REWORLDMODEL: RETHINKING VISUAL REPRESENTATION LEARNING IN LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models learn compact visual representations for predicting action-conditioned future states. However, visual information is organized hierarchically across encoder depth: low-level features retain fine-grained spatial and geometric details, favoring representational completeness, whereas high-level features form compact abstractions that favor predictability. Existing world models typically operate only on final-depth representations, potentially discarding details required for precise control. This creates a cross-level trade-off between representational completeness and predictability. To address this, we introduce ReWorldModel (ReWM), a multi-level representation architecture centered on Mixture-of-Depths Attention (MoDA). MoDA enables high-level queries to selectively retrieve low-level features from earlier encoder layers, while a modernized Transformer design improves optimization and spatial modeling. Full ReWM achieves 100% success on four of five tasks and 99.0% mean success on PushT, reaching 95% on PushT with $5\times$ fewer processed training sequences than LeWM under its official recipe. Under a shared training recipe with three paired seeds, a controlled ReWM variant improves Reacher success by 74.4 percentage points at eight encoder updates and reaches 95% with $8\times$ fewer updates on Reacher and TwoRoom. Layer-wise probing further shows that ReWM progressively organizes control-relevant position, orientation, and interaction information across encoder depth, producing final predictive representations with richer control-relevant structure. ReWM thus reshapes the completeness–predictability trade-off toward representations that maintain strong latent predictability while becoming useful for downstream control substantially earlier. These results demonstrate that selective cross-depth access improves the completeness–predictability frontier of latent world-model representations.

1 INTRODUCTION

Latent world models learn compact visual representations for predicting action-conditioned future states (Ha & Schmidhuber, 2018; Hafner et al., 2020; LeCun, 2022). LeWorldModel (LeWM) learns such representations directly from pixels with a latent prediction objective and a regularizer (Maes et al., 2026). Beyond prediction, a representation for control must also retain the distinctions that determine the agent's next action. A slight change in object position, orientation, or interaction geometry can call for a different action in an otherwise unchanged scene. The predictive objective does not explicitly require every such distinction to remain accessible, so prediction quality alone does not reveal when a representation becomes useful for control. On Reacher, two encoders that differ only in architecture reach similar prediction R^2 after 8 encoder updates, yet their frozen representations support 97.7% and 23.3% control success. Such gaps arise because a predictive representation trades off informativeness about its input, i.e., representational completeness, against predictability, and the encoder's architectural inductive bias shapes this trade-off (LeCun, 2022). Rather than biasing the representation with prediction heads for known task variables (LeCun, 2022), we ask whether the encoder architecture itself can shift this trade-off toward representations useful for control.

We analyze this trade-off across encoder depth. In a vision transformer, low-level features retain fine-grained spatial and geometric details, favoring representational completeness, whereas high-level features form compact abstractions that favor predictability (Ghiasi et al., 2022). Many latent world models operate only on final-depth representations (Hafner et al., 2020; Hansen et al., 2022;

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.

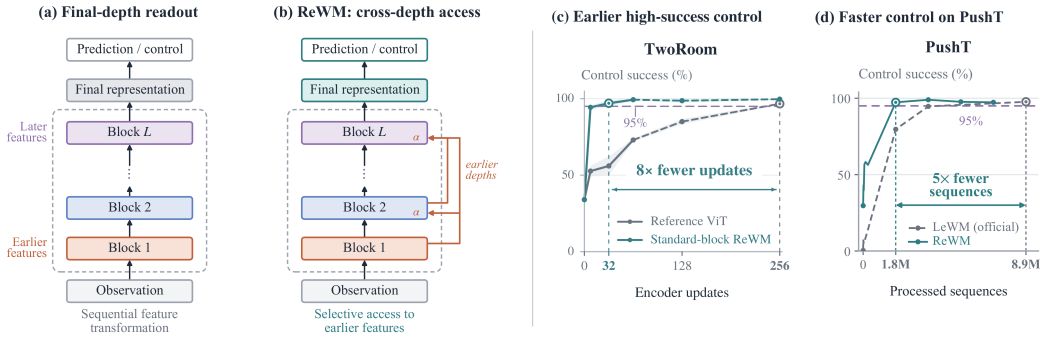


Figure 1: **Completeness, predictability, and control.** (a) A final-depth readout reaches earlier features only through successive transformations. (b) ReWM uses MoDA to selectively retrieve earlier features into its final predictive representation. (c) On TwoRoom, standard-block ReWM first reaches 95% success with $8\times$ fewer encoder updates than the reference ViT (three seeds). (d) On PushT, ReWM first reaches 95% success with $5\times$ fewer processed training sequences than the official LeWM recipe.

Maes et al., 2026), where earlier features persist only in a residual stream that later blocks transform (Fig. 1a). This turns the completeness–predictability trade-off into a cross-level trade-off: progressive abstraction favors predictability but can make the fine-grained details that control depends on increasingly difficult to access. In the official LeWM encoder, the agent-position probe R^2 on TwoRoom falls from 0.813 at block 3 to 0.619 in the final latent (Section 5.2). We therefore seek a high-level representation that retains predictive abstractions while selectively accessing earlier-depth details.

To this end, we propose **ReWorldModel (ReWM)**, a multi-level visual encoder that draws its final predictive representation from multiple encoder depths. ReWM uses **Mixture-of-Depths Attention (MoDA)** (Zhu et al., 2026), in which each query attends, within a single softmax, to spatial features at the current layer and to same-position features from earlier layers. This selective cross-depth access lets high-level queries retrieve low-level features as the representation forms. ReWM further adopts RoPE, RMSNorm, and gated feed-forward layers as a modernized Transformer design. ReWM changes only how the final predictive representation is built. The predictive objective is unchanged, and the predictor and controller still receive a single compact representation.

The Reacher encoders above are the official LeWM encoder (the reference ViT) and its standard-block ReWM variant, which adds only cross-depth access and RoPE. Under an identical recipe and three paired seeds, the variant reaches 95% success with $8\times$ fewer encoder updates than the reference ViT on Reacher and TwoRoom (Fig. 1c and Fig. 6). Compared with the official LeWM recipe, full ReWM reaches 95% success on PushT with $5\times$ fewer processed training sequences (Fig. 1d) and, through cross-depth access (Section 5.5), achieves 100% success on TwoRoom at initialization, versus 27%. Layer-wise probes show that ReWM progressively organizes control-relevant position, orientation, and interaction information across encoder depth. Component ablations identify cross-depth access as the key architectural component behind these gains, which output-level averaging or concatenation of block features does not reproduce. This selective access thus reshapes the completeness–predictability trade-off: ReWM maintains strong latent predictability while becoming useful for control substantially earlier.

Our contributions are threefold:

- We identify a cross-level trade-off in latent world models: control-relevant information retained at earlier depths can become less accessible in the final-depth representation.
- We propose ReWM, a multi-level visual encoder in which high-level queries selectively retrieve low-level features from earlier encoder layers while forming a single compact predictive representation, without changing the predictive objective or downstream interface.
- Across five navigation and manipulation environments, ReWM improves early-control performance by 73 percentage points at encoder initialization on TwoRoom, reaches 95% success on PushT with $5\times$ fewer processed training sequences than LeWM trained using its

official recipe, and achieves 100% success on four environments and 99.0% on PushT, while maintaining strong latent prediction accuracy at matched training budgets.

2 Related Work

Latent world models and predictive representations. Latent dynamics models support planning and policy learning from imagined experience (Hafner et al., 2019; 2020; Hansen et al., 2022; 2024). Joint-embedding architectures predict features without reconstructing pixels: I-JEPA learns from images (Assran et al., 2023), and V-JEPA 2 supports robotic planning through action-conditioned post-training (Assran et al., 2025). LeWM learns from pixels with latent prediction and distribution regularization, evaluating both control and physical-state probes (Maes et al., 2026). We study how these predictive representations are formed across encoder depth, and how access to earlier features affects completeness and predictability.

Control-relevant representations. State abstraction, bisimulation, contrastive learning, and self-prediction seek representations useful for decisions (Gelada et al., 2019; Zhang et al., 2021; Laskin et al., 2020; Schwarzer et al., 2021). Robot-centric pretraining provides features for policy learning and visual rewards (Nair et al., 2023; Ma et al., 2023; Majumdar et al., 2023). GC-IDM learns a goal-conditioned inverse-dynamics controller on pretrained representations (Nguyen et al., 2026), while physically grounded JEPA adds inverse-dynamics and state-alignment objectives during representation training (Liu et al., 2026). These methods bias the representation with task-oriented objectives, as suggested for JEPAs by LeCun (2022); ReWM instead biases it through the encoder architecture, retaining the predictive objective without state-label supervision.

Selective cross-depth access. Residual and dense connections propagate earlier features (He et al., 2016; Huang et al., 2017). DenseFormer learns depth-wise combinations (Pagliardini et al., 2024); DeepCrossAttention makes cross-layer mixing input-dependent (Heddes et al., 2025); Depth-Attention mixes values across layers before sequence attention (Zeng et al., 2026). MoDA jointly attends to current-layer sequence features and earlier-layer depth features under one softmax (Zhu et al., 2026). ReWM builds on this operator in visual world models, studying how cross-depth access changes the final predictive representation and when it becomes useful for control.

3 Completeness, Predictability, and Control

We characterize a predictive representation $z_t = P_\phi(E_\theta(x_t))$ through three measurements at matched encoder checkpoints. **Latent predictability** is the held-out R^2 of one-step prediction in each encoder's own latent space. We compare encoders within the same environment and at matched training budgets. **Representational completeness** is operationalized as the linear accessibility of control-relevant position, orientation, and interaction information. We fit ridge probes with episode-disjoint splits at multiple encoder depths and after the projector. **Early-control success** is closed-loop success after training a GC-IDM controller on a frozen encoder checkpoint. Every matched comparison uses the same controller training and evaluation protocol. Together, these measurements trace the completeness–predictability trade-off over training and relate it to downstream control. Task-specific probe targets, metric definitions, and evaluation settings are given in Section 5.1 and Appendix D.

4 ReWorldModel

ReWorldModel (ReWM) keeps the predictive objective and downstream interface of LeWM, but changes how the visual encoder forms its final representation (Fig. 2). Its key design is selective cross-depth access: while the representation is being formed, high-level queries can directly retrieve features retained at earlier encoder depths.

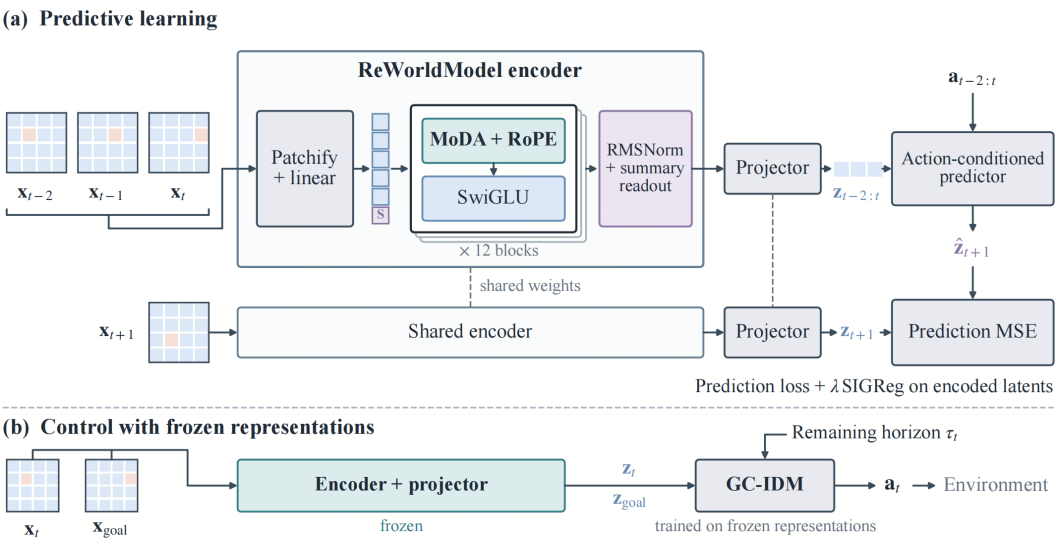


Figure 2: **ReWorldModel.** (a) A shared encoder and projector map frames to latents. An action-conditioned predictor learns to predict future latents, while SIGReg regularizes the encoded latents; only the final context prediction is shown. (b) GC-IDM is trained on frozen encoder and projector outputs for downstream control.

4.1 Multi-level visual encoder

Tokens and blocks. We linearly embed $N = 256$ patches from a 224×224 image with patch size 14 and append a learnable summary token, $H_0 = [e_1, \ldots, e_N, e_{\text{sum}}] \in \mathbb{R}^{(N+1)\times D}$. One-dimensional RoPE encodes the flattened token index. The summary token serves as the global readout over all patches. Each block computes

$$U_\ell = H_{\ell-1} + \text{MoDA}_\ell\big(\text{RMSNorm}(H_{\ell-1})\big), \tag{1}$$

$$H_\ell = U_\ell + \text{SwiGLU}_\ell\big(\text{RMSNorm}(U_\ell)\big), \tag{2}$$

where MoDA_ℓ includes the output projection and reads the cache of earlier-block keys and values defined in Section 4.2, and $\text{SwiGLU}(u) = W_{\text{down}}[\text{SiLU}(W_{\text{gate}}u) \odot W_{\text{up}}u]$. The complete encoder uses RMSNorm and SwiGLU in place of the LayerNorm and GELU MLP of the reference block.

Final representation. Despite its multi-level computation, ReWM exposes only a single final representation, $z_t = P_\phi(\text{RMSNorm}(H_L)_{\text{sum}}) \in \mathbb{R}^{d_z}$. The predictor and controller therefore use the same compact interface as LeWM. We use 12 blocks, width $D = 192$, three attention heads, and $d_z = 192$; full configurations are specified in Appendix A.

4.2 Selective cross-depth access

Standard self-attention operates on the current hidden states and provides no direct retrieval from the intermediate features produced by earlier blocks. Fine-grained information retained at shallow depths can therefore influence the final representation only through the successive transformations of later blocks. Direct retrieval allows such details to remain available to high-level queries without requiring them to survive every intervening transformation. ReWM instead gives each query direct, selective access to features from earlier encoder depths at the same token position. We implement this cross-depth access with MoDA (Zhu et al., 2026), as shown in Fig. 3.

Accessible features. For one attention head, let $u_{\ell,i} = \text{RMSNorm}(H_{\ell-1})_i$ denote the normalized hidden state of token i entering block ℓ, with query, key, and value projections $q_{\ell,i} = W_Q^\ell u_{\ell,i}$, $k_{\ell,i} = W_K^\ell u_{\ell,i}$, and $v_{\ell,i} = W_V^\ell u_{\ell,i}$ in $\mathbb{R}^{d_h}$. The query attends to two candidate sets: current-block spatial features $(k_{\ell,j}, v_{\ell,j})$ for $j \in \mathcal{S}_i$, the spatial candidates of token i, and same-position features cached from earlier blocks $(k_{m,i}, v_{m,i})$ for $m < \ell$. Patch tokens can retrieve features derived from their own earlier representations, which later blocks can aggregate into the terminal summary token,

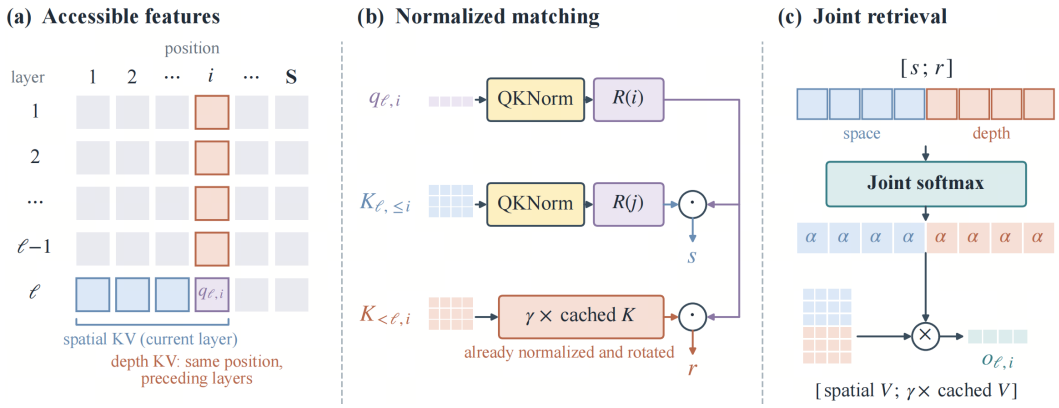


Figure 3: **Cross-depth access in ReWM.** (a) Each query accesses current-layer spatial KV pairs and same-position KV pairs from earlier layers. (b) Queries and keys are normalized and rotated; cached keys are already processed. (c) A joint softmax weights spatial values and cached values. The shared learnable scalar γ scales historical keys and values.

and the summary token analogously accesses earlier summary representations. The cache is local to each image and is empty in the first block.

Joint spatial–depth attention. Following the MoDA implementation, queries and keys are RMS-normalized with learned gains before RoPE:

$$\tilde{q}_{\ell,i} = R(i)\mathrm{RMSNorm}_{q,\ell}(q_{\ell,i}), \qquad \tilde{k}_{\ell,j} = R(j)\mathrm{RMSNorm}_{k,\ell}(k_{\ell,j}). \tag{3}$$

We cache processed keys $\tilde{k}$ and unmodified values v. The scores for spatial and earlier-depth candidates are

$$s_j = \frac{\tilde{q}_{\ell,i}^{\top}\tilde{k}_{\ell,j}}{\sqrt{d_h}} \ (j \in \mathcal{S}_i), \qquad r_m = \frac{\tilde{q}_{\ell,i}^{\top}(\gamma\tilde{k}_{m,i})}{\sqrt{d_h}} \ (m < \ell), \tag{4}$$

where γ is a learned scalar shared across heads and layers that scales the earlier-depth keys and values. The depth path reuses the key and value projections of each block and introduces only the shared scalar γ as an additional learned parameter; cached keys are not rotated again, because their rotations cancel with the query's rotation at the shared position i. Both candidate sets compete in a single softmax:

$$[\alpha^s; \alpha^d] = \mathrm{softmax}([s; r]), \qquad o_{\ell,i} = \sum_{j\in\mathcal{S}_i} \alpha_j^s v_{\ell,j} + \sum_{m<\ell} \alpha_m^d(\gamma v_{m,i}), \tag{5}$$

and head outputs are concatenated and projected back to the block width. Cross-depth access is query dependent: each query jointly selects from its current spatial context and from earlier encoder depths. Retrieved features enter the residual stream and can be further transformed by subsequent blocks, so earlier-depth information participates while the representation is being formed rather than being combined only at the encoder output. With 12 blocks, each query accesses at most 11 additional depth candidates; for the summary token, this is only 4.3% of its 257 spatial candidates (Appendix B).

4.3 Predictive Learning and Frozen-Representation Control

ReWM retains the predictive objective of LeWM without introducing additional representation losses. For $C = 3$ context positions, a causal action-conditioned predictor, including its prediction projection, produces $(\hat{z}_2, \ldots, \hat{z}_{C+1}) = F_\omega(z_{1:C}, A_\eta(a_{1:C}))$, where A_η embeds actions. The loss is

$$\mathcal{L} = \frac{1}{Cd_z}\sum_{t=1}^{C} \|\hat{z}_{t+1} - z_{t+1}\|_2^2 + \lambda\mathcal{L}_{\mathrm{SIGReg}}(z_{1:C+1}), \tag{6}$$

averaged over the batch, where SIGReg regularizes the encoded latents toward an isotropic Gaussian distribution (Maes et al., 2026). The context latents and the prediction targets come from the same

encoder and projector, with gradients through both. Representation training uses only observations and actions; physical-state labels are used only for probing.

At each evaluated checkpoint, we freeze the encoder and projector and train a GC-IDM controller (Nguyen et al., 2026), $a_t = \pi_\psi(z_t, z_g, \tau_t)$, where z_g represents the goal observation and τ_t is the remaining control horizon. Controller training and rollout settings are fixed within each matched comparison (Section 5.1).

5 EXPERIMENTS

We first examine the cross-level trade-off and how ReWM resolves it (Section 5.2), then early control (Section 5.3), the key component (Section 5.4), and where the advantage arises (Section 5.5).

5.1 EXPERIMENTAL SETUP

Tasks and encoders. We evaluate Reacher, TwoRoom, Cube, PushT, and Puzzle-3×3 using the datasets and success criteria of LeWM and GC-IDM (Maes et al., 2026; Nguyen et al., 2026). ReWM denotes the complete encoder of Section 4, and we compare it with the official LeWM encoder trained with its official recipe. We compare the two systems at matched processed training sequences. For a controlled comparison under an identical recipe, we also add cross-depth access to the official LeWM encoder through MoDA with RoPE, without the depth scale γ, and leave everything else unchanged; this is the standard-block ReWM of Section 1. We compare it with the reference ViT, the unmodified official encoder, under the same recipe and paired seeds (Table 3).

Training and evaluation. We evaluate each representation with GC-IDM, a goal-conditioned MLP that maps the current latent, the goal latent, and the remaining horizon directly to an action (Nguyen et al., 2026). It uses no planner or search, so its success reflects how accessible control-relevant state is in the frozen representation. At each encoder checkpoint, we freeze the encoder and projector and train a GC-IDM controller with four hidden layers of width 1024 for 300 epochs, with the same controller seed and settings for every encoder in a comparison. Initialization (zero processed sequences) is evaluated in the same way: the untrained encoder is frozen and only the controller is trained. Each checkpoint is evaluated on 100 rollouts, or 50 on Puzzle-3×3; PushT averages three evaluation banks of 100 start–goal pairs. ReWM reports one run per task, and component ablations share its seed and recipe (Appendix J); the representation measurements of Section 5.2 use three encoder seeds per system. The controlled comparisons use three paired encoder seeds with shared initialization of common parameters and data order.

Representation measurements. We probe the summary token after blocks 3, 6, 9, and 12 and the final latent z using episode-disjoint ridge probes. Following Section 3, the probes measure representational completeness in the task-relevant sense, as the linear accessibility of position, orientation, and interaction variables that control depends on. We pair them with one-step latent prediction R^2 on each encoder's own latent targets.

5.2 REWM RETAINS CONTROL-RELEVANT INFORMATION ACROSS DEPTH

Final-depth readout loses control-relevant information. We first probe the official encoder and ReWM at 32,768 processed training sequences (Fig. 4). In the official encoder, position accessibility is lower in the final latent than at block 3 on all three environments, e.g., 0.619 versus 0.813 on TwoRoom. This is the cross-level trade-off of Section 1.

ReWM progressively organizes this information across depth. In ReWM, position accessibility instead increases from block 3 to the final latent, reaching 0.815, 0.981, and 0.725 on Reacher, TwoRoom, and PushT, compared with 0.436, 0.619, and 0.103 for the official encoder. Its interaction probe also increases from block 3 to the final latent on every environment (0.456 to 0.507 on Reacher, 0.293 to 0.355 on TwoRoom, and 0.106 to 0.130 on PushT), as does its joint-angle probe on Reacher (0.660 to 0.811), whereas the official interaction probe falls on Reacher and TwoRoom. The position gap persists at 262,144 sequences (0.997, 0.988, and 0.899 versus 0.460, 0.817, and

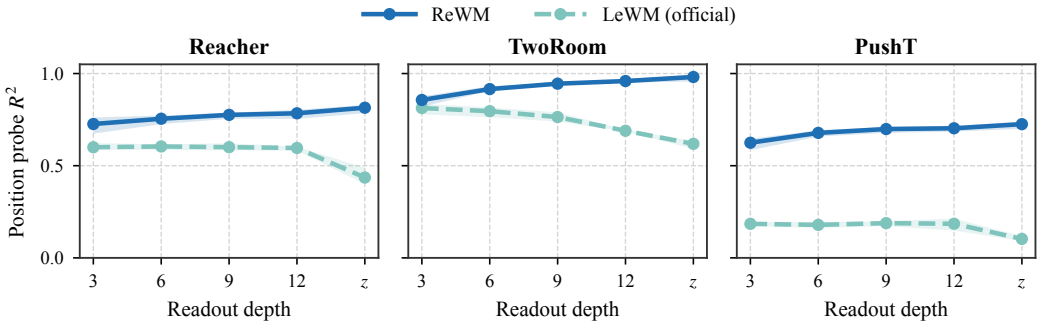


Figure 4: **Control-relevant position information across encoder depth.** Test R^2 of episode-disjoint ridge probes on the summary token after blocks 3, 6, 9, and 12 and on the final latent z, for ReWM and the official LeWM encoder after 32,768 processed training sequences. Curves show the mean over three encoder seeds per system and bands show the seed range.

Table 1: **Accessibility and predictability at update 64.** Reference ViT / standard-block ReWM, mean over three paired seeds; prediction uses training-mode normalization.

Environment	Position R^2	Prediction R^2
Reacher	0.635 / **0.858**	0.520 / **0.770**
TwoRoom	0.471 / **0.978**	0.323 / **0.810**
PushT	0.663 / **0.956**	0.607 / **0.815**

0.280), where ReWM also predicts its own latents better (0.963, 0.980, and 0.943 versus 0.374, 0.727, and 0.588).

Completeness and predictability improve together under a controlled recipe. Under an identical recipe with paired seeds, standard-block ReWM also raises both final-latent position R^2 and prediction R^2 over the reference ViT at update 64 on Reacher, TwoRoom, and PushT (Table 1; Fig. 8 in Appendix I). The controlled insertion therefore reshapes the completeness–predictability trade-off instead of exchanging one property for the other. At update 8 on Reacher, the two encoders have similar prediction R^2 in the recalibrated evaluation mode (-0.131 and -0.096; Table 10) yet reach 97.7% and 23.3% control success, so prediction quality alone does not reveal when a representation becomes useful for control.

5.3 ReWM reaches high-success control substantially earlier

Complete system. ReWM supports high-success control substantially earlier. It reaches peak success of 100% on Reacher, TwoRoom, Cube, and Puzzle-3×3, and 99.0% mean success across three PushT evaluation banks (Fig. 5). Two additional encoder seeds reproduce the early results on Reacher, TwoRoom, PushT, and Puzzle-3×3 (Appendix K). On TwoRoom and Puzzle-3×3, a controller trained on the frozen, untrained ReWM encoder already reaches 100% success, compared with 27% and 74% on the untrained official encoder. This gap exists before any predictive training, showing that the encoder architecture itself strongly biases the accessibility of control-relevant state. On Cube, ReWM reaches 100% after 114,688 processed sequences, whereas the official recipe first exceeds 95% after 1.64M sequences, $14\times$ later. On PushT, where initialization does not solve the task, predictive training raises ReWM from 29.7% to 56.3% after 262,144 processed sequences, compared with 0.3% to 14.0% for the official recipe. ReWM first exceeds 95% on PushT after one epoch (1.78M sequences), and the official recipe first does so at epoch 5 (8.92M sequences), $5\times$ later (Fig. 1d). ReWM then reaches 99.0% on PushT at epoch 2 and stays above 97% at every evaluated epoch through 8 (Table 8). PushT thus shows the two effects together: the architecture supplies a head start at initialization, and predictive training compounds it.

Controlled insertion of cross-depth access. With the recipe unchanged, standard-block ReWM makes the representation usable for control much earlier than the reference ViT (Fig. 6). On Reacher,

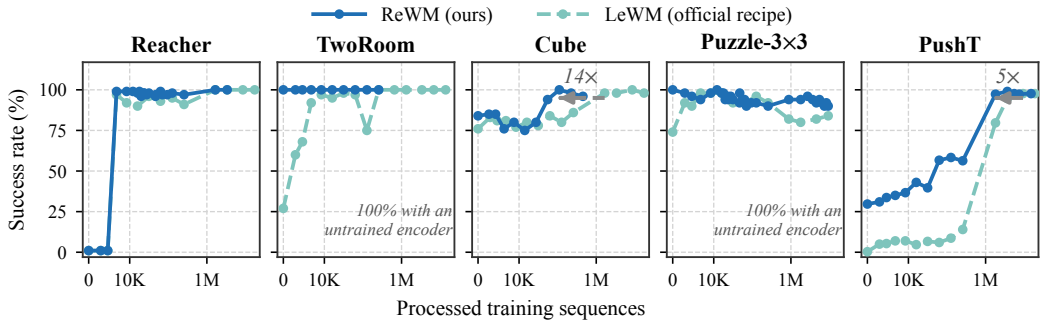


Figure 5: **Full ReWM versus the official LeWM recipe.** Success against processed training sequences (zero is initialization, with the untrained encoder frozen; PushT averages three banks). Arrows give the ratio of the first evaluated budgets reaching 95% success.

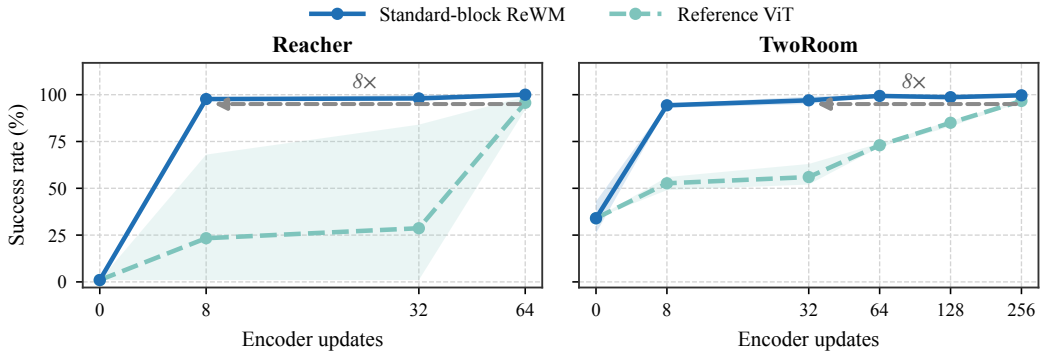


Figure 6: **Convergence to high-success control.** The reference ViT and standard-block ReWM (cross-depth access and RoPE added to the official encoder) under the same fixed recipe, three paired seeds; bands show the seed range. Arrows give the ratio of the first evaluated budgets reaching 95% mean success.

standard-block ReWM reaches 97.7% success after only eight encoder updates, compared with 23.3% for the reference ViT, a gain of 74.4 percentage points. Mean success first reaches 95% at update 8 versus 64 on Reacher, and at update 32 versus 256 on TwoRoom, i.e., with 8× fewer encoder updates on both environments. With a tuned learning rate for both encoders, the advantage also holds on PushT and Cube in every seed pair (Appendix G).

5.4 Cross-depth access is the key architectural component for early control

To find which component produces this advantage, we replace MoDA with vanilla attention, which removes cross-depth access, or replace RoPE, SwiGLU, or RMSNorm (Table 2a).

Removing cross-depth access causes the dominant early degradation. Averaged over the evaluated checkpoints within the first 64 encoder updates, replacing MoDA with vanilla attention lowers success from 79% to 37% on Reacher, from 100% to 75% on TwoRoom, from 97% to 90% on Puzzle-3×3, and from 36.8% to 7.5% on PushT. Replacing RoPE also lowers TwoRoom and PushT (90% and 18.4%), whereas replacing SwiGLU or RMSNorm stays within five points of ReWM on every task. The contrast is sharpest at update 8 (Table 2b): 99% versus 1% on Reacher in each of three encoder seeds, 100% versus 69% on TwoRoom, and 35.0% versus 9.0% on PushT, with the same gap in two additional seeds (Appendix K). Because MoDA is replaced as a whole, these ablations identify the complete cross-depth access operator as the key architectural component behind the early-control advantage.

Simple multi-level readouts do not reproduce the early gains. Two further arms replace cross-depth access with output-level combinations: the projector receives the summary tokens of blocks 3, 6, 9, and 12, averaged or concatenated through a learned linear layer (Table 2b). Averaged over the first 64 encoder updates, both output-level readouts remain substantially below ReWM (55% and

Table 2: **Component ablations and multi-level readouts of the full system.** (a) Mean success (%) over the first 64 encoder updates (checkpoints 0, 8, 16, 32, and 64), and at epoch 2 on PushT; (b) success (%) at update 8; Final layer is the vanilla-attention arm of (a), and the level readouts are defined in Section 5.4. One encoder seed per task; PushT averages three rollout banks. MoDA is replaced by vanilla attention as a whole.

(a) Component replacement	Reacher	TwoRoom	Puzzle	PushT	PushT, epoch 2
Full ReWM	79	100	97	36.8	99.0
MoDA → vanilla attention	37	75	90	7.5	96.7
RoPE → absolute positions	78	90	93	18.4	96.3
SwiGLU → GELU FFN	78	100	94	36.5	97.3
RMSNorm → LayerNorm	78	100	92	35.0	97.7

(b) Success at update 8	Full ReWM	Final layer	Level mean	Level concat.
Reacher	99	1	1	1
TwoRoom	100	69	78	80
PushT	35.0	9.0	7.3	8.0

52% versus 79% on Reacher, and 7.2% and 8.7% versus 36.8% on PushT; Appendix F), so merely exposing multiple depths at the encoder output does not reproduce its early-control advantage. The advantage emerges when earlier features are retrieved during representation formation, allowing subsequent blocks to further transform them.

5.5 The early advantage arises from cross-depth access in the encoder

On TwoRoom at initialization, the ReWM encoder reaches 100% success, whereas the official encoder reaches 27%. Removing cross-depth access from the ReWM encoder, with every other block choice of Table 3 kept, also returns success to 27%, so the advantage arises from cross-depth access rather than from the remaining differences of the full block. The controlled advantage is also retained with controller-unseen episodes and a shallower readout (Appendix E).

6 Limitations

The full system and the official LeWM recipe are compared at matched processed training sequences rather than under an identical training configuration; the controlled paired-seed comparisons isolate the architectural changes from the training recipe. We evaluate representations through a goal-conditioned inverse-dynamics controller without planning; combining ReWM with planning-based control is left for future work. All experiments use a ViT-tiny encoder and five simulated environments; extending ReWM to larger encoders and real-robot data is a natural next step.

7 Conclusion

We study how the architectural inductive bias of a latent world model shapes the completeness–predictability trade-off, and whether it can bias representations toward usefulness for control. We find that the final-depth representation of the official LeWM encoder makes control-relevant information retained at earlier depths less accessible, and propose ReWM, in which high-level queries selectively retrieve low-level features from earlier encoder depths while forming a single compact predictive representation, without task-specific prediction heads or state supervision. ReWM reaches 100% control success on four of five environments and 99.0% on PushT, and reaches 95% success on PushT with $5\times$ fewer processed training sequences than the official LeWM recipe, while organizing control-relevant information across depth and maintaining strong latent predictability. Selective cross-depth access thus offers latent world models a simple, drop-in encoder principle: keep the objective and interface, and let the final representation recover information retained at earlier depths. This design makes predictive representations usable for control substantially earlier, showing that the completeness–predictability trade-off is shaped not only by the objective but also by the architecture of the encoder.

Ethics statement

This work uses simulated environments only and involves no human data.

Reproducibility statement

The architecture, the training objective, and the evaluation protocol are described in Sections 4 and 5.1, and the hyper-parameters and training recipes are listed in Appendices A and J. We will release the code and the configurations.

AI use statement

Large language models were used to polish the writing, to draft sections of the paper from the authors' experimental records, and to assist in scheduling experiments and aggregating result files. All experimental design decisions, claims and numbers were checked by the authors.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629, June 2023.

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

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? a visual exploration, 2022. URL https://arxiv.org/abs/2212.06727.

David Ha and Jürgen Schmidhuber. World models, 2018. URL https://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 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://openreview.net/forum?id=S1lOTC4tDS.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Mike Heddes, Adel Javanmard, Kyriakos Axiotis, Gang Fu, Mohammadhossein Bateni, and Vahab Mirrokni. DeepCrossAttention: Supercharging transformer residual connections. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 22881–22903. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/heddes25a.html.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

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

Muyuan Liu, Yue Huang, Zheng Liang, and Xiang Gao. Toward physically grounded jepa world models for goal-conditioned robotic planning, 2026. URL https://arxiv.org/abs/2609.03565.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=YJ7o2wetJ2.

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

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 655–677. Curran Associates, Inc., 2023. doi: 10.52202/075280-0031. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/022ca1bed6b574b962c48a2856eb207b-Paper-Conference.pdf.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/nair23a.html.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models, 2026. URL https://arxiv.org/abs/2605.08732.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 136479–136508. Curran Associates, Inc., 2024. doi: 10.52202/079017-4336. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/f67449c7ab72f441d3a713b046c6818c-Paper-Conference.pdf.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

Boyi Zeng, Yiqin Hao, Zitong Wang, Shixiang Song, He Li, Feichen Song, Yifan Liu, Ziwei He, Xinbing Wang, and Zhouhan Lin. Depth-attention: Cross-layer value mixing for language models, 2026. URL https://arxiv.org/abs/2606.05014.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Lianghui Zhu, Yuxin Fang, Bencheng Liao, Shijie Wang, Tianheng Cheng, Zilong Huang, Chen Chen, Lai Wei, Yutao Zeng, Ya Wang, Yi Lin, Yu Li, and Xinggang Wang. Mixture-of-depths attention, 2026. URL https://arxiv.org/abs/2603.15619.

A Encoder hyper-parameters

Table 3: **Encoders compared.** All encoders have 12 blocks, width 192, three heads, and patch size 14.

Property	Reference ViT	Standard-block ReWM	ReWM (full)
Position signal	learned absolute	rotary	rotary
Block normalization	LayerNorm	LayerNorm	RMSNorm
Feed-forward	GELU, width 768	GELU, width 768	SwiGLU, width 512
Cross-depth access	none	MoDA	MoDA with scale γ
Encoder parameters	5.50M	5.50M	5.43M

Table 4: **Hyper-parameters of the ReWM encoder.**

Hyper-parameter	Value
Blocks / width / heads / head dimension	12 / 192 / 3 / 64
Input size / patch size / tokens	224×224 / 14 / 256 + 1 summary token
Feed-forward	SwiGLU, hidden width 512, three projections
Linear layers	bias-free
Latent dimension d_z	192
Context length C	3
SIGReg weight λ	0.09

B Two-source readout of cross-depth access

The joint attention of Eq. 5 can be written as a mixture of two readouts:

$$o_{\ell,i} = (1-g)\,\bar{v}_s + g\,\bar{v}_d, \qquad g = \sum_{m<\ell} \alpha_m^d = \frac{Z_d}{Z_s + Z_d}, \tag{7}$$

where $\bar{v}_s$ and $\bar{v}_d$ are the within-source softmax readouts of the spatial and the depth candidates, Z_s and Z_d are the sums of their exponentiated scores, and $\bar{v}_d$ uses the scaled values $\gamma v_{m,i}$. If all scores were equal, g would reduce to the uniform share $n_d/(n_s+n_d)$, where n_s and n_d are the numbers of spatial and depth candidates. Each query has at most $L-1=11$ depth candidates, compared with up to $N+1=257$ spatial candidates, so cross-depth access adds at most 4.3% key–value entries to the attention of the summary token.

C ADDING CROSS-DEPTH ACCESS TO A JEPA ENCODER

Cross-depth access replaces the attention of each encoder block with MoDA; the rest of the encoder, the predictor, the objective, and the controller stay unchanged. The per-block computation for one head is summarized below, following Eqs. 3–5.

```
# H: tokens of block l; cache_K, cache_V: blocks 1..l-1, same positions
u = RMSNorm(H)
q, k, v = W_Q u, W_K u, W_V u
q = RoPE(RMSNorm_q(q));  k = RoPE(RMSNorm_k(k))
s = spatial_scores(q, k) / sqrt(d_h)          # spatial scores
r = (q * gamma * cache_K).sum(-1) / sqrt(d_h) # depth scores
a = softmax(concat(s, r))                     # one joint softmax
o = a_s @ v + (a_d * gamma * cache_V).sum()   # joint readout
cache_K.append(k); cache_V.append(v)          # processed k, raw v
H = H + W_O o;  H = H + SwiGLU(RMSNorm(H))
```

The depth path reuses the key and value projections of each block and adds only the scalar γ, shared across blocks and heads. The cache is rebuilt for each image and is empty in the first block.

D PROBING DETAILS

We report each target separately. The continuous targets are finger position, joint angles (sine and cosine), and the target-minus-finger vector on Reacher; agent position and the target-minus-agent vector on TwoRoom; and pusher position, object position, object orientation (sine and cosine), and the object-minus-pusher vector on PushT. For Reacher and TwoRoom, probe training uses episodes 0–199, validation uses 200–249, and testing uses 800–899. PushT uses a fixed episode-disjoint 70/15/15 split within the extracted episode set. Encoder pretraining is self-supervised and can include these episodes. Feature and target standardization use training statistics; the ridge strength is selected on validation data from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100\}$. For a vector-valued probe target, we average test R^2 over its coordinates. Intermediate readouts after blocks 3/6/9 are raw block outputs, whereas block 12 uses the final normalization and z uses the projector. Fig. 7 shows position probes across depth, and Table 9 reports every target at update 64.

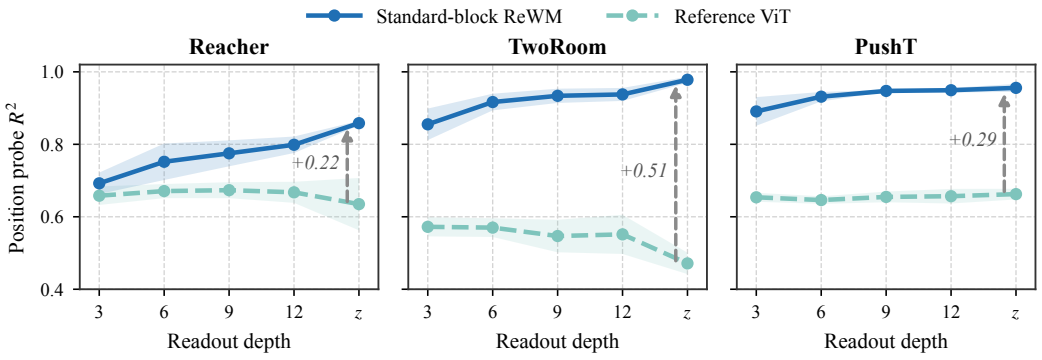


Figure 7: **Control-relevant position information across encoder depth.** Test R^2 of episode-disjoint ridge probes for standard-block ReWM and the reference ViT at update 64. Curves show means and shading shows one standard deviation over three paired seeds. Blocks 3, 6, and 9 use raw block outputs, block 12 uses the final normalized output, and z is the projector output.

E ADDITIONAL CONTROLS

Each row of Table 5 is a separate matched comparison: controller-unseen trajectories evaluates on episodes excluded from controller training, and the shallower readout uses a smaller GC-IDM.

Table 5: **Early-control success (%) under different evaluation conditions.** Reference ViT versus standard-block ReWM, mean over three seeds.

Condition	Environment and budget	Reference ViT	Standard-block ReWM
Standard evaluation	TwoRoom, update 8	52.7	**94.3**
Controller-unseen trajectories	TwoRoom, update 8 / 64	56.3 / 79.7	**93.0 / 98.3**
Shallower readout	TwoRoom, update 8	28.7	**68.7**
	Reacher, update 8	8.0	**63.3**
Tuned learning rate for both encoders	PushT, 0.5 / 3 epochs	37.0 / 89.0	**77.0 / 96.0**
	Cube, update 64 / 256	90.3 / 95.0	**96.3 / 98.3**

F MULTI-LEVEL READOUT BASELINES

The two multi-level arms remove cross-depth access (MoDA) and keep the rest of the full encoder. Each arm reads the summary tokens after blocks 3, 6, 9, and 12 and applies the final RMSNorm to each token. The mean arm averages the four tokens with equal weights and adds no parameters. The concatenation arm maps the four concatenated tokens (768 dimensions) to 192 dimensions with a learned linear layer, initialized to the equal-weight average. This layer adds 147K parameters, giving 5.58M encoder parameters in total. The resulting 192-dimensional readout enters the original projector, so the controller interface is unchanged. Both arms start from the same initialization as the full system and use its data, batch, schedule, controller, and evaluation banks (Reacher seed 0, TwoRoom seed 0, PushT seed 3072). Table 6 reports the early budgets on Reacher and PushT; on TwoRoom, both readouts give 27% at initialization and 78% (mean) and 80% (concatenation) at update 8, compared with 100% for the full system and 69% without cross-depth access.

Table 6: **Control success (%) of multi-level readouts over the first 64 encoder updates.** All arms share one encoder seed per task; PushT reports the three-bank mean.

Reacher			
Update	Full	Mean	Concat.
0	1	1	1
8	99	1	1
16	99	94	91
32	98	88	79
64	98	92	90

PushT			
Update	Full	Mean	Concat.
0	29.7	1.0	1.0
8	35.0	7.3	8.0
16	36.7	10.3	11.0
32	43.0	8.3	11.0
64	39.7	9.0	12.3

G TUNED RECIPE WITH THREE SEEDS

Table 7 compares the reference ViT and standard-block ReWM when both use the tuned learning rate of 3.5×10^{-4}.

H PUSHT UNDER THE THREE-BANK EVALUATION

The same three banks use 100 start–goal pairs each, with goal horizon 25, rollout budget 50, one GC-IDM controller of width 1024 and four hidden layers, 300 controller epochs, and controller seed 42. Temporal aggregation and action clamping are disabled. The Full encoder seed is 3072; bank seeds are 42, 93111, and 93112. The reference is the same official-recipe run used in the main comparison, with encoder initialization and training seed 3072. Full uses encoder learning rate 2.5×10^{-4}, global batch 512, and an eight-epoch training plan. The official single-GPU recipe uses learning rate 5×10^{-5}, global batch 128, and a 100-epoch plan. Both use weight decay 10^{-3} and the same training dataset. One Full encoder epoch is 3,483 updates; one official single-GPU epoch is 13,933 updates.

Table 7: **Early-control success (%) with the tuned learning rate.** Three encoder seeds.

Environment	Encoder	Budget	Per seed	Mean
Cube	Reference ViT	update 64	89, 92, 90	90.3
		update 256	95, 96, 94	95.0
		update 512	95, 94, 98	95.7
	Standard-block ReWM	update 64	96, 96, 97	**96.3**
		update 256	99, 98, 98	**98.3**
		update 512	99, 97, 99	**98.3**
PushT	Reference ViT	0.5 epochs	26, 30, 55	37.0
		1 epoch	79, 75, 81	78.3
		2 epochs	92, 91, 90	91.0
		3 epochs	86, 89, 92	89.0
	Standard-block ReWM	0.5 epochs	77, 78, 76	**77.0**
		1 epoch	87, 90, 90	**89.0**
		2 epochs	96, 93, 94	**94.3**
		3 epochs	96, 96, 96	**96.0**

Table 8: **Full PushT per evaluation bank.** Success (%) in each bank, with actual encoder update counts.

Encoder and budget	Updates	Bank 1	Bank 2	Bank 3	Mean
Full, epoch 1	3,483	100	94	98	97.3
Full, epoch 2	6,966	100	99	98	99.0
Full, epoch 3	10,449	100	96	97	97.7
Full, epoch 4	13,932	97	97	98	97.3
Full, epoch 8	27,864	99	96	98	97.7

I PAIRED REPRESENTATION MEASUREMENTS

Table 1 in the main text reports the controlled comparison at update 64.

Prediction and probes are measured on the same checkpoints and paired encoder seeds. Fig. 8 traces the two metrics over training, and Table 9 reports each state target separately.

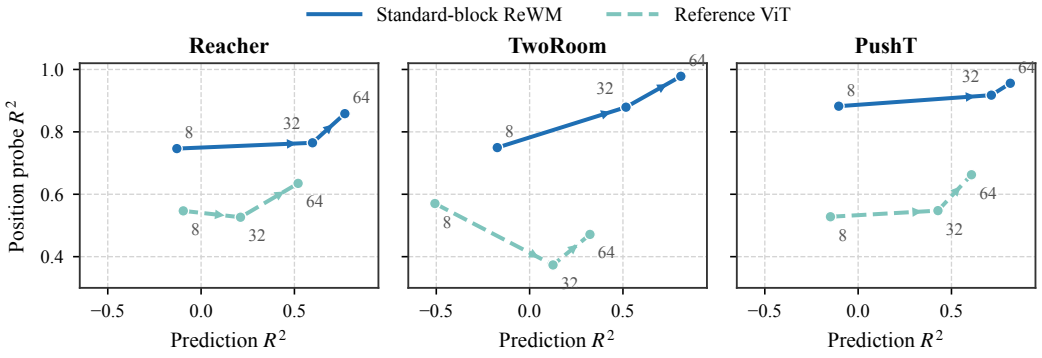


Figure 8: **Training trajectories in the completeness–predictability plane.** Each point is the mean over three paired encoder seeds at updates 8, 32, and 64 for standard-block ReWM and the reference ViT. The horizontal axis is one-step latent prediction R^2 with training-mode normalization; the vertical axis is final-latent position probe R^2.

Table 9: **All final-representation probes at update 64.** Mean ± seed standard deviation for standard-block ReWM and the reference ViT over three paired seeds; test R^2.

Environment	Target	Metric	Reference ViT	Standard-block ReWM
Reacher	Finger position	R^2	0.635 ± 0.072	0.858 ± 0.006
Reacher	Interaction vector	R^2	0.325 ± 0.029	0.447 ± 0.001
Reacher	Joint angles (sin/cos)	R^2	0.682 ± 0.061	0.859 ± 0.011
TwoRoom	Agent position	R^2	0.471 ± 0.030	0.978 ± 0.008
TwoRoom	Interaction vector	R^2	0.194 ± 0.009	0.332 ± 0.005
PushT	Interaction vector	R^2	0.535 ± 0.011	0.605 ± 0.003
PushT	Object orientation (sin/cos)	R^2	0.684 ± 0.022	0.875 ± 0.019
PushT	Object position	R^2	0.663 ± 0.015	0.956 ± 0.011
PushT	Pusher position	R^2	0.637 ± 0.008	0.794 ± 0.015

I.1 Prediction measurement

The train-BN measurement uses fixed consecutive chunks of 112 held-out samples, batch statistics in BatchNorm, frozen running statistics, and disabled dropout. Prediction R^2 is one minus total squared prediction error divided by total squared deviation of the target vectors from their validation-set mean, pooling samples and predicted time positions. Each encoder predicts its own latent targets. We also measure prediction and probes in a single evaluation mode. For each checkpoint, we recalibrate the BatchNorm running statistics on training data, keep all learned parameters fixed, and extract both the prediction targets and the probe features in evaluation mode from this state. Table 10 reports the results; these recalibrated features are not used for the control scores.

Table 10: **Prediction and position probes in one recalibrated evaluation mode.** Reference ViT / standard-block ReWM, mean over three paired encoder seeds. Position targets are finger (Reacher), agent (TwoRoom), and object (PushT) position.

Environment	Update	Prediction R^2	Position probe R^2
Reacher	8	$\mathbf{-0.096}$ / -0.131	0.445 / **0.708**
	32	0.213 / **0.600**	0.381 / **0.757**
	64	0.523 / **0.774**	0.576 / **0.856**
TwoRoom	8	-0.508 / $\mathbf{-0.174}$	0.571 / **0.741**
	32	0.126 / **0.517**	0.377 / **0.874**
	64	0.325 / **0.813**	0.450 / **0.974**
PushT	8	-0.147 / $\mathbf{-0.103}$	0.472 / **0.880**
	32	0.436 / **0.726**	0.534 / **0.922**
	64	0.620 / **0.830**	0.647 / **0.959**

J Full recipes and comparison protocols

Table 11 lists the recipe of the reported run for each task.

Table 11: **Full training recipes.** Batch is the global encoder batch, distinct from the controller minibatch. All encoders have 5,427,841 parameters, SwiGLU width 512, initial depth scale $\gamma = 1$, and SIGReg weight 0.09.

Task	Seed	Learning rate	Weight decay	Batch	Schedule / observed horizon
Reacher	0	2×10^{-4}	10^{-3}	512	Original 2-epoch recipe
TwoRoom	0	7×10^{-4}	10^{-3}	512	12-step warmup; stop at 512
Cube	60106	5×10^{-4}	2×10^{-3}	896	8-step warmup; late-half to 512
Puzzle	0	2×10^{-4}	10^{-3}	512	Original 10-epoch recipe
PushT	3072	2.5×10^{-4}	10^{-3}	512	Original 8-epoch recipe

Cube holds its peak learning rate through update 128 and then follows a cosine decrease to half that rate at update 512. TwoRoom retains its one-epoch planned schedule, with explicit 12-update warmup, and is stopped after update 512. The other tasks retain their original training schedules. All use four sampled frames per example and a frame skip of five. Reported encoder parameter counts exclude the projector, action encoder, predictor, and GC-IDM controller.

The main Full control measurements use all available trajectory frames for feature extraction (stride one), a four-hidden-layer GC-IDM of width 1024 trained for 300 epochs, and controller seed 42. The controller uses its best validation-loss checkpoint within that fixed training budget. Reacher, TwoRoom, and Cube use 100 rollouts; Puzzle uses 50. PushT uses the three 100-rollout banks in Appendix H. The five-arm Full ablation uses Reacher seed 0, TwoRoom seed 0, Puzzle seed 0, and PushT seed 3072; each arm shares its task's training recipe and controller settings. Removing cross-depth access removes MoDA as a whole. The absolute-position replacement adds its learned position-embedding parameters, while the GELU replacement uses hidden width 768 with two projections instead of SwiGLU's width 512 with three projections.

The official PushT reference uses the official single-GPU recipe and the same control evaluation as Full. In the controlled comparisons, both encoders use the same recipe.

K Additional encoder seeds

Table 12 repeats the full system with two further encoder seeds under the same recipes. The early results hold in every seed on Reacher, TwoRoom, PushT, and Puzzle-3×3, and the Reacher and PushT ablations give the same early contrast in every seed.

Table 12: **Full-system control success (%) across encoder seeds.** The first column is the run of the main text (seed 0, or 3072 on PushT); the other two are additional encoder seeds with the same recipe (seeds 1 and 2, or 3073 and 3074 on PushT). PushT reports the three-bank mean.

Task	Budget	Main	Second seed	Third seed
Reacher	update 8	99	99	99
Reacher	epoch 2	100	100	100
Reacher, −cross-depth access	update 8	1	1	1
TwoRoom	every update in 0–512	100	100	100
PushT	update 8	35.0	35.7	35.7
PushT, −cross-depth access	update 8	9.0	6.7	7.0
PushT	epoch 1	97.3	96.3	96.3
PushT	epoch 2	99.0	97.7	97.3
Puzzle-3×3	update 16	98	98	98