NESTEDWM: WORLD MODELS WITH NESTED LATENT REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with world models requires repeated rollouts, yet conventional models are trained with a fixed latent size that determines the trade-off between representation capacity and planning cost. We propose NestedWM, which learns nested latent representations that allow a single world model to predict and plan using leading prefixes of its predictive state at different budgets. Each prefix contains either the leading dimensions of a latent vector or the leading tokens of an ordered latent-token sequence. We jointly train an encoder and an action-conditioned predictor over sampled prefix budgets, using each selected prefix as both the predictor input and the target prediction space. In a dimension-wise study with the architecture held fixed, a single encoder and predictor support planning across budgets, achieving high success with only a fraction of the full latent dimensions in most evaluated visual-control and synthetic tasks. Further analyses show that leading prefixes preferentially retain planning-relevant information, while predictable distractors can consume limited representation budget. We then turn this capability into computational savings with a token-wise instantiation that processes only the selected prefix tokens during rollout. In controlled multi-token settings, a single model supports successful planning across multiple token budgets, with smaller prefixes reducing model-based cost-evaluation time.

1 INTRODUCTION

World models enable agents to plan by predicting the consequences of actions before executing them (Hafner et al., 2019; Hansen et al., 2024; Zhou et al., 2025). A planner repeatedly rolls the model forward along candidate action sequences and evaluates the predicted trajectories to choose actions. We focus on latent world models, which perform these rollouts in latent space without generating future images (Hafner et al., 2019). A line of work jointly trains the encoder and action-conditioned predictor, allowing the latent state to focus on predictive factors useful for planning rather than irrelevant visual detail. In this setting, shaping the latent representation space is a central challenge, as it influences both planning performance and rollout computation cost.

The size of these representations, however, is typically fixed when training the encoder and predictor, limiting the ability to adjust it to task demands or available computational resources. Adjusting it after training requires more than truncating the latent state of a model trained to use its full representation: information needed for prediction may be distributed across the representation, while the predictor may depend on components removed by truncation.

We therefore propose NestedWM, a world model whose rollout budget can be adjusted after training to trade off planning performance and computational cost. NestedWM learns nested latent representations in which smaller states are prefixes—the first few dimensions or tokens—of larger ones. Motivated by prior work on nested representations (Rippel et al., 2014; Kusupati et al., 2022; Bachmann et al., 2025), we sample a prefix budget at every training step and restrict both the predictor input and its prediction target to the selected prefix. This training scheme offers two benefits. First, training across all prefix budgets makes the rollout budget controllable: a single encoder and action-conditioned predictor operate at every budget, so the budget can be chosen at planning time without training a separate model for each. Second, the nested structure concentrates predictive information in the leading prefixes, since earlier positions are included in every larger budget and are trained more often; this keeps even small budgets informative enough for planning.

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.

We first examine whether small prefixes support planning and what information they retain by training with nested dropout over dimension-wise prefixes, keeping the encoder and predictor architecture of LeWorldModel (LeWM) (Maes et al., 2026) unchanged. A single trained encoder and predictor support planning across multiple prefix budgets. In most evaluated environments, high planning success is achieved using only a fraction of the full latent dimensions. Further analyses provide evidence that the learned representations concentrate information useful for dynamics prediction and planning in their leading prefixes. Distractor experiments show, however, that predictable but task-irrelevant factors can also occupy limited prefix budgets and reduce planning success at small budgets.

We then translate this capability into savings in rollout computation by applying nested dropout over token-wise prefixes, guided by a frozen full-budget teacher. In this setting, fewer prefix tokens directly reduce rollout computation, since the predictor processes only the selected tokens. At matched budgets, a single NestedWM model achieves comparable planning success than LeWM-based models trained separately for each reduced budget. This indicates that NestedWM can effectively control the rollout budget in practice while largely preserving planning performance.

Our contributions are threefold:

- We propose NestedWM, which jointly learns nested latent representations and prefix dynamics, enabling a single world model to predict and plan across representation budgets.
- In a controlled dimension-wise study, we show that small prefixes support planning and preferentially retain planning-relevant information.
- Our token-wise extension provides multiple cost–performance operating points within a single model, supporting successful planning with fewer tokens and shorter model-based cost evaluation times in a controlled multi-object task.

2 Related Work

Latent world models for planning. Latent world models differ in how they learn representations and use predicted trajectories for control. PlaNet learns dynamics with observation reconstruction and reward prediction, while TD-MPC2 combines latent prediction with reward and value learning without reconstructing observations (Hafner et al., 2019; Hansen et al., 2024). For planning toward goal observations, DINO-WM learns action-conditioned dynamics over frozen pretrained visual features and evaluates predicted trajectories in that feature space (Zhou et al., 2025). V-JEPA 2-AC learns an action-conditioned predictor over a frozen pretrained video encoder for image-goal robot planning (Assran et al., 2025). CompACT learns latent dynamics for planning over tokens from a separately trained compact tokenizer (Kim et al., 2026). PLDM and LeWorldModel instead jointly learn representations and latent dynamics for goal-directed planning without image reconstruction (Sobal et al., 2025; Maes et al., 2026). LeWorldModel's prediction-and-regularization objective provides the baseline for our dimension-wise study. We extend this joint learning setting to train the same encoder and predictor across multiple nested prefix budgets.

Nested representations. Nested representation learning makes prefixes of different sizes useful within a single representation. Nested dropout learns ordered representations by sampling a cutoff and excluding subsequent units during training, with applications to reconstruction and retrieval (Rippel et al., 2014). Matryoshka representation learning jointly optimizes objectives at several prefix sizes, enabling embeddings to be used at different dimensionalities for tasks such as classification and retrieval (Kusupati et al., 2022). One-D-Piece and ElasticTok use sampled token prefixes to support variable-length visual encodings for reconstruction (Miwa et al., 2025; Yan et al., 2025). FlexTok applies nested dropout and causal attention to learn variable-length discrete image tokens, using a rectified-flow decoder for reconstruction in a VAE latent space and autoregressive models for image generation (Bachmann et al., 2025). We use prefix sampling for action-conditioned prediction of the next latent state at the same budget, so that each prefix can support repeated rollout. Whereas FlexTok reconstructs the current image's VAE latents, our token-wise construction uses a frozen teacher's full future latent state as an auxiliary target.

3 METHOD

We first review joint encoder–predictor learning in latent world models and introduce our shared principle of predicting and planning within latent prefixes. We then present two instantiations: dimension-wise prefix learning with the architecture held fixed, and token-wise prefix learning that reduces rollout computation by processing fewer tokens.

3.1 PRELIMINARIES

Joint-embedding predictive architectures (JEPAs) learn to predict representations of a target from an observed context, rather than reconstructing the target in input space (Assran et al., 2023). For action-conditioned world models, the context consists of past observations and actions, and the target is a future observation encoded in latent space. We consider an offline dataset of visual observation sequences with their corresponding actions. With an encoder enc_θ and an action-conditioned predictor pred_ϕ, the prediction takes the form

$$\mathbf{z}_t = \mathrm{enc}_\theta(o_t), \qquad \widehat{\mathbf{z}}_{t+1} = \mathrm{pred}_\phi(\mathbf{z}_{t-h+1:t}, a_{t-h+1:t}), \tag{1}$$

where h is the context length and a_t denotes the action taking the environment from time t to $t+1$. The predictor is trained against the encoded future observation, so learning dynamics does not require pixel reconstruction or factor annotations. Our focus is on jointly learning the encoder and predictor, allowing the predictive objective to shape the representation itself.

Joint learning also admits a degenerate solution: a constant encoder output can be predicted perfectly while conveying no state information. Approaches to avoiding collapse include freezing a pretrained encoder (Zhou et al., 2025), updating a target encoder by an exponential moving average (EMA) with stop-gradient through its outputs (Grill et al., 2020; Assran et al., 2023), and explicitly regularizing the embedding distribution (Bardes et al., 2022; Balestriero & LeCun, 2025).

We adopt LeWM (Maes et al., 2026) as our baseline for joint encoder–predictor learning because of its simple objective, few objective-specific hyperparameters, and reported planning performance and planning-time efficiency across navigation and manipulation benchmarks. Its training objective combines latent prediction error with Sketched Isotropic Gaussian Regularization (SIGReg) (Balestriero & LeCun, 2025):

$$\mathcal{L}_{\mathrm{LeWM}} = \mathcal{L}_{\mathrm{pred}} + \lambda \mathcal{L}_{\mathrm{SIGReg}}, \tag{2}$$

where $\mathcal{L}_{\mathrm{pred}}$ is the mean squared error between predicted and target latent states, and λ weights the regularizer. SIGReg serves as an anti-collapse mechanism by encouraging the embedding distribution to match an isotropic Gaussian target. To avoid directly comparing high-dimensional distributions, it projects embeddings onto random unit-norm directions and evaluates each one-dimensional sample using the Epps–Pulley statistic (Balestriero & LeCun, 2025; Maes et al., 2026).

3.2 PREDICTING AND PLANNING WITH LATENT PREFIXES

Latent prefixes. Our goal is to use the same encoder and rollout predictor across multiple latent representation budgets, each defined by the size of the retained prefix. In the dimension-wise construction, the encoder maps each observation o_t to a single latent vector $\mathbf{z}_t \in \mathbb{R}^D$. Retaining its first d feature dimensions forms a *dimension-wise prefix*, denoted by $\mathbf{z}_t^{(d)} \in \mathbb{R}^d$. In the token-wise construction, the encoder maps each observation o_t to K latent tokens of width D, forming a state $\mathbf{Z}_t \in \mathbb{R}^{K \times D}$. Retaining the first k tokens forms a *token-wise prefix*, denoted by $\mathbf{Z}_t^{(k)} \in \mathbb{R}^{k \times D}$.

Training. In each construction, we train these prefixes using *nested dropout* (ND) (Rippel et al., 2014), which samples a truncation point and excludes the suffix from the prediction input. At each training step, we draw one prefix length uniformly from all available lengths, from one coordinate or token to the full representation, and use it for the entire minibatch across time steps. The encoder and predictor are trained to predict the next-state prefix from the sampled input prefix and actions. We call this same-budget prediction relation *prefix-to-prefix* (PTP). Because retaining a position always retains all preceding positions, earlier coordinates or tokens are included more frequently. Earlier positions thus participate in more prefix predictions, encouraging information needed at small budgets to be encoded early while training the predictor to operate across nested budgets. Figure 1(a) illustrates this training structure.

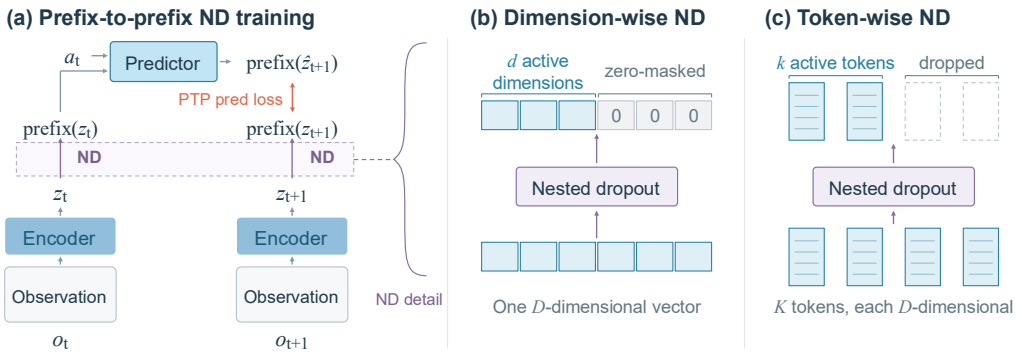


Figure 1: **Nested-prefix prediction and its two instantiations.** (a) Nested dropout (ND) samples a shared prefix budget for the latent states encoded from o_t and o_{t+1}, training the prefix-to-prefix (PTP) predictor to predict the next prefix from the current prefix and action a_t. The h-step temporal history is omitted for visualization. (b) Dimension-wise prefixes zero-mask suffix coordinates, keeping the predictor width fixed. (c) Token-wise prefixes drop suffix tokens entirely, so the PTP predictor processes only the retained tokens.

Planning. At planning time, we choose the prefix budget and apply the PTP predictor recursively to roll out candidate action sequences. We score each rollout against the goal prefix at that budget. We follow LeWM's model predictive control (MPC) procedure (Maes et al., 2026), using the cross-entropy method (CEM) to optimize actions with the model fixed and replanning after the prescribed execution interval.

3.3 Dimension-wise Prefix Learning

We use dimension-wise prefixes to study how predictive information and planning performance change with the coordinate budget while retaining LeWM's encoder–predictor architecture with minimal changes. The LeWM predictor expects a fixed-width latent vector and cannot directly accept a shorter prefix. We therefore zero-mask the suffix rather than truncate the vector, preserving the D-dimensional input shape while exposing only the selected prefix.

Prefix-to-prefix prediction. We train the encoder and predictor to predict the next coordinate prefix from the masked latent history and actions. For each minibatch, we sample d, predict the next prefix, and compute the PTP loss as the mean squared error over that prefix:

$$d \sim \mathrm{Uniform}\{1, \dots, D\}, \qquad \widetilde{\mathbf{z}}_t^{(d)} = \begin{bmatrix} (\mathbf{z}_t)_{1:d} \\ \mathbf{0}_{D-d} \end{bmatrix}, \tag{3}$$

$$\widehat{\mathbf{z}}_{t+1}^{(d)} = \left[\mathrm{pred}_\phi\big(\widetilde{\mathbf{z}}_{t-h+1:t}^{(d)}, a_{t-h+1:t}\big) \right]_{1:d}, \tag{4}$$

$$\mathcal{L}_{\mathrm{PTP}}^{(d)} = \mathbb{E}\left[\frac{1}{d} \left\| \widehat{\mathbf{z}}_{t+1}^{(d)} - \mathbf{z}_{t+1}^{(d)} \right\|_2^2 \right], \tag{5}$$

where the expectation is estimated by averaging over training sequences and supervised time steps in the minibatch. We jointly optimize the encoder and predictor, with encoder gradients through both the masked inputs and encoded targets. During rollout, each predicted prefix is zero-padded back to width D before the next prediction.

Training objective. We apply SIGReg to the sampled prefix rather than the full vector, aligning the scope of regularization with that of prediction. Full-vector SIGReg constrains only the distribution of the complete dimensional state and therefore does not prevent the leading coordinates from collapsing to low variance, which we have observed to hurt small-prefix planning. For the sampled budget d, the training objective is

$$\mathcal{L}_{\mathrm{dim}}^{(d)} = \mathcal{L}_{\mathrm{PTP}}^{(d)} + \lambda \mathcal{L}_{\mathrm{SIGReg}}^{(d)}, \tag{6}$$

where $\mathcal{L}_{\mathrm{SIGReg}}^{(d)}$ applies SIGReg to the unpadded d-dimensional prefixes in the minibatch.

3.4 TOKEN-WISE PREFIX LEARNING

We use token-wise prefixes to reduce rollout computation by letting the predictor process fewer latent tokens at smaller budgets. Unlike the dimension-wise case, the predictor can naturally accommodate different token counts while keeping each token's feature width fixed. To produce multiple latent tokens, we augment the encoder with a lightweight cross-attention resampler and adapt the predictor to process token sequences (See details in Appendix B).

Prefix-to-prefix prediction. We train the encoder and predictor to predict the next token prefix from the selected prefix history and actions. For each minibatch, we sample a token budget, predict the next prefix from only the selected tokens, and compute the PTP loss against the next prefix encoded by the same model:

$$k \sim \mathrm{Uniform}\{1, \dots, K\}, \qquad \mathbf{Z}_t^{(k)} = (\mathbf{Z}_t)_{1:k}, \tag{7}$$

$$\widehat{\mathbf{Z}}_{t+1}^{(k)} = \mathrm{pred}_\phi\big(\mathbf{Z}_{t-h+1:t}^{(k)}, a_{t-h+1:t}\big), \tag{8}$$

$$\mathcal{L}_{\mathrm{PTP}}^{(k)} = \mathbb{E}\left[\frac{1}{kD}\left\|\widehat{\mathbf{Z}}_{t+1}^{(k)} - \mathbf{Z}_{t+1}^{(k)}\right\|_F^2\right], \tag{9}$$

where the expectation is estimated over minibatch sequences and supervised time steps as in Section 3.3.

Prefix-to-full prediction. To retain the predictive information learned by a full-budget model while concentrating it in the leading prefixes, we add a *prefix-to-full* (PTF) objective. We initialize the student from a pretrained LeWM-based multi-token model trained without nested dropout, with verified full-budget planning performance. A frozen copy of the pretrained encoder serves as the teacher $\mathrm{enc}_{\bar{\theta}}$, supplying the same full next-state target $\overline{\mathbf{Z}}_{t+1} = \mathrm{enc}_{\bar{\theta}}(o_{t+1}) \in \mathbb{R}^{K\times D}$ across all budgets. A separate predictor $\mathrm{pred}_\psi^{\mathrm{full}}$, also initialized from the pretrained predictor, maps the student prefix and actions to this teacher target:

$$\widehat{\mathbf{Z}}_{t+1}^{\mathrm{full}} = \mathrm{pred}_\psi^{\mathrm{full}}\big(\mathbf{Z}_{t-h+1:t}^{(k)}, a_{t-h+1:t}\big), \tag{10}$$

$$\mathcal{L}_{\mathrm{PTF}}^{(k)} = \mathbb{E}\left[\frac{1}{KD}\left\|\widehat{\mathbf{Z}}_{t+1}^{\mathrm{full}} - \overline{\mathbf{Z}}_{t+1}\right\|_F^2\right]. \tag{11}$$

The teacher and PTF predictor are used only during training; planning uses only the student encoder and PTP predictor.

Training objective. The token-wise objective combines equally weighted PTP and PTF losses with a SIGReg term:

$$\mathcal{L}_{\mathrm{token}}^{(k)} = \mathcal{L}_{\mathrm{PTP}}^{(k)} + \mathcal{L}_{\mathrm{PTF}}^{(k)} + \lambda\mathcal{L}_{\mathrm{SIGReg}}^{(k)}. \tag{12}$$

SIGReg is applied to the concatenated kD-dimensional prefix. Both predictive losses update the student encoder and shared action encoder through the input prefix, along with their predictors pred_ϕ and $\mathrm{pred}_\psi^{\mathrm{full}}$. We stop gradients through the PTP target prefix and keep the teacher fixed.

4 EXPERIMENTS

We first use dimension-wise prefixes to evaluate planning performance and analyze the information contained in different representation budgets (Sections 4.2–4.5). We then evaluate token-wise prefixes to examine whether smaller budgets reduce rollout computation while retaining planning performance (Section 4.6).

4.1 EXPERIMENTAL SETUP

Environments. We evaluate on established visual planning benchmarks and controlled Dot environments. Following LeWM (Maes et al., 2026), we evaluate on PushT, TwoRoom, and the DeepMind Control Suite's Reacher task (Tassa et al., 2018). We also use Dot1, Dot2, and Dot3,

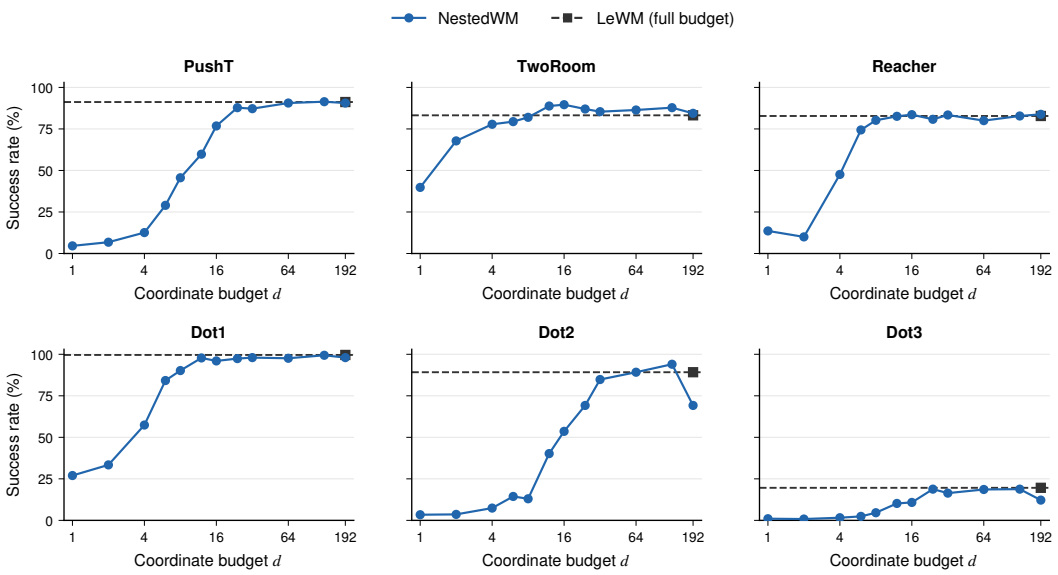


Figure 2: **Planning with dimension-wise prefixes.** Each blue curve uses a single trained NestedWM encoder and predictor across coordinate budgets d. The dashed line and square show LeWM evaluated at its full budget, $D = 192$.

controlled visual environments with one, two, and three dots, respectively, where success requires moving all dots to their target configuration. Figure 6 in the appendix shows example observations. We introduce distractor variants of the Dot environment in Section 4.5 to examine the effects of task-irrelevant visual factors. We exclude OGBench Cube from the main comparison because sampled initial states can already satisfy its success criterion, complicating the interpretation of planning success; Appendix A.3 documents this concern and retains the results.

Planning evaluation. We measure planning performance by the success rate: the fraction of trials that meet the environment's success criterion within the allotted action budget. We follow LeWM's MPC/CEM evaluation procedure (Maes et al., 2026), computing the goal cost in the selected prefix space. For the main experiments, we evaluate each setting on 500 sampled initial-state–goal-state pairs to reduce sampling variability relative to using 50 pairs. All token-wise conditions use the same evaluation pairs. Additional dimension-wise protocol details are given in Appendix A.1.

Models and training. For dimension-wise experiments, we compare NestedWM with LeWM, the baseline trained without nested dropout, using the same per-environment configuration for architecture, data, training epochs, and other training hyperparameters. Both models are trained on offline observation–action sequences, with each observation encoded as a single vector of dimension $D = 192$. For token-wise experiments, we extend LeWM to support multiple latent tokens and train token-wise NestedWM as described in Section 3.4.

4.2 Planning with Dimension-wise Prefixes

In most evaluated environments, NestedWM achieves high planning success rates using only a fraction of the full dimension budget. Figure 2 varies the prefix budget during planning while keeping a single trained encoder and predictor fixed within each environment. We evaluate LeWM at full budget for reference. Since its predictor is trained only on full representations, evaluating it on small prefixes would confound prefix quality with the predictor's lack of training on those inputs.

A high success-rate plateau below the full budget suggests that a smaller prefix retains state information and supports learned dynamics sufficient for planning in that setting. Dot3 is an exception: LeWM itself has a low success rate, and NestedWM reaches a similarly low maximum across the

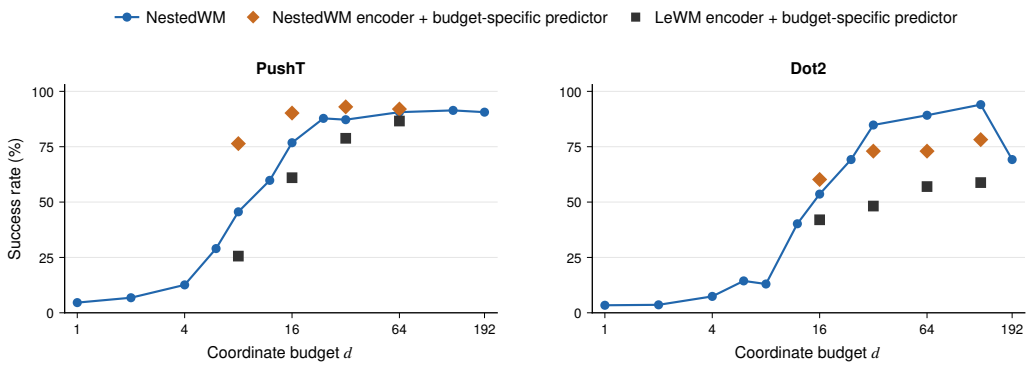


Figure 3: **Prefix planning with shared and budget-specific predictors.** The blue curve shows NestedWM with its jointly trained encoder and predictor shared across budgets. Orange diamonds and black squares show results from frozen NestedWM and LeWM encoders, respectively, each paired with a separate predictor trained for that budget under matched settings.

evaluated budgets. Appendix A.2 reports the available 50-trial benchmark results for comparison with LeWorldModel's evaluation size, and Appendix A.3 reports the excluded Cube results.

4.3 Planning Utility of Prefix Representations

Representation advantage. The orange and black markers in Figure 3 compare the prefix representations of frozen NestedWM and LeWM encoders, each paired with a randomly initialized, budget-specific predictor trained under identical settings (Appendix A.4). NestedWM prefixes yield higher planning success at all four evaluated budgets in both PushT and Dot2. Because the predictors are matched, this gap reflects what the leading coordinates contain rather than how the predictor is trained. LeWM spreads state information across all D coordinates, so truncation discards information that a dedicated predictor cannot recover, whereas NestedWM concentrates planning-relevant information in its leading prefix.

Shared-predictor model. The blue curve and black markers compare the two models at matched planning budgets: at each budget, the original NestedWM uses its single predictor shared across all budgets, whereas LeWM uses a predictor trained specifically for that budget. A single NestedWM model outperforms these budget-specialized LeWM predictors at every evaluated budget, showing that the representation advantage carries over to the model used for planning. This indicates that nested training enables a single world model to plan effectively across budgets.

4.4 Task States in Prefix Representations

We probe oracle task states to examine what information small prefixes retain: agent position, block position, and block angle in PushT, and the two dot positions in Dot2. Motivated by the use of latent distances in planning, we use k-nearest-neighbor (kNN) probing to measure state recovery through R^2, with protocol details provided in Appendix A.5.

At low budgets, NestedWM prefixes generally recover task states more accurately than random coordinate subsets of the same size, as shown by the solid and dashed blue lines in Figure 4. This advantage indicates that recoverable task information is concentrated in the leading coordinates, rather than being equally accessible from arbitrary coordinate selections. NestedWM prefixes also generally outperform LeWM prefixes at matched budgets, as shown by the blue and black solid lines. Together with the matched-predictor results in Section 4.3, this shows that the prefix advantage is reflected in both task-state recovery and planning.

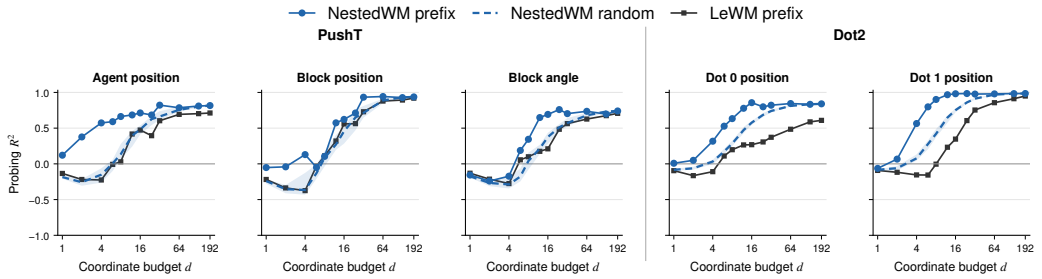


Figure 4: **Task-factor recovery with kNN probing.** Blue and black denote NestedWM and LeWM representations, respectively. Solid lines show prefix readout; the blue dashed line and shaded band show the median and interquartile range across 32 random coordinate permutations.

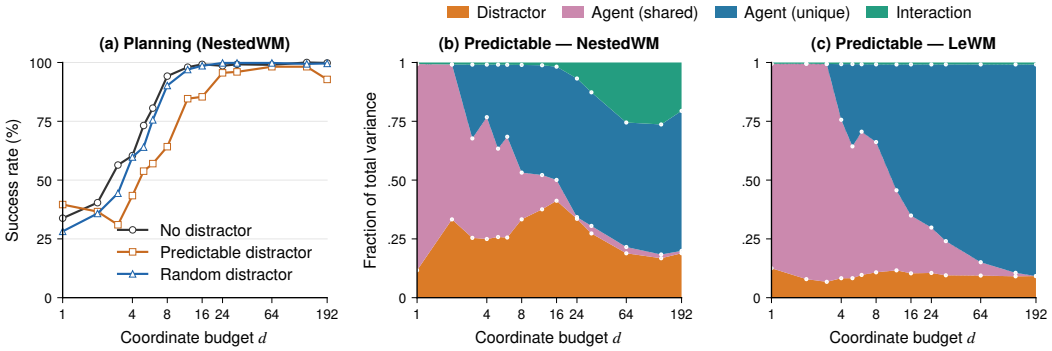


Figure 5: **Planning and variance allocation with distractors.** (a) Planning success of NestedWM in Clean, Predictable, and Random, using 500 initial-state/goal-state pairs per condition. (b–c) Variance allocation within each prefix of NestedWM and LeWM in Predictable. Agent (shared) and Agent (unique) denote agent variance along and orthogonal to the distractor's dominant variation subspace, respectively.

4.5 Distractor Effects on Prefix Representations

Learning to concentrate predictive information in leading prefixes may also retain temporally predictable factors that are irrelevant to control (Wang et al., 2022). We therefore examine how such factors occupy limited prefix budgets and affect planning. We add a predictable or randomly moving distractor to the Dot1 environment, used here as the Clean reference, while requiring only the agent to reach its goal. Figure 9 in the appendix illustrates the two distractor conditions. The predictable distractor substantially reduces low-budget planning success, whereas the random distractor produces little degradation relative to Clean (Figure 5a).

Variance allocation. Since SIGReg encourages isotropic representation variance, we use ANOVA to examine how this variance is distributed among the agent, distractor, and their interaction (Appendix A.6). In NestedWM, the predictable distractor's share of representation variance initially increases with the prefix budget, peaks at $d = 16$, and then decreases, whereas its share in LeWM remains roughly constant (Figure 5b–c). As the distractor's share decreases, planning approaches a plateau around $d = 24$. By contrast, in Random, more than 99.99% of the representation variance is attributable to the agent.

Subspace overlap. Agent variation also becomes more separated from the distractor's dominant variation directions as the NestedWM prefix grows. We measure agent variance along these directions, termed A-shared, which falls below 1% of total representation variance near the start of the planning plateau. This separation may reduce interference between task-relevant and irrelevant factors in the latent-distance comparisons used for planning.

Table 1: **Planning with fewer rollout tokens in Dot3.** LeWM-based models are trained separately for each token count, whereas NestedWM uses prefixes of a single model with $K = 3$ and $D = 192$. Reported times are mean per-call times for NestedWM on an A6000, including observation/goal encoding, predictor rollout, and latent cost evaluation.

Token count	LeWM-based SR (%)	NestedWM SR (%)	Cost evaluation (ms)
1	4.0	69.0	142.5
2	1.2	89.4	191.6
3	86.8	84.0	237.8

Table 2: **Planning with fewer rollout tokens in PushT.** LeWM-based models are trained separately for each token count, whereas NestedWM uses prefixes of a single model with $K = 24$ and $D = 8$. Reported times are mean per-call times for NestedWM on an A6000, including observation/goal encoding, predictor rollout, and latent cost evaluation.

Token count	LeWM-based SR (%)	NestedWM SR (%)	Cost evaluation (ms)
1	45.6	69.0	33.3
2	74.4	75.6	41.5
8	81.6	87.2	86.2
24	88.0	85.0	234.9

4.6 Planning with Token-wise Prefixes

We evaluate whether token-wise NestedWM can reduce rollout computation while retaining the planning capabilities of a multi-token model. We return to Dot3, where both LeWM and dimension-wise NestedWM showed low planning success in Section 4.2. A LeWM-based model with three 192-dimensional tokens achieves high success in this environment, motivating us to examine whether its planning capabilities can be retained in smaller token prefixes. For PushT, we use 24 eight-dimensional tokens, keeping the full latent dimensionality at 192 while examining a wider range of token budgets. In both settings, rollouts process only the selected prefix tokens, directly reducing predictor computation.

In Dot3, NestedWM achieves substantially higher planning success with one- and two-token prefixes than separately trained models with the corresponding token counts (Table 1). Its two-token prefix exceeds the model's observed full-budget success rate while reducing mean model-based cost evaluation time by approximately 19%. Probing further shows that small prefixes retain information about all three dots, with additional tokens improving readout accuracy (Appendix B.2).

On PushT, a single NestedWM model offers smaller prefixes that preserve high planning success and faster operating points that trade success rate for further computational savings (Table 2). The eight-token prefix achieves 87.2% planning success while reducing mean model-based cost evaluation time by approximately 63% relative to using all 24 tokens. One- and two-token prefixes provide faster operating points with lower success rates.

5 Conclusion

We have presented NestedWM, which jointly learns nested latent representations and action-conditioned dynamics so that a single world model can predict and plan across multiple prefix budgets. Our dimension-wise study shows that small prefixes support planning and concentrate planning-relevant information, while predictable distractors can occupy limited representation budget. In Dot3 and PushT, our token-wise instantiation translates smaller representation budgets into reduced rollout computation, supporting successful planning with shorter model-based cost evaluation times. Together, these results demonstrate multiple cost–performance operating points within a single trained world model.

AI USE STATEMENT

We used generative AI tools to assist with implement research methods and support translation. Other required-disclosure activities were not applicable. There were no additional uses to disclose under the recommended-disclosure categories. The authors reviewed the AI-assisted outputs through manual code review, code execution and testing, and checks of translation accuracy. Additional verification included confirming the existence and bibliographic details of cited references, checking literature summaries and related-work descriptions against original sources, cross-checking reported results and their interpretations against experimental outputs, reviewing mathematical content, and checking figures against the methods, implementation, and source materials. The authors accept responsibility for the complete paper and its accompanying artifacts.

ETHICS STATEMENT

This work is intended to support research on latent world models and model-based planning. Our experiments use simulated control environments and do not establish that the method is safe or reliable for deployment in physical systems. Any real-world deployment would require independent validation and appropriate safety assessment.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the model and training objectives in Section 3, the experimental settings and evaluation protocols in Section 4 and Appendix A, and the token-wise architecture and training details in Appendix B. Upon publication, we will release the code and experiment configurations needed to reproduce our results.

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 *CVPR*, pp. 15619–15629, 2023.

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.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. FlexTok: Resampling images into 1D token sequences of flexible length. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2241–2292. PMLR, 2025.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

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

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

Dongwon Kim, Gawon Seo, Jinsung Lee, Minsu Cho, and Suha Kwak. Planning in 8 tokens: A compact discrete tokenizer for latent world model. In *CVPR*, 2026.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *NeurIPS*, volume 35, pp. 30233–30249, 2022.

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.

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-D-Piece: Image tokenizer meets quality-controllable compression. In *ICML Workshop on Tokenization (TokShop)*, 2025.

Art B. Owen. Variance components and generalized Sobol' indices. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):19–41, 2013.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *ICLR*, 2025.

Oren Rippel, Michael Gelbart, and Ryan P. Adams. Learning ordered representations with nested dropout. In *ICML*, 2014.

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 *NeurIPS*, volume 38, 2025. doi: 10.52202/085713-1465.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tongzhou Wang, Simon S. Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22591–22612. PMLR, 2022.

Wilson Yan, Vlad Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. ElasticTok: Adaptive tokenization for image and video. In *ICLR*, 2025.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025.

A Additional Dimension-wise Evaluation Results

A.1 Evaluation Details

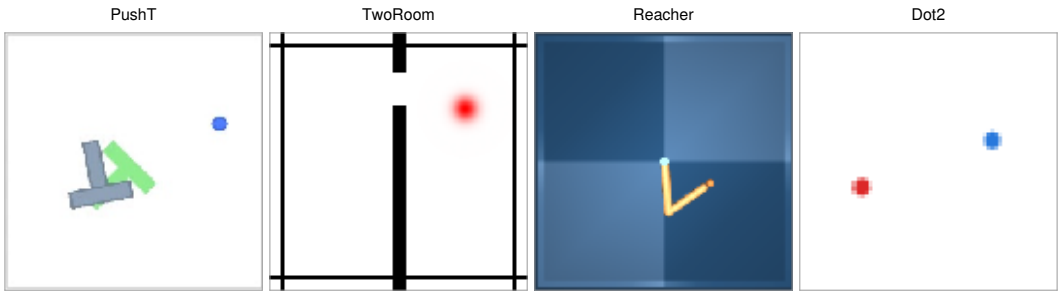


Figure 6: **Environments used for evaluation.** From left to right: PushT, a 2D manipulation task in which an agent pushes a T-shaped block to a target pose; TwoRoom, a 2D navigation task in which an agent moves between rooms to reach a target position; Reacher, a planar control task in which a two-joint arm reaches a target configuration; and Dot2, a controlled environment in which both dots must be moved to their respective target positions.

The following settings apply to the dimension-wise experiments. The LeWM and NestedWM models are trained for 20 epochs on PushT, 100 epochs on TwoRoom and Reacher, and 50 epochs on Dot1, Dot2, and Dot3. All use $D = 192$ and a history of three observations. PushT, TwoRoom, and Reacher use a ViT encoder, while the Dot environments use a CNN encoder. For PushT, TwoRoom, and Reacher, we follow the evaluation-clip sampling procedure in LeWM's released code (Maes et al., 2026). Dot evaluations use separately generated validation datasets. The 500 planning trials measure performance for each trained checkpoint rather than variability across independently trained models. The dimension-wise comparisons use training seed 3072 and evaluation seed 42.

CEM uses 300 candidates, 30 elites, and 30 iterations. The prediction horizon, replanning interval, and action block size are 5, 5, and 5 for the benchmarks and 25, 25, and 1 for the Dot environments. Goals are sampled at an offset of 25 environment steps, with an action budget of 50. For PushT, TwoRoom, and Reacher, we use the success criteria implemented in LeWM's evaluation code (Maes et al., 2026). Success requires agent and block positions within a joint distance of 20 and block angle within $\pi/9$ for PushT, agent position within 16 of the target for TwoRoom, and all joint angles within 0.05 radians for Reacher. Dot success is defined by the joint Euclidean distance between all dot coordinates and their targets being at most $0.08\sqrt{n}$, where n is the number of dots.

A.2 Results with 50 Planning Trials

We use 500 initial-state–goal-state pairs rather than the 50 used by LeWorldModel (Maes et al., 2026) to reduce sampling variability when comparing success rates across prefix budgets. With 50 pairs, the observed curves exhibit larger fluctuations across adjacent budgets than with 500 pairs. Since our analysis focuses on how planning performance changes with the prefix budget, we report the 500-pair evaluations in the main text and provide the available 50-pair results in Figure 7 for comparison.

A.3 OGBench Cube Results and Evaluation Limitation

We use the single-cube environment from OGBench (Park et al., 2025). Cube's initial-state sampling can select states already within the 4 cm goal-position threshold, a concern reported in the LeWorldModel repository.[1] We therefore retain these evaluations in Figure 8 but exclude Cube from the main comparison. With 500 trials, NestedWM reaches 49.6% success at $d = 1$ and 60.6% at $d = 192$, while LeWM reaches 60.4%. We have not measured the initial-state success fraction or a no-action reference for these evaluation cases, so the small-prefix result does not establish how much success

[1] https://github.com/lucas-maes/le-wm/issues/23

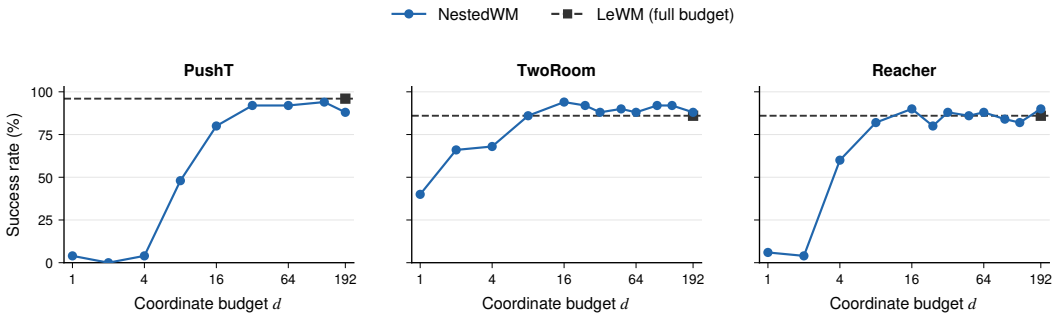


Figure 7: **Benchmark results with 50 planning trials.** NestedWM curves and full-budget LeWM references use the same conventions as Figure 2.

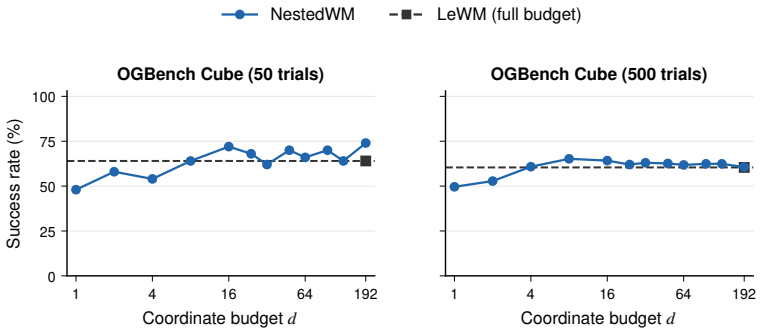


Figure 8: **Retained Cube evaluations with 50 and 500 trials.** The initial-state sampling concern applies to both panels; these are not results from a corrected sampling protocol.

depends on planning. Both models are trained for 100 epochs and otherwise follow the benchmark evaluation settings above.

A.4 Matched Predictor Training Protocol

We freeze the encoder and projection of the models used in Section 4.2. For each representation, we train separate predictors at $d \in \{8, 16, 32, 64\}$ for PushT and $d \in \{16, 32, 64, 128\}$ for Dot2, covering the region around planning saturation. The action encoder, predictor, and prediction projection are randomly initialized with matching architectures, initialization, and training data order between NestedWM and LeWM. Each run uses a fixed prefix budget and minimizes prediction loss over the active coordinates. Training lasts 10 epochs on PushT and 50 epochs on Dot2, matching the source models' training durations. Budget-specific predictors are evaluated on the same 500 initial-state/goal-state pairs within each environment, following Appendix A.1.

A.5 Task-Factor Probe Protocol

We probe the frozen encoder and projection of the 10-epoch PushT and 50-epoch Dot2 models used in the main dimension-wise evaluation. For PushT, we fit probes on 4,096 frames from 128 episodes and evaluate on 1,024 frames from 32 separate episodes in the same training dataset, sampling 32 frames per episode. The episode split separates probe fitting from evaluation; it does not establish that the evaluation episodes were unseen during world-model training. For Dot2, we use 2,048 fitting samples and 2,048 evaluation samples from independently generated crossed dot-position grids over $[-0.98, 0.98]^2$. These static samples are distinct from the planning evaluation trajectories.

We standardize each feature coordinate using its probe-fitting mean and standard deviation, then select neighbors by Euclidean distance in the retained coordinates. Predictions average the target values of the 10 nearest fitting samples. Position targets contain the two spatial coordinates, and

the PushT angle target contains its sine and cosine; we average R^2 over each target's two components. This readout evaluates neighborhoods under the standardized probe metric rather than directly testing the unstandardized planning cost. For random-coordinate comparisons, we use 32 permutations (seeds 51000–51031) and retain the first d entries of each permutation, making selections nested across budgets. At $d = D$, all selections contain the same coordinates and are evaluated once.

A.6 Distractor Evaluation and Variance Decomposition



Figure 9: **Distractor environments.** Predictable moves with the same constant velocity vector across all episodes, with periodic boundary wrapping; Random resamples its position at every step. Both distractors move independently of the agent's actions, and only the agent must reach its target. Faint dots mark subsequent distractor positions from the dataset, with arrows indicating temporal order; Random arrows do not represent continuous paths.

Clean contains one action-controlled dot, with its position clipped to $[-1, 1]^2$ after each step. Predictable adds an action-independent dot that translates by $(0.11, 0.07)$ per step with periodic boundary wrapping. Random instead redraws the second dot's position uniformly from $[-1, 1]^2$ at every step. Only the controlled agent is included in the success criterion. Goal images retain the distractor, so excluding it from the success criterion does not remove it from the observation encoded for latent goal comparison. The three conditions use separate offline datasets and validation trajectories; sharing the Dot1 task does not imply identical datasets or checkpoints. We use the same CNN-based world-model architecture as the Dot environments in Appendix A.1, train each model for 50 epochs with seed 3072, and evaluate 500 initial-state/goal-state pairs per condition with seed 42. The Predictable NestedWM and LeWM models share the architecture, dataset, and training duration. LeWM is trained at full budget and then analyzed at each prefix, without retraining.

For allocation analysis, we independently sample 256 agent positions and 64 distractor positions from $[-0.98, 0.98]^2$ and render all 16,384 combinations. NestedWM and LeWM encoders receive identical position combinations. This static crossed population differs from the trajectories used for planning evaluation. For a fixed budget d, let $z(x, y) \in \mathbb{R}^d$ be the prefix at agent position x and distractor position y. We decompose each coordinate using functional ANOVA (Owen, 2013), with expectations taken uniformly over the crossed samples:

$$\begin{aligned} \mu &= \mathbb{E}_{x,y}[z(x,y)], & f_A(x) &= \mathbb{E}_y[z(x,y)] - \mu, \\ f_D(y) &= \mathbb{E}_x[z(x,y)] - \mu, & f_I(x,y) &= z(x,y) - \mu - f_A(x) - f_D(y). \end{aligned} \tag{13}$$

The zero-mean effects are uncorrelated under this product population, so their covariance matrices satisfy

$$\Sigma_z = \Sigma_A + \Sigma_D + \Sigma_I, \qquad T = \operatorname{tr}(\Sigma_z). \tag{14}$$

We do not assume that the covariance on this analysis population is isotropic or that $T = d$. The interaction term captures dependence on the joint position combination that cannot be expressed as the sum of the two main effects; it does not imply physical interaction between the dots.

At each budget, let P_D be the orthogonal projector onto the smallest set of leading eigenvectors of Σ_D accounting for at least 95% of its trace. We split the agent contribution into

$$V_{A,\text{shared}} = \operatorname{tr}(P_D \Sigma_A), \qquad V_{A,\text{unique}} = \operatorname{tr}((I_d - P_D)\Sigma_A). \tag{15}$$

The plotted components are $\mathrm{tr}(\Sigma_D)/T$, $V_{A,\mathrm{shared}}/T$, $V_{A,\mathrm{unique}}/T$, and $\mathrm{tr}(\Sigma_I)/T$. Shared and unique therefore refer to projected variance relative to this 95% subspace, not shared or exclusive information. The projector is recomputed at each budget; when its rank equals d, all agent variance is shared by definition.

For interval comparisons, we divide each main-effect or interaction trace increase by the total trace increase over the same interval. Specifically, writing $V_F(d) = \mathrm{tr}(\Sigma_F^{(d)})$ for $F \in \{A, D, I\}$, the contribution over $d_1 \to d_2$ is

$$\frac{V_F(d_2) - V_F(d_1)}{T(d_2) - T(d_1)}. \tag{16}$$

Table 3 reports these contributions for NestedWM in Predictable.

Table 3: **Allocation of additional variance in Predictable NestedWM prefixes.** Each entry is the component's share of the total variance increase over the indicated prefix interval.

Prefix interval	Agent (%)	Distractor (%)	Interaction (%)
$8 \to 16$	48.8	48.7	2.5
$16 \to 24$	64.8	18.6	16.6
$24 \to 64$	53.3	10.4	36.3

B Token-wise Model Details

B.1 Architecture and Training

Multi-token architecture. The token-wise model replaces the dimension-wise model's single-vector output with a single-block learned-query cross-attention resampler over spatial backbone features, followed by a shared token-wise projection. Resampler queries attend to the visual features independently of other queries, allowing the first k outputs to be computed without evaluating the remaining queries. The predictor adds learned frame and token-position embeddings and uses frame-block causal attention: tokens may attend to all retained tokens in the same or earlier frames, but not to future frames. Its hidden width remains 192 in both Dot3 ($K = 3$, $D = 192$) and PushT ($K = 24$, $D = 8$).

Teacher-anchored training. Unlike dimension-wise training, token-wise training continues from a full-budget model and adds a separate PTF predictor targeting a frozen copy of the source encoder and projection. We train NestedWM for 10 additional epochs on PushT and 20 additional epochs on Dot3. The PTP and PTF predictors are initialized from separate copies of the source predictor and share the student encoder and action encoder. Projection MLPs use per-token-position batch-normalization statistics with shared weights and affine parameters. For each sampled k, PTP processes an actual k-token prefix and predicts the next student prefix with stop-gradient through the target; PTF receives a zero-masked K-token input and predicts the full teacher target. The two mean-squared-error losses have equal weights and jointly update the student through their inputs, while the teacher remains fixed in evaluation mode.

Prefix execution. Whereas dimension-wise masking preserves the predictor's full input width, token-wise planning evaluates only the first k resampler queries and processes k tokens per history frame in the PTP predictor. Predicted prefixes are fed directly into subsequent rollout steps without restoring suffix tokens, and the goal is encoded at the same budget. The image backbone still processes the full observation, while the teacher and PTF branch are omitted from planning.

B.2 Task States in Token Prefixes

In Dot3, the learned representation concentrates information about all three dots in a small prefix, while additional tokens progressively improve readout accuracy. Figure 10 shows cumulative-prefix probing of the initialization source and the model after token-wise training.

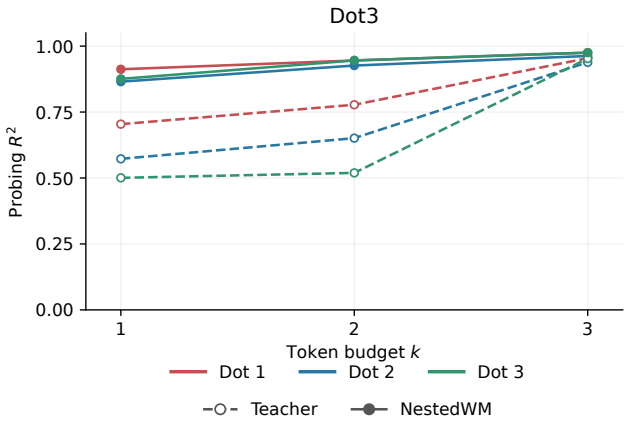


Figure 10: **Recovering dot positions from token prefixes.** Ridge probing uses the concatenated first k tokens to recover each dot's position. Dashed lines denote the teacher and solid lines NestedWM after token-wise training.