LEARNING TO IGNORE WHAT MATTERS: JEPAS IN STOCHASTIC WORLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-embedding predictive architectures (JEPAs) learn world models by predicting future representations rather than reconstructing observations. We show that this objective does not protect the information needed for control. We identify two failure mechanisms for jointly trained JEPAs in stochastic environments: the representation can discard the state that decisions depend on, and the predictor can lose decision-relevant features even when the representation keeps them. We test both by training agents entirely inside a JEPA world model and inside a generative world model with the same data and compute, then evaluating them in the real environment. The JEPA supports far weaker control: agents trained inside it survive up to 50 times less often and fail to learn multi-step behavior in Craftax. These failures are not caused by collapse, and rollouts that look correct can still teach agents to exploit errors in predicted rewards and terminations. Low latent prediction error and non-collapsed representations coexist with poor closed-loop performance, so neither is sufficient for control.

1 INTRODUCTION

Most of what a camera sees is irrelevant to the agent carrying it: the pattern of a carpet, the flicker of sensor noise, the leaves moving on a tree. Joint-embedding predictive architectures (JEPAs) promise a world model that learns to look past such things (LeCun, 2022; Assran et al., 2023; Bardes et al., 2024). Instead of reconstructing the next observation, an encoder and a predictor are trained together so that the predictor forecasts the encoder's own next output. The model never has to paint a pixel, and the encoder is free to drop whatever it cannot predict. LeWorldModel (Maes et al., 2026) turned this idea into a minimal recipe for control: a squared prediction loss, a regularizer that keeps the latents Gaussian so that they cannot collapse, and nothing else.

In a stochastic world, that freedom has a price. A single objective decides both *what* to encode and *how* to predict it, and it rewards whatever is predictable. Yet what an agent controls is often the least predictable part of the scene, pushed by unseen forces and threatened by obstacles revealed at the last moment. An objective that rewards predictability can thus learn to ignore what matters. We identify two ways this happens (Section 2). The *representation* can discard the controlled state, which nothing asks the encoder to keep and whose random perturbations cost prediction loss. The *rollout* can lose the state even when it is kept: a squared loss predicts averages, averages do not compose over time, and features the loss barely penalizes are the first to go.

To test both, we set the LeWorldModel JEPA against a generative world model built from the very ingredients a JEPA avoids: a tokenizer trained to reconstruct frames, and a flow-matching predictor that *samples* the next latent instead of averaging over the possibilities (Lipman et al., 2023; Hafner et al., 2025). Both learn from the same data under the same compute budget, though as full recipes they differ in more than the objective (Section 7). Our worlds are small visual control tasks in which each kind of randomness can be switched on by itself (background noise redrawn every frame or held fixed, a hidden wind that pushes the agent, an approaching obstacle whose opening is revealed late), and the open-world game Craftax (Matthews et al., 2024). Each model faces two tests: what its latent retains about the state, both when encoding real frames and when imagining the future; and what an agent trained entirely inside its imagination can do in the real world. A control separates the two: the same agent, trained on real experience seen through the model's decoder, shows whether the information needed for control was there at all.

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.

Our findings:

1. **Randomness in the controlled state can make the JEPA abandon it.** With background noise redrawn every frame and a hidden wind, no read-out locates the agent from the JEPA's latent better than chance, in any trained model, even at three times the compute; with the noise held fixed, the agent is located to within a pixel. Without wind, per-frame noise is instead the easier background. The failure is not collapse, nor what the objective prefers: a weaker regularizer can keep the agent and score four times better under the failing models' objective.
2. **A representation that holds the state does not guarantee a rollout that does.** In the simplest obstacle world, the JEPA encodes the approaching wall almost perfectly and then erases it from its own predictions, often at the very first step. Averaging over the two possible openings is not to blame: an exact average of both futures keeps the wall; the deletion costs the prediction loss almost nothing. Nor is the JEPA's predictor alone at fault: the flow predictor, trained on the same frozen encoder, loses the wall too.
3. **Only the generative model's imagination teaches.** Agents trained inside the flow model survive the obstacle in 97% of episodes; inside the JEPA, in 2%, and in none once background noise is added, although the same agent trained on real experience through the JEPA's own decoder survives in 49% to 98%. Worse, the JEPA's imagination is exploitable: agents learn to game its over-optimistic reward and survival predictions. In Craftax, flow-trained agents craft a wooden pickaxe in 95% of episodes in two of three runs; JEPA-trained agents never do, and unlock fewer achievements than random play.

Non-collapse and low prediction error are therefore no evidence that a world model supports control, and repairing the rollout helps only where the representation has kept the state.

2 Two ways to ignore what matters

The objective. An encoder f_θ maps an observation x_t to a latent $z_t \in \mathbb{R}^D$, and a predictor g_θ maps recent latents and actions to a forecast of z_{t+1}. LeWorldModel (Maes et al., 2026) trains both jointly by minimizing

$$\mathcal{L} = \frac{1}{D}\,\widehat{\mathbb{E}}\,\|g_\theta(z_{\le t}, a_{\le t}) - z_{t+1}\|^2 + \lambda\,\mathrm{SIGReg}(\{z_t\}), \tag{1}$$

where SIGReg is a statistical test, computed on random one-dimensional projections of a batch, of whether the latents are distributed as a standard Gaussian. The regularizer rules out the trivial solution that maps every frame to the same point; there is no stop-gradient, teacher network, decoder or reward. Two choices hide in this single line: the encoder chooses the target z_{t+1} that the predictor must hit, and the loss is a squared error. They fail in different places.

2.1 Mechanism 1: the encoder decides what is worth predicting

For a fixed encoder and an unrestricted predictor, the lowest achievable squared error is the conditional variance of the target, $\inf_g \mathbb{E}\|g(C,A) - Z'\|^2 = \mathbb{E}\,\mathrm{trCov}(Z' \mid C, A)$: whatever the past cannot predict is charged in full. Joint training lets the encoder lower that charge by choosing what Z' contains. Suppose each frame is rendered as $x_t = R(p_t, b)$ from a controlled state p_t and a nuisance b that is fixed for the episode, such as a background pattern. If b varies enough across episodes to be mapped to a standard Gaussian $\varphi(b)$, an encoder that outputs only $\varphi(b)$ is as far from collapse as possible, satisfies the regularizer exactly, and lets a predictor that copies the last latent reach zero loss, while carrying no information whatsoever about p_t (Proposition 1 in Appendix A, extending Sobal et al., 2022; proofs in Appendix B). An agent reading such a latent is no better off than one that ignores the screen.

The construction needs a constant nuisance; stochastic worlds turn its lesson into a cost. If the controlled state receives random shocks, an encoder that keeps it at a fixed scale must pay for every shock in prediction loss (Corollary 1), up to a cap set by the Gaussian marginal (Remark 1). Non-collapse says nothing about whether it pays; Section 4 measures that.

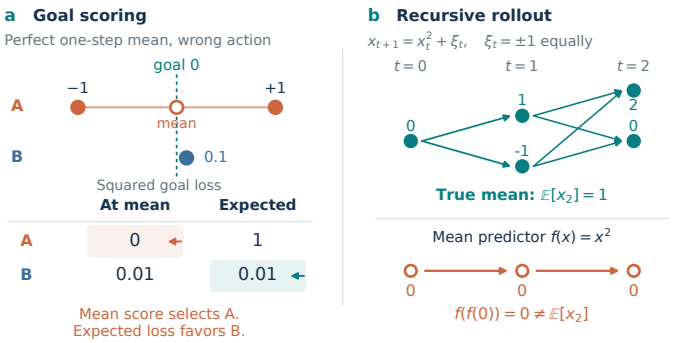


Figure 1: **A correct mean can still give a wrong decision and a wrong rollout.** Left: the mean of action A lands exactly on the goal, although every outcome of A is farther from it than B's certain outcome, so a planner that scores means prefers the worse action. Right: for $X_{t+1} = X_t^2 + \xi_t$ with fair coin flips $\xi_t = \pm 1$ and $X_0 = 0$, a rollout that feeds back its own exact conditional means stays at 0, while the true mean after two steps is 1. Both panels are exact calculations, not trained models.

2.2 Mechanism 2: The squared loss keeps only the mean

Even when the state is kept, a squared loss learns the conditional mean $m(c, a) = \mathbb{E}[Z' \mid c, a]$, whereas a planner needs expected costs. For a latent goal z_g,

$$\mathbb{E}\big[\|Z' - z_g\|^2 \mid c, a\big] = \|m(c, a) - z_g\|^2 + \mathrm{trCov}(Z' \mid c, a), \tag{2}$$

and whenever the second term depends on the action, it can change which action is best. Figure 1 (left) gives the smallest example. The goal is 0; action A lands at -1 or $+1$ with equal probability, and action B lands at 0.1 for certain. A planner that scores means sees A exactly on target and chooses it, although A always misses by a full unit and its expected cost is a hundred times that of B. No better estimate of the mean can fix this: the discarded variance is what the planner needed.

Means do not compose. The damage also accumulates along a rollout. Let $X_{t+1} = X_t^2 + \xi_t$ with independent fair coin flips $\xi_t \in \{-1, +1\}$ and $X_0 = 0$. The exact one-step conditional mean is $m(x) = x^2$, so a model fed its own mean predictions answers $0, 0, 0, \ldots$ forever. The real process moves to $X_1 = \pm 1$ and then to $X_2 \in \{0, 2\}$, whose mean is 1 (Figure 1, right). In general $\mathbb{E}[X_{t+1}] = (\mathbb{E}X_t)^2 + \mathrm{Var}(X_t)$, and substituting means discards the variance at every step. A model rolled forward on its own predictions makes exactly this substitution, plus its own approximation error; Appendix A separates the two, and Section 4 measures which a trained network commits. Approximation error can land on what matters: Proposition 2 shows that a decision-relevant feature can be worth almost nothing to the loss.

What repairs mechanism 2, and what does not. A model can be trained to predict expected costs or values directly, keeping the expectations a decision needs without ever sampling (Wan et al., 2019; Grimm et al., 2021). More generally, it can predict the whole distribution, as a flow-matching predictor does by sampling Z'. Neither touches mechanism 1: if the encoder has already dropped the state, a perfect distribution over the remaining features is a perfect distribution over nothing useful. This order, representation before rollout, is what our experiments test.

3 Worlds, models and two tests

Worlds with one knob per kind of randomness. Our control worlds are deliberately simple, so that each source of randomness can be switched on by itself. The agent steers a red 6×6-pixel block, which we call the cube, around a 63×63 color image; each of four actions moves it three pixels, and touching the border ends the episode with a penalty of -1, the only reward there is. Episodes last at most 128 steps. Three knobs set the difficulty. The *background* is black, or gray noise that is either redrawn every frame (*per-frame noise*, like sensor static: irrelevant and unpredictable) or drawn once per episode and then held fixed (*persistent noise*, like a patterned floor: irrelevant, and perfectly

predictable after the first frame). *Wind* adds a hidden drift to every move, a slow random walk the agent never observes, so that even a perfect memory of the past leaves fresh uncertainty about where the cube will be next; without it the world is *still*. In the *gate* worlds, walls sweep across the image toward the cube one after another, each with a single opening at the top or the bottom; which side is random and stays hidden until the wall is close. To survive, the agent must hold back until the opening appears, then commit to it; any strategy that ignores the screen never survives. Beyond these nine worlds we use Craftax-Classic (Matthews et al., 2024), a two-dimensional open-world game in the spirit of Minecraft, where an agent gathers resources and crafts tools and progress requires multi-step plans. All models learn from the same recordings of uniformly random play: 500,000 transitions per world (100,000 for the two black cube worlds); exact dynamics are in Appendices E and F.[1]

Three models, one budget. The *grid JEPA* uses LeWorldModel's objective (Equation 1) at its published weight $\lambda = 0.09$, with a spatial latent: each frame becomes a 4×4 grid of 32-dimensional vectors, 512 numbers in all, and a predictor attending over space and time forecasts the next grid from the last eight latents and actions. The *class-token JEPA* is our port of LeWorldModel's published design, which summarizes a frame in a single 192-dimensional vector. The *flow* model is the generative reference: a tokenizer trained to reconstruct frames, with the grid JEPA's encoder trunk and latent shape, followed by a causal flow-matching predictor that samples the next latent (Lipman et al., 2023; Hafner et al., 2025) and is trained jointly with reward and termination heads. Every model is trained from scratch on one A100 for about ten minutes (under 600 seconds per JEPA, 732 for the flow model because of its separate tokenizer stage); as the JEPA's representation failure persists at three times its budget (Appendix J), that difference is not what separates them. Whatever a JEPA needs downstream (decoders, read-outs, heads) is fitted afterwards on the frozen model, outside its budget.

Test 1: what the latent keeps. A latent can hold the state and still lose it in prediction, so we read both. We ask whether the cube's position can be recovered from the encoding of a real frame (ENC) and from the model's own prediction eight steps ahead, started from real context (PRED). A small read-out is trained on encodings of training frames and scored on held-out ones by its error in pixels. We fit one read-out on the latent and one on frames rendered by a decoder fitted to the frozen model, and report the better, so that a weak decoder can never fail a model; an error above 15 pixels is chance. In the gate worlds we also measure how much of the approaching wall survives in the decoded frames (its intensity, 1 when fully present) and whether the side of the opening can be told apart (AUROC, 0.5 at chance). In Craftax we compare decoded frames with real ones by Fréchet distance, against the baseline of repeating the last frame. Verdicts, fixed in advance: **green** when real encodings locate the cube within one cube width and predictions within two pixels of an ideal forecaster (wall: at least 80% intensity), **red** at chance (wall: below 20%), **amber** in between.

Test 2: what an agent learns inside the model. A world model is ultimately a place to practice. We therefore train one standard pixel-based agent, a convolutional network with a recurrent memory (1.7M parameters) trained by PPO (Schulman et al., 2017), entirely inside each model: from real recorded moments, the model supplies the frames through its decoder, and its own heads supply rewards and terminations; nothing from the simulator enters training. Each agent is then tested in the real world on the same 256 fixed episodes, and we report the fraction it survives. Two blind strategies, moving at random and alternating left and right, go through the same test and show what can be achieved without looking. Before any imagination result is read, the same agent is trained on *real* experience seen through each model's encoder and decoder, which we call the model's interface: if it succeeds, the latent carries what the task needs, and any later failure belongs to the imagination rather than to the interface. Verdicts: **green** for at least 80% mean survival and 50% in every run, **red** when the agent beats the better blind strategy by at most five points, **amber** in between (Appendix H).

Table 1: **Test 1: what each model's latent keeps.** Error in pixels when a read-out locates the cube from the encoding of a real held-out frame (ENC) and from the model's own prediction eight steps ahead (PRED); lower is better, above 15 is chance. Means over three independently trained models, no selection of seeds or checkpoints (every run in Table 3). With wind, an ideal forecaster errs by 7.90 at eight steps if it predicts the mean and 11.76 if it samples, as the flow model does. In the gate worlds the color also reflects the wall (details in Table 4), so a model that tracks the cube but loses the wall is **red**. Craftax: Fréchet distance $\times 10^3$, against 13.3 for repeating the last frame.

	class-token JEPA		grid JEPA		flow	
world	ENC	PRED	ENC	PRED	ENC	PRED
black, still	0.09	18.4	0.10	3.35	0.12	0.12
black, wind	0.00	8.79	0.13	11.5	0.13	11.1
per-frame noise, still	20.8	19.9	0.12	4.83	0.26	0.33
per-frame noise, wind	18.0	19.6	17.6	20.4	0.25	12.5
persistent noise, still	18.7	20.0	0.76	1.83	0.59	1.79
persistent noise, wind	16.0	19.5	0.67	11.0	0.25	13.4
black gate	6.73	17.2	0.10	3.24	0.17	0.38
gate + per-frame noise	20.2	20.3	4.74	14.5	0.26	0.43
gate + persistent noise	18.5	21.0	1.09	3.37	0.27	1.03
Craftax-Classic	18.6	24.0	8.09	11.4	0.07	1.27

4 What the models keep

Our causal claims rest on rows that differ in one property of the world, with the model held fixed.

Wind reverses the effect of the background. In still worlds, the grid JEPA keeps the cube under either background, and per-frame noise is the *easier* one: the cube is located to 0.12 pixels, against 0.76 with persistent noise. This is the JEPA promise at work: unpredictable static is exactly what the encoder should learn to ignore. Add hidden wind and the roles reverse. With persistent noise the cube is still located to within a pixel (0.67); with per-frame noise the error jumps to 17.6 pixels, chance, in all three trained models. Architecture, latent size, data, compute and read-out are identical across these cells. The failure is present in the encoding of real frames, before anything is predicted, and it is visible to the eye (Figure 2): decoded from the JEPA's latent, a real frame becomes a gray field with at most a trace of the cube, while the flow model, trained on the same data, keeps it sharp. Every read-out we tried is at chance on this latent, which shows that the information is unusable, though not that it is erased (Appendix E). Two natural explanations fail: two and three times the compute change nothing, and cutting the gradient through the prediction target, so that the encoder cannot shape its own target, does not bring the cube back (Appendix E). The regularizer's weight does matter: the seed-0 fits at $\lambda = 0.03$ keep the cube in both wind worlds, though two more seeds with per-frame noise do not; at 0.27 the persistent one degrades, and at 0.9 both are at chance (Appendix J).

This is mechanism 1, though not in its textbook form. In a still world, the cube's next position follows from the past and the action, so keeping it costs the predictor nothing. Wind attaches a cost: an encoder that keeps every state distance at scale κ pays at least $\kappa^2 \mathbb{E}\mathrm{tr}\Sigma_t / D$ in prediction loss (Corollary 1), and the Gaussian marginal caps that cost near 0.002 per latent coordinate (Remark 1). The persistent-noise models pay it and keep the cube. The failing models do something stranger. Their loss, 0.73 per coordinate, is three hundred times that cap: their latent is mostly unpredictable and not even Gaussian (Appendix B), far from the zero-loss shortcut of Proposition 1. Nor does the objective prefer it: a model trained at $\lambda = 0.03$ keeps the cube and scores four times better under the failing models' objective. The blind latent is where optimization lands, in every seed and at every budget we tried. The theory explains why the objective does not protect the cube, not why persistent noise does.

[1] In two further cube worlds with a persistent colored background, in which only the flow model was trained, its latent loses a still cube (18.4 pixels, chance) and locates a windblown one only to 4.1 pixels. The flow model is thus not immune to nuisances either.

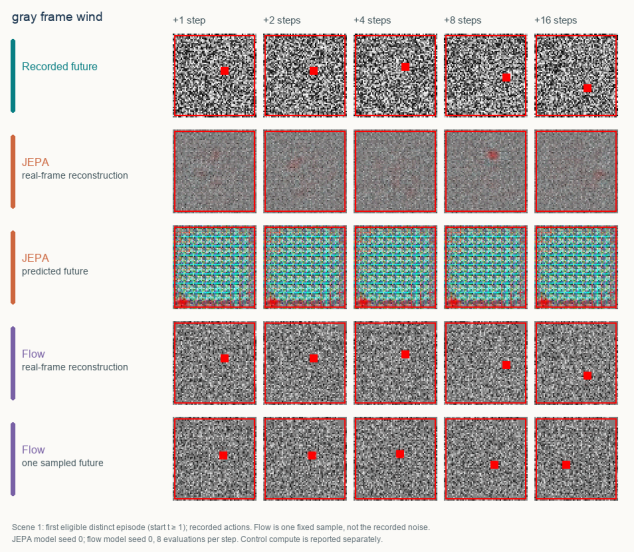


Figure 2: **Per-frame noise plus wind: the JEPA no longer encodes the cube.** From top: the recorded future under the recorded actions; the JEPA's decoding of its own encodings of those real frames; the JEPA's imagined future; the flow model's decoding of the real frames; and one future sampled by the flow model. First eligible held-out episode, chosen blind to model outputs; model seed 0. Decoders are read-outs fitted after training: rows two and four show what each latent retains.

The published single-vector design fails earlier, and differently. The class-token JEPA loses the cube in all six noisy worlds, but its most instructive failure is in the easiest one. On a black, still background it locates the cube to a tenth of a pixel from a real encoding, yet eight steps into its own rollout the read-out is at chance, although the rollout's error is a mere 0.04% of the latent variance. The information is there, but at a tiny scale: adding random noise of that size to *true* latents destroys the read-out just as thoroughly, while the same relative perturbation of the grid latent moves it by about a tenth of a pixel (Appendix E). A rollout can thus be excellent by the standard of the loss and useless to an agent. Proposition 2 shows that the objective cannot rule this out: a latent can carry the state exactly, at an arbitrarily small scale, while staying perfectly Gaussian.

In the black gate, the wall survives encoding and dies in prediction. The black gate is built to take mechanism 1 off the table. There is no background to prefer, the wall is large and slow, and the only uncertainty is which opening will appear, so even an average of the two possible futures shows the wall's solid part. The grid JEPA encodes the wall almost perfectly, 99% of its intensity with the side of the opening perfectly separable, and then its predictor deletes it. At the first predicted step two of three models show no wall at all and the third about half of it; by the eighth step none shows any wall, and the side of the opening is at chance (Figure 4 in Appendix D), while the cube's position survives. The flow model keeps both; the class-token JEPA never encodes the wall reliably.

An exact average keeps the wall. Proposition 3 splits a wrong prediction into three parts: an error of the task head, the gap between reading the conditional mean and averaging the reading over outcomes (mechanism 2), and the predictor's failure to reach that mean at all; the gate separates the last two. For every held-out moment at which the wall is still solid, we reconstruct the exact simulator state and render two futures under the same actions, the one that happened and the one with the opening on the other side, encode both with the frozen encoder, and decode their average. Because the two futures look identical until the opening is revealed, this average is the best any deterministic predictor could do. It shows the wall at 96% to 100% of its intensity in every model, and the model's own termination head responds to it exactly as it does to the real next frame. The trained prediction from the same context has lost the wall in two of three models at the first step, and in all three by the eighth. The two futures' encodings differ by at most 0.025% of the latent variance: to that precision the future was not uncertain at all, averaging contributes nothing, and the missing wall is approximation error in its entirety (Figure 5). The same measurement explains

Table 2: **Test 2: survival of one agent trained inside each model, and on real experience.** Percentage of the 256 fixed test episodes survived at the final checkpoint, mean ± standard deviation over three agent seeds, no selection of seeds or checkpoints (per-seed values, episode lengths and returns in Table 5). "Blind" is the better of two strategies that ignore the screen; in the still cube world, alternating left and right survives indefinitely, so no agent can beat it. The cube worlds and the black gate use one trained model; the noisy gate worlds pair agent seed k with model k.

		trained inside the model		trained on real frames via decoder	
world	blind (best)	grid JEPA	flow	grid JEPA	flow
per-frame noise, still	100	85±18	97±2	n/a	n/a
per-frame noise, wind	0	0±0	58±41	n/a	n/a
persistent noise, wind	0	2±1	41±42	n/a	n/a
black gate	0	2±1	97±2	97±2	81±25
gate + per-frame noise	0	0±0	88±13	66±26	83±21
gate + persistent noise	0	0±0	5±6	94±3	73±37

why the objective tolerates it. The trained prediction sits only 0.03% to 0.05% of the latent variance away from this oracle: keeping or deleting the entire wall moves the prediction loss by less than a two-thousandth of the variance it is measured against. The encoder kept an exact feature at an amplitude the loss barely penalizes, and the predictor let it go.

With a background, the rollout keeps the wall. Immediate deletion is specific to the black gate. Add a background to the same world and the grid JEPA's rollout *keeps* the wall, at 91% of its intensity with per-frame noise and 90% with persistent noise (Table 4). Test 1 alone would rank the noisy gate worlds as *easier* for the JEPA; Section 5 shows otherwise, which is why we run both tests.

A different predictor on the same encoder splits the blame. To turn the recipe gap into a controlled comparison, we train the flow model's sampling predictor, recipe and heads unchanged, on the frozen grid-JEPA encoder's latents and render its predictions with the JEPA's decoder (Appendix K). In the black gate it fails as the JEPA's own predictor does, seed for seed: the wall is gone by the eighth step and the opening is at chance, although the same predictor keeps both on its own tokenizer. With per-frame noise it keeps the wall at 99% through eight steps and reads the opening with an AUROC of at least 0.99 on two of three encoders, where the JEPA's predictor does so on one; on the third encoder both predictors lose the opening and let the wall fade to about half. The tokenizer shares the JEPA encoder's trunk, so the gap lies in the encoder's objective: every black-gate encoder and one per-frame-noise encoder store the wall at a scale neither predictor preserves, as Proposition 2 allows.

Craftax repeats the ranking. The flow model's eight-step predictions are about nine times closer to real frames than the grid JEPA's by Fréchet distance, and the grid JEPA's barely improve on repeating the last frame (11.4 against 13.3); the class-token JEPA does worse than repeating it.

5 What an agent learns inside the models

Both interfaces carry what the task needs. Trained on real experience seen through the JEPA's own decoder, the agent learns all three gate worlds: it survives 95% to 98% of episodes in the black gate, 91% to 96% with persistent noise and 49% to 95% with per-frame noise, where blind strategies never survive. Through the flow decoder it learns them too, from 31% to 97%; in two of three worlds the JEPA's interface scores higher. Whatever goes wrong next is a failure of imagination.

Inside the JEPA, agents never learn the gate. With the same agent, budget and test episodes, agents trained inside the JEPA survive 2% of the black gate's episodes, against 97% inside the flow model. They are not random: they stay alive for 42 steps on average, where blind strategies last 24, having learned the one danger their imagination still showed them (the border) in a world whose walls their model had erased. With per-frame noise the JEPA agents never survive, in any seed, while the flow agents survive 88%. Here the two tests part ways: the JEPA's rollout keeps the wall

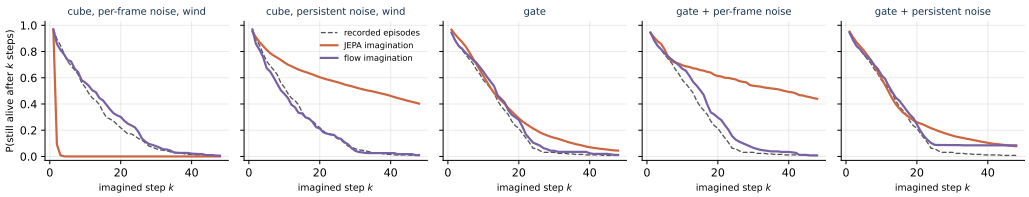


Figure 3: **What each model's imagination tells an agent about dying.** Each frozen model is rolled forward from 256 real starting points under recorded random actions, and its own termination head is read along the rollout. Solid curves show the implied probability that the imagined episode is still alive after k steps; the dashed curve shows how many of the same recorded episodes actually were. One model of each kind, those of Table 2. In the cube world with per-frame noise and wind, the JEPA's head predicts termination with probability 0.91 at every step, so its imagined episodes end at once; elsewhere it is optimistic, most of all in the gate world with per-frame noise.

(Section 4) and agents trained through its interface succeed, yet agents trained in its imagination fail. Planning directly with the JEPA's predictor, as LeWorldModel was designed to do, fails too: a search over action sequences is no better than random in either noisy gate world (Appendix I).

With persistent noise, sampling does not help either. Here *both* imaginations fail. The JEPA agents never survive and the flow agents survive 5%, although agents trained through either interface succeed, and although both latents keep the wall and both rollouts render it. We therefore cannot claim that a distributional predictor repairs imagination in general. What remains is an order: a failure of the representation is fatal, a failure of the rollout is sometimes repairable, and for both models imagination fails before the interface does.

The failing imaginations are exploitable. With persistent noise the JEPA agents die even sooner than blind play (20 steps against 24), and with per-frame noise they last only slightly longer (26). Figure 3 shows what they were taught. Rolled forward under recorded random actions, the JEPA's termination head tracks real survival for sixteen steps with persistent noise; with per-frame noise it turns optimistic from the fourth step, and after 48 steps it expects 44% of episodes to be alive where fewer than 1% are. Trained agents then find exactly the action sequences the model rates as safe. They finish with imagined returns as high as $+0.48$ in a world whose only reward is a penalty for dying, and with persistent noise the agents of *both* models end with higher imagined survival than any agent achieves in the world (Appendix M). This is the head-error term of Proposition 3: a task head read on latents that the agent's own actions pushed outside the training distribution.

Without walls, the representation decides first. With per-frame noise and wind, where the encoder lost the cube, every JEPA agent dies in every episode, after 10 to 16 steps, sooner than a blind random walk (17), while the flow agents survive 58% on average. With persistent noise and wind, where the encoder kept the cube, the JEPA agents stay alive longer than blind play (27 steps against 17 and 18) yet survive only 2% of episodes, against 41% for the flow agents. Where the representation fails, imagination is dead on arrival; where it holds, imagination is merely wrong. These worlds are sensitive to the agent's budget: with three times the updates the flow agents reach 90% and 65%, while the JEPA agents stay at 0% with per-frame noise and reach 51% with persistent noise in one seed of three (Appendix L).

In Craftax, agents exploit the JEPA's reward. Here success requires a multi-step plan. One agent per seed is trained for five million imagined steps inside each model, with rewards and terminations from the model's own heads and no demonstrations, action masking or reward shaping, and is then tested in the real game on 256 episodes. The milestone we track is the wooden pickaxe, which requires collecting wood, placing a crafting table and crafting, in that order. Inside the flow model, every seed collects wood in every episode and places a table in over 90% of them, and two of three seeds craft the pickaxe in 95% of episodes, the third in 6%. Inside the JEPA, no seed ever places a table or crafts a pickaxe, and the agents unlock fewer achievements than uniformly random play (0.02 to 0.27 per episode, against 2.34), about as many as doing nothing (Table 13). They learned what their environment paid for: the JEPA's reward head, fitted after training, paid them

0.90 to 1.00 per imagined step, thirty times what the best real agent earns, while the flow model's heads paid 0.03 to 0.06, the rate of real play. Craftax lacks an interface control, so this supports the reward-exploitation account but cannot separate interface from imagination (Appendix O).

6 Related work

Joint-embedding world models. Joint-embedding prediction avoids reconstructing observable detail (Assran et al., 2023; Bardes et al., 2024). LeWorldModel learns an encoder and an action-conditioned predictor jointly, with a squared loss and a Gaussian regularizer (Maes et al., 2026; Balestriero & LeCun, 2025), and is the recipe we study; DINO-WM and V-JEPA 2 predict on frozen pretrained features (Zhou et al., 2024; Assran et al., 2025), and PLDM learns the representation jointly with additional regularizers and plans in it for navigation (Sobal et al., 2025); we judge a model instead by what an agent trained inside it can do. Sobal et al. (2022) identified the failure in which an encoder latches onto a predictable nuisance, which Proposition 1 specializes to this objective. Our worlds add fresh randomness to the controlled state, where the effect of a persistent nuisance reverses; this bounds that finding rather than contradicting it. LeWorldModel itself reports accurate probes alongside weaker planning; Temporal-Distance JEPA, RC-aux and physically grounded variants add temporal, reachability or task-state structure (Bai & Xiong, 2026; Li et al., 2026; Yan et al., 2026; Liu et al., 2026), and Terver et al. (2026) observe that conditional means blur multimodal futures.

Generative imagination and objective mismatch. Generative world models have supported control learned in imagination (Hafner et al., 2020; Alonso et al., 2024; Hafner et al., 2025), and they have their own perception, action-conditioning and rollout failures (Hansen & Wang, 2026). FlowWM learns stochastic futures in feature space without actions (Porcher et al., 2026); our flow reference is action-conditioned and trained on the JEPA's data and budget. That a model good at prediction need not be good for control is a longstanding observation (Lambert et al., 2020): value-aware model learning, expectation models and (proper) value equivalence fit exactly the quantities a class of decisions needs (Farahmand et al., 2017; Wan et al., 2019; Grimm et al., 2020; 2021), and TD-JEPA predicts policy-conditioned successor features (Bagatella et al., 2026).

7 Discussion

Two failures, in this order. The representation failure appears before any rollout, and nothing downstream can restore a state the encoder has dropped. The rollout failure appears only once the state is kept. The flow model repairs it in two of the three gate worlds (97% and 88% survival, against 2% and 0% for the JEPA); its predictor on the JEPA's encoder keeps the wall for two of three per-frame-noise encoders and for none of the black gate's, which store it too faintly.

A good latent is not a good imagination. In the noisy gates and Craftax, Test 1 rates the JEPA partial at worst, and what fails is invisible to it: the predictor's approximation of a feature the loss barely penalizes, or a task head read on latents the agent's own actions produced. Non-collapse and low prediction error coexisted with every failure here; only Test 2 exposed them all.

Scope and limitations. Our recipes share an encoder trunk but differ in predictor, updates and supervision, so every JEPA-versus-flow gap is a recipe gap; our causal claims rest on within-recipe contrasts and the swap. We swapped predictors in one direction only, the flow predictor on the JEPA encoder, and scored the swap with Test 1 alone. The wind reversal holds at $2\times$ and $3\times$ the budget but depends on the regularizer's weight and seed (Appendix J). Our agents need the decoder; on the raw latent they fail even on real experience, for both models (Appendix I). Each behavioral result rests on one trained model per world with three agent seeds, and the black gate's JEPA was chosen during development.

Outlook. A general world model needs a principle for deciding what to keep, and predictability alone is not one. The controlled state, random shocks included, is information to model rather than noise to predict away. A world model should be judged by what its latent retains, what its rollouts preserve and what it teaches an agent.

REPRODUCIBILITY STATEMENT

Environments, models, agents, evaluation code and the scripts that generate both main tables will be released with the code. Every model number comes from a fixed final checkpoint, and every agent number from a final checkpoint evaluated on stored test episodes; the training scripts enforce the compute budgets and refuse other values. Model, data and source hashes are recorded beside every run. The protocol and its verdict thresholds were fixed before the experiments, every later amendment is dated in the repository, and a checker maps every number in this paper to the artifact it comes from. The code was extended twice during the experiments, to add the two noisy gate worlds and the longer agent budget of Appendix L; both extensions only add to the shared code path, and a second checker reports which version produced each run.

USE OF LARGE LANGUAGE MODELS

Language-model coding agents helped write some of the experiment code; the authors specified the protocols, audited the code and results, and are responsible for every claim. For retrieval and discovery, language models helped find related work. For writing, language models helped to polish and proofread the text, which the authors revised. The authors take full responsibility for all content and every claim.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning, 2025.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.00739.

Jiaxin Bai and Jiaxuan Xiong. Temporal-distance JEPA: Plan-aware representation learning for latent world model predictive control, 2026. URL https://arxiv.org/abs/2607.25337.

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

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1486–1494, 2017.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alexander Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Christopher Grimm, André Barreto, Gregory Farquhar, David Silver, and Satinder Singh. Proper value equivalence. In *Advances in Neural Information Processing Systems*, volume 34, pp. 7773–7786, 2021. URL https://proceedings.neurips.cc/paper/2021/file/400e5e6a7ce0c754f281525fae75a873-Paper.pdf.

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

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models, 2025.

Nicklas Hansen and Xiaolong Wang. Hallucination in world models is predictable and preventable, 2026. URL https://arxiv.org/abs/2606.27326.

Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. Muesli: Combining improvements in policy optimization. In *International Conference on Machine Learning*, 2021.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Conference on Learning for Dynamics and Control (L4DC)*, 2020.

Yann LeCun. A path towards autonomous machine intelligence, 2022. Version 0.9.2, OpenReview.

Wenyuan Li, Guang Li, Keisuke Maeda, Takahiro Ogawa, and Miki Haseyama. Predictive but not plannable: RC-aux for latent world models, 2026. URL https://arxiv.org/abs/2605.07278.

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

Muyuan Liu, Yue Huang, Zheng Liang, and Xiang Gao. Toward physically grounded JEPA world models for goal-conditioned robotic planning, 2026. URL https://arxiv.org/abs/2609.03565. IROS 2026 Workshop on Physical World Models for Scaling Embodied AI.

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

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

François Porcher, Nicolas Carion, Karteek Alahari, and Shizhe Chen. Flow matching in feature space for stochastic world modeling, 2026. URL https://arxiv.org/pdf/2606.29059v2.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features, 2022. URL https://arxiv.org/abs/2211.10831.

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, 2025.

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. URL https://arxiv.org/abs/2512.24497.

Yi Wan, Muhammad Zaheer, Adam White, Martha White, and Richard S. Sutton. Planning with expectation models. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3649–3655, 2019.

Haodong Yan, Jiaguan Zhu, Mingyuan Jia, Ruiqing Yin, Junjie He, Zhide Zhong, Junfeng Li, Jinxuan Lu, Hengtao Li, Tianran Zhang, Jiayi Chen, Wenxuan Song, Wen Chen, Yuxiang Gao, and Haoang Li. Is forward prediction enough? physical state grounding for JEPA world models, 2026. URL https://arxiv.org/abs/2608.06799.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning, 2024.

A What the objective does not guarantee

The experiments concern fitted finite networks. The statements below are about populations and fixed encoders: they say which guarantees follow from prediction error and a Gaussian latent marginal alone, and they separate the three places a downstream error can come from. None of them characterizes optimization, and none identifies the cause of an empirical failure; where the main text leans on one of them, it leans only on the part stated here. Propositions 1 and 2 are *counterexamples*: they show that a property cannot be inferred from the objective. Proposition 3 is an *identity*: it tells the diagnostics of Section 4 what they measure.

A.1 Non-collapse does not select the controlled state

Write observations as $x_t = R(p_t, b)$, where p_t is controlled state and b is an episode-constant nuisance recoverable from the image. Assume that b is independent of states and actions in the training data. At deployment, assume it is independent of the initial state and transition randomness, and does not directly affect dynamics or reward. Let SIGReg_∞ be the nonnegative population characteristic-function discrepancy, which vanishes for $\mathcal{N}(0, I_D)$ (Appendix B).

Proposition 1 (Gaussian nuisance optimum; extending Sobal et al. (2022))**.** *If the encoder class contains a map $f(R(p, b)) = \varphi(b)$ with $\varphi(b) \sim \mathcal{N}(0, I_D)$ and the predictor class contains the last-latent copy map, their population prediction loss and SIGReg_∞ are both zero. This is a global optimum with covariance I_D and $I(z_t; p_{1:T}) = 0$ under the training distribution. A deployed controller observing only these latents, time, and its own actions has the value of a randomized state-blind policy, one that may depend on time, its own past actions and private randomness but not on the state.*

The argument is simple: nuisance changes across episodes, satisfying the marginal constraint, but is constant within each episode, making prediction exact. Its deployment value follows by giving a state-blind policy the same private nuisance draw and controller randomness. The Gaussianization assumption requires a rich nuisance and a realizable measurable map; a finite texture does not literally instantiate it. The finite-batch regularizer also differs from the population functional. A complementary innovation bound in Appendix B shows that an encoder preserving all state distances at a fixed scale must incur prediction error for unpredictable state innovations. What the proposition says about the fits of Section 4 is only this: a healthy marginal and a low loss are compatible with an encoder that carries nothing a controller needs, so neither can be read as evidence that the state is kept. The failing per-frame-plus-wind cell is *not* an instance of the construction, since its loss is high, and the proposition does not explain why persistence protects the cube; the matched design measures that.

A.2 DECODABILITY DOES NOT SET A USEFUL PREDICTION SCALE

Even preserving all state information leaves its geometry unconstrained. For any $\varepsilon > 0$, we can place the two values of a relevant bit less than 2ε apart while keeping the *entire* latent distribution exactly standard Gaussian.

Proposition 2 (Exact information at arbitrarily small scale)**.** *For every $\varepsilon > 0$ and $D \geq 1$, there is a measurable observation process with episode-constant nuisance b and a fresh independent fair state bit p_t, and an encoder f_ε, such that $z_t \sim \mathcal{N}(0, I_D)$ and both p_t and b are exactly recoverable almost surely. For fixed b, $0 < \|f_\varepsilon(R(1,b)) - f_\varepsilon(R(0,b))\| < 2\varepsilon$. The optimal next-latent predictor is independent of the next bit and has coordinate-mean squared error that is positive and below ε^2/D. With reward $\mathbf{1}\{a_t = p_t\}$, a controller that observes z_t before acting attains value 1 per step, while every state-blind policy attains $1/2$.*

The construction partitions one Gaussian coordinate into narrow intervals and uses the bit to select the lower or upper half of each interval's conditional probability mass. The nuisance fixes the interval and the within-half quantile. This preserves the Gaussian marginal and exact decoding while making the two states arbitrarily close. The full proof is in Appendix B. It uses infinite-precision measurable coding; the next bit is intrinsically unpredictable. It therefore separates information from scale, without proving avoidable prediction error or explaining deletion of the predictable gate wall. Nor is f_ε what the objective selects: in the same world the nuisance-only encoder of Proposition 1 attains exactly zero, so f_ε is an (ε^2/D)-approximate optimum, and the construction bounds what the objective *permits*, not what it prefers. The class-token cell of Section 4, where a rollout error of 0.0004 of the latent variance destroys a read-out that is exact on clean latents, is the finite analogue of this scale mismatch, observed and then reproduced by injecting noise of that size; the proposition says only that the objective could not have prevented it.

A.3 WHERE A DOWNSTREAM ERROR COMES FROM

Fix a frozen encoder and a context c, the latent and action history the predictor reads, and let Z' be the next latent with conditional law P_c and conditional mean $m(c) = \mathbb{E}[Z' \mid c]$. Let $h : \mathbb{R}^D \to \mathbb{R}$ be a task quantity of the next latent (the probability that the transition it starts is fatal, its reward, a decoded pixel statistic) and $\widehat{h}$ a fitted head for it. What imagination has to supply is $u(c) = \mathbb{E}[h(Z') \mid c]$. A deterministic predictor $\widehat{g}$ supplies $\widehat{h}(\widehat{g}(c))$; a sampler with conditional law $\widehat{Q}_c$ supplies $\mathbb{E}_{\widehat{Q}_c}\widehat{h}(\widehat{Z}')$.

Proposition 3 (Three terms)**.** *For every context c,*

$$\widehat{h}(\widehat{g}(c)) - u(c) = \underbrace{[\widehat{h} - h](\widehat{g}(c))}_{\text{head error, off distribution}} + \underbrace{h(\widehat{g}(c)) - h(m(c))}_{\text{predictor approximation}} + \underbrace{h(m(c)) - \mathbb{E}[h(Z') \mid c]}_{\text{read-out gap of the mean}}. \tag{3}$$

If h is L-Lipschitz, the second term is at most $L\,\|\widehat{g}(c) - m(c)\|$ in absolute value and the third at most $L\sqrt{\mathrm{trCov}(Z' \mid c)}$; if h is twice differentiable with $\|\nabla^2 h\| \leq M$, the third is at most $\frac{M}{2}\mathrm{trCov}(Z' \mid c)$, and it vanishes when h is affine on the support of P_c or when P_c is a point mass. For a sampler,

$$\mathbb{E}_{\widehat{Q}_c}\widehat{h}(\widehat{Z}') - u(c) = \mathbb{E}_{\widehat{Q}_c}[\widehat{h} - h](\widehat{Z}') + \Big(\mathbb{E}_{\widehat{Q}_c} h(\widehat{Z}') - \mathbb{E}_{P_c} h(Z')\Big), \tag{4}$$

where the second term is at most $L\,W_1(\widehat{Q}_c, P_c)$ in absolute value, with W_1 the Wasserstein-1 distance.

The identity is a rearrangement and the bounds are Lipschitz continuity, a second-order Taylor remainder and Kantorovich–Rubinstein duality (Appendix B); its use is in what it separates. The *read-out gap of the mean* is mechanism 2 of Section 2: it needs both a spread conditional law and a read-out that is not affine across that spread, and it is the only term a distributional predictor removes, at the price of the transport term in Equation 4. The *predictor approximation* term is what a trained network adds on top of the ideal conditional mean; a small latent error controls it only relative to the read-out's sensitivity at that scale (Equation 5), which Proposition 2 shows the objective does not constrain. The *head error* is evaluated at the predicted latent, a point the head was never fitted on, and is the term the imagination policies of Section 5 are paid by. In a rollout the three terms recur at every step with the imagined history in place of c; nothing here bounds their compounding.

Two consequences organize the diagnostics. Before the wall's opening is revealed in the black gate, the two possible futures render the same solid wall, so P_c is a point mass on everything but the opening, the third term is exactly zero for any read-out of the shared wall, and a wall missing from the predicted frame is the second term; Section 4 measures it by decoding the oracle mean and the trained prediction through the same decoder. And the sampler's version replaces the read-out gap by a transport term but keeps the head term, so a world where sampling does not repair imagination (the gate with a persistent background, Section 5) is one where the head term or the transport term dominates, and the decomposition says which two quantities to measure there.

Why scale matters for a downstream read-out. For any fixed L-Lipschitz downstream score q and any coupled real and predicted next latents,

$$\left|\mathbb{E}q(Z') - \mathbb{E}q(\widehat{Z}')\right| \le L\sqrt{\mathbb{E}\|Z' - \widehat{Z}'\|^2}. \tag{5}$$

Prediction error controls score error only relative to the read-out's sensitivity, and exact decodability alone supplies no bound on that sensitivity. With $\widehat{Z}' = m(c)$ the right-hand side is $L\sqrt{\mathbb{E}\mathrm{tr}\mathrm{Cov}(Z' \mid c)}$, the read-out gap of Proposition 3; and a planner needs an argmax rather than an expectation, which survives whenever twice the right-hand side is below the read-out's gap between the best and second-best action. We do not estimate L or claim a control guarantee for the fitted networks.

B Proofs

B.1 Empirical objective and population idealization

For a training batch of B windows and H predicted time positions, the implemented loss is

$$\mathcal{L}_{\mathrm{pred}} = \frac{1}{BHD}\sum_{b,h}\|\widehat{z}_{b,h+1} - z_{b,h+1}\|^2. \tag{6}$$

The notation in Equation 1 averages over time and uses this coordinate-mean convention. One-step targets are teacher-forced at every temporal position, with a shorter history near the start of a window. For encoded windows of T frames, P random unit projection directions u_p, K knots $\tau_k \in [0, 3]$, and $w_k = e^{-\tau_k^2/2}$, the regularizer is

$$R = \frac{B}{TP}\sum_{t,p,k} q_k w_k \left[\left(\frac{1}{B}\sum_b \cos(\tau_k u_p^\top z_{b,t}) - w_k\right)^2 + \left(\frac{1}{B}\sum_b \sin(\tau_k u_p^\top z_{b,t})\right)^2\right], \tag{7}$$

where $\Delta = 3/(K-1)$, with $q_k = \Delta$ at the endpoints and 2Δ in the interior. The characteristic function is averaged across batch examples separately at each temporal position; the factor B is part of the implementation with $\lambda = 0.09$.

The population statements use the nonnegative discrepancy

$$\mathrm{SIGReg}_\infty(\mathcal{L}(z)) = \int_{\mathbb{S}^{D-1}}\int_0^3 w(\tau)\left[\left(\mathbb{E}\cos(\tau\langle z, u\rangle) - w(\tau)\right)^2 + \left(\mathbb{E}\sin(\tau\langle z, u\rangle)\right)^2\right] d\tau\, d\sigma(u), \tag{8}$$

where $w(\tau) = e^{-\tau^2/2}$ and σ is uniform on the sphere. Only the fact that this functional vanishes for a standard Gaussian is needed. It is not a theorem about exact minima of the finite-batch objective.

B.2 Nuisance optimum and the cost of innovations

Proposition 1.

Proof. Let β recover the nuisance from the observation, set $f^\star = \varphi \circ \beta$, and let $g^\star$ copy the last latent. Then $z_t = \varphi(b) = z_{t+1}$, so every prediction residual vanishes. The Gaussian marginal gives covariance I_D and zero population regularizer. Both objective terms are nonnegative, so their sum is globally minimized at zero. Training-distribution independence gives $I(\varphi(b); p_{1:T}) = 0$.

For deployment, draw nuisance B and the controller's random seed U at reset, independently of the initial state and transition randomness. Run the original controller in the environment with nuisance B. Let a state-blind policy with private seed (B, U) feed constant latent history $\varphi(B)$ to a copy of that controller and reproduce its actions with U. Coupled to the same initial state and transition randomness, the policies take the same first action. If their actions agree through step t, their states, supplied latent histories, and seeds agree, so the next actions agree as well. Induction gives identical trajectories, returns, and terminations. This does not assume independence between B and states subsequently influenced by its induced actions. □

Corollary 1 (Innovation cost for uniformly state-resolving encoders)**.** *Let $p_{t+1} = p_t + \mu(a_t) + w_t$, $\mathcal{H}_t = \sigma(p_{\le t}, b, a_{\le t})$, and $\Sigma_t = \mathrm{Cov}(w_t \mid \mathcal{H}_t)$. Assume finite second moments of states and encoded observations, $x_t = R(p_t, b)$, and $\|f(R(p, b)) - f(R(p', b))\| \ge \kappa\|p - p'\|$ for all admissible p, p', b, with $\kappa > 0$; the nuisance is $\mathcal{H}_t$-measurable, so it is fixed under the conditioning below. Every square-integrable predictor of latent and action history satisfies*

$$\frac{1}{D}\mathbb{E}\|g - f(x_{t+1})\|^2 \ge \frac{\kappa^2}{D}\mathbb{E}\mathrm{tr}\Sigma_t. \tag{9}$$

When $\mathbb{E}\mathrm{tr}\Sigma_t > 0$, this exceeds the nuisance encoder's zero prediction loss under the assumptions of Proposition 1.

The assumption is stronger than injectivity or task sufficiency: every state distance must be preserved at a uniform scale. Innovation is measured after conditioning on state history and nuisance; persistent wind is not automatically fresh innovation.

Remark 1 (The bound has a ceiling). The Gaussian marginal caps κ. If p and b are independent and $\mathrm{Cov}(z) = I_D$ with $\mathbb{E}z = 0$, two independent state draws P, P' sharing one nuisance give latents with $\mathbb{E}\|z - z'\|^2 \le 2\mathbb{E}\|z\|^2 + 2\mathbb{E}\|z'\|^2 = 4D$, while the expansion gives $\mathbb{E}\|z - z'\|^2 \ge \kappa^2\mathbb{E}\|P - P'\|^2 = 2\kappa^2\mathrm{trCov}(p)$. Hence $\kappa^2 \le 2D/\mathrm{trCov}(p)$ and the certified cost in Equation 9 is at most $2\,\mathbb{E}\mathrm{tr}\Sigma_t/\mathrm{trCov}(p)$, twice the ratio of the world's fresh noise to its state spread, in the per-coordinate units of Table 1. In the wind worlds $\mathbb{E}\mathrm{tr}\Sigma_t \approx 0.35$ and $\mathrm{trCov}(p) \approx 300$ square pixels, so the corollary can certify at most 0.0024 of prediction loss. The persistent-noise pair of Section 4 pays $0.0030 - 0.0014 = 0.0016$ extra for the wind and keeps the cube; the per-frame pair pays 0.733 extra, three hundred times the certified price, and loses it. The corollary prices the incentive; it does not explain the failing cell.

Corollary 1.

Proof. Put $Z = f(x_{t+1})$ and $\mathcal{C}_t = \sigma(z_{\le t}, a_{\le t})$. Because observations and latents are measurable functions of state and nuisance, $\mathcal{C}_t \subseteq \mathcal{H}_t$. Any predictor using a shorter window is also $\mathcal{C}_t$-measurable. Conditional least squares and the tower property give

$$\begin{aligned}\mathbb{E}\|g - Z\|^2 &\ge \mathbb{E}\|Z - \mathbb{E}[Z \mid \mathcal{C}_t]\|^2 \\ &= \mathbb{E}\|Z - \mathbb{E}[Z \mid \mathcal{H}_t]\|^2 + \mathbb{E}\|\mathbb{E}[Z \mid \mathcal{H}_t] - \mathbb{E}[Z \mid \mathcal{C}_t]\|^2 \\ &\ge \mathbb{E}\mathrm{trCov}(Z \mid \mathcal{H}_t).\end{aligned}$$

The cross term vanishes because its second factor is $\mathcal{H}_t$-measurable, by $\mathcal{C}_t \subseteq \mathcal{H}_t$, and its first has conditional mean zero given $\mathcal{H}_t$. Take conditionally independent copies P, P' of p_{t+1} given $\mathcal{H}_t$. Such copies exist for Euclidean state spaces, using their regular conditional distributions. The nuisance b and current state p_t are fixed under this conditioning. The conditional independent-copy variance identity yields

$$\begin{aligned}\mathrm{trCov}(Z \mid \mathcal{H}_t) &= \tfrac{1}{2}\mathbb{E}[\|f(R(P, b)) - f(R(P', b))\|^2 \mid \mathcal{H}_t] \\ &\ge \tfrac{\kappa^2}{2}\mathbb{E}[\|P - P'\|^2 \mid \mathcal{H}_t] \\ &= \kappa^2\mathrm{trCov}(p_{t+1} \mid \mathcal{H}_t) = \kappa^2\mathrm{tr}\Sigma_t.\end{aligned}$$

The final equality follows because $p_t + \mu(a_t)$ is $\mathcal{H}_t$-measurable. Taking expectations and dividing by D proves Equation 9. Averaging these inequalities gives the corresponding bound for a multi-time prediction objective. Proposition 1 supplies the zero-loss comparator when its assumptions also hold. □

B.3 CONDITIONAL PREDICTION AND GEOMETRIC SCALE

Lemma 1 (Conditional mean)**.** *For a fixed encoder and data law with $\mathbb{E}\|z'\|^2 < \infty$, the unique squared-error minimizer over measurable square-integrable predictors from history c and action a is $m(c,a) = \mathbb{E}[z' \mid c, a]$, up to null sets.*

Proof. Let $m = \mathbb{E}[z' \mid c, a]$. For every square-integrable measurable $g(c,a)$, expand $g - z' = (g - m) + (m - z')$. The cross term has expectation zero since $\mathbb{E}[m - z' \mid c, a] = 0$. Hence

$$\mathbb{E}\|g - z'\|^2 = \mathbb{E}\|g - m\|^2 + \mathbb{E}\mathrm{trCov}(z' \mid c, a).$$

The second term does not depend on g. The first is nonnegative and vanishes exactly when $g = m$ almost surely. □

Proposition 3.

Proof. Adding and subtracting $h(\widehat{g}(c))$ and $h(m(c))$ gives Equation 3. If h is L-Lipschitz, $|h(\widehat{g}(c)) - h(m(c))| \le L\|\widehat{g}(c) - m(c)\|$, and by Jensen's inequality $|h(m(c)) - \mathbb{E}[h(Z') \mid c]| \le \mathbb{E}[|h(m(c)) - h(Z')| \mid c] \le L\,\mathbb{E}[\|Z' - m(c)\| \mid c] \le L\sqrt{\mathrm{trCov}(Z' \mid c)}$. If h is twice differentiable with $\|\nabla^2 h\| \le M$, the integral form of Taylor's theorem gives $h(Z') = h(m) + \nabla h(m)^\top (Z' - m) + \rho$ with $|\rho| \le \frac{M}{2}\|Z' - m\|^2$; the linear term has conditional mean zero, so $|\mathbb{E}[h(Z') \mid c] - h(m)| \le \frac{M}{2}\mathbb{E}[\|Z' - m\|^2 \mid c] = \frac{M}{2}\mathrm{trCov}(Z' \mid c)$. If h is affine on the support of P_c the remainder is zero, and if P_c is a point mass then $Z' = m(c)$ almost surely. For the sampler, adding and subtracting $\mathbb{E}_{\widehat{Q}_c} h(\widehat{Z}')$ gives Equation 4, and the Kantorovich–Rubinstein duality $W_1(\widehat{Q}, P) = \sup_{\mathrm{Lip}(\phi)\le 1} |\mathbb{E}_{\widehat{Q}}\phi - \mathbb{E}_P\phi|$ applied to $\phi = h/L$ gives the transport bound. All expectations are finite because h is Lipschitz and the latents have finite second moments. □

The finite-sample regularizer. The finite-sample regularizer differs from Equation 8. For independent identically distributed samples with characteristic function ϕ and empirical estimate $\widehat{\phi}_n$,

$$\mathbb{E}|\widehat{\phi}_n(t) - w(t)|^2 = |\phi(t) - w(t)|^2 + \frac{1 - |\phi(t)|^2}{n}. \tag{10}$$

Indeed, $\mathbb{E}\widehat{\phi}_n = \phi$ and $\mathbb{E}|\widehat{\phi}_n - \phi|^2 = (1 - |\phi|^2)/n$, since each complex exponential has modulus one; expanding around ϕ gives the identity. At a Gaussian law, the residual equals $(1 - e^{-t^2})/n$. This is estimator bias, not an optimization floor. Changing the law changes both terms; dependent temporal samples introduce additional covariance terms. The implementation multiplies this statistic by batch size B, so its Gaussian finite-batch bias does not shrink as $1/B$ after that scaling. The second term decreases in $|\phi|$, so the finite-batch statistic pays a small premium for concentration: at a single knot its minimizer over real $\phi \in [0,1]$ is $\phi^\star = wB/(B-1)$, displaced $O(1/B)$ from the Gaussian in the collapse direction. For an exactly Gaussian batch of independent rows the expected value of Equation 7 is $\sum_k q_k w_k (1 - e^{-\tau_k^2}) = 1.05$ with the implementation's 17 knots on $[0, 3]$, independent of B; this is the reference against which the fitted values of Section 4 are read. Finite projections and knots, and the bounded range $\tau \le 3$, do not characterize Gaussianity.

Proposition 2.

Proof. Partition $\mathbb{R}$ into $I_j = [2\varepsilon j, 2\varepsilon(j+1))$, $j \in \mathbb{Z}$. Let $q_j = \Pr(G \in I_j)$ for $G \sim \mathcal{N}(0,1)$. Every q_j is positive and their sum is one. Let F_j be the conditional Gaussian distribution function on I_j, with inverse quantile function Q_j on $(0,1)$. The conditional CDF is continuous and strictly increasing on the interval interior, so its inverse is measurable and takes values in that interior.

Draw J with probabilities q_j, $U \sim \mathrm{Uniform}(0,1)$, and $V \sim \mathcal{N}(0, I_{D-1})$, independently. Set $b = (J, U, V)$; for $D = 1$, omit V. Take $R(p, b) = (p, b)$ and define

$$f_\varepsilon(R(p,b)) = \big(Q_J((U+p)/2), V\big). \tag{11}$$

The mixture $(U + p_t)/2$ is uniform on $(0,1)$, since p_t is an independent fair bit. Conditional on $J = j$, the first coordinate therefore has the Gaussian law restricted to I_j. Mixing with weights q_j recovers $\mathcal{N}(0,1)$. Independence of V gives the joint law $\mathcal{N}(0, I_D)$ and zero population regularizer.

Given the first coordinate y, its interval determines J almost surely. Write $v = F_J(y)$. Then $p_t = \mathbf{1}\{v > 1/2\}$ and $U = 2v - p_t$ almost surely; the other coordinates recover V. Thus the latent recovers both state and nuisance. For fixed b, the two first coordinates are distinct and lie inside the same interval of width 2ε. All remaining coordinates agree, proving the distance bound.

Latent history reveals b, while the next bit remains independent and fair. The next latent consequently has two equiprobable values, say $z^{(0)}(b)$ and $z^{(1)}(b)$. Lemma 1 gives $m(b) = (z^{(0)}(b) + z^{(1)}(b))/2$. Its conditional squared error is $\|z^{(1)}(b) - z^{(0)}(b)\|^2/4$, strictly positive and strictly below ε^2 for every b. The midpoint is square-integrable by Jensen's inequality, since both state encodings have finite second moments. Taking expectations and dividing by D proves the coordinate-mean loss bound, including strict positivity since the conditional error is positive almost surely. The midpoint depends only on b, so it is independent of p_{t+1}. Finally, a controller that observes z_t before acting decodes the current bit and plays $a_t = p_t$. A state-blind action is independent of the current fair bit, giving success probability $1/2$. The construction is not selected by the objective: the nuisance (J, U, V) alone maps to $(Q_J(U), V) \sim \mathcal{N}(0, I_D)$, an encoder of the form of Proposition 1 with exactly zero prediction loss and zero regularizer in the same world, so f_ε is an (ε^2/D)-approximate global optimum that the objective strictly prefers to discard. □

C PER-SEED TABLES BEHIND EVERY CELL

Both scorecards are generated from saved artifacts by the scripts released with the code; the tables below list every fit and every policy run that enters them, so each cell of Table 1 and Table 2 can be opened. No row is excluded and no seed or checkpoint is selected: every fit that completed its registered budget appears, and cells with fewer than three entries print their n in the main tables.

What each imagination paid. Those last two columns say what each policy was rewarded for, beside what it then did in the real world. The flow model is roughly calibrated where it works, imagining 0.88 of episodes alive on the black gate against 0.97 real, and optimistic where it does not, up to 0.71 imagined against 0.11 real under a persistent background. The JEPA's currency is wrong in both directions: in the per-frame wind world its heads end at 0.04 alive with an imagined death probability of 0.89 per step, so a dream episode lasts two steps and the gradient sees almost no consequence of an action, while in the still sibling they pay an imagined return of 8.0, 2.5 and 9.5 where the real world's best is 0. Those policies are not solving the task; they are collecting the model's mistakes, which is the same reward farming the Craftax rows show.

D DECODED STRIPS

E THE CUBE WORLDS AND THE READ-OUT PROTOCOL

Environment. The scene contains a red 6×6 object in a 63×63 RGB image with a red boundary. Four cardinal actions move the object three pixels. Reset positions are uniformly sampled from integer coordinates in $[16, 42]^2$, and initial wind is zero. In wind conditions, each wind coordinate independently increments by -1, 0, or 1 with probabilities 0.1, 0.8, and 0.1, clipped to $[-2, 2]$; the updated wind is added to the action displacement. Background pixels are uniform integers from 0 to 255; gray noise shares the same sample across RGB channels. Contact with the boundary gives reward -1 and terminates; other rewards are zero, with a 128-step cap. The gray-noise cube worlds and the gate worlds use $500{,}000$ training transitions from 100 parallel environments; the two black cube worlds use $100{,}000$; every world holds out $25{,}000$ from the same collector (Appendix F).

Read-out fitting. The nonlinear latent position probe is fitted on training data and evaluated on independent held-out frames. Its spatial softmax-and-offset output is produced by two hidden layers of width 128, trained for $10{,}000$ updates with batch $2{,}048$ and Adam at initial learning rate 10^{-3} with cosine decay. Grid inputs are standardized using training statistics. For class-token inputs, standardization and PCA whitening are selected using the final 10% of fitting rows, then the chosen variant is refitted on the full fitting split; whitening floors singular values at 1% of the largest. The pixel probe is a separate supervised CNN read on frames from the off-budget decoder, trained with synthetic background replacement using ground-truth cube masks; both probes are diagnostics,

Table 3: **Scorecard 1, per fit.** Both pre-registered read-outs for every cell: the latent probe on the frozen encoding and the pixel probe on decoded frames, on encodings of held-out observations (ENC) and at the eighth predicted step (PRED), with the exact step-8 floor, the persistence reference, and the gate quantities where they apply. The main table reports the better of the two probes; both are here for every seed. Every fit that completed its registered budget appears, in seed order, with no seed or checkpoint selection.

world	arm	seed	ENC	PRED	floor	latent_enc	latent_pred	pixel_enc	pixel_pred	wall_enc	wall_pred1	auroc_pred	updates
black, still	class-token JEPA	0	0.09	18.41	0.00	0.09	45.30	17.01	18.41	–	–	–	23750
black, still	class-token JEPA	1	0.09	18.40	0.00	0.09	44.88	17.00	18.40	–	–	–	23750
black, still	class-token JEPA	2	0.09	18.40	0.00	0.09	45.61	17.01	18.40	–	–	–	23750
black, still	grid JEPA	0	0.10	3.42	0.00	0.10	3.93	0.27	3.42	–	–	–	8500
black, still	grid JEPA	1	0.10	3.57	0.00	0.10	3.89	0.27	3.57	–	–	–	8500
black, still	grid JEPA	2	0.09	3.05	0.00	0.09	3.57	0.27	3.05	–	–	–	8500
black, still	flow	0	0.13	0.13	0.00	0.13	0.13	0.23	0.24	–	–	–	–
black, still	flow	1	0.12	0.11	0.00	0.12	0.11	0.23	0.24	–	–	–	–
black, still	flow	2	0.12	0.12	0.00	0.12	0.12	0.23	0.24	–	–	–	–
black, wind	class-token JEPA	0	0.00	8.85	7.90	0.00	9.14	0.22	8.85	–	–	–	23750
black, wind	class-token JEPA	1	0.00	8.79	7.90	0.00	9.11	0.22	8.79	–	–	–	23750
black, wind	class-token JEPA	2	0.00	8.72	7.90	0.00	8.82	0.22	8.72	–	–	–	23750
black, wind	grid JEPA	0	0.13	11.74	7.90	0.13	11.74	0.26	12.10	–	–	–	8500
black, wind	grid JEPA	1	0.14	11.49	7.90	0.14	11.52	0.29	11.49	–	–	–	8500
black, wind	grid JEPA	2	0.13	11.30	7.90	0.13	11.30	0.26	11.37	–	–	–	8500
black, wind	flow	0	0.13	11.22	11.76	0.13	11.28	0.22	11.22	–	–	–	–
black, wind	flow	1	0.14	11.27	11.76	0.14	11.30	0.22	11.27	–	–	–	–
black, wind	flow	2	0.13	10.89	11.76	0.13	10.93	0.22	10.89	–	–	–	–
per-frame noise, still	class-token JEPA	0	20.80	20.29	0.00	23.96	20.29	20.80	22.99	–	–	–	24000
per-frame noise, still	class-token JEPA	1	20.76	20.18	0.00	23.91	20.18	20.76	21.52	–	–	–	24000
per-frame noise, still	class-token JEPA	2	20.82	19.13	0.00	23.98	19.86	20.82	19.13	–	–	–	24000
per-frame noise, still	grid JEPA	0	0.12	4.27	0.00	0.12	4.27	0.27	4.46	–	–	–	8500
per-frame noise, still	grid JEPA	1	0.13	6.09	0.00	0.13	6.15	0.27	6.09	–	–	–	8500
per-frame noise, still	grid JEPA	2	0.10	4.11	0.00	0.10	4.11	0.27	4.26	–	–	–	8500
per-frame noise, still	flow	0	0.26	0.39	0.00	3.54	3.20	0.26	0.39	–	–	–	–
per-frame noise, still	flow	1	0.26	0.31	0.00	2.91	2.68	0.26	0.31	–	–	–	–
per-frame noise, still	flow	2	0.26	0.30	0.00	3.14	2.55	0.26	0.30	–	–	–	–
per-frame noise, wind	class-token JEPA	0	17.95	19.57	7.90	25.29	19.57	17.95	22.37	–	–	–	24000
per-frame noise, wind	class-token JEPA	1	18.09	19.74	7.90	25.22	19.74	18.09	22.94	–	–	–	24000
per-frame noise, wind	class-token JEPA	2	18.09	19.53	7.90	25.36	19.53	18.09	20.54	–	–	–	24000
per-frame noise, wind	grid JEPA	0	17.54	19.70	7.90	21.35	19.70	17.54	48.07	–	–	–	8500
per-frame noise, wind	grid JEPA	1	17.63	20.03	7.90	21.37	20.03	17.63	33.42	–	–	–	8500
per-frame noise, wind	grid JEPA	2	17.71	21.44	7.90	21.26	21.44	17.71	24.73	–	–	–	8500
per-frame noise, wind	flow	0	0.25	12.25	11.76	3.68	13.96	0.25	12.25	–	–	–	–
per-frame noise, wind	flow	1	0.25	12.74	11.76	2.44	13.77	0.25	12.74	–	–	–	–
per-frame noise, wind	flow	2	0.25	12.63	11.76	2.87	14.10	0.25	12.63	–	–	–	–
persistent noise, still	class-token JEPA	0	18.57	19.61	0.00	21.32	21.28	18.57	19.61	–	–	–	24000
persistent noise, still	class-token JEPA	1	18.34	19.72	0.00	20.82	21.21	18.34	19.72	–	–	–	24000
persistent noise, still	class-token JEPA	2	19.15	20.76	0.00	21.48	20.95	19.15	20.76	–	–	–	24000
persistent noise, still	grid JEPA	0	0.71	1.78	0.00	0.71	1.78	1.44	3.14	–	–	–	8500
persistent noise, still	grid JEPA	1	0.84	1.64	0.00	0.84	1.64	1.98	3.71	–	–	–	8500
persistent noise, still	grid JEPA	2	0.73	2.07	0.00	0.73	2.07	1.63	3.85	–	–	–	8500
persistent noise, still	flow	0	0.40	1.54	0.00	3.38	3.67	0.40	1.54	–	–	–	–
persistent noise, still	flow	1	0.74	2.66	0.00	12.36	16.02	0.74	2.66	–	–	–	–
persistent noise, still	flow	2	0.61	1.16	0.00	6.16	5.49	0.61	1.16	–	–	–	–
persistent noise, wind	class-token JEPA	0	15.98	19.46	7.90	22.97	27.42	15.98	19.46	–	–	–	23750
persistent noise, wind	class-token JEPA	1	15.81	19.25	7.90	22.75	25.97	15.81	19.25	–	–	–	23750
persistent noise, wind	class-token JEPA	2	16.26	19.75	7.90	22.75	25.08	16.26	19.75	–	–	–	23750
persistent noise, wind	grid JEPA	0	0.72	11.55	7.90	0.72	11.55	1.22	13.01	–	–	–	8500
persistent noise, wind	grid JEPA	1	0.67	10.77	7.90	0.67	10.77	1.09	11.08	–	–	–	8500
persistent noise, wind	grid JEPA	2	0.62	10.53	7.90	0.62	10.53	1.20	10.96	–	–	–	8500
persistent noise, wind	flow	0	0.25	13.98	11.76	2.51	14.71	0.25	13.98	–	–	–	–
persistent noise, wind	flow	1	0.25	13.14	11.76	2.33	13.49	0.25	13.14	–	–	–	–
persistent noise, wind	flow	2	0.25	13.23	11.76	2.92	14.24	0.25	13.23	–	–	–	–
gate	class-token JEPA	0	7.60	17.21	0.00	7.60	42.52	15.50	17.21	0.33	0.65	0.50	23500
gate	class-token JEPA	1	6.18	17.22	0.00	6.18	42.54	15.52	17.22	1.00	1.00	0.49	23500
gate	class-token JEPA	2	6.42	17.21	0.00	6.42	39.84	15.51	17.21	0.32	0.64	0.50	23500
gate	grid JEPA	0	0.10	3.66	0.00	0.10	3.82	0.23	3.66	1.00	0.00	0.50	8500
gate	grid JEPA	1	0.10	3.45	0.00	0.10	3.55	0.27	3.45	0.99	0.00	0.50	8500
gate	grid JEPA	2	0.10	2.61	0.00	0.10	2.61	0.23	2.80	0.99	0.53	0.56	8500
gate	flow	0	0.18	0.48	0.00	0.18	0.48	0.23	0.64	1.00	1.00	1.00	–
gate	flow	1	0.17	0.34	0.00	0.17	0.34	0.23	0.45	1.00	1.00	1.00	–
gate	flow	2	0.17	0.32	0.00	0.17	0.32	0.23	0.42	1.00	1.00	1.00	–
gate + per-frame noise	class-token JEPA	0	20.17	17.23	0.00	20.20	20.59	20.17	17.23	0.44	1.00	0.50	24000
gate + per-frame noise	class-token JEPA	1	20.04	21.03	0.00	20.27	21.03	20.04	24.18	0.44	1.00	0.50	24000
gate + per-frame noise	class-token JEPA	2	20.27	22.60	0.00	20.28	22.71	20.27	22.60	0.44	1.00	0.50	24000
gate + per-frame noise	grid JEPA	0	0.38	5.01	0.00	0.68	5.01	0.38	5.75	1.00	0.94	0.64	8500
gate + per-frame noise	grid JEPA	1	12.59	23.02	0.00	13.57	23.02	12.59	25.50	1.00	1.00	0.95	8500
gate + per-frame noise	grid JEPA	2	1.24	15.47	0.00	6.26	15.47	1.24	21.76	1.00	0.78	0.50	8500
gate + per-frame noise	flow	0	0.26	0.48	0.00	2.63	2.62	0.26	0.48	1.00	1.00	1.00	–
gate + per-frame noise	flow	1	0.26	0.38	0.00	2.62	2.55	0.26	0.38	1.00	1.00	1.00	–
gate + per-frame noise	flow	2	0.27	0.42	0.00	2.89	2.96	0.27	0.42	1.00	1.00	1.00	–
gate + persistent noise	class-token JEPA	0	18.79	21.08	0.00	20.53	21.08	18.79	21.28	0.22	0.23	0.52	24000
gate + persistent noise	class-token JEPA	1	18.30	20.93	0.00	20.78	25.03	18.30	20.93	0.22	0.22	0.52	24000
gate + persistent noise	class-token JEPA	2	18.43	21.01	0.00	20.29	23.14	18.43	21.01	0.22	0.24	0.52	24000
gate + persistent noise	grid JEPA	0	1.02	3.10	0.00	1.02	3.10	1.42	4.41	0.83	0.86	0.99	8500
gate + persistent noise	grid JEPA	1	1.10	3.41	0.00	1.10	3.41	1.26	4.44	0.83	0.91	0.98	8500
gate + persistent noise	grid JEPA	2	1.15	3.60	0.00	1.15	3.60	1.28	4.65	0.82	0.92	1.00	8500
gate + persistent noise	flow	0	0.27	1.22	0.00	2.88	5.23	0.27	1.22	1.00	1.00	1.00	–
gate + persistent noise	flow	1	0.26	1.24	0.00	2.93	6.74	0.26	1.24	1.00	0.97	1.00	–
gate + persistent noise	flow	2	0.26	0.62	0.00	2.66	3.36	0.26	0.62	1.00	1.00	1.00	–

world	arm	seed	ENC	PRED	persistence	AF	updates
Craftax-Classic	class-token JEPA	0	0.01843	0.02428	0.01333	0.702	24000
Craftax-Classic	class-token JEPA	1	0.01917	0.02413	0.01333	0.707	24000
Craftax-Classic	class-token JEPA	2	0.01821	0.02360	0.01333	0.716	24000
Craftax-Classic	grid JEPA	0	0.00860	0.01137	0.01333	0.885	8500
Craftax-Classic	grid JEPA	1	0.00789	0.01125	0.01333	0.886	8500
Craftax-Classic	grid JEPA	2	0.00779	0.01160	0.01333	0.878	8500
Craftax-Classic	flow	0	0.00007	0.00124	0.01333	0.931	9000+16000
Craftax-Classic	flow	1	0.00007	0.00122	0.01333	0.942	9000+16000
Craftax-Classic	flow	2	0.00008	0.00136	0.01333	0.921	9000+16000

never a policy or world-model training signal. Applying either read-out to predicted latents tests prediction and read-out extrapolation together, which is why a cell is reported as the better of the two and why the behavioural scorecard, not the probe, carries the claims about imagination. Rollout windows are sampled with replacement and overlap and exclude death or reset during the eight context and 16 forecast frames; the benchmark uses 2,048 windows for video metrics and the first 512 for position diagnostics. Surviving windows are a selected subset of wind trajectories, so PRED does not by itself establish death prediction or closed-loop control.

What the wind-world reference scores are. The two numbers printed against PRED in the wind worlds are computed exactly by a dynamic programme over the simulator's position-and-wind law from the last context frame under the recorded actions, on the same 512 windows the cells use. They are the scores of two ideal forecasters that know the true dynamics but not that the window was later selected for surviving all 24 frames: an exact conditional-mean forecast scores 7.90 pixels at step 8, an exact one-sample forecast of the same law 11.76. Neither is a lower bound on the metric. Because the realized position is that of a selected window, a forecast biased toward the safe interior scores better than either: the infimum over all point forecasts, the median of the selected law, is 7.44,

Table 4: **The gate worlds' wall and gap quantities.** Decoded intensity of the wall's shared solid part on real encodings and at the model's own first and eighth predicted step, and the AUROC for the revealed opening's side on real encodings and on predicted latents; mean ± sample standard deviation over the same fits as Table 1, every fit entering with no seed or checkpoint selection. The quantities on real encodings set the **red** and **green** conditions of the gate rows. Intensity on decoded predictions saturates when a decoder renders a bright field, so AUROC is the discriminative quantity of the pair.

world	recipe	wall, real	wall, pred 1	wall, pred 8	AUROC, real	AUROC, pred
gate	class-token JEPA	0.55±0.39	0.76±0.20	0.26±0.44	0.55±0.09	0.50±0.01
gate	grid JEPA	0.99±0.00	0.18±0.31	0.00±0.01	1.00±0.00	0.52±0.03
gate	flow	1.00±0.00	1.00±0.00	0.96±0.02	1.00±0.00	1.00±0.00
gate + per-frame noise	class-token JEPA	0.44±0.00	1.00±0.00	0.00±0.00	0.50±0.00	0.50±0.00
gate + per-frame noise	grid JEPA	1.00±0.00	0.91±0.12	0.54±0.43	1.00±0.00	0.69±0.23
gate + per-frame noise	flow	1.00±0.00	1.00±0.00	0.94±0.02	1.00±0.00	1.00±0.00
gate + persistent noise	class-token JEPA	0.22±0.00	0.23±0.01	0.22±0.00	0.52±0.00	0.52±0.00
gate + persistent noise	grid JEPA	0.83±0.00	0.90±0.03	0.50±0.14	1.00±0.00	0.99±0.01
gate + persistent noise	flow	1.00±0.00	0.99±0.02	0.89±0.14	1.00±0.00	1.00±0.00

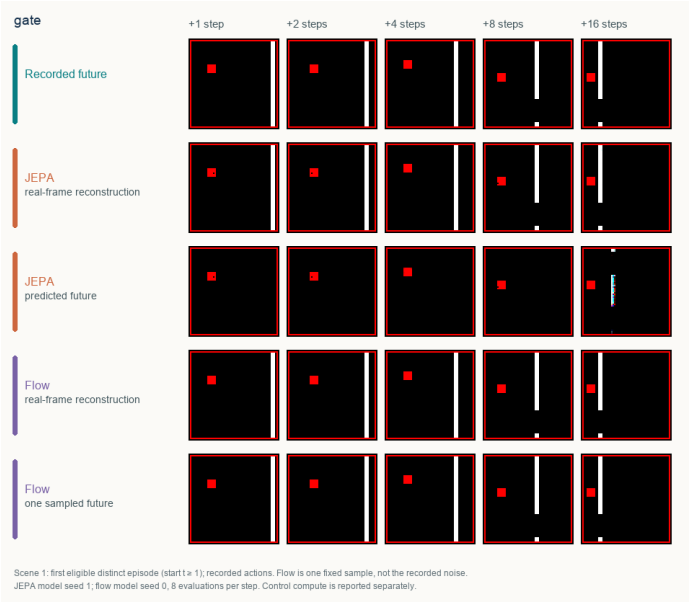


Figure 4: **The wall survives encoding and dies in prediction.** From one pre-reveal context and the recorded actions: the simulator; the JEPA's decoded encodings of the real future; the JEPA's decoded rollout; the flow model's decoded encodings and one sampled future. First eligible held-out episode, chosen without looking at model outputs; JEPA model seed 1, flow seed 0. Quantities for all three seeds are in Table 1; a decoding failure on predicted latents mixes prediction error with off-distribution read-out, which is why the behavioural comparison in Section 5 is what turns this picture into a claim.

and one sample of the selected law scores 10.60; the published median reference of 8.14 exceeds the mean reference for the same reason. A cell can therefore sit below its reference, as the flow arm's black-wind cell does at 11.1. The reference is assigned by what the arm's predictor is by construction, a deterministic map or a sampler, and this assignment decides one colour: the grid JEPA's black-wind cell reads 11.5 against 7.90 and is **amber**, the flow's 11.1 against 11.76 is **green**, a difference of 0.4 pixels in the measurement. Read on the one-parameter family "conditional mean plus a fraction ℓ of a sampled innovation", whose step-8 score rises from 7.90 at $\ell = 0$ through 8.76, 9.58, 10.60 and 11.17 at $\ell = 0.4, 0.6, 0.8, 0.9$ to 11.76 at $\ell = 1$, the same row reads: class token $\ell \approx 0.4$, grid JEPA $\ell \approx 0.95$, flow $\ell \approx 0.9$; the grid JEPA's predicted latents read out at nearly full dispersion. Both probes are nonlinear, so a probe applied to a predicted latent does not commute

Table 5: **Scorecard 2, per policy run.** Survival, walls passed, control length and return of every policy run on the world's fixed 256-episode packet, with the model fit and policy seed it used and the number of PPO updates in its registered budget. State-blind rules are listed with the same evaluator; their control length is the quantity the noisy gate worlds are compared on, since every rule and policy there scores zero survival. The last two columns are the imagined side of the same run at its final update: the fraction of the model's own rollouts still alive and the return its own heads paid, which is what the policy was optimizing.

world	arm	model seed	policy seed	updates	survival	walls passed	control length	return	imagined alive	imagined return
per-frame noise, still	grid JEPA, imagined	0	0	50	0.965	0.00	119.9	-0.035	0.94	8.01
per-frame noise, still	grid JEPA, imagined	0	1	50	0.938	0.00	117.8	-0.062	0.82	2.45
per-frame noise, still	grid JEPA, imagined	0	2	50	0.633	0.00	90.3	-0.367	0.94	9.51
per-frame noise, still	flow, imagined	0	0	50	0.953	0.00	119.2	-0.047	0.91	-0.01
per-frame noise, still	flow, imagined	0	1	50	0.969	0.00	119.5	-0.031	0.89	-0.03
per-frame noise, still	flow, imagined	0	2	50	0.992	0.00	120.6	-0.008	0.87	-0.01
per-frame noise, still	blind: random	–	–	–	0.238	–	74.0	–	–	–
per-frame noise, still	blind: alternate	–	–	–	1.000	–	121.0	–	–	–
per-frame noise, wind	grid JEPA, imagined	0	0	50	0.000	0.00	10.2	-1.000	0.04	-1.54
per-frame noise, wind	grid JEPA, imagined	0	1	50	0.000	0.00	11.4	-1.000	0.04	-1.53
per-frame noise, wind	grid JEPA, imagined	0	2	50	0.000	0.00	16.0	-1.000	0.04	-1.53
per-frame noise, wind	flow, imagined	0	0	50	0.945	0.00	118.1	-0.055	0.98	-0.03
per-frame noise, wind	flow, imagined	0	1	50	0.129	0.00	53.9	-0.871	0.78	-0.34
per-frame noise, wind	flow, imagined	0	2	50	0.652	0.00	102.2	-0.348	0.95	-0.10
per-frame noise, wind	blind: random	–	–	–	0.000	–	17.4	–	–	–
per-frame noise, wind	blind: alternate	–	–	–	0.000	–	18.2	–	–	–
persistent noise, wind	grid JEPA, imagined	0	0	50	0.027	0.00	32.6	-0.973	0.56	-0.67
persistent noise, wind	grid JEPA, imagined	0	1	50	0.020	0.00	31.1	-0.980	0.55	-0.69
persistent noise, wind	grid JEPA, imagined	0	2	50	0.000	0.00	16.7	-1.000	0.45	-0.71
persistent noise, wind	flow, imagined	0	0	50	0.891	0.00	115.9	-0.109	0.98	-0.05
persistent noise, wind	flow, imagined	0	1	50	0.207	0.00	69.2	-0.793	0.80	-0.62
persistent noise, wind	flow, imagined	0	2	50	0.137	0.00	51.9	-0.863	0.53	-0.71
persistent noise, wind	blind: random	–	–	–	0.000	–	17.4	–	–	–
persistent noise, wind	blind: alternate	–	–	–	0.000	–	18.2	–	–	–
gate	grid JEPA, imagined	1	0	150	0.027	0.73	46.2	-0.973	0.28	-0.10
gate	grid JEPA, imagined	1	1	150	0.008	0.23	36.2	-0.992	0.37	-0.05
gate	grid JEPA, imagined	1	2	150	0.027	0.70	44.0	-0.973	0.37	-0.06
gate	flow, imagined	0	0	150	0.988	3.97	120.3	-0.012	0.88	-0.16
gate	flow, imagined	0	1	150	0.941	3.86	117.7	-0.059	0.88	-0.15
gate	flow, imagined	0	2	150	0.977	3.93	119.5	-0.023	0.85	-0.18
gate	grid JEPA decoded, real	1	0	150	0.977	3.95	119.9	-0.023	0.85	-0.20
gate	grid JEPA decoded, real	1	1	150	0.980	3.96	120.1	-0.020	0.85	-0.18
gate	grid JEPA decoded, real	1	2	150	0.945	3.91	118.9	-0.055	0.85	-0.20
gate	flow decoded, real	0	0	150	0.957	3.88	118.2	-0.043	0.87	-0.17
gate	flow decoded, real	0	1	150	0.516	3.36	103.8	-0.484	0.80	-0.24
gate	flow decoded, real	0	2	150	0.953	3.92	118.9	-0.047	0.83	-0.21
gate	blind: random	–	–	–	0.000	–	24.0	–	–	–
gate	blind: alternate	–	–	–	0.000	–	24.6	–	–	–
gate + per-frame noise	grid JEPA, imagined	0	0	150	0.000	0.07	33.2	-1.000	0.89	0.48
gate + per-frame noise	grid JEPA, imagined	1	1	150	0.000	0.13	22.7	-1.000	0.11	-0.43
gate + per-frame noise	grid JEPA, imagined	2	2	150	0.000	0.00	21.9	-1.000	0.77	0.40
gate + per-frame noise	flow, imagined	0	0	150	0.980	3.98	120.6	-0.020	0.89	-0.13
gate + per-frame noise	flow, imagined	1	1	150	0.941	3.77	115.5	-0.059	0.87	-0.16
gate + per-frame noise	flow, imagined	2	2	150	0.730	3.21	101.9	-0.270	0.79	-0.28
gate + per-frame noise	grid JEPA decoded, real	0	0	150	0.953	3.89	118.5	-0.047	0.87	-0.18
gate + per-frame noise	grid JEPA decoded, real	1	1	150	0.523	2.56	84.6	-0.477	0.69	-0.45
gate + per-frame noise	grid JEPA decoded, real	2	2	150	0.492	3.35	103.5	-0.508	0.81	-0.23
gate + per-frame noise	flow decoded, real	0	0	150	0.945	3.88	118.2	-0.055	0.87	-0.17
gate + per-frame noise	flow decoded, real	1	1	150	0.582	2.83	93.4	-0.418	0.76	-0.34
gate + per-frame noise	flow decoded, real	2	2	150	0.961	3.97	120.3	-0.039	0.78	-0.26
gate + per-frame noise	blind: random	–	–	–	0.000	–	24.0	–	–	–
gate + per-frame noise	blind: alternate	–	–	–	0.000	–	24.6	–	–	–
gate + persistent noise	grid JEPA, imagined	0	0	150	0.000	0.04	26.6	-1.000	0.48	-0.45
gate + persistent noise	grid JEPA, imagined	1	1	150	0.000	0.00	19.1	-1.000	0.31	-0.68
gate + persistent noise	grid JEPA, imagined	2	2	150	0.000	0.00	15.3	-1.000	0.40	-0.56
gate + persistent noise	flow, imagined	0	0	150	0.000	0.00	21.5	-1.000	0.25	-0.99
gate + persistent noise	flow, imagined	1	1	150	0.039	0.79	47.8	-0.961	0.57	-0.76
gate + persistent noise	flow, imagined	2	2	150	0.113	1.70	67.8	-0.887	0.71	-0.46
gate + persistent noise	grid JEPA decoded, real	0	0	150	0.906	3.79	116.0	-0.094	0.84	-0.21
gate + persistent noise	grid JEPA decoded, real	1	1	150	0.961	3.88	118.3	-0.039	0.83	-0.21
gate + persistent noise	grid JEPA decoded, real	2	2	150	0.945	3.91	118.8	-0.055	0.82	-0.22
gate + persistent noise	flow decoded, real	0	0	150	0.973	3.94	119.6	-0.027	0.87	-0.17
gate + persistent noise	flow decoded, real	1	1	150	0.309	3.04	95.7	-0.691	0.82	-0.26
gate + persistent noise	flow decoded, real	2	2	150	0.922	3.84	117.2	-0.078	0.84	-0.21
gate + persistent noise	blind: random	–	–	–	0.000	–	24.0	–	–	–
gate + persistent noise	blind: alternate	–	–	–	0.000	–	24.6	–	–	–

with the expectation inside it, and no sentence in the paper reads a model's predictor off its probe score. The "better of two probes" rule biases PRED downward, toward **green**; it was registered to protect a cell from a weak decoder, and the per-probe values are in Table 3.

Detached-target refits. For the two per-frame gray conditions, three model seeds per world are refitted with the prediction target detached from the encoder's gradient (`stop_gradient` on z_{t+1}

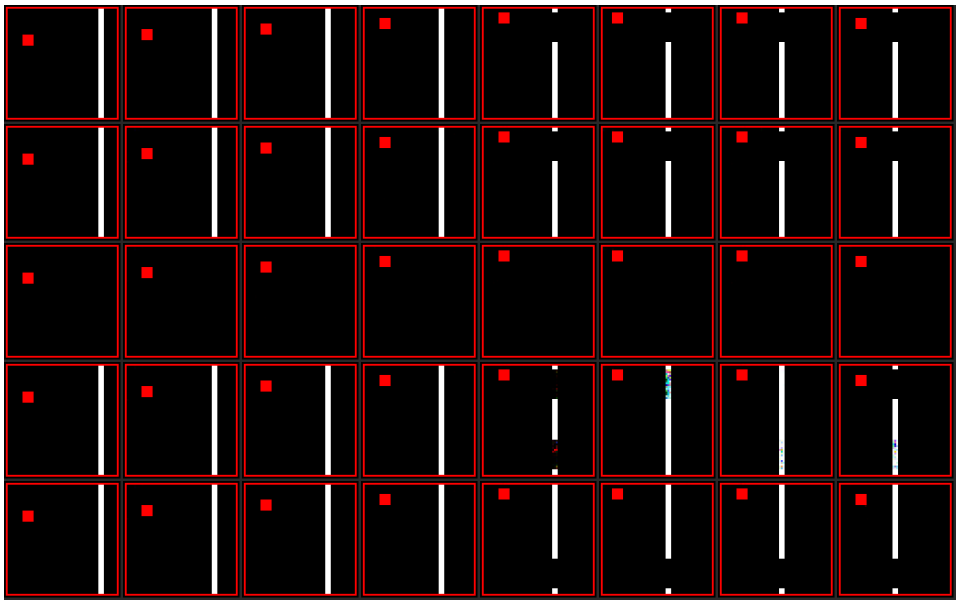


Figure 5: **The oracle conditional mean keeps the wall; the trained predictor does not.** Black gate, JEPA model seed 0, one held-out context whose wall is still solid at $x \in \{51, 54\}$ and whose opening is revealed at the fourth or fifth predicted step; eight predicted steps under the recorded actions. Rows: the real future; the decoded encoding of the real future; the trained predictor's rollout; the decoded average of the frozen encodings of the two possible futures, the recorded opening and the other one, which is the exact conditional mean given the context before the reveal; the decoded encoding of the other-opening future. Faint artefacts at the opening in the fourth row after the reveal are the decoder rendering the average of two encodings. Quantities for every context and fit are in Section 4; the strip is the first context of its group, chosen without looking at model output.

Table 6: **Detaching the target does not rescue the wind cell and costs the deterministic one.** Position read-out error (mean absolute pixel error per coordinate), mean $\pm$ sample standard deviation over three model seeds; each seed pair shares initialization and data order, and every fit enters with no seed or checkpoint selection.

Per-frame gray noise	Joint target	Detached target
Deterministic	0.37 ± 0.05	10.93 ± 4.90
Wind	14.51 ± 0.12	14.51 ± 0.07

in Equation 1); the predictor-input and SIGReg gradients, architecture, data order and initialization are unchanged, and the joint control of each seed is retrained by the same script so that the pair differs only in the target gradient (a bit-exact equivalence test against the original trainer is part of the test suite). A GELU read-out with two hidden layers of width 256 on the raw 512 latent coordinates is fitted on 50,000 training encodings per model and scored on 10,000 frames of an independently collected archive as mean absolute pixel error per coordinate (Table 6); this protocol differs from the probe of Table 1, so the two tables are not numerically comparable. Detaching the target is the obvious suspicion about mechanism 1 — that the encoder is shaping its own target — and it does not rescue the failing cell: the wind world stays at chance and the deterministic sibling gets worse.

Amplitude intervention. For the black still world, the same latent tile probe is read on three inputs over the first 128 metric windows of the cell evaluation, for the seed-0 fit of each arm: the true encodings of the recorded future, the model's own predicted latents, and the true encodings perturbed by isotropic Gaussian noise whose variance equals that arm's own rollout mean squared error at that step (3.5×10^{-4} of the variance for the class token, 2.2×10^{-4} for the grid). The

perturbation therefore has exactly the size of each predictor's error but none of its structure; the 18.40 quoted in Section 4 is the cell's pixel probe over 512 windows and is not the same read-out. The class-token model's rollout error is 0.0004 of its latent variance at every step; its read-out gives 0.0005 pixels on clean latents, 39.4 on predicted ones and 46.5 under the noise. The grid model's rollout error grows from 0.0002 to 0.20 of its variance over eight steps; its read-out gives 0.09 pixels clean, 0.21 to 3.57 on predicted latents and 0.13 to 2.65 under the matched noise. Two readings follow: the read-out's sensitivity to an unstructured perturbation of the predictor's own magnitude accounts for both arms' rollout errors, and at equal *relative* error the grid latent is two orders of magnitude more robust than the class token. Nothing is trained in this diagnostic and no seed or window is selected.

F HYPERPARAMETERS AND COMPUTE

The gate and gray-noise cube conditions supply 500,000 uniform-random transitions each. The earlier black and color-noise cube datasets contain 100,000 transitions. Craftax uses its separate 500,000-transition archive. Model training uses bfloat16 compute, float32 parameters, and one A100. A 500-step calibration estimates the training length fitting 575 seconds including compilation; the grid gate model trains for 8,500 steps (17,000 and 34,000 in the earlier budget extensions of Appendix P), the class-token cube models for 23,750–24,000. Model seeds 0, 1, 2 of the grid JEPA differ only in the initialisation and data-order seed.

JEPA settings. Both JEPA architectures use batch 128 and AdamW with learning rate and weight decay 10^{-3}, 10% warm-up followed by cosine decay to zero, and gradient clipping at 1.0. SIGReg uses weight 0.09, 17 knots on $[0, 3]$, and 1024 random projections. Neither architecture uses a stop-gradient, teacher, decoder, or reward during world-model fitting. Linear–BatchNorm–GELU–linear projection heads flatten batch, time, and spatial tokens for normalization. Training-mode statistics therefore couple examples and temporal positions; at inference, running statistics define a fixed pointwise projection and the attention predictor is causal. The theoretical causal-map formulation is an inference/population idealization.

	Class-token JEPA	Grid JEPA
encoder	ViT, 7×7 patches	tile transformer + conv
width / depth / heads	192 / 6 / 3	128 / 6 / 8
encoder MLP	768	$4\times$ width
read-out	class token	4×4 grid, 32 channels
latent dimension	192	512
projector	BN projector, width 2048	same, hidden 512
predictor	AdaLN-zero AR	spatial + causal temporal
predictor depth / heads / dim	6 / 16 / 64	6 / 8 / 32
predictor MLP / width	2048 / latent	$2\times$ width / 256
history	3 frames	8 frames
dropout	0.1	0
parameters	15.25M	11.12M

Flow settings. The tokenizer uses 9,000 steps, batch 256, and learning rate 9×10^{-4}; dynamics 16,000 steps, batch 64, the same learning rate, width 256, six layers, eight spatial and temporal attention heads, a 32-frame context, independent per-frame noise levels, a shortcut loss of weight 0.25 every fourth step after the first quarter of training, history dropout 0.25, two history frames resampled every eight steps after half of training, and a temporal loss weight of 0.25. The symlog reward head has 255 bins over $[-20, 20]$; reward and termination losses have weight 0.1 and enter the trunk at full strength for half of training, then at factor 0.1. Inference uses four function evaluations per frame. We corrected terminal-transition masking in the recipe's reward loss (Appendix N).

JEPA read-outs. Gate post-hoc heads use the frozen encoder and the first 800 time rows (80,000 frames) of the same archive used for world-model training. The following 200 time rows are held out from head fitting only; the independent world-model evaluation archive is `eval.npz`. The pixel decoder uses 8,000 steps, batch 256, learning rate 10^{-3}, width 256, six residual blocks; it supports the figures and the decoded-latent control, in which its frozen output, clipped to $[0, 255]$, replaces the pixels of the real-gate PPO; no true pixel enters that policy after the encoder. Reward, termination, and behaviour-value heads are two-layer perceptrons of width 512 on the latent and the one-hot action, 6,000 steps, batch 1024, learning rate 10^{-3}, Monte Carlo value targets at discount 0.99, value loss weight 0.1. The residual predictor models $z_{t+1} - m$ with $m = g(z_{t-7:t}, a_{t-7:t})$ the original grid prediction, conditioned on m, z_t, and the action: three residual blocks of width 1024, 8,000 steps, and batch 2048. The final 20,000 valid windows of the same world-model training archive are excluded from residual fitting and used for its quality check. They are not independent world-model evaluation data. For encoder seed 0, coordinate-mean MSE is 3.07×10^{-4} for the original predictor, 2.05×10^{-4} for one residual sample, and 1.32×10^{-4} for the mean of eight. This

quality check uses eight function evaluations per sample; the reported PPO runs instead use four. Averaging eight learned samples is a random Monte Carlo average, not an exact conditional mean.

Policy optimisation (historical protocols). The settings in this paragraph are those of the earlier gate and cube studies of Appendix P; the shared-policy comparison of the main text uses Appendix H and the Craftax comparison Appendix O. PPO uses discount 0.99, GAE parameter 0.9, clipping 0.2, value coefficient 0.5, entropy coefficient 0.01, AdamW at 2.5×10^{-4} with weight decay 0.01, four epochs per batch, and gradient clipping at 1.0. Pixels pass through a three-block Impala torso (Espeholt et al., 2018), latents through a two-layer perceptron; a GRU receives that embedding and the previous action; the critic uses 64 categorical bins with histogram-Gaussian targets (Farebrother et al., 2024). Real-environment PPO uses 128 environments and 64 steps per iteration with four minibatch groups, truncated backpropagation over 16 steps, and 1.5M environment steps. In-model PPO uses 150 updates of 512 trajectories of 16 steps (1.2M imagined steps) with minibatch size 128; half of the starts are resets, half dataset states with their real eight-frame context. All reported policy rows use final checkpoints.

Compute. On a shared A100 the seed-0 grid-gate JEPA completes its $8{,}500$ updates in 573 seconds against the 575-second budget, with a logged total of 586.5 seconds including compilation. Every read-out of a frozen model is off budget, and each such stage — probe, head and decoder fitting, the residual sampler, one policy run, the latent diagnostic — costs minutes rather than hours; the sample-averaging rollouts of Appendix P are the most expensive, tens of minutes per run. Per-stage wall-clock is logged beside every run.

G Evaluation protocol

This section states the earlier gate protocol of Appendix P; the main text's policy comparison follows Appendix H instead, and the Craftax one Appendix O. Every gate-policy endpoint uses the final checkpoint and 256 real episodes from reset, with the 128-step cap. Survival is the fraction still alive after step 128; death on that step is not survival. Walls passed counts wall exits before or on the terminal transition, including an exit on the same step as death. Discovery is final survival strictly above 50%. Real-training and imagined-training runs advance their random keys under different iteration schedules; their final evaluations are not paired episode trials. Policy seeds are separate runs conditional on a fitted model; independently fitted encoders provide a different replication level. That protocol has a single flow checkpoint, against three independent fits per recipe in the campaign of Table 2. Reported means weight policy runs equally; standard deviations describe variation across those runs, not uncertainty across model families.

Gate specification. The gate uses the same image and object sizes and cardinal action displacement. A white vertical wall of width three pixels spawns at $x = 60$ every 22 steps and moves left three pixels per step. Its 16-pixel gap is sampled at spawn, top or bottom with equal probability, and becomes visible when the wall reaches $x \leq 40$. The gap location is hidden state before that reveal. Border or wall contact produces reward -1 and termination; the time cap is 128 steps.

Latent diagnostic. For each encoder, the diagnostic uses 64 real contexts with the wall still solid at $x \in \{42, 45\}$. Reconstructed simulator states generate recorded futures and matched wall-free futures under the recorded actions. The recorded branch survives the diagnostic horizon by construction. At each step, wall amplitude is the squared Euclidean distance between encodings of the wall-present and wall-absent frames. The teacher-forced residual uses real encoded history and is a total squared error, summed over latent coordinates. The wall coordinate of a prediction is its projection onto the wall-present minus wall-absent latent difference, with endpoints 0 and 1. Decoded wall intensity averages the minimum colour channel on shared wall pixels, excluding either possible gap band. These measurements compare features within one encoder; raw distances across models are not an information metric.

H SHARED-POLICY COMPARISON: PROTOCOL

This is the protocol behind Section 5 and the behavioral rows of Section 4. It was fixed before any of its policy outcomes existed (`research/shared_decoded_cnn_protocol.md` and, for the two worlds added afterwards, `research/scorecard_protocol.md` in the repository).

Matrix. Six worlds: per-frame gray noise without wind, per-frame gray noise with wind, persistent gray noise with wind, the gate, and the gate with a per-frame or a persistent gray background. For each world, the grid JEPA and the flow model each train three policy seeds $(0, 1, 2)$ inside imagination; all three gate worlds additionally receive three decoded-real control policies per family. That is 54 policies, all of which finished their registered budget and are reported; two of them, policy seeds 1 and 2 of the flow model's imagination in the gate world with a persistent background, were stopped once by an infrastructure failure and rerun from scratch under the same protocol.

In the four worlds of the original campaign, the three policy seeds run on one model fit each: gray-world JEPA checkpoints are model seed 0, the black gate's JEPA is model seed 1 (preselected on prior development evidence that its decoded-real CNN could discover the strategy), and every flow checkpoint is the seed-0 fit. In the two gate worlds added for this version, the design is stronger: policy seed k runs on model fit k, so the three runs of each cell are three independent world-model fits as well as three policy initializations. Policy seeds alone are not independent world-model fits, and the table states which regime each row uses.

Policy and optimizer. The policy is the unmodified pixel actor-critic of the original study: a three-block Impala CNN on native 63×63 RGB, a GRU that also receives the previous action, a categorical actor and a 64-bin histogram critic; 1,702,980 parameters. Both families use the same recurrent PPO: AdamW at 2.5×10^{-4} with weight decay 0.01, four epochs, minibatch 128, 16-step truncated backpropagation, $\gamma = 0.99$, GAE $\lambda = 0.9$, value coefficient 0.5, entropy coefficient 0.01, clipping 0.2, global gradient clipping 1.0, batch 512 and horizon 48. The three non-gate worlds use 50 updates (1,228,800 positions, 2,400 optimizer steps); every gate policy, imagined or decoded-real, uses 150 updates (3,686,400 positions, 7,200 optimizer steps). The training script refuses any other budget unless run as an explicitly non-scientific two-update smoke.

Observations. Each policy sees only its own model's decoded RGB: decoded posterior encodings of real frames during burn-in and evaluation, decoded model predictions in imagination. No raw pixel reaches the actor after the encoder. Training starts are sampled uniformly from valid eight-frame archive histories with no reset in the history; burn-in processes the first seven decoded frames with their real incoming actions. Latents keep the native $4 \times 4 \times 32$ interface without resizing, whitening or amplification; decoder quantization is retained.

Task signals. Flow uses its native sampled dynamics at four function evaluations and its jointly fitted reward and termination heads. The JEPA uses its deterministic predictor, the existing 80,000-frame decoder, and reward and termination heads of the original architecture refitted on all 500,000 training transition labels with the original 6,000-update recipe. The refit never updates encoder, predictor or decoder. Equal label counts do not equalize supervision stage: the flow heads send full-strength gradients into the dynamics trunk for the first half of its training and 0.1-scaled gradients afterwards, while JEPA world-model fitting is reward-free.

Evaluation. 256 fresh reset episodes are admitted by surviving seven uniformly random actions; the admission never looks at the policy. The same accepted prefix identities and all environment and action keys are reused across families and policy seeds, so every contrast is paired. The actor then controls 121 steps under the 128-step cap. Survival is return above -0.5; walls passed counts wall exits before or on the terminal transition. Every policy is its final checkpoint. Native-RGB trajectories of the first three accepted episodes are stored for each policy regardless of outcome.

Validation before the runs. Independent CPU checks of the training and evaluation code; GPU codec parity between the imagination path and the evaluation path for every checkpoint (context versus single-step encoding, 256- and 512-frame batches, head-cache encoding); unchanged frozen-state hashes; exact gate-state reconstruction; shared initial-context and evaluation-packet finger-

prints. Two-update smokes per family and path supplied wall-clock estimates used only for scheduling.

For the two gate worlds added in this version, imagination needs the simulator state behind each archive context, and the archives store position, wind, wall position and gap side but not the per-episode background texture. A sidecar replays the collector from the recorded seed and emits the texture key, and is accepted only when the replayed frames, actions, positions, wall positions and gap sides equal the archive's exactly. The reconstructed state is then checked against the archive frame by frame on $1{,}024$ sampled transitions per world before any policy runs: pixel-exact in the black and persistent-background worlds, and structurally exact on the drawn pixels where the background is redrawn each frame. All three gate worlds passed on every sampled transition.

State-blind reference rows. Two rules that never read an observation are run through the same paired evaluator on the same packets: uniform random actions, and strict left–right alternation, which is the trivial survival strategy of the still worlds (Table 7). They calibrate the survival numbers of Table 2. Alternation survives the per-frame still world in every episode; both rules survive no episode of any wind world or of any gate world. A trained policy whose control length falls below these values is steering into the boundary faster than chance. Because a state-blind rule ignores the observation and the three gate worlds share reset and action keys, their blind rows are identical by construction; only the background differs between them.

Table 7: State-blind rules on the shared evaluation packets: survival, control length and, for the gate worlds, walls passed, over the same 256 paired episodes as Table 2. Both rules are run through the identical evaluator on the identical episode keys; no episode or seed selection applies.

world	rule	survival	length	walls
per-frame noise, still	uniform random	0.24	74.0	–
per-frame noise, still	left–right alternation	1.00	121.0	–
per-frame noise, wind	uniform random	0.00	17.4	–
per-frame noise, wind	left–right alternation	0.00	18.2	–
persistent noise, wind	uniform random	0.00	17.4	–
persistent noise, wind	left–right alternation	0.00	18.2	–
gate	uniform random	0.00	24.0	0.07
gate	left–right alternation	0.00	24.6	0.05
gate + per-frame noise	uniform random	0.00	24.0	0.07
gate + per-frame noise	left–right alternation	0.00	24.6	0.05
gate + persistent noise	uniform random	0.00	24.0	0.07
gate + persistent noise	left–right alternation	0.00	24.6	0.05

Imagined termination probe. To separate what the policy learns from what its training environment shows it, each frozen model is rolled forward for 48 steps from 256 uniformly sampled real eight-frame contexts under the recorded actions (the data policy is uniform random), and its own termination head is read on two inputs: the model's predicted latent, exactly as in imagination, and the encoder latent of the real frame at the same step. The recorded termination labels give the ground truth. Nothing is trained and the adapters are byte-identical to those of the campaign. Figure 3 plots the implied probability that an imagined episode is still alive after k steps against the recorded survival. Per-step outputs, including latent distances between predicted and encoder latents, are stored with the campaign results.

I DECODER-FREE INTERFACES ON THE NOISY GATE WORLDS

Section 5 reads both models through a decoder. The two interfaces here read the frozen latent directly, so that no decoder stands between a model and the policy or planner. Both use the same three model fits per world as Table 2, policy seed k on fit k, and every number is a final update; no seed, checkpoint or episode is selected.

Latent-observation PPO. The policy replaces the pixel CNN by a two-layer perceptron on the 512-dimensional latent, keeps the GRU, the previous-action input, the categorical actor and the histogram critic, and is trained by the same recurrent PPO with the same settings and the same 150-update budget of 512 trajectories of 48 steps inside each frozen model, half of the starts from resets and half from recorded eight-frame contexts, as in the historical protocol of Appendix F. The JEPA supplies its deterministic predictor and refitted heads; the flow model its sampled dynamics at four function evaluations and its jointly fitted heads. Evaluation is 256 fresh episodes from reset under the 128-step cap with lengths counted from the reset and no admission prefix, so the lengths are not those of Table 2; the two state-blind rules are run through the same evaluator on the same keys and printed in the table. Real-transition controls through the same latent interface, one policy per fit, are listed where they completed.

Planners on the JEPA. Three planners run on the frozen JEPA in the closed-loop evaluator of the historical gate study, 128 environments for 128 steps from a uniformly random eight-step warm-up, lengths counted from the reset, the cap not counted as a death. *Goal-latent search* is the LeWorld-Model planner: every five-action sequence is rolled through the predictor and the one whose final latent is closest to the encoding of a goal frame, the cube at the frame's centre with no wall, is executed one step. *Head-scored search* rolls the same sequences and scores them by the refitted reward and termination heads with a bootstrapped value at the horizon; *one step* does the same at horizon one. Random and a fixed centre-holding rule are run through the same evaluator.

Table 8: **Decoder-free interfaces on the two noisy gate worlds.** Survival to the cap, mean episode length from reset and walls passed per episode, per seed (0 / 1 / 2), policy seed k on model fit k; planners are deterministic given the fit and the evaluator's keys. PPO rows use the 256-episode evaluator of `studies/stoch/rl.py`, planner rows the 128-environment evaluator of `studies/stoch/control.py`; the two blind rows are from the PPO evaluator, and the random-action row among the planners is from the planner evaluator.

World	Interface / training	Survival	Length	Walls
gate + per-frame noise	blind, uniform random	0.00	30.4	0.06
gate + per-frame noise	blind, left–right alternation	0.00	32.4	0.02
gate + per-frame noise	real transitions, latent, JEPA	0.031 / 0.000 / 0.000	59.5 / 39.2 / 39.0	0.91 / 0.19 / 0.06
gate + per-frame noise	real transitions, latent, flow	0.008 / 0.000 / 0.016	46.4 / 33.4 / 49.1	0.61 / 0.25 / 0.80
gate + per-frame noise	imagined, latent, JEPA	0.000 / 0.000 / 0.000	40.6 / 18.7 / 29.4	0.03 / 0.00 / 0.00
gate + per-frame noise	imagined, latent, flow	0.000 / 0.000 / 0.000	24.5 / 21.9 / 33.3	0.09 / 0.00 / 0.03
gate + per-frame noise	planner on the JEPA, uniform random	0.000 / 0.000 / 0.000	32.2 / 29.2 / 30.0	0.09 / 0.02 / 0.06
gate + per-frame noise	planner on the JEPA, goal-latent search, horizon 5	0.000 / 0.000 / 0.000	28.0 / 19.5 / 28.2	0.02 / 0.00 / 0.02
gate + per-frame noise	planner on the JEPA, head-scored search, horizon 5	0.016 / 0.000 / 0.000	48.7 / 35.3 / 37.1	0.55 / 0.06 / 0.06
gate + per-frame noise	planner on the JEPA, head-scored one step	0.008 / 0.000 / 0.000	52.9 / 33.8 / 33.5	0.81 / 0.03 / 0.05
gate + persistent noise	blind, uniform random	0.00	30.4	0.06
gate + persistent noise	blind, left–right alternation	0.00	32.4	0.02
gate + persistent noise	real transitions, latent, JEPA	0.000 / 0.000 / 0.000	33.0 / 34.6 / 32.5	0.04 / 0.06 / 0.07
gate + persistent noise	real transitions, latent, flow	0.016 / 0.004 / 0.000	43.3 / 39.4 / 35.6	0.54 / 0.48 / 0.29
gate + persistent noise	imagined, latent, JEPA	0.000 / 0.000 / 0.000	9.4 / 11.0 / 12.1	0.00 / 0.00 / 0.00
gate + persistent noise	imagined, latent, flow	0.000 / 0.000 / 0.000	25.0 / 30.7 / 31.1	0.07 / 0.05 / 0.07
gate + persistent noise	planner on the JEPA, uniform random	0.000 / 0.000 / 0.000	32.2 / 29.2 / 30.0	0.09 / 0.02 / 0.06
gate + persistent noise	planner on the JEPA, goal-latent search, horizon 5	0.000 / 0.000 / 0.000	30.5 / 29.6 / 30.3	0.02 / 0.00 / 0.01
gate + persistent noise	planner on the JEPA, head-scored search, horizon 5	0.000 / 0.000 / 0.000	26.7 / 29.6 / 26.5	0.02 / 0.11 / 0.02
gate + persistent noise	planner on the JEPA, head-scored one step	0.000 / 0.000 / 0.000	23.5 / 25.8 / 23.6	0.02 / 0.04 / 0.01

Reading. The latent interface does not qualify. Trained on *real* transitions through the frozen latent, the perceptron policy survives at most 0.03 of episodes through the JEPA's latent and at most 0.02 through the flow's, in either world, where the same PPO through the decoders reaches 0.49 to 0.98 (Table 5); on the black gate, in the earlier single-seed protocol, the same interface reached 0.98 inside the flow model, so it is the noisy backgrounds that defeat it at this budget. The twelve imagined-latent policies all end at 0.000 survival with 0.71 to 0.99 of their imagined rollouts alive, but with the control failing they say nothing about imagination that the decoded rows do not already say, and the paper's interface claims rest on the decoder alone. What the runs do establish is negative: the flow model's decoded-interface success in the per-frame gate world is not reproduced through its latent by this policy, so the repair Section 5 credits to sampling is a repair *through the decoder*. Among the planners, the head-scored ones pass walls, up to 0.81 per episode in one fit, and extend control by ten steps on average under a per-frame background, which shows that the refitted heads carry the wall's position; the goal-latent planner, which reads only the predictor, is at or below

random in both worlds, as it was on the black gate (32.0 steps against 32.2, historical protocol). A policy or planner that reads the grid latent with a convolutional torso, or an amplified wall direction, was not run.

J THE REVERSAL ACROSS REGULARIZER WEIGHTS AND BUDGETS

The reversal of Section 4 is measured at one regularizer weight, $\lambda = 0.09$, and one budget, 8,500 updates. Eight further grid-JEPA fits, all seed 0, vary one of the two: the SIGReg weight at 0.03, 0.27 and 0.9 in both wind worlds, and the budget at 17,000 and 25,500 updates, two and three times the registered one, in the per-frame wind world. Everything else is the campaign recipe, and each fit is scored by the same cell evaluation as Table 1, both probes, the same 512 windows.

Table 9: **Per-frame noise plus wind against persistent noise plus wind, one fit each, as the regularizer weight and the budget change.** ENC and PRED as in Table 1, the better of the two probes; the registered cell's seed-0 values are repeated for reference. Every fit that was run appears.

World	Fit	ENC	PRED	pred. loss	SIGReg
per-frame noise, wind	$\lambda = 0.09$, 8,500 updates (registered)	17.54	19.70	0.7345	2.29
per-frame noise, wind	$\lambda = 0.03$	0.14	9.13	0.0343	2.16
per-frame noise, wind	$\lambda = 0.27$	17.71	19.11	0.8821	1.24
per-frame noise, wind	$\lambda = 0.9$	17.78	20.13	0.9504	1.05
per-frame noise, wind	17,000 updates	17.56	20.77	0.7355	2.28
per-frame noise, wind	25,500 updates	17.62	23.11	0.7343	2.26
persistent noise, wind	$\lambda = 0.09$, 8,500 updates (registered)	0.72	11.55	0.0031	1.07
persistent noise, wind	$\lambda = 0.03$	0.42	10.15	0.0013	1.16
persistent noise, wind	$\lambda = 0.27$	8.70	16.20	0.0099	0.96
persistent noise, wind	$\lambda = 0.9$	18.14	22.02	0.0180	1.04

Budget. Two and three times the registered budget change nothing in the failing cell: ENC 17.56 and 17.62 pixels, prediction loss 0.7355 and 0.7343 against 0.7345 at 8,500 updates, the same latent to three digits. The fit is in a stable basin, not on its way somewhere else.

Regularizer weight. The weight moves the odds. At $\lambda = 0.03$ the per-frame wind world keeps the cube in seed 0, ENC 0.14 pixels with prediction loss 0.034, but not in two further seeds, run after the rest of this study (ENC 17.7 pixels in both, loss 0.48). In seed 0 it is once more the *easier* background, as in the still worlds, with the persistent sibling at 0.42. At 0.09 the per-frame cell fails and the persistent one keeps the cube, which is the reversal of Table 1. At 0.27 the persistent cell degrades to 8.70 pixels, and at 0.9 both are at chance. The two backgrounds fail at different weights and with different loss signatures: every failing per-frame fit has a prediction loss between 0.73 and 0.95 per coordinate, a latent driven by the unpredictable noise, while the failing persistent fit at $\lambda = 0.9$ has a loss of 0.018, a nearly predictable latent that carries the episode's texture instead of the cube, which is the cheap non-collapsed optimum of Proposition 1 in all but the exactly-Gaussian clause.

What the objective prefers. The final training values of the two loss terms let each fit be scored under the other's weight. Under the registered $\lambda = 0.09$ the failing seed-0 fit scores $0.734 + 0.09 \times 2.29 = 0.94$, while the cube-keeping $\lambda = 0.03$ fit scores $0.034 + 0.09 \times 2.16 = 0.23$ under the same weight; under $\lambda = 0.27$ the comparison is 1.22 against 0.62; only at $\lambda = 0.9$ does the failing fit score better, 1.90 against 1.98, a gap inside the batch noise of the SIGReg statistic. So at the registered weight, and at three times it, the objective prefers a latent that keeps the cube by a factor of two to four, and every fit we trained at those weights, three seeds and three budgets, lands in the other basin. The failing cell is where optimization goes, not what the objective asks for, which is the part of the phenomenon Section A declines to characterize. One fit per variant is a check of direction, not a spread; the registered cell's three fits differ by 0.08 pixels.

K PREDICTOR SWAP

The swap of Section 4 keeps every part of the flow recipe except its tokenizer. The frozen grid-JEPA encoder of a gate fit, the tokenizer's encoder trunk trained with the JEPA objective instead of reconstruction (it differs only in reading one frame instead of two and in ending with LeWorldModel's projector), replaces the tokenizer's encoder, its 512-dimensional latent read as the $4 \times 4 \times 32$ grid the flow predictor expects, and the JEPA's own pixel decoder, fitted after training, replaces the tokenizer's decoder. The flow predictor is then trained exactly as in the flow recipe: the same $16{,}000$ updates, per-coordinate latent standardization computed on the training archive, and reward and termination heads trained jointly. No tokenizer is trained, so the swap costs about 400 seconds of training. Both the swap and the JEPA's own predictor are scored by the gate evaluation of Table 4, run by the same script on the same contexts: the decoded shared-wall intensity on 256 contexts before the opening is revealed, and the opening-side AUROC averaged over the eight predicted steps after it. The script is `studies/scorecard/swap.py`.

Table 10: **Predictor swap on the frozen grid-JEPA encoder.** Wall intensity at predicted steps 1 and 8 (encoded = real frames) and the mean opening-side AUROC over the eight predicted steps, for the JEPA's own predictor and for the flow predictor trained on the same frozen encoder. One row per encoder seed; every fit that was run appears.

			JEPA predictor		flow predictor (swap)	
world	seed	encoded	wall 1 / 8	AUROC	wall 1 / 8	AUROC
black gate	0	1.00	0.00 / 0.00	0.50	0.00 / 0.00	0.50
black gate	1	0.99	0.00 / 0.01	0.50	0.00 / 0.00	0.50
black gate	2	0.99	0.53 / 0.00	0.56	0.50 / 0.01	0.51
gate + per-frame noise	0	1.00	0.94 / 0.14	0.64	1.00 / 0.99	0.99
gate + per-frame noise	1	1.00	1.00 / 1.00	0.95	1.00 / 0.99	1.00
gate + per-frame noise	2	1.00	0.78 / 0.50	0.50	0.95 / 0.57	0.53

L POLICY BUDGET: EVERY IMAGINED CELL AT THREE TIMES THE REGISTERED UPDATES

Table 2 reports each world's registered budget, fixed before any policy ran: 150 updates in the gate worlds, 50 in the cube worlds. Two of the flow arm's cube cells had a wide seed spread at that budget, which is exactly the shape a policy-optimization artifact takes, so before reading anything into them we reran *every* imagined cube cell, both families and all three seeds, at 150 updates. Nothing else changed: same checkpoints, same head refits, same archive, same start protocol, same evaluation packet, same optimizer, same evaluator, and the extended budget is recorded in each run's provenance (`--budget extended`).

Table 11: **The same imagined cells at the registered and the extended policy budget.** Final-checkpoint survival in per cent on each world's fixed 256-episode packet, mean $\pm$ sample standard deviation over the three policy seeds, with the per-seed values beside it. Only the number of PPO updates differs between the columns; every run is reported and no seed or checkpoint is selected.

world	recipe	50 updates	150 updates	seeds (50)	seeds (150)
per-frame noise, wind	grid JEPA	0 ± 0	0 ± 0	0.000, 0.000, 0.000	0.000, 0.000, 0.000
per-frame noise, wind	flow	58 ± 41	90 ± 1	0.945, 0.129, 0.652	0.891, 0.918, 0.895
persistent noise, wind	grid JEPA	2 ± 1	18 ± 29	0.027, 0.020, 0.000	0.512, 0.000, 0.016
persistent noise, wind	flow	41 ± 42	65 ± 5	0.891, 0.207, 0.137	0.703, 0.602, 0.633

Three readings, in order of what they cost us.

First, the flow arm's weak cells were the budget. With a per-frame background its three seeds go from 0.945, 0.129, 0.652 to 0.891, 0.918, 0.895, and with a persistent background from 0.891, 0.207, 0.137 to 0.703, 0.602, 0.633: in both worlds the spread collapses and the low seeds were

unfinished optimization, not a model that cannot teach. The persistent-noise cell is still below the colour threshold at the longer budget, mean 0.65, and it stays **amber**.

Second, the JEPA's per-frame-noise cell is not a budget artifact. It is 0.000 in all three seeds at 50 updates and 0.000 in all three at 150. That is the world where the encoder lost the cube (Section 4), and no amount of policy optimization inside such a model recovers it.

Third, the JEPA's persistent-noise cell does move: 0.027, 0.020, 0.000 becomes 0.512, 0.000, 0.016. One seed of three reaches half the episodes. We report this because it qualifies the sentence it bears on: in the world where the cube survives encoding, the JEPA's imagination is not uniformly unable to teach the task, it is unreliable, and one fit in three teaches a policy that survives half the time given three times the budget. The ordering against the flow reference is the same at either budget, mean 0.18 against 0.65 and worst seed 0.000 against 0.602.

What this changes about how to read the paper: the cube worlds' imagined cells are budget-sensitive in both families, so the matched rows of Table 2 should be read as what one fixed budget buys, not as a ceiling. The gate worlds, which carry the paper's claims, are registered at the longer budget for both families and for both training sources, and their cells are not close to a threshold: the JEPA's imagination is at 0.00 to 0.03 and the flow's decoded-real controls and black-gate imagination are above 0.88.

M One repair attempt: a Muesli-inspired unroll auxiliary over the frozen models

One cell of Table 2 fails for neither of the paper's two reasons. In the gate world with a persistent background both families' representations keep the wall, both rollouts render it, the decoded-real controls pass on the same checkpoints, and yet both imaginations fail. In that cell the flow arm's policies *improve inside the dream* — the last update of the three runs has 0.25, 0.57 and 0.71 of imagined rollouts alive — and transfer nothing, 0.000, 0.039 and 0.113. That signature is model exploitation rather than lost state, so it is the one place where a change to the *algorithm* rather than to the model deserves a measurement, and we ran exactly one such change.

What was added. The design note was written before the runs and fixes everything below. The world model stays frozen: no gradient from either new term reaches an encoder, predictor, decoder or head, and nothing is trained on imagined data except the policy and its critic. The policy keeps the campaign's CNN + GRU, the same parameter count and the same optimizer. Two terms are added to the campaign's PPO loss, both read off the frozen model's own lookahead, in the spirit of Muesli's unroll auxiliary and its regularized policy improvement (Hessel et al., 2021):

1. a K-step value target along the imagined on-policy trajectory, $y_t^{(K)} = \sum_{j<K} \gamma^j c_{t,j} \hat{r}_{t+j} + \gamma^K c_{t,K} V_{\bar{\theta}}(s_{t+K})$ with $c_{t,j}$ the model's own continuation product and $\bar{\theta}$ the rollout-time critic, regressed with the campaign's HL-Gauss loss at weight 0.5; and
2. a clipped advantage-weighted policy target. For every action a — including the ones the policy did not take — the frozen model is stepped once under a, continued $K-1$ steps under the policy's greedy action and bootstrapped, giving $\hat{q}_K(s_t, a)$; with $\hat{A} = \hat{q}_K - V_{\bar{\theta}}$ and β the standard deviation of $\hat{A}$ over the imagined step, the target is $\pi_{\mathrm{CMPO}}(a \mid s_t) \propto \pi_{\bar{\theta}}(a \mid s_t) \exp(\mathrm{clip}(\hat{A}/\beta, -1, 1))$ and $\mathrm{KL}(\pi_{\mathrm{CMPO}} \,\|\, \pi_\theta)$ enters at weight 0.5 on every eighth imagined row.

The clip is what makes this safe on a wrong model: each unnormalized weight lies in $[e^{-1}, e]$, so after normalization an advantage the model invents can move the target policy's likelihood ratio $\pi_{\mathrm{CMPO}}(a \mid s)/\pi_{\bar{\theta}}(a \mid s) = e^{\mathrm{clip}(\hat{A}_a/\beta, -1, 1)}/Z$ by at most a factor e^2 in either direction, whatever it claims, since the normalizer Z is itself in $[e^{-1}, e]$. $K = 3$ was chosen before the runs from $K \in \{2, 3, 4\}$, both families get the intervention in the same world with the same K and the same three seeds, and the matched rows of Table 2 are untouched.

That the rest is unchanged is measured, not asserted. The campaign's collector and the auxiliary collector were run on the same model, the same contexts and the same key, and every field the

campaign's PPO reads was compared. Bit-identity is not available in this stack for a reason the campaign's own code records — bfloat16 kernels change with the batch shape, so adding one unrelated codec call to a program already moves the JEPA's decoded context by up to 9 levels of 255 — so the check measures that floor directly, by recompiling the campaign's collector and by putting it in a program that holds one extra codec call. At the first imagined step the flow collector deviates by exactly zero on every field, and the JEPA collector's deviations equal the extra-call floor to every printed digit, with both models' own reward and termination diagnostics inside one bfloat16 unit in the last place.

Table 12: **The gate + persistent-noise imagination cells with and without the unroll auxiliary.** Final-checkpoint survival in per cent on the world's fixed 256-episode packet, three policy seeds, policy seed k on model fit k, with the per-seed values; "imagined alive" is the mean fraction of the model's own rollouts still alive at the last update, which is what the policy was paid. Only the two auxiliary loss terms differ between a recipe's two rows. The verdict column applies the note's rule, fixed before the runs: improved if the three-seed mean gains at least 0.10 with no seed worse than its own matched counterpart by more than 0.05.

recipe	imagination	survival	per seed	imagined alive	verdict
grid JEPA	matched PPO	0±0	0.000, 0.000, 0.000	0.40	
grid JEPA	+ unroll aux (K=3)	0±0	0.000, 0.000, 0.000	0.47	unchanged
flow	matched PPO	5±6	0.000, 0.039, 0.113	0.51	
flow	+ unroll aux (K=3)	3±4	0.000, 0.016, 0.070	0.54	unchanged

It does not work, and the way it fails is informative. Neither family improves. The grid JEPA stays at 0.000 in all three seeds, and the flow arm moves from 0.000, 0.039, 0.113 to 0.000, 0.016, 0.070 — slightly worse, well inside the spread of a cell whose three matched seeds already differ by a factor of three. By the rule fixed before the runs both rows read *unchanged*, so no sentence about the intervention enters the main text and $K = 2$ and $K = 4$ were not run: the note conditioned them on $K = 3$ moving a cell.

What did change is the wrong quantity. Both families end *more* successful inside the model they were trained in: the mean fraction of imagined rollouts still alive at the last update rises from 0.40 to 0.47 for the JEPA and from 0.51 to 0.54 for the flow model, while real survival does not follow. The added terms were live, not silently zero — the K-step value loss runs at 1.2–2.1, comparable to the PPO critic's own, and the policy term's mean KL per update is $2.2\text{–}2.6 \times 10^{-2}$ for the JEPA and 5.9×10^{-3}–2.8×10^{-2} for the flow model. A lookahead through a frozen model can only sharpen what that model already reports, and in this cell the report itself is what is wrong: the deficit is in what the imagination pays, so making the policy better at collecting it moves survival no closer. We also checked the cheapest alternative explanation, that the imagined policies simply collapse to a single action and no target could move them: only one matched seed and one auxiliary seed end below a policy entropy of 0.05, and the two matched seeds that end at 0.94 and 0.89 fail just as completely (`output/scorecard/aux_entropy.json`).

N TERMINAL REWARDS IN THE FLOW RECIPE

The original recipe masks transitions entering reset frames out of reward training. Here the only nonzero reward occurs on terminal transitions, so that mask removes the reward signal. The corrected head predicts the next reward from the current frame summary and action with terminal transitions retained in its loss. The reported flow-policy outcomes use this corrected convention. The single-checkpoint reference therefore denotes the complete tokenizer, dynamics, reward and termination package, rather than an isolated change in transition stochasticity.

O CRAFTAX INVENTORY COMPARISON: PROTOCOL

The protocol behind the Craftax paragraph of Section 5; fixed before the policies were trained (`research/craftax_native_inventory_protocol.md` and `research/craftax_inventory_protocol.md`). Table 13 lists every policy.

Table 13: **Craftax-Classic, one pixel policy trained for five million imagined steps inside each model,** 256 **native episodes per policy, final checkpoints.** Wood and table: fraction of episodes with the achievement; pickaxe: episodes ending with a wooden pickaxe. "Imagined r": mean reward per imagined step paid by the model's own head at the last update. All three policy seeds are listed; no seed is selected, and the state-blind rows use the identical evaluator on the same episode keys.

Policy trained in	seed	wood	table	pickaxe	achievements	return	imagined r
flow imagination	0	100%	96%	242/256	6.04	5.14	0.06
flow imagination	1	100%	91%	15/256	3.97	3.03	0.03
flow imagination	2	100%	97%	242/256	5.87	4.94	0.06
JEPA imagination	0	22%	0%	0/256	0.27	-0.65	0.90
JEPA imagination	1	4%	0%	0/256	0.07	-0.86	0.96
JEPA imagination	2	2%	0%	0/256	0.02	-0.91	1.00
uniform random	–	43%	9%	1/256	2.34	1.28	–
no-op	–	0%	0%	0/256	0.00	-0.92	–

Environment. Craftax-Classic-Pixels version 1.5.0 with unchanged defaults: 17 actions, 64×64 procedurally generated maps, a 10,000-step native episode limit, 22 achievements. The native reward is the change in the achievement count plus 0.1 times the change in health; the evaluator checks the telescoping identity between summed rewards and terminal achievements and health on every episode. Termination occurs at the step limit, on lava, or at non-positive health. The evaluator uses the no-auto-reset environment, stops each episode at its first native termination and freezes its metrics.

World models. Both models are trained on the same 500,000-transition archive of uniformly random play from 250 parallel streams under the same ten-minute budget with the same $4 \times 4 \times 32$ latent shape. The JEPA is the seed-0 grid fit of Table 1, with one-step factored predictor and eight-frame history; its pixel decoder and its reward and termination heads are refitted afterwards on the frozen latent with a fixed recipe (decoder: 8,000 steps, batch 256; heads: 6,000 steps, batch 1,024, hidden 512) whose arguments and resulting hashes are checked at load time. The flow model is the native recipe with jointly fitted heads, trained separately for this comparison under the same recipe and budget; it is not one of the three Craftax flow fits of Table 1, and the two checkpoints' hashes are recorded beside the runs. The frozen model weights of the JEPA are hash-verified to be byte-identical before and after head fitting.

Policy training in imagination. The policy is the original Craftax pixel actor-critic, a non-recurrent CNN of $6{,}092{,}177$ parameters with the original Craftax pixel-PPO settings; it is not the $1{,}702{,}980$-parameter CNN + GRU of the cube and gate worlds. It is trained for a nominal five million imagined steps (4,998,400 executed) inside each model from a shared bank of genuine eight-frame episode-start contexts. The JEPA adapter reads reward and termination from (z_t, a_t), predicts once per action and samples termination from the head's probability; the flow adapter runs its native expanding-window rollout with four function evaluations. Imagined episodes are capped at 1,024 steps. No behavior cloning, action masking, reward alteration, resource injection or checkpoint selection is used; a non-finite loss fails the run rather than being skipped. Three policy seeds per family, one model seed.

Evaluation. Each final policy is evaluated on 256 native episodes with evaluation seed 123. Environment keys derive from episode identity and step only, action-sampling keys from a separate stream; the construction is stored with the results. The policy observes native RGB and its previous action, whereas during training it observed its model's decoded RGB. Unlike the gate worlds, this comparison has no decoded-real control and no evaluation on decoded frames of the native game, so an interface mismatch between decoded and native pixels is not separated from an imagination failure by any run; the imagined-reward column and the in-model achievement logs are the quantities that do not depend on the evaluation channel. Building that matched evaluator was outside the scope of this revision and is listed as unavailable. Primary endpoint: the fraction of episodes in which the wooden-pickaxe achievement is true at the end of the episode. Retained: all 22 achievement rates with Wilson intervals, achievement count, native return, executed steps, survival, termination causes,

first-acquisition steps, final inventory, and the terminal native score where a native termination occurred. Uncertainty intervals are across evaluation episodes for one fixed policy and do not estimate variation across model fits.

Reference rows. Two state-blind policies are run through the identical evaluator, keys and aggregation: uniform random actions and the no-op action. Uniform random play reaches one wooden pickaxe in 256 episodes, collects wood in 43% of episodes, places a table in 9% and averages 2.34 achievements per episode; the no-op policy earns no achievement and a return of -0.92 from health loss. A trained policy below the random row has learned something that hurts.

Imagined reward. The training logs record the mean reward per imagined step paid by each model's reward head. In the real game uniformly random play collects about 0.01 per step and the best policy of this campaign 0.03 (return over executed steps). Both readings are reported in Table 13; a policy whose imagined reward exceeds what any real policy collects is exploiting its reward head rather than the game.

P Earlier unmatched gate experiments

The experiments in this appendix predate the matched shared-policy protocol of Appendix H: real and imagined policies did not share evaluation keys, latent-MLP and decoded-CNN policies are mixed, and the summary table (Table 14) is regenerated from the last evaluation of each run. They are reported because they cover interventions the matched campaign does not, in particular sampling from the JEPA's predictor. Of 27 policies trained inside the JEPA's imagination, none discovered the gate strategy, including nine that used a sampling head fitted on the predictor's residual and nine that averaged eight of its samples. Those runs share the fitted heads and the PPO protocol of that era, so they test this particular intervention rather than distributional prediction in general; the licensed statement in the main text is only that the deficit is in the rollout.

Reference model. For the gate, a reconstruction-trained tokenizer and causal flow-matching predictor provide a feasibility reference (Lipman et al., 2023; Chen et al., 2024; Hafner et al., 2025). It uses the same latent shape, data, and nominal model-training budget, with a 32-frame context and jointly fitted reward and termination heads. Architecture, supervision, and history length differ from the JEPA. One dynamics checkpoint is evaluated, so its policy runs show variation conditional on that fit rather than variation across fitted flow models.

P.1 Useful wall information remains in the frozen representation

To test whether the wall information visible in decoded real encodings can support behavior, we train a convolutional policy in the real gate on the frozen decoder's output. Two of nine runs discover the survival strategy, both using encoder seed 1; the corresponding real-pixel CNN discovers it in two of three runs. This establishes useful information in at least one fitted encoder under the decoder-and-CNN read-out, and it is the reason the matched campaign qualifies every interface on real transitions before reading an imagination result. The decoder adds reconstruction supervision outside the world-model budget, so this is a test of recoverable information rather than unsupervised control sufficiency.

The policy interface matters. With an MLP torso, real pixels yield 0.91 walls per episode and frozen latents yield 0.94, with no discoveries in either group. An MLP on decoded latents similarly yields 0.92 walls. Thus failure of this MLP training recipe is not evidence that the encoder has lost the obstacle. These comparisons do not set a capacity limit for all MLP policies; they locate a limitation of the tested read-out and optimization procedure, which is why the matched campaign fixes one stock convolutional policy for every arm.

P.2 The tested imagination pipeline does not discover the strategy

In the earlier unmatched protocols, all three JEPA predictor variants produce poor downstream control (Table 14). The original predictor yields 0.27 walls per episode; one residual sample and the mean of eight each yield 0.26. No policy among these 27 runs discovers the strategy; the highest

Table 14: **Final gate outcomes of the earlier unmatched campaign, with predictor variants kept separate.** Mean ± sample standard deviation across policy runs; discovery is final survival above 50%. JEPA imagination variants and real-latent conditions each contain 3 models × 3 policies; pixel controls have three policies. Every run enters at its last evaluation, with no selection over seeds or checkpoints. Real training changes rewards, terminations, and PPO protocol alongside dynamics. Flow is a single-model reference.

Training / observation	Predictor or torso	Walls	Discovery
Imagined JEPA	Original	0.27 ± 0.22	$0/9$
	Residual sample	0.26 ± 0.32	$0/9$
	Residual mean of 8	0.26 ± 0.30	$0/9$
Real / frozen JEPA	MLP	0.94 ± 0.04	$0/9$
Real / pixels	MLP	0.91 ± 0.08	$0/3$
Real / decoded JEPA	CNN	1.57 ± 1.26	$2/9$
Real / pixels	CNN	2.91 ± 1.65	$2/3$
Imagined flow	Sample	1.45 ± 1.23	$1/6$
	Mean of 8	0.88 ± 0.14	$0/3$

final survival is 3.1%. Real-transition PPO on the same frozen encoders yields 0.94 walls. The difference measures the combined effect of learned transitions, fitted heads, and the changed PPO protocol. Even the stronger real-latent MLP result falls far short of reliable survival.

The flow reference supports feasibility at this task and nominal model budget: one of its nine policies discovers the strategy. The remaining eight pass 0.72–1.18 walls per episode. Since all nine use one dynamics fit with a different architecture and supervision, this reference does not establish a model-family ranking. Training the gate JEPA for twice and four times the steps also leaves imagined-policy survival at 0–3% in the tested extensions (encoder seed 0, policy seed 0, each predictor variant).

P.3 Prediction and fitted heads provide distinct failure signals

The same decoder that renders real encoded walls removes them from original predicted rollouts: across all three encoder seeds, decoded shared-wall intensity is at most 0.02 by the second predicted step. The shared wall pixels occur in both possible gap futures, so a pixelwise mixture of those futures would retain them. This observation rules out that simple pixel-space picture; it does not determine what averaging or distribution shift does in the learned latent space.

A within-encoder counterfactual diagnostic makes the scale of the feature explicit. On 64 pre-reveal contexts for seed 0, the first-step squared distance between wall-present and wall-absent encodings is 0.035, while total teacher-forced prediction error is 0.145; the same ordering holds for all three seeds. This comparison concerns sensitivity to wall removal versus error in all features. It neither identifies a causal mechanism nor permits comparing information content through raw distances across independently fitted latent spaces.

The fitted heads expose another limitation. For encoder 0, termination AUC is 0.96 for border deaths but 0.65 for wall deaths. Predicted rewards for the two death causes are -0.89 and -0.05, respectively, against the true -1. An imagined policy therefore trains on a weak wall-collision signal. The residual-sampling variants retain these heads and the policy protocol, so their poor control outcomes test this particular intervention rather than sampling in general.