RULES ARE RARE: WHEN WORLD MODELS MEMORISE RATHER THAN ACQUIRE RULES

Anonymous authors
Paper under double-blind review

ABSTRACT

World models are trained to predict the next state, and by that objective they also learn the rules of their environment. We show that the world models we train learn them by memorising. On Craftax, a transformer world model (TWM) is as accurate on a rule before the episode has shown it as after (88% against 89%), and when the rules are resampled each episode it falls to the accuracy of predicting that nothing changes. Training on changing rules does not repair this: on XLand-MiniGrid, a Long-Context Transformer model that sees the whole episode memorises one rule table per training episode instead; updating the weights does, by fine-tuning, replay, hidden-parameter inference or online test-time adaptation (AdaJEPA), at three separable costs: interference, forgetting and erosion. The hidden parameter in these environments is not one vector: it splits by interaction type, and the interaction is observable while its effect is not. So we introduce PRISMO. It keeps one entry per interaction type in a store outside the frozen host, writes it from a single observation with no gradient step, and clears it when the rules may change. On two frozen Craftax hosts it raises rule-state accuracy from 29% to 87% (TWM) and from 24% to 79% (a long-context Transformer) while every other predicted token is unchanged, and on XLand two rules observed once each compose into a chain never seen executed.

1 INTRODUCTION

World models are trained to predict the next state (Hafner et al., 2024; Ha & Schmidhuber, 2018; Micheli et al., 2023; Dedieu et al., 2025). Under this objective they also pick up the rules of their environment: which potion heals, which two objects combine into which third. When the rules are fixed, this works well, and a well-trained model predicts the effect of each rule accurately. But it stores the rules in its weights. In both of our settings, its accuracy before an episode has shown a rule is the same as after (Figure 1): it never needs to see the rule in action. When the rules change, it fails at once: its accuracy on the variables a rule governs (its *rule-state accuracy*, defined in §2) falls to that of predicting no change at all (§3.1). This is the "average model" of the hidden-parameter MDP literature: behaving as if it had no variable for the hidden rule, the model fits one transition function to every rule set at once (Doshi-Velez & Konidaris, 2013; Killian et al., 2017). For discrete rules the mode of that average is "no change", which is the starting point of our diagnosis (§4). Adapting to new rules at test time is an important capability for world models (Wang et al., 2026b).

A natural fix is to train on environments whose rules change every episode, so that the model must infer the current rule from context. Recent work shows that, given enough diverse environments and a long enough context, world models switch from recognising a known environment to learning a new one in context (Wang et al., 2026a). We find that this does not happen for rules, even when the whole episode is in context (§3.1). Sparsity offers one possible explanation: dynamics provide evidence at every step, whereas a rule takes effect only once or twice per episode. A second problem, which to our knowledge has not been posed before, follows from the first. Rules should also compose: the product of one should be recognised as the ingredient of the next in a chain the model has never seen executed. A model that memorises rules cannot do this, because it has nothing to apply outside the case it memorised (§5.4). This paper asks three questions. What stops a world model from acquiring new rules? Why do the existing remedies fail? And what works instead?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

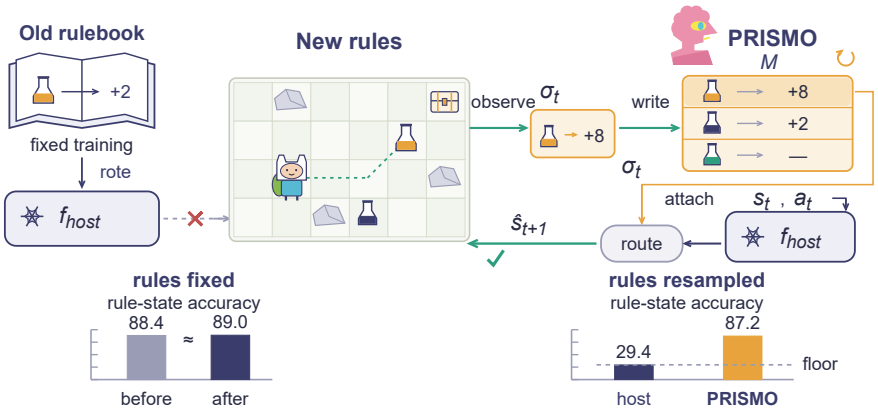


Figure 1: **Rules in the weights, or in a store.** Left: with rules fixed, TWM is as accurate before a rule is observed as after (88.4 ≈ 89.0). Right: with rules resampled it predicts no change (29.4, floor 29.6); PRISMO keeps the host frozen and holds each rule in a store M addressed by the interaction σ_t, written from one observation, read thereafter (87.2), cleared when the rules may change. Right-hand values: Table 3.

Existing adaptation changes the parameters, and pays for it. Longer context does not help (§3.2). What does help is changing weights: fine-tuning, the hidden-parameter MDP approach of inferring a latent task variable while also updating the dynamics model (Killian et al., 2017; Perez et al., 2020), and test-time adaptation of world models such as AdaJEPA (Wang et al., 2026b). None of them separates the rule-governed quantities from the rest of what the model knows, so what adapts is the same set of parameters that carries everything else, and adaptation pays up to three separable costs (§3.2). *Interference* with the next rule set, *forgetting* of the earlier ones, and *erosion* of general competence; replay repairs only the second.

Taking the rule out of the parameters. In these environments the hidden parameter is not one vector. It splits by interaction type, and the interaction — what acts on what — is visible even when its effect is not (§4). **PRISMO** (Figure 1) uses the interaction as an *address* into a store kept outside the parameters of the host world model (§2), and fills in the *value* from observation: one observation suffices, the store is cleared when the rules may change, and the host stays frozen. An entry written from one observation composes: two rules seen once each form a two-step chain the model has never observed, with the model recognising its own predicted product as the next rule's ingredient, and the chain breaks exactly where a rule is withheld (§5.4).

Contributions. (i) *A diagnosis*: at the corpus sizes we train, a pretrained world model does not read in-episode evidence — it lands on the predict-unchanged floor, chooses at random at a decision, and under per-episode rule diversity memorises at the granularity of the episode instead — made visible by a protocol that scores each transition type against a predict-unchanged floor and an encoder ceiling. (ii) *An account of why adaptation by weights is not enough*: it pays interference, forgetting and erosion, and none of the weight-updating remedies we test removes all three. What removes them is a principle, not an implementation: take the rule out of the weights and address it by interaction type. A latent built on both removes them too, at about 200 gradient steps per observation; PRISMO does so with none. (iii) *PRISMO*, which acquires a rule from one observation without a gradient step. (iv) *Composition*, a test we introduce: two rules observed once each compose into a two-step chain the model has never seen executed (94.8%), a negative control shows the chain breaks when one rule is withheld (0.6%), and every model without in-episode memory scores below 2%.

2 Instrument

A model with the rule in its weights and one that reads it from evidence differ only when the rule is not the one they were trained on; our apparatus is built around that manipulation (details in Appendix A.1).

Fixed or resampled. Each environment is parameterised by a rule set, drawn at the start of an episode, held constant within it, and never observed. Under FIXED, one rule set is reused for every episode, while layouts, placements and trajectories still vary; under RESAMPLED, a fresh rule set is drawn each episode and nothing else changes. In both we report **rule-state accuracy**: on the transitions where a rule takes effect, the accuracy of predicting the variables that rule governs (the *rule* column of every table; *other* is accuracy on all remaining transitions). We read it *before* the episode has shown the rule and *after*. A model that memorised the rule is equally accurate in both; a model that acquires it is at chance before and accurate after.

Two rule forms. In **Craftax** (Matthews et al., 2024), each consumable item (a potion, identified by its colour) has an effect on the agent's attributes, resampled per episode: *entity* $\rightarrow$ *scalar increment*, independent rules that take effect many times. We refer to a Craftax rule by its potion's colour. It has a published world-model architecture (TWM, §2), so we diagnose an architecture we did not design. In **XLand-MiniGrid** (Nikulin et al., 2024), placing two objects side by side produces a third: *entity pair* $\rightarrow$ *entity*, rules that *compose* and take effect rarely, which lets us ask whether an entry written from one observation composes into a chain never seen executed (§5.4).

Two hosts on Craftax. A *host* is a world model we diagnose and, in §5.3, attach PRISMO to; every host is trained on our Craftax corpus before any experiment and frozen thereafter. The first is the transformer world model of Dedieu et al. (2025), with the architecture used unchanged and a 20-step context; we call it TWM, as they do. Their released checkpoint is for Craftax-Classic (9×9 token grid); our environment is Craftax with potions (13×11), so we train the host ourselves (Appendix A.2). The second is a long-context Transformer of our own (Micheli et al., 2023; Robine et al., 2023), on the same tokeniser and predicting the same tokens, with a 256-step context that covers roughly ten repeat encounters of a rule. XLand has no published world model; the models there are ours (§5.4).

Adaptation baselines. Every remedy in §3.2 is applied to these same frozen hosts, so that only the adaptation rule varies. For AdaJEPA (Wang et al., 2026b) we reproduce its adaptation protocol (a gradient step on a few layers of the host at every step, a short buffer, a reset at the episode boundary) on our hosts, not its full model; the reason and the selected configurations are in Appendix A.2.

Three layers of measurement. Whole-frame accuracy is dominated by state that stays the same, so we report accuracy *by transition type*, grouped by what causes a variable to change (§3.1), with a PREDICT-UNCHANGED *floor* and an *encoder ceiling* (the true next observation through the same frozen perception) for every group (Appendix A.1).

Collection-time modifications. In their default configurations rule events are too rare to measure, so we raise their density at collection time (Appendix A.3). All methods see the identical corpus, the rules are untouched, and every change increases a baseline's exposure to rules, so it can only make our negative result harder to obtain; our numbers are not comparable to published ones.

3 WHAT GOES WRONG, AND WHY THE OBVIOUS REMEDIES DO NOT FIX IT

3.1 SYMPTOMS

A reasonable expectation, which we call H0 and which we held ourselves, is that a world model trained where the rules keep changing learns to work out from its observations which rules are currently in force: once no single rule set is right across training, memorising one should stop paying. Both Craftax hosts and the XLand models are trained this way, with the rules redrawn every episode. This section shows that H0 fails; §3.2 shows that the remedies available today do not repair it. We begin in Craftax (§2), where each consumable item shifts one of the agent's attributes by an amount resampled per episode. With the rules held fixed, the host's rule-state accuracy (§2) is 88.44%, against a PREDICT-UNCHANGED floor of 3.77%, and it is as accurate before an episode has shown the rule as after (88.44% against 88.96%; Figure 1): the rule is in the weights. Once the rules are resampled per episode, accuracy after a rule's first observation falls to 26.93%, 0.06 points below the 26.99% of predicting no change (29.4 against 29.6 on the common set of Table 3, the values in Figure 1); the apparent in-episode gain ($5.80\% \rightarrow 26.93\%$) is the floor moving ($5.39\% \rightarrow 26.99\%$), and predictions not governed by a rule are unaffected (88.41%, floor 89.00%). The model is not weak. Consuming an item both reduces its count, the same in every episode, and changes an attribute by the

Table 1: **Craftax: at a decision, the host is indistinguishable from choosing at random.** Decision set D1, horizon h=1.

	picks the oracle's best candidate	regret	harmful choices	mean tied candidates
Oracle (true mapping)	*100 %*	*0*	*n/a*	*n/a*
random mapping + **PRISMO**	**61.6%** [46.9, 76.1]	0.754	**6.4%**	1.15
fixed-mapping host	45.5% [35.4, 56.9]	1.80	15.8%	1.58
random mapping + fine-tune	27.5% [21.2, 34.5]	2.72	47.2%	4.00
random-mapping host	27.2% [20.7, 34.3]	2.83	48.6%	5.21
random policy	*25.8%* [20.5, 31.9]	*2.86*	*49.5%*	*1.00*

Table 2: **XLand-MiniGrid: a second rule form, and the ceiling of memorisation.** Left: the best a memorising strategy can do, by corpus. Right: every route without in-episode memory, resampled rules; $\pm$ is the s.d. over three seeds.

Memorisation optimum

Corpus	`pair-lookup`
fixed	100.00
resampled	3.01

Resampled corpus, no in-episode memory

Route	first	after first
PREDICT-UNCHANGED	0.00	0.00
MAJORITY-PRODUCT	0.44	0.66
`pair-lookup` (train-time)	2.20	3.61
long-context Transformer	2.04 ± 1.08	1.29 ± 0.30
our model, read path zeroed	1.31 ± 1.14	1.52 ± 0.43
encoder ceiling	*99.6–100*	*100.00*

episode's rule; the host predicts the first 66 points above floor and the second 0.06 below it. It learned the half that could be memorised. The failure carries into decisions. At a Craftax decision point the model imagines acting on each of several objects, rolls each forward h steps, and the best-rated object is chosen (Table 1; protocol in Appendix A.13). Table 1 is condition D1 at h=1, where PRISMO's store is pre-filled with all six colours from their real first observations. The host picks the oracle's choice 27.2% of the time and a random policy 25.8%, because it predicts the same outcome for 5.21 of a median six candidates and so produces no ranking. The fixed-mapping host, which holds the rule in its weights, ties 1.58 and picks the oracle's choice 45.5% of the time: the decision fails exactly when the model has no access to the rule. More varied rules over a longer window do not help. On Craftax the long-context Transformer, whose 256-step window covers about ten repeat encounters of a rule, sits below its floor like the host (24.1 against 25.2; Table 3). In XLand-MiniGrid (§2), where a rule combines two objects into a third and a long-context Transformer sees the whole episode, a lookup table built from the training set scores 100.00 on fixed rules and 3.01 on resampled ones, although 69% of test pairs occurred in training: the entry is found, and it is stale (Table 2). The Transformer, with 1.63× our parameters and 12.6× our data per gradient step, scores 1.29 after a rule's first observation and 2.04 before it, yet 91% on its own training episodes before the rule has been shown at all (Appendix A.10): it recognises the episode from its scene and recalls that episode's rules. Diversity made it memorise one rule table per episode; the encoder ceiling (99.6–100) rules out perception. A memorised rule also cannot be used beyond the case it was memorised for: on a two-step chain, where the product of one rule is the ingredient of the next, every model without in-episode memory scores below 2% (§5.4).

3.2 WHY THE AVAILABLE REMEDIES FALL SHORT

Longer context does not help. Extending the host's context twentyfold, to its architectural limit, moves rule-state accuracy from 29.4% to 30.0%, while accuracy on all other transitions rises by 1.8 points (Table 3): the added context is used, but not for rules. Widening the Transformer's evaluation window from one step to its whole 256-step domain moves it from 23.9% to 24.1%, still below its floor of 25.2%, although within that domain the previous observation of the same colour lies a median 21 steps back. The XLand Transformer, which sees the whole episode, is no different (Table 2; Appendix A.8).

Table 3: **Are the available remedies enough?** Craftax. *Rule*: rule-state accuracy on the events every route shares (1,764 on TWM, 508 on the Transformer). *Other*: paired change (pp) on all other transitions against the frozen host. TWM ±: episode-bootstrap s.e.; Transformer ±: s.d. over three seeds. Replay: Figure 2, Table 4b. Notes, including *rules fixed*, in Appendix A.17.

	TWM host		**Long-context Tf**		
Route	rule	other	rule	other	cost
no adaptation	29.4 ± 2.0	—	23.9 ± 1.2	—	—
+ longer context	30.0 ± 1.9	$+1.8$	24.1 ± 1.5	$+0.5$	0 steps
+ fine-tune	54.8 ± 1.9	-6.9	25.9 ± 1.5	-36.9	200 / 50 steps
+ fine-tune (head)	39.1 ± 1.8	-0.7	29.3 ± 0.7	-0.1	50 steps
+ online TTA	30.4 ± 1.9	-0.8	32.3 ± 2.5	-5.3	201 / 386 steps/ep.
+ HiP-MDP, published	85.8 ± 1.2	-51.6	n/a	n/a	∼580 steps/obs.
+ HiP-MDP, frozen, per-type latent	86.8 ± 1.2	0	n/a	n/a	∼200 steps/obs.
+ episode-local store	72.7 ± 2.7	0	58.7	0	0 steps
+ **PRISMO**	$\mathbf{87.2 \pm 1.5}$	**0**	**79.3**	**0**	**0 steps**
floor	*29.6*		*25.2*		
rules fixed	*88.96*		*74.71*		pretrained

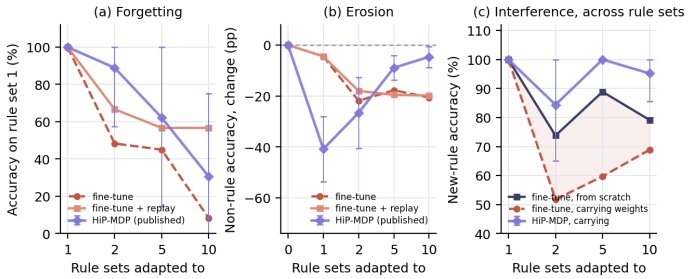
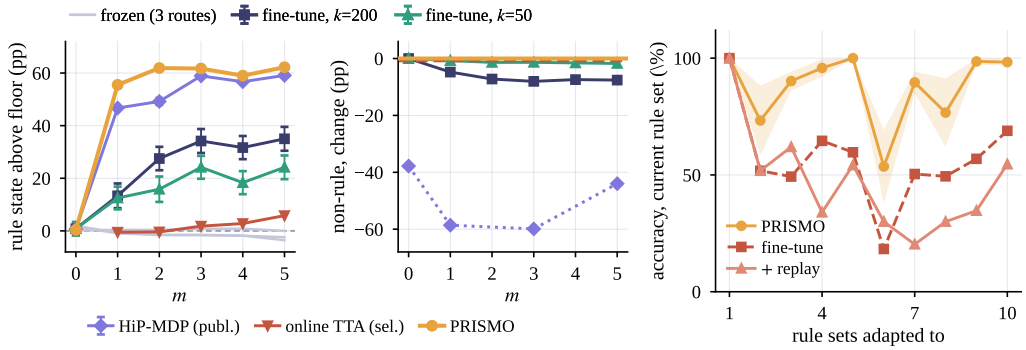


Figure 2: **Three costs of adapting by weight update.** Fine-tuning with and without replay, and the published HiP-MDP method, which also updates the dynamics model's weights.

Changing the weights works, at three separable costs. Fine-tuning the host on the rule events observed in an episode reaches 54.8% rule-state accuracy (200 gradient steps; Table 3) and loses 6.9 points on all other transitions; more steps make both worse. Over a sequence of rule sets the cost separates into three parts (Figure 2). *Interference*: a model that already holds one rule set learns the next one 10 to 29 points worse than a fresh copy. This answers a stronger form of H0: training on rules did not teach the model to acquire them. *Forgetting*: accuracy on the first rule set falls from 100% to 8.3% by the tenth, although the rule sets differ only in the mapping. *Erosion*: accuracy on all other transitions falls from 86.5% to 65.9%, most of it within the first two adaptations. Replay repairs forgetting in part (56.7% on the first rule set) but not erosion (66.7%), so erosion is a separate cost. Updating only the head buys little (39.1% at 50 steps, 50.0% at 200) and erodes by 200 steps (-0.7 and -4.7 points on other transitions). The long-context Transformer fails in the same way although its predictions on other transitions clear their floor on the full set (92.26 against 88.04); full fine-tuning costs it 36.9 points on other transitions for a 2-point gain on rules (Table 3). Online test-time adaptation in the manner of AdaJEPA (its protocol on our hosts, §2) takes a gradient step on one block of the host at every step of the episode. On the Transformer it reaches 32.3% after the first observation, at 386 gradient steps per episode and 5.3 points on all other transitions; on the full set that cost grows with the length of the episode, from 0.2 points in the first twenty steps to 11.9 after step 200, and before a rule has been observed the adapted model is worse than the frozen one ($7.3 \rightarrow 5.2$). On the TWM host the same protocol acquires nothing: 30.4% against 29.4% without adaptation for about 200 gradient steps per episode, with the logits unmoved ($0.755 \rightarrow 0.757$). TWM was pretrained on this buffer, so its updated layers already sit at a minimum of the adaptation loss (Appendix A.2).

Separating the rule into a latent helps, but not while the weights still move. The published HiP-MDP method infers a latent task variable while updating the dynamics model. The latent removes



(a) Exposure curves, TWM host, above each route's own floor.

(b) Accuracy on the current rule set, ten rule sets.

Figure 3: **Acquisition within an episode and across rule sets.** m: earlier observations of the colour in the episode. Notes in Appendix A.17.

interference: carrying it to the next rule set costs nothing (Figure 2c). But the weights still move, so the model still forgets (Figure 2a) and, within an episode, loses 38 to 60 points on other transitions while acquiring the rule (Figure 3a). Separating the rule from the weights removes interference; keeping the weights fixed removes the other two costs. The next section does both.

4 Taking the rule out of the parameters

We offer one account of the failures of §3 and test the prediction it makes. It has two parts. Models that do not update their weights do not acquire the rule because they behave as if the episode's history were absent; models that do update them pay the three costs because the rule shares parameters with everything else. In both, the models tested leave the history unused, so the rule is held only in parameters shared with everything else. The account is supported on the hosts and data regimes studied here; it is not a proof that no model or corpus could do otherwise.

Why the models fail. Our setting is a hidden-parameter MDP (Doshi-Velez & Konidaris, 2013): a family of MDPs whose dynamics depend on a hidden parameter θ, drawn at the start of each episode, fixed within it, and never observed. A potion mapping is one such θ. Given θ, all rules of the episode are known. Both hosts see the history $\mathcal{H}_t$ (20 steps; 256 steps), so the predictor they could in principle represent is the Bayes-optimal one,

$$p(s_{t+1} \mid s_t, a_t, \mathcal{H}_t) \;=\; \int p(s_{t+1} \mid s_t, a_t, \theta)\; p(\theta \mid \mathcal{H}_t)\, d\theta, \tag{1}$$

in which $p(\theta \mid \mathcal{H}_t)$ concentrates once the episode has shown a rule take effect, and the prediction is then correct. The trained models instead behave as if $p(\theta \mid \mathcal{H}_t)$ were the prior $p(\theta)$, as if the history did not exist: the mode of that marginal is "nothing changed", exactly where §3.1 found the host (26.93 against a floor of 26.99). Fixed rules are the special case in which θ has a single value, so prior and posterior coincide. We attribute the behaviour to what gradient descent finds, not to what the objective permits: a rule takes effect once or twice per episode, so reading it from the history earns almost no training signal, while identifying the episode and recalling its rules is rewarded at every step. The XLand Transformer's 90.8% on its own training episodes at a rule's first observation (Appendix A.10) is consistent with that solution, and holding the scene fixed does not change it, which suggests the model identifies episodes by whatever differs between them (Appendix A.11). At these corpus sizes the models do not learn to use the history; a larger corpus is untested (Appendix A.7).

The structure they do not exploit. Methods for HiP-MDPs infer θ as a single vector from the episode as a whole. In our environments θ has more structure than that. We call the combination of the entities involved and the action taken an *interaction type* σ (for instance: the agent, the third potion, *drink*), and write Σ for the finite set of such types. Each rule governs one type, so θ splits into one component per type,

$$\theta \;=\; \{\,\theta_\sigma\,\}_{\sigma \in \Sigma}, \qquad \sigma_t \;=\; g(s_t, a_t), \tag{2}$$

Table 4: **Gradient adaptation against PRISMO.** Same episodes, same rule events; **n/a** is by construction, not a favourable measurement.

(a) H1: m=1 on the E1 site sample, then each route at its best on the common set of Table 3.

	Route	Gradient steps	Rule state	Non-rule cost
$m = 1$	Fine-tune	200	39.15	-6.17
	PRISMO	**0**	79.6	**identical**
best point	Fine-tune	200	54.8	-6.9
	PRISMO (all)	**0**	**87.2**	**identical**

(b) H2: the three costs over ten rule sets, and what PRISMO does with each.

Cost	What it measures	Fine-tune	+ replay	**PRISMO**
Forgetting	accuracy on rule set 1, after ten	$100 \to 8.3$	$100 \to 56.7$	**n/a**
Erosion	non-rule accuracy, after ten	$86.5 \to 65.9$	$86.5 \to 66.7$	**86.5**, identical
Interference	new-rule acc., old weights vs. fresh	-10 to -29	-22 to -35	**n/a**
Adaptation cost	per rule set	200 gradient steps		**0 steps**

where g reads the interaction type at step t from the state and the action. θ_σ determines how the rule-state variables of type σ change (§2); we write $\mathcal{R}(\sigma)$ for that set of variables. Only θ_σ is hidden. σ_t is observable, because the entities and the action are part of the state and the input. The interaction type therefore serves as an address, and the rule is the value stored at it. Two properties follow. One clean observation of σ fixes θ_σ, with nothing left to infer. And updating θ_σ cannot disturb $\theta_{\sigma'}$ for $\sigma' \neq \sigma$. A single entangled vector has no such guarantee (§5.2).

The method. Figure 1 sketches the architecture. PRISMO keeps a store M with one entry per interaction type. The entry $M[\sigma]$ holds the current estimate of θ_σ, that is, of what rule σ does to the variables in $\mathcal{R}(\sigma)$: in Craftax, how much the potion shifts an attribute; in XLand, which object the pair produces. The store replaces the integral in (1):

$$\hat{s}^{\,i}_{t+1} \;=\; \begin{cases} h\big(\mathrm{read}(M[\sigma_t],\, v^i)\big) & \text{if } i \in \mathcal{R}(\sigma_t) \text{ and } M[\sigma_t] \text{ is filled,} \\ f_{\text{host}}(s_{t-k:t},\, a_t)^{\,i} & \text{otherwise.} \end{cases} \tag{3}$$

In words: once a rule has been observed, the variables it governs are predicted from the store and every other variable by the host, unchanged. Here v^i is the current value of variable i, and h maps the stored effect into the host's output format. The host is frozen throughout ($\partial\mathcal{L}/\partial\phi_{\text{host}} = 0$).

Each entry is an interval $[\ell_\sigma, u_\sigma]$ on the effect rather than a point, because an observation can be clipped: if a potion takes health from 7 to its ceiling of 9, the observed $+2$ is only a lower bound on the effect. An unclipped observation writes the exact effect, replaces whatever was stored, and makes the entry usable at once; a clipped one narrows the interval. The write and read rules are given in full in Appendix A.5 (Algorithm 1). No gradient step occurs anywhere, and an empty entry is itself the signal to observe and write, so no surprise detector is needed.

M shares no parameters with the host (§3.2): separate entries cannot interfere, and the host is never updated, so every variable outside $\mathcal{R}(\sigma)$ is predicted exactly as before. Separation has one cost: after the rules change, a stale entry is worse than an empty one, so M is cleared whenever θ may have been redrawn, here at the episode boundary. Two frozen-host HiP-MDP baselines optimise only θ: a single vector, which tests separate addressing (§5.2), and a latent per interaction type, which shares PRISMO's (§5.3).

5 New Hypotheses

5.1 H1: A separated θ_σ is acquired where gradients do not arrive

Table 4a compares fine-tuning and PRISMO at matched evidence and at their best. With one observation of a rule, fine-tuning reaches 39.15% after 200 gradient steps; PRISMO reaches 79.6% with none. At their best, on the events every route shares, fine-tuning reaches 54.8% and PRISMO

Table 5: **Interference on the frozen TWM host.** Drop in rule-state accuracy on colour B (points; positive means worse) after one and after five interposed observations of other colours. Control: the same gradient steps spent on B itself.

Arm	after 1	after 5
PRISMO	**0.00** [0, 0]	**0.00**
HiP-MDP, one latent vector		
published	+4.0	−0.66
frozen host, d=16	+0.2	+2.1
frozen host, d=2	+13.4	+35.7
incremental, d=16	+5.4	+20.3
control: same steps on B	−20.3 (B improves)	

Table 6: **The prediction follows whatever the store holds.** Interventions on PRISMO's store, TWM host; rule-state accuracy after a colour's first observation, over all 4,764 rule events.

Store contents	accuracy
Correct rule, from observation	**85.1**
Empty	26.9
Attached, write path disabled	26.9
Another colour's entry	8.0
Random vectors	15.6
Last episode's entries, not cleared	24.9

87.2%; PRISMO's figure pools all observations after the first (Figure 3a). Over a fifty-step closed-loop rollout with the same six observations, fine-tuning is no better than the untouched host on the rule-governed variable, while PRISMO cuts its error from 2.78 to 1.70 (Appendix A.9). Among routes that update the host's weights, the published HiP-MDP method comes closest (the frozen-host latent is in §5.3): at hundreds of gradient steps per observation (Table 3) it is below PRISMO for $m = 1$–4 and above it for $m \geq 5$ (89.8 against 87.4; Figure 3a), at a non-rule cost.

5.2 H2: A SEPARATED θ_σ CAN BE REPLACED WITHOUT HARMING WORLD UNDERSTANDING

Table 4b answers the three panels of Figure 2 in turn. Two of PRISMO's entries are not applicable by construction (Appendix A.17); the erosion entry is an identity, because the host is never updated. Accuracy on the current rule set does not reduce to one number and is plotted instead (Figure 3b). Replay does not remove interference on this host (−22.0, −34.7, −24.6 against a fresh copy at stages 2, 5, 10, worse than plain fine-tuning at 5 and 10); on the Transformer it does (59.7 against 61.0 at stage 10; caveat in Appendix A.17). Across the sequence PRISMO shows no downward trend: the slope is +0.18 pp per rule set (95% interval [−1.80, +2.23], sixty rule-set episodes), against −1.98 for fine-tuning and −4.56 with replay: no trend, not stage-to-stage stability (Appendix A.17).

Interference within an episode. This is the direct test of the decomposability claim of §4, that entries with separate addresses cannot disturb one another. We observe colour B, evaluate it, interpose observations of other colours and evaluate B again; a control re-observes B for the same number of gradient steps. The comparison arm is a single HiP-MDP latent optimised through the same interface on the same frozen host. Table 5 gives the drop after one and after five interposed colours. PRISMO's is 0.00 on every one of the same 983 problems. Interference appears when the latent is updated incrementally (+5.4, then +20.3, while the control improves B by 20.3) and when it is too small to hold every effect (d=2: +13.4, then +35.7). A frozen single-vector latent of adequate size interferes little (d=16: +0.2, +2.1), and the published method's prioritised replay, which keeps the first observation of B in its buffer, largely prevents it: its drop is +4.0 after one interposed colour and −0.66 after five, both at the noise level, so the protection is not confined to the first interposed colour. What separates PRISMO from the published method is therefore cost and erosion (Table 3), not interference.

5.3 H3: PRISMO ATTACHES TO A FROZEN HOST AND FILLS ITS BLIND SPOT

On both hosts PRISMO fills the blind spot at zero gradient steps and leaves every other token unchanged: on the events every route shares, rule-state accuracy rises from 29.4% to 87.2% on TWM and from 23.9% to 79.3% on the Transformer (Table 3). On TWM the two latent baselines are level with PRISMO within their intervals: 85.8% for the published HiP-MDP method and 86.8 ± 1.2% for a frozen-host variant that infers one latent per interaction type. Built on the same two principles, that variant also pays none of the three costs, but spends about 200 gradient steps per observation where PRISMO spends none; the published method, which updates the weights, spends about 580 and loses 51.6 points on other transitions by the end of an episode. Before a rule is observed, host and PRISMO predict identically (Appendix A.12). At the decision points of Table 1, PRISMO picks

Table 7: **One rule, then two.** Left: a single rule, the minimal contrast on identical data. Right: composition. $\pm$ is the s.d. over three seeds.

Panel A — a single rule

	after 1st
long-context Transformer	1.29 ± 0.30
ours, read path zeroed	1.52 ± 0.43
ours	99.27 ± 0.39

Panel B — composition

Condition	memory on	memory off	teacher
C1 both rules, two steps (n=859)	**94.8** [93.1, 96.1]	1.9	94.2
C2 parent only (n=476)	**0.6**	1.9	93.1
C3 child, one step (n=1054)	**97.1** [95.9, 97.9]	1.8	96.6

the oracle's choice 61.6% of the time (95% interval $[46.9, 76.1]$) with 1.15 ties; the point estimate is above the fixed-mapping host's 45.5%, but the intervals overlap.

The gain comes from the entry, not from the module. Table 6 keeps the model fixed and changes only what the store holds: another colour's entry, random vectors and last episode's entry left in place all fall below the empty store after the first observation (8.0, 15.6 and 24.9 against 26.9%), which is what the invalidation rule of §4 rests on. At the first observation a stale entry scores 18.8 against 5.8 for an empty store: with six colours it matches the new mapping about one time in six, which is chance, not knowledge. The prediction follows whatever the entry holds. A plain episode-local store with the same interaction address and the same overwrite interface already reaches 72.7 on the same events at no gradient cost (Table 3); writing clipped observations as bounds rather than values accounts for the remaining 14.5 points on TWM and 20.6 on the Transformer. The gain comes from where the rule is held and how it is addressed, not from an elaborate mechanism.

5.4 H4: AN ENTRY WRITTEN FROM ONE OBSERVATION COMPOSES

This test asks whether an entry written from one observation can be used in a state the model has never seen: as the input to a second rule, whose ingredient is the model's own predicted product. Craftax rules do not compose, so the test runs on XLand (Table 7). Panel A is the minimal contrast on identical data (full baselines in §3.1). Under the same budget the context model memorises episodes and PRISMO reads the rule. Panel B is the composition test. With both rules observed once each, the model executes a two-step chain it has never seen executed 94.8% of the time (C1). C1 is a rollout: the second step's input is the model's own first-step prediction, so the model must recognise its own predicted product as the next rule's ingredient. In the negative control C2, with only the parent rule observed, accuracy is 0.6%, while the same model handed the full rule table reaches 93.1%: the chain is not hard, the child rule is missing. A model trained on one fixed rule set scores 0% on every condition, the teacher row included (Appendix A.14). Writing the memory from observation costs nothing against being handed it (Appendix A.17).

6 RELATED WORK

Latent task parameters. Our setting is a *hidden-parameter MDP* (Doshi-Velez & Konidaris, 2013): the dynamics depend on a parameter drawn once per instance and fixed within it, to be inferred rather than averaged over; later work infers a deep latent at test time (Killian et al., 2017; Perez et al., 2020). Ours follows it and differs in three ways. The latent there is a low-dimensional vector describing an instance as a whole, and inference relies on nearby parameters producing nearby dynamics; a rule set relabelled each episode has no such geometry. Inference there is posterior inference or optimisation over a batch; ours is one write at first observation, addressed by interaction type. Their estimate is kept; ours must also be discarded on a schedule. What is ours is the finding that a published architecture does not perform the inference at all, and the measurement that shows it.

Adaptation and change. In-episode adaptation without weight updates is a target capability (Adaptive Agent Team et al., 2023; Lee et al., 2023; Grigsby et al., 2024; Laskin et al., 2022; Gao et al., 2025), and forgetting under change is a recognised obstacle (Liu et al., 2025; Pan et al., 2025), which one line of work avoids by construction (Liu et al., 2025); we keep the host and move the rules out (§3.2).

External memory. Memory for a predictor is not new (Blundell et al., 2016; Goyal et al., 2022; Pritzel et al., 2017); we claim only that an interaction address makes one observation sufficient and that a rule store needs a discard rule.

REPRODUCIBILITY STATEMENT

The environments, hosts and adaptation baselines are described in §2, with the full experimental setup, model configurations, training settings and collection-time modifications in Appendices A.1–A.3; PRISMO's write and read rules and pseudocode are in Appendix A.5, and the protocol and evaluation set behind every main-text table and figure are in Appendix A.17. Every number in the paper is read from a logged result file. Code, collection configurations and checkpoints will be released with the camera-ready version.

USE OF LARGE LANGUAGE MODELS

Large language models were used to assist with writing the text and with preparing figures and tables. All data in the paper come from the authors' experiments; no result was generated or altered by a language model, and the authors take full responsibility for every claim, number and citation.

REFERENCES

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control, 2016. URL https://arxiv.org/abs/1606.04460.

Antoine Dedieu, Joseph Ortiz, Xinghua Lou, Carter Wendelken, Wolfgang Lehrach, J Swaroop Guntupalli, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Improving transformer world models for data-efficient rl, 2025. URL https://arxiv.org/abs/2502.01591.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations, 2013. URL https://arxiv.org/abs/1308.3513.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions, 2025. URL https://arxiv.org/abs/2503.18938.

Anirudh Goyal, Abram L. Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, and Charles Blundell. Retrieval-augmented reinforcement learning, 2022. URL https://arxiv.org/abs/2202.08417.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents, 2024. URL https://arxiv.org/abs/2310.09971.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Taylor Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden-parameter markov decision processes, 2017. URL https://arxiv.org/abs/1706.06544.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation, 2022. URL https://arxiv.org/abs/2210.14215.

Jonathan N. Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning, 2023. URL https://arxiv.org/abs/2306.14892.

Zichen Liu, Guoji Fu, Chao Du, Wee Sun Lee, and Min Lin. Continual reinforcement learning by planning with online world models, 2025. URL https://arxiv.org/abs/2507.09177.

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, 2024. URL https://arxiv.org/abs/2402.16801.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models, 2023. URL https://arxiv.org/abs/2209.00588.

Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Artem Agarkov, Viacheslav Sinii, and Sergey Kolesnikov. Xland-minigrid: Scalable meta-reinforcement learning environments in jax, 2024. URL https://arxiv.org/abs/2312.12044.

Minting Pan, Wendong Zhang, Geng Chen, Xiangming Zhu, Siyu Gao, Yunbo Wang, and Xiaokang Yang. Continual visual reinforcement learning with a life-long world model, 2025. URL https://arxiv.org/abs/2303.06572.

Christian F. Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps transferable model-based rl in a handful of trials, 2020. URL https://arxiv.org/abs/2002.03072.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control, 2017. URL https://arxiv.org/abs/1703.01988.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions, 2023. URL https://arxiv.org/abs/2303.07109.

Fan Wang, Zhiyuan Chen, Yuxuan Zhong, Sunjian Zheng, Pengtao Shao, Bo Yu, Shaoshan Liu, Jianan Wang, Ning Ding, Yang Cao, and Yu Kang. Context and diversity matter: The emergence of in-context learning in world models, 2026a. URL https://arxiv.org/abs/2509.22353.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model, 2026b. URL https://arxiv.org/abs/2606.32026.

A Appendix

The appendix gives the experimental setup in full (§A.1), the hosts and adaptation baselines (§A.2), the environment modifications (§A.3), the design notes and pseudocode for PRISMO (§A.5), ablations and controls, the closed-loop rollout and decision protocols, and the notes to every main-text table and figure (§A.17).

A.1 Experimental setup in full

The manipulation: rules held fixed, or resampled. Each environment we use is parameterised by a rule set, drawn at the start of an episode, held constant within it, and never observed by the model. We run every experiment under two regimes that differ in exactly one respect:

Fixed. One rule set is sampled once and reused for every episode in training and evaluation. The world is still regenerated each episode — layouts, object placements, and trajectories all vary — so the model faces the usual diversity. Only the rules stand still.

Resampled. A fresh rule set is drawn for each episode. Everything else is identical, including the data collection procedure and the training budget.

Under Fixed, a rule is a stable fact about the world and can legitimately be stored in weights. Under Resampled, it cannot be: no setting of the parameters is correct across episodes. The contrast isolates the variable we care about, and it does so without changing the architecture, the objective, the data volume, or the observation distribution.

Two readings are taken in both regimes, and their relationship is what makes the instrument work. We measure accuracy on transitions where a rule takes effect *before* that rule has been observed earlier in the same episode, and *after*. A model that has memorised the rule is equally accurate in both — it never needed the observation. A model that acquires the rule is at chance before and accurate after. **The gap between these two readings is the measurement.** A single accuracy number, however high, cannot distinguish the two.

Two environments, two rule forms. We use two environments whose rules differ structurally, so that our conclusions are not artefacts of one way of encoding a rule.

Craftax associates each of several visually distinct consumable items with an effect on the agent's attributes. The mapping from item to effect is resampled per episode by the environment's own design. A rule here is *entity* $\rightarrow$ *scalar increment*, rules are mutually independent, and a rule may take effect many times in one episode.

XLand-MiniGrid uses production rules: placing two objects of specified types adjacent to one another consumes both and produces a third. A rule here is *entity pair* $\rightarrow$ *entity transformation*; rules *compose*, since the product of one may be the input to another; and a rule consumes its ingredients, so it takes effect rarely. Rule sets are drawn from the environment's pre-sampled pool, and we resample per episode explicitly in the collection loop — the environment supports this but does not do it by default.

These differences matter for what each environment can establish. Craftax has a published world-model architecture (TWM), so the diagnosis runs on an architecture we did not design as well as on our own. XLand's rules compose, so it can answer a question Craftax structurally cannot: whether an entry written from one observation composes into a chain never seen executed. We allocate the experiments accordingly, and say so at the point of use.

Measurement: three layers, each for a reason. Whole-frame next-state accuracy — the standard metric — cannot support the argument we are making, for a reason that has nothing to do with our method. In environments of this kind, most transitions change little or nothing, so the metric is dominated by the majority of the state that was going to stay the same anyway. Rule-governed variables are a small fraction of the predicted quantity. A model can score well on the whole frame while being at chance on every rule in the environment, and nothing in the number would reveal it.

We therefore report three layers together.

Decomposition by transition type, not by position. We partition the predicted state into groups and report each separately. The partition is by *what causes a variable to change*, not by where the variable sits, because the same variables are moved both by fixed dynamics and by per-episode rules — consuming an item decrements its count (deterministic, constant across episodes) and alters an attribute (rule-governed, resampled). Grouping by position would merge these; grouping by cause separates them, and that separation carries a substantial part of our argument (§3.1). The partition is derived from the rule set, which is known at collection time, and was fixed before any result was produced.

A PREDICT-UNCHANGED floor for every group. For each group we report the accuracy of predicting that the next state equals the current one. Without it, accuracies in this setting are uninterpretable: a group where little changes will show a high number for any method. The floor also exposes a trap we encountered and report in §3 — an apparent improvement that turns out to track a shift in the floor rather than anything the model learned.

An encoder ceiling for every group. Both the hosts and our method use a frozen perception front end, which necessarily loses some information. We measure how much by encoding the *true* next observation and scoring the result under the same criterion: this is what a perfect dynamics model would achieve given the same perception, and no model can exceed it. The ceiling separates three failure modes that a bare accuracy conflates — what the model cannot see, what it can see but has not learned, and what is not predictable from observation at all. We report it per group and use it throughout; in §3.1 it rules out perception as the cause of the XLand Transformer's failure.

A.2 HOSTS IN DETAIL

TWM host. The world-model architecture of Dedieu et al. (2025), whose agent was the first to exceed the human-expert reward on Craftax-Classic: a GPT with 3 layers, embedding width 128, 8 heads, feed-forward width four times the embedding, dropout 0.1, absolute and spatio-temporal position embeddings, and a context of 20 blocks of 144 tokens (143 state tokens and the action). Observations are tokenised by a nearest-neighbour tokeniser with a 4,096-entry codebook, patch size 10 on a 13×11 grid; the codebook is static once grown, so a code keeps one meaning. The model has about 2.0 M parameters (computed from the architecture, not read from the checkpoint). It is trained with Adam at learning rate 10^{-3}, $\varepsilon = 10^{-5}$, global-norm clipping 0.5, batch 16, from a replay buffer of 48,000 transitions, over 50,000 environment interactions with a 10,000-step warm-up; the checkpoint used is step 49,152, trained under the FIXED or RESAMPLED regime and frozen thereafter.

Hardware. Each Craftax host trains on one RTX 4090 in about four hours. The exposure, sequential, rollout and decision experiments on the TWM host run on one V100; the AdaJEPA and HiP-MDP grids on L40S, V100 and 4090 nodes; the XLand long-context Transformer on L40S, 4090 and H100 nodes. The Craftax long-context Transformer, our XLand model and all XLand data collection run on CPU only (32 cores). Where GPU types were mixed within one experiment, JAX matrix precision was set to `highest`, and the results reproduce across types.

Why the host is ours. The authors' checkpoint is trained on Craftax-Classic, a 9×9 token grid, with a replay buffer of 128,000, batch 48 and 1,000,000 interactions. Our environment is Craftax with potions on a 13×11 grid, which the rule events require and which no released checkpoint covers, so their setup cannot be reproduced as is and a host had to be trained for it; we use their architecture and optimiser unchanged. Under fixed rules this host reaches 88.44% rule-state accuracy and 88–91% accuracy elsewhere, which is the competence the diagnosis requires, and every comparison in the paper is between adaptation routes on this same host, not between this host and theirs. Longer training did not help: two runs with ten times the interactions diverged at the same step under different seeds, so the 50,000-step host is the one used throughout.

AdaJEPA: protocol, not model. AdaJEPA is a latent-space (JEPA) world model with its own encoder and predictor, trained for continuous control and adapted online during planning. Porting the full model would replace the host, and with it the tokens, the corpus and the perception front end that every other column of Table 3 shares; the comparison would then measure a different architecture rather than a different adaptation rule. We therefore apply its adaptation protocol to each host, not its system. After every observed transition the host takes gradient steps on its own pretraining loss (token cross-entropy for both hosts, following AdaJEPA's principle of reusing the pretraining objective) over a buffer of recent transitions, updating only a subset of layers, and is reset to its pretrained checkpoint at the start of every episode.

Grid. 48 configurations: buffer policy {most recent, highest loss} $\times$ buffer size {5, 20} $\times$ learning rate {1, 3, 10}$\times$ the host's pretraining rate $\times$ steps per transition {1, 3} $\times$ adapted layers {final block and output heads, first predictor block}. On both hosts the first predictor block is the first Transformer block (attention, feed-forward and its two layer norms; 198,272 parameters on each), and the final setting is the last block plus the output heads (≈ 0.78 M parameters on TWM, 204,335 on the Transformer); embeddings and input projections are never updated, and the frozen parameters are hash-checked at every step. The default is AdaJEPA's own setting: the five most recent transitions, the pretraining learning rate, one step, final layers.

Selection. For the long-context Transformer, the configuration for each test fold f is the one maximising after-first accuracy on fold $(f+1) \bmod 5$; every configuration was run on every episode, so this required no extra runs. For the TWM host a full-sweep run of one configuration costs up to about fourteen GPU-hours, so the grid was run on one fold only, the configuration chosen there, and every TWM online-adaptation number excludes that fold (189 episodes, 3,858 after-first rule events). The selected TWM configuration is the twenty most recent transitions, three times the pretraining learning rate of 10^{-3}, one step, first predictor block. On the Transformer the selected configurations all update the first predictor block, with a twenty-transition buffer and three steps on four of five folds; the multiplier applies to the Transformer's pretraining learning rate of 3×10^{-4}.

Why the two hosts respond differently. The TWM host was pretrained on this buffer, so the layers the protocol updates already sit at a minimum of the loss it re-applies; one step per transition on five to twenty recent transitions changes little. The Transformer's fold models were trained from scratch

on four of the five folds each (197 of 247 episodes), had more to move, and paid for it in non-rule accuracy.

Native setting. We also attempted to run AdaJEPA on the kind of host it was designed for: a frozen DINOv2 ViT-S/14 encoder with a predictor trained on Craftax. The check was gated on whether the frozen features represent attribute values at all. A decoder trained on the frozen features reached 13.3–18.7% joint accuracy on the five attribute tokens at the protocol's input resolution, and 15.8% at twice that resolution, against thresholds of 90% joint and 97% per attribute fixed in advance. Craftax's attribute digits occupy a few pixels and fall inside one or two 14-pixel patches, aligned with none. The check was therefore not run. Whether reusing a token loss disadvantages the protocol is answered instead on the main hosts by the logit diagnostic: the correct token's probability moves from 0.755 to 0.757 on TWM and from 0.792 to 0.814 on the Transformer across the step that includes a rule event.

Long-context Transformer, Craftax. It reads the same 143 tokens per frame from the same codebook. Each token is embedded in 8 dimensions and projected to width 128; an action embedding and a learned absolute position embedding are added. The backbone is a pre-norm causal Transformer encoder with 2 layers, 4 heads, feed-forward width four times the model width and dropout 0.1, with five output heads for the attribute tokens trained by mean cross-entropy; 620,335 parameters. Its domain is each episode's first 256 transitions, a truncation rather than a sliding window, identically at training and evaluation time. It is trained with AdamW at learning rate 3×10^{-4}, weight decay 10^{-2}, clipping 1.0, batch 8 episodes, for up to 6,000 steps with early stopping on validation loss. The 247 episodes are split into five folds and each fold's held-out model is the one evaluated, with configurations selected per fold; seeds vary initialisation, dropout and sampling. It has no store and no pretraining: the context is its only route to the rule. A *null-context* variant that sees only the current frame serves as the control for whether history is used (Appendix A.8). For the TWM host, by contrast, the online-adaptation grid was run on one fold and selection was made there; every TWM rule cell of Table 3 is on the rule events all routes share (Appendix A.17).

Long-context Transformer, XLand. It shares our model's parameter-free patch readout and tile embedding. Each step's 25 cells and pocket are embedded and summed with an action embedding and a sinusoidal position; the backbone is a pre-norm causal Transformer with 2 layers, 4 heads and model width 128, and the product head is weight-tied to the tile embedding. It has 606,257 parameters against our model's 372,644, and its context is the whole episode prefix, up to 2,000 steps. Both are trained for 20,000 steps with Adam at learning rate 3×10^{-4} and clipping 1.0 on the same 2,700 training and 300 validation episodes. The baseline's batch of four whole episodes contains 6,437 transitions per step, against our 512: twelve and a half times more data per step, in the baseline's favour.

The tokeniser is shared and frozen. On Craftax the host, the Transformer baseline and PRISMO read the same tokens from the same codebook; on XLand both models share one parameter-free readout. Perception is therefore not a source of any difference between them.

A.3 Environment modifications

Rule-bearing transitions are rare by construction in both environments, and in their default configurations they are rare enough that the phenomenon we study is not measurable at all. In XLand, production rules consume their inputs, and across ten thousand default episodes we found fewer than ten instances in which a rule took effect twice within one episode — so no method that depends on reuse, ours or a baseline's, could be evaluated. We therefore modify data collection in both environments to raise the density of rule events and, in XLand, the rate at which a rule recurs within an episode.

Three properties of these modifications are worth stating here, because they determine whether the results can be trusted.

First, **all methods see the same data.** The modifications are applied at collection time and every model — hosts, baselines, and ours — is trained and evaluated on the identical corpus.

Second, **every modification increases a baseline's exposure to rules, never decreases it.** The hypothesis under test is that these models fail to acquire rules *even when given ample opportunity*.

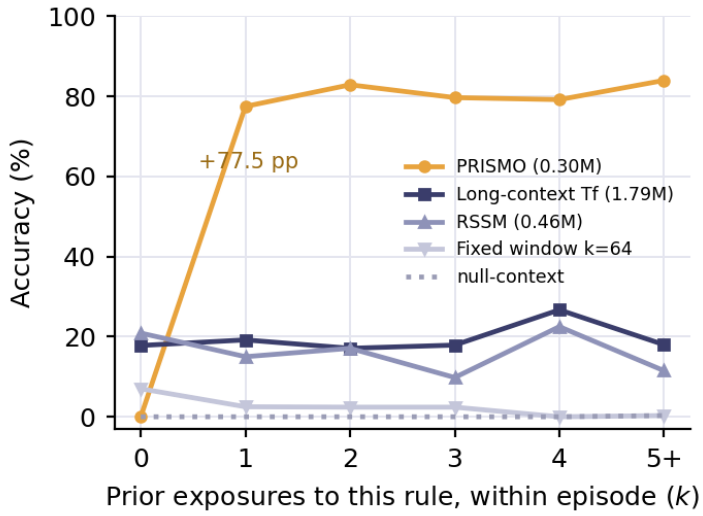


Figure A1: Exposure curves on the self-trained CNN-VAE panel. Not comparable to Figure 3a: different perception front end.

Measures that increase opportunity can only make a negative result harder to obtain and a positive result easier. This class of intervention cannot manufacture the conclusion we draw.

Third, **the rules themselves are untouched.** We change how often an agent encounters a rule, not what any rule does. The per-episode resampling that the entire study rests on is the environments' own behaviour in Craftax, and the environments' own pre-sampled rule pool in XLand.

One consequence must be stated plainly: because our corpus differs from the environments' default distributions, our numbers are not directly comparable to published results on these benchmarks, and we do not present them as such. Published figures are cited to describe existing evaluation practice, not as baselines.

XLand-MiniGrid collection. Rule sets are filtered toward those containing a two-step production chain before collection; the filter is the only collection lever used. Two others were tried and rejected: seeding more ingredients and lengthening episodes both *reduced* the number of chains whose two rules were both observed, because the collection policy spent the surplus on rules it could already apply. Ingredients are seeded; products never are, so any product on the map was made during the episode. The final corpus holds 3,000 episodes drawn from 2,729 distinct rule sets out of a candidate pool of about 15,800, and 1,104 chains testable by rollout. Seeding is neutral with respect to executability: chains execute on 88.93% of seeded maps, the same rate as on the same maps unseeded. Every (object type, colour) combination appears in at least 117,000 frames.

Craftax collection. Episodes are collected on Craftax with potions, with 20 potions of each colour placed on the map and a random drink action taken with probability 0.12, so that rule events are frequent enough to measure. The potion effect mapping itself is untouched and resampled per episode by the environment. The fixed-mapping control host is identical in every respect except that the mapping is held fixed.

A.4 The self-trained panel (CNN-VAE front end)

The numbers in this section use a different perception front end and are not comparable to the main-text panel: the same PREDICT-UNCHANGED baseline reads 96.3% here against 37.6% under the TWM tokeniser. They are reported separately for that reason.

Algorithm 1 One episode with PRISMO attached to a frozen host f_{host}.

$M[\sigma] \leftarrow (-\infty, \infty)$ for all $\sigma \in \Sigma$ ▷ cleared at the episode boundary
for $t = 1, 2, \ldots$ **do**
  $\hat{s}_{t+1} \leftarrow f_{\text{host}}(s_{t-k:t}, a_t)$ ▷ host prediction; weights frozen
  $\sigma_t \leftarrow g(s_t, a_t)$ ▷ interaction type, read from state and action
  if $\sigma_t \neq \varnothing$ **and** $M[\sigma_t]$ is filled **then**
    for each variable $i \in \mathcal{R}(\sigma_t)$ **do**
      $\hat{s}^i_{t+1} \leftarrow h\big(\textsc{Read}(M[\sigma_t],\, s^i_t)\big)$ ▷ overwrite the rule-state variable only
    end for
  end if
  act, observe s_{t+1}
  if $\sigma_t \neq \varnothing$ **then**
    for each variable $i \in \mathcal{R}(\sigma_t)$ **do**
      $\Delta \leftarrow s^i_{t+1} - s^i_t$; $[\ell, u] \leftarrow M[\sigma_t]$
      if $L < s^i_{t+1} < U$ **then** $M[\sigma_t] \leftarrow [\Delta, \Delta]$ ▷ exact; replaces what was stored
      else if $s^i_{t+1} = U$ **then** $M[\sigma_t] \leftarrow [\max(\ell, \Delta),\, u]$ ▷ clipped at the ceiling: effect $\geq \Delta$
      else if $s^i_{t+1} = L$ **then** $M[\sigma_t] \leftarrow [\ell,\, \min(u, \Delta)]$ ▷ clipped at the floor: effect $\leq \Delta$
      end if
    end for
  end if
end for

function $\textsc{Read}([\ell, u],\, v)$
  if $\ell = u$ **or** $\mathrm{clip}(v + \ell, L, U) = \mathrm{clip}(v + u, L, U)$ **then**
    return $\mathrm{clip}(v + \ell, L, U)$
  else
    return $\mathrm{clip}(v + \hat{e}, L, U)$, with $\hat{e}$ the most frequent training-set effect in $[\ell, u]$
  end if
end function

A.5 Design notes for the store

Algorithm 1 gives one episode of PRISMO attached to a frozen host, in the notation of §4.

No gradient is taken anywhere in Algorithm 1; the only trainable parameters are in g (the signature encoder) and, on XLand, in h.

Write rule. For a variable with bounds $[L, U]$, current value v, observed post-value v' and $\Delta = v' - v$, an entry starts at $(-\infty, \infty)$ and is updated as

$$M[\sigma] \leftarrow \begin{cases} [\Delta,\ \Delta] & L < v' < U \quad \text{(exact; replaces what was stored)}, \\ [\max(\ell_\sigma, \Delta),\ u_\sigma] & v' = U \quad \text{(clipped at the ceiling: effect} \geq \Delta), \\ [\ell_\sigma,\ \min(u_\sigma, \Delta)] & v' = L \quad \text{(clipped at the floor: effect} \leq \Delta). \end{cases}$$

Rules are fixed within an episode, so an exact entry stays exact. One unclipped observation makes an entry exact and usable; a clipped observation makes it usable as a bound, which later observations tighten. Reading returns $v + \ell_\sigma$, clipped to $[L, U]$, when the interval is a point or when clipping makes its two ends agree; the remaining case is below.

Why there is no surprise threshold. This is a design we arrived at only after trying the alternative. Our first version gated writes on prediction error and classified errors by whether a rule could explain them. Once Σ was discretised both became redundant: an empty slot is already the signal, and an error at a signature with a populated slot is by construction not a missing-rule error. We report this because the redundancy is not obvious in advance and the machinery is a natural thing to build.

Reading an interval that is not a point. When $\ell_\sigma < u_\sigma$ and clipping does not make the two ends agree, the read returns $\mathrm{clip}(v + \hat{e}, L, U)$, where $\hat{e}$ is the most frequent effect for this variable and clip direction in the training episodes, restricted to $[\ell_\sigma, u_\sigma]$. This is a fixed table computed offline once; it involves no learned parameters and no gradient.

Table A1: **What to store.** Write strategies on the frozen TWM host, after-first potion frames ($n = 4{,}764$). Net accuracy is close across the top three, but the error structure differs: thresholded writes damage fewer of the host's already-correct predictions, at the cost of more missed entries.

	Accuracy (%)		Error decomposition (frames)			
Strategy	rule state	all tokens	total	module wrong	stored nothing	host-correct broken
TWM host (no module)	26.9	6.5	—	—	—	—
uniform	53.3	10.9	—	—	—	—
max-magnitude	57.6	11.4	—	—	—	—
uniform + clamp	67.7	13.6	1537	1047	784	547
threshold-2 + clamp	73.0	14.3	1286	699	860	**191**
threshold-3 + clamp	72.1	14.1	1328	535	1064	181
max-magnitude + clamp	**73.1**	**14.3**	1280	765	795	199
oracle ceiling	*100.0*	*19.7*				

The *clamp* fix addresses a read-path defect: when the target value fell outside the legal range the module abandoned the correction and deferred to the host, which happens precisely where the rule's effect is largest. A counterfactual predicted 64.9% for this fix; the measured value was 67.7%.

Usable after one observation, not final. Requiring a second confirmation before use was measurably worse (§A.6). The distinction matters for reading our results: figures indexed by exposure count report what PRISMO can do after k observations, while aggregate numbers pool all observations after the first and are not one-observation figures.

A.6 What to store: write-strategy ablation

Table A1 should be read together with its error decomposition: net accuracy is close across the top strategies, but the error structure differs, and the clamp fix is the single largest effect.

A.7 Training-set size

Over training sets from 1k to 30k transitions, the baseline's rule-state accuracy lies between 3% and 33% with no monotone trend. This bounds the claim of §4 to that range: within it, more data did not make the models use the history; whether a larger corpus would is not established here.

A.8 Context window, training distribution, and what Route 1 can claim

The long-context Transformer was trained on 256-step sequences: each episode is truncated to its first 256 valid transitions, identically at training and evaluation time. Its results therefore cover only rule events within the first 256 steps of an episode.

The context-extension result in Section 3.2 carries a qualification that we state ourselves rather than leave to be found.

The host's window is in distribution, but the pattern is not. The TWM host has an architectural limit of 20 blocks and was trained on 21-step sequences, so evaluating it at a 20-step context is not an out-of-distribution probe. What is out of distribution is the *pattern the model would have to learn*. Two consumptions of the same item are separated by a median of 24 steps, which exceeds the 21-step training sequence. Within a training window, therefore, the same item taking effect twice is a rare event, and the operation at issue — find the previous interaction of this kind and reuse its outcome — is close to absent from the data the host was fitted on.

This matters for what the host's $+0.42$ pp over floor on the full sweep can be used to argue. It is evidence that the window does not reach the relevant evidence, and that the training distribution rarely rewarded reaching for it. **It is not, on its own, evidence that the architecture cannot learn retrieve-and-copy.** We do not use it that way.

The long-context Transformer is what carries the claim. The load-bearing comparison is the model whose view is an order of magnitude larger (full sweep):

Model	View	Trained on	Floor-relative
TWM host	20 steps	21-step sequences	$+0.42$
Long-context Transformer	256-step prefix	256-step sequences	-1.60

Within the Transformer's 256-step domain the previous same-colour observation lies a median 21 steps back (3,440 after-first events); it is within 20 steps for 49.6% of them, within 50 for 79.2%, within 100 for 95.3%, and inside the domain for all of them. On the full corpus the median gap is 24 steps. Roughly ten repeat opportunities fall inside the view, and the corresponding pattern is present in the training distribution rather than rare in it. The model nevertheless sits below its floor (-1.60). **This is the comparison that supports a claim about capability**, and Route 1 should rest on it.

The window-coverage split points the same way. Restricting to rule events whose previous occurrence falls inside the host's window gives $+0.99$ pp, against $+0.11$ pp for those outside it. The split behaves as the window account predicts — and it also bounds what the window account can rescue. Even where the evidence is demonstrably reachable, the gain is about one point.

Longer context helps, just not on rules. Extending the host's context twentyfold, to its architectural limit, moves rule-state accuracy on the full sweep from 26.93% to 27.41%. Over the same extension, accuracy on all other transitions rises from 88.41% to 90.25%. Longer context gives this architecture about two points of generic benefit; on rule frames it gives half a point. The context mechanism is working, and the model is using it. It is not using it for rules.

The reachable fraction bounds this further. On the full corpus, twenty steps covers 47.4% of repeat occurrences (median gap 24 steps). Had the model exploited the evidence on that fraction — lifting those frames from 27% to, say, 70% — the aggregate would rise by roughly $47\% \times 43 \approx 20$ points. The measured rise is 0.48, about 2% of the fully-exploiting case.

Three windows, one outcome. Widening the window a hundredfold does not change the result. The three rows below are different environments and different models, so the numbers are not directly comparable; what the table shows is that the outcome does not move as the view grows.

	Window	Coverage	Result
TWM host (Craftax)	20 steps	$\approx 47\%$ of repeat occurrences	at floor
Transformer (Craftax)	256 steps	≈ 10 repeat opportunities	$\approx$ null-context
Transformer (XLand)	whole episode ($\leq$ 2,000)	every earlier occurrence — but only ≈ 1.7 per episode	at floor

Why we did not retrain the host with a longer context. Two reasons. The 20-block limit is architectural rather than a configuration we chose, so changing it produces a different model rather than the same model with a longer view. And the question it would answer — does a larger view help when the pattern is present in training? — is already answered by the long-context Transformer, at an order of magnitude more view.

What we conclude. The qualification makes the argument narrower and firmer at the same time. A model whose window cannot reach the evidence, trained where the pattern barely occurs, teaches us little by failing. A model that sees ten repeat opportunities, trained where the pattern is common, and that still cannot be told apart from a model with no history, is the observation that earns the word *capability*.

A.9 Closed-loop rollout: what happens over an imagined trajectory

This section reports the comparison of §3.1 over a closed-loop rollout, in which the model is fed its own predictions and no further observation enters after the store has been filled (Table A2, Table A3).

With the evidence held equal, the gradient route moves nothing and PRISMO does not. Both adapted arms see the same six observations. On the variable the rule governs, the paired difference for fine-tuning is 0.00 ± 0.05 at step 10 and $+0.04 \pm 0.09$ at step 50; the store's error is 0.229 at step 2 and 1.70 at step 50, against the host's 0.656 and 2.78 (paired difference $+1.078 \pm 0.216$). Giving

the gradient route more evidence does not rescue it: at $m = 5$ observations per colour the paired difference at step 50 is -0.10 ± 0.08 — no better, and if anything worse. The contrast with the single-frame measurement is the point. On one frame the two routes differ by about 32 points (54.8 against 87.2 on the common set); under fifty steps of closed-loop compounding the gradient route's advantage over the untouched host disappears entirely.

On the rest of the state PRISMO helps and fine-tuning hurts, for different reasons. Fine-tuning loses drift accuracy at every step (-0.02 to -0.11 per step), and by step 50 it is worse than the untouched host on the host's own territory (-0.109 ± 0.041). This is the 6.9-point non-rule loss of §3.2 expressed over a trajectory rather than a frame. The store's improvement on the same variables (1.72 against 1.96 at step 50) is *not* the module modelling drift — it writes to no variable in Panel B. It is a consequence of the loop: a corrected frame fed back is a better base from which to predict the next one.

In rollout, a rule kept in the weights still does better than one kept in a store. We report this plainly. At step 50 the fixed-mapping host stands at 0.80 and PRISMO at 1.70; in horizon terms, 43 steps against 32, with the untouched host at 19. In this host, **putting the rule in a store buys about thirteen steps of usable imagination and putting it in the weights buys about twenty-four**. The remainder is structural: the store's value is computed from an imagined state that has itself drifted.

This does not contradict §3.1, where PRISMO's point estimate is above the fixed-mapping host's. A decision depends on the one attribute the potion changes, which is where PRISMO concentrates its accuracy; the length of an imagined rollout depends on every attribute, including drift the store does not model.

The error does not compound; the contamination spreads. On a variable the rule governs the error is a single step of roughly 3.5 units and is then flat — the untouched host's merged error is no higher at step 50 than at step 5. What grows is the number of variables affected, from 0.9 to 2.5 across the rollout, together with drift on the rest. Reporting a merged mean without this decomposition would invite the opposite reading.

The control. With the store empty, the method's curve is identical to the host's: the paired difference is exactly 0.000 at every step. The module can only observe the model's own imagination, so writing a prediction into the store would feed the host's own error back to it as though it were evidence; the store must be filled by real observation, and this is the measurement that says so.

A.10 Post-hoc: what the XLand Transformer memorised

This is a post-hoc diagnostic, run after the baseline result was in. Scored on its own *training* episodes, the long-context Transformer reaches 90.81 ± 2.25 on first exposures and 91.39 ± 3.11 on later ones (mean $\pm$ s.d. over three seeds).

The first-exposure number carries the most weight. On a first exposure the rule has not appeared anywhere earlier in context, so no retrieval can answer it; a score of 91% there is evidence for episode-specific memorisation, recognising the episode from its scene — layout and object set — and recalling that episode's rule table. The two exposures also score the same, whereas a model reading from context would be expected to score higher on later exposures than on the first.

It is the signature of §3.1 in a different form. The fixed-mapping host memorised one rule; this model memorised one rule table per training episode, keyed on the scene. On a new episode the key matches nothing, and it falls to the floor (§3.1). Training on resampled rules does not by itself force a model to acquire rather than memorise.

A.11 Holding the scene fixed does not close the memorisation route

A control on XLand-MiniGrid in which every episode starts from the same layout, agent pose and heading, and only the rules are resampled (Table A4). The long-context Transformer still identifies its training episodes: it scores 90.6% on them at a rule's first observation, against 90.8% in the existing corpus, and stays at the floor on validation episodes. The reason is that in XLand the objects on the map are the rule set's ingredients, so the initial object set alone identifies 99.3% of training episodes. The memorisation route is therefore not spatial: the model uses whatever identifies the episode. Our

Table A2: **Closed-loop rollout, mean absolute error ± s.e.** The model is fed its own predictions, and no further observation enters after the store is filled; lower is better. Panel A is the attribute the episode's rule governs; Panel B is the state the module never writes to. *Usable horizon*: per rollout, the first imagined step at which the mean absolute error over the five attribute tokens, in attribute-value units, exceeds 1.0 (51 if never crossed), reported as the median over rollouts; the threshold, the 50-step length and the 20-step context were fixed before any rollout was run. Actions are taken from the recorded episode; states are the model's own predictions, never re-grounded.

Panel A — the attribute the rule governs

	fixed-mapping host	random-mapping host	+ **PRISMO**	+ fine-tune (matched)	+ fine-tune (m=5/col.)
step 2	*0.064*	0.656	**0.229**	0.66±0.13	0.72±0.14
step 10	*0.25*±0.06	1.87±0.20	**0.79±0.15**	1.87±0.20	1.87±0.20
step 20	*0.33*±0.05	2.35±0.19	**1.20±0.17**	2.17±0.19	2.22±0.19
step 50	*0.80*	2.78	**1.70**	2.73±0.21	2.88±0.21
usable horizon (steps)	*43*	19	**32**	19.0	19.5

Panel B — the four state variables the module never writes to

	fixed-mapping host	random-mapping host	+ **PRISMO**	+ fine-tune (matched)
step 20	*0.62*±0.06	0.97±0.07	**0.77±0.06**	1.07±0.07
step 50	*1.42*	1.96	**1.72**	2.07±0.09

The fixed-mapping column is set in italics because it is a *reference*, not a competitor: it is the same architecture with the episode's rule already in its weights, and it is the ceiling this comparison is measured against, not an arm that adapts.

Table A3: **Paired differences against the untouched host** (positive = better; same episodes, same seeds). With the same six observations, fine-tuning at matched evidence moves nothing.

	Panel A — attribute the rule governs			*Panel B* — the rest of the state	
	+ **PRISMO**	+ fine-tune (matched)	+ fine-tune ($m = 5$)	+ **PRISMO**	+ fine-tune (matched)
step 10	$\mathbf{+1.081 \pm 0.193}$	0.00 ± 0.05	0.00 ± 0.07	$\mathbf{+0.064 \pm 0.040}$	-0.064 ± 0.031
step 50	$\mathbf{+1.078 \pm 0.216}$	$+0.04 \pm 0.09$	-0.10 ± 0.08	$\mathbf{+0.242 \pm 0.080}$	-0.109 ± 0.041

model is unaffected. This control does not separate insufficient data from a stable obstacle; it shows only that removing one identifying cue does not remove the route.

Table A4: Fixed-scene control on XLand-MiniGrid: pair accuracy (%), mean over three seeds.

	existing corpus	fixed scene
long-context Transformer, validation, after first	1.29	1.15
long-context Transformer, training episodes, first observation	90.8	90.6
ours, validation, after first	99.27	98.37
ours, read path zeroed	1.52	2.45

A.12 HOST AND PRISMO ON THE TWM HOST, WITH THE NO-LEAKAGE CONTROL

Table A5 gives the conjunctive rule-state accuracy of the host and PRISMO over the full evaluation set (all 4,764 TWM rule events), a complement to the common-set values of Table 3 (1,764 events), with the *before 1st* row as the no-leakage control.

Table A5: **Conjunctive rule-state accuracy on the TWM host, host and PRISMO.** Full evaluation set, not the common set of Table 3. The *before 1st* row is the no-leakage control: the store is still empty, so host and PRISMO predict identically.

Split	n	Host	+ PRISMO	Gain
after 1st	4,764	26.93	**85.1**	+58.2
before 1st	1,225	5.80	5.80	0.00

A.13 Decision experiment: protocol notes

Ties. Ties are broken uniformly at random in every condition, and we report the tie count. Breaking them by candidate index would disguise the candidates' presentation order as signal, and would make a model that produces no ordering look better than chance.

A second implementation. The equivalence between the resampled host and a random chooser also holds on the earlier CNN-VAE panel (Appendix A.4): 63.0% against 64.2%. Two implementations with nothing in common reach the same conclusion.

A.14 XLand: the fixed-ruleset control

Why the comparison is not in the main text. A host trained on a fixed ruleset reaches 100.00 ± 0.00 — but so does a lookup table built from the training corpus, because that corpus contains one ruleset with two rules. The comparison cannot separate learning a rule from holding two rules in the weights, so it settles nothing, and we replace it with the lookup table on both corpora (Table 2).

What the control did show. The fixed-ruleset model scores 0% on *every* condition of the composition experiment (§5.4), including the teacher row, where the rule table is handed to it. It holds one rule table in its weights and has no path by which a rule can be supplied at inference time. This separates *knowing a rule*, which weights can do, from *being able to receive one*, which needs a read path. Such a model looks competent and is inert.

A.15 HiP-MDP settings

The main text uses the settings of Killian et al. (2017): latent dimension $|w_b| = 5$, $N_u = 2$ alternations per tuning event, latent learning rate 5×10^{-4}, network learning rate 2.5×10^{-4}, $\alpha = 0.5$, prioritised replay with exponents 0.2 and 0.1. To check that these are not unfavourable here, we swept latent dimension and alternation count, then step size at the best cell (Table A6). The best configuration found, $|w_b| = 16$, $N_u = 4$ with step size doubled, reaches 79.7 against the paper's 79.3, within noise, so every reported figure uses the paper's settings.

Table A6: HiP-MDP sweep: pooled rule-state accuracy after a colour's first observation (%), adapter-only variant, selection fold, one seed. Bold: the published setting. The surface is flat for $|w_b| \leq 16$.

$N_u \setminus \lvert w_b \rvert$	2	5	16	32
1	77.8	79.1	77.4	65.7
2	78.8	**79.3**	78.8	73.7
4	77.9	78.3	79.5	77.8

A.16 Limitations

PRISMO improves prediction only on the variables that per-episode rules govern, so its whole-frame effect is small; it repairs one blind spot of a frozen host rather than replacing the host, and it requires that the interaction type σ_t can be read from the host's input and that the variables each rule governs are known in advance (§4). The XLand context-route failure is established in this data regime, not in principle: the training corpus holds $2{,}700$ episodes with roughly 1.7 TileNear events each, so few in-context examples exist in which retrieval is the only way to reduce the loss, and every episode

can be memorised instead. A larger corpus or a longer schedule might make retrieval cheaper than memorisation; that was not run, and nothing was tuned after the result.

A.17 NOTES TO THE MAIN-TEXT TABLES AND FIGURES

Table 1 (decision). 640 decision points per condition, a median of 6 candidates per point; 95% intervals are clustered by episode. The random policy's tie count of 1.00 is definitional: a random policy produces no ordering to tie. The PRISMO row reports the values as run (oracle's choice 61.6% $[46.9, 76.1]$, regret 0.754 $[0.440, 1.156]$, harmful 6.4% $[0.0, 18.1]$); the interval is wide because the bootstrap is over 36 episodes. A deterministic expected-tie-break computation agrees with the as-run values within 0.5 pp.

Table 2 (XLand baselines). Mean $\pm$ s.d. ($n - 1$) over three seeds, except MAJORITY-PRODUCT and train-time `pair-lookup`, which are seed 0 only. Every row uses the same validation episodes and the same per-seed n.

Table 3 (remedies). All rule cells on TWM are on the $1{,}764$ events every route shares; all rule cells on the Transformer are on its 508 fine-tuning site-events; *other* is each route's paired change on its own frames. Each such value is a difference against the frozen host on the same frames, so the difficulty of those frames largely cancels; the frame sets, and how far into adaptation they are measured, still differ between routes. The zeros are structural and hold on any frames. The abstract, Figure 1 and every comparison of routes in the main text use these values. Figures over the full evaluation set ($4{,}764$ TWM events, $3{,}440$ Transformer events) are PRISMO-only or host-only measurements and say so: the collapse in §3.1, the interventions of Table 6, the write variants of Table A1, and the growth of the online-adaptation cost with episode length. Over the full set PRISMO reaches 85.1 on TWM and 83.6 on the Transformer. The *episode-local store* is a plain key-value memory on PRISMO's interface: same interaction address, same overwrite of the rule-state outputs, but each entry holds the observed change as a value (the max-magnitude + clamp row of Table A1, 73.1 over all $4{,}764$ TWM events). The *frozen-host* HiP-MDP keeps the host frozen and infers one latent per interaction type, about 200 gradient steps per observation. The *other* entries of 0 for PRISMO, the episode-local store and the frozen-host latent are not near misses: with the host frozen and corrections confined to rule-state tokens, output on every other token is identical to the host's, asserted on $384/384$ frames rather than measured. The published HiP-MDP *other* entry (-51.6, from the start to the end of an episode) is measured on held-out frames; the method trains on every transition, so in-episode frames would be contaminated. The HiP-MDP Transformer cells are not run by decision: the costs of updating weights are a property of gradient optimisation rather than of architecture, and are established on the TWM host. RSSM is omitted because it falls below the copy floor on non-rule frames and does not qualify as a baseline. The Transformer is the better-behaved baseline: on the full set it clears its non-rule floor (92.26 against 88.04) where the TWM host falls below its own (88.41 against 89.00), while its rule predictions still fail. *No adaptation* is each host's shortest evaluation window (TWM ctx 1, Transformer window 1); *longer context* is its longest (TWM ctx 20, the Transformer's full 256-step domain). *Rules fixed* is a separate host trained with the rules held fixed, on its own corpus: a reference, not a remedy, and not on the common set. On the common-set episodes the frozen TWM host scores 90.2% on other transitions.

Table 4 (gradient against PRISMO). In panel (a), the $m = 1$ row is on the E1 site sample and the best-point row on the $1{,}764$-event common set of Table 3 ($k = 200$); PRISMO pools all observations after the first. The 40-point gap at $m = 1$ is not a matter of sample: both routes are scored on the same sites there. The replay entry in the interference row is new-rule accuracy against a fresh copy at stages 2, 5 and 10 (-22.0, -34.7, -24.6 on TWM); these rest on 2–8 evaluation rule events per sequence, with per-sequence differences ranging from $+50$ to -100, so only the direction should be read. In the forgetting row the store is cleared at the rule change, so nothing is left to forget. In the erosion row the identity is structural, asserted on $384/384$ frames, not a measurement. In the interference row old entries are discarded, so the comparison has no second arm.

Table 5 (interference). 983 problems, seed 0, for PRISMO and the frozen and incremental arms. The published HiP-MDP entry after one interposed colour is on the first 200 problems; its entry after five interposed colours is -0.66 pp $[-3.55, +2.08]$ on $n = 151$ of the first 200 problems, one seed; the re-observation control on the same 151 problems is also -0.66.

Table 7 (composition). *Teacher* is the same model handed the rule table instead of writing it from observation. Against the teacher, writing from observation is $+0.6$ on C1 and $+0.5$ on C3, both intervals spanning zero. Composition adds no cost beyond the product of the per-step accuracies, and 2 of 25 first-step errors are recovered at the second step.

Figure 3b (sequential rule sets). Each stage holds only 14–45 rule events, so the figure carries a band and no per-stage point value is quoted.

Figure 3a (exposure curves). Fine-tuning reloads the base checkpoint at each site; HiP-MDP adapts continuously, so its value at m includes all tuning up to that prediction. Every curve in panel (a) is plotted above its own *per-*m floor rather than the pooled floor of Table 3, because the floor moves with m (5.4 at m=0 against 25.2 at m=5 on the full sweep) and a single pooled value would distort the step at m=1. HiP-MDP and AdaJEPA (online TTA) pool $m \geq 5$; AdaJEPA reports no m=0 point. The AdaJEPA curve is the selected configuration, evaluated on the 3,858 rule events that remain after the selection fold is excluded, against the other curves' full sweep; its per-m floors are the full-sweep ones. In the right-hand plot of Figure 3a the AdaJEPA line is a single pooled value (-0.8 pp) drawn flat, not a measured curve; HiP-MDP's non-rule change is reported at $m = 0, 1, 3, 5$ only and is measured against a frozen reference of 90.0 on held-out frames, because that method trains on every transition of the episode.