

CIS 6280 · FALL 2026

WORLD MODELS

LECTURE 8

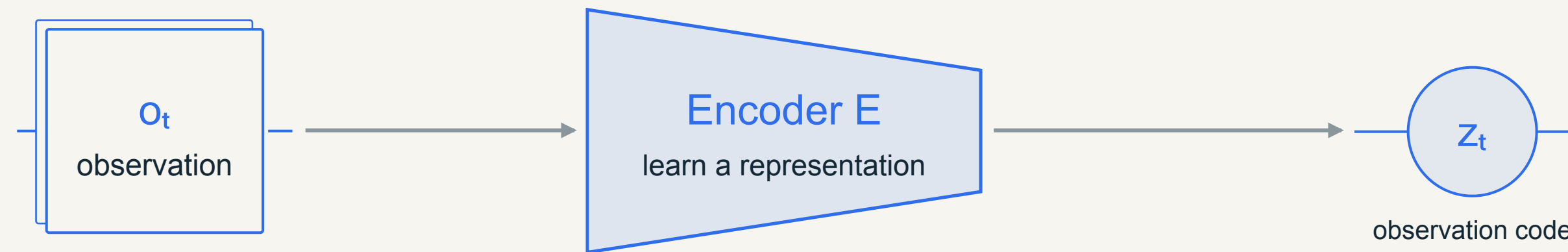
Latent World Models

Course announcements

Item	Details
Assignment 1	Due Thursday, September 24 . Check Canvas for the exact submission deadline.
September 22 class	Tentative: Zoom . Check Canvas for confirmation and the link.
September 24 class	May also be on Zoom due to travel . Check Canvas for confirmation.
Office hours this week	Moved to Friday, September 25 at 5:00 PM .

Today: design, train, and evaluate latent predictive models.

State, representation, and inference



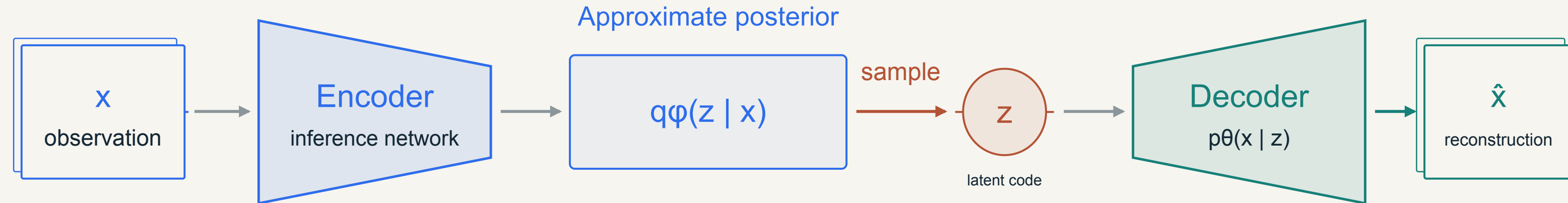
SSM: a state explains how observations evolve.

Representation learning: an objective decides what the code preserves.

An observation code is not automatically a sufficient predictive state.

VAE at the observation interface

L7 notation: x is one observation. In L8, o_t denotes an observation at time t .

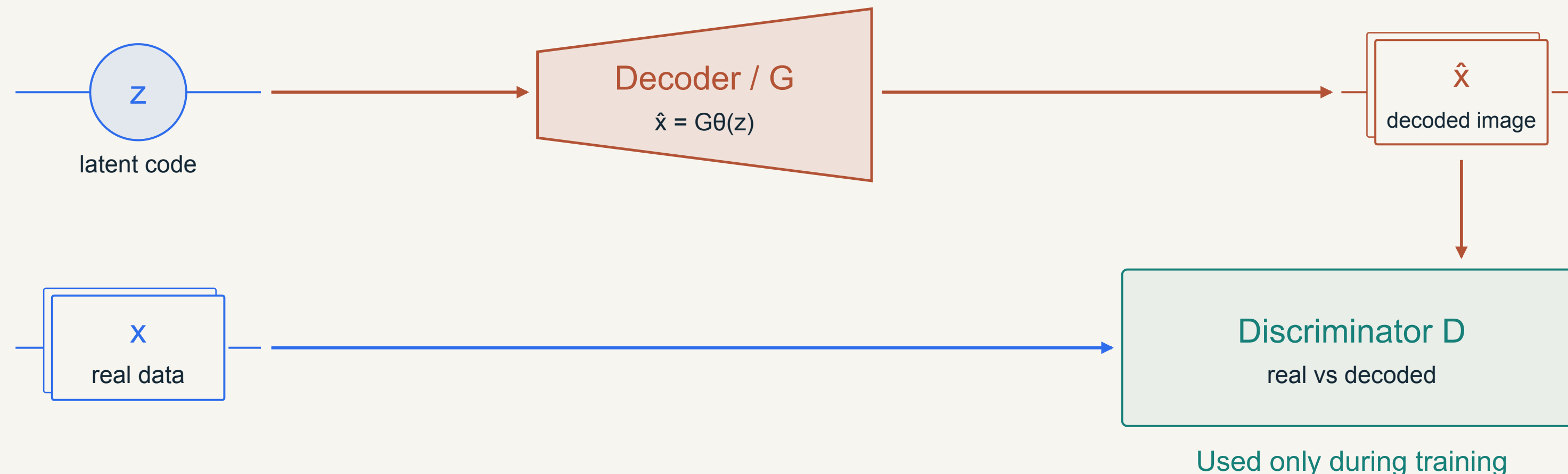


The encoder specifies $q\phi(z | x)$; sampling produces a code.

Prior $p(z)$ + decoder define the observation generative model.

A temporal transition is an additional modeling choice.

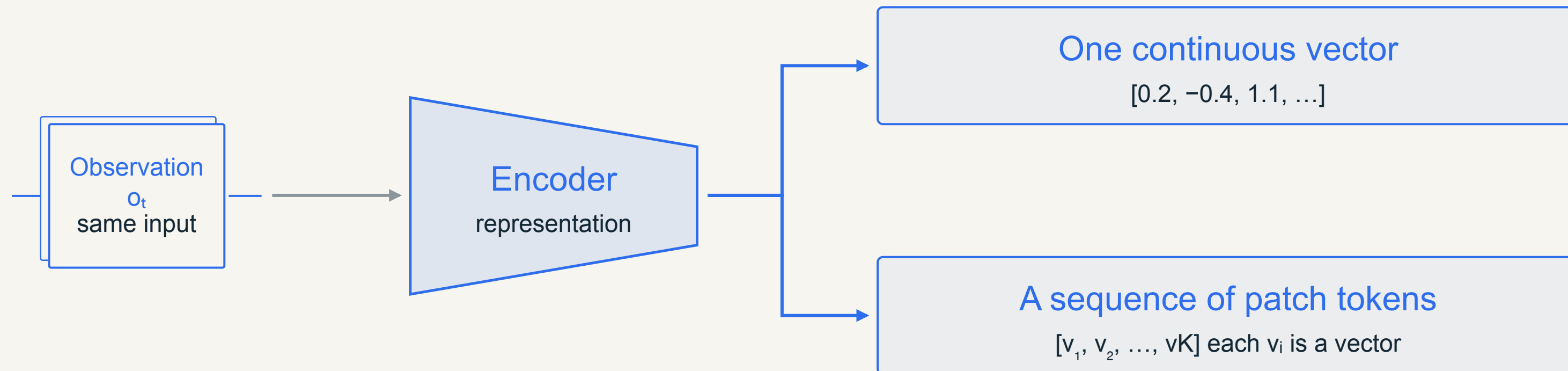
Decoders can use adversarial training



Combine reconstruction / perceptual losses with adversarial feedback from D .

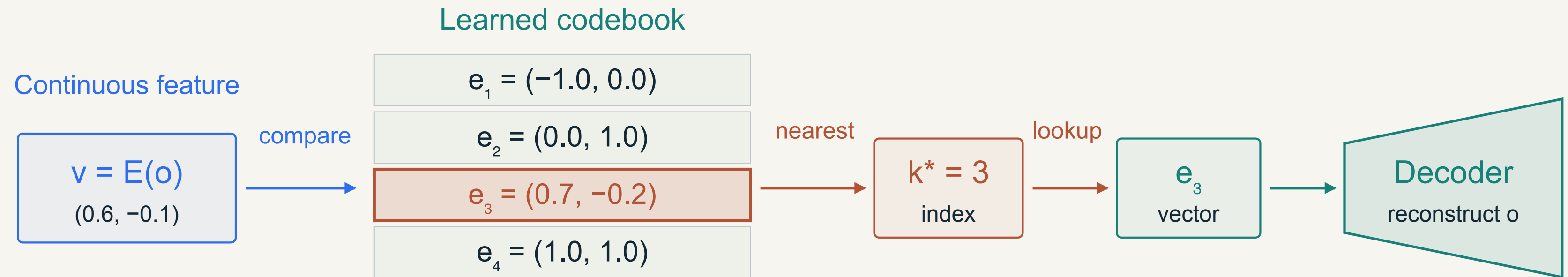
Example: VQGAN learns a latent autoencoder with an adversarially trained decoder.

Continuous features and tokens



A token is a unit of representation. It can be a continuous vector.

From continuous features to discrete codes



Repeat for each feature position to obtain a sequence or grid of code indices.

$$k^* = \arg \min_{k \in \{1, \dots, V\}} \|v - e_k\|_2^2, \quad z_q = e_{k^*}$$

V codebook entries. The encoder, codebook, and decoder learn from reconstruction and quantization objectives. Toy values shown.

Latent type and encoder randomness

	Continuous output	Discrete output
Deterministic encoding	$z = E(o)$	Nearest codebook index
Stochastic encoding	$z \sim q_{\phi}(z \mid o)$; distribution over vectors	Sample a categorical code

The encoder distribution and the future-prediction distribution are separate choices.

LGSSM: world model and belief update

$$s_{t+1} = As_t + Ba_t + \epsilon_t^s, \quad o_t = Cs_t + \epsilon_t^o$$

$\epsilon_t^s \sim \mathcal{N}(0, Q)$, $\epsilon_t^o \sim \mathcal{N}(0, R)$. Gaussian initial state; independent noise; model parameters fixed.

Current belief $p(s_t \mid H_t) = \mathcal{N}(\mu_{t|t}, P_{t|t})$

1. World-model prediction

$$p(s_{t+1} \mid H_t, a_t) = \mathcal{N}(\mu^-, P^-)$$

$$\mu^- = A\mu_{t|t} + Ba_t$$

$$P^- = AP_{t|t}A^\top + Q$$

$$S = CP^-C^\top + R$$

$$p(o_{t+1} \mid H_t, a_t) = \mathcal{N}(C\mu^-, S)$$

2. Belief update after o_{t+1}

Use the new observation to correct the prediction.

$$\nu = o_{t+1} - C\mu^-$$

$$K = P^-C^\top S^{-1}$$

$$\mu_{t+1|t+1} = \mu^- + K\nu$$

$$P_{t+1|t+1} = P^- - KSK^\top$$

Keep the updated mean and covariance, then repeat for the next action and observation.

$H_t = (o_{0:t}, a_{0:t-1})$. This is the observation/action history called h_t in L4.

General latent world models: two challenges

1. World model: represent and predict the environment

$$s_{t+1} \sim p_{\theta}(\cdot \mid s_t, a_t), \quad o_{t+1} \sim p_{\theta}(\cdot \mid s_{t+1})$$

Nonlinear dynamics and image observations require choices about the state, representation, and predictive distributions.

2. Belief update: infer state from observation history

$$p_{\theta}(s_{t+1} \mid H_t, a_t) = \int p_{\theta}(s_{t+1} \mid s_t, a_t) p_{\theta}(s_t \mid H_t) ds_t$$

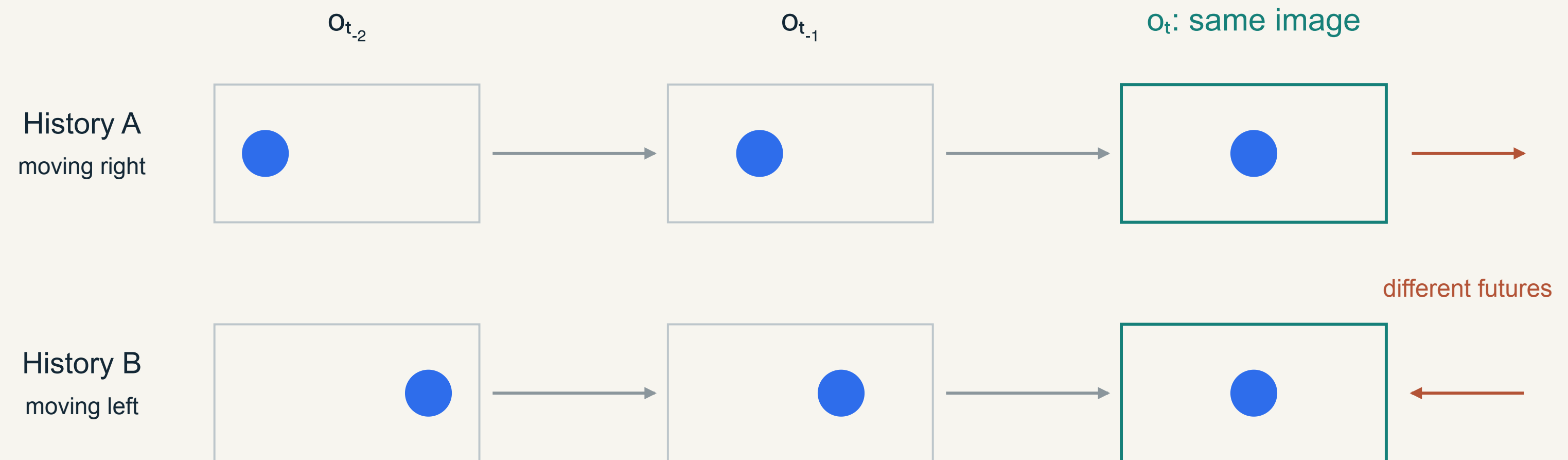
$$p_{\theta}(s_{t+1} \mid H_{t+1}) \propto p_{\theta}(o_{t+1} \mid s_{t+1}) p_{\theta}(s_{t+1} \mid H_t, a_t)$$

The integrals and posterior may have no closed form. A mean and covariance may no longer describe the belief.

We must design both the latent world model and how observation history informs predictions.

Both LGSSMs and general models can have learned parameters. Here, H_t is history; h_t will denote recurrent memory.

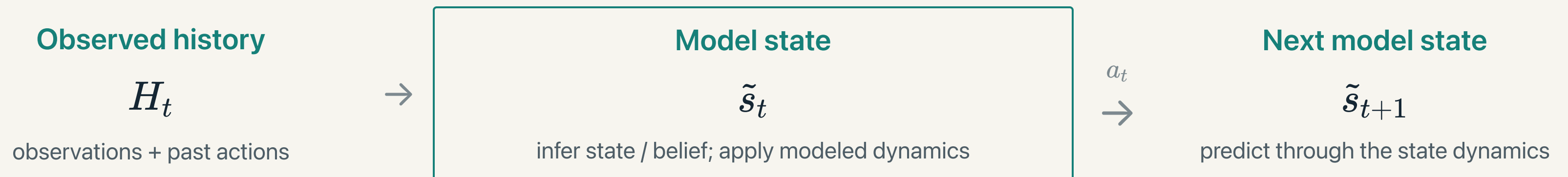
The same frame can hide different motion



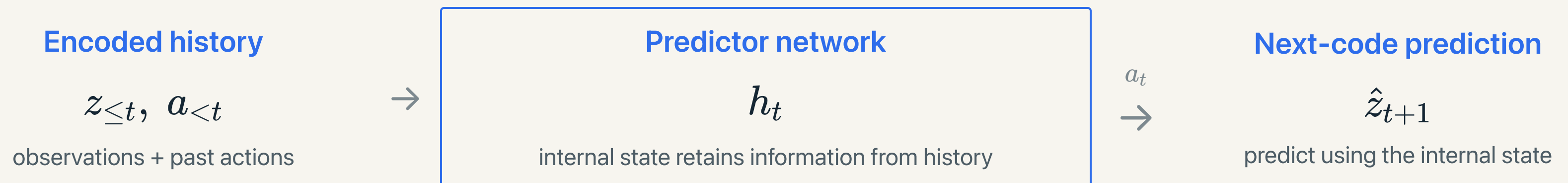
Identical frame inputs cannot distinguish these two motion histories.

State and history: two modeling views

Describe state evolution



Condition predictions on retained history



Both retain information from history. These views can overlap in the same learned model.

From one-step prediction to multi-step rollout

Explicit model state: propagate the state

Inferred state

 $\tilde{s}_t$ a_t 

Predicted state

 $\tilde{s}_{t+1}$ a_{t+1} 

Predicted state

 $\tilde{s}_{t+2}$

History-conditioned prediction: retain earlier codes

Observed prefix

 $\dots, z_t$ a_t 

One predicted step

 $\dots, z_t, \hat{z}_{t+1}$ a_{t+1} 

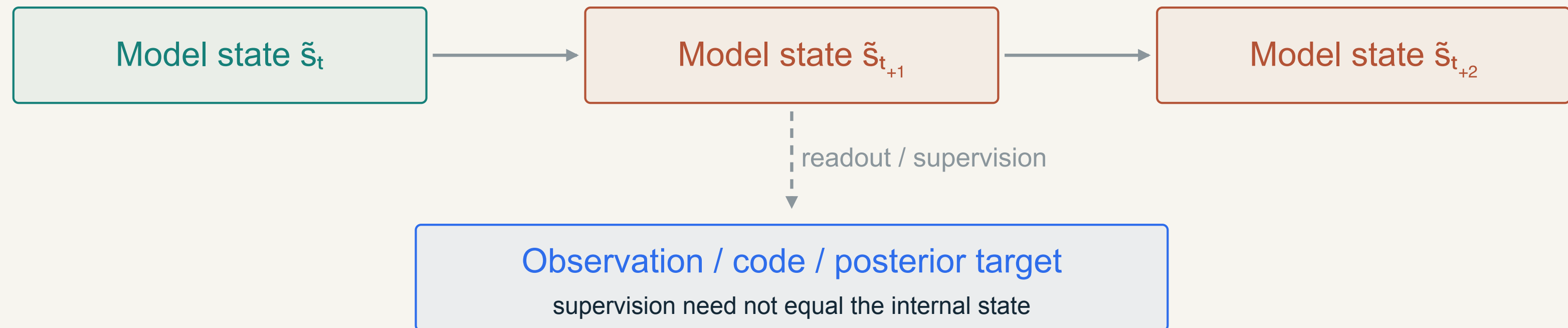
Two predicted steps

 $\dots, z_t, \hat{z}_{t+1}, \hat{z}_{t+2}$

Actions are supplied at each step; earlier actions also remain in the history. A hat marks a generated code.

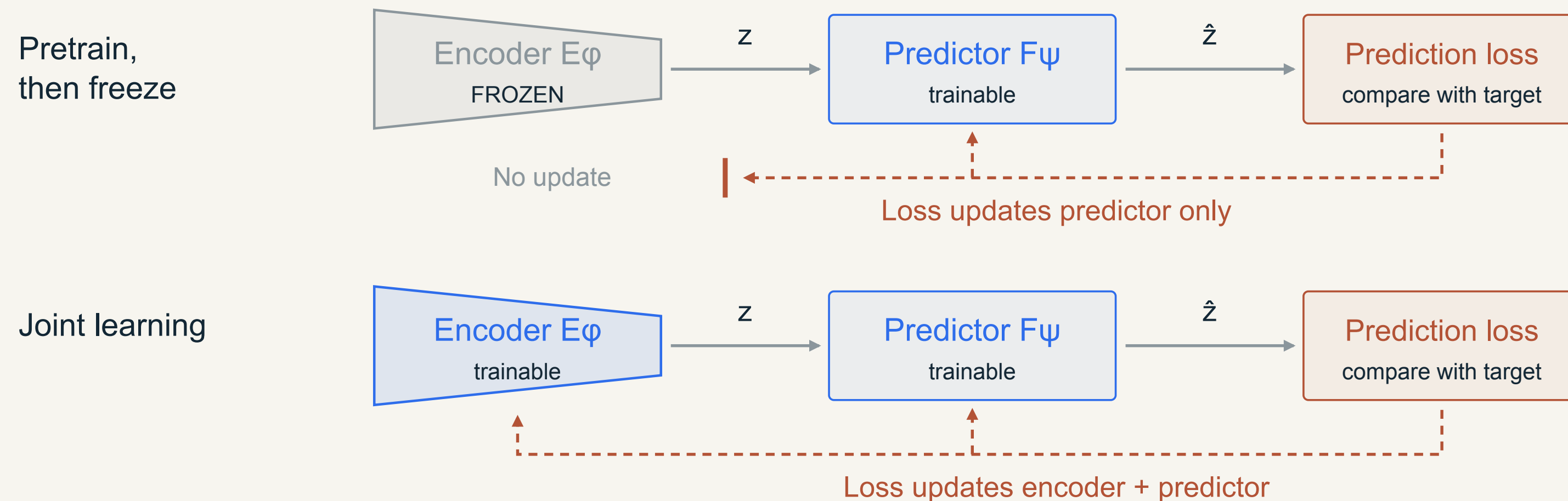
At each step, use the model's output to continue predicting the future.

Rollout variables and training targets



What the model propagates and what supervises it need not be the same variable.

Learning representations with dynamics



Solid arrows: forward computation. Dashed orange arrows: prediction-loss gradients.

A constant representation is easy to predict. What prevents that solution?

Autoregressive trajectory prediction

Start from the observed seed $\hat{z}_0 = z_0$. Each prediction uses the earlier outputs and supplied actions.

Deterministic prediction

$$\hat{z}_{t+1} = f_{\theta}(\hat{z}_{0:t}, a_{0:t})$$

Directly compute the next code. No distribution or sampling is required.

Probabilistic prediction

$$\hat{z}_{t+1} \sim p_{\theta}(\cdot \mid \hat{z}_{0:t}, a_{0:t})$$

Sample the next code from a predicted conditional distribution.

In the probabilistic case, the trajectory distribution factorizes:

$$p_{\theta}(z_{1:T} \mid z_0; a_{0:T-1}) = \prod_{t=0}^{T-1} p_{\theta}(z_{t+1} \mid z_{0:t}, a_{0:t})$$

Autoregressive: reuse earlier outputs to predict later ones. The prediction may be deterministic or stochastic.

Teacher forcing vs. autoregressive rollout

At time t , both use supplied actions. Rollout history starts from observations, then includes model predictions.

Teacher forcing · training

Input: recorded history

$$z_{\leq t}$$



Predict the next code

$$\hat{z}_{t+1} = f_{\theta}(z_{\leq t}, a_{\leq t})$$



Compare with recorded z_{t+1} to compute the training loss.

Next step receives recorded z_{t+1} .

Autoregressive rollout · inference

Input: observed prefix + predicted continuation

$$\hat{z}_{\leq t}$$



Predict the next code

$$\hat{z}_{t+1} = f_{\theta}(\hat{z}_{\leq t}, a_{\leq t})$$



No recorded future z_{t+1} is supplied during rollout.

Next step receives generated $\hat{z}_{t+1}$.

Teacher forcing feeds recorded history. Rollout feeds generated history; future observations are unavailable.

Discrete prediction: a distribution over codes

4	11	3	2
5	7	9	1
8	6	0	10

For the next code

$$\pi = \text{softmax}(f_{\theta}(\text{context}))$$

$$\mathcal{L} = -\log \pi_{k^*}$$

Sample · stochastic

$$\hat{k} \sim \text{Categorical}(\pi)$$

Argmax · deterministic

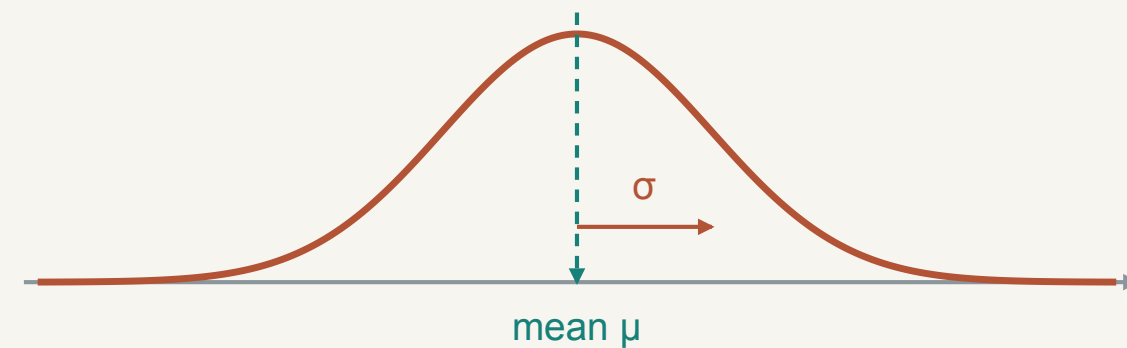
$$\hat{k} = \arg \max_k \pi_k$$

Prediction can be stochastic or deterministic.

Continuous prediction: a Gaussian distribution

Let c be the available code history and actions; $y = z_{t+1}$ is the next continuous code.

One continuous coordinate



Predict location and spread from context.

For the next continuous code

$$p_{\theta}(y \mid c) = \mathcal{N}(y; \mu_{\theta}(c), \Sigma_{\theta}(c))$$

$$\mathcal{L} = -\log p_{\theta}(y^* \mid c)$$

Sample · stochastic

$$\hat{y} \sim \mathcal{N}(\mu_{\theta}(c), \Sigma_{\theta}(c))$$

Mean · deterministic

$$\hat{y} = \mu_{\theta}(c)$$

Prediction can be stochastic or deterministic.

When Gaussian NLL becomes MSE

$$p_{\theta}(y \mid c) = \mathcal{N}(y; \mu_{\theta}(c), \sigma^2 I)$$

$$-\log p_{\theta}(y \mid c) = \frac{\|y - \mu_{\theta}(c)\|^2}{2\sigma^2} + C$$

Fixed variance

Learned variance

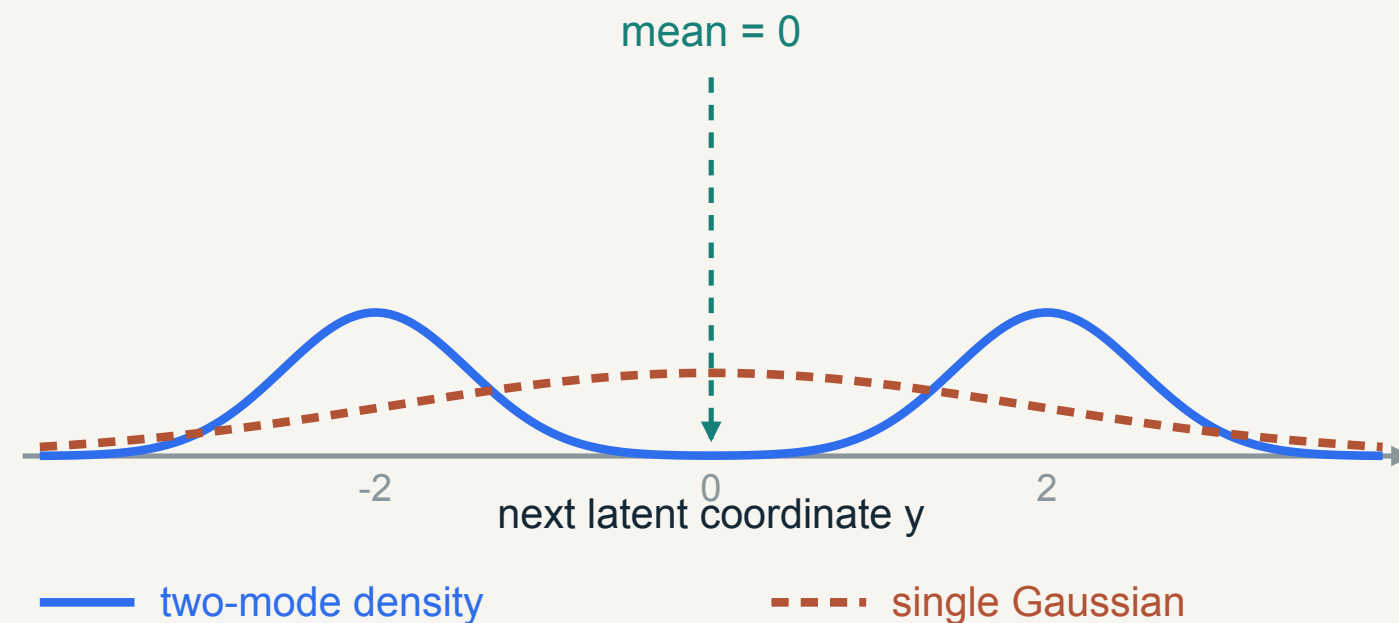
Optimize the mean: scaled MSE

Residual weighting and log-scale terms also matter

$$\mu^*(c) = \mathbb{E}[y \mid c]$$

Squared error targets the conditional mean. That mean need not be a typical future.

Mixtures represent distinct possible futures



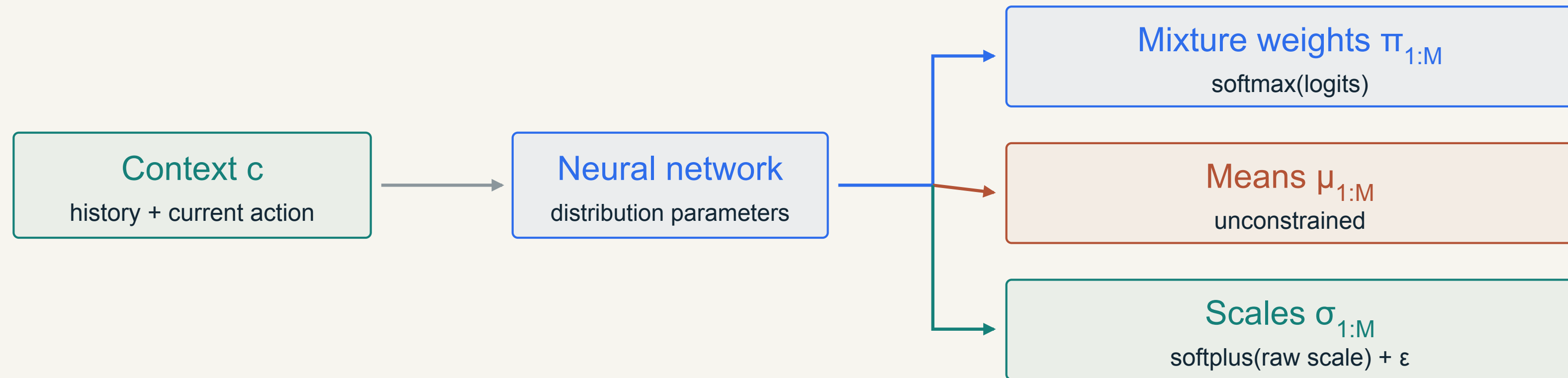
A mixture of M conditional densities

Each component describes one region of possible outcomes.

Weights determine how likely those regions are.

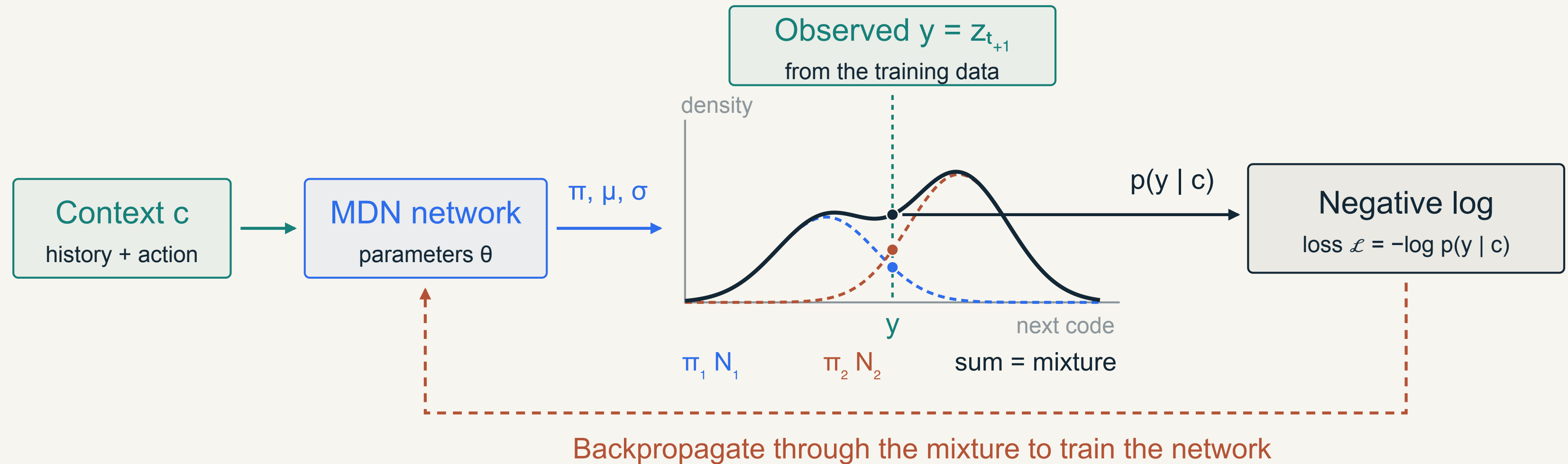
$$p(y \mid c) = \sum_{k=1}^M \pi_k(c) \mathcal{N}(y; \mu_k(c), \Sigma_k(c))$$

MDN: what the network outputs



A mixture density network predicts parameters; it does not output one chosen future.

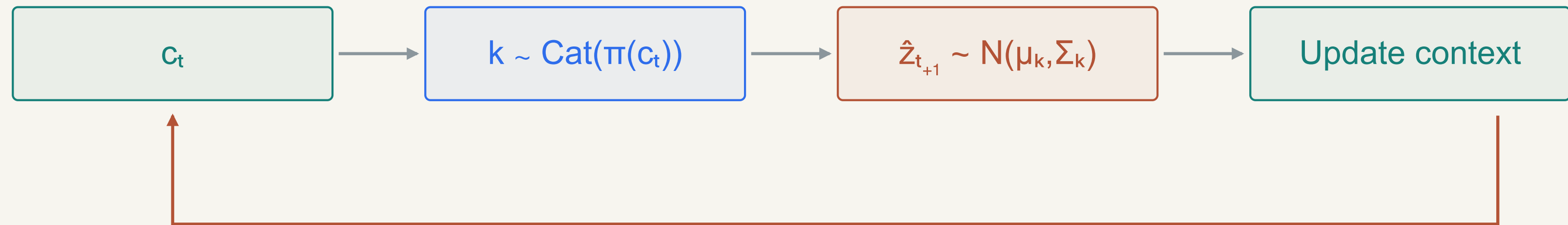
MDN: how the loss is computed



$$\mathcal{L}(c, y) = -\log \left[\sum_{k=1}^M \pi_k(c) \mathcal{N}(y; \mu_k(c), \sigma_k^2(c)) \right]$$

Evaluate every component at the observed y . Training requires no component sampling.

MDN: sampling and rolling forward

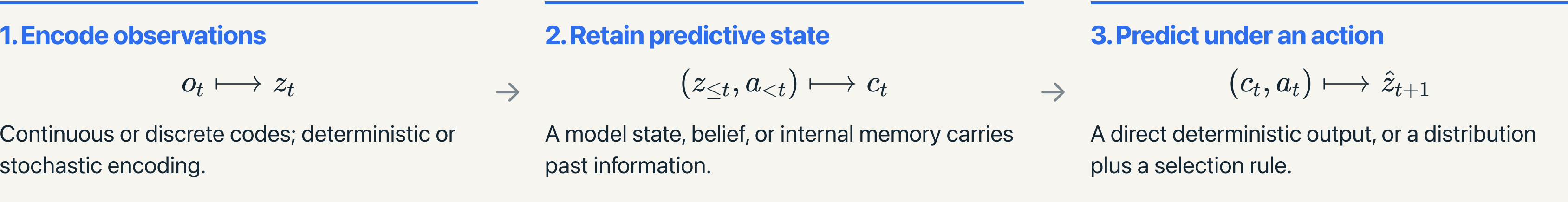


Repeat with the next supplied action.

Richer alternatives: diffusion and flow models — later lectures.

Latent world models: what we have built

A latent world model predicts how an environment evolves under supplied actions, using a learned internal representation.



Train on recorded trajectories

Compare predictions with observed codes or observations. Learn the representation and dynamics jointly or in stages.

Roll forward

Reuse the predicted state / code for the next action. Decode to observations when the model includes a decoder.

The representation, retained state, prediction rule, and training objective are the main design choices.

TD-MPC

Temporal Difference Learning for Model Predictive Control

Nicklas Hansen · Xiaolong Wang · Hao Su
2022

arXiv:2203.04955v2 [cs.LG] 19 Jul 2022

Temporal Difference Learning for Model Predictive Control

Nicklas Hansen¹ Xiaolong Wang^{*1} Hao Su^{*1}

Abstract

Data-driven model predictive control has two key advantages over model-free methods: a potential for improved sample efficiency through model learning, and better performance as computational budget for planning increases. However, it is both costly to plan over long horizons and challenging to obtain an accurate model of the environment. In this work, we combine the strengths of model-free and model-based methods. We use a learned task-oriented latent dynamics model for local trajectory optimization over a short horizon, and use a learned terminal value function to estimate long-term return, both of which are learned jointly by temporal difference learning. Our method, **TD-MPC**, achieves superior sample efficiency and asymptotic performance over prior work on both state and image-based continuous control tasks from DMControl and MetaWorld. Code and videos are available at <https://nicklashansen.github.io/td-mpc>.

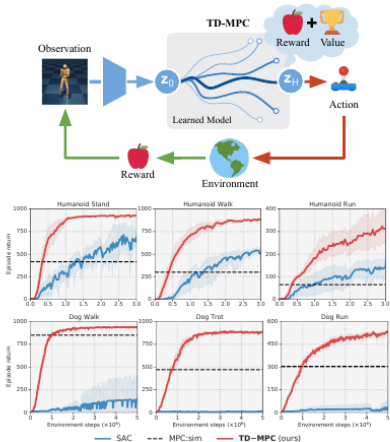


Figure 1. Overview. (Top) We present a framework for MPC using a task-oriented latent dynamics model and value function learned jointly by temporal difference learning. We perform trajectory optimization over model rollouts and use the value function for long-term return estimates. (Bottom) Episode return of our method, SAC, and MPC with a ground-truth simulator on challenging, high-dimensional Humanoid and Dog tasks (Tassa et al., 2018). Mean of 5 runs; shaded areas are 95% confidence intervals.

1. Introduction

To achieve desired behavior in an environment, a Reinforcement Learning (RL) agent needs to iteratively interact and consolidate knowledge about the environment. Planning is a powerful approach to such sequential decision making problems, and has achieved tremendous success in application areas such as game-playing (Kaiser et al., 2020; Schrittwieser et al., 2020) and continuous control (Tassa et al., 2012; Chua et al., 2018; Janner et al., 2019). By utilizing an internal model of the environment, an agent can plan a trajectory of actions ahead of time that leads to the desired behavior; this is in contrast to *model-free* algorithms that learn a policy purely through trial-and-error.

Concretely, prior work on model-based methods can largely be subdivided into two directions, each exploiting key ad-

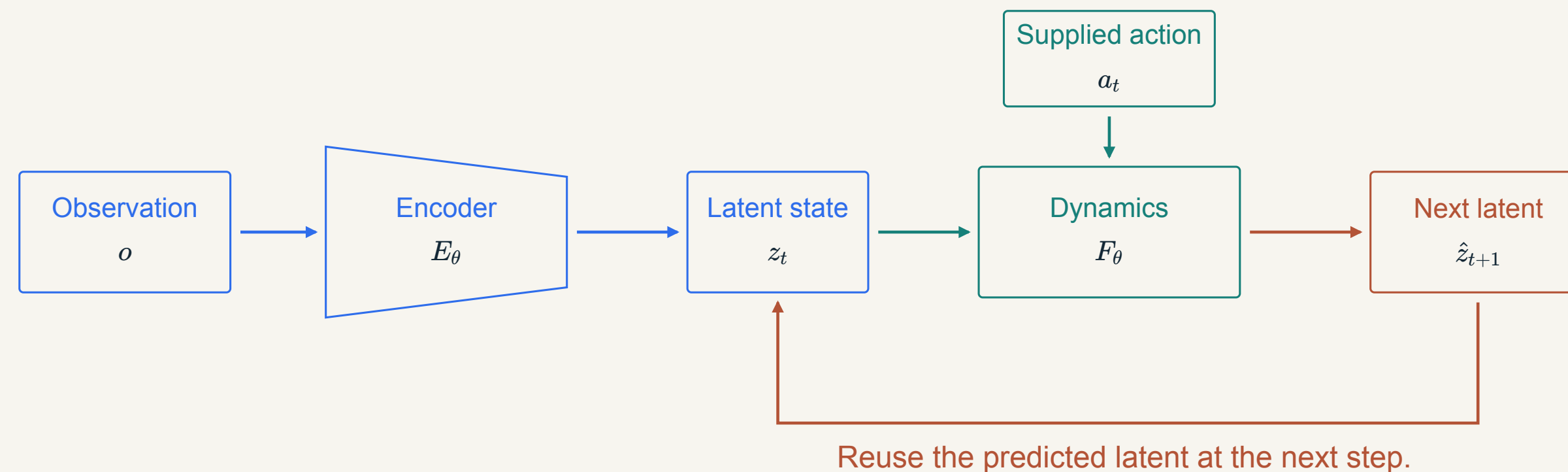
vantages of model-based learning: (i) planning, which is advantageous over a learned policy, but it can be prohibitively expensive to plan over long horizons (Janner et al., 2019; Lowrey et al., 2019; Hafner et al., 2019; Argenson & Dulac-Arnold, 2021); and (ii) using a learned model to improve sample-efficiency of model-free methods by e.g. learning from generated rollouts, but this makes model biases likely to propagate to the policy as well (Ha & Schmidhuber, 2018; Hafner et al., 2020b; Clavera et al., 2020). As a result, model-based methods have historically struggled to outperform simpler, model-free methods (Srinivas et al., 2020; Kostrikov et al., 2020) in continuous control tasks.

Can we instead augment model-based planning with the strengths of model-free learning? Because of the immense cost of long-horizon planning, Model Predictive Control

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

^{*}Equal contribution ¹UC San Diego. Correspondence to: Nicklas Hansen <nihansen@ucsd.edu>.

TD-MPC: latent dynamics



$$z = E_\theta(o), \quad \hat{z}_{t+1} = F_\theta(z_t, a_t)$$

The observation input may include recent frames. Rollout reuses the predicted latent.

Encoder

Deterministic: $z = E_\theta(o)$

Prediction

Deterministic autoregressive latent prediction.

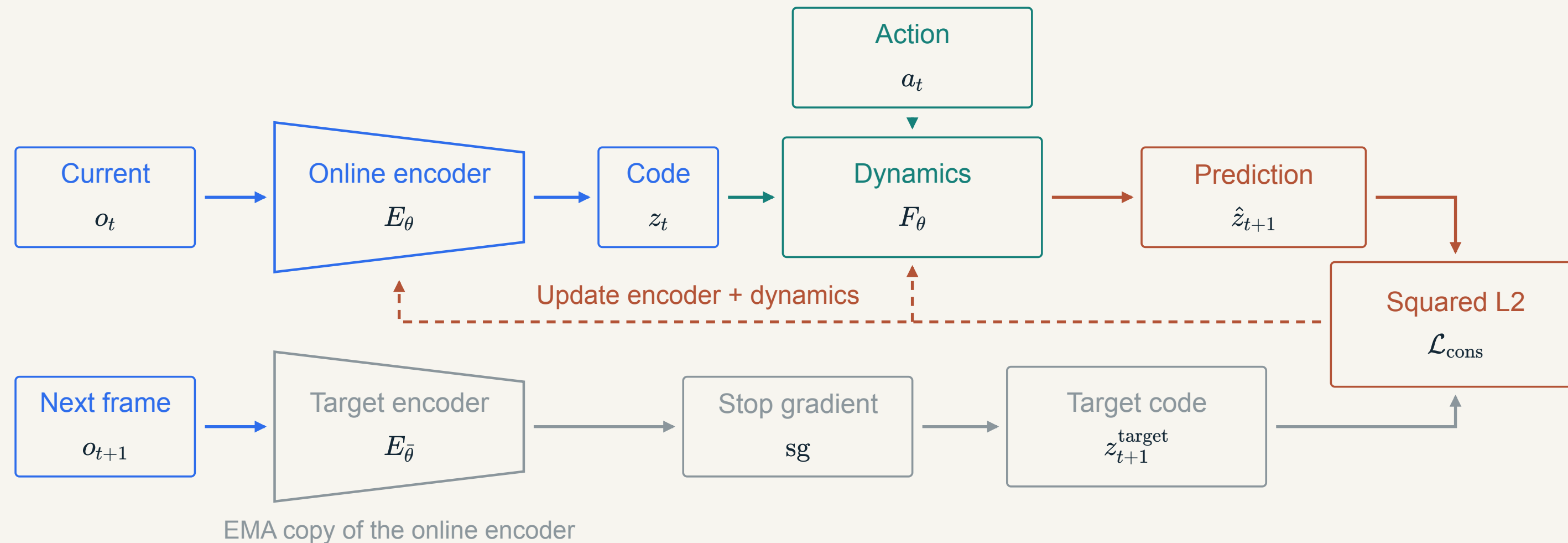
Loss

Squared L2 latent consistency.

Full method

Reward / value losses also matter; L2 consistency alone permits collapse.

TD-MPC: L2 consistency training



$$\mathcal{L}_{\text{cons}} = \|F_\theta(E_\theta(o_t), a_t) - \text{sg}[E_{\bar{\theta}}(o_{t+1})]\|_2^2$$

The target encoder follows the online encoder by EMA: $\bar{\theta} \leftarrow (1 - \tau)\bar{\theta} + \tau\theta$.

TD-MPC: learning the transition

Predicted next latent

$$\hat{z}_{t+1} = F_{\theta}(z_t, a_t)$$

Target from the next observation

$$z_{t+1}^{\text{target}} = \text{sg}[E_{\bar{\theta}}(o_{t+1})]$$

$$\mathcal{L}_{\text{cons}} = \|\hat{z}_{t+1} - z_{t+1}^{\text{target}}\|_2^2$$

sg stops target gradients; the target encoder is updated by an exponential moving average.

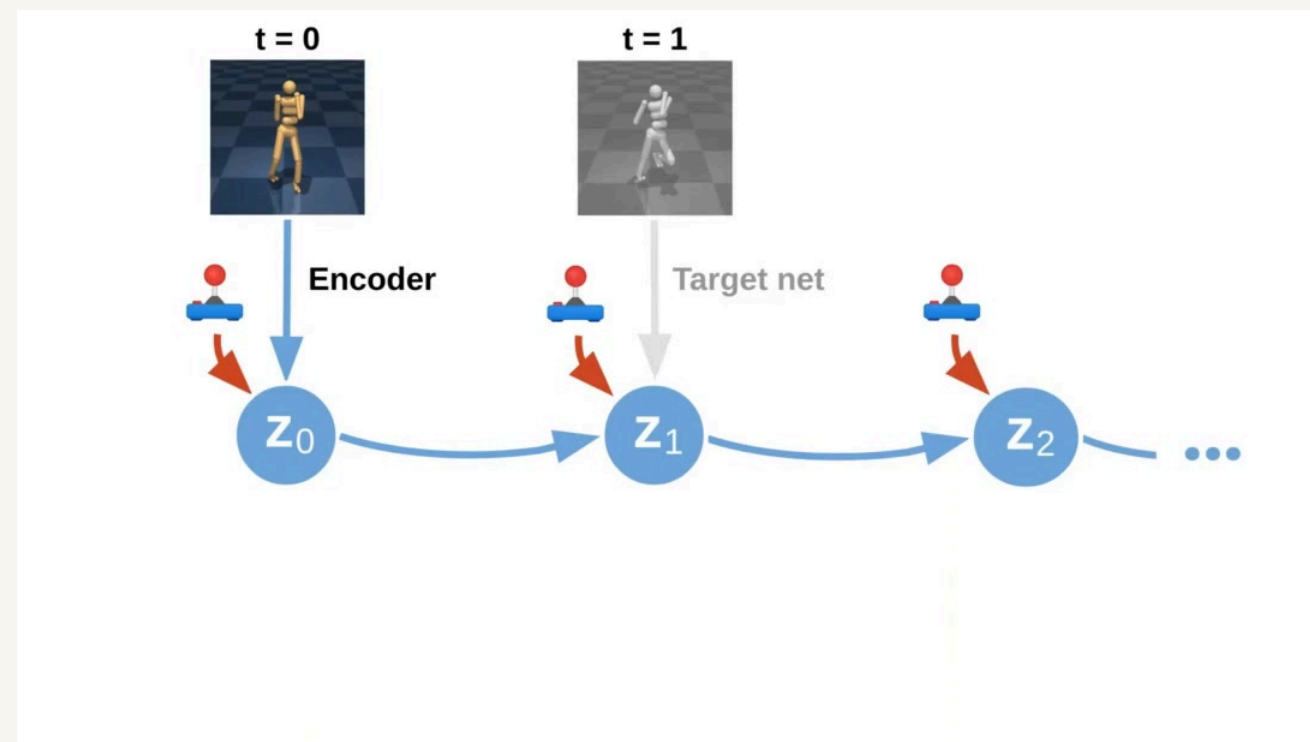
L2 consistency alone permits collapsed codes.

$$E(o) \equiv c, \quad F(c, a) \equiv c \quad \Rightarrow \quad \mathcal{L}_{\text{cons}} = 0$$

The full TD-MPC objective also includes reward prediction and value learning.

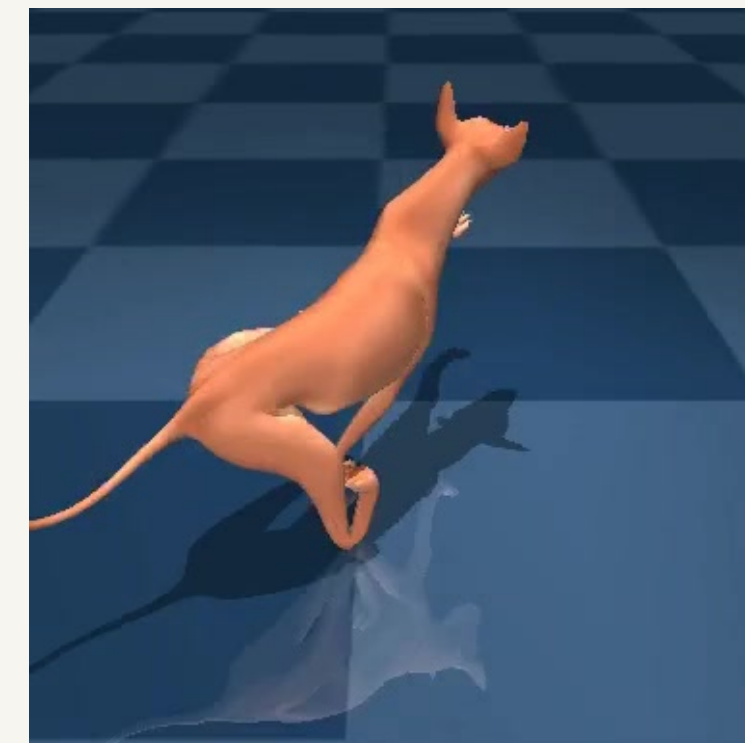
TD-MPC: latent rollouts and control

Learning the latent transition



Original training animation

Using the model for control



Agent execution · Dog Run

Predictions stay in latent space; the task video shows the resulting behavior in the environment.

V-JEPA 2-AC

V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning

Mahmoud Assran et al.

2025

We focus on the action-conditioned world-model stage.

arXiv:2506.09985v1 [cs.AI] 11 Jun 2025

V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning

Mahmoud Assran^{1,*}, Adrien Bardes^{1,*}, David Fan^{1,*}, Quentin Garrido^{1,*}, Russell Howes^{1,*}, Mojtaba Komeili^{1,*}, Matthew Muckley^{1,*}, Ammar Rizvi^{1,*}, Claire Roberts^{1,*}, Koustuv Sinha^{1,*}, Artem Zholus^{1,2,*}, Sergio Arnaud^{1,*}, Abha Gejj^{1,*}, Ada Martin^{1,*}, François Robert Hogan^{1,*}, Daniel Dugas^{1,*}, Piotr Bojanowski¹, Vasil Khalidov¹, Patrick Labatut¹, Francisco Massa¹, Marc Szafraniec¹, Kapil Krishnakumar¹, Yong Li¹, Xiaodong Ma¹, Sarath Chandar², Franziska Meier^{1,*}, Yann LeCun^{1,*}, Michael Rabbat^{1,*}, Nicolas Ballas^{1,*}

¹FAIR at Meta, ²Mila – Quebec AI Institute and Polytechnique Montréal
*Core Team

A major challenge for modern AI is to learn to understand the world and learn to act largely by observation (LeCun, 2022). This paper explores a self-supervised approach that combines internet-scale video data with a small amount of interaction data (robot trajectories), to develop models capable of understanding, predicting, and planning in the physical world. We first pre-train an action-free joint-embedding-predictive architecture, V-JEPA 2, on a video and image dataset comprising over 1 million hours of internet video. V-JEPA 2 achieves strong performance on motion understanding (77.3 top-1 accuracy on Something-Something v2) and state-of-the-art performance on human action anticipation (39.7 recall-at-5 on Epic-Kitchens-100) surpassing previous task-specific models. Additionally, after aligning V-JEPA 2 with a large language model, we demonstrate state-of-the-art performance on multiple video question-answering tasks at the 8 billion parameter scale (e.g., 84.0 on PerceptionTest, 76.9 on TempCompass). Finally, we show how self-supervised learning can be applied to robotic planning tasks by post-training a latent action-conditioned world model, V-JEPA 2-AC, using less than 62 hours of unlabeled robot videos from the Droid dataset. We deploy V-JEPA 2-AC zero-shot on Franka arms in two different labs and enable picking and placing of objects using planning with image goals. Notably, this is achieved without collecting any data from the robots in these environments, and without any task-specific training or reward. This work demonstrates how self-supervised learning from web-scale data and a small amount of robot interaction data can yield a world model capable of planning in the physical world.

Date: June 13, 2025
Correspondence: Nicolas Ballas <ballasn@meta.com> and Michael Rabbat <mikerabbat@meta.com>
Code: <https://github.com/facebookresearch/vjepa2>
Blogpost: <https://ai.meta.com/blog/v-jepa-2-world-model-benchmarks>

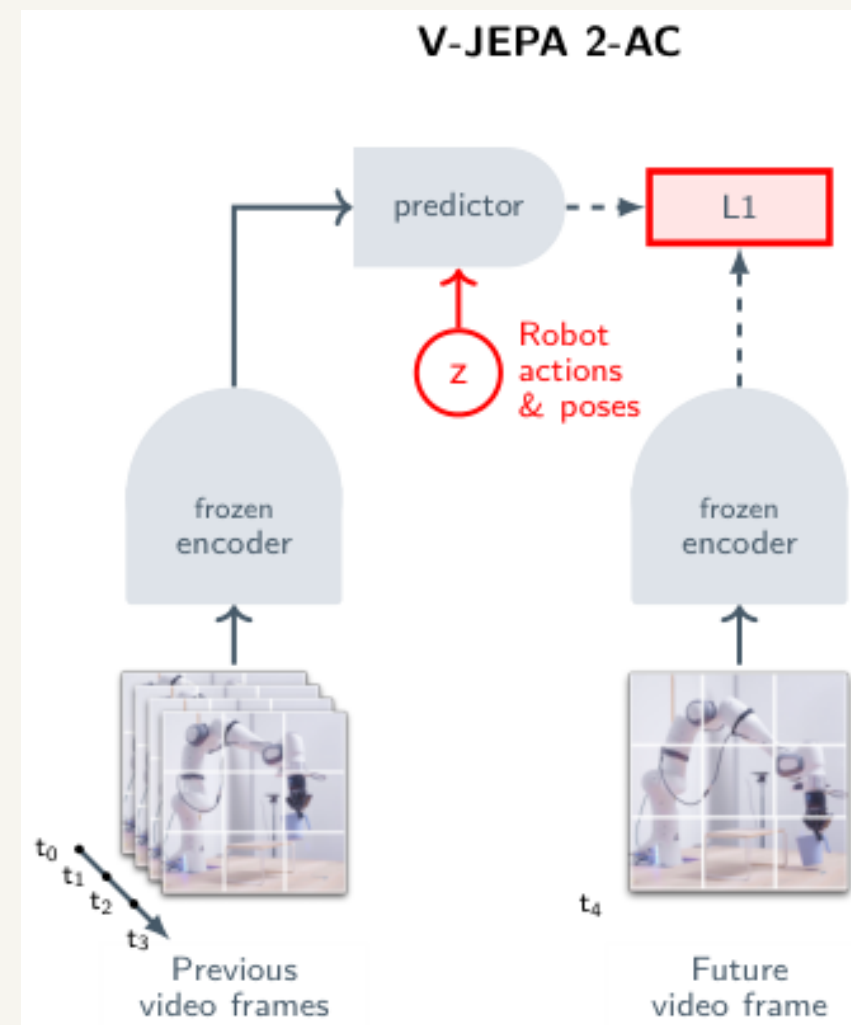
 Meta

1 Introduction

Humans have the ability to adapt and generalize when taking on new tasks and operating in unfamiliar environments. Several cognitive learning theories suggest that humans learn an internal model of the world by integrating low-level sensory inputs to represent and predict future states (Craig, 1967; Rao and Ballard, 1999), and they further posit that this world model shapes our perception at any given moment, playing a crucial role in informing our understanding of reality (Friston, 2010; Clark, 2013; Nortmann et al., 2015). Moreover, our ability to predict the effects of our actions on future states of the world is also essential for goal-oriented planning (Sutton and Barto, 1981, 1998; Ha and Schmidhuber, 2018; Wolpert and Ghahramani, 2000). Building artificial agents that learn a world model from sensory data, such as video, could enable them to *understand* the physical world, *predict* future states, and effectively — like humans — *plan* in new situations, resulting in systems capable of tackling tasks that have not been encountered before.

Previous works have explored the development of predictive world models from interaction data consisting

V-JEPA 2-AC: a pretrained encoder



Original V-JEPA 2-AC training diagram

The predictor also receives robot pose and the supplied actions.

Encoder

Deterministic pretrained features; frozen during AC training.

State

Feature / action / pose history inside the predictor.

Prediction

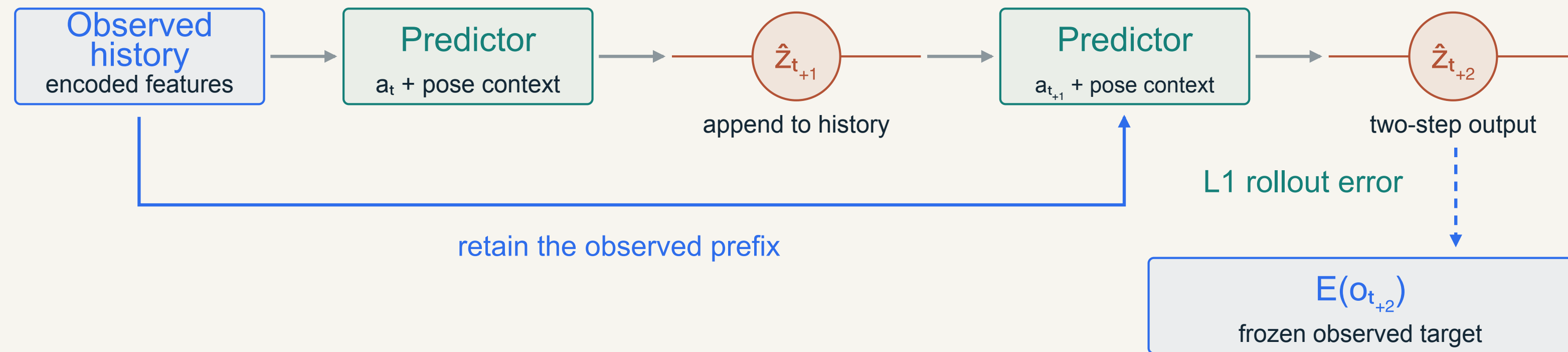
Deterministic autoregressive latent prediction.

Loss

L1 teacher-forced prediction + L1 short rollout loss.

V-JEPA 2-AC: predict and roll out features

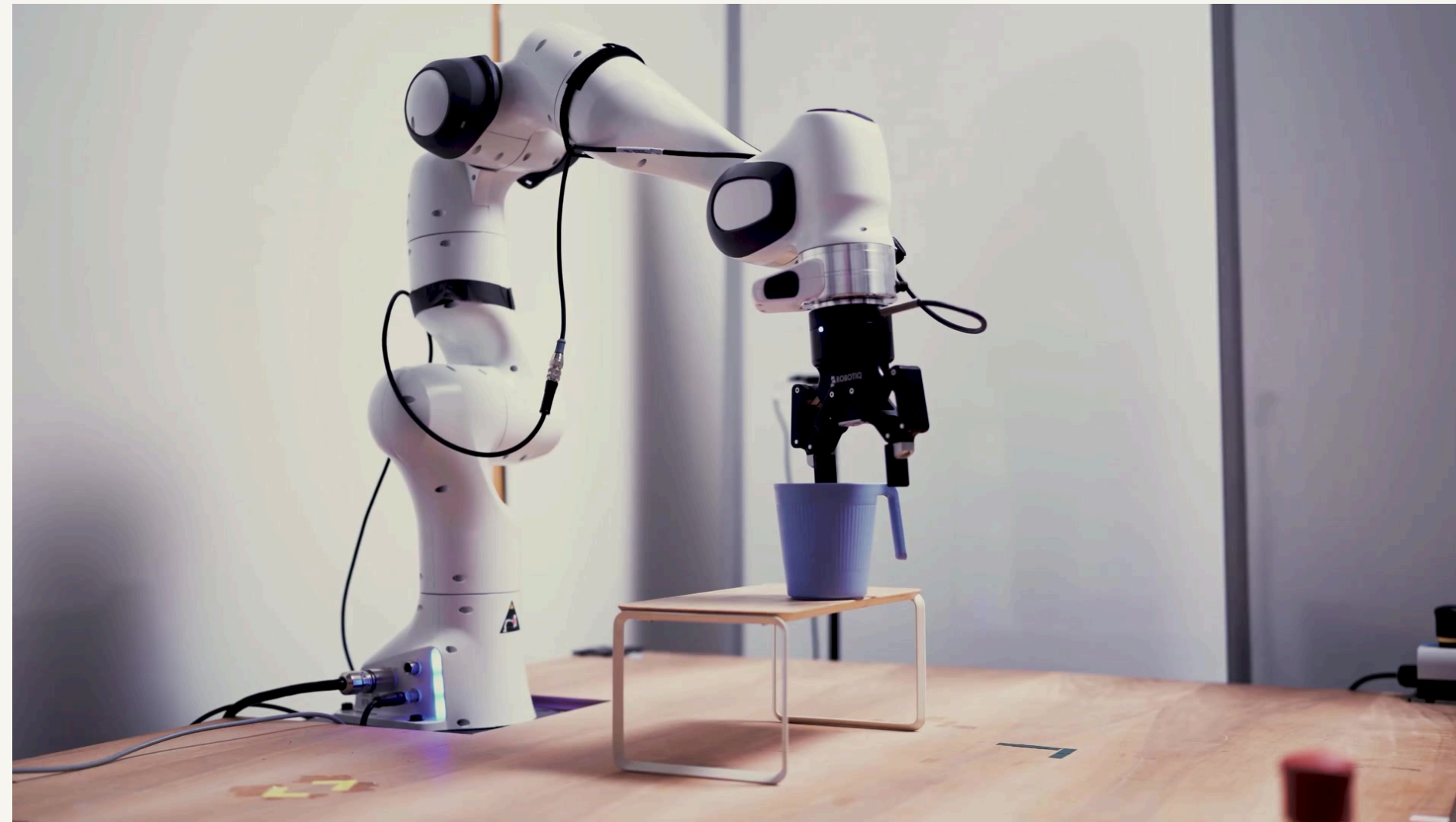
Retain feature history, actions, and robot-pose context at each step



$$\mathcal{L} = \mathcal{L}_{\text{teacher forcing}}^{L_1} + \mathcal{L}_{\text{two-step rollout}}^{L_1}$$

Both terms compare features with L1 error. The observation encoder stays frozen.

V-JEPA 2-AC: robot execution



Robot task execution · original V-JEPA 2 demonstration

The robot uses predictions in feature space to choose actions.

LeWorldModel

LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels

Lucas Maes et al.
2026

arXiv:2603.19312v3 [cs.LG] 3 Jun 2026

LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels

Lucas Maes^{*1} Quentin Le Lidec^{*2} Damien Scieur^{1,3} Yann LeCun² Randall Balestriero⁴
¹Mila & Université de Montréal ²New York University ³Samsung SAIL ⁴Brown University

[Website](#) [Code](#)

Abstract

Joint Embedding Predictive Architectures (JEPAs) offer a compelling framework for learning world models in compact latent spaces, yet existing methods remain fragile, relying on complex multi-term losses, exponential moving averages, pre-trained encoders, or auxiliary supervision to avoid representation collapse. In this work, we introduce LeWorldModel (LeWM), the first JEPA that trains stably end-to-end from raw pixels using only two loss terms: a next-embedding prediction loss and a regularizer enforcing Gaussian-distributed latent embeddings. This reduces tunable loss hyperparameters from six to one compared to the only existing end-to-end alternative. With 15M parameters trainable on a single GPU in a few hours, LeWM plans up to $48\times$ faster than foundation-model-based world models while remaining competitive across diverse 2D and 3D control tasks. Beyond control, we show that LeWM’s latent space encodes meaningful physical structure through probing of physical quantities. Surprise evaluation confirms that the model reliably detects physically implausible events. Code available [here](#).

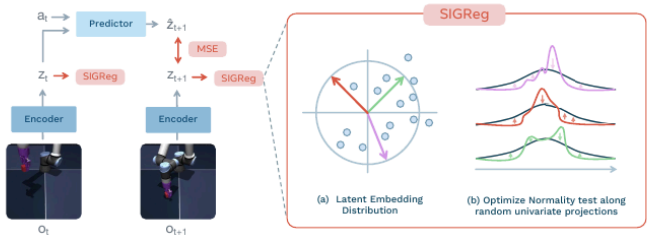
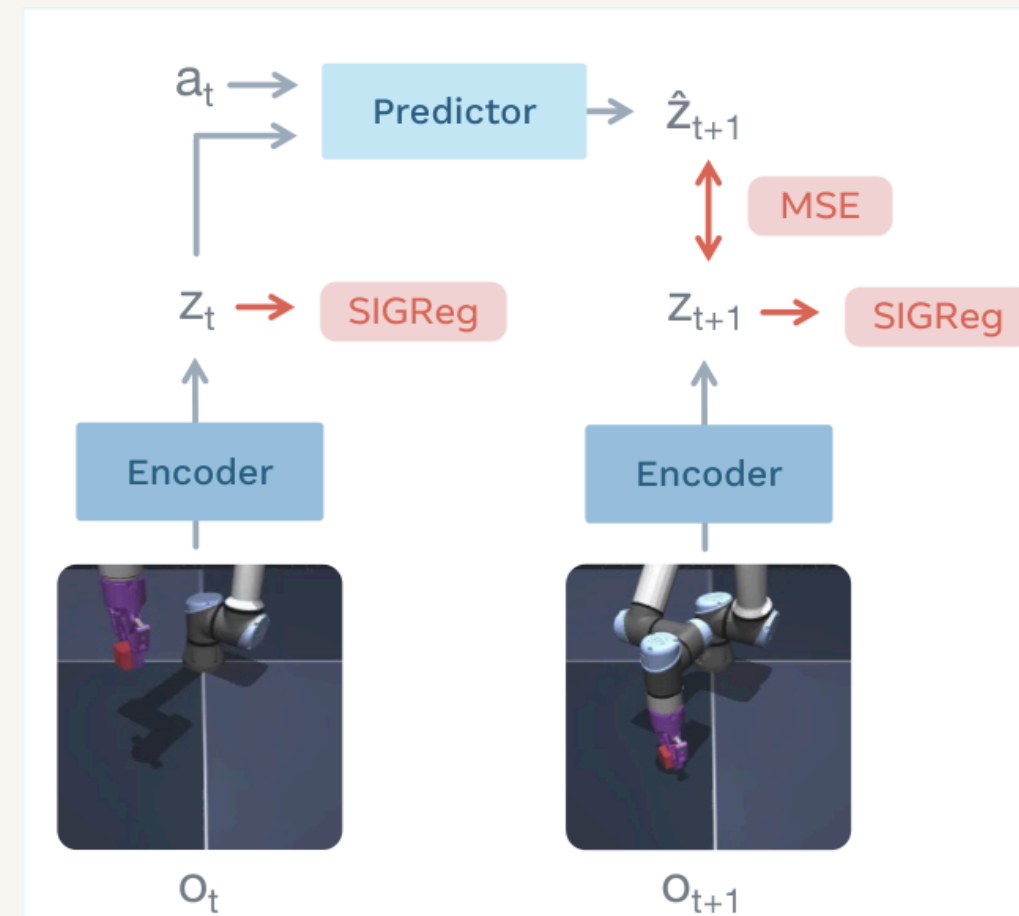


Figure 1: **LeWorldModel Training Pipeline.** Given frame observations $\mathbf{o}_{1:T}$ and actions $\mathbf{a}_{1:T}$, the encoder maps frames into low-dimensional latent representations $\mathbf{z}_{1:T}$. The predictor models the environment dynamics by autoregressively predicting the next latent state $\mathbf{z}_{t+1}$ from the current latent state $\mathbf{z}_t$ and action $\mathbf{a}_t$. The encoder and predictor are jointly optimized using a mean-squared error (MSE) prediction loss. LeWM does not rely on any training heuristics, such as stop-gradient, exponential moving averages, or pre-trained representations. To prevent trivial collapse, the SIGReg regularization term enforces Gaussian-distributed latent embeddings, promoting feature diversity. For tractability, latent embeddings are projected onto multiple random directions, and a normality test is applied to each one-dimensional projection. Aggregating these statistics encourages the full embedding distribution to match an isotropic Gaussian.

^{*} Equal contribution. Correspondence to lucas.maes@mila.quebec

Preprint.

LeWorldModel: joint prediction



Original joint encoder/predictor training pipeline

$$\hat{z}_{t+1} = F_{\psi}(z_{\text{history}}, a_{\text{history}})$$

Encoder

Deterministic continuous features, learned jointly.

State

History retained inside the predictor.

Prediction

Deterministic autoregressive latent prediction.

Loss

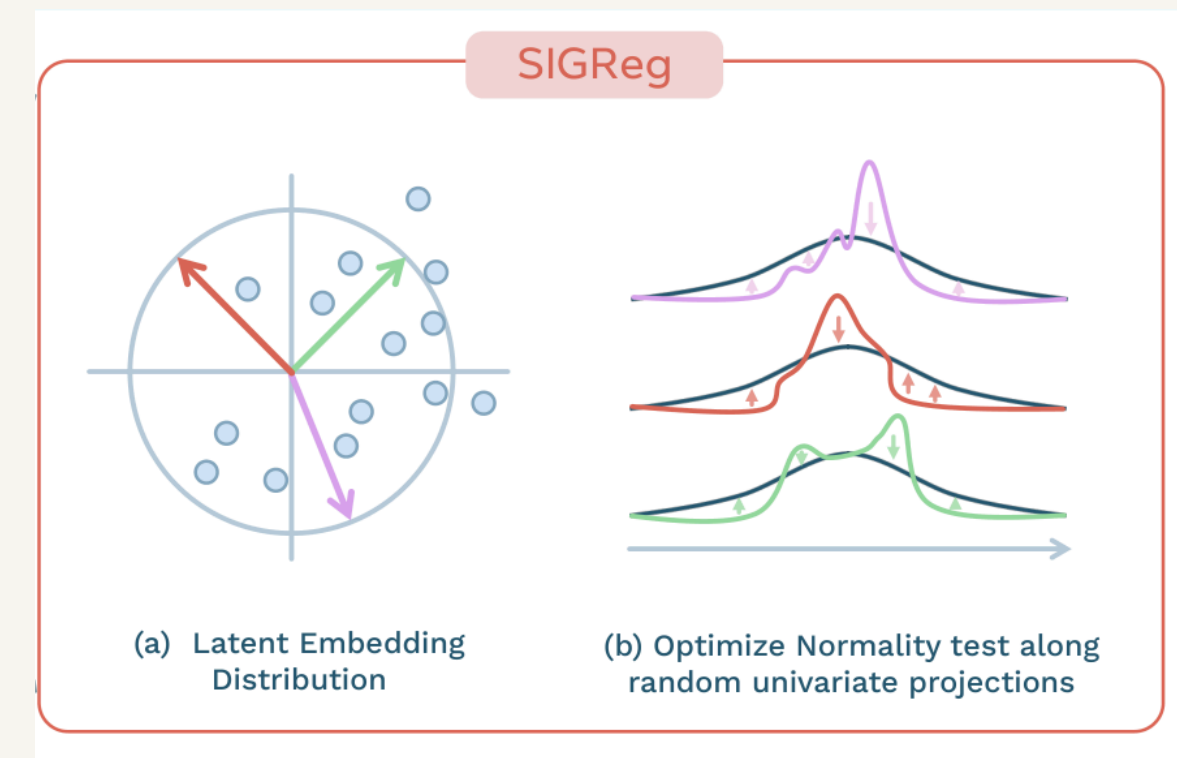
MSE prediction + SIGReg to constrain the representations.

LeWorldModel: what prevents collapse?

Constant codes



Easy to predict, no information



Random projections: encourage each marginal toward $N(0, 1)$

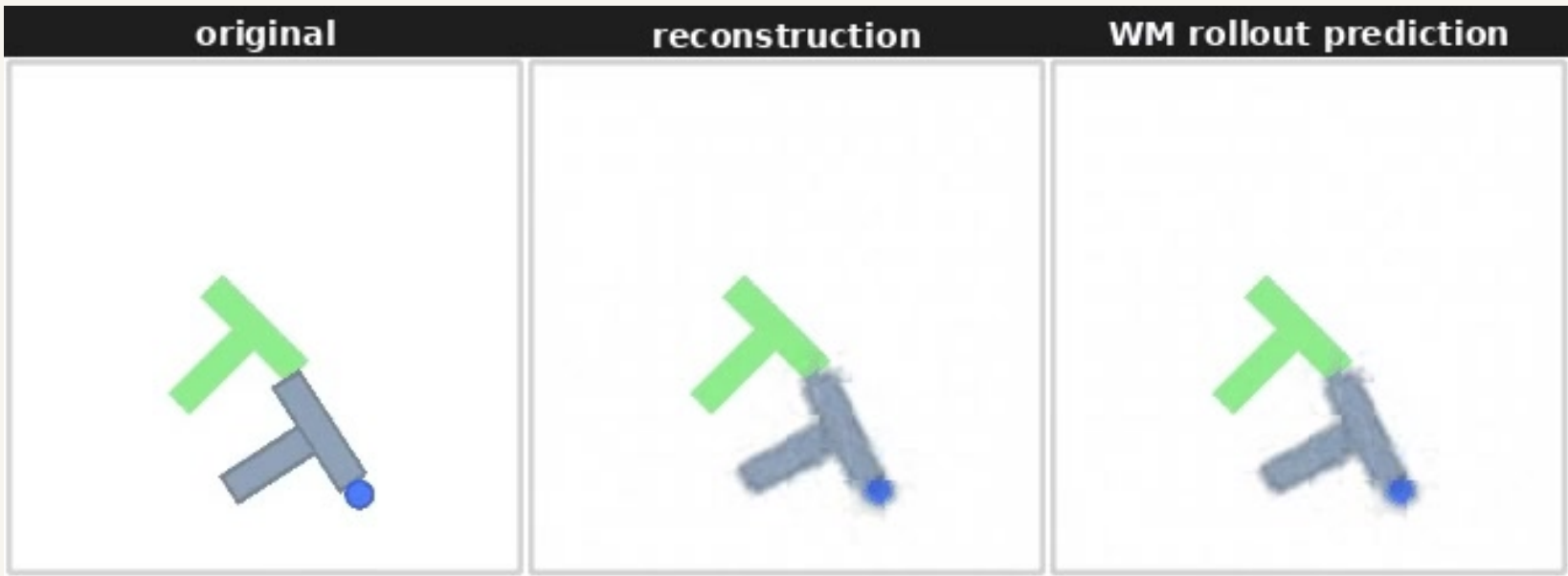
$z_i = \text{constant} \Rightarrow$ small prediction loss, uninformative representation

Across samples, SIGReg encourages embeddings toward $N(0, I)$.

This constraint does not make $p(z_{t+1} | \text{history, action})$ Gaussian.

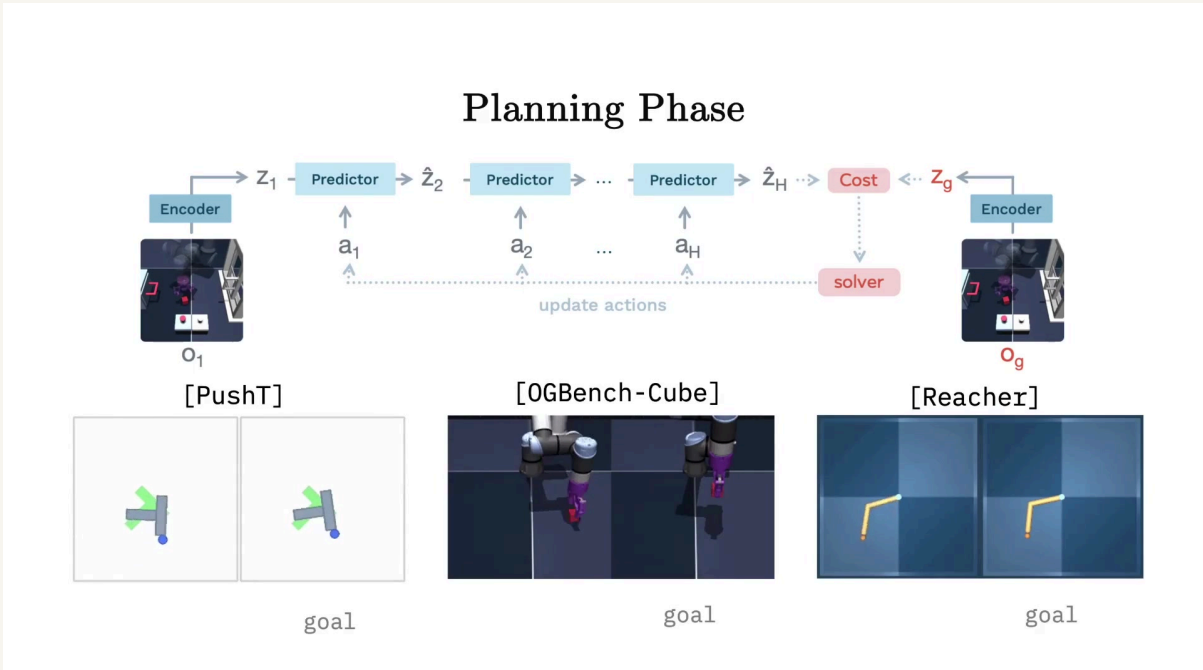
LeWorldModel: prediction and control

Decoded latent prediction



Push-T: observed / reconstructed / predicted

Using predictions for control



Original project video · 53–68 s excerpt

The visualization decoder is separate from world-model training; control uses latent predictions.

World Models (2018)

World Models

David Ha · Jürgen Schmidhuber

2018

arXiv:1803.10122v4 [cs.LG] 9 May 2018

World Models

David Ha¹ · Jürgen Schmidhuber^{2,3}

Abstract

We explore building generative neural network models of popular reinforcement learning environments. Our *world model* can be trained quickly in an unsupervised manner to learn a compressed spatial and temporal representation of the environment. By using features extracted from the world model as inputs to an agent, we can train a very compact and simple policy that can solve the required task. We can even train our agent entirely inside of its own hallucinated dream generated by its world model, and transfer this policy back into the actual environment.

An interactive version of this paper is available at <https://worldmodels.github.io>

1. Introduction

Humans develop a mental model of the world based on what they are able to perceive with their limited senses. The decisions and actions we make are based on this internal model. Jay Wright Forrester, the father of system dynamics, described a mental model as:

The image of the world around us, which we carry in our head, is just a model. Nobody in his head imagines all the world, government or country. He has only selected concepts, and relationships between them, and uses those to represent the real system. (Forrester, 1971)

To handle the vast amount of information that flows through our daily lives, our brain learns an abstract representation of both spatial and temporal aspects of this information. We are able to observe a scene and remember an abstract description thereof (Cheang & Tsao, 2017; Quiroga et al., 2005). Evidence also suggests that what we perceive at any given moment is governed by our brain’s prediction of the future based on our internal model (Nortmann et al., 2015; Gerrit et al., 2013).

One way of understanding the predictive model inside of our brains is that it might not be about just predicting the future in general, but predicting future sensory data given our

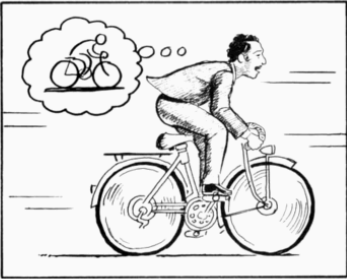


Figure 1. A World Model, from Scott McCloud’s *Understanding Comics*. (McCloud, 1993; E, 2012)

current motor actions (Keller et al., 2012; Leinweber et al., 2017). We are able to instinctively act on this predictive model and perform fast reflexive behaviours when we face danger (Mobbs et al., 2015), without the need to consciously plan out a course of action.

Take baseball for example. A batter has milliseconds to decide how they should swing the bat – shorter than the time it takes for visual signals to reach our brain. The reason we are able to hit a 100 mph fastball is due to our ability to instinctively predict when and where the ball will go. For professional players, this all happens subconsciously. Their muscles reflexively swing the bat at the right time and location in line with their internal models’ predictions (Gerrit et al., 2013). They can quickly act on their predictions of the future without the need to consciously roll out possible future scenarios to form a plan (Hirshon, 2013).

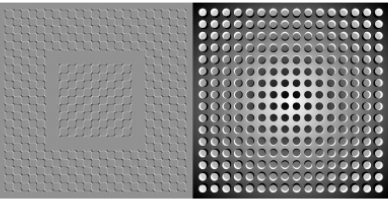
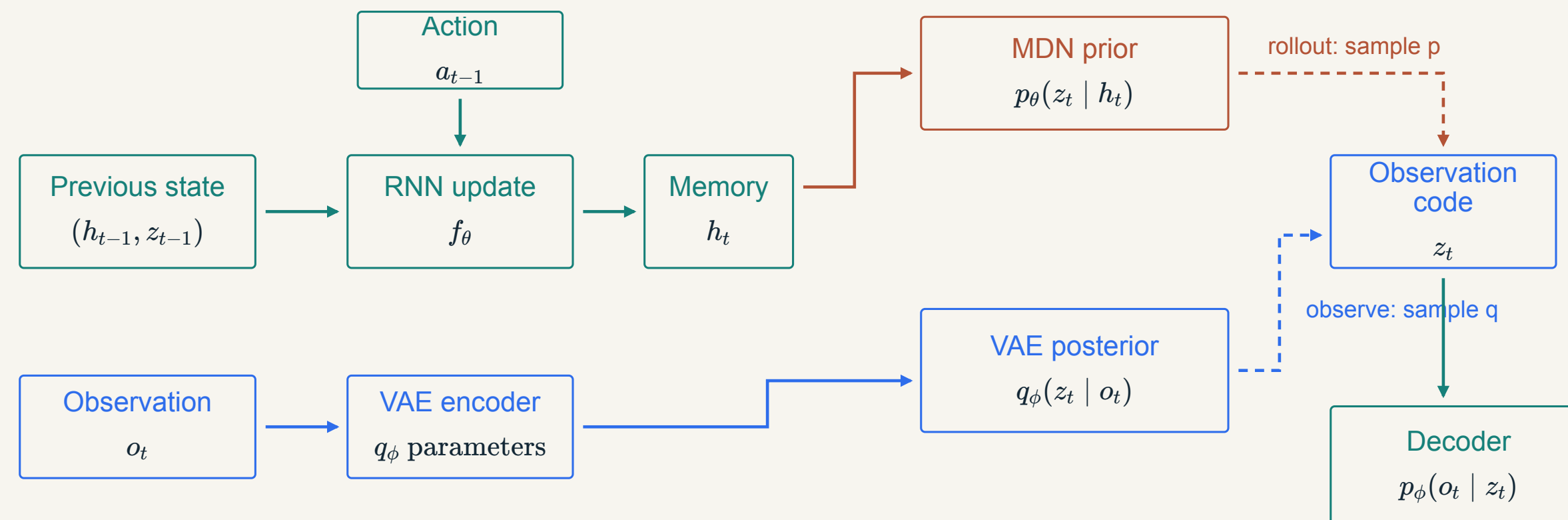


Figure 2. What we see is based on our brain’s prediction of the future (Kitaoka, 2002; Watanabe et al., 2018).

¹Google Brain ²NNAISENSE ³Swiss AI Lab, IDSIA (USI & SUPSI)

World Models: vision and memory



Encode the current frame as z ; retain earlier history in h .

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

While observing, sample the frame VAE posterior. During rollout, sample the MDN prior.

Encoder

Stochastic VAE code:

$$z_t \sim q_\phi(\cdot | o_t)$$

Latent meaning

z_t encodes the current frame; h_t summarizes earlier codes and actions.

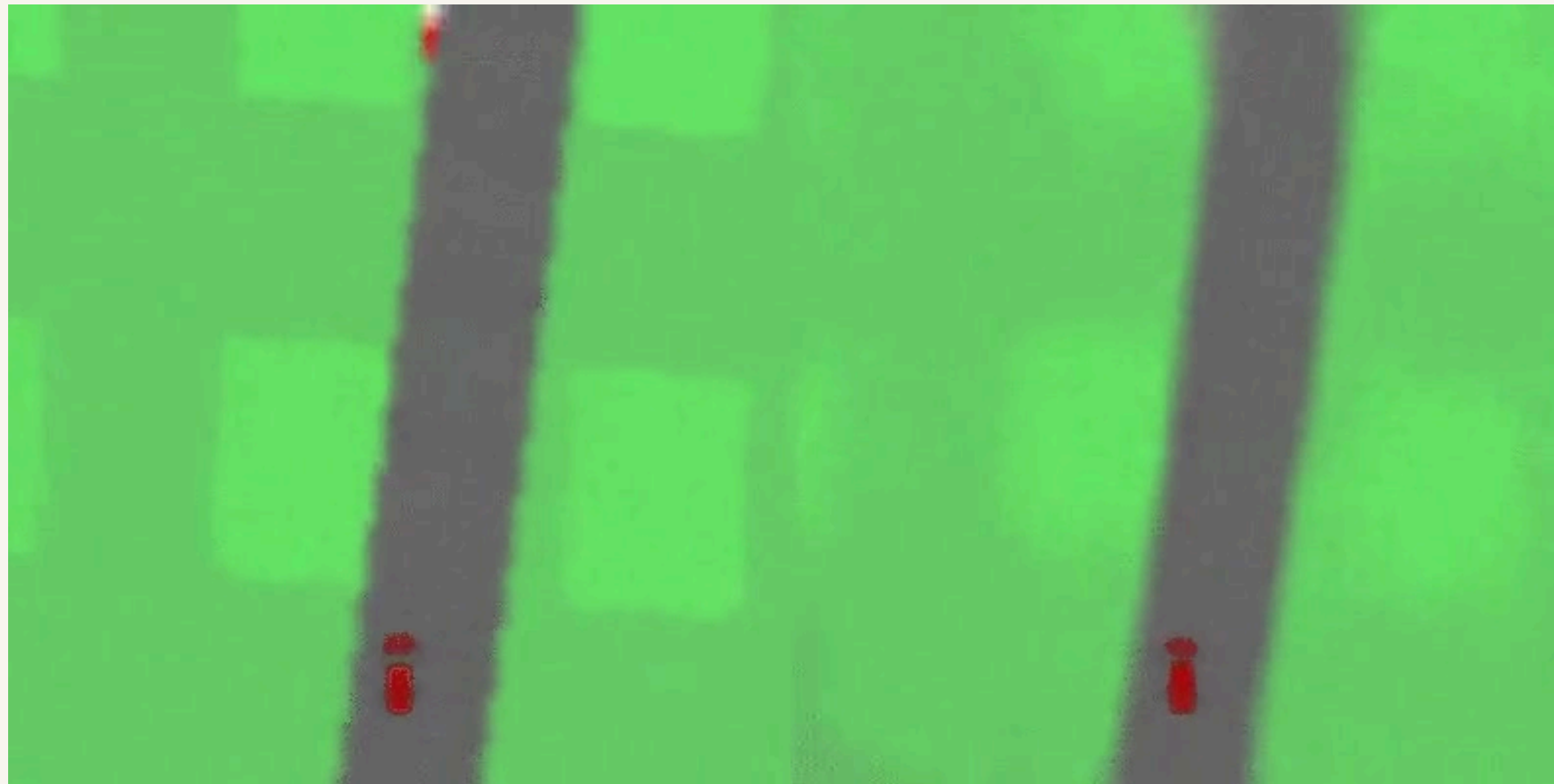
Prediction

Deterministic h update; probabilistic z update through an MDN.

Loss

VAE objective, then next-code MDN negative log likelihood.

World Models: encoding observed frames



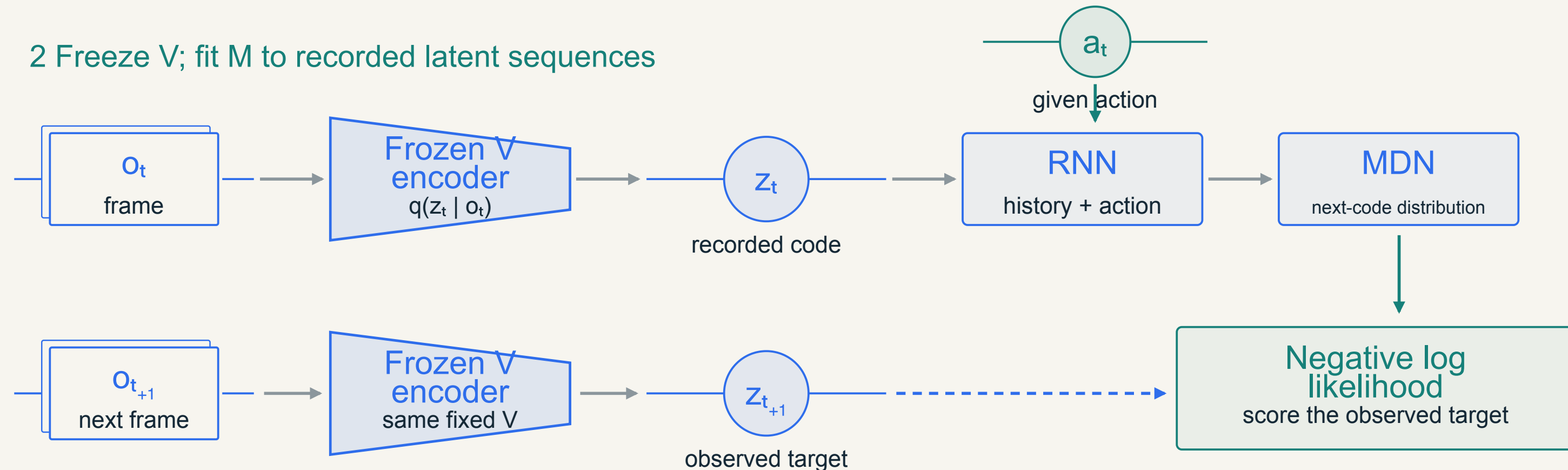
CarRacing observations and VAE reconstructions · original project video

The VAE encodes and reconstructs observed frames. Temporal prediction is the MDN-RNN's job.

World Models: staged training

1 Train V: reconstruct each observed frame + VAE KL

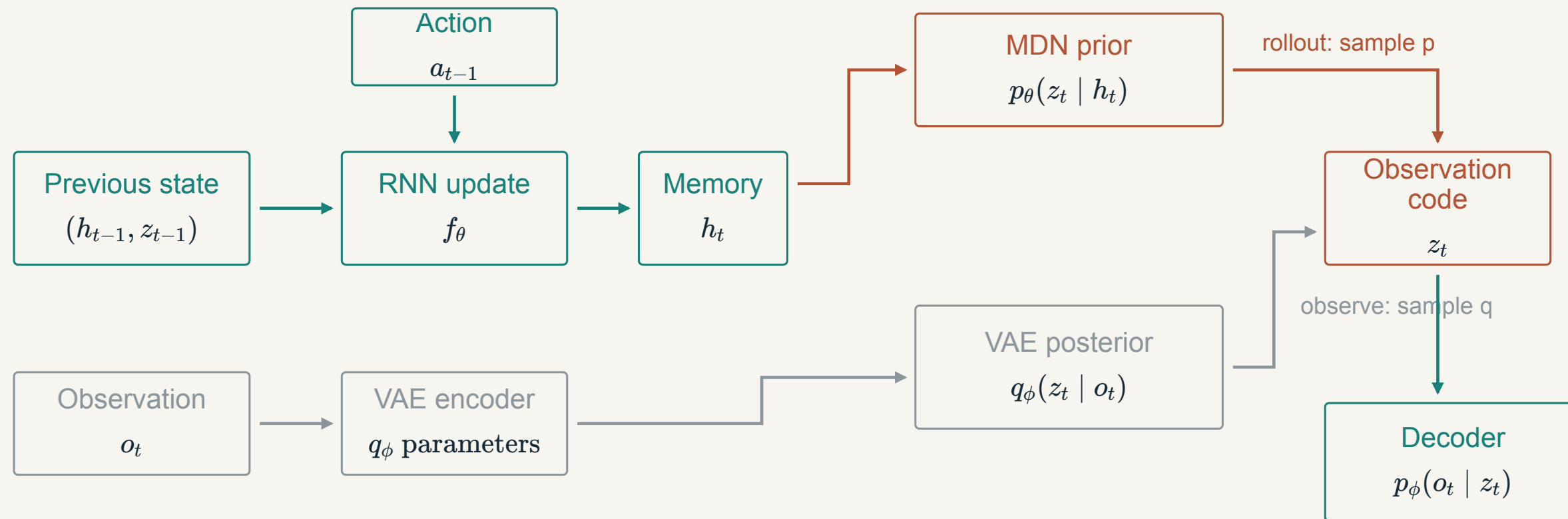
2 Freeze V; fit M to recorded latent sequences



$$\mathcal{L}_M = -\log p_\psi(z_{t+1} | z_{\leq t}, a_{\leq t})$$

M scores the recorded next code. Sampling and optional decoding appear on the rollout page.

World Models: imagined rollouts



Encode the current frame as z ; retain earlier history in h .

$$\hat{z}_t \xrightarrow{\text{decoder}} \hat{o}_t$$

The latent and memory continue the rollout. The decoded observation is a model output.

RSSM / Dreamer

Learning Latent Dynamics for Planning from Pixels

Danijar Hafner et al.

2019 · PlaNet

RSSM is introduced in PlaNet and reused in the Dreamer family.

arXiv:1811.04551v5 [cs.LG] 4 Jun 2019

Learning Latent Dynamics for Planning from Pixels

Danijar Hafner^{1,2} Timothy Lillicrap³ Ian Fischer⁴ Ruben Villegas^{1,5}
David Ha¹ Honglak Lee¹ James Davidson¹

Abstract

Planning has been very successful for control tasks with known environment dynamics. To leverage planning in unknown environments, the agent needs to learn the dynamics from interactions with the world. However, learning dynamics models that are accurate enough for planning has been a long-standing challenge, especially in image-based domains. We propose the Deep Planning Network (PlaNet), a purely model-based agent that learns the environment dynamics from images and chooses actions through fast online planning in latent space. To achieve high performance, the dynamics model must accurately predict the rewards ahead for multiple time steps. We approach this using a latent dynamics model with both deterministic and stochastic transition components. Moreover, we propose a multi-step variational inference objective that we name latent overshooting. Using only pixel observations, our agent solves continuous control tasks with contact dynamics, partial observability, and sparse rewards, which exceed the difficulty of tasks that were previously solved by planning with learned models. PlaNet uses substantially fewer episodes and reaches final performance close to and sometimes higher than strong model-free algorithms.

enough for planning has been a long-standing challenge. Key difficulties include model inaccuracies, accumulating errors of multi-step predictions, failure to capture multiple possible futures, and overconfident predictions outside of the training distribution.

Planning using learned models offers several benefits over model-free reinforcement learning. First, model-based planning can be more data efficient because it leverages a richer training signal and does not require propagating rewards through Bellman backups. Moreover, planning carries the promise of increasing performance just by increasing the computational budget for searching for actions, as shown by Silver et al. (2017). Finally, learned dynamics can be independent of any specific task and thus have the potential to transfer well to other tasks in the environment.

Recent work has shown promise in learning the dynamics of simple low-dimensional environments (Deisenroth & Rasmussen, 2011; Gal et al., 2016; Amos et al., 2018; Chua et al., 2018; Henaff et al., 2018). However, these approaches typically assume access to the underlying state of the world and the reward function, which may not be available in practice. In high-dimensional environments, we would like to learn the dynamics in a compact latent space to enable fast planning. The success of such latent models has previously been limited to simple tasks such as balancing cartpoles and controlling 2-link arms from dense rewards (Watter et al., 2015; Banijamali et al., 2017).

In this paper, we propose the Deep Planning Network (PlaNet), a model-based agent that learns the environment dynamics from pixels and chooses actions through online planning in a compact latent space. To learn the dynamics, we use a transition model with both stochastic and deterministic components. Moreover, we experiment with a novel generalized variational objective that encourages multi-step predictions. PlaNet solves continuous control tasks from pixels that are more difficult than those previously solved by planning with learned models.

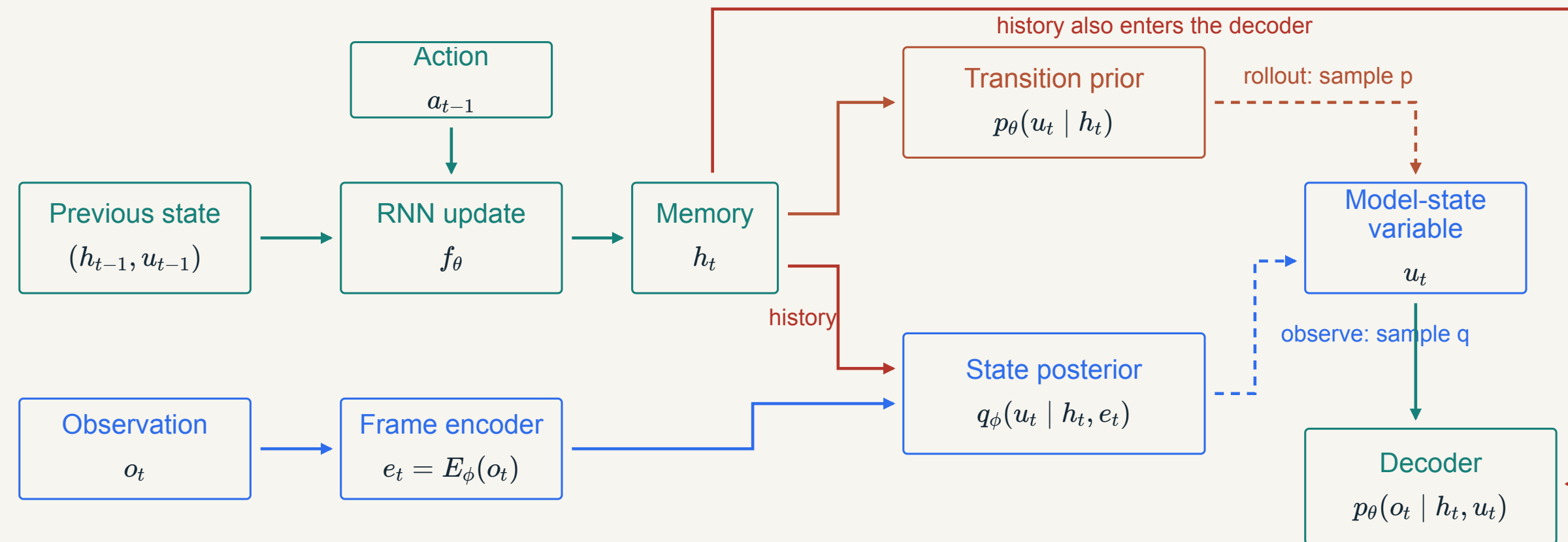
Key contributions of this work are summarized as follows:

- **Planning in latent spaces** We solve a variety of tasks from the DeepMind control suite, shown in Figure 1, by learning a dynamics model and efficiently planning in

¹Google Brain ²University of Toronto ³DeepMind ⁴Google Research ⁵University of Michigan. Correspondence to: Danijar Hafner <mail@danijar.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

RSSM: PlaNet and Dreamer



Infer the latent state u from history + the current observation.

$$h_t = f_\theta(h_{t-1}, u_{t-1}, a_{t-1}), \quad \tilde{s}_t = (h_t, u_t)$$

Same RNN + probabilistic latent structure. Red arrows show RSSM's added history dependencies.

Latent meaning

u_t is inferred using history and the current observation.

Posterior

$q_\phi(u_t | h_t, E(o_t))$ uses the temporal context.

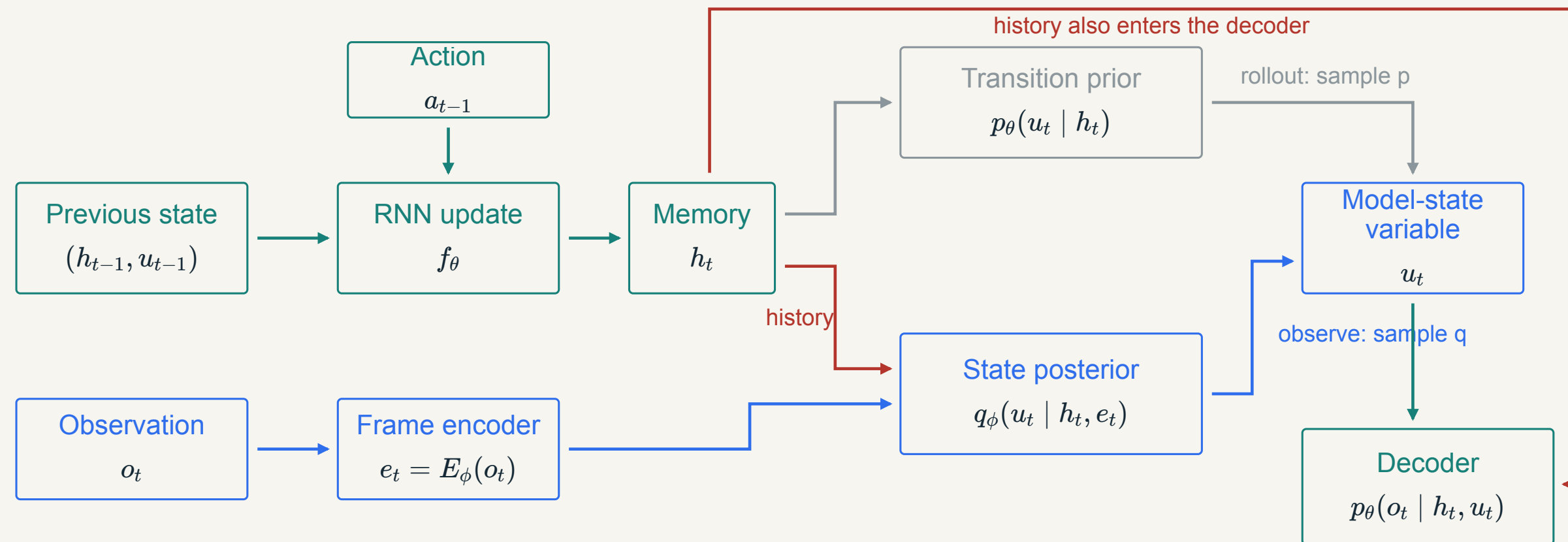
Shared structure

Deterministic RNN memory + probabilistic latent transitions.

Difference: training

Joint observation NLL + KL to the temporal prior.

RSSM: updating with observations



Infer the latent state u from history + the current observation.

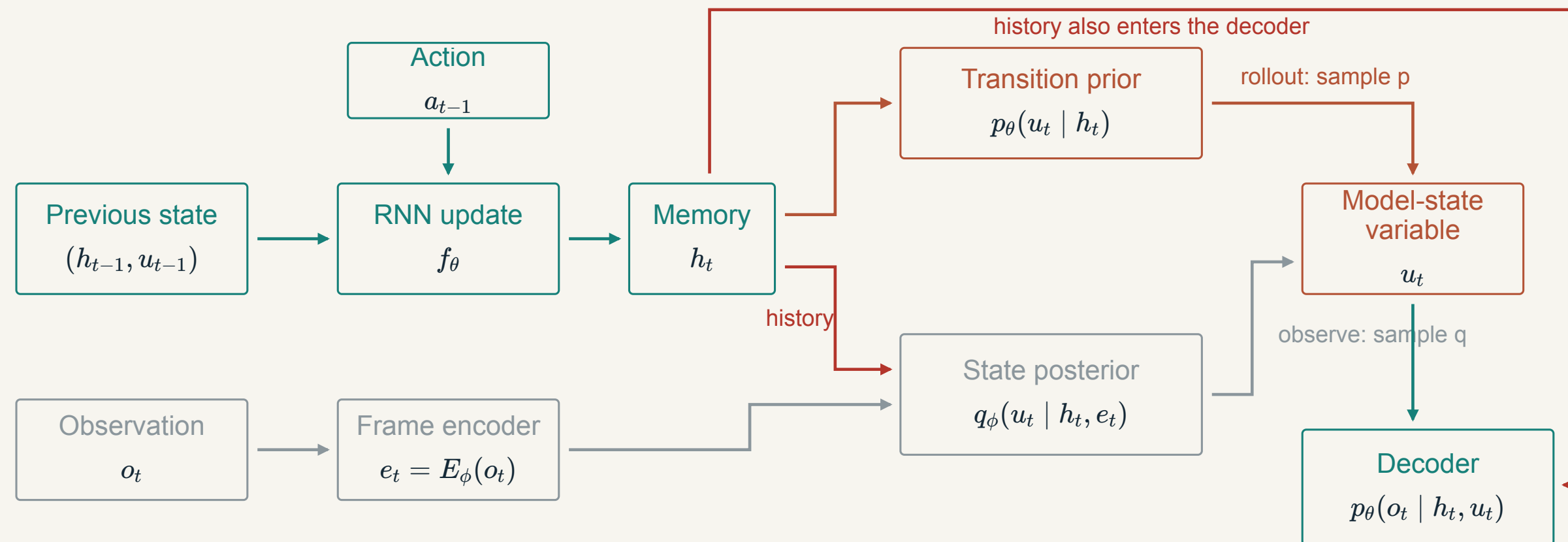
World Models 2018: current frame

$$z_t \sim q_{\phi}(\cdot \mid o_t)$$

RSSM: history + current observation

$$u_t \sim q_{\phi}(\cdot \mid h_t, E_{\phi}(o_t))$$

RSSM: imagining without observations

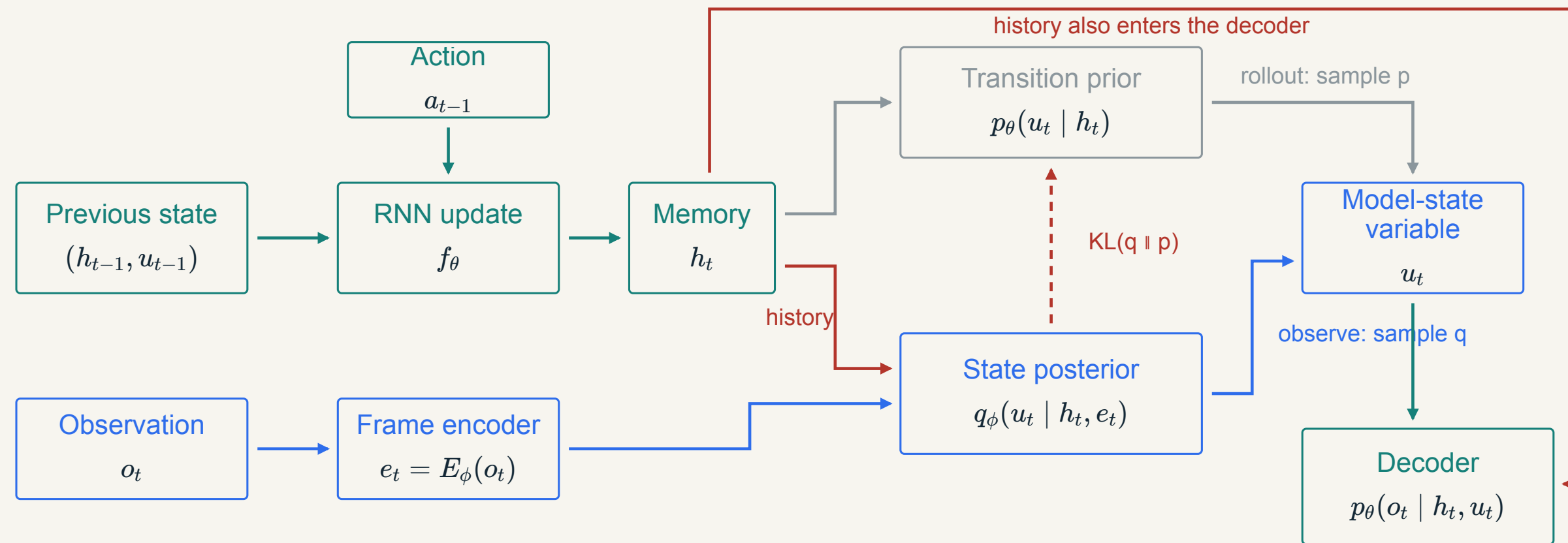


Infer the latent state u from history + the current observation.

$$u_t \sim p_\theta(u_t | h_t), \quad h_{t+1} = f_\theta(h_t, u_t, a_t)$$

During imagination, sample the prior and continue. The observation branch is unavailable.

RSSM: learning the model



Infer the latent state u from history + the current observation.

$$\mathcal{L}_t = \mathbb{E}_q[-\log p_\theta(o_t | h_t, u_t)] + \beta D_{\text{KL}}(q_\phi(u_t | h_t, e_t) || p_\theta(u_t | h_t))$$

Joint temporal training: the posterior is matched to a history-dependent prior.

Dreamer V1–V4: world-model choices

V1–V3: recurrent state-space world models

V1

Gaussian stochastic state

V2

Categorical state
KL balancing

V3

Categorical RSSM
More robust training

V4: a Transformer world model

Video frames

$O_{\leq t}$



Causal tokenizer

continuous video tokens



Block-causal dynamics

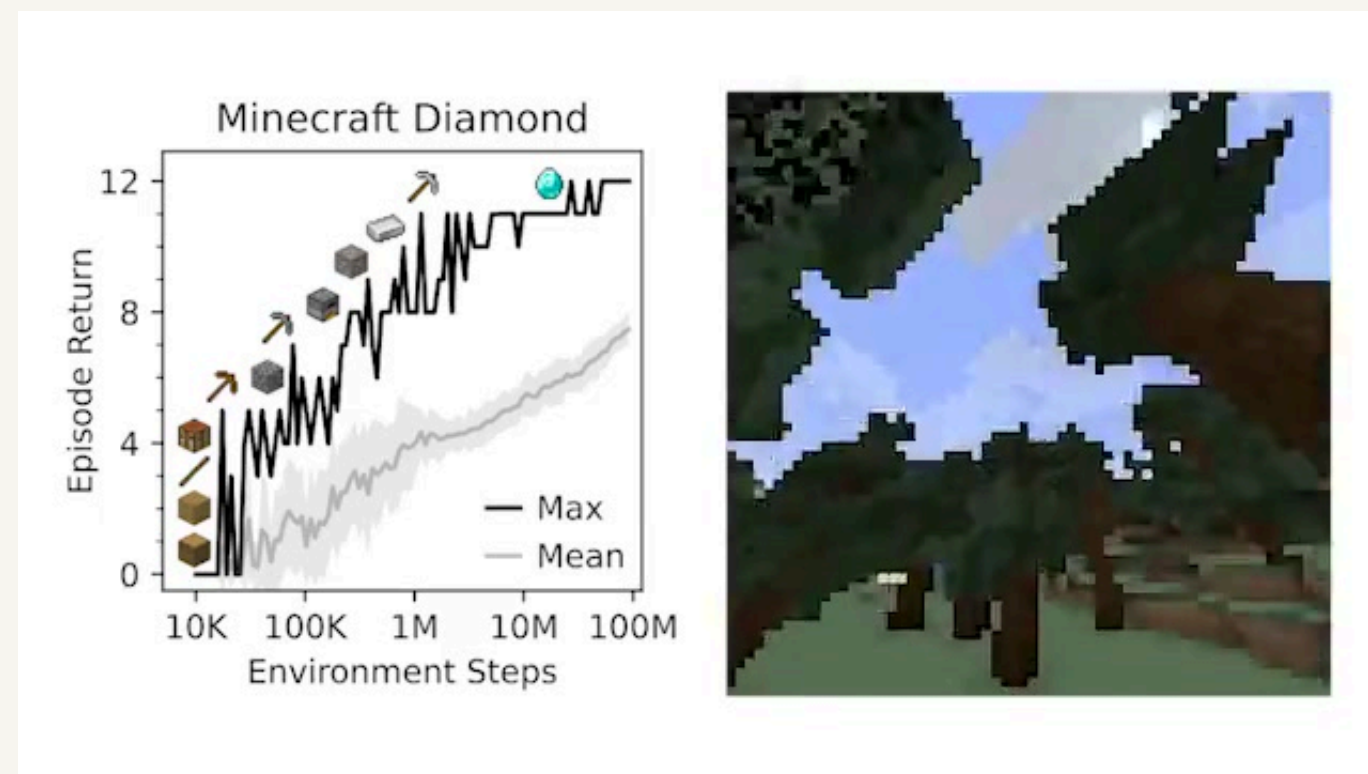
tokens + supplied actions

Shortcut forcing trains the dynamics to denoise future representations with few generation steps.

V4 changes the world-model architecture. Latent imagination remains the shared idea.

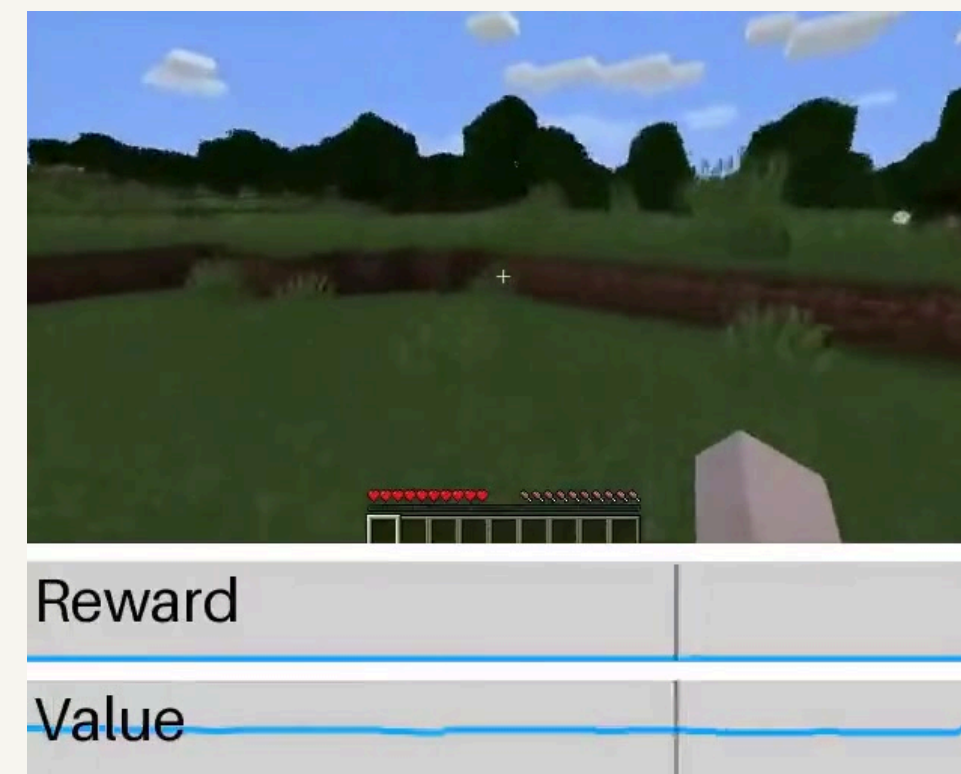
Dreamer: environment behavior and imagination

V3: agent execution



Minecraft environment · DreamerV3 agent

V4: imagined trajectory



Generated Minecraft frames · Dreamer 4

The left clip shows behavior in the environment; the right clip shows a model-generated future.

IRIS

Transformers are Sample-Efficient World Models

Vincent Micheli · Eloi Alonso · François Fleuret

2022 preprint · ICLR 2023

arXiv:2209.00588v2 [cs.LG] 1 Mar 2023

Published as a conference paper at ICLR 2023

TRANSFORMERS ARE SAMPLE-EFFICIENT WORLD MODELS

Vincent Micheli*
University of Geneva

Eloi Alonso*
University of Geneva

François Fleuret
University of Geneva

ABSTRACT

Deep reinforcement learning agents are notoriously sample inefficient, which considerably limits their application to real-world problems. Recently, many model-based methods have been designed to address this issue, with learning in the imagination of a world model being one of the most prominent approaches. However, while virtually unlimited interaction with a simulated environment sounds appealing, the world model has to be accurate over extended periods of time. Motivated by the success of Transformers in sequence modeling tasks, we introduce IRIS, a data-efficient agent that learns in a world model composed of a discrete autoencoder and an autoregressive Transformer. With the equivalent of only two hours of gameplay in the Atari 100k benchmark, IRIS achieves a mean human normalized score of 1.046, and outperforms humans on 10 out of 26 games, setting a new state of the art for methods without lookahead search. To foster future research on Transformers and world models for sample-efficient reinforcement learning, we release our code and models at <https://github.com/eloialonso/iris>

1 INTRODUCTION

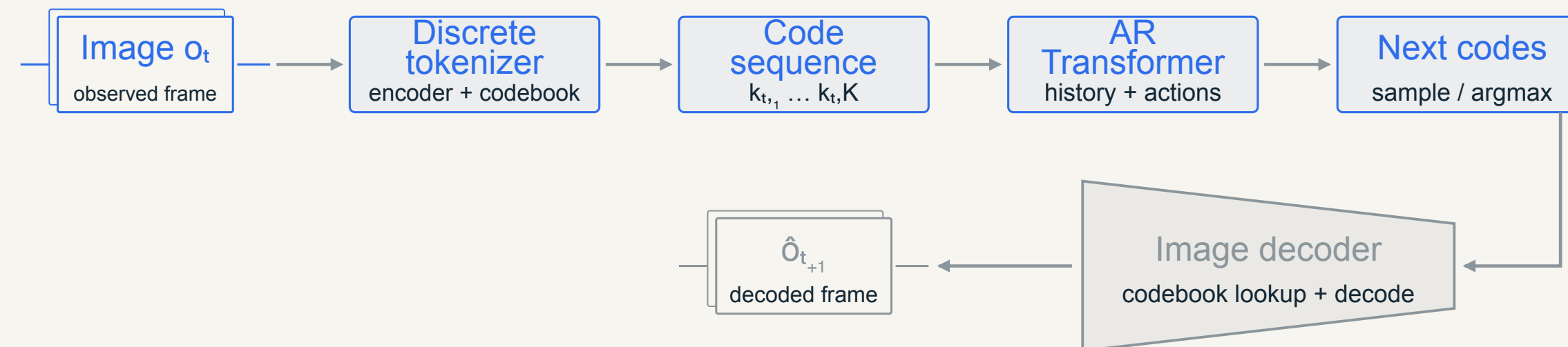
Deep Reinforcement Learning (RL) has become the dominant paradigm for developing competent agents in challenging environments. Most notably, deep RL algorithms have achieved impressive performance in a multitude of arcade (Mnih et al. [2015], Schrittwieser et al. [2020], Hafner et al. [2021]), real-time strategy (Vinyals et al. [2019], Berner et al. [2019]), board (Silver et al. [2016], [2018], Schrittwieser et al. [2020]) and imperfect information (Schmid et al. [2021], Brown et al. [2020a]) games. However, a common drawback of these methods is their extremely low sample efficiency. Indeed, experience requirements range from months of gameplay for DreamerV2 (Hafner et al. [2021]) in Atari 2600 games (Bellemare et al. [2013b]) to thousands of years for OpenAI Five in Dota2 (Berner et al. [2019]). While some environments can be sped up for training agents, real-world applications often cannot. Besides, additional cost or safety considerations related to the number of environmental interactions may arise (Yampolskiy [2018]). Hence, sample efficiency is a necessary condition to bridge the gap between research and the deployment of deep RL agents in the wild.

Model-based methods (Sutton & Barto [2018]) constitute a promising direction towards data efficiency. Recently, world models were leveraged in several ways: pure representation learning (Schwarzer et al. [2021]), lookahead search (Schrittwieser et al. [2020], Ye et al. [2021]), and learning in imagination (Ha & Schmidhuber [2018], Kaiser et al. [2020], Hafner et al. [2020], [2021]). The latter approach is particularly appealing because training an agent inside a world model frees it from sample efficiency constraints. Nevertheless, this framework relies heavily on accurate world models since the policy is purely trained in imagination. In a pioneering work, Ha & Schmidhuber (2018) successfully built imagination-based agents in toy environments. SimPLe recently showed promise in the more challenging Atari 100k benchmark (Kaiser et al. [2020]). Currently, the best Atari agent learning in imagination is DreamerV2 (Hafner et al. [2021]), although it was developed and evaluated with two hundred million frames available, far from the sample-efficient regime. Therefore, designing new world model architectures, capable of handling visually complex and partially observable environments with few samples, is key to realize their potential as surrogate training grounds.

The Transformer architecture (Vaswani et al. [2017]) is now ubiquitous in Natural Language Processing (Devlin et al. [2019], Radford et al. [2019], Brown et al. [2020b], Raffel et al. [2020]), and is also gaining traction in Computer Vision (Dosovitskiy et al. [2021], He et al. [2022]), as well as in Offline

*Equal contributions, order determined by a coin flip. Correspondence: {first.last}@unige.ch

IRIS: images as discrete codes



Separate losses; repeated updates of both autoencoder and Transformer.

This diagram shows IRIS's observation and dynamics components.

Encoder

Deterministic assignment to discrete image codes.

State

Code and action history inside the sequence model.

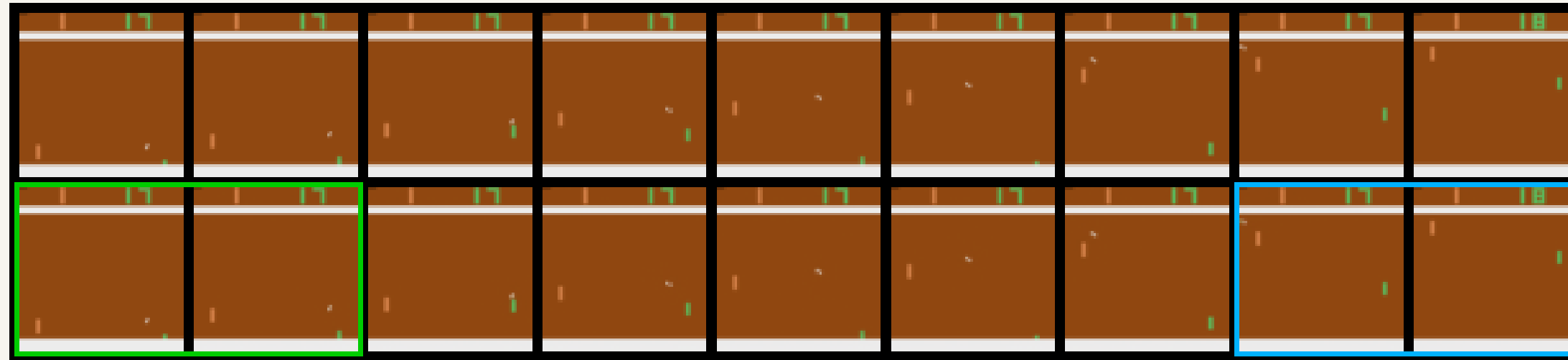
Prediction

Probabilistic AR: categorical next-token distribution.

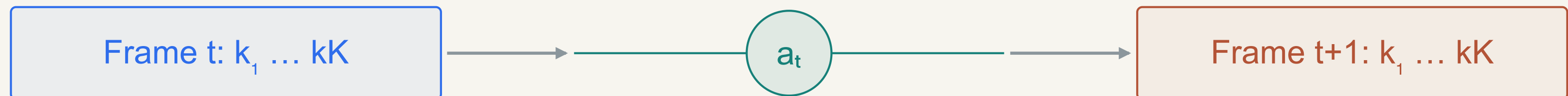
Loss

Autoencoder reconstruction / codebook terms; token cross-entropy for dynamics.

IRIS: generating future frames



Recorded trajectory (top) and model rollout under recorded actions (bottom). Green: conditioning frames. Original paper, Fig. 3.



There are two generation orders: across time and within each frame.

IRIS: agent behavior in Atari



Official IRIS agent demonstration · Atari environments

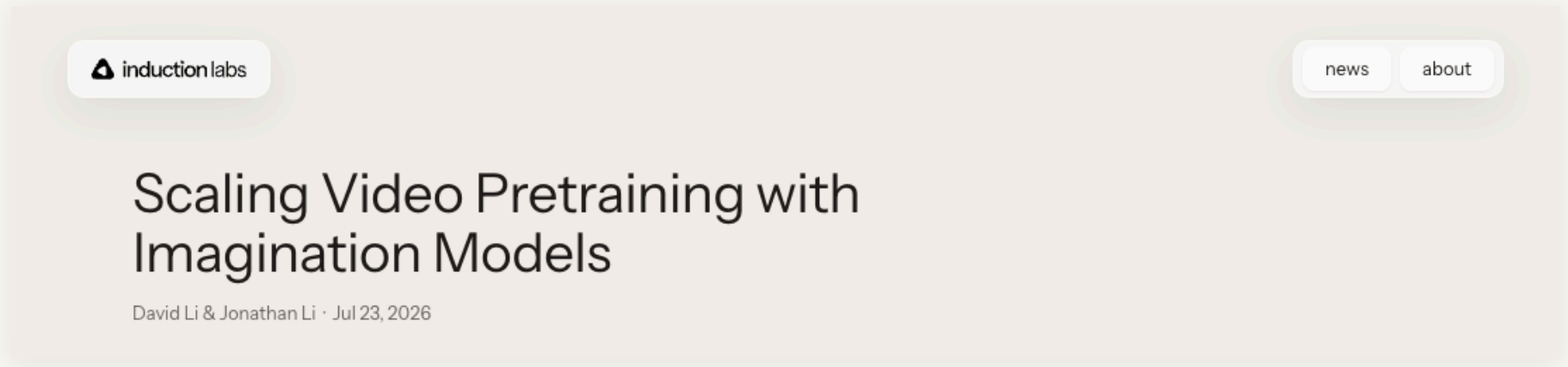
An agent trained with imagined trajectories acts in the Atari environments.

Photon-1

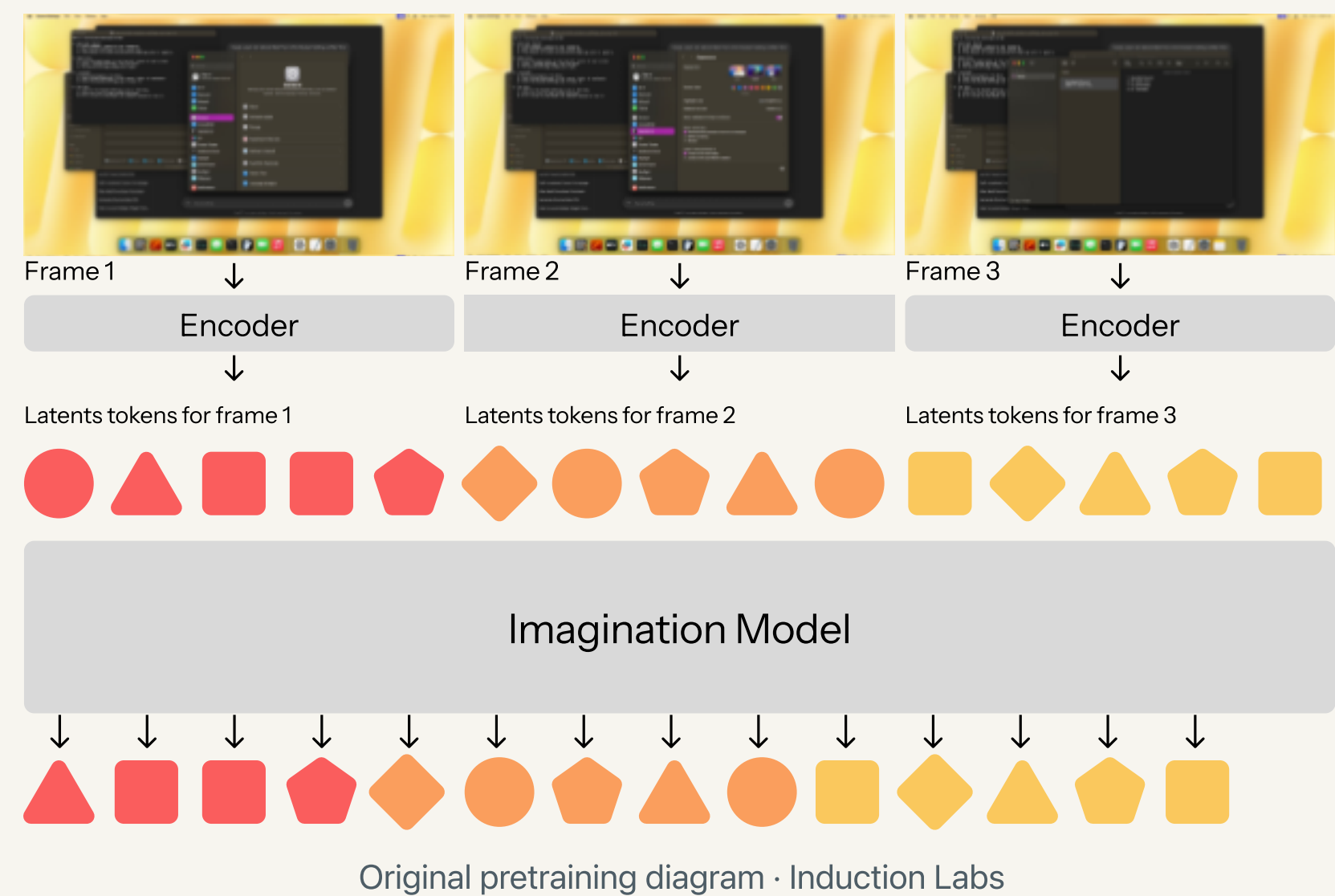
Scaling Video Pretraining with Imagination Models

David Li · Jonathan Li

July 2026 · Induction Labs technical report



Photon-1: video pretraining at scale



106B parameters / 5B active · 575M frames · 960 discrete tokens per frame

- Encoder**
Differential FSQ: discrete frame-change codes.
- State**
Token history inside the Transformer.
- Prediction**
Probabilistic autoregressive next-token prediction.
- Learning**
Video pretraining; later action SFT and RL.

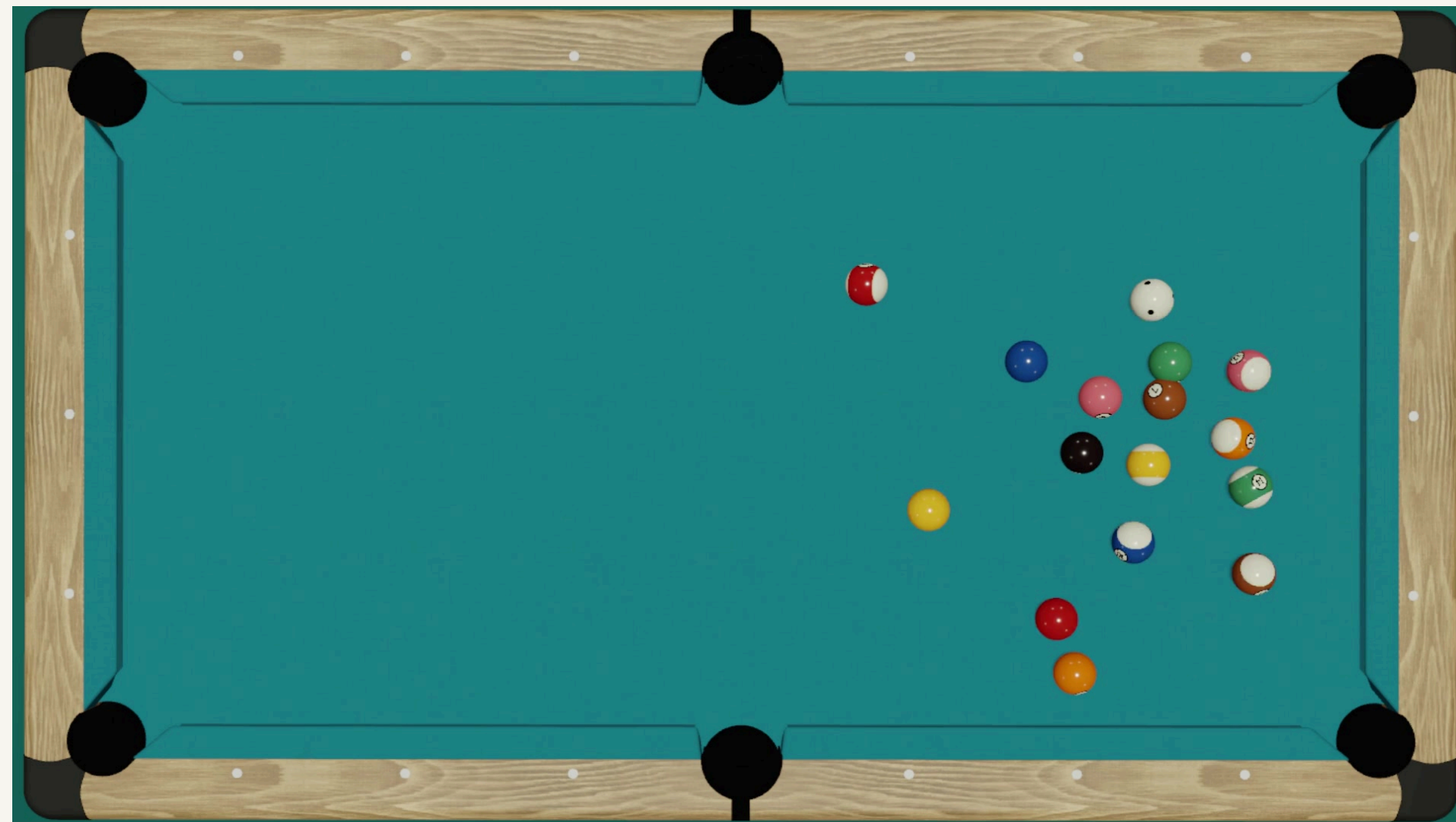
IRIS and Photon-1: shared recipe, different training

	IRIS	Photon-1
Visual representation	VQ image codes	FSQ codes of frame differences
Dynamics data	Action-labeled Atari interactions	Video pretraining without action labels
Actions	Condition imagined next frames	After SFT: predict next state, then action
Image decoder	Autoencoder reconstruction; rollout rendering	Auxiliary visualization only

Shared recipe: discrete visual representations + autoregressive next-token prediction.

A scaled extension of this modeling family; the representation and training stages also change.

Photon-1: predicted billiard dynamics



Official Photon-1 billiards demonstration

Predicted ball positions are rendered after billiards finetuning.

Comparing the model structures

Model	Representation / history	Next-step prediction
TD-MPC	Stacked frames → compact latent state	Deterministic latent transition
V-JEPA 2-AC	Frozen features + action/pose history	Point prediction (L1 training)
LeWorldModel	Jointly learned features + causal history	Point prediction (MSE training)
World Models (2018)	Frame VAE code + RNN memory	MDN over the next code
RSSM / Dreamer V1–V3	Memory + stochastic model state	Prior; posterior when observations arrive
IRIS	Discrete image codes + causal history	Categorical next-token probabilities
Photon-1	Discrete frame-change codes + causal history	Categorical next-token probabilities

These are model-component comparisons, not complete agent descriptions.

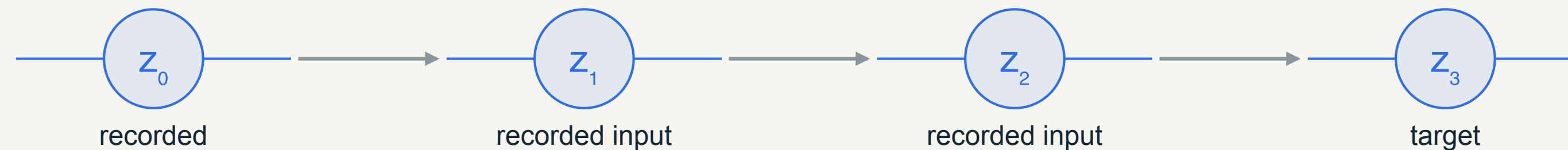
Comparing how the models are learned

Model	Representation and dynamics	What constrains the representation?
TD-MPC	Joint task-oriented learning	Consistency + reward/value signals
V-JEPA 2-AC	Pretrain, then freeze encoder	Fixed pretrained target features
LeWorldModel	Joint encoder and predictor	Predictive objective + SIGReg
World Models (2018)	VAE, then MDN-RNN	Reconstruction + framewise VAE KL
RSSM / Dreamer V1–V3	Joint temporal inference and dynamics	Observation modeling + prior/posterior agreement
IRIS	Separate losses; repeated updates of both modules	Reconstruction through discrete codes
Photon-1	Video pretraining, then action learning	Encoder training details not specified in report

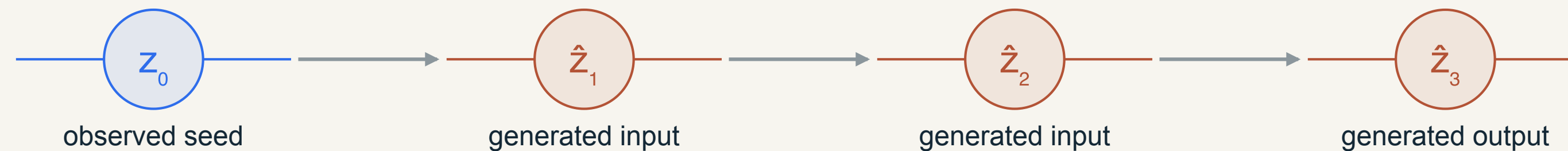
A learned representation becomes useful for world modeling through its temporal and action-conditioned role.

Training-rollout mismatch

Observed-history training



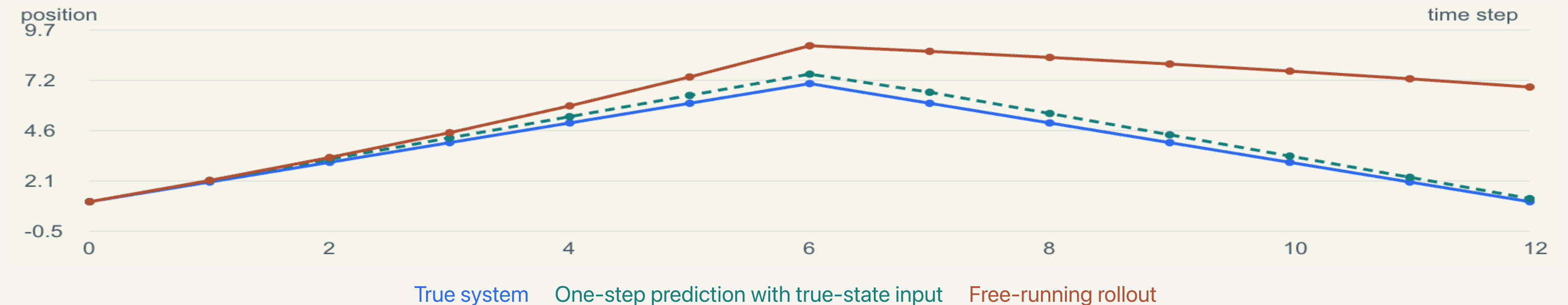
Free-running rollout



Remedy: train on multi-step predictions that feed generated outputs back as inputs.

Example: V-JEPA 2-AC combines teacher forcing with a two-step rollout loss.

Compounding error and observation updates



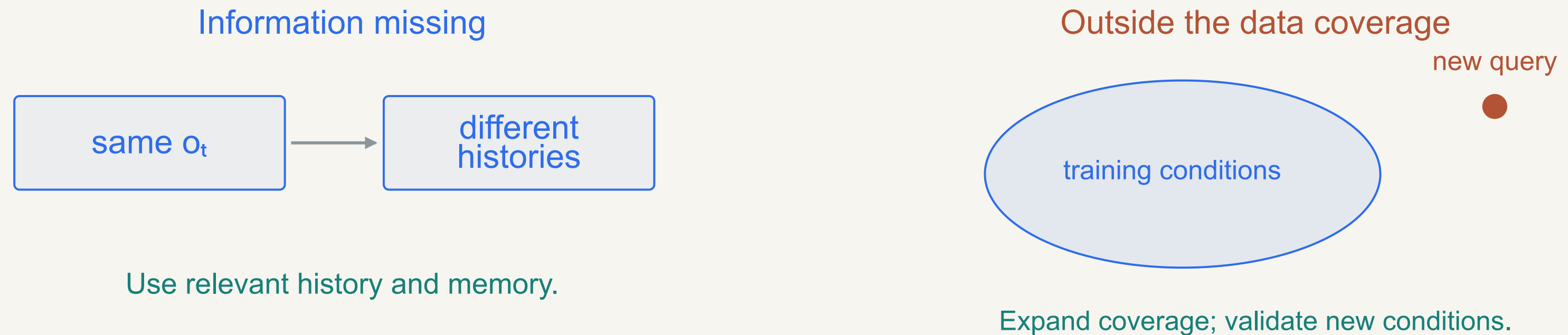
$$x_0 = \hat{x}_0 = 1; g = 1.08; a_0 \dots a_5 = +1; a_6 \dots a_{11} = -1; e_t = \hat{x}_t - x_t.$$

Step 12/12 · one-step error: 0.160 · rollout error: 5.823

$$x_{t+1} = x_t + a_t, \quad \hat{x}_{t+1} = g\hat{x}_t + a_t \quad \Rightarrow \quad e_{t+1} = ge_t + (g - 1)x_t$$

Remedies: multi-step consistency, a validated rollout horizon, and state updates when real observations arrive.

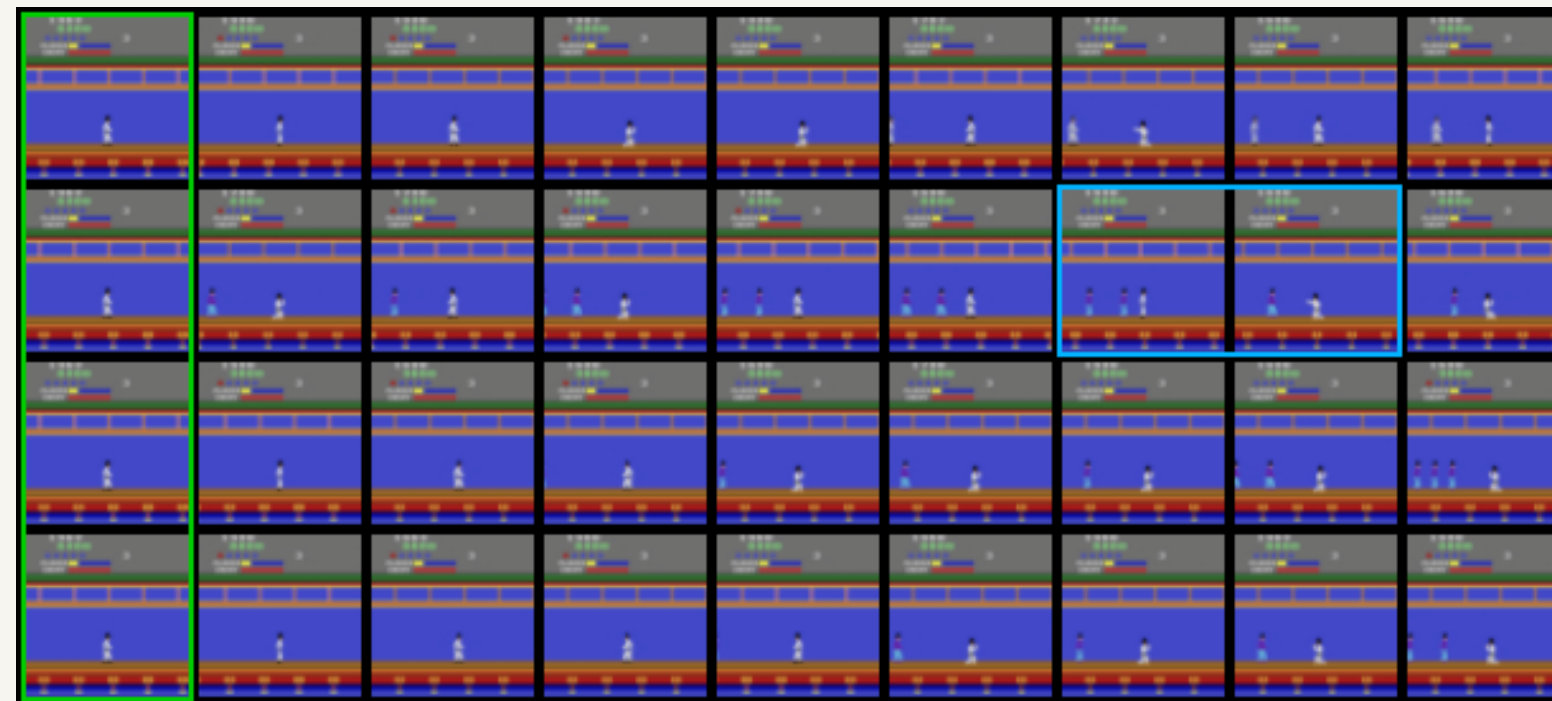
Missing information and unfamiliar inputs



A richer output distribution cannot recover information that the representation discarded.

Longer context and ensemble disagreement can help; neither guarantees reliable out-of-distribution prediction.

Multiple futures and uncertainty

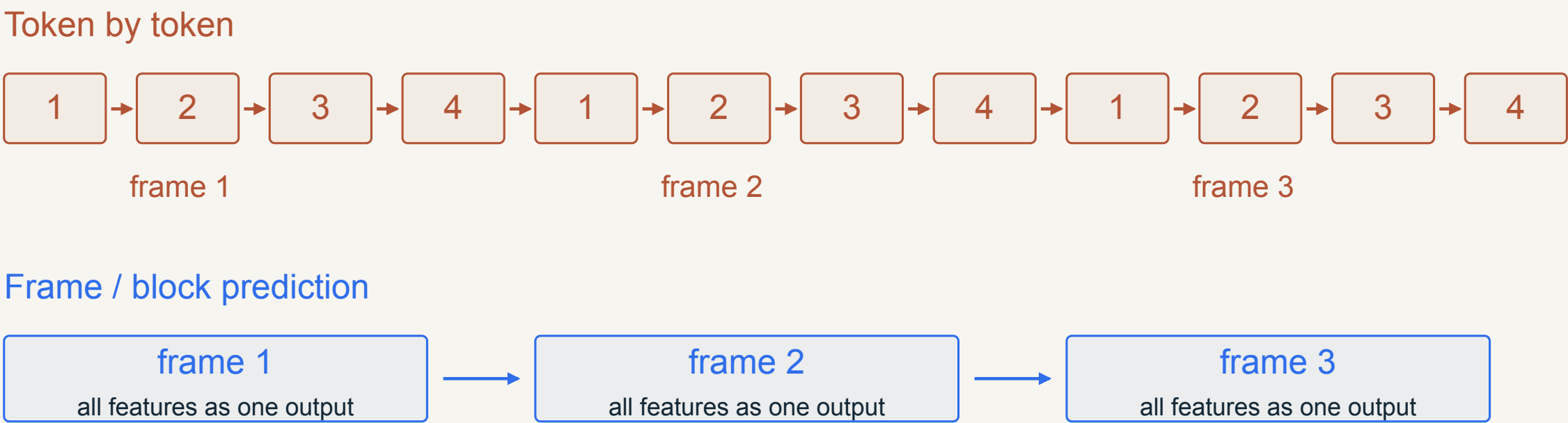


Four imagined trajectories from the same initial frame · IRIS, Fig. 2

Use probabilistic dynamics and several samples when the future is genuinely ambiguous.

More randomness does not repair model bias. Evaluate the predicted distribution, not only one sample.

Sequential generation cost



Cost	Possible mitigation	Tradeoff
Many serial token steps	Fewer tokens; frame/block prediction	Compression may lose information
Repeated computation / long context	Caching; context compression	Memory use or forgotten history
Many iterative sampling steps	Distillation / shortcut methods	Additional training; later lectures

H frames $\times$ K serial codes $\approx HK$ prediction steps. Compare actual latency too: steps have different costs.

Discussion: how would you evaluate this model?

Check	Concrete test
Data split	Hold out entire episodes, not adjacent frames.
Rollout horizon	Same initial evidence and actions; compare one-step and H-step rollout.
Probabilistic output	Use held-out NLL when available; check calibration or coverage.
Representation scale	Raw latent MSE is not comparable across different encoders.
Use conditions	Test new scenes/actions separately and measure generation latency.

Discuss: choose one model and explain which checks apply.

Next: use predicted futures for planning, then learn behavior with world models.