

Dreamer and Related World Models

Zachary McCaw

2026-10-08

PlaNet

Hafner *et al.* [2] propose PlaNet, a model-based agent that learns latent dynamics directly from image observations and performs online planning in the learned latent space. The central modeling contribution is the Recurrent State Space Model (RSSM), which combines deterministic recurrence with stochastic latent dynamics. Given action a_{t-1} , deterministic state h_{t-1} , and stochastic state s_{t-1} , the recurrent state is updated as

$$h_t = f_\theta(h_{t-1}, s_{t-1}, a_{t-1}),$$

after which the stochastic state, observation, and reward are modeled as

$$s_t \sim p_\theta(s_t | h_t), \quad o_t \sim p_\theta(o_t | h_t, s_t), \quad r_t \sim p_\theta(r_t | h_t, s_t).$$

The deterministic path h_t provides memory over long horizons, while the stochastic state s_t allows the model to represent uncertainty and multiple plausible futures. The paper argues that both components are important: purely stochastic models can struggle to retain information over time, while purely deterministic models cannot represent multi-modal or uncertain futures as naturally.

Because the latent states are unobserved and the dynamics and observation models are nonlinear, the marginal likelihood

$$p_\theta(o_{1:T}, r_{1:T} | a_{1:T}) = \int p_\theta(s_{1:T}, o_{1:T}, r_{1:T} | a_{1:T}) ds_{1:T}$$

cannot in general be evaluated tractably. PlaNet therefore introduces an approximate filtering posterior $q_\phi(s_t | h_t, o_t)$ and maximizes a variational lower bound. In simplified form, the per-time-step objective contains an observation reconstruction term, a reward-prediction term, and a KL term,

$$\mathcal{L}_t = \mathbb{E}_{q_\phi(s_t | h_t, o_t)} [\ln p_\theta(o_t | h_t, s_t) + \ln p_\theta(r_t | h_t, s_t)] - \mathbb{KL} [q_\phi(s_t | h_t, o_t) \| p_\theta(s_t | h_t)].$$

The reconstruction term $p_\theta(o_t | h_t, s_t)$ forces the latent state to preserve information needed to explain the current observation, the reward term $\ln p_\theta(r_t | h_t, s_t)$ preserves control-relevant information, and the KL term trains the transition prior by matching it to the observation-informed posterior. Thus, during training the posterior acts as a target that has access to the true observation, whereas during planning the model must roll forward using only the learned transition prior.

PlaNet also introduces *latent overshooting* to improve multi-step prediction. Rather than training only one-step priors, the transition model is rolled forward open-loop for several steps and the resulting d -step prior is matched to the posterior at the corresponding future time:

$$\mathbb{KL}[q_\phi(s_t \mid o_{\leq t}, a_{< t}) \parallel p_\theta(s_t \mid s_{t-d}, a_{t-d:t-1})].$$

This directly penalizes compounding latent rollout error without requiring multi-step image reconstruction. In the final RSSM experiments, however, latent overshooting was not essential.

At decision time, PlaNet uses model-predictive control with the cross-entropy method. Candidate action sequences are sampled, rolled forward entirely in latent space under the RSSM, and scored using the learned reward model. CEM repeatedly refits its proposal distribution to the highest-return sequences, after which PlaNet executes only the first action and replans after receiving the next observation. Thus, PlaNet uses the learned world model directly for online control rather than learning an actor or value function.

Dreamer

Dreamer [1] retains PlaNet’s recurrent state-space model (RSSM) but replaces online CEM planning with an actor-critic learned through latent imagination. The world model consists of a representation model, transition prior, observation model, and reward model,

$$\begin{aligned} p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t), \\ q_\theta(s_t \mid s_{t-1}, a_{t-1}), \\ q_\theta(o_t \mid s_t), \\ q_\theta(r_t \mid s_t), \end{aligned}$$

and is trained from replayed real experience using the same reconstruction, reward-prediction, and KL-regularization objective as PlaNet. The observation model is used only to train the latent representation; imagined trajectories subsequently evolve entirely in latent space.

Dreamer introduces an action model $q_\phi(a_\tau \mid s_\tau)$ and value model $v_\psi(s_\tau)$. Starting from posterior states s_t inferred from real replay sequences, the agent rolls the transition model forward for an imagination horizon H using actions sampled from q_ϕ . Because maximizing only rewards occurring within these H steps would ignore consequences beyond the rollout boundary, Dreamer bootstraps from the value model. For an imagined state s_τ and a k -step return, define $h = \min(\tau + k, t + H)$ and

$$\begin{aligned} V_N^k(s_\tau) &= \mathbb{E}_{q_\theta, q_\phi} \left\{ \sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\psi(s_h) \right\}, \\ V_\lambda(s_\tau) &= (1 - \lambda) \sum_{k=1}^{H-1} \lambda^{k-1} V_N^k(s_\tau) + \lambda^{H-1} V_N^H(s_\tau). \end{aligned}$$

Here, τ indexes imagined time, t is the real-data state from which imagination begins, H is the maximum imagination horizon, k is the bootstrap horizon, h is the corresponding endpoint truncated at $t + H$, r_n is the reward predicted by the world model, γ is the discount factor, and λ controls the weighting of shorter versus longer bootstrapped returns. Shorter returns rely more heavily on the learned value function, whereas longer returns rely more heavily on imagined rewards; V_λ averages across these horizons. For continuous-control experiments, Dreamer uses $H = 15$, $\gamma = 0.99$, and $\lambda = 0.95$. Bootstrapping with v_ψ therefore allows behavior at s_τ to depend on rewards expected beyond the finite imagination horizon, which makes the learned policy substantially less sensitive to the choice of H than optimizing imagined rewards alone.

The critic regresses toward the stop-gradient λ -return,

$$\mathcal{L}_V(\psi) = \mathbb{E}_{q_\theta, q_\phi} \left[\sum_{\tau=t}^{t+H} \frac{1}{2} \{v_\psi(s_\tau) - \text{sg}[V_\lambda(s_\tau)]\}^2 \right],$$

whereas the actor maximizes the same estimated returns,

$$J(\phi) = \mathbb{E}_{q_\theta, q_\phi} \left[\sum_{\tau=t}^{t+H} V_\lambda(s_\tau) \right].$$

For continuous actions, Dreamer uses the reparameterization

$$a_\tau = \tanh\{\mu_\phi(s_\tau) + \sigma_\phi(s_\tau)\epsilon_\tau\}, \quad \epsilon_\tau \sim \mathcal{N}(0, I),$$

and similarly reparameterizes stochastic latent states. The actor gradient is then propagated analytically through the learned dynamics, rather than treating imagined trajectories as non-differentiable samples. For example, the contribution of the next-state value to the gradient of an action at s_τ contains

$$\frac{\partial v_\psi(s_{\tau+1})}{\partial \phi} = \frac{\partial v_\psi(s_{\tau+1})}{\partial s_{\tau+1}} \frac{\partial s_{\tau+1}}{\partial a_\tau} \frac{\partial a_\tau}{\partial \phi}.$$

More generally, backpropagation through successive transitions propagates gradients from future predicted rewards and values through the entire imagined trajectory. This is why a separate action-value function $Q(s_\tau, a_\tau)$ is unnecessary: the dependence of future return on a_τ is supplied explicitly by the differentiable transition model. For discrete actions, the action model instead outputs a categorical distribution and Dreamer uses straight-through gradients during imagination.

World-model and policy learning remain separated. Actor and critic gradients pass through the world model but do not update its parameters, which are learned only from real experience. Training instead iterates among three stages: update the world model from replayed environment trajectories, update the actor and critic using latent trajectories imagined from posterior replay states, and execute the updated actor in the real environment to collect new experience for the replay dataset. In the reported implementation, Dreamer alternates 100 gradient-update steps with collection of one new episode. Thus Dreamer amortizes PlaNet’s online planning into a learned policy at action-selection time, but training itself remains online and requires repeated interaction with the environment.

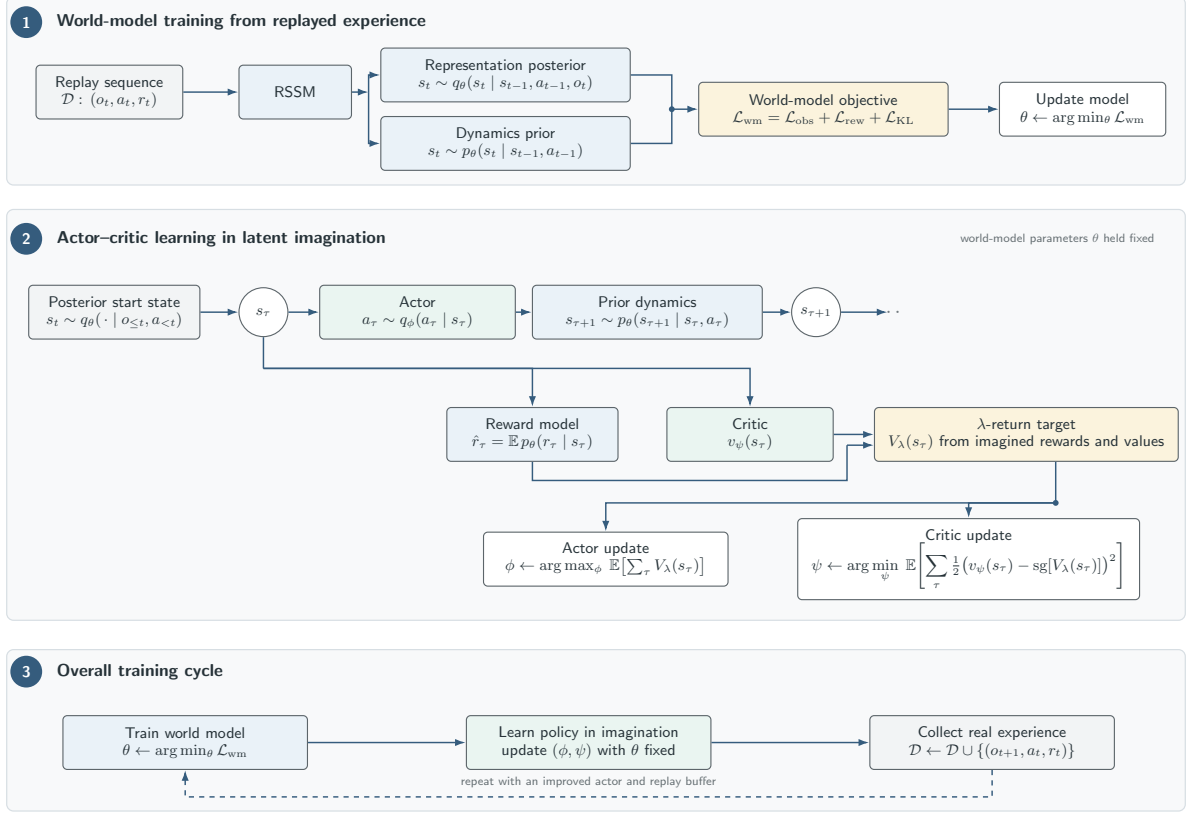


Figure 1: Overview of the Dreamer training procedure. (1) The recurrent state-space model is trained on replayed experience by jointly optimizing observation and reward prediction losses together with a regularization term aligning the representation posterior and dynamics prior. (2) Posterior states inferred from replayed observations initialize imagined trajectories. With the world model fixed, predicted rewards and critic values form bootstrapped λ -return targets used to update the actor and critic. (3) Training alternates between world-model learning, policy learning in latent imagination, and collection of new experience with the improved policy.

DreamerV2

DreamerV2 [3] extends Dreamer to Atari while retaining its basic structure: an RSSM world model is learned from real experience, and an actor and critic are then trained from trajectories imagined within the learned latent dynamics. The world model consists of a deterministic recurrent state h_t , stochastic state z_t , posterior q_ϕ , prior p_ϕ , image decoder, reward predictor, and continuation predictor:

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

The posterior uses the current observation to infer the realized state, whereas the prior predicts the state without observing x_t and is therefore used during imagination. Al-

though h_t is deterministic conditional on $(h_{t-1}, z_{t-1}, a_{t-1})$, it is marginally stochastic through its dependence on previous z_t . Stochastic latents encode that multiple future states may be compatible with the current state and action. The encoder, decoder, RSSM, and prediction heads are trained jointly from scratch.

The principal architectural change from DreamerV1 is that z_t consists of 32 categorical variables with 32 classes each rather than Gaussian variables. Discrete latents provide a natural inductive bias for abrupt state changes. They may also make the prior easier to match to the aggregate posterior. Consider the posterior:

$$q_\phi(z_t | h_t, x_t) = \text{Cat}\{z_t; \pi_\phi(h_t, x_t)\}$$

Independent of the distribution of $x_t | h_t$, the marginal distribution is:

$$\bar{q}_\phi(z_t | h_t) = \int q_\phi(z_t | h_t, x_t) p(x_t | h_t) dx_t = \text{Cat}[z_t; \mathbb{E}\{\pi_\phi(h_t, X_t) | h_t\}].$$

Thus mixtures of categorical posteriors remain categorical, whereas mixtures of Gaussian posteriors need not be Gaussian. DreamerV2 uses a factorized vector of categoricals, however, so marginalizing over x_t can induce dependence across components and this closure argument does not hold exactly for the full joint latent. Hard one-hot categorical states are used in the forward pass, with gradients propagated through their softmax probabilities using a straight-through estimator.

The world-model objective combines image reconstruction, reward prediction, continuation prediction, and prior-posterior agreement. Defining

$$\begin{aligned} \ell_{x,t} &= -\ln p_\phi(x_t | h_t, z_t), \\ \ell_{r,t} &= -\ln p_\phi(r_t | h_t, z_t), \\ \ell_{\gamma,t} &= -\ln p_\phi(\gamma_t | h_t, z_t), \\ \ell_{\text{KL},t} &= \alpha \text{KL}[\text{sg}\{q_\phi(z_t | h_t, x_t)\} \| p_\phi(z_t | h_t)] \\ &\quad + (1 - \alpha) \text{KL}[q_\phi(z_t | h_t, x_t) \| \text{sg}\{p_\phi(z_t | h_t)\}]. \end{aligned}$$

the overall loss is

$$\mathcal{L}_{\text{model}}(\phi) = \mathbb{E}_{q_\phi} \left\{ \sum_t (\ell_{x,t} + \ell_{r,t} + \ell_{\gamma,t} + \beta \ell_{\text{KL},t}) \right\}.$$

The KL-balancing parameter $\alpha = 0.8$ places greater gradient weight on fitting the prior to the more informed posterior than on regularizing the posterior toward the prior. Both categorical latents and KL balancing substantially improve Atari performance relative to their DreamerV1 counterparts.

Reinforcement learning

Policy learning is separated from world-model learning: the world-model parameters ϕ are fixed during actor-critic updates. Imagined trajectories start from posterior states inferred from real data and then evolve using the actor and learned prior,

$$\hat{a}_t \sim \pi_\psi(\hat{a}_t | \hat{s}_t), \quad \hat{h}_{t+1} = f_\phi(\hat{h}_t, \hat{z}_t, \hat{a}_t), \quad \hat{z}_{t+1} \sim p_\phi(z_{t+1} | \hat{h}_{t+1}),$$

where $\hat{s}_t = (\hat{h}_t, \hat{z}_t)$. The critic is trained toward the λ -return

$$V_t^\lambda = \hat{r}_t + \hat{\gamma}_t \left[(1 - \lambda)v_\xi(\hat{s}_{t+1}) + \lambda V_{t+1}^\lambda \right],$$

with $\lambda = 0.95$.

DreamerV2 considers REINFORCE and dynamics backpropagation as alternative actor-gradient estimators. The REINFORCE loss uses the critic as a baseline,

$$\mathcal{L}_{\text{RF}}(\psi) = -\mathbb{E} \left\{ \sum_t \ln \pi_\psi(\hat{a}_t \mid \hat{s}_t) \text{sg} [V_t^\lambda - v_\xi(\hat{s}_t)] \right\}.$$

For dynamics backpropagation, write the sampled action as $\hat{a}_t(\psi) = G_\psi\{\hat{s}_t(\psi)\}$ and the learned dynamics as $\hat{s}_{t+1}(\psi) = F_\phi\{\hat{s}_t(\psi), \hat{a}_t(\psi)\}$. Because the imagined states and actions depend recursively on ψ , so does the imagined return, which we denote $V_t^\lambda(\psi)$. The dynamics-backpropagation loss is therefore

$$\mathcal{L}_{\text{DB}}(\psi) = -\mathbb{E} \left\{ \sum_t V_t^\lambda(\psi) \right\}.$$

Its pathwise gradient is obtained by differentiating through the imagined trajectory,

$$\frac{\partial \hat{a}_t}{\partial \psi} = \frac{\partial G_\psi}{\partial \psi} + \frac{\partial G_\psi}{\partial \hat{s}_t} \frac{\partial \hat{s}_t}{\partial \psi},$$

$$\frac{\partial \hat{s}_{t+1}}{\partial \psi} = \frac{\partial F_\phi}{\partial \hat{s}_t} \frac{\partial \hat{s}_t}{\partial \psi} + \frac{\partial F_\phi}{\partial \hat{a}_t} \frac{\partial \hat{a}_t}{\partial \psi},$$

$$\frac{\partial \mathcal{L}_{\text{DB}}}{\partial \psi} = -\mathbb{E} \left\{ \sum_t \frac{\partial V_t^\lambda(\psi)}{\partial \psi} \right\}.$$

Thus changes in the actor parameters alter imagined actions, which alter subsequent latent states, rewards, and returns; dynamics backpropagation propagates the resulting return gradient through this entire computation graph. Because Atari actions and DreamerV2 latent states are discrete, the required derivatives through sampled actions and categorical states are supplied by straight-through gradients.

Defining the entropy regularizer

$$\mathcal{L}_{\text{ent}}(\psi) = -\mathbb{E} \left\{ \sum_t \mathcal{H}\{\pi_\psi(\cdot \mid \hat{s}_t)\} \right\},$$

the overall actor loss is

$$\mathcal{L}_{\text{actor}}(\psi) = \rho \mathcal{L}_{\text{RF}}(\psi) + (1 - \rho) \mathcal{L}_{\text{DB}}(\psi) + \eta \mathcal{L}_{\text{ent}}(\psi).$$

For Atari, REINFORCE performs substantially better and DreamerV2 uses $\rho = 1$; for continuous control, dynamics backpropagation performs better and $\rho = 0$. Training alternates between collecting real experience, updating the world model from replay, and updating the actor and critic from imagined trajectories.

DreamerV3

DreamerV3 [4] is an online, model-based RL algorithm that learns a latent world model jointly with an actor and critic. The world model serves as a learned simulator, allowing the actor and critic to be trained on trajectories simulated in latent space rather than solely through direct environment interaction. The model, actor, and critic are updated continually as new experience is collected and added to the replay buffer.

4.1 Notation

Let x_t denote the observation, a_t the action, r_t the immediate reward, and $c_t \in \{0, 1\}$ the continuation indicator at time t . The latent model state is $s_t = (h_t, z_t)$, where h_t is the deterministically generated recurrent state and z_t is the stochastically generated discrete latent state.

4.1.1 World model

The **world model** comprises:

1. The encoder $q_\phi(z_t | h_t, x_t)$, which approximates the posterior.
2. The recurrent transition function $h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$.
3. The latent dynamics prior $p_\phi(z_t | h_t)$.
4. The observation likelihood or decoder $p_\phi(x_t | h_t, z_t)$.
5. The reward predictor $p_\phi(r_t | s_t)$.
6. The continuation predictor $p_\phi(c_t | s_t)$.

4.1.2 Actor-critic

The policy (actor) is denoted by $\pi_\theta(a_t | s_t)$. The critic predicts a return distribution $p_\psi(\cdot | s_t)$, whose mean defines the value function $V_\psi(s_t) = \mathbb{E}_{G \sim p_\psi(\cdot | s_t)}(G)$. The discounted long-term return from time t is

$$G_t = \sum_{\tau=0}^{\infty} \gamma^\tau r_{t+\tau}.$$

Its bootstrapped λ -return estimate is defined recursively as

$$G_t^\lambda = r_t + \gamma c_t \left\{ (1 - \lambda) V_\psi(s_t) + \lambda G_{t+1}^\lambda \right\}, \quad G_T^\lambda = V_\psi(s_T).$$

4.2 World model learning

The world model is trained on sequences $\{(x_t, a_t, r_t, c_t)\}_{t=1}^T$ sampled from the replay buffer. For each sequence, the recurrent transition propagates h_t from the preceding latent state and action $h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$, while the encoder samples z_t conditional on both h_t

and the current observation x_t : $z_t \sim q_\phi(z_t | h_t, x_t)$. The parameters ϕ are optimized using a weighted combination of prediction, dynamics, and representation losses:

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

The prediction loss trains the latent state to reconstruct the observation and predict the reward and continuation indicator:

$$\mathcal{L}_{\text{pred},t}(\phi) = -\ln p_\phi(x_t | s_t) - \ln p_\phi(r_t | s_t) - \ln p_\phi(c_t | s_t).$$

The dynamics and representation losses align the latent dynamics prior with the encoder posterior:

$$\begin{aligned} \mathcal{L}_{\text{dyn},t}(\phi) &= \max \{1, \mathbb{KL} [\text{sg} \{q_\phi(z_t | h_t, x_t)\} \| p_\phi(z_t | h_t)]\}, \\ \mathcal{L}_{\text{rep},t}(\phi) &= \max \{1, \mathbb{KL} [q_\phi(z_t | h_t, x_t) \| \text{sg} \{p_\phi(z_t | h_t)\}]\}. \end{aligned}$$

The stop-gradient operator $\text{sg}(\cdot)$ causes the dynamics loss to update the prior toward the encoder posterior, whereas the representation loss updates the encoder posterior toward the prior. Clipping both KL losses below one nat implements free bits, allowing the representation to retain information needed for reconstruction without being forced to match the prior more closely once the divergence is already small.

4.2.1 Posterior estimation

For an ordinary variational objective without the unequal loss weights, stop-gradient operators, and free-bits lower bound used by DreamerV3, the reconstruction and KL terms form a negative evidence lower bound. In that idealized setting, and for fixed generative-model parameters, minimizing the objective with respect to the encoder is equivalent to minimizing

$$\mathbb{KL}\{q_\phi(z_t | h_t, x_t) \| p_\phi(z_t | h_t, x_t)\}.$$

Thus, the variational interpretation motivates treating $q_\phi(z_t | h_t, x_t)$ as an approximation to the true posterior under the learned world model. DreamerV3’s actual weighted, stop-gradient objective is a deliberately modified training criterion rather than the exact negative evidence lower bound.

4.2.2 Preventing distribution collapse

To prevent categorical distributions from collapsing to nearly deterministic predictions, Dreamer replaces each predicted distribution with a mixture containing 99% of the original distribution and 1% of the uniform distribution:

$$\tilde{p} = 0.99p + 0.01u,$$

where u is uniform over the categorical outcomes. This guarantees that every outcome retains some probability mass, which prevents excessively large KL divergences and preserves useful gradients when the prior and posterior disagree.

4.3 Actor–critic learning

The actor and critic are trained on simulated trajectories of length H generated by the world model. Each rollout begins from a latent state $s_0 = (h_0, z_0)$ inferred from a replayed observation:

$$z_0 \sim q_\phi(z_0 \mid h_0, x_0).$$

For $t = 0, \dots, H - 1$, the current actor selects an action and the world model generates the next latent state:

$$\begin{aligned} a_t &\sim \pi_\theta(a_t \mid s_t), \\ h_{t+1} &= f_\phi(h_t, z_t, a_t), \\ z_{t+1} &\sim p_\phi(z_{t+1} \mid h_{t+1}), \\ s_{t+1} &= (h_{t+1}, z_{t+1}). \end{aligned}$$

The corresponding reward and continuation indicator are generated from the simulated state:

$$\begin{aligned} r_{t+1} &\sim p_\phi(r_{t+1} \mid s_{t+1}), \\ c_{t+1} &\sim p_\phi(c_{t+1} \mid s_{t+1}). \end{aligned}$$

The world-model parameters ϕ are held fixed while the actor and critic are updated from these simulated trajectories.

The return-prediction loss fits the critic to the bootstrapped return targets:

$$\mathcal{L}_{\text{return}}(\psi) = - \sum_{t=1}^H \ln p_\psi(G_t^\lambda \mid s_t).$$

Let $\bar{\psi}$ denote an exponential moving average of the critic parameters, updated via

$$\bar{\psi} \leftarrow \rho \bar{\psi} + (1 - \rho) \psi.$$

In addition to the return-prediction loss, DreamerV3 regularizes the current critic toward outputs of this slowly moving critic. Schematically, write the combined loss as

$$\mathcal{L}_{\text{critic}}(\psi) = \mathcal{L}_{\text{return}}(\psi) + \mathcal{L}_{\text{slow}}(\psi; \bar{\psi}).$$

Here, $\rho \in (0, 1)$ controls the smoothing of the EMA parameters. The first term fits the trajectory-based target G_t^λ , and the second denotes regularization toward the stopped outputs of the slowly moving critic. The paper’s main exposition specifies this regularization but does not give a closed-form discrepancy or coefficient there, so the display is intentionally schematic rather than an asserted exact objective. The critic likelihood is applied to both imagined trajectories and replay trajectories, with respective loss scales 1 and 0.3.

The actor is optimized using REINFORCE with the critic as a baseline. Defining the advantage estimate as

$$A_t = G_t^\lambda - V_\psi(s_t),$$

the actor loss is

$$\mathcal{L}_{\text{actor}}(\theta) = - \sum_{t=1}^H \left[\text{sg} \left\{ \frac{A_t}{\max(1, S)} \right\} \ln \pi_{\theta}(a_t | s_t) + \eta \mathcal{H}\{\pi_{\theta}(\cdot | s_t)\} \right].$$

Here, $\text{sg}(\cdot)$ denotes the stop-gradient operator, η controls entropy regularization, and S is an exponential moving average, with decay 0.99, of the difference between the 95th and 5th return percentiles. This robust return-range normalization makes the relative strength of entropy regularization less sensitive to the scale of rewards across environments.

4.4 Robust prediction of large targets

4.4.1 Symlog and symexp

Observation values, rewards, and returns can differ substantially in scale across environments. Dreamer therefore uses the symmetric logarithmic transformation and its inverse:

$$\begin{aligned} \text{symlog}(y) &= \text{sign}(y) \ln(|y| + 1), \\ \text{symexp}(y) &= \text{sign}(y) \{\exp(|y|) - 1\}. \end{aligned}$$

The symlog transformation is approximately linear near zero but logarithmically compresses large positive and negative values, reducing the influence of target magnitude on optimization. Vector observations are transformed by symlog before being passed to the encoder and used as decoder targets.

For deterministic continuous targets, a network $f_{\omega}(u)$ is trained by squared error in symlog space:

$$\begin{aligned} \mathcal{L}_{\text{symlog}}(\omega) &= \frac{1}{2} \{f_{\omega}(u) - \text{symlog}(y)\}^2. \\ \hat{y} &= \text{symexp}\{f_{\omega}(u)\}. \end{aligned}$$

Thus, movement by a fixed amount in the transformed space corresponds to a multiplicative rather than additive change for large targets.

4.4.2 Symexp two-hot loss

For potentially stochastic scalar targets, including rewards and returns, Dreamer instead uses a symexp two-hot loss. The network predicts a categorical distribution over bins $b_1, \dots, b_K$ that are evenly spaced in symlog space and therefore exponentially spaced in the original scale:

$$b_k = \text{symexp}(v_k), \quad v_k \in [-20, 20].$$

A target y is represented by a two-hot vector whose nonzero weights interpolate between the two adjacent bins containing y . If $\mathbf{q}_{\omega}(u)$ denotes the predicted bin probabilities and $\mathbf{y}^{\text{2hot}}$ the encoded target, the loss and prediction are

$$\begin{aligned} \mathcal{L}_{\text{2hot}}(\omega) &= - \sum_{k=1}^K y_k^{\text{2hot}} \ln q_{\omega,k}(u), \\ \hat{y} &= \sum_{k=1}^K q_{\omega,k}(u) b_k. \end{aligned}$$

Because the loss depends on predicted probabilities rather than the numerical distances between bin values, its gradient scale is largely decoupled from the magnitude of the target. Dreamer applies this loss to the reward predictor and critic.

4.5 Overall training loop

Dreamer alternates among environment interaction, world-model learning, and actor-critic learning. First, the current actor interacts with the environment, and the resulting observations, actions, rewards, and continuation indicators are added to the replay buffer:

$$a_t \sim \pi_\theta(a_t \mid s_t),$$

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_t, a_t, r_t, c_t)\}.$$

Second, fixed-length sequences are sampled from $\mathcal{D}$ and used to update the world-model parameters ϕ through the prediction, dynamics, and representation losses. Third, latent states inferred from replayed observations initialize simulated trajectories from the updated world model, which are used to update the critic parameters ψ and actor parameters θ while holding ϕ fixed.

The cycle can therefore be summarized as

$$\begin{aligned} \pi_\theta &\longrightarrow \text{environment interaction} \longrightarrow \mathcal{D}, \\ \mathcal{D} &\longrightarrow \text{world-model update} \longrightarrow \phi, \\ (\mathcal{D}, \phi, \pi_\theta) &\longrightarrow \text{simulated trajectories} \longrightarrow (\psi, \theta). \end{aligned}$$

As the actor improves, it collects experience from new regions of the environment; this experience updates the world model, which in turn supplies improved simulations for subsequent actor-critic learning.

TD-MPC2

TD-MPC2 [6] learns a decoder-free, control-centric world model and performs model-predictive control directly in the learned latent space. The model consists of an encoder h , deterministic latent dynamics d , reward predictor R , Q -function, and policy prior p :

$$z = h(s, e), \quad z' = d(z, a, e), \quad \hat{r} = R(z, a, e), \quad \hat{q} = Q(z, a, e), \quad \hat{a} = p(z, e),$$

where e is an optional learned task embedding for multitask learning. Unlike Dreamer, TD-MPC2 does not reconstruct observations; instead, the latent representation is trained directly to retain information useful for control through joint-embedding prediction, reward prediction, and TD value prediction.

For replayed trajectories, the world model minimizes

$$\mathcal{L}(\theta) = \mathbb{E} \left[\sum_{t=0}^H \lambda^t \left\{ \|z'_t - \text{sg}\{h(s'_t)\}\|_2^2 + \text{CE}(\hat{r}_t, r_t) + \text{CE}(\hat{q}_t, q_t) \right\} \right],$$

with TD target

$$q_t = r_t + \gamma \bar{Q}\{z'_t, p(z'_t)\},$$

where $\bar{Q}$ is an exponential-moving-average target network. Thus, in contrast to purely self-supervised joint-embedding prediction, the reward and value terms explicitly shape the representation toward features relevant for control. Reward and value prediction use discrete regression in a log-transformed space, reducing sensitivity to differences in reward scale across tasks.

The latent state is normalized using SimNorm, which partitions z into groups and applies a softmax within each group:

$$g_i = \text{softmax}(z_i/\tau), \quad z^\circ = [g_1, \dots, g_L].$$

Each group therefore lies on a simplex, making SimNorm a continuous relaxation of the vector-of-categoricals representation used in Dreamer. One possible interpretation of the continuous representation is that, without a decoder, it need only preserve predictable, control-relevant information rather than support a generative distribution over observations. This interpretation is not itself established by the paper’s ablations.

TD-MPC2 trains an ensemble of five separately parameterized Q -functions to reduce overestimation bias, forming each TD target from the minimum of two randomly selected EMA target critics. Planning uses model-predictive path integral (MPPI) control, which is closely related to the cross-entropy method (CEM): both iteratively sample action sequences from a Gaussian proposal distribution and refit the proposal toward high-return samples. Whereas CEM updates the Gaussian mean using a uniformly weighted set of elite samples, MPPI uses return-dependent weights that give higher-return trajectories greater influence. After several iterations, only the first action of the optimized sequence is executed before replanning from the next state.

Dreamer 4

Dreamer 4 [5] is an offline, model-based RL algorithm that learns a scalable generative video world model and improves a task-conditioned policy entirely through simulated interaction. A causal tokenizer compresses video into continuous latent representations, and an action-conditioned transformer learns their dynamics using shortcut forcing. The transformer is subsequently fine-tuned for policy and reward prediction using offline demonstrations, after which it is frozen while the policy is improved through reinforcement learning in imagination. Unlike DreamerV3, Dreamer 4 requires no environment interaction during training.

6.1 Notation

Let x_t denote the video frame at time t , a_t the action, r_t the reward, and q_t the task identifier. A causal tokenizer encodes the video history into a fixed-length sequence of

continuous visual tokens:

$$z_t = E_\phi(x_{\leq t}).$$

A shared transformer F_θ processes these representations in two ways. For dynamics prediction, it receives corrupted latent representations $\tilde{z}$, actions, signal levels τ , and shortcut step sizes δ , and predicts the clean latent endpoint:

$$\hat{z}^1 = F_\theta(\tilde{z}, a, \tau, \delta).$$

For agent prediction, task tokens are additionally provided:

$$h_t = F_\theta(z_{\leq t}, a_{< t}, q_t),$$

where h_t denotes the contextualized representation of the task token. Separate policy, reward, and value models operate on h_t , while the dynamics predictions use the world-model token representations. Asymmetric attention prevents task information from directly influencing the world dynamics.

6.2 Causal tokenizer

The tokenizer is a block-causal transformer autoencoder with a Perceiver-style bottleneck. Learned latent tokens attend to image-patch tokens and preceding video, after which only the latent-token outputs are retained and projected through a low-dimensional tanh bottleneck. Writing P_t for the image patches and L for the learned latent tokens,

$$H_t = E_\phi(P_{\leq t}, L), \quad z_t = \tanh\{W_\phi H_t^{(L)}\},$$

where $H_t^{(L)}$ denotes the outputs corresponding to the latent tokens. The decoder reconstructs image patches from these compressed representations, remaining causal in time to support frame-by-frame generation.

The tokenizer is trained with pixelwise MSE and LPIPS:

$$\mathcal{L}_{\text{tok}}(\phi) = \mathcal{L}_{\text{MSE}}(\phi) + 0.2\mathcal{L}_{\text{LPIPS}}(\phi).$$

LPIPS compares reconstructed and target images using intermediate features of a fixed pretrained vision network, encouraging perceptual fidelity beyond pixelwise agreement. Training also uses masked autoencoding: a random fraction of image patches is replaced by learned mask embeddings, encouraging spatially coherent representations rather than direct copying of local content.

6.3 Interactive dynamics

After tokenizer training, the tokenizer is frozen. An action-conditioned, block-causal transformer F_θ learns to generate future latent representations. Its inputs include visual tokens, actions, learned register tokens, and embeddings of the signal level τ_t and shortcut step size δ_t . Tokens attend freely within a time step and causally to preceding time steps.

Rather than directly regressing future latents, Dreamer 4 uses a conditional generative model based on flow matching. A clean latent z^1 is corrupted by Gaussian noise z^0 :

$$z^0 \sim \mathcal{N}(0, I), \quad \tilde{z} = (1 - \tau)z^0 + \tau z^1.$$

Here $\tau = 0$ corresponds to pure noise and $\tau = 1$ to clean data. Given the corrupted representations, actions, signal levels, and step sizes, the transformer predicts

$$\hat{z}^1 = F_\theta(\tilde{z}, \tau, \delta, a).$$

Future latent frames are generated by progressively transforming Gaussian noise toward clean representations, conditioned on preceding video and actions.

6.4 Shortcut forcing

Shortcut forcing augments flow matching by conditioning the model on the integration step size δ . Under the flow interpretation, latent states follow an ODE with velocity $v(z, \tau)$, approximated by forward Euler integration:

$$z_{\tau+\delta} = z_\tau + \delta v(z_\tau, \tau).$$

Shortcut forcing trains the network to approximate multiple small integration steps with a single larger step. The smallest step size is supervised directly using the clean latent target z^1 ; larger steps are trained by distilling two half-steps.

For a step of size δ , the first half-step predicts the velocity and intermediate latent

$$v_{0.5} = \frac{F_\theta(\tilde{z}, \tau, \delta/2, a) - \tilde{z}}{1 - \tau}, \quad z_{0.5} = \tilde{z} + \frac{\delta}{2} v_{0.5}.$$

The second half-step predicts

$$v_1 = \frac{F_\theta(z_{0.5}, \tau + \delta/2, \delta/2, a) - z_{0.5}}{1 - (\tau + \delta/2)}.$$

Their average $\bar{v} = (v_{0.5} + v_1)/2$ serves as a stop-gradient target for the larger step. Thus, two finer Euler-style updates are recursively distilled into one coarser update.

Unlike conventional flow matching, Dreamer 4 predicts the clean latent endpoint rather than velocity. The implied velocity is

$$\hat{v} = \frac{\hat{z}^1 - \tilde{z}}{1 - \tau},$$

giving the shortcut loss

$$\mathcal{L}_{\text{shortcut}} = (1 - \tau)^2 \|\hat{v} - \text{sg}(\bar{v})\|_2^2.$$

Defining the target endpoint $\bar{z} = \tilde{z} + (1 - \tau)\bar{v}$, we have

$$(1 - \tau)^2 \|\hat{v} - \bar{v}\|_2^2 = \|\hat{z}^1 - \bar{z}\|_2^2.$$

Thus, the scaled velocity loss is equivalent to MSE between clean latent endpoints. This retains the velocity formulation needed for shortcut composition while allowing the network to predict more structured latent targets, which the authors find reduces error accumulation during autoregressive generation.

6.5 Autoregressive latent generation

At inference, each future latent frame is initialized from Gaussian noise:

$$z_{t+1}^0 \sim \mathcal{N}(0, I).$$

At signal level τ , the model predicts a clean endpoint, converts it to velocity, and advances the latent:

$$\hat{z}_{t+1}^1 = F_\theta(z_{t+1}^\tau, \tau, \delta, a_{\leq t}), \quad \hat{v}_{t+1}^\tau = \frac{\hat{z}_{t+1}^1 - z_{t+1}^\tau}{1 - \tau}, \quad z_{t+1}^{\tau+\delta} = z_{t+1}^\tau + \delta \hat{v}_{t+1}^\tau.$$

Dreamer 4 uses four updates with $\delta = 1/4$ and $\tau \in \{0, 1/4, 1/2, 3/4\}$. The resulting latent frame is appended to the context, and subsequent frames are generated autoregressively.

6.6 Agent fine-tuning

After world-model pretraining, a task-conditioned agent token is introduced at each time step. It attends to the visual representations, actions, and preceding context, but world-model tokens cannot attend back to it. This asymmetric attention allows the agent to use task information without directly altering the simulated dynamics. Let h_t denote the contextualized agent representation.

Separate policy and reward MLPs map h_t to action and reward distributions. Using offline demonstrations, Dreamer 4 trains both with multi-token prediction:

$$\mathcal{L}_{\text{agent}}(\theta) = - \sum_{n=0}^H \ln \pi_\theta(a_{t+n} \mid h_t) - \sum_{n=0}^H \ln p_\theta(r_{t+n} \mid h_t).$$

The prediction horizon is $H = 8$. Rewards use the symexp two-hot distribution, while actions use categorical or vectorized binary distributions. The pretrained transformer is fine-tuned jointly with these objectives, while the video-prediction loss remains active to preserve its generative capabilities.

6.7 Imagination training

Dreamer 4 subsequently improves the policy entirely through imagined interaction with the world model. Rollouts start from offline dataset contexts, with actions sampled from the current policy:

$$a_t \sim \pi_\theta(\cdot \mid h_t).$$

The frozen dynamics model generates the next latent representation, and the resulting imagined trajectory supplies subsequent agent representations and predicted rewards $\hat{r}_t$. Thus, the policy interacts on-policy with the learned simulator without further real-world experience.

A value MLP maps h_t to a return distribution $p_\psi(\cdot | h_t)$ with expectation $V_\psi(h_t)$. The value model is trained using bootstrapped λ -returns:

$$R_t^\lambda = \hat{r}_t + \gamma c_t \{(1 - \lambda)V_\psi(h_t) + \lambda R_{t+1}^\lambda\}, \quad R_T^\lambda = V_\psi(h_T).$$

The value loss is the negative log likelihood of these returns under a symexp two-hot distribution:

$$\mathcal{L}_{\text{value}}(\psi) = - \sum_{t=1}^T \ln p_\psi(R_t^\lambda | h_t).$$

During standard imagination training, the tokenizer and dynamics transformer are frozen; only the policy and value models are updated.

6.8 Policy optimization

Dreamer 4 uses PMPO, which depends on the sign rather than the magnitude of the estimated advantage:

$$A_t = R_t^\lambda - V_\psi(h_t).$$

Define $\mathcal{D}^+ = \{t : A_t \geq 0\}$ and $\mathcal{D}^- = \{t : A_t < 0\}$. PMPO increases the likelihood of actions with nonnegative advantages and decreases that of actions with negative advantages:

$$\begin{aligned} \mathcal{L}_{\text{PMPO}}(\theta) = & \frac{1 - \alpha}{|\mathcal{D}^-|} \sum_{t \in \mathcal{D}^-} \ln \pi_\theta(a_t | h_t) \\ & - \frac{\alpha}{|\mathcal{D}^+|} \sum_{t \in \mathcal{D}^+} \ln \pi_\theta(a_t | h_t) \\ & + \frac{\beta}{T} \sum_{t=1}^T \text{KL}\{\pi_\theta(\cdot | h_t) \| \pi_{\text{prior}}(\cdot | h_t)\}. \end{aligned}$$

Dreamer 4 uses $\alpha = 0.5$, balancing the aggregate contributions of positive- and negative-advantage transitions. Discarding advantage magnitudes makes the objective insensitive to reward scale, avoiding DreamerV3’s explicit return normalization.

The reference policy π_{prior} is a frozen copy of the behavior-cloned policy. The reverse-KL penalty $\text{KL}\{\pi_\theta \| \pi_{\text{prior}}\}$ discourages departures from behaviors represented in the offline dataset, reducing the risk of exploiting inaccuracies in poorly supported regions of the learned world model.

6.9 Overall training procedure

Dreamer 4 follows three principal stages:

offline video $\longrightarrow$ causal tokenizer $\longrightarrow$ shortcut-trained dynamics model,
offline actions, tasks, and rewards $\longrightarrow$ task-conditioned policy and reward prediction,
imagined trajectories $\longrightarrow$ value learning and PMPO policy improvement.

The tokenizer is frozen after reconstruction training. During agent fine-tuning, the dynamics transformer and agent prediction models are trained jointly while retaining video-prediction supervision. Finally, the learned simulator is frozen and the policy is improved using imagined trajectories, a learned value function, and regularization toward the behavioral prior. Unlike DreamerV3, the entire procedure uses a fixed offline dataset rather than alternating policy improvement with new environment interaction.

Contrastive World Models

Li [7] proposes Contrastive World Models (CWM). CWM replaces Dreamer’s pixel-reconstruction objective with a contrastive mutual-information objective while retaining the Recurrent State Space Model (RSSM) and actor-critic framework. Given an observation o_t , an encoder first produces $z_t = E(o_t)$. The RSSM maintains a recurrent latent state that combines information propagated through the dynamics with the current observation: we distinguish the transition-prior draw $\tilde{s}_t$ from the transition-posterior draw s_t as

$$\tilde{s}_t \sim p_\theta(\tilde{s}_t \mid s_{t-1}, a_{t-1}), \quad s_t \sim q_\theta(s_t \mid \tilde{s}_t, z_t), \quad r_t \sim p_\theta(r_t \mid s_t).$$

The transition posterior therefore incorporates the new observation embedding into the recurrent state, while the transition prior predicts forward using only the preceding state and action. Standard Dreamer additionally decodes s_t to reconstruct o_t ; CWM removes this pixel decoder, sharing LeWM’s motivation of learning world models directly in representation space rather than forcing the latent state to retain all reconstructable pixel detail. Unlike LeWM, however, CWM retains Dreamer’s stochastic RSSM, reward prediction, and actor-critic learning, and replaces reconstruction with a contrastive rather than direct latent-regression objective.

To define the contrastive objective, let $c_{t,h} = [s_t, a_{t:t+h-1}]$ denote the current recurrent state concatenated with the h intervening actions, and let $z_{t+h}^{m,n} = E_{m,n}(o_{t+h})$ denote the local encoder feature at spatial location (m,n) of the future observation. We use a half-open action slice so that $c_{t,1} = (s_t, a_t)$; this resolves the paper’s inclusive-looking shorthand $a_{t:t+h}$ without changing its intended semantics. CWM seeks to maximize the mutual information between the state-action context and future local features,

$$I(c_{t,h}; z_{t+h}^{m,n}) = \mathbb{E} \left[\ln \frac{p(c_{t,h}, z_{t+h}^{m,n})}{p(c_{t,h})p(z_{t+h}^{m,n})} \right] = \mathbb{E} \left[\ln \frac{p(z_{t+h}^{m,n} \mid c_{t,h})}{p(z_{t+h}^{m,n})} \right].$$

The model borrows from Deep InfoMax the idea of relating a *global* representation to many *local* features: here the global representation is the recurrent state together with the action sequence, while the local features are spatial embeddings of the future observation. Conceptually, the model is trained to distinguish local features belonging to the actual future observation from features of other observations, encouraging $c_{t,h}$ to retain information that is consistently predictive across the future scene without requiring reconstruction of every pixel. Using the bilinear score adopted in the paper, the corresponding InfoNCE objective is

$$J_I^t = \frac{1}{MN} \sum_{m,n} \ln \frac{\exp(c_{t,h}^\top W_\theta z_{t+h}^{m,n})}{\sum_{b \in B} \exp(c_{t,h}^\top W_\theta z_b^{m,n})}.$$

CWM combines this representation objective with the RSSM transition and reward objectives,

$$J_D^t = -\mathbb{KL} \{q_\theta(\cdot \mid \tilde{s}_{t+1}, z_{t+1}) \parallel p_\theta(\cdot \mid s_t, a_t)\}, \quad J_R^t = \ln p_\theta(r_t \mid s_t),$$

and optimizes

$$J_{\text{CWM}} = \mathbb{E} \left\{ \frac{1}{T} \sum_t (J_I^t + \lambda_1 J_D^t + \lambda_2 J_R^t) \right\}.$$

The comparison with LeWM is instructive. Given z_t and a_t , LeWM learns to predict z_{t+1} by minimizing the MSE. By contrast, given $c_{t,1} = (s_t, a_t)$, CWM learns to predict z_{t+1} by maximizing mutual information. Thus, the key differences are that (i) CWM conditions on the posterior recurrent state s_t rather than the direct observation encoding z_t and (ii) CWM maximizes mutual information with the future encoding rather than minimizing prediction MSE.

References

- [1] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv*, 12 2019. <https://arxiv.org/abs/1912.01603>.
- [2] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv*, 11 2018. <https://arxiv.org/abs/1811.04551>.
- [3] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 10 2020. <https://arxiv.org/abs/2010.02193>.
- [4] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv*, 01 2023. <https://arxiv.org/abs/2301.04104>.
- [5] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv*, 09 2025. <https://arxiv.org/abs/2509.24527>.
- [6] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. *arXiv*, 10 2023. <https://arxiv.org/abs/2310.16828>.
- [7] Bonnie Li. Contrastive world models. *arXiv*, 08 2026. <https://arxiv.org/abs/2609.22175>.