POLEWM: VISUAL WORLD MODELS WITH POLE-BASED ACTION RESPONSES

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual planning requires predicting how candidate actions unfold over time. Direct world models avoid recursive state rollout, but direct prediction alone does not specify how action effects should share temporal structure. We introduce PoleWM, which represents a latent trajectory as an initial-condition response plus delayed, action-conditioned responses. Nonlinear networks determine response coefficients from visual history and causal action prefixes; a shared pole basis determines their evolution with elapsed time. This factorization preserves nonlinear action interactions while providing exact initial-state anchoring, non-anticipation, and bounded responses to finite plans. We also characterize the additional basis direction needed to represent re-anchored tails, separating these structural properties from restart consistency. Across four image-goal control environments, PoleWM achieves 90.83% mean success over two training seeds, exceeding the strongest baseline mean by 6.08 percentage points and leading on three tasks. In a first-epoch Cube study with one training seed and no restart penalty, neural and pole response heads improve over a generic direct head by 20.67 and 23.34 points, respectively. Separately, optimized single-frame planning takes 74–87 ms per complete CEM solve on an A100. These results support explicit action-response factorization as an effective inductive bias for direct visual world models, with poles providing an analytically structured realization.

1 INTRODUCTION

World models make visual control possible by predicting the consequences of actions in a learned representation. From locally linear latent control to planning and policy learning through imagined trajectories, the organization of predictive dynamics has been central to this approach (Watter et al., 2015; Hafner et al., 2019; 2020). For image-goal planning, a compact interface is to encode the current observation and goal, predict a candidate plan's future embedding, and minimize its distance to the goal (Zhou et al., 2025; Maes et al., 2026). This places the predictive model directly inside a repeated search over actions.

One-step models reach distant times by repeatedly applying a transition to predicted states. Direct models instead condition on the observed state and an action prefix, as in Fast-LeWM (Gao & Xu, 2026). This removes predicted-state recursion, but leaves the temporal organization of action effects unspecified. A general direct head can learn that organization from data. An alternative is to build it into the predictor: separate the initial condition from subsequent action changes, and let each change initiate a response whose evolution is shared across query times. We ask whether this *action-response factorization* is a useful inductive bias for direct visual planning.

Consider a plan that changes a control twice. The first change starts a response; the second starts another, conditioned on the preceding actions. At any query time, only responses that have already started contribute. PoleWM implements this picture in latent space (Figure 1). Visual history determines the initial state and its free response. A nonlinear prefix network generates coefficients for each action innovation. A shared temporal basis evaluates every response at the time elapsed since its onset. Thus the network learns *what response an action initiates*, while explicit delays and basis functions determine *how that response unfolds*. Responses add, but their coefficients depend on the full causal prefix, allowing nonlinear interactions between actions.

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.

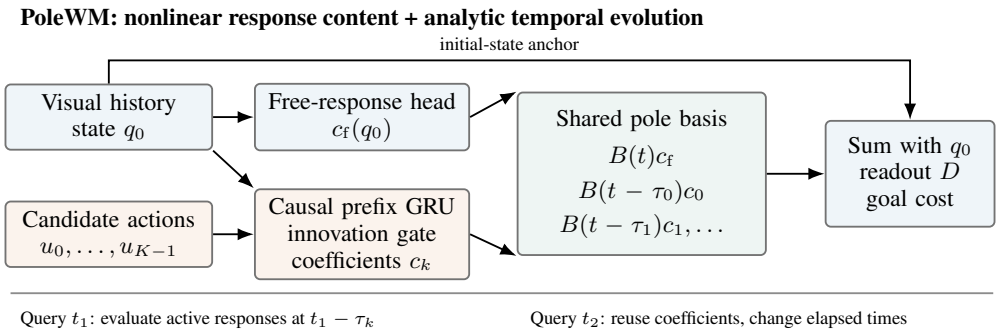


Figure 1: **Separate response content from temporal evolution.** Visual history determines the free response; causal action prefixes and innovations determine event responses. Their common pole basis is evaluated at elapsed times $t - \tau_k$, with zero contribution before onset. Nonlinear prefix conditioning lets later actions interact with earlier ones. Different queries reuse coefficients within the same model call. Replacing the pole basis with a neural time basis retains this response organization.

PoleWM realizes this factorization with a learned bank of stable complex poles and an initial-value anchor. The construction separates persistent displacement from decaying transients, gives each event an exact zero onset value, and supports analytic evaluation without an ODE solver or numerical inverse Laplace transform. It connects classical response models (Åström & Murray, 2008) with nonlinear coefficient–coordinate factorizations (Lu et al., 2021). Our contribution lies in organizing these ingredients into a visual predictor with prefix-conditioned, innovation-gated action events and explicit temporal constraints.

The experiments support this structural view. PoleWM attains the highest mean success in a matched four-environment comparison, with gains on Cube, TwoRoom, and Reacher. In early Cube training, replacing a generic direct head with either a neural response head or a pole response head yields the larger observed improvement; the difference between response bases is smaller. Poles therefore supply a concrete analytic realization of the response hypothesis, while the study also reveals which level of the design has the clearest empirical support. A separate complete-solve benchmark establishes practical planning cost.

Our contributions are:

- **A structured direct predictor.** We factor visual dynamics into a free response and delayed action-innovation responses, combining nonlinear prefix conditioning with shared temporal evaluation.
- **An analytic realization and its structural analysis.** We construct an anchored pole basis, establish non-anticipation and bounded finite-plan responses, and identify the missing basis direction that separates response boundedness from restart closure.
- **Evidence at three levels.** We demonstrate 90.83% mean control success, examine response organization through a no-flow head comparison, and measure 74–87 ms optimized complete-solve latency under a common single-frame protocol.

2 Related Work

Latent dynamics and visual control. Learning a control-oriented latent space predates current joint-embedding world models. E2C learns locally linear dynamics from images (Watter et al., 2015), and PlaNet combines recurrent latent dynamics with online planning (Hafner et al., 2019). PETS develops uncertainty-aware sampling-based model predictive control (Chua et al., 2018); Dreamer learns behavior through latent imagination (Hafner et al., 2020). TD-MPC and TD-MPC2 couple latent trajectory optimization with learned value estimates (Hansen et al., 2022; 2024). These works establish the broader model-based control setting. PoleWM focuses on offline image-goal

prediction with a terminal embedding cost, rather than learning rewards, values, or a policy through online reinforcement learning.

Predictive visual representations and direct world models. DINO learns visual features by self-distillation (Caron et al., 2021), while I-JEPA learns by predicting target representations from image context (Assran et al., 2023). VICReg studies explicit variance and covariance regularization against representation collapse (Bardes et al., 2022). In visual planning, DINO-WM builds dynamics on pretrained features (Zhou et al., 2025), LeWM jointly learns features and dynamics (Maes et al., 2026), and QQWorld uses quantile matching to regularize embeddings (Yu et al., 2026). Fast-LeWM predicts action-prefix outcomes directly (Gao & Xu, 2026). PoleWM adopts learned embeddings and QQ regularization; its contribution is the internal organization of direct prediction into delayed responses. Our generic direct head is an internal comparator, separate from the complete Fast-LeWM system.

Response models and structured state spaces. The decomposition into initial-condition and forced responses is classical in linear systems (Åström & Murray, 2008). S4 brings structured state-space kernels to sequence modeling (Gu et al., 2022), and Mamba introduces input-dependent selection into state-space dynamics (Gu & Dao, 2024). PoleWM uses a shared analytic kernel with a specific event structure: an action innovation gates a prefix-conditioned coefficient, and its contribution is evaluated at elapsed time since onset. Both nonlinear networks built around state-space layers and PoleWM can model nonlinear sequence dependence. PoleWM's distinction is the explicit decomposition of a visual prediction into anchored, delayed action responses, together with direct queries of their sum.

Operator learning and continuous-time prediction. DeepONet separates input-function encoding from output-coordinate evaluation (Lu et al., 2021); Fourier neural operators learn kernels in the Fourier domain (Li et al., 2021). PoleWM shares the coefficient–coordinate viewpoint, with additional initial-state anchoring, causal prefix conditioning, and event-wise time shifts. Neural ODEs compute trajectories by integrating learned vector fields (Chen et al., 2018). Neural Laplace instead predicts transform-domain functions and uses numerical inversion (Holt et al., 2022), with an extension to delayed control (Holt et al., 2023). The Laplace neural operator uses pole-residue structure for differential-equation operators (Cao et al., 2024). PoleWM evaluates its finite time-domain basis analytically and learns its coefficients jointly with visual representations for action-conditioned planning.

Spectral dynamics and composition. DMD with control separates state evolution from actuation in data-driven linear models (Proctor et al., 2016); learned Koopman embeddings seek coordinates with linear evolution (Lusch et al., 2018). PoleWM's poles parameterize latent response shapes, while nonlinear history and action-prefix conditioning remain part of prediction. They are therefore not identified physical eigenmodes. Separately, BRo-JEPA and group-action world models structure latent transformations and their composition (Jha et al., 2026; Wang et al., 2026), and Fast-LeWM uses planning-time consistency (Gao & Xu, 2026). Our training-time restart penalty asks for agreement after reinitialization. The analysis in Section 4 identifies why exact anchoring and decaying pole transients alone do not guarantee that agreement.

3 Action-Response World Models

3.1 Prediction Interface and Response Factorization

Let o_i be an RGB observation and $e_i = E_\theta(o_i) \in \mathbb{R}^D$ its visual embedding. An encoder H_θ summarizes the observed history and executed controls into q_0. A nonlinear readout D_θ maps predicted states back to the embedding space:

$$q_0 = H_\theta(e_{-L+1:0}, U_{-L+1:-1}), \qquad \widehat{e}(t) = D_\theta(\widehat{q}(t)). \tag{1}$$

Here U_i groups the raw controls between adjacent observed frames. The model predicts embeddings rather than pixels. The learned history state is not assumed to be a sufficient physical Markov state.

For a candidate sequence of K normalized raw actions $u_0, \ldots, u_{K-1}$, define absolute event times $\tau_k = k\delta$. A direct head can map the history state, a causal action-prefix representation, and query

time to an outcome. We instead factor its latent trajectory as

$$\boxed{\widehat{q}(t) = q_0 + B(t)c_{\mathrm{f}}(q_0) + \sum_{k:\tau_k \le t} B(t-\tau_k)c_k(q_0, u_{0:k}), \qquad B(0) = 0.} \tag{2}$$

B is a shared bank of scalar time functions; the products contract its channels with vector-valued coefficients. The free response captures evolution from the initial condition, while event k begins at τ_k. The same coefficient set serves all queries for the same history and plan. Although the responses add in state space, this is not a linear control model: each c_k can depend nonlinearly on the entire preceding action sequence, and the visual readout is nonlinear.

Where the inductive bias enters. Equation (2) ties predictions at different times through a common response family. A later action can change its own response coefficients through prefix conditioning, but cannot rewrite contributions before its onset. This separates three roles: the observed history sets the initial condition, the action prefix sets response content, and elapsed time selects a point on each response. A neural time basis can preserve this organization; PoleWM specifies its analytic form below.

3.2 Analytic pole basis and action innovations

PoleWM uses a real anchor and compensated complex exponentials. For $\kappa > 0$, define

$$A(t) = \frac{1 - e^{-\kappa t}}{\kappa}, \qquad C_p(t) = e^{pt} - e^{-\kappa t} - (p + \kappa)te^{-\kappa t}. \tag{3}$$

They satisfy $A(0) = 0$, $A'(0) = 1$, and $C_p(0) = C_p'(0) = 0$. With M conjugate pole pairs, representatives are parameterized by

$$p_m = -\big[d_{\min} + \mathrm{softplus}(\alpha_m)\big] + \mathrm{i}\big[\omega_{\min} + \mathrm{softplus}(\beta_m)\big], \qquad d_{\min}, \omega_{\min} > 0. \tag{4}$$

Negative real parts produce decaying transients. The anchor tends to $1/\kappa$, allowing a persistent displacement in addition to the transient components.

For $v \in \mathbb{R}^D$ and $b_m \in \mathbb{C}^D$, write

$$\phi(t; v, b) = A(t)v + 2\,\mathrm{Re}\sum_{m=1}^{M} C_{p_m}(t)b_m. \tag{5}$$

The free head predicts $(v_{\mathrm{f}}, b_{\mathrm{f}})$ from q_0. An action GRU initialized from q_0 processes (u_k, τ_k) to obtain h_k. Event coefficients are generated with an innovation gate:

$$\eta_k = g_\theta([q_0, h_k]) \odot W_\theta(u_k - u_{k-1}), \qquad (v_k, b_k) = \mathrm{unpack}(\eta_k). \tag{6}$$

Both factors in the product have $(1 + 2M)D$ real entries: D drift entries and $2MD$ real/imaginary residue entries. W_θ is linear and has no bias. Thus an unchanged action starts no new response, while preceding responses persist and the prefix GRU may continue to evolve.

The compensation in C_p gives the coefficient groups distinct roles: v controls the onset slope, while b_m controls transient curvature without changing the initial value or slope. The negative-real-part constraint bounds these transients for a fixed coefficient set. This is the analytic role of poles in PoleWM; the response factorization itself also admits other time bases.

We set $u_{-1} = 0$ and append $u_K = 0$ to define the continuation after the finite plan. Zero refers to normalized action coordinates, not necessarily zero physical actuation. The full predictor is

$$\widehat{q}(t) = q_0 + \phi(t; v_{\mathrm{f}}, b_{\mathrm{f}}) + \sum_{k=0}^{K} \mathbf{1}_{t \ge \tau_k}\phi(t - \tau_k; v_k, b_k). \tag{7}$$

The terminal event contributes zero at the planning endpoint $K\delta$. Every earlier coefficient depends on its own prefix and absolute event time, not on the eventual plan length.

3.3 LEARNING AND PLANNING

Training windows provide observed future embeddings e_j and states q_j obtained by applying H_θ to sliding observed histories. The three prediction/readout terms are

$$\mathcal{L}_{\text{pred}} = \text{MSE}(D_\theta(\widehat{q}(t_j)), e_j), \quad \mathcal{L}_{\text{state}} = \text{MSE}(\widehat{q}(t_j), q_j), \quad \mathcal{L}_{\text{read}} = \text{MSE}(D_\theta(q_j), e_j). \tag{8}$$

Both target and prediction branches receive gradients; there is no EMA encoder. QQ regularization is applied to the observed image embeddings e_j across all frames of the training window, rather than to the predicted states or readout outputs (Yu et al., 2026).

Writing F_θ for Eq. (7), an auxiliary restart term compares a suffix reinitialized at the predicted intermediate state with the direct prediction:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_r \, \text{MSE}_{j>r}\Big(F_\theta(t_j - t_r; \widehat{q}(t_r), u^{(r)}), \text{sg}[\widehat{q}(t_j)]\Big). \tag{9}$$

The restart state remains differentiable. Each restart initializes a new action-prefix GRU and resets the preceding action to normalized zero. The complete-system objective is

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + 0.25\mathcal{L}_{\text{state}} + 0.25\mathcal{L}_{\text{read}} + 3.5\mathcal{L}_{\text{QQ}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow}}, \tag{10}$$

with $\lambda_{\text{flow}} = 0.05$ for the main control model. The early response-head study uses zero flow for every variant. The penalty encourages an approximate agreement; it does not enforce the exact closure discussed below.

For a goal embedding e_g, the cross-entropy method (CEM), used for sampling-based planning in learned dynamics models (Chua et al., 2018; Hafner et al., 2019), minimizes the terminal objective

$$J(u) = \|D_\theta(F_\theta(K\delta; q_0, u)) - e_g\|_2^2. \tag{11}$$

Each solve encodes the observed history and goal, scores candidate plans through Eq. (7), and returns controls for execution. Visual embeddings are cached within a solve. Although the model supports multiple query times, the reported planner uses a terminal goal cost, not path constraints. Dense response aggregation costs $O(QKMD)$ for Q query times after coefficient generation. The action GRU remains sequential in K; direct temporal querying does not remove action-prefix computation. No numerical inverse Laplace transform is required.

4 STRUCTURAL PROPERTIES AND RESTART CLOSURE

The response construction enforces constraints that a general head would otherwise have to learn. We first state the properties of a single conditioned trajectory, then examine the stronger requirement of representing that trajectory after a restart.

Proposition 1 (Anchoring and non-anticipation)**.** *For finite coefficients, Eq. (7) is continuous with $\widehat{q}(0) = q_0$. An action event at or after query time t cannot change $\widehat{q}(t)$. Appending normalized-zero actions preserves the predicted function.*

The zero onset values and causal prefixes establish the proposition without learned cancellation of future-action effects. Here causality refers to information flow. A causal direct head can also satisfy non-anticipation; PoleWM additionally enforces its explicit response decomposition and anchor.

Proposition 2 (Bounded finite-plan response)**.** *For a fixed finite initial state and action plan, finite coefficients, $\kappa > 0$, and $\text{Re}\, p_m < 0$, Eq. (7) is bounded for all $t \geq 0$, and*

$$\lim_{t\to\infty} \widehat{q}(t) = q_0 + \kappa^{-1}\left(v_{\text{f}} + \sum_{k=0}^{K} v_k\right). \tag{12}$$

The statement fixes the coefficients and plan. It is not a uniform bound over all conditioning inputs, a guarantee of accurate extrapolation, or a closed-loop stability theorem. Appendix A supplies both proofs.

Why restart agreement is a separate question. Re-anchoring a trajectory does not simply preserve the original response family. Define $\Psi_\kappa(s) = A(s) - se^{-\kappa s}$. An exact identity is

$$C_p(r+s) - C_p(r) = e^{pr}C_p(s) + C_p'(r)A(s) + (p+\kappa)(e^{-\kappa r} - e^{pr})\Psi_\kappa(s). \tag{13}$$

For distinct nonreal poles, Ψ_κ is generally outside the real span of the original anchor and transients. Hence the basis need not represent every re-anchored tail, even with arbitrary new coefficients. Adding this direction closes the anchored exponential span, but would neither restore missing action-prefix context nor guarantee closure of the nonlinear coefficient map. We do not add it to the evaluated model. This analysis defines the scope of the restart regularizer rather than attributing an observed control error to an unmeasured mechanism.

5 Experiments

We evaluate three levels of the proposal: **control quality** of the complete world model, **early learning** with alternative prediction heads, and **planning cost** of the optimized implementation. The full-model comparison uses two training seeds and actual online history. The head comparison uses one training seed and zero flow for all variants. The timing benchmark standardizes the input to a single current frame and goal image.

5.1 Setup

Tasks and learning protocol. We evaluate image-goal planning on Cube, TwoRoom, PushT, and Reacher. The primary model uses a from-scratch ViT-Tiny/14 encoder at 224×224, a 192-dimensional history state, three history frames, five predicted macro steps, and 25 conjugate pole pairs. Training uses 20 epochs, global batch 128, and AdamW with learning rate 5×10^{-5} and weight decay 10^{-3}. Appendix B specifies the remaining settings.

The shared data protocol uses a 90/10 random split of overlapping observation windows (seed 3072), full-dataset action statistics, and common development control pools. This is a matched control evaluation under the stated protocol; it is not a trajectory-held-out generalization test. Split, preprocessing, and task-pool details are given in Appendix B.

Planning and scoring. All method comparisons use a common training/data protocol and planning budget. CEM uses 300 candidates, 30 iterations, 30 elites, and a horizon of five macro actions, each containing five raw controls. The primary control evaluation executes up to 25 controls before replanning, with success measured within 50 environment steps. The first solve uses the current image; the second uses real observations at steps 15/20/25 and the intervening executed controls. Physical action candidates are converted into each model's normalized coordinates. Main-model evaluation uses the checkpoint's raw BN statistics.

Success follows environment predicates: Cube position error at most 4 cm; Reacher per-joint position error below 0.05 radians; PushT joint agent/block position error below 20 with wrapped block-angle error below $\pi/9$; TwoRoom goal distance below 16 pixels. PushT uses this state predicate rather than IoU. Appendix B gives exact semantics. The primary PoleWM endpoint is fixed epoch 20. Validation-loss checkpoint selection is reported separately, without selecting the best control score.

5.2 Four-environment control

Higher mean success and gains in three environments. Table 1 compares PoleWM with LeWM, Fast-LeWM, its consistency variant, and QQWorld. PoleWM achieves 90.83% macro-average success, improving by 6.08 points over QQWorld and 7.58 points over Fast-LeWM. It leads on Cube, TwoRoom, and Reacher. Relative to the best baseline in each of these environments, the respective gains are 12.00, 5.34, and 10.33 points. The improvement is therefore distributed across three tasks rather than concentrated in a single high-scoring environment.

QQWorld remains strongest on PushT, where PoleWM reaches 82.67% versus 91.67%. The results establish a favorable mean and three-task improvement while retaining a task-specific performance gap; the aggregate scores alone do not determine its mechanism.

Method	Cube	TwoRoom	PushT	Reacher	Mean
LeWM	69.67	83.67	84.00	78.00	78.83
Fast-LeWM	73.33	90.33	87.33	82.00	83.25
Fast-LeWM + consistency	76.67	90.33	86.33	80.67	83.50
QQWorld	71.00	92.33	**91.67**	84.00	84.75
PoleWM	**88.67**$_{\pm 0.47}$	**97.67**$_{\pm 0.94}$	82.67$_{\pm 1.41}$	**94.33**$_{\pm 0.00}$	**90.83**$_{\pm 0.47}$

Table 1: **Goal-reaching success (%).** PoleWM uses fixed epoch 20 and reports mean and sample SD across training seeds 17 and 29, with 300 tasks per environment and checkpoint. Baselines are point estimates from the matched comparison. Bold marks the highest mean. The final column averages environments equally; uncertainty is descriptive rather than a cross-method significance test.

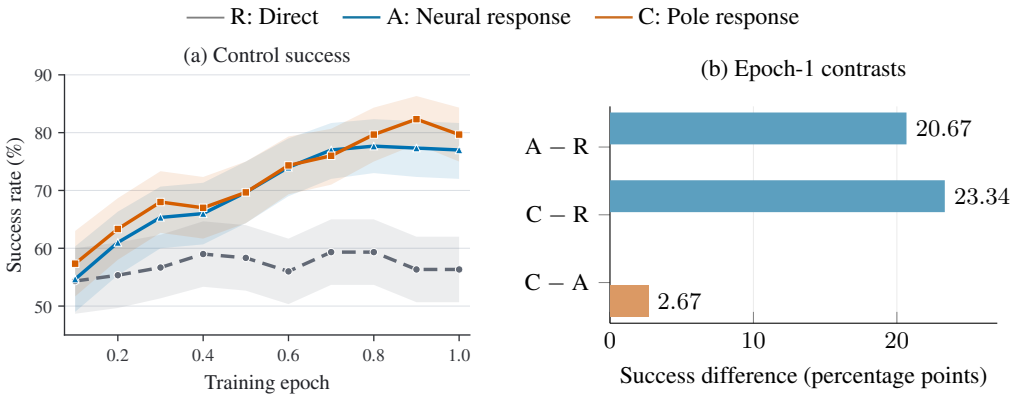


Figure 2: **Response organization is the larger observed early contrast.** Left: Cube success evaluated every 0.1 epoch, with one training seed and no flow. Shading denotes pointwise 95% task-bootstrap intervals from 10,000 resamples, not variation across trained models. Right: differences computed from the displayed epoch-1 percentages; these are descriptive contrasts, not separate experiments or significance tests. The comparison covers the first epoch only.

The mean improvement persists under either checkpoint rule. Selecting PoleWM checkpoints by validation prediction loss gives 91.21% mean success. Only seed 17's Cube and Reacher checkpoints change, from epoch 20 to 19; all other selections remain at epoch 20. The primary table uses fixed epoch 20 throughout, so its improvement does not depend on selecting a checkpoint by control success. Appendix C reports raw counts, both selection rules, and all 20 epochs for each training seed.

5.3 Response structure in early training

We compare three predictor organizations on Cube: R, a generic direct head; A, a neural-time response head; and C, a pole response head. The neural variant retains the free-plus-delayed-response organization of Eq. (2), replacing the analytic pole time basis with a neural one. R maps the conditioned inputs directly to a prediction rather than explicitly summing delayed responses. These are internal head comparisons, not substitutes for the external systems in Table 1.

All three variants use training seed 17, zero flow weight, global batch 128, and the same initial/goal task pool. One epoch comprises 11,390 updates; evaluation occurs every 1,139 updates (0.1 epoch). Each checkpoint contains 300 tasks, with 50 tasks for each of six evaluation seeds. Task identities and images are checked across models and checkpoints. All use actual observations and executed-action history, and the same training windows for BN recalibration. Each model runs its own CEM. This comparison tests alternative prediction heads under a common training and evaluation setting.

Both response heads improve early control. At epoch 1, A exceeds R by 20.67 points and C exceeds R by 23.34 points, whereas C exceeds A by 2.67 points (Table 2). At epoch 0.5, both

Training epoch	R: Direct	A: Neural response	C: Pole response
0.5	58.33	69.67	69.67
1.0	56.33	77.00	79.67

Table 2: **Cube success (%) at two fixed early checkpoints.** All heads use one training seed, the same 300 tasks, and no flow penalty. These checkpoints are separate from the two-seed, raw-BN experiment in Table 1.

Method	Cube	TwoRoom	PushT	Reacher	Task average
LeWM	548	551	549	550	549.5
Fast-LeWM	91	92	91	94	92.0
PoleWM (optimized)	**87**	**74**	**75**	**74**	**77.5**

Table 3: **Single-frame CEM-solve latency (ms).** Each task entry is the median of nine timed solves after four warmups on an exclusive A100-SXM4-80GB. Timing includes encoding, transfers, search, synchronization, and action return; it excludes image preprocessing, environment execution, and rendering. Task average is the mean of four medians, not a pooled median. The benchmark uses a common single-frame protocol and is separate from the actual-history success experiment.

response heads reach 69.67%, versus 58.33% for R. The larger contrast is associated with organizing prediction into responses, and it appears with either a learned neural time basis or an analytic pole basis. This is consistent with the factorization being a useful inductive bias beyond a single basis choice. The pole head attains the highest epoch-1 score while supplying the structural properties in Section 4.

Figure 2 shows the first-epoch learning curves. Its pointwise 95% intervals use 10,000 task-bootstrap resamples and are conditional on the trained seed-17 models. With flow disabled in every variant, the response-head advantage is present without a restart penalty. The study measures early control at a fixed update budget; it does not measure asymptotic success or training-seed robustness. Appendix D specifies the bootstrap and shared BN-calibration procedure.

5.4 Optimized planning cost

We time a complete CEM solve under a common input and search budget (Table 3). Methods run sequentially on one exclusive A100-SXM4-80GB. Each solve receives one current image and one goal image and starts a fresh visual cache; first encoding is included. The common budget is 300 candidates, 30 CEM iterations, 30 elites, and 25 raw controls. CUDA synchronization brackets every timed call. Appendix D gives precision and timing boundaries.

Analytic responses remain practical inside the full search. The optimized PoleWM implementation takes 74–87 ms, versus 91–94 ms for Fast-LeWM and 548–551 ms for LeWM. Its mean task median is 15.8% lower than Fast-LeWM's, with per-task reductions from 4.4% in Cube to 21.3% in Reacher. These measurements include encoding and all 30 search iterations, demonstrating that the response construction can be used at practical latency in the complete planner.

The timing result characterizes the optimized complete system under the common single-frame protocol. It is separate from the actual-history success experiment and does not isolate the cost of poles relative to the untimed R/A heads. We report medians from nine calls, without a tail-latency or cross-method significance claim.

6 Discussion and conclusion

PoleWM introduces an explicit temporal organization for direct visual prediction: nonlinear action prefixes generate response coefficients, and a shared pole basis assembles their delayed effects. This couples expressive conditioning with exact initial-value and information-flow constraints. The restart analysis further distinguishes properties of an individual response from closure under reinitialization, a distinction relevant to consistency regularization in direct models.

The full model improves the four-task mean by 6.08 points over the strongest baseline mean. The early head comparison locates the larger observed contrast at response organization, and complete-solve timing shows that the analytic construction is compatible with practical search. Together, these findings make action-response factorization a useful design choice for visual planning and motivate studying its temporal sharing beyond the terminal costs evaluated here.

Scope and limitations. QQWorld remains stronger on PushT. Two training seeds for full control and one for the early study limit evidence about training variability; the first-epoch comparison does not measure asymptotic performance. No benefit is attributed to the auxiliary flow penalty. The overlapping-window split and reused development pools do not establish trajectory-held-out generalization. Timing uses a separate single-frame protocol, so it is not a joint success–latency measurement. Finally, bounded finite-plan responses do not imply closed-loop stability or path safety.

AI USE STATEMENT

OpenAI ChatGPT/Codex assisted with literature discovery, conceptual and mathematical formulation, proofs, experimental design, result interpretation, manuscript editing, figure code, and deterministic checks. The basis diagnostic uses synthetic mathematical functions; all control results are measured outcomes. The authors are responsible for the claims, proofs, citations, implementations, and submitted artifacts.

REPRODUCIBILITY STATEMENT

Appendix A provides proofs and a deterministic basis diagnostic; Appendix B specifies the model, training, preprocessing, history alignment, and success predicates. Appendices C–D give per-seed counts, checkpoint selection, curves, and head-comparison and timing protocols. Anonymous supporting materials include aggregate data, figure sources, and diagnostic scripts, but not complete training or optimized planning implementations.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2301.08243.

Karl J. Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008. URL https://www.cds.caltech.edu/~murray/books/AM08/.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2105.04906.

Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6:631–640, 2024. doi: 10.1038/s42256-024-00844-4.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021. URL https://arxiv.org/abs/2104.14294.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1806.07366.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1805.12114.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026. URL https://arxiv.org/abs/2606.26217.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2111.00396.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.16828.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pp. 8387–8406, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Neural Laplace: Learning diverse classes of differential equations in the Laplace domain. In *International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/holt22a.html.

Samuel Holt, Alihan Hüyük, Zhaozhi Qian, Hao Sun, and Mihaela van der Schaar. Neural Laplace control for continuous-time delayed systems. In *International Conference on Artificial Intelligence and Statistics*, pp. 1747–1778, 2023. URL https://proceedings.mlr.press/v206/holt23a.html.

Divyansh Jha, Yuanfang Xie, Brennen Yu, and Varan Mehra. BRo-JEPA: Learning modular transformations in latent space. *arXiv preprint arXiv:2606.01372*, 2026. URL https://arxiv.org/abs/2606.01372.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.08895.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9:4950, 2018. doi: 10.1038/s41467-018-07210-0.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026. URL https://arxiv.org/abs/2603.19312.

Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016. doi: 10.1137/15M1013857.

Zijie Wang, Wei Zhang, Weiming Zhang, Fanqi Zhang, Xiao Tan, Yipeng Qin, and Guanbin Li. World models as group actions. *arXiv preprint arXiv:2605.24578*, 2026. URL https://arxiv.org/abs/2605.24578.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://arxiv.org/abs/1506.07365.

Zhoushun Yu, Xiaoyu Hu, and Xiangyu Xu. QQWorld: Quantile-quantile matching for world model regularization. *arXiv preprint arXiv:2607.28415*, 2026. URL https://arxiv.org/abs/2607.28415.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, pp. 79115–79135, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A PROOFS AND MATHEMATICAL DETAILS

A.1 INITIAL VALUES AND ACTION CAUSALITY

From Eq. (3), $A(0) = C_p(0) = C_p'(0) = 0$ and $A'(0) = 1$. At $t = 0$ the free and first event responses vanish; later events are inactive. Hence $\widehat{q}(0) = q_0$. Every event begins with a zero value and has a continuous post-onset response, proving continuity.

For a query t, an event with $\tau_k > t$ contributes nothing. An event with $\tau_k = t$ also contributes zero. Every earlier event coefficient depends only on its own control prefix, and the free response depends only on q_0. Thus future actions cannot change the query value. This proof uses absolute model event times, not time normalized by the total plan length.

When zeros are appended, the original final transition to zero still occurs at the same event time with the same prefix. Every additional event has $u_k - u_{k-1} = 0$, and its bias-free innovation gate is zero. Although its hidden state may change, its entire response coefficient vanishes. Earlier coefficients are independent of the new total length. The complete function is therefore unchanged. At an event time, the added right derivative equals v_k, so derivative changes do not contradict state continuity.

A.2 BOUNDS AND TRANSFORM

For $t \geq 0$, $|A(t)| \leq 1/\kappa$. If $\operatorname{Re} p < 0$,

$$|C_p(t)| \leq e^{\operatorname{Re} p\, t} + e^{-\kappa t} + |p + \kappa| t e^{-\kappa t} \leq 2 + \frac{|p + \kappa|}{e\kappa}. \tag{14}$$

The triangle inequality bounds the finite sum in Eq. (7). Each transient tends to zero, and every finite event is eventually active with anchor limit $1/\kappa$, proving Proposition 2. Coefficients may be arbitrarily large across conditioning inputs. Repeated nonlinear restarts are not covered; for example $F_t(q) = (2 - e^{-t})q$ is bounded in t for fixed q but repeated positive-time restarts can grow.

The analytic response has the transform representation

$$\widetilde{A}(s) = \frac{1}{s(s + \kappa)}, \qquad G_p(s) = \frac{(p + \kappa)^2}{(s - p)(s + \kappa)^2}, \qquad \operatorname{Re} s > 0. \tag{15}$$

In particular, Laplace transformation gives

$$\mathcal{L}\{C_p\}(s) = \frac{1}{s - p} - \frac{1}{s + \kappa} - \frac{p + \kappa}{(s + \kappa)^2}.$$

Combining these fractions gives Eq. (15). The shift theorem supplies $e^{-s\tau_k}$, and the constant initial value contributes q_0/s. The stated half-plane $\operatorname{Re} s > 0$ is sufficient for all terms. These are identities for the learned latent function; the nonlinear readout need not have a finite rational transform.

A.3 RESTART IDENTITY AND SPAN RESTRICTION

Expand both sides of Eq. (13) in the functions $1, e^{-\kappa s}, se^{-\kappa s}, e^{ps}$, using

$$C_p'(r) = pe^{pr} + \kappa e^{-\kappa r} - (p + \kappa)(1 - \kappa r)e^{-\kappa r}.$$

Matching the four coefficients proves the identity. With distinct nonreal p_m, the exponentials $e^{p_m s}$ and their conjugates are linearly independent from the anchor functions. To express Ψ_κ as a combination of $A, \operatorname{Re} C_{p_m}, \operatorname{Im} C_{p_m}$, all complex exponential coefficients must vanish. The remaining anchor multiple cannot supply the $se^{-\kappa s}$ term. Thus this direction is missing from the ordinary real span.

Adjoining Ψ_κ gives the anchored subspace of $\operatorname{span}_{\mathbb{R}}\{1, e^{-\kappa s}, se^{-\kappa s}, \operatorname{Re} e^{p_m s}, \operatorname{Im} e^{p_m s}\}$. That exponential-polynomial space is shift-invariant; subtracting the shifted initial value returns to its anchored subspace. This proves closure of the enlarged response basis, not closure of the neural coefficient map. Also $\Psi_\kappa(0) = \Psi_\kappa'(0) = 0$ but $\Psi_\kappa(\infty) = 1/\kappa$, so adding the term changes the model's asymptotic coefficient formula.

Deterministic basis diagnostic. Set $\kappa = 1$, $p = -0.35 + 2\mathrm{i}$, and restart $r = 0.6$. Fit $2\,\mathrm{Re}[C_p(r + s) - C_p(r)]$ on the even-indexed points of a 401-point grid over $s \in [0, 2]$ and evaluate on odd points. Float64 least squares with cutoff 10^{-12} gives relative ℓ_2 error 5.6210×10^{-2} for the original three-column basis and 3.80×10^{-16} after adding Ψ_κ. The identity residual is below 6×10^{-16}. This tests the function class, not a trained model; the error is not a general lower bound for learned trajectories.

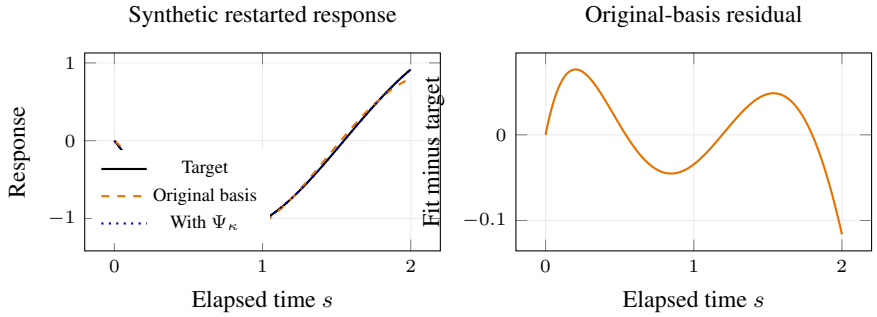


Figure 3: Deterministic basis check from Eq. (13). Fits use even-indexed grid points; the reported errors use odd-indexed points. The extended fit overlaps the target to numerical precision. This figure contains no observations from a trained world model.

A.4 State scale and restart context

QQ regularizes visual embeddings e, rather than states q directly. At the function level, replacing q by aq, $F_t(q, u)$ by $aF_t(q/a, u)$, and $D(q)$ by $D(q/a)$ preserves decoded predictions while scaling state and flow MSE by a^2. The implemented residual networks and weight decay can restrict this rescaling. Raw state-space losses alone therefore do not certify physical accuracy or restart closure. No collapse conclusion is inferred from this coordinate argument.

At restart, the implemented preceding action is normalized zero and the action-prefix GRU is reinitialized from the predicted state. Supplying the actual previous action would change the first innovation, but would not restore the old prefix-GRU hidden state. Exact initial-state anchoring still holds because every response has zero onset value; the right derivative and later response may differ. This is the convention used by Eq. (9), not an assumed exact composition law.

B Implementation and evaluation details

Window counts and compute. A sample contains eight strided observations with seven intervening macro action blocks. The fixed window split, rather than the training seed, controls train/validation membership. Overlapping windows from one episode may appear in both partitions, and the control pools were reused during development. These choices define the reported comparison and do not establish trajectory-held-out generalization. The dataset-window counts and final optimizer updates are: Full-dataset action means and sample standard deviations are retained as part of the evaluated protocol. Images undergo float conversion, ImageNet normalization, and resizing to 224×224. Global batch size is 128. We do not report a matched training GPU-hour comparison.

Clocks and history. The model macro interval is 0.25 for every task, so model raw-event interval is 0.05. Native nominal macro intervals are 0.25, 0.20, and 0.50 seconds for Cube, Reacher, and PushT, giving model/physical time ratios 1, 1.25, and 0.5. TwoRoom's rendering interval is not interpreted as verified physical time. Saved model parameters and clocks remain unchanged during evaluation.

The online-history policy consumes actual transition receipts and resets history between episodes. At step 25 its three frames are steps 15/20/25 and its two executed action blocks span 15:20 and

Component	Setting
Visual encoder	ViT-Tiny/14, 224×224, trained from scratch
Projector	$192 \to 2048 \to 192$, BatchNorm
History state / readout	$D = 192$; GRU history; residual MLP readout
Action prefix / coefficient width	512
Pole pairs / anchor	$M = 25$, $\kappa = 1$
Minimum decay / frequency	$d_{\min} = 0.05$, $\omega_{\min} = 0.1$
History / prediction / raw controls per macro	$3/5/5$
Cold-history augmentation	Probability 0.25
Training seeds / split seed	$17, 29$ / 3072
Optimizer	AdamW, LR 5×10^{-5}, decay 10^{-3}
Global batch / epochs / clipping	$128/20/1.0$
QQ	1024 projections, mean reduction, current batch
Training numerical mode	BF16 visual AMP; FP32 coefficient networks
Analytic accumulation	FP64 / complex128; output FP32
Control BN	Raw checkpoint running statistics, unchanged
CEM	300 candidates, 30 iterations, 30 elites

Table 4: Full-model configuration. Model-time and environment-time distinctions are given below.

Task	Episodes	Train windows	Validation windows	Updates
Cube	10,000	1,458,000	162,000	227,800
Reacher	10,000	1,458,000	162,000	227,800
PushT	18,685	1,447,219	160,802	226,120
TwoRoom	10,000	477,786	53,087	74,640

Table 5: Training counts per seed. Equal epochs do not imply equal updates across datasets.

20:25. The current frame's outgoing action slot is zero. Candidate and history actions are converted through the physical coordinate system when the model scaler differs from the planner scaler; padding slots remain zero.

Native task predicates. All tasks score whether success occurs within 50 environment actions, rather than only at the final state. Cube uses a single cube's Euclidean position error ≤ 0.04 meters; orientation is not scored. Reacher uses an absolute per-joint qpos difference < 0.05 radians without angle wrapping, evaluated by the native task at internal steps. PushT requires the Euclidean error of the first four state coordinates (agent and block xy jointly) < 20, together with wrapped block-angle error $< \pi/9$. TwoRoom uses agent-goal distance < 16 pixels. Reset uses the saved environment-specific state; goal and success data are not input features to the learned cost beyond the goal image.

Selection and statistical unit. Six task/planner seeds contribute 50 episodes each. Shared task pools and repeated epochs do not add independent training replicas. Control-best checkpoint selection is not used. Validation-selected checkpoints minimize prediction loss within the same run; raw embedding losses are not compared across separately learned representations to select a winning model.

Control outcomes are scored from the native Boolean success predicates. The reported tables do not use training-loss aggregates as a surrogate for physical prediction accuracy. Baseline values are local evaluations under the confirmed matched protocol, rather than scores transferred from published tables with different task metrics.

Training mechanics. The learning-rate schedule uses warmup followed by cosine decay; gradient norm is clipped at 1. Visual training uses BF16 AMP, while coefficient networks remain FP32 and analytic accumulation uses FP64/complex128 before returning FP32 states. Training includes cold-history augmentation with probability 0.25. The projector is a $192 \to 2048 \to 192$ MLP with BN; the history encoder and readout use residual corrections. Three-frame history is aligned as $[(e_{-2}, U_{-2}), (e_{-1}, U_{-1}), (e_0, 0)]$, with each nonzero slot carrying the outgoing controls to the next observation.

QQ uses 1024 random projections, mean reduction, and ranking over the current batch, with distributed gathering when applicable and no history queue. It is applied to all eight observed visual positions in each training sample. Both embedding-target and prediction branches receive gradients. The restart loss samples one macro split per training batch; validation averages valid splits. The state supplied to the restarted predictor remains differentiable, and the direct suffix is stop-gradient.

C SEED-LEVEL RESULTS AND STATISTICAL SCOPE

Task	Train seed	Epoch 20 successes	Selected epoch	Selected successes
Cube	17	265/300	19	268/300
Cube	29	267/300	20	267/300
Reacher	17	283/300	19	289/300
Reacher	29	283/300	20	283/300
PushT	17	251/300	20	251/300
PushT	29	245/300	20	245/300
TwoRoom	17	295/300	20	295/300
TwoRoom	29	291/300	20	291/300

Table 6: Raw counts underlying Table 1. Every denominator is 300. Validation selection precedes the use of control results for the table.

For task-specific seed scores S_{17}, S_{29}, we report their arithmetic mean and sample standard deviation, using denominator $n - 1$. This SD is neither a standard error nor a confidence interval. For the macro result, average the four task rates within each training seed and then compute the mean/SD across the two seed-level averages. Calculations use raw counts before rounding.

The full-model evaluation includes $4 \times 2 \times 20 = 160$ checkpoints and $160 \times 6 = 960$ completed seed cells. The table uses only training seeds 17 and 29 under this protocol; no differently trained checkpoint is treated as an additional seed.

Two training seeds provide limited information about training variability. Task resampling cannot replace training replicas, and reused tasks across epochs do not create independent tests. Baseline training-seed uncertainty is unavailable in the reported summaries; no cross-method significance claim is made.

Environment	Fixed epoch 20 (primary)	Validation-selected epoch
Cube	88.67 ± 0.47	89.17 ± 0.24
Reacher	94.33 ± 0.00	95.33 ± 1.41
PushT	82.67 ± 1.41	82.67 ± 1.41
TwoRoom	97.67 ± 0.94	97.67 ± 0.94
Macro mean	90.83 ± 0.47	91.21 ± 1.00

Table 7: PoleWM success (%), mean $\pm$ sample SD across two training seeds. The primary method comparison consistently uses the fixed endpoint. Validation selection is within each training run and is not based on control success.

D RESPONSE-HEAD AND TIMING PROTOCOLS

D.1 EARLY RESPONSE-HEAD STUDY

The first-epoch Cube study compares a generic direct head (R) with neural-time (A) and pole-time (C) response heads. A/C share the free-plus-delayed-response organization. We interpret this as a comparison of predictor designs, rather than attributing the performance difference to the temporal basis alone.

Each head uses training seed 17, global batch 128, and zero training flow weight. One epoch comprises 11,390 updates; evaluations occur at 1,139-update intervals. Evaluation seeds 42, 100, 2026,

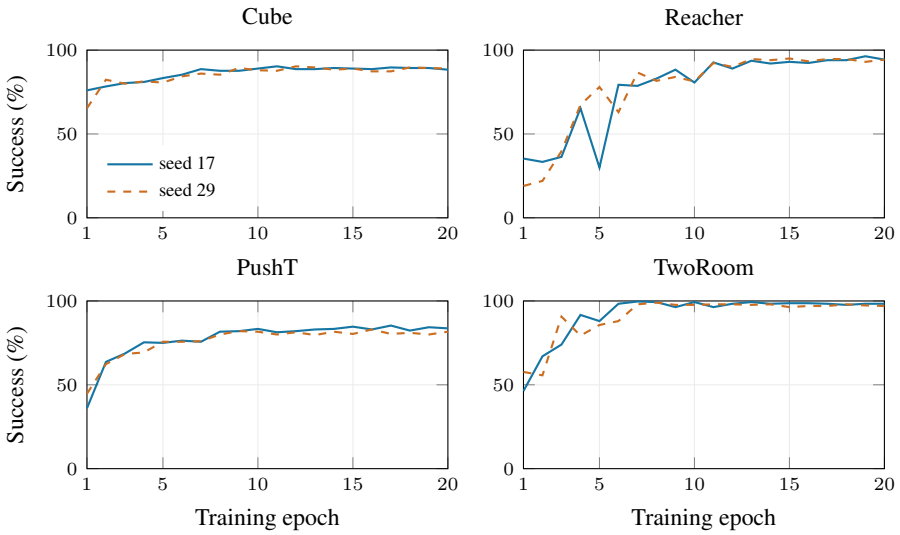


Figure 4: Full-model success at all 20 training epochs, with 300 tasks per point. These are separate from the first-epoch no-flow study. Checkpoint selection does not use the highest control score, and reused pools do not constitute 20 independent tests.

3407, 1234, and 4444 each contribute 50 initial/goal tasks, totaling 300. Task identities and images are checked across heads and checkpoints. These are evaluation seeds, not six independent trained models. Inputs use the original dataset initial/goal images and subsequently actual observations and executed-action history. Every model runs its own CEM; common initial tasks do not imply identical later candidate sequences or trajectories.

Each evaluation recalibrates BN on the same training windows with budget 128×64 and seed 314159. The primary two-seed experiment instead uses raw checkpoint BN, so the two experiments are analyzed separately.

Shading in Figure 2 represents pointwise 95% intervals from 10,000 task-bootstrap resamples. It is conditional on the fitted seed-17 models and excludes training variability. It is neither a simultaneous band over epochs nor a paired confidence interval for A/R or C/A differences. We use the reported endpoints descriptively, without reconstructing per-task outcomes from rounded percentages. The exact numerical snapshots are in Table 2; the full first-epoch success curve preserves the supplied vector plot.

D.2 Complete CEM-solve timing

Methods run sequentially on a single exclusive A100-SXM4-80GB. For each condition, four complete solves warm up the implementation and are excluded from statistics; nine further solves provide the reported single-environment median. Calls to `torch.cuda.synchronize()` occur before timing and after the complete solve. The timer includes transfers, visual encoding, search updates, synchronization, and action return. It excludes image preprocessing, environment execution, rendering, and model loading. These are solve times, not end-to-end robot-cycle times.

All methods receive one current image and one goal image and use a fresh visual cache per solve. First encoding is included; embeddings are reused within that solve. CEM uses 300 candidates, 30 iterations, 30 elites, and five macro actions of five raw controls each. Neural networks use FP32 without AMP. Matrix-multiplication TF32 is disabled and cuDNN TF32 enabled. PoleWM retains FP64 analytic-kernel construction and final summation. We report the optimized implementation as measured, without attributing its timing to a particular compiler transformation or a pole-only complexity reduction.

Table 3 averages the four task medians rather than pooling all 36 calls. No interval is reported.