GRANULARITY IS A DECISION: JOINT HORIZON-DESCRIPTION SELECTION IN WORLD MODELS

Anonymous authors
Paper under double-blind review

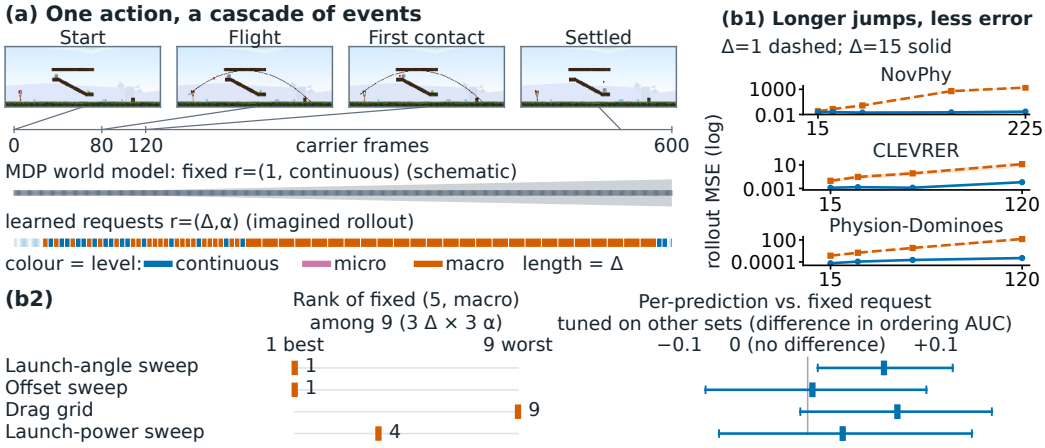


Figure 1: Prediction granularity can change at every query. (a) One real NovPhy launch. The MDP track is schematic; the lower track shows the requests $r = (\Delta, \alpha)$ our learned policy chose while imagining this launch (decision-only, not closed-loop). (b1) Rollout error of a continuous-only model stepping by $\Delta = 1$ versus 15 (three-seed mean and range). (b2) We plan by imagining each candidate launch and picking the lowest predicted cost; ordering AUC is the probability that a launch that succeeds in the engine is predicted to cost less than one that fails. Left: rank of one fixed request among all nine ($\Delta \in \{1, 5, 15\} \times \alpha \in \{\text{continuous}, \text{micro}, \text{macro}\}$).

ABSTRACT

Embodied agents increasingly plan with world models, rolling candidate actions forward and executing the one whose predicted outcome best reaches the goal. Such models are typically trained as Markov decision process (MDP) transitions $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, binding every prediction to one environment step and one state representation. However, many physical interactions are action-sparse with persistent effects: a single throw sends an object through flight, collision, and settling, a cascade that unfolds without further action yet decides success. Step-wise prediction of such cascades compounds error, coarse horizons skip decisive contacts, and full detail wastes capacity. All three costs stem from one fixed choice of granularity, made once at design time. Thus, we propose making granularity a variable that the world model chooses per prediction. We formalize the idea as an MDP while every model query carries a prediction request $r = (\Delta, \alpha)$ that advances the imagined rollout Δ steps at description level α. Across three physics benchmarks, granularity proves consequential: predicting in long jumps cuts compounding rollout error by orders of magnitude, yet no fixed request suits every test set: the one that best separates successful from failed launches on two sets ranks worst on a third. That variance is exploitable: choosing the request per prediction beats a fixed request tuned on other test sets, and choosing both axes matches a horizon-only adaptive baseline at a third of its compute. These results establish granularity as a decision variable rather than a design constant: consequential, unsolvable by any fixed setting, and cheap to adapt.

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.

1 INTRODUCTION

Embodied agents increasingly plan with learned world models, imagining the outcomes of candidate actions and executing the one predicted to best reach the goal (Hafner et al., 2025; Hansen et al., 2022; Assran et al., 2025; Zhou et al., 2025). Many physical interactions, however, are action-sparse with persistent effects: after a single throw, push or release, objects keep moving under physical law, strike other objects and set off chain reactions that reshape the scene long after the action ends. Imagining the outcome of such an action amounts to simulating physical motion. In physical simulation, two choices have long been central and are fitted to the situation rather than fixed. One is time resolution: integrators control their step size (Hairer et al., 1993), and rigid-body simulators take smaller steps as they approach a collision (Mirtich, 2000). The other is model resolution: less important parts of a scene are simulated at a lower level of detail (Carlson & Hodgins, 1997), and learned simulators refine or coarsen their mesh as they run (Pfaff et al., 2021). Together these set a simulator's granularity, how far each step reaches and at what level of description it advances the state. A world model that imagines a physical cascade faces the same two choices.

Yet the world models used by embodied agents treat both choices as design constants. Formalized as a Markov decision process (MDP), they learn $s_{t+1} \sim P(\cdot \mid s_t, a_t)$: every query advances one fixed step in one fixed representation, whether latent states (Hafner et al., 2025; Hansen et al., 2022), visual features (Assran et al., 2025; Zhou et al., 2025) or video frames (Yang et al., 2024; Bruce et al., 2024; Cen et al., 2025). For action-sparse physical environments such as NovPhy (Pinto et al., 2024) and PHYRE (Bakhtin et al., 2019), this formalism fits poorly. The cascade becomes a long run of no-op actions predicted one step at a time; options, action persistence and delayed MDPs change when the agent acts rather than what the model predicts (Sutton et al., 1999; Metelli et al., 2020; Katsikopoulos & Engelbrecht, 2003); and PHYRE's own contextual-bandit framing abstracts the cascade into a single outcome (Bakhtin et al., 2019). Even work that does vary granularity has not made the two axes one choice. Hierarchical JEPA calls for prediction "at different time scales and different levels of abstraction" but ties each time scale to one abstraction level (LeCun, 2022); other models vary the horizon (Nhu et al., 2025; Gumbsch et al., 2024) or the description (Wang et al., 2023; Liang et al., 2025) one axis at a time, or select among fixed repertoires or separately trained models (Huang & Freed, 2026; Jang et al., 2026). To our knowledge, no world model chooses horizon and description jointly, per prediction, over one shared latent state (Section 2).

To close this gap, we propose to make granularity a variable that the world model chooses for every prediction. Specifically, our approach introduces three key innovations:

- **A per-prediction request within the MDP.** Because the agent issues no further action once a cascade starts, a jump of Δ steps needs no intermediate actions. The environment can therefore remain an ordinary MDP while every model query carries a *prediction request* $r = (\Delta, \alpha)$: how far to advance and at what level of description.
- **One world model for all requests.** A single JEPA-style latent predictor takes the request as input and advances one shared latent state under every request, so an imagined rollout can switch granularity at any step without translating between models.
- **A learned request policy.** The request for each prediction is chosen by a policy distilled from dynamic-programming labels that trade prediction error against compute, requiring neither reinforcement learning nor task-outcome labels.

Our experiments on NovPhy (Pinto et al., 2024), an Angry-Birds-style physics benchmark that, to our knowledge, has not been used as a world-model testbed, together with CLEVRER (Yi et al., 2020) and Physion-Dominoes (Bear et al., 2021), show that granularity is consequential: long jumps curb the compounding error of step-wise rollouts (Talvitie, 2014; Venkatraman et al., 2015; Janner et al., 2019) by orders of magnitude (Figure 1, b1). No fixed request suits every situation, as the request that best ranks candidate launches on two launch sets ranks worst on a third (Figure 1, b2). This variance is exploitable: choosing the request per prediction outperforms a fixed request tuned on other launch sets, and choosing both axes matches a horizon-only adaptive baseline at a third of its compute. These results suggest that, for agents that act once and then wait on physics, granularity is better treated as a decision variable than as a design constant.

2 Related Work

Adapting granularity. Horizon-adaptive world models jump to predictable or key frames (Jayaraman et al., 2019; Pertsch et al., 2020; Kim et al., 2019), condition on the time step (Nhu et al., 2025), or segment time by context change or surprise (Gumbsch et al., 2024; Iiyama et al., 2026), while what they predict stays fixed; VisualPredicator instead invents predicates over a fixed schedule (Liang et al., 2025). Hierarchical models fix which time scale goes with which level of abstraction: H-JEPA assigns short-term, low-level prediction to one level and longer-term, higher-level to the next (LeCun, 2022, §4.4, §4.6), and Clockwork VAE, MSPred and Director fix these scales in the architecture (Saxena et al., 2021; Villar-Corrales et al., 2022; Hafner et al., 2022). SALT learns both abstractions jointly, but once, as a fixed repertoire for skill planning (Huang & Freed, 2026). Others select per state, but a different variable: Wang et al. (2023) choose a model resolution at each MPC step over a fixed horizon, and MuSix routes each state to one world model in a mixture (Jang et al., 2026). Granularity thus changes by switching level, skill or model, whereas simulators adjust step and detail within one run (Mirtich, 2000; Carlson & Hodgins, 1997). To our knowledge, no world model chooses horizon and description jointly, per prediction, over one shared latent state.

Action-sparse physics. Physical-reasoning benchmarks pose one-shot interventions or passive dynamics (Bakhtin et al., 2019; Bear et al., 2021; Yi et al., 2020), and forward-prediction agents on PHYRE use object- or pixel-based models of fixed granularity (Girdhar et al., 2020). To our knowledge, NovPhy (Pinto et al., 2024) has not been used to evaluate a learned world model; Appendix D gives more detail.

3 Problem Formulation

An action-sparse persistent-effect environment is a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A} \cup \{\emptyset\}, P, R)$ over engine steps t, where $\emptyset$ is the null action, R is the task reward, and actions are sparse:

$$s_{t+1} \sim P(\cdot \mid s_t, a_t), \qquad a_t \neq \emptyset \text{ only if } t \in \mathcal{T}_{\mathrm{dec}}, \tag{1}$$

with $\mathcal{T}_{\mathrm{dec}}$ the small set of decision steps. In NovPhy an action is one slingshot launch, and the agent observes an RGB frame o_t. Because the agent issues no further action between decision steps, the state n steps ahead depends only on the current state and the action taken there, provided no decision step lies in $(t, t+n)$:

$$P^n(s_{t+n} \mid s_t, a_t) = \int P(s_{t+1} \mid s_t, a_t) \prod_{\tau=t+1}^{t+n-1} P(s_{\tau+1} \mid s_\tau, \emptyset)\, \mathrm{d}s_{t+1:t+n-1}. \tag{2}$$

A prediction may therefore jump any number of steps without intermediate actions, and $\mathcal{M}$ remains an ordinary MDP. Treating the launch as the only decision would instead fold the cascade into the outcome of (s, a), the contextual-bandit framing of PHYRE (Bakhtin et al., 2019); we keep the cascade and let the model decide how to predict it.

A frozen object-slot encoder E maps an observation to the *carrier* $z = E(o) \in \mathbb{R}^{236}$, a continuous latent state (Section 4). We index carrier frames by k, with z_k the carrier of the k-th captured frame after the decision step; on N1 captures one carrier frame spans 50 engine steps. Every world-model query carries a *prediction request*

$$r = (\Delta, \alpha) \in \mathcal{R} = \{1, 5, 15\} \times \{\mathsf{cont}, \mathsf{micro}, \mathsf{macro}\}, \qquad |\mathcal{R}| = 9. \tag{3}$$

The horizon Δ is the number of carrier frames the prediction advances. The description level α sets what the prediction reasons over. Under cont (continuous) it is the carrier alone, which lists each object's presence, kind, position and velocity. Under micro it adds relations between pairs of objects: which objects touch and which object supports which. Under macro it adds two predicates of the whole scene: whether the scene has come to rest and whether the structure is unstable. In the scene of Figure 2, micro records that the pig touches the platform beneath it and that this platform supports the pig, while macro records only that the scene is not yet at rest. Under micro and macro the model reads these predicates from its state and feeds them back into the transition, so α changes what is rolled forward. One predictor serves every request and returns a carrier in the same

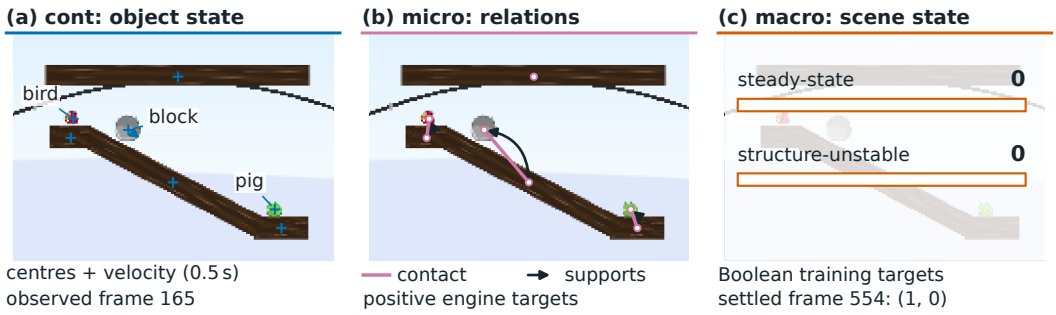


Figure 2: **Three description levels on one real frame.** A NovPhy frame during the cascade, described at each level α. (a) cont: each object's centre and velocity. (b) micro: objects in contact (lines) and support relations (arrows, from supporter to supported). (c) macro: two scene predicates, both false here; once the scene settles (frame 554), steady-state becomes true. All panels show engine states and the engine labels used as training targets, not model outputs.

space, $\hat{z}_{k+\Delta} = F_\theta(\hat{z}_k, a, r) \approx z_{k+\Delta}$, so a rollout can change its request at any step. The launch a conditions every prediction.

Given a launch a and a rollout endpoint of K carrier frames, the model rolls forward from $\hat{z}_0 = z_0$. At frame k, with $\kappa_k = K - k$ frames remaining, a request policy π_ψ issues one feasible request and the model advances by its horizon:

$$r_k = \underset{r \in \mathcal{R}:\ \Delta(r) \le \kappa_k}{\arg\max} \ \pi_\psi(r \mid \hat{z}_k, a, \kappa_k), \tag{4}$$

$$\hat{z}_{k+\Delta(r_k)} = F_\theta(\hat{z}_k, a, r_k), \qquad \kappa_{k+\Delta(r_k)} = \kappa_k - \Delta(r_k). \tag{5}$$

The rollout stops at $\kappa = 0$. Its steps last $\Delta(r_k)$ frames, so the rollout is a semi-MDP over carrier frames with state $(\hat{z}_k, \kappa_k)$ and the requests as actions, while the environment stays the MDP of Equation 1. A good schedule predicts accurately at little compute. On a training trajectory $z_{0:L}$ with launch a, a schedule $r_{1:J}$ whose horizons sum to L costs

$$G(r_{1:J}) = \sum_{j=1}^{J} \Big[\Delta(r_j)\, \tfrac{1}{d} \big\| F_\theta(\tilde{z}_{k_j}, a, r_j) - z_{k_j + \Delta(r_j)} \big\|_2^2 + \lambda\, \mathrm{MAC}(r_j) \Big], \tag{6}$$

where step j starts at frame k_j from the context carrier $\tilde{z}_{k_j}$, $d = 236$, and $\mathrm{MAC}(r)$ is the multiply-accumulate count of one transition under r, policy included, relative to the continuous transition at $\Delta = 15$. The error term is the duration-weighted error of one prediction from its context. Equation 6 needs the observed future, so it is defined on training trajectories only, whereas π_ψ must choose from $(\hat{z}_k, a, \kappa_k)$ alone. Section 4 solves Equation 6 by dynamic programming and distills the solutions into π_ψ.

At a decision state s, the agent imagines every launch in a candidate set $\mathcal{C}(s)$ by Equation 5 and reads a task cost on the predicted endpoint carrier $\hat{z}_K$:

$$\hat{J}(a) = 1000 \sum_{i \in \mathcal{I}_{\mathrm{pig}}} \bar{p}_i(\hat{z}_K) + \sum_{i \in \mathcal{I}_{\mathrm{block}}} \bar{p}_i(\hat{z}_K), \qquad \hat{a} = \underset{a \in \mathcal{C}(s)}{\arg\min}\, \hat{J}(a), \tag{7}$$

where $\bar{p}_i(z)$ is slot i's presence entry clipped to $[0, 1]$, and $\mathcal{I}_{\mathrm{pig}}$ and $\mathcal{I}_{\mathrm{block}}$ index the target (pig) and block slots. Ties go to the lower candidate ordinal. The engine outcome $y(s, a) \in \{0, 1\}$ is 1 when the engine, executing a from s, removes the target at the first shot; it enters neither training nor ranking. Because the agent acts on the order of $\hat{J}$, we score that order by the *ordering AUC*, the probability that a successful launch is predicted to cost less than a failed one:

$$\mathrm{AUC}(s) = \Pr\big[\hat{J}(a^+) < \hat{J}(a^-)\big] + \tfrac{1}{2} \Pr\big[\hat{J}(a^+) = \hat{J}(a^-)\big], \qquad y(s, a^+) = 1,\ y(s, a^-) = 0, \tag{8}$$

with a^+ and a^- drawn uniformly from $\mathcal{C}(s)$, defined when $\mathcal{C}(s)$ contains both outcomes. To isolate the request schedule, we compare π_ψ with fixed requests ($r_k \equiv r$) on the same checkpoint, candidates, endpoint and cost; the fixed-request reference $r^\dagger$ is the request with the best mean AUC on the other launch sets.

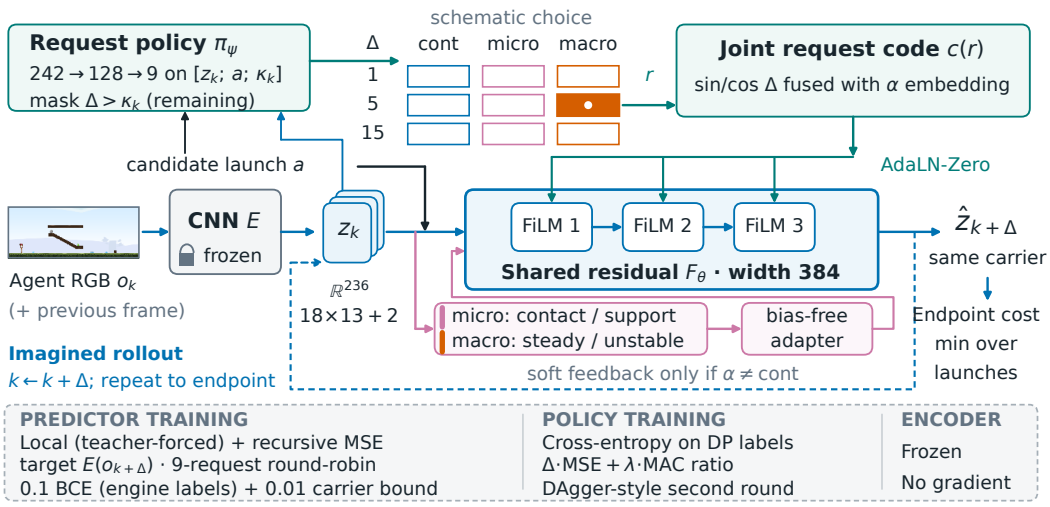


Figure 3: **Model overview.** One request-conditioned predictor serves all nine (Δ, α) requests. This JEPA-style latent predictor over a frozen object-slot encoder advances a continuous carrier (blue), receives request modulation (teal), and feeds soft predicates back only for micro/macro requests (violet/orange); training signals (bottom) are separate from runtime inputs. The highlighted cell is schematic; there is no EMA target encoder, and the policy was evaluated only in decision-only imagined rollouts, never in closed loop.

4 Method

The model has four parts (Figure 3): the frozen encoder E, the request-conditioned predictor F_θ, two predicate heads, and the request policy π_ψ. F_θ and the heads train jointly, and π_ψ trains afterwards with F_θ frozen. The predictor is JEPA-style in that it predicts future carriers in latent space without a pixel decoder; its targets are the frozen encoder's carriers of observed future frames, with no EMA target encoder or stop-gradient branch.

Encoder and carrier. E is a small CNN object-slot parser that reads the 96×64 agent frame into 18 named slots from a fixed vocabulary. It was trained once with supervised slot and predicate labels and stays frozen during all dynamics training. The carrier stacks two global entries g, whether a prior frame exists and the elapsed time, with 13 features per slot,

$$z = [g;\, f_1; \ldots; f_{18}] \in \mathbb{R}^{2+18\cdot 13} = \mathbb{R}^{236}, \tag{9}$$

where each slot's velocity is its displacement from the prior frame.

Request-conditioned predictor. F_θ is a residual MLP trunk of width 384 whose three blocks are all modulated by one code for the whole request. The code fuses sinusoidal features $\phi(\Delta)$ of the horizon with a learned embedding e_α of the description level in one MLP, so each of the nine requests receives its own modulation rather than a sum of a horizon term and a description term. Modulation is FiLM-style (Perez et al., 2018), with scale and shift initialized at zero as in AdaLN-Zero (Peebles & Xie, 2023), so training starts from an unmodulated trunk:

$$c(r) = \mathrm{MLP}_c\big([\phi(\Delta);\, e_\alpha]\big), \tag{10}$$

$$x^{(0)} = W_{\text{in}}\,[\hat{z}_k;\, a] + \mathbf{1}[\alpha \neq \textsf{cont}]\; U_\alpha\, h_\alpha(\hat{z}_k), \tag{11}$$

$$x^{(\ell)} = x^{(\ell-1)} + \mathrm{MLP}_\ell\Big(\mathrm{LN}\big(x^{(\ell-1)}\big) \odot \big(1 + \gamma_\ell(c)\big) + \beta_\ell(c)\Big), \qquad \ell = 1, 2, 3, \tag{12}$$

$$\hat{z}_{k+\Delta} = \hat{z}_k + W_{\text{out}}\,\mathrm{LN}\big(x^{(3)}\big), \tag{13}$$

where $a \in \mathbb{R}^5$ encodes the launch. The head $h_{\textsf{micro}}$ outputs contact and support probabilities over 18×18 slot pairs, and $h_{\textsf{macro}}$ outputs the two scene predicates, each with an availability output; the bias-free adapter U_α adds them to the trunk input, and a cont request runs no head. Per call, a micro request costs 25% more multiply-accumulates than a cont request and a macro request 2% more, so the compute of a schedule is set mainly by how many calls it makes, that is, by its horizons.

Training. Each update trains one request, cycling $r = \mathcal{R}[u \bmod 9]$ at update u without a curriculum, so all nine requests share every weight of the trunk. For a training window of observed carriers $z_{0:L}$ ($L \le 60$) with launch a, the loss unrolls $M \le 4$ steps of horizon Δ ending at $k_j = j\Delta \le L$:

$$\mathcal{L}(\theta; r) = \frac{1}{M} \sum_{j=1}^{M} \Big[\mathrm{mse}\big(F_\theta(z_{k_{j-1}}, a, r), z_{k_j}\big) + \mathrm{mse}\big(\hat{z}_j, z_{k_j}\big) + \lambda_{\mathrm{sym}}\, \ell_\alpha^{(j)} + \lambda_{\mathrm{bd}}\, b(\hat{z}_j) \Big], \quad (14)$$

where $\hat{z}_0 = z_0$ and $\hat{z}_j = F_\theta(\hat{z}_{j-1}, a, r)$ rolls without teacher forcing, $\mathrm{mse}(x, x') = \frac{1}{d}\|x - x'\|_2^2$, b penalizes carrier entries beyond ± 2, $\lambda_{\mathrm{sym}} = 0.1$ and $\lambda_{\mathrm{bd}} = 0.01$. The symbolic term $\ell_\alpha^{(j)}$ is a masked binary cross-entropy of head α on the predicted carrier and on the observed context, and $\ell_{\mathsf{cont}} \equiv 0$. Its labels come from the engine: contact and support relations projected from the captured object records for micro, and stability events and support-set changes for macro. Of the 2,409 training lineages, the 2,400 of the base corpus are captured at every engine step, so there Δ counts engine steps rather than the 50-step frames of the N1 captures.

Request policy. π_ψ is an MLP ($242 \to 128 \to 9$) on $[\hat{z}_k;\, a;\, \min(\kappa_k, 60)/60]$ whose logits for requests with $\Delta > \kappa_k$ are set to $-\infty$. With F_θ frozen, a backward dynamic program finds the schedule that minimizes Equation 6 on each training window, and π_ψ is trained by cross-entropy on its choices. With $g_k(r)$ the summand of Equation 6 for a step that starts at frame k, and $\lambda = 10^{-4}$,

$$V(k) = \min_{r \in \mathcal{R}:\ k + \Delta(r) \le L} \Big[g_k(r) + V\big(k + \Delta(r)\big) \Big], \qquad V(L) = 0, \quad (15)$$

$$\mathcal{L}_\pi(\psi) = -\sum_{k < L} \log \pi_\psi\big(r_k^\star \mid \tilde{z}_k, a, L - k\big), \quad (16)$$

where $r_k^\star$ is the minimizer in Equation 15, with ties broken by a fixed order of $\mathcal{R}$. In round 1 the contexts $\tilde{z}_k$ are the observed carriers. Round 2 is DAgger-style: it takes as contexts the carriers that the round-1 policy visits in its own rollout of the window, recomputes the labels against the same observed targets, and trains on both rounds' labels.

Horizon-only baseline. The baseline θ^{con} is the continuous-only model of Figure 1(b1) and, with its own policy, the horizon-only adaptive baseline of Section 1. It uses the trunk of Equations 12–13 and the loss of Equation 14 without symbolic terms, but its code depends on the horizon alone, $c(\Delta) = \mathrm{MLP}_c(\phi(\Delta))$; it has no heads, adapters or description embedding and accepts only cont. Its trunk width of 456 is the width nearest the hybrid's parameter count (1,862,396 against 1,837,690). Its policy chooses among the three continuous requests and is labelled by the same dynamic program with its own compute counts. Both models share seeds, minibatches and update budget, and at each update the baseline trains the same Δ as the hybrid. The compute comparison uses a training of both models by the same recipe on the base corpus alone (Appendix F).

5 Experimental Setup

Benchmarks and models. On NovPhy (Pinto et al., 2024) we use levels with normal physics, and the frozen encoder of Section 4 parses the rendered frames. CLEVRER (Yi et al., 2020) and Physion-Dominoes (Bear et al., 2021) have no agent actions; there we build the carrier from the released object annotations in the same 236-value layout and train both the hybrid and the baseline from scratch with the same architecture and update budget. Macro predicates exist only on NovPhy, so the other two benchmarks use cont and micro. Every model is trained with three seeds, and each benchmark is reported separately.

Rollout error. From one observed carrier, a fixed request (Δ, cont) is applied recursively, each prediction feeding the next without resets, and we report the mean squared error of the predicted object positions against the observed ones. On NovPhy the rollouts start at the launch frame of four held-out levels, one per candidate launch (13 per level), and run to 225 carrier frames; on CLEVRER and Physion-Dominoes they start from 48 context frames in 12 evaluation scenes and run to 120 frames.

Ranking candidate launches. A decision state is a NovPhy level with a set of candidate launches, each executed once in the engine to record its outcome y. Four launch sets vary the launch differently: a sweep of launch angles (9–13 candidates), a sweep of 11 other angles, a 4×4 grid of drag

Table 1: **Rollout error at the last endpoint.** Mean squared error of the predicted object positions after 225 carrier frames on NovPhy and 120 frames on CLEVRER and Physion-Dominoes, rolled recursively from one observed carrier (three-seed mean). The baseline is the continuous-only model of Section 4; the hybrid serves all nine requests with one set of weights. Macro predicates exist only on NovPhy.

Model	Request r	NovPhy	CLEVRER	Physion-Dominoes
Baseline	$(1, \mathsf{cont})$	2224	13.5	182.6
	$(5, \mathsf{cont})$	0.281	0.0471	0.0176
	$(15, \mathsf{cont})$	0.0370	0.0116	0.00114
Hybrid	$(1, \mathsf{cont})$	3141	2.98	167.4
	$(5, \mathsf{cont})$	0.200	0.0444	0.0156
	$(15, \mathsf{cont})$	0.0296	0.0170	0.00128
	$(15, \mathsf{micro})$	0.0313	0.0190	0.00199
	$(15, \mathsf{macro})$	0.0397	–	–

vectors, and a sweep of angle and launch power whose powers lie outside the training range (20 candidates). The sets cover the same 15 levels, and a level is scored in a set only when the set holds both a success and a failure, which leaves 7–14 levels per set. Every candidate is imagined from the level's launch-frame carrier to 225 carrier frames, and the ordering AUC (Equation 8) is computed per level and seed. On the same hybrid checkpoints we compare the request policy with all nine fixed requests: the cross-fitted reference $r^\dagger$ of Section 3 and, as an upper reference, the fixed request that is best on each set in hindsight.

Compute. The comparison with the horizon-only baseline uses 200 development states with 12 candidate launches each, rolled to 225 carrier frames, and the three seeds of the training in Appendix F. Each system chooses the launch of lowest predicted cost; the choice is scored by its regret in the count cost that the engine realizes at the end of the capture window, normalized per state to $[0, 1]$, and compute is the linear multiply-accumulates of all predictor and policy calls per candidate and state.

Statistics and scope. Each experiment ran under a plan frozen before the outcomes it scores existed (Appendix B). Intervals are 95% percentile bootstrap intervals that resample levels, scenes or states with all their seeds together, and are descriptive.

6 Results

6.1 Granularity changes rollout error

On all three benchmarks, a rollout that jumps 15 frames per prediction ends three to five orders of magnitude closer to the observed positions than one that steps frame by frame (Table 1), and the gap grows with the endpoint (Figure 1, b1): on NovPhy the two are within a factor of 1.5 after 15 frames and 6×10^4 apart after 225. This is the compounding error of step-wise rollouts (Talvitie, 2014; Venkatraman et al., 2015; Janner et al., 2019), and the horizon of each prediction controls it. The shared model keeps this accuracy: serving all nine requests with one set of weights, the hybrid stays within a factor of 1.5 of the dedicated continuous-only baseline at $\Delta = 15$ on every benchmark.

6.2 No fixed request suits every launch set

Granularity also decides how well imagined outcomes rank candidate launches. Within a level, the best and the worst of the nine fixed requests differ by 0.44–0.51 in ordering AUC on average, and which request is best depends on the launch set (Figure 4). The request $(5, \mathsf{macro})$ ranks first of nine on the angle and offset sweeps (AUC 0.63 and 0.59) but last on the drag grid (0.40), where $(1, \mathsf{macro})$ is best (0.51); $(1, \mathsf{macro})$ is also best on the power sweep (0.53). The two axes move separately: the best request is macro on every set, while its horizon is 5 on two sets and 1 on the other two. A fixed request tuned on the other three sets therefore transfers poorly. On no set does $r^\dagger$ coincide with the set's best request, and it trails that request by 0.03–0.13.

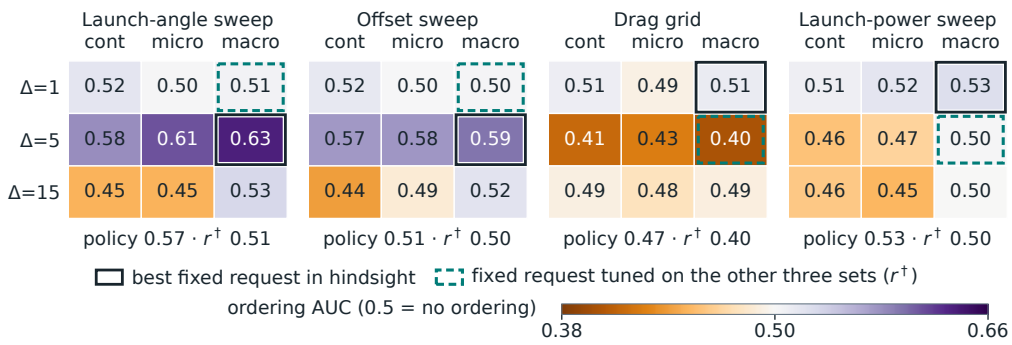


Figure 4: **The best fixed request changes with the launch set.** Ordering AUC of each of the nine fixed requests on each launch set (hybrid checkpoints, mean over levels and seeds). Solid box: best fixed request in hindsight. Dashed box: the fixed request tuned on the other three sets ($r^\dagger$). Below each panel: the AUC of the request policy and of $r^\dagger$.

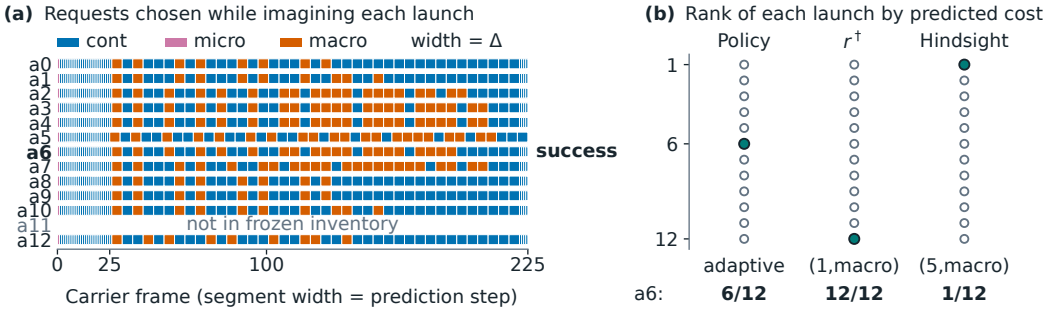


Figure 5: **One decision, seen through its requests.** The level of Figures 1 and 2, with 12 candidate launches from flat (a0) to steep (a12); only a6 succeeds in the engine. (a) Requests the policy chooses while imagining each launch; each segment is one prediction, coloured by α, with width Δ. (b) Rank of each launch by predicted cost (1 = chosen) under the policy, the transferred fixed request $r^\dagger$ and the best fixed request in hindsight; the filled dot is a6. One decision, not a frequency estimate.

6.3 Choosing the request per prediction

The request policy changes the request from one prediction to the next, using six or seven of the nine requests in each seed's rollouts. Figure 1(a) shows one such schedule: single-frame steps right after the launch, alternating $(5, \mathsf{cont})$ and $(5, \mathsf{macro})$ steps through the interaction, and 15-frame macro steps late in the rollout. On the same checkpoints, this per-prediction choice orders launches better than the transferred fixed request $r^\dagger$ on all four sets: by $+0.060$ on the angle sweep, $+0.004$ on the offset sweep, $+0.070$ on the drag grid and $+0.027$ on the power sweep (Figure 1, b2).

Choosing both axes also saves compute. On the 200 development states, the hybrid's policy matches the horizon-only adaptive baseline, with a regret of 0.350 against 0.367, at a third of its compute: 118.7M against 356.0M multiply-accumulates per candidate and state, and 1.0 s against 2.0 s of wall time per state. The saving comes from taking longer steps. The hybrid's policy makes 80 predictor calls per candidate and state where the baseline's makes 189, and a call costs about the same under every request (Section 4).

6.4 How the policy chooses requests

Figure 5 opens one decision. While imagining each launch, the policy starts with single-frame steps and then alternates $(5, \mathsf{cont})$ and $(5, \mathsf{macro})$, and where it places the macro steps depends on the launch: the 12 candidates receive eight different schedules. The resulting order sits between the two fixed references, as in the aggregate. The policy ranks the one successful launch, a6, sixth of 12, the

transferred fixed request $(1, \mathsf{macro})$ ranks it last, and the best fixed request in hindsight, $(5, \mathsf{macro})$, ranks it first. The same behaviour holds across all 2,634 imagined rollouts. In 77% of level–seed pairs the schedule differs between candidate launches, so the choice depends on the imagined state and not only on the time step. Longer and coarser requests become more frequent as the rollout proceeds: the share of $\Delta = 5$ rises from 2% of predictions in the first 25 frames to 16% after frame 100, and the share of macro from 59% to 83%.

7 Discussion

Granularity as a decision. The three results fit together. The horizon of each prediction sets rollout error by orders of magnitude, and the request that ranks candidate launches best depends on the situation along both axes; granularity is therefore worth choosing, and no single choice is safe. A policy that chooses the request per prediction, over one shared latent state, improves on a fixed request carried over from other launch sets and matches a horizon-only adaptive baseline at a third of its compute. These results make the case for treating how far and at what level to predict as part of each query to a world model, rather than as a constant fixed at design time.

Relation to hierarchies and simulation. Hierarchical JEPA and multi-timescale world models tie each time scale to one level of abstraction (LeCun, 2022; Saxena et al., 2021). In our results the two axes vary separately: the best request uses macro on every launch set, while its horizon moves between 1 and 5. This suggests exposing horizon and description as independent coordinates of one request, which a hierarchy could still serve, rather than as levels of a stack. The learned schedule of Figure 1(a) also resembles adaptive stepping in simulation (Hairer et al., 1993; Mirtich, 2000), with short steps while the launch unfolds and long, coarse steps late in the rollout, a pattern that comes from trading prediction error against compute rather than from a hand-set schedule.

Scope and next steps. Macro predicates exist only on NovPhy; the other two benchmarks exercise the continuous and micro levels. The natural next steps are to let the chosen requests drive actions in closed loop, to learn the description levels together with the encoder, and to train the request policy on a decision-aware cost.

8 Conclusion

We made prediction granularity a per-prediction decision of one world model. An action-sparse MDP stays an ordinary MDP while every model query carries a request $r = (\Delta, \alpha)$; one JEPA-style predictor serves all nine requests over one shared latent state; and a policy distilled from dynamic-programming labels chooses the request at each step. On NovPhy, which we introduce as a world-model testbed, and on CLEVRER and Physion-Dominoes, granularity changes rollout error by orders of magnitude, no fixed request ranks candidate launches best everywhere, and choosing the request per prediction improves on a transferred fixed request while matching a horizon-only adaptive baseline at a third of its compute. We hope these results encourage world models that ask how far and at what level to predict at every query, rather than answering once at design time.

References

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, 2023. doi: 10.1109/CVPR52729.2023.01499.

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

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2.

Daniel M. Bear, Elias Wang, Damian Mrowca, Felix J. Binder, Hsiao-Yu Fish Tung, R. T. Pramod, Cameron Holdaway, Sirui Tao, Kevin A. Smith, Fan-Yun Sun, Li Fei-Fei, Nancy Kanwisher, Joshua B. Tenenbaum, Daniel L. K. Yamins, and Judith E. Fan. Physion: Evaluating physical prediction from vision in humans and machines. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021), Datasets and Benchmarks Track*, 2021.

André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. Temporl: Learning when to act. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR / OpenReview.net, 2024.

Tim Brüdigam, Daniel Prader, Dirk Wollherr, and Marion Leibold. Model predictive control with models of different granularity and a non-uniformly spaced prediction horizon. In *2021 American Control Conference (ACC)*, pp. 3876–3881. IEEE, 2021. doi: 10.23919/ACC50511.2021.9482617.

Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In Wayne A. Davis, Marilyn M. Mantei, and R. Victor Klassen (eds.), *Proceedings of the Graphics Interface 1997 Conference, May 21-23, 1997, Kelowna, BC, Canada*, pp. 1–8. Canadian Human-Computer Communications Society, 1997.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model, 2025. URL https://arxiv.org/abs/2506.21539.

Tianqi Du, Qi Zhang, Yifei Wang, and Yisen Wang. Beyond the next step: Variable-length latent world models for long-horizon planning. *arXiv preprint arXiv:2606.21775*, 2026. doi: 10.48550/arXiv.2606.21775.

Rohit Girdhar, Laura Gustafson, Aaron Adcock, and Laurens van der Maaten. Forward prediction for physical reasoning, 2020. URL https://arxiv.org/abs/2006.10734.

Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *International Conference on Learning Representations (ICLR)*, 2024.

Muhammad Burhan Hafez, Cornelius Weber, Matthias Kerzel, and Stefan Wermter. Improving robot dual-system motor learning with intrinsically motivated meta-control and latent-space experience imagination. *Robotics and Autonomous Systems*, 133:103630, 2020. doi: 10.1016/j.robot.2020.103630.

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, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 26091–26104, 2022. doi: 10.52202/068431-1892. URL https://papers.nips.cc/paper_files/paper/2022/hash/a766f56d2da42cae20b5652970ec04ef-Abstract-Conference.html.

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

Ernst Hairer, Gerhard Wanner, and Syvert P Nørsett. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

William Huang and Benjamin Freed. Salt: Learning state- and temporally-abstracted world models for offline long-horizon decision-making. In *ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation*, 2026.

Tomoshi Iiyama, Masahiro Suzuki, and Yutaka Matsuo. Sunta: Hierarchical video prediction with surprise-based chunking. *arXiv preprint arXiv:2607.02087*, 2026. doi: 10.48550/arXiv.2607.02087.

Jinwoo Jang, Daniel J. Rho, Sihyung Yoon, Hyunsuk Cho, and Honguk Woo. Multi-scale mixture of world models for embodied agents in evolving environments. *arXiv preprint arXiv:2607.00457*, 2026. doi: 10.48550/arXiv.2607.00457.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12498–12509, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Dinesh Jayaraman, Frederik Ebert, Alexei A. Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=SyzVb3CcFX.

K. V. Katsikopoulos and S. E. Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *IEEE Transactions on Automatic Control*, 48(4):568–574, 2003. doi: 10.1109/TAC.2003.809799.

Taesup Kim, Sungjin Ahn, and Yoshua Bengio. Variational temporal abstraction. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/b5d3ad899f70013367f24e0b1fa75944-Abstract.html.

Yann LeCun. A path towards autonomous machine intelligence, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. OpenReview archive, version 0.9.2.

Shiqian Li, Kewen Wu, Chi Zhang, and Yixin Zhu. I-phyre: Interactive physical reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B. Tenenbaum, Tom Silver, João F. Henriques, and Kevin Ellis. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In *International Conference on Learning Representations (ICLR)*, 2025.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026. doi: 10.48550/arXiv.2603.19312.

Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 6862–6873, 2020. URL https://proceedings.mlr.press/v119/metelli20a.html.

Brian Mirtich. Timewarp rigid body simulation. In Judith R. Brown and Kurt Akeley (eds.), *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000, New Orleans, LA, USA, July 23-28, 2000*, pp. 193–200. ACM, 2000. doi: 10.1145/344779.344866. URL https://doi.org/10.1145/344779.344866.

Anh N. Nhu, Sanghyun Son, and Ming Lin. Time-aware world model for adaptive prediction and control. In *Proceedings of Machine Learning Research (ICML 2025 Workshop track), Vol. 267*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 4172–4182. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00387. URL https://doi.org/10.1109/ICCV51070.2023.00387.

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

Karl Pertsch, Oleh Rybkin, Jingyun Yang, Shenghao Zhou, Konstantinos Derpanis, Kostas Daniilidis, Joseph Lim, and Andrew Jaegle. Keyframing the future: Keyframe discovery for visual prediction and planning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, pp. 969–979, 2020. URL https://proceedings.mlr.press/v120/pertsch20a.html.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Vimukthini Pinto, Chathura Gamage, Cheng Xue, Peng Zhang, Ekaterina Nikonova, Matthew Stephenson, and Jochen Renz. Novphy: A physical reasoning benchmark for open-world ai systems. *Artificial Intelligence*, 336:104198, 2024. doi: 10.1016/j.artint.2024.104198.

Vaibhav Saxena, Jimmy Ba, and Danijar Hafner. Clockwork variational autoencoders. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pp. 29246–29257, 2021. URL https://papers.nips.cc/paper_files/paper/2021/hash/f490d0af974fedf90cb0f1edce8e3dd5-Abstract.html.

Daniel Scarafoni, Irfan Essa, and Thomas Ploetz. Finding islands of predictability in action forecasting. *arXiv preprint arXiv:2210.07354*, 2022. doi: 10.48550/arXiv.2210.07354.

Vaisakh Shaj, Saleh Gholam Zadeh, Ozan Demir, Luiz Ricardo Douat, and Gerhard Neumann. Multi time scale world models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

Erik Talvitie. Model regularization for stable sample rollouts. In Nevin L. Zhang and Jin Tian (eds.), *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014*, pp. 780–789. AUAI Press, 2014. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2514&proceeding_id=30.

Rishi Upadhyay, Howard Zhang, Jim Solomon, Ayush Agrawal, Yunhao Ba, Alex Wong, Celso M. de Melo, and Achuta Kadambi. Worldbench: Benchmarking physical understanding of world models by isolating physics concepts. *arXiv preprint arXiv:2601.21282*, 2026.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In Blai Bonet and Sven Koenig (eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pp. 3024–3030. AAAI Press, 2015. doi: 10.1609/AAAI.V29I1.9590. URL https://doi.org/10.1609/aaai.v29i1.9590.

Angel Villar-Corrales, Ani Karapetyan, Andreas Boltres, and Sven Behnke. Mspred: Video prediction at multiple spatio-temporal scales with hierarchical recurrent networks. In *Proceedings of the 33rd British Machine Vision Conference (BMVC)*, 2022.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026. doi: 10.48550/arXiv.2606.32026.

Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-resolution model learning for object pile manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.047.

Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=sFyTZEqmUY.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations (ICLR)*, 2020.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal Patel, Paul Liang, Daniel Khashabi, Cheng Peng, Rama Chellappa, Tianmin Shu, Alan Yuille, Yilun Du, and Jieneng Chen. World-in-world: World models in a closed-loop world. In *International Conference on Learning Representations (ICLR)*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2025. doi: 10.48550/arXiv.2411.04983.

A Use of LLM Statement

An LLM was used only for polishing language; the scientific content is entirely original.

B Protocol Mechanics and Measurement Conventions

This section defines the protocol lifecycle and the measurement conventions that every downstream section uses. Throughout the appendices, h denotes the requested horizon Δ of Section 3, counted in carrier frames. Every executed artifact in this paper is produced under a frozen plan. The lifecycle has five stages: freeze, validate, execute, publish, exact-validate. At the freeze stage the runner writes `plan.json`, which pins membership, templates, seeds, estimands, metrics, contrasts, compute accounting, and stop rules; a freeze contains actual numerical values, and approximations or example values in place of values are banned. After the freeze, loading the plan re-hashes the source tree and the runner refuses to execute after any source edit. The validate stage checks the plan before any outcome is computed, execution then runs every scheduled cell, and publication writes the #77 layout (`summary.json`, `comparisons.csv`, `findings.md`) beside the producing module. The exact-validate stage reruns a `--validate` mode that recomputes every published table from the artifacts and must exit 0. Campaign runners additionally write `inventory.json`, `campaign.json`, `ledger.json`, `coverage.json`, per-branch `markers/`, `receipts/`,

and `attempts/`, and their `--validate` mode re-derives `coverage.json` from the retained ledger. A recorded reseal guard regenerates a plan through its own prepare module and aborts unless the only drift is recorded source text, with membership, branches, and prior-exposure byte-identical. Superseded pre-review plans stay on disk and were never used for any outcome.

A typed terminal failure is a scheduled cell retained with its failure kind when it cannot complete. Every scheduled cell is executed or retained as a typed terminal failure. There is no outcome-conditioned exclusion, retry, replacement, or re-freeze. A terminal-incomplete ledger refuses top-up permanently, and all failure records are kept. Allowances were recorded per ticket, and an unfunded cell is recorded `unsupported`, never dropped.

Measurement discipline is uniform. All paired contrasts carry bootstrap intervals over paired units, labelled descriptive, and no interval is read as a significance test. No multiplicity adjustment is applied. Zero-shot and few-shot conditions are reported separately and never pooled. Evaluation states are held-out lineages with at least one coverage-admissible branch, taken in sorted identity order. In the N1 record the candidate inventory is the frozen 13-candidate set (reference `a00` plus angular candidates `a01`–`a12`), with per-state counts reduced by typed branch failures.

Compute accounting reports per-state wall time, active-parameter step counts, and candidate counts. Training is equalized: 9,000 updates with batch 64 per cell, which is 576,000 examples, and identical minibatches per paired seed across arms. Linear MACs are reported rather than full FLOPs. The capacity contract matches the continuous width to the hybrid total parameter count within a 2% tolerance; the #74 fits measure 1,862,396 continuous versus 1,837,690 hybrid parameters, a relative difference of 0.0134. All fits ran on one RTX 3090 under an exclusive lock, and every port targets the existing $\sim$1.8M-parameter frozen-carrier predictor scale. All fits ran on one RTX 3090 under an exclusive lock, and every port targets the existing $\sim$1.8M-parameter frozen-carrier predictor scale.

Paired checkpoints use three seeds, 20260908, 20260909, and 20260910, across the retraining families. Campaign generation and engine seeds are pinned disjoint from all earlier ranges in each frozen plan. The #77 bases are 764,000,000+ordinal for generation and 764,100,000+ordinal for the engine, with 764,200,000+, 764,300,000+, and 764,400,000+ reserved for the #75-adjacent N2 capture, engine, and adaptation streams. Table 2 inventories the per-seed artifacts.

The protocol's design discipline is **frozen plans** (every campaign, training run, and evaluation run executes a plan frozen before launch and re-hashed at launch, refusing the run after any source edit); **sealed membership** (cell membership, held-out state identities, and the candidate inventory are pinned before any rendered run or model score, and no outcome moves membership); **typed terminal failures** (every scheduled unit ends admissible or as a typed failure with a recorded kind, retained and reported per cell, never silently dropped or worst-cased); and **equal-update training with work reporting** (paired arms share minibatches and update budgets, and the continuous width is matched to the hybrid parameter count within a width tolerance). Its scoring discipline is **paired descriptive inference** (contrasts report mean paired differences over paired units with descriptive 95% paired-bootstrap intervals); a **declared candidate inventory** (the 13 candidate actions per state, one reference drag plus 12 release-angle actions, are pinned before scoring, typed branch failures remove candidates without refill, and CLEVRER runs no inventory); and **every attempted cell reported** (the accounting is fixed in advance and reported in full, with all typed failures retained).

C CLEVRER Replication Details

This section documents the CLEVRER replication: data provenance, frozen membership rule, Phase-0 vocabulary mapping, endpoint grid, and training protocol. The data source is the official CLEVRER release at `data.csail.mit.edu/clevrer` under a CC0 license (Yi et al., 2020). Only the annotation archives were used: train at 156,761,889 bytes with `sha256` prefix `e7f9185a` and validation at 78,386,966 bytes with prefix `d89e7471`. No videos were downloaded. Train scenes 0–9999 and validation scenes 10000–14999 ship annotations, and the test split ships none. Each annotation contains `object_property` records for color, material, and shape, per-frame `motion_trajectory` entries with per-object location, orientation, velocity, and angular velocity, collision events with `frame_id`, and an `inside_camera_view` flag. That flag is the availability mask, applied in slot columns 7 and 10 and in every masked estimand, and unavailable fields are zeroed exactly as the deployment carrier zeroes them. Vertical location and velocity,

Table 2: Per-seed checkpoint inventory. All rows use paired seeds 20260908, 20260909, and 20260910 with identical minibatches per paired seed.

Family and root	Arms	Per-seed artifacts and budget
#74 matched systems, `.local-artifacts/issue-74-matched-dynamics-v1`	`continuous`, `hybrid`	`seed-*/{continuous,hybrid}/{predictor,controller}.pt`; six fits, 9,000 updates each, selection at the last update
#77 N1 retraining, `.local-artifacts/issue-77-n1-dynamics-v1`	`continuous`, `hybrid`	`seed-*/{continuous,hybrid}/`; six fits, 9,000 updates each, identical recipe to #74
#79 CLEVRER replication, `.local-artifacts/issue-79-clevrer-boundary-v1`	`continuous`, `hybrid`	`seed-*/{continuous,hybrid}/predictor.pt`; six fits, 9,000 updates each

orientation matrices, and angular velocities are parsed into per-scene shards for the parser-contract record with no carrier-column referent. No official generation or simulation code exists. The public CLEVRER repository at pinned commit `98b8420` contains model code only, and the paper's Bullet-plus-Blender pipeline was never released, so regenerated scenes are not a fallback. Had the released fields proved insufficient for the feature-parser contract, the frozen rule required publishing `readiness_or_precision_insufficient` naming the exact blocker.

Membership was frozen before any training. The rule selects the first 12 collision-complete scenes in ascending `scene_index` within the validation-split annotations, where collision-complete means a contiguous 128-frame trajectory for every declared object and at least one collision event, with no outcome-based selection. The selected scenes are 10000 through 10011, with 4 to 6 objects and 1 to 4 collisions each. Each scene yields 40 training windows with starts 0 through 39 and length 61, for 480 windows in total, and the evaluation contexts $\{0, 1, 2, 3\}$ give 48 scene–context units per seed. The scan visited exactly these 12 scenes and skipped none, and the membership function returns typed failures for skipped scenes rather than dropping them silently.

The Phase 0 vocabulary mapping was declared before any training. The carrier reuses the unchanged #71 layout of 2 global values plus 18 slots of 13 columns, with kind vocabulary cube, cylinder, and sphere, position scale 5.0, velocity scale 3.0, and speed-tier edges frozen at $[0.3499, 1.6807]$. The micro predicates are contact, derived from collision events as symmetric channel 0, and velocity-bin, a same-tier indicator at the frozen edges as symmetric channel 1. CLEVRER scenes contain no support structures, so `supports` and `structure-unstable` have no referent. The macro arm is therefore not trained or evaluated, and its estimand is dropped with no invented vocabulary. A null action tensor held constant serves both arms.

The endpoint grid is $\{15, 30, 60, 120\}$ CLEVRER frames, fixed by a divisibility rule. Every endpoint is divisible by all $h \in \{1, 5, 15\}$, and every context plus endpoint stays within the 128-frame clip, which holds for all four contexts since the largest sum is $3 + 120 = 123 \leq 127$. The NovPhy grid $\{15, \ldots, 600\}$ does not transfer because 600 exceeds the clip, and divisibility is the freezing rule. The estimand is a recursive fixed-pair rollout from the true carrier at each context, with one shared initial carrier, predictions feeding the next transition, and no truth resets. Metrics are position, velocity, carrier, presence, and kind mean squared error, with position and velocity masked by `inside_camera_view` at the target frame. Ranking regret is not an estimand here because no candidate inventory exists on CLEVRER. Engine-annotation features are the input on both arms, there is no visual perception on either arm, and the parser contract is shared and frozen.

Both arms were retrained from scratch, pure continuous versus the micro-symbolic hybrid, for 3 paired seeds at 9,000 updates with identical minibatches. The objective is local MSE plus recursive MSE plus the carrier-bound penalty at the #74 weight of 0.01, and the hybrid adds micro symbolic losses at weight 0.1. The hybrid macro head and adapter receive no CLEVRER gradient and are excluded from the expected-gradient parameter count. The compute allowance was 6 GPU-hours, and actual consumption was 1,494.6 seconds of training plus 52.6 seconds of evaluation, approximately 0.43 GPU-hours, with peak CUDA allocation of 105 MiB and zero typed failures. Units per arm

were 432 continuous versus 864 hybrid, and active-parameter counts, linear MACs per step, and transition-step counts per system are recorded in `summary.json`.

D Extended Related Work

Existing methods reasonably fix predictive horizons and targets to suit their intended objectives. Latent-dynamics control methods fix temporal targets or planning horizons and reach further by recursive one-step rollout with value bootstrapping (Hafner et al., 2020; Hansen et al., 2022). Joint-embedding approaches fix the predictive representation and target at training time (Assran et al., 2023; Bardes et al., 2024). Recent latent world models on frozen or end-to-end features inherit the same fixed target (Zhou et al., 2025; Maes et al., 2026). This task-specific design is a widespread default. We ask whether one action-conditioned model serving heterogeneous decisions should instead select prediction horizon and output-description mode jointly at each decision.

Recent work relaxes one dimension of this contract at a time. On the temporal side, variable-length latent world models vary prediction offsets (Du et al., 2026), surprise signals set hierarchical chunk boundaries in open-loop video prediction (Iiyama et al., 2026), and reliability estimates adapt roll-out depth during imagination (Hafez et al., 2020). On the description side, published work varies the output form: state-dependent scene resolution (Wang et al., 2023) and per-frame abstraction of action forecasts (Scarafoni et al., 2022). Fixed couplings of the two also exist: hierarchical prediction across fixed time scales (Shaj et al., 2023; Villar-Corrales et al., 2022) and short-detailed versus long-simplified segments in one model-predictive controller (Brüdigam et al., 2021). Wang et al. (2026) adapt model parameters at test time, not granularity selection. None of these lines treats horizon and output-description mode jointly as one state-dependent prediction request to an action-conditioned world model, which is the narrow gap the granularity program targeted.

Temporal abstraction in reinforcement learning answers a different question. Options and their learned variants decide when and for how long the agent acts (Sutton et al., 1999; Bacon et al., 2017), and Biedenkapp et al. (2021) learn when to commit to an action repetition. These methods adapt the agent's control schedule, not what the world model predicts or how far ahead it extends. Our question sits on the prediction side of that boundary.

In the *action-sparse persistent-effect environment*, actions occur infrequently while their physical effects continue autonomously for many fixed simulation steps. A single intervention can set off a cascade through collision onset and active rearrangement before settling. We build on NovPhy (Pinto et al., 2024), a published physical-reasoning benchmark (Bakhtin et al., 2019; Bear et al., 2021; Yi et al., 2020) evaluating how agents detect and adapt to novel physical dynamics across scenarios. Each cascade phase, we hypothesize, asks for different prediction granularity: fine temporal resolution and relation-level detail near the intervention, coarse state judgments near settling. We call this tension *granularity mismatch* and treat it as a heterogeneity hypothesis. Effects, if any, depend on the decision's phase, the physical family, and the scored horizon. An average over cells can hide either sign.

Granularity can enter a world model on the temporal axis, on the description axis, or on both at once. The paragraphs above survey fixed designs and present them as reasonable task-specific choices. The subsections below cover methods that make at least one axis adaptive, the closest joint neighbour, and the evaluation platforms that currently set the standard for judging world models. Table 3 summarizes the positioning, verified against each paper's full text. The one entry without public text is marked unverified.

D.1 Temporal adaptation of prediction

The temporal-abstraction paragraph above draws the boundary between prediction-side granularity and the control schedule. The options framework, its learned descendants, and TempoRL sit on the acting side of that line, deciding when and for how long the agent acts (Sutton et al., 1999; Bacon et al., 2017; Biedenkapp et al., 2021). They adapt the agent's commitment rather than the prediction, so we treat them as the boundary of the survey class rather than as members of it.

Within the prediction side, TAWM conditions predictions on the time-step size Δt and trains across a range of step sizes, so a single model serves tasks observed at different temporal grains (Nhu et al.,

2025). The grain at inference is imposed by the environment's observation rate rather than chosen per decision, so TAWM tolerates rate changes instead of adapting its grain per state. VLWM predicts future latent states conditioned on action sequences of variable length and extends the horizon through a training curriculum (Du et al., 2026). Its planner is tailored to the variable-length predictor. THICK learns a hierarchy over discrete latent dynamics in which parts of the latent state update sparsely, forming invariant contexts whose durations are learned (Gumbsch et al., 2024). A higher level predicts when those contexts change. SUNTA varies chunk length at inference inside a hierarchical video predictor, splitting the rollout where surprise spikes, and reports prediction staying accurate to 250 steps where the compared baselines degrade within ten (Iiyama et al., 2026). Its scope is open-loop video prediction with a fixed output format. TAWM and VLWM express the prediction in a single latent format, while THICK fixes a two-level discrete latent hierarchy. None of the four casts grain selection as an explicit per-decision request over a prediction surface.

D.2 Description adaptation

The description axis has its own adaptive tradition. VisualPredicator learns abstract world models over an invented first-order predicate language, with an online algorithm that invents predicates and plans over the resulting abstract states (Liang et al., 2025). Resolution-adaptive scene representations and per-frame abstraction of action forecasts, both introduced above, also adapt what the model expresses (Wang et al., 2023; Scarafoni et al., 2022). On this axis the adaptation selects the state description. The prediction length remains fixed, so the joint request over (Δ, α) remains open.

D.3 Joint state and temporal abstraction

SALT addresses state- and temporally-abstracted world models for offline long-horizon decision-making (Huang & Freed, 2026), the nearest title-level overlap with our joint request. No public text is available beyond the workshop's poster listing, so the SALT row of Table 3 is marked unverified. Our surface makes the abstraction a per-decision prediction request, selecting which supervised readout is scored and how many steps the carrier advances, and scores that request against fixed-granularity references as a property of prediction.

D.4 Evaluation zeitgeist

Two 2026 evaluation efforts set the current standard for judging world models. World-in-World benchmarks world models inside a closed loop with a unified online planning strategy and a standardized action interface (Zhang et al., 2026). Task success rather than visual fidelity is the primary metric. Reported findings include that visual quality alone does not guarantee task success and that more inference-time compute improves closed-loop performance. WorldBench evaluates video-based world models with physics concepts isolated one at a time, at two levels: intuitive concepts such as object permanence, and low-level constants such as friction coefficients and viscosity (Upadhyay et al., 2026). Both platforms target diagnostic criteria over aggregate quality. Neither platform treats granularity as the object of evaluation or asks where an adaptive-granularity advantage stops.

PHYRE certifies task solvability by construction and scores agents by AUCCESS over up to 100 attempts per task (Bakhtin et al., 2019), and I-PHYRE reports human success alongside an experimenter upper bound (Li et al., 2024). Physion evaluates event-prediction accuracy, not agent task completion, so it is outside this success-rate comparison (Bear et al., 2021).

Horizon-resolved or rate-resolved plots do appear individually: VLWM reports success against goal offset, TAWM reports success against the evaluation time step, THICK sweeps corridor length, and SUNTA reports horizon-resolved degradation of open-loop video prediction. These are post-hoc, single-system-suite analyses without frozen protocols, uncertainty discipline, or family resolution, and none reports where the advantage inverts. Across the surveyed class, verified per paper against the full texts (Table 3), no method reports a pre-registered, matched-system comparison together with a family- and horizon-resolved boundary curve.[1] The claim is scoped to the surveyed papers with public text.

[1] SALT has no public text, so we could not verify its protocol.

Method	Temporal adaptive	Description adaptive	Pre-registered protocol	Matched-system comparison	Family- and horizon-resolved boundary curve
TAWM (Nhu et al., 2025)	Partial	No	No	Partial	Partial
VLWM (Du et al., 2026)	Yes	No	No	Partial	Partial
THICK (Gumbsch et al., 2024)	Yes	Partial	No	No	Partial
SUNTA (Iiyama et al., 2026)	Yes	No	No	No	Partial
Options line (Sutton et al., 1999; Bacon et al., 2017; Biedenkapp et al., 2021)	Yes (control schedule)	No	–	–	–
VisualPredicator (Liang et al., 2025)	No	Yes	No	No	No
SALT (Huang & Freed, 2026)	?	?	?	?	?
World-in-World (Zhang et al., 2026)	No	No	–	Partial	–
WorldBench (Upadhyay et al., 2026)	No	No	No	No	Partial
This paper	Yes	Yes	Yes	Yes	Yes

Table 3: Positioning matrix. Cells follow full-text verification of the surveyed papers as of September 2026. The SALT row is marked "?" because no public text exists. "Partial" in the temporal column records that TAWM's inference time step is imposed by the environment's observation rate rather than selected per decision. In the description column it records that THICK's two-level hierarchy is fixed by its architecture. In the matched-system column it records training at equal samples and iterations without inference-work accounting (TAWM), an implementation- and planner-matched comparison on a single seed (VLWM), and a fixed planning strategy with a standardized action interface that holds the decision layer constant across heterogeneous world models (World-in-World). In the boundary column it records resolution short of the full conjunct: success against the evaluation time step (TAWM), success against goal offset with post-hoc oracle-per-cell planner selection (VLWM), a corridor-length sweep of task horizon in one environment (THICK), horizon-resolved degradation of open-loop video prediction without family resolution or work accounting (SUNTA), and concept-resolved failure patterns without a horizon axis (WorldBench). "–" marks properties that do not apply to an entry's role: the options line acts on the control schedule and makes no prediction-granularity claim, and World-in-World evaluates rather than trains adaptive-granularity methods. For this paper, the adaptive columns record that the request policy selects horizon and description per prediction, the matched-system column records that one checkpoint serves all nine requests under a matched training recipe, and the boundary column records the horizon-resolved rollout error of Figure 1(b1) and the per-launch-set request rankings of Figure 4. The protocol column refers to the frozen plans of Appendix B.

Estimand	**What varies**	**What stays fixed**
Training $\tau_{\mathsf{tr}}(\Delta)$	**training recipe**	readout and request
Execution $\tau_{\mathsf{ex}}(\Delta, \alpha)$	**executed readout**	checkpoint and horizon
Adaptation τ_{ad}	**selection schedule**	training + readout pairs

Figure 6: Three comparison targets. Training-effect contrasts vary the training recipe; execution-effect contrasts vary the executed readout at a shared checkpoint; adaptation-effect contrasts vary the per-decision selection schedule. Other factors must be matched to isolate each target.

E Estimand Taxonomy Details

Here, an estimand is the effect a comparison aims to measure. Comparisons across the nine-pair surface differ in what they hold fixed and what they let vary. A single aggregate would conflate three distinct questions: whether symbol-supervised training changes the shared carrier, whether executing a symbolic readout changes prediction at a fixed checkpoint, and whether per-decision selection improves on fixed pairs. The three require different evidence, and an answer to one does not transfer to the others. We define each below as a contrast in which exactly one factor differs (Figure 6 and Table 4).

Table 4: The three estimands. "Varies" names the factor that differs between the two sides of a contrast. The remaining two factors are identical by construction, which is what fixes the estimand's meaning.

Estimand	Contrast (tested vs. reference)	Varies / isolates
Training effect $\tau_{\mathrm{tr}}(\Delta)$	hybrid-trained model vs. independently trained pure-continuous model, both executing $(\Delta, \mathrm{continuous})$	training recipe, isolating what symbol-supervised training does to the carrier
Symbolic-execution effect $\tau_{\mathrm{ex}}(\Delta, \alpha)$	fixed (Δ, α) with $\alpha \in \{\mathrm{micro}, \mathrm{macro}\}$ vs. $(\Delta, \mathrm{continuous})$, inside one hybrid checkpoint	executed readout, isolating what executing a symbolic readout contributes at a fixed checkpoint
Adaptation effect τ_{ad}	per-decision selection policy vs. fixed-pair controls, within an arm at matched training	selection schedule, isolating the value of adapting granularity per decision; defined here

Each estimand varies exactly one factor and holds the other two fixed by construction. The *training effect* $\tau_{\mathrm{tr}}(\Delta)$ varies the training recipe, comparing a model trained with symbol-supervised readouts (the hybrid arm) against an independently trained pure-continuous model, both executing the fixed continuous pair $(\Delta, \mathrm{continuous})$. The *symbolic-execution effect* $\tau_{\mathrm{ex}}(\Delta, \alpha)$ varies the executed readout, comparing a fixed symbolic readout (Δ, α) against the fixed continuous readout at the same Δ, inside one hybrid checkpoint. The *adaptation effect* τ_{ad} varies the selection schedule, comparing a per-decision selection policy against fixed-pair controls within an arm at matched training. In each contrast the remaining factors are identical: readout and selection policy for τ_{tr}, trained model and selection policy for τ_{ex}, checkpoints and readouts for τ_{ad}.

Classification is mechanical. For any contrast, examine what differs between its two sides: the training recipe alone gives a training effect, the readout at a shared checkpoint gives a symbolic-execution effect, and the selection schedule at shared checkpoints gives an adaptation effect. Under this rule, every model-comparison claim in the paper attaches to exactly one estimand: Table 1 reports the training effect of symbol-supervised training at fixed requests and the symbolic-execution effect at a fixed horizon inside one hybrid checkpoint, and Section 6 scores the request at one fixed checkpoint. The taxonomy extends beyond this program: any evaluation with a carrier, supervised readouts, and a selection policy can classify its contrasts the same way.

F Implementation Details

Encoder and carrier. The 13 features of each slot are its presence probability, kind index and kind confidence, center position, velocity from the prior frame, expected-kind probability, and availability flags. The parser was trained once, on all 3,000 training lineages, and none of the calibration lineages was used for fitting the dynamics.

Predictor. The horizon features are $\phi(\Delta) = [\sin(\Delta\omega_j), \cos(\Delta\omega_j)]_{j=0}^{7}$ with $\omega_j = 10000^{-j/8}$, the description embedding is $e_\alpha \in \mathbb{R}^{128}$, and the code is $c(r) \in \mathbb{R}^{128}$. Each MLP_ℓ is a two-layer SiLU block, and γ_ℓ, β_ℓ are one zero-initialized linear map of c per block. The launch $a \in \mathbb{R}^5$ holds the normalized drag release, hold and tap times, and a constant. The micro head symmetrizes contact and keeps support directed; both are gated by the presence of both slots and by the head's availability output. The adapters are bias-free linear maps $650 \to 384$ (micro) and $4 \to 384$ (macro). The hybrid has 1,837,690 parameters. Counting linear layers only, one call costs 1,432,448 multiply-accumulates under cont, 1,795,456 under micro and 1,464,704 under macro, independent of Δ; the policy adds 32,128.

Training. Each model trains for 9,000 updates of batch 64 with AdamW (learning rate 10^{-4}, weight decay 10^{-4}, gradient clip 1.0), and the last update is kept, with no best-validation selection. The symbolic loss uses positive weight 5 on the predicate logits plus a cross-entropy on the availability output, with unavailable frames masked.

Request policy. Each of the two rounds runs 1,800 updates of batch 128 (learning rate 10^{-3}) on the first window of each shot from 603 policy lineages. The compute term of Equation 6 counts linear multiply-accumulates of the predictor and the policy, and its normalizer is the predictor-only continuous transition at $\Delta = 15$. The hybrid policy (width 128) has 32,265 parameters.

Baseline. The baseline width is the width in $\{384, 392, \ldots, 512\}$ whose parameter count is nearest the hybrid's, with ties to the smaller width; its policy width, 131, is chosen by the same rule against the hybrid policy (32,229 parameters). Parameter counts are matched.

Training used for the compute comparison. The compute comparison of Section 1 uses an earlier training of both models by the same recipe on the 2,400 base-corpus lineages alone, with 600 policy lineages and the same three seeds.

G Reproduction Commands

Activate the environment before running any command: `conda activate novphy && source env.sh`, where `env.sh` puts the repository root on `PYTHONPATH`. All commands below run from the NovPhy repository root. The recorded source revisions are `env@254720f` for the #74 and #77 records and commits `0a0dd73` plus `ac2a4e1`, `8399f57`, and `9e369c4` for the #79 line. Each `--validate` mode recomputes the published tables from the retained artifacts and must exit 0. The validators never retrain a model. The external-baseline validator was verified on both cpu and cuda. Table 5 lists one row per artifact family.

Table 5: Exact validation commands per artifact family, with artifact roots relative to the NovPhy repository root.

Artifact family	Root(s)	Command
#74 matched systems	`.local-artifacts/issue-74-matched-dynamics-v1/`	`python -u -m scripts.run_issue_74_matched_dynamics --validate`
#77 N1 campaign	`.local-artifacts/issue-77-n1-v1/`	`python scripts/run_issue_77_n1.py --validate`
#77 N1 training and diagnostic	`.local-artifacts/issue-77-n1-dynamics-v1/`, `.local-artifacts/issue-77-n1-diagnostic-v1/`	`python -u -m scripts.run_issue_77_n1_diagnostic --validate`
#79 CLEVRER boundary replication	`.local-artifacts/issue-79-clevrer-boundary-v1/`	`python -u -m scripts.run_clevrer_boundary_replication --validate`
#84 third family	`.local-artifacts/issue-84-third-family-v1/`	`python -u -m scripts.run_third_family_boundary_replication --validate`
#92 lookahead control	`.local-artifacts/issue-92-lookahead-control-v1/`	`python -u -m scripts.run_lookahead_control --validate`
#96 within-checkpoint `tau_ad`	`.local-artifacts/issue-96-tau-ad-within-checkpoint-v1/`	`python -u -m scripts.run_tau_ad_within_checkpoint --validate`

The evaluation validators recompute every published table without new inference. The CLEVRER module records its own exact validation command in its module documentation, `python -u -m scripts.run_clevrer_boundary_replication --validate`, which recomputes every published table and exits 0.

The kept runners bind a frozen `plan.json` with sha256-pinned inputs, write per-cell markers and receipts to a resumable ledger, and record every unit that does not complete as a typed failure. The small-artifact subset of every root in Table 5 (plans, summaries, comparisons, findings, records, receipts, and ledgers) is version-controlled through commit `2b4d897`, and the #96 root through `0e7f921c`. Frames and native chunks, about 15 GB per pool, are excluded.

H NUMBER-TO-ARTIFACT AUDIT

Table 6 maps every number printed in this paper to the artifact that holds it. Rows group numbers by source. Paths are relative to the NovPhy repository root, and each root validates with the command of Table 5.

Table 6: Number-to-artifact audit. "Value as printed" lists every number the paper prints for the claim, in the paper's rounding.

Claim	Value as printed	Artifact path	Ticket
Table 1 (rollout error at the last endpoint)	Recursive position MSE, three-seed mean. NovPhy, 225 frames: baseline $(1, 5, 15, \mathsf{cont}) = 2224$, 0.281, 0.0370; hybrid 3141, 0.200, 0.0296, plus $(15, \mathsf{micro}) = 0.0313$ and $(15, \mathsf{macro}) = 0.0397$. CLEVRER, 120 frames: baseline 13.5, 0.0471, 0.0116; hybrid 2.98, 0.0444, 0.0170, plus $(15, \mathsf{micro}) = 0.0190$. Physion-Dominoes, 120 frames: baseline 182.6, 0.0176, 0.00114; hybrid 167.4, 0.0156, 0.00128, plus $(15, \mathsf{micro}) = 0.00199$	`.local-artifacts/issue-77-n1-diagnostic-v1/summary.json`, `.local-artifacts/issue-79-clevrer-boundary-v1/summary.json`, `.local-artifacts/issue-84-third-family-v1/summary.json` (`per_seed` curves `position_mse` for `continuous_h1/h5/h15`, `hybrid_continuous_h1/h5/h15`, `hybrid_micro_h15`, `hybrid_macro_h15`)	#77, #79, #84
Figure 1 (real cascade, imagined requests, recursive error curves)	Start, video frames 80/120 and final video frame 554 map to carrier frames $0, 80, 120, 553.38$ (native steps $30000, 34000, 36000, 57669$, origin 30000, stride 50); the final photo is the engine-window endpoint, not the imagined endpoint 600; the request trace comes from a decision-only imagined rollout (candidate 6: modal 1-continuous, $29/94$ queries); F(5,macro) rank among nine in launch-angle, offset, drag-grid and launch-power order $1, 1, 9, 4$; $J - F^*$ AUC differences $+.0595$, $+.0037$, $+.0700$, $+.0273$	`data/issue-87-closed-loop-oracle/manifest.json` (state/ordinal frame pointers); `.local-artifacts/issue-87-closed-loop-oracle-v1/attempts/oracle--issue-77-n1-001-a00--a06/shot-1/observation-trace/observation_trace_manifest.json` (frame_records fixed_step); `.local-artifacts/issue-92-lookahead-control-v1/records/decision--hybrid-adaptive-e600--seed20260908--issue-77-n1-001-a00.json` (decision.horizon_trace["6"]); `.local-artifacts/issue-96-tau-ad-within-checkpoint-v1/comparisons.csv`	#87, #92, #96
Section 6.2 and Figure 4 (per-set request ranking)	Ordering AUC per set: $(5, \mathsf{macro})$ 0.63 (angle) and 0.59 (offset) and 0.40 (drag grid, last of nine), $(1, \mathsf{macro})$ 0.51 (drag grid) and 0.53 (launch power); best-minus-worst spread 0.44–0.51; $r^\dagger$ trails the set's best request by 0.03–0.13	`figures/build_results_grid.py` over `.local-artifacts/issue-96-tau-ad-within-checkpoint-v1/comparisons.csv` (`auc_F-*`, `spread`)	#96
Section 6.3 (policy against the transferred fixed request)	$J - r^\dagger$: $+0.060$ (angle sweep), $+0.004$ (offset sweep), $+0.070$ (drag grid), $+0.027$ (launch power)	`.local-artifacts/issue-96-tau-ad-within-checkpoint-v1/comparisons.csv` (`inventory_J-F*`)	#96
Section 6.3 (compute)	Ranking regret 0.350 (hybrid adaptive) against 0.367 (horizon-only adaptive); 118.7M against 356.0M linear MACs per candidate and state; 1.0 s against 2.0 s per state; 80 against 189 predictor calls per candidate and state	`.local-artifacts/issue-74-matched-dynamics-v1/readiness.json`	#74

Claim	Value as printed	Artifact path	Ticket
Section 6.4 (request schedules)	12 candidates receive eight schedules; the successful launch a6 is ranked sixth of 12 by the policy, last by $(1, \mathsf{macro})$ and first by $(5, \mathsf{macro})$; 2,634 imagined rollouts; 77% of level–seed pairs differ between candidates; $\Delta = 5$ share $2\% \to 16\%$; macro share $59\% \to 83\%$	`.local-artifacts/issue-96-tau-ad-within-checkpoint-v1/records/` (per-cell `decision.schedules`), aggregated and plotted by `figures/build_qualitative_requests.py`	#96
Ranker architecture	depth-3 predictor, 236-dimensional carrier; 1,837,690 (hybrid) and 1,862,396 (continuous) parameters; 9,000 updates, batch 64; 2,409 fitting lineages	`.local-artifacts/issue-77-n1-dynamics-v1/plan.json` (`contract.config`, `training`, parameter-count block)	#77

I Reproducibility Statement

Section 5 defines the evaluation protocol, Appendix B records its mechanics and typed-failure taxonomy, and Appendix G gives the validation command for every cited artifact root. The records behind Sections 6 and 7 recompute from retained artifacts: the #74 matched-systems runner, the #77 N1 training and diagnostic runner, the CLEVRER replication runner, the third-family runner, the #92 lookahead-control runner, and the #96 within-checkpoint runner, each rerun as `python -u -m scripts.{runner} --validate`, with module names and commands listed in Table 5. Each run binds a frozen `plan.json` with sha256-pinned inputs and writes per-cell markers and receipts to a resumable ledger, and every unit that does not complete ends as a typed failure of a recorded kind. The small-artifact subset of every cited root (plans, summaries, comparisons, findings, records, receipts, and ledgers) is version-controlled through commit `2b4d897`, and the within-checkpoint root through `0e7f921c`. Frames and native chunks (about 15 GB per pool) are excluded. Table 6 maps every printed number to its artifact path.

J Ethics Statement

This work uses simulated physics environments only: NovPhy scenarios and the CLEVRER annotation release, which is distributed under a CC0 license. It involves no human subjects, personal data, or deployed systems.