MOTO-WM: LEARNING MULTIPLE TASKS ONLINE WITH ONE WORLD MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

World models support model-based reinforcement learning through planning and imagined policy optimization. Extending them to multiple tasks requires sharing predictive structure while preserving differences between task objectives. Through empirical and representation analyses of independently trained DreamerV3 agents, we find shared physical structure in recurrent states but objective-dependent reward and policy readouts. Motivated by this observation, we propose Multi-task Online Training with One World Model (MOTO-WM), which shares a world model and actor–critic across tasks while conditioning feedback and control on task identity. A unified action interface supports different action dimensions. MOTO-WM learns online from scratch without demonstrations, precollected data, or pretrained models. On 20 visual DeepMind Control tasks, a single 10.55M-parameter model outperforms the evaluated model-based baselines at 500K steps per task with only 1.48 GPU-days of training. Meta-World results, ablations, and controlled task interventions further validate the method and its task-conditioned behavior. Essential scripts and checkpoint are provided in the supplementary material.

1 INTRODUCTION

World models learn predictive representations of environment dynamics from experience and have become a rapidly evolving research direction (Ding et al., 2025). Recent work spans generative environment simulation (Bruce et al., 2024), self-supervised predictive representation learning (Assran et al., 2023; Bardes et al., 2024), and large-scale multimodal modeling over long video–language sequences (Liu et al., 2025). In this work, we focus on model-based reinforcement learning, where action-conditioned dynamics are learned from interaction and used for planning or imagined policy learning (Hafner et al., 2019b). Such models can reuse predictive structure to improve control beyond direct environment interaction.

However, Dreamer-style world-model methods are typically trained separately for individual tasks, even when tasks share the same physical system but differ in their objectives. Prior work has shown that shared dynamics can support transfer across reward functions (Laroche & Barlier, 2017), and PlaNet demonstrated multi-task learning with a single latent dynamics model (Hafner et al., 2019b). The key challenge is to reuse shared predictive structure while preserving objective-specific feedback and control.

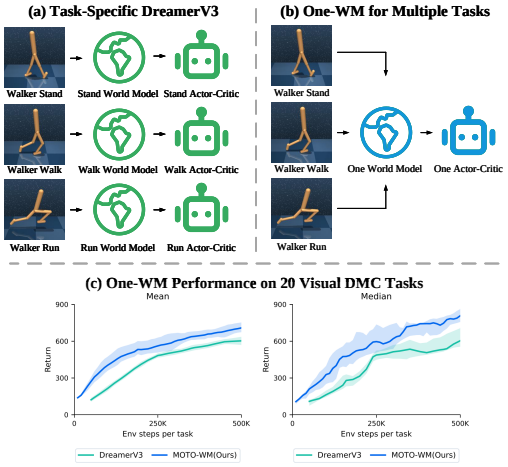


Figure 1: **From task-specific DreamerV3 agents to MOTO-WM.** (a) DreamerV3 trains a separate agent for each task. (b) MOTO-WM shares one agent across tasks, with task conditioning distinguishing different objectives. (c) At 500K steps/task, MOTO-WM reaches 722.86/855.17 mean/median return versus 606.57/625.79 for DreamerV3 on DMC20.

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.

Recent studies have extended world-model-based reinforcement learning to multi-task control through large task-conditioned models, modular or expert architectures, and demonstration- or pretraining-based approaches (Hansen et al., 2024; Zhang et al., 2026; Pu et al., 2026; Hansen et al., 2026). These works establish the feasibility of multi-task world modeling, but compact and highly shared models learned online from scratch remain less explored. The central challenge is to reuse common predictive structure across tasks while preserving the distinct feedback and control required by different objectives.

To understand how this trade-off can be resolved, we first examine what is already shared and what remains task dependent in independently trained single-task DreamerV3 agents. Under matched observation–action histories, their deterministic recurrent states exhibit substantial structural similarity and retain accessible physical information, while downstream reward and policy readouts remain differentiated across objectives. These observations support a shared world-model backbone, with explicit task conditioning reserved for reward prediction and control. Motivated by these observations, we propose **Multi-task Online Training with One World Model (MOTO-WM)**, which learns multiple tasks online with a single shared world model and actor–critic, while explicitly conditioning feedback prediction and control on task identity. A unified action interface accommodates different action dimensions, while task-specific environment instances and per-task replay buffers support joint online learning across tasks. Experiments across visual DeepMind Control and Meta-World show that MOTO-WM can learn multiple tasks with a single compact model while reducing both task-specific model duplication and overall interaction requirements. Our contributions are summarized as follows:

- We provide an empirical analysis of independently trained DreamerV3 agents, revealing substantial shared structure in recurrent world-model states alongside objective-dependent reward and policy readouts.
- We propose MOTO-WM, a compact framework that learns multiple tasks online with one shared world model and actor–critic, using explicit task conditioning and a unified action interface.
- We demonstrate strong multi-task performance on visual DMC and Meta-World, with substantially reduced model duplication and overall interaction requirements, supported by ablations and controlled task-conditioning analyses.

2 Related Work

2.1 World Models for Control

World models support control through different forms of predictive modeling. Latent world models learn compact action-conditioned dynamics for planning or imagined policy learning (Hafner et al., 2019b;a; 2025; Hansen et al., 2022; Micheli et al., 2023; Zhang et al., 2023), while other approaches predict future observations directly through generative models (Alonso et al., 2024). Several studies further investigate what information world-model representations should preserve for control (Fu et al., 2021; Deng et al., 2022; Wang et al., 2022). More recently, predictive world modeling has also been integrated with vision-language-action models for robotic control (Cen et al., 2025). Most control-oriented world models, however, are developed and evaluated for individual tasks, leaving the extent to which their learned predictive representations can be shared across different control objectives less understood.

2.2 Multi-Task Model-Based Reinforcement Learning

Multi-task MBRL has explored several forms of model sharing across tasks. PlaNet showed that a single latent dynamics model can support multiple control tasks, while Task Aware Dreamer studied shared dynamics with task-dependent rewards and generalization (Hafner et al., 2019b; Xinning et al., 2023). More recent work scales multi-task world models through task conditioning, pretraining, or large-scale joint training, including TD-MPC2, PWM, and Newt (Hansen et al., 2024; Georgiev et al., 2025; Hansen et al., 2026).

Other methods introduce explicit specialization through task grouping, routed experts, sparse mixture-of-experts, dynamic capacity expansion, or hierarchical sharing, as in Mixture-of-World

Models, ScaleZero, and TaskLoom (Zhang et al., 2026; Pu et al., 2026; Zeng et al., 2026). These studies provide diverse solutions to multi-task world modeling, while compact and highly shared models learned entirely from online interaction remain less explored.

3 PRELIMINARIES

3.1 MULTI-TASK REINFORCEMENT LEARNING

We consider a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, P the transition kernel, r the reward function, and γ the discount factor. A policy π_θ maximizes

$$J(\theta) = \mathbb{E}_{P,\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{1}$$

Model-based RL learns a dynamics model $\hat{P}_\phi \approx P(s_{t+1} \mid s_t, a_t)$ and uses it for policy learning.

In multi-task RL, we consider K tasks $\mathcal{T} = \{\mathcal{T}^1, \ldots, \mathcal{T}^K\}$ with $\mathcal{M}^k = (\mathcal{S}^k, \mathcal{A}^k, P^k, r^k, \gamma^k)$. A shared policy is optimized over all tasks:

$$J_{\mathrm{MT}}(\theta) = \sum_{k=1}^{K} \mathbb{E}_{P^k,\pi_\theta}\left[\sum_{t=0}^{\infty} (\gamma^k)^t r^k(s_t^k, a_t^k)\right]. \tag{2}$$

3.2 DREAMERV3

DreamerV3 is a model-based Actor–Critic method that learns a world model from replay and trains its policy on imagined latent trajectories (Hafner et al., 2025). Its recurrent state-space model (RSSM) represents the latent state as $s_t = (h_t, z_t)$, where h_t is deterministic and z_t is stochastic:

$$\begin{aligned}
h_t &= f_\phi(h_{t-1}, z_{t-1}, a_{t-1}),\\
z_t &\sim q_\phi(z_t \mid h_t, x_t),\\
\hat{z}_t &\sim p_\phi(\hat{z}_t \mid h_t),\\
\hat{r}_t &\sim p_\phi(\hat{r}_t \mid h_t, z_t),\\
\hat{c}_t &\sim p_\phi(\hat{c}_t \mid h_t, z_t),\\
\hat{x}_t &\sim p_\phi(\hat{x}_t \mid h_t, z_t).
\end{aligned} \tag{3}$$

The posterior incorporates the current observation, while the prior enables latent imagination. The world model is trained with prediction, dynamics, and representation losses:

$$\mathcal{L}_{\mathrm{WM}} = \mathbb{E}_{q_\phi}\left[\sum_{t=1}^{T} \left(\beta_{\mathrm{pred}}\mathcal{L}_{\mathrm{pred}}^t + \beta_{\mathrm{dyn}}\mathcal{L}_{\mathrm{dyn}}^t + \beta_{\mathrm{rep}}\mathcal{L}_{\mathrm{rep}}^t\right)\right]. \tag{4}$$

DreamerV3 then learns the Critic from imagined λ-returns,

$$\mathcal{L}_{\mathrm{critic}}(\psi) = -\mathbb{E}_{\hat{\tau}}\left[\sum_{t=1}^{H} \log p_\psi(G_t^\lambda \mid s_t)\right], \tag{5}$$

and optimizes the Actor using $A_t^\lambda = G_t^\lambda - v_\psi(s_t)$:

$$\mathcal{L}_{\mathrm{actor}}(\theta) = -\mathbb{E}_{\hat{\tau}}\left[\sum_{t=1}^{H} \left(\mathrm{sg}(A_t^\lambda) \log \pi_\theta(a_t \mid s_t) + \eta \mathcal{H}(\pi_\theta(\cdot \mid s_t))\right)\right]. \tag{6}$$

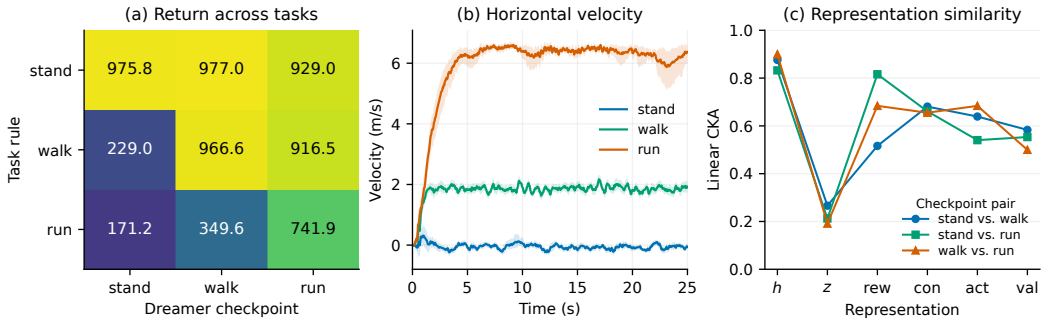


Figure 2: **Native single-task Dreamer diagnostics.** (a) Mean return under different reward rules and controllers. (b) Horizontal velocity over 25 s with 95% pair-bootstrap intervals. (c) Same-history CKA for h, sampled z, and reward/continuation/actor/critic hidden features.

Thus, DreamerV3 learns latent dynamics for imagination and trains the Actor–Critic entirely within the learned latent space.

4 Motivation

Tasks may share the same embodiment, dynamics, and action space while pursuing different objectives. To isolate objective differences from changes in physical structure, we analyze three independently trained DreamerV3 checkpoints on DMC Walker *stand*, *walk*, and *run*. These tasks share the same Walker system and six-dimensional action space but require distinct behaviors. We therefore examine which components exhibit reusable structure and where explicit task information remains necessary.

We freeze the three checkpoints and replay matched observation–action histories from 20 paired initial states, yielding 960 aligned readout points per model (Appendix A; Tables 5–7).

Shared physics does not imply shared behavior. Cross-task evaluation shows asymmetric compatibility between objectives. The *stand* controller performs poorly under *walk* and *run*, whereas the *run* controller also achieves high *stand* and *walk* returns (Figure 2a), partly because fast upright motion satisfies both reward rules. Their mean velocities are -0.047, 1.809, and 5.917 m/s, respectively (Figure 2b). Thus, identical physics can support clearly different objective-dependent behaviors.

The recurrent state retains shared structure. We use linear CKA (Kornblith et al., 2019) to compare representations across independently trained networks without assuming neuron-wise alignment. For column-centered, row-aligned feature matrices X and Y,

$$\mathrm{CKA}(X, Y) = \frac{\|X^\top Y\|_F^2}{\|X^\top X\|_F \, \|Y^\top Y\|_F}. \tag{7}$$

The deterministic RSSM state h shows the highest cross-model similarity, with CKA values of 0.832–0.900 (Figure 2c). Reward, actor, and critic features are less similar, indicating shared recurrent structure but more objective-specific downstream readouts.

The recurrent state preserves physical information. We further test whether this shared structure contains meaningful physical content using linear probes on frozen recurrent states. Across the three checkpoints, horizontal-velocity decoding yields RMSEs of 0.816–0.973 m/s and $R^2 = 0.866$–0.906. Similar trends are observed for body height and uprightness (Appendix A), showing that h retains linearly accessible physical information.

Implications for Multi-Task World Modeling. These results reveal a consistent separation: recurrent world-model states retain substantial shared physical structure, whereas behavior and down-

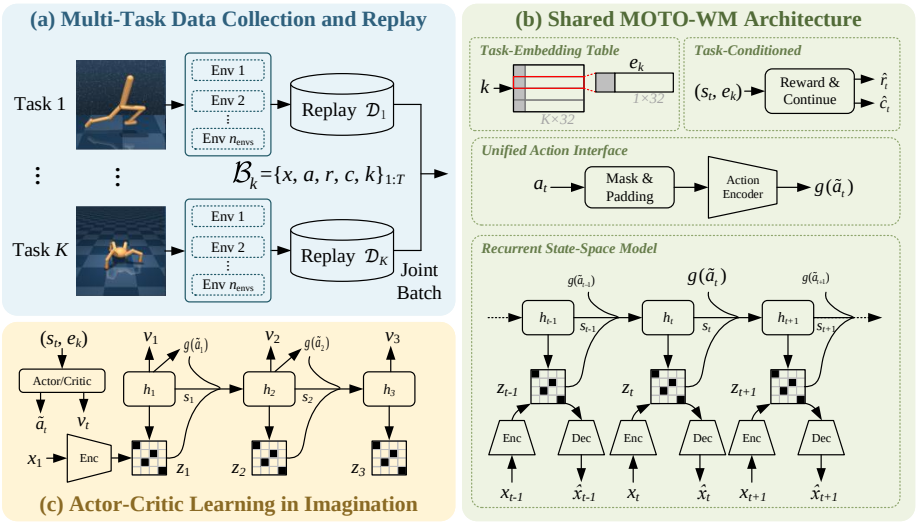


Figure 3: **MOTO-WM overview.** (a) Each task uses multiple environment instances and a dedicated replay buffer; sampled sequences form a joint multi-task batch. (b) The shared model includes a task-embedding table, task-conditioned reward and continuation predictors, a unified action interface, and an encoder–RSSM–decoder backbone. (c) The shared actor–critic learns from imagined trajectories using $s_t = (h_t, z_t)$ and task embedding e_k; task identity remains fixed within each rollout, and recurrent states remain trajectory specific.

stream readouts vary across objectives. This motivates sharing the world-model backbone while conditioning reward prediction and control on task identity.

5 Multi-task Online Training with One World Model

5.1 Methodology Overview

MOTO-WM shares the predictive backbone and readout networks while task embeddings distinguish feedback and control objectives (Figure 3). Parallel environments maintain independent recurrent states and populate per-task replay buffers. A joint batch trains the observation encoder and RSSM through a unified action interface. Actor–critic learning starts from replay-inferred states and uses imagined actions, rewards, continuation probabilities, and values; the task ID remains fixed within each rollout. The updated policy then collects further experience. All components learn online from scratch, without demonstrations, offline data, or pretrained models.

5.2 Shared World Model with Task-Conditioned Feedback and Control

The motivation analysis in Sec. 4 indicates that recurrent representations retain substantial common structure across tasks. Meanwhile, downstream predictions and control behaviors differ across objectives. MOTO-WM therefore uses a shared world-model backbone together with task-conditioned readouts.

MOTO-WM uses the shared RSSM defined in Eq. 3. The main modification to the recurrent transition is that actions from different tasks are first mapped to a unified interface and then processed by a shared action encoder:

$$h_t = f_\phi(h_{t-1}, z_{t-1}, g_\phi(\tilde{a}_{t-1}))\,, \tag{8}$$

where $\tilde{a}_{t-1}$ denotes the unified action vector and g_ϕ is the shared action encoder introduced in Sec. 5.3.

A learnable table $\mathbf{E} \in \mathbb{R}^{K \times d_e}$ assigns task k an embedding $e_k = \mathbf{E}[k]$, where $d_e = 32$. The task embedding is concatenated with the latent state and provided to four shared task-conditioned

readouts:

$$\begin{aligned} &p_\phi(r_t \mid s_t, e_k), \qquad p_\phi(c_t \mid s_t, e_k), \\ &\pi_\theta(a_t \mid s_t, e_k), \qquad p_\psi(G_t^\lambda \mid s_t, e_k). \end{aligned} \tag{9}$$

As motivated in Sec. 4, the recurrent state retains substantial shared physical structure across tasks, while task differences are more pronounced in downstream reward and policy readouts. We therefore restrict explicit task conditioning to the feedback and actor-critic readouts.

This differs from other multi-task world-model approaches that inject task information more broadly into the predictive model, such as TD-MPC2 (Hansen et al., 2024), or introduce explicit specialization through task-conditioned experts or task grouping, as in MoW (Zhang et al., 2026) and TaskLoom (Zeng et al., 2026). MOTO-WM instead keeps a highly shared world-model backbone and uses task identity only where different objectives need to be distinguished.

For policy learning, imagined trajectories start from replay posterior states, with task identity fixed throughout each rollout. The shared RSSM predicts future latent states, while the actor, reward, continuation, and value readouts are conditioned on that task identity.

5.3 Unified Action Interface

Different tasks may have different numbers of continuous action dimensions, which prevents direct sharing of the actor output and RSSM action input. MOTO-WM therefore maps all actions to a fixed-width interface with $d_{\max} = 16$.

For task k with $d_k \le d_{\max}$, we define a fixed mask $m_{k,j} = \mathbf{1}[j \le d_k]$. The shared actor always outputs a $d_{\max}$-dimensional action distribution, while only the first d_k coordinates are valid. For example, a six-dimensional task uses $m_k = [1, 1, 1, 1, 1, 1, 0, \ldots, 0] \in \{0, 1\}^{16}$, where the remaining ten coordinates are zero and excluded from policy log-probability and entropy. The unified action is

$$\tilde{a}_t^{(k)} = m_k \odot \mathrm{clip}(a_t, -1, 1). \tag{10}$$

The same interface is used for environment interaction, replay, and imagination. Before entering the RSSM, $\tilde{a}_t^{(k)}$ is processed by a shared nonlinear action encoder:

$$g_\phi(\tilde{a}_t^{(k)}) = \mathrm{SiLU}\Big(\mathrm{RMSNorm}\Big(W_2\,\mathrm{SiLU}(W_1\tilde{a}_t^{(k)} + b_1) + b_2\Big)\Big), \tag{11}$$

with dimensions $16 \to 64 \to 256$. This allows the same actor and RSSM to operate across tasks with different action dimensions.

5.4 Online Multi-Task Data Collection and Replay

MOTO-WM separates task-specific experience collection from shared model learning. For each of the K tasks, we instantiate n_{envs} parallel environments, resulting in Kn_{envs} environments in total. All environments use the same policy parameters while maintaining independent recurrent states, and transitions from task k are stored in a dedicated replay buffer $\mathcal{D}_k$.

Each learner batch samples b sequences from every task:

$$\mathcal{B} = \mathrm{concat}\left(\mathcal{B}_1, \ldots, \mathcal{B}_K\right), \qquad |\mathcal{B}_k| = b, \qquad B = Kb. \tag{12}$$

Each sequence contains T training positions and a fixed task identity. The per-task sampling quota prevents larger replay buffers from dominating training. Within each buffer, newly collected sequences are prioritized before falling back to uniform replay. Besides,a global replay ratio ρ controls the amount of learning relative to online interaction.

5.5 Joint Optimization and Data-Parallel Training

Joint learning objective. MOTO-WM applies the objectives in Eqs. 4–6 jointly across tasks. The overall objective is

$$\mathcal{L}_{\text{MOTO-WM}} = \frac{1}{K}\sum_{k=1}^{K}\left[\mathcal{L}_{\text{WM}}^{(k)} + \mathcal{L}_{\text{critic}}^{(k)} + \mathcal{L}_{\text{actor}}^{(k)} + \beta_{\text{rv}}\mathcal{L}_{\text{rv}}^{(k)}\right], \tag{13}$$

Table 1: Per-task DMC20 returns after 500K environment steps per task.

Task	PPO	DreamerV3	HarmonyDream	TD-MPC2	**MOTO-WM**
Acrobot Swingup	0.07	72.33	209.45	130.52	39.39
Ball in Cup Catch	300.00	978.94	480.38	830.72	953.00
Cartpole Balance	467.91	984.92	993.60	978.66	994.98
Cartpole Balance Sparse	141.20	692.58	965.25	932.72	1000.00
Cartpole Swingup	237.46	833.65	848.53	796.30	835.19
Cartpole Swingup Sparse	0.00	369.22	770.50	615.78	637.42
Cheetah Run	48.04	578.34	774.89	461.84	681.63
Finger Spin	11.60	898.75	487.75	984.47	914.08
Finger Turn Easy	195.40	562.88	695.63	701.51	941.92
Finger Turn Hard	200.00	568.33	120.13	711.34	875.14
Hopper Hop	1.42	65.21	133.34	185.73	227.46
Hopper Stand	0.19	673.25	887.08	820.44	893.65
Pendulum Swingup	0.00	792.58	842.13	826.99	303.17
Quadruped Run	6.25	328.65	395.74	210.20	452.00
Quadruped Walk	196.65	321.62	460.56	191.01	474.42
Reacher Easy	188.40	964.67	967.13	828.61	965.42
Reacher Hard	48.40	33.92	116.50	467.78	697.81
Walker Run	24.66	565.76	581.46	682.43	641.74
Walker Stand	154.20	967.94	958.80	920.85	963.85
Walker Walk	37.64	877.85	871.77	948.49	964.94
Task mean	112.97	606.57	628.03	661.32	**722.86**
Task median	48.22	625.79	733.06	753.82	**855.17**

where the reward, continuation, actor, and critic terms are conditioned on the task embedding e_k, and $\mathcal{L}_{\mathrm{rv}}^{(k)}$ denotes the replay-value objective. A single optimizer updates all shared parameters and task embeddings. The average over tasks follows from the fixed sampling quota rather than an adaptive task-balancing objective.

Per-task return scaling. Because return scales can differ across tasks, MOTO-WM maintains separate return statistics for each task. Let $Q_{5,k}$ and $Q_{95,k}$ denote moving percentiles of imagined λ-returns:

$$S_k = \max(1, Q_{95,k} - Q_{5,k}),$$
$$\widehat{A}_t^{(k)} = \frac{G_t^{\lambda,(k)} - v_\psi(s_t, e_k)}{S_k}. \tag{14}$$

The scaled advantage is used by the shared actor, while reward and value targets remain in their original scales.

Data-parallel execution. Learner updates can be distributed across multiple GPUs using synchronous data parallelism. The global multi-task batch is sharded across devices, which jointly update the same shared model; per-task return statistics are computed from the corresponding samples in the global batch.

6 Experiments

Experimental setup. We evaluate MOTO-WM on 20 visual DeepMind Control tasks (Tassa et al., 2018) and ten state-based Meta-World MT10 tasks (Yu et al., 2020). Within each benchmark, a single model learns all tasks online from scratch. We compare with PPO (Schulman et al., 2017), DreamerV3 (Hafner et al., 2025), HarmonyDream (Ma et al., 2024), and TD-MPC2 (Hansen et al., 2024), with DreamerV3 serving as the primary task-specific specialist reference. Full benchmark, training, and network details are provided in Appendices B–E (Figure 6; Tables 8–13).

Table 2: Task-mean success rates on Meta-World MT10.

Task	DreamerV3	HarmonyDream	TD-MPC2	MOTO-WM(Ours)
Task mean	98.67%	70%	85.4%	87%

Table 3: Ablation results aggregated over 20 DMC tasks at 500K environment steps per task.

Metric	SharedDreamer	SharedEmbedding	Feedback StopGrad	Full Path	Ours
Mean	465.41	554.89	596.37	699.16	**722.86**
Median	432.06	499.62	600.26	745.59	**855.17**

Multi-task performance. Table 1 summarizes DMC20 returns after 500K environment steps per task. One shared MOTO-WM model outperforms the aggregate of 20 DreamerV3 specialists in both task mean and median (Figure 1c). PPO supplies a model-free reference, while DreamerV3, HarmonyDream, and TD-MPC2 provide model-based comparisons. Results with dispersion and learning curves appear in Appendix F (Table 14; Figure 7). We additionally evaluate smaller-capacity variants, with detailed results provided in Appendix I. On MT10 (Table 2), MOTO-WM attains an 87% task-mean success rate. Its performance is below that of separately trained single-task DreamerV3 agents, but is comparable to the multi-task baseline TD-MPC2. Appendix G reports per-task returns and evaluation provenance (Table 15) and learning curves (Figure 8).

Training efficiency and multi-GPU scaling. MOTO-WM uses 1.48 GPU-days at a parameter scale comparable to one DreamerV3 agent (Table 4). Synchronous two-GPU training reduces wall-clock time from 35.52 to 22.57 hours , at a total cost of 1.88 GPU-days. Thus, lower elapsed time does not imply lower total GPU cost. Appendix H provides the corresponding learning curves (Figure 9).

Architectural ablations. We compare four variants (Table 3; Appendix J, Tables 17–18 and Figure 11). *SharedDreamer* retains the multi-task action interface and collection scheme but omits explicit task conditioning. *SharedEmbedding* maps every task ID to one shared embedding. Both perform worse than the full model, supporting objective differentiation in this setting. *Feedback StopGrad* blocks reward and continuation losses from updating the shared RSSM representation; its lower aggregate returns suggest that feedback supervision helps shape that representation. *Full Path* conditions the predictive backbone as well as downstream heads. It performs relatively well but does not improve on downstream-only conditioning. These comparisons support the chosen conditioning scope on DMC20; they do not establish that it is optimal for every task family.

One checkpoint, different task-dependent behaviors. We analyze a frozen seed checkpoint through Walker task-ID interventions (Appendix K). Without parameter updates, matched-task returns are 944.49/950.76/623.96 for *stand*/*walk*/*run*. Switching IDs at identical simulator and model states produces different continuations (Figure 4(a)), with late-episode velocities near 0, 2.27–2.29, and 5.08–5.33 m/s. Appendix K gives the return (Table 19) and all six switches (Figures 13–14).

Shared states, task-conditioned readouts, and prediction. Changing only the task ID leaves the inferred state unchanged under fixed histories and randomness, while downstream readouts differ. At 960 fixed points, task-conditioned reward MSE is 0.00719 versus a task-blind lower bound of 0.32592, supporting objective-specific prediction from shared states. At 15 decisions, Walker pixel MSE is 0.0188/0.0221 under recorded/imagined actor actions (Figure 4(b)), while Figure 5 provides complementary visual predictions through action index 15 after context ending at index 4. Appendix K reports invariance, calibration, and task-information diagnostics (Tables 20–25; Figures 12–16), and Appendix L reports reward errors (Figure 17; Table 26) and all 20-task visual examples (Figures 18–21). These results characterize task-conditioned readouts and short-horizon prediction fidelity for the evaluated checkpoint, rather than long-horizon stability or the unique necessity of a particular conditioning architecture.

Table 4: Model size and total training cost on DMC20 at 500K steps per task.

Metric	TD-MPC2	DreamerV3	HarmonyDream	Ours
Parameters (M)	5.17	10.49	15.60	**10.55**
GPU-days	3.24	5.49	4.74	**1.48**

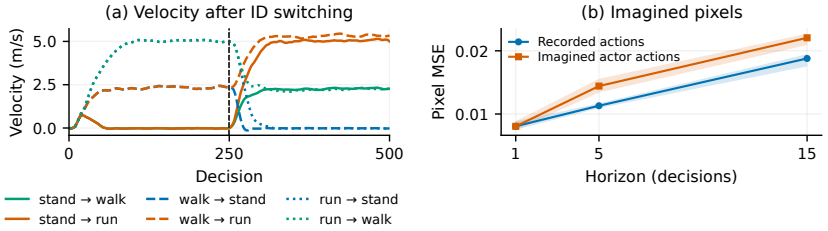


Figure 4: Task-dependent behavior and short-horizon visual prediction. (a) Velocity after task-ID switches at decision 250 over 20 paired starts. (b) Walker pixel MSE under recorded and imagined actor actions. Shading shows 95% intervals over 20 and four initial-state pairs, respectively, for one training seed.

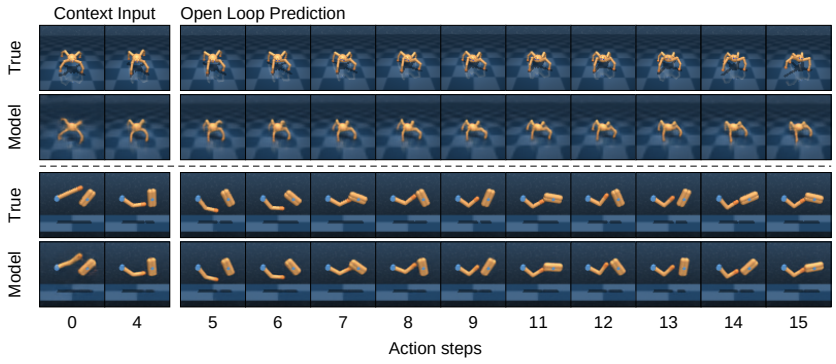


Figure 5: Open-loop predictions on Quadruped Walk and Finger Spin. Context is shown at action indices 0 and 4, followed by predictions through index 15 without observation correction. True and Model denote environment observations and model predictions. The horizontal coordinate is the sequence index, not forecast distance from the last context frame.

7 Conclusion

We presented MOTO-WM, a compact framework for learning multiple tasks online with one shared world model while preserving objective-specific feedback and control. On 20 visual DeepMind Control tasks at 500K environment steps per task, MOTO-WM achieves higher aggregate performance than the task-specific DreamerV3 and HarmonyDream baselines and the multi-task TD-MPC2 baseline, while requiring only 1.48 GPU-days of training. A single 10.55M-parameter model serves all 20 tasks, avoiding the need to train separate task-specific agents.

Experiments on Meta-World, architectural ablations, model-capacity studies, task-ID interventions, and open-loop prediction further validate the shared multi-task design and characterize its behavior. Multi-GPU training also reduces wall-clock time through synchronous data parallelism. Our current study is limited by the explored model scale and by training DMC and Meta-World separately. Future work will investigate larger shared models, joint learning across heterogeneous task suites, and deployment in real-world robotic control.

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. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

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 *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

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. Featured Certification.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first international conference on machine learning*, 2024.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model. *CoRR*, abs/2506.21539, June 2025. URL https://doi.org/10.48550/arXiv.2506.21539.

Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International conference on machine learning*, pp. 4956–4975. PMLR, 2022.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, et al. Understanding world or predicting future? a comprehensive survey of world models. *ACM Computing Surveys*, 58(3):1–38, 2025.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3480–3491. PMLR, 18–24 Jul 2021.

Ignat Georgiev, Varun Giridhar, Nick Hansen, and Animesh Garg. Pwm: Policy learning with multi-task world models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 13737–13757, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/240ea1741b205ea295721d55184ac43b-Paper-Conference.pdf.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Nick Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. In *International Conference on Learning Representations*, volume 2026, pp. 5048–5091, 2026.

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

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Romain Laroche and Merwan Barlier. Transfer reinforcement learning with shared dynamics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. doi: 10.1609/aaai.v31i1.10796.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with blockwise ringattention. In *International Conference on Learning Representations*, volume 2025, pp. 45953–45977, 2025.

Haoyu Ma, Jialong Wu, Ningya Feng, Chenjun Xiao, Dong Li, Jianye Hao, Jianmin Wang, and Mingsheng Long. Harmonydream: Task harmonization inside world models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Yuan Pu, Yazhe Niu, Jia Tang, Junyu Xiong, Shuai Hu, and Hongsheng Li. One model for all tasks: Leveraging efficient world models in multi-task planning. In *International Conference on Learning Representations*, volume 2026, pp. 47870–47924, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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 Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22591–22612. PMLR, 17–23 Jul 2022.

Zhou Xinning, Ying Chengyang, Hao Zhongkai, Su Hang, Liu Songming, Yan Dong, and Zhu Jun. Task aware dreamer for task generalization in reinforcement learning. *SCIENCE CHINA Information Sciences*, 2023.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100, 2020.

Qingzhang Zeng, Peixi Peng, Hang Li, Luntong Li, and Yonghong Tian. Taskloom: Weaving knowledge across tasks in world models. In *Forty-third International Conference on Machine Learning*, 2026.

Boxuan Zhang, Weipu Zhang, Zhaohan Feng, Wei Xiao, Jian Sun, Jie Chen, and Gang Wang. Mixture-of-world models: Scaling multi-task reinforcement learning with modular latent dynamics. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qUQARlAx5y.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: Efficient stochastic transformer based world models for reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=WxnrX42rnS.

AI USE STATEMENT

In this work, we used generative AI tools for design or provide feedback on research methodology or experiments, assist with translation.

We have not used generative AI tools for help develop theoretical models or conceptual frameworks, formulate mathematical claims, assist in the writing of proofs, propose or refine hypotheses, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. And generating synthetic data sets is not applicable to this work.

Additionally, we used generative AI tools for create or modify scientific figures or images, create or edit software code and identify relevant literature.

We have reviewed all AI-assisted work. An author carefully reviewed the experimental script that was designed with the help of AI tools. For content involving AI-assisted translation, such as technical terms, an author checked relevant literature to ensure appropriate wording. AI-assisted modifications to scientific figures were also reviewed by an author to verify the accuracy of the data. After the project was completed, AI tools were used to help reorganize and simplify the project code, reduce memory usage, and improve readability, with the final code repository reviewed by an author. AI tools were also used to help check references for possible errors, and the results were verified by an author.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The experimental setup and evaluation protocols are detailed in Appendix C, with training hyperparameters and network configurations provided in Appendices D and E. Procedures for the motivation diagnostics, task-conditioned behavior and representation analyses, and short-horizon prediction checks are described in Appendices A, K, and L, including the evaluation settings, diagnostic definitions, and aggregation conventions.

The supplementary material submitted with this paper includes the essential scripts for MOTO-WM and a lightweight checkpoint, reduced in size and compressed to meet the 100 MB submission limit.

APPENDIX

The appendix first presents the motivation diagnostics, evaluation environments, experimental setup, training hyperparameters, and network parameters. It then follows the main experiments with per-task DMC20 and MT10 results, GPU scaling, capacity and architecture ablations, task-conditioned behavior and representation analyses, and short-horizon prediction checks.

A MOTIVATION DIAGNOSTICS

Checkpoints and paired evaluation. The motivation analysis uses three independently trained seed-0 DreamerV3 checkpoints for Walker *stand*, *walk*, and *run*, each labeled 500K by its original training counter. Compatibility loading leaves the checkpoint parameters unchanged; the diagnostic does not independently reconstruct the original training runs. Each controller is evaluated from 20 paired initial states for 500 action decisions (25 seconds), with 64×64 RGB observations, six effective action dimensions, and action repeat two. Rewards are summed across both native substeps, so decision rewards lie in $[0, 2]$ and episode returns in $[0, 1000]$.

Each physical trajectory is rescored under all three reward definitions without changing its controller (Table 5). The stand reward measures height and upright posture; it does not require zero velocity. Walk saturates at a target speed of 1 m/s, whereas run targets 8 m/s. Consequently, high off-diagonal returns can reflect compatible reward rules, not successful execution of a different instruction. Figure 2(b) shows unsmoothed velocities at all 501 physical-state time points. Intervals use 2,000 cluster-bootstrap resamples of the 20 initial-state pairs (seed 20260916); they describe evaluation variation, not uncertainty across training seeds.

Table 5: Cross-reward evaluation of the three native DreamerV3 controllers. Rows denote reward rules and columns denote history-generating controllers.

Rule / Controller	Stand	Walk	Run
Stand	975.75	977.01	929.03
Walk	228.99	966.65	916.45
Run	171.17	349.58	741.94

Common histories and representational similarity. The readout bank contains $3 \times 20 \times 16 = 960$ points: three controllers, 20 paired starts, and endpoints at decisions 1, 34, 67, 100, 133, 167, 200, 233, 266, 299, 333, 366, 399, 432, 465, and 499. Every reader replays the same complete observation–action prefixes with native recurrent carries and posterior inference. Recorded pre-clipping actions enter the recurrent replay, whereas the environment executes clipped actions. Each model infers its own state; latent states are never exchanged between checkpoints.

Linear CKA, defined in Eq. 7, is computed without projection on h (2048 dimensions), flattened sampled z (512), and the last hidden layer of each head (256). Table 6 provides the values underlying Figure 2(c). Similarity in h is high across checkpoint pairs, but head similarity is heterogeneous; for example, stand–run reward CKA is also high. These comparisons concern aligned representations, not causal necessity or interchangeable network weights.

Table 6: Cross-model linear CKA on the aligned history set.

Task pair	h	z	Rew	Con	Act	Val
Stand–Walk	0.8759	0.2662	0.5162	0.6806	0.6391	0.5832
Stand–Run	0.8323	0.2125	0.8159	0.6611	0.5404	0.5533
Walk–Run	0.8998	0.1909	0.6838	0.6551	0.6838	0.4997

Physical-variable probes. We decode velocity, height, and upright posture from a fixed 32-dimensional orthogonal random projection of h (projection seed 2000). Five folds group all points

by their initial-state pair: four pairs test, four validate, and twelve fit, using a split generated with seed 20260916. The ridge penalty is selected from $\{0.01, 0.1, 1, 10, 100\}$ by normalized validation MSE. Feature scaling and target normalization are fitted only on training data; after selection the model is refitted on non-test pairs. Table 7 reports held-out RMSE in original physical units and R^2 across all 960 points, with pair-bootstrap RMSE intervals. These results support shared physical content in the deterministic state; they do not establish task-invariant information on arbitrary trajectories.

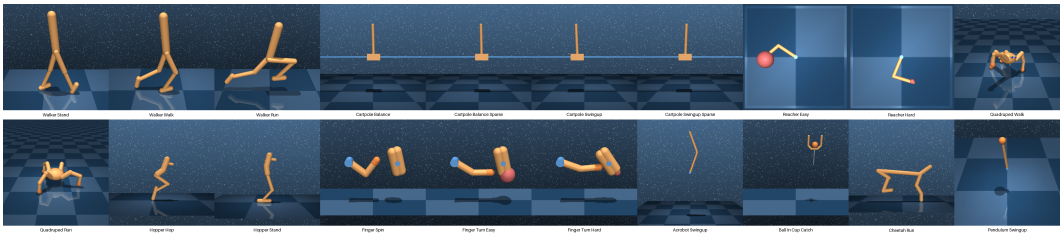
Table 7: Physical-variable decoding from the deterministic recurrent state h.

Model	Target	RMSE	95% pair interval	R^2
Stand	velocity (m/s)	0.9729	[0.9137, 1.0291]	0.8662
Stand	height (m)	0.0681	[0.0576, 0.0790]	0.5454
Stand	upright	0.2450	[0.2189, 0.2683]	0.4537
Walk	velocity (m/s)	0.9325	[0.8748, 0.9866]	0.8771
Walk	height (m)	0.0686	[0.0577, 0.0795]	0.5392
Walk	upright	0.2074	[0.1875, 0.2264]	0.6085
Run	velocity (m/s)	0.8156	[0.7727, 0.8580]	0.9060
Run	height (m)	0.0694	[0.0568, 0.0817]	0.5283
Run	upright	0.2233	[0.2034, 0.2421]	0.5463

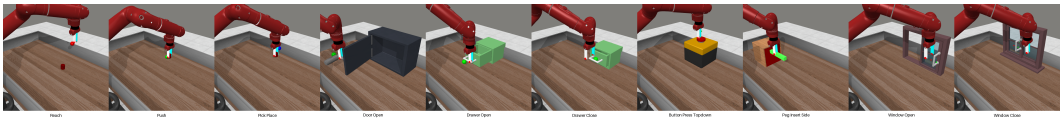
Scope. The evidence motivates sharing predictive state while differentiating downstream objectives; it does not establish that our conditioning architecture is uniquely necessary. The three checkpoints provide one training seed per task. Walker's nonterminal samples also provide no positive evidence for task-dependent continuation targets. Matched-training architectural comparisons are reported separately in Appendix J.

B Evaluation environments

DMC20 uses visual observations from the 20 tasks listed in Table 14; Meta-World MT10 uses 39-dimensional state observations for the ten manipulation tasks listed in Table 15. Figure 6 gives an overview of their physical settings.



(a) DMC Visual Control 20 Tasks



(b) MetaWorld MT10 Tasks

Figure 6: Evaluation benchmarks: (a) 20 visual DMC tasks and (b) 10 state-based Meta-World MT10 tasks.

C Experimental setup

One shared model is trained online from scratch within each benchmark. Table 8 separates total action decisions from nominal environment steps. The latter multiply decisions by action repeat; the

per-task counts divide the total budget by the number of tasks. Reset transitions and evaluation are not training interactions.

DreamerV3 is the task-specific specialist reference. Main DMC20 comparisons use a nominal 500K environment steps per task. The MT10 return table reports a distinct evaluation protocol, including a pretrained TD-MPC2 checkpoint; its caption retains the supplied seed-selection and evaluation details. The single-checkpoint Walker analyses have their own paired-start protocols (Appendices K and L); their confidence intervals must not be read as training-seed uncertainty.

Table 8: Training budgets and evaluation settings. Environment-step counts are nominal counts obtained by multiplying action steps by the action repeat.

Training/evaluation setting	DMC20	Meta-World MT10
Total training action steps	5,000,000	5,000,000
Action repeat	2	2
Nominal total env. steps	10,000,000	10,000,000
Action steps/task	250,000	500,000
Nominal env. steps/task	500,000	1,000,000
Parallel envs./task	4	4
Total parallel environments	80	40
Parallel environment execution	True	True
Training GPUs	1	1
Evaluation interval (action steps)	100,000	100,000
Evaluation episodes/task	4	10
Evaluation action selection	Stochastic sampling	Stochastic sampling
Evaluation seed offset	100,000	100,000
Meta-World benchmark seed	–	0

D TRAINING HYPERPARAMETERS

The settings below apply to the main experiments, except for the explicitly changed component in a capacity or architecture ablation. Table 9 specifies replay and optimization; Table 10 gives world-model and prediction-loss weights; Table 11 describes imagined and replay actor–critic training. Network sizes are listed separately in Appendix E.

Table 9: Replay and optimizer settings.

Setting	DMC20	MT10
Batch size	20	40
Sequences/task/batch	1	4
Training sequence length	64	64
Replay context	1	1
Sample length (with context)	65	65
Replay capacity/task	250,000	50,000
Total replay capacity	5,000,000	500,000
Replay chunk	1024	1024
Online replay	True	True
Task sampling	Equal quota/batch	Equal quota/batch
Replay ratio	256	256
Learning rate	1×10^{-4}	4×10^{-5}
LR warmup	1000 updates	1000 updates
Post-warmup LR	Constant	Constant
LR annealing	None	None
Optimizer	LaProp	LaProp
First moment	0.9	0.9

Table 9 (continued)

Setting	DMC20	MT10
Second moment	0.999	0.999
Optimizer ϵ	10^{-20}	10^{-20}
Adaptive grad. clipping	AGC (0.3)	AGC (0.3)
Weight decay	0	0

Table 10: World-model and prediction-loss settings shared by both benchmarks.

Setting	Value
KL free nats	1.0
Reconstruction loss	1.0
Dynamics KL loss	1.0
Representation KL loss	0.1
Reward prediction loss	1.0
Continue prediction loss	1.0

Table 11: Actor–critic training settings shared by both benchmarks.

Setting	Value
Imagination horizon	15
Discount horizon	333
Discount factor	$1 - 1/333 \approx 0.996997$
Return lambda	0.95
Actor loss weight	1.0
Critic loss weight	1.0
Replay critic loss weight	0.3
Actor entropy coefficient	3×10^{-4}
Critic EMA regularization	1.0
Replay critic EMA regularization	1.0
Critic EMA decay	0.98
Critic EMA update rate	0.02
Critic EMA interval	Every update
Return norm. scope	Per-task
Return norm. lower percentile	5%
Return norm. upper percentile	95%
Return norm. scale	$P_{95}(R_T) - P_5(R_T)$
Return norm. scale floor	1.0
Return norm. EMA decay	0.99
Return norm. bias correction	False
Value normalization	None
Extra advantage normalization	None

Here P_q denotes percentile q. Return-scale statistics are maintained separately for each task; this changes the scope of the statistics, not the definition of the underlying Dreamer-style normalization.

E NETWORK PARAMETERS

Both benchmarks use the `size12m` preset, SiLU activations, RMSNorm, and `bfloat16` computation. Preset names are not exact parameter counts: the main DMC20 model contains 10.55M parameters, while input and output modules depend on the benchmark. Tables 12 and 13 separate the shared predictive state from downstream head architectures.

Table 12: Shared recurrent state and latent-distribution architecture.

Component	Value
Hidden width	256
RSSM deterministic dimension	2048
Stochastic categorical variables	32
Categories/variable	16
Flattened stochastic dimension	512
World-model feature dimension	2560
RSSM recurrent blocks	8
RSSM dynamics hidden layers	1
Prior hidden layers	2
Posterior hidden layers	1
Prior/posterior hidden width	256
Latent unimix	0.01

Observation encoding and reconstruction. DMC20 receives $64 \times 64 \times 3$ images, preprocessed as $x/255 - 0.5$. Its four-stage CNN encoder uses base width 16 with channel multipliers $[2, 3, 4, 4]$, giving channels $[32, 48, 64, 64]$; each stage uses a 5×5 kernel and 2×2 max-pooling. The encoder output has 1024 dimensions. A CNN decoder reconstructs images using image MSE. MT10 instead receives a 39-dimensional state vector with symlog preprocessing. Its encoder and decoder are three-hidden-layer, width-256 MLPs; the encoder produces a 256-dimensional vector, and reconstruction uses symlog MSE.

Task representation and unified actions. A learnable task lookup table has 32 columns: 20×32 (640 parameters) for DMC20 and 10×32 (320 parameters) for MT10, initialized with a scaled truncated normal. Its output conditions reward, continuation, actor, and critic, but is not explicitly supplied to the observation encoder or RSSM. Concatenating it with $[h_t, z_t] \in \mathbb{R}^{2560}$ gives a 2592-dimensional downstream input.

Actions are padded to 16 dimensions, with invalid coordinates handled by the task-specific mask. The action encoder is $16 \to 64 \to 256$: the first linear layer is followed by SiLU, and the second by RMSNorm and SiLU. This shared action pathway does not receive the task embedding.

Table 13: Task-conditioned head architectures and output distributions.

Component	Value
Reward predictor hidden layers	1
Reward predictor hidden width	256
Continue predictor hidden layers	1
Continue predictor hidden width	256
Actor hidden layers	3
Actor hidden width	256
Critic hidden layers	3
Critic hidden width	256
Reward output distribution	Symexp two-hot
Reward bins	255
Continue output distribution	Bernoulli
Critic output distribution	Symexp two-hot
Critic bins	255
Continuous policy distribution	Bounded normal
Minimum policy std.	0.1
Maximum policy std.	1.0

F DMC20 RESULTS

Table 1 in the main text contains all five methods' per-task point estimates. Table 14 adds the reported dispersion for the three methods for which it was supplied; Figure 7 gives the corresponding MOTO-WM learning curves.

Table 14: DMC20 returns with reported dispersion. PPO and TD-MPC2 point estimates are in Table 1; emphasis follows the best result across all five methods, not only the columns shown here.

Task	DreamerV3	HarmonyDream	MOTO-WM
Acrobot Swingup	72.33 ± 30.11	209.45 ± 106.79	39.39 ± 19.87
Ball in Cup Catch	978.94 ± 7.38	480.38 ± 679.35	953.00 ± 14.49
Cartpole Balance	984.92 ± 5.58	993.60 ± 1.02	$\mathbf{994.98 \pm 0.52}$
Cartpole Balance Sparse	692.58 ± 532.46	965.25 ± 49.14	$\mathbf{1000.00 \pm 0.00}$
Cartpole Swingup	833.65 ± 16.29	848.53 ± 33.43	835.19 ± 53.83
Cartpole Swingup Sparse	369.22 ± 354.23	770.50 ± 50.20	637.42 ± 276.33
Cheetah Run	578.34 ± 29.21	774.89 ± 18.16	681.63 ± 25.56
Finger Spin	898.75 ± 43.13	487.75 ± 152.38	914.08 ± 35.28
Finger Turn Easy	562.88 ± 223.27	695.63 ± 65.23	$\mathbf{941.92 \pm 44.17}$
Finger Turn Hard	568.33 ± 268.11	120.13 ± 169.88	$\mathbf{875.14 \pm 86.42}$
Hopper Hop	65.21 ± 92.22	133.34 ± 19.03	$\mathbf{227.46 \pm 58.97}$
Hopper Stand	673.25 ± 131.59	887.08 ± 14.19	$\mathbf{893.65 \pm 44.07}$
Pendulum Swingup	792.58 ± 29.81	842.13 ± 65.94	303.17 ± 525.10
Quadruped Run	328.65 ± 179.53	395.74 ± 12.26	$\mathbf{452.00 \pm 12.13}$
Quadruped Walk	321.62 ± 51.65	460.56 ± 133.61	$\mathbf{474.42 \pm 14.94}$
Reacher Easy	964.67 ± 2.16	967.13 ± 7.95	965.42 ± 6.29
Reacher Hard	33.92 ± 56.38	116.50 ± 160.87	$\mathbf{697.81 \pm 70.91}$
Walker Run	565.76 ± 59.82	581.46 ± 48.78	641.74 ± 12.62
Walker Stand	967.94 ± 3.94	958.80 ± 21.09	963.85 ± 16.63
Walker Walk	877.85 ± 124.59	871.77 ± 28.93	$\mathbf{964.94 \pm 8.43}$
Task mean	606.57	628.03	**722.86**
Task median	625.79	733.06	**855.17**

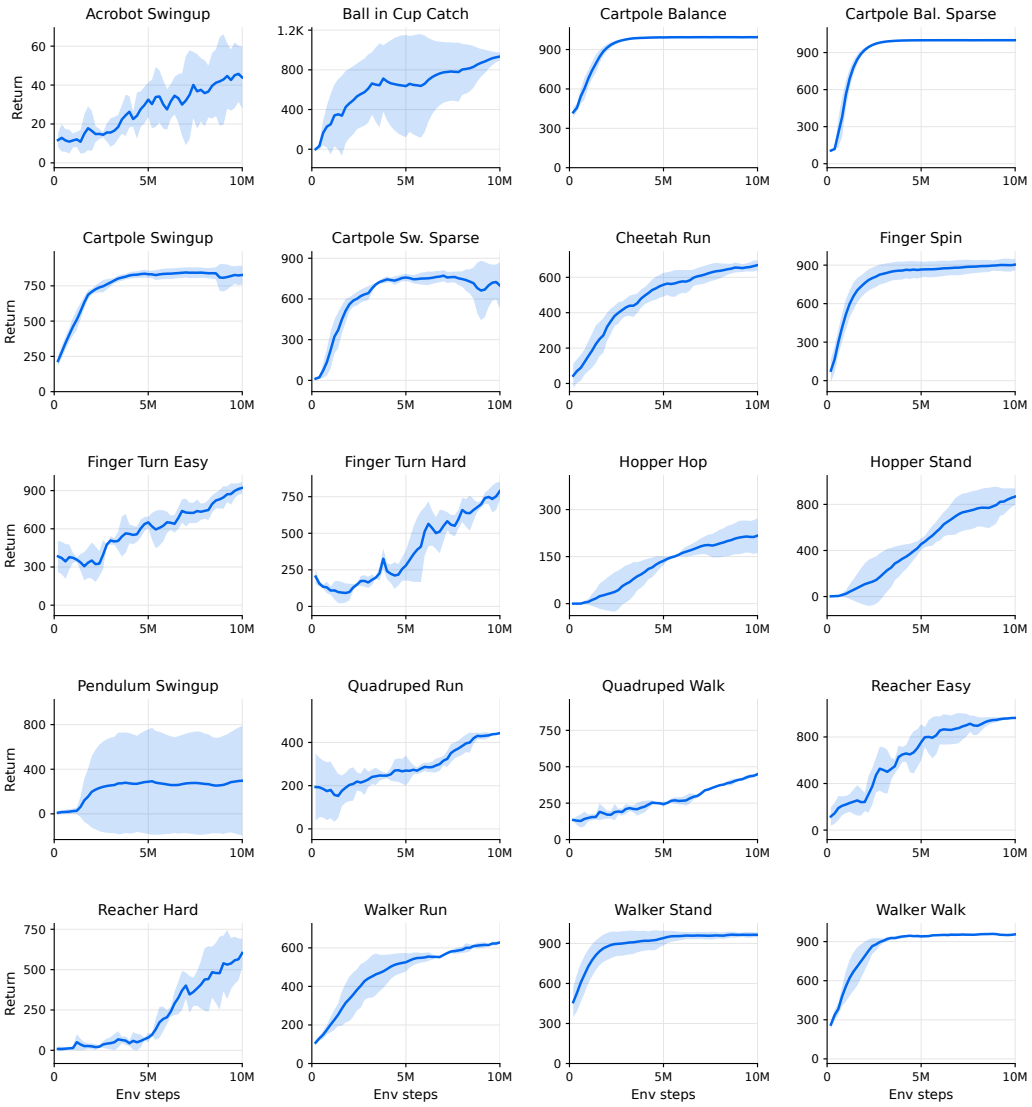


Figure 7: Per-task learning curves of MOTO-WM on the 20 DMC tasks.

G META-WORLD MT10 RESULTS

Table 15: Per-task returns on Meta-World MT10. Ours uses processed results with seeds selected by final success rate, whereas DreamerV3 and HarmonyDream use raw results. TD-MPC2 uses the mt80-5M checkpoint with 50 evaluation episodes per task, each containing 250 decision steps (500 environment steps).

Task	DreamerV3	HarmonyDream	TD-MPC2	MOTO-WM
Reach	4866.01	4803.98	4675.10	3411.56
Push	4520.41	17.98	3136.85	3998.10
Pick Place	2974.60	87.34	2782.11	1208.96
Door Open	4556.13	4475.96	4341.06	4107.96
Drawer Open	4786.53	4701.94	3041.91	3979.33

Table 15 (continued)

Task	DreamerV3	HarmonyDream	TD-MPC2	MOTO-WM
Drawer Close	4855.24	4733.11	3598.43	4849.15
Button Press Topdown	3621.73	3389.08	3826.21	3195.04
Peg Insert Side	4386.67	7.89	4097.14	3252.53
Window Open	4539.73	4454.58	4569.95	4068.52
Window Close	4593.44	4440.25	4431.11	3925.31
Task mean	4370.05	3111.21	3849.98	3599.65

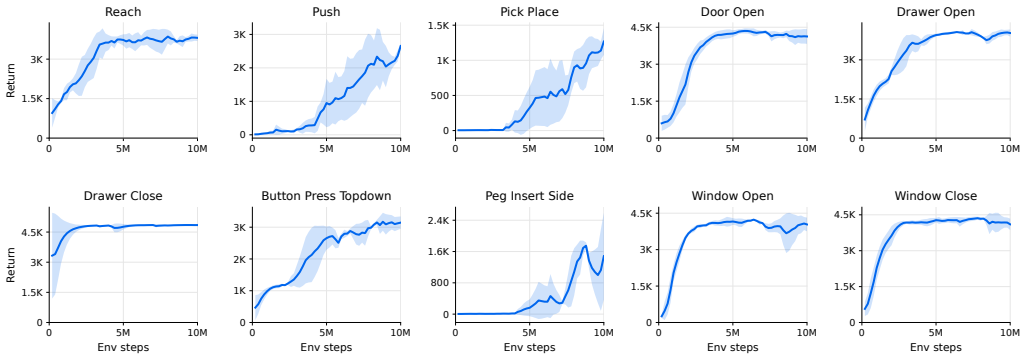


Figure 8: Per-task learning curves of MOTO-WM on the ten Meta-World MT10 tasks.

H SINGLE- AND DUAL-GPU SCALING

Figure 9 compares learning under the same interaction budget; the main text reports elapsed time and GPU cost.

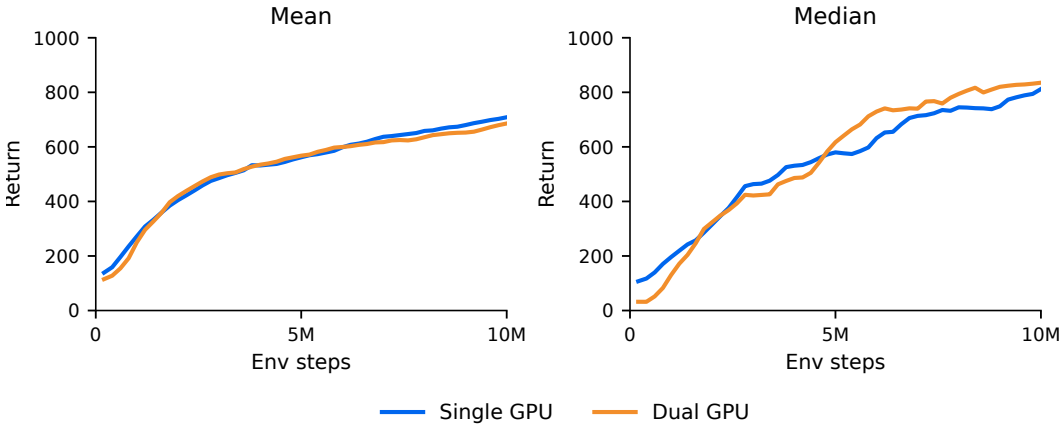


Figure 9: Aggregate mean and median DMC20 learning curves for single- and dual-GPU MOTO-WM training under the same interaction budget. Curves are plotted against environment steps; dual-GPU execution primarily reduces wall-clock training time rather than changing the interaction budget.

I MODEL CAPACITY

Table 16: Processed per-task DMC20 returns for the 10.55M- and 0.81M-parameter MOTO-WM configurations (`size12m` and `size1m` presets) after 10M total environment steps (500K per task).

Task	MOTO-WM (0.81M)	MOTO-WM (10.55M)
Acrobot Swingup	26.46 ± 3.87	39.39 ± 19.87
Ball in Cup Catch	748.92 ± 124.34	953.00 ± 14.49
Cartpole Balance	988.60 ± 1.59	994.98 ± 0.52
Cartpole Balance Sparse	994.00 ± 10.39	1000.00 ± 0.00
Cartpole Swingup	826.85 ± 51.52	835.19 ± 53.83
Cartpole Swingup Sparse	496.25 ± 245.19	637.42 ± 276.33
Cheetah Run	277.50 ± 87.93	681.63 ± 25.56
Finger Spin	552.75 ± 33.59	914.08 ± 35.28
Finger Turn Easy	376.75 ± 87.27	941.92 ± 44.17
Finger Turn Hard	150.00 ± 156.81	875.14 ± 86.42
Hopper Hop	131.39 ± 101.18	227.46 ± 58.97
Hopper Stand	471.05 ± 213.30	893.65 ± 44.07
Pendulum Swingup	24.83 ± 22.95	303.17 ± 525.10
Quadruped Run	127.94 ± 126.29	452.00 ± 12.13
Quadruped Walk	212.53 ± 33.76	474.42 ± 14.94
Reacher Easy	381.92 ± 130.97	965.42 ± 6.29
Reacher Hard	7.25 ± 9.69	697.81 ± 70.91
Walker Run	335.15 ± 113.64	641.74 ± 12.62
Walker Stand	933.24 ± 31.21	963.85 ± 16.63
Walker Walk	890.06 ± 53.05	964.94 ± 8.43
Task mean	447.67	722.86
Task median	379.33	855.17

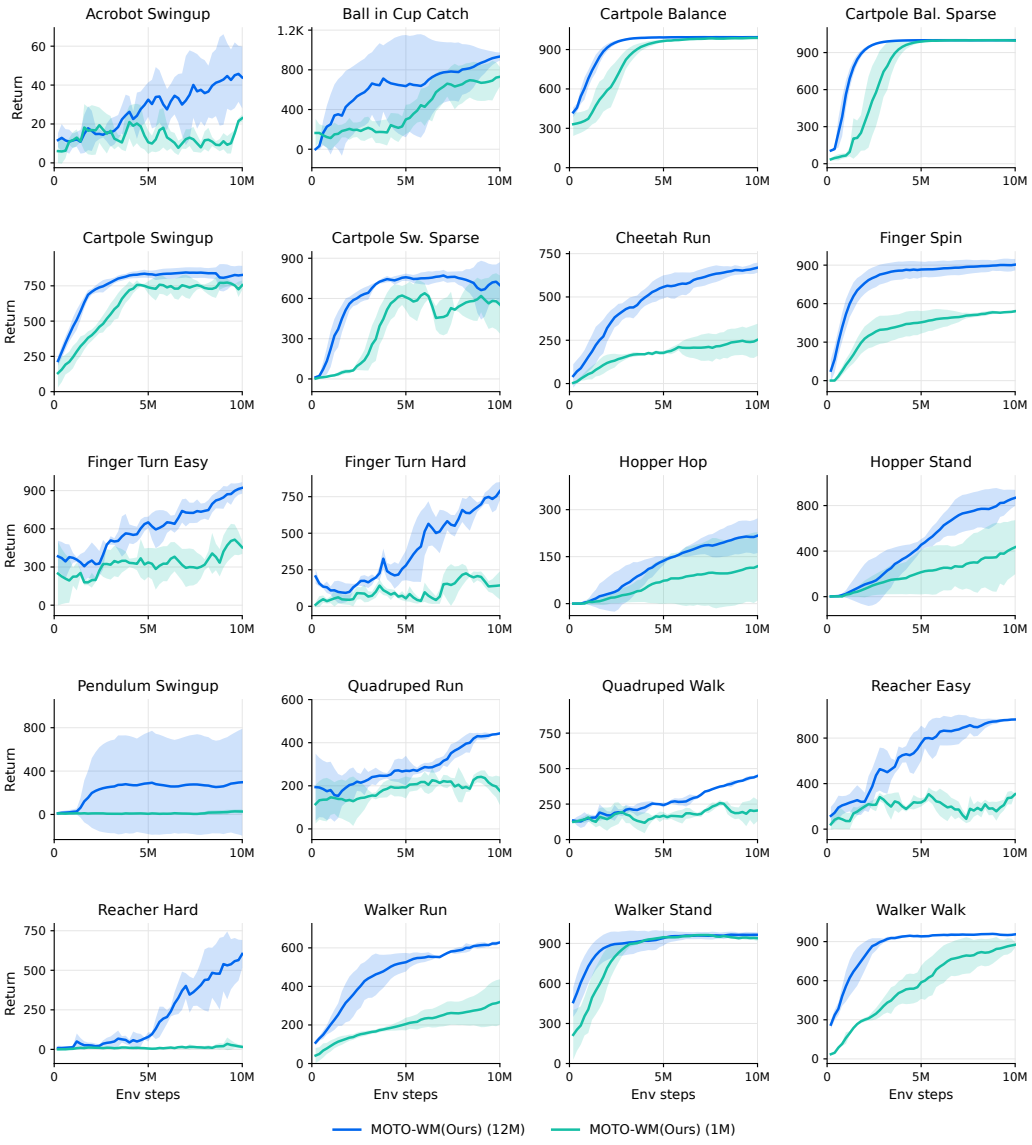


Figure 10: Per-task learning curves of MOTO-WM with the 12M and 1M size presets on the 20 DMC tasks.

J DMC20 ABLATIONS

Tables 17 and 18 split the five-variant comparison into two panels to keep every reported mean and dispersion on one line. MOTO-WM is repeated as a common reference; bold denotes its results that are best among all five variants.

Table 17: Per-task DMC20 ablations: shared-model and shared-embedding variants.

Task	SharedDreamer	SharedEmbedding	MOTO-WM
Acrobot Swingup	20.21 ± 21.25	20.58 ± 13.72	39.39 ± 19.87
Ball in Cup Catch	919.13 ± 4.42	948.25 ± 13.44	953.00 ± 14.49
Cartpole Balance	864.61 ± 2.44	993.75 ± 1.67	$\mathbf{994.98 \pm 0.52}$

Table 17 (continued)

Task	SharedDreamer	SharedEmbedding	MOTO-WM
Cartpole Balance Sparse	897.63 ± 103.41	1000.00 ± 0.00	$\mathbf{1000.00 \pm 0.00}$
Cartpole Swingup	572.70 ± 154.22	806.03 ± 57.79	835.19 ± 53.83
Cartpole Swingup Sparse	615.00 ± 49.14	743.50 ± 15.20	637.42 ± 276.33
Cheetah Run	408.11 ± 64.10	478.25 ± 84.95	$\mathbf{681.63 \pm 25.56}$
Finger Spin	379.13 ± 62.40	340.50 ± 53.03	$\mathbf{914.08 \pm 35.28}$
Finger Turn Easy	429.00 ± 89.45	653.88 ± 100.94	$\mathbf{941.92 \pm 44.17}$
Finger Turn Hard	222.25 ± 243.24	521.00 ± 148.14	$\mathbf{875.14 \pm 86.42}$
Hopper Hop	4.45 ± 6.30	88.42 ± 56.90	227.46 ± 58.97
Hopper Stand	99.55 ± 139.82	421.94 ± 85.89	$\mathbf{893.65 \pm 44.07}$
Pendulum Swingup	25.00 ± 35.36	397.63 ± 511.41	303.17 ± 525.10
Quadruped Run	435.13 ± 44.73	367.68 ± 53.62	452.00 ± 12.13
Quadruped Walk	689.83 ± 131.06	410.51 ± 3.44	474.42 ± 14.94
Reacher Easy	724.50 ± 3.89	847.88 ± 186.50	965.42 ± 6.29
Reacher Hard	1.00 ± 1.41	0.13 ± 0.18	$\mathbf{697.81 \pm 70.91}$
Walker Run	295.13 ± 5.47	323.44 ± 86.02	641.74 ± 12.62
Walker Stand	862.19 ± 16.99	879.80 ± 51.64	963.85 ± 16.63
Walker Walk	843.71 ± 18.22	854.57 ± 64.74	964.94 ± 8.43
Task mean	465.41	554.89	**722.86**
Task median	432.06	499.62	**855.17**

Table 18: Per-task DMC20 ablations: gradient-path variants. MOTO-WM repeats the reference column of Table 17.

Task	Feedback StopGrad	Full Path	MOTO-WM
Acrobot Swingup	26.32 ± 36.19	46.37 ± 18.24	39.39 ± 19.87
Ball in Cup Catch	954.13 ± 4.42	954.00 ± 0.00	953.00 ± 14.49
Cartpole Balance	992.63 ± 0.30	992.49 ± 1.15	$\mathbf{994.98 \pm 0.52}$
Cartpole Balance Sparse	1000.00 ± 0.00	1000.00 ± 0.00	$\mathbf{1000.00 \pm 0.00}$
Cartpole Swingup	871.91 ± 1.20	871.17 ± 2.10	835.19 ± 53.83
Cartpole Swingup Sparse	781.63 ± 65.23	764.38 ± 52.86	637.42 ± 276.33
Cheetah Run	465.05 ± 43.70	662.49 ± 97.57	$\mathbf{681.63 \pm 25.56}$
Finger Spin	388.38 ± 0.18	685.75 ± 216.73	$\mathbf{914.08 \pm 35.28}$
Finger Turn Easy	500.00 ± 0.00	496.88 ± 4.42	$\mathbf{941.92 \pm 44.17}$
Finger Turn Hard	361.88 ± 158.22	520.63 ± 133.47	$\mathbf{875.14 \pm 86.42}$
Hopper Hop	134.88 ± 21.35	247.44 ± 148.41	227.46 ± 58.97
Hopper Stand	700.52 ± 264.38	813.90 ± 142.09	$\mathbf{893.65 \pm 44.07}$
Pendulum Swingup	729.63 ± 49.67	819.63 ± 110.84	303.17 ± 525.10
Quadruped Run	296.96 ± 98.04	504.74 ± 71.81	452.00 ± 12.13
Quadruped Walk	382.13 ± 151.46	726.81 ± 233.49	474.42 ± 14.94
Reacher Easy	966.88 ± 7.95	912.25 ± 107.13	965.42 ± 6.29
Reacher Hard	19.75 ± 12.37	337.38 ± 460.86	$\mathbf{697.81 \pm 70.91}$
Walker Run	474.88 ± 134.37	682.68 ± 4.58	641.74 ± 12.62
Walker Stand	948.30 ± 37.09	974.75 ± 9.13	963.85 ± 16.63
Walker Walk	931.51 ± 31.82	969.59 ± 11.70	964.94 ± 8.43
Task mean	596.37	699.16	**722.86**
Task median	600.26	745.59	**855.17**

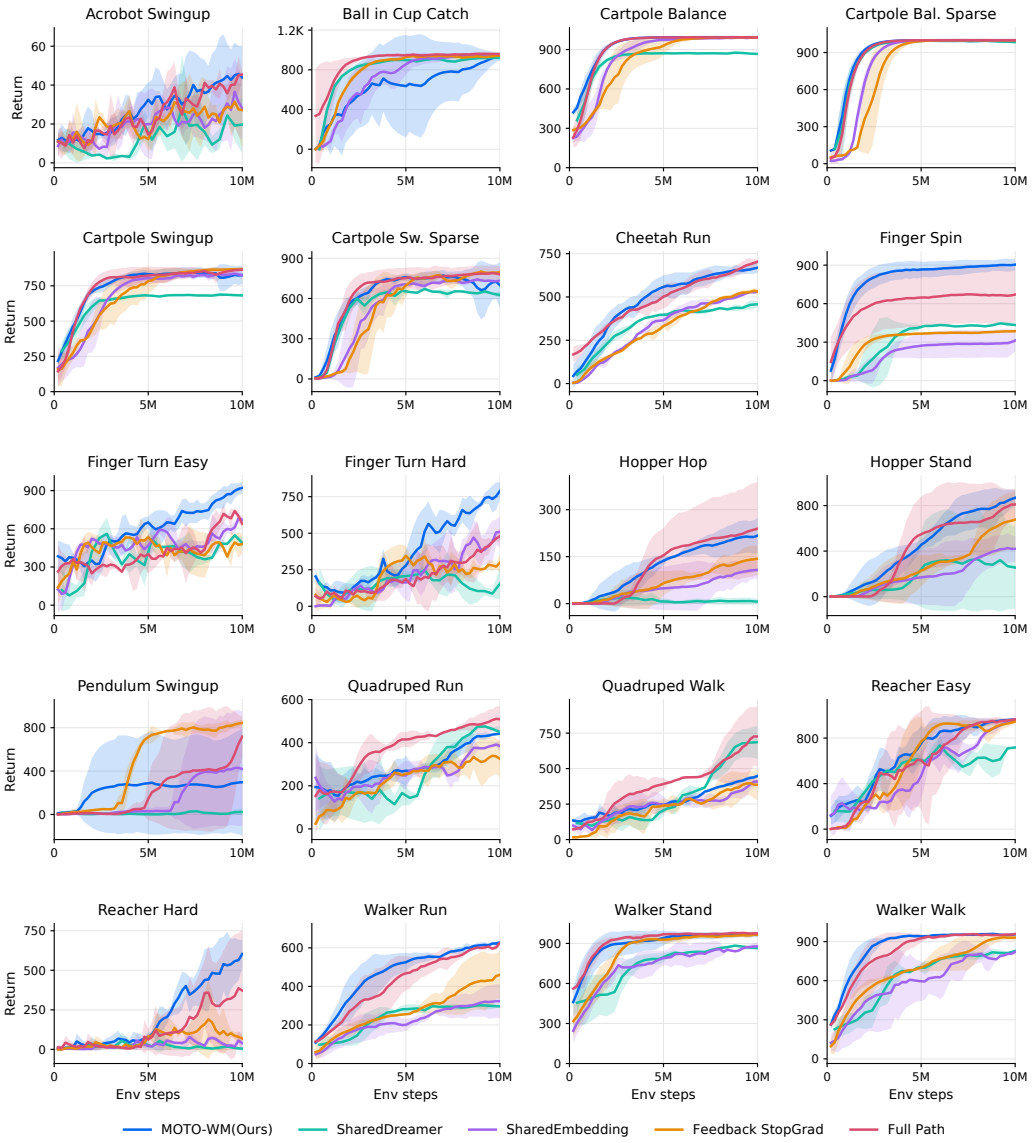


Figure 11: Per-task learning curves for the DMC ablation variants.

K TASK-CONDITIONED BEHAVIOR AND REPRESENTATIONS

Checkpoints and evaluation protocol. All MOTO-WM interventions use one frozen seed-0 checkpoint trained on 20 DMC tasks for five million action decisions. Task identity enters downstream readouts, not the encoder or RSSM transition and posterior networks. Native Dreamer uses three independently trained Walker checkpoints. Shared uses one learnable embedding across 16 tasks and a checkpoint trained for 2.5 million action decisions; it is a descriptive comparison, not a matched-training ablation. No parameters are updated during evaluation. We use 20 paired initial states, 500 decisions (25 seconds), 64×64 RGB observations, and six effective action dimensions. Action repeat is two; summing both substep rewards gives decision rewards in $[0, 2]$ and episode returns in $[0, 1000]$. Each trajectory is rescored under all three reward rules without changing its controller. Stand rewards upright posture without requiring zero speed; walk saturates at 1 m/s and run targets 8 m/s. High off-diagonal returns can therefore reflect compatible rewards.

The MOTO-WM bank contains $20 \times 3 \times 16 = 960$ points from its three task-conditioned controllers. The comparison bank contains $20 \times 7 \times 16 = 2,240$ points from three MOTO-WM conditions, three Dreamer checkpoints, and Shared. Each reader replays identical observation–action histories and infers its own latent state; states are never exchanged between models. Results from these banks are not interchangeable. Unless stated otherwise, 95% cluster-bootstrap intervals resample initial-state pairs, keeping their histories and conditions together. They measure evaluation variability conditional on the available checkpoints, not variation across training seeds.

One checkpoint under three task IDs. Table 19 reports the task-rule-by-input-ID matrix with pair-bootstrap intervals. Episode-mean velocities are 0.023, 2.223, and 4.665 m/s under stand, walk, and run IDs, with identical parameters and action spaces. All 20 starts are retained, including the run-conditioned failure at pair 0 (return 56.55); aggregate performance does not imply success from every start.

Table 19: One checkpoint under three task IDs: mean return and 95% pair-bootstrap interval.

Task rule	stand ID	walk ID	run ID
stand	944.5 [883.8, 978.6]	964.2 [951.1, 975.4]	917.7 [853.9, 953.6]
walk	186.5 [172.6, 196.1]	950.8 [935.9, 963.3]	891.1 [800.5, 940.8]
run	161.8 [151.5, 167.7]	389.1 [381.6, 395.4]	624.0 [561.5, 658.7]

Fixed-history interventions and invariance. At a fixed cached posterior sample $x = (h, z)$, the 960-point bank gives matched reward MAEs of 0.02059/0.07710/0.03146 for stand/walk/run. Actor-mean RMS differences are 0.39168/0.52271/0.40994 for stand–walk/stand–run/walk–run, demonstrating task-dependent readouts at an identical inferred state. For balanced labels, let $r_{j,t}$ be the true reward and $\bar{r}_j = \frac{1}{3}\sum_t r_{j,t}$. Any task-blind scalar prediction f_j satisfies

$$\frac{1}{3N}\sum_{j,t}(f_j - r_{j,t})^2 = \frac{1}{N}\sum_{j}(f_j - \bar{r}_j)^2 + \frac{1}{3N}\sum_{j,t}(r_{j,t} - \bar{r}_j)^2 \geq 0.32592. \tag{15}$$

MOTO-WM's reward MSE is 0.00719 on the same support. This bound isolates ambiguous reward labels at fixed inputs; it neither evaluates Shared's learned representation nor uniquely identifies a conditioning architecture.

The invariance audit fixes observations, actions, initial carries, resets, and random streams. Native-forward checks cover nine full histories under three IDs; diagnostic checks cover posterior logits/probabilities, h, and sampled z at 960 endpoints, including agreement between diagnostic and native h. The exact maximum differences in Table 20 verify the implemented conditioning path, not task independence on naturally different trajectories. Changing task identity at a fixed history isolates readout dependence, whereas comparing trajectories generated under different task IDs also changes actions and subsequent observations. The latter can therefore produce different recurrent states despite the former's exact invariance.

Table 20: Fixed-history invariance audit. Every zero is the recorded maximum, not a rounded average.

Audit	Measured quantity	Maximum difference
Native replay	native h across task IDs	0.0
Native replay	native z_sample across task IDs	0.0
Native replay	native replay h	0.0
Native replay	native replay z_sample	0.0
Native replay	native replay action	0.0
Posterior audit	joint max: posterior logits/probs, diagnostic h/z	0.0
Posterior audit	diagnostic h vs native h	0.0

Mid-episode task switching. At decision 250, each trajectory branches into three IDs while retaining its simulator snapshot and full model carry, without updates or resets. The $20 \times 3 \times 3 = 180$

branches each run another 250 decisions with matched random streams; unchanged-ID branches are checked against cached continuations. Figures 13 and 14 show all six directed switches and unchanged-source controls. Both branches are scored under the destination rule throughout decisions 0–500, avoiding a scoring-induced discontinuity. Reward and velocity use a ten-decision trailing mean within each trajectory before aggregation. Recurrent-state distance is unsmoothed:

$$d_h(t) = \sqrt{\frac{1}{D_h} \left\| h_t^{\text{switch}} - h_t^{\text{unchanged}} \right\|_2^2}. \tag{16}$$

Here D_h is state dimensionality. Initial branch states coincide; cached recurrent states span decisions 0–499, rewards 1–500, and physical states 0–500, without extrapolating h_{500}. These distances compare states within one model, not representations from independently trained networks. Figure 12 summarizes the six switch directions, complementing the velocity changes in Figure 4(a).

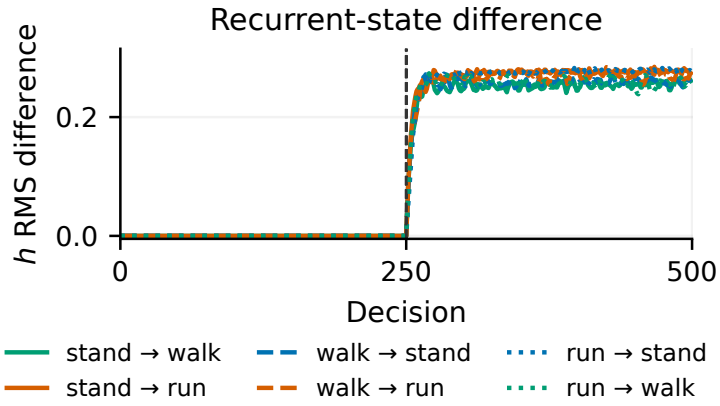


Figure 12: Recurrent-state divergence after switching the task ID at decision 250. Curves show h RMS distance from the unchanged-ID branch; shading denotes 95% intervals over 20 paired starts for one training seed. Divergence follows different actions and observations, not direct task-ID conditioning of the RSSM.

Mean velocities over decisions 450–500 are approximately -0.020 m/s for either switch to stand, 2.268/2.288 m/s for run/stand to walk, and 5.080/5.334 m/s for stand/walk to run. Destination rewards for stand-to-run and walk-to-run rise from 0.327/0.817 over decisions 200–249 to 1.349/1.412 over decisions 450–500. Stand reward need not rise when upright motion already scores well. Post-switch state divergence follows differing actions and environmental feedback, not an instantaneous task-ID rewrite of h or a direct measure of adaptation quality.

Behavior and reward calibration across models. Table 21 compares matched-task returns. Shared uses one controller at mean speed 2.834 m/s; its three scores relabel the same trajectories. Its paired return differences from MOTO-WM are $+26.73$ $[-7.67, 87.54]$, $+2.22$ $[-13.60, 17.70]$, and -168.17 $[-203.14, -105.95]$ for stand/walk/run. Thus, its compatible stand/walk scores coexist with lower run performance.

Table 21: Matched-task returns. Shared uses one policy, rescored under three rules. Training budgets and task sets differ.

Task rule	MOTO-WM	Native Dreamer	Shared
stand	944.5 [883.8, 978.6]	975.8 [967.9, 982.6]	971.2 [965.4, 976.5]
walk	950.8 [935.9, 963.3]	966.6 [961.9, 971.4]	953.0 [943.4, 960.8]
run	624.0 [561.5, 658.7]	741.9 [728.1, 754.5]	455.8 [450.9, 460.2]

Table 22 reports reward MAE on the 2,240-point comparison bank. Figure 15 plots mean predicted against mean true reward in width-0.2 bins, aggregating within pairs before averaging represented

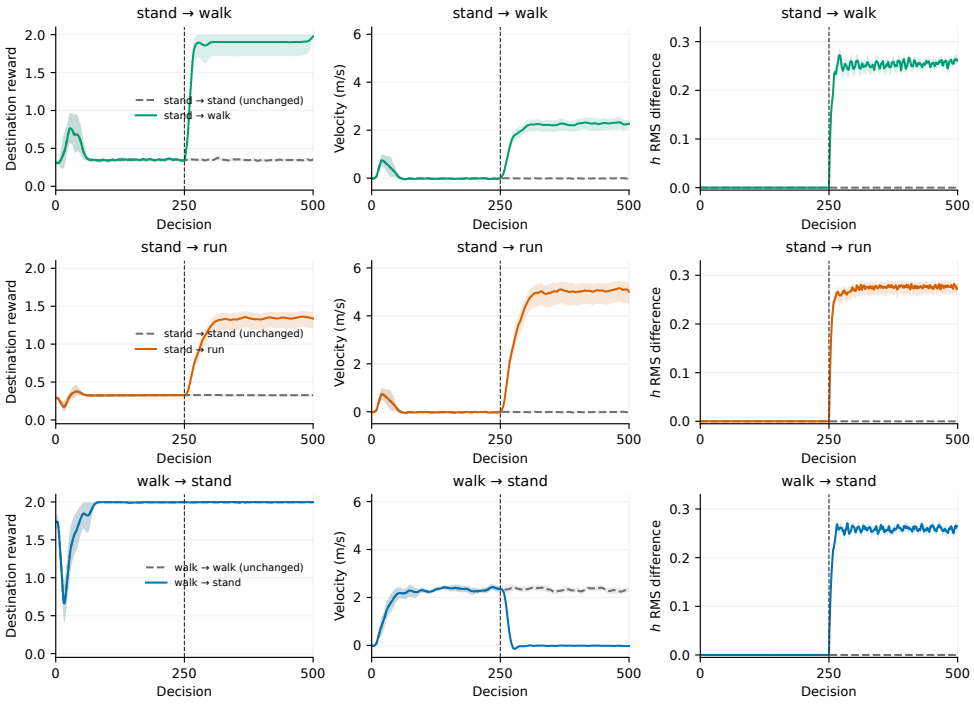


Figure 13: **Task switches, part I.** Colored curves switch IDs at decision 250; gray controls retain the source ID. Both use destination rewards. Shading shows 95% intervals over 20 paired starts. Reward and velocity use ten-decision trailing means; h RMS is unsmoothed.

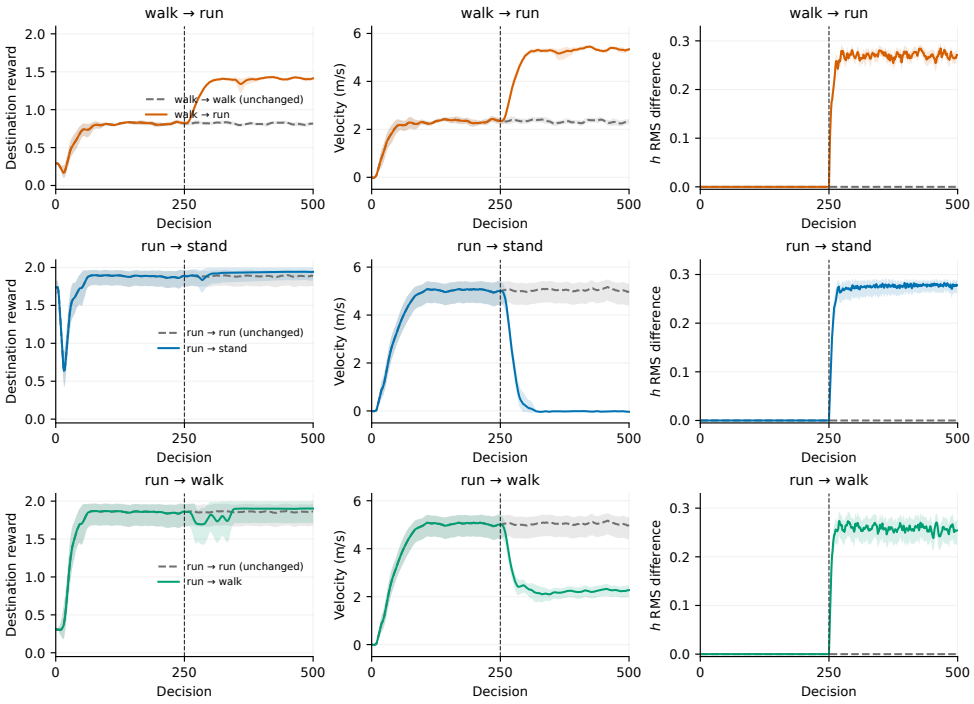


Figure 14: **Task switches, part II.** Remaining directions, using the protocol and scales of Figure 13.

pairs; sparse bins have different precision. The run curve's high-reward downturn indicates under-prediction on cached states, not temporal deterioration. Shared's larger errors can coexist with high stand/walk returns because calibration uses mixed histories, whereas returns evaluate its own actor, which does not plan online through the reward head. Compatible reward rules provide an additional explanation. These results do not make reward accuracy irrelevant during training or imply that Shared predicts the mean reward label. MOTO-WM improves over Shared on these labels, but not uniformly over native Dreamer.

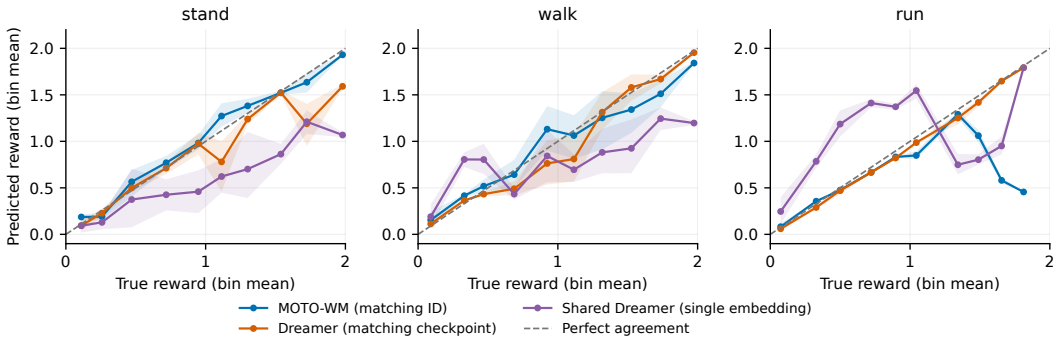


Figure 15: **Reward calibration on identical histories.** Blue: matching MOTO-WM ID; orange: matching native Dreamer; purple: Shared. The diagonal denotes perfect calibration; shading shows pair-bootstrap intervals for predicted bin means. Model colors and axes are shared across panels.

Table 22: Reward MAE on 2,240 common-history readout points; 95% intervals resample initial-state pairs.

Model	stand	walk	run
MOTO-WM	0.056 [0.051, 0.062]	0.148 [0.138, 0.157]	0.181 [0.174, 0.187]
Native Dreamer	0.373 [0.351, 0.390]	0.045 [0.041, 0.049]	0.068 [0.059, 0.082]
Shared	0.883 [0.856, 0.909]	0.671 [0.646, 0.696]	0.597 [0.572, 0.623]

On the same comparison bank, actor-mean RMS differences for stand–walk/stand–run/walk–run are 0.387/0.489/0.401 for MOTO-WM IDs and 0.920/1.080/1.073 for Dreamer checkpoints. MOTO-WM changes the ID at one inferred state, while each Dreamer checkpoint infers its own state from the common history. Shared has one actor readout at a given state. The magnitudes describe specialization rather than action quality, and should not be compared with the 960-point bank as if the evaluated histories were identical.

Critic calibration. We evaluate 12 starting histories (four pairs, three Dreamer sources), with four independent own-policy continuations of 2,048 real decisions per policy. Targets use $\gamma = 332/333$ without a learned bootstrap tail; time-limit truncations are disabled after restoration and genuine terminations respected. The maximum omitted tail is $2\gamma^{2048}/(1-\gamma) \approx 1.41$. Table 23 shows no uniform improvement in MOTO-WM's MAE, particularly on stand. The Dreamer-generated prefixes are cross-policy inputs for MOTO-WM. Shared is excluded because its critic targets a multi-task mixture, not one uniquely correct Walker reward.

Table 23: Critic error against independent 2,048-step own-policy Monte Carlo returns on 12 fixed starting histories. Intervals resample four initial-state pairs.

Task	MOTO-WM MAE	Native Dreamer MAE
stand	48.19 [20.92, 78.56]	6.52 [4.16, 8.88]
walk	16.08 [8.71, 24.16]	22.51 [20.40, 24.63]
run	24.22 [7.17, 49.07]	24.88 [22.18, 29.25]

Task information in targets and readouts. Let T be a balanced task ID and J a fixed cached history/state, including its posterior sample. The conditioning variable is fixed before task labels vary; it is not the identity of the controller that generated the history. We distinguish information in the true reward target from information in trained action and critic readouts. All information values use natural logarithms, are bounded by $\log 3$, and describe finite support rather than population conditional MI in latent space. For true rewards at 960 points, we use $Q_{w,\phi}(r) = \lfloor r/w + \phi \rfloor$ and empirical $I(Q_{w,\phi}(R); T \mid J)$. Width 0.1 and phase zero quantify reward-label distinguishability at that resolution (Table 24).

For six-dimensional executed actions, let π_t^{clip} denote the clipped policy distribution at x_j. We estimate

$$I(A; T \mid J) = \frac{1}{N} \sum_j \frac{1}{3} \sum_{t=1}^{3} \mathrm{KL}\left(\pi_t^{\text{clip}}(\cdot \mid x_j) \,\middle\|\, \frac{1}{3} \sum_{u=1}^{3} \pi_u^{\text{clip}}(\cdot \mid x_j) \right). \tag{17}$$

The calculation accounts for clipping-boundary masses; its Monte Carlo repeat-mean standard deviation is approximately 0.000184 nats. Information divided by $\log 3$ is not classification accuracy or a requirement on every optimal policy. Critic predictions use the same quantization: width 10 gives 1.03219 nats at phase zero but 0.74289 $[0.72603, 0.76311]$ at phase 0.5. Table 25 reports all widths and phases. This dependence on bin boundaries limits numerical interpretation; task-dependent predictions alone do not establish calibration or superior returns.

Table 24: Task-conditional information at 960 fixed states. Binned statistics use phase zero; these distinct quantities are not a head-importance ranking.

Quantity	Definition	Information (nats)
True reward	Binned, width 0.1	0.65992 [0.65192, 0.66858]
Executed action	Clipped-policy mixture	1.03134 [1.02359, 1.03880]
Critic prediction	Binned, width 10	1.03219 [1.02255, 1.04133]

Table 25: Sensitivity of information in the critic prediction to quantization. Values are in nats.

Bin width	Phase 0	Phase 0.5
1	1.07503	1.07503
5	1.05096	0.94169
10	1.03219	0.74289
25	0.68417	0.68706
50	0.66780	0.65914

Task information in real future returns. We restore 60 histories and simulator states (20 pairs, three source conditions) at decision 250. For each start and ID, 16 independent continuations yield 2,880 branches of 100 decisions. The first action is sampled independently from the recomputed cached-state policy; later actions use the native stochastic policy with independent keys. Task identity determines both policy and reward scoring. Normalized discounted returns are

$$\bar{G}_H = \frac{\sum_{k=0}^{H-1} \gamma^k r_{k+1}^{(T)}}{\sum_{k=0}^{H-1} \gamma^k}, \qquad H \in \{50, 100\}, \quad \gamma = 332/333, \tag{18}$$

without a critic tail. The 50-step target uses the first half of each branch; the horizons are not independent experiments. Normalization preserves decision-reward units.

We average frequency-based information between T and $Q_{w,\phi}(\bar{G}_H)$ over starts. Figure 16 compares the observed statistic with a reference from 199 within-start task-label permutations preserving counts. Observed-minus-reference is a diagnostic contrast, not an unbiased MI estimator; the shuffled reference captures finite-sample upward bias. At width 0.1 and phase zero, observed values are 0.79523/0.72041 nats for 50/100 decisions, references are 0.10563/0.08723, and differences are 0.68959/0.63318. The observed pair-bootstrap intervals are $[0.77205, 0.81754]$ and

$[0.69407, 0.75019]$, respectively. All audited widths (0.05/0.1/0.2) and phases (0/0.5) have positive differences and permutation $p = 0.005$, the minimum resolution with 199 permutations. These are descriptive sensitivity checks, not independent confirmatory tests. The positive contrast shows that the desired task helps distinguish real future returns after fixing the starting history and physical state, supporting task-dependent value targets beyond differences in a critic's own predictions. Because task identity changes both policy and scoring, this experiment neither separates their contributions nor proves direct critic-ID wiring necessary or establishes infinite-horizon accuracy.

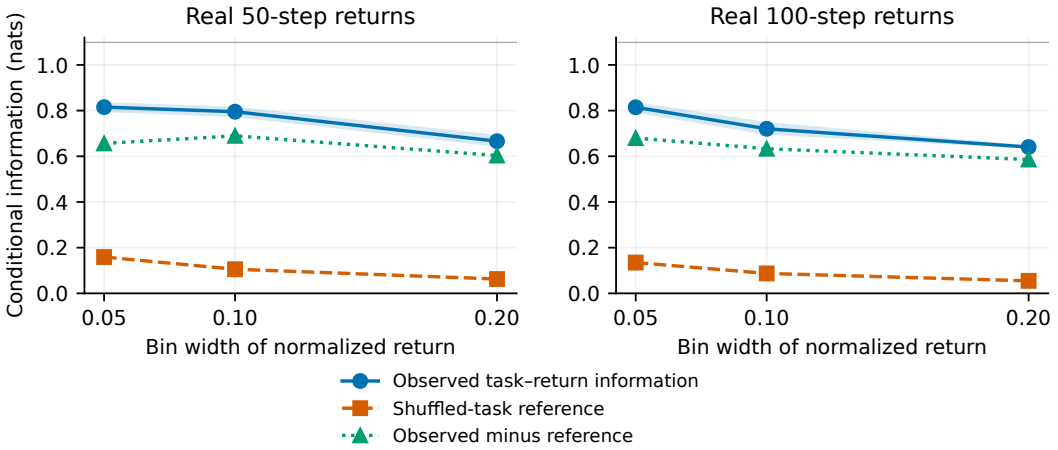


Figure 16: **Task information in real future returns.** At phase zero, quantization width varies along the horizontal axis. Blue: observed information with pair-bootstrap intervals; orange: shuffled-task reference; green: their difference. The horizontal line is $\log 3$. These are real 50/100-step outcomes, not critic errors.

Scope. Reward, action, and critic statistics use different targets and estimators and cannot rank head importance. Continuation targets are constant at $332/333$ in this nonterminal bank, yielding zero task information and no evidence that continuation requires task conditioning. The diagnostics support shared physical-history states with task-dependent readouts in this setting. They do not establish sufficiency across arbitrary tasks or replace matched-training ablations and additional training seeds. Short-horizon prediction errors are evaluated separately in Appendix L.

L SHORT-HORIZON PREDICTION CHECKS

The Walker pixel check (Figure 4(b)) and reward check (Figure 17) compare predicted and executed futures. The DMC20 image examples use separate evaluation banks and sequence indices, as specified below.

Walker protocol and errors. We use the frozen seed-0 checkpoint from Appendix K and 36 histories: four initial-state pairs, three native Dreamer controllers, and decisions 100, 250, and 400. Each history is tested under three task IDs. *Recorded actions* replay the observed future sequence; with fixed actions and stochastic draws, imagined h, sampled z, and decoded images are identical across IDs, whereas reward and value readouts may differ. For *imagined actor actions*, we execute the task-conditioned actor's imagined sequence open-loop from the same restored real simulator snapshot. This tests prediction, not closed-loop policy return.

At horizons $k \in \{1, 5, 15\}$, pixel MSE averages over $[0, 1]$ RGB pixels and channels; reward error is $|\hat{r}_k - r_k^{(T)}|$ under the matching reward rule. These are errors at k, not cumulative or rollout-averaged errors. Each mode/horizon has 108 condition-specific observations clustered into four initial-state pairs (Table 26). At 15 decisions, reward errors reach 0.88233/0.75242 for recorded/imagined actions on the $[0, 2]$ decision-reward scale. The modes visit different futures, so their error difference does not establish that actor-selected actions improve the model.

Table 26: Pointwise prediction errors at each horizon, with 95% intervals over four initial-state pairs.

Action sequence	Horizon	Pixel MSE	Reward abs. error
Recorded	1	0.0081 [0.0075, 0.0088]	0.2411 [0.1774, 0.3047]
Recorded	5	0.0113 [0.0109, 0.0117]	0.3110 [0.2843, 0.3409]
Recorded	15	0.0188 [0.0175, 0.0195]	0.8823 [0.8394, 0.9241]
Imagined actor	1	0.0080 [0.0074, 0.0089]	0.2356 [0.1788, 0.2924]
Imagined actor	5	0.0144 [0.0134, 0.0156]	0.2265 [0.1980, 0.2586]
Imagined actor	15	0.0221 [0.0209, 0.0227]	0.7524 [0.6768, 0.8115]

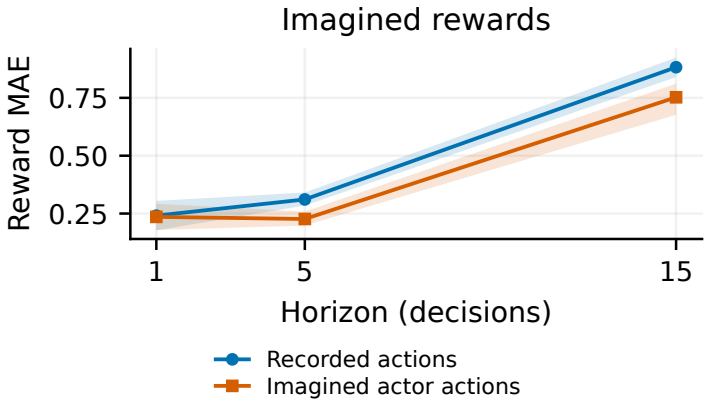


Figure 17: Walker reward prediction errors at horizons 1, 5, and 15 under the protocol of Figure 4(b). Shading shows 95% intervals over four initial-state pairs for one training seed.

Visual predictions across DMC20. Figures 18–21 extend Figure 5 to all 20 tasks. With context at action indices 0 and 4, subsequent actions generate predictions at indices 5–15 without observation correction (index 10 is not displayed). Index 15 is eleven indices beyond the last context, not a 15-step forecast. True and Model denote real and decoded observations. Walker confidence intervals do not transfer to these other environments, and neither check establishes uniform error bounds or long-horizon stability.

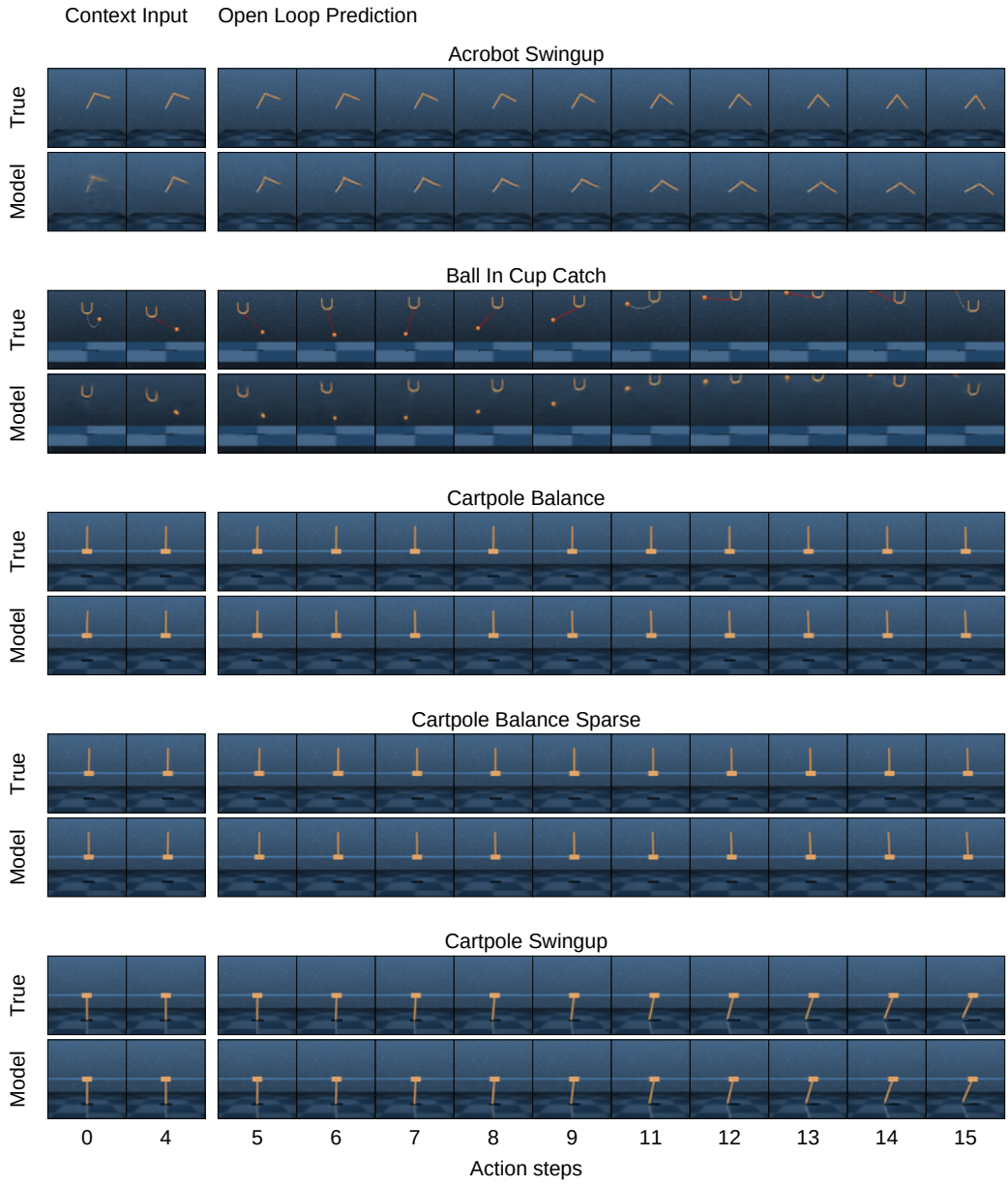


Figure 18: Open-loop predictions on Acrobot Swingup, Ball in Cup Catch, Cartpole Balance, Cartpole Balance Sparse, and Cartpole Swingup.

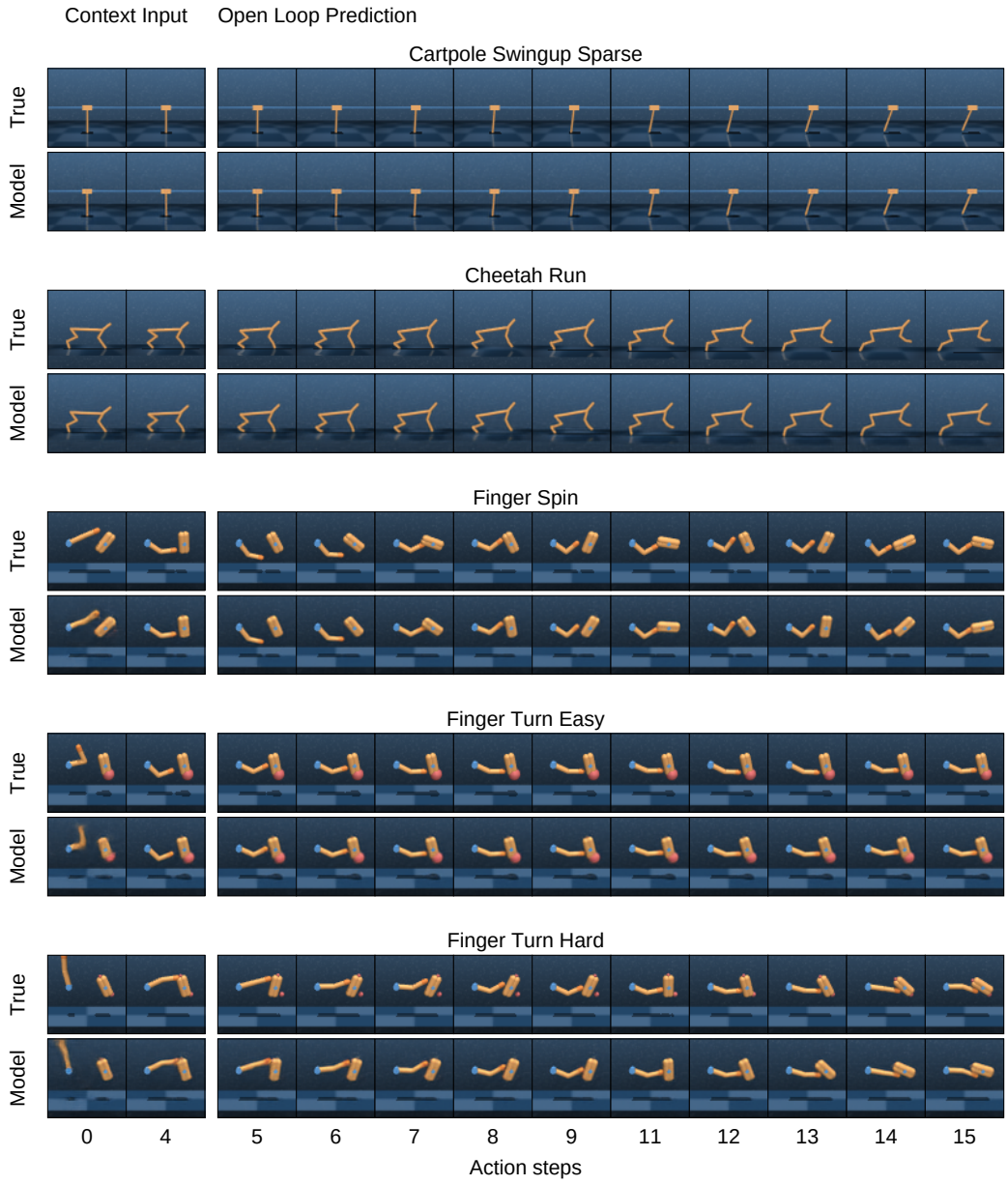


Figure 19: Open-loop predictions on Cartpole Swingup Sparse, Cheetah Run, Finger Spin, Finger Turn Easy, and Finger Turn Hard.

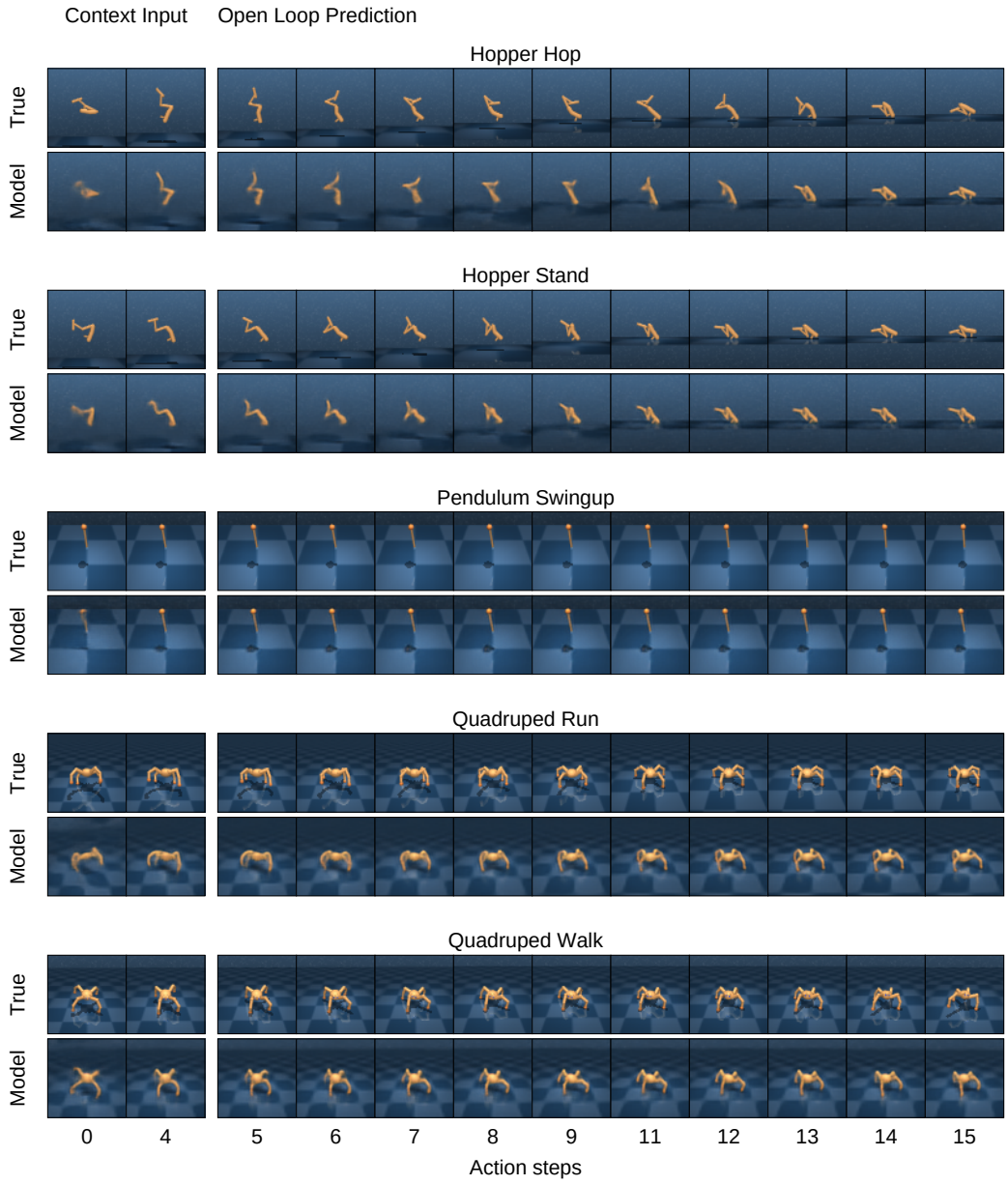


Figure 20: Open-loop predictions on Hopper Hop, Hopper Stand, Pendulum Swingup, Quadruped Run, and Quadruped Walk.

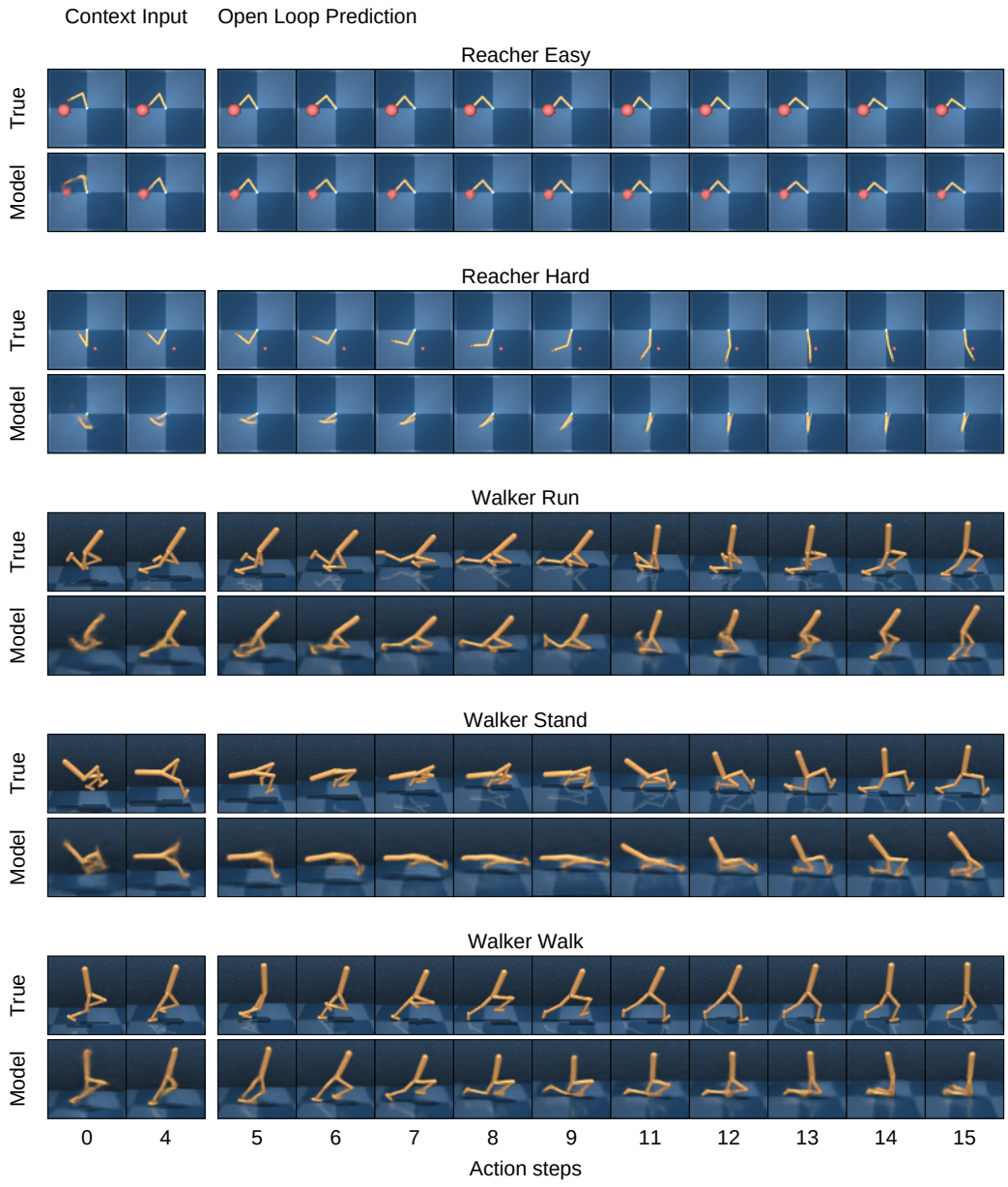


Figure 21: Open-loop predictions on Reacher Easy, Reacher Hard, Walker Run, Walker Stand, and Walker Walk.