall, PointMaze-UMaze, PointMaze-Medium) as well as Push-T are used with the two DINO-WM variants.

A.2 World Models

LeWorldModel (LWM) (Maes et al., 2026b) is a joint-embedding predictive world model (JEPA) trained end-to-end from raw pixels. A vision-transformer encoder maps each frame to a single latent vector, and a causal transformer predictor forecasts the next-frame latent from the current one and an embedding of the executed action, injected through adaptive layer normalization. It is trained with a teacher-forced prediction loss $\mathcal{L}_{\text{pred}}$, the squared error between the predicted and the encoded next latent, together with the Sketched-Isotropic-Gaussian Regularizer (SIGReg) (Balestriero & LeCun, 2025), an anti-collapse term that pushes the embedding distribution toward an isotropic Gaussian by matching random one-dimensional projections to a standard normal. The complete objective is

$$\mathcal{L}_{\text{LeWM}} \triangleq \mathcal{L}_{\text{pred}} + \lambda \text{SIGReg}(Z), \tag{6}$$

where Z is the batch of latent embeddings collected over a history, and λ weights the regularizer.

The model has roughly 15M parameters and trains on a single GPU in a few hours. We use the publicly released LeWorldModel checkpoints. The encoder is a ViT-tiny (patch size 14, images of 224×224, latent dimension 192) and the predictor is a ViT-S backbone with learned positional embeddings and causal masking over the observation history, whose length is three frames for Push-T and OGBench-Cube and one for Two-Room. We refer to Maes et al. (2026b) and Balestriero & LeCun (2025) for additional details on LWM and SigReg.

PLDM (Sobal et al., 2025) is a joint-embedding predictive world model. We adopt the PLDM baseline configuration of Maes et al. (2026b), which uses the same ViT-tiny encoder and ViT-S predictor as LeWorldModel. PLDM prevents representation collapse differently from LeWorldModel: Rather than SIGReg, it regularizes the embedding space with variance and covariance terms in the style of VICReg (Bardes et al., 2022), together with temporal-similarity, temporal-variance, temporal-covariance, and inverse-dynamics terms. The training objective is

$$\mathcal{L}_{\text{PLDM}} \triangleq \mathcal{L}_{\text{pred}} + \alpha\mathcal{L}_{\text{var}} + \beta\mathcal{L}_{\text{cov}} + \gamma\mathcal{L}_{\text{time-sim}} + \zeta\mathcal{L}_{\text{time-var}} + \nu\mathcal{L}_{\text{time-cov}} + \mu\mathcal{L}_{\text{IDM}}. \tag{7}$$

Here $\mathcal{L}_{\text{pred}}$ is the squared error between the predicted and the encoded next latent; $\mathcal{L}_{\text{var}}$ and $\mathcal{L}_{\text{cov}}$ are the variance and covariance regularizers; $\mathcal{L}_{\text{time-sim}}$, $\mathcal{L}_{\text{time-var}}$, and $\mathcal{L}_{\text{time-cov}}$ apply the temporal analogues, encouraging smooth latent trajectories; and $\mathcal{L}_{\text{IDM}}$ is an inverse-dynamics term predicting the executed action from two consecutive latents. We train PLDM from scratch. Following the setting of Maes et al. (2026b), we set the coefficients to $\alpha = 18$, $\beta = 12$, $\gamma = 0.2$, $\zeta = 0.7$, $\nu = 0$, and $\mu = 0$.

DINO-WM (Zhou et al., 2025) is a latent world model built on frozen pretrained visual features. Each frame is encoded by a frozen DINOv2 vision transformer (Oquab et al., 2024) into a grid of patch tokens, and a ViT transition model predicts the next-frame patch features from a short history, conditioned on embeddings of the action and, where available, the proprioceptive state. Unlike LeWorldModel and PLDM, which pool each frame to a single latent vector, DINO-WM retains the full spatial grid of patch features. Planning rolls out the transition model and matches the predicted patch features to those of the goal image. We use the two variants introduced by Wang et al. (2026a). The Projected variant augments the frozen DINOv2 features with a learned channel projector that reduces each patch from 384 to 8 dimensions, yielding a compact latent for planning. The Straightened variant uses the same projector but is additionally trained with a temporal straightening loss that aligns consecutive latent velocities, so that latent trajectories curve as little as possible. The motivation is that straighter latent trajectories induce a better-conditioned planning objective, which makes gradient- and sampling-based optimization more effective (Wang et al., 2026a).

A.3 Planners

All planners in this work optimize the same objective, the terminal cost $J(A \mid z_0, z_g)$ between the latent reached by rolling the world model out under an action sequence A and the goal latent z_g (Eq. 2). Because the number of initial candidates itself affects success, we evaluate every planner at a fixed, shared sample budget of $N = 300$ candidates under every world model. This budget applies uniformly to CEM, GD, the three priors used on their own, and each prior seeding CEM or GD, so the comparison isolates how each planner searches rather than how many candidates it draws. A prior used on its own returns its lowest-cost sample without further optimization (Eq. 5) and is described in the main text and not repeated here.

Cross-entropy Method (CEM) (Rubinstein, 1997; Botev et al., 2013) maintains a factorized Gaussian over action sequences. Starting from a zero-mean Gaussian with unit variance ($\mu^{(0)} = \mathbf{0}$, $\sigma^{(0)} = \mathbf{1}$), each iteration i draws N candidate sequences $\mathcal{P}^{(i)} = \{A^{(j)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(j)}\}_{j=1}^{N}$ with $\epsilon^{(j)} \sim \mathcal{N}(0, I)$, scores them by the rollout cost J, keeps the K lowest-cost *elites* $\mathcal{E}^{(i)}$, and refits the Gaussian to them, elementwise over the (H, d_a) action grid:

$$\mathcal{E}^{(i)} = \underset{A \in \mathcal{P}^{(i)}}{\arg\min\text{-K}}\, J(A \mid z_0, z_g), \qquad \mu^{(i+1)} = \tfrac{1}{K} \sum_{A \in \mathcal{E}^{(i)}} A, \qquad \sigma^{(i+1)} = \mathrm{std}\big(\mathcal{E}^{(i)}\big). \tag{8}$$

After I iterations the planner acts the elite mean $\mu^{(I)}$.

Gradient Descent (GD). Because the world model is differentiable, the terminal cost $J(A \mid z_0, z_g)$ can be minimized directly by backpropagating through the latent rollout into the action sequence (Srinivas et al., 2018). We optimize with Adam for S steps at a constant learning rate and act the lowest-cost sequence after the final step. Contrary to the reference gradient-descent planner of Wang et al. (2026a), which optimizes a single action sequence per episode, we optimize N sequences in parallel, initialized from a Gaussian around zero, so that GD is evaluated at the same sample budget as the other planners. The N sequences are optimized as one batch but never interact: Each descends only its own cost, so they are N independent gradient descents, and we act the lowest-cost one.

Prior+CEM seeds the CEM search with one of the three priors, and we write Flow+CEM, Gaussian+CEM and BC+CEM for the three instances. At iteration 0 we replace the Gaussian samples entirely with N samples from the prior:

$$\mathcal{P}^{(0)} = \{A^{(j)} \sim p_\psi(z_0, z_g)\}_{j \le N}. \tag{9}$$

For the later refinement steps $i \ge 1$ the algorithm proceeds as ordinary CEM, sampling and refitting according to Equation 8, and after I iterations we act the elite mean $\mu^{(I)}$. Comparing Prior+CEM with CEM isolates the effect of the learned prior. Sampling only a fraction of N from a learned prior and the remainder from a Gaussian is ablated in Figure 8 and Tables 14 and 15.

Prior+GD. Gradient descent is a local, gradient-based optimizer. Unlike the derivative-free, proposal-based search of CEM, which exhibits lower susceptibility to local optima (Pinneri et al., 2020), gradient descent is prone to being trapped by them. Consequently, its performance depends on the initialization, as it may fail to recover from a poor starting point. An informed proposal directly addresses this limitation. In Prior+GD, we replace the standard Gaussian initializations of the GD baseline with N samples from the prior, which serve as N parallel starts, while leaving the optimizer, learning rate, and step count unchanged:

$$A_{[0]}^{(j)} \sim p_\psi(z_0, z_g), \qquad A_{[s+1]}^{(j)} = \mathrm{Adam}\Big(A_{[s]}^{(j)},\ \nabla_{A^{(j)}} \textstyle\sum_{j'=1}^{N} J\big(A_{[s]}^{(j')} \mid z_0, z_g\big),\ \eta_s\Big), \tag{10}$$

for $s = 0, \ldots, S-1$. Since only the initialization differs from the GD baseline, a comparison between the two isolates the effect of the prior.

A.4 Implementation Details

For the world models and the evaluation environments we follow the setups of the LeWorldModel (Maes et al., 2026b) and temporal-straightening (Wang et al., 2026a) codebases, described in Section A.2 and Section A.1. Below we document only where our protocol deviates from those references, together with the architecture and training of the three priors. All hyperparameters are collected in Table 4.

Matched sample budget and gradient-descent settings. The reference gradient-descent planner optimizes a single action sequence per episode (Wang et al., 2026a). To compare every planner under an equal budget, we run CEM, GD, the three priors and each prior seeding CEM or GD from the same number of initial sequences $N = 300$ under every world model. We further fix the gradient-descent budget to $S = 5$ Adam steps at a constant learning rate of 0.1 for both GD and Prior+GD, dropping the reference cosine schedule so that the only difference between the two is

the initialization. A prior used on its own takes no gradient steps ($S = 0$) and acts its lowest-cost sample. The CEM and Prior+CEM refinement schedules are inherited per environment and listed in Table 4; all remaining world-model, environment and CEM settings are taken unchanged from the reference codebases.

Prior architecture. All three priors share one backbone: A residual MLP of width d_h and depth L, into every block of which the conditioning is FiLM-modulated (Perez et al., 2018), with a zero-initialized output projection so that training starts from a null prediction. The conditioning is the start and goal latents z_0, z_g taken from the frozen world model. With LWM and PLDM that latent is a single vector; with the DINO-WM variants, whose latent is a grid of patch tokens, we condition on the flattened patch tokens and, on Push-T, additionally on the proprioceptive latent. The priors differ only in what enters the backbone beside the conditioning and in what the output layer emits. The Flow, a conditional flow-matching model (Lipman et al., 2023) over normalized action chunks $A \in \mathbb{R}^{H \times d_a}$, additionally consumes a noisy action chunk and a sinusoidal embedding of the Flow time τ, and emits a velocity field. Behavior cloning and the Gaussian head have no such per-sample input and read a learned query token in its place, a single trainable vector that is identical for every input and serves only to give the backbone something to transform, so that the prediction is carried entirely by the FiLM conditioning. They emit a mean chunk, the Gaussian head emitting a per-dimension scale as well. Because width and depth are identical across the three, their parameter counts land within 4% of one another, so the comparison isolates the distributional form rather than capacity.

Conditional flow matching with optimal-transport coupling. Let $p_0 = \mathcal{N}(0, I_{Hd_a})$ and $p_1 = q(\cdot \mid z_0, z_g)$. Flow matching (Lipman et al., 2023) learns a field whose ODE transports p_0 to p_1. We use the linear (rectified) interpolant (Liu et al., 2023), whose conditional velocity is *constant along the path*: $x_\tau = (1-\tau)x_0 + \tau A^\star$ and $\dot{x}_\tau \triangleq \mathrm{d}x_\tau/\mathrm{d}\tau = A^\star - x_0$ for $\tau \in [0,1]$. Pairing each $A^\star$ with an independent x_0 leaves a large irreducible variance in the regression target: The same x_τ is reachable from many pairs with conflicting velocities and the model can only learn their conditional mean. We therefore use a minibatch optimal-transport coupling (Tong et al., 2024). To minimize path intersections, we solve a linear assignment problem per batch of size B. Letting $\mathfrak{S}_B$ denote the set of all possible permutations of B elements, we use the Hungarian algorithm to efficiently compute the exact permutation $\pi^\star \in \mathfrak{S}_B$ that minimizes the total squared Euclidean distance between the paired samples:

$$\pi^\star = \operatorname*{arg\,min}_{\pi \in \mathfrak{S}_B} \sum_{i=1}^{B} \left\| x_0^{(\pi(i))} - A^{\star(i)} \right\|_2^2, \qquad \text{pairing } A^{\star(i)} \text{ with } x_0^{(\pi^\star(i))}. \tag{11}$$

This straightens the marginal trajectories, motivating the coarse fixed-step ODE integration used at sampling time (Table 4). Letting $\pi^\star$ also denote the empirical joint distribution it induces, the objective is:

$$\mathcal{L}(\psi) = \mathop{\mathbb{E}}_{(z_0, z_g, A^\star) \sim \mathcal{D},\ \tau \sim \mathcal{U}[0,1],\ x_0 \sim \pi^\star(\cdot \mid A^\star)} \left\| v_\psi\big(x_\tau, \tau, z_0, z_g\big) - \big(A^\star - x_0\big) \right\|_2^2. \tag{12}$$

Note what Equation 12 omits: No rollout of Φ_θ, no gradient through the dynamics and no reward. The Flow sees only the encoder's output and the demonstrated actions, never the cost J its samples will be scored against. While each backbone needs its own Flow, no planner enters Flow training.

Prior training and generation. Training tuples come from hindsight relabeling (Equation 4), pairing each start with the observation H decision steps later and the chunk executed in between. We train one prior of each type per (environment, world model, horizon), using a $90\%/10\%$[1] episode holdout for the LWM and PLDM experiments and the datasets' existing train and test splits for the DINO-WM experiments, and we report on the held-out episodes. At planning time we generate candidate chunks by integrating the Flow ODE from $\tau = 0$ to 1 with 20 midpoint (RK2) steps.

Hardware. Every training and evaluation run used a single NVIDIA H100 GPU, up to 32 CPU threads, and at most 125 GB of RAM for evaluation and 200 GB for training.

[1] For Push-T its 1,818 out of 18,685 episodes (9.73%) to prevent leakage.

Table 4: Implementation details of the priors and the shared planner budgets.

Prior architecture (shared)	
Residual blocks L	4
Hidden dimension d_h	384
Time embedding dim	128 (Flow only)
Conditioning	FiLM
Prior training and generation	
Objective (Flow)	OT-CFM (Equation 12)
Objective (Gaussian)	β-NLL, $\beta = 0.5$ (Seitzer et al., 2022)
Objective (BC)	MSE, $\|\mu_\psi - A^\star\|_2^2$
Optimizer	AdamW, lr 10^{-3}, weight decay 10^{-4}
LR schedule	short linear warmup to 10^{-3}, then cosine to 0
Batch size	512
Epochs	10
Precision, gradient clip	bf16, 1.0
Horizon H	5 (short), 10 (long)
ODE integrator (planning)	RK2, 20 steps (Flow only)
BC jitter σ_{jit}	0.3, one candidate unperturbed
Planner budgets	
Sample size N	300
GD, Prior+GD	5 Adam steps, lr 0.1 constant
Priors alone	0 steps (lowest-cost sample)
CEM iterations	LWM/PLDM: 30 (Push-T), 10 (others); DINO-WM: 10

B Additional Results

B.1 Priors seeding CEM

Table 5 extends Table 2 to every short-horizon configuration, adding LWM and PLDM and the GD column.

Table 5: Success rate in % of the three priors seeding CEM, on every short-horizon configuration.

Planner	Flow+CEM	Gaussian+CEM	BC+CEM	CEM	GD
LWM					
Two-Room	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	84.67±1.61	94.50±1.00
Push-T	**89.83±2.75**	88.00±3.61	87.00±2.18	86.33±1.04	73.00±5.57
OGBench-Cube	**99.50±0.00**	99.33±0.58	98.83±0.76	71.83±1.53	61.83±3.55
PLDM					
Two-Room	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**	95.50±2.65	94.00±1.80
Push-T	**93.33±1.89**	93.17±2.08	89.33±3.88	78.00±5.77	70.83±3.01
OGBench-Cube	99.17±0.58	99.83±0.29	**100.00±0.00**	59.17±3.55	56.17±1.04
DINO-WM					
Wall	**98.17±0.76**	98.00±0.87	94.50±0.87	94.00±1.50	92.17±0.29
Push-T	**89.17±2.57**	87.83±3.21	85.50±3.97	74.17±2.08	39.17±1.53
PointMaze-UMaze	**85.00±3.28**	83.67±2.52	77.50±1.32	83.50±4.09	76.33±2.36
PointMaze-Medium	87.33±1.26	88.50±1.32	81.17±2.84	**88.83±2.02**	68.50±5.89
DINO-WM (Straightened)					
Wall	**98.67±0.58**	97.33±1.53	94.17±1.44	98.50±0.50	94.50±0.50
Push-T	**88.33±1.61**	**88.33±1.76**	85.00±3.50	78.50±1.50	43.50±5.07
PointMaze-UMaze	93.67±1.04	92.83±2.93	**96.83±0.58**	93.50±1.50	84.00±2.00
PointMaze-Medium	86.17±2.25	**88.00±2.60**	87.50±3.12	85.00±2.50	72.33±3.82

B.2 LONG HORIZON RESULTS

Four tables report the long-horizon results the main text summarized for Push-T. Tables 6 and 7 are the padded setting behind Figure 2, with the priors used on their own and seeding CEM; Tables 8 and 9 repeat both with priors trained natively at $H = 10$.

Two observations are specific to the natively trained priors. Long-horizon Push-T is the one configuration in which the Flow separates from the two policy heads on its own, leading them by between 7.0 and 14.5 percentage points under all four world models. On PointMaze-Medium the priors alone reach 33.0 to 46.0% against CEM's 84.2 to 86.0%, and only seeding CEM closes that gap. Notably, it does not close the gap for behavior cloning, which trails the other two refined priors by between 10.7 and 21.5 percentage points.

Table 6: Success rate in % at the long horizon with *padded* candidate chunks. This table contains part of the data behind Figure 2. Each prior is the short $H{=}5$ candidate chunk Gaussian-padded out to $H{=}10$ decision steps.

Planner	Flow	Gaussian	BC	CEM	GD
LWM					
Push-T	**39.17±4.73**	**39.17±2.52**	38.50±0.87	**38.17±2.08**	34.67±4.16
OGBench-Cube	62.00±1.50	**64.00±0.87**	60.83±2.08	**50.67±4.16**	41.33±2.25
PLDM					
Push-T	32.83±6.45	**33.50±2.18**	30.00±2.00	**35.17±1.53**	31.67±2.52
OGBench-Cube	54.67±1.61	**56.00±1.00**	55.50±2.29	**46.17±2.57**	37.83±1.61
DINO-WM					
Push-T	**8.50±1.80**	7.33±1.76	7.33±2.02	**8.50±2.29**	6.00±1.00
PointMaze-Medium	40.33±3.55	39.50±3.12	**46.00±2.00**	**86.00±3.12**	58.00±4.09
DINO-WM (Straightened)					
Push-T	4.50±0.87	**7.00±1.50**	6.50±1.00	**7.83±0.76**	5.17±2.57
PointMaze-Medium	39.17±2.02	42.67±4.65	**44.83±5.77**	**84.17±3.01**	58.50±1.73

Table 7: Success rate in % at the long horizon with *padded* candidate chunks and CEM. This table contains part of the data behind Figure 2. The CEM and GD columns repeat Table 6.

Planner	Flow+CEM	Gaussian+CEM	BC+CEM	CEM	GD
LWM					
Push-T	48.00±7.55	**51.00±8.05**	50.83±3.79	**38.17±2.08**	34.67±4.16
OGBench-Cube	65.50±1.00	**65.67±1.15**	65.00±2.29	**50.67±4.16**	41.33±2.25
PLDM					
Push-T	48.83±4.86	**52.83±1.53**	50.33±2.57	**35.17±1.53**	31.67±2.52
OGBench-Cube	56.00±1.80	57.33±0.76	**58.83±2.93**	**46.17±2.57**	37.83±1.61
DINO-WM					
Push-T	18.17±3.75	**21.17±3.82**	19.17±2.75	**8.50±2.29**	6.00±1.00
PointMaze-Medium	83.33±3.75	**85.33±2.02**	82.17±0.76	**86.00±3.12**	58.00±4.09
DINO-WM (Straightened)					
Push-T	**15.17±2.08**	15.00±1.32	14.00±0.50	**7.83±0.76**	5.17±2.57
PointMaze-Medium	82.33±3.55	**82.83±2.31**	81.50±3.50	**84.17±3.01**	58.50±1.73

B.3 PLANNING TIMES

Table 10 reports the wall-clock time each planner needs to select an action chunk for execution, including the world-model rollouts.

Table 8: Success rate in % at the long horizon with priors trained *natively* at $H = 10$ and used on their own. This table contains part of the data behind Figure 2. The CEM and GD columns repeat Table 6.

Planner	Flow	Gaussian	BC	CEM	GD
LWM					
Push-T	**48.17±5.35**	39.67±3.62	36.50±6.61	**38.17±2.08**	34.67±4.16
OGBench-Cube	**89.83±1.44**	84.83±1.61	**89.83±3.21**	**50.67±4.16**	41.33±2.25
PLDM					
Push-T	**46.17±2.02**	37.33±2.02	39.17±2.75	**35.17±1.53**	31.67±2.52
OGBench-Cube	78.00±1.80	79.00±1.80	**85.33±1.04**	**46.17±2.57**	37.83±1.61
DINO-WM					
Push-T	**34.00±1.80**	19.50±1.32	20.00±3.04	**8.50±2.29**	6.00±1.00
PointMaze-Medium	**42.00±0.50**	41.83±5.01	33.00±3.50	**86.00±3.12**	58.00±4.09
DINO-WM (Straightened)					
Push-T	**26.00±3.04**	17.67±3.40	17.00±1.50	**7.83±0.76**	5.17±2.57
PointMaze-Medium	43.17±5.97	43.67±1.61	**46.00±6.26**	**84.17±3.01**	58.50±1.73

Table 9: Success rate in % at the long horizon with natively trained $H = 10$ priors seeding CEM. This table contains part of the data behind Figure 2. The CEM and GD columns repeat Table 6.

Planner	Flow+CEM	Gaussian+CEM	BC+CEM	CEM	GD
LWM					
Push-T	**66.33±2.47**	61.50±3.61	59.50±1.73	**38.17±2.08**	34.67±4.16
OGBench-Cube	86.50±0.50	85.00±3.28	**88.50±1.80**	**50.67±4.16**	41.33±2.25
PLDM					
Push-T	**66.67±3.62**	65.67±3.06	62.83±2.57	**35.17±1.53**	31.67±2.52
OGBench-Cube	79.50±0.50	81.50±0.50	**88.50±2.65**	**46.17±2.57**	37.83±1.61
DINO-WM					
Push-T	**36.33±1.53**	36.00±2.65	34.17±1.04	**8.50±2.29**	6.00±1.00
PointMaze-Medium	85.50±1.32	**85.67±2.47**	64.17±2.89	**86.00±3.12**	58.00±4.09
DINO-WM (Straightened)					
Push-T	22.67±3.62	**25.17±3.79**	24.00±3.12	**7.83±0.76**	5.17±2.57
PointMaze-Medium	**84.83±1.76**	82.00±3.00	71.33±4.91	**84.17±3.01**	58.50±1.73

B.4 Success Rate vs. Planning Time

Table 11 shows the success rate at different numbers of optimizer or refinement steps, with the corresponding planning time in milliseconds given in parentheses.

B.5 Sample size

Table 12 provides every value behind Figure 4, including the single-candidate column the figure leaves out. At $N = 1$ the world model has nothing to choose between, so the run evaluates the candidate chunk without planning.

B.6 Replanning cadence and cost weighting

Table 13 repeats the grid of Table 3 for PLDM and the two DINO-WM configurations.

Table 10: Per-sample planning time in milliseconds. The time one planner needs to propose an action sequence (including world-model rollouts).

Planner	Flow	Gauss.	BC	Flow+CEM	Gaussian+CEM	BC+CEM	CEM	GD
LWM								
Two-Room	69	36	36	187	158	158	151	213
Push-T	68	36	36	469	438	438	430	204
OGBench-Cube	74	38	38	194	161	163	157	233
PLDM								
Two-Room	70	38	37	231	203	203	195	239
Push-T	69	38	38	601	578	580	571	253
OGBench-Cube	76	45	46	247	213	212	202	278
DINO-WM								
Wall	633	613	617	6089	6053	6007	6037	9586
Push-T	646	628	630	6068	6043	5999	6186	9853
PointMaze-UMaze	709	611	685	6041	6015	5977	6074	9717
PointMaze-Medium	641	619	619	6067	6060	6022	6001	9649
DINO-WM (Straightened)								
Wall	629	611	615	6083	6042	6001	6039	9559
Push-T	618	601	602	6085	6061	6038	5912	9475
PointMaze-UMaze	621	604	606	6119	6082	6038	5989	9545
PointMaze-Medium	644	623	623	6181	6155	6123	6050	9691

B.7 WALL TRAJECTORY VISUALIZATION

Figure 7 compares CEM and Flow+CEM in the Wall environment using the straightened DINO-WM, showing both the simulator space and the first two principal components of the latent rollout. For CEM (two left panels), the first-iteration action sequence (yellow dotted line) stops prematurely near the wall barrier and fails to reach the goal (black star) from the start (black cross). Only after iterative refinement does the final action sequence (solid blue line) pass through the gap and reach the goal. For Flow+CEM (two right panels), the initial action sequence already routes through the gap to the goal in the first iteration, and it almost perfectly overlaps the final sequence in both simulator and latent space, indicating that further refinement was largely unnecessary. At the level of a single trajectory, this mirrors the aggregate finding that an informed prior places the search near a solution from the outset.

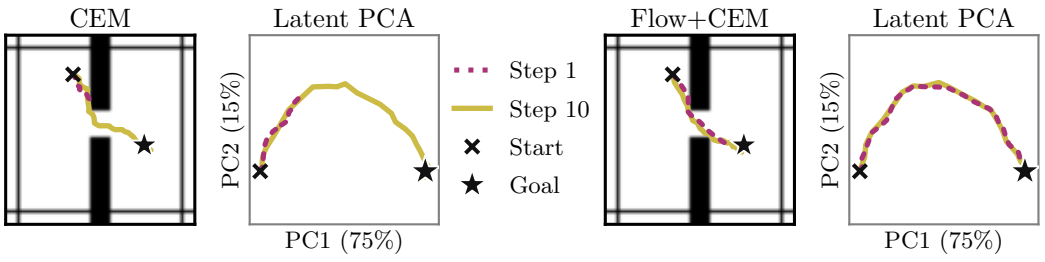


Figure 7: Comparison between CEM and Flow+CEM on the Wall environment (Simulator and PCA) using straightened DINO-WM. The purple line indicates the action sequence proposed at the first step and the solid yellow line indicates the action sequence proposed at the last step. Axis-labels of the PCA indicate the explained variance of the components.

Table 11: Success rate in % on Push-T with per-sample planning time in milliseconds in parentheses. This table holds the values behind Figure 3. The long-horizon rows use the natively trained $H = 10$ flow.

Steps	CEM	Flow+CEM	GD	Flow+GD
LeWorldModel (SH)				
0/1	12.33±2.08 (24)	79.67±2.47 (61)	44.67±1.76 (27)	82.17±3.62 (68)
5	66.67±0.29 (81)	87.00±1.80 (118)	72.83±4.86 (212)	87.33±0.76 (252)
10	82.33±0.76 (152)	88.67±1.61 (188)	79.33±2.36 (399)	88.33±2.93 (439)
15	83.00±2.18 (223)	88.67±2.08 (258)	77.83±2.52 (588)	89.00±3.28 (627)
20	85.17±2.02 (293)	90.00±3.61 (329)	80.33±4.31 (767)	88.00±1.73 (807)
25	82.17±2.02 (363)	89.17±2.75 (398)	81.50±2.65 (953)	88.67±2.89 (1000)
30	86.33±0.58 (431)	90.50±3.50 (468)	81.67±5.01 (1133)	90.00±3.50 (1177)
PLDM (SH)				
0/1	19.67±2.75 (25)	76.00±0.50 (62)	45.17±1.04 (28)	83.17±0.29 (69)
5	65.67±3.21 (102)	90.00±0.87 (138)	71.17±2.47 (252)	88.83±1.89 (287)
10	73.33±2.93 (199)	91.50±2.00 (233)	73.50±4.58 (469)	90.83±2.52 (523)
15	75.67±5.01 (293)	92.83±1.04 (327)	77.17±0.29 (692)	90.50±1.00 (744)
20	77.00±3.61 (381)	94.00±2.18 (420)	81.83±0.76 (919)	93.00±2.18 (961)
25	77.17±6.37 (484)	93.17±0.76 (515)	84.67±3.55 (1135)	91.83±1.26 (1175)
30	77.83±5.84 (577)	93.83±1.89 (615)	84.50±1.80 (1357)	91.83±0.76 (1414)
LeWorldModel (LH)				
0/1	1.00±0.50 (38)	32.67±0.76 (77)	14.17±1.44 (42)	47.67±3.25 (82)
5	12.17±3.51 (149)	55.50±5.89 (186)	34.50±4.44 (406)	58.17±5.01 (447)
10	25.33±1.44 (288)	62.33±1.61 (323)	37.33±3.40 (771)	62.67±2.57 (814)
15	29.83±1.15 (427)	63.33±0.29 (460)	45.17±3.25 (1148)	62.17±2.02 (1178)
20	35.33±1.26 (565)	65.00±3.77 (598)	48.83±2.08 (1505)	64.67±0.29 (1537)
25	36.50±1.00 (704)	64.17±3.01 (736)	49.33±3.69 (1861)	67.33±4.48 (1903)
30	38.67±2.93 (842)	66.83±3.75 (873)	48.50±3.97 (2230)	67.33±5.80 (2250)
PLDM (LH)				
0/1	3.17±0.29 (40)	29.67±0.58 (77)	14.33±2.52 (43)	44.67±5.01 (84)
5	17.33±2.84 (173)	55.00±0.50 (211)	31.50±2.00 (461)	64.33±6.43 (494)
10	27.50±2.78 (338)	62.83±2.25 (375)	38.00±1.73 (865)	63.83±3.01 (905)
15	30.67±2.47 (507)	66.67±4.01 (538)	41.67±0.58 (1281)	68.33±6.01 (1322)
20	32.33±3.40 (678)	66.67±3.01 (697)	46.17±1.26 (1698)	70.00±5.50 (1736)
25	35.50±3.97 (838)	66.67±2.25 (872)	48.83±1.53 (2115)	72.17±2.84 (2199)
30	34.83±2.08 (995)	66.17±3.33 (1044)	47.17±3.25 (2515)	76.00±1.73 (2574)

B.8 Ablation: Hybrid CEM

Instead of replacing all candidate chunks as in Equation 9, we can sample a fraction ρ of them from a prior and sample the remainder from the Gaussian:

$$\mathcal{P}^{(0)} = \underbrace{\left\{A^{(j)} \sim p_\psi(z_0, z_g)\right\}_{j \le \rho N}}_{\text{prior proposals}} \cup \underbrace{\left\{\mu^{(0)} + \sigma^{(0)} \odot \epsilon^{(j)}\right\}_{j > \rho N}}_{\text{Gaussian exploration}}, \quad \epsilon^{(j)} \sim \mathcal{N}(0, I), \tag{13}$$

so that $\rho = 1$ is the default of Prior+CEM and $\rho = 0$ recovers standard CEM exactly. Figure 8 depicts the effect of different ratios of samples drawn from the Flow prior instead of a Gaussian on latent planning with different world models across two distinct environments: OGBench-Cube and Push-T. The left subfigure shows a sharp increase in success rate as the Flow Ratio increases from 0% to 25% for both LWM and PLDM on OGBench-Cube, reaching a near-perfect success rate. The right subfigure depicts the performance of the straightened DINO-WM and also shows a large, although more gradual, increase in success rate from 0% to 25%. Across both tasks and all evaluated world models, increasing the Flow Ratio correlates with an improved success rate.

Table 12: Success rate in % against sample size N on short-horizon Push-T. This table holds the values behind Figure 4. CEM's elite fraction is fixed at 10% of N throughout.

N	1	30	50	100	300	500
LWM						
Flow	50.83±3.21	76.83±2.75	80.00±2.18	80.17±1.04	81.50±3.28	83.00±0.87
Gaussian	58.50±5.63	73.83±2.36	74.50±0.50	77.00±2.29	79.67±3.69	78.50±1.80
BC	67.17±1.04	74.00±2.65	73.83±1.61	76.33±2.25	77.83±2.47	79.17±2.84
Flow+CEM	–	85.00±1.32	86.67±2.57	89.83±2.25	89.83±2.75	89.83±3.40
Gaussian+CEM	–	85.00±2.60	86.00±3.00	87.83±2.93	88.00±3.61	89.17±2.36
BC+CEM	–	83.67±2.36	85.33±2.57	86.50±2.29	87.00±2.18	87.50±2.29
CEM	0.17±0.29	71.67±1.76	78.33±0.58	82.67±2.36	86.33±1.04	84.67±1.61
PLDM						
Flow	49.00±2.65	72.83±2.57	76.00±3.28	76.50±3.97	80.17±3.79	83.67±3.21
Gaussian	54.00±5.22	72.17±2.47	74.67±4.48	76.17±1.53	80.33±2.31	81.67±3.25
BC	59.50±2.60	72.67±4.04	75.00±1.80	75.83±2.52	78.00±2.00	79.67±3.75
Flow+CEM	–	89.33±0.76	90.50±2.78	93.17±2.93	93.33±1.89	95.17±2.02
Gaussian+CEM	–	88.67±2.57	90.67±2.89	93.17±1.26	93.17±2.08	93.67±2.89
BC+CEM	–	86.50±5.27	88.67±5.13	88.67±2.75	89.33±3.88	89.33±3.40
CEM	0.17±0.29	62.83±3.40	70.83±1.15	74.67±4.51	78.00±5.77	79.17±4.86
DINO-WM						
Flow	30.50±3.97	62.33±1.61	65.83±1.76	70.67±1.61	77.00±1.50	79.33±2.89
Gaussian	42.17±2.08	58.50±4.09	63.17±2.93	65.33±2.84	70.33±3.75	72.67±3.88
BC	51.17±0.76	67.67±1.53	69.00±5.00	71.17±3.62	72.00±4.92	73.33±4.25
Flow+CEM	–	83.00±3.77	85.67±3.06	86.33±2.52	89.17±2.57	88.17±3.06
Gaussian+CEM	–	81.67±5.13	85.33±3.25	86.00±4.50	87.83±3.21	87.83±3.62
BC+CEM	–	82.00±4.27	85.17±3.55	86.17±3.51	85.50±3.97	85.50±3.97
CEM	0.17±0.29	54.50±1.80	63.00±3.61	70.00±3.46	74.17±2.08	76.67±1.15
DINO-WM (Straightened)						
Flow	29.67±2.08	64.83±2.02	66.33±2.08	70.33±0.76	75.83±1.76	77.67±1.26
Gaussian	46.67±5.51	59.33±2.75	61.50±3.97	66.50±3.50	71.83±2.93	74.17±2.84
BC	47.83±6.01	64.83±3.21	67.33±3.01	70.17±3.75	70.00±4.27	72.00±4.77
Flow+CEM	–	84.17±2.89	84.67±1.04	85.67±1.44	88.33±1.61	89.17±1.61
Gaussian+CEM	–	84.17±5.80	86.17±4.01	87.33±3.01	88.33±1.76	89.00±2.00
BC+CEM	–	81.17±3.62	82.33±3.40	84.33±3.33	85.00±3.50	85.00±3.61
CEM	0.17±0.29	54.00±5.41	64.17±3.18	71.00±2.29	78.50±1.50	76.17±2.47

B.9 Success rates of other planners

For completeness we also evaluate three further sampling-based planners as uninformed baselines. Improved CEM (iCEM) (Pinneri et al., 2020) augments CEM with temporally correlated (colored) noise for the action samples (exponent $\beta = 2$), a set of elites carried across iterations (5 elites), and an exponential moving average of the mean and standard deviation (momentum 0.1). Model Predictive Path Integral control (MPPI) (Williams et al., 2015) replaces CEM's hard elite selection with a soft, information-theoretic update, weighting every sampled sequence by the exponential of its negative cost and setting the next mean to the resulting cost-weighted average (temperature 0.5). Predictive Sampling (Howell et al., 2022) is the simplest of the three, drawing a single round of N Gaussian samples (scale 1) around zero and keeping the lowest-cost one without any refinement. It is worth noting that a prior used on its own is essentially Predictive Sampling with its zero-centered Gaussian replaced by draws from the learned prior. Table 16 reports their success rates. Among the three, iCEM is generally the strongest, but plain CEM outperforms it in five of the six configurations, by up to 10.0 pp on OGBench-Cube, while MPPI and Predictive Sampling trail both, most clearly on the contact-rich Push-T task.

Table 13: Replanning cadence R against cost weighting b on short-horizon Push-T for PLDM and the two DINO-WM configurations.

R	b	Flow	Gaussian	BC	CEM
PLDM					
5	0.5	51.83±3.01	60.67±5.35	60.83±5.20	43.83±1.76
5	1	69.17±5.01	70.50±2.65	70.33±2.08	57.67±2.36
5	2	77.83±1.44	74.67±2.36	74.67±4.25	71.33±4.37
5	∞	**83.17±1.26**	**80.50±2.18**	**79.00±3.50**	**78.17±5.51**
2	0.5	59.67±2.08	64.00±1.32	**57.67±4.25**	43.67±2.25
2	1	75.17±2.75	**66.33±3.21**	52.00±5.63	59.83±3.88
2	2	**76.00±3.61**	56.50±1.73	43.67±3.33	**73.17±4.80**
2	∞	28.33±4.54	32.67±4.93	25.50±4.77	34.67±2.36
1	0.5	62.83±1.61	58.83±0.29	**45.33±3.55**	46.50±1.32
1	1	**78.50±0.50**	**61.17±1.04**	43.33±4.54	62.67±3.82
1	2	77.00±1.32	46.00±1.32	29.83±5.30	**73.17±5.53**
1	∞	23.83±3.69	26.00±1.50	18.33±4.37	28.17±1.53
DINO-WM					
5	0.5	36.83±1.76	34.67±3.88	37.83±1.76	38.83±6.33
5	1	55.83±3.01	51.17±2.47	54.50±4.77	57.00±1.32
5	2	73.83±2.25	68.33±2.47	70.50±1.73	76.83±2.02
5	∞	**86.33±1.61**	**79.50±2.29**	**80.33±3.79**	**83.50±1.00**
2	0.5	56.83±4.65	**55.83±0.76**	**53.67±2.02**	50.67±5.01
2	1	72.33±4.25	53.83±4.54	49.83±2.75	68.17±3.25
2	2	**72.83±1.76**	44.17±1.04	42.67±0.76	**71.67±3.25**
2	∞	10.17±1.61	24.67±5.11	20.50±3.04	22.50±3.00
1	0.5	66.33±5.20	**60.33±2.93**	**52.50±1.73**	55.33±7.15
1	1	**80.83±0.76**	55.17±3.21	46.67±3.06	72.17±3.69
1	2	76.67±0.29	37.33±1.89	35.00±3.77	**76.67±0.29**
1	∞	12.33±1.15	17.33±1.15	19.83±2.75	27.00±1.80
DINO-WM (Straightened)					
5	0.5	35.33±5.06	34.33±3.82	38.00±3.97	38.33±2.84
5	1	56.33±2.93	48.17±3.33	57.33±2.47	58.50±4.77
5	2	71.67±0.76	67.83±2.02	71.83±5.92	76.67±3.75
5	∞	**85.67±2.36**	**81.00±2.65**	**78.50±5.22**	**85.50±1.50**
2	0.5	55.83±6.90	50.83±3.75	**46.50±1.73**	48.67±6.29
2	1	**74.33±2.02**	**51.50±2.18**	44.33±2.36	67.67±5.53
2	2	72.00±2.60	37.83±2.93	35.83±1.61	**71.67±6.37**
2	∞	10.50±2.18	18.00±1.32	18.50±3.12	21.67±2.47
1	0.5	61.50±6.24	**54.50±3.04**	**43.00±2.65**	52.50±4.82
1	1	79.17±3.69	46.00±5.41	37.17±1.04	74.00±5.57
1	2	**82.67±3.06**	30.67±2.93	27.50±5.57	**77.50±2.78**
1	∞	10.33±1.26	15.50±0.87	14.00±2.78	27.17±1.89

Table 14: Evaluating LWM and PLDM on the short-horizon OGBench-Cube task using a mixture of candidate chunks sampled from the Flow prior and sampled from a Gaussian.

Flow Ratio	0%	25%	50%	75%	100%
LWM	71.83±2.02	99.33±0.29	99.33±0.76	99.33±0.29	99.50±0.00
PLDM	56.50±4.36	98.33±1.15	99.33±0.29	99.17±0.76	99.33±1.15

B.10 Data Efficiency

Because a prior is trained only on offline demonstrations, its practical value also depends on how much data it requires. Figure 9 reports the success rate of all three priors, on their own and seeding CEM, when each is trained on 25%, 50%, 75% or 100% of the available Push-T demonstrations.

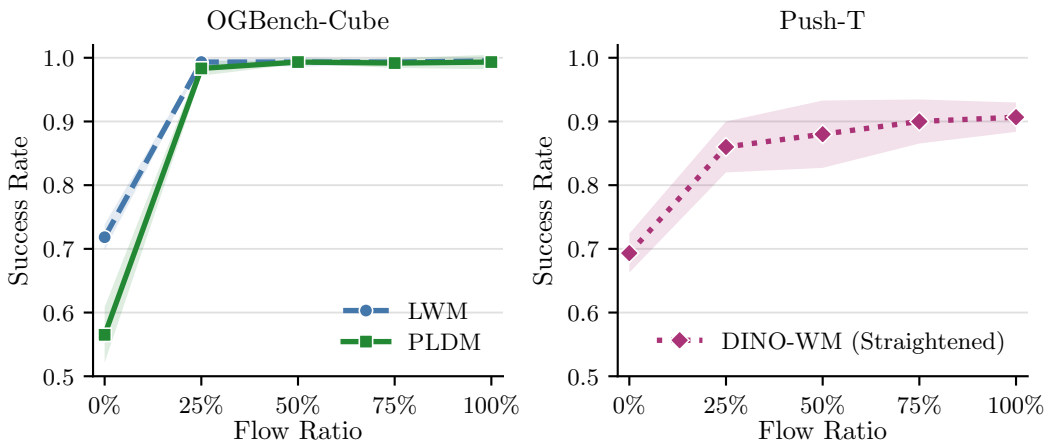


Figure 8: **HybridCEM.** Evaluating LWM and PLDM on the short-horizon OGBench-Cube task and DINO-WM (Straightened) on the short-horizon Push-T task using a mixture of candidate chunks sampled from the Flow prior and sampled using a Gaussian. Shades indicate $\pm 1\sigma$. The exact success rates can be found in Tables 14 and 15.

Table 15: Evaluating two DINO-WM versions on the short-horizon Push-T task using a mixture of candidate chunks sampled from the Flow prior and sampled from a Gaussian.

Flow Ratio	0%	25%	50%	75%	100%
DINO-WM	70.00±6.00	86.00±5.29	88.00±5.29	90.67±5.03	90.00±4.00
DINO-WM (Straightened)	69.33±3.06	86.00±4.00	88.00±5.29	90.00±3.46	90.67±2.31

Table 16: Success rate in % of three additional uninformed samplers.

Planner	**iCEM**	**Pred.Samp.**	**MPPI**
LeWorldModel			
Two-Room	87.67±1.61	86.83±1.61	76.67±3.33
Push-T	82.83±1.15	43.67±2.52	61.50±1.32
OGBench-Cube	61.83±4.93	55.83±3.51	51.67±3.01
PLDM			
Two-Room	95.20±2.60	91.17±0.58	77.67±0.58
Push-T	74.17±2.93	45.67±2.75	64.17±1.89
OGBench-Cube	51.00±6.54	52.50±2.78	49.50±2.29

Every curve is close to flat, so almost the entire benefit obtained with the full dataset is already retained with a quarter of it. The Flow is the flattest of the three, moving by under 2 percentage points between 25% and 100% under either world model, whereas the two other priors still gain a little from more data, by up to 5.8 percentage points for BC under PLDM. Once a prior seeds CEM the fraction stops mattering altogether: All six refined curves sit within 2 percentage points of their full-data value at every fraction. The ordering of the priors by success rate is largely unaffected by the amount of data.

B.11 Training time of the priors

Table 18 reports the time to train each prior in minutes, counted as the sum of the per-epoch durations over the ten training epochs. The one-off precomputation of the world-model embeddings is excluded. All three priors share the data, the optimizer and the epoch count, so the remaining difference is attributable to the objective. On the larger datasets the Flow costs four to eight times

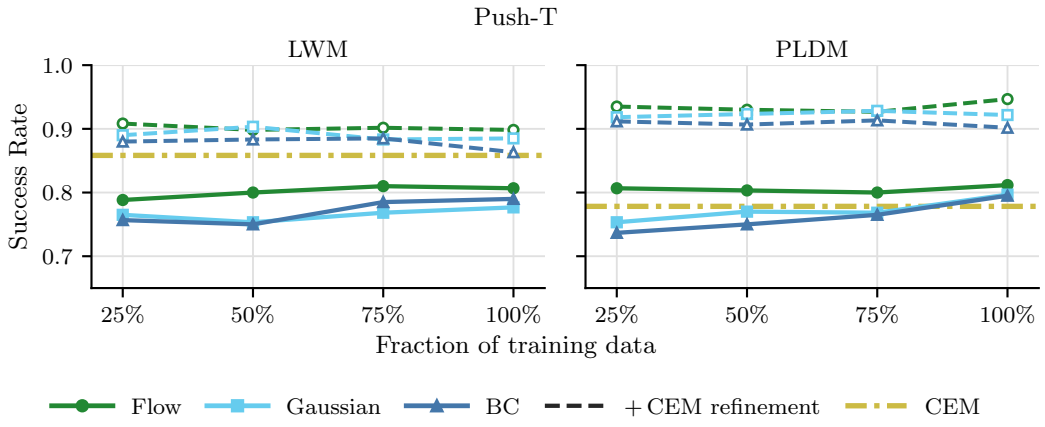


Figure 9: **Data efficiency.** Success rate on short-horizon Push-T with different fractions of training episodes for each prior. The underlying values are in Table 17.

Table 17: Success rate in % on short-horizon Push-T against the fraction of the training episodes each prior was fitted on. This table holds the data behind Figure 9. CEM is listed once per world model because it does not depend on the fraction.

Training data	25%	50%	75%	100%
LWM				
Flow	78.83±0.76	80.00±3.12	**81.00±2.60**	80.67±1.61
Gaussian	76.50±3.77	75.33±4.01	76.83±2.75	**77.67±3.18**
BC	75.67±4.51	75.00±3.91	78.50±4.92	**79.00±2.65**
Flow+CEM	**90.83±1.53**	89.83±2.84	90.17±2.02	89.83±2.84
Gaussian+CEM	89.00±2.65	**90.33±2.75**	88.33±4.04	88.50±2.60
BC+CEM	88.00±2.60	88.33±1.61	**88.50±3.04**	86.33±1.89
CEM		85.83±0.58		
PLDM				
Flow	80.67±3.62	80.33±4.48	80.00±2.29	**81.17±2.02**
Gaussian	75.33±4.37	77.00±1.32	76.83±2.36	**79.67±2.84**
BC	73.67±3.62	75.00±3.04	76.50±2.78	**79.50±4.36**
Flow+CEM	93.50±1.32	93.00±2.00	92.67±0.76	**94.67±1.15**
Gaussian+CEM	91.83±2.57	92.33±2.52	**92.83±2.02**	92.17±2.52
BC+CEM	91.17±2.31	90.67±3.01	**91.33±3.01**	90.17±3.62
CEM		77.83±6.03		

as much to train as either policy head. On the smallest datasets all three are under a minute and the difference disappears. Every prior in this paper trains in at most eleven minutes on one GPU.

Table 18: Training time of each prior in minutes, measured as the sum of the per-epoch durations over the 10 training epochs. The one-off precomputation of the world-model embeddings is excluded. Values in parentheses are the long-horizon (H=10) priors.

World model	Environment	Flow	Gaussian	BC
LWM	Two-Room	1.6	0.7	0.7
	Push-T	10.1 (7.3)	2.5 (1.8)	2.3 (1.8)
	OGBench-Cube	10.2 (8.2)	2.0 (1.9)	1.9 (2.0)
PLDM	Two-Room	1.6	0.7	0.7
	Push-T	10.3 (7.2)	2.5 (1.9)	2.3 (1.7)
	OGBench-Cube	10.2 (8.2)	1.9 (2.0)	1.8 (1.9)
DINO-WM	Wall	0.2	0.2	0.2
	Push-T	8.2 (5.9)	1.1 (0.9)	1.1 (0.8)
	PointMaze-UMaze	0.4	0.2	0.2
	PointMaze-Medium	0.7 (0.5)	0.3 (0.3)	0.3 (0.3)
DINO-WM (Straightened)	Wall	0.2	0.2	0.2
	Push-T	8.1 (5.9)	1.1 (0.9)	1.1 (0.9)
	PointMaze-UMaze	0.4	0.2	0.2
	PointMaze-Medium	0.7 (0.5)	0.3 (0.3)	0.3 (0.3)