LOOP LEWORLDMODEL: RELIABLE RECURSION FOR DEPTH REUSE AND TEMPORAL ROLLOUT IN LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual world models based on joint embedding predictive architectures (JEPAs) support planning by predicting action consequences before execution. As planning repeatedly applies the predictor to imagined latent states, recursive layer sharing offers a natural way to organize iterative latent processing within each model call. However, naively looping pretrained layers across depth erases depth-specific computations, and recursively feeding predicted states back often accumulates errors and destabilizes imagined trajectories. To address these two issues, we propose Loop LeWorldModel, a loop-based adaptation of pretrained visual world models. Specifically, we utilize depth-specific low-rank adapters to preserve distinct transformations at each loop. Additionally, we design a training-free self-correction gate to suppress abnormal prediction steps using recent motion statistics. Experiments across four visual-control tasks show that Loop LeWorldModel improves planning success with fewer stored parameters and reduces the magnitudes of large latent transitions during rollout. *All source codes will be released online.*

1 INTRODUCTION

A significant goal in visual control is to learn models that predict how actions change the visual world and utilize those predictions to select future actions. World models (Ha & Schmidhuber, 2018) support this goal by evaluating candidate futures in imagination rather than executing every possibility in the environment. This capability is particularly useful in offline visual control, where dynamics learned from fixed trajectories must generalize to new goals. Joint-embedding predictive architectures (JEPAs) provide an appealing formulation by predicting embeddings in a compact latent space instead of reconstructing every pixel (Assran et al., 2023; Bardes et al., 2024). Recent systems such as LeWorldModel (Maes et al., 2026) and DINO-WM (Zhou et al., 2025) use Transformer-based encoders and action-conditioned predictors during cross-entropy method (CEM) search (de Boer et al., 2005), repeatedly evaluating candidate futures before an action is executed. This repeated use makes the organization of latent computation particularly important. Since the predictor repeatedly processes its own outputs during autoregressive rollout, reusing transformations within each model call offers a complementary form of iterative latent processing.

This form of iterative computation has also been studied in large language models through recursive or recurrent-depth Transformers, where a smaller set of blocks is repeatedly reused across depth. Specifically, Saunshi et al. (2025) show that looping-inspired regularization of corresponding layers improves performance on selected reasoning benchmarks, supporting an inductive bias toward iterative latent processing. Recursive conversion of pretrained Transformers can recover depth-dependent computation using layer-wise low-rank adaptations (Bae et al., 2025), while recurrent-depth architectures show that repeated latent computation can provide additional effective depth (Geiping et al., 2025). More recently, Looped World Models (Lu et al., 2026) extended this principle to world modeling through iterative latent refinement and adaptive computation. Together, these studies motivate depth reuse as a computational structure for repeatedly processing latent representations within autoregressive world models.

However, few works have explored the *post-training recursive conversion of pretrained visual JEPAs for planning*. Unlike prior work that either converts pretrained language models or develops looped world models as native architectures, we start from LeWorldModel(Maes et al., 2026), a pretrained visual encoder-predictor model, and convert it into a recursive architecture while preserving the

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.

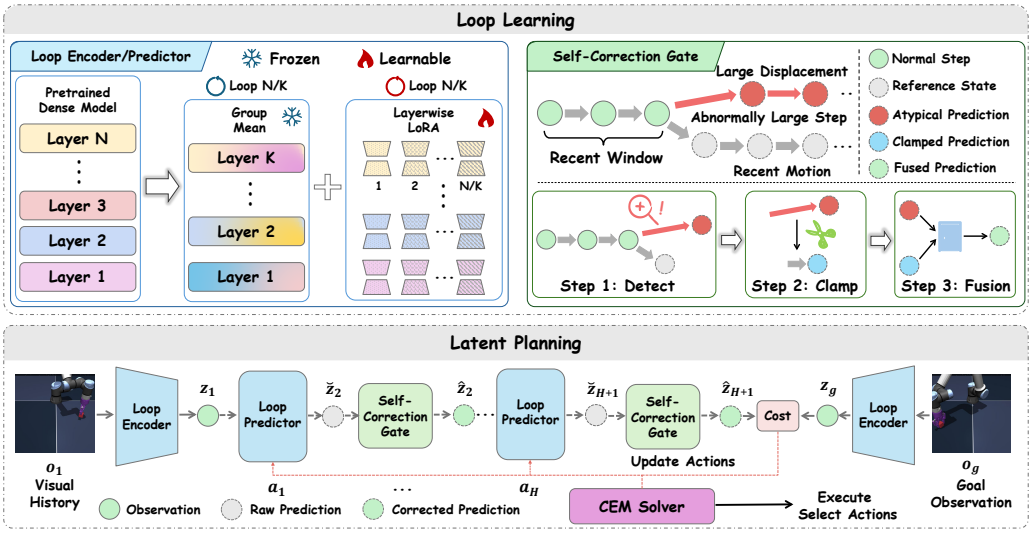


Figure 1: **Overview of Loop LeWorldModel. Loop Learning:** The Loop Encoder/Predictor reuses frozen group-mean layers with learnable layerwise LoRA. The Self-Correction Gate detects large steps using the recent window, clamps their magnitudes, and fuses raw and clamped predictions. **Latent Planning:** The Loop Encoder embeds history and goal observations. The Loop Predictor and gate generate corrected rollouts under candidate actions. The terminal goal-matching cost guides the CEM Solver to update and select actions for execution.

layer-specific functions acquired during pretraining. Forcing multiple pretrained layers to reuse the same parameters can erase these layer-specific functions. Meanwhile, recursive latent rollout introduces a separate challenge: the predictor consumes its own predictions during planning (Maes et al., 2026), allowing errors to accumulate over time. We therefore consider two forms of recursion: *depth recursion* from repeated layer reuse and *temporal recursion* from autoregressive rollout.

To address these challenges, we propose a loop-based adaptation of pretrained visual world models, namely Loop LeWorldModel. Figure 1 summarizes how the two forms of recursion are handled within latent planning. For depth recursion, we convert the pretrained encoder–predictor pair into a Loop Encoder/Predictor by reusing frozen group-mean blocks. Depth-specific low-rank adapters, initialized from pretrained layer residuals through truncated singular value decomposition, retain the distinct transformations at each effective depth without storing separate dense layers. We then jointly recover the encoder and action-conditioned predictor to preserve their compatibility for goal-conditioned planning. For temporal recursion, the training-free Self-Correction Gate detects atypical steps relative to recent latent motion, clamps their magnitudes, and fuses raw and clamped predictions. The corrected states then enter candidate evaluation and subsequent rollout steps. Because correction requires no weight updates, it also applies to untied predictors and addresses temporal reliability independently of depth sharing. Together, these components support compact and reliable latent planning.

We evaluate Loop LeWorldModel on four visual-control tasks spanning manipulation, navigation, and locomotion in both 2D and 3D environments. The experiments examine planning after recursive conversion and the additional benefit of temporal correction, showing improved success with fewer parameters than the original pretrained model.

In summary, our key contributions are listed as follows:

- We develop a Loop Encoder/Predictor with joint adapter recovery to retain depth-specific computation and compatible latent dynamics with fewer stored parameters.
- We introduce a training-free Self-Correction Gate that uses recent latent motion to suppress atypical rollout transitions and improve candidate evaluation.
- We evaluate depth reuse and temporal correction on four control tasks, showing stronger planning with fewer parameters and correction gains on untied models.

2 Related Work

In this section, we briefly review world models for planning, the visual representations used for latent prediction, and recursive sharing with parameter-efficient adaptation.

Latent World Models and Planning. World models support control by training policies on imagined trajectories (Hafner et al., 2020; 2021), with applications to a wide range of control tasks (Hafner et al., 2025) and self-supervised exploration (Sekar et al., 2020). Other methods evaluate candidate actions through tree search (Schrittwieser et al., 2020; Ye et al., 2021) or model predictive control (Hansen et al., 2022; 2024). The first group uses model predictions to train a policy, while the second uses them to select actions during planning. Within this landscape, model design and application form complementary comparison axes. For model design, IRIS, TWM, and STORM use Transformers for latent dynamics (Micheli et al., 2023; Robine et al., 2023; Zhang et al., 2023b), whereas DIAMOND generates images through diffusion (Alonso et al., 2024). For applications, Genie and UniSim generate interactive environments (Bruce et al., 2024; Yang et al., 2024b), whereas DINO-WM and LeWorldModel match predicted embeddings to visual goals for planning (Zhou et al., 2025; Maes et al., 2026). However, autoregressive errors can alter candidate rankings. Receding-horizon planning obtains new observations (Maes et al., 2026), while self-correcting dynamics models train on their generated states (Talvitie, 2017). Different from these reliability mechanisms, our gate corrects atypical steps within imagined rollouts without further observations or training.

Joint-Embedding Predictive Architectures. Visual pretraining learns features through reconstruction (He et al., 2022; Tong et al., 2022) or self-distillation (Caron et al., 2021; Oquab et al., 2024). I-JEPA and V-JEPA instead predict embeddings of masked image or video regions, avoiding pixel-level reconstruction (Assran et al., 2023; Bardes et al., 2024). VICReg controls feature variance and covariance to prevent feature collapse (Bardes et al., 2022), while LeJEPA encourages embeddings to follow an isotropic Gaussian distribution (Balestriero & LeCun, 2025). Predicting future features as an additional training task (Schwarzer et al., 2021) and pretraining on videos without action labels (Seo et al., 2022) also help learn representations for control. Building on these visual representations, DINO-WM and V-JEPA 2-AC learn predictors using features from frozen visual encoders (Zhou et al., 2025; Assran et al., 2025), whereas LeWorldModel trains its encoder and predictor together (Maes et al., 2026). These models use the same latent space to predict state changes and compare predicted states with visual goals. Changing the encoder can therefore change the predictor's inputs and targets, as well as the distances between predicted states and goals. Existing JEPA-based world models focus on learning these representations and dynamics. Less work addresses how to keep the encoder and predictor compatible when the model structure is changed after training.

Recursive Depth Sharing and Parameter-Efficient Adaptation. Sharing parameters across layers supports iterative latent processing. ALBERT shares parameters across layers (Lan et al., 2020), while looped Transformers repeatedly apply the same blocks for algorithmic computation (Giannou et al., 2023; Yang et al., 2024a) and reasoning (Saunshi et al., 2025; Geiping et al., 2025). This allows deep computation without separate weights at every layer. A related line of work reduces the number of parameters updated during adaptation. LoRA keeps pretrained weights fixed and learns low-rank updates (Hu et al., 2022). PiSSA initializes adapters from the leading singular components of pretrained weights (Meng et al., 2024), while AdaLoRA adjusts adapter ranks within a parameter budget (Zhang et al., 2023a). Relaxed Recursive Transformers combine sharing with low-rank adaptation by adding depth-specific LoRA to shared blocks, recovering layer-specific transformations from a pretrained model (Bae et al., 2025). Training after conversion updates both the shared backbone and the adapters. A shared block can then apply different transformations at different depths without storing a full layer for each depth. In world modeling, LoopWM uses a shared dynamics block to refine latent states over multiple passes, with adaptive depth and spectral constraints (Lu et al., 2026). These studies explore recursive conversion of pretrained language models and the design of looped world models. By contrast, our setting involves converting a pretrained visual JEPA into a looped model while preserving the computation learned at each layer of both the encoder and predictor.

3 METHOD

As illustrated in Figure 1, our method consists of three main components: Loop Encoder, Loop Predictor, and training-free Self-Correction Gate. The Loop Encoder and Predictor address depth recursion by reusing shared blocks with layerwise adapters, while the Self-Correction Gate addresses temporal recursion by suppressing atypical transitions during latent rollout. These components are integrated into latent planning to evaluate candidate futures and guide the agent toward goal-directed action selection.

3.1 LOOP ENCODER/PREDICTOR

Let o_t and a_t denote the image observation and action at time t. A visual JEPA consists of an encoder E_θ and an action-conditioned predictor P_ϕ, parameterized by θ and ϕ, respectively. The encoder produces latent states $z_t = E_\theta(o_t) \in \mathbb{R}^{d_z}$, where d_z is the latent dimension, and the predictor produces $\check{z}_{t+1} = P_\phi(z_{\leq t}, a_{\leq t})$, where the subscript $\leq t$ denotes the available history up to t. We reserve z for encoded observations and $\check{z}$ for raw predictions; Section 3.2 introduces clamped states $\tilde{z}$ and corrected states $\hat{z}$.

A configuration $(s_e, \ell_e; s_p, \ell_p)$ specifies s_e shared encoder blocks repeated ℓ_e times and s_p shared predictor blocks repeated ℓ_p times. The effective depths are $D_e = s_e\ell_e$ and $D_p = s_p\ell_p$. We retain LeWorldModel's depths $D_e = 12$ and $D_p = 6$; for example, $(4, 3; 2, 3)$ repeats four encoder blocks and two predictor blocks three times each.

Shared Bases and Layerwise Adapters. Following relaxed recursive conversion (Bae et al., 2025), we initialize shared blocks from pretrained layers. For either network, let D be its effective depth and s its number of shared blocks, where s can be divided by D. Index depths by $d \in \{0, \ldots, D-1\}$ and shared slots by $j \in \{0, \ldots, s-1\}$. Define that Slot $j(d) = d \bmod s$ is used at depth d, and $\mathcal{S}_j = \{d : j(d) = j\}$ collects its assigned depths. Let m index an attention or MLP linear projection, with input and output widths d_m^{in} and d_m^{out}. Its pretrained weight is $W_{m,d}^{\text{off}} \in \mathbb{R}^{d_m^{\text{out}} \times d_m^{\text{in}}}$.

Lemma 1 (Least-Squares Optimal Shared Representative)**.** *For a nonempty slot $\mathcal{S}_j$, the unique matrix minimizing the sum of squared Frobenius distances to its pretrained weights is their mean:*

$$\bar{W}_{m,j} = \arg\min_W \sum_{d \in \mathcal{S}_j} \|W_{m,d}^{\text{off}} - W\|_F^2 = \frac{1}{|\mathcal{S}_j|} \sum_{d \in \mathcal{S}_j} W_{m,d}^{\text{off}}, \tag{1}$$

where W has the same dimensions as $W_{m,d}^{\text{off}}$, $|\mathcal{S}_j|$ is the number of assigned depths, and $\|\cdot\|_F$ is the Frobenius norm.

Hard tying discards the layer-specific residual $\Delta W_{m,d} = W_{m,d}^{\text{off}} - \bar{W}_{m,j(d)}$. To retain these differences without storing dense residuals, we use rank-r adapters, where r controls their capacity.

Lemma 2 (Optimal Rank-r Matrix Residual)**.** *Fix $\bar{W}_{m,j(d)}$ and an integer $1 \leq r \leq \min(d_m^{\text{out}}, d_m^{\text{in}})$. Let $\Delta W_{m,d} = U\Sigma V^\top$ be a singular value decomposition, with orthonormal singular vectors in U and V and descending nonnegative singular values in the diagonal matrix Σ. We write $U_{:,r}$ and $V_{:,r}$ for their first r columns, and $\Sigma_{r,r}$ for the leading $r \times r$ block of Σ. The Eckart–Young–Mirsky theorem minimizes Frobenius-norm error under a rank limit of r by retaining the r largest singular values and their vectors. Thus,*

$$\Delta W_{m,d}^{(r)} = U_{:,r}\Sigma_{r,r}V_{:,r}^\top \in \arg\min_{\text{rank}(X) \leq r} \|\Delta W_{m,d} - X\|_F, \tag{2}$$

where $X \in \mathbb{R}^{d_m^{\text{out}} \times d_m^{\text{in}}}$ is a candidate residual approximation and $\text{rank}(X)$ is its matrix rank. The low-rank factors are initialized as $B_{m,d} = U_{:,r}\Sigma_{r,r}$ and $A_{m,d} = V_{:,r}^\top$.

Both lemmas concern matrix approximation; the nonlinear model still requires recovery. Their proofs appear in Appendix B. The effective weight at depth d is

$$W_{m,d}^{\text{loop}} = \bar{W}_{m,j(d)} + B_{m,d}A_{m,d}, \quad A_{m,d} \in \mathbb{R}^{r \times d_m^{\text{in}}}, \quad B_{m,d} \in \mathbb{R}^{d_m^{\text{out}} \times r}. \tag{3}$$

Let x_d denote the hidden token sequence at depth d and F_j the block in slot j. The predictor's action context and depth-specific adaptive layer normalization (AdaLN) (Peebles & Xie, 2023) parameters are collected in η_d, an argument omitted for the encoder. Repeated computation follows

$$x_{d+1} = F_{j(d)}\left(x_d; \{W_{m,d}^{\text{loop}}\}_m, \eta_d\right), \tag{4}$$

where $\{W_{m,d}^{\text{loop}}\}_m$ collects the effective weights in Eq. (3) across attention and MLP projections m, with depth d fixed. Appendix C.3 reports parameter counts.

Adapter-Only Uptraining. We recover the converted model on replay trajectories, training only adapter factors and predictor AdaLN copied per depth. Shared weights, averaged biases, and averaged normalization parameters remain frozen. Appendix A provides conversion details, including the encoder's auxiliary pre-MLP normalization. The recovery objective is

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda_{\text{sig}}\mathcal{L}_{\text{sig}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}, \tag{5}$$

where $\mathcal{L}_{\text{pred}}$ and $\mathcal{L}_{\text{sig}}$ follow the next-embedding prediction loss and SIGReg regularizer of LeWorldModel (Maes et al., 2026). These two terms align the encoder with the action-conditioned dynamics and discourage collapse through SIGReg's isotropic Gaussian embedding target (Balestriero & LeCun, 2025). The nonnegative weights λ_{sig} and λ_{dist} control regularization and distillation.
For the Loop student S and frozen pretrained teacher T, we use

$$\mathcal{L}_{\text{dist}} = \text{MSE}(\breve{z}^S, \breve{z}^T) + \tfrac{1}{2}\,\text{MSE}(z^S, z^T), \tag{6}$$

where z^S and z^T are embeddings, $\breve{z}^S$ and $\breve{z}^T$ denote one-step predictions from the same sampled replay observations and actions, and each predictor uses its own encoder's embeddings. Mean squared error (MSE) averages over the batch, matched sequence positions, and latent dimensions. The prediction term retains teacher behavior; the embedding term anchors encoder geometry. Appendix C gives training and SIGReg settings.

3.2 Self-Correction Gate

The gate follows the Detect, Clamp, and Fusion steps in Figure 1, using recent latent motion as its reference.
Within a planning call, let t_0 be the first available observation time and $H_k = (h_{t_0}, \ldots, h_k)$ the history at time k, with $n_k = |H_k|$ states. Each h_i is an encoded observation or an appended corrected prediction. The proposal is $\breve{z}_{k+1} = P_\phi(H_k, a_{\le k})$, where $a_{\le k}$ contains past and candidate actions up to k. The predictor reads its context window; the gate retains the full history. Define its displacement $\delta_k = \breve{z}_{k+1} - h_k$ and magnitude $q_k = \max(\|\delta_k\|_2, \epsilon)$, where $\|\cdot\|_2$ is the Euclidean norm and $\epsilon > 0$ prevents division by zero. The minimum history length $k_0 \ge 2$ and reference-window length $w \ge 1$ are integers. For $n_k \ge k_0$, let r_k be the lower median of the last $\min(w, n_k - 1)$ displacement norms in H_k, floored at ϵ. For an even count, the lower median selects the smaller central value, avoiding averaging a large step into the reference.

Detect. Let $u_k = q_k / r_k$ measure the proposal's size relative to recent motion, $\tau > 0$ be the activation threshold, and $\alpha > 0$ control transition sharpness. The soft trigger is

$$g_k = \sigma(\alpha(u_k - \tau)), \qquad n_k \ge k_0, \tag{7}$$

where $\sigma(v) = 1/(1 + e^{-v})$ is the logistic sigmoid. For $n_k < k_0$, we set $g_k = 0$ and $\hat{z}_{k+1} = \breve{z}_{k+1}$, skipping Clamp and Fusion.

Clamp. For eligible steps, let $c > 0$ be the maximum clamped displacement relative to r_k. The displacement multiplier s_k and clamped state $\tilde{z}_{k+1}$ are

$$s_k = \min\left(1, \frac{c\, r_k}{q_k}\right), \qquad \tilde{z}_{k+1} = h_k + s_k \delta_k. \tag{8}$$

Clamping retains the proposal's direction while limiting its magnitude.

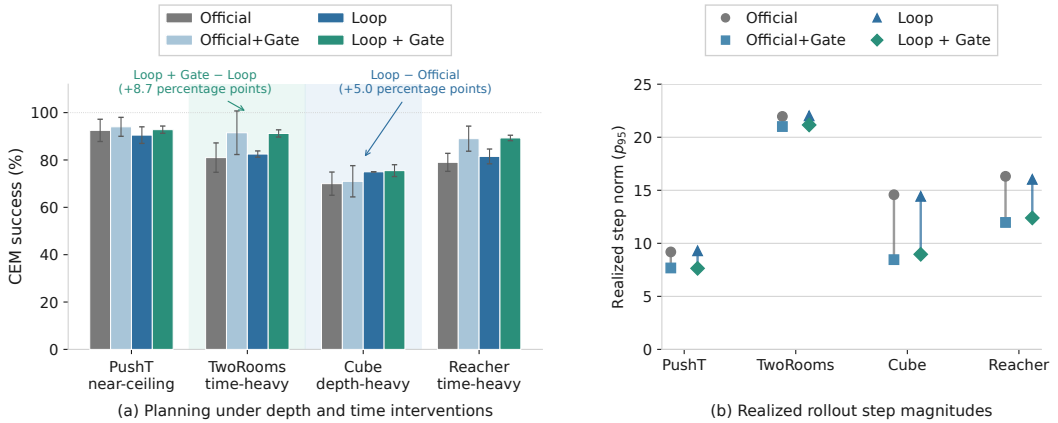
Fusion. With a mixing strength $\rho \in [0, 1]$, the corrected state is

$$\hat{z}_{k+1} = \breve{z}_{k+1} + \rho\, g_k(\tilde{z}_{k+1} - \breve{z}_{k+1}). \tag{9}$$

Steps well below the threshold change little; large steps are shortened along the predicted direction. We set $h_{k+1} = \hat{z}_{k+1}$, giving $H_{k+1} = (h_{t_0}, \ldots, h_k, \hat{z}_{k+1})$ for the next prediction. Its new displacement $\|\hat{z}_{k+1} - h_k\|_2$ also enters the next reference window. All gate hyperparameters are calibrated once on PushT with $(4, 3; 2, 3)$ and then fixed across tasks, configurations, and the untied Official model (Table 6 in Appendix C.2). Algorithm 1 in Appendix A.2 summarizes the correction step.

3.3 Latent Planning

In the lower panel of Figure 1, the Loop Encoder maps observation history through the current time t into $H_t = (z_{t_0}, \ldots, z_t)$ and the goal observation o_g into $z_g = E_\theta(o_g)$; the subscript g denotes the goal. For a candidate action sequence $\mathbf{a} = (a_t, \ldots, a_{t+H-1})$, where the positive integer H



(a) Planning under depth and time interventions

(b) Realized rollout step magnitudes

Figure 2: **Planning gains and reduced rollout step magnitudes.** (a) CEM success for $(4, 3; 2, 3)$ and Official. (b) The 95th percentile of realized latent step norms in free rollouts; gated variants are measured after correction.

is the planning horizon, the Loop Predictor and Self-Correction Gate append corrected states for $k = t, \ldots, t + H - 1$. We score the resulting terminal prediction by

$$J(\mathbf{a}) = \|\hat{z}_{t+H}(\mathbf{a}) - z_g\|_2^2, \tag{10}$$

where J is the terminal goal cost and $\hat{z}_{t+H}(\mathbf{a})$ denotes the final corrected state. The CEM Solver (de Boer et al., 2005) iteratively refits its sampling distribution to the lowest-cost sampled sequences, then selects actions for receding-horizon execution. Model weights and gate settings remain fixed; correction affects action selection through candidate rollouts.

4 Experiments

We evaluate Loop LeWorldModel for autoregressive latent planning through depth reuse and temporal correction across diverse visual-control tasks and recursive configurations. Section 5 further analyzes the contributions of depth recovery and temporal correction through ablation studies.

4.1 Experimental Setup

We evaluate on the same four visual-control tasks as LeWorldModel (Maes et al., 2026): PushT, TwoRooms, Cube, and Reacher, with Cube implemented in OGBench (Park et al., 2025). For each task, we convert the official checkpoint into 8 configurations, using the notation $(s_e, \ell_e; s_p, \ell_p)$ defined in Section 3.1. All preserve the effective encoder and predictor depths ($D_e = 12$, $D_p = 6$) and receive 5 epochs of adapter-only uptraining under three independent training seeds ($\{3072, 0, 2027\}$). Adapter ranks are 24 for PushT, TwoRooms, and Reacher, and 16 for Cube. Planning uses CEM with 300 samples, 30 iterations, and horizon 5. Each checkpoint is evaluated over four CEM seeds ($\{42, 123, 456, 789\}$), with 50 episodes per seed. We average these evaluations within each training run and summarize performance across the three runs. The released Official checkpoint is evaluated over the same four CEM seeds. Table entries and error bars indicate means and standard deviations (SD): across training runs for the main Loop results, and across CEM seeds for Official controls and other single-checkpoint ablations. With one initial observation and horizon 5, the gate first becomes eligible at the terminal prediction and affects action selection through its goal cost. Training and planning settings appear in Appendix C.

4.2 Results

Improvements of the Loop Encoder/Predictor. Table 1 compares eight sharing schedules to assess how encoder/predictor depth reuse with recovery affects planning. Every ungated schedule improves mean success over Official across four tasks. The largest mean gain, 1.8 percentage points, comes from $(4, 3; 2, 3)$, with 43–47% fewer stored parameters (Table 8 in Appendix C.3). For this configuration, Cube benefits most from Loop, gaining 5.0 points; The performance on TwoRooms and Reacher are also improved. Although PushT loses 2.0 points, its success rate remains above 90% and retaining six encoder blocks narrows or reverses this loss under the two predictor schedules, suggesting that lighter encoder sharing better preserves the transformations needed for PushT.

Table 1: **Complete planning-success matrix (%).** Configurations follow the notation in Section 3.1. "Avg." is the unweighted mean across the four tasks; Δ Official is its gain over Official in percentage points, computed before rounding.

Factorization	Variant	PushT	TwoRooms	Cube	Reacher	Avg.	Δ Official
Official	Untied	92.5 ±4.7	81.0 ±6.2	70.0 ±4.9	79.0 ±3.8	80.6	–
$(4, 3; 2, 3)$	Loop	90.5 ±3.5	82.5 ±1.3	75.0 ±0.0	81.5 ±3.1	82.4 ±1.6	+1.8
	Loop + Gate	92.8 ±1.5	91.2 ±1.5	75.5 ±2.5	89.3 ±1.2	87.2 ±1.3	+6.6
$(4, 3; 3, 2)$	Loop	90.8 ±1.6	82.3 ±0.3	71.5 ±1.3	79.5 ±2.0	81.0 ±0.4	+0.4
	Loop + Gate	91.7 ±0.6	91.7 ±0.3	72.3 ±2.1	86.5 ±2.2	85.5 ±1.1	+4.9
$(2, 6; 2, 3)$	Loop	90.8 ±2.6	81.2 ±1.6	74.8 ±1.5	81.2 ±2.1	82.0 ±1.3	+1.4
	Loop + Gate	92.2 ±0.8	91.7 ±0.3	74.5 ±2.2	87.3 ±1.6	86.4 ±0.6	+5.8
$(2, 6; 3, 2)$	Loop	90.5 ±1.3	81.5 ±2.0	72.0 ±1.0	83.2 ±0.6	81.8 ±0.6	+1.2
	Loop + Gate	93.5 ±0.5	**92.7 ±0.6**	73.5 ±0.5	87.8 ±2.8	86.9 ±0.7	+6.3
$(3, 4; 2, 3)$	Loop	90.7 ±1.4	82.3 ±0.3	73.5 ±0.9	80.8 ±2.0	81.8 ±0.3	+1.2
	Loop + Gate	92.5 ±0.9	91.5 ±1.8	74.0 ±2.0	88.2 ±2.6	86.5 ±0.8	+5.9
$(3, 4; 3, 2)$	Loop	92.2 ±1.0	82.5 ±0.9	71.5 ±3.9	80.8 ±4.2	81.8 ±1.8	+1.1
	Loop + Gate	93.8 ±1.0	91.5 ±1.8	72.7 ±0.8	**89.7 ±1.4**	86.9 ±0.4	+6.3
$(6, 2; 2, 3)$	Loop	92.3 ±0.8	81.8 ±0.6	72.5 ±0.5	82.0 ±0.9	82.2 ±0.2	+1.5
	Loop + Gate	93.3 ±1.9	91.2 ±2.1	**76.5 ±1.0**	88.7 ±2.8	**87.4 ±0.9**	+6.8
$(6, 2; 3, 2)$	Loop	93.2 ±1.3	82.0 ±0.5	73.3 ±1.6	80.5 ±3.5	82.3 ±0.1	+1.6
	Loop + Gate	**94.2 ±1.0**	91.3 ±0.3	73.7 ±2.4	88.7 ±2.9	87.0 ±1.1	+6.3

Effectiveness of the Self-Correction Gate. The Gate improves mean success across all eight schedules in Table 1 without adding learned parameters. Its largest task-level gain is 11.2 points on TwoRooms with $(2, 6; 3, 2)$; its only loss is 0.3 points on Cube with $(2, 6; 2, 3)$. The best combined model, $(6, 2; 2, 3)$, exceeds Official by 6.8 points on average. Furthermore, Figure 2(a) compares Official and Loop $(4, 3; 2, 3)$, each with and without the Gate. Both backbones gain most from correction on TwoRooms and Reacher, suggesting greater sensitivity to candidate-cost errors. Autoregressive feedback can accumulate errors that make candidates appear spuriously close to the goal; the Gate shortens atypical terminal jumps before CEM scoring. Figure 2(b) further tests whether correction limits large latent transitions in free rollouts. The Gate lowers the 95th percentile of realized step norms for both backbones on all four tasks. For Loop, the reduction is largest on Cube (37.9%) and smallest on TwoRooms (3.9%). These reductions show that correction limits the large steps fed into later predictions, complementing its terminal-cost intervention during planning. Detailed diagnostics are available in Table 11 in Appendix F.1.

Task-Dependent Contributions of Loop and Gate. As shown in Table 1, with $(4, 3; 2, 3)$, Loop contributes most of Cube's improvement, whereas TwoRooms and Reacher benefit more from the Gate. On PushT, correction offsets Loop's small loss. The combined model therefore exceeds Official on all four tasks. More training-run results are available in Appendix G.

5 Ablation Studies

To explain the planning gains, we ablate $(4, 3; 2, 3)$, which has the highest ungated equal-task mean among the eight configurations. Rank and training-scope comparisons use training seed 3072 and four CEM seeds, so their SD measures planning variability; Loop references in Figure 3 aggregate three training runs.

5.1 Loop Training and Depth-Specific Computation

Loop Training and Effective Depth. Figure 3 assesses Loop training and effective depth. *Hard-tie* uses group-mean pretrained bases without adapters ($r = 0$); *Conversion-only* adds depth-specific residual adapters initialized by singular value decomposition (SVD). Neither receives Loop training. Both and the Loop variants use $(4, 3; 2, 3)$: four shared encoder and two shared predictor blocks repeated three times, giving depths 12 and 6. *Shallow + LoRA* instead applies only the first four encoder and first two predictor blocks once ($(4, 1; 2, 1)$). It and Loop train only adapters and predictor AdaLN for 5 epochs with frozen bases. Conversion-only and trained variants use $r = 24$ except $r = 16$ on Cube; only Loop + Gate corrects rollouts. Official is the untied pretrained model

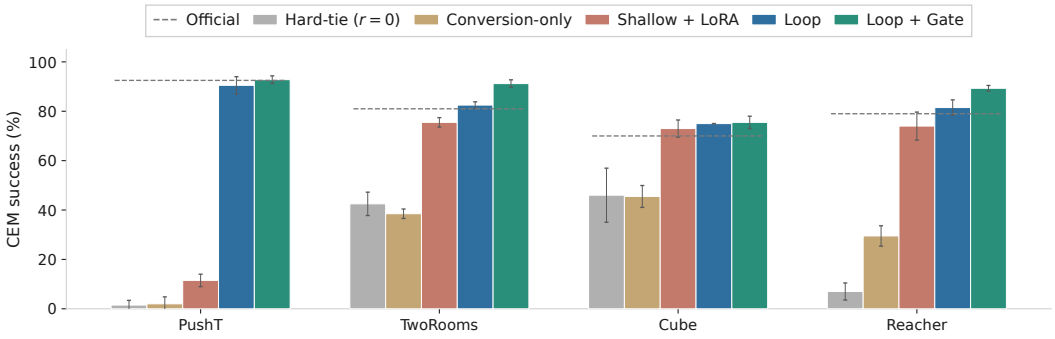


Figure 3: **Loop training and effective-depth controls.** Bars compare planning results for conversion controls, reduced-depth adaptation, and trained Loop variants across four tasks. Dashed lines denote Official.

Table 2: **Functional contribution of Loop adapters.** Frobenius ratios report mean $\|BA\|_F/\|\bar{W}\|_F$ as percentages. MSE ratios compare shared-only/full predictor error with Loop encoder inputs and targets fixed. Encoder cosine compares shared-only/full outputs in units of 10^{-3}. Stable rank averages $\|BA\|_F^2/\|BA\|_2^2$; $\|\cdot\|_2$ is the spectral norm.

Task	Encoder Ratio (%)	Predictor Ratio (%)	Shared-Only MSE Ratio	Encoder Cosine ($\times 10^{-3}$)	Encoder Stable Rank	Predictor Stable Rank
PushT	85.3	79.3	14.5×	4.6	12.1	11.7
TwoRooms	93.5	84.6	53.9×	−10.2	9.7	10.5
Cube	79.0	75.9	28.5×	−8.7	8.9	9.3
Reacher	91.1	83.5	16.6×	−6.7	10.3	12.5

with depths 12 and 6. On PushT, trained Loop reaches 90.5%, compared with 1.5% for Hard-tie, 2.0% for Conversion-only, and 11.5% for trained Shallow + LoRA. The gain over Conversion-only establishes the contribution of Loop training; the advantage over Shallow + LoRA across all four tasks supports retaining effective depth for iterative latent processing.

Functional Contribution of Adapters. Table 2 quantifies the adapters' functional contribution. Mean adapter-to-base Frobenius norm ratios reach 75.9–93.5%. Removing predictor adapters increases one-step MSE 14.5–53.9-fold with full-encoder inputs and targets fixed; removing encoder adapters makes outputs nearly orthogonal to full-encoder embeddings. The low-rank residuals therefore supply transformations essential to both Loop representations and prediction.

For a graded intervention, we scale each module's adapters separately to $\gamma B_{m,d}A_{m,d}$, $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$, fixing inputs and references. Predictor MSE decreases monotonically with its adapter scale on every task. More functional and weight analyses are available in Appendix H; atlas construction and profile interpretation are in Appendices H.2–H.3.

5.2 Adapter Capacity and Training Scope

Adapter Rank. Table 3 identifies compact rank choices by varying r from 8 to 32. PushT and TwoRooms peak at $r = 24$ (94.0% and 84.0%) with 10.3M parameters, outperforming the larger 11.1M rank-32 models. Cube peaks at $r = 16$ (75.0%) with 9.6M parameters. Reacher benefits from $r = 32$ (88.5%), while $r = 8$ provides a smaller 8.9M option at 85.0%. Thus, moderate ranks capture the strongest results for three tasks, while Reacher offers a measurable size–success trade-off. We retain earlier CEM-selected ranks; Appendix C.1 documents overlapping selection and evaluation seeds. More per-seed results are available in Appendix D.

Training Scope. Table 4 compares adapter-and-AdaLN training with full-backbone fine-tuning (Full-FT). Loop achieves higher mean success on all four tasks, with and without gating. For example, ungated Loop reaches 94.0% on PushT and 84.0% on Reacher, compared with 89.0% and 76.0% for Full-FT. After gating, Loop retains a 2.5-point advantage on Cube (78.0% versus 75.5%). Shared-weight updates affect every recurrence, whereas LoRA and AdaLN provide separate adjustments at each depth. The advantage over Full-FT suggests that depth-specific specialization is a more effective use of training capacity in this recursively shared model.

Table 3: **Adapter-rank ablation for $(4, 3; 2, 3)$ (ungated success, %).** The table shows planning results for training seed 3072; † marks the rank used in the main experiments.

Task	$r = 8$	$r = 16$	$r = 24$	$r = 32$
PushT	84.0 ±2.8	89.5 ±2.5	**94.0 ±3.3**†	93.5 ±3.4
TwoRooms	76.5 ±4.4	80.5 ±4.1	**84.0 ±5.2**†	81.5 ±6.6
Cube	71.5 ±1.0	**75.0 ±4.8**†	73.0 ±2.6	73.0 ±5.3
Reacher	85.0 ±3.5	81.5 ±4.1	84.0 ±4.3†	**88.5 ±2.5**
Total Params	8.9M	9.6M	10.3M	11.1M

Table 4: **Planning with and without the fixed gate (success, %).** Results use four CEM seeds per checkpoint. Loop and Full-FT use $(4, 3; 2, 3)$, training seed 3072; Official uses its released checkpoint. Full-FT and Loop differ in implementation and optimization settings.

	Official		Loop		Full-FT	
Task	No Gate	+ Gate	No Gate	+ Gate	No Gate	+ Gate
PushT	92.5 ±4.7	94.0 ±4.0	94.0 ±3.3	**94.5 ±2.5**	89.0 ±4.8	90.0 ±3.3
TwoRooms	81.0 ±6.2	91.5 ±9.2	84.0 ±5.2	**92.5 ±4.4**	82.5 ±6.2	92.0 ±5.9
Cube	70.0 ±4.9	71.0 ±6.6	75.0 ±4.8	**78.0 ±4.3**	71.0 ±10.5	75.5 ±5.0
Reacher	79.0 ±3.8	89.0 ±5.3	84.0 ±4.3	**90.0 ±2.8**	76.0 ±6.7	86.0 ±5.2

5.3 Temporal Correction Beyond Recursive Sharing

To test correction without sharing, Table 4 applies the same fixed gate to Official. Mean success rises on all four tasks, most on TwoRooms and Reacher. Specifically, Official improves from 81.0% to 91.5% on TwoRooms and from 79.0% to 89.0% on Reacher. These gains arise with the pretrained weights fixed, showing that correction can improve candidate evaluation without changing the learned representation. The shared pattern of gains in Official and Loop shows that the same gate improves planning in both untied and recursively shared models.

Correction During Planning and Rollouts. In five-step CEM planning, the gate first acts on the terminal prediction. It changes the goal cost used to rank candidate action sequences, allowing correction to improve action selection with model weights fixed. Longer rollouts add a second role: the corrected state becomes an input to later predictions. We therefore examine the latent steps propagated through these continued rollouts.

Table 11 in Appendix F.1 reports lower 95th-percentile step magnitudes on all four tasks. For Loop, the value falls from 16.0 to 12.4 on Reacher and from 14.4 to 9.0 on Cube. Figure 2(b) visualizes these realized step magnitudes. Cube has the largest step reduction but a smaller planning gain than TwoRooms and Reacher: step norms quantify local motion, whereas CEM evaluates distance to the goal. The two measurements therefore assess different effects of correction. More diagnostics and state-correction versus cost-penalty controls are available in Appendices F.1 and F.2, respectively.

6 Conclusion

In this paper, we propose Loop LeWorldModel, a post-training framework for iterative latent processing in visual JEPA world models. Its components address depth reuse and temporal correction. The Loop Encoder/Predictor combines frozen shared bases with depth-specific adapters to preserve encoder–predictor compatibility while retaining the original effective depths with fewer stored parameters. The training-free Self-Correction Gate uses recent motion statistics to correct atypical latent transitions, refining terminal goal costs during planning and limiting large steps during longer rollouts. Experiments across visual-control tasks demonstrate improved mean planning success over the pretrained model with substantially fewer stored parameters. Ablations show that Loop training and retained effective depth improve planning over conversion-only and shallow controls, and that the same gate also improves untied predictors. Although these results are promising, our study considers fixed-depth schedules and a limited set of tasks. In the future, we will explore adaptive computation depth and broader transfer of temporal correction.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for World Modeling: Visual Details Matter in Atari. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6bdde0373d53d4a501249547084bed43-Abstract-Conference.html.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images With a Joint-Embedding Predictive Architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Assran_Self-Supervised_Learning_From_Images_With_a_Joint-Embedding_Predictive_Architecture_CVPR_2023_paper.html.

Mahmoud 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. *arXiv preprint arXiv:2506.09985*, 2025. doi: 10.48550/arXiv.2506.09985. URL https://arxiv.org/abs/2506.09985.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed Recursive Transformers: Effective Parameter Sharing with Layer-wise LoRA. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WwpYSOkkCt.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and Scalable Self-Supervised Learning Without the Heuristics. *arXiv preprint arXiv:2511.08544*, 2025. doi: 10.48550/arXiv.2511.08544. URL https://arxiv.org/abs/2511.08544.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting Feature Prediction for Learning Visual Representations from Video. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2. Featured Certification.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative Interactive Environments. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021. doi: 10.1109/ICCV48922.2021.00951. URL https://openaccess.thecvf.com/content/ICCV2021/html/Caron_Emerging_Properties_in_Self-Supervised_Vision_Transformers_ICCV_2021_paper.html.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134(1):19–67, February 2005. ISSN 1572-9338. doi: 10.1007/s10479-005-5724-z. URL https://doi.org/10.1007/s10479-005-5724-z.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped Transformers as Programmable Computers. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

David Ha and Jürgen Schmidhuber. World Models. *arXiv preprint arXiv:1803.10122*, 2018. doi: 10.48550/arXiv.1803.10122. URL https://arxiv.org/abs/1803.10122.

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.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Control Tasks through World Models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal Difference Learning for Model Predictive Control. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022. doi: 10.1109/CVPR52688.2022.01553. URL https://openaccess.thecvf.com/content/CVPR2022/html/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Hongyuan Adam Lu, Z. L. Victor Wei, Qun Zhang, Jinrui Zeng, Bowen Cao, Lingwei Meng, Mocheng Li, Zezhong Wang, Haonan Yin, Naifu Xue, Minyu Chen, Cenyuan Zhang, Zefan Zhang, Hao Wei, Jiawei Zhou, Haoran Xu, Hao Yang, Ronglai Zuo, Tongda Xu, Yonghao Li, Jian Chen, Hebin Wang, Zeyu Gao, Yang Li, Wei Zhao, Qimin Zhong, Siqi Liu, Yumeng Zhang, Leyan Cui, Zhangyu Wang, and Wai Lam. Looped World Models. *arXiv preprint arXiv:2606.18208*, 2026. doi: 10.48550/arXiv.2606.18208. URL https://arxiv.org/abs/2606.18208.

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. doi: 10.48550/arXiv.2603.19312. URL https://arxiv.org/abs/2603.19312.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/db36f4d603cc9e3a2a5e10b93e6428f2-Abstract-Conference.html.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers Are Sample-Efficient World Models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research*, 2024. 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.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, October 2023.

Lorenzo Posani, Shuqi Wang, Samuel P. Muscinelli, Liam Paninski, and Stefano Fusi. Rarely Categorical, Highly Separable Representations along the Cortical Hierarchy. *Nature*, 2026. doi: 10.1038/s41586-026-10668-4. URL https://www.nature.com/articles/s41586-026-10668-4.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based World Models Are Happy With 100k Interactions. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TdBaDGCpjly.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with Latent Thoughts: On the Power of Looped Transformers. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2676109d49d1eb26d6bc584a8f556305-Paper-Conference.pdf.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588(7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4. URL https://www.nature.com/articles/s41586-020-03051-4.

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.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to Explore via Self-Supervised World Models. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592, 2020. URL https://proceedings.mlr.press/v119/sekar20a.html.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement Learning with Action-Free Pre-Training from Videos. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19561–19579, 2022. URL https://proceedings.mlr.press/v162/seo22a.html.

Erik Talvitie. Self-Correcting Models for Model-Based Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, volume 31, pp. 2597–2603, 2017. doi: 10.1609/aaai.v31i1.10850. URL https://ojs.aaai.org/index.php/AAAI/article/view/10850.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/416f9cb3276121c42eebb86352a4354a-Abstract-Conference.html.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped Transformers Are Better at Learning Learning Algorithms. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b8402301e7f06bdc97a31bfaa653dc32-Abstract-Conference.html.

Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning Interactive Real-World Simulators. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=sFyTZEqmUY.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari Games with Limited Data. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d5eca8dc3820cad9fe56a3bafda65ca1-Abstract.html. EfficientZero.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=lq62uWRJjiY.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: Efficient Stochastic Transformer based World Models for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://openreview.net/forum?id=WxnrX42rnS.

Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.