MATRYOSHKA WORLD MODELS: TEST-TIME SCALING BEYOND SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with learned world models requires trading off predictive fidelity against computational cost. Existing approaches typically vary search effort while keeping the latent representation fixed, or train separate models for different representation sizes. We introduce **Matryoshka World Models** (MWM), a framework for training world models to operate at multiple nested predictive fidelities. Unlike independently trained fixed-dimensional models, MWM learns a shared hierarchy of predictive representations that can be truncated or expanded at test time without retraining. This variable-fidelity world model enables a new form of planning-time flexibility: predictive fidelity can change across episode-level replanning steps, CEM optimization iterations, and rollout depth. The planner can therefore allocate high-fidelity computation selectively while using cheaper representations where fine-grained prediction is unnecessary. We evaluate variable-fidelity planning against standard fixed-fidelity planning baselines across navigation and manipulation tasks. MWM matches the success rate of the full-fidelity baseline at up to **108.8× less compute** and exceeds the best baseline by up to **59% at the same compute budget**. By making predictive fidelity a resource that can be dynamically allocated across different stages of planning, MWM establishes a new axis of **test-time scaling** for world models.

Project website: https://study-9f2c7a.pages.dev

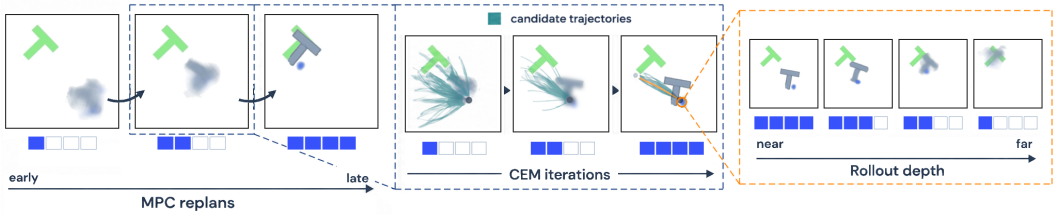


Figure 1: **Predictive fidelity as a planning resource.** MWM allows fidelity to vary across MPC replans, CEM iterations, and along rollout depth. This gives the planner control over both how much computation it spends and where it spends it, supporting different trade-offs between planning performance and cost.

1 INTRODUCTION

Reasoning models let users trade speed for performance by varying the computation spent before answering (OpenAI, 2025). World-model planners face an analogous trade-off before an agent acts, balancing the cost of imagining the future against the quality of the action they choose. Existing planners scale this effort by sampling more candidate trajectories, looking farther ahead, or refining their search for longer (Hansen et al., 2022). However, when we plan, spending more effort need not mean considering more alternatives or looking farther ahead: it can also mean imagining the same course of action in greater detail.

We introduce **Matryoshka World Models (MWM)**, a training framework that brings this flexibility to latent world-model planning. A single trained hierarchy supports multiple predictive fidelities that

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.

can be scheduled throughout planning, extending test-time scaling from the extent of search to the detail of prediction.

MWM builds this hierarchy by applying Matryoshka Representation Learning (Kusupati et al., 2022) during world-model training. A shared encoder learns nested latent prefixes of increasing dimension, each paired with a dynamics module trained to predict at that level. This nested structure provides many ways to exploit variable fidelity predictions. For example, a rollout can move to a smaller level by truncating its predicted state and continuing with the corresponding dynamics module. The construction reuses the underlying world model's training objective and can be applied to different encoder–dynamics architectures.

We use the hierarchy through predefined schedules that select fidelity at three nested scales: successive model predictive control (MPC) replans over an episode, cross-entropy method (CEM) iterations within each replan, and prediction steps within each candidate rollout. Changing these schedules varies both the total prediction budget and where it is spent, without retraining the world model.

We instantiate MWM with LeWorldModel (Maes et al., 2026b) and evaluate visual goal reaching across navigation and manipulation tasks at multiple goal offsets. Joint training improves fixed-level planning: at the same latent dimension, MWM outperforms independently trained single-level models (Single-d). Scheduling fidelity further expands the performance–compute Pareto frontier relative to holding a jointly trained level fixed. Overall, MWM matches the success rate of the full-dimensional Single-d model with up to a 108.8-fold reduction in dynamics-predictor FLOPs and exceeds the best affordable Single-d configuration by up to 59% at the same compute budget. MWM thus trades additional training computation for stronger fixed-level planning and a broader performance–compute frontier at inference, establishing predictive fidelity as a new axis of test-time scaling for world-model planning.

2 Related Work

2.1 World Models for Planning and Control

PlaNet (Hafner et al., 2019) learns latent dynamics from pixels for online trajectory optimization, while Dreamer (Hafner et al., 2020) uses imagined latent trajectories to train an actor and value function. TD-MPC (Hansen et al., 2022) combines short latent rollouts with a learned terminal value and studies how planning horizon and optimization iterations affect the trade-off between inference cost and control performance.

Visual world models may use pretrained features or learn representations jointly with dynamics. V-JEPA (Bardes et al., 2024) learns video representations through masked feature prediction, and V-JEPA 2-AC (Assran et al., 2025) learns action-conditioned dynamics on pretrained video features. DINO-WM (Zhou et al., 2025) learns dynamics over frozen DINOv2 patch features, whereas LeWorldModel (Maes et al., 2026b) jointly learns a compact visual representation and its dynamics. In this paper, we instantiate MWM with LeWorldModel for its compact architecture and simple, stable end-to-end training procedure. We inherit LeWorldModel's encoder-dynamics setup and visual-goal planner, which follows DINO-WM (Zhou et al., 2025) in using CEM to minimize terminal latent goal distance within an MPC loop.

2.2 Multi-resolution Representation Learning

Matryoshka Representation Learning (MRL) (Kusupati et al., 2022) applies a training objective at selected prefixes of one embedding, jointly training representations at several dimensions. MRL reports that jointly trained prefixes can match or improve on independently trained embeddings at the same dimension. It also demonstrates retrieval cascades that use shorter prefixes to screen candidates and longer prefixes to rerank a shortlist.

Other architectures vary computation by changing network capacity or tokenization. MatFormer (Devvrit et al., 2024) nests Transformer feed-forward parameters to provide models at several computational scales, including for autoregressive generation. FlexiViT (Beyer et al., 2023) varies patch size and the resulting token count within a vision transformer. MWM extends MRL

to action-conditioned predictive states, using joint training to strengthen fixed-level planning and enabling fidelity to change during planning. Because the levels share latent coordinates, a state predicted at a larger level can be truncated and passed directly to a smaller dynamics module within the same rollout.

2.3 Adaptive Computation and Test-Time Compute

Multi-resolution control offers direct precedents for allocating predictive computation. Wang et al. (2023) learn dynamics over particle graphs of different sizes and a resolution regressor that selects the particle count at each MPC replan. Model Hierarchy Predictive Control (Li et al., 2021) joins predefined physical models along a trajectory through a state projection; its quadruped example uses detailed dynamics near the present and a reduced trunk model farther ahead.

DIRECT (Dao et al., 2026) learns cost-aware selection among configurations of high-level vision–language planners as a task progresses. Related forms of adaptive computation appear in language-model reasoning and embodied models. TOPS (Yang et al., 2025) trains on the shortest correct responses sampled at different reasoning efforts. MoLe-VLA (Zhang et al., 2026) selects which language-model layers execute at a configured layer count, while M3W (Zhao et al., 2025) uses soft mixing of dynamics experts and sparse routing of reward experts.

MWM brings multi-resolution prediction into learned latent space through jointly trained, compatible latent prefixes. Predefined schedules select these levels across replans, optimization iterations, and rollout depth.

3 Method

3.1 Learning Nested Predictive Representations

Consider a latent world model with a trainable representation module E_θ and an action-conditioned dynamics module G_ϕ. With a frozen encoder, E_θ can include a trainable projection of its features. Given an observation o_t and action a_t, the model computes

$$z_t = E_\theta(o_t) \in \mathbb{R}^D, \qquad \widehat{z}_{t+1} = G_\phi(z_t, a_t).$$

Here a_t may denote either a single action or an action block; we use a single transition for notational simplicity.

We write the training objective of this base setup as $\mathcal{L}_{\text{WM}} = \mathcal{L}_{\text{base}} + \mathcal{R}_{\text{base}}$, where $\mathcal{L}_{\text{base}}$ contains its nonregularizing loss terms and $\mathcal{R}_{\text{base}}$ contains representation regularization. We leave their arguments implicit, since they depend on the base setup.

MWM applies Matryoshka Representation Learning to this setup. We choose levels with latent dimensions $d_1 < d_2 < \cdots < d_K \leq D$[1] and define the prefix at level d_k as $z_t^{(k)} = z_t[1 : d_k]$.

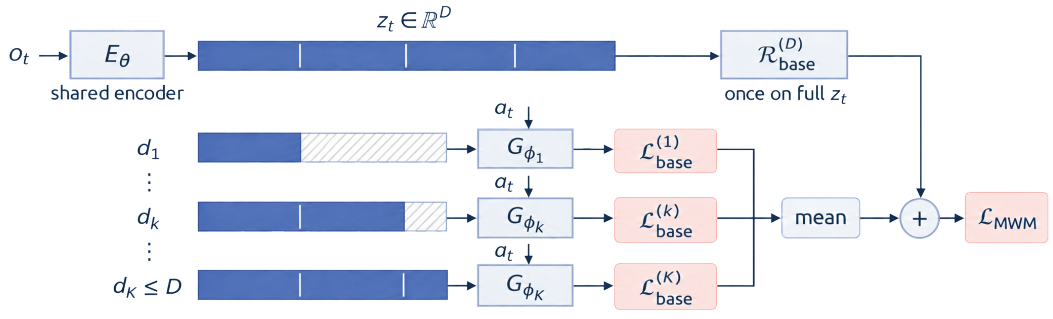


Figure 2: **Matryoshka training.** Hatched suffixes are omitted at smaller levels. The objective averages the per-level losses and adds the full-representation regularizer.

[1] in practice d_K need not be upper-bounded by D, but for our setup we have $d_K = D$.

MWM changes how E_θ is trained, keeping its architecture unchanged. All levels share E_θ, while each has its own dynamics module (Figure 2):

$$\widehat{z}_{t+1}^{(k)} = G_{\phi_k}\left(z_t^{(k)}, a_t\right).$$

At level d_k, both the latent input and output of G_{ϕ_k} have dimension d_k.

Let $\mathcal{L}_{\text{base}}^{(k)}$ denote the nonregularizing base objective evaluated at level d_k, using its latent prefix and corresponding dynamics module. Following equal-weight MRL, we optimize these objectives jointly:

$$\mathcal{L}_{\text{MWM}} = \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_{\text{base}}^{(k)} + \mathcal{R}_{\text{base}}^{(D)}.$$

Here $\mathcal{R}_{\text{base}}^{(D)}$ denotes the base regularizer applied once to the full representation.[2]

MWM scales each dynamics module's internal capacity with its latent dimension. A prediction at level d_k uses only G_{ϕ_k}; smaller levels therefore use smaller dynamics modules and fewer FLOPs per prediction. Appendix A.6 specifies the scaling rule. This lets MWM models operate at different levels of fidelity that trade off accuracy with cost. When $K = 1$, MWM trains a single dynamics module at level d_1; we call this instantiation Single-d. Single-d is the natural point of comparison for evaluating the benefit of jointly training multiple levels. When $d_1 = D$, this single-level instantiation reduces to the underlying world model and its original training objective.

3.2 Allocating fidelity during planning

MWM unlocks changing fidelities on 3 nested axes when planning with MPC and CEM: successive replans over an episode, CEM optimization iterations within each replan, and prediction steps within each candidate rollout inside the model (Figure 1).

These scales serve different roles. Across replans, the need for precision may rise near the goal or peak earlier, for example while passing through a narrow doorway. CEM begins with broad search and progressively distinguishes among similar candidate trajectories. Within a trajectory, early prediction errors compound and uncertainty increases; it may be beneficial to discard noisier, uncertain parts of the latent as the rollout progresses. **These considerations motivate increasing fidelity across CEM iterations and decreasing it along rollout depth; the appropriate allocation across replans depends on where the task requires precision.**

Let e index MPC replans, $i = 1, \ldots, I$ index CEM iterations, and $h = 1, \ldots, H$ index rollout steps. Each inner scheduler S receives the level selected by its enclosing scheduler, giving the level at (e, i, h) as

$$d_{e,i,h} = S_{\text{rollout}}(h; S_{\text{CEM}}(i; S_{\text{episode}}(e))), \qquad d_{e,i,h} \in \{d_1, \ldots, d_K\}.$$

We denote the resulting rollout schedule by $\mathbf{d}_{e,i} = (d_{e,i,1}, \ldots, d_{e,i,H})$. An inner scheduler can hold its inherited level, denoted outer, fixed or use it as an endpoint of a varying schedule.

Our experiments use simple predefined schedules that interpolate linearly over each scale's progress. Future work could make level selection responsive to the current task state and planning feedback, potentially through a learned policy (Section 5). Appendix A.2 gives the schedule construction and a worked example.

3.3 Predicting and scoring candidate trajectories

At MPC replan e and CEM iteration i, let a candidate action sequence $U = (a_0, \ldots, a_{H-1})$ follow level schedule $\mathbf{d}_{e,i}$. The initial latent state is $\widehat{z}_0 = E_\theta(o_t)$. The predicted trajectory then follows

$$\widehat{z}_h = G_{\phi_k}(\widehat{z}_{h-1}[1:d_k], a_{h-1}) \quad \text{when } d_{e,i,h} = d_k, \qquad h = 1, \ldots, H.$$

For dynamics conditioned on a latent history, every state in that history is truncated to the selected level (Figure 3).

[2] Other world-model setups may call for a different scheme, such as applying the regularizer at each level.

Because rollout states are predicted rather than re-encoded from observations, the level may remain fixed or decrease within a rollout but cannot increase once latent dimensions have been discarded:

$$d_{e,i,h+1} \leq d_{e,i,h}, \qquad h = 1, \ldots, H-1.$$

In the visual-goal planning setup used here, let $z_g = E_\theta(o_g)$ be the representation of goal observation o_g. We use the underlying planner's cost form $\mathcal{C}$ at the selected terminal level:

$$J_{e,i}(U) = \mathcal{C}(\widehat{z}_H, z_g[1 : d_{e,i,H}]) .$$

For our LeWorldModel instantiation, $\mathcal{C}(x, y) = \|x - y\|_2^2$. Appendix Algorithm 1 summarizes the planner.

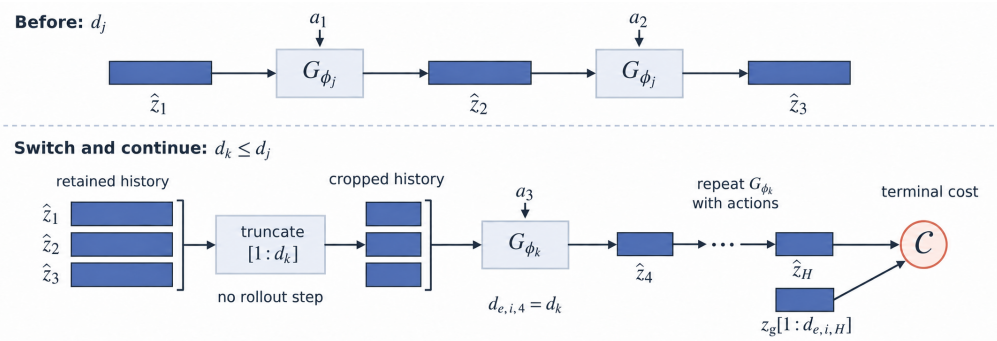


Figure 3: **Changing levels within a rollout.** Bar lengths denote latent dimension. After the retained history is truncated from d_j to $d_k \leq d_j$, prediction continues with G_{ϕ_k}, and the terminal state is compared with the matching goal prefix.

4 Experiments

4.1 Experimental Setup

Tasks. We evaluate visual goal-conditioned planning in TwoRoom (navigation), PushT (planar manipulation), OGBench-Cube (object manipulation), and Reacher (reaching). We sample goal observations $\Delta \in \{25, 50\}$ frames after the initial observation: $\Delta = 25$ follows LeWorldModel, while $\Delta = 50$ tests goals farther apart in time. The rollout horizon H is the number of predicted transitions in each candidate rollout. Increasing Δ also increases the action budget and, for some tasks, H (Appendix A.1).

World models. In each environment, we instantiate MWM with LeWorldModel and train the hierarchy $\mathcal{D} = \{48, 72, 96, 120, 144, 168, 192\}$, retaining LeWorldModel's latent dimension $D = 192$ as the largest level. We average LeWorldModel's predictive objective across levels and apply SIGReg once to the full representation (Section 3.1). With $\mathcal{D} = \{192\}$, our single-level implementation matches or exceeds the published weights' mean planning success in paired evaluations across all four tasks (Appendix A.5).

Comparisons. **Single-d** trains a separate model with $\mathcal{D} = \{d\}$ for each evaluated level and uses that level throughout planning. **MWM (fixed)** uses one jointly trained level throughout planning; **MWM (scheduled)** varies levels through the nested schedules. Both MWM settings use the same checkpoint in each comparison.

Planning. All settings use LeWorldModel's planning setup: CEM optimizes candidate trajectories within an MPC loop. We vary CEM population size and iteration count to span planning costs. After training, we choose the minimum scheduling level $d_{\min}$ as the smallest level where MWM (fixed) exceeds 60% success (Appendix A.1).

Metrics. We report task success rate and mean dynamics GFLOPs per episode, counting the full dynamics module across all candidate rollouts, CEM iterations, and replans. All settings use the same encoder architecture and incur the same cost per encoding pass (Appendix A.6); we exclude this shared cost to isolate dynamics computation.

Each configuration pairs CEM settings with a fixed level or a predefined schedule (Table 1). During planning, these schedules determine level choices without using a GFLOP budget or a model of expected planning performance. We measure each configuration's success and mean dynamics cost. At budget B, the empirical frontier reports the highest observed success among configurations with mean cost at most B. Appendix A.1 gives the full evaluation protocol.

4.2 Joint Training Improves Fixed-Level Planning

Joint training substantially improves success at low and intermediate levels in TwoRoom, PushT, and Reacher (Figure 4). Its advantage narrows at the largest levels, consistent with joint training being most valuable when latent capacity is limited. In OGBench-Cube, MWM's success is comparatively stable across levels, and its advantage over Single-d is smaller. This suggests that MWM allows some predictors to become more effective at planning time without increasing the computation required for each prediction.

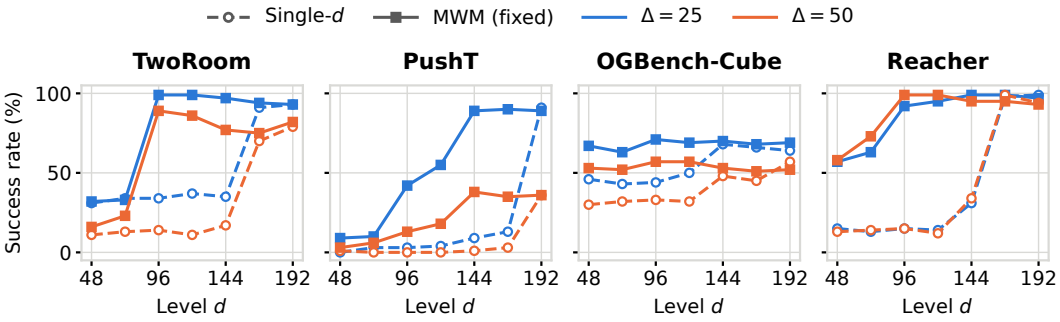


Figure 4: **Planning at a fixed level.** Success rate of Single-d and MWM (fixed) across latent levels. All levels use the same CEM settings (Appendix A.1).

Figure 5 illustrates how predictive detail changes across levels within MWM. Lower levels preserve coarse scene structure but distort control-relevant details: in TwoRoom, the layout remains clear while the agent's reconstructed position becomes less accurate. Differences become even more pronounced 25 environment steps ahead, with higher levels better preserving object states.

4.3 MWM Expands the Performance–Compute Frontier

The fixed-level results show that joint training improves the representations available to the planner. We next ask whether scheduling fidelity during planning provides a better performance–compute trade-off than holding one level fixed. Figure 6 compares the resulting frontiers as CEM search effort varies.

Joint training provides a stronger starting point for compute-efficient planning: MWM (fixed) reaches Single-d's peak success with 2–35$\times$ less dynamics compute in six of eight settings. Scheduling extends this benefit beyond choosing a cheaper constant level: MWM (scheduled) reaches the highest success attained by both MWM variants with less compute in all eight settings. In OGBench-Cube, the main benefit is reduced computation at similar success rates; in TwoRoom and Reacher, scheduling also improves attainable success. It exceeds Single-d's peak success in seven of eight settings. The exception is PushT at $\Delta = 25$, where Single-d outperforms both MWM variants: it reaches 93% success, compared with 87% for MWM (scheduled) and 85% for MWM (fixed).

At the larger goal offset $\Delta = 50$, scheduling yields particularly large gains in TwoRoom and Reacher. In TwoRoom, scheduled MWM reaches the full-dimensional Single-d model's 71% peak success with $108.8\times$ less dynamics computation. At a budget of 47.56 GFLOPs per episode in Reacher, MWM (scheduled) achieves 78% success versus 19% for the best affordable Single-d con-

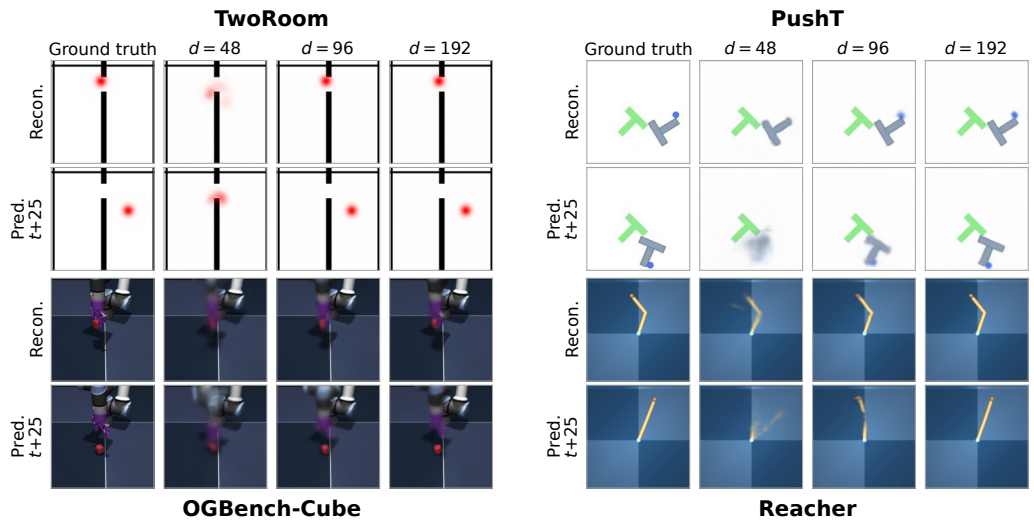


Figure 5: **Reconstruction of MWM nested latents.** For each task, all levels use the same initial observation and 25-step recorded action trajectory. See Appendix A.3 for details and all seven levels.

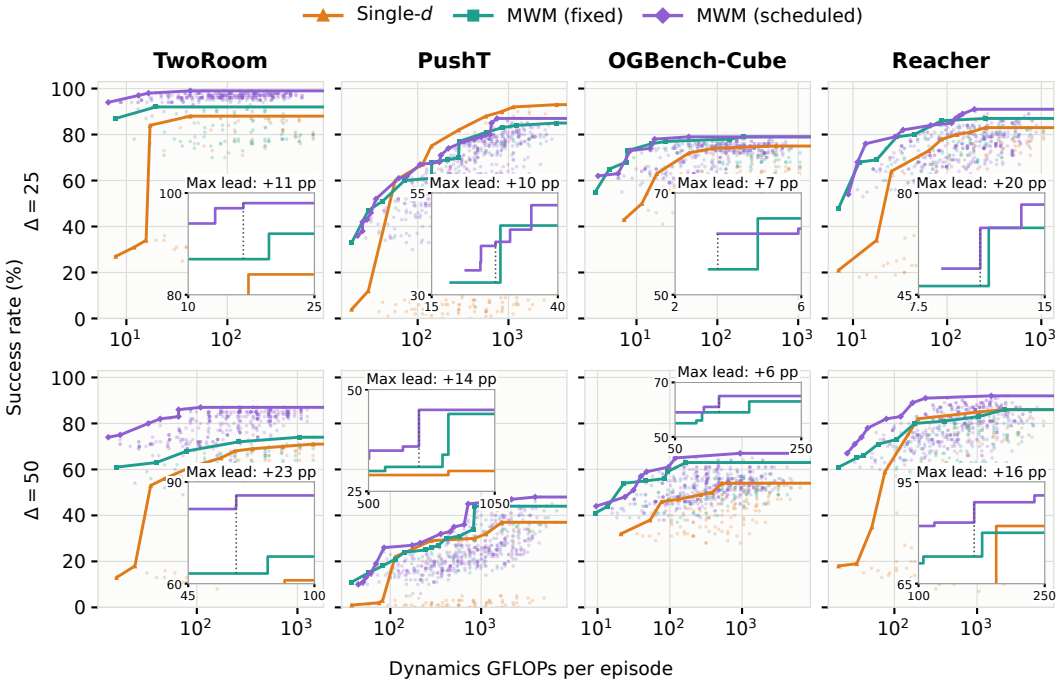


Figure 6: **Planning performance across compute budgets.** Faint points show evaluated configurations; solid lines connect empirical Pareto frontiers. Horizontal extensions retain the best observed success at larger budgets, where the same cheaper configuration remains available; they do not represent additional evaluations. Insets highlight the largest observed success-rate advantage of MWM (scheduled) over MWM (fixed) under the same compute budget. Main compute axes are logarithmic; inset axes are linear and panel-specific.

figuration, a gain of 59 percentage points. The PushT ranking also reverses at this offset: scheduled MWM reaches 48% success, compared with 44% for MWM (fixed) and 37% for Single-d. These results show that the benefit depends on the task and goal offset, with gains in both compute efficiency and peak success in the larger-offset evaluations.

Note that these gains arise from a deliberately restricted schedule family. We choose $d_{\min}$ using the heuristic in Appendix A.1, fix the upper endpoint at d_K, and use linear transitions without searching over endpoint values. The selected range may include levels that are too coarse for some decisions or unnecessarily costly for others. Thus, the reported frontier demonstrates what simple predefined schedules can achieve, while leaving open the gains from adapting both the levels used and when they are used.

We next examine which planning scales account for these gains.

4.4 How Should Fidelity Be Allocated During Planning?

To identify where fidelity scheduling helps, we apply it at one planning scale at a time, using MWM-Fixed at $d_K = 192$ as the reference. Episode-only and CEM-only schedules increase fidelity; rollout-only schedules decrease it. Any scheduler outside the scheduled scale passes down d_K, while any scheduler inside it uses `outer` to inherit the scheduled choice (Section 3.2). Figure 7 summarizes the effect of scheduling at each planning scale across compute budgets. Appendix A.4 gives the schedule endpoints and full frontiers, including the comparison against the best evaluated fixed level.

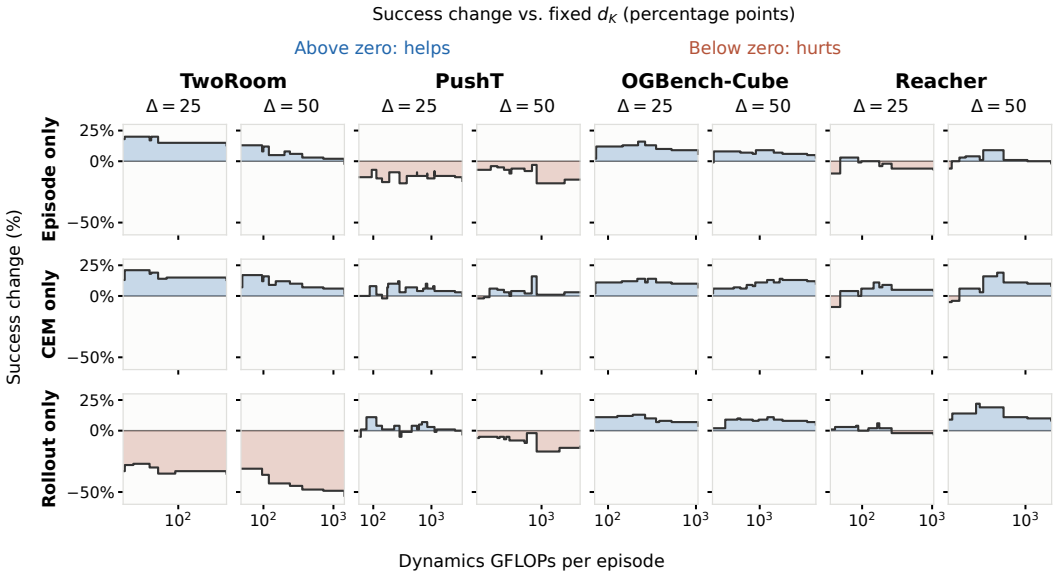


Figure 7: **Where does scheduling help?** Change in best observed success at matched compute budgets relative to MWM held at d_K, with scheduling applied at one scale at a time. The three rows share the same budget range within each task–offset setting.

CEM scheduling refines predictions before actions are executed. CEM-only scheduling gives the most consistent gains across the eight settings. Early CEM iterations use inexpensive predictions to explore broadly, while later iterations use finer predictions to distinguish increasingly similar trajectories. In every replan, the final CEM population is evaluated at d_K, so the last update is guided by full-fidelity predictions. In PushT at $\Delta = 25$, this schedule preserves the fixed model's 85% peak success with $3.2\times$ less compute. The advantage over the best constant level is smaller in some tasks (Appendix A.4).

Episode scheduling depends on when precision is needed. Across replans, the value of fidelity depends on the decisions encountered during execution. A chosen level affects every CEM iteration and candidate rollout in that replan, so episode scheduling can concentrate computation on stages requiring precise control. The tested increasing schedule improves efficiency in TwoRoom and OGBench-Cube: in TwoRoom at $\Delta = 50$, episode-only scheduling reaches 72% success with $7.2\times$ less compute than fixed d_K. Its weaker results in PushT show that the same progression does not

transfer uniformly across tasks. These results motivate choosing fidelity based on current control demands as well as episode progress.

Rollout scheduling can concentrate fidelity on early predictions. As a rollout extends, prediction errors compound and epistemic uncertainty grows. A decreasing fidelity schedule is intended to mirror this decline in confidence: the planner can drop noisier suffix states and spend less compute where accumulated model error already limits reliability. The gains in OGBench-Cube at both offsets and Reacher at $\Delta = 50$ are consistent with this allocation; in Reacher, rollout-only scheduling raises peak success from 83% to 91%. The retained prefix must still support terminal goal comparison (Section 3.3), with the losses in TwoRoom suggesting that the tested schedule reduces fidelity too far for that purpose. This motivates choosing the terminal level to preserve the distinctions needed for goal evaluation while using finer predictions earlier in the rollout.

Together, these results motivate schedules that respond to the current state and search uncertainty, as well as progress through the episode or rollout. The interactions in Appendix A.4 also reinforce the need to coordinate allocation across scales: combining individually useful schedules can erase their gains.

5 Conclusion and Future Work

We introduced Matryoshka World Models, which train a shared hierarchy of predictive representations and allow planners to allocate fidelity across MPC replans, CEM iterations, and rollout depth. Joint training improves fixed-level planning. In all of our settings, scheduling reaches the highest success rate attained by both MWM variants with less computation than fixed-level planning. At $\Delta = 50$, scheduled MWM meets the full-dimensional Single-d baseline's success rate with $108.8\times$ fewer dynamics-predictor FLOPs in TwoRoom and improves success by 59 percentage points over the best affordable Single-d configuration in Reacher.

The scheduling analysis shows that effective fidelity allocation depends on where computation is spent. Increasing fidelity across CEM iterations gives the most consistent gains, while episode and rollout schedules show stronger task dependence.

Shared dynamics. Each additional MWM level currently requires another dynamics module, increasing model storage. A dynamics module that operates at multiple levels could reduce this overhead through parameter sharing. One possibility is a transformer that predicts dynamics over a variable-length sequence of latent tokens, with each token carrying the information added by one level. Smaller levels would use shorter sequences, so the shared network could make cheaper predictions at lower fidelity.

Designing the hierarchy. The number and spacing of levels are currently chosen heuristically. Future work could jointly choose which levels to train and how much dynamics capacity to assign to each, balancing planning performance across compute budgets against training cost. Training objectives could also reflect each level's role in planning. For smaller levels, the objective could penalize screening out promising trajectories; for larger levels, it could emphasize distinctions between closely matched candidates. This adapts MRL's proposal to use different losses at different fidelities to the planning setting (Kusupati et al., 2022).

Adaptive planning. Our present schedules choose levels from planning progress. An adaptive planner could learn to maximize expected planning success under a specified compute budget. It could use the observation, goal, search state, the hierarchy's observed performance at different levels, and remaining budget to decide which levels to use and when to switch. For example, it might increase fidelity as CEM's distribution narrows, or use cheaper levels to evaluate more trajectories within the same budget.

These directions build on MWM's central contribution: making predictive fidelity a controllable resource, so that world-model planners can scale both the extent of search and the detail of their predictions.

AI USE STATEMENT

Generative AI tools, including Astra, 5.6 Sol, and Claude Sonnet, assisted with software implementation, debugging, monitoring training and evaluation runs, and developing plotting scripts and figures. They also supported literature searches and citation checks, manuscript drafting and revision, mathematical exposition, and feedback on experimental design and the interpretation and presentation of results. The authors developed the research idea, hypotheses, and core method. Generative AI was not used to create training or evaluation data. Software checks included automated tests and the single-level planning-performance comparison with published LeWorldModel weights reported in Appendix A.5. The authors reviewed and revised AI-assisted manuscript content and are responsible for the paper's contents.

ETHICS STATEMENT

All experiments were conducted in simulation using datasets released with LeWorldModel. Our evaluation covers visual planning in navigation and manipulation benchmarks. The planner uses CEM within an MPC loop to select actions using predictions from a learned world model; no reasoning language model participates in the control loop.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the method and experimental protocol, with additional implementation and evaluation details in Appendix A.1. The accompanying repository provides the exact code revision, configurations, checkpoints, data-acquisition instructions, evaluation episode definitions and seeds, raw results, source data for quantitative plots, dependencies, hardware specifications, and training-cost records. The project website is available at https://study-9f2c7a.pages.dev. Appendix A.5 reports the planning-performance comparison between a model trained with $\mathcal{D} = \{192\}$ and the published LeWorldModel weights across all four evaluated tasks, using the same evaluation pipeline for both.

REFERENCES

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. URL https://arxiv.org/abs/2506.09985.

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. ISSN 2835-8856. URL https://openreview.net/forum?id=QaCCuDfBk2.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14496–14506, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Beyer_FlexiViT_One_Model_for_All_Patch_Sizes_CVPR_2023_paper.html.

Jadelynn Dao, Milan Ganai, Yasmina Abukhadra, Ajay Sridhar, Mozhgan Nasr Azadani, Katie Luo, Clark Barrett, Jiajun Wu, Chelsea Finn, and Marco Pavone. DIRECT: When and where should you allocate test-time compute in embodied planners? *arXiv preprint arXiv:2606.12402*, 2026. URL https://arxiv.org/abs/2606.12402.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. MatFormer: Nested transformer for elastic inference. In *Advances in Neural Information Processing Systems*, volume 37, pp. 140535–140564, 2024. doi: 10.52202/079017-4461. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/fe066022bab2a6c6a3c57032a1623c70-Abstract-Conference.html.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

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 *Advances in Neural Information Processing Systems*, 2022.

He Li, Robert J. Frei, and Patrick M. Wensing. Model hierarchy predictive control of robotic systems. *IEEE Robotics and Automation Letters*, 6(2):3373–3380, April 2021. doi: 10.1109/LRA.2021.3061322. URL https://doi.org/10.1109/LRA.2021.3061322.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation. *arXiv preprint arXiv:2605.21800*, 2026a. URL https://arxiv.org/abs/2605.21800.

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*, 2026b. URL https://arxiv.org/abs/2603.19312.

OpenAI. Introducing GPT-5 for developers, August 2025. URL https://openai.com/index/introducing-gpt-5-for-developers/.

Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-resolution model learning for object pile manipulation. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.047. URL https://www.roboticsproceedings.org/rss19/p047.html.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 43605–43631, 2025. doi: 10.52202/085713-1452. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e22bea3b170f4c2aebb9c48d98ae64d-Abstract-Conference.html.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. MoLe-VLA: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18764–18772, 2026. doi: 10.1609/aaai.v40i22.38945. URL https://ojs.aaai.org/index.php/AAAI/article/view/38945.

Zijie Zhao, Zhongyue Zhao, Kaixuan Xu, Yuqian Fu, Jiajun Chai, Yuanheng Zhu, and Dongbin Zhao. Learning and planning multi-agent tasks via an MoE-based world model. In *Advances in Neural Information Processing Systems*, volume 38, pp. 40857–40890, 2025. doi: 10.52202/085713-1364. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3a2d96d2eb2902043c2db705ca03e9a2-Abstract-Conference.html.

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