NORMALIZING FLOWS ARE CAPABLE WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models predict how the world evolves in response to actions and have developed along two main directions. Video world models, most of which are built on diffusion models, generate future observations with impressive quality but typically require large models and multiple denoising steps per prediction. Latent world models instead predict and plan in a learned representation space. Those based on Joint Embedding Predictive Architectures (JEPAs) learn this space jointly with the predictor through a regression objective, but require an additional anti-collapse mechanism, and their deterministic predictors cannot represent multiple possible futures. To bridge these directions, we present Normalizing-Flow World Model (NFWM), an autoregressive normalizing flow consisting of an invertible encoder and a probabilistic predictor. Both components are trained jointly by exact maximum likelihood, which trains the predictor to model a distribution over future latent states and prevents representation collapse without an additional anti-collapse mechanism. Since the encoder is invertible, NFWM can generate future observations like a video world model, while predicting in a single step and planning in its learned latent space like a latent world model. Although the encoder preserves all information, its latent coordinates are still learned, so we can shape them for planning with an auxiliary inverse-dynamics objective. Empirically, NFWM achieves the highest average success on deterministic goal-conditioned planning and outperforms JEPA-based world models in stochastic environments and on robotic manipulation, while also supporting policy learning in imagination. These results provide initial evidence that normalizing flows are capable world models, establishing them as a promising direction for connecting video and latent world models.

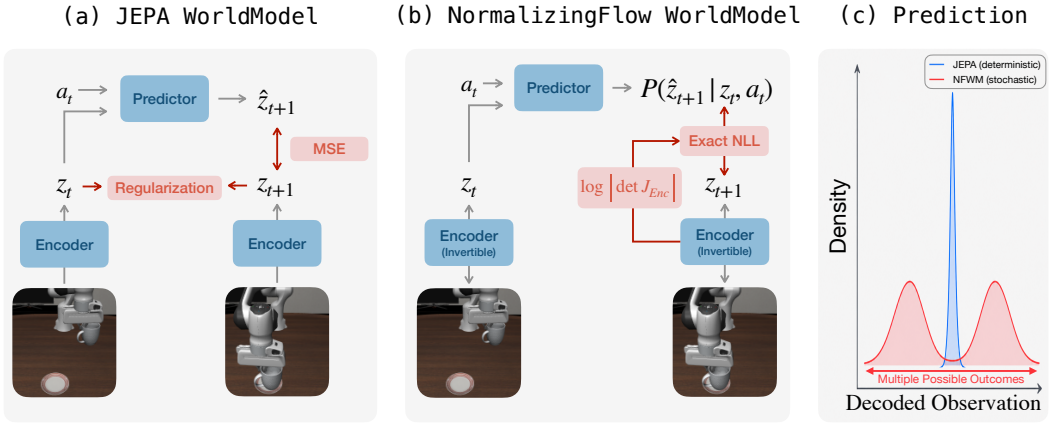

Figure 1: **Method overview.** **(a)** JEPA-based world models predict a single future latent state with a deterministic predictor, trained with an MSE loss and an additional anti-collapse mechanism (e.g., SIGReg or inverse dynamics). **(b)** NFWM predicts a distribution over future latent states. Its invertible encoder and probabilistic predictor form a single normalizing flow and are trained jointly by minimizing the negative log-likelihood (NLL), which prevents collapse without an additional anti-collapse mechanism. **(c)** Schematic comparison of deterministic and stochastic predictions.

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.

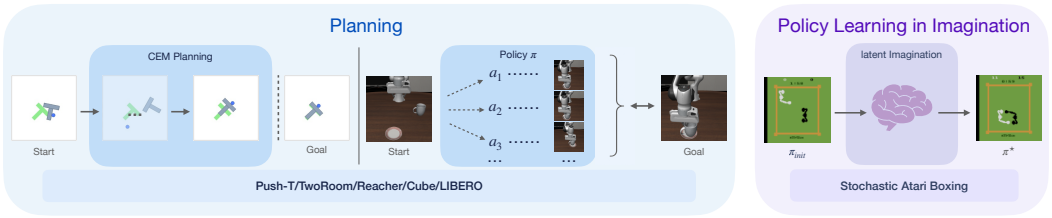

Figure 2: **Applications of NFWM.** The framework supports goal-conditioned planning, action selection, and policy learning across deterministic and stochastic environments. The figure illustrates these uses with examples from Push-T (Maes et al., 2026; Ivashkov et al., 2026), LIBERO (Liu et al., 2023), and Stochastic Atari Boxing.

1 Introduction

The ability to predict how the world will change is central to intelligence (Friston, 2010; Clark, 2013). World models learn to predict future states from observations and actions, and these predictions can support anticipation of future events and decision making (Ha & Schmidhuber, 2018; Hafner et al., 2019b; 2023; 2025; Assran et al., 2025). Recent work has followed two main directions: video world models, which predict future observations, and latent world models, which predict future states in a learned representation space.

Video world models learn distributions over future observations from large video datasets, enabling the generation of diverse, interactive environments (Bruce et al., 2024; Parker-Holder & Fruchter, 2025; Agarwal et al., 2025; Ali et al., 2025; Sun et al., 2025; Alonso et al., 2024; Valevski et al., 2025). Most recent video world models are built on diffusion models (Wan et al., 2025; Kong et al., 2024; Agarwal et al., 2025; Alonso et al., 2024; Valevski et al., 2025). Although diffusion-based world models achieve impressive generation quality, they typically rely on large models, often with billions of parameters (Wan et al., 2025; Kong et al., 2024; Agarwal et al., 2025), and each prediction requires multiple denoising steps through such a model. This cost multiplies when a planner evaluates many candidate action sequences before each decision. Few-step sampling can reduce the number of denoising steps, but it typically requires additional distillation or specialized training objectives (Zhu et al., 2026; Hafner et al., 2025).

Latent world models instead predict future states in a learned representation space, without generating future observations (Hafner et al., 2019b; 2023; Zhou et al., 2025a; Maes et al., 2026; Ivashkov et al., 2026; Zhang et al., 2026). Among latent world models, Joint Embedding Predictive Architectures (JEPAs) have recently emerged as a promising direction (Assran et al., 2023; 2025). Recent JEPA-based world models jointly train an encoder and a predictor from pixels with a regression objective in latent space, such as the mean squared error (MSE) between predicted and encoded future states (Maes et al., 2026; Ivashkov et al., 2026; Zhang et al., 2026) (Figure 1(a)). However, this approach has two limitations. First, because the objective is defined only in latent space, joint training can collapse: the encoder can map every observation to the same latent state. Preventing this requires an additional anti-collapse mechanism (Assran et al., 2023; 2025; Balestriero & LeCun, 2025; Zimmermann et al., 2025; Maes et al., 2026; Ivashkov et al., 2026; Zhang et al., 2026; Wu et al., 2026), which keeps the representation informative only approximately and may compress away information useful for downstream tasks. Second, and more fundamentally, the predictor is deterministic: it predicts a single future latent state rather than a distribution over possible futures. For a fixed encoder, the conditional mean minimizes expected squared error over unrestricted predictors. Averaging distinct futures can yield an invalid state (Alonso et al., 2024).

We introduce Normalizing-Flow World Model (**NFWM**), a world model that bridges video and latent world models (Figure 1(b)). NFWM is based on an autoregressive normalizing flow (Kumar et al., 2019; Zhai et al., 2025; Gu et al., 2025a;b) that consists of an invertible encoder and a probabilistic predictor. The encoder maps each observation to a latent state, and the predictor outputs a Gaussian distribution over the next latent state given past latent states and actions. Because the encoder is invertible, the change-of-variables formula gives the likelihood of the next observation, and we train the encoder and predictor jointly by minimizing its negative log-likelihood (NLL). The NLL consists of a prediction term and a log-Jacobian term, which address the two limitations of JEPA-based world

models. The prediction term trains the predictor to model a distribution over future latent states rather than a single future latent state. The log-Jacobian term penalizes volume contraction by the encoder. Together with invertibility, which preserves all information in each observation by construction, it prevents representation collapse without an additional anti-collapse mechanism.

NFWM can be viewed as both a video world model and a latent world model. Since the encoder is invertible, each predicted latent state can be decoded into an observation, allowing NFWM to generate future observations like a video world model. At the same time, NFWM predicts each future latent state in a single step and plans directly in its latent space, like a latent world model. Preserving all information does not mean giving up representation learning: the encoder still learns a coordinate transformation, which determines the latent geometry. For example, to support planning based on Euclidean distance in the latent space (Ivashkov et al., 2026; Zhang et al., 2026; Destrade et al., 2025; Wang et al., 2026), we can add auxiliary objectives that shape this geometry alongside the generative objective. Here, we use inverse dynamics to encourage sensitivity to action-induced changes (Ivashkov et al., 2026; Zhang et al., 2026). Since the NLL already prevents collapse, this objective serves only to shape the latent geometry.

Following JEPA-based world models (Zhou et al., 2025a; Maes et al., 2026), we evaluate NFWM on goal-conditioned planning, where it achieves the highest average success on deterministic tasks and outperforms these models in a stochastic environment and on robotic manipulation (Figure 2). Following Dreamer (Hafner et al., 2023; 2025), we further show that NFWM supports policy learning in imagination. For reference, we also report results for a video world model fine-tuned from a 1.3B-parameter Wan model (Wan et al., 2025; Zhu et al., 2026). We believe these results provide initial evidence that normalizing flows are capable world models, bridging the two main directions of world modeling.

2 Background

Video world models We consider an offline dataset $\mathcal{D} = \{(o_t, a_t, o_{t+1})\}$, where o_t is a visual observation and a_t is the action taken between consecutive observations. A video world model learns a conditional distribution over future observations (Alonso et al., 2024; Valevski et al., 2025). For a one-step context, it generates the next observation as

$$\hat{o}_{t+1} \sim p_\phi(o_{t+1} \mid o_t, a_t).$$

Sampling from this distribution allows the model to represent different possible outcomes of the same action.

JEPA JEPA-based world models predict in a learned latent space. An encoder maps observations to latent states,

$$z_t = \mathrm{Enc}_\theta(o_t), \qquad z_{t+1} = \mathrm{Enc}_\theta(o_{t+1}),$$

and a predictor estimates the next latent state:

$$\hat{z}_{t+1} = \mathrm{Pred}_\phi(z_t, e_{a_t}; h_t),$$

where $e_{a_t} = \mathrm{MLP}(a_t)$ is the action embedding and $h_t = (z_0, a_0, \ldots, z_{t-1}, a_{t-1})$ denotes the previous history. The model minimizes the prediction loss

$$\mathcal{L}_{\mathrm{fwd}} = \|\hat{z}_{t+1} - z_{t+1}\|_2^2.$$

This loss alone admits a trivial solution: the encoder and predictor can output the same constant latent state for every observation. To prevent representation collapse, LeWM regularizes the latent distribution using SIGReg (Maes et al., 2026), while SMWM and Delta-JEPA use inverse dynamics to require the latent states to retain action information (Ivashkov et al., 2026; Zhang et al., 2026).

Normalizing flows A normalizing flow models a data distribution with an invertible transformation f_θ whose Jacobian determinant is tractable. The change-of-variables formula (Dinh et al., 2016) gives

$$p_\theta(x) = p_0(f_\theta(x)) \left|\det J_{f_\theta}(x)\right|,$$

where p_0 is a prior distribution and J_{f_θ} is the Jacobian of f_θ. This allows direct maximum likelihood training by minimizing the negative log-likelihood (NLL)

$$\mathcal{L}_{\mathrm{nll}} = -\log p_0(f_\theta(x)) - \log \left|\det J_{f_\theta}(x)\right|.$$

Samples are generated by drawing $z \sim p_0$ and applying the inverse transformation $x = f_\theta^{-1}(z)$. TARFlow parameterizes the invertible transformation using a Transformer, allowing efficient computation of the Jacobian determinant (Zhai et al., 2025).

Following TARFlow (Zhai et al., 2025), we add Gaussian input noise to smooth the sharply concentrated image distribution for likelihood training, using the task-specific standard deviations in Table 14.

3 METHOD

3.1 NORMALIZING-FLOW WORLD MODEL

Model architecture NFWM consists of an invertible encoder Enc_θ and a predictor Pred_ϕ. Given a transition sample (o_t, a_t, o_{t+1}), we encode both observations as

$$z_t = \mathrm{Enc}_\theta(o_t), \qquad z_{t+1} = \mathrm{Enc}_\theta(o_{t+1}).$$

We predict a distribution over the next latent state to account for uncertainty in future outcomes. We model this distribution as a diagonal Gaussian:

$$\hat{z}_{t+1} \sim \mathcal{N}(\mu_{t+1}, \sigma_{t+1}^2), \qquad (\mu_{t+1}, \sigma_{t+1}) = \mathrm{Pred}_\phi(z_t, e_{a_t}; h_t),$$

where $e_{a_t} = \mathrm{MLP}(a_t)$ is the action embedding, and μ_{t+1} and σ_{t+1} are the predicted mean and standard deviation. We implement the encoder using TARFlow and the predictor using a Transformer.

Maximum likelihood Invertibility allows us to train latent prediction by maximizing the likelihood of the next observation. Applying the change-of-variables formula, we minimize

$$\mathcal{L}_{\mathrm{nll}} = -\log p_\phi(z_{t+1} \mid z_t, e_{a_t}; h_t) - \log \left|\det J_{\mathrm{Enc}_\theta}(o_{t+1})\right|, \tag{1}$$

where p_ϕ is the predicted Gaussian density and J_{Enc_θ} is the encoder Jacobian. For the diagonal Gaussian, the first term is

$$-\log p_\phi(z_{t+1} \mid z_t, e_{a_t}; h_t) = \frac{1}{2} \sum_j \frac{(z_{t+1,j} - \mu_{t+1,j})^2}{\sigma_{t+1,j}^2} + \sum_j \log \sigma_{t+1,j} + \mathrm{const.}$$

Equation 1 has an intuitive interpretation: the first term encourages accurate prediction of the next latent state through its conditional likelihood, while the second term penalizes local volume contraction by the encoder. Together, the invertible encoder and the log-Jacobian term help prevent representation collapse.

3.2 SHAPING LATENT GEOMETRY FOR PLANNING

Planning objective Planning can optimize different task objectives, such as expected reward or a distance to a goal. Following LeWM (Maes et al., 2026), we use the squared L_2 distance in the learned latent space for goal-conditioned planning. Given a goal observation o_g, we encode it as $z_g = \mathrm{Enc}_\theta(o_g)$. For each candidate action sequence a_{traj}, NFWM samples intermediate latent states and scores the final conditional mean:

$$a^*_{\mathrm{traj}} = \arg\min_{a_{\mathrm{traj}}} \left\| \mu_{\mathrm{final}}(a_{\mathrm{traj}}) - z_g \right\|_2^2. \tag{2}$$

This objective can be optimized over action sequences or used to rank a fixed set of candidates.

Inverse dynamics Likelihood training does not directly require this distance to be useful for choosing actions. We therefore add an auxiliary inverse-dynamics objective to shape the latent geometry for planning by encouraging sensitivity to action-induced changes (Ivashkov et al., 2026; Zhang et al., 2026).

An inverse-dynamics model (IDM), denoted by IDM_ψ, predicts the action embedding from two consecutive latent states. We minimize

$$\mathcal{L}_{\mathrm{inv}} = \left\| \hat{e}_{a_t} - \mathrm{sg}(e_{a_t}) \right\|_2^2,$$

where $\hat{e}_{a_t} = \mathrm{IDM}_\psi(z_t, z_{t+1})$, and sg stops gradients through the target action embedding e_{a_t}. We implement IDM using a two-layer Transformer and an MLP head. Implementation details are provided in Appendix D.1 (Table 13).

Training objective We combine the two losses:

$$\mathcal{L} = \mathcal{L}_{\text{nll}} + \lambda_{\text{inv}} \mathcal{L}_{\text{inv}},$$

where λ_{inv} controls the weight of the inverse-dynamics loss.

3.3 Generation and decoding

Latent-space generation Given context observations and an action sequence, we generate a future trajectory directly in latent space by repeatedly sampling from the predicted Gaussian:

$$\hat{z}_{t+1} = \mu_{t+1} + \sigma_{t+1} \odot \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, I), \tag{3}$$

where $\odot$ denotes elementwise multiplication. Each sampled latent state is used for the next prediction, allowing uncertainty to propagate through the trajectory.

Decoding and denoising The invertible encoder also provides a decoder through its inverse, which maps each sampled latent state back to an observation, $\hat{o}_{t+1} = \text{Enc}_\theta^{-1}(\hat{z}_{t+1})$. Because the model is trained on noisy observations, decoded samples also contain observation noise. For visualization, we apply one conditional score-denoising step following TARFlow (Zhai et al., 2025) to obtain cleaner images. This step uses the gradient of the learned log density and requires no additional training. Appendix D.5 gives the denoising procedure.

4 Experiments

We evaluate our model through goal-conditioned planning, robotic action selection, and policy learning across deterministic and stochastic environments. We compare our model with JEPA-based world models, LeWM (Maes et al., 2026) and SMWM (Ivashkov et al., 2026). All three use encoders of approximately 5M parameters and predictors of approximately 10M parameters. For reference, we also report a video world model fine-tuned from a 1.3B-parameter autoregressive Wan model (Wan et al., 2025; Zhu et al., 2026) with 1, 2, and 4 function evaluations (NFE). It scores planning candidates by the distance between its predicted final state and the goal in VAE latent space. Its model size, training budget, and checkpoint selection differ from those of the latent world models (Appendix D.2).

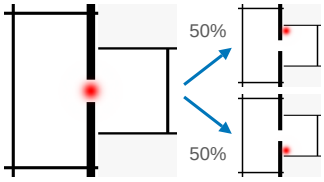

Figure 3: **Teleport environment.** Passing through the doorway teleports the agent to the upper or lower destination with equal probability.

4.1 Goal-conditioned planning

Environment and data Following LeWM (Maes et al., 2026), we use Push-T, Reacher, Cube, and TwoRoom. To evaluate planning under stochastic dynamics, we introduce Teleport, a version of TwoRoom in which entering the doorway sends the agent to one of two destinations with equal probability (Figure 3). Dataset and environment details are provided in Appendix A.1.

Setup We optimize the planning objective in Equation 2 using the Cross-Entropy Method (CEM) (Rubinstein, 1999). Each candidate uses one rollout.

On the deterministic tasks, each plan spans 25 environment steps, with replanning allowed within a total budget of 50 steps. Teleport uses a single 25-step plan without replanning, so actions are chosen before observing the teleport destination. Success is therefore theoretically bounded by 50%.

Results Figure 4 reports planning success. On the deterministic tasks, NFWM performs comparably to LeWM on Push-T, performs between the two JEPA-based baselines on Reacher and Cube, and achieves the highest success on TwoRoom. On Teleport, NFWM achieves higher success than both JEPA-based baselines.

Discussion Figure 5 shows decoded rollouts in Teleport. Both JEPA-based baselines initially produce predictions that average over the two possible futures, resulting in invalid states. By modeling

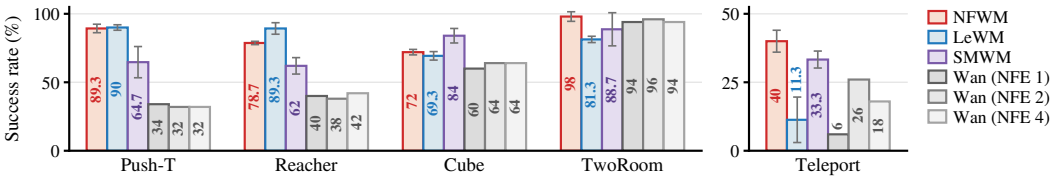

Figure 4: **Planning performance across environments.** Mean ± std over three training runs. Averaged over the four deterministic tasks, NFWM, LeWM, and SMWM achieve 84.5%, 82.5%, and 74.9% success, respectively.

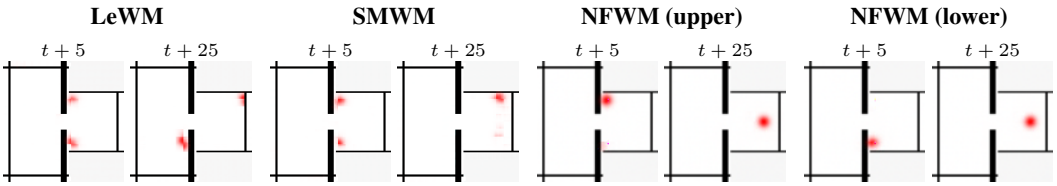

Figure 5: **Predictor rollouts on Teleport.** We visualize decoded latent states predicted from context observations and action sequences. Step $t+5$ corresponds to right after the agent passes through the doorway. Latent states are decoded by a decoder trained with the encoder frozen for LeWM and SMWM, and by the inverse of the encoder for NFWM.

a bimodal future distribution, NFWM can sample distinct branch continuations. Detailed analyses of LeWM and SMWM predictions and further qualitative results are provided in Appendices A.4 and D.5.

4.2 Action Selection

Environment and data We use a custom LIBERO task in which a robot places a white mug on a plate (Liu et al., 2023). The world model receives images from one external camera and the robot's actions. The world models are trained on successful and unsuccessful trajectories collected using a pretrained policy, $\pi_{0.5}$ (Physical Intelligence et al., 2025). Evaluation uses 50 scenes with 50 candidate action sequences per scene. Dataset and evaluation details are provided in Appendix B.

Setup We evaluate whether a world model can support robotic action selection on LIBERO. We generate candidate action sequences using $\pi_{0.5}$ and perturb them with noise. Each latent world model scores every candidate with Equation 2 and selects the lowest-cost one. The selected sequence is executed in full without replanning.

Results Figure 6 reports action-selection success. NFWM achieves higher success than both JEPA-based baselines.

Discussion We analyze failed selections of NFWM on LIBERO. Figure 7 shows the five lowest-cost candidates from one failed scene. They end in similar gripper and object configurations, yet the lowest-cost candidate fails while some others succeed. This failure may reflect limited sensitivity of latent distance to end-effector differences rather than prediction errors, since NFWM achieves similar success when model predictions are replaced with encoded final observations from actual executions (Table 1). To compare what each representation preserves, we reconstruct observations from their encoded latent states (Figure 8; SMWM in Appendix B.6). Decoders trained on LeWM and SMWM latents fail to recover details such as the gripper and mug, while NFWM preserves them. We suspect that excessive compression may discard information needed for manipulation.

Table 1: **Ranking diagnosis on LIBERO (%).** Predictor and simulator rollouts rank candidates by predicted and actual final states, respectively.

	Predictor rollout		Simulator rollout
	Top-1	Top-5	Top-1
NFWM	36.0±4.0	80.7±1.2	42.0±3.5

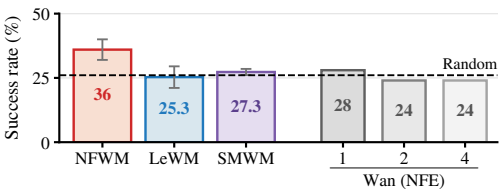

Figure 6: **Action selection performance on LIBERO.** Mean ± std over three training runs.

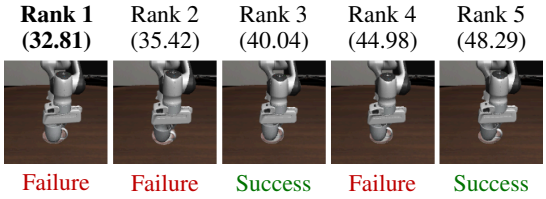

Figure 7: **Failure case on LIBERO.** Final states of the five lowest-cost candidates of NFWM.

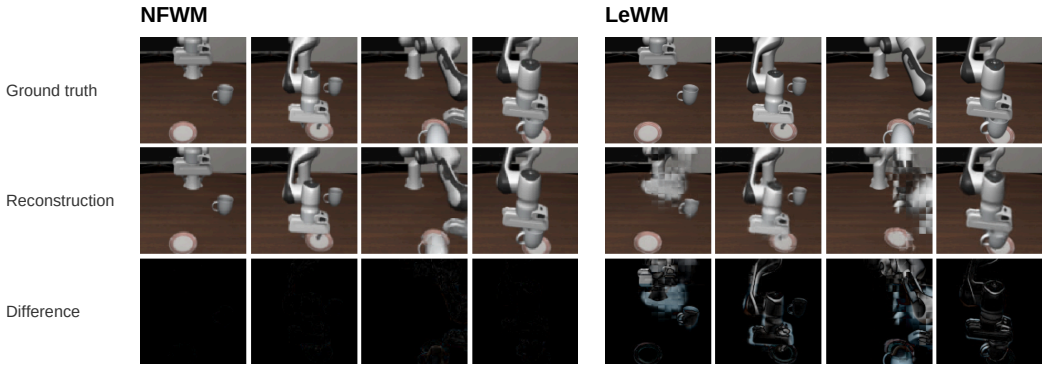

Figure 8: **Image reconstructions on LIBERO.** We show observations reconstructed from their encoded latent states and their differences from the original observations. Decoders are trained with the encoders frozen.

4.3 Policy Learning

Environment and data To evaluate policy learning under stochastic dynamics, we introduce random opponent actions into Atari Boxing. On each emulator frame, the opponent's action is replaced with probability 0.5 by a randomly selected different action. The opponent's actions are not provided to the world model. Similar to Berdica et al. (2024), the offline dataset combines trajectories from ten PPO (Schulman et al., 2017) checkpoints at different training stages (Appendix C.1).

Setup We evaluate whether a world model can support policy learning in Atari Boxing. Following Dreamer 4 (Hafner et al., 2025), we initialize an actor through behavior cloning on the offline data and then train an actor–critic inside the world model, with a KL penalty to the frozen behavior-cloned policy. For this imagination training, we freeze the world model trained on the offline data and learn reward and termination predictors from its features using a smooth L_1 loss and a class-weighted binary cross-entropy loss, respectively. Imagined trajectories start from encoded observations and use the frozen predictors for rewards and termination probabilities. We evaluate the trained policy in the environment using actual observations at every decision and report mean return over 50 evaluation episodes with sampled actions. Appendix C.3 details training, checkpoint selection, and probability calibration.

Results Table 2 reports policy-learning returns after imagination training. NFWM achieves higher return than both JEPA-based baselines. It improves over its behavior-cloned initialization in all three runs, whereas LeWM's return decreases in all three (Table 10).

Discussion To investigate the differences in returns on Atari Boxing, we first examine the policy-learning curves (Figure 9(a)). Both JEPA-based baselines improve initially, but their gains diminish with further imagination training. LeWM shows a particularly sharp decline, whereas NFWM continues to improve. In Atari Boxing, unobserved opponent actions lead to multiple possible futures. Deterministic predictors trained with MSE can average over these futures. We hypothesize that feeding these averaged predictions back into the model causes rollout drift, in which imagined latent

Table 2: **Policy-learning returns on Atari Boxing.** Mean ± std over three training runs.

Model	Return
NFWM	77.37 ± 9.28
LeWM	-35.19 ± 23.97
SMWM	40.78 ± 38.34
Wan (NFE 1)	45.1
Wan (NFE 2)	59.9
Wan (NFE 4)	63.1

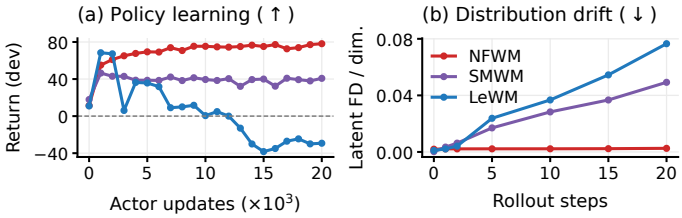

Figure 9: **Policy learning and rollout drift on Atari Boxing.** (a) Development returns during imagination training, averaged over three training runs. Step 0 is the behavior-cloned policy. (b) Fréchet distance between open-loop rollout latents and training latents, divided by the latent dimension.

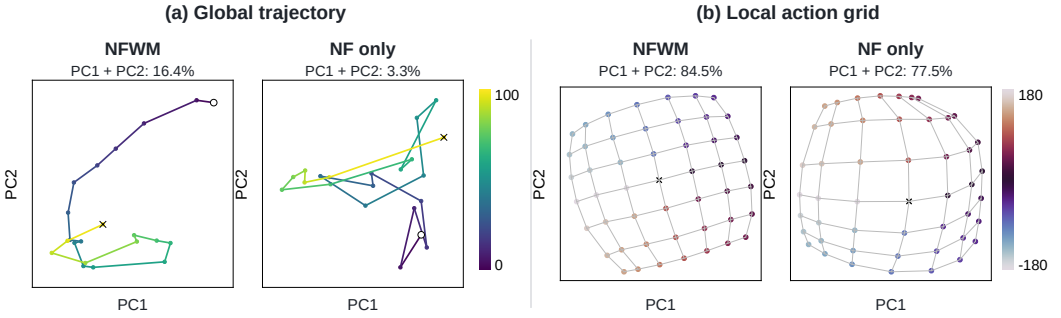

Figure 10: **Latent geometry of NFWM.** PCA of latent states from NFWM and its variant without inverse dynamics (w/o InvDyn), fitted separately for each model. (a) A held-out trajectory, colored by time; the circle and cross mark its start and end. (b) Predicted states for a grid of action offsets from one state, colored by action direction.

states move away from the data distribution, making imagined trajectories less reliable for policy learning.

To examine whether rollout drift contributes to this deterioration, we measure the Fréchet distance (FD) between rollout latent distributions and a fixed reference distribution encoded from training observations (Figure 9(b)). For NFWM, whose latent state consists of patch tokens, we compute FD at each patch position and average it over positions (patch FD). We interpret each curve relative to its model's real-future control, obtained by encoding actual future observations, rather than comparing absolute FD across models. Relative to these controls, SMWM shows slower long-horizon drift than LeWM, while NFWM's patch FD remains close to its control (Appendix C.5). These patterns suggest that rollout drift caused by averaged predictions partly explains the differences in policy-learning returns. See Appendix C.5 for a more detailed discussion of these drift patterns.

4.4 Ablations and Analysis

To assess the contribution of each objective, we compare NFWM with two variants: NLL training without inverse dynamics (w/o InvDyn), and forward MSE training with inverse dynamics (w/o NLL) (Table 3, one training run per model). Push-T planning success decreases from 86% for the full model to 34% without inverse dynamics and 28% when NLL training is replaced with forward MSE. These results support using both NLL training and the inverse-dynamics objective for planning.

Table 3: **Objective ablations.** Push-T planning success (%).

Variant	Success (%)
NFWM	86.0
w/o InvDyn	34.0
w/o NLL	28.0

Figure 10 visualizes latent geometry using PCA. NFWM exhibits fewer abrupt turns in the trajectory projection and a more regular grid of predicted states under changes in action. Without inverse

dynamics, the invertible encoder still prevents representation collapse, but latent Euclidean distances appear less useful for planning.

5 Related Work

JEPA JEPAs learn representations by predicting in latent space without reconstructing observations (Assran et al., 2023; 2025). Recent JEPA-based world models plan using the Euclidean distance between predicted and goal latent states (Maes et al., 2026; Ivashkov et al., 2026; Zhang et al., 2026). Our model plans in the same way, but predicts a distribution over future latent states and requires no anti-collapse mechanism. JEDI also jointly learns representations and future latent distributions through conditional diffusion, but focuses on policy learning rather than goal-conditioned planning (Lim et al., 2026).

Generative latent world models Generative latent world models learn distributions over future latent states (Ha & Schmidhuber, 2018). Recurrent state-space models (RSSMs) jointly learn representations and stochastic dynamics by maximizing a variational bound, and use them for model-predictive control (Hafner et al., 2019b) and policy learning in imagination (Hafner et al., 2019a; 2020; 2023). Dreamer 4 instead trains its dynamics model on top of a separately trained tokenizer (Hafner et al., 2025). These models learn policies from imagined trajectories, so policy learning depends on how well the predicted distribution captures possible futures. Our model learns policies in the same way, but also uses its latent states for goal-conditioned planning. It trains the encoder and the action-conditioned predictor jointly by exact maximum likelihood, without a variational bound or a separately trained tokenizer.

Normalizing flows Normalizing flows learn invertible mappings between data and a simple prior distribution, which allows exact likelihood training and sampling (Dinh et al., 2016). Normalizing flows have become competitive generative models for images (Zhai et al., 2025; Gu et al., 2025a), videos (Gu et al., 2025b), multimodal generation (Shen et al., 2026), and latent reasoning (Tu et al., 2026). These models use the latent space of the flow for generation and reasoning. In this paper, we instead use this space as the state space of a world model, in which we plan based on Euclidean distance and train policies.

6 Conclusion

We presented NFWM, a world model based on autoregressive normalizing flows that bridges video and latent world models. NFWM trains an invertible encoder and a probabilistic predictor jointly by exact maximum likelihood, which lets the predictor model a distribution over future latent states and prevents representation collapse without an additional anti-collapse mechanism. Because its encoder is invertible, NFWM can decode its predictions into observations like a video world model, while predicting in a single step and planning in latent space like a latent world model. Although the encoder preserves all information, its coordinate transformation is still learned, and we shape it for planning with an auxiliary inverse-dynamics objective. As shown experimentally, NFWM achieves the highest average success on deterministic goal-conditioned planning and outperforms JEPA-based world models in stochastic environments and on robotic manipulation. As a bonus, the same model supports policy learning in imagination. We hope this work encourages further study of normalizing flows as a common foundation for world models.

Limitations and future work We evaluate NFWM only in small-scale simulated environments; scaling it to larger and real-world datasets remains future work. The predictor models each future latent state with a diagonal Gaussian, which may limit its ability to represent complex, multimodal futures. NLL training alone does not yield a latent geometry well suited to Euclidean planning, so NFWM relies on the auxiliary inverse-dynamics objective (Table 3); learning such geometry directly from the generative objective is an open direction. Finally, like other action-conditioned world models, NFWM requires action-labeled trajectories, which prevents it from directly learning from action-free video.

AI USE STATEMENT

In this work, we used generative AI tools to help implement methods and to create and edit software code, including code for training models, running and managing experiments, and producing figures. We also used generative AI tools to draft and revise parts of the manuscript and to improve its clarity and readability. We have not used generative AI tools to generate synthetic data, formulate mathematical claims, or propose research hypotheses. All AI-assisted code was reviewed and tested by the authors, and all AI-assisted text was reviewed and edited by the authors. The authors take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 describes the model and training objectives, and Appendix D.1 gives the architectures and training settings of NFWM and the baselines (Tables 13 and 14). Environments, datasets, and evaluation protocols are described in Appendix A for goal-conditioned planning, Appendix B for action selection, and Appendix C for policy learning. Appendix D.2 describes the video world model reference, and Appendix D.6 specifies the number of training runs, evaluation seeds, and reporting protocol. We will release our code upon publication.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

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

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.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613/jair.3912.

Uljad Berdica, Kelvin Li, Michael Beukman, Alexander D. Goldie, Perla Maiolino, and Jakob Foerster. Robust offline learning via adversarial world models. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024. URL https://openreview.net/forum?id=dwKIyvYURI.

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

Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with jepa world models. *arXiv preprint arXiv:2601.00844*, 2025.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

D. C. Dowson and B. V. Landau. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. doi: 10.1016/0047-259X(82)90077-X. URL https://doi.org/10.1016/0047-259X(82)90077-X.

Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2): 127–138, 2010.

Jiatao Gu, Tianrong Chen, David Berthelot, Huangjie Zheng, Yuyang Wang, Ruixiang ZHANG, Laurent Dinh, Miguel Ángel Bautista, Joshua M. Susskind, and Shuangfei Zhai. STARFlow: Scaling latent normalizing flows for high-resolution image synthesis. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=3YguS2rxdk.

Jiatao Gu, Ying Shen, Tianrong Chen, Laurent Dinh, Yuyang Wang, Miguel Angel Bautista, David Berthelot, Josh Susskind, and Shuangfei Zhai. Starflow-v: End-to-end video generative modeling with normalizing flows. *arXiv preprint arXiv:2511.20462*, 2025b.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

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, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

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.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics. *arXiv preprint arXiv:2606.20104*, 2026.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A conditional flow-based model for stochastic video generation. *arXiv preprint arXiv:1903.01434*, 2019.

Jing Yu Lim, Rushi Shah, Zarif Ikram, Samson Yu, Haozhe Ma, Tze-Yun Leong, and Dianbo Liu. Jedi: Joint embedding diffusion world model for online model-based reinforcement learning. *arXiv preprint arXiv:2605.13013*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

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.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models. *Google DeepMind Blog*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143. URL https://doi.org/10.1023/A:1010091220143.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Ying Shen, Tianrong Chen, Yuan Gao, Yizhe Zhang, Yuyang Wang, Miguel Ángel Bautista, Shuangfei Zhai, Joshua M Susskind, and Jiatao Gu. Starflow2: Bridging language models and normalizing flows for unified multimodal generation. *arXiv preprint arXiv:2605.08029*, 2026.

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. Worldplay: Towards long-term geometric consistency for real-time interactive world modeling. *arXiv preprint arXiv:2512.14614*, 2025.

Guancheng Tu, Xiangjun Fu, Suhao Yu, Yao Tang, Haoqiang Kang, Lianhui Qin, Yizhe Zhang, and Jiatao Gu. Latent reasoning with normalizing flows, 2026. URL https://arxiv.org/abs/2606.06447.

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.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Jiawei Wang, Ke Rui, Yushen Zuo, Yichun Feng, and Minglei Li. Decision-metric alignment in latent world models: Diagnostics and action-conditioned objectives for mpc planning. *arXiv preprint arXiv:2608.18746*, 2026.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Haiyu Wu, Randall Balestriero, and Morgan Levine. Visreg: Variance-invariance-sketching regularization for jepa training. *arXiv preprint arXiv:2606.02572*, 2026.

Shuangfei Zhai, Ruixiang ZHANG, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Ángel Bautista, Navdeep Jaitly, and Joshua M. Susskind. Normalizing flows are capable generative models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2uheUFcFsM.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, et al. Delta-jepa: Learning action-sensitive world models via latent difference decoding. *arXiv preprint arXiv:2606.31232*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=D5RNACOZEI.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025b. URL https://arxiv.org/abs/2510.03827.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=BYInOck3gr.

Eric Zimmermann, Harley Wiltzer, Justin Szeto, David Alvarez-Melis, and Lester Mackey. Kerjepa: Kernel discrepancies for euclidean self-supervised learning. *arXiv preprint arXiv:2512.19605*, 2025.

A GOAL-CONDITIONED PLANNING

A.1 ENVIRONMENTS AND DATASETS

A.1.1 DETERMINISTIC CONTROL

We evaluate navigation, articulated control, planar manipulation, and three-dimensional manipulation using TwoRoom, Reacher, Push-T, and Cube. We follow LeWM's datasets and planning setup (Maes et al., 2026), with SMWM's held-out splitting protocol (Ivashkov et al., 2026). We randomly split whole episodes into 90% training and 10% validation data (Table 4) and sample eligible validation windows uniformly without replacement. Table 5 lists the window counts and goal offsets.

Table 4: Episode-held-out partitions for deterministic control. Action dimension is per raw environment step.

Task	Training episodes	Validation episodes	Action dimension
Push-T	16,816	1,869	2
Cube	9,000	1,000	5
Reacher	9,000	1,000	2
TwoRoom	9,000	1,000	2

A.1.2 TELEPORT

Teleport modifies TwoRoom so that reaching the doorway sends the agent to one of two destinations with equal probability. We contract the right-hand room to keep the stochastic transition within the evaluation horizon (Figure 11). On the 224×224 canvas, the door is centered at $(112, 112)$ and the destinations are $(128, 72)$ and $(128, 152)$. Beyond its entrance, the right room spans $y \in [56, 168]$. Initial positions lie in $x \in [21, 98]$, $y \in [21, 202]$, at least 85 pixels from the door center; training targets lie in $x \in [180, 202]$, $y \in [88, 136]$. The speed parameter is 5 and the expert action-noise parameter is 0.5.

The training set contains 20,000 episodes: 9,000 successful trajectories arranged as 4,500 branch pairs, 9,000 opposite-branch action replays, and 2,000 collision or exploration trajectories. Successful trajectories reach the exact target through executed corrective actions. Opposite-branch replays execute a recorded continuation from the other destination and retain both successful and unsuccessful outcomes. Pairing continuations with both branches reduces the association between expert corrections within an action block and the realized branch. Stored trajectories retain at least ten raw steps before teleportation to supply the training history.

We generate 25 additional branch pairs for validation. Each pair shares the initial state, target, and pre-teleport actions, with mirrored expert continuations and equal episode lengths. Targets have $y = 112$ and $x \in [180, 202]$. Centering the target height makes the two continuations mirror-symmetric; the generator verifies equal lengths and identical pre-teleport states. Since evaluation windows end at episode completion, unequal episode lengths could give the two branches different starting images. Equal lengths keep the window starts at the same pre-teleport state, while canonical goal rendering removes branch-dependent endpoint offsets. Together, these checks ensure that both branches have identical planner inputs. Episodes contain at least 25 transitions and reach the target within 20 transitions after teleportation. The final 25-transition window therefore begins before the stochastic event. Both branches use a canonical goal image with the agent exactly at the target; the planner receives this image and the window's initial observation. Branch identity is fixed by the episode and hidden from the model. Success uses a 16-pixel distance threshold. A predictor of the realized branch from the shared initial and goal images has chance accuracy 50% on these balanced pairs.

A.2 WORLD-MODEL TRAINING

We train each model on the corresponding dataset described above. Appendix D.1 gives the architecture, preprocessing, and optimization settings for each model; Table 14 lists the reporting epochs, observation noise, and objective weights. We normalize continuous actions using statistics from the training trajectories.

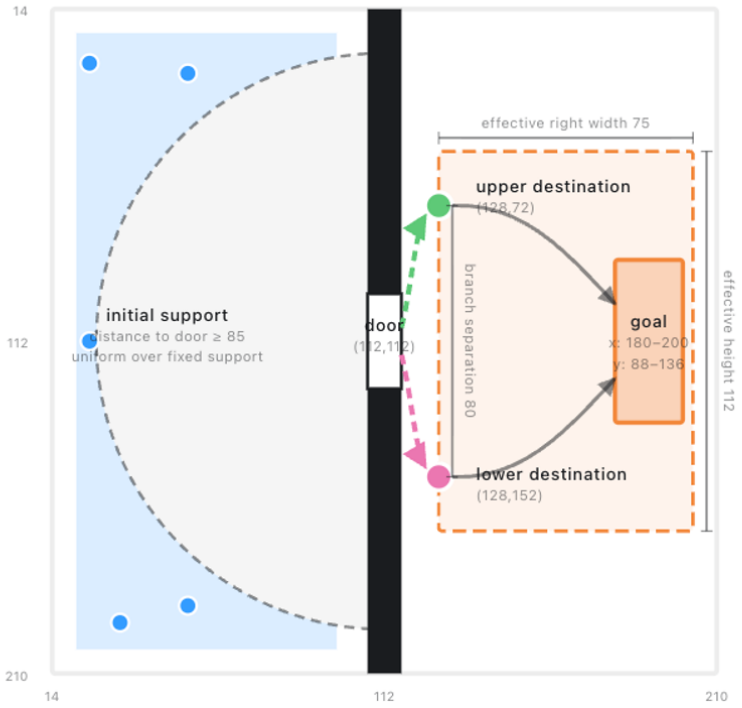

Figure 11: **Teleport geometry.** Schematic of the constrained initial region, doorway, and two teleport destinations. The contracted right room allows both branches to reach the goal within the evaluation horizon. The shaded goal box illustrates the training target region; paired evaluation fixes the target height at $y = 112$. Exact experimental ranges are specified in the text.

A.3 Planning and Evaluation

Goal-conditioned planning We rank candidate action sequences by the mean squared distance between their predicted terminal states and the encoded goal. NFWM uses all image-patch coordinates; LeWM and SMWM use their compact embeddings. CEM updates a Gaussian action distribution from the best candidates using the settings in Table 5.

Table 5: Shared CEM settings and evaluation budgets. A raw step is one environment transition, not one five-action prediction block.

Setting	Four deterministic tasks	Teleport
Candidates / elites / iterations	300 / 30 / 30	300 / 30 / 30
Initial action variance (normalized)	1	1
Fixed validation cases	50 windows per task	25 pairs / 50 windows
Initial observations / past actions	1 / none	1 / none
Goal offset (raw steps)	25	25
Predicted blocks H	5	5
Actions per block m	5	5
Executed blocks before replanning	5	All 5; no replanning
Total execution budget (raw steps)	50	25

We allow at most two planning calls on deterministic tasks and execute one plan without intermediate feedback on Teleport. We use simulator states to restore initial conditions and determine success. The model receives only images and action history. Each task uses its native success criterion, including Teleport's positional threshold described above.

Terminal conditional-mean scoring A sampled terminal state adds randomness to the goal cost even when the preceding imagined history is fixed. The *Final Mean* protocol reduces this source of noise by scoring the final conditional mean. Intermediate states are still sampled as in Equation 3:

$$C(a_{\text{traj}}) = \frac{1}{D} \left\| \mu_\phi\left(\hat{z}_{t+H-1}, a_{t+H-1}, \hat{h}_{t+H-2}\right) - z_g \right\|_2^2 .$$

The hatted history contains observed context followed by sampled states. We use one rollout per candidate and apply Final Mean to NFWM in Figure 4. The *Sample* variant also samples the final transition. Final Mean removes this last source of cost variation while retaining uncertainty from earlier transitions. It omits the variance term $D^{-1} \sum_j \sigma_j^2$ that would appear in the expected squared distance conditioned on the same history.

A.4 Stochastic Prediction Analysis

We inspect decoded Teleport predictions to assess whether they depict valid outcomes and cover both branches. This audit supports the analysis in Section 4.1.

Shared inputs and sampled predictions We use the 25 upper/lower pairs in the fixed-height Teleport validation set and the epoch-150 checkpoints from the reference training run. The initial observation is four raw steps before the teleport action; the five-action block ends at the first post-teleport frame. Initial pixels and all five actions are verified identical within each pair. No goal image or post-branch expert correction is supplied.

The gallery contains 50 rows, one per recorded episode. For each pair, NFWM receives a fixed noisy context, and we display its first two sampled predictions without selection by branch, image quality, or success. Each draw occupies one reference row, but the reference branch is not provided to the model. LeWM and SMWM receive clean contexts and produce one deterministic prediction per pair, repeated in both rows. The audit therefore covers 50 displayed images from 25 conditioning states. The selected rollouts in Figure 14 are separate qualitative examples, described in Appendix D.5.

Decoding and branch assignment NFWM uses the inverse flow without score denoising. LeWM and SMWM use matched frozen-encoder decoders trained for 75 epochs with pure pixel MSE on rendered states spanning both rooms, including intermediate positions. We visually inspect each decoded prediction and assign it to the upper or lower branch only when the agent is clearly visible at the corresponding landing. All other images remain unassigned. Either valid branch is accepted, regardless of which branch occurred in the reference episode. Table 6 reports the counts.

Table 6: **Visual assessment of decoded Teleport transitions.** Counts and percentages use 50 displayed images per model. Unassigned includes all images not accepted as either valid branch. Branch assignments use visual inspection; deterministic baseline predictions repeat within each pair. Counts describe the gallery, not calibrated conditional probabilities.

Model	Upper	Lower	Unassigned
NFWM	14 (28%)	8 (16%)	28 (56%)
LeWM	0 (0%)	0 (0%)	50 (100%)
SMWM	10 (20%)	0 (0%)	40 (80%)

Among the valid NFWM images, 63.6% depict the upper branch and 36.4% the lower branch. With two draws per condition, these are pooled descriptive frequencies.

Conditional expressivity and sample fidelity For a fixed history and action, NFWM samples a diagonal Gaussian in latent space and applies the nonlinear inverse flow. This transformation can produce a multimodal image distribution without a discrete branch selector or mixture-Gaussian head. It does not, however, constrain every sample to a valid landing region. Imperfect fitting can produce intermediate positions, fragmented agents, or other implausible images.

All histories share the same invertible encoder, so each conditional distribution must be expressed through this transformation and its diagonal-Gaussian parameters. This restriction, finite training, and observation noise may limit sample fidelity.

B Action Selection

B.1 Environment and Dataset

We use a deterministic custom version of the *put the white mug on the plate* task from LIBERO (Liu et al., 2023), through the LIBERO-Pro implementation (Zhou et al., 2025b). Initial object configurations are perturbed with magnitude $d = 0.115$. After reset and object placement, we advance the simulator for ten zero-motion control steps with the gripper commanded open, allowing contacts and object poses to settle before recording the initial state. These initialization steps precede the evaluated action sequence. This collection setup gives the $\pi_{0.5}$ policy approximately 70% success. World models use one external RGB camera and seven-dimensional actions describing translation, rotation, and gripper control. Training includes successful and unsuccessful episodes. We train every method on the same 18,685 regenerated episodes containing 4,142,719 frames. Inputs are limited to external-camera images and actions; stored wrist images and proprioception are excluded. Appendix B.3 describes the shared evaluation pool.

B.2 World-Model Training

All methods use the matched LIBERO collection described above, with the architecture and optimization settings in Appendix D.1. The LIBERO row of Table 14 specifies the reporting epoch, observation noise, and inverse-dynamics weight.

B.3 Candidate Construction

We construct a fixed pool of 50 scenes with 50 candidates per scene. From each reset state, $\pi_{0.5}$ generates fresh action sequences and may use closed-loop feedback during collection. We then freeze the sequences and add independent Gaussian noise to the first six action dimensions. Its standard deviation is half the corresponding training-action standard deviation. We clip perturbed values to the training-action extrema and leave the gripper command unchanged. Incomplete final blocks are padded with zero motion and the last gripper command for both scoring and execution.

We adjust the pool difficulty while retaining at least one candidate per scene that succeeds in the environment, without inserting a ground-truth action sequence. The pool contains 651 successful candidates out of 2,500, giving uniform random selection 26.04% expected success and an oracle 100% success. These statistics characterize a success-conditioned ranking task. LeWM (25.3%) and SMWM (27.3%) perform close to random selection. Candidate sequences, outcomes, and reset states are shared across all models.

B.4 Scoring and Execution

Each model receives the initial image and the terminal goal image from a successful reference trajectory. We predict the outcome of each candidate using the action blocks and context settings in Table 12, then execute the sequence with the lowest terminal latent cost from the saved reset state. Execution uses no replanning, and success follows the native LIBERO task criterion, checked throughout the sequence. All models share the candidate pool and evaluation randomness, with reporting checkpoints and observation noise specified in Table 14. NFWM uses a clean goal and one rollout per candidate; baseline inputs are clean.

B.5 Candidate-Ranking Diagnosis

We compare rankings based on predicted and actual terminal states to separate prediction errors from limitations of the latent distance metric. Table 1 reports the mean $\pm$ std of success across three training runs of NFWM at epoch 30, using the same frozen candidate pool and evaluation randomness. We first rank candidates by their predicted terminal costs. Top-1 measures the success of the selected candidate; Top-5 measures whether any of the five highest-ranked candidates succeeds. Uniform random ranking yields 70.30% top-five coverage, averaging the without-replacement probability across scenes. We then replace each predicted terminal state with the encoding of its actual terminal image, keeping the goal and squared-distance metric unchanged. For NFWM, actual images receive

the usual observation noise, shared across candidates within a scene; goals remain clean. Success can occur at any step, whereas the score uses only the terminal observation.

B.6 Image Reconstruction

We reconstruct LIBERO observations from their encoded latent states to examine which visual details each representation preserves (Figure 12). We use Transformer decoders with architectures similar to each model's encoder and train them for 100 epochs with pixel MSE while keeping the encoders frozen. For a fair comparison, we train a decoder on the latent states of NFWM instead of inverting its encoder. For LeWM and SMWM, the initial 12-layer decoders produce poor reconstructions, and increasing their depth to 24 layers still leaves visible errors, as shown in the figure.

NFWM more closely preserves the gripper, mug, and plate in these examples. LeWM and SMWM recover the broad scene layout but distort or blur parts of the robot and objects. We suspect that their representations discard some visual details that are difficult for a decoder to recover, even with additional layers. Losing these details may also make it harder to distinguish the small changes in gripper pose and object alignment needed for LIBERO action selection.

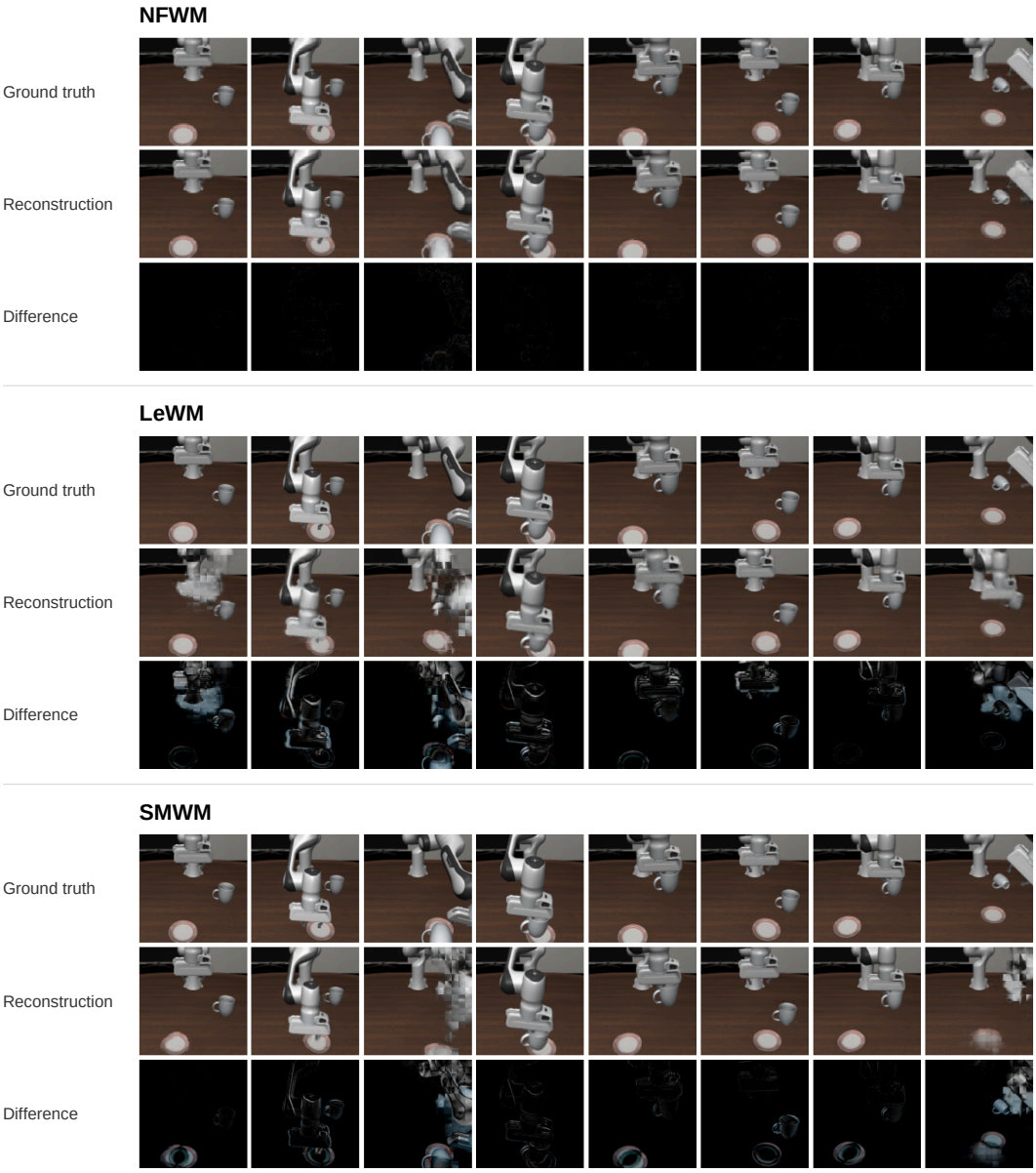

Figure 12: **Image reconstructions on LIBERO.** Each model shows eight original observations, their reconstructions from encoded latent states, and difference images. LeWM and SMWM use 24-layer decoders.

C Policy Learning

C.1 Environment and Dataset

We introduce stochastic opponent behavior into a two-player Atari Boxing environment built on the Arcade Learning Environment (ALE) (Bellemare et al., 2013). A synchronized single-player ROM supplies the opponent's actions. On each ALE frame, we independently replace its action with probability 0.50, sampling uniformly from the other 17 actions. The agent chooses among 18 actions and repeats each choice for five ALE frames; opponent randomness is applied throughout this repetition. Sticky actions are disabled. Observations are 96×96 RGB images, and unclipped player-one rewards are summed over the repeated frames. Episodes terminate at ALE game-over, with no time-limit truncations in the collected data.

We collect trajectories from ten checkpoints of one PPO run, spanning roughly one to ten million training environment steps. Each checkpoint generates 73 complete episodes and remains fixed within an episode. We sample actions from temperature-scaled categorical logits. Stronger checkpoints use higher temperatures to target returns near 30, while the earliest low-return checkpoints use temperature one. We retain all collected episodes regardless of return, yielding 1,029,628 transitions in 730 episodes with a mean return of 25.30 across the collection.

For each collector checkpoint, we assign 66 episodes to training and seven to offline validation (Table 7). We split the 70 validation episodes equally between head selection and calibration fitting (35 episodes) and a separate calibration audit (35 episodes). This partition is fixed across runs. Final policy evaluation uses independent environment episodes, as described in Appendix C.3.

Table 7: Atari Boxing dataset. Counts refer to agent-decision transitions after five-frame action repetition, not raw ALE frames.

Partition	Episodes	Transitions	Mean episode return
Training	660	930,868	25.53
Offline validation	70	98,760	23.19
Full collection	730	1,029,628	25.30

C.2 World-Model Training

We train on consecutive stored transitions, each containing five ALE frames and one categorical action, without an additional stride. LeWM and SMWM use ViT-Tiny encoders with 8×8 patches on 96×96 images and ImageNet normalization. NFWM follows Table 13, with an 18-way inverse-dynamics head trained by cross-entropy with weight 0.1. Gaussian observation noise has standard deviation 0.05 for NFWM only; the same noisy encoding is used for frozen-head training, BC, policy training, and environment evaluation. LeWM and SMWM receive clean images. LeWM uses SIGReg weight 0.09, and SMWM uses inverse-dynamics cross-entropy with weight 1. Reward and termination heads are trained after freezing the world model. Table 8 specifies the budgets for all stages.

C.3 Policy Training and Evaluation

We train a policy using experience generated by the frozen world model. We first fit reward and termination predictors, initialize an actor through behavior cloning, and then optimize an actor–critic on imagined trajectories.

Reward and termination prediction We use an MLP to jointly predict reward and a termination logit from the dynamics predictor's final hidden representation, which already encodes the history and current action. For NFWM, the head operates on each image-patch token. We average reward predictions and termination logits over tokens, then apply a sigmoid to the averaged logit. LeWM and SMWM use their compact predictor features.

We train the head on detached offline-training features using the budget in Table 8. Reward prediction uses SmoothL1 loss; termination prediction uses binary cross-entropy with positive examples

Table 8: Atari Boxing settings, fixed across models and training repetitions except for the model-specific objectives and input noise described above. Selection follows Appendix C.3.

Setting	Value
Images / action space	96×96 RGB / 18 categorical actions
World-model training	50 epochs; learning rate 5×10^{-5}; shared settings in Table 12
Frozen predictor-head training	50 epochs; batch 128; AdamW; weight decay 10^{-3}
BC budget	30,000 updates from scratch
BC / RL development evaluation	Every 1,000 updates; 30 episodes; sampled actions
Imagination budget	1,000 critic-only warm-up updates; then 20,000 actor updates
Imagination horizon / batch	20 agent steps (100 ALE frames) / 32 starts
Actor–critic optimization	Learning rate 10^{-4}; weight decay 0; gradient norm clip 1
Return / regularization	$\gamma = 0.985$; $\lambda = 0.95$; entropy coefficient 10^{-3}; BC-reference KL coefficient 0.1; critic SmoothL1 weight 0.5
Final policy reporting	Final 20,000 actor updates; 50 held-out episodes with sampled actions

weighted by 1,000. The shared head is LayerNorm–Linear(256)–GELU–Linear(2), with no separate encoder-reward branch. Both targets describe the transition (o_t, a_t, o_{t+1}). Termination denotes ALE game-over rather than truncation, and the world model remains frozen throughout head training.

Termination estimates determine imagined continuation and therefore affect the policy targets. We calibrate these estimates and select the checkpoint with the lowest reward-validation loss among those passing the checks in Table 9. We then freeze the head and calibration parameters. A one-time audit uses disjoint episodes; if it fails, we stop the pipeline rather than select another checkpoint.

Table 9: Reward/termination head selection. Calibration uses $\hat{p}_t = \mathrm{sigmoid}(a\ell_t + b)$ with $a > 0$. All methods use the same rule; head and calibration parameters are frozen before policy improvement.

Component	Setting
Selection / audit	35 / 35 disjoint episodes
Calibration validation	Two-fold episode-level cross-fitting; at least ten positive examples per fold
Eligibility	Finite metrics; calibrated Brier score and unweighted binary NLL each ≤ 1.25 times the constant-prevalence reference; mean predicted termination rate 0.25–4 times the observed rate
Selection and final fit	Lowest reward SmoothL1 among eligible checkpoints; refit calibration on selection episodes only
Initialization	Analytical $-\log(1000)$ logit correction for the positive-class training weight

Actor–critic architecture The actor–critic uses a separate attention network to process latent history and intervening actions, with the context length in Table 12. We project states to width 256; actions use 32-dimensional embeddings followed by projection. Token, time, and source-type embeddings identify the inputs. Actor and critic queries attend to this history through two cross-attention blocks with eight heads and MLP width 1,024. Separate outputs produce 18 action logits and a scalar value. The inputs are NFWM's full patch latent state or the baselines' compact states. These policy queries are separate from the auxiliary inverse-dynamics readout.

Behavior cloning and selection We train the BC actor from scratch on the offline training split using the budget and evaluation schedule in Table 8. For each run, we select the checkpoint with the highest mean development return, breaking ties by the earliest update. Selection is independent across runs, without matching returns across models or filtering by offline NLL. We freeze a copy

of the selected actor as the BC reference and begin imagination training with a critic-only warm-up. Table 10 lists the selected updates.

Imagination learning We initialize each rollout from a training observation using the context settings in Table 12 and the horizon in Table 8. The actor samples actions throughout the rollout. For NFWM, sampled latent states enter the dynamics history, while the actor and critic receive the corresponding conditional means μ after the initial observation. Using the mean removes transition-sampling noise from the actor's input, which we expect to make policy optimization more stable. This follows the same rationale as terminal conditional-mean scoring in Appendix A.3. The frozen BC reference uses the same history as the actor. Reward and termination predictions condition on the sampled dynamics history. LeWM and SMWM use their deterministic predictions for both histories.

We optimize the actor with REINFORCE (Williams, 1992) using a critic baseline and λ-returns. We detach advantages and standardize them within each batch; calibrated termination probabilities provide continuation weights. The loss also includes actor entropy, critic SmoothL1, and $D_{\mathrm{KL}}(\pi(\cdot \mid h)\|\pi_{\mathrm{BC}}(\cdot \mid h))$, with coefficients in Table 8. Gradients update only the actor–critic. We report the fixed final update as the primary result and use development-selected endpoints for secondary comparisons.

Environment evaluation and independent training runs At each environment decision, the actor encodes the new observation and samples an action using the real latent and action history. Evaluation retains the training observation preprocessing, five-frame action repetition, opponent randomness, and unclipped rewards. Development and reporting use separate episode sets with sizes given in Table 8. Environment and action-sampling randomness are fixed across models and training runs. Execution is closed loop.

Each independent run retrains the world model, predictor head, BC actor, and imagination policy with fresh training randomness. The dataset, validation partition, evaluation randomness, and other settings remain fixed. For NFWM, LeWM, and SMWM, Table 2 reports the mean and std of the three run-level mean returns at the final 20,000-update actor checkpoint. Thus, the error bars describe variation across training runs. We use neither development-best RL checkpoints nor held-out returns to select this endpoint. For Wan, we train one world model with jointly trained reward and termination heads and follow the same BC and imagination pipeline. The actor–critic reads Wan VAE latents, the reward and termination heads read the outputs of the 14th transformer block, and imagination rolls out Wan-generated latents with 1, 2, or 4 NFE. The actor is initialized from the final 30,000-update BC checkpoint, which achieves a mean held-out return of 43.1.

Table 10: Atari Boxing results by training seed. BC and final RL are mean sampled returns over the same 50 held-out environment seeds. The BC checkpoint is selected on separate development episodes; final RL always uses 20,000 actor updates after critic warm-up.

Model	Run	Selected BC update	BC	Final RL
NFWM	1	29,000	7.68	86.52
NFWM	2	27,000	9.30	77.64
NFWM	3	16,000	10.58	67.96
LeWM	1	24,000	7.54	−54.80
LeWM	2	22,000	7.52	−42.30
LeWM	3	30,000	5.46	−8.46
SMWM	1	26,000	16.84	54.64
SMWM	2	27,000	14.24	70.26
SMWM	3	19,000	15.86	−2.56

NFWM improves over its BC initialization in all three runs, whereas LeWM degrades in all three. SMWM improves in two runs and degrades in one, producing a positive aggregate but large training-seed variation.

Policy learning curves Figure 9a tracks development return from the selected BC initialization through imagination training. At each point, we average real-environment returns over three indepen-

dent training runs, using the schedule in Table 8 and identical environment seeds across updates and models. The JEPA-based baselines improve initially and then deteriorate, whereas NFWM continues to improve. These development curves use separate episodes from the held-out BC and final-policy results in Table 2.

C.4 Input-Noise Control

We test whether observation noise explains NFWM's advantage by making its representations more robust. We retrain both baselines with the same noise level throughout training and evaluation. LeWM still deteriorates during policy learning, while SMWM loses its gain (Table 11). Matched input noise therefore does not reproduce NFWM's improvement.

We repeat the complete LeWM and SMWM pipelines with independent Gaussian noise of standard deviation 0.05 in $[0, 1]$ RGB units, without clipping. We add noise before ImageNet normalization, or equivalently rescale it by channel after normalization. Noise is applied to every observed encoder input, including world-model contexts and targets, head training and calibration, BC, imagination initialization, and environment evaluation. The deterministic latent transitions receive no additional noise.

We retain the data splits, training randomness, architectures, and objectives from reference run 1, with budgets from Table 8. We reselect each head by minimum reward loss among checkpoints passing termination checks and select BC actors independently under the same budget. Each control uses one training run and evaluates the final actor on the same held-out environment episodes.

Table 11: **Boxing input-noise control (one training seed).** BC and final-RL columns are mean sampled held-out returns. Reward loss is the selected predictor head's SmoothL1 on the head-selection split, not an independent test loss. All rows use reference run 1 rather than the three-seed aggregate. The NFWM row is the reference run; the additional two runs retrain each baseline with noisy encoder inputs.

Model	Pixel-noise σ	BC	Final RL	Reward loss
NFWM	0.05	7.68	86.52	0.02423
LeWM	0	7.54	−54.80	0.02854
LeWM	0.05	6.62	−45.38	0.02604
SMWM	0	16.84	54.64	0.02676
SMWM	0.05	16.06	15.86	0.02655

Noise improves LeWM's final return relative to its clean-input run, but the result remains far below BC. SMWM loses its imagination-learning gain, while BC performance and head-selection reward loss change little. For SMWM, the final 1,000-update averages of imagined per-step reward and KL to BC rise from 0.135 to 0.362 and from 1.67 to 3.39, respectively, despite lower real return. This pattern is consistent with model-biased policy optimization. The policies visit different imagined states, so these averages do not directly measure reward error on matched states. Since noise is present during both training and evaluation, the control assesses the complete pipeline rather than isolating data augmentation. Its contribution may still depend on the model architecture.

C.5 Latent-Distribution Drift and Effective Dimension

Open-loop evaluation with fixed contexts We measure how latent predictions move away from the training distribution as rollout length increases. We use the original epoch-50 models, with clean inputs for LeWM and SMWM and training-time observation noise for NFWM. We fix 50,000 training-reference states and 50,000 distinct held-out rollout origins, allocating 5,000 origins to each PPO collector checkpoint and balancing them across its episodes. The training and held-out splits contain 660 and 70 episodes, respectively. Eligible origins have three context frames and a complete 20-step future without crossing an episode boundary. All models recursively predict along the same recorded player-one actions at horizons $h \in \{0, 1, 2, 5, 10, 15, 20\}$, with no future observation feedback. Each step is five ALE frames. Predicted termination does not censor trajectories. NFWM samples transitions at prior scale one, with one sampled rollout per origin; its encoded-frame noise is cached and shared across overlapping contexts. No actor, decoder, or visualization denoising is used.

Vector and patch-level Fréchet distance We compute Fréchet distance (FD) directly between latent distributions (Dowson & Landau, 1982). For empirical Gaussian moments (m, C) and (m_R, C_R), the distance is

$$\mathrm{FD}(Q, R) = \|m - m_R\|_2^2 + \mathrm{tr}\Big(C + C_R - 2(C_R^{1/2} C C_R^{1/2})^{1/2}\Big).$$

Before computing these moments, each coordinate is standardized with its training-reference mean and standard deviation. Variances are floored at $\max(10^{-12}, 10^{-6}\bar{v})$, where $\bar{v}$ is the reference's mean coordinate variance, computed separately at each patch position. LeWM and SMWM each use their full 192-dimensional vector and report FD $/192$.

For NFWM's 144×192 patch latent state, we compute FD at each spatial position across samples and average over positions:

$$D_{\mathrm{patch}}(h) = \frac{1}{144} \sum_{p=1}^{144} \frac{\mathrm{FD}(Q_{h,p}, R_p)}{192}.$$

This statistic preserves within-patch covariance but omits cross-patch dependence. We interpret drift relative to each model's real-future control, since the models use different representations even after standardization and division by dimension.

Observed drift and real-future control Figure 9b compares predicted endpoint distributions with the fixed training reference. At $h = 0$, the real starting latents are reused, but FD need not be zero because training and held-out samples differ. LeWM, SMWM, and NFWM begin at 0.00108, 0.00026, and 0.00195; at $h = 20$ they reach 0.07654, 0.04925, and 0.00256, respectively. SMWM's vector FD is higher than LeWM's at one and two steps but grows less at longer horizons. Encoding the actual recorded future instead of predicting it gives 20-step scores of 0.00100, 0.00235, and 0.00201 against the same reference. The baseline prediction curves therefore depart substantially from their real-future controls, while NFWM's patch distributions remain close to the level obtained from actual observations.

Effective latent dimension To examine whether representation geometry helps explain SMWM's slower drift, we measure how concentrated its latent variation is. For the unstandardized 192-dimensional embeddings, let λ_i denote training-reference covariance eigenvalues. We compute the participation ratio $r_{\mathrm{eff}} = (\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$. It is 55.59 for LeWM and 1.95 for SMWM; 54 and three principal components, respectively, explain 95% of variance. Although both embeddings have the same ambient dimension, SMWM concentrates variation in far fewer directions, consistent with its reported representation analysis (Ivashkov et al., 2026). We hypothesize that this lower effective dimension helps limit error accumulation during rollout, despite SMWM's averaged predictions.

FD measures changes in marginal means and covariances. It does not test whether individual predicted futures match their conditioning histories.

D Shared Settings and Additional Analyses

D.1 World-Model Training and Baselines

We train NFWM and the JEPA-based baselines from scratch using the common settings in Table 12. LeWM and SMWM retain their respective architectures and objectives, and all models use the same data splits on deterministic tasks. During training, predictions condition on observed histories. During multi-step rollout, predicted states enter the history recursively.

Table 12: Common world-model training and context settings. Diagnostic experiments specify any changes to rollout initialization.

Setting	Value
Training batch	128 windows
Maximum observation context	Three frames
Training window	Four observations: up to three context frames and one target
Continuous-action transitions	Five raw steps between observations; concatenate the five intervening actions
Continuous-action normalization	Statistics fitted on training trajectories for all reported evaluations
Decision-rollout initialization	One observed frame; grow the predicted context to the maximum length
World-model optimizer	AdamW; weight decay 10^{-3}; gradient-norm clip 1; bfloat16

Architecture and preprocessing Table 13 summarizes NFWM's architecture. Prediction and planning use the full image-patch latent state, while an auxiliary readout supplies features for inverse-dynamics training. We take the difference between normalized query outputs from consecutive observed states to predict a detached action encoding. Gradients update both state encodings and the readout. We average the continuous-action inverse loss over target coordinates and normalize latent NLL and log-determinant by image dimensionality, combining the latter two terms with equal weights throughout training.

Table 13: NFWM architecture for continuous-action tasks.

Component	Setting
Image / latent	96×96 RGB; 8×8 patches; 144 tokens of width 192
Flow	Four blocks, three Transformer layers per block; three heads of width 64; MLP width 768; alternating token order
Affine scale	Soft clipping at 4; positive softplus scale, lower bound 0.1
Latent predictor	Six layers; token width 192; 16 heads of width 64; MLP width 2048; dropout 0.1; action AdaLN
Conditional variance	Learned diagonal standard deviation; lower bound 10^{-4}
Auxiliary readout	Eight queries, two self-attention layers; three heads of width 64; MLP width 768; no dropout
Readout input/output	Learned slot and query embeddings; 1,536-dimensional flattened queries with non-affine LayerNorm
IDM head	MLP hidden width 128; difference of readouts; detached encoded-action target of width 192

For continuous-action tasks, NFWM resizes images to 96×96, scales them to $[0, 1]$, and adds Gaussian noise to training contexts and targets. At evaluation, observed context images receive this noise but the goal image is clean. The flow has no auxiliary register coordinates. The eight learned readout queries receive no noise and are excluded from the predictive state and planning cost. Scoring uses the latent predictions directly. LeWM and SMWM use clean 224×224 images, ImageNet normalization, and ViT-Tiny encoders with 14×14 patches. Resizing is bilinear with antialiasing and no random cropping. Their six-layer predictors operate on compact frame embeddings. Input resolution, representation size, and training objective therefore differ across methods. We match their evaluation conditions without requiring identical parameter counts.

Optimization For continuous-action tasks, NFWM and LeWM use learning rate 5×10^{-5}, and SMWM uses 10^{-4}. Table 14 gives NFWM's task settings. LeWM uses SIGReg weight 0.09. SMWM predicts normalized raw action blocks with inverse-loss weights 30, 5, 1, 0.1, 0.1, and 1 for Push-T, Reacher, Cube, TwoRoom, Teleport, and LIBERO, respectively. The four deterministic-task inverse weights are those reported by SMWM (Ivashkov et al., 2026); the SIGReg coefficient follows the published baseline setting (Maes et al., 2026; Ivashkov et al., 2026). For Teleport and LIBERO, which extend beyond those papers' control tasks, we use the reproduction settings listed above. Across tasks, we standardize batch size and context length using Table 12.

Table 14: Continuous-action training settings. Reporting epochs are fixed across methods. NFWM uses learned conditional standard deviations; image noise is measured in $[0, 1]$ units.

Task	Reporting epoch	Noise std.	λ_{inv}
Push-T	10	0.05	0.03
Cube	10	0.05	10
Reacher	10	0.05	20
TwoRoom	10	0.05	0.1
Teleport	150	0.05	1
LIBERO	30	0.08	10

D.2 Video World Model Reference (Wan)

We fine-tune the frame-wise autoregressive Wan2.1-T2V-1.3B model from Causal Forcing (Wan et al., 2025; Zhu et al., 2026) as an action-conditioned video world model. It predicts Wan VAE latents of 224×224 frames, and each latent corresponds to one five-action block (four actions on Boxing). The text input is a fixed task prompt. We encode actions with a randomly initialized encoder that follows the LeWM action-encoder architecture, and each transformer block adds a zero-initialized AdaLN projection of the action embedding to its timestep modulation. We fine-tune all parameters with flow matching and teacher forcing for 100k iterations (250k on LIBERO) without EMA, using the settings in Table 15.

At inference, the model generates each latent with 1, 2, or 4 Euler steps, conditioning on the previously generated latents. For planning and LIBERO, we score candidates by the distance between the predicted final latent and the goal latent in VAE latent space. Planning uses the CEM settings in Table 5, and LIBERO uses the candidate pool in Appendix B.

Table 15: Wan fine-tuning settings.

Setting	Value
Initialization	Frame-wise autoregressive Wan2.1-T2V-1.3B
Action conditioning	Per-block zero-initialized AdaLN
Objective	Flow matching with teacher forcing
Batch / iterations	16 clips / 100k (250k on LIBERO); no EMA
Optimizer	AdamW; learning rate 10^{-5}; weight decay 10^{-2}; gradient-norm clip 1; bfloat16
Sampling	Euler; 1, 2, or 4 NFE per latent

D.3 Objective Ablations

We ablate the training objectives to assess their contributions to Push-T planning (Table 3). The full model minimizes $\mathcal{L}_{\text{mle}} + \lambda_{\text{inv}}\mathcal{L}_{\text{inv}}$. The NF-only variant (w/o InvDyn) sets $\lambda_{\text{inv}} = 0$, retaining observation likelihood training, learned variance, and the unchanged patch-state architecture. The unused auxiliary reader is retained but receives no IDM gradient and is excluded from planning costs.

The second variant (w/o NLL) replaces both the Gaussian latent NLL and the log-determinant loss with forward MSE:

$$\mathcal{L}_{\text{MSE}} = \mathbb{E}\big[D^{-1}\|z_{t+1} - \mu_\phi(z_{\le t}, a_{\le t})\|_2^2\big] ,$$

Table 16: **Additional Push-T representation probes.** Held-out R^2 for the full reference model and its NF-only ablation at the fixed reporting epoch.

Model	Action R^2	State R^2	Displacement R^2
NFWM	**0.669**	**0.619**	**0.354**
NF only	-0.003	0.301	-0.006

where $z_t = \mathrm{Enc}_\theta(o_t)$ contains all image-patch tokens. Its objective is $\mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{inv}}\mathcal{L}_{\mathrm{inv}}$. The variance head is absent, input noise is zero, and rollout always uses the deterministic prediction. Gradients pass through the future latent target. IDM retains the two-layer query reader and detached encoded-action targets, as in the full model.

All three runs use the same held-out Push-T split and the shared settings in Table 12. We evaluate each variant at the reporting epoch in Table 14. The full and forward-MSE variants use inverse weight 0.1. Full and NF-only training retain the main model's image noise and learned variance. All three use the CEM settings and fixed cases in Table 5, with fixed evaluation randomness and all-patch MSE cost. Under the historical evaluation protocol, both the initial and goal images are noisy for the two likelihood-trained runs; the deterministic variant uses clean images.

Table 3 uses Final Mean for the likelihood-trained models and deterministic rollout for the forward-MSE variant. Entirely sampled rollout gives 84% for the full model and 34% for NF-only. Removing inverse dynamics leaves the other settings unchanged, while the forward-MSE variant also changes observation noise and rollout sampling. This ablation uses inverse weight 0.1 and noisy goals for the likelihood-trained models; the main planning result uses weight 0.03 and a clean goal. With these settings and a single training run, the full-model entry in Table 3 differs from the NFWM result in Figure 4.

D.4 Representation Analysis

We examine how inverse dynamics changes latent geometry using PCA, held-out linear probes, and local action interventions. We freeze the Push-T models at the objective ablation's reporting epoch and analyze their flattened image-patch latents. Auxiliary query outputs are excluded, since they are untrained in the NF-only variant. All variants use identical physical states and actions. The full model and NF-only share pixel-noise draws, while the forward-MSE variant uses clean inputs. These diagnostics use one checkpoint per variant. PCA illustrates the geometry; linear probes and local fits quantify it in the full latent space.

Global trajectory PCA We encode six predetermined held-out trajectories, each containing 21 observations spaced five raw steps apart. We fit PCA separately for each model using the pooled 126 states. Figure 10a shows the first trajectory for the full model and NF-only; Figure 13 adds the second trajectory and the forward-MSE variant. The trajectories follow their original order and are not selected by planning success. They contain encoded real observations rather than imagined states. The first two components explain 16.4%, 3.3%, and 70.2% of variance for the full, NF-only, and forward-MSE models, respectively. IDM-trained representations show more temporally coherent paths in these examples, whereas NF-only has more abrupt turns and overlapping segments. For NF-only, the projection captures only a small fraction of total variance, motivating the full-dimensional probes below.

Local predicted-state PCA For a fixed held-out state, we add a 7×7 grid of normalized two-dimensional offsets in $[-0.3, 0.3]^2$ to the same five-action expert block. Each offset is applied to all five actions, with native environment bounds enforced for this diagnostic. The model predicts one block from the same initial observation for each candidate. Full and NF-only predictions use a shared latent noise draw across candidates; the forward-MSE variant is deterministic. Sharing the draw controls sampling variation while actions change and applies only to this diagnostic. Figure 10b shows the full model and NF-only; Figure 13c adds the forward-MSE variant. The latter has a regular projection despite weaker planning. We therefore also measure action effects using encoded outcomes from the actual environment.

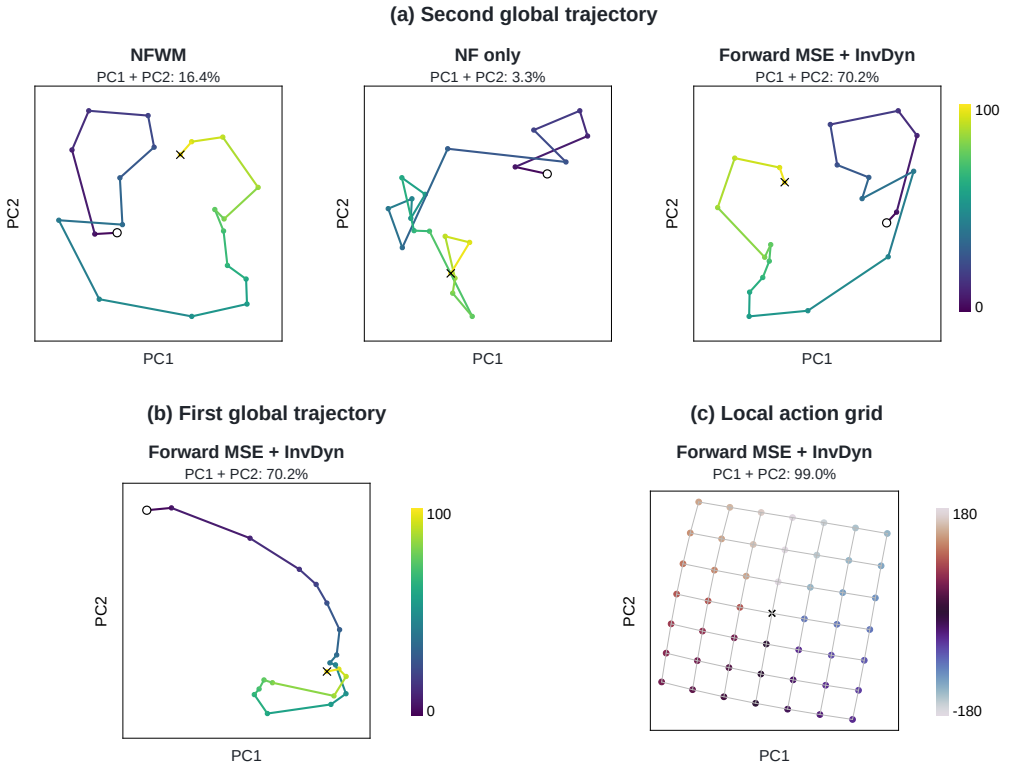

Figure 13: **Additional Push-T projections.** (a) A second real trajectory across all three objectives. (b) The first trajectory under forward MSE + IDM, complementing Figure 10a. Color denotes elapsed time. (c) Local predictions for forward MSE + IDM under the same action grid as Figure 10b; color denotes action direction. Axes are fitted separately. The deterministic variant can have orderly projections despite weaker planning.

Held-out linear probes We fit ridge probes for the full model and NF-only on 800 distinct world-model-training episodes. We use 200 additional training episodes to select regularization and 400 held-out episodes for evaluation, sampling one five-step transition per episode. We fit centering, feature scaling, and target normalization on the probe-training split; the regularization grid is $10^{-6}, \ldots, 10^{2}$. State targets comprise agent and block positions and the sine/cosine of block orientation. Displacement targets comprise their translations and wrapped block rotation. The action target is the normalized raw five-action block. State probes read z_t; displacement and action probes read $z_{t+1} - z_t$. Table 16 report test R^2, averaged uniformly over target coordinates. The scores measure how easily a linear probe can recover information. The invertible NF-only encoder may retain the same information in a form that requires a nonlinear readout.

Quantifying local linearity We also reset the actual environment to each of 16 fixed held-out states and execute the action grid at normalized radii 0.1 and 0.3. We fit an affine map from the two-dimensional action offset to the full encoded endpoint displacement on 30 actions and test it on 10 different actions. The remaining nine actions select regularization for a complementary inverse ridge probe. Common pixel noise across endpoints controls nuisance variation in the representation comparison. The forward score is $1 - \mathrm{SSE}/\mathrm{SST}$, where SST uses the fit-action mean, averaged over the 16 states. At radius 0.3, the full model scores 0.696 versus 0.497 for NF-only; at radius 0.1, the scores are 0.797 and 0.690. Within these tested states and perturbations, inverse dynamics makes the encoded effects of actions more closely follow an affine map.

The reverse local probe, from actual latent displacement to action offset, is near ceiling for both models (0.997 versus 0.991 at radius 0.3). This indicates that action effects remain recoverable without IDM. The full model makes these effects more accessible to linear readout and more regular in the forward direction. Combined with the forward-MSE variant's orderly projections but weaker planning, this suggests that useful geometry must also support accurate prediction and goal comparison.

D.5 Decoding and Prediction Visualizations

Decoding and display denoising For these visualizations, we train LeWM and SMWM decoders with pixel MSE while keeping their encoders frozen. We use epoch-75 decoder checkpoints for Teleport and epoch-200 checkpoints for Boxing. NFWM maps sampled latents through the inverse flow. For display, we apply one conditional score step inspired by TarFlow (Zhai et al., 2025):

$$y_{\mathrm{vis}} = y + \sigma_{\mathrm{obs}}^2 \nabla_y \log p_\theta(y \mid h, a), \qquad \sigma_{\mathrm{obs}} = 0.05.$$

We hold the sampled history and action fixed and clip the resulting values for display. The step uses no goal or future observation, and its output is used only for visualization.

D.6 Reporting Protocol and Statistical Details

We evaluate independently trained NFWM, LeWM, and SMWM checkpoints on the same cases or episodes, using the budgets in Tables 5 and 8. LIBERO reuses the candidate pool in Appendix B. Evaluation seeds fix CEM randomness and, for NFWM, observation noise and latent sampling. Across training runs, we vary initialization, shuffling, and training randomness while fixing the data split, hyperparameters, reporting epoch, and evaluation cases. Wan results come from a single training run per task, so we report them without standard deviations.

For success percentages $r_1, \ldots, r_n$, we report $\bar{r} = n^{-1} \sum_i r_i$ and sample standard deviation $s = \sqrt{\sum_i (r_i - \bar{r})^2/(n-1)}$ when $n > 1$.

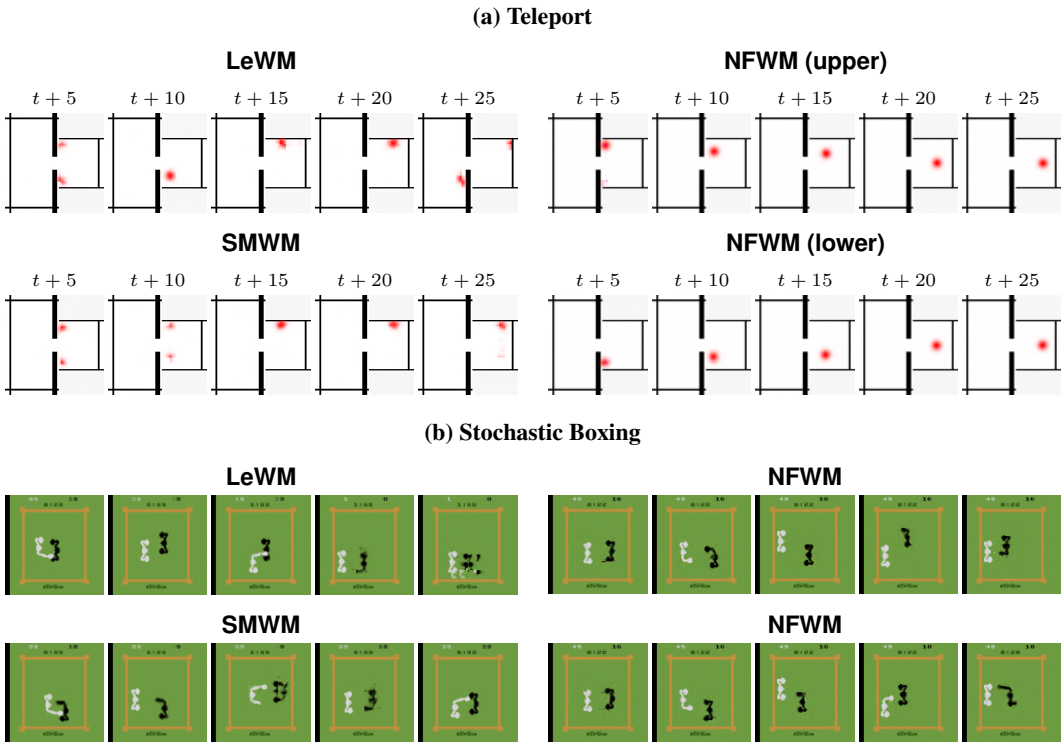

Figure 14: **Decoded predictions in stochastic environments.** (a) Selected Teleport predictions at five-step intervals. The panels share the initial observation and first action block; upper and lower NFWM continuations then follow branch-specific expert actions. (b) Atari Boxing predictions, progressing from left to right. Teleport and Atari Boxing NFWM images receive one score-denoising step for display.