Dreamer

But not only v1

William Ruys

Published as a conference paper at ICLR 2020

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

Published as a conference paper at ICLR 2021

MASTERING ATARI WITH DISCRETE WORLD MODELS

Danijar Hafner *
Google Research

Timothy Lillicrap
DeepMind

Mohammad Norouzi
Google Research

Jimmy Ba
University of Toronto

Article

Mastering diverse control tasks through world models

https://doi.org/10.1038/s41586-025-08744-2
Received: 5 April 2024

Danijar Hafner[1⊠], Jurgis Pasukonis[1], Jimmy Ba[2] & Timothy Lillicrap[1]

Dreamer

But not only v1

William Ruys

v1

Published as a conference paper at ICLR 2020

Dream to Control: Learning Behaviors by Latent Imagination

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

v2

Published as a conference paper at ICLR 2021

Mastering Atari with Discrete World Models

Danijar Hafner *
Google Research

Timothy Lillicrap
DeepMind

Mohammad Norouzi
Google Research

Jimmy Ba
University of Toronto

v3

Article

Mastering diverse control tasks through world models

https://doi.org/10.1038/s41586-025-08744-2

Received: 5 April 2024

Danijar Hafner[1⊠], Jurgis Pasukonis[1], Jimmy Ba[2] & Timothy Lillicrap[1]

Dreamer

But not only v1

William Ruys

Architecture + Idea **v1**

Published as a conference paper at ICLR 2020

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

v2

Published as a conference paper at ICLR 2021

MASTERING ATARI WITH DISCRETE WORLD MODELS

Danijar Hafner *
Google Research

Timothy Lillicrap
DeepMind

Mohammad Norouzi
Google Research

Jimmy Ba
University of Toronto

v3

Article

Mastering diverse control tasks through world models

https://doi.org/10.1038/s41586-025-08744-2

Received: 5 April 2024

Danijar Hafner[1⊠], Jurgis Pasukonis[1], Jimmy Ba[2] & Timothy Lillicrap[1]

Dreamer

But not only v1

Architecture + Idea **v1**

Published as a conference paper at ICLR 2020

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

Modify for Atari
Discrete Latent Space
Split KL-objective **v2**

Published as a conference paper at ICLR 2021

MASTERING ATARI WITH DISCRETE WORLD MODELS

Danijar Hafner *
Google Research

Timothy Lillicrap
DeepMind

Mohammad Norouzi
Google Research

Jimmy Ba
University of Toronto

v3

Article

Mastering diverse control tasks through world models

https://doi.org/10.1038/s41586-025-08744-2
Received: 5 April 2024

Danijar Hafner[1⊠], Jurgis Pasukonis[1], Jimmy Ba[2] & Timothy Lillicrap[1]

William Ruys

Dreamer

But not only v1

William Ruys

Architecture + Idea **v1**

Published as a conference paper at ICLR 2020

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

Modify for Atari
Discrete Latent Space
Split KL-objective **v2**

Published as a conference paper at ICLR 2021

MASTERING ATARI WITH DISCRETE WORLD MODELS

Danijar Hafner *
Google Research

Timothy Lillicrap
DeepMind

Mohammad Norouzi
Google Research

Jimmy Ba
University of Toronto

Reduce hyperparameter tuning
Reward Scaling
Distributional RL
Vanishing KL-objective **v3**

Article

Mastering diverse control tasks through world models

https://doi.org/10.1038/s41586-025-08744-2
Received: 5 April 2024

Danijar Hafner[1&2], Jurgis Pasukonis[1], Jimmy Ba[2] & Timothy Lillicrap[1]

Dreamer

But not only v1

William Ruys

Architecture + Idea **v1**

Published as a conference paper at ICLR 2020

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

Modify for Atari
Discrete Latent Space
Split KL-objective **v2**

Published as a conference paper at ICLR 2021

MASTERING ATARI WITH DISCRETE WORLD MODELS

Danijar Hafner *
Google Research

Timothy Lillicrap
DeepMind

Mohammad Norouzi
Google Research

Jimmy Ba
University of Toronto

Reduce hyperparameter tuning
Reward Scaling
Distributional RL **v3**
Vanishing KL-objective

Article

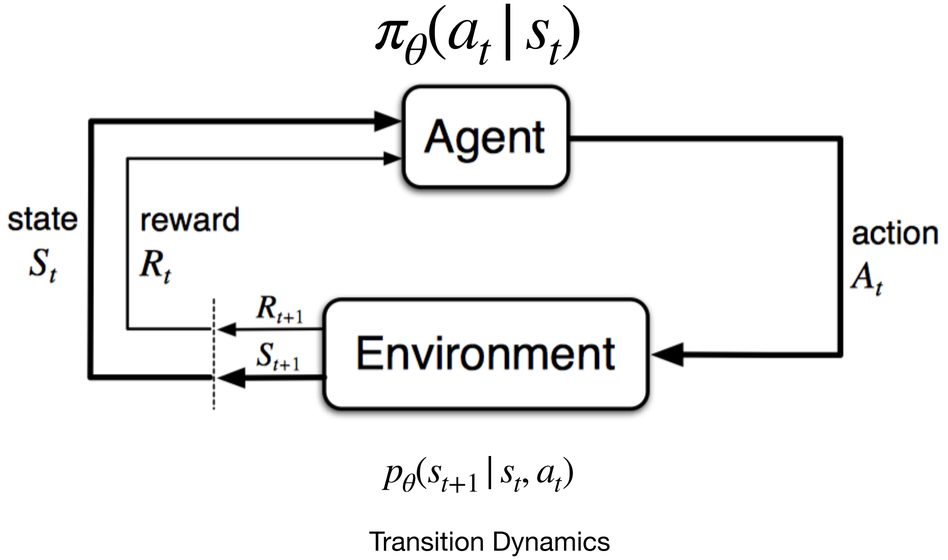
Mastering diverse control tasks through world models

https://doi.org/10.1038/s41586-025-08744-2
Received: 5 April 2024

Danijar Hafner[1&2], Jurgis Pasukonis[1], Jimmy Ba[2] & Timothy Lillicrap[1]

Reinforcement Learning

- We seek to learn a policy $\pi_\theta(a_t \mid s_t)$ to maximize total expected return
- Return of Trajectory $R(\tau) = \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \qquad (s_t, a_t) \in \tau$
- Maximize: $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[R(\tau)\right]$

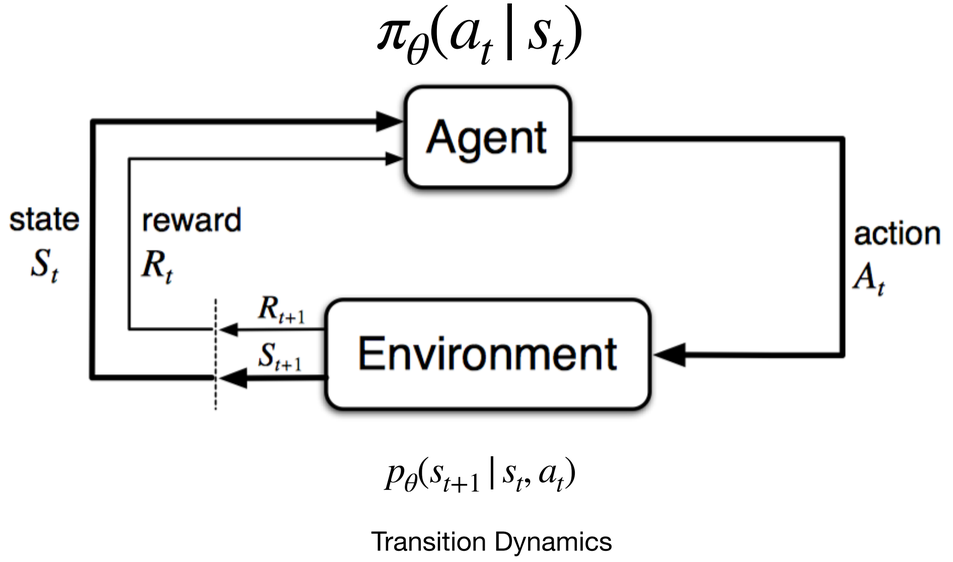


- Might only have observations not states
- Rewards might be stochastic

Policy Gradient Methods

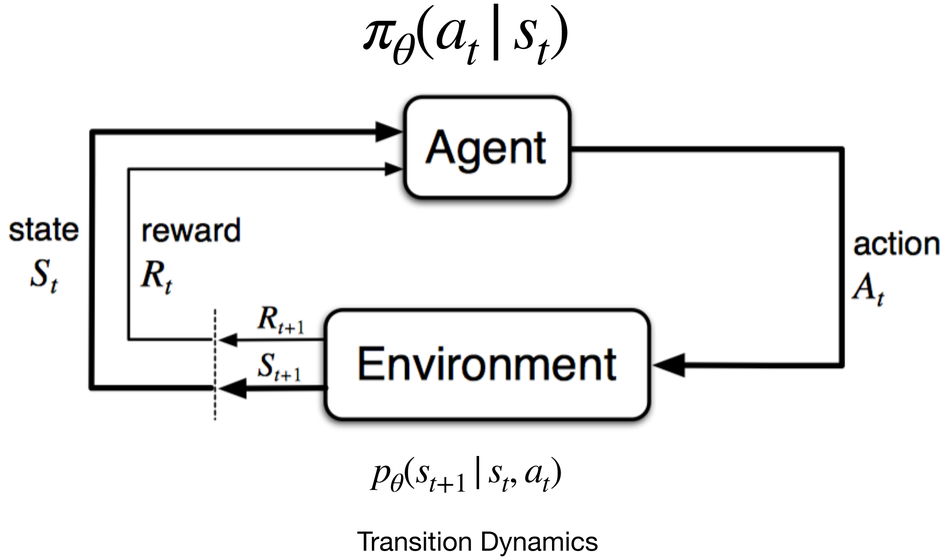
- Optimize the objective $J(\theta)$ directly.
  - Gradient ascent on: $\nabla_\theta J(\theta) \propto \mathbb{E}_{a_t \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) Q^{\pi_\theta}(s_t, a_t)\right]$

- Collect transitions (s_t, a_t, s_{t+1}, r_t) from env
- Increase log-likelihood of high return actions
- Repeat



What is Dreamer?

- Policy Gradient methods are *sample inefficient*
  - Need many environment interactions (~100k - 1B transitions)
  - Environment may be ‘slow’ or costly (real time, etc)

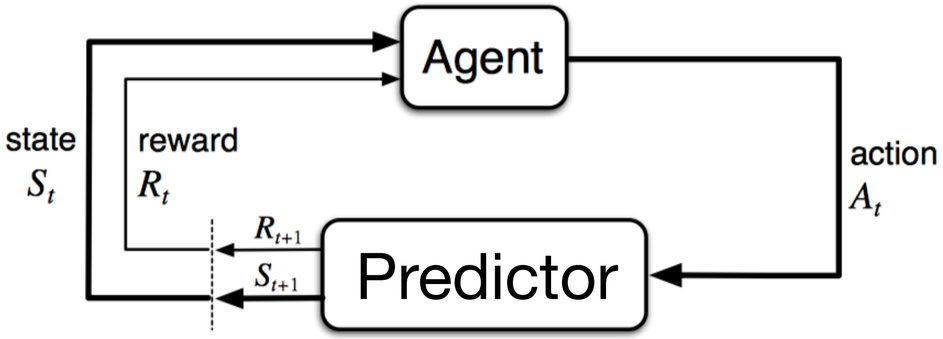


What is Dreamer?

- Policy Gradient methods are *sample inefficient*
  - Need many environment interactions (~100k - 1B transitions)
  - Environment may be ‘slow’ or costly (real time, etc)

- Replace real interactions with prediction
  - Neural network: model transitions and rewards
  - Network inference faster than real interaction
  - Perform policy gradient with predicted dynamics (“dream”)

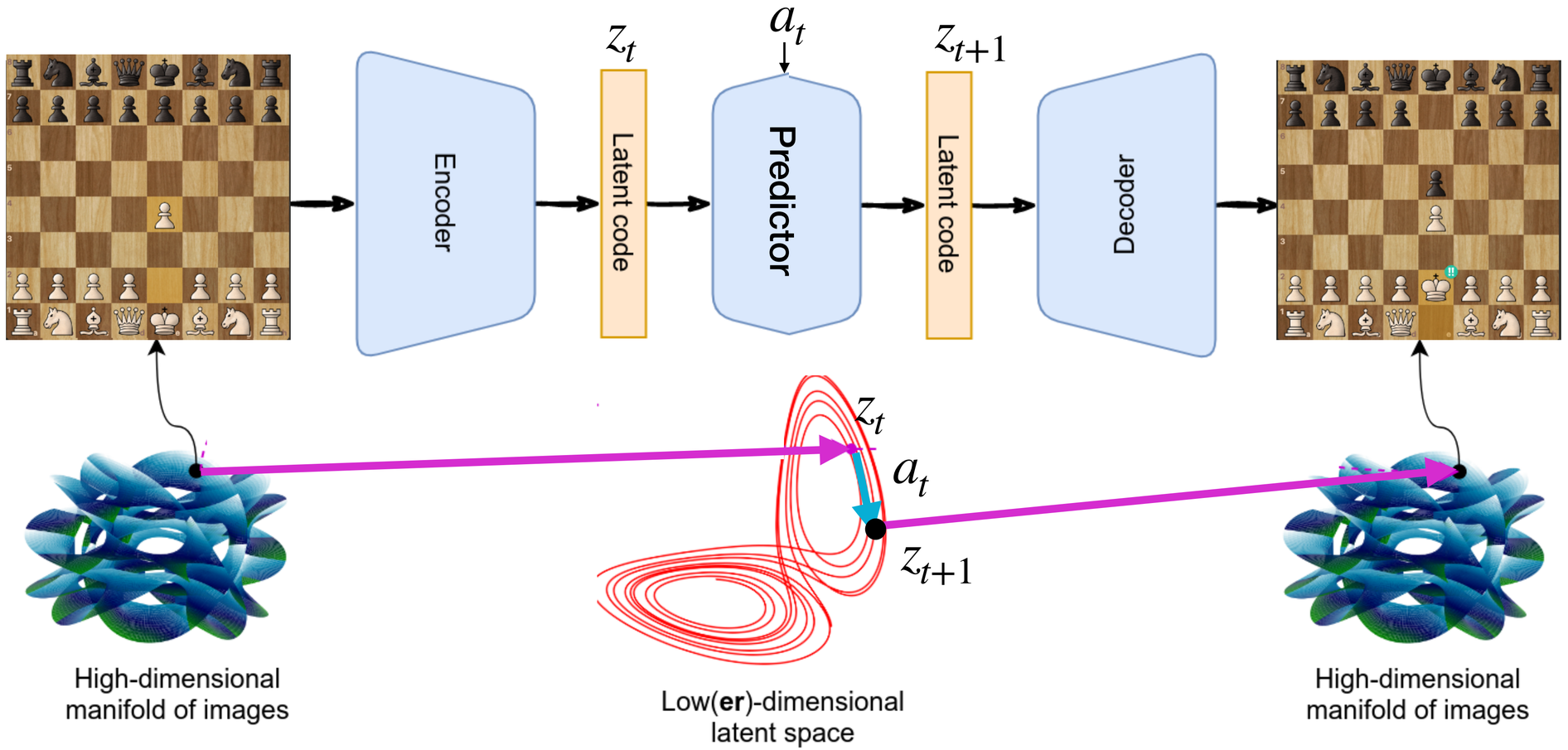
$\pi_\theta(a_t \mid s_t)$



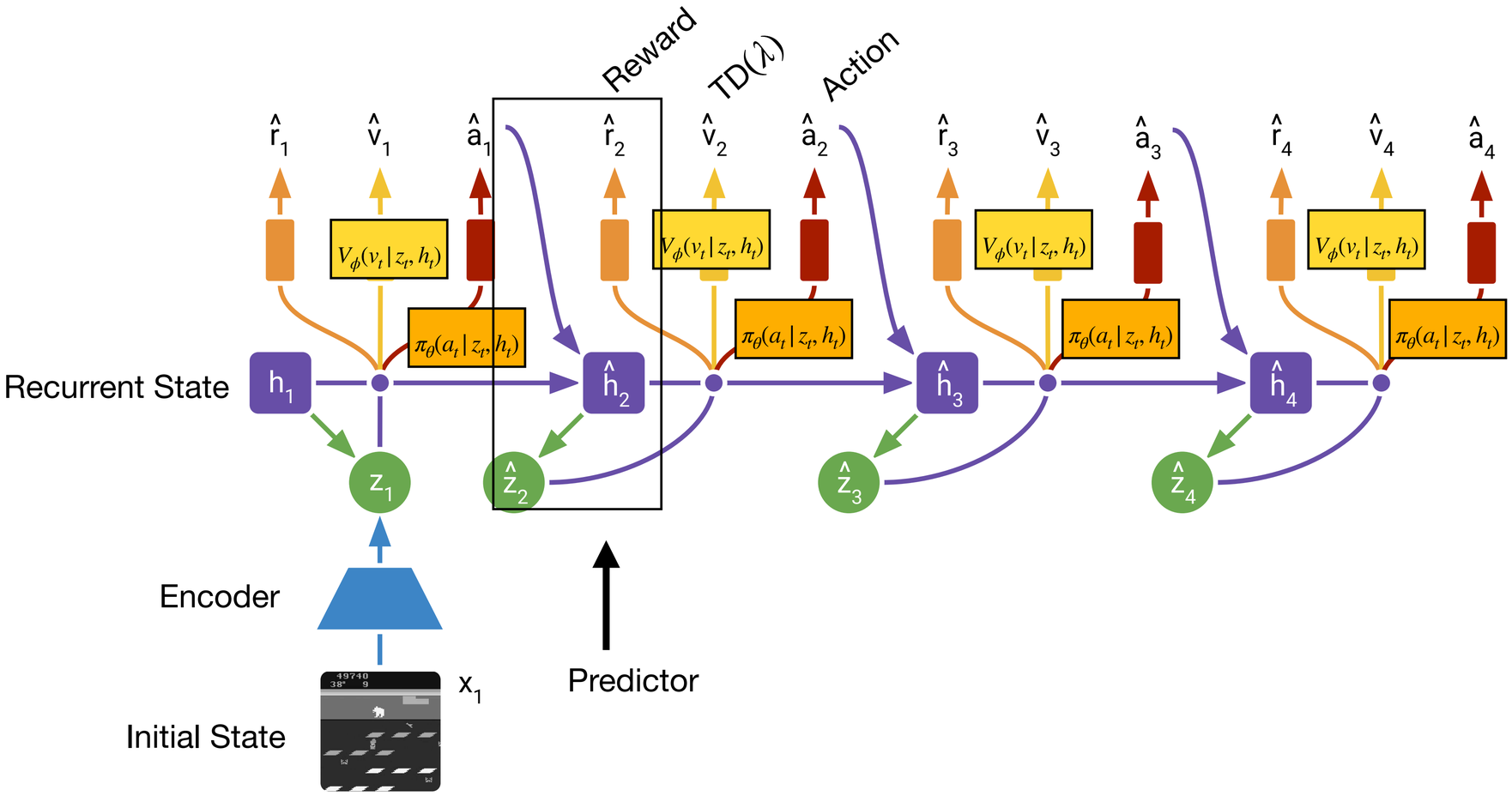
$p_\theta(s_{t+1} \mid s_t, a_t)$

Transition Dynamics

Learn dynamics in lower-dimensional space

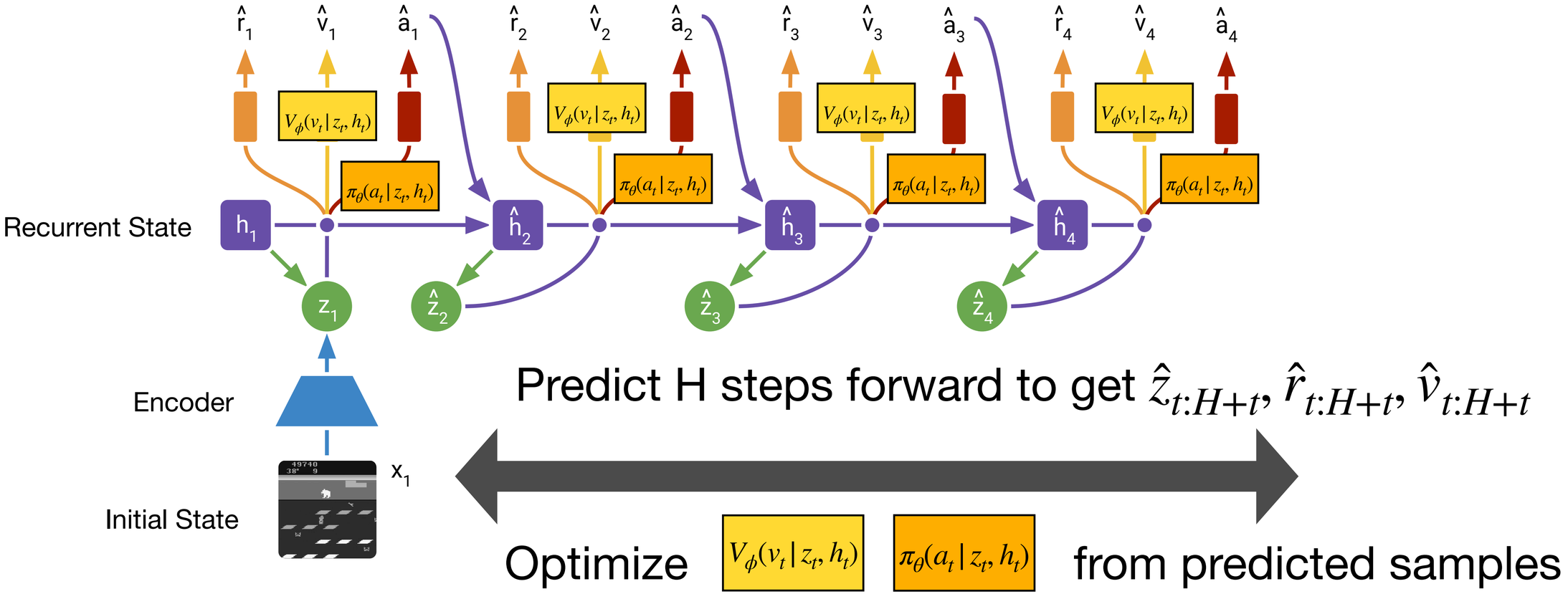


Recurrent State Space Model (RSSM)



Dreamer v2, "Mastering Atari with Discrete World Models"

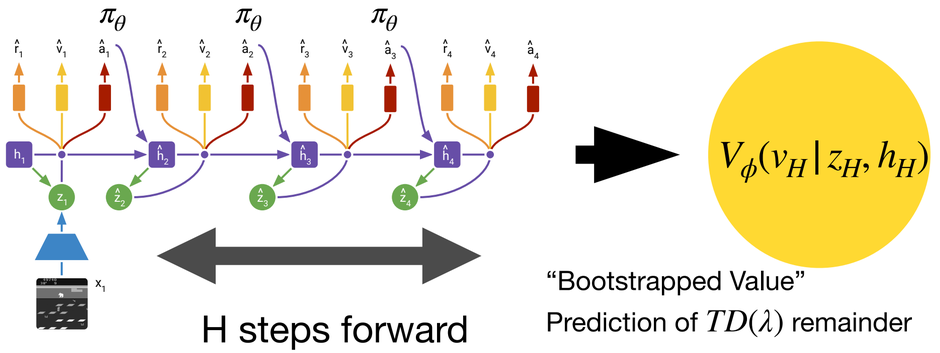
Recurrent State Space Model (RSSM)



Dreamer v2, "Mastering Atari with Discrete World Models"

Behavior Learning / Policy Improvement

- Sample a starting state s_t
- 'Imagine' trajectories $\{(s_k, a_k, r_k)\}_{k=t}^{t+H}$ from s_t using π_θ
- Compute value estimates:



$$V_{\text{TD}(\lambda)}(s_t) = (1-\lambda)\sum_{k=1}^{H}\lambda^{k-1}V(s_t,k) \qquad V(s_t,k) = \sum_{j=k}^{t+k}\gamma^{j-t}r_j + \gamma^{k-t}V_\phi(v_{t+k}\mid z_{t+k}, h_{t+k})$$

- Policy Update: $\max_\theta \mathbb{E}_{\pi_\theta, v_\phi}\left[\sum_{k=t}^{t+H} V_{\text{TD}(\lambda)}(s_k)\right]$

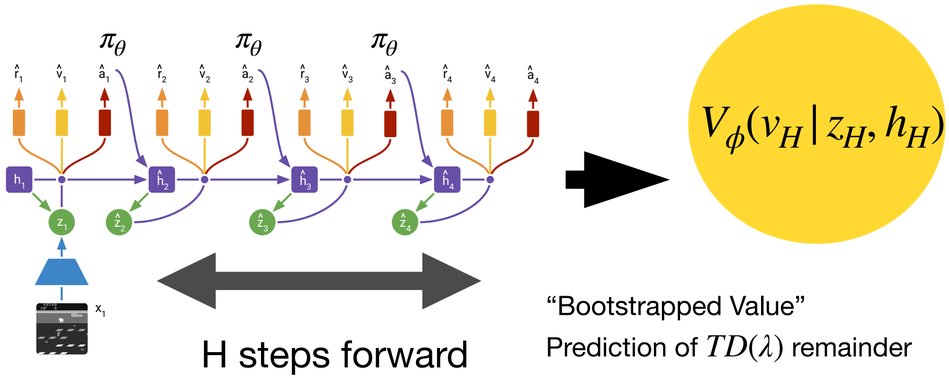
$\pi_\theta(a_t \mid z_t, h_t)$

- Critic / Bootstrap Update: $\min_\phi \mathbb{E}_{\pi_\theta, v_\phi}\left[\sum_{k=t}^{t+H}\frac{1}{2}||V_\phi(s_k) - V_{\text{TD}(\lambda)}(s_k)||^2\right]$

$V_\phi(v_t \mid z_t, h_t)$

Behavior Learning / Policy Improvement

- Sample a starting state s_t
- 'Imagine' trajectories $\{(s_k, a_k, r_k)\}_{k=t}^{t+H}$ from s_t using π_θ
- Compute value estimates:



$$V_{\text{TD}(\lambda)}(s_t) = (1-\lambda)\sum_{k=1}^{H}\lambda^{k-1}V(s_t,k) \qquad V(s_t,k) = \sum_{j=k}^{t+k}\gamma^{j-t}r_j + \gamma^{k-t}V_\phi(v_{t+k}\mid z_{t+k}, h_{t+k})$$

- Policy Update: $\max_\theta \mathbb{E}_{\pi_\theta, v_\phi}\left[\sum_{k=t}^{t+H} V_{\text{TD}(\lambda)}(s_k)\right]$

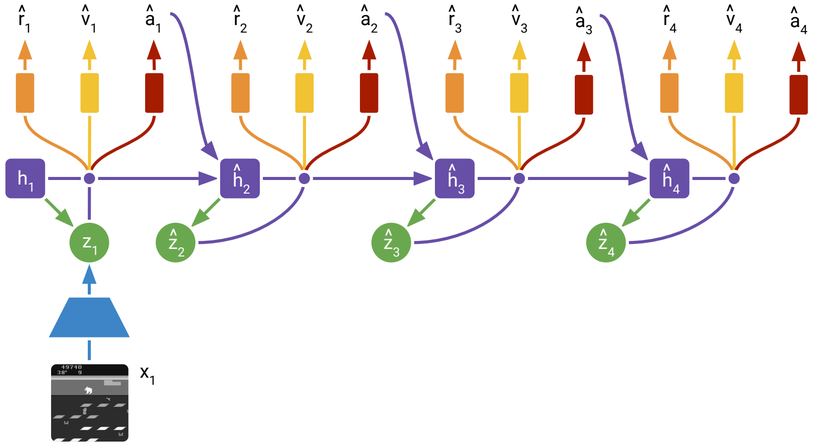
$\pi_\theta(a_t \mid z_t, h_t)$

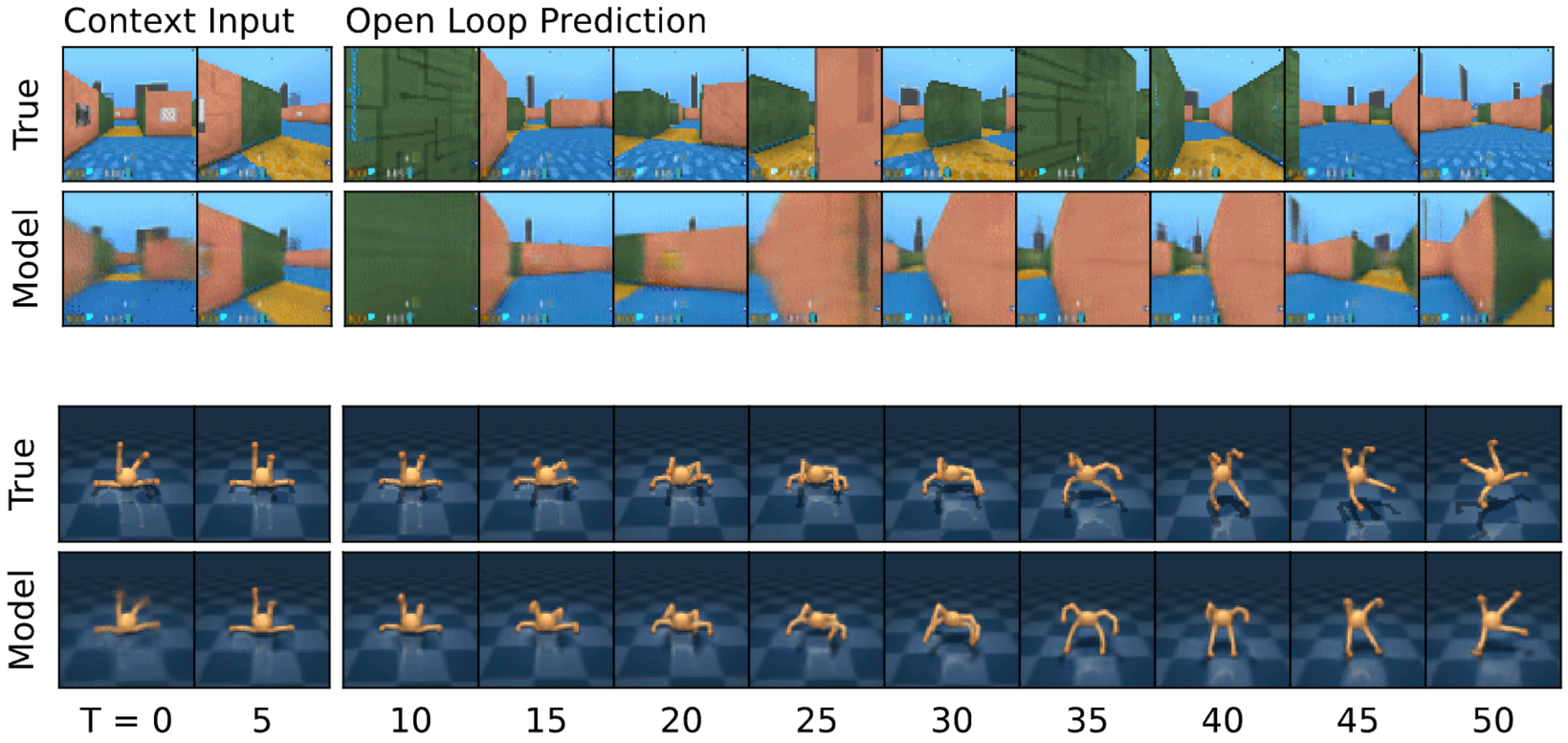
- Critic / Bootstrap Update: $\min_\phi \mathbb{E}_{\pi_\theta, v_\phi}\left[\sum_{k=t}^{t+H}\frac{1}{2}||V_\phi(s_k) - V_{\text{TD}(\lambda)}(s_k)||^2\right]$

$V_\phi(v_t \mid z_t, h_t)$

- Use any method we've discussed so far.
- Dreamer uses REINFORCE to train policy
  - W/ batch-wise entropy regularization

Need accurate predictor





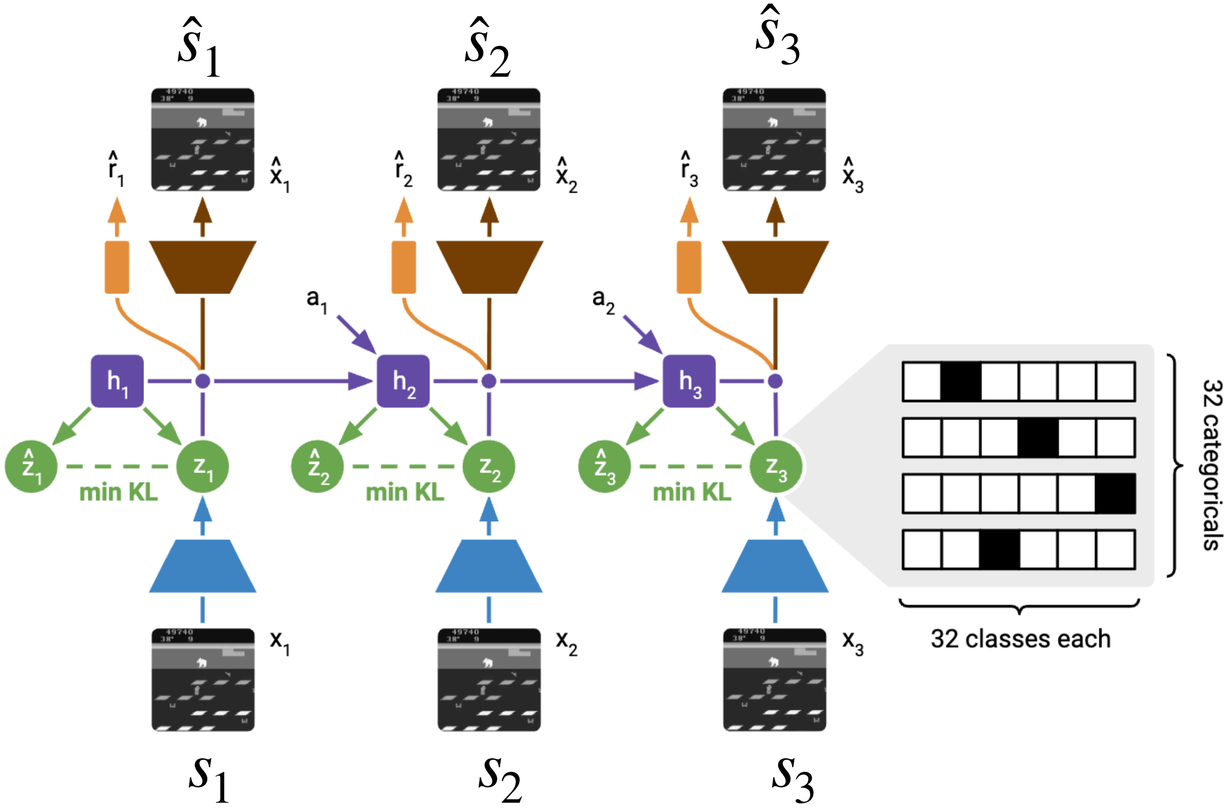
Dreamer v3

How do we build a predictor?

World Model Training

- Given representative trajectory $\{(s_k, a_k, r_k)\}_{k=t}^{t+H}$ (assume collected with ϵ-Greedy on current policy)
- Learn a stochastic model to generate it using **only** lower dimensional features

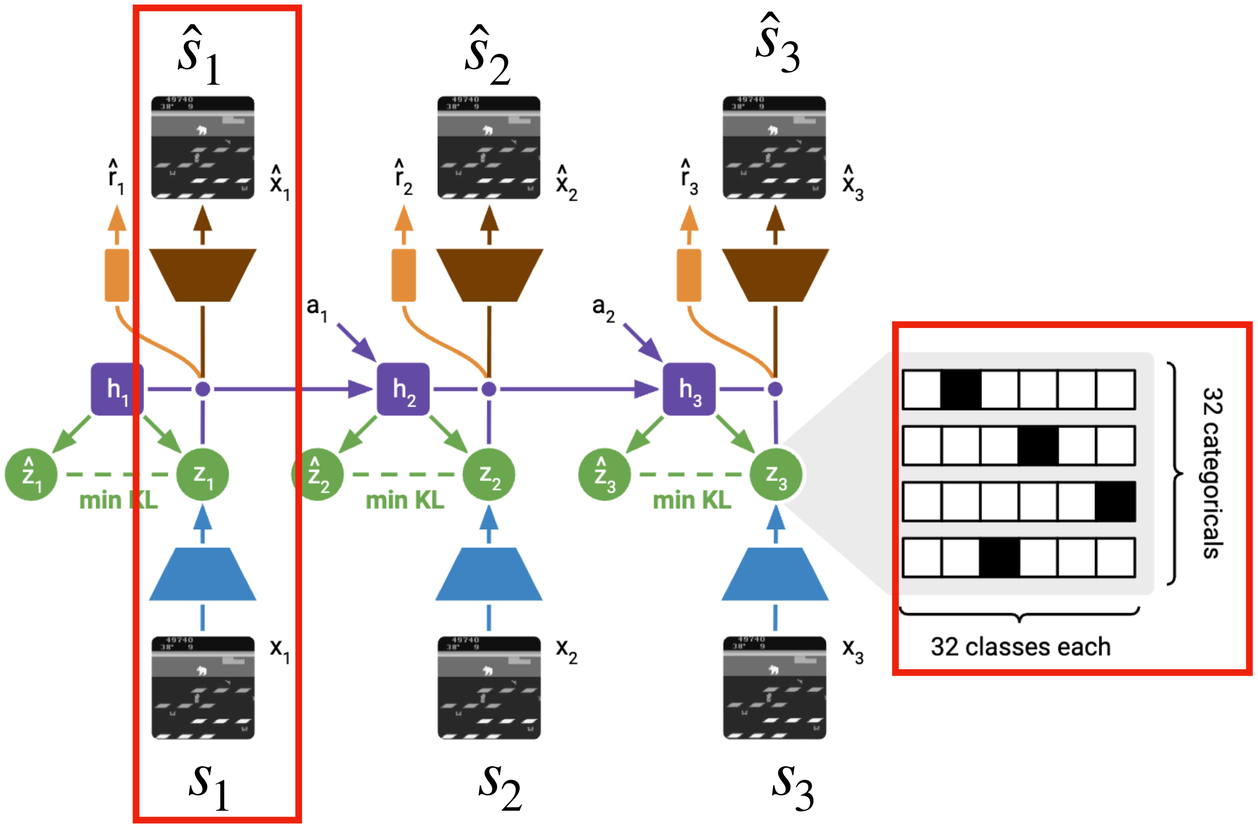
- Multiobjective:
  - Reconstruction from lower dimension:
    - $d(s_t, \hat{s}_t)$ small, learn good state encoder
    - $d(r_t, \hat{r}_t)$ small, predict rewards from state-actions
  - Predict state from history
    - $d(z_t, \hat{z}_t)$ small
    - Predict next encoder without real system observation



Building the RSSM

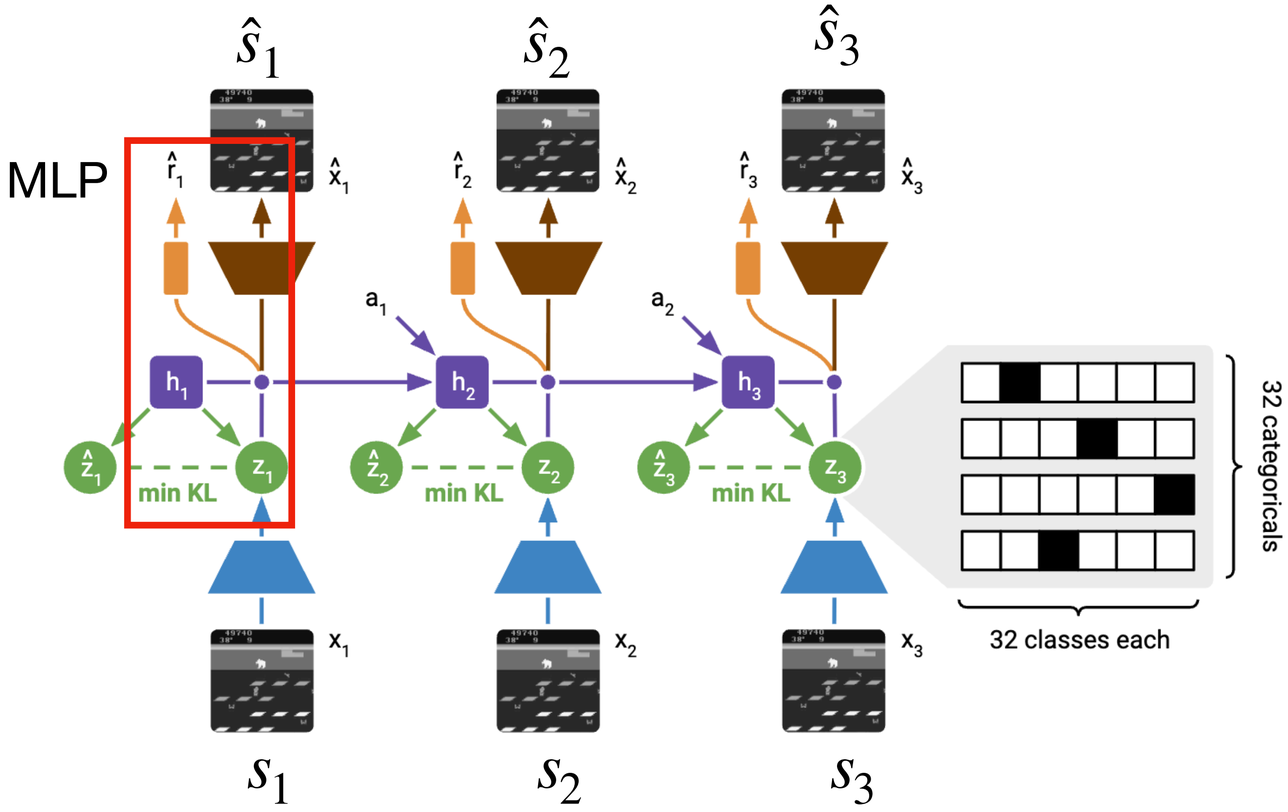
- Enforce z_t is from categorical distribution
- Encoder: $z_t \sim q_\theta(z_t \mid h_t, s_t)$
- Decoder: $\hat{s}_t \sim p_\theta(\hat{s}_t \mid h_t, z_t)$

Sequential Discrete Variational Autoencoder (VQ-VAE)



Building the RSSM

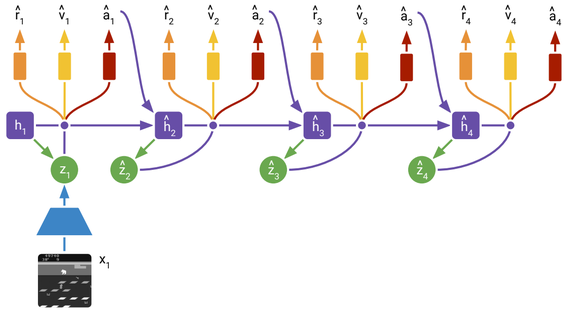
- Reward Model: $r_t \sim p_\theta(r_t \mid h_t, z_t)$

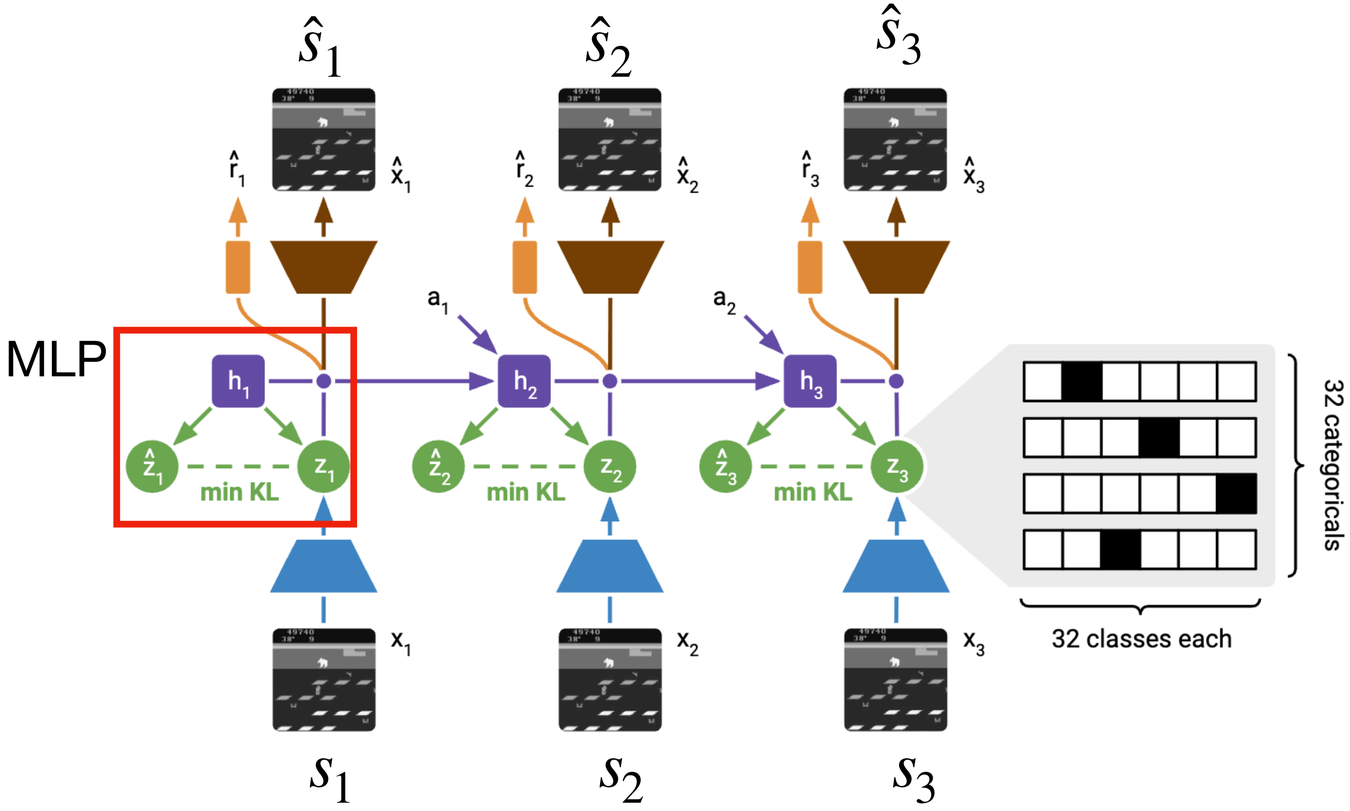


Building the RSSM

- Transition Predictor: $\hat{z}_t \sim p_\theta(\hat{z}_t \mid h_t)$

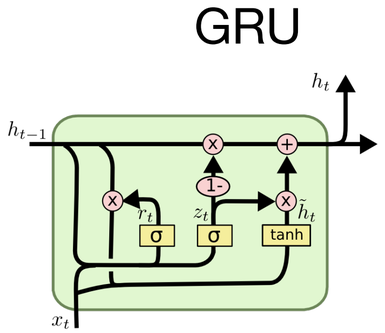
“Replacement for encoder during imagined rollouts”





Building the RSSM

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

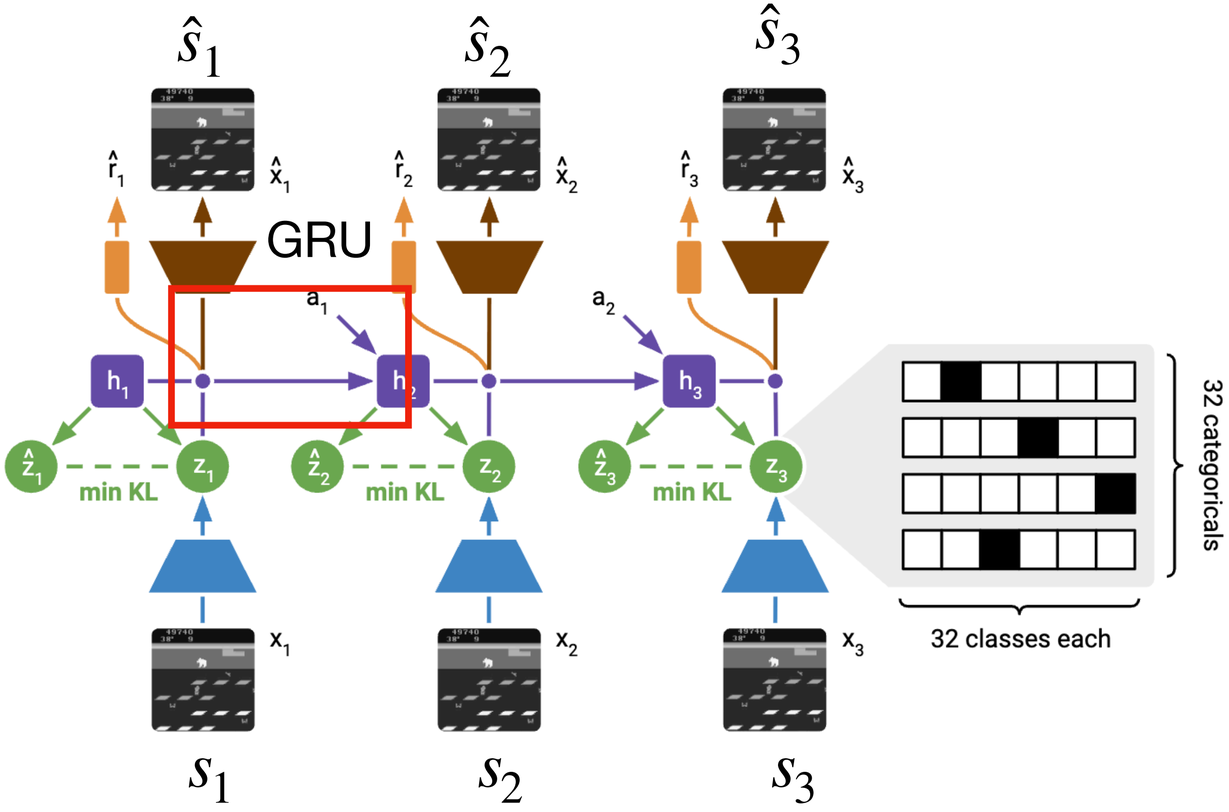


$$x_t = [z_{t-1}, a_{t-1}]$$
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

How much of the past should the current rely on

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

How much the state should reflect current

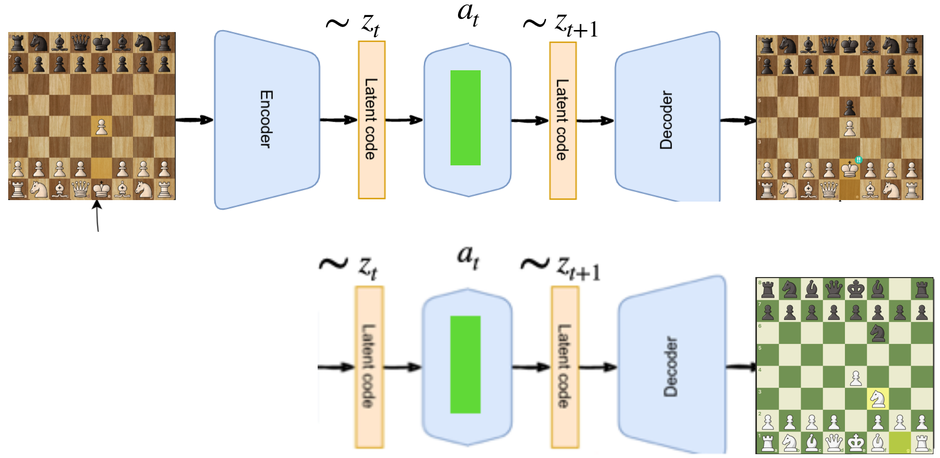


Building the generative RSSM

- Enforce z_t is from categorical distribution
- Encoder: $z_t \sim q_\theta(z_t \mid h_t, s_t)$
- Decoder: $\hat{s}_t \sim p_\theta(\hat{s}_t \mid h_t, z_t)$
- Reward Model: $r_t \sim p_\theta(r_t \mid h_t, z_t)$
- Transition Predictor: $\hat{z}_t \sim p_\theta(\hat{z}_t \mid h_t)$

These are probabilistic models

Randomness from sampling latent $z_t, \hat{z}_t$



Why?

- Capture uncertain dynamics (don't know opponents move or inherent randomness)
- Capture uncertainty in current state from partial observations
- Pragmatically: Variational Autoencoders are effective representative models
- See: MuZero for a deterministic latent space model

World Model Training

- Pull trajectories from buffer (of interactions with real environment)
- All components share parameters
- Minimize:

$$\mathscr{L}(\phi) \doteq \mathbb{E}_{q_\phi(z_{1:T}|a_{1:T},s_{1:T})}\left[\sum_{t=1}^{T} \underbrace{-\beta_1 \ln p_\phi(s_t \mid h_t, z_t)}_{\text{state log loss}} + \underbrace{-\beta_2 \ln p_\phi(r_t \mid h_t, z_t)}_{\text{reward log loss}} + \underbrace{\beta_3 \,\mathrm{KL}\left[q_\phi(z_t \mid h_t, s_t) \parallel p_\phi(\hat{z}_t \mid h_t)\right]}_{\text{KL loss}}\right].$$

KL Term does not work well as written

- Minimize: $$\mathscr{L}(\phi) \doteq \mathbb{E}_{q_\phi(z_{1:T}|a_{1:T},s_{1:T})}\left[\sum_{t=1}^{T} \underbrace{-\beta_1 \ln p_\phi(s_t \mid h_t, z_t)}_{\text{state log loss}} + \underbrace{-\beta_2 \ln p_\phi(r_t \mid h_t, z_t)}_{\text{reward log loss}} + \underbrace{\beta_3 \mathrm{KL}\left[q_\phi(z_t \mid h_t, s_t) \parallel p_\phi(\hat{z}_t \mid h_t)\right]}_{\text{KL loss}}\right].$$
- Problems from multi-objective optimization:
  - An accurate $p_\phi(\hat{z}_t \mid h_t)$ is more important than an accurate $q_\phi(z_t \mid h_t, s_t)$
    - Solution: Split the KL term with separate scaling coefficients ($B_{3_1} > B_{3_2}, B_{3_1} + B_{3_2} = 1,$)
    - $$\mathrm{KL}\left[q_\phi(z_t \mid h_t, s_t) \parallel p_\phi(\hat{z}_t \mid h_t)\right] = B_{3_1}\mathrm{KL}\left[\mathrm{sg}[q_\phi(z_t \mid h_t, s_t)] \parallel p_\phi(\hat{z}_t \mid h_t)\right] + B_{3_2}\mathrm{KL}\left[q_\phi(z_t \mid h_t, s_t) \parallel \mathrm{sg}[p_\phi(\hat{z}_t \mid h_t)]\right]$$
      - “sg” is the stop gradient operator. “detach” in PyTorch.

DreamerV2

KL Term does not work well as written

- Minimize: $$\mathscr{L}(\phi) \doteq \mathbb{E}_{q_\phi(z_{1:T}|a_{1:T},s_{1:T})}\left[\sum_{t=1}^{T} \underbrace{-\beta_1 \ln p_\phi(s_t \mid h_t, z_t)}_{\text{state log loss}} + \underbrace{-\beta_2 \ln p_\phi(r_t \mid h_t, z_t)}_{\text{reward log loss}} \underbrace{-\beta_3 \operatorname{KL}\left[q_\phi(z_t \mid h_t, s_t) \parallel p_\phi(\hat{z}_t \mid h_t)\right]}_{\text{KL loss}}\right].$$
- Problems from multi-objective optimization:
  - Degenerate case (z_t easy to predict) not informative.
    - Focus should be on state and reward recovery once predicted embedding is “good enough”
  - Solution: clip minimum, allow “free unoptimized bits of difference between them”
  - $B_{3_1} \min[c, \operatorname{KL}\left[\text{sg}[q_\phi(z_t \mid h_t, s_t)] \parallel p_\phi(\hat{z}_t \mid h_t)\right]] + B_{3_2} \min[c, \operatorname{KL}\left[q_\phi(z_t \mid h_t, s_t) \parallel \text{sg}[p_\phi(\hat{z}_t \mid h_t)]\right]]$

DreamerV3

Putting it all together

- Collect transitions from real env with epsilon-greedy
- Train predictive generative world model on real transitions
- Train policy and value models on generated transitions
- Repeat

Results (DreamerV3)

- Surprisingly sample efficient (in terms of interactions with real environment)

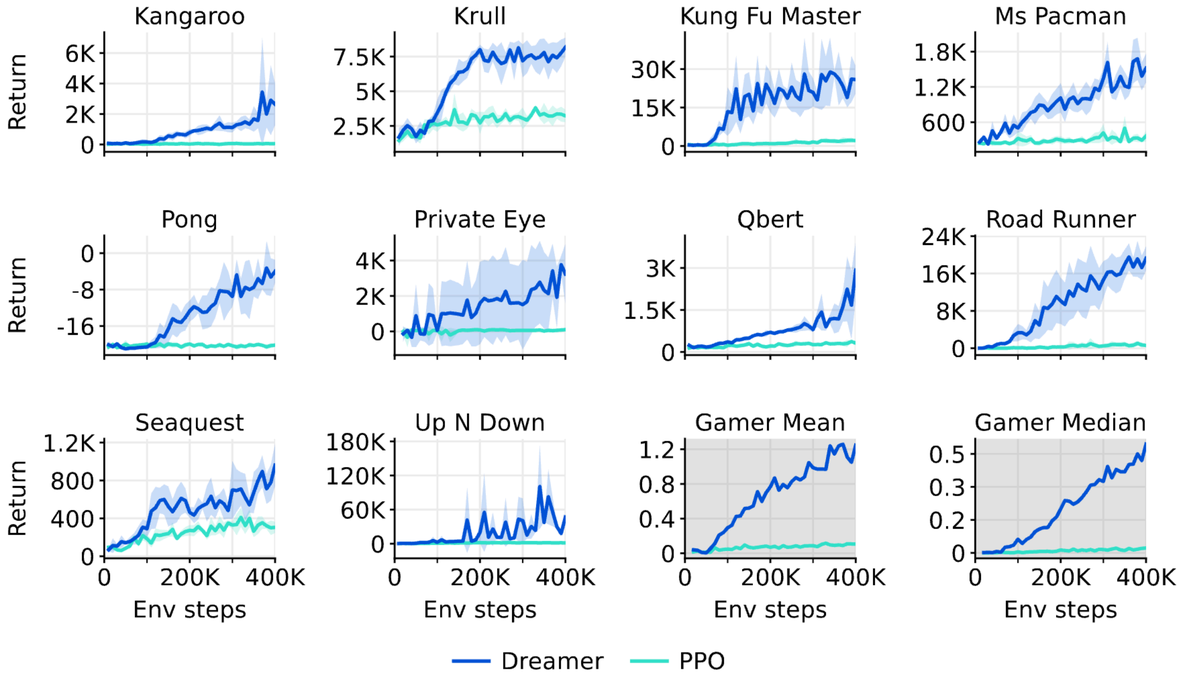


Figure 13: Atari100k learning curves.

Results (DreamerV3)

- Surprisingly sample efficient (in terms of interactions with real environment)

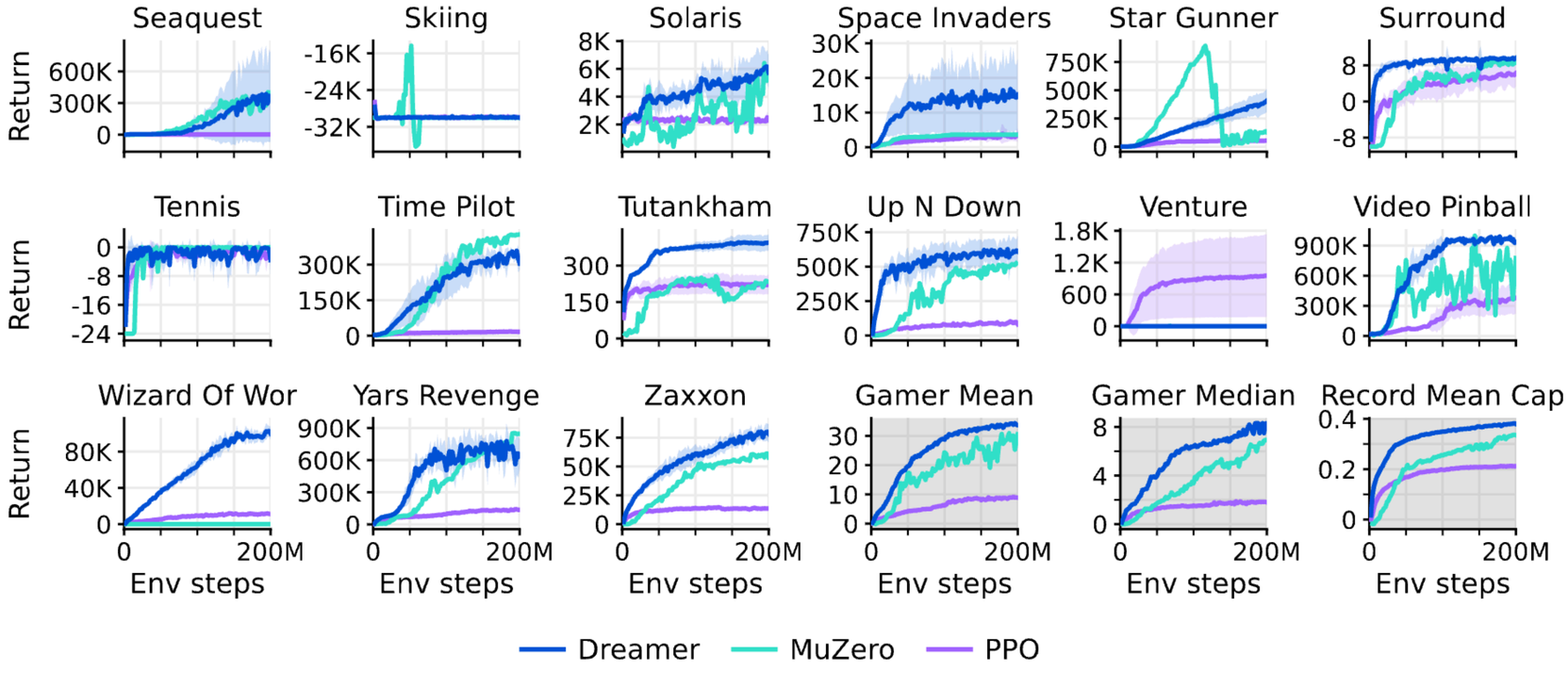


Figure 10: Atari learning curves.

Results (DreamerV3)

each room has a different layout, treasure, and enemies, and some rooms have special features, such as moving walls

- Surprisingly sample efficient (in terms of interactions with real environment)

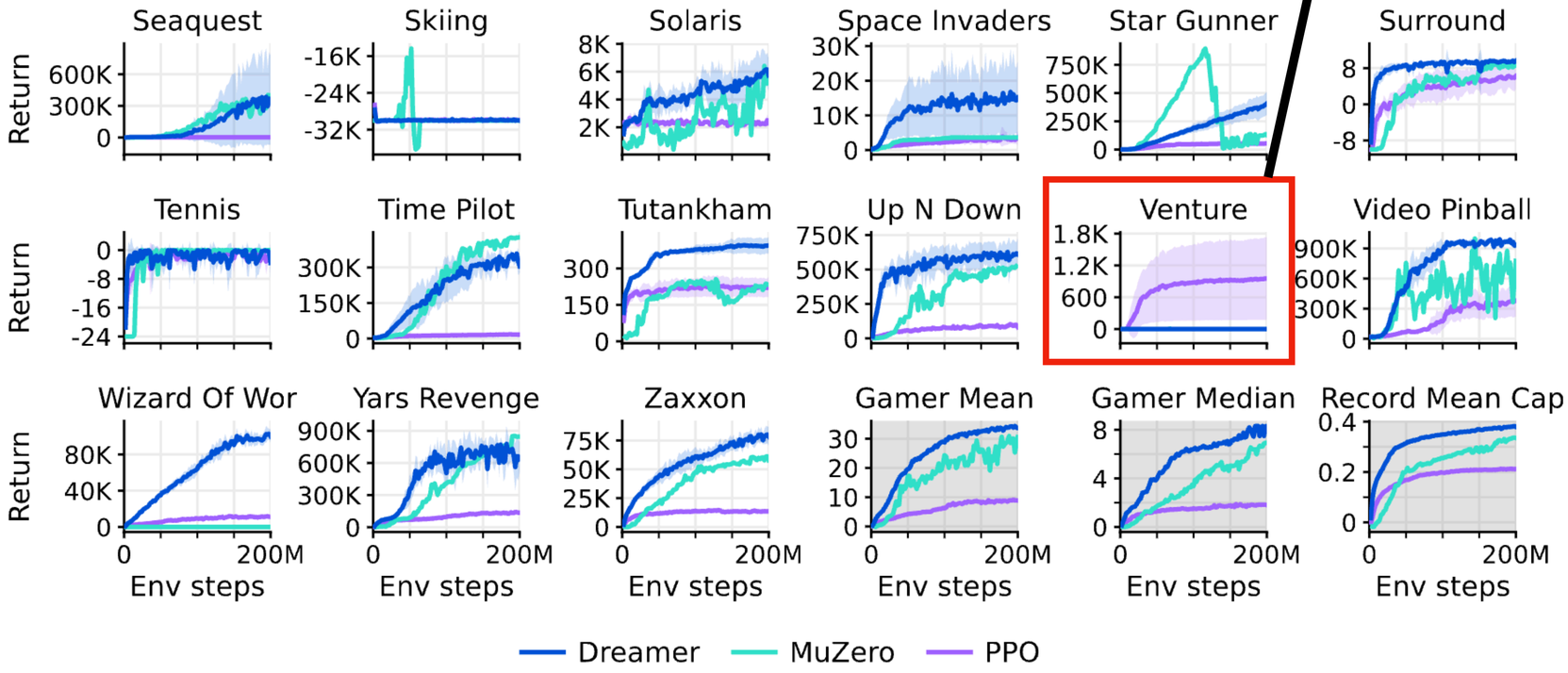
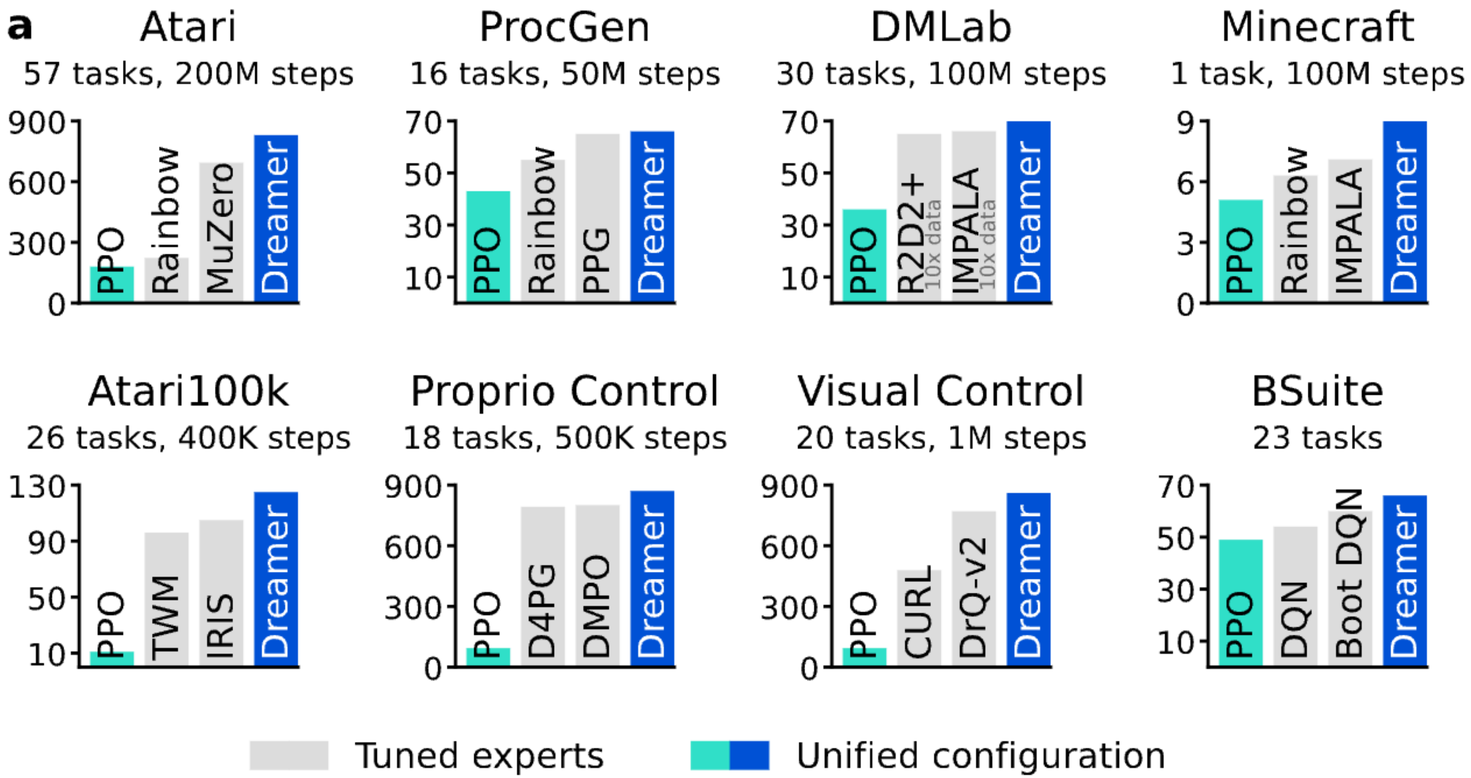


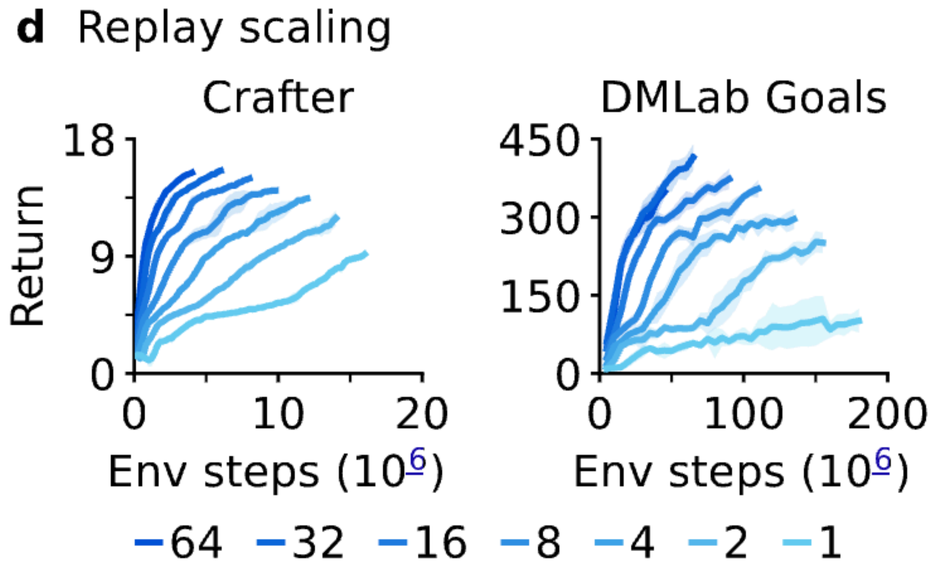
Figure 10: Atari learning curves.

Results (DreamerV3)

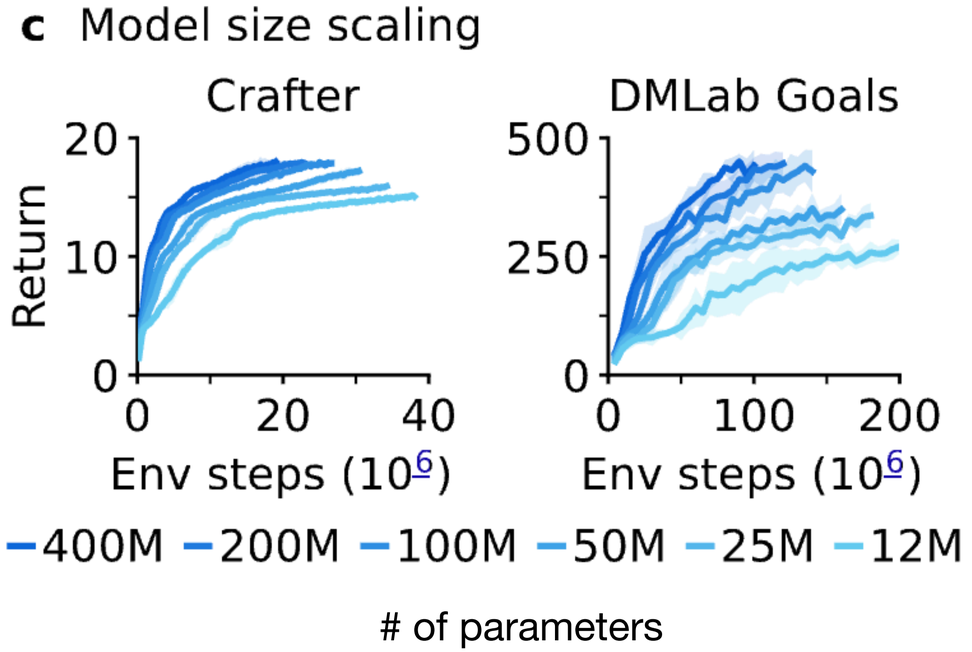
- Claimed effective across a range of tasks without hyperparameter search



Results (DreamerV3)



- Imagination vs Buffer ratio
  - X-axis: Real steps to train prediction
  - Lines: Ratio of imagined steps to train policy per real step to train predictor generative model
  - Without a reference A2C or PPO on this plot it is hard to compare sample efficiency to not using a generative model



- Models are **LARGE**
- But performance increases as parameters increase for same setup

Crafter is a synthetic 2D Minecraft

DMLab is 3D Maze navigation