CriticJEPA: World Models Are Better Critics Than Optimizers

Xvyuan Liu[1,*], Jianjie Fang[1,*], Honglin Zhang[1,*], Zhaolu Wang[2], Xin Zhang[3], Wei Wu[3], Fengli Xu[1], Chen Gao[1,†], Yong Li[1,†]

[1]Tsinghua University [2]Zhongguancun Academy [3]Manifold AI

[*]Equal contribution [†]Corresponding authors

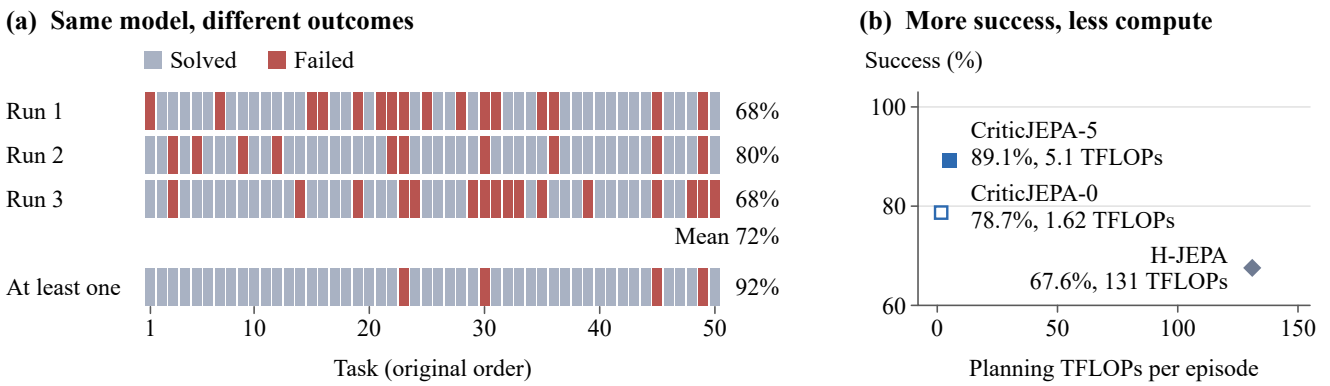


Figure 1. **CriticJEPA improves success with less planning compute. (a)** Three H-JEPA restarts on the same 50 AntMaze tasks in original order. Mean single-run success is 72%; the union row covers 92%. **(b)** Success and compute across three independently trained models on Standard AntMaze. Suffixes denote refinement steps per level; candidate counts follow Tab. 3.

Abstract

Visual world models enable goal-directed planning by predicting action outcomes. Gradient-based planners optimize actions to lower predicted goal cost, but continued optimization can exploit prediction errors after execution stops improving. Comparing plans requires informative relative scores; improving them also requires reliable update directions. Exact-state execution reveals that a frozen hierarchical world model ranks goal-conditioned proposals even when its rankings among outputs of prolonged gradient search are uninformative. Effective planning pairs goal-directed candidates with predictions that distinguish their outcomes. We introduce CriticJEPA*, which learns goal-conditioned flow proposals from reward-free trajectories and uses frozen rollouts to select among them at every hierarchy level. Forward selection forms a complete planner; short anchored updates refine candidates before selection. Across visual navigation and manipulation, CriticJEPA increases success at a fraction of the planning compute, transferring shared settings across independently trained models, hierarchy depths, and unseen tasks while keeping every world-model weight fixed.*

1. Introduction

Visual world models predict how the environment changes under a sequence of actions, enabling agents to plan toward goal images in a learned latent space [4, 18, 22, 25, 32]. Gradient-based planners use these predictions both to evaluate plans and to construct them, repeatedly updating candidate actions to reduce predicted goal cost. H-JEPA [30] extends this process across temporal scales, from high-level subgoals to low-level controls.

With the world model fixed, H-JEPA restarts solve different sets of tasks, as shown in Fig. 1a. The same planning stack reaches more goals across runs than a single search reliably recovers. Tracing the action updates reveals a mismatch between predicted improvement and execution: early updates improve both, but further updates lower predicted endpoint error after execution stalls, as shown in Fig. 2a. Repeated optimization can exploit prediction errors, favoring actions that look increasingly successful to the model without moving closer to the goal in execution.

Evaluating plans and optimizing them require different properties from the predictor. Choosing among given plans requires scores that preserve their relative quality. Improving a plan through gradients also requires reliable directions as the actions change. Errors that leave the ordering of a candidate set intact can still redirect optimization toward artificially favorable predictions. A predictor can distinguish promising plans even when further optimization stops improving their execution.

This distinction appears when we execute alternative plans from identical simulator states, holding the goal and

horizon fixed. The same frozen predictor gives informative rankings of learned goal-conditioned proposals and near-chance rankings of H-JEPA search outputs, as shown in Tab. 1. Goal-conditioned proposals yield better selected outcomes than retrieved training actions: retrieval matches the starting latent state, whereas the proposer also uses the current goal to generate alternatives. A critic's value depends on both how well it compares plans and which plans it is given.

These observations lead to *CriticJEPA*, which separates the generation of plans from their evaluation. Goal-conditioned flow heads [19, 20] learn action sequences from reward-free trajectories paired with the states they actually reach [2, 13]. Frozen world-model rollouts then select among the generated alternatives. Short gradient updates capture the early gains in execution, with an anchor penalizing departure from each proposal.

The design operates throughout the temporal hierarchy, as illustrated in Fig. 3: selected upper-level plans supply subgoals, and bottom-level plans supply controls. Forward selection constructs complete plans; the refinement variant updates every candidate before selection. In contrast to prior-guided iterative search in flat JEPA models [10, 29], generation and frozen selection form the planner at every level, including the macro-actions that determine subsequent subgoals.

Across visual navigation and manipulation, CriticJEPA reaches more goals with substantially less planning compute, as shown for AntMaze in Fig. 1b. Our contributions are:

- We demonstrate through exact-state AntMaze execution that a frozen predictor distinguishes goal-conditioned proposals when prolonged gradient search produces uninformative rankings. The candidate source determines how ranking translates into executed outcomes.
- We introduce CriticJEPA, a complete hierarchical planner that pairs goal-conditioned flow proposals with frozen rollout selection and short anchored updates for local refinement.
- We demonstrate success and compute gains across visual navigation and manipulation, transferring shared planning settings across independently trained models, hierarchy depths, and unseen AntMaze tasks.

2. Related Work

Planning with self-supervised world models. Learning latent dynamics enables planning from high-dimensional visual observations [12, 14, 15]. Joint-Embedding Predictive Architectures (JEPA) [3, 5, 8, 18] learn predictive representations without reconstructing pixels. Visual world models including PLDM [25], DINO-WM [32], V-JEPA 2 [4], NWM [7], and LeWorldModel [22] support goal-image planning from reward-free trajectories; Terver *et al*. [27] study the design choices behind such planners. HWM [31] and H-JEPA [30] extend latent prediction across temporal scales. With pretrained H-JEPA hierarchies, we show that changing how plans are generated and judged improves control without changing the learned dynamics.

Model exploitation in trajectory optimization. Optimizing actions against learned dynamics can favor out-of-distribution inputs on which predictions are erroneously optimistic [12, 16], paralleling adversarial examples for classifiers [26]. In a two-level LeWorldModel, Mind the Gap [9] observes that CEM over macro-actions exploits the high-level predictor and regularizes search toward the data. Our exact-state execution shows that a frozen predictor distinguishes goal-conditioned proposals when rankings among its gradient-search outputs are uninformative. The candidate source determines how these comparisons translate into the quality of the selected action.

Generative priors and amortized proposals. Policy priors accelerate model-based search [15, 28], while diffusion models generate trajectories and actions [1, 11, 17]. Recent work brings learned priors to flat JEPA world models: PRISM [29] adds a Gaussian prior to iterative MPPI, SAGE [10] trains subgoal and action generators to initialize CEM, and LePlanner [6] amortizes planning into a recurrent controller. In CriticJEPA, proposal generation and frozen rollout selection extend throughout the temporal hierarchy, including the macro-actions that determine subgoals. Forward evaluation alone supports planning at every level, and anchored refinement spends further computation on short corrections to the proposals.

3. Diagnosing the Search Bottleneck

Planning with a learned predictor couples the model's predictions to the actions produced by search. We separate these effects through exact-state execution with one frozen three-level H-JEPA hierarchy [30] on Visual AntMaze. Upper levels choose subgoals, and the bottom level supplies controls. The H-JEPA planner initializes candidates from Gaussian distributions and minimizes predicted goal-distance cost with up to 90 AdamW steps [21].

3.1. Predicted Improvement, Stalled Execution

At each saved state, we fix the current level-m subgoal $g^{(m)}$. For each candidate, we predict and execute the same replanning window: 10 primitive steps, or 5 at the shortest horizon. Let $\tilde{z}^{(m)}_{\text{end},i}$ be the predicted endpoint projected through frozen encoders into the target space, and $E^{(m)}(o^{(i)}_{\text{end}})$ the reached image encoded in that space. The window-level diagnostic compares

$$\begin{aligned} C^{\text{win}}_{\text{pred},i} &= \|\tilde{z}^{(m)}_{\text{end},i} - g^{(m)}\|_2^2, \\ C^{\text{win}}_{\text{true},i} &= \|E^{(m)}(o^{(i)}_{\text{end}}) - g^{(m)}\|_2^2. \end{aligned} \tag{1}$$

Table 1. **Goal-conditioned proposals yield the best selected actions.** At each of 300 AntMaze states, eight candidates per source are executed from the same simulator state. *Rank* ρ: within-state Spearman correlation of predicted and realized endpoint errors. *Top-1*: how often the model selects the best of the eight; chance is 12.5%. *Selected error*: mean realized error of the selected action. *Mean optimism*: realized minus predicted endpoint error, averaged over all candidates.

Candidates	Ranking		Selected error↓	Mean optimism↓
	Rank ρ ↑	Top-1 (%)↑		
Optimized	-0.03	12.0	24.3	17.4
Dataset	**0.41**	**27.0**	30.1	1.9
Flow proposals	0.28	22.0	**23.0**	**1.2**

The simulator is reset to the same full state before each execution. These endpoint errors measure progress over the replanning window; the planner optimizes the full-horizon cost in Sec. 4.

Figure 2a tracks the gradient planner across iterations $k \in \{0, 10, 30, 90\}$ on 720 replanning states. True executed error falls from 25.3 to 23.0 during the first 10 steps, then stalls at 22.8 after 30 steps and 23.4 after 90. Predicted error keeps dropping, while optimism nearly doubles from 8.6 to 16.7. The extra optimization chiefly improves the model score rather than the executed outcome.

Comparison and optimization impose different requirements. For matched predicted and realized costs, write $C_{\text{pred}}(a) = C_{\text{true}}(a) - \epsilon(a)$, where ϵ denotes optimism. A local differentiable view gives

$$-\nabla_a C_{\text{pred}}(a) = -\nabla_a C_{\text{true}}(a) + \nabla_a \epsilon(a). \qquad (2)$$

The update direction contains a term that improves the true objective and a term that increases model optimism. This gives search a route to lower predicted cost without better execution, matching the divergence in the iteration probe.

Comparison makes a different demand. On a fixed candidate pool, a shared prediction-error offset leaves every ranking unchanged. What matters is the error difference between the supplied plans. Gradient search also responds to how the error changes as actions move; even small value errors can vary sharply. Ranking given plans and supplying update directions are distinct predictive abilities.

3.2. Candidate Generation Shapes Selection

This difference between comparison and optimization makes candidate generation central to model-based selection. We execute plans from different sources at identical saved simulator states, with the same subgoal and horizon, and compare the predicted and realized endpoint errors.

We sample 15 replanning states from each of 20 validation episodes, saving the full MuJoCo physics state and fixing the current level-m subgoal $g^{(m)}$ at all 300 states. We evaluate eight level-1 candidates from each of three sources. **Gradient search** supplies the winners of eight independent H-JEPA AdamW solves, each with 100 initializations and up to 90 steps. **Training dataset** supplies eight recorded action sequences from the 500 nearest-neighbor level-1 latent states in the training set. **Flow proposals** are eight direct samples from the goal-conditioned proposer in Sec. 4.

The frozen predictor distinguishes candidates from recorded trajectories, as shown in Tab. 1. Its rankings correlate with executed outcomes at ρ=0.41, and top-1 selection reaches 27.0%, over twice the chance rate. On H-JEPA search outputs, ranking correlation is ρ=-0.03 and top-1 selection is at chance level at 12.0%. Mean optimism, averaged over all candidates, rises from 1.9 for recorded actions to 17.4 for search outputs. The predictor's ability to compare actions thus survives on a different candidate distribution.

Goal-conditioned proposals deliver the best executed choices and the lowest mean optimism: 23.0 and 1.2, respectively. Flow proposals combine informative rankings at $\rho = 0.28$ with lower selected error than retrieval, which yields $\rho = 0.41$ and error 30.1. Retrieval matches only the starting latent state, whereas the flow also conditions on the current goal. Ranking agreement summarizes the whole pool; control depends on what the selected action achieves. Candidate generation must therefore support both informative comparisons and progress toward the target.

The selected outcome separates into the best executed cost available in the pool and the extra cost of choosing another candidate. For a fixed pool, the selected plan has no higher predicted cost than the true best plan, so the extra selection cost is bounded by the range of prediction errors across the pool. A shared prediction bias cancels in comparisons, leaving the error differences that affect which action is selected. This gives proposal learning and model evaluation complementary roles: supplying goal-directed alternatives and preserving the comparisons that identify which to execute.

Forward selection among 100 zero-gradient CriticJEPA proposals achieves a true error of 20.2 on the 720 states of Fig. 2a, below every tested step count of H-JEPA gradient search.

3.3. Search Variability Across the Hierarchy

The consequences of search extend through the hierarchy: upper-level choices set targets for lower-level control, so changing either search can change task completion. With the frozen model and initial states held fixed, three H-JEPA restarts on the 50 official tasks produce the outcomes in Fig. 1a. Each run has a 350-step budget and independently resampled initial actions. Individual runs solve 68%, 80%, and

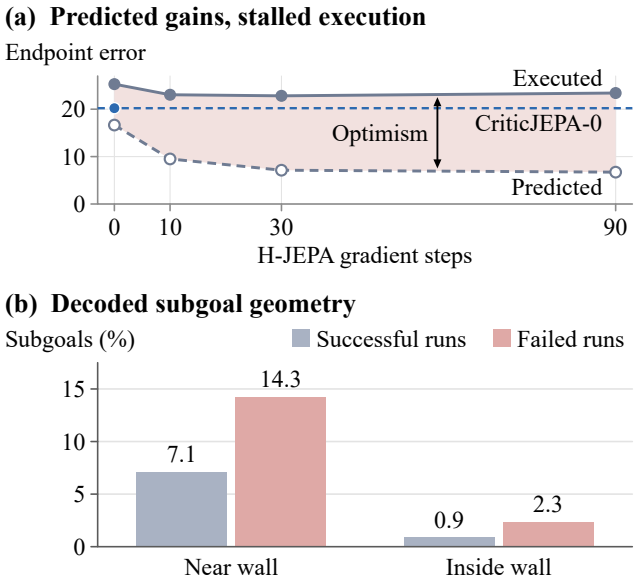


Figure 2. **Optimizing predictions can diverge from execution.** **(a)** Predicted and executed endpoint errors at 720 AntMaze states with one frozen model. Shading shows optimism: executed minus predicted error. The blue reference selects among 100 zero-gradient proposals. **(b)** Decoded L3 subgoals on the 24 tasks whose outcomes change across restarts. Near-wall subgoals lie within 0.8 m of a wall.

68% of tasks, averaging 72.0%, while their union reaches 46 of 50 tasks, or 92.0%. Twenty-two tasks are solved by all three runs, and 48% change outcome across restarts. The gap between individual success and combined coverage identifies a consistency problem: the unchanged planning stack reaches many more goals than a typical run finds. The challenge is to recover this successful behavior reliably from the same frozen model.

Increasing the level-1 pool fivefold to $N_1{=}500$ gives 70.0% mean success across three further runs. Combining all six runs expands coverage to 48 of 50 tasks, or 96.0%. Successful behavior is recovered across many tasks by the unchanged planning stack, but a larger initial pool alone leaves single-run success near its original level.

Both high-level and low-level search flip task outcomes. Varying search randomness separately at each level reveals instability at both timescales. With lower-level random draws fixed, changing only top-level initialization flips outcomes on 44% of the 50 tasks. With upper-level draws fixed, changing only bottom-level initialization flips 38%. The outcome changes at both timescales connect candidate generation to the complete control loop: the upper search chooses where to go next, and the lower search determines how to get there.

At the top level, unsuccessful search has a geometric signature. We train a linear probe on real observations to decode level-3 latents into maze coordinates, obtaining $R^2 \approx 0.998$. Figure 2b compares successful and failed runs on the 24 tasks whose outcomes change across restarts. In failed runs, 14.3% of subgoals decode within 0.8 m of a wall and 2.34% inside walls. The corresponding rates for successful runs are 7.1% and 0.89%. Failed runs thus more often produce subgoals decoded near or inside walls. Across all 50 tasks, CriticJEPA-0 reduces the fraction decoded inside walls from 2.31% to 0.31%. Coordinates are used for this geometric evaluation; training and planning operate in the frozen latent and action spaces.

The hierarchy therefore needs goal-directed alternatives for both subgoal selection and primitive control. Recorded trajectories provide state-to-goal connections at each timescale, from which a generator can learn the plans that the frozen predictor will compare.

4. CriticJEPA

CriticJEPA generates goal-directed alternatives at every temporal level and uses frozen world-model predictions to choose among them. Short anchored updates refine every candidate before selection in the gradient-based variant. Figure 3 shows the computation, with its execution order in Tab. 2.

At level ℓ, the frozen world model has latent state $z_t^{(\ell)}$, predictor $F^{(\ell)}$, and temporal stride Δ_ℓ. A plan $a_{1:h_\ell}^{(\ell)}$ contains control blocks at the bottom level and learned macro-actions above it. The predictor rolls out $\hat{z}_k^{(\ell)} = F^{(\ell)}(z_0^{(\ell)}, a_{1:k}^{(\ell)})$. A target $g^{(m)}$ can arrive from level $m \geq \ell$. Frozen upper encoders project the rollout into that target space using the H-JEPA temporal windows, yielding $\tilde{z}_k^{(m)}$; no projection is needed when $m = \ell$.

Propose: condition actions on state, target, and timescale. Several action sequences can connect a state to the same target. We learn a distribution of plans conditioned on the goal and timescale, using a flow proposer at each level [19, 20]:

$$a_{1:h_\ell}^{(\ell)} \sim \pi_\theta^{(\ell)}(\cdot \mid c), \quad m \geq \ell, \\ c = \big(z_0^{(\ell)}, z_0^{(m)}, g^{(m)} - z_0^{(m)}, h_\ell, m\big). \tag{3}$$

One sample produces an entire level-ℓ plan spanning $h_\ell \Delta_\ell$ primitive steps. The condition supplies the current state in the planning and target spaces, the displacement to the goal, and embeddings for the horizon and target level. When horizon decay skips an upper level, the H-JEPA rule targets the goal image in the skipped upper space for AntMaze/FourRoom and in the active level's space for Cube/Push-T. An AntMaze level-1 proposer can thus receive a level-3 goal while still producing a short control block.

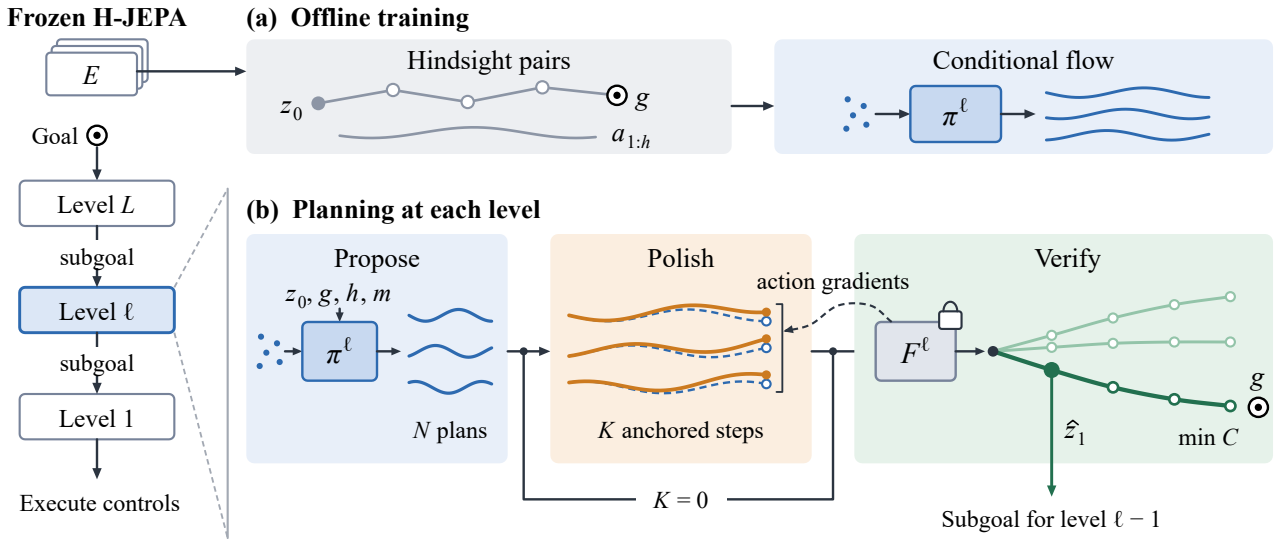


Figure 3. **CriticJEPA: proposals supply plans; the world model judges them. (a)** Frozen encoders turn observed training trajectories into hindsight pairs; only the conditional flow heads are trained. **(b)** At each level, the trained head proposes N plans conditioned on state, target, horizon h, and target level m. K anchored updates refine the entire pool through frozen-predictor action gradients. Final rollout selection uses the H-JEPA cost C; $K = 0$ bypasses refinement. The chosen upper-level plan's first predicted state becomes the next subgoal, while the bottom level executes controls. Curves are schematic.

We train on the world model's reward-free training trajectories using hindsight goals [2, 13]. A real action segment $x_1 = a^{(\ell)}_{t:t+h-1}$ is paired with its starting latent and the latent it actually reaches after $h\Delta_\ell$ primitive steps. Short segments are padded and masked. For Gaussian x_0 and uniform $\tau \in [0, 1]$, we interpolate $x_\tau = (1-\tau)x_0 + \tau x_1$ and train a conditional velocity field:

$$\mathcal{L}^{(\ell)}_{\text{flow}} = \mathbb{E}\Big[\big\|v^{(\ell)}_\theta(x_\tau, \tau, c) - (x_1 - x_0)\big\|^2_{\text{valid}}\Big]. \quad (4)$$

The squared error uses the valid, unpadded action entries. Hindsight pairs connect each starting state to an observed outcome. Independent noise samples generate alternatives for the critic to compare.

All encoders, action encoders, and predictors remain frozen. At inference, four Euler steps generate N_ℓ candidates in parallel, clipped to the H-JEPA action bounds.

Verify: use the world model as a rollout critic. Once proposals are available, the frozen predictor chooses among them using its relative judgments. For the current state, target, and upper-level plan, we retain the H-JEPA rollout cost:

$$C^{(\ell)}(a) = \sum_{k \in \mathcal{T}_\ell} \big\|\tilde{z}^{(m)}_k - g^{(m)}\big\|^2_2 + \beta_\ell R^{(\ell)}_{\text{act}}(a),$$
$$i^\star = \arg\min_{i=1,\dots,N_\ell} C^{(\ell)}(a^{(i)}). \quad (5)$$

Here $\mathcal{T}_\ell$ contains the final scoring steps used by H-JEPA. The action-consistency term $R^{(\ell)}_{\text{act}}$ sums squared differences between actions encoded into the upper action space and the selected upper plan over their matching window. Its weight is $\beta_1 = 0.2$ in Cube and Push-T when an upper action target is present, and zero otherwise. The highest level targets the encoded goal image.

For **CriticJEPA-0**, one batched evaluation selects the lowest-cost proposal. Candidates run in parallel, while each rollout advances through the horizon. Higher levels pass the selected plan's first predicted state downward; the bottom level executes its control block. This forward selection forms a complete planner throughout the hierarchy, using the original dynamics and temporal abstraction.

Polish: anchor short updates to the proposals. The early gains in the iteration probe motivate short local refinement of the proposed plans. For **CriticJEPA-K**, we retain all N_ℓ proposals and refine them in parallel for K AdamW steps before the final critic selection. Let $a^{(i)}_{\text{prop}}$ be proposal i and d_a the number of action entries. We optimize

$$\mathcal{J}^{(\ell)}(a^{(i)}) = C^{(\ell)}(a^{(i)}) + \frac{\lambda}{d_a}\|a^{(i)} - a^{(i)}_{\text{prop}}\|^2_2, \quad (6)$$

with $\lambda = 0.1$ in the normalized action space. The anchor penalizes departures from the proposal while allowing local corrections. Final selection uses the rollout cost $C^{(\ell)}$

Table 2. **One hierarchy, two modes.** $K = 0$ uses forward verification alone; $K > 0$ polishes the full candidate set before selection. Target-space projections, horizon decay, and execution windows follow H-JEPA.

Algorithm 1: CriticJEPA at each replanning window

Input: frozen hierarchy, proposers $\pi_\theta^{(\ell)}$, counts N_ℓ, horizons h_ℓ, polish steps K

Encode the current observation at all levels; initialize target $g \leftarrow E^{(L)}(o_g)$ and target level $m \leftarrow L$

for $\ell = L, \dots, 1$ **do**

  if $\ell > 1$ and $h_\ell = 1$: skip this level using the H-JEPA goal-space rule; continue

  Form condition c from Eq. (3); sample N_ℓ plans from $\pi_\theta^{(\ell)}(\cdot \mid c)$

  If $K > 0$, polish *all* plans for K steps on Eq. (6)

  Verify all final plans; select $i^\star = \arg\min_i C^{(\ell)}(a^{(i)})$

  If $\ell > 1$, set $g \leftarrow F^{(\ell)}(z_0^{(\ell)}, a_1^{(i^\star)})$ and $m \leftarrow \ell$

Execute the selected bottom-level plan for the H-JEPA execution window and replan

without the anchor penalty: the anchor shapes local refinement, while the critic compares the resulting plans. This gives test-time compute two distinct roles. Candidate count controls the breadth of comparison by supplying alternatives to the critic; polish steps control local adaptation around the original proposals.

Where planning computation goes. Zero-gradient planning uses four proposer evaluations and one predictive rollout per candidate. Each polish step adds a rollout and its action-gradient computation. This applies to macro-actions above and control blocks below. Including proposer cost, the savings come from replacing repeated world-model optimization with a short generative pass and a final comparison.

5. Experiments

5.1. Experimental Setup

Benchmarks and frozen world models. We evaluate pretrained H-JEPA hierarchies [30] across four control domains, using three independently trained world models for each environment and hierarchy depth. **Visual AntMaze** uses a three-level model for quadruped maze navigation from 64×64 top-down pixels and proprioception, with a 350-step budget. **Push-T** uses two- and three-level models for contact-rich planar pushing of a T-block to a target pose, with a 90-step budget. **OGBench Cube** uses both hierarchy depths for 6-DoF robotic cube manipulation [23], with a 30-step budget. **FourRoom** uses a three-level model for multi-room navigation through narrow doorways, with a 210-step budget. All 18 world models remain frozen; each model's flow proposers learn from its own encoded training trajectories. On Visual AntMaze, three planning restarts on each of the 50 official tasks give 450 episodes per method across the three models. Diagnostics and component ablations hold one AntMaze model fixed to isolate changes in planning. A further 50 tasks generated after method development form the *Unseen* set, with 150 episodes per method. Each remaining setting uses 50 tasks per world model, also giving 150 episodes per method.

Methods and metrics. The official **H-JEPA** gradient planner runs up to 90 AdamW steps per level with its original early stopping. To separate generation from selection, we include **Proposal**, a single-proposal ablation that executes one deterministic flow sample per level at each replanning step without rollout verification. **CriticJEPA-0** verifies N_ℓ candidates in one batched rollout; **CriticJEPA-K** applies K anchored refinement steps to every candidate before selection. On AntMaze the H-JEPA candidate counts are $100/100/400$: L1 samples a standard Gaussian, and upper levels sample Gaussians fitted to training macro-actions with H-JEPA action clipping. We report success as mean $\pm$ standard error across the three world models and median environment steps in successful episodes. Planner TFLOPs per episode count encoder, proposer, and predictor operations, including refinement backward passes.

Training and shared planning settings. The velocity field is a six-block residual MLP of width 512 with AdaLN conditioning [24]. Training uses 25 epochs, batch size 1024, and a cosine learning-rate schedule starting at 3×10^{-4}; the final epoch is used for every model.

All experiments retain the H-JEPA hierarchy, horizon schedule, and execution windows. Stochastic proposals use temperature 1.0; the deterministic ablation starts the flow at zero noise. We fix four Euler steps and $\lambda = 0.1$. The cross-environment comparison uses $N_1/N_2/N_3 = 100/100/200$ for CriticJEPA-0 and $25/25/100$ with $K = 5$ for CriticJEPA-5, using the first two counts for two-level models. We establish this configuration on one AntMaze model and transfer it unchanged to the other models, environments, and unseen tasks. The AntMaze budget sweep varies only the stated candidate counts and polish steps.

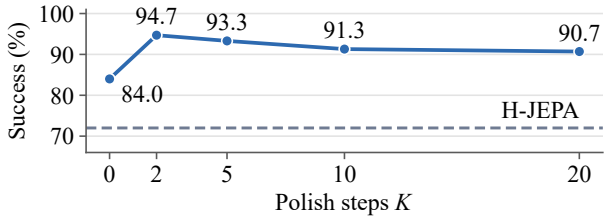
5.2. Main Results on Visual AntMaze

Frozen rollout selection turns generated alternatives into more successful control. In Tab. 3, selecting from a proposal pool raises Standard success from 44.7% for a single proposal to 78.7% for CriticJEPA-0. The latter reaches 87.3% on Unseen tasks at 1.62 TFLOPs per episode. These gains of 11.1 and 12.6 percentage points over H-JEPA use the same frozen dynamics. The small pool reaches 68.7% at 0.36 TFLOPs, exceeding H-JEPA's 67.6% at 131.0 TFLOPs with about $360\times$ less planning compute.

Table 3. **Visual AntMaze: higher success with lower planning compute.** Three frozen world models, 450 Standard and 150 Unseen episodes per row. Success is in percent, mean ± SE across models; Steps is the median among successful episodes. Candidate counts $N_1/N_2/N_3$ are $8/8/16$ for the small variant, $100/100/200$ for CriticJEPA-0, and $25/25/100$ for CriticJEPA-5. TFLOPs is mean model compute per Standard episode.

Method	Standard	Unseen	TFLOPs	Steps
H-JEPA	67.6±2.5	74.7±2.4	131.0	226
Single proposal	44.7±4.1	45.3±3.5	0.25	226
CriticJEPA-0 (small)	68.7±1.7	78.7±3.5	**0.36**	267
CriticJEPA-0	78.7±4.8	87.3±2.7	1.62	234
CriticJEPA-5	**89.1**±2.1	**92.0**±1.2	5.11	**199**

(a) Refinement steps



(b) Push-T execution budget

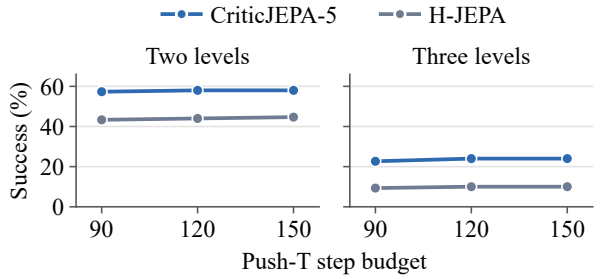


Figure 4. **Success peaks with short refinement; gains persist across execution budgets.** **(a)** The AntMaze sweep holds the model and $25/25/100$ candidate counts fixed, averaging three restarts on 50 tasks. **(b)** Push-T success across three independently trained models, with 150 episodes per curve. The same runs continue beyond the official 90-step budget at the minimum horizon, retaining the official horizon schedule through step 90.

Short refinement provides a stronger operating point with fewer candidates. CriticJEPA-5 reaches 89.1% success at 5.11 TFLOPs, a gain of 21.5 percentage points over H-JEPA with 26× less planning compute. Success improves for every independently trained world model. The same proposal heads and frozen predictors reach 92.0% on Unseen tasks, compared with H-JEPA's 74.7%, carrying the gain to new start–goal pairs. Successful episodes also finish sooner, with a median of 199 steps versus 226. This configuration combines a smaller pool with five anchored updates; Fig. 4a follows refinement length at a fixed pool size.

Table 4. **Shared settings improve success and cut compute across domains and depths.** Success in percent, reported as mean ± standard error over three independently trained models per row. *Proposal* executes a single proposal per level without verification. *Compute reduction* divides H-JEPA planner compute by CriticJEPA-5 compute.

Env.	Levels	H-JEPA	Proposal	CriticJEPA $K=0$	CriticJEPA $K=5$	Compute reduction
Cube	L=2	49.3±8.4	**100**±0.0	**100**±0.0	92.0±1.2	44×
Cube	L=3	48.7±4.1	**100**±0.0	**100**±0.0	**100**±0.0	50×
Push-T	L=2	43.3±1.8	34.7±4.8	52.0±3.1	**57.3**±1.3	79×
Push-T	L=3	9.3±2.9	18.0±2.0	18.7±1.8	**22.7**±1.3	13×
FourRoom	L=3	96.0±1.2	**100**±0.0	**100**±0.0	**100**±0.0	14×
AntMaze	L=3	67.6±2.5	44.7±4.1	78.7±4.8	**89.1**±2.1	26×

The same trained proposer supports two ways of spending planning compute: generating more alternatives and adapting those already proposed. Two-step refinement reaches 88.2% at 2.51 TFLOPs, while five steps with a larger $50/50/200$ pool reach 90.9% at 9.97 TFLOPs. Across the AntMaze budget sweep, planning compute ranges from 0.36 to 18.63 TFLOPs per episode without retraining.

5.3. Across Domains and Hierarchy Depths

The same candidate budgets and refinement settings carry the AntMaze gains to the other domains and hierarchy depths. Across all six settings in Tab. 4, CriticJEPA-5 improves on H-JEPA while cutting planning compute by 13× to 79×. The tasks reveal distinct roles for generation, selection, and local refinement.

On Cube and FourRoom, the generator supplies successful behavior directly. A single deterministic proposal solves **100%** of episodes on every world model at both Cube depths and on three-level FourRoom. CriticJEPA-0 preserves full success without gradient refinement; H-JEPA reaches 49.3% and 48.7% on two- and three-level Cube. With smaller candidate pools, CriticJEPA-5 reaches 92.0% on two-level Cube, a 42.7-point gain over H-JEPA at 44× lower compute, and 100% on three-level Cube and FourRoom.

Push-T, like AntMaze, benefits from having alternatives to compare. On two-level Push-T, generating and verifying a candidate pool raises success from 34.7% for a single proposal to 52.0%. The five-step configuration reaches 57.3% at 79× lower compute, compared with H-JEPA's 43.3%. On three-level Push-T, where short episodes restrict valid pretraining clips [30], CriticJEPA-5 more than doubles H-JEPA success, from 9.3% to 22.7%. Extending the same runs from 90 to 150 environment steps preserves the advantage in Fig. 4b. With 67% more execution steps, H-JEPA success changes by at most 1.3 points, while CriticJEPA-5 maintains

Table 5. **Hierarchical proposals combine control gains and compute savings.** All comparisons share the diagnostic AntMaze model from Sec. 3 and three restarts on 50 tasks; Tab. 3 averages over three models. Success is percent; TFLOPs is mean per episode. In (a), $N_1/N_2/N_3$ is $1/1/1$ for single proposals, $100/100/200$ for CriticJEPA-0, and $25/25/100$ for CriticJEPA-5. In (b), Upper denotes L2/L3; H-JEPA denotes gradient search. The $K = 0$ columns form a matched 2×2 comparison. At $K = 10$, L1-only and all-level proposals use 100 and 50 L1 candidates.

(a) Planning components

Configuration	Verify	Polish K	Performance Success	Performance TFLOPs
H-JEPA	–	–	72.0	131.6
Proposal, stochastic	–	–	33.3	0.25
Proposal, deterministic	–	–	52.0	0.25
CriticJEPA-0	✓	0	86.7	1.62
CriticJEPA-5	✓	5	**93.3**	5.11

(b) Proposal placement

Upper	L1	$K = 0$ Success	$K = 0$ TFLOPs	$K = 10$ Success	$K = 10$ TFLOPs
H-JEPA	H-JEPA	72.0	131.6	72.0	131.6
Proposals	H-JEPA	63.3	15.86	76.0	31.59
H-JEPA	Proposals	84.7	112.96	88.7	115.71
Proposals	Proposals	**86.7**	**1.62**	**90.7**	**18.63**

higher success across all budgets.

5.4. Understanding the Planning Gains

From a learned action to a choice of plans. On the diagnostic AntMaze model, single stochastic and deterministic proposals solve 33.3% and 52.0% of tasks, respectively, as shown in Tab. 5a. Generating multiple alternatives and selecting by frozen rollout scores raises success to 86.7%. The learned distribution supplies the candidate behaviors, and comparison adapts the choice to the current state and goal. With a smaller pool and five anchored updates, CriticJEPA-5 reaches 93.3% at 5.11 TFLOPs per episode.

The hierarchy links control quality to search cost. Applying the design at both timescales brings together higher success and lower compute. In the $K = 0$ comparison of Tab. 5b, replacing L1 search raises success from 72.0% to 84.7%. The upper-level optimizers still consume most of the computation, leaving the total at 112.96 TFLOPs per episode. With L1 proposals in place, replacing upper-level search further raises success to 86.7% and cuts compute to 1.62 TFLOPs, a 70× reduction. The lower level improves the actions executed; the upper levels make the resulting controller inexpensive to run.

Upper-level proposals are most effective with proposal-based control. With H-JEPA at L1, they reach 63.3% at $K = 0$. Their $K = 10$ configuration reaches 76.0% at 31.59 TFLOPs, surpassing H-JEPA's 72.0% with about one quarter of the compute. At $K = 10$, the full hierarchy reaches 90.7% at 18.63 TFLOPs, compared with 88.7% at 115.71 TFLOPs for L1-only proposals. Pairing proposals at every level delivers the highest success and lowest compute at both refinement budgets.

Short corrections improve complete-task success. Short refinement captures the largest gains in complete-task success. With the world model and candidate counts fixed, two anchored steps raise success from 84.0% to **94.7%**, as shown in Fig. 4a. Ten and twenty steps retain 91.3% and 90.7% success; the peak is reached with two updates. The three-model AntMaze sweep shows the same preference for short refinement: with $50/50/200$ candidates, five steps achieve 90.9% and ten achieve 89.8%. Gradients are most effective here as local corrections to generated plans, while the frozen critic makes the final choice. This complete-task profile connects the planner's compute allocation to the early execution gains in Sec. 3.1.

The same frozen hierarchy also succeeds more consistently across restarts. CriticJEPA-5 averages 93.3% success per run, exceeding the 92.0% coverage obtained by combining three H-JEPA restarts in Sec. 3.3. Three runs of CriticJEPA-10 solve **all 50 tasks** in their union, and the fraction of tasks whose outcomes change across those runs drops from 48% to 22%.

6. Discussion and Conclusion

Frozen predictions support better plan selection after continued gradient search stops improving execution. Exact-state AntMaze execution reveals this distinction in the same frozen predictor: its rankings distinguish goal-conditioned proposals and are uninformative for H-JEPA search outputs. The candidate source determines both which plans the model compares and what its selected action achieves.

CriticJEPA turns this insight into a complete hierarchical planner. Flow heads learn goal-directed alternatives from reward-free trajectories, frozen rollouts choose among them, and short anchored updates capture the early gains from gradients before selection. This division operates on both macro-actions and primitive controls. Forward selection preserves full success on Cube and FourRoom, where proposals already supply task-solving behavior. On AntMaze and Push-T, comparing alternatives and applying short corrections yield further gains from the same frozen predictors. Shared settings carry these gains across independently trained world models, hierarchy depths, and unseen tasks. Every world-model weight stays fixed.

Planning compute now serves two complementary roles: generating alternatives and locally refining them before frozen rollout selection. Candidate generation shapes what can be achieved, while the model's comparisons determine which action is executed. This division recovers more successful behavior from unchanged world models.

References

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

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

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

[4] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

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

[6] Saksham Bansal, Om Naphade, Chayan Aggarwal, and Vrishin M. LePlanner: An iterative amortized controller for world models. *arXiv preprint arXiv:2609.13845*, 2026.

[7] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[8] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

[9] Niccolò Caselli, Francesco Massafra, Samuele Punzo, Salvatore Lo Sardo, Ippokratis Pantelidis, and Sathya Kamesh Bhethanabhotla. Mind the gap: Promises and pitfalls of hierarchical planning in LeWorldModel. *arXiv preprint arXiv:2607.12547*, 2026.

[10] Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-conditioned action generation for latent world model planning. *arXiv preprint arXiv:2607.17973*, 2026.

[11] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

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

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

[14] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2019.

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

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

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

[18] Yann LeCun. A path towards autonomous machine intelligence. OpenReview preprint, 2022.

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

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

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

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

[24] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision*, 2023.

[25] Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[27] Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026.

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

[29] Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: PRior-guided imagination sampling in world models. *arXiv preprint arXiv:2606.07974*, 2026.

[30] Wancong Zhang, Basile Terver, Michael Rabbat, Yann LeCun, and Randall Balestriero. H-JEPA: End-to-end learning of hierarchical world models for visual planning. *arXiv preprint arXiv:2610.06805*, 2026.

[31] Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

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