ENCODING MEMORY IN WORLD MODELS WITH TEST-TIME MODEL-BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning (MBRL) approaches have demonstrated strong performance across diverse domains in recent years, learning accurate world models from agent trajectories. However, despite this progress, current approaches still struggle to surpass human performance on tasks that require long-term memory and planning. In this work, we address the problem of long-term memory in MBRL by moving beyond the commonly studied paradigm of architecture design. We introduce Test-Time Model-Based Reinforcement Learning (TT-MBRL), using gradient descent at test time to encode the memory of episode observations into world model parameters. This enables the agent to learn a distinct world model for each episode, improving memory capabilities compared to traditional approaches that solely rely on contextual memory. We first establish a proof of concept on a simplified two-dimensional variant of the Memory Maze environment, which we refer to as Memory Maze 2D, showing consistent improvements across different world model architectures. We then demonstrate the effectiveness of TT-MBRL in the 3D domain, achieving new state-of-the-art results on the Memory Maze benchmark. Finally, we show that TT-MBRL maintains strong performance on non-memory benchmarks such as DeepMind Control and Atari 100k. Our code will be made publicly available for full reproducibility.

1 INTRODUCTION

Model-based reinforcement learning has gained increasing research attention in recent years (Kaiser et al., 2020; Schrittwieser et al., 2020; Hafner et al., 2020; 2021; 2023; 2025; Hansen et al., 2022; 2024; 2026; Micheli et al., 2024; Alonso et al., 2024), driven in part by advances in hardware computing capabilities. These improvements have enabled the training of world models from high-dimensional observations using deep neural networks as function approximators (Hafner et al., 2019). World models (Sutton, 1991; Ha & Schmidhuber, 2018) capture an agent's experience into a predictive function that can be used in place of the real environment to learn complex behaviors. By leveraging a model of the environment, agents can simulate multiple possible future trajectories in parallel, leading to improved generalization, greater sample efficiency, and more effective decision-making via planning.

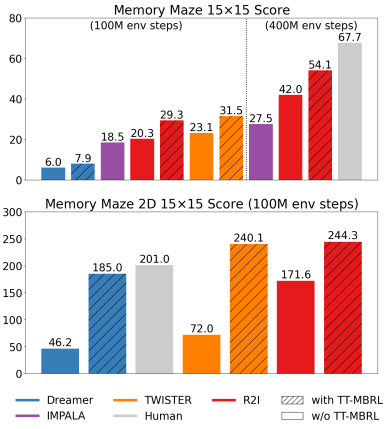


Figure 1: Performance improvements brought by TT-MBRL on Memory Maze 2D and Memory Maze.

Among these approaches, the Dreamer algorithm (Hafner et al., 2020; 2021; 2023) has demonstrated strong performance across diverse environments by learning a world model capable of generating accurate trajectories in latent space. However, despite making realistic predictions for fully observable settings, Dreamer struggles to maintain temporal coherence in partially observable environments like Memory Maze (Figure 2). World model prediction error accumulates over time, leading to long-horizon trajectories that become increasingly inconsistent with past observations.

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.

To remedy this problem, recent approaches have proposed architectural improvements to better handle past context information and enhance the memory capabilities of world models. Pašukonis et al. (2023) proposed to augment Dreamer with Truncated Backpropagation Through Time (TBTT), conditioning training sequences on previous context representations. TransDreamer (Chen et al., 2021) proposed a Transformer-based world model to bypass the limitations of recurrent neural networks and process long context with self-attention (Vaswani et al., 2017). Building on this direction, TWISTER (Burchi & Timofte, 2025) further improved Transformer-based world models by incorporating a contrastive predictive coding objective to enhance temporal representation learning. Following the success of State Space Models (SSMs) in capturing dependencies across long sequences (Gu et al., 2022; Nguyen et al., 2022; Smith et al., 2023), several works have also explored SSM-based world models. Recall to Imagine (R2I) (Samsami et al., 2024) improved training efficiency and long-term memory using a modification of the S4 model (Gu et al., 2022), outperforming previous approaches on the Memory Maze benchmark (Pašukonis et al., 2023). Concurrently, Deng et al. (2023) proposed S4WM, further demonstrating that SSM-based architectures outperform Transformer-based world models in terms of long-term memory.

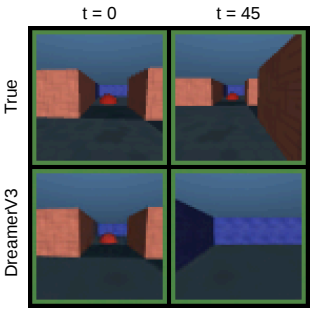


Figure 2: DreamerV3 failure case where the world model predicts an incoherent future after returning to the original position in Memory Maze.

However, despite this progress, existing approaches still encounter difficulties surpassing human performance on tasks requiring long-term memory and planning (Samsami et al., 2024). Most prior works focus on architecture design, assuming that improved world model architectures can effectively exploit past context to capture long-term dependencies. These approaches rely on learning a single world model that generalizes across episodes, requiring it to extract common patterns and adapt to the unique structure of each new episode through contextual representations. While shared patterns can be learned effectively, modeling episode-specific information through context alone remains challenging, particularly in partially observable settings like Memory Maze. In contrast, biological systems are believed to encode long-term memory through persistent modifications of synaptic strengths, a process often associated with Long-Term Potentiation (LTP) in neural circuits (Bliss & Collingridge, 1993). Motivated by this perspective, and by the observation that Test-Time Training (TTT) can improve predictive models when training and test data come from different distributions (Sun et al., 2020), we propose a learning strategy that encodes episode-specific memory directly into the model parameters via gradient-based updates at test time. This approach enables the agent to capture episode-specific information by adapting its world model online, rather than relying solely on contextual memory.

In this work, we address the challenge of long-term memory in MBRL by moving beyond the commonly studied paradigm of architecture design. Inspired by test-time training methods that adapt model parameters at inference time (Sun et al., 2020; Gandelsman et al., 2022; Wang et al., 2025), we introduce Test-Time Model-Based Reinforcement Learning (TT-MBRL), using gradient descent at test time to encode the memory of episode-specific information into world model parameters. This enables the agent to learn a distinct world model for each episode, improving memory capabilities compared to traditional approaches that solely rely on contextual memory. We first establish a proof of concept on a simplified two-dimensional variant of the Memory Maze environment, which we refer to as Memory Maze 2D, showing consistent improvements across different world model architectures. We then demonstrate the effectiveness of TT-MBRL in the 3D domain, achieving new state-of-the-art results on the Memory Maze benchmark (Pašukonis et al., 2023). We find that the effectiveness of TT-MBRL scales proportionally to the number of environment steps collected at test-time, increasing the amount of rewards obtained per step by 53.7% for R2I after 4000 environment steps on Memory Maze 15×15 (Figure 5c). Finally, we show that TT-MBRL maintains strong performance on non-memory benchmarks such as DeepMind Control and Atari 100k. Figure 1 summarizes the results obtained on memory benchmarks.

2 Background

We evaluate the impact of TT-MBRL within the Dreamer family of world model architectures. The Dreamer algorithm was originally introduced in Hafner et al. (2020) and later refined in subsequent

works (Hafner et al., 2021; 2023) to master diverse environments with fixed hyperparameters. It has since become a prominent baseline in MBRL, demonstrating strong performance across a wide range of tasks from high-dimensional visual observations. The agent consists of three main components: The world model that learns to simulate the environment to generate imaginary trajectories, the critic network that estimates state values, and the actor network that learns to select optimal actions.

2.1 World Model Learning

Dreamer learns a world model of the environment in latent space using a sequential autoencoder architecture (Hinton & Salakhutdinov, 2006; Kingma & Welling, 2014; Chung et al., 2015). Specifically, the world model encodes observations o_t into compact stochastic representations z_t learned via a reconstruction objective, while a Recurrent State-Space Model (RSSM) (Hafner et al., 2019) predicts future states z_{t+1} given actions a_t. In addition, the world model includes multi-layer perceptron (MLP) heads for predicting rewards r_t and episode continuation c_t signals from model states $s_t = \{h_t, z_t\}$. The trainable subcomponents of the world model are the following:

$$\text{RSSM} \begin{cases} \text{Sequential Network:} & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \text{Encoder Network:} & z_t \sim q_\phi(z_t \mid h_t, o_t) \\ \text{Dynamics Predictor:} & \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\ \text{Reward Predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid s_t) \\ \text{Continue Predictor:} & \hat{c}_t \sim p_\phi(\hat{c}_t \mid s_t) \\ \text{Decoder Network:} & \hat{o}_t \sim p_\phi(\hat{o}_t \mid s_t) \end{cases} \tag{1}$$

While the original Dreamer algorithm uses a Gated Recurrent Unit (GRU) (Cho et al., 2014) for the sequential network, other variants have been proposed using Transformer-based world models with self-attention (Vaswani et al., 2017) and Structured State Space Sequence models (S4) (Gu et al., 2022). These variants allow computing deterministic states $h_{1:T}$ in parallel using a non-recurrent encoder network $z_t \sim q_\phi(z_t \mid o_t)$ and a parallelizable sequential network $h_{1:T}, x_{1:T} = f_\phi(z_{0:T-1}, a_{0:T-1}, x_0)$, where x_t are SSM hidden states. In this paper, we experiment with three recently proposed world model designs based on the Dreamer algorithm: DreamerV3 (Hafner et al., 2023), which uses an RNN-based world model; TWISTER (Burchi & Timofte, 2025), which uses a Transformer-based world model with self-attention; and R2I (Samsami et al., 2024), which introduced the Structured State-Space Model (S3M), a modification of S4 to handle temporal dependencies.

The world model is trained on trajectories sampled from a replay buffer. Trajectories are collected during the exploration phase, where the current policy interacts with the environment. Given an input batch containing B sequences of T image observations $o_{1:T}$, actions $a_{1:T}$, rewards $r_{1:T}$, and episode continuation flags $c_{1:T}$, the world model parameters (ϕ) are optimized to minimize the following loss function:

$$L(\phi) = \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \Big[\beta_{\text{pred}} L_{\text{pred}}(\phi) + \beta_{\text{dyn}} L_{\text{dyn}}(\phi) + \beta_{\text{rep}} L_{\text{rep}}(\phi) \Big] \tag{2}$$

The prediction loss L_{pred} learns hidden representations with a reconstruction loss and predicts environment quantities such as rewards and continuation flags:

$$L_{\text{pred}}(\phi) = \underbrace{-\log p_\phi(o_t \mid s_t)}_{\text{reconstruction loss}} \underbrace{-\log p_\phi(r_t \mid s_t)}_{\text{reward predictor loss}} \underbrace{-\log p_\phi(c_t \mid s_t)}_{\text{continue predictor loss}} \tag{3}$$

The dynamics loss L_{dyn} trains the dynamics predictor to predict the next state representation by minimizing the KL divergence between the predictor $p_\phi(\hat{z}_t \mid h_t)$ and the next stochastic representation $q_\phi(z_t \mid h_t, o_t)$. The representation loss trains the representations to become more predictable if the dynamics predictor cannot predict their distribution. Both loss terms use the stop gradient operator $sg(\cdot)$ to prevent the gradients of targets from being backpropagated and are scaled by the corresponding loss weights β_{dyn} and β_{rep}, respectively:

$$\begin{aligned} L_{\text{dyn}}(\phi) &= \max\big(1, \text{KL}\big[\, sg(q_\phi(z_t \mid h_t, o_t)) \;\|\; p_\phi(\hat{z}_t \mid h_t) \,\big]\big) \\ L_{\text{rep}}(\phi) &= \max\big(1, \text{KL}\big[\, q_\phi(z_t \mid h_t, o_t) \;\|\; sg(p_\phi(\hat{z}_t \mid h_t)) \big]\big) \end{aligned} \tag{4}$$

2.2 ACTOR-CRITIC LEARNING

The agent's policy and value functions are learned by imagining trajectories using the world model. Learning takes place entirely in latent space, which allows the agent to process large batch sizes and increase generalization. To do so, the model states of the sampled sequence are flattened along batch and time dimensions to generate $B_{img} = B \times T$ sample trajectories. The world model imagines $H = 15$ steps into the future using the sequential network and dynamics predictor, selecting actions by sampling from the actor network. Analogously to world model predictor networks, actor and critic networks are designed as simple MLPs with parameter vectors (θ) and (ψ), respectively. We describe the actor-critic learning process of the agent in more detail in Appendix A.3.

3 METHOD

We introduce TT-MBRL, a novel learning framework for MBRL that takes advantage of test-time observations to learn episode-specific patterns and acquire long-term memory. TT-MBRL learns a unique world model and policy for each evaluation episode. In contrast to training time, where model parameters are shared across all episodes in the replay buffer, this allows the agent to fine-tune its parameters to each individual episode. An overview of the proposed method is provided in Figure 3.

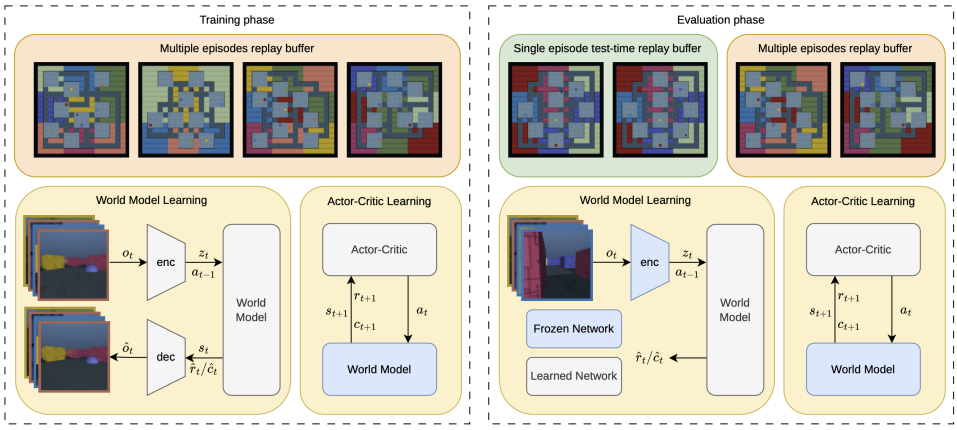


Figure 3: During training, the world model learns to predict the common patterns shared among multiple episode trajectories collected during exploration. The agent can solely rely on contextual memory to identify patterns that are unique to each new episode, which can lead to incoherent future predictions. At test time, the agent can leverage both contextual and neural memory. This allows the world model to make more accurate predictions and learn an improved policy for each new episode.

3.1 TEST-TIME OPTIMIZATION

At test time, the agent initializes test-time network parameters (ϕ', θ', ψ') with the set of weights learned during the training phase (ϕ, θ, ψ) at the beginning of each episode. A dedicated test-time replay buffer is also created to store observations collected within the current evaluation episode. Analogous to the training procedure, the agent samples mini-batches of trajectories to update its parameters. We find it beneficial to draw half of the trajectories from the training replay buffer and the other half from the test-time buffer. This strategy acts as a form of regularization, mitigating overfitting to the limited test-time observations, which could otherwise degrade performance. When the number of test-time samples is smaller than the required world model sequence length, mini-batches are concatenated with trajectories from the training buffer. We also freeze the encoder and decoder parameters and drop the reconstruction loss at test time to improve TT-MBRL efficiency, preserving the representation space learned during training. Unlike the sequential and dynamics networks, which must adapt to the specific layout and structure of each new episode, the encoder and decoder learn general visual representations that are independent of their temporal ordering. Freezing these parameters significantly reduces evaluation time without impacting performance. We provide an empirical analysis of these design choices in Section 4.3. For each new policy step,

the world model loss optimizes the test-time parameters ϕ' as follows:

$$L(\phi') = \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \Big[\beta_{\text{pred}} L_{\text{pred}}(\phi') + \beta_{\text{dyn}} L_{\text{dyn}}(\phi') + \beta_{\text{rep}} L_{\text{rep}}(\phi') \Big] \tag{5}$$

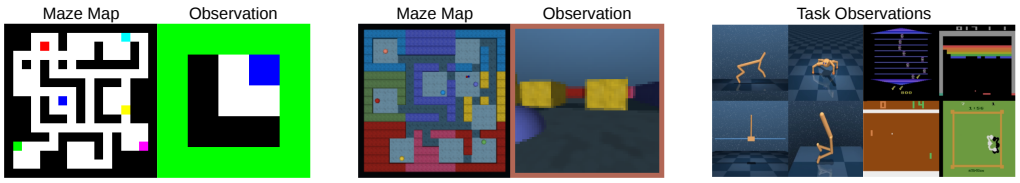
With the test-time prediction loss $L_{\text{pred}}(\phi')$ omitting the reconstruction objective:

$$L_{\text{pred}}(\phi') = \underbrace{-\log p_{\phi'}(r_t \mid s_t)}_{\text{reward predictor loss}} \underbrace{-\log p_{\phi'}(c_t \mid s_t)}_{\text{continue predictor loss}} \tag{6}$$

The agent's behavior at test time is optimized using the adapted world model to learn an improved policy. By reducing world model prediction errors, TT-MBRL enables more accurate estimation of imagined returns, which in turn allows the actor and critic to be more effectively optimized.

4 Experiments

In this work, we investigate the potential of TT-MBRL to learn long-term memory via gradient descent. To this end, we evaluate our method on several tasks that require long-term memory and planning. We first establish a proof of concept by experimenting with a simplified two-dimensional variant of the Memory Maze environment, which we refer to as Memory Maze 2D, showing consistent improvements across different world model architectures. We then demonstrate the effectiveness of TT-MBRL in the 3D domain, achieving new state-of-the-art results on the Memory Maze benchmark (Pašukonis et al., 2023). Finally, we conduct a study to identify effective strategies for performing MBRL at test time and show that TT-MBRL maintains strong performance on non-memory benchmarks such as the DeepMind Control Suite (DMC) (Tassa et al., 2018) and Atari 100k (Kaiser et al., 2020).



(a) Memory Maze 2D (b) Memory Maze (c) DMC/Atari

Figure 4: Randomly generated environment map and agent observation for Memory Maze 2D and Memory Maze benchmarks. The agent receives partial observations of the environment and must rely on long-term memory to learn an optimal policy. In contrast, DMC and Atari 100k tasks are mostly fully observable and solvable without extensive memory.

4.1 Memory Maze 2D

Benchmark. In order to demonstrate the effectiveness of TT-MBRL, we first design a simplified and efficient two-dimensional variant of the Memory Maze environment in which the agent observes a top-down 3×3 local context centered on its current position. To enable goal identification, we augment the observation with colored borders that indicate the target color to be reached for obtaining the next reward. The observation is then flattened and normalized, forming a vector of dimension $75 = 5 \times 5 \times 3$. Figure 4 illustrates a comparison between our modified setting and the original Memory Maze environment. The agent operates in the maze using a discrete action space consisting of five actions: no-op, move left, move right, move up, and move down. We consider the same four maze sizes as in the original benchmark (9×9, 11×11, 13×13, and 15×15), along with their corresponding hyperparameters. Further details on the proposed Memory Maze 2D environment are provided in Appendix A.6.

Results. To evaluate the effectiveness of TT-MBRL, we consider three recent world model architectures: DreamerV3 (Hafner et al., 2023), which employs an RNN-based world model; TWISTER (Burchi & Timofte, 2025), which relies on a Transformer-based architecture; and

Table 1: Memory Maze 2D benchmark comparison for all maze sizes (100M env steps). We show the impact of TT-MBRL on the performance for each method. The reported scores are averaged over three runs with 100 evaluation episodes. We also report Human mean scores over 10 episodes.

Method	Mem Maze 2D 9×9	Mem Maze 2D 11×11	Mem Maze 2D 13×13	Mem Maze 2D 15×15
Human	126.2	172.0	195.0	201.0
Oracle	137.2	215.4	276.6	311.4
DreamerV3 (TBTT)	121.2 ± 3.0	185.9 ± 11.3	108.0 ± 8.8	46.2 ± 7.6
+ TT-MBRL (Ours)	$\mathbf{122.1 \pm 1.1}$	$\mathbf{189.4 \pm 6.8}$	$\mathbf{207.2 \pm 10.9}$	$\mathbf{185.0 \pm 17.6}$
TWISTER	84.8 ± 0.8	100.0 ± 5.9	95.3 ± 10.2	72.0 ± 3.2
+ TT-MBRL (Ours)	$\mathbf{113.7 \pm 1.8}$	$\mathbf{178.7 \pm 4.4}$	$\mathbf{224.2 \pm 6.1}$	$\mathbf{240.1 \pm 17.6}$
R2I	$\mathbf{123.1 \pm 0.2}$	$\mathbf{200.3 \pm 6.3}$	222.9 ± 6.6	171.6 ± 24.3
+ TT-MBRL (Ours)	$\mathbf{123.1 \pm 0.2}$	199.2 ± 4.2	$\mathbf{233.4 \pm 8.3}$	$\mathbf{244.3 \pm 7.2}$

Table 2: Evaluation of the impact of TT-MBRL on world model predictions on Memory Maze 2D 15×15. We aggregate all metrics over 100 unseen trajectories of 64 frames from distinct episodes.

Method	KL div ↓	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
DreamerV3 (TBTT)	114.6	104.5	10.3	0.552	0.283
+ TT-MBRL (Ours)	**66.0**	**55.2**	**16.3**	**0.642**	**0.241**
TWISTER	28.7	89.1	13.1	0.578	0.292
+ TT-MBRL (Ours)	**13.2**	**57.1**	**15.7**	**0.641**	**0.262**
R2I	52.0	115.4	10.1	0.541	0.304
+ TT-MBRL (Ours)	**13.8**	**45.7**	**15.8**	**0.675**	**0.246**

R2I (Samsami et al., 2024), which is built on S4. Table 1 shows the results obtained after training 100M environment steps on each maze size. We also record the performance of a human player to estimate the capabilities of each method compared to human memory abilities. Oracle defines the performance obtained when following the shortest path to the target color. We find that R2I achieves the best results on all maze sizes with a score of 171.6 on Memory Maze 2D 15×15. TWISTER outperforms DreamerV3 on the 15×15 maze size due to its Transformer-based world model with self-attention. However, we find that it struggles to learn an optimal policy on smaller maze sizes. We suppose this is due to the nature of the self-attention mechanism that is not biased toward recent observations and requires accurately selecting the needed information at each time step, while DreamerV3 carries past information into a recurrent hidden state. We find that TT-MBRL improves the performance of all models when a near optimal policy is not already achieved. It allows R2I to outperform human performance on all maze sizes, except 9×9. This is due to the policy occasionally failing on a subset of episodes, despite achieving higher performance on the majority of them. We also evaluate the impact of TT-MBRL on world-model prediction accuracy using a combination of standard video prediction metrics (Table 2). We measure the world model KL divergence, Fréchet Video Distance (FVD) (Unterthiner et al., 2018), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) (Wang et al., 2004) and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) over 100 unseen trajectories of 64 frames from distinct episodes. We find that TT-MBRL consistently decreases the world model prediction error in latent space. This results in more accurate predictions and better performance in video prediction metrics.

4.2 Memory Maze

Benchmark. The Memory Maze benchmark, introduced by Pašukonis et al. (2023), consists of a 3D domain of randomized mazes specifically designed to evaluate long-term memory in agents. In this environment, the agent must locate a target object of a specific color, indicated by the border color in the observation image. Upon reaching and interacting with the correct object, the agent receives a reward of +1, after which a new target object is randomly selected. The benchmark

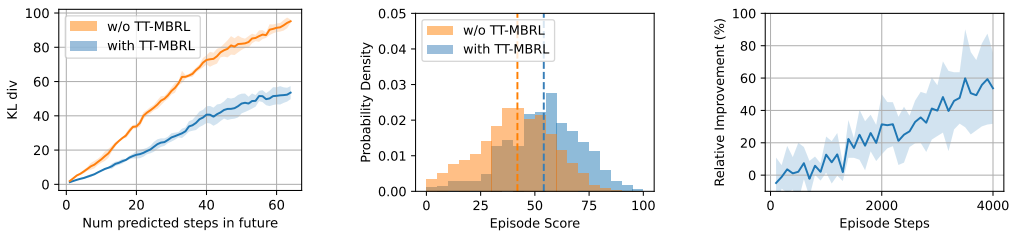
includes four tasks of increasing size and difficulty: Memory Maze 9×9, 11×11, 13×13, and 15×15. While the smaller maze sizes contain three objects, 15×15 mazes include six objects and a greater number of rooms, making the task more challenging while remaining solvable for human players.

Table 3: Memory Maze benchmark comparison (100M env steps) for all maze sizes. We show the impact of TT-MBRL on the performance for each method. The reported scores are averaged over three runs with 100 episodes for evaluation. † denotes the results taken from Samsami et al. (2024). We also report TT-MBRL results for R2I with a budget of 400M environment steps over six runs.

Method	Mem Maze 9×9	Mem Maze 11×11	Mem Maze 13×13	Mem Maze 15×15
Human	26.4	44.3	55.5	67.7
Oracle	34.8	58.0	74.5	87.7
IMPALA†	23.5	28.5	27.0	18.5
IMPALA† (400M steps)	26.5	35.0	36.5	27.5
DreamerV3 (TBTT)	**32.7 ± 0.8**	33.0 ± 1.4	22.1 ± 1.8	6.0 ± 0.3
+ TT-MBRL (Ours)	32.5 ± 0.7	**37.8 ± 1.1**	**32.5 ± 3.2**	**7.9 ± 1.3**
TWISTER	28.2 ± 0.6	33.5 ± 0.9	29.7 ± 2.4	23.1 ± 0.5
+ TT-MBRL (Ours)	**30.6 ± 0.8**	**42.1 ± 0.3**	**42.3 ± 1.9**	**31.5 ± 1.8**
R2I	**32.5 ± 0.6**	45.9 ± 1.6	45.9 ± 1.3	20.3 ± 1.1
+ TT-MBRL (Ours)	32.3 ± 0.5	**47.8 ± 2.6**	**53.0 ± 1.8**	**29.3 ± 4.7**
R2I (400M steps)	33.7 ± 0.3	52.3 ± 0.9	60.1 ± 1.6	42.0 ± 5.8
+ TT-MBRL (Ours)	**33.8 ± 0.6**	**52.4 ± 1.4**	**63.0 ± 1.5**	**54.1 ± 3.5**

Table 4: Evaluation of the impact of TT-MBRL on world model predictions on Memory Maze 15×15. We aggregate all metrics over 100 unseen trajectories of 64 frames from distinct episodes.

Method	KL div ↓	FVD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
DreamerV3 (TBTT)	21.7	67.7	25.0	0.771	0.211
+ TT-MBRL (Ours)	**20.1**	**61.7**	**26.2**	**0.789**	**0.190**
TWISTER	51.8	63.8	24.6	0.756	0.230
+ TT-MBRL (Ours)	**46.3**	**49.1**	**26.2**	**0.792**	**0.187**
R2I	61.3	114.1	21.1	0.687	0.319
+ TT-MBRL (Ours)	**38.3**	**85.2**	**24.0**	**0.750**	**0.252**
R2I (400M steps)	54.2	111.1	21.7	0.697	0.290
+ TT-MBRL (Ours)	**29.3**	**70.7**	**26.0**	**0.784**	**0.192**



(a) KL div (↓) over time steps (b) Evaluation histogram (c) Average obtained reward per step

Figure 5: R2I world model KL divergence over 64 predicted steps, evaluation episodes histogram, and relative improvement in obtained rewards brought by TT-MBRL on Memory Maze 15×15.

Results. We train DreamerV3, TWISTER, and R2I world models for 100M environment steps on all Memory Maze sizes. Following Samsami et al. (2024), we further train R2I up to 400M environment steps and include the results obtained by IMPALA (Espeholt et al., 2018), which serves

as a strong model-free baseline for comparison. Table 3 compares the results obtained by each method. Despite the increased observation complexity of the 3D domain, where observations of the same position vary with viewing angle, unlike the fixed top-down views of Memory Maze 2D, TT-MBRL successfully identifies structural similarities between observed views to build a more accurate episode-specific world model. TT-MBRL yields consistent improvements across all world model approaches on larger maze sizes, achieving substantial gains: The performance of R2I increases from 60.1 to 63.0 on Memory Maze 13×13, and from 42.0 to 54.1 on Memory Maze 15×15, establishing new state-of-the-art results and the closest performance to the human baseline (67.7) among all proposed methods. On the smaller maze sizes, where DreamerV3 and R2I already achieve near-optimal performance, the gains are more modest, consistent with the saturation effects observed in Memory Maze 2D. We further assess the quality of world model predictions in Table 4, comparing video prediction metrics. Across all five metrics, TT-MBRL substantially reduces prediction error. Figure 5 further illustrates the dynamics of adaptation within a single episode for R2I on Memory Maze 15×15. Figure 5a shows the world model KL divergence over 64 predicted steps: Without TT-MBRL, prediction error accumulates rapidly beyond the first 20 steps, while the adapted model maintains substantially lower divergence. Figure 5b shows the episode return distribution, with TT-MBRL shifting mass toward higher scores and reducing the frequency of low-reward episodes. Figure 5c shows the per-step relative improvement in obtained rewards brought by TT-MBRL across the episode: The effectiveness of TT-MBRL grows steadily as more observations accumulate in the test-time replay buffer, reaching 53.7% after 4000 environment steps.

4.3 TT-MBRL Analysis

Training Data Regularization. We conduct a set of studies to identify effective strategies for performing model-based reinforcement learning at test time. Table 5 reports the impact of different sampling schemes on mitigating overfitting. We first consider a naive approach in which the model is updated solely using trajectories collected at test time, without enforcing a minimum number of samples to prefill the test-time replay buffer before learning begins (Test Data Only, No Prefill). Although this strategy improves performance in many episodes, it also degrades performance in others, as the model overfits to limited early observations, resulting in suboptimal policies. To address this issue, we introduce a delay mechanism that postpones learning until the test-time replay buffer is populated with a sufficient number of samples (1024 Prefill). This reduces overfitting but introduces a trade-off, as it delays the benefits of adaptation. To overcome both limitations, we propose mixing trajectories collected at test time with samples from the training replay buffer (Test+Train Data, No Prefill), effectively acting as a form of regularization. This approach mitigates early overfitting while enabling immediate adaptation, leading to more stable and consistent performance improvements. To verify that the gain in performance is due to TT-MBRL rather than additional inference compute, we also perform a compute-matched evaluation. We study the impact of additional compute on performance when only training trajectories are available (Train Data Only). The comparison confirms that the performance gain is due to TT-MBRL rather than additional compute budget.

Table 5: Impact of replay buffer sampling strategy on TT-MBRL performance on Memory Maze 2D 15×15. Combining training and test-time trajectories (Test+Train Data, No Prefill) consistently achieves the best performance across all world model architectures.

Method	**w/o TT-MBRL**	**Train Data Only**	**Test Data Only**		**Test+Train Data**	
			No Prefill	**1024 Prefill**	**No Prefill**	**1024 Prefill**
DreamerV3	46.2	47.0	117.6	173.5	**185.0**	149.6
TWISTER	72.0	69.2	167.6	210.1	**240.1**	207.1
R2I	171.6	171.7	218.8	240.1	**244.3**	237.5

Reconstruction Loss and Parameter Freezing. Table 6 shows the impact of freezing or updating encoder/decoder parameters using the reconstruction loss at test time. We find that the two variants achieve comparable performance across different world model architectures. These findings are as expected, given that encoder representations are independent of their temporal ordering. The episode-specific information is instead captured by the sequential and dynamics network parameters. However, unfreezing these parameters comes at a non-trivial computational cost, as gradient back-

propagation must be carried through the decoder and encoder, resulting in longer evaluation times. This is especially the case for Memory Maze, where observations are 64×64 pixels images. We therefore adopt parameter freezing for encoder and decoder networks as our default setting, as it improves evaluation efficiency without sacrificing performance.

Table 6: Impact of updating encoder/decoder using the reconstruction loss on performance and evaluation time. *Frozen* denotes our default TT-MBRL setting where encoder and decoder parameters are fixed at test time; *Unfrozen* allows them to be updated. We report the evaluation time per episode.

Method	**w/o TT-MBRL**	**Episode Score**		**Eval. Time (hours)**	
		Frozen	**Unfrozen**	**Frozen**	**Unfrozen**
Memory Maze 2D 15×15					
DreamerV3	46.2	**185.0**	181.4	**0.31**	0.35
TWISTER	72.0	240.1	236.4	**0.25**	0.27
R2I	171.6	244.3	**247.5**	**0.06**	0.06
Memory Maze 15×15					
R2I	42.0	**54.1**	53.8	**0.65**	1.04

4.4 Application of TT-MBRL to Non-Memory Domains

In order to evaluate the capacity of TT-MBRL to maintain strong performance in non-memory domains, domains not designed primarily to test long-term episodic memory, we evaluate its impact on two widely adopted benchmarks that do not require long-term memory or planning: The visual DeepMind Control Suite (DMC) (Tassa et al., 2018), which comprises 20 continuous control tasks, and the Atari 100k benchmark (Kaiser et al., 2020), which includes 26 Atari games with discrete actions. Both benchmarks are largely fully observable and can be solved without extensive memory. We report aggregated mean and median scores across all 20 DMC tasks in Table 7 and human-normalized scores on Atari 100k in Table 8. Overall, we find that TT-MBRL does not degrade the performance obtained by DreamerV3. Moreover, we find that TT-MBRL consistently improves performance on *Reacher Hard* and *Finger Turn Hard*, where the agent occasionally fails to accurately reach the target sphere. TT-MBRL mitigates this issue by enabling the agent to better detect and correct such errors. Detailed results for individual tasks are provided in Appendix A.10.

Table 7: DMC results (20 tasks, 1M env steps).

Method	**Mean**	**Median**
DreamerV3	776.2	842.2
+TT-MBRL (Ours)	**788.2**	**850.3**

Table 8: Atari results (26 tasks, 400K env steps).

Method	**Mean**	**Median**	**IQM**
DreamerV3	109.5%	44.3%	0.502
+TT-MBRL (Ours)	**111.1%**	**46.1%**	**0.514**

5 Conclusion

In this work, we addressed the challenge of long-term memory in MBRL. While prior approaches have primarily focused on improving world model architectures to better exploit contextual information, we instead explored an orthogonal direction: encoding memory directly into model parameters. To this end, we introduced TT-MBRL, a framework that leverages gradient-based updates at test time to incorporate episode-specific information into the world model. By enabling the agent to learn a distinct set of parameters for each episode, TT-MBRL moves beyond shared representations and allows rapid adaptation to the current episode, effectively forming a form of neural memory. Through experiments on both a simplified setting and the Memory Maze benchmark, we demonstrated that TT-MBRL consistently improves performance across multiple world model architectures. Notably, our method achieves state-of-the-art results on Memory Maze. We find that the effectiveness of TT-MBRL scales proportionally to the number of environment steps collected at test-time. These results highlight the potential of parameter-based memory as a powerful complement to contextual memory in MBRL systems. More broadly, our findings suggest that test-time learning constitutes a promising direction for developing agents capable of long-term planning.

AI USE STATEMENT

In this work, we used generative AI tools only to identify and correct grammar and spelling mistakes. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results.

ETHICS STATEMENT

The development of autonomous agents for real-world applications introduces various safety and environmental concerns. In real-world scenarios, an agent might cause harm to individuals and damage to its surroundings during both training and deployment. Although using world models during training can mitigate these risks by allowing policies to be learned through simulation, some risks may still persist. This statement aims to inform users of these potential risks and emphasize the importance of AI safety in the application of autonomous agents to real-world scenarios.

REPRODUCIBILITY STATEMENT

We aim to provide all information necessary to ensure the reproducibility of our results. The TT-MBRL algorithm is described in full via pseudocode in the Appendix (Algorithm 1), detailing the test-time parameter initialization, replay buffer construction, and update procedure at each evaluation step. Our method builds on DreamerV3, TWISTER, and R2I, whose original implementations are all publicly accessible, allowing our approach to be integrated into existing codebases. All hyperparameters used across our experiments, for both standard training and test-time optimization, are reported in Appendix A.8 (Tables 16, 17 and 18), along with the model sizes used for each benchmark (Table 15). Since Memory Maze 2D is a new environment introduced in this work, we provide a complete description of its design and configuration in Appendix A.6, including the observation and action space, and maze-size parameters (Table 13). Finally, as noted in the abstract, we will publicly release our code to enable full reproducibility of the results presented in this paper.

REFERENCES

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J 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.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15791–15801, 2025.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Tim VP Bliss and Graham L Collingridge. A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361(6407):31–39, 1993.

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 *International Conference on Machine Learning*, 2024.

Maxime Burchi and Radu Timofte. Learning transformer-based world models with contrastive predictive coding. In *International Conference on Learning Representations*, 2025.

Chang Chen, Jaesik Yoon, Yi-Fu Wu, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. In *Deep RL Workshop NeurIPS 2021*, 2021.

Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, 28, 2015.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International Conference on Computers and Games*, pp. 72–83. Springer, 2006.

Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: Rnns, transformers, and s4. *Advances in Neural Information Processing Systems*, 36:72904–72930, 2023.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1407–1416. PMLR, 2018.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems*, 31, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*, 2021.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. *International Conference on Learning Representations*, 2026.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. *arXiv preprint arXiv:2305.18466*, 2023.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Anthony Hu, Václav Volhejn, Adrien Ramanana Rahary, Chris Mulder, Aditya Makkar, Amélie Royer, Manu Orsini, Alyx Liao, Adam Jelley, Eloi Alonso, et al. Multiplayer interactive world models with representation autoencoders. *arXiv preprint arXiv:2607.05352*, 2026.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jia-Hua Lee, Bor-Jiun Lin, Wei-Fang Sun, and Chun-Yi Lee. EDELINE: Enhancing memory in diffusion-based world models via linear-time sequence modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ph1V6n7BSv.

Vincenzo Lomonaco, Karan Desai, Eugenio Culurciello, and Davide Maltoni. Continual reinforcement learning in 3d non-stationary environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 248–249, 2020.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient world models with context-aware tokenization. In *International Conference on Machine Learning*, pp. 35623–35638. PMLR, 2024.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937. PmLR, 2016.

Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. S4nd: Modeling images and videos as multidimensional signals with state spaces. *Advances in Neural Information Processing Systems*, 35:2846–2861, 2022.

Jurgis Pašukonis, Timothy P Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. In *International Conference on Learning Representations*, 2023.

Tim Pearce and Jun Zhu. Counter-strike deathmatch with large-scale behavioural cloning. In *2022 IEEE Conference on Games (CoG)*, pp. 104–111. IEEE, 2022.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *International Conference on Learning Representations*, 2023.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *International Conference on Learning Representations*, 2024.

Nedko Savov, Naser Kazemi, Deheng Zhang, Danda Pani Paudel, Xi Wang, and Luc Van Gool. StateSpaceDiffuser: Bringing Long Context to Diffusion World Models. In *Advances in Neural Information Processing Systems*, 2025.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pp. 9229–9248. PMLR, 2020.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, volume 2025, pp. 73754–73776, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A Efros, and Xiaolong Wang. Test-time training on video streams. *Journal of Machine Learning Research*, 26(9): 1–29, 2025.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. Efficientzero v2: Mastering discrete and continuous control with limited data. In *International Conference on Machine Learning*, pp. 51041–51062. PMLR, 2024.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:27147–27166, 2023.

A Appendix

A.1 Related Works

A.1.1 Model-based Reinforcement Learning

Model-based reinforcement learning approaches use a model of the environment to simulate agent trajectories, enabling better generalization, sample efficiency, and decision-making through planning. Following the success of deep neural networks for learning function approximations (LeCun et al., 2015), researchers proposed to learn world models using gradient backpropagation. One of the first approaches to apply model-based reinforcement learning to high-dimensional visual observations is SimPLe (Kaiser et al., 2020). The authors proposed to learn a predictive world model for Atari games directly in pixel space using a convolutional autoencoder. SimPLe's model predicts future observations and rewards conditioned on past frames and actions, enabling the training of a Proximal Policy Optimization (PPO) agent (Schulman et al., 2017) using simulated rollouts.

PlaNet (Hafner et al., 2019) subsequently introduced Recurrent State-Space Models (RSSMs), combining deterministic recurrent dynamics implemented with Gated Recurrent Units (GRUs) (Cho et al., 2014) and stochastic latent variables to learn compact world models in latent space. The agent learns hidden representations by encoding observations using a convolutional variational autoencoder (VAE) (Kingma & Welling, 2014) while predicting future latent states and rewards, enabling planning via model predictive control. Building on this framework, Dreamer (Hafner et al., 2020) improved performance by learning a policy and value function directly within the learned latent space. DreamerV2 (Hafner et al., 2021) extended this approach to Atari games by introducing categorical latent variables trained with straight-through gradients (Bengio et al., 2013), improving representational capacity and stability in discrete domains. More recently, DreamerV3 (Hafner et al., 2023) demonstrated that a single architecture with shared hyperparameters can scale across diverse domains, from continuous control to Atari. This was achieved through a set of architectural and optimization refinements, including symlog-scaled reward and value predictions to handle scale variation, layer normalization (Ba et al., 2016) for improved robustness, and return normalization using an exponential moving average (EMA) of return percentiles. With these changes, DreamerV3 achieved state-of-the-art performance across a wide range of benchmarks, often outperforming both specialized model-free and other model-based methods.

In parallel to the Dreamer line of work, other approaches have explored alternative world model designs. Beyond recurrent architectures, Transformer-based world models have been proposed to improve long-term memory and representation learning (Micheli et al., 2023; Chen et al., 2021; Robine et al., 2023; Zhang et al., 2023). Diffusion-based world models (Sohl-Dickstein et al., 2015) have also emerged as a compelling alternative to discrete latent variable sequence modeling, demonstrating strong performance across a range of settings (Alonso et al., 2024; Valevski et al., 2025; Bar et al., 2025; Hafner et al., 2025; Hu et al., 2026). Genie (Bruce et al., 2024) proposed to learn a latent actions world model from a large dataset of public gaming videos. Another line of work has investigated multitask world models, training a single world model jointly across multiple tasks to improve generalization and achieve stronger performance than single-task models (Hansen et al., 2024; 2026). Model-based reinforcement learning approaches using Monte-Carlo tree search (MCTS) (Coulom, 2006) have also been explored following the success of Alpha Go (Silver et al., 2017; 2018; Schrittwieser et al., 2020; Ye et al., 2021; Wang et al., 2024).

A.1.2 Memory in World Models

Multiple approaches have focused on improving the memory capabilities of world models. Pašukonis et al. (2023) proposed to augment Dreamer with Truncated Backpropagation Through Time (TBTT), conditioning training sequences on previous context representations to preserve information over time. TransDreamer (Chen et al., 2021) replaced Dreamer's RSSM by a Transformer State-Space Model (TSSM) using masked self-attention to process past context more easily. The agent was evaluated on Hidden Order Discovery tasks requiring long-term memory and reasoning, demonstrating better performance. Recall To Imagine (R2I) (Samsami et al., 2024) proposed the Structured State-Space Model (S3M) to improve computational efficiency when handling long sequences. This allows R2I to process very long sequences and achieve strong performance on the Memory Maze benchmark. Concurrently, Deng et al. (2023) proposed S4WM, demonstrating that

the S4 architecture outperforms Transformer-based world models in terms of long-term memory. Alternatively, EDELINE (Lee et al., 2025) proposed to augment the DIAMOND world model (Alonso et al., 2024) with a Mamba (Gu & Dao, 2023) SSM to enhance memory capabilities. A similar approach was concurrently proposed in Savov et al. (2025) where the authors use Mamba to improve model predictions on CSGO (Pearce & Zhu, 2022).

A.1.3 Test-Time Training

Test-Time Training (TTT) was originally proposed in Sun et al. (2020) for improving the performance of predictive models when training and test data come from different distributions. This work proposed to turn a single unlabeled test sample into a self-supervised learning problem, on which to update the model parameters before making a prediction. The initial TTT paper used image rotation prediction (Gidaris et al., 2018) as a self-supervised task to update model weights and achieve higher classification accuracy on corrupted test images. The method was later improved with TTT-MAE (Gandelsman et al., 2022), using MAE (He et al., 2022) instead of rotation prediction to reach higher performance. Test-time training was subsequently applied to multiple domains like reinforcement learning (Hansen et al., 2021), language modeling (Hardt & Sun, 2023), and video processing (Wang et al., 2025). In the area of reinforcement learning, Hansen et al. (2021) proposed Policy Adaptation during Deployment (PAD) (Hansen et al., 2021), applying TTT to model-free reinforcement learning algorithms (Haarnoja et al., 2018; Mnih et al., 2016). They demonstrated that PAD improves performance under distribution shifts on the DeepMind Control (Tassa et al., 2018) and CRLMaze (Lomonaco et al., 2020) benchmarks. They also showed that PAD enables more effective sim-to-real transfer for robotic manipulation tasks.

A.2 Additional Ablations

Test-Time Optimization Objective. In order to clarify that the performance gains achieved by TT-MBRL are the results of its key differences rather than solely the product of applying test-time adaptation, we perform a comparison with PAD and study the impact of loss components. Table 9 shows the impact of test-time objectives on performance. As expected, we find that the KL objective is essential to TT-MBRL success. Removing the KL objective (w/o KL loss) causes the agent to fail because the dynamics predictor cannot adapt to moving world model representations and makes incorrect predictions. On the other hand, the agent appears robust to removing the reward and continue objectives (w/o rwd-cont loss), signaling that it is not essential to TT-MBRL success. However, given the dependency of reward and continue MLP heads on moving representations, we suppose this design choice could have a negative impact on performance in the long run. We also find that removing the actor-critic objective (w/o actor-critic) can decrease performance. The performance of the agent deteriorates along the episode, given that the actor and critic cannot adapt to moving representations. We also experiment with PAD (Hansen et al., 2021), which was originally proposed in the model-free context. PAD only adapts encoder parameters using a self-supervised objective and does not use training samples for regularization, leaving policy and value networks unchanged at test time. As expected, we find that PAD has a detrimental effect on performance despite relying on test-time training.

Table 9: Ablation study on Test-Time Optimization objective.

Method	PAD	TT-MBRL	w/o KL loss	w/o rwd-cont loss	w/o actor-critic loss
DreamerV3	25.5	185.0	12.3	**187.9**	47.8
TWISTER	35.5	240.1	12.9	**240.9**	50.2
R2I	42.3	**244.3**	14.9	242.7	158.5

TT-MBRL Train/Test Sampling Ratio. The default TT-MBRL setting uses a 50/50 sampling ratio for train/test trajectories. To confirm that our method is robust to this hyper-parameter choice and does not depend on a narrow setting, we study the impact of the train/test ratio by varying the ratio of training samples among the global batch. Table 10 shows the results that we obtained on Memory Maze 2D 15×15. We find that sampling only training trajectories (100/0) does not improve performance because the world model cannot learn about episode-specific patterns. Using only test

trajectories (0/100) decreases performance, confirming the important role of training trajectories in regularization. Overall, we find that a sampling ratio between 25% and 75% achieves robust performance, showing the flexibility of this hyperparameter.

Table 10: Ablation study on TT-MBRL train/test sampling ratio.

Method	**w/o TT-MBRL**	**100/0 (train-only)**	**75/25**	**50/50**	**25/75**	**0/100 (test-only)**
DreamerV3	46.2	47.0	160.9	**185.0**	179.4	117.6
TWISTER	72.0	69.2	**244.5**	240.1	235.2	167.6
R2I	171.6	171.7	242.5	244.3	**249.7**	218.8

Test-Time Gradient Update Frequency. We perform a study on compute–performance tradeoff by varying the number of policy steps per gradient step. Table 11 shows the impact of test-time gradient update frequency on TT-MBRL efficacy. We find that increasing the update frequency by a factor of 2 does not lead to a significant performance improvement while increasing evaluation time by the same factor. Decreasing the frequency of TT-MBRL updates slows down adaptation, which results in lower final performance. However, we find that TT-MBRL is quite robust to this hyper-parameter, achieving significant improvement with a larger factor of policy steps per gradient step while also decreasing evaluation time by the same factor. We find these results encouraging for the development of an efficient manner to apply TT-MBRL.

Table 11: Study on test-time gradient update frequency.

Policy Step/Grad Step	**w/o TT-MBRL**	**32**	**16**	**8**	**4**	**2**	**1**	**1/2**
DreamerV3	46.2	79.9	107.3	132.1	152.9	165.7	**185.0**	178.7
TWISTER	72.0	108.9	139.8	178.3	198.5	225.8	240.1	**245.6**
R2I	171.6	182.3	190.2	205.8	213.6	230.9	244.3	**251.8**

A.3 Actor Critic Learning

Analogously to world model predictor networks, actor and critic networks are designed as simple MLPs with parameter vectors (θ) and (ψ), respectively:

$$\text{Actor:} \quad a_t \sim \pi_\theta(a_t|s_t), \quad \text{Critic:} \quad v_\psi(s_t) \approx \mathrm{E}_{p_\phi,\pi_\theta}[R_t^\lambda] \tag{7}$$

The critic network learns to minimize the symlog-transformed cross-entropy loss (Hafner et al., 2023) with discretized λ-returns obtained from imagined trajectories:

$$R_t^\lambda = \hat{r}_{t+1} + \gamma \hat{c}_{t+1}\Big((1-\lambda)v_\psi(s_{t+1}) + \lambda R_{t+1}^\lambda\Big) \qquad R_{H+1}^\lambda = v_\psi(s_{H+1}) \tag{8}$$

The critic also relies on its own predictions for estimating rewards beyond the prediction horizon, which requires adding a regularizing term toward the outputs of its own EMA network. Equation 9 defines the critic loss as follows:

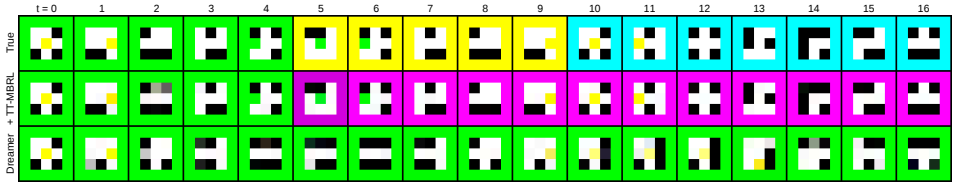
$$L_{\text{critic}}(\psi) = \frac{1}{BH}\sum_{b=1}^{B}\sum_{t=1}^{H}\Big[\underbrace{-\log p_\psi(R_t^\lambda \mid s_t)}_{\text{discrete returns regression}} \underbrace{-\log p_\psi(v'_\psi(s_t) \mid s_t)}_{\text{critic EMA regularizer}}\Big] \tag{9}$$

The actor network is trained to select actions that maximize obtained advantages using Reinforce (Williams, 1992). It also maximizes the policy entropy to ensure sufficient exploration during both data collection and imagination. The actor loss is defined as follows:

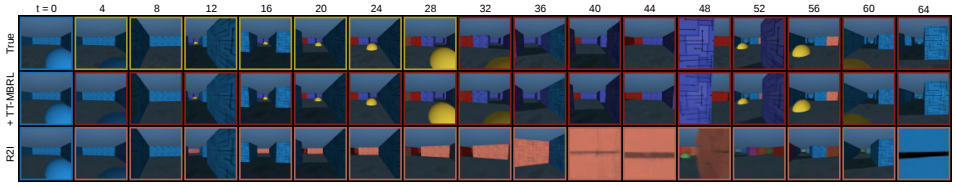
$$L_{\text{actor}}(\theta) = \frac{1}{BH}\sum_{b=1}^{B}\sum_{t=1}^{H}\Big[\underbrace{-\log \pi_\theta(a_t \mid s_t) sg(A_t^\lambda)}_{\text{reinforce}} \underbrace{-\eta \mathrm{H}[\pi_\theta(a_t \mid s_t)]}_{\text{entropy regularizer}}\Big] \tag{10}$$

Where $A_t^\lambda = \big(\hat{R}_t^\lambda - v_\psi(s_t)\big)/\max(1, S)$ defines advantages computed using normalized returns. The returns are scaled using exponentially moving average statistics of their 5th and 95th batch percentiles to ensure stable learning across environments with different return scales:

$$S = \mathrm{EMA}(\mathrm{Per}(R_t^\lambda, 95) - \mathrm{Per}(R_t^\lambda, 5), momentum = 0.99) \tag{11}$$



(a) DreamerV3 world model predictions over 16 time steps: Memory Maze 2D 15×15



(b) R2I world model predictions over 64 time steps: Memory Maze 15×15

Figure 6: Impact of TT-MBRL on world model predictions and actor-critic learning. We let the agent explore and learn about the environment, then we compare world model predictions to real observations. First row shows unseen real environment observations. Second row shows world model predictions with TT-MBRL. Third row shows world model predictions without TT-MBRL. Without test-time optimization, the model accumulates prediction errors and struggles to make accurate long-term predictions. This makes it difficult to learn an optimal policy for the actor-critic.

A.4 Experiments Compute Resources

We perform Memory Maze 2D and Memory Maze experiments on Nvidia L40 GPUs with 48GB of memory along with 40 CPU workers. During training, we use two GPUs for training R2I on Memory Maze experiments while a single GPU is sufficient for other experiments. For DMC and Atari 100K benchmarks, we used RTX 3090 GPUs with 24GB of memory along with 16 CPU workers. A single GPU per task was sufficient for the two benchmarks. Table 12 describes the execution time and resources required for each experiment.

Table 12: Resources and execution time used for each experiments.

Environment	#Tasks	Method	GPU	#GPUs	#Env Steps	Training Time (days/task)
Memory Maze 2D	4	DreamerV3	L40	1	100M	8
Memory Maze 2D	4	TWISTER	L40	1	100M	6
Memory Maze 2D	4	R2I	L40	1	100M	2.9
Memory Maze	4	DreamerV3	L40	1	100M	13.5
Memory Maze	4	TWISTER	L40	1	100M	23
Memory Maze	4	R2I	L40	2	400M	28
DMC	20	DreamerV3	RTX 3090	1	1M	0.8
Atari 100k	26	DreamerV3	RTX 3090	1	400K	0.35

A.5 Limitations and Future Work.

The primary limitation of our approach is the reduced evaluation efficiency resulting from the use of gradient-based optimization at test time. TT-MBRL requires learning a distinct world model and behavior policy for each evaluation episode, leading to increased computational cost and longer evaluation times. While this overhead remains manageable in environments such as Memory Maze, it may become prohibitive in settings that require a large number of environment interactions. As a direction for future work, we plan to extend our approach to operate during the exploration phase. We hypothesize that integrating TT-MBRL into exploration could improve sample efficiency and overall performance by enabling the collection of more informative and diverse trajectories. Consequently, improving the computational efficiency of our approach will be crucial to making it practical for online adaptation during exploration.

A.6 MEMORY MAZE 2D ENVIRONMENT

We design Memory Maze 2D to evaluate the capacity of agents to develop long-term memory by exploring a partially observable environment. Memory Maze 2D serves as a simplified and more efficient version of the Memory Maze environment, where the agent observes a top-down 3×3 local context centered on its current position. To facilitate goal identification, we augment the observation with colored borders that indicate the target color to be reached for obtaining the next reward. The observation is then flattened and normalized, forming a vector of dimension $75 = 5 \times 5 \times 3$. The agent operates in the maze using a discrete action space consisting of five actions: no-op, move left, move right, move up, and move down. Following (Pašukonis et al., 2023), we design four maze tasks with varying sizes and difficulty: Memory Maze 2D 9×9, Memory Maze 2D 11×11, Memory Maze 2D 13×13, and Memory Maze 2D 15×15. Table 13 shows the parameters associated with each maze size and Figure 7 depicts randomly generated environment maps.

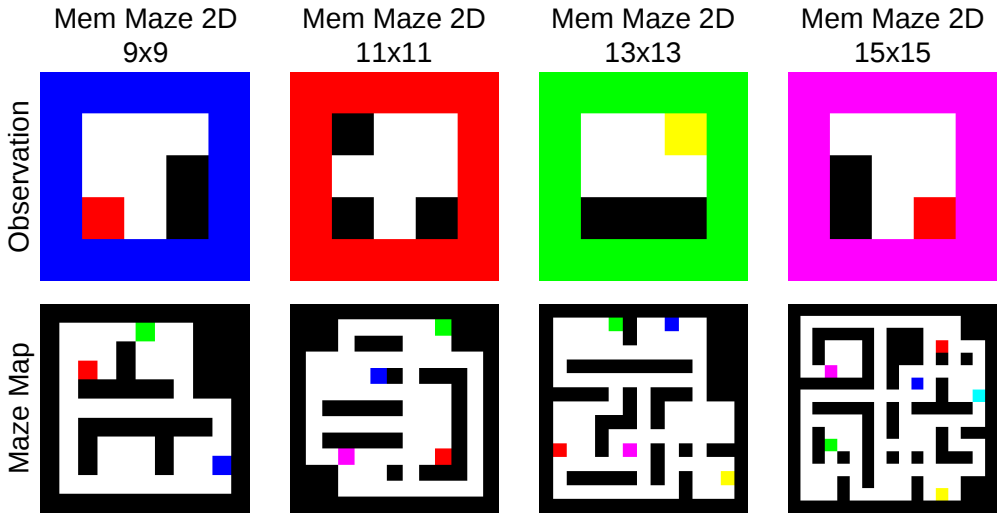


Figure 7: Randomly generated environment map for Memory Maze 2D (9×9, 11×11, 13×13 and 15×15). The agent receives a top-down 3×3 local context centered on its current position and must reach the target color to receive the next reward.

Table 13: Memory Maze 2D environment details.

Parameter	Mem Maze 2D 9×9	Mem Maze 2D 11×11	Mem Maze 2D 13×13	Mem Maze 2D 15×15
Number of actions	5	5	5	5
Observation dim	75	75	75	75
Number of colors	3	4	5	6
Number of rooms	$3-4$	$4-6$	$5-6$	9
Room size	$3-5$	$3-5$	$3-5$	3
Episode length	1000	2000	3000	4000
Mean maximum score (Oracle)	137.2	215.4	276.6	311.4

Table 14: Environment efficiency comparison. Episode average step time measured on a single Intel Core i7-12700 4.9GHz CPU thread and a RTX 3090 GPU.

Step time (ms)	9×9	11×11	13×13	15×15
Memory Maze 2D	0.16	0.16	0.16	0.16
Memory Maze	47.28	61.82	76.48	90.36

A.7 Algorithm Pseudocode

Algorithm 1 TT-MBRL, our proposed algorithm for applying model-based reinforcement learning at test time. At the beginning of each evaluation episode, the agent initializes test-time parameters (ϕ', θ', ψ') from the trained parameters (ϕ, θ, ψ) and creates a dedicated test-time replay buffer $\mathbf{R}'$ to collect episode observations. At each environment step, the world model is updated by sampling from both the test-time buffer $\mathbf{R}'$ and the training buffer $\mathbf{R}$, acting as a regularizer that prevents overfitting to limited early observations. The actor-critic is then updated using the adapted world model, enabling the agent to learn an improved policy.

Require: Trained parameters (ϕ, θ, ψ), training replay buffer $\mathbf{R}$, and environment env

```
while not done evaluation do
    (φ', θ', ψ') ← (φ, θ, ψ)          ▷ Initialize test-time parameters from training.
    R' ← {}                            ▷ Initialize empty test-time replay buffer.
    obs ← ENV_RESET(env)               ▷ Reset environment.
    R' ← R' ∪ {obs}                    ▷ Store initial observation.
    while not done episode do
        obs ← ENV_STEP(env, φ', θ')    ▷ Step environment using current policy.
        R' ← R' ∪ {obs}                ▷ Update test-time replay buffer.
        φ' ← TRAIN_WORLD_MODEL(R, R')  ▷ Update world model using train-test data.
        (θ', ψ') ← TRAIN_ACTOR_CRITIC(φ')  ▷ Update actor-critic using world model.
    end while
end while
```

A.8 Experimental Details and Hyperparameters

Table 15: World model sizes. N/A refers to the use of a MLP encoder for low-dimensional inputs, and the removal of the AC-CPC objective in the case of TWISTER.

(a) DreamerV3 model sizes.

Environment	Mem Maze 2D	Mem Maze	DMC	Atari 100k
CNN Base Dim	N/A	96	32	32
Hidden Size	512	1024	512	512
GRU Recurrent Units	512	4096	512	512
MLP Layers	2	5	2	2
Number Params	8.7M	200.7M	17.9M	17.9M

(b) TWISTER model sizes.

Environment	Mem Maze 2D	Mem Maze
CNN Base Dim	N/A	48
Hidden Size	512	640
Transformer Blocks	4	5
Feed-Forward Ratio	2	3
Number of Heads	8	10
Dropout Prob	0.1	0.1
MLP Layers	2	3
Attention Context	256	256
AC-CPC Steps	N/A	10
Number params	15.9M	58.8M

(c) R2I model sizes.

Environment	Mem Maze 2D	Mem Maze
CNN Base Dim	N/A	48
Hidden Size	1024	2048
SSM Layers	3	5
SSM State Dim	128	512
SSM Output Dim	1024	1024
MLP Layers	2	4
Number params	11.6M	92.8M

Table 16: DreamerV3 hyperparameters.

Hyperparameter	Memory Maze 2D	Memory Maze
General		
Batch Size (B)	64	16
Sequence Length (T)	64	64
Environment Instances	8	32
Policy Step per Grad Step	64	64
Replay Buffer Capacity	10M	10M
TT-MBRL Policy Step / Grad Step	1	1
TT-MBRL Batch Size	32	8
TT-MBRL Train/Test Ratio	50/50	50/50
World Model		
Prediction Loss Scale (β_{pred})	1.0	1.0
Dynamics Loss Scale (β_{dyn})	0.5	0.5
Representation Loss Scale (β_{rep})	0.1	0.1
Stochastic State Features	32	32
Classes per Feature	32	32
Learning Rate	10^{-4}	10^{-4}
Gradient Clipping	1000	1000
Actor-Critic		
Imagination Horizon (H)	15	15
Return Discount (γ)	0.997	0.997
Return Lambda (λ)	0.95	0.95
Critic EMA Decay	0.98	0.98
Return Normalization Momentum	0.99	0.99
Actor Entropy Scale (η)	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Learning Rate	$3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$
Gradient Clipping	100	100

Table 17: TWISTER hyperparameters.

Hyperparameter	Memory Maze 2D	Memory Maze
General		
Batch Size (B)	4	4
Sequence Length (T)	1024	1024
Environment Instances	8	32
Policy Step per Grad Step	64	64
Replay Buffer Capacity	10M	10M
TT-MBRL Policy Step / Grad Step	1	1
TT-MBRL Batch Size	2	2
TT-MBRL Train/Test Ratio	50/50	50/50
World Model		
Prediction Loss Scale (β_{pred})	1.0	1.0
Dynamics Loss Scale (β_{dyn})	0.5	0.5
Representation Loss Scale (β_{rep})	0.1	0.1
Stochastic State Features	32	32
Classes per Feature	32	32
Learning Rate	10^{-4}	10^{-4}
Gradient Clipping	1000	1000
Actor-Critic		
Imagination Horizon (H)	15	15
Return Discount (γ)	0.997	0.997
Return Lambda (λ)	0.95	0.95
Critic EMA Decay	0.98	0.98
Return Normalization Momentum	0.99	0.99
Actor Entropy Scale (η)	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Learning Rate	$3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$
Gradient Clipping	100	100

Table 18: R2I hyperparameters.

Hyperparameter	Memory Maze 2D	Memory Maze
General		
Batch Size (B)	4	4
Sequence Length (T)	1024	1024
Environment Instances	8	40
Policy Step per Grad Step	64	64
Replay Buffer Capacity	10M	10M
TT-MBRL Policy Step / Grad Step	1	1
TT-MBRL Batch Size	2	2
TT-MBRL Train/Test Ratio	50/50	50/50
World Model		
Prediction Loss Scale (β_{pred})	1.0	1.0
Dynamics Loss Scale (β_{dyn})	0.8	0.8
Representation Loss Scale (β_{rep})	0.2	0.2
Stochastic State Features	32	32
Classes per Feature	32	32
Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Gradient Clipping	200	200
Actor-Critic		
Imagination Horizon (H)	15	15
Return Discount (γ)	0.997	0.997
Return Lambda (λ)	0.95	0.95
Critic EMA Decay	0.98	0.98
Return Normalization Momentum	0.99	0.99
Actor Entropy Scale (η)	10^{-3}	10^{-3}
Learning Rate	10^{-4}	10^{-4}
Gradient Clipping	200	200

A.9 TRAINING CURVES

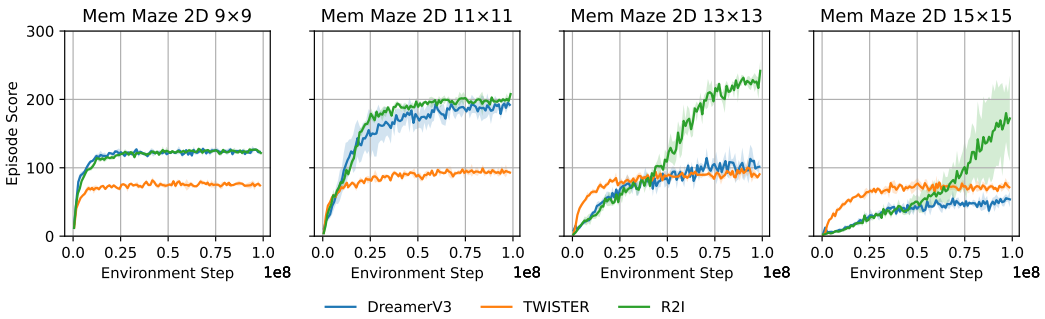


Figure 8: Memory Maze 2D training curves (100M environment steps).

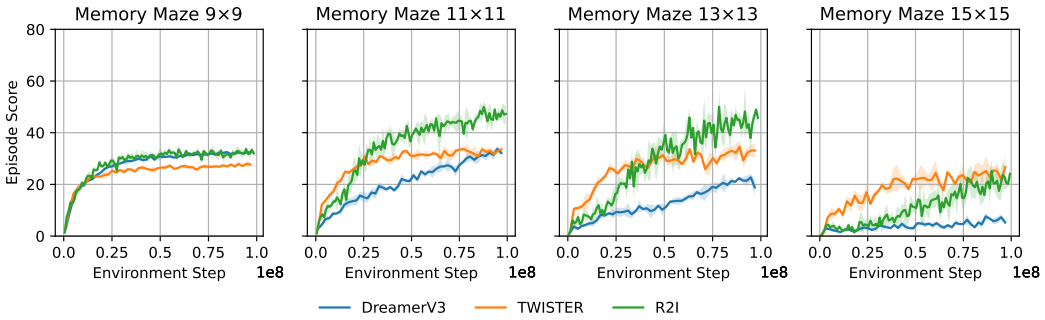


Figure 9: Memory Maze training curves (100M environment steps).

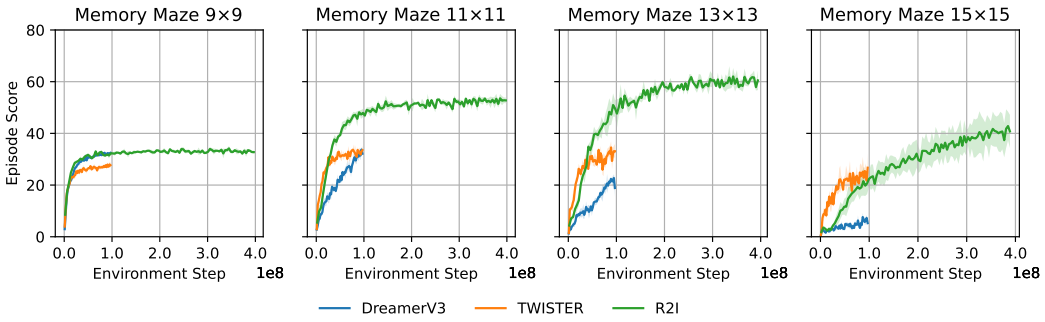


Figure 10: Memory Maze training curves (400M environment steps).

A.10 EVALUATION HISTOGRAMS

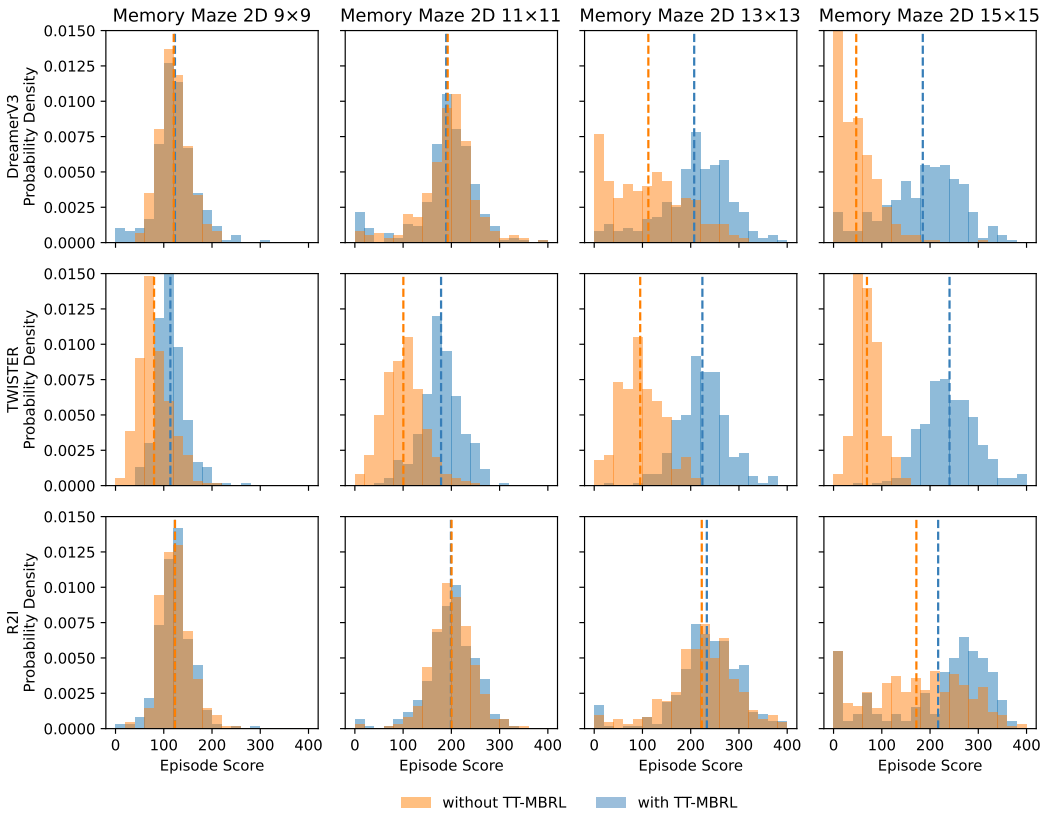


Figure 11: Evaluation episode return distribution on Memory Maze 2D (100M environment steps). We compare the impact of TT-MBRL on episode return for DreamerV3, TWISTER and R2I.

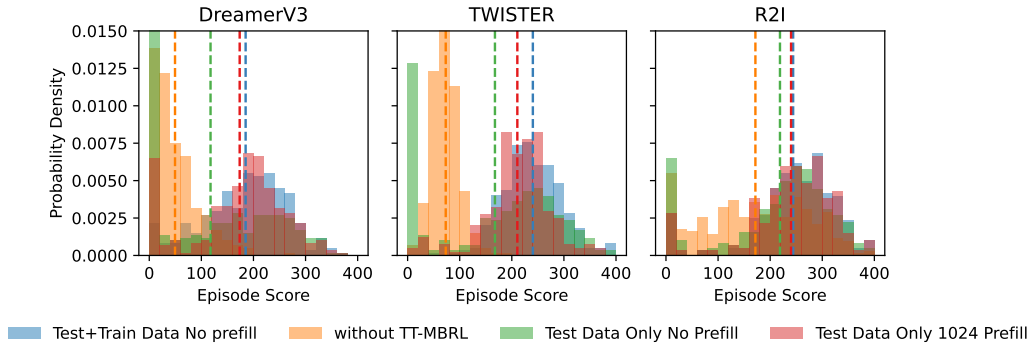


Figure 12: Evaluation episode return distribution on Memory Maze 2D 15×15. We compare the impact of training trajectories regularization and replay buffer prefill on TT-MBRL efficacy.

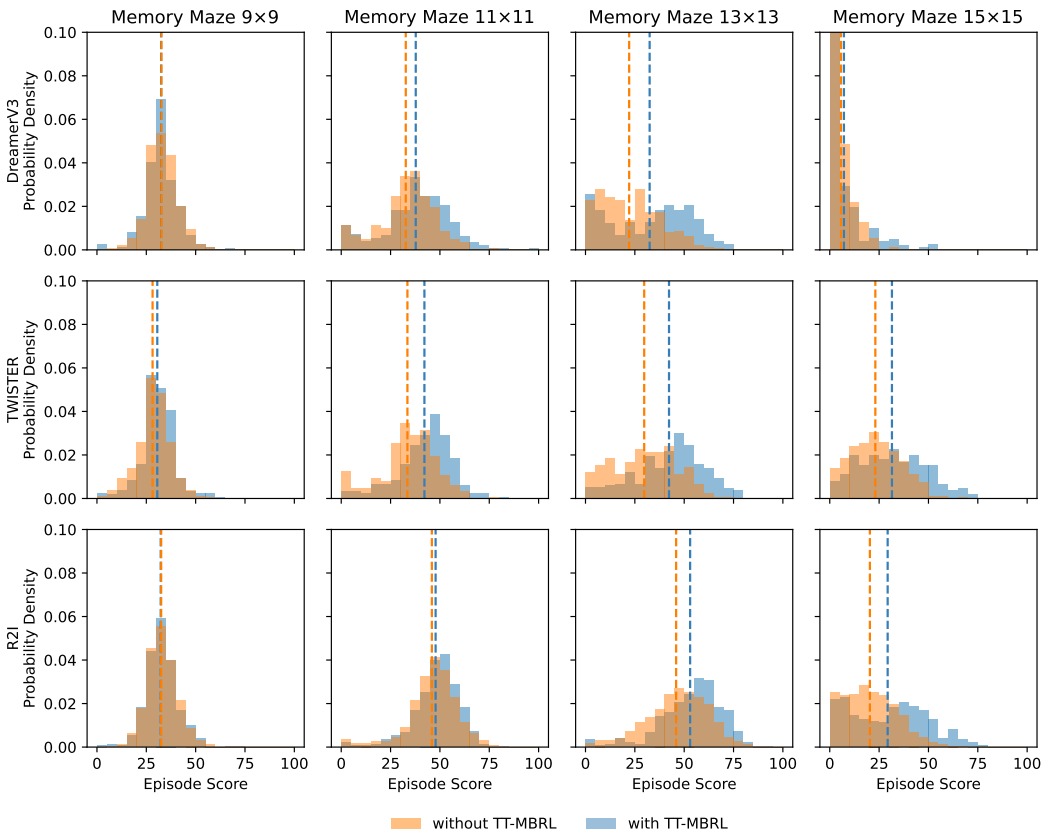


Figure 13: Evaluation episode return distribution on Memory Maze (100M environment steps). We show the impact of TT-MBRL on episode return for DreamerV3, TWISTER and R2I.

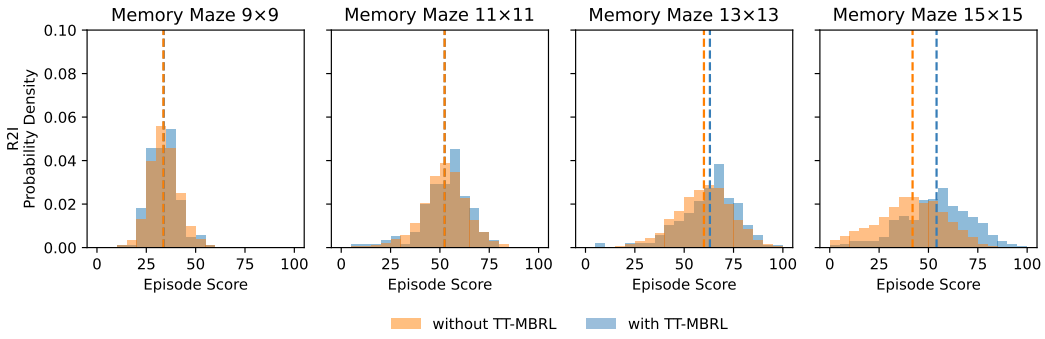


Figure 14: Evaluation episode return distribution on Memory Maze (400M environment steps). We show the impact of TT-MBRL on episode return for R2I.

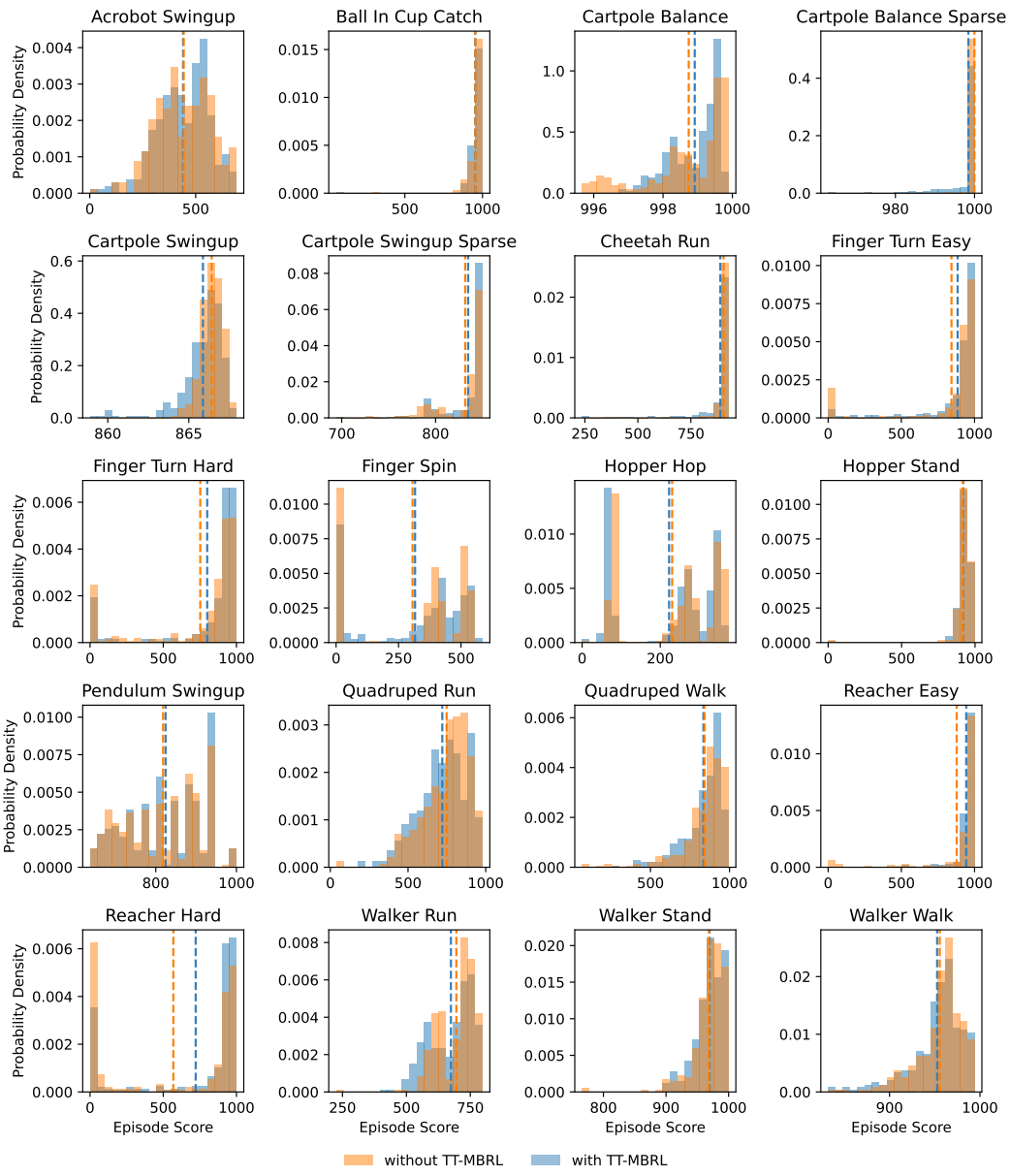


Figure 15: Evaluation episode return distribution on the DeepMind Control Suite (DMC). We show the impact of TT-MBRL on episode return for DreamerV3.

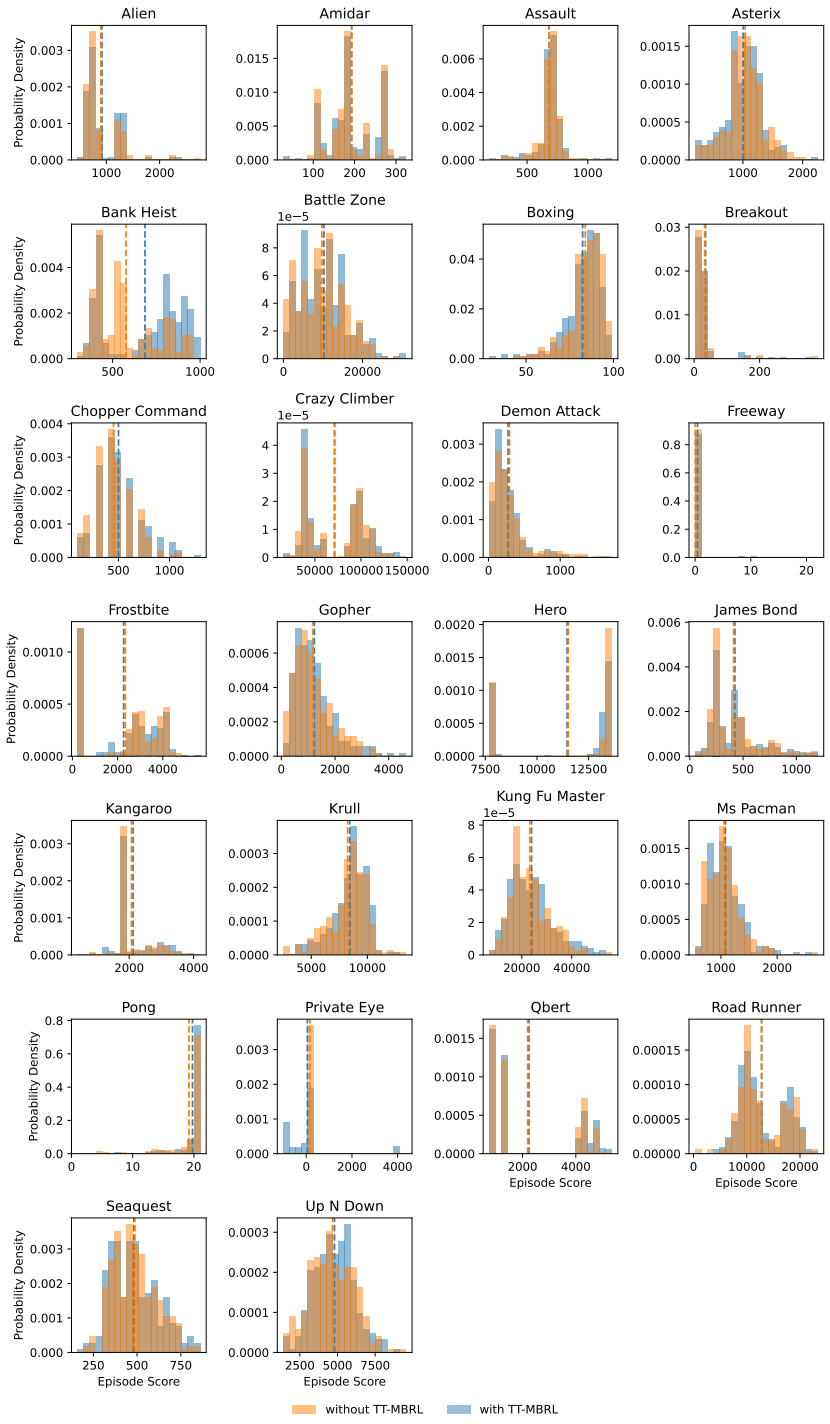


Figure 16: Evaluation episode return distribution on the Atari 100k benchmark. We show the impact of TT-MBRL on episode return for DreamerV3.

A.11 TT-MBRL Relative Improvement

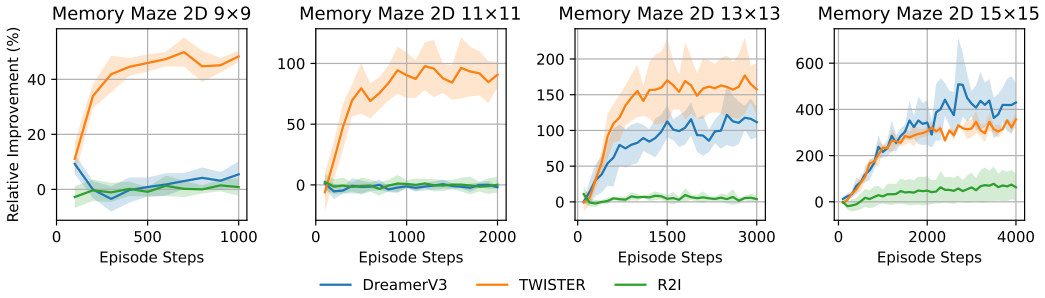


Figure 17: Relative improvement in obtained rewards brought by TT-MBRL on Memory Maze 2D.

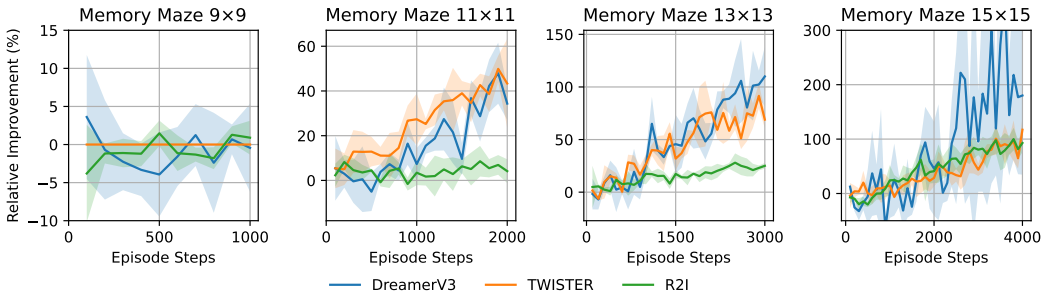


Figure 18: Relative improvement in obtained rewards brought by TT-MBRL on Memory Maze (100M environment steps).

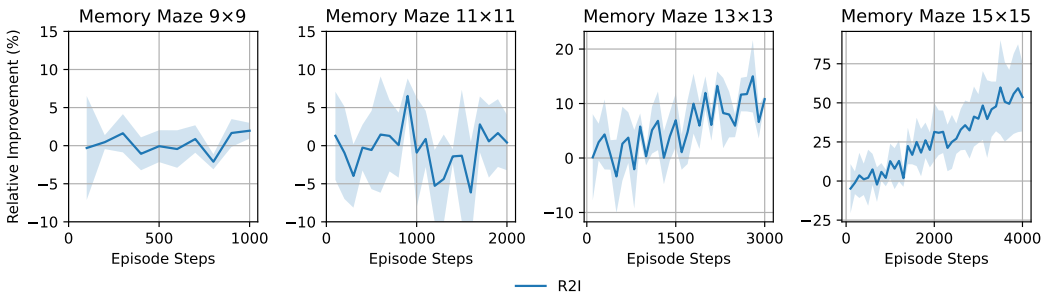


Figure 19: Relative improvement in obtained rewards brought by TT-MBRL on Memory Maze (400M environment steps).

A.12 World Model Prediction Metrics

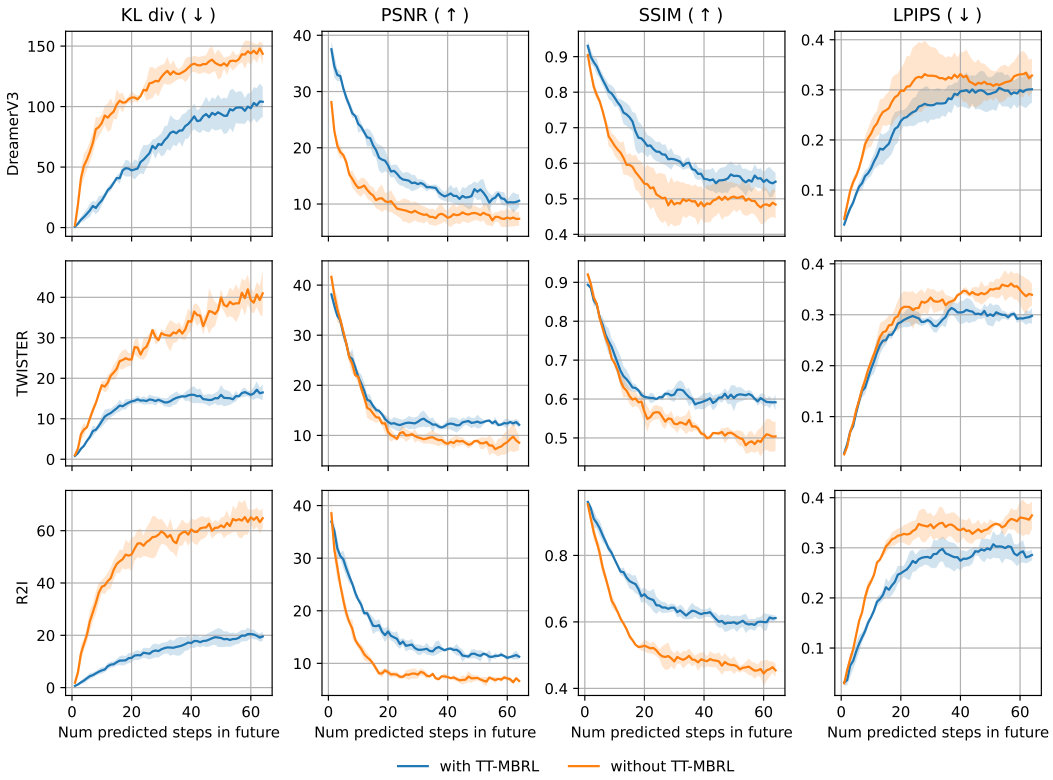


Figure 20: DreamerV3, TWISTER and R2I world model prediction metrics on Memory Maze 2D 15×15.

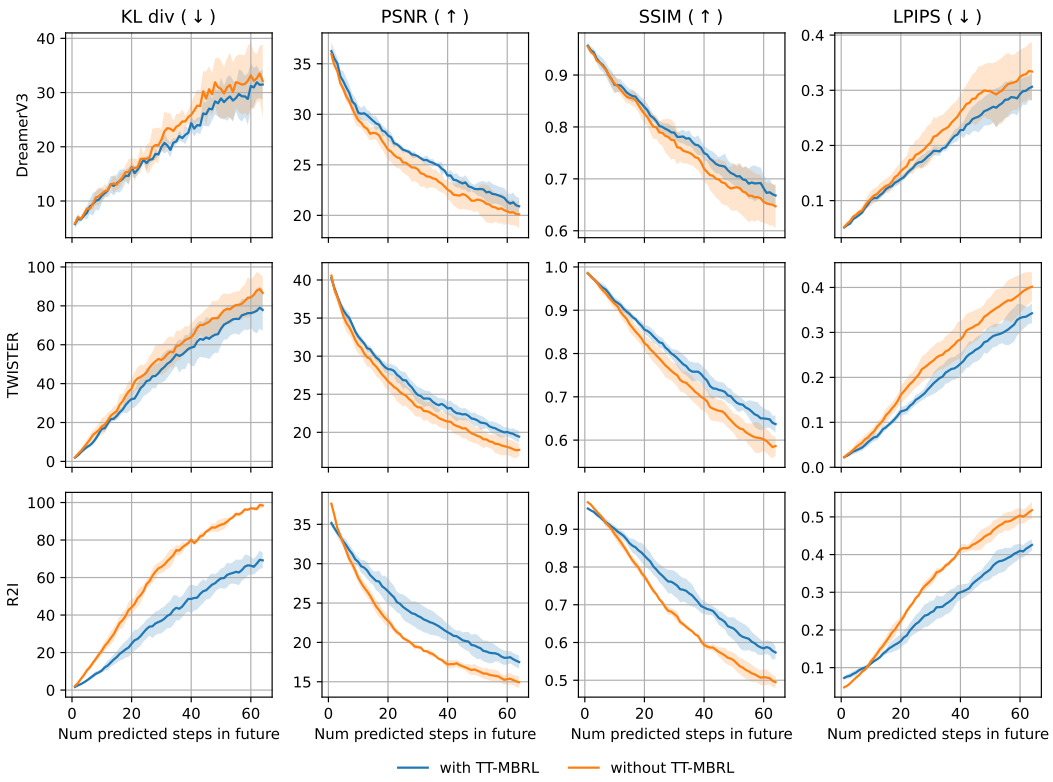


Figure 21: DreamerV3, TWISTER and R2I world model prediction metrics on Memory Maze 15×15 (100M environment steps).

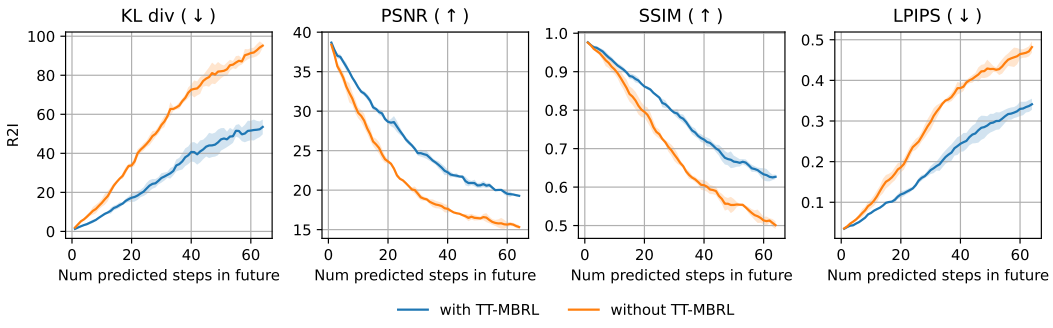


Figure 22: R2I world model prediction metrics on Memory Maze 15×15 (400M environment steps).

A.13 World Model Predictions

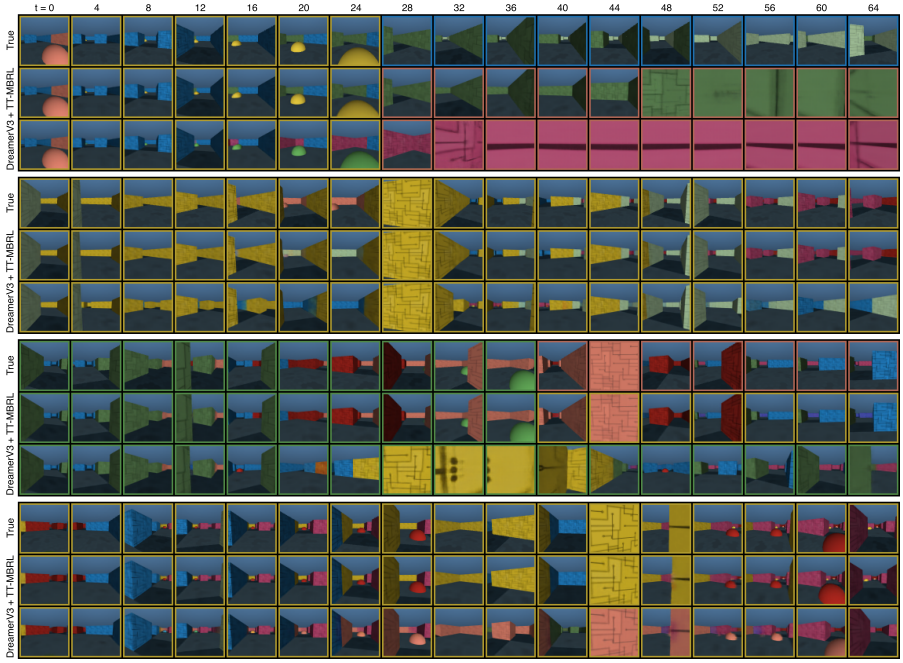


Figure 23: DreamerV3 world model predictions on Memory Maze.

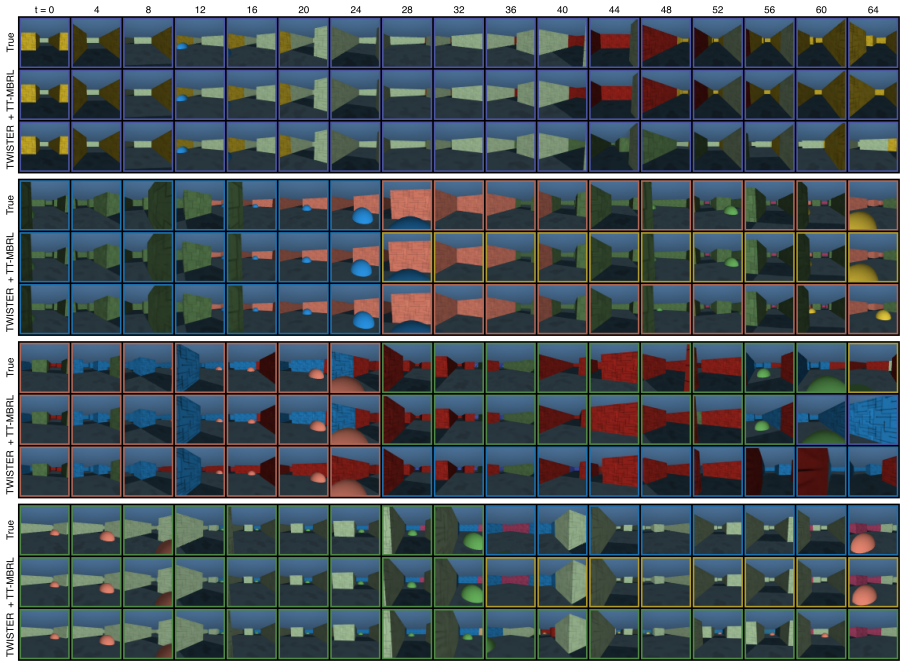


Figure 24: TWISTER world model predictions on Memory Maze.

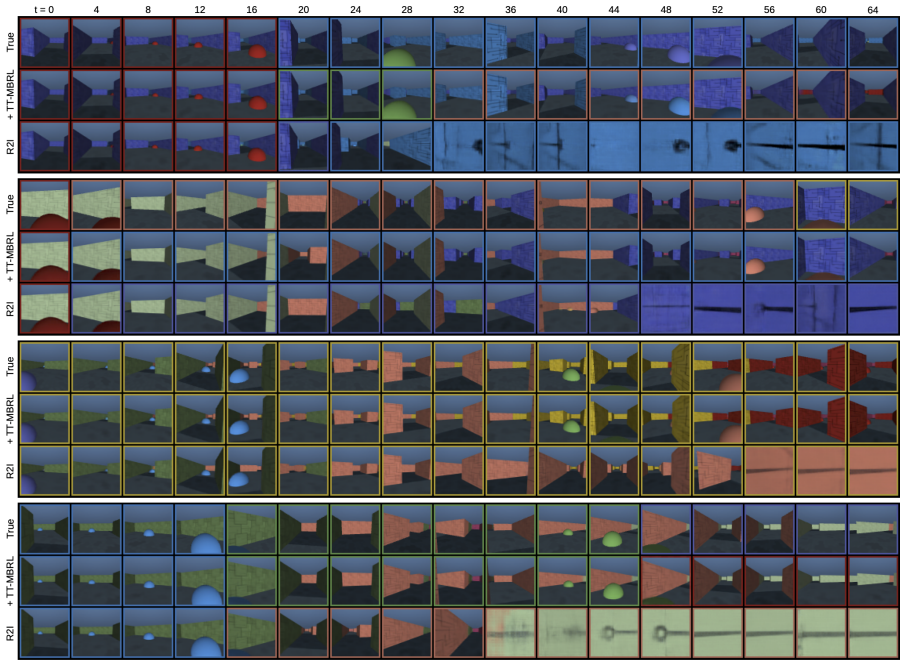


Figure 25: R2I world model predictions on Memory Maze.

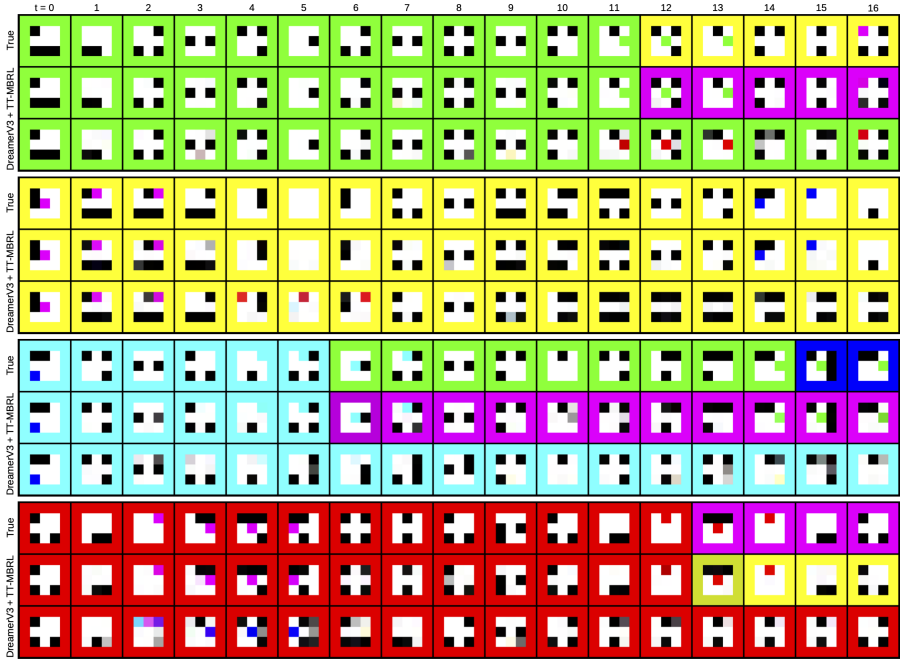


Figure 26: DreamerV3 world model predictions on Memory Maze 2D.

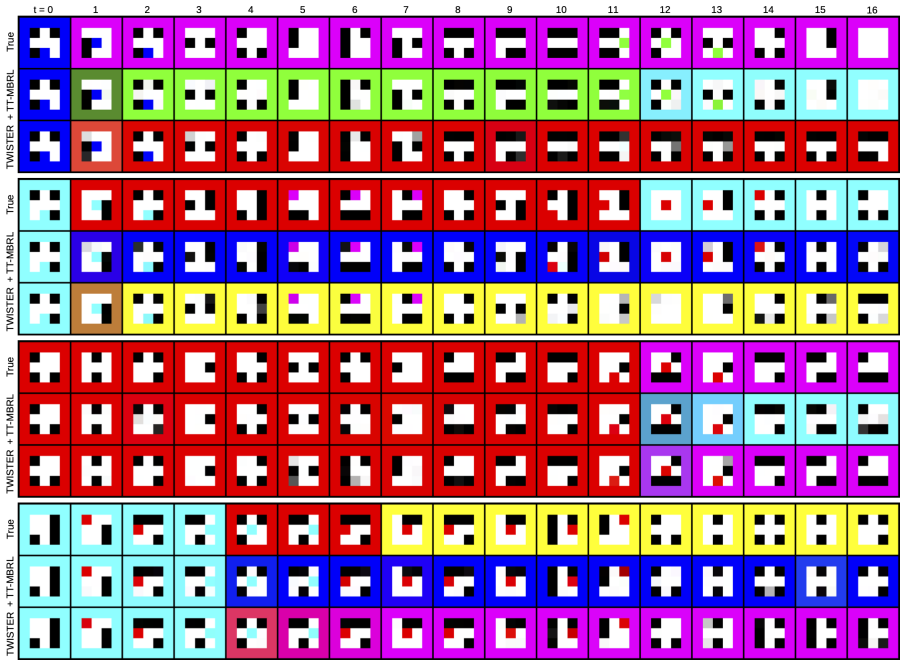


Figure 27: TWISTER world model predictions on Memory Maze 2D.

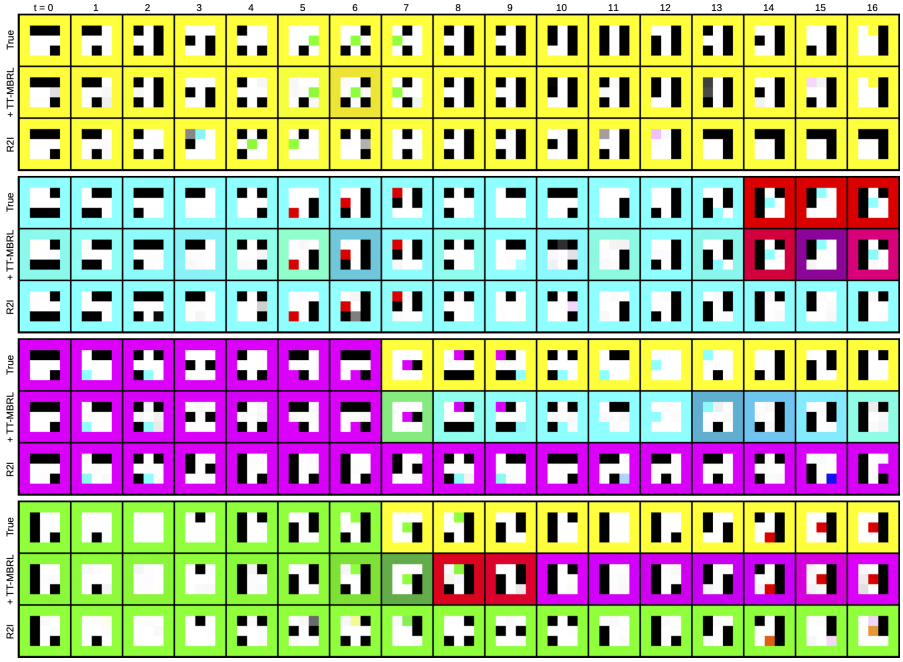


Figure 28: R2I world model predictions on Memory Maze 2D.