JEPA-TTT: PERSISTENT TEST-TIME TRAINING OF LATENT WORLD MODELS FOR PLANNING UNDER DYNAMICS SHIFTS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models enable agents to plan by predicting future states of the environment, but their predictions can become unreliable when test-time dynamics differ from those seen during training. We present JEPA-TTT, which adapts the latent dynamics predictor of a pretrained action-conditioned Joint-Embedding Predictive Architecture world model throughout test time. Self-supervised updates accumulate across episodes, while the visual encoder and reward head remain fixed, preserving the pretrained representation and task objective. Planning requires neither a goal image nor online environment reward. JEPA-TTT uses dense replay, which forms prediction windows at every temporal offset, retains them in a growing buffer, and samples minibatches from that buffer for predictor updates. Across eight dynamics shifts in four continuous-control environments, JEPA-TTT improves planning on every shift. After 500 test-time episodes, it reduces autoregressive latent prediction error by 83% on average and improves planning performance by 153% over the frozen JEPA world model. These results show that persistent self-supervised test-time training can adapt a pretrained latent world model under changed dynamics.

1 INTRODUCTION

Agents can learn predictive models of environment dynamics (i.e., world models) to support decision-making (Ha & Schmidhuber, 2018). Action-conditioned Joint-Embedding Predictive Architecture (JEPA) world models predict future latent representations from observations and actions (LeCun, 2022; Maes et al., 2026). The effectiveness of a world model for planning depends critically on the accuracy of its learned dynamics. Conventional world models typically assume that the transition dynamics are the same during training and testing (Hafner et al., 2019b; Maes et al., 2026). However, in practice, this assumption can be violated by changes in actuator responses, contact dynamics, or object motion, causing a model trained under one set of dynamics to produce systematic prediction errors at test time and thus leading to poor planning performance.

Figure 1 illustrates this failure: a pretrained world model predicts the training dynamics, while the action produces a different outcome after the dynamics shifts in the test environment. Given the current observation o_t, the agent executes action a_t and receives the next observation o_{t+1}. The transition (o_t, a_t, o_{t+1}) provides a self-supervised target for updating the dynamics model.

We study whether a pretrained action-conditioned JEPA world model can adapt to new dynamics throughout test time and thereby improve planning. We introduce JEPA-TTT, which adapts the model's latent dynamics predictor through self-supervised updates that accumulate across episodes. Since JEPA learns predictive representations without a reconstruction objective, it eliminates the computational cost of a decoder, making it a practical choice for test-time training under a limited compute budget.

The model's visual encoder maps observations to latent states, and its predictor predicts how actions change those states. JEPA-TTT updates only the predictor using executed actions and consecutive encoded observations. The visual encoder and offline-trained reward head remain fixed, preserving the visual representation and task objective. During planning, model predictive control (MPC) uses the adapted predictor and fixed reward head to rank candidate actions without requiring goal images

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

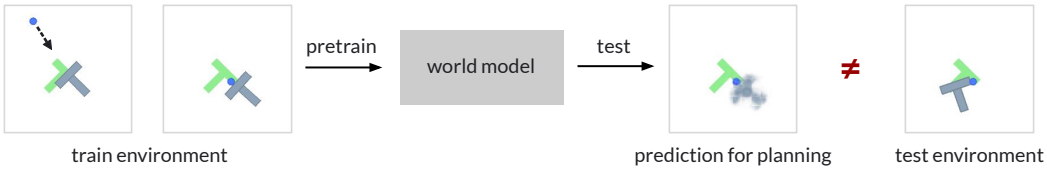


Figure 1: A dynamics shift can cause a pretrained world model to fail during planning. The model predicts the outcome of a candidate action based on the dynamics learned during training, but the test environment has different transitions. This mismatch can cause the planner to select wrong actions.

or online environment rewards. To train from online experience, JEPA-TTT uses dense replay, which forms prediction windows at every temporal offset and samples minibatches from a growing buffer.

We evaluate two controlled dynamics shifts in each of four continuous-control environments. The shifts span changes to actuation, contact dynamics, and state-dependent action mappings. JEPA-TTT improves planning on all eight shifts. It achieves a mean normalized held-out planning-score AUC of 0.571, compared with 0.267 for the frozen JEPA world model, and raises mean best held-out score from 0.267 to 0.678. After 500 test-time episodes, the adapted predictor reduces autoregressive latent prediction error by 83% on average. We also conduct ablation studies to assess persistence, temporal coverage, and replay.

Contributions. To summarize, our main contributions are: (1) JEPA-TTT, which adapts a pretrained JEPA predictor for planning without goal images or online environment rewards, using an offline-trained reward head; (2) consistent improvements in planning and latent prediction across eight dynamics shifts; and (3) controlled ablations of test-time training.

2 Related Work

Latent World Models and Joint-Embedding Predictive Architectures. Generative world models perform prediction in observation space or through explicit scene representations (Finn & Levine, 2017; Du et al., 2023; 2024; Bar et al., 2025; Zhang et al., 2025; Zhen et al., 2025; Hafner et al., 2025; Zhang et al., 2026; Yin et al., 2026), while latent world models predict compact representations that can discard visual details irrelevant to control (Ha & Schmidhuber, 2018; Hafner et al., 2019b;a). Dreamer learns behavior through latent imagination, while TD-MPC2 uses latent dynamics for trajectory optimization (Hafner et al., 2019a; Hansen et al., 2024). Joint-Embedding Predictive Architectures provide a particularly natural foundation for latent world modeling because they learn predictive representations without requiring pixel reconstruction (LeCun, 2022; Assran et al., 2023; Bardes et al., 2024). Recent works have developed action-conditioned JEPA world models specifically for planning. V-JEPA 2-AC post-trains an action-conditioned predictor for robot planning with image goals (Assran et al., 2025). DINO-WM predicts over frozen pretrained visual features and demonstrates zero-shot latent planning (Zhou et al., 2024). PLDM learns a latent dynamics model with VICReg-style regularization from offline data (Sobal et al., 2025a). LeJEPA introduces SIGReg for stable end-to-end JEPA training (Balestriero & LeCun, 2025), and LeWorldModel extends this approach to action-conditioned world modeling directly from pixels (Maes et al., 2026). Wang et al. (2026b) reshape latent trajectories to improve their suitability for optimization-based planning. These works primarily study world-model training and planning rather than persistent adaptation to a new dynamics regime. JEPA-TTT instead investigates how a pretrained JEPA world model adapts its dynamics online for dynamics shift during deployment.

Test-Time Training and Online Dynamics Adaptation. Test-time training adapts model parameters using data available during inference, typically through self-supervised objectives or entropy minimization (Sun et al., 2020; Wang et al., 2020). Continual and online variants carry adaptation state across sequential image and video inputs, allowing recent observations to inform subsequent predictions (Wang et al., 2022; 2025). Self-supervised policy adaptation updates visual representations without online rewards (Hansen et al., 2020). JEPA-TTT applies this principle to action-conditioned latent dynamics by using observed transitions as self-supervised targets for test-time training. Related work has also viewed parameter updates as a form of memory for long input

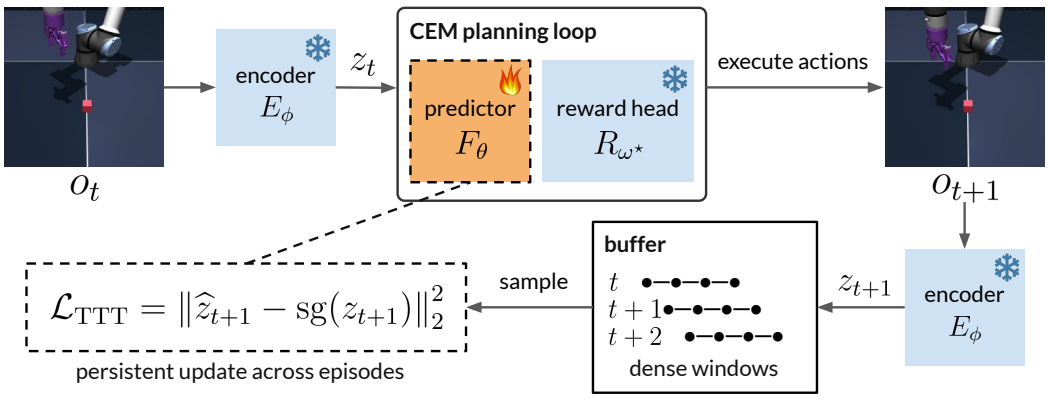


Figure 2: Overview of JEPA-TTT. Cross Entropy Method (CEM) plans with the current latent dynamics predictor F_θ and frozen reward head $R_{\omega^\star}$. After executing the actions, the frozen visual encoder maps the new observation o_{t+1} to a target latent z_{t+1}. JEPA-TTT stores prediction windows at every temporal offset and samples from the growing buffer to update F_θ with a self-supervised latent prediction objective. The adapted predictor is retained across episodes.

streams (Hardt & Sun, 2024; Sun et al., 2024). Online adaptation has a long history in learning-based control, including adaptive dynamics models for MPC (Nagabandi et al., 2018; Williams et al., 2020) and latent environment representations inferred from recent interaction histories (Kumar et al., 2021). Offline world models can also be finetuned through online model-based RL with environment rewards (Feng et al., 2023). Off-dynamics RL instead corrects the rewards used for source-domain policy training to account for differences in transition dynamics (Eysenbach et al., 2020). Feng et al. (2026) further explores test-time adaptation from human demonstrations in world-action models. Unlike these approaches, JEPA-TTT updates a pretrained JEPA world model directly from the agent's actions and observations in online transitions. The most closely related work is the concurrent work AdaJEPA, which also adapts a JEPA world model from observed transitions within an MPC loop (Wang et al., 2026a). AdaJEPA resets its adapted model and buffer at each episode, conditions planning on a goal image that is unavailable in our setting, and updates from a short within-episode history. In contrast, JEPA-TTT retains adaptation across episodes, plans with a frozen reward head and no goal image, and uses dense replay over accumulated transitions.

3 Persistent Test-Time Training of Latent World Models

3.1 Problem Setting

We study finite-horizon control after a single change in transition dynamics that remains fixed throughout test time. The task itself does not change: the state space, action space, observation mapping, score function, initial-state distribution, and episode horizon are identical before and after the shift. Only the transition dynamics differ. All offline training data are collected under the dynamics P_{train}, while testing is under dynamics P_{test}.

At time t, the environment has a physical state $s_t \in \mathcal{S}$. The agent receives an observation o_t, selects an action $a_t \in \mathcal{A}$, and the next state is sampled from $P(\cdot \mid s_t, a_t)$. Each episode lasts at most T steps. The dynamics shift satisfies

$$P_{\text{test}}(\cdot \mid s, a) \neq P_{\text{train}}(\cdot \mid s, a) \quad \text{for some } (s, a) \in \mathcal{S} \times \mathcal{A}. \tag{1}$$

The task defines a scalar score $r(s)$ that measures task progress. This score is available during offline training and used for evaluation. It is not observed during test-time training or planning. The objective is to recover high performance under P_{test} by adapting to the changed dynamics. Our experiments instantiate this setting with RGB observations and continuous actions.

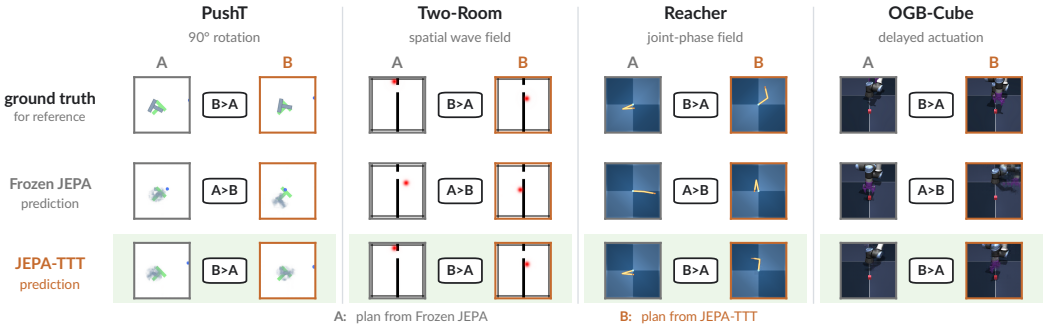


Figure 3: Test-time training can correct the ranking of candidate plans under dynamics shift. For each environment, Plan A is optimized using the frozen JEPA world model, and Plan B is optimized using JEPA-TTT from the same initial state. The frozen JEPA world model predicts that A will achieve a higher task score, but the ground-truth rollouts show that B is better. After adaptation, JEPA-TTT predicts the correct ordering. Both planners use the same frozen reward head $R_{\omega^\star}$, so the change in plan ranking comes from the adapted latent dynamics predictor. A pretrained decoder is used only to visualize latent predictions.

3.2 Action-Conditioned JEPA World Model and Planning

We use an action-conditioned JEPA world model (LeCun, 2022; Maes et al., 2026) with a visual encoder E_ϕ and latent dynamics predictor F_θ. The trainable predictor parameters θ include the action encoder, autoregressive predictor, and prediction projection. Observations are separated by a frame skip of K environment steps, and the K actions between consecutive observations are grouped into one action block. For a sampled trajectory, let x_k denote the observation at frame index k, let b_k denote the corresponding action block, and let $z_k = E_\phi(x_k)$ be the encoded latent. Given up to C previous latent states and action blocks, F_θ predicts the next latent state as $\widehat{z}_{k+1} = F_\theta(z_{k-C+1:k}, b_{k-C+1:k})$.

Because test-time planning receives neither a goal image nor an online environment reward, we train a reward head before deployment. The reward head maps frozen encoder features to the task score using observation-score pairs collected under the training dynamics. Let $\mathcal{D}^r_{\text{train}}$ denote this offline dataset of pairs (o, y), where $y = r(s)$:

$$\omega^\star = \arg\min_{\omega} \; \mathbb{E}_{(o,y)\sim\mathcal{D}^r_{\text{train}}} \left[\left(R_\omega(E_\phi(o)) - y \right)^2 \right]. \tag{2}$$

Both the visual encoder E_ϕ and the fitted reward head $R_{\omega^\star}$ remain frozen throughout test time. Freezing the encoder gives the test-time training objective a stable representation space, while freezing the reward head keeps the planning objective fixed as the dynamics predictor changes. Keeping the reward head fixed separates adaptation of environment dynamics from adaptation of the task objective. The planner must therefore improve by predicting the consequences of actions more accurately rather than by changing how predicted states are valued.

At each decision step, Cross-Entropy Method (CEM) (Rubinstein & Kroese, 2004) planning samples candidate sequences of H action blocks. The planning horizon spans HK environment steps. The current dynamics predictor rolls each candidate forward autoregressively in latent space, and the frozen reward head scores the predicted trajectory. For a candidate sequence $\mathbf{b} = (b_k, \ldots, b_{k+H-1})$, the planner maximizes the discounted sum of predicted return:

$$\widehat{J}_k(\mathbf{b}) = \sum_{h=1}^{H} \gamma^h R_{\omega^\star}(\widehat{z}_{k+h}(\mathbf{b}, \theta_k)). \tag{3}$$

CEM repeatedly refits its action distribution toward the highest-scoring candidates. The agent executes only the first action block of the optimized sequence, observes the updated state, and replans. Test-time training therefore changes planning only through the dynamics prediction. The encoder and reward head remain fixed.

Dynamics prediction error leads to incorrect planning because it changes the ordering of candidate actions. Figure 3 illustrates this failure mode. Under each shown shift, the frozen JEPA world

Algorithm 1: JEPA-TTT

Input: Pretrained predictor F_{θ_0}, frozen reward head $R_{\omega^\star}$, batch size B
Initialize once: $\theta \leftarrow \theta_0$, replay buffer $\mathcal{Q} \leftarrow \varnothing$;
for *each replanning step in test episodes* **do**
  $\mathbf{b}^\star \leftarrow \mathrm{CEM}(F_\theta, R_{\omega^\star})$;
  Execute the first action block $b_0^\star$;
  Add all new dense windows to $\mathcal{Q}$;
  if *a scheduled full minibatch is available* **then**
    Sample $\mathcal{B}$ uniformly from $\mathcal{Q}$;
    Update θ using $\mathcal{L}_{\mathrm{TTT}}(\theta, \mathcal{B})$;
  end
end

model assigns the higher predicted return to a plan that performs worse in the environment, while the adapted predictor recovers the correct ordering. These examples are qualitative illustrations. Sections 4.2 and 4.3 provide the quantitative planning and prediction evaluations.

3.3 JEPA-TTT: Persistent Self-Supervised Test-Time Training

During test-time interaction, the agent stores its observations and executed actions. Each training window is encoded by the frozen visual encoder to produce latent prediction targets. For window w, let $(z_{w,0}, \ldots, z_{w,C})$ denote the encoded latents and let $(b_{w,0}, \ldots, b_{w,C-1})$ denote the corresponding action blocks. The predictor then rolls forward from the preceding latent context, conditioned on the executed actions. At prediction position j, the model estimates $\widehat{z}_{w,j+1}(\theta)$ from $z_{w,0:j}$ and $b_{w,0:j}$. It is trained to match the frozen encoder representation $z_{w,j+1}$ of the observation. For a minibatch $\mathcal{B}$ and latent dimension d_z, we minimize:

$$\mathcal{L}_{\mathrm{TTT}}(\theta, \mathcal{B}) = \frac{1}{|\mathcal{B}| C d_z} \sum_{w \in \mathcal{B}} \sum_{j=0}^{C-1} \left\| \widehat{z}_{w,j+1}(\theta) - \mathrm{sg}(z_{w,j+1}) \right\|_2^2 . \tag{4}$$

Here, sg denotes the stop-gradient operator. Because E_ϕ is frozen, the target representation remains fixed and test-time training updates only θ. We perform at most one AdamW update at each replanning step when a complete minibatch is available. Freezing the encoder also prevents the target representation from drifting during online predictor optimization. The test-time loss itself is intentionally simple. Our focus is on whether persistent self-supervised prediction updates can adapt to dynamics shift, and on how online data should be organized to make those updates effective.

JEPA-TTT retains learned dynamics across test-time episodes. The predictor parameters, optimizer state, and online training buffer all persist across episode boundaries. Let $\theta_{e,t}$, $\nu_{e,t}$, and $\mathcal{Q}_{e,t}$ denote these states at interaction index t of episode e, where $t = 0$ is the state before the first interaction. If episode e ends with predictor state θ_{e,T_e}, optimizer state ν_{e,T_e}, and buffer $\mathcal{Q}_{e,T_e}$, episode $e+1$ begins from exactly these states: $(\theta_{e+1,0}, \nu_{e+1,0}, \mathcal{Q}_{e+1,0}) = (\theta_{e,T_e}, \nu_{e,T_e}, \mathcal{Q}_{e,T_e})$. Algorithm 1 summarizes JEPA-TTT and Section 3.4 defines its dense replay update rule.

3.4 Dense Replay

Experience replay reuses online interaction data (Lin, 1992). Reactor extends prioritized experience replay to overlapping sequences (Gruslys et al., 2017). JEPA-TTT uses dense replay to train from accumulated online experience. It forms prediction windows at every valid temporal offset, retains them in a growing buffer, and uniformly samples minibatches for predictor updates.

Dense replay starts a new window at every environment step. Let $a_{u:v} = (a_u, \ldots, a_v)$. For an episode trajectory, the window beginning at environment step t contains $C + 1$ observations and the C action blocks between them: $\mathcal{W}_t = \left((o_{t+jK})_{j=0}^{C}, \, (a_{t+jK:t+(j+1)K-1})_{j=0}^{C-1} \right)$. Each $\mathcal{W}_t$ is one training window, and $\mathcal{Q}$ stores the valid windows accumulated across test-time episodes.

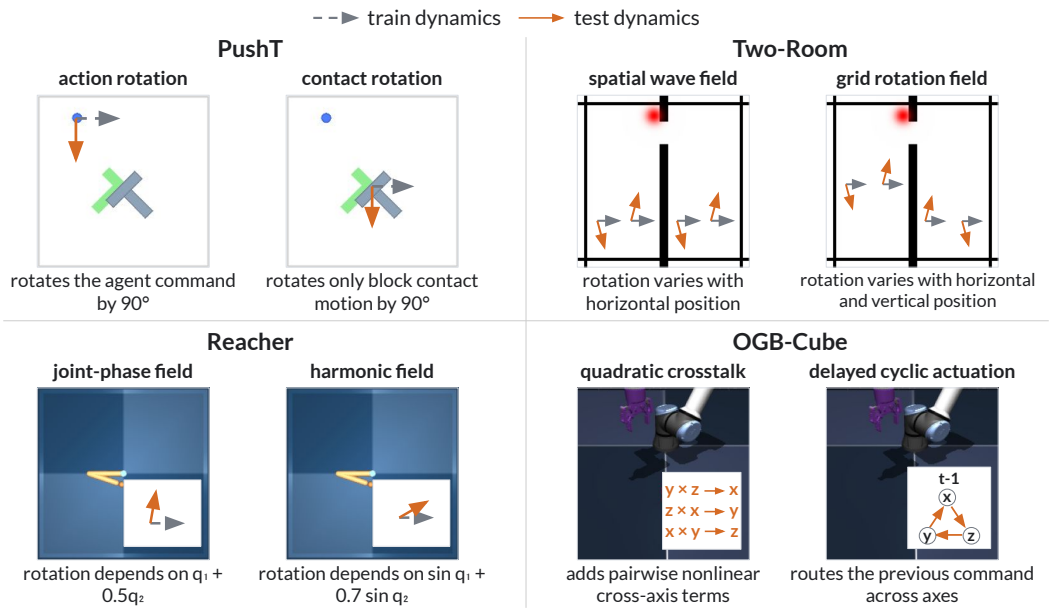


Figure 4: Train and test dynamics illustrated for the eight shifts in our evaluation suite.

Replay capacity and the ratio of learning updates can both affect performance (Fedus et al., 2020), motivating the following controlled comparisons. Dense construction begins a window at every environment step, whereas sparse construction begins only at action-block boundaries. Thus, dense construction includes all K possible offsets within each action block and yields approximately K times as many windows. The *sparse stream* processes sparse windows chronologically and uses each once. The *compute-matched sparse stream* repeats new sparse windows to match the dense presentation budget without expanding temporal coverage. The *dense stream* processes all dense windows chronologically and uses each once. Dense replay instead samples uniformly from all dense windows accumulated in the buffer.

The comparisons isolate the three factors in sequence: sparse versus compute-matched sparse tests additional optimization; compute-matched sparse versus dense stream tests temporal coverage; and dense stream versus dense replay tests replay from the same available dense window set under a matched cumulative update budget. Appendix A.1.2 specifies the detailed window validity and minibatch scheduling.

4 Experiments

Our experiments address three research questions: **RQ1: Planning recovery.** Does JEPA-TTT recover planning performance under dynamics shifts? **RQ2: Predictive accuracy.** Does the adapted predictor become more accurate on state prediction? **RQ3: Ablation studies.** How do temporal coverage, replay, and persistence affect planning performance?

4.1 Experimental Setup

Tasks and shifts. We evaluate four diverse continuous-control environments: PushT (Florence et al., 2022; Chi et al., 2025), Two-Room (Sobal et al., 2025b), Reacher from the DeepMind Control Suite (Tassa et al., 2018), and OGBench-Cube (Park et al., 2025). All four were also used in LeWorldModel (Maes et al., 2026). Each environment has a fixed task objective with a score in $[0, 1]$. For every task, we design two test-time dynamics shifts while keeping the observation mapping, task score, action space, and episode horizon unchanged. The resulting eight shifts cover changes in actuator mapping, contact response, state-dependent control transformations, cross-axis coupling, and delayed actuation. Figure 4 illustrates the shifts, and Appendix A.2 gives their detailed definitions.

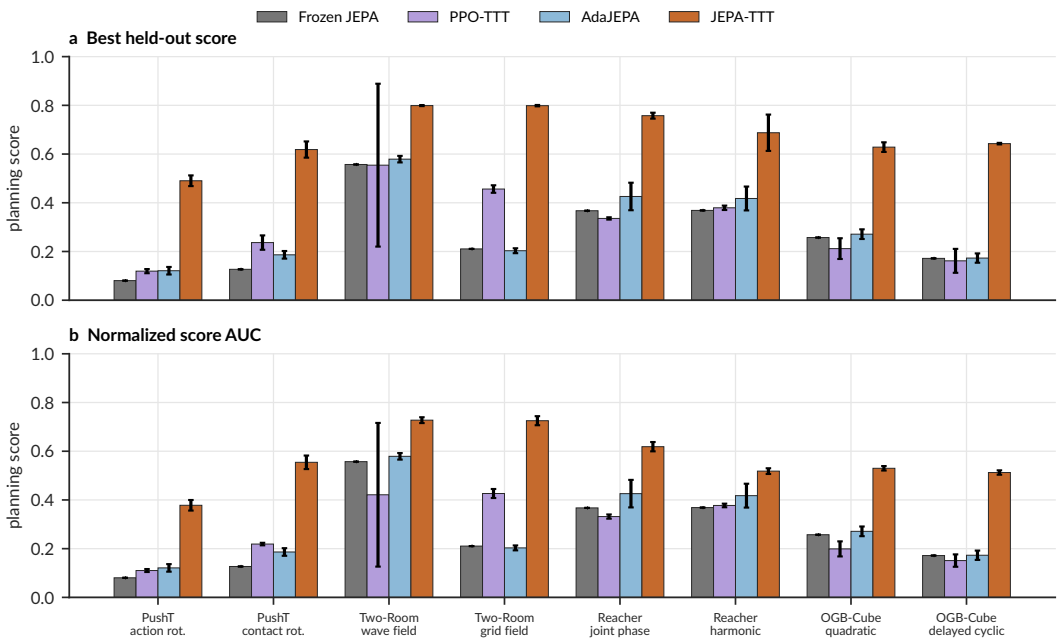


Figure 5: Planning performance under the test-time dynamics shift. Panel (a) reports the best held-out score. Panel (b) reports normalized held-out planning-score AUC from episode 0 to 500. Bars and error bars report means and standard deviations over three deployment runs. JEPA-TTT outperforms other methods including the frozen JEPA world model baseline, PPO test-time training (PPO-TTT), and AdaJEPA.

Offline training. For each task, we collect 3,000 complete episodes in the training environment and pretrain one action-conditioned JEPA world model (i.e., LeWorldModel). We then freeze its encoder and fit a reward head from offline score labels collected under the same dynamics. Appendix A.1.1 provides the data collection, architecture, and optimization details.

Test-time protocol. Each JEPA-TTT deployment consists of 500 episodes under one fixed set of test-time dynamics. At episodes $0, 50, \ldots, 500$, we evaluate the current model on the same fixed set of 100 held-out episodes. We report means and standard deviations over three deployment runs with different environment seeds. Appendix A.1.2 provides the complete test-time training and planning settings.

Metrics. We report the best held-out planning score (best score), defined as the maximum three-run mean across evaluations at episodes $50, 100, \ldots, 500$. We also report the normalized area under the held-out planning-score curve, averaged over three runs (mean AUC). For each run, we compute the area from episode 0 to 500 by trapezoidal integration and divide by 500. Aggregate results average both metrics across shifts. The best score measures the strongest planning performance reached, while mean AUC summarizes performance throughout test-time training. For dynamics prediction, we report five-block autoregressive latent MSE on fixed trajectories collected under the test-time dynamics. Appendix A.5 details trajectory collection and error aggregation.

Baselines. Frozen JEPA uses the same pretrained LeWorldModel, reward head, and CEM planner as JEPA-TTT, but does not update the world model at test time. PPO-TTT tests whether a pretrained PPO policy can adapt from oracle environment rewards in the same 500-episode interactions (Schulman et al., 2017). AdaJEPA is the concurrent work that also adapts a JEPA world model from online transitions within each episode (Wang et al., 2026a). For a controlled comparison, we adapt AdaJEPA to our goal-image-free setting, initialize it from the same pretrained LeWorldModel, and use the same frozen reward head and CEM planner as JEPA-TTT.

4.2 Test-Time Training Improves Planning under Dynamics Shifts

For RQ1, JEPA-TTT improves planning across all eight dynamics shifts. Relative to the frozen JEPA world model, JEPA-TTT improves best score by 153% and mean AUC by 113%, with both metrics averaged across shifts. JEPA-TTT outperforms all three baselines on every evaluated shift for both reported planning metrics. Figure 5 summarizes these results, and Appendix Table 3 reports the exact values.

The large PPO-TTT variance on the Two-Room spatial-wave shift reflects qualitatively different learning outcomes across deployment runs. One run remains trapped in the first room, one discovers the doorway after roughly 350 test episodes, and one discovers it within the first 50 episodes.

The full learning curves in Figure 7 show that JEPA-TTT improves as interaction accumulates. Its best score and mean AUC improve over the frozen JEPA world model on all eight shifts. We include a data-rich offline reference trained directly under the test-time dynamics to contextualize the remaining adaptation gap. It is not a deployable baseline and is not a mathematical upper bound because it uses a different offline data collection and uses 3,000 complete episodes from the shifted environment for 20 epochs. On six of the eight shifts, the best score of JEPA-TTT is very close to, or even slightly surpasses, the offline reference using only 500 episodes of online interaction. Appendix Table 9 reports the reference values.

4.3 Test-Time Training Improves Dynamics Prediction

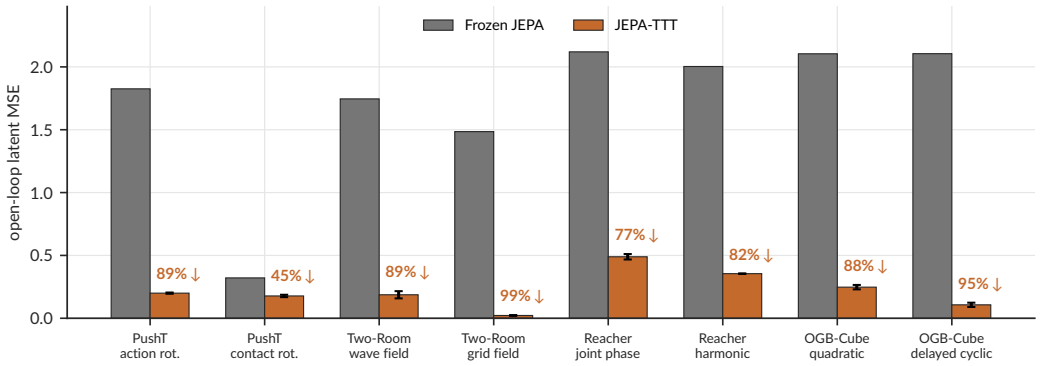


Figure 6: Five-block autoregressive latent prediction error under the test-time dynamics. The orange bars show JEPA-TTT's prediction errors drop significantly after 500 test-time episodes. Percent labels show the relative MSE reduction after adaptation, and error bars show standard deviations over the three adapted models.

For RQ2, we evaluate the frozen JEPA predictor and the adapted predictor on identical held-out trajectories. Both models receive the same observed context and future action sequence, then predict five latent states autoregressively without intermediate observations. JEPA-TTT lowers latent prediction MSE on every shift. The mean reduction is 83% after 500 test-time episodes. These reductions indicate that the predictor has become more accurate under the changed dynamics. Figure 6 reports the comparison.

The planning and prediction results are therefore consistent with the intended mechanism: JEPA-TTT improves dynamics prediction under the shifted dynamics and supports better planning.

4.4 Ablation Studies

For RQ3, we ablate dense replay by separately varying update count, temporal coverage, and replay. Averaged across shifts, the compute-matched sparse stream improves both best score and mean AUC over the sparse stream, indicating a benefit from additional optimization. The dense stream further improves both metrics under the same presentation budget, suggesting a benefit from broader temporal coverage. Dense replay achieves the highest aggregate best score and mean AUC among the four update rules. Compared with the dense stream, it reaches a similar best score (0.678 versus

Shift	Metric	Sparse stream	Compute-matched sparse stream	Dense stream	Dense replay
PushT: action rotation	Best	0.253 ± 0.063	0.236 ± 0.049	$\mathbf{0.511 \pm 0.009}$	0.490 ± 0.022
	AUC	0.169 ± 0.007	0.169 ± 0.021	0.359 ± 0.044	$\mathbf{0.378 \pm 0.022}$
PushT: contact rotation	Best	0.317 ± 0.258	0.521 ± 0.319	$\mathbf{0.646 \pm 0.044}$	0.619 ± 0.033
	AUC	0.205 ± 0.082	0.320 ± 0.157	0.490 ± 0.041	$\mathbf{0.555 \pm 0.028}$
Two-Room: spatial wave	Best	0.505 ± 0.031	0.790 ± 0.002	0.797 ± 0.002	$\mathbf{0.799 \pm 0.002}$
	AUC	0.427 ± 0.034	0.672 ± 0.020	0.697 ± 0.012	$\mathbf{0.728 \pm 0.012}$
Two-Room: grid rotation	Best	0.786 ± 0.004	0.794 ± 0.003	0.782 ± 0.000	$\mathbf{0.799 \pm 0.003}$
	AUC	0.662 ± 0.009	0.717 ± 0.026	0.686 ± 0.020	$\mathbf{0.725 \pm 0.019}$
Reacher: joint phase	Best	0.557 ± 0.025	$\mathbf{0.768 \pm 0.062}$	0.725 ± 0.018	0.758 ± 0.012
	AUC	0.401 ± 0.007	0.545 ± 0.051	0.593 ± 0.013	$\mathbf{0.619 \pm 0.019}$
Reacher: harmonic	Best	0.446 ± 0.046	0.588 ± 0.048	0.610 ± 0.039	$\mathbf{0.688 \pm 0.075}$
	AUC	0.398 ± 0.007	0.488 ± 0.014	0.492 ± 0.006	$\mathbf{0.518 \pm 0.012}$
OGB-Cube: quadratic	Best	0.520 ± 0.053	0.652 ± 0.010	$\mathbf{0.653 \pm 0.004}$	0.628 ± 0.020
	AUC	0.390 ± 0.005	0.528 ± 0.015	0.528 ± 0.013	$\mathbf{0.530 \pm 0.009}$
OGB-Cube: delayed cyclic	Best	0.591 ± 0.044	0.646 ± 0.006	$\mathbf{0.646 \pm 0.004}$	0.643 ± 0.003
	AUC	0.290 ± 0.022	0.522 ± 0.006	$\mathbf{0.542 \pm 0.006}$	0.513 ± 0.009
Mean	Best	0.497 ± 0.028	0.625 ± 0.025	0.671 ± 0.008	$\mathbf{0.678 \pm 0.013}$
	AUC	0.368 ± 0.007	0.495 ± 0.025	0.548 ± 0.007	$\mathbf{0.571 \pm 0.004}$

Table 1: Comparisons of different update rules for test-time training. The sparse stream uses its natural number of updates. Entries report means and standard deviations over three runs. The Mean rows summarize the three run-level averages across shifts. Bold marks the highest unrounded mean for each metric and shift. Dense replay has the highest aggregate mean best score and AUC.

0.671) while improving mean AUC (0.571 versus 0.548). Table 1 reports the full comparison. These results support dense temporal coverage and suggest a small additional benefit from replay.

We also test the effect of retaining adaptation across episodes. In a matched control with episode-local dense windows, retaining the predictor and optimizer state increases the best score from 0.289 to 0.729 and mean AUC from 0.273 to 0.637, with the same number of predictor updates. Appendix Table 6 reports the full results.

5 Conclusion

We studied whether a pretrained action-conditioned JEPA world model can adapt its dynamics model during deployment. JEPA-TTT updates only the latent dynamics predictor from observed transitions while keeping the visual encoder and reward head fixed. The adapted predictor is retained across episodes, allowing adaptation to accumulate over time. Across eight dynamics shifts in four continuous-control environments, JEPA-TTT improves planning on every shift. The planning gains are accompanied by lower latent prediction error. Ablation studies show the benefit of dense temporal windows, with a small further gain from replay.

Scope of dynamics shifts. We study a single dynamics change that remains fixed throughout test time. The visual observation mapping, task score, action interface, and episode horizon do not change. Because the encoder and reward head remain frozen, JEPA-TTT is not designed to handle simultaneous visual or reward shifts.

Continual adaptation and catastrophic forgetting. We do not study sequences of dynamics changes or a return to the training dynamics after adaptation. Persistent predictor updates may degrade predictions for previously learned dynamics, including the training dynamics. Our experiments therefore do not measure under continual learning settings.

AI USE STATEMENT

In this work, we used generative AI tools for searching related work, implementation, and polishing writing. We have not used generative AI tools for ideation or for developing the conceptual framework, and formulating mathematical claims and writing proofs are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures and summarizing existing literature. We have reviewed all AI-assisted work. We have carefully checked the LLM-generated code for correctness and polishing changes. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our experiments study test-time adaptation in simulated control environments. Deployment in physical systems would require additional safety evaluation and safeguards, as prediction errors and online model updates could lead to unsafe actions.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 describe our method. Appendix A.1 documents data collection, model architectures, and training and evaluation settings. Appendices A.2–A.3 specify the dynamics shifts and baselines, while Appendix A.5 details the prediction-error evaluation.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

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

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801. IEEE, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, Leslie Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning. In *International Conference on Learning Representations*, volume 2024, pp. 31138–31155, 2024.

Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International conference on machine learning*, pp. 3061–3071. PMLR, 2020.

Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world. *arXiv preprint arXiv:2310.16029*, 2023.

Yusen Feng, Bingchen Han, Jiangran Lyu, Kai Liu, Yixin Zheng, Yuxuan Wan, Weiheng Liu, Sun Han, Ruiqin Li, Yulong Zhang, et al. Wam-ttt: Steering world-action models by watching human play at test time. *arXiv preprint arXiv:2607.06988*, 2026.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022.

Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc Bellemare, and Remi Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. *arXiv preprint arXiv:1704.04651*, 2017.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

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

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.

Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54625–54640, 2024.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022. URL `https://openreview.net/pdf?id=BZ5a1r-kVsf`.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

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

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *Advances in Neural Information Processing Systems*, 38:43905–43941, 2025a.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Stress-testing offline reward-free reinforcement learning: A case for planning with latent dynamics models. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025b.

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.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

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.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022.

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.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. Adajepa: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026a.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim GJ Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026b.

Grady R Williams, Brian Goldfain, Keuntaek Lee, Jason Gibson, James M Rehg, and Evangelos A Theodorou. Locally weighted regression pseudo-rehearsal for adaptive model predictive control. In *Conference on Robot Learning*, pp. 969–978. PMLR, 2020.

Yifan Yin, Zehao Wen, Suyu Ye, Jieneng Chen, Zehan Zheng, Nanru Dai, Haojun Shi, Aydan Huang, Zheyuan Zhang, Alan Yuille, et al. 3d-belief: Embodied belief inference via generative 3d world modeling. *arXiv preprint arXiv:2605.11367*, 2026.

Hongxin Zhang, Zeyuan Wang, Qiushi Lyu, Zheyuan Zhang, Sunli Chen, Tianmin Shu, Behzad Dariush, Kwonjoon Lee, Yilun Du, and Chuang Gan. Combo: Compositional world models for embodied multi-agent cooperation. In *International Conference on Learning Representations*, volume 2025, pp. 49996–50019, 2025.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal Patel, Paul Liang, et al. World-in-world: World models in a closed-loop world. In *International Conference on Learning Representations*, volume 2026, pp. 55660–55699, 2026.

Haoyu Zhen, Qiao Sun, Hongxin Zhang, Junyan Li, Siyuan Zhou, Yilun Du, and Chuang Gan. Tesseract: learning 4d embodied world models. *arXiv preprint arXiv:2504.20995*, 2025.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A Appendix

A.1 Experimental Protocol and Implementation

A.1.1 World-Model Pretraining

All observations are 224×224 RGB images. All four world models use the same LeWorldModel configuration. The image encoder is a ViT-Tiny with 14×14 patches and 192-dimensional outputs. The autoregressive predictor has six transformer layers, 16 attention heads of dimension 64, hidden dimension 192, MLP dimension 2048, and dropout 0.1. The action encoder maps one five-action block to 192 dimensions. The latent and prediction projections are two-layer MLPs with hidden dimension 2048 and BatchNorm. Each sample contains three context keyframes and one prediction keyframe separated by five raw simulator steps.

We collect 3,000 complete episodes under the training dynamics for each task. PushT uses an equal mixture of block-local and arena-wide target actions held for four steps. Two-Room uses the upstream noisy heuristic, Reacher uses random actions, and OGB-Cube uses an equal mixture of guided and Gaussian exploration, with 30% uniform actions during guided episodes. Maximum collection horizons are 300, 100, 200, and 50 steps, respectively. These collection distributions do not use task rewards and need not equal the fixed-task evaluation reset distribution. Models are initialized from scratch and trained for 20 epochs using AdamW, batch size 128, maximum learning rate 5×10^{-5}, weight decay 10^{-3}, gradient clipping 1.0, linear warmup followed by cosine decay, and bfloat16 precision. We use a random 90/10 window split and the predeclared epoch-20 checkpoint rather than selecting an epoch on test performance. The upstream LeWorldModel latent prediction and SIGReg objectives are unchanged. SIGReg uses weight 0.09, 17 knots, and 1,024 projections.

After pretraining, the reward head is fitted on an episode-disjoint split collected under the training dynamics. It is a sigmoid MLP with dimensions $192 \to 256 \to 256 \to 1$ and ReLU hidden activations. Training uses AdamW, batch size 512, learning rate 10^{-3}, weight decay 10^{-5}, and early stopping with patience 15 for at most 100 epochs. The selected epochs are 67, 64, 82, and 61 for PushT, Two-Room, Reacher, and OGBench-Cube. The optional decoder is trained for 20 epochs with AdamW, batch size 32, learning rate 2×10^{-4}, weight decay 10^{-4}, gradient clipping 1.0, and bfloat16 autocast. Neither the decoder nor its output is used for planning.

A.1.2 Planning and Test-Time Training

For dense construction, a window start becomes valid only after a clip spanning $(C + 1)K$ raw observation rows from that start is available. The loss uses the keyframes and first C action blocks in $\mathcal{W}_t$. Sparse construction creates its first window after C complete action blocks. For the compute-matched sparse stream, each new sparse window is repeated $N_\tau - N_{\tau-1}$ times, where N_τ is the number of dense windows available after replanning step τ.

Test-time interaction continues to the fixed episode horizon after success so that data supply does not depend on the current model. Held-out evaluation episodes use normal task termination. The sparse stream, compute-matched sparse stream, and dense stream update only when a full batch is available. Windows that do not complete a batch remain pending at scheduled evaluations and at episode 500 rather than being flushed as a smaller batch. Dense replay receives the same number of updates as the dense stream. After N unique dense windows, both have $\lfloor N/64 \rfloor$ updates. Each replay batch samples without replacement within that batch, while all stored windows remain valid for later updates.

For JEPA-TTT and each dense replay ablation, all three deployment runs begin from the same fixed pretrained world model. The runs differ only in their test-time environment-seed sequences and therefore in the interaction data accumulated during adaptation. Held-out evaluation uses a separate fixed seed suite that is disjoint from all deployment runs, and evaluation trajectories never enter the online buffer. The reported standard deviations therefore characterize variability in test-time interaction and adaptation conditional on one pretrained model. They do not measure variability across independently pretrained world models.

Component	Setting
Test-time interaction horizon	500 episodes under the test-time dynamics
Episode horizon	PushT: 300 raw steps, other tasks: 50
Held-out evaluations	episodes $0, 50, \ldots, 500$
Evaluation size	100 held-out episodes per evaluation
CEM	256 candidates, 6 iterations, 32 elites
Planning horizon	5 world-model blocks
Execution	1 block, then replan
Action block	5 raw actions
CEM proposal	zero-centered Gaussian, initial std. 1.0, std. floor 0.05
Reward discount	0.99
Reward-head scale	PushT: 10, other tasks: 1 (rank-invariant for CEM)
Test-time optimizer	AdamW, LR 5×10^{-5}, WD 10^{-3}
Test-time batch size	64
Gradient clipping	1.0
Trainable-module mode	evaluation mode; normalization statistics remain fixed
Update rate	at most 1 update per replan
Trainable components	action encoder, predictor, prediction projection
Frozen components	image encoder, latent projector, reward head, decoder
Test-time inputs	current RGB observations and executed actions

Table 2: Shared test-time planning and training settings. These settings define the common test-time protocol for all eight shifts.

A.2 Tasks and Dynamics Shifts

Let

$$\mathbf{R}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{5}$$

denote a planar rotation. Every shift below changes only how the commanded action produces the next state. The visual goal, score, action bounds, and initial-state distribution are identical to the corresponding task under the training dynamics.

A.2.1 PushT

The action is a normalized two-dimensional target for the circular agent. The instantaneous score is the overlap coverage between the movable T block and the fixed green target, and the reported episode score is its peak value. Under *action rotation*, a target u_t is rotated by 90° around the current agent position p_t before the base controller receives it:

$$u_t^{\text{apply}} = p_t + \mathbf{R}(\pi/2)(u_t - p_t). \tag{6}$$

Under *contact rotation*, the agent follows the commanded target normally. At each physics substep, only the contact-induced block displacement Δq is replaced by $\mathbf{R}(\pi/2)\Delta q$. This separates an actuator-frame shift from a contact-response shift.

A.2.2 Two-Room

The agent begins in the left room and moves toward a fixed goal $g = (164, 112)$ in pixel coordinates. Let $d = (112, 49)$ be the doorway waypoint. The score is $\exp(-D(s, g)/64)$, where $D(s, g) = \|s - d\|_2 + \|d - g\|_2$ while the agent is left of the wall and $D(s, g) = \|s - g\|_2$ after it crosses. A run succeeds within 16 pixels of g.

Both shifts apply $a_t^{\text{apply}} = \mathbf{R}(\alpha(s_t))a_t$. For the spatial wave field,

$$\alpha(s) = 75^\circ \sin\left(\frac{2\pi(x - 20)}{88}\right). \tag{7}$$

For the grid field,

$$\alpha(s) = 75^\circ \sin\left(\frac{2\pi(x-20)}{96}\right)\cos\left(\frac{2\pi(y-20)}{96}\right). \tag{8}$$

The first field changes smoothly with horizontal position; the second changes over both visible room coordinates.

A.2.3 REACHER

The goal is the fixed joint configuration $(0, 0)$. Let $\widetilde{q} \in [-\pi, \pi)^2$ be the wrapped joint-angle error. The score is $\exp(-\operatorname{mean}(\widetilde{q}^2)/2)$, and success requires $|\widetilde{q}_i| < 0.05$ for both joints. The shifts rotate the two-dimensional motor command by an angle that depends on the current configuration and vanishes near the goal.

Define

$$h(\widetilde{q}) = 1 - \exp\left(-\frac{\operatorname{mean}(\widetilde{q}^2)}{2(0.20)^2}\right), \qquad \alpha(\widetilde{q}) = \alpha_{\max} h(\widetilde{q}) \tanh\left(\frac{\psi(\widetilde{q})}{\kappa}\right). \tag{9}$$

The joint-phase field uses $\psi(\widetilde{q}) = \widetilde{q}_1 + 0.5\widetilde{q}_2$, $\kappa = 0.35$, and $\alpha_{\max} = 155^\circ$. The harmonic field uses $\psi(\widetilde{q}) = \sin\widetilde{q}_1 + 0.7\sin\widetilde{q}_2$, $\kappa = 0.30$, and $\alpha_{\max} = 165^\circ$. In both cases, $a_t^{\text{apply}} = \operatorname{clip}(\mathbf{R}(\alpha(\widetilde{q}_t))a_t, -1, 1)$.

A.2.4 OGBENCH-CUBE

We retain the OGBench-Cube simulator, UR5e robot, camera, table, and physical red cube, and use a three-dimensional reach task. The target is a fixed hover point 0.10 m above the cube at $(0.425, 0, 0.12)$ m. The action contains Cartesian x, y, z commands; yaw and gripper commands are fixed at zero. For end-effector error e, the score is $\exp(-\|e\|_2/0.10)$ and success requires $\|e\|_2 \leq 0.05$ m.

For quadratic crosstalk, a command $a = (a_x, a_y, a_z)$ becomes

$$a^{\text{apply}} = \operatorname{clip}(a + 3(a_y a_z, a_z a_x, a_x a_y), -1, 1). \tag{10}$$

For delayed cyclic actuation, the command applied at raw step t is

$$a_t^{\text{apply}} = (a_{t-1,y}, a_{t-1,z}, a_{t-1,x}), \tag{11}$$

with the previous command initialized to zero at reset. This combines a one-step delay with a fixed cross-axis routing.

A.3 BASELINE DETAILS

AdaJEPA. We evaluate AdaJEPA's episodic adaptation procedure using the same pretrained model, reward head, and CEM planner in JEPA-TTT without a goal image. We do not use its original goal-conditioned planning formulation to ensure fairness. Each episode restores the pretrained parameters and clears the history of the five most recent executed action segments. Every replanning step initializes a new Adam optimizer and performs one update. All adaptation is discarded at the end of the episode. The predictor learning rate is 5×10^{-4} and the encoder-head rate, when adapted, is 10^{-5}, without weight decay or gradient clipping. We evaluate adaptation on the last layer of the predictor, either with a frozen encoder head (pred-last+enc-frozen) or with the default trainable projection head (pred-last+enc-last). We evaluate AdaJEPA in three runs of 100 episodes. Before every episode, AdaJEPA reloads the pretrained model, adapts only from transitions observed during that episode, and produces one task score. We average the 100 episode scores within each run, then report the mean and sample standard deviation across the three run averages. Because no adaptation carries between episodes, we use this per-episode performance estimate for both best score and mean AUC.

PPO-TTT. Base PPO policies use a CNN encoder with 32, 64, and 64 channels, followed by a 256-dimensional feature layer and diagonal-Gaussian actor and value heads. PushT requested 10 million training steps, stopped after 2.40 million by a predeclared validation-plateau rule, and selected the checkpoint at 1.30 million steps. Two-Room, Reacher, and OGB-Cube each trained for

2 million steps and selected checkpoints at 1.5, 1.2, and 1.7 million steps, respectively, using fixed held-out validation episodes. The selected policy, value function, and a fresh optimizer then persist across 500 test episodes and receive the online environment reward. Every 10 episodes, Adam runs 10 epochs with learning rate 3×10^{-4}, minibatch size 512, discount 0.99, GAE parameter 0.95, clipping coefficient 0.2, value coefficient 0.5, entropy coefficient 0, and gradient clipping 0.5. Test-time actions are stochastic. Held-out evaluation actions are deterministic.

A.4 Complete Planning Results

Shift	Metric	Frozen JEPA	PPO-TTT	AdaJEPA	JEPA-TTT
PushT: action rotation	Best	0.080	0.119 ± 0.009	0.121 ± 0.015	$\mathbf{0.490 \pm 0.022}$
	AUC	0.080	0.110 ± 0.006	0.121 ± 0.015	$\mathbf{0.378 \pm 0.022}$
PushT: contact rotation	Best	0.127	0.237 ± 0.029	0.186 ± 0.016	$\mathbf{0.619 \pm 0.033}$
	AUC	0.127	0.219 ± 0.005	0.186 ± 0.016	$\mathbf{0.555 \pm 0.028}$
Two-Room: spatial wave	Best	0.557	0.554 ± 0.334	0.579 ± 0.013	$\mathbf{0.799 \pm 0.002}$
	AUC	0.557	0.421 ± 0.295	0.579 ± 0.013	$\mathbf{0.728 \pm 0.012}$
Two-Room: grid rotation	Best	0.211	0.456 ± 0.015	0.203 ± 0.010	$\mathbf{0.799 \pm 0.003}$
	AUC	0.211	0.426 ± 0.019	0.203 ± 0.010	$\mathbf{0.725 \pm 0.019}$
Reacher: joint phase	Best	0.367	0.336 ± 0.005	0.426 ± 0.056	$\mathbf{0.758 \pm 0.012}$
	AUC	0.367	0.332 ± 0.009	0.426 ± 0.056	$\mathbf{0.619 \pm 0.019}$
Reacher: harmonic	Best	0.369	0.379 ± 0.009	0.418 ± 0.049	$\mathbf{0.688 \pm 0.075}$
	AUC	0.369	0.377 ± 0.007	0.418 ± 0.049	$\mathbf{0.518 \pm 0.012}$
OGB-Cube: quadratic	Best	0.257	0.212 ± 0.043	0.271 ± 0.020	$\mathbf{0.628 \pm 0.020}$
	AUC	0.257	0.199 ± 0.031	0.271 ± 0.020	$\mathbf{0.530 \pm 0.009}$
OGB-Cube: delayed cyclic	Best	0.172	0.162 ± 0.049	0.173 ± 0.019	$\mathbf{0.643 \pm 0.003}$
	AUC	0.172	0.151 ± 0.025	0.173 ± 0.019	$\mathbf{0.513 \pm 0.009}$
Mean	Best	0.267	0.307	0.297	**0.678**
	AUC	0.267	0.279	0.297	**0.571**

Table 3: Exact values for Figure 5. JEPA-TTT and PPO-TTT's entries are means and sample standard deviations over three deployment runs. For AdaJEPA, each run averages the task scores from 100 episodes, with the pretrained model reloaded before every episode. Entries report the mean and sample standard deviation across three runs. Because AdaJEPA does not retain adaptation across episodes, the same score is reported for both metrics. JEPA-TTT outperforms all three baselines on every shift under both metrics.

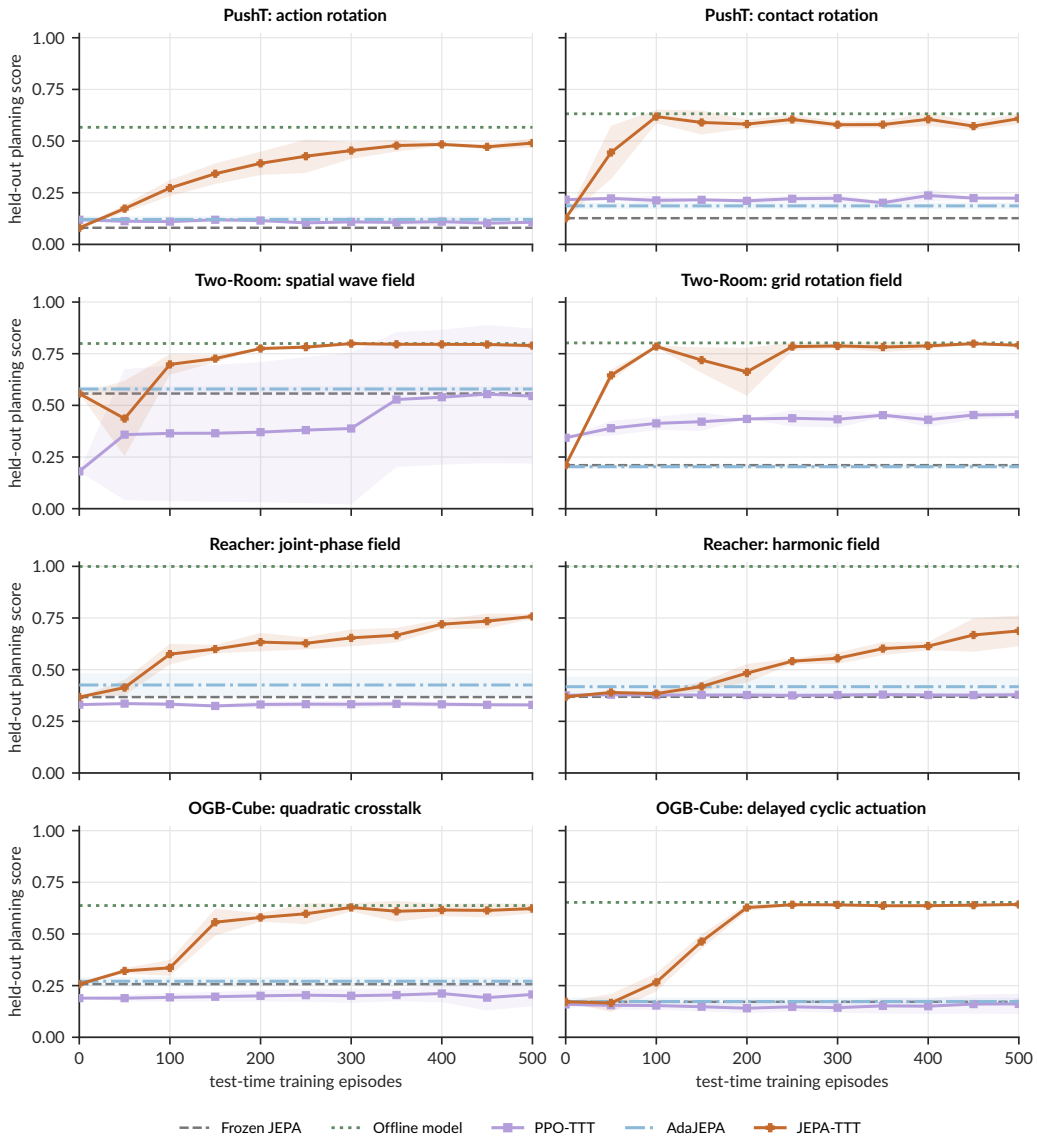


Figure 7: Held-out planning performance throughout 500 test-time training episodes. Curves show means over three deployment runs evaluated on the same fixed set of 100 held-out episodes, and shaded regions show standard deviations. JEPA-TTT is very close to, or even slightly surpasses, the offline reference using only 500 episodes of online interaction, and outperforms other methods including the frozen JEPA world model baseline, PPO test-time training (PPO-TTT), and AdaJEPA.

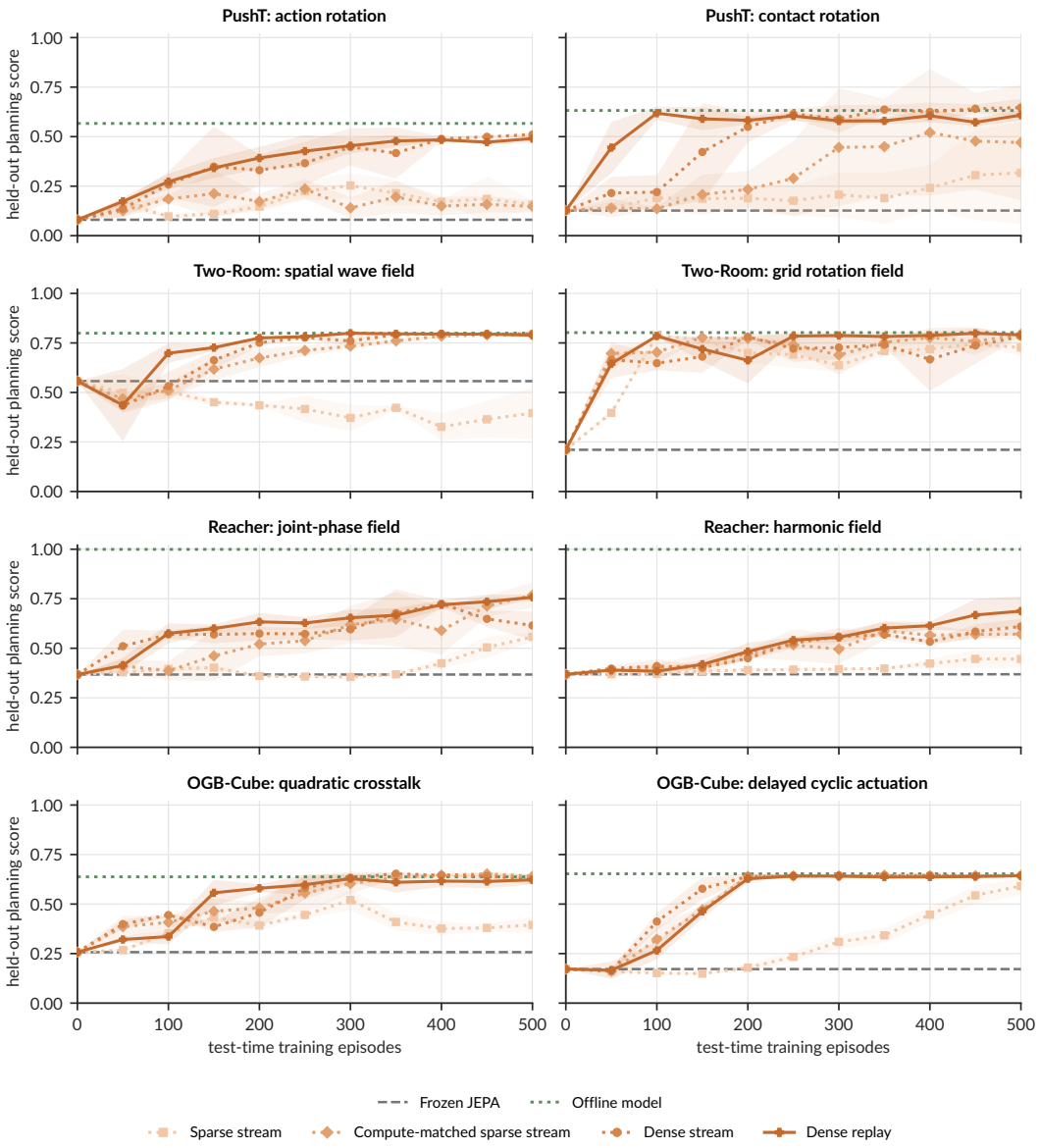


Figure 8: Full held-out score curves for JEPA-TTT and the three ablations of dense replay. Curves show means and sample standard deviations over three runs. Dashed gray lines show the frozen JEPA world model. Dotted green lines show the data-rich offline reference trained under the test-time dynamics. This reference is neither a deployable baseline nor a mathematical upper bound because it uses a different data-collection distribution and receives 3,000 complete offline episodes from the shifted environment (Table 9). Dense replay achieves the highest aggregate best score and AUC.

A.4.1 TRAINABLE COMPONENTS IN ADAJEPA

Shift	pred-last + enc-last (default)	pred-last + enc-frozen
PushT: action rotation	$\mathbf{0.1210 \pm 0.0155}$	0.1042 ± 0.0163
PushT: contact rotation	$\mathbf{0.1864 \pm 0.0156}$	0.1527 ± 0.0145
Two-Room: spatial wave	0.5791 ± 0.0134	$\mathbf{0.5850 \pm 0.0125}$
Two-Room: grid rotation	0.2031 ± 0.0101	$\mathbf{0.2470 \pm 0.0028}$
Reacher: joint phase	0.4259 ± 0.0565	$\mathbf{0.4266 \pm 0.0552}$
Reacher: harmonic	0.4176 ± 0.0490	$\mathbf{0.4183 \pm 0.0487}$
OGB-Cube: quadratic	0.2711 ± 0.0199	$\mathbf{0.2965 \pm 0.0144}$
OGB-Cube: delayed cyclic	$\mathbf{0.1730 \pm 0.0191}$	0.1608 ± 0.0204
Mean	0.2972	**0.2989**

Table 4: AdaJEPA performance with two choices of trainable components. Each run averages the task scores from 100 episodes, with the pretrained model reloaded before every episode. Entries report the mean and sample standard deviation across three runs. Because adaptation does not carry across episodes, the same performance estimate is used for both best held-out score and normalized held-out planning-score AUC. Bold marks the higher mean within each shift. The difference between the two configurations is only 0.0017. All main comparisons use the default pred-last+enc-last configuration.

A.5 PREDICTION AND DECODER DIAGNOSTICS

A.5.1 FIXED-TRAJECTORY LATENT ERROR

Within each row of Table 5, trajectories are collected once under the test-time dynamics and then reused unchanged for the frozen JEPA world model and the adapted model. They are neither expert demonstrations nor trajectories generated by an adapted model. PushT uses 100 fixed, seed-deterministic 300-step behavior trajectories. Every four simulator steps, this behavior policy samples a target with equal probability either uniformly over the arena or near the movable block with 60-pixel Gaussian jitter. Two-Room, Reacher, and OGB-Cube each use 100 fixed 50-step trajectories generated by CEM using the frozen JEPA world model and reward head. Observation and action hashes verify that every model receives exactly the same inputs.

We extract every sliding window containing three observed context frames and five future frames, with adjacent frames separated by five environment steps. The frozen image encoder maps all RGB observations to target latents. Starting from the three observed context latents, the predictor autoregressively predicts five future latents under the executed actions without receiving intermediate observations. At each horizon, MSE is averaged over the latent dimensions. Windows are averaged within each episode before averaging episodes, so long trajectories do not receive greater weight. Table 5 averages over the five horizons. Adapted values are means and sample standard deviations over the three dense replay models after episode 500. Raw magnitudes should not be compared across tasks because their latent spaces are trained independently.

Shift	Frozen JEPA MSE	Adapted MSE	Reduction
PushT action rotation	1.82540	0.19988 ± 0.00491	89.05%
PushT contact rotation	0.32117	0.17751 ± 0.00953	44.73%
Two-Room spatial wave	1.74541	0.18697 ± 0.02870	89.29%
Two-Room grid field	1.48501	0.02205 ± 0.00345	98.52%
Reacher joint phase	2.11969	0.48927 ± 0.02162	76.92%
Reacher harmonic	2.00316	0.35488 ± 0.00216	82.28%
OGB-Cube quadratic	2.10429	0.24754 ± 0.01766	88.24%
OGB-Cube delayed	2.10517	0.10688 ± 0.01710	94.92%

Table 5: Five-block autoregressive latent MSE before and after dense replay adaptation. Adaptation reduces MSE on all eight shifts, with reductions from 45% to 99%.

Matched MSE under the training dynamics is 0.10458 for PushT, 0.11622 for Two-Room, 1.27430 for Reacher, and 0.02131 for OGBench-Cube. These checks use the same task-specific behavior families as the diagnostics under the test-time dynamics.

A.5.2 Decoded Predictions

Figures 9 and 10 decode autoregressive latent predictions with the frozen pretrained decoder. Each panel uses the same initial observation and future action sequence for ground truth, the frozen JEPA world model, and the adapted model. The examples are qualitative diagnostics and are not used to compute planning scores or prediction MSE.

A.6 Ablations and Offline References

A.6.1 Matched Persistence Control

Both variants use dense windows from the current episode, batch size 8, the same prediction loss, identical optimizer hyperparameters, the same CEM planner, and the same number of predictor updates. A batch size of 8 is used because many individual episodes do not contain enough windows to form a batch of 64. The persistent variant carries the adapted predictor and optimizer state across episodes. The reset variant restores the pretrained predictor and reinitializes the optimizer at every episode boundary.

Shift	Metric	Persistent	Episodic reset	Difference
PushT action rotation	Best	0.4683 ± 0.0239	0.1049 ± 0.0168	$+0.3634$
	AUC	0.3863 ± 0.0247	0.0934 ± 0.0016	$+0.2929$
PushT contact rotation	Best	0.6393 ± 0.0292	0.1735 ± 0.0924	$+0.4658$
	AUC	0.5324 ± 0.0352	0.1254 ± 0.0127	$+0.4069$
Two-Room spatial wave	Best	0.7971 ± 0.0016	0.5972 ± 0.0135	$+0.1999$
	AUC	0.7692 ± 0.0041	0.5614 ± 0.0089	$+0.2078$
Two-Room grid field	Best	0.7892 ± 0.0157	0.2178 ± 0.0061	$+0.5714$
	AUC	0.7189 ± 0.0193	0.2127 ± 0.0043	$+0.5062$
Reacher joint phase	Best	0.9394 ± 0.0717	0.3931 ± 0.0275	$+0.5463$
	AUC	0.7329 ± 0.1311	0.3816 ± 0.0162	$+0.3513$
Reacher harmonic	Best	0.9153 ± 0.1150	0.3909 ± 0.0379	$+0.5244$
	AUC	0.7426 ± 0.1098	0.3810 ± 0.0211	$+0.3616$
OGB-Cube quadratic	Best	0.6385 ± 0.0085	0.2622 ± 0.0013	$+0.3763$
	AUC	0.5996 ± 0.0067	0.2569 ± 0.0017	$+0.3427$
OGB-Cube delayed	Best	0.6465 ± 0.0010	0.1723 ± 0.0064	$+0.4742$
	AUC	0.6119 ± 0.0042	0.1686 ± 0.0021	$+0.4433$
Mean	Best	0.7292	0.2890	$+0.4402$
	AUC	0.6367	0.2726	$+0.3641$

Table 6: Best held-out score and normalized held-out planning-score AUC for the matched persistence-versus-reset control. Entries are means and sample standard deviations over three runs. Persistence improves both metrics on every shift.

A.6.2 Continued Training under the Training Dynamics

We run dense replay for 500 episodes under the unchanged training dynamics as a no-shift control. Across the four tasks, the best score averages 0.679 for the frozen JEPA world model and 0.691 after continued training. Mean AUC is 0.679 for the frozen JEPA world model and 0.676 after continued training. These changes are small and inconsistent across tasks, so continued training under the training dynamics provides no consistent benefit. Table 7 reports both metrics for each task.

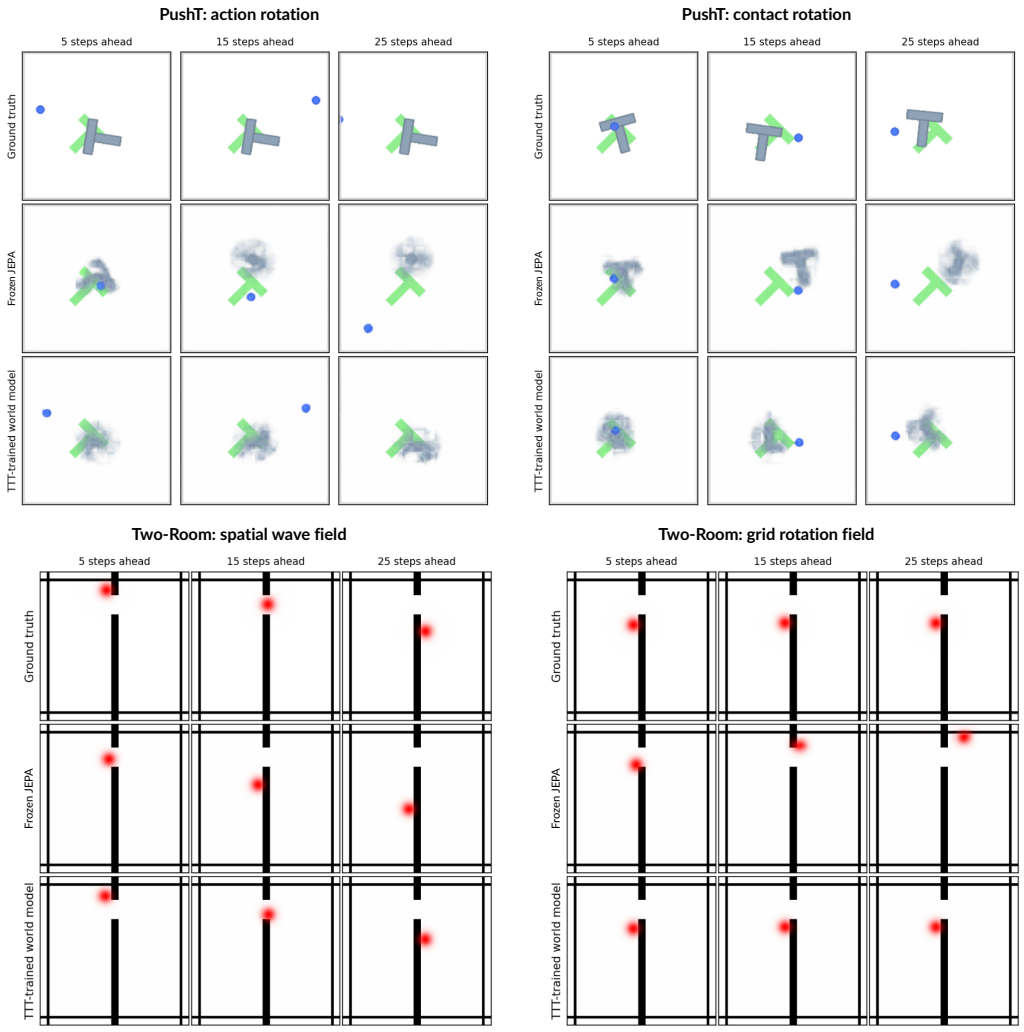


Figure 9: Decoded predictions for both PushT and Two-Room shifts. Rows within each subpanel are ground truth, Frozen JEPA prediction, and TTT-trained-world-model prediction; columns are 5, 15, and 25 raw steps ahead. After adaptation, decoded predictions more closely follow the ground truth across prediction horizons.

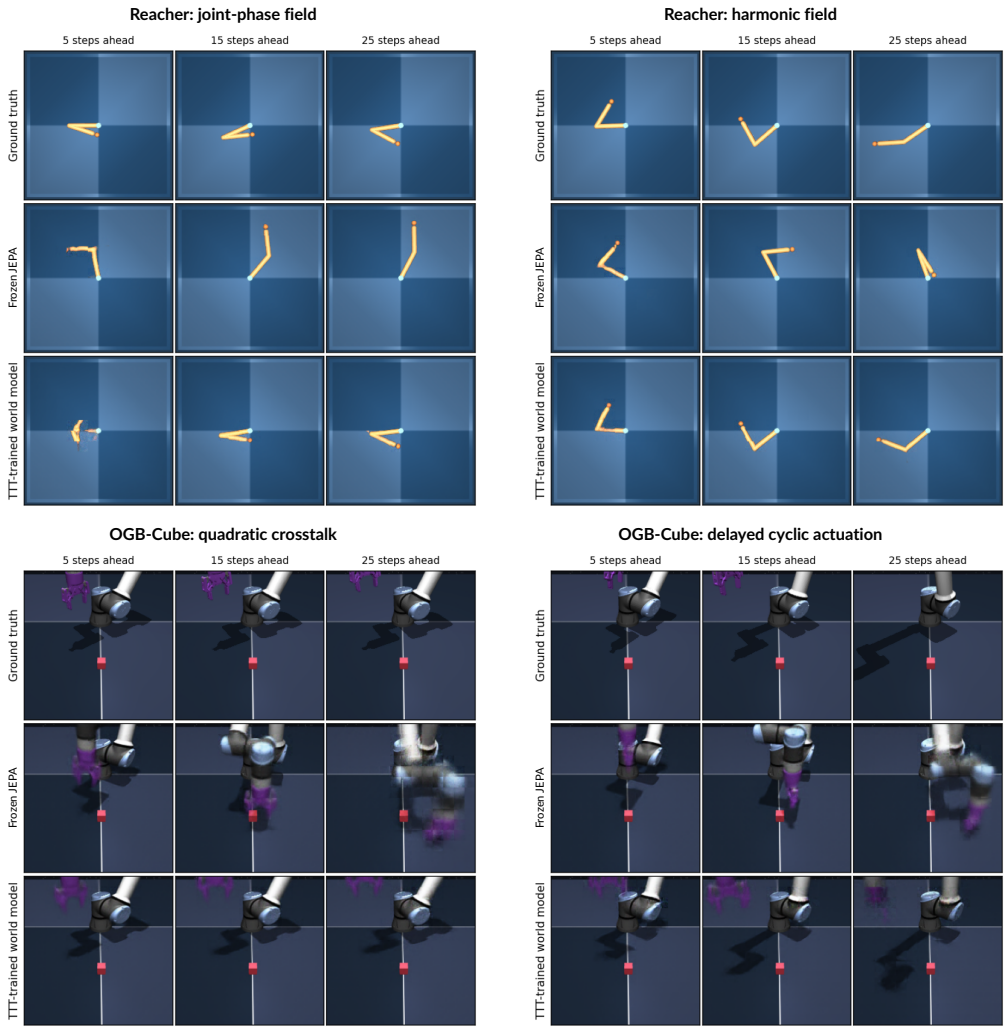


Figure 10: Decoded predictions for both Reacher and OGB-Cube shifts, using the same layout as Figure 9. After adaptation, decoded predictions more closely follow the ground truth across prediction horizons.

Task	Frozen JEPA best/AUC	Continued training best	Continued training AUC
PushT	0.2682	0.3599 ± 0.0187	0.3247 ± 0.0264
Two-Room	0.8039	0.8013 ± 0.0019	0.7894 ± 0.0084
Reacher	0.9995	0.9520 ± 0.0065	0.9438 ± 0.0042
OGB-Cube	0.6439	0.6492 ± 0.0010	0.6465 ± 0.0003

Table 7: Best held-out score and normalized held-out planning-score AUC for dense replay under the unchanged training dynamics. Continued training produces no consistent improvement when the dynamics do not change.

A.6.3 Training-Dynamics Checks

Task	Planning score	Strict success	Reward-head MSE	Decoder MSE
PushT	0.2682	0%	2.54×10^{-4}	8.10×10^{-4}
Two-Room	0.8039	100%	1.24×10^{-6}	5.21×10^{-6}
Reacher	0.9995	100%	1.36×10^{-6}	3.85×10^{-5}
OGB-Cube	0.6439	100%	3.34×10^{-5}	5.77×10^{-4}

Table 8: Planning and held-out reward-head and decoder diagnostics under the training dynamics. All reward heads and decoders have low held-out MSE, while PushT is the only task without strict CEM success.

The epoch-20 upstream validation prediction losses are 0.01291 for PushT, 0.00802 for Two-Room, 0.02702 for Reacher, and 0.01534 for OGBench-Cube. These diagnostics use overlapping random clips, so the episode-held-out errors under the training dynamics in Section A.5 are the stronger generalization check.

PushT's aggregate reward-head MSE is dominated by low-score states. Of 90,300 validation frames, 86,466 have scores below 0.1, while only 114 have scores of at least 0.5. MSE rises to 0.0154 in the score interval $[0.5, 0.7)$ and 0.0683 in $[0.7, 1.0)$, with systematic underprediction. A score-balanced diagnostic head changes CEM coverage under the training dynamics from 0.2682 to 0.3137 but produces no strict successes with either head, so we retain the original protocol and report this limitation directly.

A.6.4 Offline Reference under the Test-Time Dynamics

We include a data-rich offline reference trained directly under each set of test-time dynamics to contextualize the remaining adaptation gap. To construct it, we collect 3,000 complete episodes from the shifted environment, train a new world model from scratch for 20 epochs, fit a reward head using score labels from the same dynamics, and evaluate on 100 held-out episodes. It is not a deployable baseline and is not a mathematical upper bound because it uses a different data-collection distribution and receives 3,000 complete offline episodes from the shifted environment. Specifically, its collection policy is not task-optimal, whereas JEPA-TTT receives on-policy CEM trajectories during test-time training. The persistent result can therefore be slightly higher in some cases. Bracketed intervals below are the mean plus or minus 1.96 standard errors across the 100 episodes.

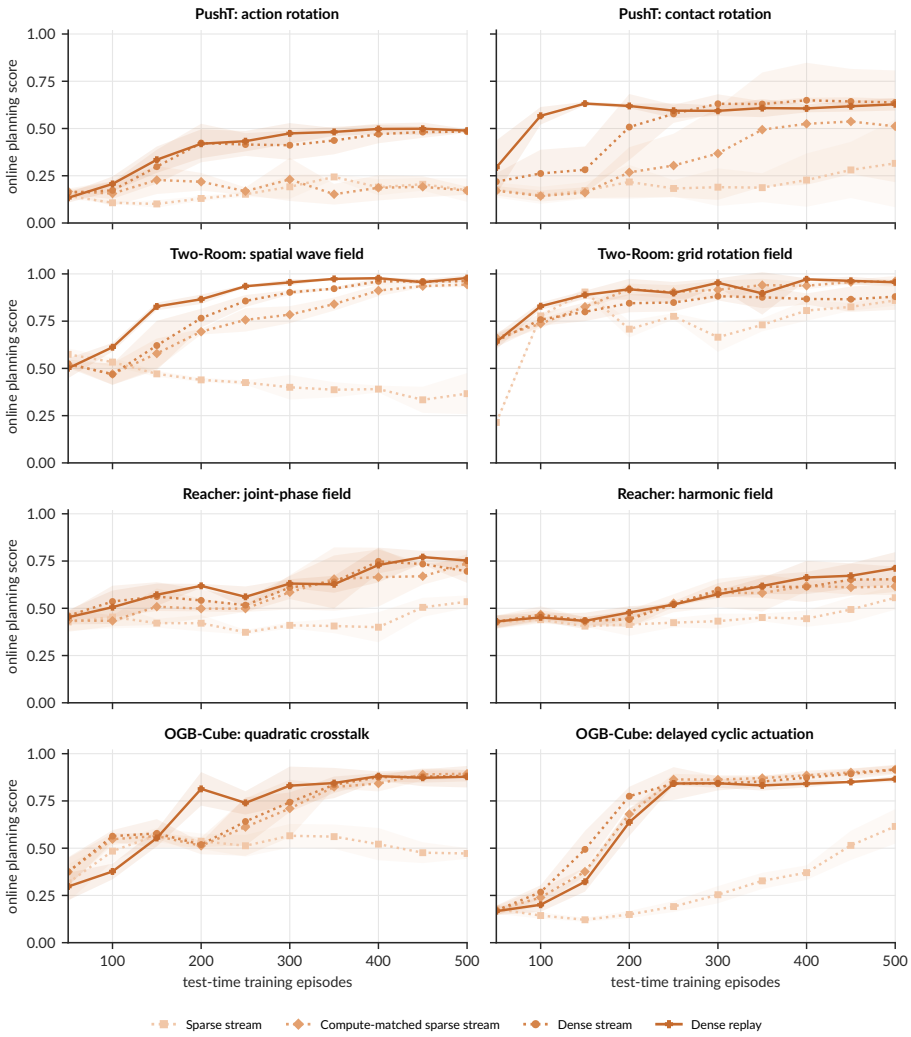


Figure 11: Online test-time performance for JEPA-TTT and the three ablations of dense replay. Curves show means ± sample standard deviations over three runs, binned into consecutive groups of 50 episodes. Dense replay achieves the highest aggregate online score.

Shift	Frozen JEPA	JEPA-TTT best	JEPA-TTT AUC	Offline model [95% CI]
PushT action rotation	0.0804	0.490 ± 0.022	0.378 ± 0.022	$0.5667\ [0.5447, 0.5886]$
PushT contact rotation	0.1268	0.619 ± 0.033	0.555 ± 0.028	$0.6321\ [0.6145, 0.6497]$
Two-Room spatial wave	0.5572	0.799 ± 0.002	0.728 ± 0.012	$0.7993\ [0.7970, 0.8017]$
Two-Room grid field	0.2106	0.799 ± 0.003	0.725 ± 0.019	$0.8021\ [0.7993, 0.8050]$
Reacher joint phase	0.3673	0.758 ± 0.012	0.619 ± 0.019	$0.9994\ [0.9993, 0.9994]$
Reacher harmonic	0.3687	0.688 ± 0.075	0.518 ± 0.012	$0.9994\ [0.9994, 0.9995]$
OGB-Cube quadratic	0.2572	0.628 ± 0.020	0.530 ± 0.009	$0.6375\ [0.6322, 0.6428]$
OGB-Cube delayed	0.1717	0.643 ± 0.003	0.513 ± 0.009	$0.6527\ [0.6465, 0.6589]$

Table 9: Offline references trained under the test-time dynamics. The best held-out score and normalized held-out planning-score AUC follow the definitions in Section 4.1. The offline reference is a separately pretrained model evaluated once on 100 held-out episodes. The best JEPA-TTT score is within 0.08 of the offline reference on six shifts, with larger gaps on the two Reacher shifts.

A.7 Online Test-Time Performance

The main paper evaluates planning performance on fixed held-out episodes. Figure 11 instead reports the scores obtained during the 500 episodes that supply test-time training data. Each point averages a consecutive block of 50 test episodes. These curves include the feedback between the changing planner and the data it collects, so they are noisier than held-out evaluations but show the performance experienced online.