How State-Change Prediction Shapes Representations in Latent World Models

Anonymous authors
Internal draft for discussion

Abstract

Latent world models commonly predict the full representation of a future observation. We study how predicting its change from the current representation affects action-related information and its use in control. Within LeWM, we compare next-state and state-change prediction under matched architectures, training data, and representation regularization. State-Change LeWM predicts a latent update and adds it to the current representation. It achieves higher mean planning success on all four control tasks, raising the equally weighted average from 80.90% to 86.15%. Policies trained on frozen State-Change representations also improve success by 5.2 and 3.8 percentage points on Reacher and Cube. Crossed encoder–predictor comparisons and action readouts connect these gains to both the prediction form and the learned representation. The identity path carries the current state forward while the predictor models change; multi-step rollouts further show lower latent error on three tasks and higher decoded PSNR on all four.

1 Introduction

World models support planning and decision-making by predicting the consequences of actions (Ha & Schmidhuber, 2018; Hafner et al., 2019). In visual environments, pixel-space prediction must capture both environmental dynamics and substantial observational detail. Latent world models predict in a compact representation space, allowing candidate actions to be evaluated directly from predicted representations. LeWorldModel (LeWM) provides an end-to-end instance of this approach (Maes et al., 2026).

Future-state representations contain both persistent content and changes caused by a transition. In a jointly trained latent world model, the prediction loss also shapes what the encoder retains (Assran et al., 2023; Van Assel et al., 2025). Predicting a change rather than the full future representation may therefore affect action-related information in the learned representation. We ask whether this target improves the representation's decision utility and how it changes multi-step dynamics prediction.

We compare two prediction forms in LeWM. Next-State LeWM predicts future representations from latent histories and action sequences. State-Change LeWM predicts the difference between sampled representations and adds it to the current representation. The variants have the same encoder architecture, prediction backbone, training data, and representation regularization, and are trained separately end-to-end. Figure 1 shows their computational paths.

State differencing cancels additive latent components that persist over the prediction interval. The identity path carries the current representation forward, leaving the predictor to model its update. Because the encoder is jointly trained with the predictor, this change in parameterization can affect both the learned dynamics and the resulting representation.

We assess the complete model through planning and the learned representation through policies trained on frozen encoders. Crossed encoder–predictor combinations and action readouts examine the sources and content of the planning gains; multi-step rollouts assess prediction behavior.

State-Change LeWM achieves higher mean planning success on all four control tasks, increasing the equally weighted average from 80.90% to 86.15%. With the pretrained predictor removed, policies trained on frozen State-Change representations improve success by 5.2 and 3.8 percentage points on Reacher and Cube. Crossed combinations associate the planning gains with both prediction form

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.

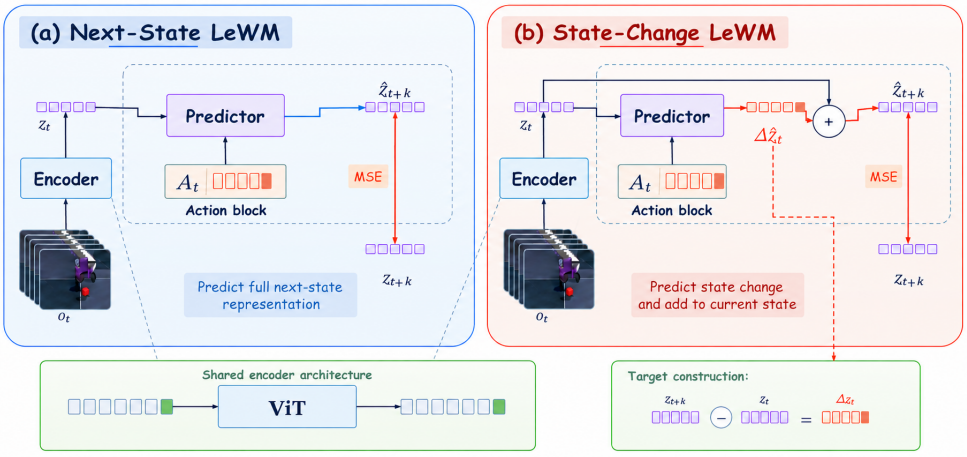


Figure 1: Prediction targets in Next-State LeWM and State-Change LeWM. The upper panels show direct future-state prediction and state-change prediction with an identity path. Next-State predicts z_{t+k}; State-Change predicts $\widehat{\Delta z}_t$ and reconstructs $z_t + \widehat{\Delta z}_t$, with target $\Delta z_t = z_{t+k} - z_t$. The lower panels show the shared encoder architecture and target construction.

and encoder source. State-Change also reduces ten-step latent error on three tasks and improves decoded-rollout PSNR on all four. Prediction target choice thus affects both representation utility and the behavior of the learned dynamics.

2 Related Work

Latent world models for visual control. DINO-WM learns dynamics in a pretrained visual feature space (Zhou et al., 2025), while LeWM trains the encoder and predictor jointly with latent regularization (Maes et al., 2026). Sub-JEPA further studies how that regularization affects end-to-end learning (Zhao et al., 2026). We use LeWM as a controlled setting in which the architecture, data, and regularizer remain fixed while the prediction form changes.

Latent changes and action information. Delta-JEPA retains next-state forward prediction and trains an auxiliary decoder to recover actions from observed latent differences (Zhang et al., 2026). Action-Discriminative World Models (AD-WM) combines residual latent prediction with action recovery from predicted transitions; its Cube ablation also evaluates residual prediction alone (Qiu et al., 2026). Our study examines this prediction form across four control tasks and traces its effects on the jointly learned representation and predictor through frozen-encoder policies, crossed encoder–predictor comparisons, action readouts, and multi-step rollouts.

3 Prediction Target Parameterization

3.1 Setting and Controlled Comparison

We learn latent dynamics from reward-free offline trajectories of visual observations o_t and actions a_t, where t indexes the original recorded time steps. The visual encoder and projector e_θ, with trainable parameters θ, produce the representation $z_t = e_\theta(o_t) \in \mathbb{R}^d$, where d is the latent dimension. Observations used for prediction are sampled k recorded steps apart, with intervening action block $A_t = (a_t, \ldots, a_{t+k-1})$. The predictor context c_t contains the available latent history, including z_t, and the aligned action blocks.

Next-State LeWM and State-Change LeWM use matching visual and action encoder architectures, prediction backbones, training data, and representation regularization, and are trained separately. They differ in the prediction target and the identity path used by State-Change LeWM. Architecture and training settings are detailed in Appendix A.

3.2 PREDICTION TARGETS AND TRAINING

Let f_ϕ and g_ψ denote the Next-State and State-Change prediction mappings, with trainable parameters ϕ and ψ. The state-change target is $\Delta z_t = z_{t+k} - z_t$. The two models predict

$$\hat{z}^{\mathrm{N}}_{t+k} = f_\phi(c_t), \tag{1}$$

$$\widehat{\Delta z}_t = g_\psi(c_t), \qquad \hat{z}^{\mathrm{S}}_{t+k} = z_t + \widehat{\Delta z}_t, \tag{2}$$

where $\widehat{\Delta z}_t$ is the predicted change, and $\hat{z}^{\mathrm{N}}_{t+k}$ and $\hat{z}^{\mathrm{S}}_{t+k}$ are the predicted future representations for Next-State and State-Change LeWM, respectively.

Both models minimize mean-squared prediction error with the same SIGReg anti-collapse regularizer. Writing $\mathcal{R}(Z)$ for SIGReg applied to a batch of projected representations Z and λ for its weight, the objectives are

$$\mathcal{L}_{\mathrm{N}} = \mathbb{E}\left[\frac{1}{d}\|f_\phi(c_t) - z_{t+k}\|_2^2\right] + \lambda\mathcal{R}(Z), \tag{3}$$

$$\mathcal{L}_{\mathrm{S}} = \mathbb{E}\left[\frac{1}{d}\|g_\psi(c_t) - \Delta z_t\|_2^2\right] + \lambda\mathcal{R}(Z). \tag{4}$$

Here, $\mathbb{E}$ averages over sampled sequences and valid prediction positions, and $\|\cdot\|_2^2$ denotes squared Euclidean error. Within each model, the same encoder processes context and target observations. Gradients propagate through both z_t and z_{t+k}, including both terms in Δz_t.

Relation between the targets. For the same State-Change computation graph, state-change regression equals future-state regression with an identity path:

$$\|g_\psi(c_t) - \Delta z_t\|_2^2 = \|z_t + g_\psi(c_t) - z_{t+k}\|_2^2. \tag{5}$$

This identity frames the comparison as a choice of prediction parameterization: direct future-state prediction versus a learned update with an identity path. We examine its effect on finite-model prediction and the jointly trained encoder. Appendix A.1 gives the fixed-encoder relation.

4 ACTION-RELATED REPRESENTATION LEARNING

Planning tests the complete world model. We then remove the pretrained predictor and train policies on frozen encoder outputs, compare crossed encoder–predictor combinations, and read out actions and kinematics from real transitions. These evaluations test whether the planning gains extend to the learned representation.

4.1 DECISION UTILITY IN PLANNING

We first compare latent world models on goal-conditioned planning. The planner rolls out candidate action sequences in each model's latent space and selects actions according to the predicted future representations. Table 1 summarizes planning success for Next-State LeWM, State-Change LeWM, Planning with Latent Dynamics Models (PLDM), DINO-WM, and Sub-JEPA across four control environments (Sobal et al., 2025; Zhou et al., 2025; Zhao et al., 2026). All five models use the shared task-specific planning protocol detailed in Appendix A.

State-Change LeWM outperforms its matched Next-State counterpart in every environment and achieves the highest equally weighted mean planning success among all evaluated models, with leading results on PushT and Cube. These results place it among the strongest models in the comparison and show that state-change prediction strengthens planning across navigation, pushing, reaching, and manipulation within the same LeWM architecture.

Figure 2 aligns the real reference trajectory with paired Cube rollouts from the same initial state and goal. The reference reaches the goal at $t = 25$ and holds its terminal observation thereafter. State-Change LeWM maintains control of the cube and reaches the same configuration, whereas Next-State LeWM loses contact with the cube.

Table 1: Planning success (%) under the shared task-specific cross-entropy method (CEM) evaluation protocol. All five models use the same planning settings and ten evaluation seeds. Values are mean ± standard deviation; aggregation conventions are detailed in Appendix A. Green/red mark column maxima/minima; bold marks maxima.

World model	TwoRoom	PushT	Reacher	Cube
Next-State LeWM	87.20 ± 2.30	84.20 ± 4.70	78.40 ± 5.10	73.80 ± 8.30
State-Change LeWM	91.40 ± 2.99	$\mathbf{91.80} \pm 3.58$	82.00 ± 7.77	$\mathbf{79.40} \pm 7.66$
PLDM	96.40 ± 2.50	74.40 ± 5.20	$\mathbf{84.40} \pm 6.05$	61.40 ± 9.21
DINO-WM	$\mathbf{99.60} \pm 0.84$	63.80 ± 9.02	78.00 ± 5.33	75.60 ± 6.10
Sub-JEPA	92.60 ± 3.27	91.60 ± 2.63	56.80 ± 7.90	78.60 ± 7.49

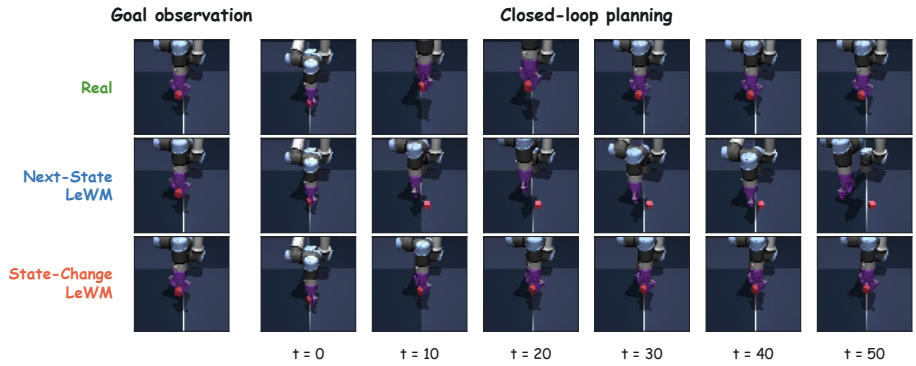


Figure 2: Paired Cube trajectories under matched CEM planning (seed 42, paired case 29). The left column shows the shared goal observation; the remaining columns align the real reference and model-controlled environment observations at $t = 0, 10, \ldots, 50$. The real trajectory terminates at $t = 25$, and its terminal observation is held in the later columns. Next-State LeWM releases the cube early, while State-Change LeWM maintains the grasp and reaches the goal configuration.

Encoder–predictor cross-comparisons. We compare native and crossed combinations to examine how planning varies with encoder source and prediction form. The first term in each row names the encoder training form and the second names the prediction form. Native pairs are trained jointly; crossed pairs refit a predictor on a frozen encoder.

Table 2: Planning success (%) for native and crossed LeWM encoder–predictor combinations. The first label identifies the encoder pretraining target and the second the prediction form. Native pairs are trained jointly; crossed predictors are refit on frozen encoders.

LeWM encoder / predictor	TwoRoom	PushT	Reacher	Cube
Next-State / Next-State	87.20 ± 2.30	84.20 ± 4.70	78.40 ± 5.10	73.80 ± 8.30
Next-State / State-Change	87.2 ± 3.7	92.2 ± 4.9	80.4 ± 6.9	75.0 ± 8.9
State-Change / Next-State	89.6 ± 2.1	85.8 ± 4.9	82.6 ± 5.7	75.6 ± 8.0
State-Change / State-Change	91.40 ± 2.99	91.80 ± 3.58	82.00 ± 7.77	79.40 ± 7.66

Panel 3(a) compares prediction forms with the encoder source fixed. State-change prediction improves planning success in most comparisons and raises the four-task average under both encoder sources, with its clearest gains on PushT. This advantage across representations learned with different targets supports the effectiveness of state-change prediction in using a given representation for action selection.

Panel 3(b) compares encoder sources under each prediction form. The State-Change encoder improves success in most comparisons and achieves a higher four-task average with both next-state and state-change prediction. Its advantage across prediction forms indicates that state-change training produces representations that support more effective planning. The two panels thus provide complementary evidence for contributions from both the prediction form and the learned representation.

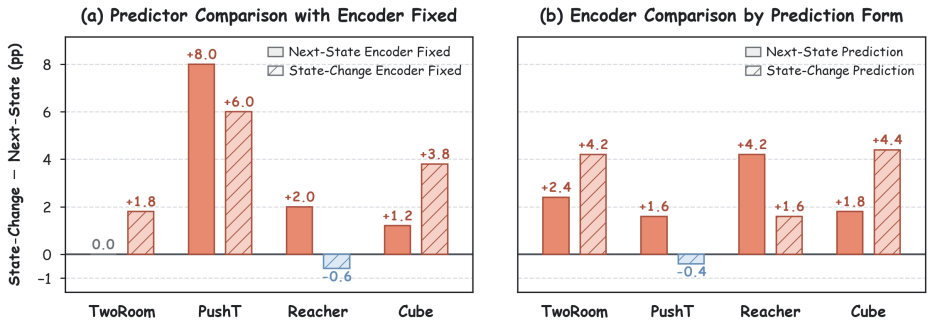


Figure 3: Controlled comparisons from Table 2. Panel (a) fixes the encoder source and reports State-Change-predictor minus Next-State-predictor success; solid/hatched bars use encoders trained with Next-State/State-Change targets. Panel (b) fixes the prediction form and reports State-Change-encoder minus Next-State-encoder success; solid/hatched bars use next-state/state-change predictors. Crossed predictors are refit on frozen encoders. Values are percentage points; coral/blue denote positive/negative differences.

4.2 Frozen-Encoder Policy Utility

We remove the pretrained predictor, freeze the encoder, and train a goal-conditioned behavioral cloning (GCBC) policy on its outputs. This tests the representation through a separate decision-making interface.

Table 3: Frozen-encoder GCBC success (%). A new policy is trained on each frozen encoder; DINOv2-S is an external representation baseline (Oquab et al., 2024). Values are mean ± standard deviation over ten evaluation seeds. Green/red mark column maxima/minima; bold marks maxima. Equal columns are unshaded.

Frozen encoder	TwoRoom	PushT	Reacher	Cube
Next-State LeWM	**100.00** ± 0.00	87.00 ± 4.83	89.80 ± 2.74	79.40 ± 4.22
State-Change LeWM	**100.00** ± 0.00	**87.40** ± 3.78	**95.00** ± 3.16	83.20 ± 7.13
DINOv2-S	**100.00** ± 0.00	79.40 ± 5.66	1.20 ± 1.40	**90.40** ± 5.87

After removing the pretrained predictor, State-Change representations retain their utility for policy learning. They match the high success of Next-State on TwoRoom and slightly exceed it on PushT, with larger mean gains on Reacher and Cube. These results suggest that state-change training makes action-relevant information more readily usable by GCBC, allowing the learned encoder to support effective decisions through a separately trained policy.

Table 4: Frozen-encoder OFT-style success on LIBERO (%) (Liu et al., 2023). The same action head is trained on each frozen, suite-specific encoder. Each entry reports one encoder–suite run. Green/red mark column maxima/minima; bold marks maxima.

Frozen encoder	Goal	Spatial	Object
Next-State LeWM	32.00	23.00	6.20
State-Change LeWM	**33.60**	**25.20**	**11.20**

State-Change also improves policy success across the LIBERO Goal, Spatial, and Object suites. The consistent gains with a second policy interface extend the evidence for its downstream utility to a broader set of manipulation tasks, supporting the value of state-change training for learning representations that action heads can use effectively.

4.3 Decodability of Action and Kinematic Information

We examine action-related information by decoding executed action blocks and kinematic variables from frozen representations of real transitions. Linear and MLP probes use the latent change or concatenated before-and-after representations. Performance is measured by Pearson correlation r; higher values indicate more accessible target information.

We first recover the executed action block from the latent state change

$$\Delta z_t = e_\theta(o_{t+k}) - e_\theta(o_t). \tag{6}$$

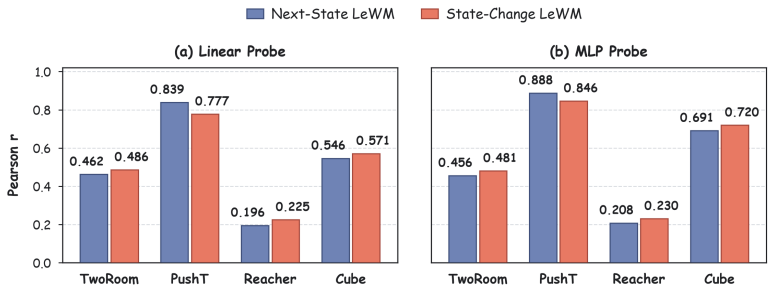


Figure 4: Action-block readout from real transitions. Linear and MLP probes predict the executed five-step action block from latent differences produced by frozen LeWM encoders. Bars report Pearson r; full metrics are in Appendix B.2.

Figure 4 shows higher action-readout correlation for State-Change LeWM on TwoRoom, Reacher, and Cube under both linear and MLP probes. Its equally weighted mean Pearson correlation across the four tasks is also higher for both probe families. The agreement between linear and nonlinear readouts supports improved accessibility of executed actions from the learned latent changes across most evaluated tasks.

We then concatenate the before-and-after representations, $[z_t, z_{t+k}]$, and decode joint velocity, gripper velocity, and the action block on Cube. State-Change leads on all three targets under both probe families, with a pronounced advantage in MLP action readout over both Next-State LeWM and DINOv2-S (Oquab et al., 2024). The improvement extends from executed actions to joint and gripper motion, supporting the view that state-change training makes action-related dynamics more accessible in the transition representation.

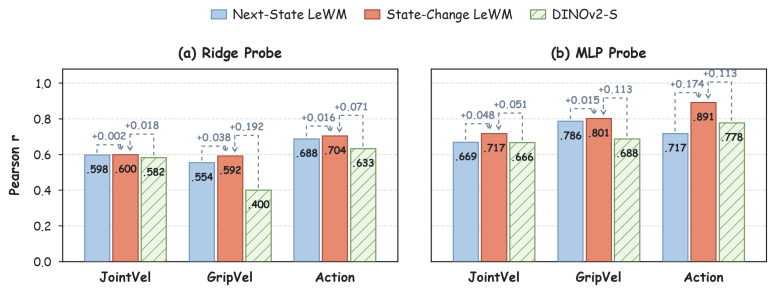


Figure 5: Cube transition readout from $[z_t, z_{t+k}]$. Ridge and MLP probes predict JointVel, GripVel, and Action from DINOv2-S, Next-State LeWM, and State-Change LeWM representations. Bars report Pearson r; dashed brackets show the indicated State-Change differences.

The policy and readout experiments provide complementary evidence for action-related representation learning. Policies trained on frozen State-Change encoders achieve stronger downstream performance, while probes recover actions and associated motion more accurately across most evaluated settings. These findings support the view that state-change prediction shapes representations whose transition information is both accessible to readout models and useful for action selection.

5 Prediction Behavior Under State-Change Parameterization

State-change prediction separates state propagation from dynamic updates: the identity path preserves the current representation, while the predictor learns subsequent changes. This division focuses the learned output on state evolution and provides a state reference for multi-step prediction. With architecture and parameter count held fixed, latent prediction error, decoded image quality, and planning under appearance shifts provide complementary evidence of the effects of this parameterization.

5.1 MULTI-STEP LATENT PREDICTION

State-change prediction achieves lower normalized cumulative latent error on multiple tasks, demonstrating improved multi-step prediction. Figure 6 compares ten-step open-loop rollouts under the same recorded actions. At the tenth prediction step, the cumulative normalized error, averaged equally across the four tasks, is approximately 10.1% lower for State-Change than for Next-State.

This result shows that the effect of state-change parameterization extends to recurrent dynamics prediction. The identity path directly propagates the current state, allowing the predictor to construct subsequent predictions around state updates rather than reproduce the full future state through its learned output. The reduction in average cumulative error is consistent with the preceding predictor comparisons under fixed encoders, supporting the benefit of separating state persistence from change modeling for multi-step prediction.

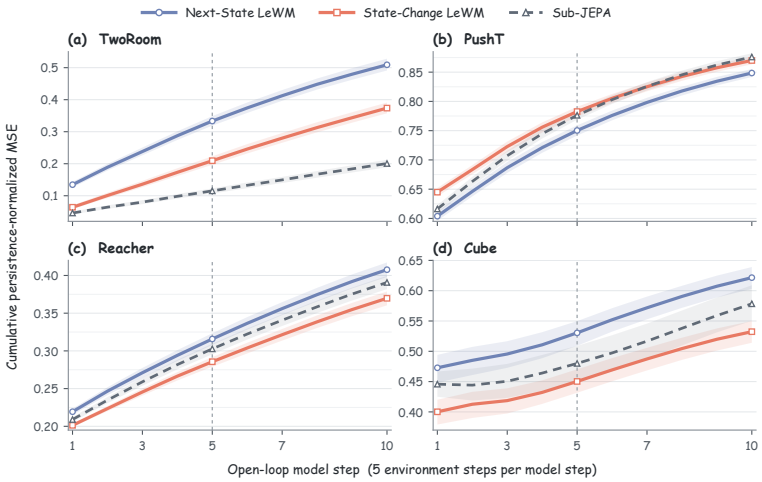


Figure 6: Open-loop prediction error accumulation under recorded action sequences. Curves show cumulative MSE normalized by each model's persistence baseline; lower is better. Each model step corresponds to five environment steps, and the dashed line at $K = 5$ marks the planning horizon. Shaded bands show episode-bootstrap 95% confidence intervals.

5.2 DECODED ROLLOUTS

The advantage of state-change prediction is also reflected in the image reconstruction quality supported by its predicted representations. We map predicted latents back to image space through a separately trained CNN decoder for each model. Figure 7 shows that State-Change achieves higher PSNR on all four tasks, indicating that its decoded images are closer to the real observations at the pixel level.

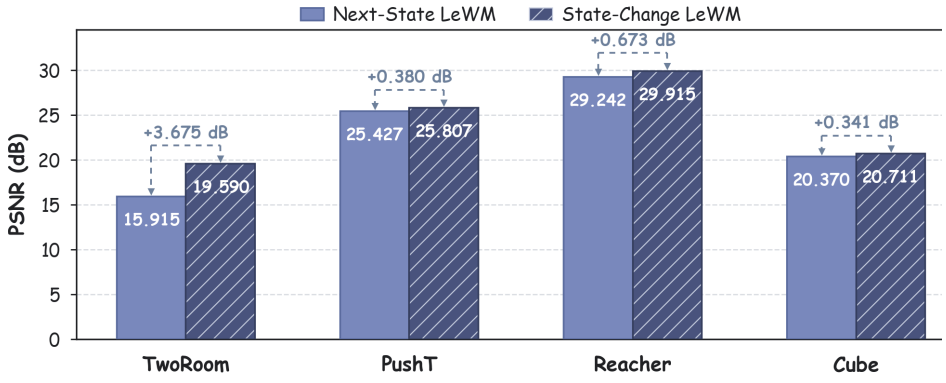


Figure 7: Decoded open-loop rollout PSNR (dB; higher is better). Each model uses a separately trained CNN decoder; annotations show the PSNR gain of State-Change over Next-State.

These results provide image-space evidence of improved latent prediction: State-Change's predicted representations support more accurate visual reconstruction. Using the decoded ground-truth latents of each model as a reference, the smaller PSNR losses on most tasks further indicate that State-Change better preserves decodable visual content during rollout relative to each model's own reconstruction capability.

Figure 8 further illustrates how prediction errors accumulate under image feedback. Predicted images are re-encoded for the next prediction, allowing visual deviations introduced at one step to propagate through the recurrence. Next-State progressively loses the room geometry and displaces the ball from its real trajectory, whereas State-Change better preserves the scene layout and ball position in subsequent predictions.

This comparison illustrates the reconstruction differences reflected in PSNR through observable changes in the scene. In this sequence, improved prediction quality is expressed through the continued preservation of static structure and dynamic position. Consistent with the multi-step latent prediction results, this observation supports the role of state-change modeling in maintaining visual consistency during recurrent prediction.

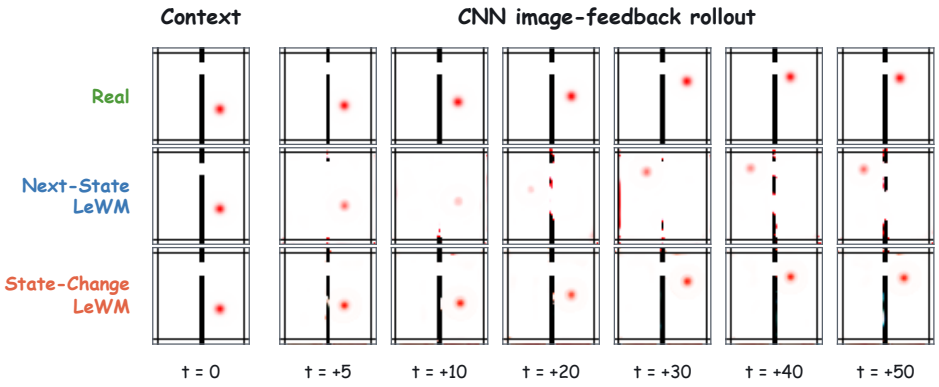


Figure 8: CNN image-feedback rollout on a TwoRoom sequence. Both models use the same $t = 0$ context. At each step, the predicted representation is decoded to an RGB image and re-encoded, then combined with the recorded action block for the next prediction. Six of ten prediction steps are shown.

5.3 Planning under appearance shifts

State-change prediction continues to improve planning when observation appearance changes. We train the models on the original PushT observations and introduce seven visual perturbations at evaluation: recoloring the background, agent, T-shaped block, or goal region, along with darkening, Gaussian blur, and salt-and-pepper noise. Each recoloring condition changes only the specified component to a common replacement color, leaving the other components unchanged. All conditions preserve the original physics, task goals, and planning settings, allowing us to examine how visual changes affect action selection. Figure 9 illustrates the conditions, and Table 5 summarizes planning success.

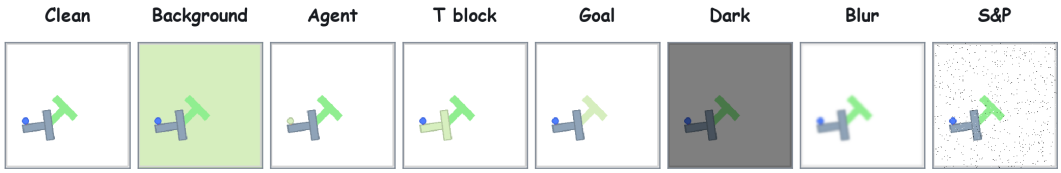


Figure 9: Appearance variations in PushT, shown at the same physical state and goal. Clean shows the original rendering. Background, Agent, T block, and Goal each recolor only the named component. Dark reduces image brightness, Blur applies Gaussian blurring, and S&P adds salt-and-pepper noise.

Table 5: PushT planning success (%) under seven visual perturbations. Values are mean ± standard deviation. Bold marks the highest mean in each row, including ties.

Condition	Next-State LeWM	State-Change LeWM	PLDM	Sub-JEPA
Background	15.6 ± 4.7	21.8 ± 4.5	19.0 ± 5.3	$\mathbf{25.8} \pm 7.7$
Agent	2.4 ± 1.7	$\mathbf{3.4} \pm 2.0$	2.8 ± 2.2	$\mathbf{3.4} \pm 2.5$
T block	11.0 ± 5.8	27.2 ± 9.3	$\mathbf{27.8} \pm 8.6$	27.4 ± 7.3
Goal	76.4 ± 4.5	85.2 ± 4.6	53.0 ± 8.8	$\mathbf{86.2} \pm 5.3$
Dark	3.4 ± 1.8	1.2 ± 1.3	3.0 ± 2.2	$\mathbf{4.0} \pm 2.5$
Blur	85.8 ± 3.5	84.4 ± 4.8	67.0 ± 8.0	$\mathbf{90.8} \pm 3.9$
S&P	77.6 ± 8.5	82.8 ± 5.0	64.0 ± 8.2	$\mathbf{86.2} \pm 6.0$

State-Change outperforms Next-State under all four recoloring conditions, with the largest difference when the T-shaped block changes color. Its advantage thus extends from background changes to changes in the appearance of the object being manipulated. With geometry and dynamics unchanged, these results support the ability of state-change training to improve adaptation to appearance changes, allowing predicted representations to remain useful for action selection.

The benefit also extends to salt-and-pepper noise. Averaged equally across the seven perturbations, success increases from 38.9% for Next-State to 43.7% for State-Change. The conditions reveal distinct sensitivities: the two LeWM variants perform similarly under blur, whereas agent recoloring and darkening reduce success to low levels for all four models. Changes to individual components and image-wide degradations therefore affect planning differently; the improvements from state-change prediction are concentrated in recoloring and salt-and-pepper noise.

The external models provide context for these results. Sub-JEPA leads in average success, while State-Change exceeds PLDM on average and ties Sub-JEPA for the highest mean under agent recoloring. The matched LeWM comparison provides more direct evidence for our study: changing the prediction parameterization while preserving the architecture improves planning under several appearance perturbations. This pattern is consistent with explicitly propagating the current state and learning subsequent changes, further indicating that the prediction target affects how a model uses dynamics for decision-making under altered visual conditions.

6 Discussion and Conclusion

Prediction targets and action-related representations. State-change prediction shapes the encoder through joint training, making actions and associated dynamics easier to extract and use. Policy learning on frozen encoders shows that this advantage persists after removing the original predictor; action and kinematic readouts provide supporting evidence about representation content. Encoder cross-comparisons further show that State-Change representations achieve higher average planning performance under both prediction forms. These results support the role of prediction targets in shaping transition information and its decision utility.

Separating state persistence from change modeling. State-Change LeWM propagates the current representation through an identity path while the predictor learns subsequent updates. Additive components that remain constant over the prediction interval cancel from the difference target, focusing the learned output on state changes. Planning gains under fixed encoders and lower multi-step latent error on most tasks support this prediction form. Higher decoded PSNR and structural preservation in the TwoRoom image-feedback rollout further demonstrate improved prediction in image space. This division of labor supports more effective dynamics prediction and planning at the same model size.

Conclusion. We study how prediction targets affect action-related representations and dynamics learning in LeWM under matched architectures, training data, and regularization. State-Change LeWM achieves higher mean planning success than Next-State on every control task and the highest four-task average among the compared models. Representation-side and predictor-side experiments show that prediction targets shape the information the encoder retains and organizes, as well as how the predictor uses it to roll out future states. In this study, learning state changes while propagating the current state through an identity path improves both representation utility and prediction performance.

AI USE STATEMENT

Generative AI assisted with manuscript structure, wording revisions, claim–evidence checks, and figure layout. The authors take responsibility for the final text, claims, and artifacts.

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*, 2023. URL https://arxiv.org/abs/2301.08243.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2303.04137.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 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 *Proceedings of the 36th International Conference on Machine Learning*, 2019. URL https://arxiv.org/abs/1811.04551.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

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.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

Jiabin Qiu, Zixuan Chen, Hongye Cao, Jieqi Shi, Jing Huo, and Yang Gao. AD-WM: Action-discriminative world models for counterfactual model predictive control. arXiv preprint arXiv:2609.30264, 2026. URL https://arxiv.org/abs/2609.30264.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.14819.

Yuval Tassa, Yotam Doron, Alistair Muldal, et al. Deepmind control suite. arXiv preprint arXiv:1801.00690, 2018. URL https://arxiv.org/abs/1801.00690.

Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. Joint embedding vs reconstruction: Provable benefits of latent space prediction for self supervised learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1fa81061d6d4d7fea88f803d89ae9d6e-Abstract-Conference.html.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861. URL https://doi.org/10.1109/TIP.2003.819861.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. URL https://arxiv.org/abs/1801.03924.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding. arXiv preprint arXiv:2606.31232, 2026. URL https://arxiv.org/abs/2606.31232.

Kai Zhao, Dongliang Nie, Yuchen Lin, Zhehan Luo, Yixiao Gu, Deng-Ping Fan, and Dan Zeng. Sub-JEPA: Subspace gaussian regularization for stable end-to-end world models. arXiv preprint arXiv:2605.09241, 2026. URL https://arxiv.org/abs/2605.09241.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A Method and Experimental Details

A.1 Relation between the Prediction Targets

For a fixed encoder and a context c_t containing z_t, the targets are related by a conditional translation:

$$\mathbb{E}[\Delta z_t \mid c_t] = \mathbb{E}[z_{t+k} \mid c_t] - z_t, \qquad \mathrm{Var}(\Delta z_t \mid c_t) = \mathrm{Var}(z_{t+k} \mid c_t). \tag{7}$$

Thus, $g^*(c_t) = f^*(c_t) - z_t$, and both targets share the same minimum squared-error risk. Equation 5 also gives identical gradients within the same State-Change graph; differences between f_ϕ and $z_t + g_\psi$ reflect finite capacity, initialization, optimization, and the joint encoder response.

A.2 Model and Training

Main world models. We evaluated both released LeWM checkpoints and checkpoints trained using the official recipe. Our reproduced Next-State LeWM achieved a slightly higher four-task mean planning success, so we use the self-trained checkpoints throughout the reported experiments. Both target variants use the shared architecture and training settings detailed below; the additional data-budget runs and suite-specific LIBERO encoders are separate experiments. The evaluation tasks use TwoRoom from PLDM, PushT from Diffusion Policy, Reacher from the DeepMind Control Suite, and Cube from OGBench (Sobal et al., 2025; Chi et al., 2023; Tassa et al., 2018; Park et al., 2025).

The ViT-Tiny/14 image encoder processes 224×224 observations and produces a 192-dimensional class-token feature, which is projected into the prediction space. The predictor has six layers, 16 attention heads, head dimension 64, and feed-forward width 2048. The embedding and prediction projectors have hidden width 2048 and use batch normalization. The configured history and prediction lengths are three and one, respectively, with sampling interval $k = 5$. Each action block concatenates five consecutive actions, giving 10 input coordinates for TwoRoom, PushT, and Reacher, and 25 for Cube. During training, four sampled observations supply three aligned one-step targets to the causal predictor; the loss is averaged over examples, prediction positions, and latent coordinates.

Both variants use ten training epochs, batch size 128, AdamW with learning rate 5×10^{-5} and weight decay 10^{-3}, and BF16 precision. The SIGReg weight is $\lambda = 0.09$ (Maes et al., 2026). SIGReg uses 17 integration knots and 1024 random projections, regularizing the minibatch distribution at each sampled time and averaging over time. The encoder is updated through both context

and target embeddings, without stop-gradient or an exponential-moving-average target encoder. No state-change target normalization, variant-specific auxiliary loss, or zero-initialized output is used. LeWM's internal predictor initialization is retained.

A.3 Evaluation Protocols

Planning series. Table 1 uses ten evaluation seeds, $5, 10, \ldots, 50$, with 50 episodes per seed. All five self-trained checkpoints use 30 CEM iterations on PushT and 10 on the other three environments, with 300 samples and 30 elites. State-Change LeWM, DINO-WM, and Sub-JEPA report sample standard deviation; PLDM reports population standard deviation. These statistics describe repeated evaluation of fixed checkpoints, not independent world-model training runs. Appendix B.1 reports a separate six-seed comparison with 30 CEM iterations on every task.

Visual perturbations. The PushT evaluation (Chi et al., 2023) compares fixed Next-State LeWM, State-Change LeWM, PLDM, and Sub-JEPA checkpoints trained on the original observations. The seven test-time conditions comprise four single-component recolorings and three image-wide degradations. Each recoloring changes the background, agent, T-shaped block, or goal region to the same replacement color, leaving all other components at their original colors. The remaining conditions reduce brightness, apply Gaussian blur, or add salt-and-pepper noise. Physics, task goals, and planning settings remain unchanged across conditions. Figure 9 uses the same physical state and goal to illustrate the original rendering and each perturbation; Table 5 reports results for the seven perturbed conditions.

GCBC. The four-task main encoders are frozen while the goal-conditioned policy is trained. TwoRoom is evaluated at policy epoch 10; the other tasks use epoch-20 policy checkpoints. The State-Change Reacher policy uses a weights warm-start from epoch 10. This downstream training detail is separate from the ten-epoch world-model pretraining protocol. DINOv2 uses a different visual feature interface from LeWM and is included as an external reference (Oquab et al., 2024).

Frozen OFT. For each LIBERO suite (Liu et al., 2023), Stage 1 trains a separate encoder for 40 epochs and Stage 2 freezes it while training an OFT head for 30 epochs. The policy uses two camera views, two-frame context, proprioception, a task identifier, and an eight-action chunk, with action L1 supervision for the head. Each result summarizes ten tasks with 50 rollouts each and a 600-step limit. Evaluation uses the first 50 provided initial states per task. Table 4 reports one run per model and suite, rather than variation across independent encoder seeds.

Cross-combinations. Each crossed predictor is trained on the frozen representation of the other target variant. Table 2 reports the native and crossed pairs. Native pairs are trained end-to-end, whereas crossed pairs refit only the predictor; the table is not a fully matched four-cell factorial design. The reported contrasts use differences of means and do not infer paired uncertainty from marginal dispersion values.

Inverse-action probes. The readout split is episode-disjoint 80/10/10 with split seed 3072. Inputs are real projected latent state changes over a five-action block, and targets are standardized actions. This action standardization is confined to the probe and does not normalize the world-model prediction target. Linear regularization and MLP checkpoint selection use validation data; the MLP uses ten epochs and probe-fit seeds 0, 1, and 2. The split separates probe training from testing but does not establish a held-out test set for world-model pretraining.

Cube transition probes. The Cube analysis uses 100,000 OGBench-Cube frame pairs (Park et al., 2025) and an episode-disjoint 80/10/10 probe split (seed 3072) shared across encoders. From $[z_t, z_{t+k}]$ with $k = 5$, probes predict six-dimensional joint velocity, one-dimensional gripper velocity, or the 25-dimensional action block. Ridge regularization is selected from 10^{-6} to 10^{2} on validation data; the two-layer 256-unit GELU MLP uses validation-based early stopping. We report test Pearson correlations macro-averaged across target dimensions.

Open-loop prediction. The analysis uses one start per episode, history length three, a five-record frame interval, real action blocks, and up to ten free-recurrence prediction steps. There are 928

episodes for TwoRoom and 1000 for each other task. Figure 6 reports the ratio of cumulative squared prediction error to the cumulative error of repeatedly copying the starting latent; shaded bands are episode-bootstrap 95% intervals. These trajectories were not held out from world-model pretraining; the evaluation measures rollout behavior on that data distribution, not out-of-distribution generalization.

CNN image-feedback visualization. Figure 8 uses held-out TwoRoom start 58004 from episode 634. From 64 starts pre-ranked by their pure-latent LPIPS gap, we evaluate image-feedback recurrence and select the case with the largest mean Next-State-minus-State-Change LPIPS gap over horizons 5–10. The selection intentionally favors a visually discriminative example and is used only for qualitative illustration. At each step, the predicted latent is decoded by the model's separately trained CNN decoder; the RGB output is quantized, re-encoded, and supplied to the next prediction together with the recorded action block. The ten-step mean LPIPS is .225141 for Next-State and .038510 for State-Change. The figure displays the shared $t = 0$ context and steps $5, 10, 20, 30, 40, 50$.

Decoded rollout evaluation. Each target variant uses a separately trained CNN decoder. Table 8 reports the selected decoder-checkpoint comparison. The decoder maps the 192-dimensional projected latent to a $512 \times 7 \times 7$ feature map and applies five upsampling stages to produce a 224×224 RGB image. For each rollout, the model encodes three real context frames at times $0, 5, 10$ and then predicts ten latent states at times $15, 20, \ldots, 60$ using recorded five-action blocks. Decoded images are used only for this aggregate evaluation and are never re-encoded. Ground-truth actions are standardized using the population statistics of the complete H5 action column before being grouped into blocks of five. Full-frame PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) are averaged over the evaluated future frames. Table 9 subtracts decoded-rollout PSNR from the PSNR obtained by decoding the encoded ground-truth future latent with the same model-specific decoder.

B Complete Results

B.1 Additional Planning Results

Table 6: Planning success (%) under the Sub-JEPA evaluation protocol, using six evaluation seeds and a shared CEM configuration. This series is separate from the LeWM-protocol main comparison. Next-State and State-Change LeWM use the main checkpoints; Sub-JEPA remains an external comparison rather than a target-only ablation.

Task	Next-State LeWM	State-Change LeWM	Sub-JEPA
TwoRoom	86.33 ± 6.86	90.67 ± 4.84	93.00 ± 5.02
PushT	82.33 ± 6.38	91.67 ± 5.85	90.00 ± 3.35
Reacher	81.00 ± 2.76	81.67 ± 7.31	58.33 ± 2.94
Cube	68.33 ± 6.38	82.33 ± 6.98	76.00 ± 5.66

The evaluation seeds are $42, 100, 2026, 3407, 1234, 4444$, with 50 episodes per seed. All tasks use 30 CEM iterations, 300 samples, and top-$k = 30$. The evaluator configuration sets the horizon, receding horizon, and action block to five, the goal offset to 25, and the evaluation budget to 50. State-Change means exceed Next-State means in all four tasks, with a small difference on Reacher.

B.2 Representation Readouts

B.3 Decoded Rollout Metrics

Table 9: Oracle-relative PSNR loss (dB), computed as ground-truth-latent reconstruction PSNR minus decoded-rollout PSNR. Lower values indicate that prediction stays closer to the reconstruction quality supported by the model's own encoder–decoder pair. Bold marks the lower loss.

Task	Next-State	State-Change
TwoRoom	9.332	**.075**
PushT	.771	**.555**
Reacher	1.532	**.407**
Cube	**.005**	.206

Table 7: Complete action-block readouts for Next-State and State-Change LeWM. Linear/Ridge and MLP probes predict executed actions from real latent differences. MLP dispersion is across three probe fits per fixed encoder.

		Linear/Ridge		MLP	
Task	Encoder	MSE ↓	r ↑	MSE ↓	r ↑
TwoRoom	Next-State	.7848	.4621	$.7936 \pm .0026$	$.4559 \pm .0018$
	State-Change	.7618	.4861	$.7697 \pm .0028$	$.4806 \pm .0017$
PushT	Next-State	.2929	.8388	$.2117 \pm .0014$	$.8879 \pm .0012$
	State-Change	.3921	.7774	$.2846 \pm .0045$	$.8462 \pm .0023$
Reacher	Next-State	.9420	.1956	$.9462 \pm .0014$	$.2082 \pm .0025$
	State-Change	.9226	.2248	$.9331 \pm .0012$	$.2304 \pm .0010$
Cube	Next-State	.5863	.5465	$.4538 \pm .0048$	$.6913 \pm .0039$
	State-Change	.5781	.5709	$.4327 \pm .0051$	$.7195 \pm .0038$

Table 8: Complete decoded open-loop rollout metrics at the selected decoder checkpoints. Each LeWM variant uses its separately trained CNN decoder. Higher PSNR and SSIM are better; lower LPIPS is better. N/S denote Next-State/State-Change, and Δ denotes S minus N.

	PSNR ↑			SSIM ↑			LPIPS ↓		
Task	N	S	Δ	N	S	Δ	N	S	Δ
TwoRoom	15.915	**19.590**	+3.675	.9227	.9416	+.0189	.1954	.0866	−.1089
PushT	25.427	**25.807**	+.380	.9286	.9156	−.0130	.1022	.1234	+.0212
Reacher	29.242	**29.915**	+.673	.9612	.9602	−.0010	.0875	.0795	−.0081
Cube	20.370	**20.711**	+.341	.8064	.7944	−.0120	.3217	.3377	+.0160