ASSOCIATIVE DREAMER: PREDICTING ACTION CONSEQUENCES WITH ASSOCIATIVE MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

In partially observed environments, identical current images and actions can have different consequences because of earlier events. We study how a world model can recover and use the relevant historical evidence within a bounded online state. We introduce Associative Dreamer, which learns to write visual associations into a fixed-size fast-weight memory, retrieve evidence through autoregressive queries, and condition its action-dependent latent prior on that evidence. At known retrieval times, an auxiliary objective aligns retrieved representations with exponential moving average (EMA) targets of annotated, query-relevant past observations. Targets come from the same training run and serve only as loss supervision. We construct a benchmark of three controlled pixel tasks whose future action consequences depend on earlier evidence, and evaluate short open-loop prediction on fixed training trajectories. Within a 20,000-update budget, the aligned model achieves 99.58% and 100% mean final-window consequence accuracy in Transport and Power, compared with approximately 50% for both an ordinary recurrent model and memory trained without alignment. At 40,000 updates in Assembly, aligned and prediction-only memory average 77.01% and 75.97% balanced accuracy, respectively. Disabling future reads removes the successful models' predictive advantage. These results show that Associative Dreamer combines retrieved historical evidence with the current state and actions to predict future consequences that depend on earlier experience.

1 INTRODUCTION

Consider a mobile robot inspecting cooling equipment powered by two redundant supplies, PSU-1 and PSU-2. It first observes that the left branch connects to PSU-1 and the right to PSU-2, then learns from a diagnostic panel that PSU-1 has an overload fault. By the time it reaches the isolation controls, neither clue is visible. Disconnecting the left branch isolates the fault; reversing the earlier wiring would make the same action disconnect the healthy supply. Thus identical current images and actions can imply different outcomes. Our inspection task abstracts this dependency using visual markers.

Recurrent world models such as Dreamer (Hafner et al., 2020; 2021; 2025a) compress observations into a latent state. In the example above, a useful state must preserve briefly visible facts, relate them across intervening observations, and recover their implications for a later action. Adding memory introduces a learning problem of its own: which associations should be stored, what should a query recover, and how should the retrieved information change a prediction? These requirements connect memory learning to dynamics learning.

We address this problem with *Associative Dreamer* (Figure 1). The model conditions future prediction on both its recurrent state and evidence retrieved from bounded episode-local memory. Our contributions are:

1. **Associative Dreamer.** We develop a world model that learns to write visual associations into a fixed-size fast-weight memory and retrieve them through a shared autoregressive Reader. Retrieved Values condition the action-dependent latent prior, connecting historical evidence to generated physical consequences.
2. **Evidence-aligned learning.** We formulate retrieved content as latent evidence and introduce an auxiliary Value-alignment objective using same-training exponential moving average

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.

(EMA) targets. Query–history correspondences supervise evidence recovery through the loss; prediction uses the model's own memory reads; reference Values serve only as loss targets.

3. **A benchmark suite for history-dependent prediction.** We introduce a new controlled suite comprising Transport, Power, and Assembly. The suite provides fixed-action trajectories, paired histories that hold current views and future actions fixed, and physical-outcome metrics. These test whether short open-loop predictions respond to relevant earlier events while remaining insensitive to irrelevant ones.

On fixed training supports with known retrieval times, our experiments show that EMA-aligned memory learns Transport and Power outcomes. Disabling future reads removes the successful predictors' advantage, linking their performance to the retrieval pathway.

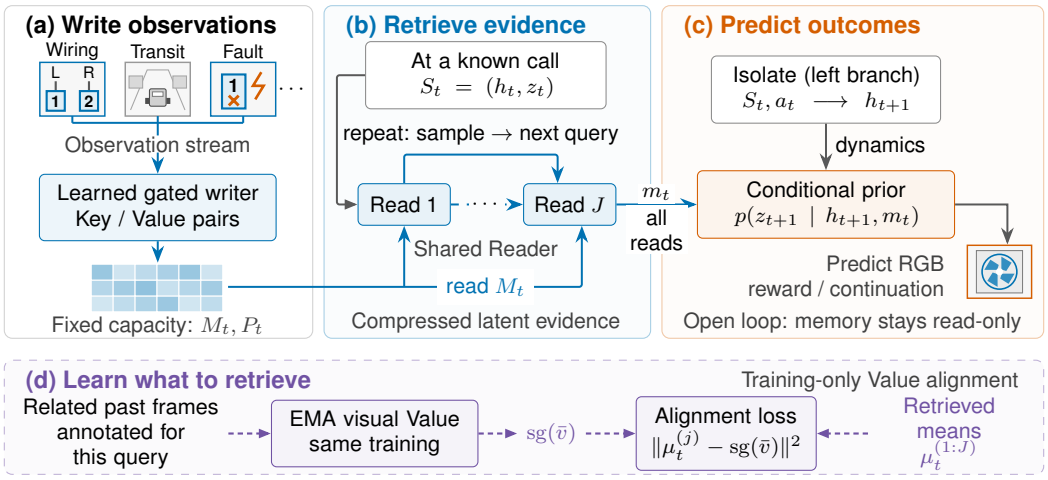

Figure 1: Associative Dreamer: from past observations to future consequences. (a) Inspection observations—a circuit's wiring, transit, and its fault—update a fixed-capacity associative memory through learned writes. (b) At a known call, one shared Reader performs J autoregressive reads; each sample helps form the next query. Intermediate reads are omitted; our experiments set $J = 2$. (c) Retrieved evidence $m_t = [e_t^{(1)}; \cdots ; e_t^{(J)}]$ conditions the action-dependent latent prior. The fused read also reaches the reward head (Equation 3). Memory is read-only during open-loop prediction. (d) Relevant historical frames provide same-training EMA targets for a loss on retrieved means, never forward-pass inputs. The numbered power units illustrate the Introduction's motivating scenario; experimental markers are shown in Figure 4. Icons are schematic, not model outputs or write-selection labels.

2 Related Work

World models and historical context. Early world models combine visual compression with recurrent prediction (Ha & Schmidhuber, 2018); PlaNet introduces deterministic and stochastic RSSM states (Hafner et al., 2019). Dreamer learns behavior in latent imagination (Hafner et al., 2020), followed by categorical states and KL balancing in DreamerV2 (Hafner et al., 2021) and broader task coverage in DreamerV3 (Hafner et al., 2025a). Dreamer4 instead uses Transformer video dynamics (Hafner et al., 2025b). For historical context, R2I uses a state-space backbone (Samsami et al., 2024), ESWM completes transitions from sparse episodic banks (He et al., 2026), and MemLearner queries context tokens within a pretrained video model (Yu et al., 2026).

Fast-weight memory updates. Fast-weight networks encode temporary information in dynamically updated weights (Schmidhuber, 1992; Ba et al., 2016). Delta-rule updates correct stored key–value associations (Schlag et al., 2021a); Gated DeltaNet combines these corrections with adaptive decay (Yang et al., 2025). Titans learns neural memory through online key–value regression (Behrouz

et al., 2025), while test-time regression places RLS and related updates in a common framework (Wang et al., 2026). These write-time objectives differ from supervision of later query-dependent retrieval.

Multi-hop associative retrieval. End-to-End Memory Networks feed each memory read into the next query state (Sukhbaatar et al., 2015); Key-Value Memory Networks separate addressing and retrieved content across hops (Miller et al., 2016). Fast Weight Memory recycles retrieved Values into subsequent queries within one input step (Schlag et al., 2021b). This within-call inference differs from autoregressive prediction across time. Ba et al.'s fast-weight network likewise performs repeated memory-conditioned state refinement within a time step (Ba et al., 2016).

Evidence supervision and representation targets. Memory Networks train retrieval with labeled supporting facts (Weston et al., 2015), whereas their end-to-end successor learns from answer supervision (Sukhbaatar et al., 2015). In representation learning, BYOL trains against a slowly averaged target network (Grill et al., 2020); I-JEPA uses EMA target representations for prediction of image regions (Assran et al., 2023). Evidence selection and target stabilization thus have distinct precedents.

3 Problem Setting

Observations and prediction target. We study action-conditioned visual prediction under partial observation. Physical rules are fixed across episodes and learned from data, while each episode has its own configuration and history. At time t, the observed history is $\mathcal{H}_t = (x_{0:t}, a_{0:t-1})$, containing RGB images and executed actions. Public remaining-time values τ and a known retrieval-call schedule χ are also available. Given L subsequent actions, the model predicts the joint distribution of future images, rewards, and continuation indicators:

$$p(y_{t+1:t+L} \mid \mathcal{H}_t, a_{t:t+L-1}, \tau, \chi)\,, \qquad y_s = (x_s, r_s, c_s).$$

Rewards r_s and continuation indicators c_s are prediction targets rather than additional history inputs. For $L > 1$, rollout is open-loop: it receives no intermediate true images. The predicted distribution can retain uncertainty when the history does not determine a unique outcome.

Historical dependence under bounded memory. The same current image, future actions, and public conditions can accompany different physical outcomes when earlier observations differ. Relevant facts may occur at separate times, with unrelated observations in between. We seek a predictor that preserves these consequential differences using a causally updated online state, including any memory, whose size is independent of history length. Training and evaluation may replay the observed prefix to reconstruct this state. The challenge is to retain and retrieve the historical information needed for a later prediction.

Available information and supervision. The schedule $\chi_t \in \{0, 1\}$ specifies when a memory read is called, not which evidence it should recover. It is supplied before rollout and shared across candidate actions from the same prediction origin. Hidden physical configurations are unavailable to the predictor. In the supervised setting, training additionally identifies the relevant earlier observations for each call. These correspondence annotations select reference representations for the retrieval loss. Neither the annotations nor the reference representations enter prediction or determine which frames are written. All real frames remain eligible for writing. Our experiments evaluate learning on a fixed set of training trajectories.

4 Associative Dreamer

Associative Dreamer adds a learned associative pathway to a Dreamer-style recurrent state-space model (RSSM) (Hafner et al., 2019; 2025a). After observing x_t, its online state comprises $S_t = (h_t, z_t)$, with deterministic recurrent state h_t and categorical stochastic state z_t, together with a Value-by-Key matrix M_t and inverse Key covariance P_t. The writer associates new visual content with its preceding state–action context; the Reader recovers query-dependent evidence; the RSSM uses that evidence to predict action consequences. Figure 1 follows this write–retrieve–predict path and separates it from the loss-only reference branch.

The memory changes through updates to fixed-size matrices, not by appending observations. Any requested read precedes the prediction; the new real observation is written afterward. During open-loop prediction, the memory remains read-only. We describe the four parts of the method below; implementation constants are collected in Appendix C.

4.1 Writing visual associations

The writer learns an address for an observation from the context that led to it. After action a_t produces a real image x_{t+1}, the Key k_{t+1} is computed from (S_t, a_t) and the Value v_{t+1} from the new visual embedding $u_{t+1} = \mathrm{Enc}(x_{t+1})$. Keys have unit norm. A sigmoid gate β_{t+1} uses the state–action context and visual embedding to control write strength. We update the association with sample-weighted recursive least squares (RLS; Haykin, 2014, Chap. 10):

$$\begin{aligned} k_{t+1} &= K(S_t, a_t), \\ v_{t+1} &= W_V \,\mathrm{RMSNorm}(u_{t+1}) + b_V, \\ g_{t+1} &= \frac{\beta_{t+1} P_t k_{t+1}}{1 + \beta_{t+1} k_{t+1}^\top P_t k_{t+1}}, \\ M_{t+1} &= M_t + (v_{t+1} - M_t k_{t+1}) g_{t+1}^\top, \\ P_{t+1} &= P_t - g_{t+1} k_{t+1}^\top P_t. \end{aligned} \tag{1}$$

Here $M_t k_{t+1}$ is the memory's current prediction of the new Value. The residual corrects that association, while P_t adjusts the update using previously written Key directions. For fixed Keys, Values, and weights, this is online weighted ridge regression (Wang et al., 2026); β_i weights an association without imposing temporal forgetting. All real frames, including the initial frame, are eligible for writing. The learned gate controls their contribution, and the update remains differentiable through the observed sequence; clue annotations do not select writes.

4.2 Retrieving query-dependent evidence

An initial read can recover evidence that helps determine what to retrieve next. Adapting this read–query feedback principle from Fast Weight Memory (Schlag et al., 2021b), we perform J successive reads with a shared Reader at each known call, for a prespecified positive integer J. Its initial state $b_t^{(0)}$ is projected from the RMS-normalized $[h_t; z_t]$, without the candidate action. A shared projection and hop embedding produce a unit-normalized query. For $\chi_t = 1$ and $j = 1, \ldots, J$,

$$\begin{aligned} q_t^{(j)} &= Q(b_t^{(j-1)}, j), & \mu_t^{(j)} &= M_t q_t^{(j)}, \\ e_t^{(j)} &= \mu_t^{(j)} + \sigma \epsilon_t^{(j)}, & \epsilon_t^{(j)} &\sim \mathcal{N}(0, I), \\ b_t^{(j)} &= b_t^{(j-1)} + F_{\mathrm{read}}(b_t^{(j-1)}, e_t^{(j)}), & & \\ m_t &= [e_t^{(1)}; \cdots; e_t^{(J)}], & d_t &= b_t^{(J)} - b_t^{(0)}. \end{aligned} \tag{2}$$

The memory supplies the mean $\mu_t^{(j)}$ of a Gaussian evidence sample. Feedback uses the actual sample $e_t^{(j)}$, so later queries depend on earlier retrieved evidence. This defines an autoregressive distribution π over the ordered bundle $\mathbf{E}_t = (e_t^{(1)}, \ldots, e_t^{(J)})$. The two readouts have different roles: m_t concatenates all samples for the latent prior, whereas d_t summarizes the Reader's state change for the reward head. For Value width d_v, m_t has width Jd_v. All reported experiments use $J = 2$ and $\sigma = 0.25$. When $\chi_t = 0$, the Reader and Gaussian sampling are skipped and $m_t = d_t = 0$.

4.3 Predicting with retrieved evidence

Retrieved evidence enters the generative path by conditioning the next-state prior. The action advances the deterministic state h_{t+1}; the prior then combines this state with m_t to predict the stochastic state. The posterior and output heads retain their distinct inputs:

$$\begin{aligned} h_{t+1} &= F_\theta(h_t, z_t, a_t), \\ p_\theta^z &= p_\theta(z_{t+1} \mid h_{t+1}, m_t), \\ q_\psi &= q_\psi(z_{t+1} \mid h_{t+1}, x_{t+1}), \\ p_\theta^x &= p_\theta(x_{t+1} \mid h_{t+1}, z_{t+1}), \\ p_\theta^r &= p_\theta(r_{t+1} \mid h_{t+1}, z_{t+1}, d_t, \tau_{t+1}), \\ p_\theta^c &= p_\theta(c_{t+1} \mid h_{t+1}, z_{t+1}, \tau_{t+1}). \end{aligned} \tag{3}$$

The prior receives the concatenation $[h_{t+1}; m_t]$. The RGB decoder and continuation head use the resulting latent state; only the reward head also receives d_t directly.

With real observations, the model performs any scheduled read from M_t and forms the prior, then uses x_{t+1} to infer the posterior state and write its visual Value through Equation 1. This ordering prevents the new image from being read back to predict itself. In open-loop rollout, z_{t+1} is sampled from the prior. The model advances the public clock and repeats retrieval and prediction while keeping M/P fixed; generated observations and states are not written.

4.4 Learning to retrieve predictive evidence

The world-model objective trains memory through its effect on prediction. To supervise retrieved content directly, we additionally align each read with a representation of its relevant historical observation. For episode n, $\mathcal{R}^{(n)}(t, j) \le t$ identifies the annotated frame for read j at time t. An EMA copy of this training run's visual encoder and Value projection produces its reference $\bar{v}_i$. References are computed after the online forward pass and used only as stopped-gradient loss targets. For a batch of B episodes, we optimize

$$\begin{aligned} \mathcal{J} &= \frac{1}{B}\sum_{n=1}^{B}\big(L_{\mathrm{WM}}^{(n)} + L_{\mathrm{align}}^{(n)}\big), \\ L_{\mathrm{align}}^{(n)} &= \frac{1}{2\sigma^2 N_n}\sum_{t:\chi_t^{(n)}=1}\sum_{j=1}^{J}\Big\|\mu_t^{(n,j)} - \operatorname{sg}\bar{v}_{\mathcal{R}^{(n)}(t,j)}\Big\|_2^2. \end{aligned} \tag{4}$$

N_n counts all real observation rows, including the initial row. Alignment is evaluated at active calls and summed over reads and Value coordinates, then averaged over rows and episodes. It trains the actual retrieval means under the Reader's own sampled prefixes. L_{WM} contains RGB, reward, and continuation prediction losses plus the gradient-balanced RSSM KL terms (Hafner et al., 2021; 2025a); their scales and reduction details are given in Appendix C.

Gradients flow through online writes and reads; the reference branch is stopped. Its Values and annotation indices never enter the Reader, feedback, writer, or prediction heads. EMA parameters start from the online initialization and update once after each complete optimizer step. They are outside the online optimizer and deployed state; no independently trained teacher is used.

Why align retrieved Values? The latent-evidence model marginalizes retrieved evidence to define its outcome distribution. A history-conditioned evidence proposal gives an evidence lower bound (ELBO) containing a KL to the Reader (Appendix A). At a fixed read prefix, let ρ_j be a Gaussian centered on the historical reference and π_j the Reader's conditional, both with covariance $\sigma^2 I$. Then

$$D_{\mathrm{KL}}\Big(\mathcal{N}(\bar{v}_{\mathcal{R}(t,j)}, \sigma^2 I)\,\|\,\mathcal{N}(\mu_t^{(j)}, \sigma^2 I)\Big) = \frac{\|\mu_t^{(j)} - \bar{v}_{\mathcal{R}(t,j)}\|_2^2}{2\sigma^2}. \tag{5}$$

This identity motivates the auxiliary Value-alignment objective: matching retrieval means to stop-gradient reference Values at Reader-generated prefixes. Appendix A details its relationship to the proposal ELBO.

We denote the prediction-only fast-weight variant by *FW* (fast weights), and the version with EMA-target Value alignment by *EMA+V*. Both use the same online memory architecture. FW retains the Reader, noise, and calling schedule but removes reference construction, alignment, and EMA; the ordinary control has no memory.

5 Experiments

Our experiments ask three questions: Can associative memory support history-dependent physical prediction? Do the predictions use the retrieved evidence? How do evidence supervision and target updating affect learning?

5.1 EXPERIMENTAL SETUP

Tasks. The benchmark comprises three pixel environments with fixed dynamics and episode-specific configurations. In *Transport*, an earlier departure and diversion determine where a floating platform arrives; a later docking action forms a bridge or lets it drift past. In *Power*, wiring and overload events determine whether isolating a branch preserves power or damages equipment. In *Assembly*, a valve connection and a heat pulse determine a material's hidden curing state; processing then produces a flat, torn, or brittle plate. Paired histories hold the current image and future actions fixed while changing an earlier fact and its physical consequence. Facts can be combined before the query, so the tasks do not prescribe a retrieval algorithm. Appendix B gives the environment rules.

Transport and Power each provide 384 complete training trajectories; Assembly provides 486. Evaluation uses 128 prespecified Transport trajectories and the full Power and Assembly supports. These are *training-support readouts*. All models receive native 128×128 RGB, actions, and public remaining time, without coordinates or hidden states. All real frames are eligible for memory writing. Both memory variants receive known invocation times; only the aligned variant receives query–evidence correspondences, exclusively for constructing loss targets.

Comparisons. We compare the ordinary recurrent world model (Ordinary) with two training recipes for Associative Dreamer: prediction losses alone (FW), and prediction losses with EMA-based Value alignment (EMA+V). FW retains the same stochastic Reader and call schedule but removes alignment, correspondence annotations, and the target branch. The memory variants have matched online architectures and initialization; Ordinary shares the backbone initialization. This is a system comparison, not an equal-parameter or equal-compute comparison.

Transport and Power use 20,000 updates from scratch; Assembly uses 40,000 total updates through continuation of the same model, optimizer, and applicable EMA state. Each update uses four complete episodes with equal episode weighting, learning rate 4×10^{-5}, and 48 initial warmup updates. Arms share training support and episode schedule within each task; evaluation occurs every 500 updates. All tasks use model seeds 17, 29, and 43 in the matched summaries. Data seed is 17 throughout. Superscripts n count independent training runs, not significance. Appendix C gives continuation details.

Prediction and metrics. H1 predicts a physical consequence in one step from a later observed prefix. Open-loop prediction starts earlier and receives no intermediate true RGB: two actions in Transport and Power, and two or three in Assembly, depending on the processing plan. We average RGB predictions from two fixed sampling streams before classifying the physical outcome. Outcome regions and references are evaluator-only; they do not provide crop-based training supervision.

Transport averages four outlet–action groups and Power six target–action groups. Assembly first balances the two reachable classes within each material–plan group, then averages six groups; its majority-class predictor scores $2/3$ in raw accuracy but $1/2$ in balanced accuracy. For each run, we average the five checkpoints over 18k–20k for Transport and Power, and 38k–40k for Assembly. We then average these summaries across model seeds. *Worst* averages each run's lowest group-window mean; *Last* separately averages scores at the corresponding budget endpoint. Checkpoints and prediction samples are not independent training runs. Appendix C also reports event reward MAE on each task's own scale.

5.2 HISTORY-DEPENDENT PHYSICAL PREDICTION

Evidence alignment improves Transport and Power (Table 1). In Power, EMA+V achieves 100% open-loop accuracy in every group throughout the final window in all three runs; Ordinary scores 50% and FW 50.61%. Transport shows a similar contrast: 99.58% for EMA+V versus 50% for both controls. Adding the memory architecture alone is therefore insufficient to recover the aligned system's accuracy on these supports under the tested recipe.

At 40k in Assembly, EMA+V and FW average 77.01% and 75.97% balanced open-loop accuracy, versus 50% for Ordinary. Their means differ by only 1.04 percentage points, with substantial variation across runs: EMA+V scores 93.36%, 50.00%, and 87.65% (Appendix C). FW attains 100% H1

Table 1: Physical consequence prediction on fixed training supports (%, higher is better). Superscript n is the number of independent training runs for the entire row: All tasks use model seeds 17, 29, and 43. Scores average the last five scheduled checkpoints within the indicated budget; Last is the budget endpoint. Worst averages each run's minimum group-window mean. A dash denotes an unavailable matched summary or no Reader.

Task	Model	H1	Open	Worst	Last	No read
Transport (20k)	Ordinary	50.00	$50.00^{n=3}$	50.00	50.00	—
Transport (20k)	FW	50.00	$50.00^{n=3}$	50.00	50.00	50.00
Transport (20k)	EMA+V	100.00	$99.58^{n=3}$	98.75	100.00	50.00
Power (20k)	Ordinary	50.00	$50.00^{n=3}$	50.00	50.00	—
Power (20k)	FW	51.08	$50.61^{n=3}$	50.00	52.78	50.00
Power (20k)	EMA+V	99.76	$100.00^{n=3}$	100.00	100.00	50.00
Assembly (40k)	Ordinary	50.00	$50.00^{n=3}$	50.00	50.00	—
Assembly (40k)	FW	100.00	$75.97^{n=3}$	57.90	72.27	50.00
Assembly (40k)	EMA+V	83.52	$77.01^{n=3}$	60.56	78.34	50.00

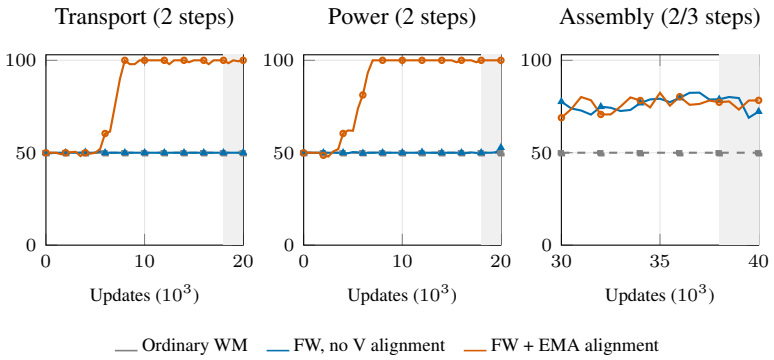

Figure 2: Open-loop learning curves using the same seeds as Table 1: Three runs are averaged for each task. Assembly shows the common 30k–40k continuation segment (note the different x-axis). Every scheduled readout in the displayed segment is shown; shading marks the task's fixed reporting window, not uncertainty.

accuracy but only 57.90% for the worst open-loop group; one-step success does not ensure consistent multi-step prediction.

Learning dynamics. Figure 2 uses the same runs as the table. Assembly shows the common 30k–40k continuation segment, with its reported window fixed at 38k–40k.

Reward prediction. EMA+V reduces event reward MAE from Ordinary's 5.00006 to 0.00937 in Power, and from 2.43854 to 1.28276 in Assembly; FW scores 6.58013 and 1.71707, respectively (Appendix Table 5). Transport evaluates terminal rewards over three- or four-step recorded continuations. These action suffixes respond to the visible docking outcome, making its reward score conditional on that response; all three methods achieve MAE below 0.0012 there. Each task retains its own event, horizon, and reward scale.

5.3 Use of Historical Evidence

Paired-history interventions distinguish responses to relevant facts from arbitrary changes in prediction (Table 2). For an earlier relevant event, we measure $S_{\mathrm{rel}} = \langle \Delta\hat{x}, \Delta x \rangle / \|\Delta x\|_2^2$ in the outcome region, where $\Delta\hat{x}$ and Δx are predicted and true image differences. A perfect response scores 1, ignoring the change scores 0, and reversing it scores -1. For irrelevant events, whose true outcome is unchanged, D_{irr} is the mean squared prediction difference. We average pairs within each intervention type, then types, checkpoints, and runs. EMA+V responds to relevant changes with sensitivities 0.8676 in

Table 2: Historical-evidence interventions with fixed current RGB and future actions. Related-event response S_{rel} and unrelated-event drift D_{irr} use the same runs and 18k–20k window as Table 1. Superscript n applies to both metrics. Sensitivity is not a percentage.

Task	Model	S_{rel} (target 1)	D_{irr} (target 0)
Transport	Ordinary	$0.0000^{n=3}$	3.73×10^{-13}
Transport	FW	$-0.0001^{n=3}$	1.03×10^{-5}
Transport	EMA+V	$0.8676^{n=3}$	4.18×10^{-5}
Power	Ordinary	$0.0000^{n=3}$	2.18×10^{-11}
Power	FW	$0.0031^{n=3}$	4.56×10^{-5}
Power	EMA+V	$0.8438^{n=3}$	5.77×10^{-6}

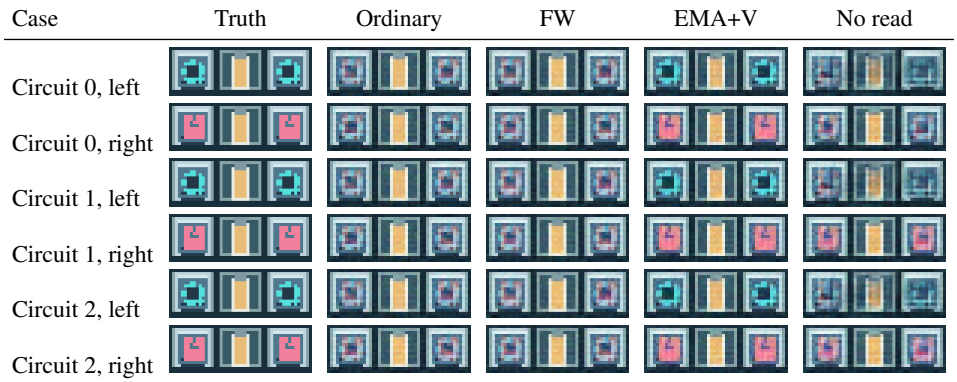

Figure 3: Power inspection: two-step open-loop predictions at update 20k, model/data seed 17. All six saved cases are shown. Within each circuit, the left/right pair shares the observed history and current RGB but supplies a different lateral action, followed by isolate. The two coolers either receive power or burn. No read disables future retrieval in EMA+V. Each cell shows the same camera-relative 48×16 physical strip for that action; predictions average streams 0 and 1 before display clipping and nearest-neighbor enlargement. Blur and errors are retained. These are action comparisons, not same-action history interventions.

Transport and 0.8438 in Power, while drift from irrelevant events remains small. Ordinary is nearly invariant to both types: low drift alone does not indicate successful use of history.

Disabling future reads returns memory models to 50% accuracy (Table 1). The observed prefix remains intact; the intervention disables prior conditioning and the reward head's direct read. This establishes use of retrieval; its architectural necessity requires alternative-memory controls. Under local route perturbations, EMA+V retains 100% two-step accuracy in Transport and Power across three seeds at 20k (Appendix D.4).

Figure 3 shows what this pathway changes in the seed-17 Power model. Each circuit shares an observed history and current image across left and right isolation actions. EMA+V predicts the corresponding powered or burnt coolers; Ordinary, FW, and read-disabled EMA+V produce less differentiated outcomes. Appendix Figure 5 complements this action comparison by holding current RGB and actions fixed while changing earlier relevant or irrelevant facts. All cases follow task specifications without consulting predictions and use the 20k checkpoint. Blur and errors are retained.

5.4 Effect of the target update

The Transport ablation keeps Value alignment in both arms and changes only the target update. The online stop-gradient target yields 78.44% open-loop window accuracy and 58.75% for the worst group; EMA reaches 100% on both (Table 3). The online-target run falls from 87.50% at 18.5k to 68.75% at 20k. This comparison supports the EMA choice in this training recipe, with one run per target. FW versus EMA+V in the main comparison changes both the alignment signal and target construction and cannot isolate the EMA effect.

Table 3: Transport target-update comparison with V alignment in both arms. All scores are percentages from one training run per arm ($n = 1$, seed 17). Open and worst group use the same fixed window; last denotes the final checkpoint.

Loss target	Open-loop window	Worst group window	Last
Online SG target	$78.44^{n=1}$	58.75	68.75
EMA target	$100.00^{n=1}$	100.00	100.00

6 Discussion and Limitations

Associative Dreamer treats historical evidence as an explicit input to prediction. The recurrent state represents the evolving scene, while the associative pathway provides access to earlier observations through learned queries. This division connects remembering to predicting: retrieved content conditions the latent prior from which action consequences are generated. It does not enforce an exclusive separation of recent and remote information; the recurrent state and memory may encode overlapping facts. Prediction losses teach the model to use retrieved content, while Value alignment directly supervises what the Reader recovers.

Assumptions and retrieval learning. The current formulation assumes known retrieval times, and its aligned variant requires query–history correspondences during training. One direction is to learn when to retrieve and how to identify relevant evidence without explicit correspondence labels. Moreover, alignment matches learned visual representations, whose retention of symbolic facts depends on the learned encoding. EMA controls target drift but cannot ensure that those representations retain every detail needed for prediction, or that matching a Value is sufficient to generate the correct consequence.

Memory and rollout boundaries. A fixed-size matrix compresses many associations into shared parameters. In our implementation, M_t and P_t together contain 528 FP32 values (2,112 bytes), independent of history length; the RSSM state is stored separately. Both matrices are reset at the start of each episode. No episode-specific memory is transferred between trajectories. Correlated Keys and competing contents can interfere; covariance-aware updates do not make storage lossless. The Reader also uses a prespecified budget J: an inaccurate read can affect subsequent queries, and the model does not learn when to stop. During imagination, memory is read-only. Predicted consequences evolve through the RSSM but cannot become new stored associations for later retrieval. This preserves the distinction between observed experience and imagined content, while leaving longer chains of hypothetical evidence outside the current memory update rule.

Toward planning and model-based reinforcement learning. A natural extension is to compare candidate action sequences from the state and memory reconstructed from the same real history, using retrieved evidence to condition their predicted consequences, rewards, and continuation. A planner could execute the first action and update memory with the resulting real observation before replanning. Alternatively, actor and value networks could learn from memory-conditioned latent rollouts following Dreamer's imagination-based approach (Hafner et al., 2025a). Such extensions require a retrieval-call rule available from observed or predicted states, an interface through which the policy can use memory, and control of prediction errors under policy-selected actions. Whether this improves closed-loop decisions is a question for future work.

7 Conclusion

We introduced Associative Dreamer, which learns to store visual associations, retrieve relevant evidence, and use it to predict action consequences. The model combines bounded episodic memory with recurrent dynamics and an optional evidence-alignment objective. The controlled prediction results show that retrieved history can change generated physical consequences. This work connects the learning of useful memory reads to their role in world-model prediction and motivates their future use in planning and model-based reinforcement learning.

AI USE STATEMENT

The research ideas and primary software architecture were conceived and designed by the human authors. AI assistants helped with literature and related open-source code searches, experimental code generation and debugging, test development and execution, checks of code and data correctness, discussions of experimental design, result organization, manuscript drafting and language polishing, and figure preparation. Image-generation tools were used to explore layouts for the method illustration, which was ultimately drawn as editable vector graphics. Numerical results come from actual experimental records, and prediction images come from saved model outputs. The human authors led the research decisions and conclusions and take full responsibility for the accuracy and integrity of the paper, code, data, and references.

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*, pp. 15619–15629, 2023.

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019.

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

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021.

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

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b.

Simon Haykin. *Adaptive Filter Theory*. Pearson, 5th edition, 2014.

Zizhan He, Maxime Daigle, and Pouya Bashivan. Building spatial world models from sparse transitional episodic memories. In *International Conference on Learning Representations*, 2026.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1400–1409, 2016. doi: 10.18653/v1/D16-1147.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *International Conference on Learning Representations*, 2024.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366, 2021a.

Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. Learning associative inference using fast weight memory. In *International Conference on Learning Representations*, 2021b.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *Journal of Machine Learning Research*, 27 (169):1–41, 2026.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

Jiwen Yu, Jianxiong Gao, Jianhong Bai, Yiran Qin, Kaiyi Huang, Quande Liu, Xintao Wang, Pengfei Wan, Kun Gai, and Xihui Liu. MemLearner: Learning to query context memory for video world models. In *European Conference on Computer Vision*, 2026.

A Deriving a memory-augmented RSSM from first principles

The derivation proceeds in three stages. First, we add a deterministic memory to an ordinary RSSM and derive its usual prediction–KL objective. Next, we make the retrieved evidence stochastic. Finally, we explain how a historical reference motivates the additional Value-alignment signal. This order separates *adding a memory to the predictor* from *adding supervision for retrieval*.

A.1 What is observed, what is latent, and what is optimized?

Write $y_t = (x_t, r_t, c_t)$ for the observed RGB, reward, and continuation indicator. The actions $A = a_{0:T-1}$ are supplied. We seek parameters that give high probability to the observed outcome sequence $Y = y_{1:T}$:

$$\max_{\theta,\phi} \log p_{\theta,\phi}(Y \mid A, \mathcal{C}_0). \tag{A1}$$

Here θ denotes world-model parameters and ϕ memory parameters. $\mathcal{C}_0$ fixes the starting RSSM state $S_0 = (h_0, z_0)$ and memory (M_0, P_0) after the initial observation or an observed prefix. Public clock values and the known retrieval schedule are also supplied conditions; we suppress them in sequence-level notation. This is a likelihood conditional on that realized starting state. A likelihood conditional only on raw history would additionally integrate its initial-state uncertainty.

Symbol	Role in the derivation
x_t, a_t, r_t, c_t	Observed image, supplied action, reward, continuation
h_t	Deterministic recurrent state
z_t	Stochastic latent state, inferred during training
M_t, P_t	Causally updated memory matrix and inverse covariance
m_t, d_t	Retrieved features for the prior and reward head
$\mathbf{E}_t$	Stochastic retrieved evidence, introduced in Section A.5

A deterministic state is not an additional independently sampled latent variable. Once a latent path and its observation prefix are specified, h_t, M_t, P_t are determined by their update rules. Importantly, M_t is not a supplied summary of the whole trajectory: it cannot use $x_{t+1:T}$.

A.2 Start with an ordinary RSSM

After observing x_t, an ordinary RSSM uses the current state and the next action to form a deterministic transition and a stochastic prior:

$$h_{t+1} = F_\theta(h_t, z_t, a_t), \qquad z_{t+1} \sim p^0_\theta(z_{t+1} \mid h_{t+1}). \tag{A2}$$

The decoder and outcome heads then predict y_{t+1}. Denote their normalized joint emission by $b^0_\theta(y_{t+1} \mid h_{t+1}, z_{t+1})$. The next image becomes available to the training posterior $q_\psi(z_{t+1} \mid h_{t+1}, x_{t+1})$, whereas the prior must predict without that image.

For $Z = z_{1:T}$, the ordinary joint model factors as

$$p^0(Y, Z \mid A, \mathcal{C}_0) = \prod_{t=0}^{T-1} p^0_\theta(z_{t+1} \mid h_{t+1}) b^0_\theta(y_{t+1} \mid h_{t+1}, z_{t+1}). \tag{A3}$$

The action does not disappear: it enters h_{t+1} through the deterministic transition. The recurrent states in this product are recomputed along each candidate latent path Z.

A.3 Add memory before changing the loss

To isolate the effect of adding memory, first consider a deterministic Reader. This is an explanatory intermediate model, not a change to the stochastic Reader used in our experiments:

$$(m_t, d_t) = \mathrm{Read}_\phi(S_t, M_t, \chi_t), \qquad p^M_\theta(z_{t+1} \mid h_{t+1}, m_t). \tag{A4}$$

The raw retrieved feature m_t changes the latent prior. The fused feature d_t also conditions the reward head. Consistently with the main model, define

$$\begin{aligned} b^M_{\theta,t} := {} & p^x_\theta(x_{t+1} \mid h_{t+1}, z_{t+1}) \\ & \times p^r_\theta(r_{t+1} \mid h_{t+1}, z_{t+1}, d_t, \tau_{t+1}) p^c_\theta(c_{t+1} \mid h_{t+1}, z_{t+1}, \tau_{t+1}). \end{aligned} \tag{A5}$$

Thus, conditioning the predictor on memory does not require a direct memory input to every head. After x_{t+1} is observed, the writer updates

$$(M_{t+1}, P_{t+1}) = \mathrm{Write}_\phi(M_t, P_t, S_t, a_t, x_{t+1}). \tag{A6}$$

The order is *read old memory, predict, receive the new image, write*. During evaluation of a likelihood on an observed sequence, the writer uses that sequence's observed prefix. In generative sampling it would use the prefix sampled so far; neither version looks into the future.

The augmented joint has the same product structure as the ordinary RSSM:

$$p^M(Y, Z \mid A, \mathcal{C}_0) = \prod_{t=0}^{T-1} \underbrace{p_\theta^M(z_{t+1} \mid h_{t+1}, m_t)}_{\text{memory-conditioned prior}} \underbrace{b_{\theta,t}^M}_{\text{predicted image and outcomes}}. \tag{A7}$$

Both the joint and the inference distribution use the same deterministic update rules for each realized path. No distribution $q(M_t)$ or independently sampled memory has been introduced, so no separate memory KL is needed.

A.4 Derive the memory-RSSM ELBO step by step

Step 1: marginalize the unknown latent states. The observed sequence likelihood is obtained by summing or integrating out Z:

$$\log p^M(Y \mid A, \mathcal{C}_0) = \log \int p^M(Y, Z \mid A, \mathcal{C}_0)\, dZ. \tag{A8}$$

For our categorical latents, the integral denotes a sum. Direct enumeration over all latent sequences is expensive.

Step 2: introduce a tractable approximate posterior. Use the causal inference model

$$q_\psi(Z \mid Y, A, \mathcal{C}_0) = \prod_{t=0}^{T-1} \underbrace{q_\psi(z_{t+1} \mid h_{t+1}, x_{t+1})}_{q_{t+1}}. \tag{A9}$$

It can examine the new image when explaining a recorded transition. Unlike the prior, it is not the distribution used to imagine the next unseen state.

Step 3: multiply and divide by that posterior, then apply Jensen. Suppress the common conditions on the right. Assuming common support and finite displayed expectations,

$$\begin{aligned} \log p^M(Y \mid A, \mathcal{C}_0) &= \log \mathbb{E}_q\left[\frac{p^M(Y, Z \mid A, \mathcal{C}_0)}{q(Z \mid Y, A, \mathcal{C}_0)}\right] \\ &\geq \mathbb{E}_q[\log p^M(Y, Z \mid A, \mathcal{C}_0) - \log q(Z \mid Y, A, \mathcal{C}_0)] =: \mathcal{L}_M. \end{aligned} \tag{A10}$$

This is the evidence lower bound (ELBO): a tractable quantity no larger than the observed-data log likelihood. Exact equality in the importance-expectation step requires the inference distribution to cover the posterior support. More generally the usual KL-gap identity below establishes the lower bound when q is absolutely continuous with respect to the true posterior.

Step 4: expand the products into sums. Let $p_{t+1}^M = p_\theta^M(z_{t+1} \mid h_{t+1}, m_t)$. Substitution gives

$$\mathcal{L}_M = \sum_{t=0}^{T-1} \mathbb{E}_q[\log b_{\theta,t}^M + \log p_{t+1}^M - \log q_{t+1}]. \tag{A11}$$

The last two terms are the negative KL between the training posterior and the memory-conditioned prior, averaged over the already sampled prefix:

$$\boxed{\mathcal{L}_M = \sum_{t=0}^{T-1} \mathbb{E}_{q_{1:t}}\left[\mathbb{E}_{q_{t+1}} \log b_{\theta,t}^M - D_{\mathrm{KL}}(q_{t+1} \| p_{t+1}^M)\right].} \tag{A12}$$

Here $q_{1:t}$ is the marginal over the previous latent states $z_{1:t}$; at $t = 0$ the starting context is fixed. Keeping this expectation matters: the same observed history can give different sampled latent paths and therefore different memory writes and queries.

What the two terms teach. The first term trains predictions of the image, reward, and continuation. The second teaches a predictor that has not seen x_{t+1} to match a posterior that has. In an ordinary RSSM the prior conditions on h_{t+1}; with memory it also conditions on m_t. Memory parameters receive gradients through the prediction paths that use their reads. The absence of a separate memory KL does not mean the memory is untrained.

The bound's gap is exactly

$$\log p^M(Y \mid A, \mathcal{C}_0) - \mathcal{L}_M = D_{\mathrm{KL}}\big(q(Z \mid Y, A, \mathcal{C}_0) \,\big\|\, p^M(Z \mid Y, A, \mathcal{C}_0)\big) \geq 0. \tag{A13}$$

Adding memory changes the model inside this bound. It does not automatically make this bound tighter than the bound of a different, memory-free model.

A.5 Now make the retrieved evidence stochastic

Let J be a preset retrieval depth. The Reader samples an ordered bundle of J Values rather than returning only a deterministic feature:

$$\begin{aligned} \mathbf{E}_t &= (e_t^{(1)}, \ldots, e_t^{(J)}), \\ \pi_\phi(\mathbf{E}_t \mid \mathcal{C}_t) &= \prod_{j=1}^{J} \mathcal{N}\Big(e_t^{(j)}; \mu_t^{(j)}(\mathcal{C}_t, e_t^{(<j)}), \sigma^2 I\Big). \end{aligned} \tag{A14}$$

The derivation allows a general preset J; the implementation and all experiments in this paper use $J = 2$. $\mathcal{C}_t$ denotes the current causal state, including memory and public information. The raw concatenation m_t and fused feature d_t are functions of these actual samples. On inactive calls, both model and inference use the same empty atom $\mathbf{E}_t = \varnothing$, with $m_t = d_t = 0$ and no sampling. A zero Gaussian mean on an active call is not this empty branch. We write $\pi_\phi^0 = \delta_\varnothing$ and $\pi_\phi^1 = \pi_\phi$; the superscript selects a branch, not a power.

Let $\mathbf{E} = \mathbf{E}_{0:T-1}$. The stochastic joint adds one factor per call:

$$p(Y, Z, \mathbf{E} \mid A, \mathcal{C}_0) = \prod_{t=0}^{T-1} \pi_\phi^{\chi_t}(\mathbf{E}_t \mid \mathcal{C}_t)\, p_{t+1}^M\, b_{\theta,t}^M. \tag{A15}$$

We now integrate both Z and $\mathbf{E}$ for the predictive likelihood. A simple valid inference choice uses the same Reader and the image posterior:

$$Q_\pi(Z, \mathbf{E} \mid Y, A, \mathcal{C}_0) = \prod_{t=0}^{T-1} \pi_\phi^{\chi_t}(\mathbf{E}_t \mid \mathcal{C}_t)\, q_{t+1}. \tag{A16}$$

For each realized path, the $\log \pi_\phi$ terms cancel in $\log p - \log Q_\pi$. Repeating Steps 3–4 yields

$$\mathcal{L}_\pi = \mathbb{E}_{Q_\pi} \sum_{t=0}^{T-1} [\log b_{\theta,t}^M + \log p_{t+1}^M - \log q_{t+1}]. \tag{A17}$$

The expectation now includes retrieved samples as well as latent states. The explicit matching penalty cancels, while both the sampled evidence and the predictions remain dependent on the Reader parameters. This explains how a stochastic memory model can be trained through prediction without a separate Value target.

A.6 Where can an additional evidence KL come from?

During supervised training, relevant historical observations are known. Introduce a distribution ρ_t centered on those historical reference Values. It is a training-side proposal for useful evidence, not an additional deployment input. At one fixed context, replacing π_t by ρ_t in the inference distribution gives

$$\widetilde{q}_t(\mathbf{E}_t, z_{t+1}) = \rho_t^{\chi_t}(\mathbf{E}_t)\, q_{t+1}. \tag{A18}$$

The proposal may use the observed history and the training correspondence $\mathcal{R}$; these conditions are suppressed only to keep the expression short. Expanding the same $\log p - \log \widetilde{q}_t$ calculation now

leaves $\log \pi_t - \log \rho_t$. Consequently,

$$\mathcal{L}_{\rho,t} = \underbrace{\mathbb{E}_{\rho_t^{\chi_t} q_{t+1}} \log b_{\theta,t}^M}_{\text{predict the observed outcome}} \underbrace{- \mathbb{E}_{\rho_t^{\chi_t}} D_{\mathrm{KL}}(q_{t+1} \| p_{t+1}^M)}_{\text{match the state posterior}} - \underbrace{\chi_t D_{\mathrm{KL}}(\rho_t \| \pi_t)}_{\text{match historical evidence}} . \tag{A19}$$

The extra KL appears because two *different evidence distributions* are being matched. When $\chi_t = 0$, the common empty atom makes this KL zero. A sequence-level $\mathcal{L}_\rho$ also averages each $\mathcal{L}_{\rho,t}$ over the earlier latent states and evidence drawn from $Q_\rho = \prod_s \rho_s^{\chi_s} q_{s+1}$, with $\rho_s^0 = \delta_\varnothing$ and $\rho_s^1 = \rho_s$. Contexts are recomputed with the shared causal update rules. An arbitrary historical proposal gives a valid bound under the stated support conditions but need not approximate the true outcome-conditioned posterior well.

A.7 From the evidence KL to a Value-matching loss

For one hop at a fixed prefix, choose two Gaussians in the same Value coordinates with the same positive covariance:

$$\rho_j = \mathcal{N}(\bar{v}_{\mathcal{R}(t,j)}, \sigma^2 I), \qquad \pi_j = \mathcal{N}(\mu_t^{(j)}, \sigma^2 I) \quad \Longrightarrow \quad D_{\mathrm{KL}}(\rho_j \| \pi_j) = \frac{\|\mu_t^{(j)} - \bar{v}_{\mathcal{R}(t,j)}\|_2^2}{2\sigma^2}. \tag{A20}$$

This is the origin of the squared Value error and its scale. The reference vector specifies a Gaussian mean; treating it as a Dirac distribution against a positive-variance Gaussian would not yield this finite KL.

For a J-step Reader, the exact chain rule is

$$D_{\mathrm{KL}}(\rho_t \| \pi_t) = \sum_{j=1}^{J} \mathbb{E}_{\rho_t(e_t^{(<j)})} D_{\mathrm{KL}}(\rho_j(\cdot \mid e_t^{(<j)}) \| \pi_j(\cdot \mid e_t^{(<j)})). \tag{A21}$$

Thus a full ρ-ELBO would use proposal-generated evidence both to predict the outcome and to form the earlier-hop context for this KL. The slot order is provided by the historical correspondence; this is not an unordered-set loss.

A.8 What is actually optimized in this paper?

Our forward pass always retrieves from its own memory, using π, including during training. The reference Value is evaluated afterward and only supplies the stopped target in

$$\mathcal{J}_{\mathrm{impl}} = \frac{1}{B} \sum_{n=1}^{B} \left[L_{\mathrm{WM}}^{(n)} + \frac{1}{2\sigma^2 N_n} \sum_{t:\chi_t^{(n)}=1} \sum_{j=1}^{J} \|\mu_t^{(n,j)} - \operatorname{sg} \bar{v}_{\mathcal{R}^{(n)}(t,j)}\|_2^2 \right]. \tag{A22}$$

This matches Equation 4, evaluated at $J = 2$ in our experiments. Here n indexes episodes in a batch of size B, and N_n counts all real observation rows, including the initial row. The numerator sums all J retrieval steps and all Value coordinates; J denotes depth, whereas $\mathcal{J}_{\mathrm{impl}}$ denotes the objective. Equal-length groups are weighted by episode count. The block derivation above conditions on the already encoded initial observation, whereas the implementation also trains its reset-row targets.

There are two distinct choices here. The prediction path has the structure of Q_π in Equation A16; the auxiliary penalty uses the Gaussian matching form but at student-generated prefixes. It is therefore not the historical-proposal objective $\mathcal{L}_\rho$. Subtracting a nonnegative penalty from an exact lower bound would still give a looser scalar lower bound, but would not make it the ρ-ELBO or establish a tighter bound.

The actual L_{WM} also uses Dreamer's engineering choices: summed RGB squared error, reward two-hot cross-entropy, binary cross-entropy against γc with $\gamma = 1 - 1/333$, and $\max(1, D_{\mathrm{KL}}(\operatorname{sg} q \| p)) + 0.1 \max(1, D_{\mathrm{KL}}(q \| \operatorname{sg} p))$. KL balancing, free nats, straight-through categorical gradients, and soft targets must be distinguished from the exact normalized-likelihood derivation. The EMA reference

copies only the online encoder and Value projection at initialization, stays outside the optimizer and online state, and is updated once after each full optimizer step with decay 0.995. Online writes retain gradients. None of these choices proves convergence or prevents collapse.

Finally, observed-sequence learning writes real observations after prediction. Our short open-loop evaluation instead freezes M/P at its origin. That read-only rollout is an explicit evaluation convention, not automatically the exact multi-step marginal of the write-enabled sequence model.

Summary. Adding deterministic memory changes what the RSSM prior can condition on, while retaining the familiar prediction term and state KL. Making retrieval stochastic adds an evidence variable. Matching a separate historical proposal then explains an evidence KL, whose Gaussian form motivates the implemented auxiliary Value loss. The implementation keeps that target entirely outside the prediction path.

B TASK ENVIRONMENTS AND PREDICTION PROBLEMS

The three environments implement different physical dependencies: transporting a released object, protecting a damaged circuit, and processing material with an unobserved internal state. Earlier events change the later scene through the environment dynamics. Our experiments model recorded action sequences; they do not train a policy to complete the games.

Figure 4 shows each task's relevant evidence, other-object interference, an intervening scene, the later query, and the resulting physical change. For these examples, the query follows the displayed later relevant fact by 26, 44, and 20 environment steps, respectively. These timelines document the observation sequence; separating the effects of delay and interference requires controlled comparisons.

B.1 COMMON OBSERVATIONS AND EXPERIMENTAL BOUNDARY

Each observation is a native 128×128 RGB frame. Transport and Power use a forward-facing camera covering seven columns and five rows; horizontal movement does not rotate the agent. Nearby machinery exposes its pictograms in a readable detail panel. Earlier stations leave the camera as the agent advances. Assembly instead shows a stationary workbench, where successive displays replace earlier evidence. Public clocks and visible machine counters remain in the images. The model also receives executed actions and public remaining time, but not coordinates, pose, hidden configurations, or the environment's action mask. Rewards and continuation indicators are prediction targets, not additional observation features.

Routes and processing plans are supplied in advance, independently of the hidden correct choice. Subsequent recovery may respond to damage that has already become visible. Both memory models receive the same known retrieval times. Only EMA+V receives the correspondence between a query and its two relevant past observations, exclusively for its alignment loss; all real frames remain eligible for writing. The readouts below use the training support, with no held-out generalization claim.

B.2 TRANSPORT: SOURCE AND DIVERSION DETERMINE A LATER BRIDGE

The agent carries cargo through an outdoor yard with two symbol-marked waterways, obstacles, and a downstream dock. Its actions are *forward*, *left*, and *right*. Passing each upstream pedal releases a floating platform from a visible left or right source. A later mechanism reveals whether that waterway preserves or swaps the sides. Travel between stations and observations of the other waterway separate the relevant events from the dock. A symbol at the dock identifies the queried waterway, without showing the incoming platform.

For source side $s \in \{-1, +1\}$ and diversion bit b, the arrival side is $s(-1)^b$. A lateral docking action selects a latch; the platform arrives two actions after entering the dock. A matching latch forms a bridge, whereas a mismatch leaves the crossing blocked. Thus, flipping either relevant fact changes the physical consequence under identical dock RGB and actions. Changing the other waterway does not. The complete game offers cargo delivery for $+10$ or safe storage for $+2$, before action costs and optional subgoal rewards. These game rewards are not measured policy returns.



Figure 4: Real observation timelines from the fixed task examples, ordered left to right. Each row includes two relevant facts, an unrelated object's clue, an intervening scene, the query, and its physical outcome. Other waterway, Other circuit, and Other valve show information about a non-target waterway, circuit, or unpulsed valve. Labels and timestamps are outside the model input. Reset is $t = 0$; $t_q - t_B$ measures the interval from the displayed later relevant fact to the query, including any subsequent visible frames of that fact. The last visible evidence frame can occur later than t_B. The remaining intermediate frames are omitted from the figure and retained in training. Every tile contains the complete native 128×128 RGB, with nearest-neighbor enlargement.

The training support crosses 32 physical configurations, two obstacle detours, two short excursions, and three dock actions: 384 trajectories. The fixed readout uses 128 of these, retaining one excursion and the two lateral actions. Its observed dock prefix ends at step 36; open-loop prediction applies the lateral action and then *forward*, reaching the consequence at step 38. H1 observes the intervening frame and predicts the same consequence. Classification uses only the outlet's physical image region and averages four waterway–action groups. Straight-ahead storage trajectories remain in training but are not part of this history-dependent consequence score.

B.3 Power: Wiring and Overload Determine Safe Isolation

The agent crosses an indoor service corridor with partitions and equipment cabinets. Three symbol-marked circuits each connect a left and a right branch to triangle and circle modules. Wiring windows expose these connections; later test windows show which module was damaged by an actual overload. The route visits all six windows, so evidence about two non-target circuits acts as interference. The target circuit is identified publicly, including at the final service station, but its wiring and damage are no longer visible there. Actions are *forward*, *left*, *right*, and *isolate*.

Let $w(s)$ denote the module connected to branch s, and let d be the damaged module. Isolating s preserves power exactly when $w(s) = d$; otherwise the damaged branch remains live and the controller burns. Either a wiring swap or a changed overload can therefore reverse the consequence while the current station image and supplied action remain unchanged. The rendered outcome directly

shows running or burnt equipment. Correct isolation permits cooling and eventual delivery for $+20$; burning or spoilage incurs -10, isolation costs 2, and actions and movement also have costs.

All 2^3 wiring assignments, 2^3 overload assignments, three target circuits, and two isolation sides form 384 complete trajectories, split evenly between preserved power and burning. Fixed evaluation uses all 384. The agent reaches the station at step 58, moves sideways at 59, and isolates at 60. H1 predicts the isolation consequence from the observed step-59 prefix; two-step open-loop prediction starts at 58 without receiving frame 59. The score compares the two cooler regions and averages six target–side groups. Successful trajectories continue to delivery; failed ones stop at the actual terminal event, without padded post-terminal frames.

B.4 Assembly: a heat pulse changes hidden material state

A stationary workbench has feed, heat, and release controls. Actions move the cursor *left* or *right*, or *press* the selected control. During preparation, three displays successively show which patterned material each valve serves. The mapping may be many-to-one. A visible pulse at step 12 preheats the material connected to the selected valve. The pulse display then disappears; the first wrapped plate arrives at step 32. The wrapper identifies the material but conceals its curing state. The other valve connections and the intervening preparation frames supply interference and delay.

For first material u, valve mapping c, and pulsed valve v, its initial heat level is $q = \mathbf{1}[c(v) = u]$. The supplied plan adds either zero or one heat press before release. Total levels zero, one, and two produce torn, flat, and brittle plates, respectively. The same wrapped plate and processing actions can therefore yield different visible deformation after different histories. A flat plate reaches the rack one step after release and earns $+6$; a damaged plate jams the line and requires clearance. The full game processes three plates with two available heat units and ends when all have been delivered or discarded.

The support contains 27 mappings, three pulse valves, three cyclic arrival orders, and two first-plate plans, giving 486 trajectories. Training includes all three plates, but the principal score concerns the first. Prediction starts at arrival and reaches its released shape after two or three actions, depending on the heat press; H1 starts immediately before release. Reward evaluation extends one further step to include actual delivery. Within each of six material–plan groups, preheated and unheated cases have ratio $1 : 2$. We therefore balance the two reachable outcome classes before averaging groups: a majority guess scores $2/3$ raw accuracy but $1/2$ balanced accuracy. The reference shape region excludes delivery counters and feedback icons.

C Implementation and Reporting Details

The backbone uses 2,048 deterministic units, 32 categorical variables with 16 classes each, eight recurrent blocks, and hidden width 256. Native RGB observations are 128×128. Memory uses key width 12, Value width 32, Reader width 32, two reads per invocation, and zero write delay. A key memory matrix and its inverse-covariance state contain $32 \cdot 12 + 12^2 = 528$ FP32 entries (2,112 bytes) per episode, excluding the RSSM state and activations. For the three-action Transport and Assembly tasks, Ordinary and memory models have 12,009,411 and 12,536,988 online trainable parameters, respectively; four-action Power uses 12,009,667 and 12,537,308. The EMA arm additionally maintains 351,520 target parameters, outside the online optimizer and deployed state.

Writer initialization and numerics. Reset clears the RSSM state and initializes $M = 0$ and $P = 100I$ before the first observation, whose write uses a zero pre-transition state and action. The implementation symmetrizes P after each update to limit numerical asymmetry. For fixed associations and weights, the weighted RLS update is equivalent to standard ridge RLS applied to $(\sqrt{\beta_i}k_i, \sqrt{\beta_i}v_i)$; there is no temporal forgetting factor.

Reader feedback and prior initialization. Feedback projects the actual sampled Value, caps the projection's RMS at one, applies $\tanh$, and multiplies it by a sigmoid gate computed from the preceding retrieval state. The cap applies only to feedback, not to the raw samples in m_t. With $J = 2$ and $d_v = 32$, m_t is 64-dimensional and the fused d_t is 32-dimensional. An active call with $M = 0$ still samples Gaussian noise, unlike an inactive call that skips the Reader. The additional prior-input

columns are initialized to zero, so the initial prior depends on h alone. The two memory variants share online initialization at each matched model seed.

Loss reduction and target updates. The world-model objective averages the RGB squared-error sum, reward two-hot cross-entropy, continuation binary cross-entropy, dynamics KL, and representation KL over real rows, including initial reset targets. Their scales are $1, 1, 1, 1, 0.1$, with one free nat for each KL. Dynamics KL stops the posterior and representation KL stops the prior, using the same pair of distributions. Continuation targets retain Dreamer's discounted-continuation convention. Alignment sums squared errors over Value coordinates and reads without an additional division by either dimension. Equal-length episode groups are weighted by episode count to preserve the episode-macro average. Targets are recomputed for each update. After the complete online optimizer step, the target parameters update once as $\bar{\theta} \leftarrow 0.995\bar{\theta} + 0.005\theta$; they have no optimizer and are not used to write M.

The current code fixes $J = 2$. The general J formulation requires matching hop embeddings, the Jd_v-wide prior input, and supervision correspondences at model construction. It does not imply that a trained checkpoint supports extra reads or adaptive stopping. Persistent M/P size is independent of J, but read computation, temporary evidence, and some slow parameters are not.

The learning-rate warmup, episode ordering, observation processing, invocation schedule, Gaussian sampling scale, and fixed evaluation subset are kept within each task's comparison. Gradient paths through online writes and reads remain active; stop-gradient applies to reference targets. Computing targets after the reference-free forward pass prevents the loss-side reference from becoming a prediction input. EMA updates once after each complete optimizer step. Real histories are replayed causally to produce the query state; future RGB is not provided to the open-loop predictor.

Seed 17 was used during method development; seeds 29 and 43 are fixed-recipe replications. The reported means are descriptive summaries of these runs, not an independent confirmatory test. Assembly is reported only at 40k, using the fixed 38k–40k window. The budget was extended after inspecting earlier results. Seed 17 continues through 20k–30k–40k, while seeds 29 and 43 continue from 20k to 40k. Model, optimizer, and applicable EMA state are restored without restarting warmup. Torch sampling state was not saved and is reseeded, so these continuations are not bitwise-equivalent to uninterrupted training.

Table 4: Per-run open-loop accuracy (%) underlying Table 1. Each entry is one training run's fixed final-window mean at the task budget.

Task	Model seed	Ordinary	FW	EMA+V
Transport	17	50.00	50.00	100.00
Transport	29	50.00	50.00	100.00
Transport	43	50.00	50.00	98.75
Power	17	50.00	50.05	100.00
Power	29	50.00	50.10	100.00
Power	43	50.00	51.67	100.00
Assembly	17	50.00	95.09	93.36
Assembly	29	50.00	64.78	50.00
Assembly	43	50.00	68.02	87.65

Table 5 reports the distinct reward events, complementing physical image scores. Transport uses terminal storage or delivery rewards, Power uses the isolation event, and Assembly uses delivery after release. For Transport, we separately evaluate the saved 18k–20k checkpoints from the observed dock prefix, following each recorded three- or four-action suffix to its terminal reward (1.9 for storage or 9.9 for delivery, including action cost). Prediction streams 0 and 1 are averaged before computing each absolute error; all 128 trajectories receive equal weight. The recorded suffix responds to the visible docking outcome, so this evaluates conditional terminal-reward fitting.

History-intervention aggregation. Table 2 uses the recorded two-step predictions, with the same prediction-sample averaging as the outcome scores. Relevant interventions change the target waterway's source or diversion in Transport, and the target circuit's wiring or overload in Power. Irrelevant interventions change the other waterway or either of the other two circuits. Drift is measured in the

Table 5: Event reward MAE (lower is better), averaged over each run's fixed window and then over the same seeds and budgets as Table 1. Task reward scales and horizons differ. Transport uses recorded-action terminal rollouts.

Model	Transport (20k)	Power (20k)	Assembly (40k)
Ordinary	$0.00091^{n=3}$	$5.00006^{n=3}$	$2.43854^{n=3}$
FW	$0.00117^{n=3}$	$6.58013^{n=3}$	$1.71707^{n=3}$
EMA+V	$0.00097^{n=3}$	$0.00937^{n=3}$	$1.28276^{n=3}$

fixed target outcome region. Types receive equal weight within each relevant/irrelevant category, followed by the fixed checkpoint window and the main table's model seeds.

D TASK-SPECIFIC PREDICTION RESULTS

The examples in the main text and this appendix complement the full-support scores. The Transport and Power examples use the final 20k checkpoint of each development-seed run, fixed across the displayed cases. Their numerical summaries below refer to those seed-17 case studies; Assembly instead reports all three runs at 40k. The multi-run summaries are in Table 1, with individual runs in Table 4. The displayed outcome images are two-step open-loop predictions in Transport and Power. H1 uses a later observed prefix and is not the first frame of these open-loop rollouts. All saved cases are retained in Figures 5 and 3, including incorrect and blurred predictions.

D.1 TRANSPORT: RESPONDING TO RELEVANT HISTORY

Figure 5 holds the dock view and future actions fixed while changing the earlier source or diversion. The true platform switches sides under either relevant change and stays on the original side when the other waterway changes. EMA+V predicts this distinction; Ordinary and FW produce much less differentiated outcomes. The complete 128-trajectory readout gives final-window accuracy of 100% for EMA+V and 50% for both controls. EMA+V also scores 100% in every waterway–action group; disabling future retrieval reduces its score to 50%.

The two retrieved Values are therefore useful to the deployed predictor in this recipe, but these results alone do not establish that two hops or a matrix are necessary. The completed FW comparison is important: it shares the memory interface and known call schedule, yet does not learn the dependency under the same update budget without Value alignment. Terminal reward prediction is reported separately in Table 5, using the recorded three- or four-action continuation from the dock prefix.

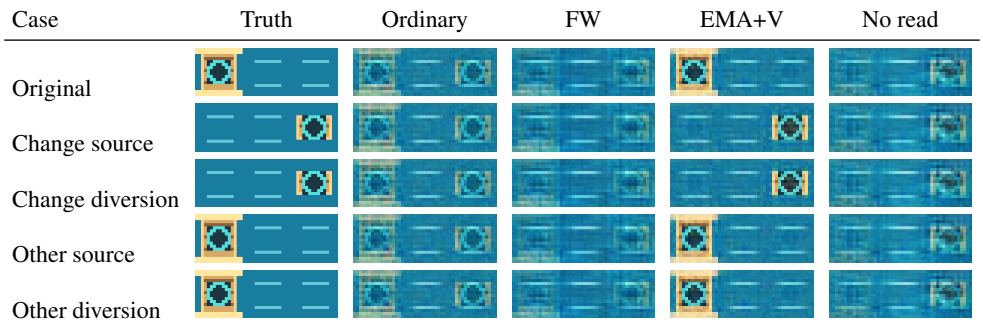

Figure 5: Transport at update 20k, model/data seed 17. All five saved cases share the same current RGB and actions (left, forward); only earlier events change. Source and diversion changes concern the target waterway; the last two rows change the other waterway. Cells show the same native 48×16 outlet region at the two-step consequence. All predictions are the saved mean over sampling streams 0 and 1, clipped to the display range and enlarged by nearest neighbor. The No read column disables future retrieval in EMA+V from its normally observed prefix. Blur and incorrect predictions are retained; these examples do not replace the full-support metrics.

D.2 POWER: PREDICTING AN ACTION'S PHYSICAL CONSEQUENCE

Figure 3 shows the saved six cases: three target circuits, each paired with left and right isolation. Wiring is direct and module zero is damaged in these examples, so left isolation leaves the coolers running and right isolation burns them. The choice changes the supplied action while holding observed evidence fixed. This figure illustrates action-conditioned prediction; history interventions are evaluated over the full support.

Across all 384 trajectories, EMA+V attains 100% final-window outcome accuracy in all six groups. Ordinary obtains 50%, and FW obtains 50.05%, with a 50% final checkpoint. Reward prediction shows the same contrast: the isolation-event MAE is 0.00121 for EMA+V, versus 5.00000 for Ordinary and 6.91114 for FW. These values measure the model's reward prediction error. Future-read disablement returns EMA+V to 50% outcome accuracy, while preserving the normally observed prefix. It disables both prior evidence and the reward head's direct read, so it does not isolate the prior path alone.

D.3 ASSEMBLY: PREDICTING HIDDEN MATERIAL STATE ACROSS PROCESSING ACTIONS

Figure 6 shows all six prespecified cases for the seed-17 model at 40k. Quantitative results below retain all three seeds. The shape crops use the same region across methods; neither cases nor checkpoints are selected by prediction quality.

Assembly tests whether the model carries the effect of an earlier heat pulse through subsequent processing. At first-plate arrival, the wrapper identifies the material but hides whether it was preheated. The earlier valve mapping and pulse jointly determine that state. Releasing immediately or heating before release produces a visible shape after two or three actions, respectively (Appendix B.4). The open-loop score therefore measures the ability to use earlier cues to predict subsequent physical transitions.

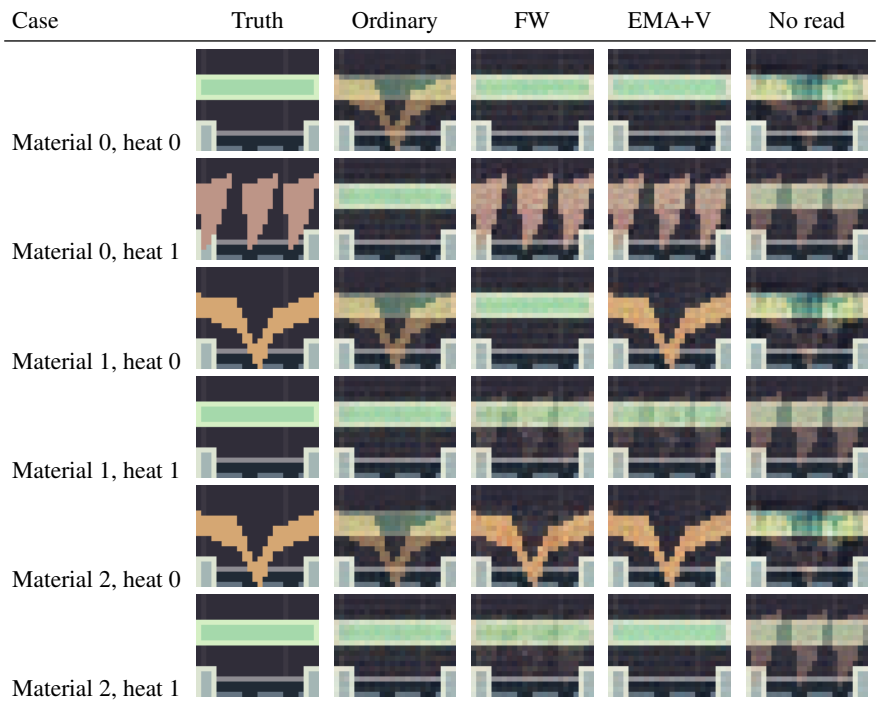

Figure 6: Assembly at 40k, model seed 17, data seed 17. All six prespecified material–plan cases are shown; heat 0/1 uses two/three open-loop actions. Columns compare the true released shape, three model predictions, and EMA+V with future reads disabled from the normally observed prefix. Each cell is the same native 24×20 plate region. Saved predictions average sampling streams 0 and 1; display uses clipping to $[0, 1]$, 8-bit RGB rounding, and nearest-neighbor enlargement. Errors and blur are retained. These examples do not replace the full-support, multi-checkpoint scores.

At 40k, the 38k–40k window over all 486 trajectories gives mean balanced open-loop accuracy of 50.00% for Ordinary, 75.97% for FW, and 77.01% for EMA+V. The close memory-model means conceal different run outcomes. For seeds 17, 29, and 43, FW scores 95.09%, 64.78%, and 68.02%; EMA+V scores 93.36%, 50.00%, and 87.65%, respectively (Table 4). Thus prediction losses alone can train a useful retrieval pathway here, while alignment does not consistently improve every run. The 1.04-point mean difference does not establish a reliable ordering between the two recipes.

The distinction between observed-prefix and open-loop prediction is also important. FW reaches 100% H1 accuracy in every run, yet its mean worst-group open-loop accuracy is 57.90%. EMA+V averages 83.52% on H1 and 60.56% on the worst open-loop group. H1 starts immediately before release, whereas the longer rollout must generate the intervening states without true images. These scores locate a gap between the two prediction settings; they do not by themselves identify retrieval or rollout error as its cause.

Disabling future reads yields 50% balanced consequence accuracy for both memory recipes in every run, removing their successful predictions' advantage while retaining the observed prefix. Delivery-event reward MAE, using the same windows and seeds, is 2.43854 for Ordinary, 1.71707 for FW, and 1.28276 for EMA+V (Table 5). The reward evaluation extends one action beyond release to include delivery; it measures the model's reward prediction error. Both memory recipes improve the mean consequence and reward predictions over Ordinary, but neither achieves uniformly accurate open-loop prediction across runs and groups.

D.4 Robustness to local route perturbations

We evaluate the fixed 20k models on new observation–action histories under the same physical configurations. Transport moves a two-action lateral excursion two steps later, changing RGB at steps 17–19. Power swaps adjacent actions at positions 22/23 and 38/39, changing RGB at steps 22 and 38. Both changes preserve trajectory length, all annotated evidence images and times, the query image and time, and all subsequent actions and images. Each perturbed action prefix differs from every training prefix. We verify these properties by replaying the real environment before model evaluation.

The comparison covers the same 128 Transport and 384 Power conditions as the original readouts, with model seeds 17, 29, and 43 and prediction streams 0 and 1. Each fixed model is evaluated on both routes without parameter updates. Original-route H1, H2, and read-disabled group accuracies exactly match the saved 20k evaluations. These are endpoint comparisons; the main table continues to report five-checkpoint window means.

Table 6: Local route perturbations: two-step outcome accuracy (%), averaged over model seeds 17, 29, and 43 at fixed 20k checkpoints ($n = 3$ per row). No read disables future retrieval on the perturbed route; a dash denotes no Reader.

Task	Model	Original	Perturbed	No read
Transport	Ordinary	50.00	50.00	—
Transport	FW	50.00	50.00	50.00
Transport	EMA+V	100.00	100.00	50.00
Power	Ordinary	50.00	50.00	—
Power	FW	52.78	52.78	50.00
Power	EMA+V	100.00	100.00	50.00

EMA+V retains 100% two-step accuracy on both routes in every run, and disabling future retrieval reduces its perturbed-route accuracy to 50%. Ordinary and FW also retain their original scores. Power FW's mean of 52.78% comprises two 50% runs and one 58.33% run. The results support robustness to these local route changes, which affect only a few intervening observations. Generalization to substantially new paths, physical configurations, or dynamics remains untested. Assembly is excluded because changing preparation actions leaves its observed images unchanged.