Reinforcement Learning

CS 59300: RL1

September 30, 2025

Joseph Campbell

Department of Computer Science

PURDUE UNIVERSITY®

Discussion on GAE

Controlling the bias/variance trade-off

Consider three cases:

1. 1-step TD. Lowest variance, highest bias.

$$A^{(1)}(s_t, a_t) = r_{t+1} + \gamma V(s_t) - V(s_t)$$

2. N-step TD. Intermediate variance, intermediate bias.

$$A^{(n)}(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^n V(s_{t+n}) - V(s_t)$$

3. Monte Carlo (w/ baseline). Highest variance, lowest bias

$$A^{(\infty)}(\mathrm{s_t}, \mathrm{a_t}) = G_t - V(s_t)$$

Generalized advantage estimation

$$A^{\mathrm{GAE}(\gamma,\lambda)} = (1-\lambda)(A^{(1)} + \lambda A^{(2)} + \lambda^2 A^{(3)} + \cdots)$$

This is an exponentially-weighted average over n-step advantages

Consider $\lambda = 0.1$

$$A^{\mathrm{GAE}(\gamma,\lambda)} = (0.9 * A^{(1)} + 0.09 * A^{(2)} + 0.009 * A^{(3)} + \cdots)$$

Consider $\lambda = 0.9$

$$A^{\mathrm{GAE}(\gamma,\lambda)} = (0.1 * A^{(1)} + 0.09 * A^{(2)} + 0.081 * A^{(3)} + \cdots)$$

Schulman et al, High-Dimensional Continuous Control Using Generalized Advantage Estimation, 2016.

Generalized advantage estimation

Larger λ means we place more weight on the later terms.

Smaller λ means weight is concentrated on earlier terms.

This s

Consider $\lambda = 0.1$

$$A^{\mathrm{GAE}(\gamma,\lambda)} = (0.9 * A^{(1)} + 0.09 * A^{(2)} + 0.009 * A^{(3)} + \cdots)$$

Consider $\lambda = 0.9$

$$A^{\mathrm{GAE}(\gamma,\lambda)} = (0.1 * A^{(1)} + 0.09 * A^{(2)} + 0.081 * A^{(3)} + \cdots)$$

Schulman et al, High-Dimensional Continuous Control Using Generalized Advantage Estimation, 2016.

Generalized advantage estimation

Later advantage terms have a more heavily discounted bootstrap

$$A^{(n)}(s_t, a_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^n V(s_{t+n}) - V(s_t)$$

Larger n = more realized returns + discount on bootstrap = less bias

- GAE averages over all n-step advantages
- Lambda controls how much we weight each n-step term, thus bias
- In practice n-step is only computed over a truncated rollout
- Discount factor determines TD target, lambda determines estimator

Discussion on PPO

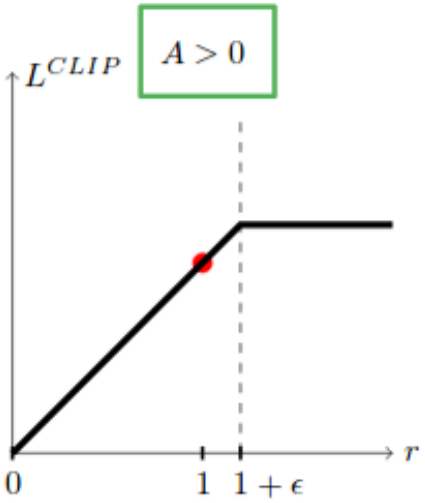
PPO Clipped objective

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}[\min(r(\theta)A(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A(s,a)]$$

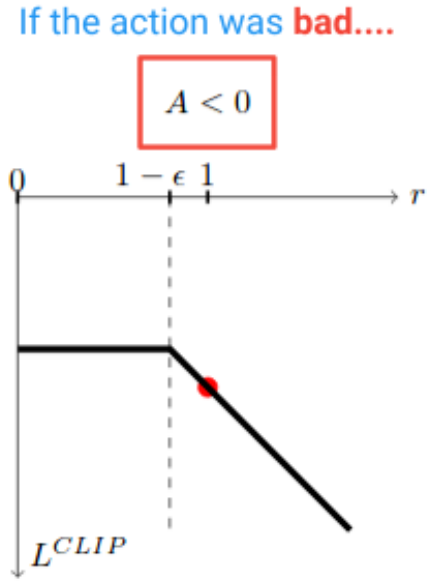
TRPO unclipped objective

Clipped objective

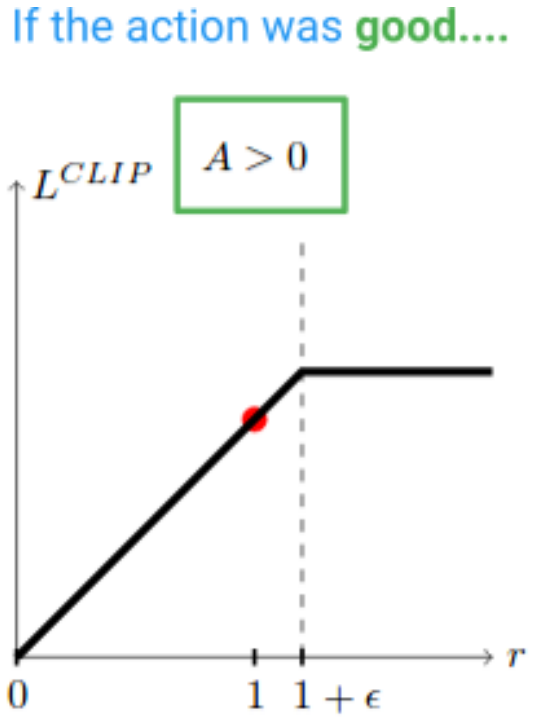
If the action was **good**....

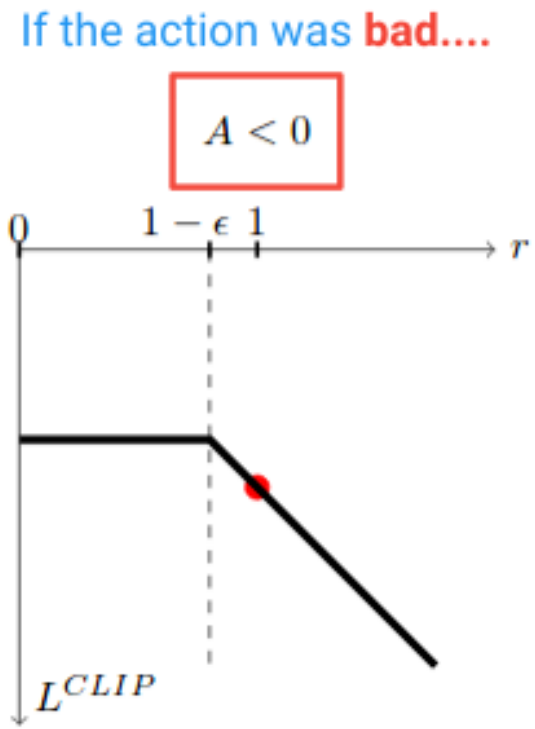


If the action was **bad**....



$$clip(f(x), a, b) = \begin{cases} f(x) & a \le f(x) \le b \\ a & f(x) \le a \\ b & f(x) \ge b \end{cases}$$

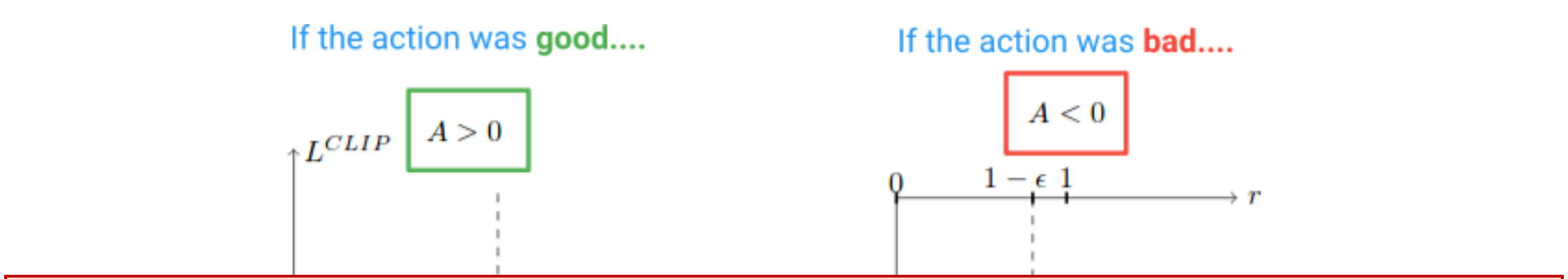




Clipping serves to flatten the gradient when we take a good step

- If action was good and we have increased probability
- If action was bad and we have decreased probability

We have already stepped far enough, do not continue stepping

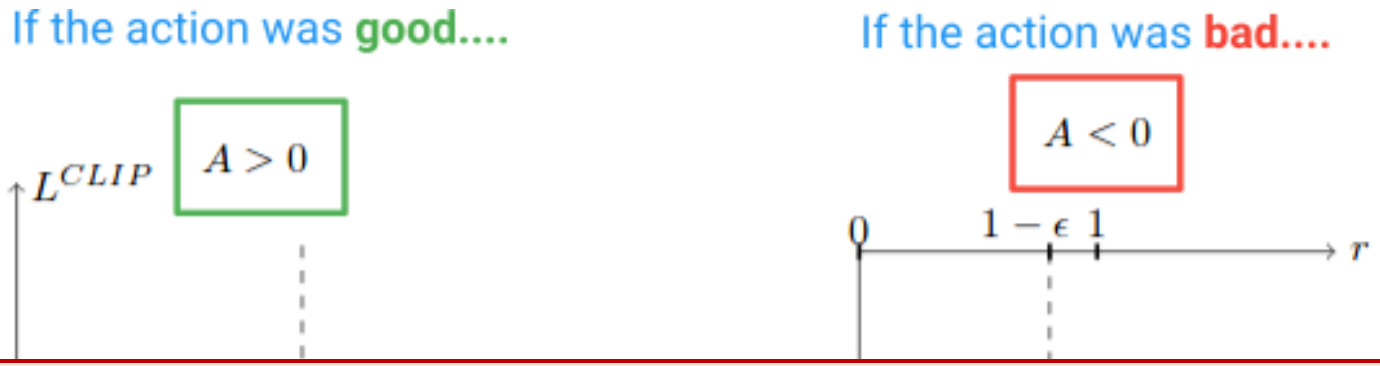


Why does clipping “flatten the gradient”?

Clipping serves to flatten the gradient when we take a good step

- If action was good and we have increased probability
- If action was bad and we have decreased probability

We have already stepped far enough, do not continue stepping



Why does clipping "flatten the gradient"?

Because the derivative is 0 beyond the ϵ-threshold.
(And clipping gives a constant with a 0 partial derivative w.r.t. θ)

Clipping serves to flatten the gradient when we take a good step

- If action was good and we have increased probability
- If action was bad and we have decreased probability

We have already stepped far enough, do not continue stepping

So why the minimum?

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}[\min(r(\theta)A(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A(s,a)]$$

This occurs when we have taken a bad step

- If action was bad but we have increased probability
- If action was good but we have decreased probability

Even if we have exceeded the ϵ-threshold, do not flatten the gradient because we need to be able to step backwards

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

PPO performs multiple minibatch updates

- On first update, $r(\theta) = 1$ because we haven't stepped yet
- On subsequent updates $r(\theta)$ is computed with respect to original θ

Today's lecture

1. High-dimensional "pixel-space" models

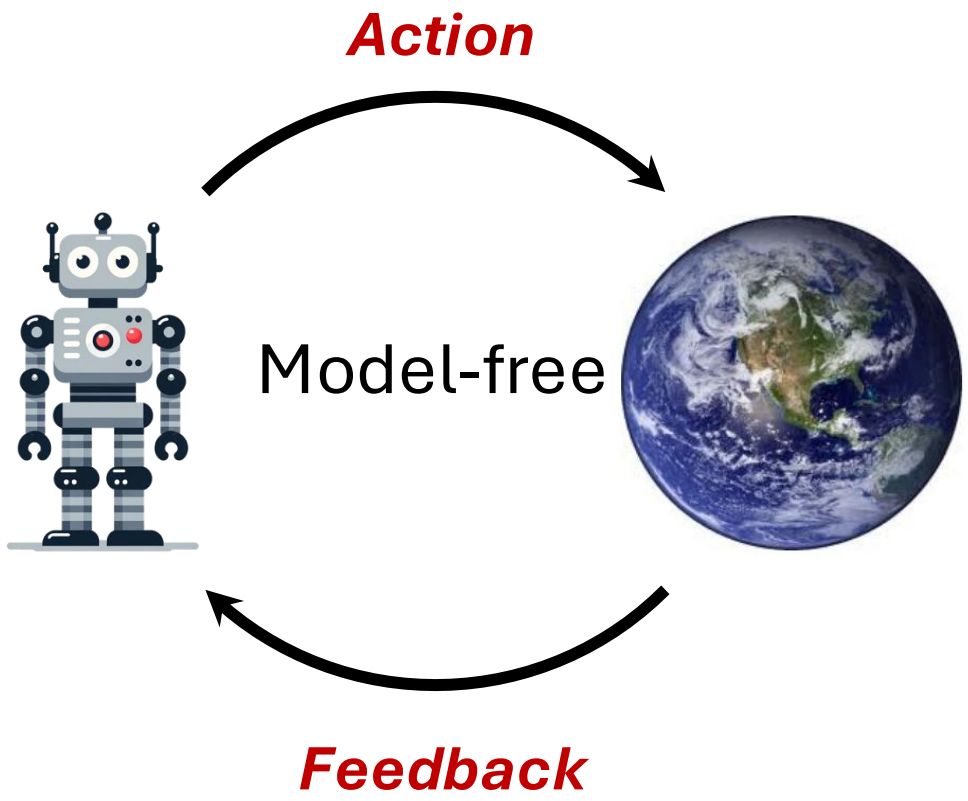
2. Low-dimensional "latent-space" models

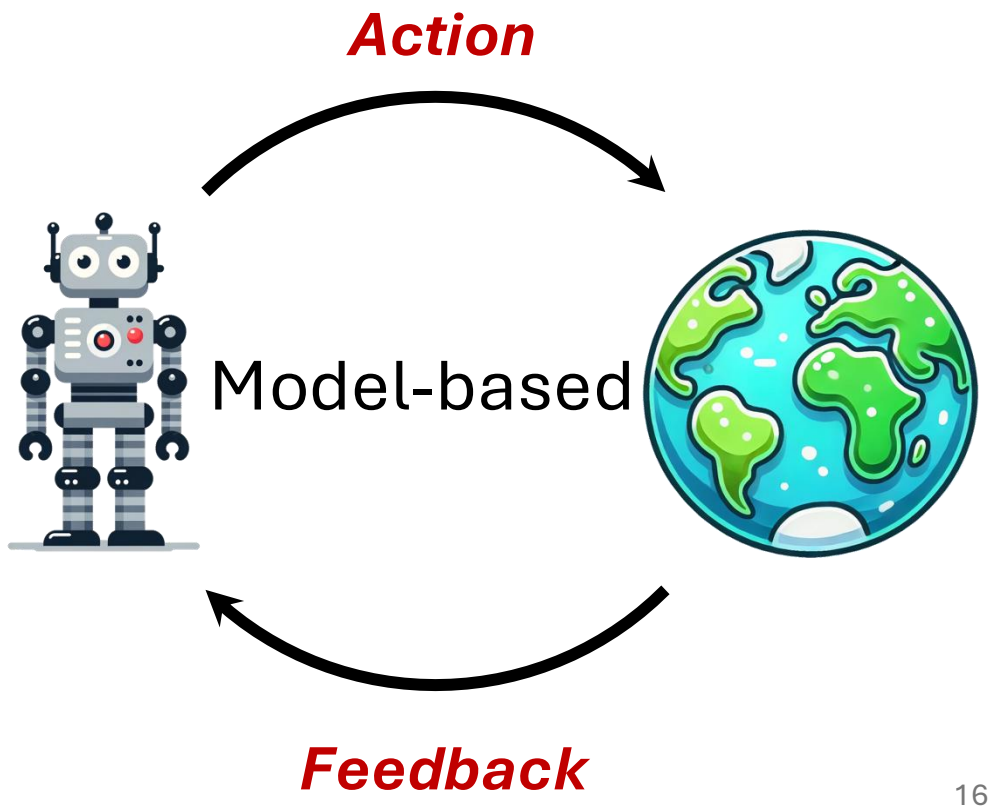
Some content inspired by Katerina Fragkiadaki's CMU 10-403

High-dimensional “pixel-space” models

Recap: Model-based reinforcement learning

Idea: In MBRL we learn a model of the environment and reward and use this to perform reinforcement learning





Recap: Why do we care? What are the benefits?

- In some cases, it is more sample-efficient to learn a model of the environment than to directly learn a policy in the environment
  - Especially important if interactions are safety-critical!
- Once a model is learned, it can be re-used many times
- We can plan ahead by simulating what will happen before acting
- Can be used to guide exploration and representation learning
  - Uncertain what next state looks like? Then we haven't been there enough!

Recap: What are the drawbacks?

- The learned model may not be accurate (e.g. high dimensions)
  - Particularly dangerous as errors propagate over time!
- If dynamics are hard to predict, may result in under-modeling
  - Faster learning at beginning, but non-optimal asymptotic performance
- If model is overfit, may lead to policy taking advantage of mistakes
  - Overfitting can induce spurious correlations in resulting policies
- Capturing stochasticity in future events is hard

Another approach: explicitly planning ahead

Value functions implicitly encode dynamics with respect to a policy

- What is the expected future reward if I follow my policy?

But if we have a model of the environment, we can explicitly simulate what will happen!

This means we can decouple dynamics from rewards

- Allows for planning optimal actions

Recap: Model predictive control

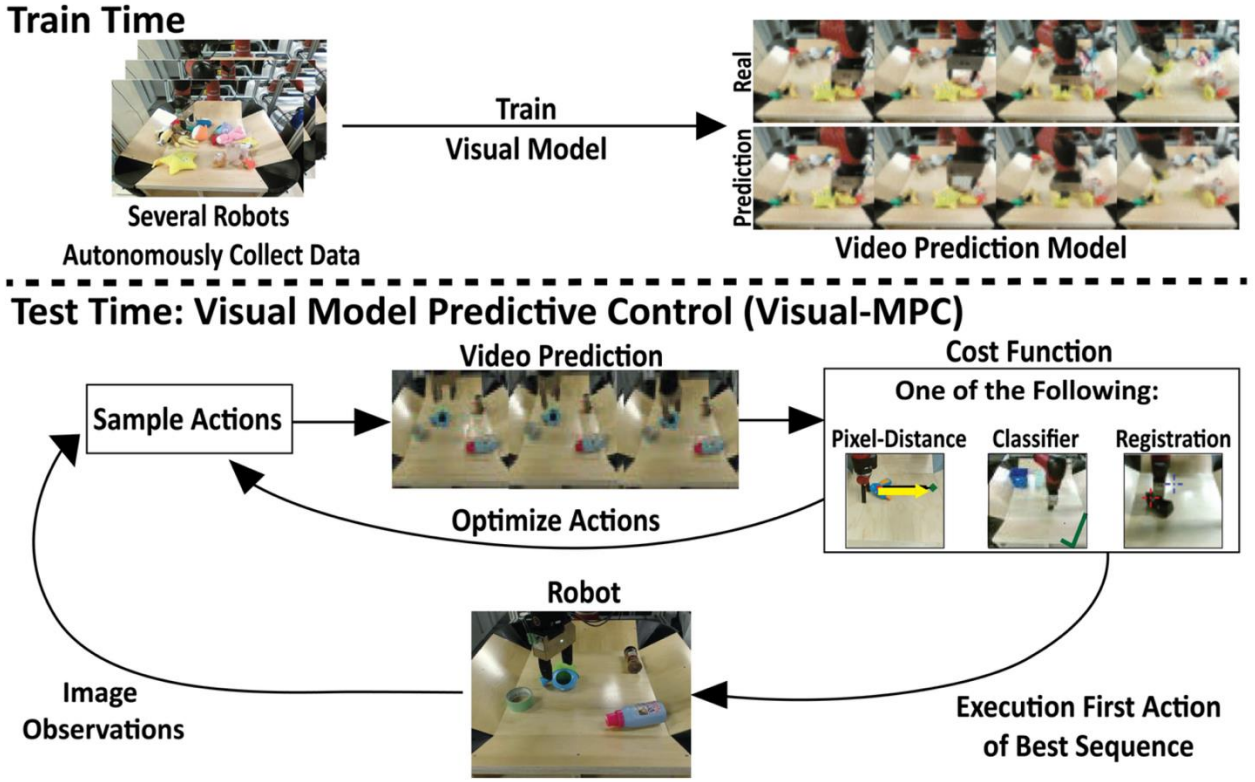
Definition: feedback control method that optimizes future behavior over finite time horizon

Three steps:

1. Predict future states using **system model**
2. Solve optimization problem to **minimize cost** over future states
3. Apply only the **first action** and repeat process next step

Deep visual foresight

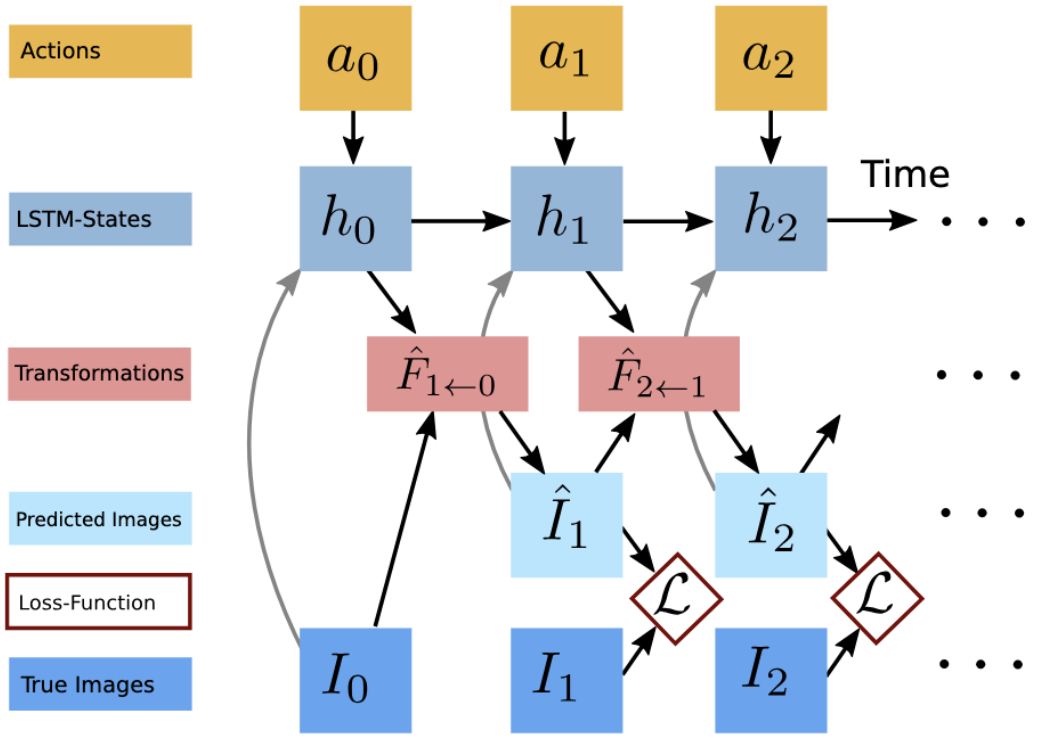
Idea: learn a video prediction model which predicts future image states and use this for model predictive control



Ebert et al, Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control, 2018.

Video prediction model

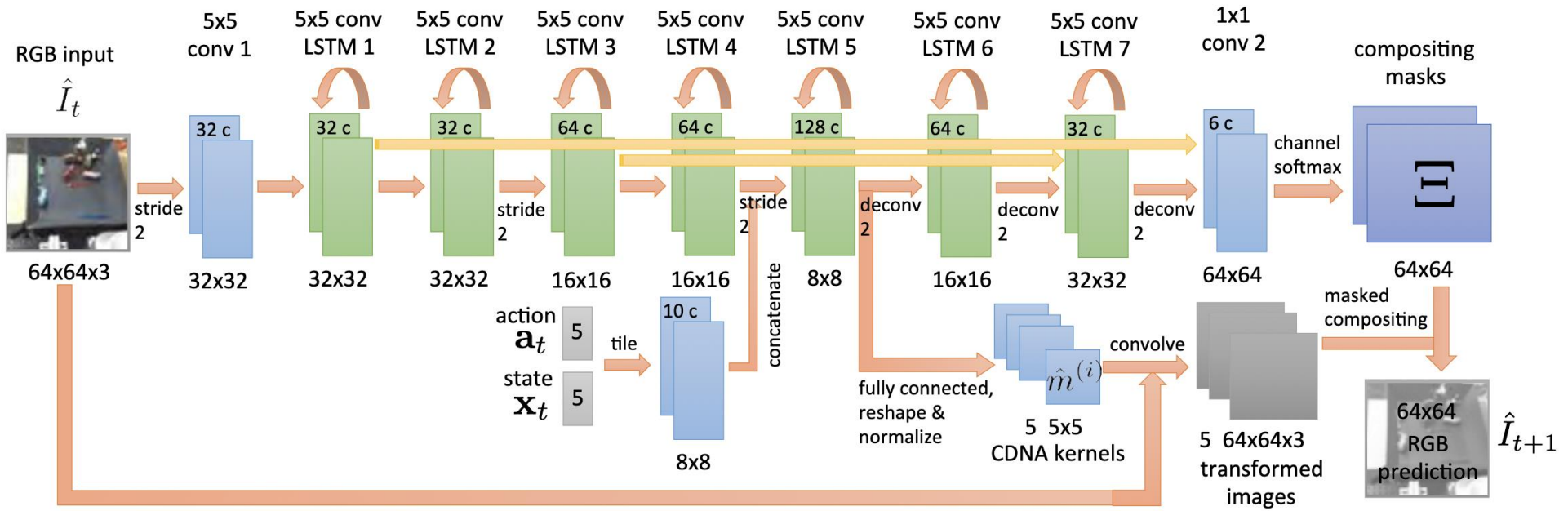
Given the current RGB image and a sequence of actions, predict the corresponding sequence of future images



Video prediction model

Trained using supervised learning

- Dataset consists of task demonstrations (~50k in this paper)
- Chicken-and-egg problem



Visual model predictive control

Use model predictive control to choose the sequence of actions that maximize the probability of achieving the desired goal

- Goal in this work: pushing objects to desired location
- Note: this is planning not learning!

MPC in a nutshell: use (video) model to observe effect of actions such that chosen actions are optimal over a finite horizon

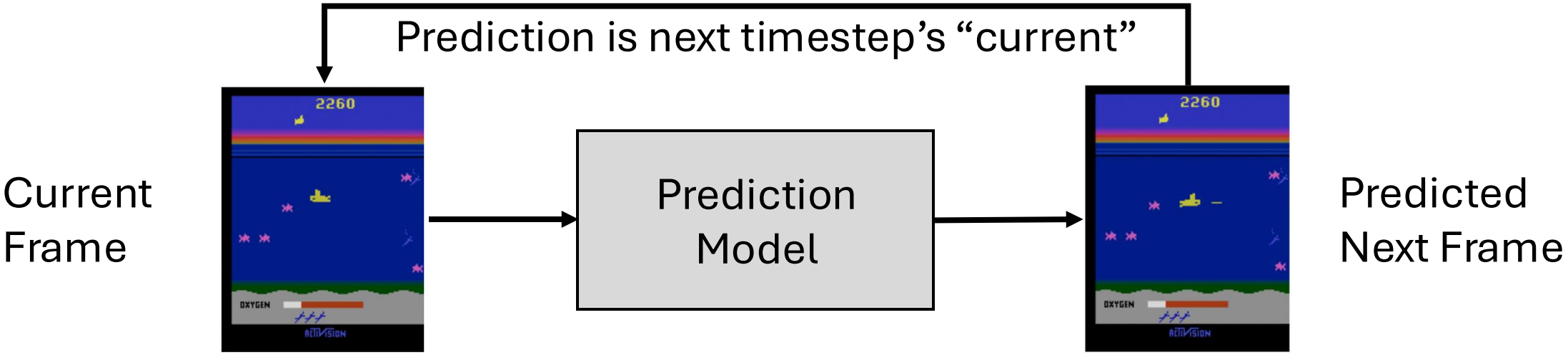
Since the policy is goal-conditioned and the prediction is in pixel-space, no need to learn a reward function!

https://youtu.be/6k7GHG4IUCY?si=04p3OqTHHF3fUCLI&t=33

Accounting for distribution shift

Supervised prediction models make errors

What happens if the predicted next frame has errors?



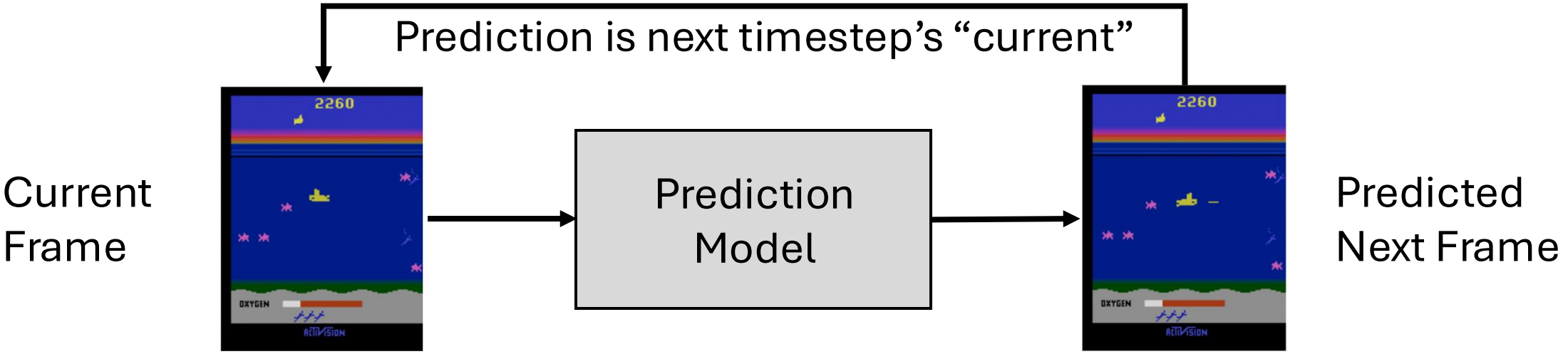
Oh et al, Action-Conditional Video Prediction using Deep Networks in Atari Games, 2015.

Accounting for distribution shift

Supervised prediction models make errors

What happens if the predicted next frame has errors?

If the model has not seen errors during training, it will perform poorly!



Oh et al, Action-Conditional Video Prediction using Deep Networks in Atari Games, 2015.

Training models over their own outputs

Assume k-step prediction horizon

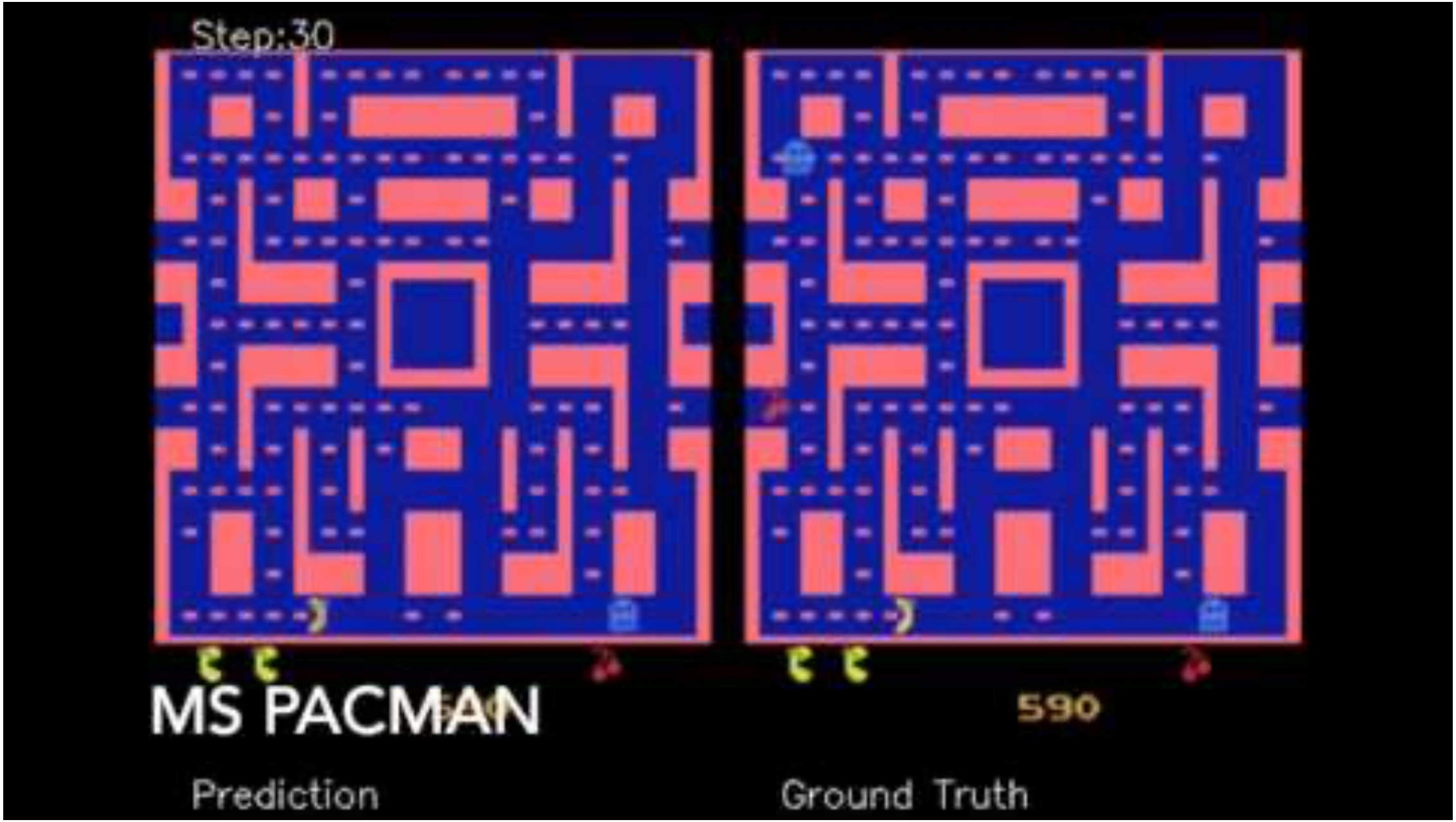
$$\mathcal{L}_K(\theta) = \frac{1}{2K} \sum_i \sum_t \sum_{k=1}^{K} \left\| \hat{x}^i_{t+k} - x^i_{t+k} \right\|^2$$

Time step

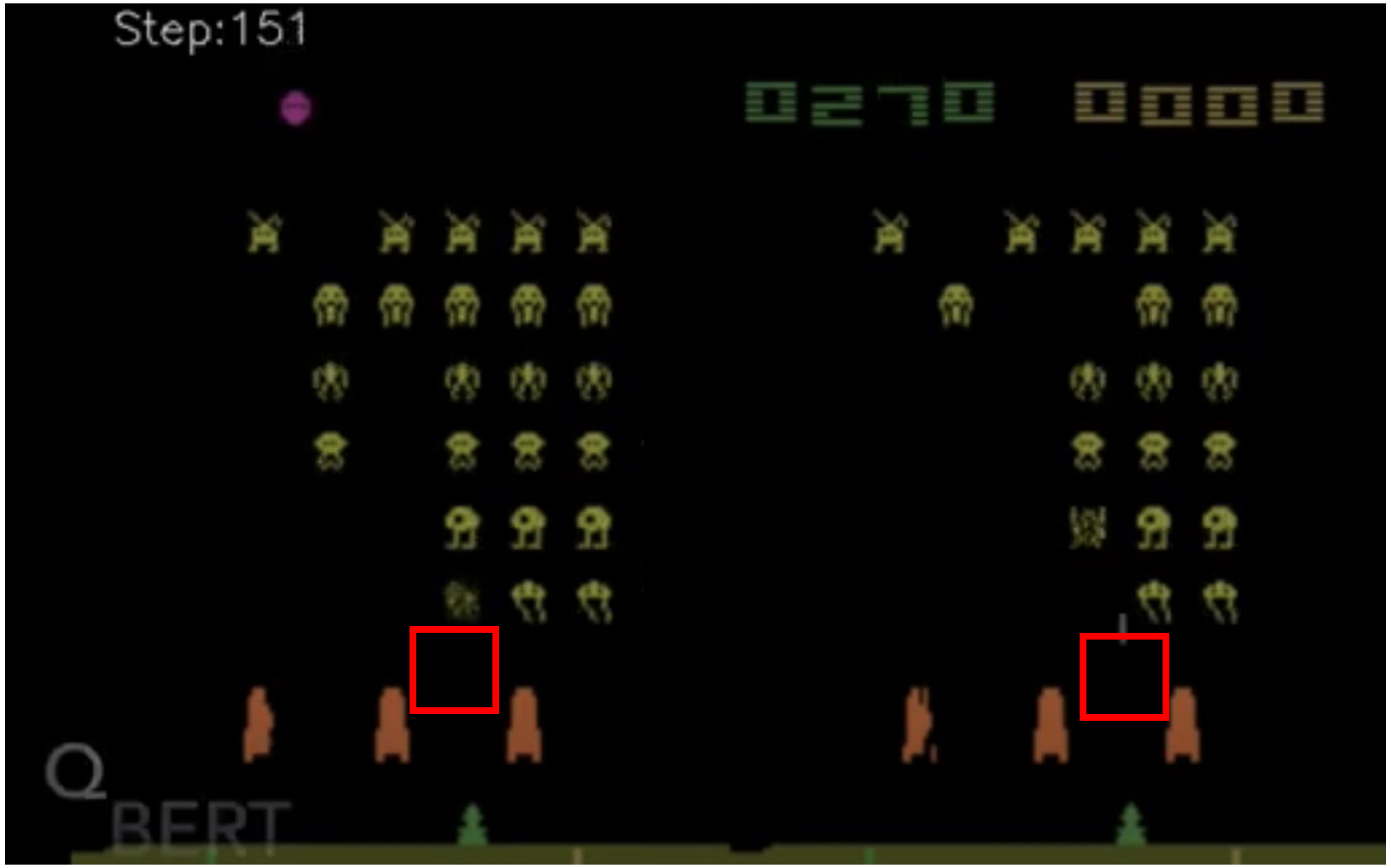
Horizon

Use multiple training phases of increasing K

- Creates a curriculum: predict longer horizons after learning shorter



https://youtu.be/7FBFhG2LgNQ?si=bFFjrM_5Tili59AU



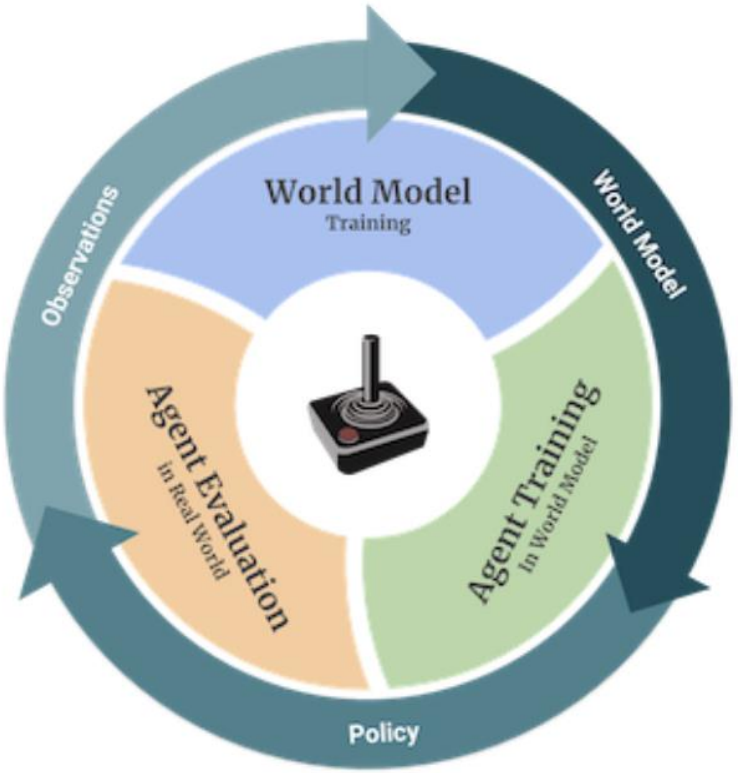
Small details are not always captured, even if they are important

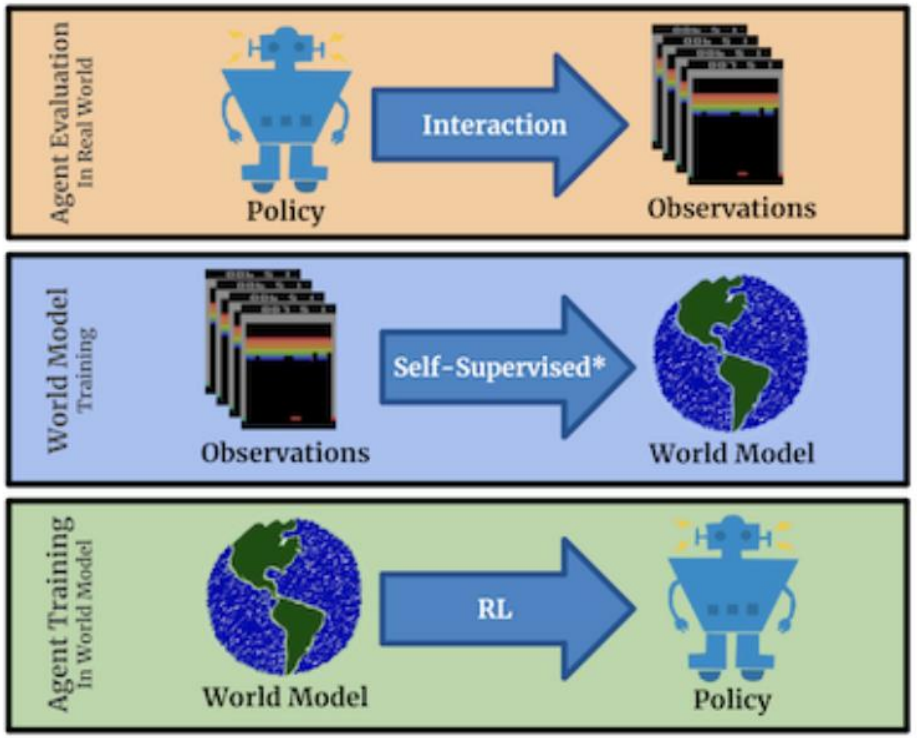
- Video prediction does not consider task-relevance

What about rewards?

So far we have had to define our own reward function

Can we also learn to predict them?





Simulated policy learning

Idea: predict the next image (like visual foresight) but also the reward, then use predicted images/rewards to train a policy

Unlike the robotic pushing task we cannot automatically specify rewards in Atari even with images

Unlike visual foresight, model is learned online

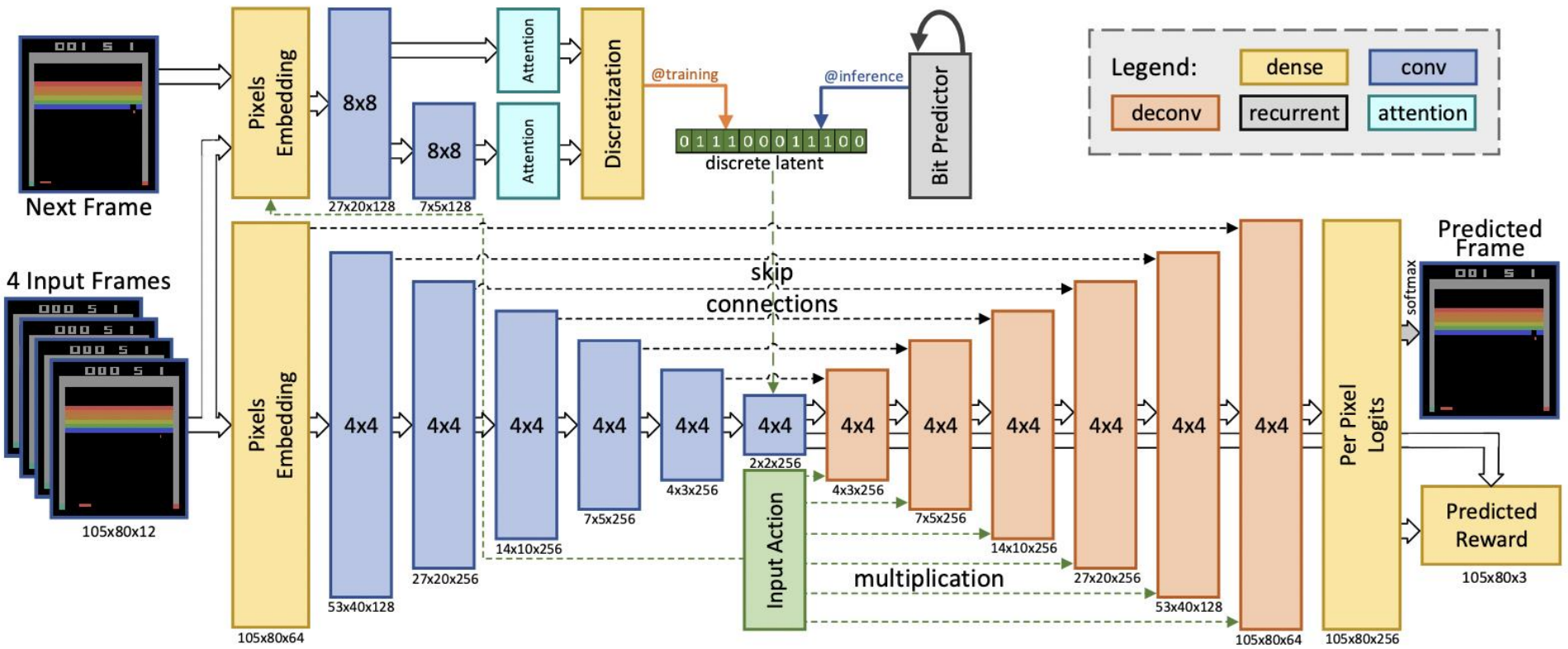
Algorithm 1: Pseudocode for SimPLe

Initialize policy π
Initialize model parameters θ of env'
Initialize empty set $\mathbf{D}$
while not done **do**
  ▷ collect observations from real env.
  $\mathbf{D} \leftarrow \mathbf{D} \cup \text{COLLECT}(env, \pi)$
  ▷ update model using collected data.
  $\theta \leftarrow \text{TRAIN\_SUPERVISED}(env', \mathbf{D})$
  ▷ update policy using world model.
  $\pi \leftarrow \text{TRAIN\_RL}(\pi, env')$
end while

Kaiser et al, Model-Based Reinforcement Learning for Atari, 2020

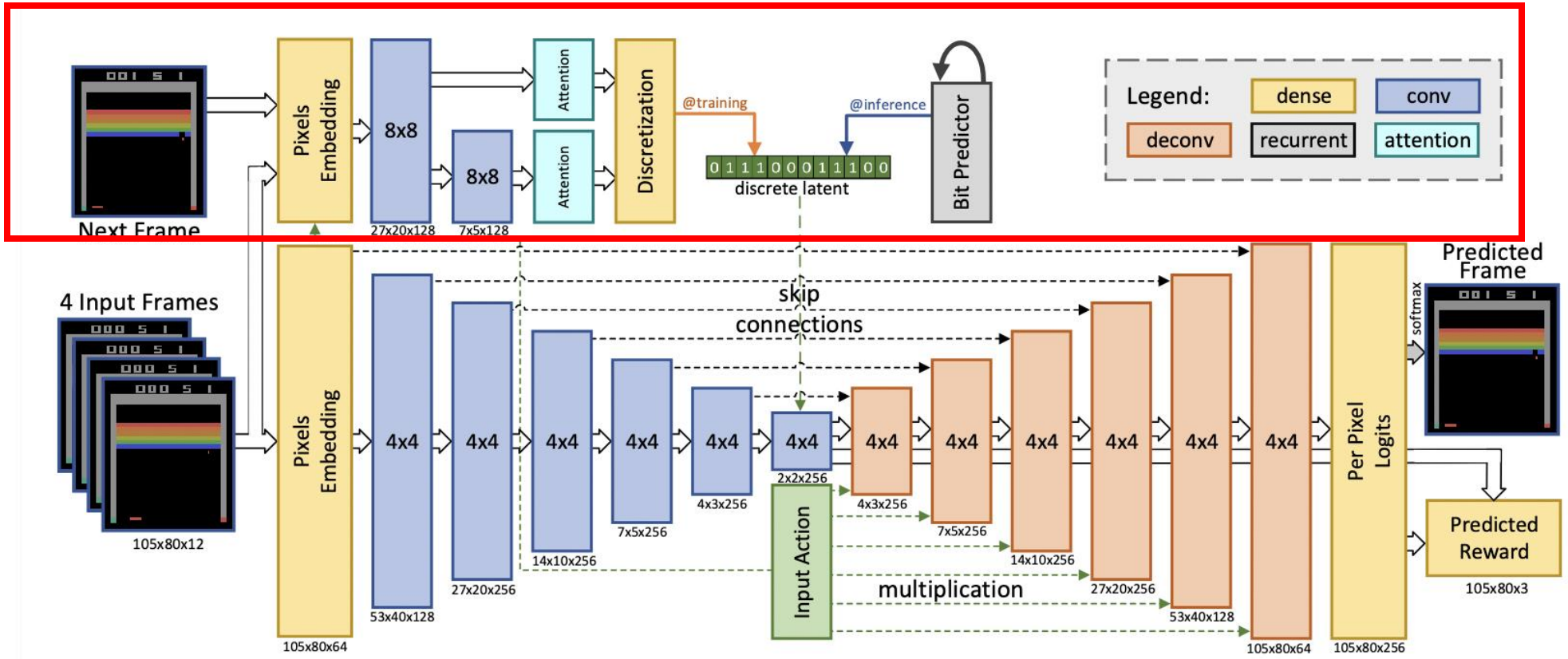
Simulated policy learning

Image and reward are supervised, latent is self-supervised



Simulated policy learning

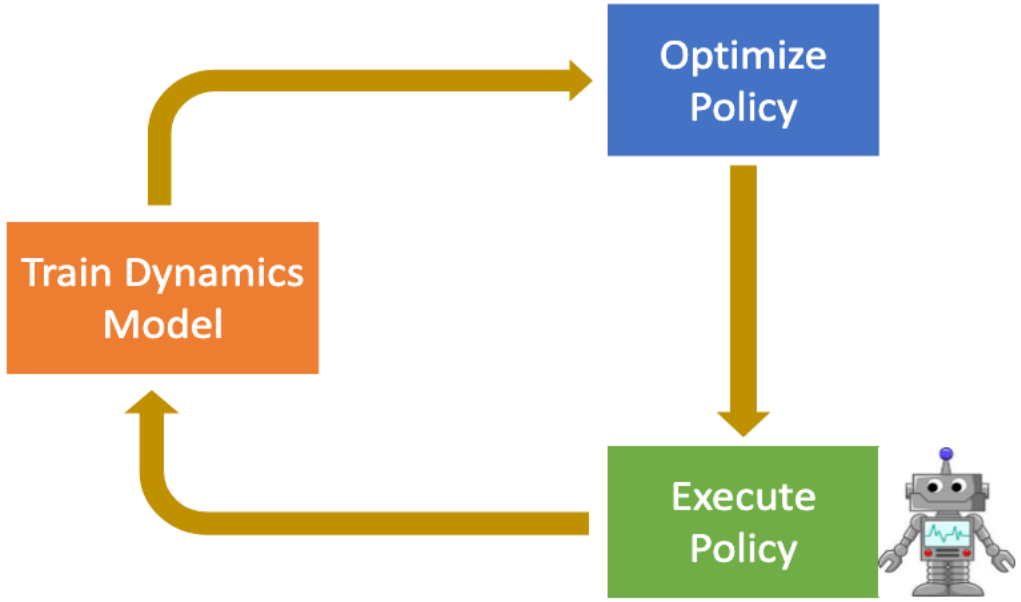
Image and reward are supervised, latent is self-supervised



Only needed if stochastic

Alternating model and policy learning

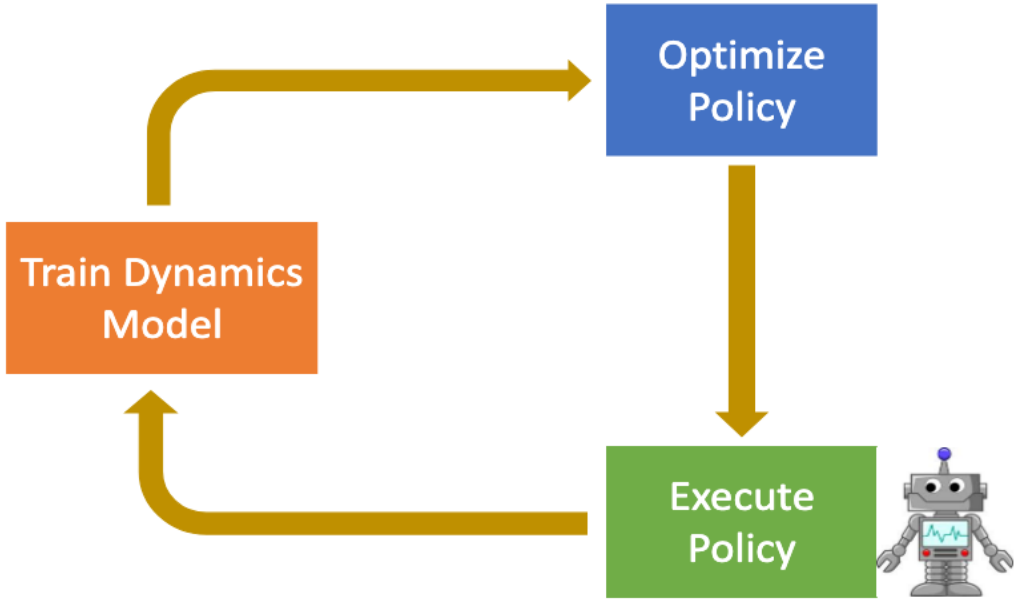
It is common to alternate model/policy learning. **Why?**

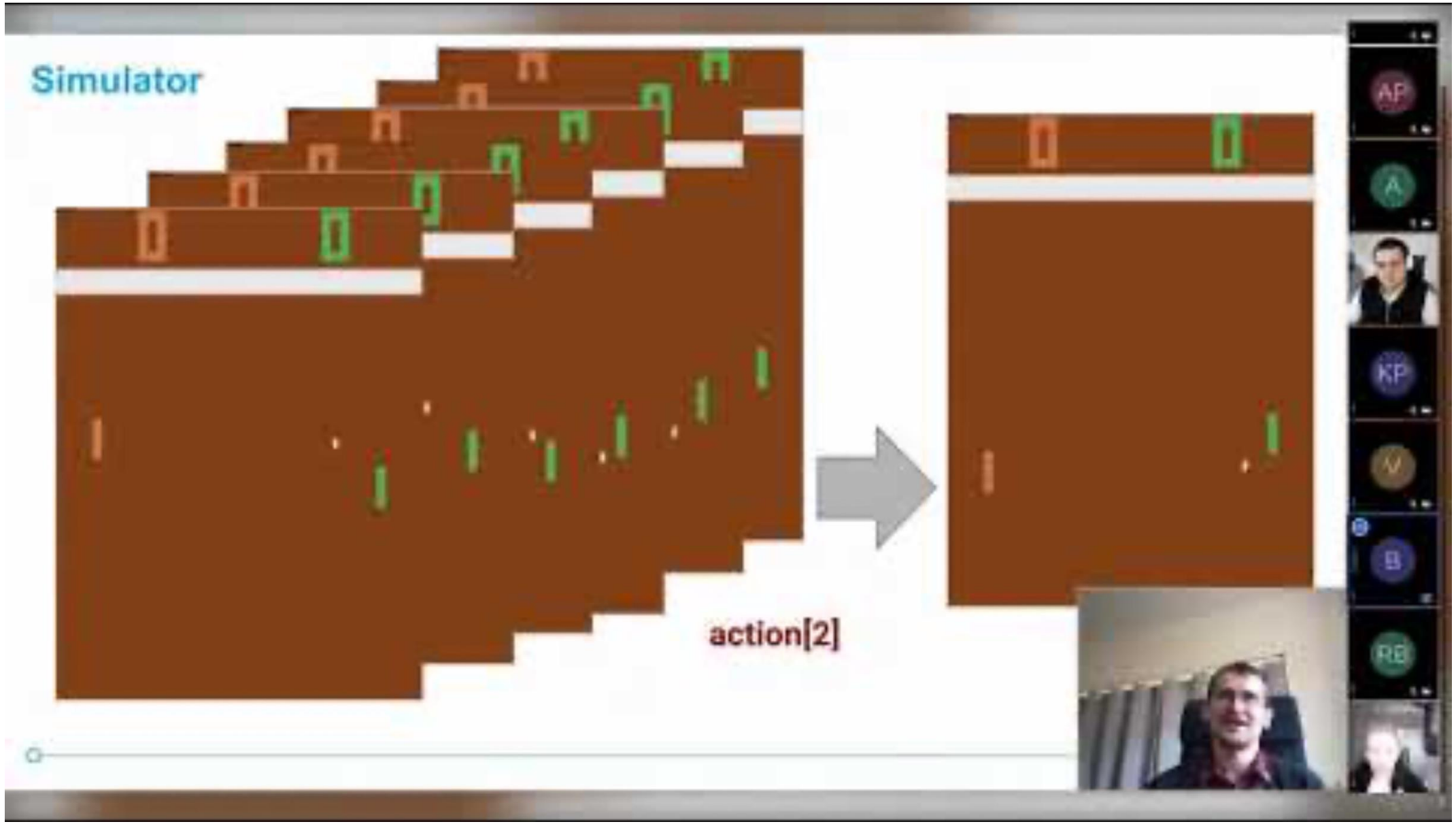


Alternating model and policy learning

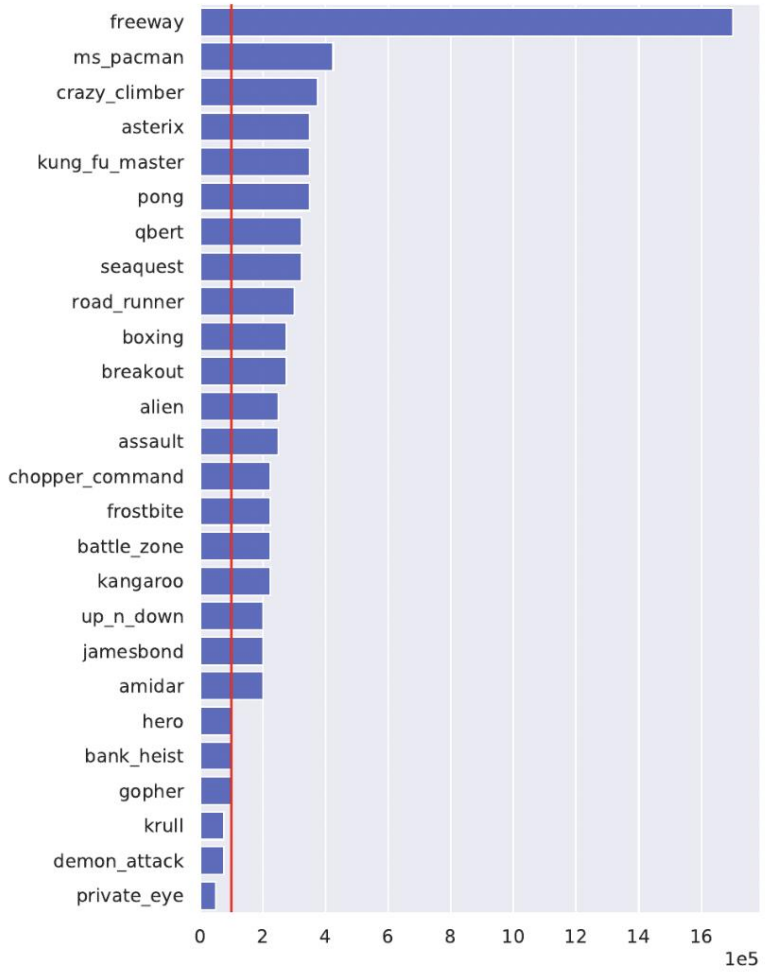
It is common to alternate model/policy learning. **Why?**

- Model dynamics only accurate close to training data
- Updating model with policy actions brings training support of model close to state distribution visited by policy

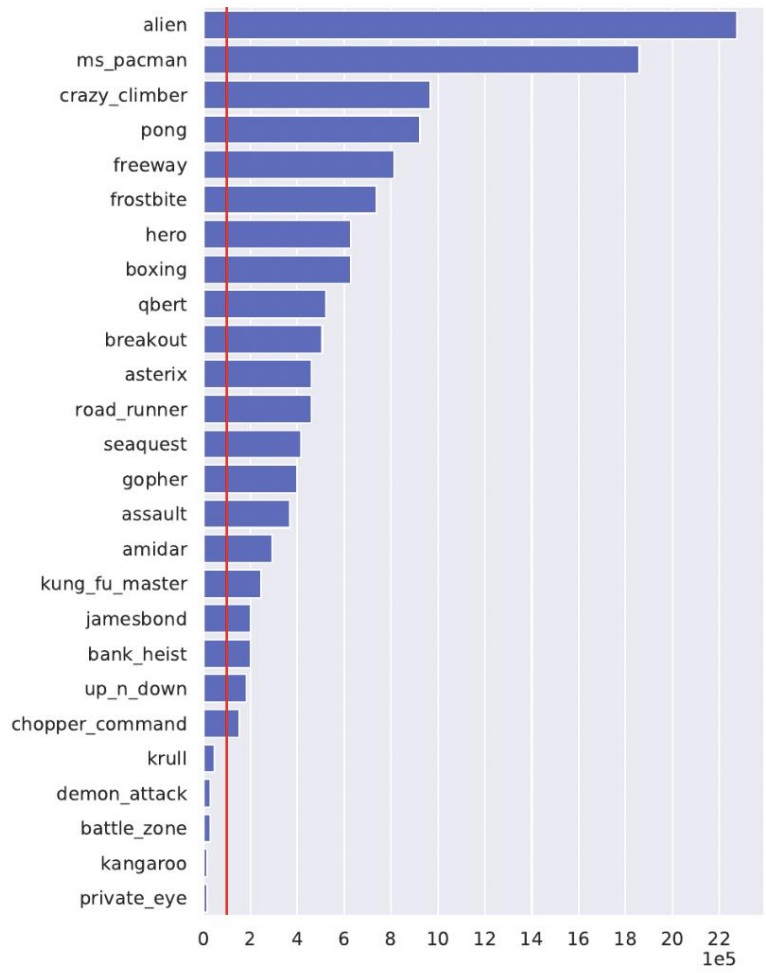




https://youtu.be/fblsbJL2ycU?si=cSW44Iu-9KgNfrHA&t=1866



Rainbow



PPO

Model-based RL far more sample-efficient than model-free

Low-dimensional “latent-space” models

What is the problem with image prediction?

What is the problem with image prediction?

It’s really, really hard!

- Inaccuracies in the model are propagated forward in time
- Errors can lead to spurious correlations in the learned policy
- Generating images is computationally expensive

What can we do to solve this?

Learn a lower-dimensional “world model”!

World Models

David Ha [1] **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:

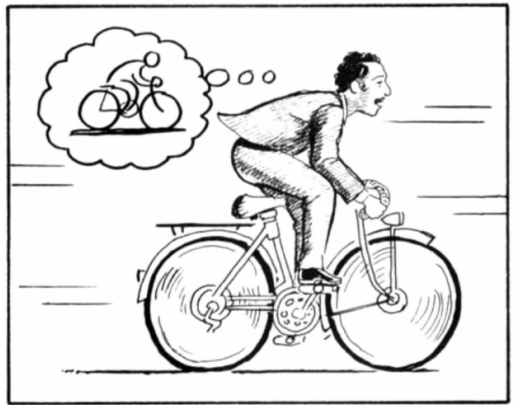


Figure 1. A World Model, from Scott McCloud’s *Understanding Comics.* (McCloud, 1993; E, 2012)

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

Dream to Control: Learning Behaviors by Latent Imagination

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

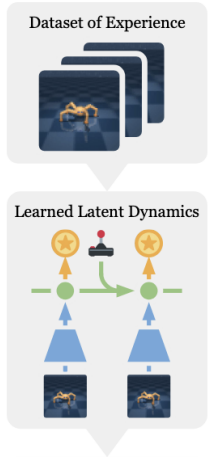
Mohammad Norouzi
Google Brain

Abstract

Learned world models summarize an agent’s experience to facilitate learning complex behaviors. While learning world models from high-dimensional sensory inputs is becoming feasible through deep learning, there are many potential ways for deriving behaviors from them. We present Dreamer, a reinforcement learning agent that solves long-horizon tasks from images purely by latent imagination. We efficiently learn behaviors by propagating analytic gradients of learned state values back through trajectories imagined in the compact state space of a learned world model. On 20 challenging visual control tasks, Dreamer exceeds existing approaches in data-efficiency, computation time, and final performance.

1 Introduction

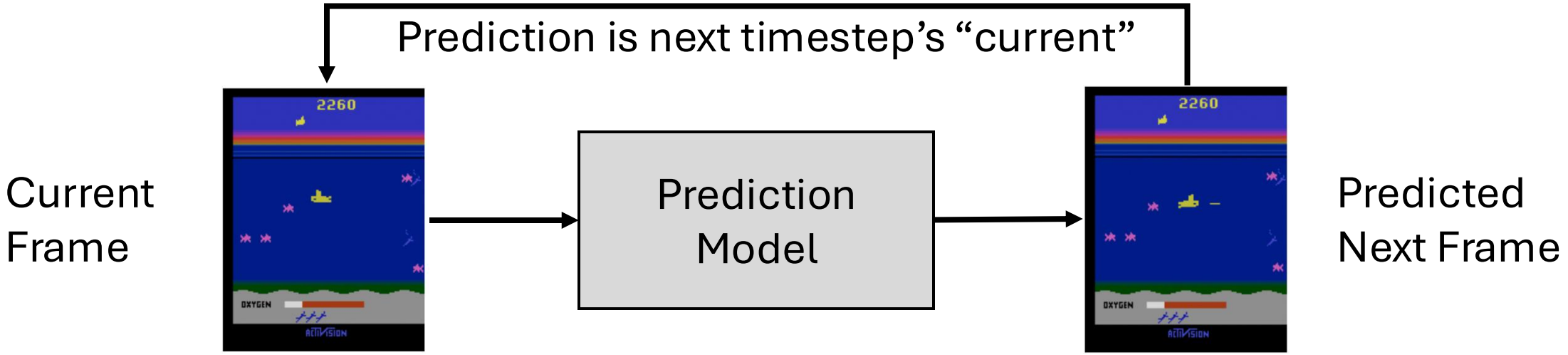
Intelligent agents can achieve goals in complex environments even though they never encounter the exact same situation twice. This ability requires building representations of the world from past experience that enable generalization to novel situations. World models offer an explicit way to represent an agent’s knowledge about the world in a parametric model that can make predictions about the future.

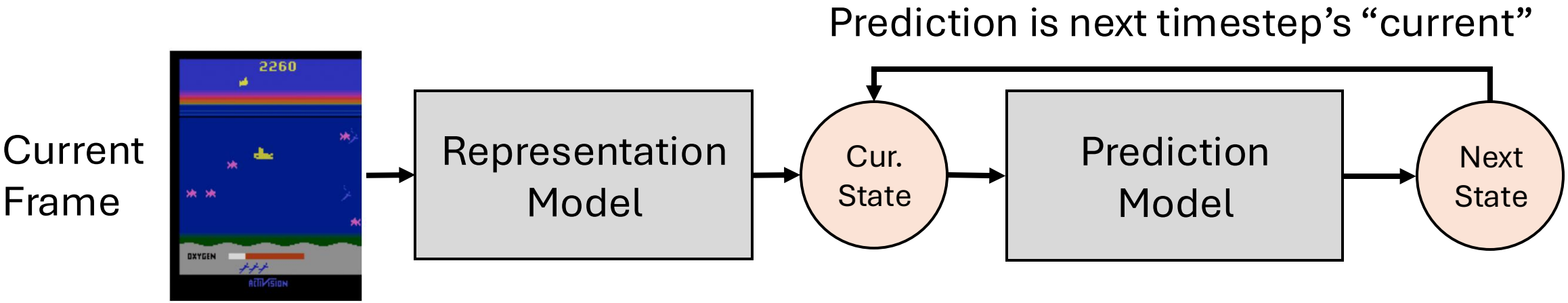


When the sensory inputs are high-dimensional images, latent dynamics models can abstract observations to predict forward in compact state spaces (Watter et al., 2015; Oh et al., 2017; Gregor et al., 2019). Compared to predictions in image space, latent states have a small memory footprint that enables imagining thousands of trajectories in parallel. Learning effective latent dynamics models is becoming feasible through advances in deep learning and latent variable models (Krishnan et al., 2015; Karl et al., 2016; Doerr et al., 2018; Buesing et al., 2018).

Behaviors can be derived from dynamics models in many ways. Often, imagined rewards are maximized with a parametric policy (Sutton, 1991; Ha and Schmidhuber, 2018; Zhang et al., 2019) or by online planning

Learn a lower-dimensional “world model”!





MBRL with latent world models

Idea: learn a compressed latent representation of the environment (world model) and then plan/learn in this latent space.

Pros:

- The latent representation abstracts away irrelevant details and leads to less noisy predictions (compared to pixel-space models)
- The latent representation is far more compact which means synthetic experiences can be generated more efficiently

MBRL with latent world models

Idea: learn a compressed latent representation of the environment (world model) and then plan/learn in this latent space.

Cons:

- ???

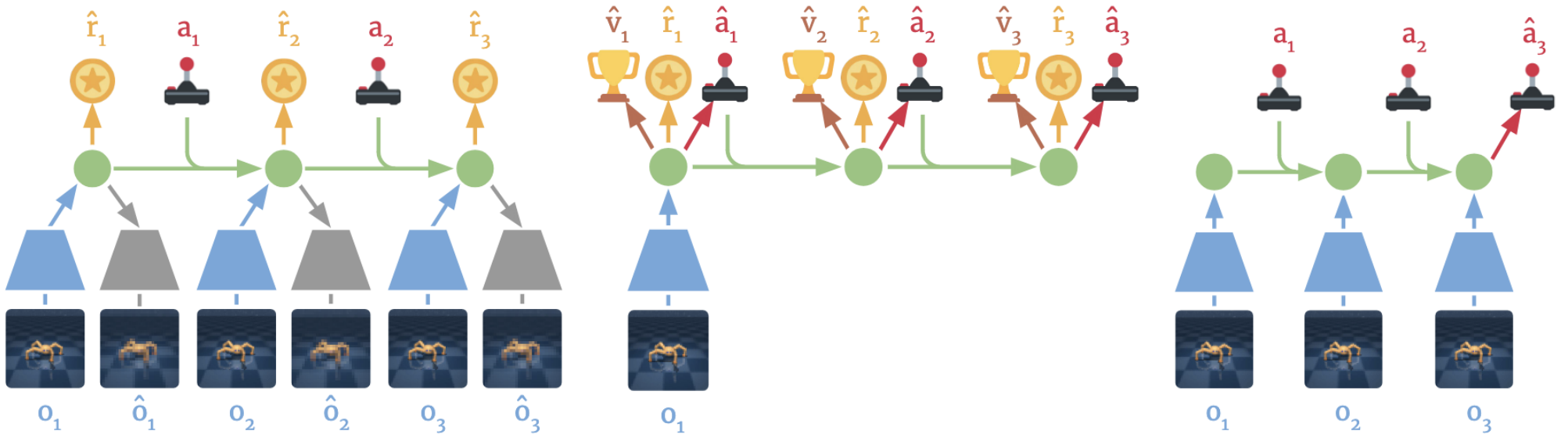
MBRL with latent world models

Idea: learn a compressed latent representation of the environment (world model) and then plan/learn in this latent space.

Cons:

- The latent representation may abstract away important fine-grained details that limit asymptotic performance
- The latent space is uninterpretable. How do we know if the predicted future latent states are accurate or useful?
- Mismatch in latent vs real-world dynamics

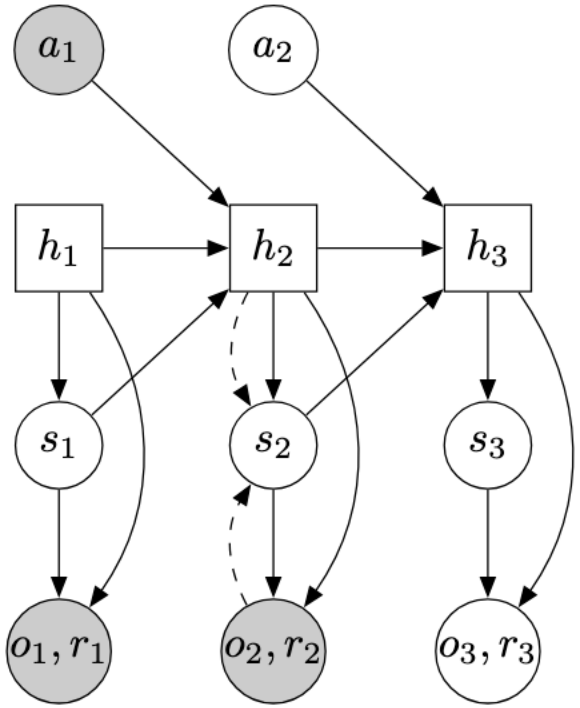
Dreamer (v1)



(a) Learn dynamics from experience (b) Learn behavior in imagination (c) Act in the environment

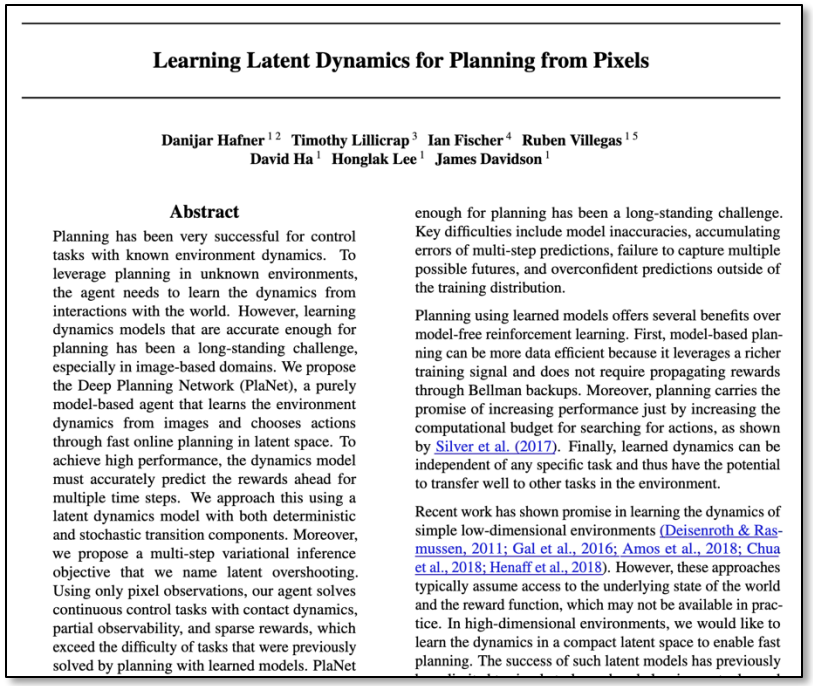
Hafner et al, Dream to Control: Learning Behaviors By Latent Imagination, 2020.

Recurrent state space model



Goal is to predict future latent state s_t and reward r_t

$$
\begin{aligned}
&\text{Transition model:} && s_t \sim p(s_t \mid s_{t-1}, a_{t-1}) \\
&\text{Observation model:} && o_t \sim p(o_t \mid s_t) \\
&\text{Reward model:} && r_t \sim p(r_t \mid s_t),
\end{aligned} \tag{2}
$$



Learning Latent Dynamics for Planning from Pixels

Danijar Hafner [1 2] Timothy Lillicrap [3] Ian Fischer [4] Ruben Villegas [1 5]
David Ha [1] Honglak Lee [1] James Davidson [1]

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 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

Algorithm 1: Dreamer

Initialize dataset $\mathcal{D}$ with S random seed episodes.
Initialize neural network parameters θ, ϕ, ψ randomly.
while *not converged* **do**
  for *update step* $c = 1..C$ **do**
    `// Dynamics learning`
    Draw B data sequences $\{(a_t, o_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D}$.
    Compute model states $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$.
    Update θ using representation learning.
    `// Behavior learning`
    Imagine trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ from each s_t.
    Predict rewards $\mathrm{E}\big(q_\theta(r_\tau \mid s_\tau)\big)$ and values $v_\psi(s_\tau)$.
    Compute value estimates $\mathrm{V}_\lambda(s_\tau)$ via Equation 6.
    Update $\phi \leftarrow \phi + \alpha \nabla_\phi \sum_{\tau=t}^{t+H} \mathrm{V}_\lambda(s_\tau)$.
    Update $\psi \leftarrow \psi - \alpha \nabla_\psi \sum_{\tau=t}^{t+H} \frac{1}{2} \big\| v_\psi(s_\tau) - \mathrm{V}_\lambda(s_\tau) \big\|^2$.
  `// Environment interaction`
  $o_1 \leftarrow$ `env.reset()`
  for *time step* $t = 1..T$ **do**
    Compute $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$ from history.
    Compute $a_t \sim q_\phi(a_t \mid s_t)$ with the action model.
    Add exploration noise to action.
    $r_t, o_{t+1} \leftarrow$ `env.step(`a_t`)`.
  Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$.

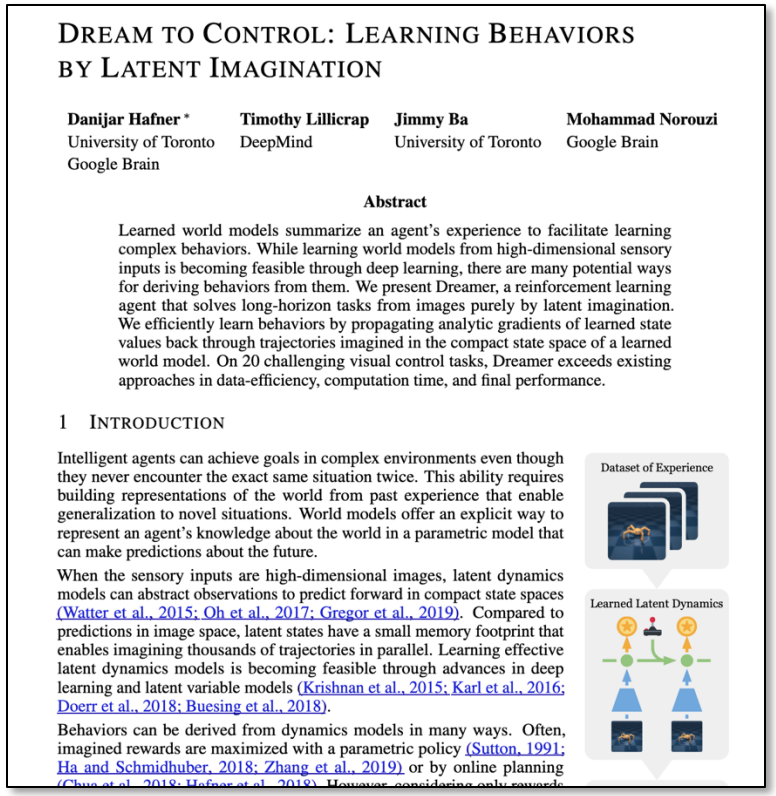
Model components

Representation	$p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$
Transition	$q_\theta(s_t \mid s_{t-1}, a_{t-1})$
Reward	$q_\theta(r_t \mid s_t)$
Action	$q_\phi(a_t \mid s_t)$
Value	$v_\psi(s_t)$

Hyper parameters

Seed episodes	S
Collect interval	C
Batch size	B
Sequence length	L
Imagination horizon	H
Learning rate	α

Dreaming and dreaming and dreaming



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

Abstract

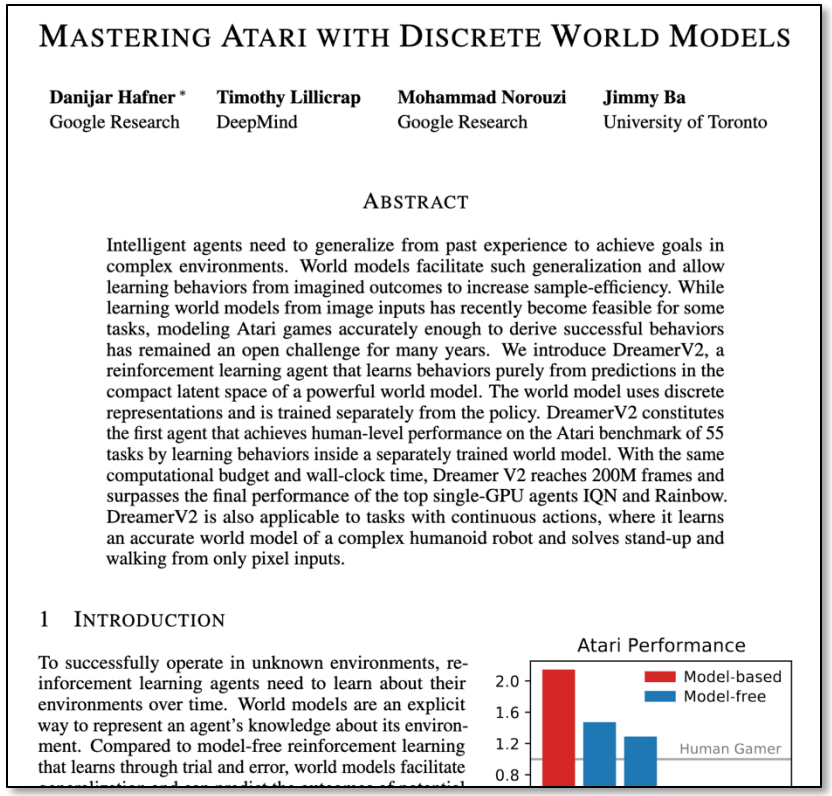
Learned world models summarize an agent's experience to facilitate learning complex behaviors. While learning world models from high-dimensional sensory inputs is becoming feasible through deep learning, there are many potential ways for deriving behaviors from them. We present Dreamer, a reinforcement learning agent that solves long-horizon tasks from images purely by latent imagination. We efficiently learn behaviors by propagating analytic gradients of learned state values back through trajectories imagined in the compact state space of a learned world model. On 20 challenging visual control tasks, Dreamer exceeds existing approaches in data-efficiency, computation time, and final performance.

1 INTRODUCTION

Intelligent agents can achieve goals in complex environments even though they never encounter the exact same situation twice. This ability requires building representations of the world from past experience that enable generalization to novel situations. World models offer an explicit way to represent an agent's knowledge about the world in a parametric model that can make predictions about the future.

When the sensory inputs are high-dimensional images, latent dynamics models can abstract observations to predict forward in compact state spaces (Watter et al., 2015; Oh et al., 2017; Gregor et al., 2019). Compared to predictions in image space, latent states have a small memory footprint that enables imagining thousands of trajectories in parallel. Learning effective latent dynamics models is becoming feasible through advances in deep learning and latent variable models (Krishnan et al., 2015; Karl et al., 2016; Doerr et al., 2018; Buesing et al., 2018).

Behaviors can be derived from dynamics models in many ways. Often, imagined rewards are maximized with a parametric policy (Sutton, 1991; Ha and Schmidhuber, 2018; Zhang et al., 2019) or by online planning



MASTERING ATARI WITH DISCRETE WORLD MODELS

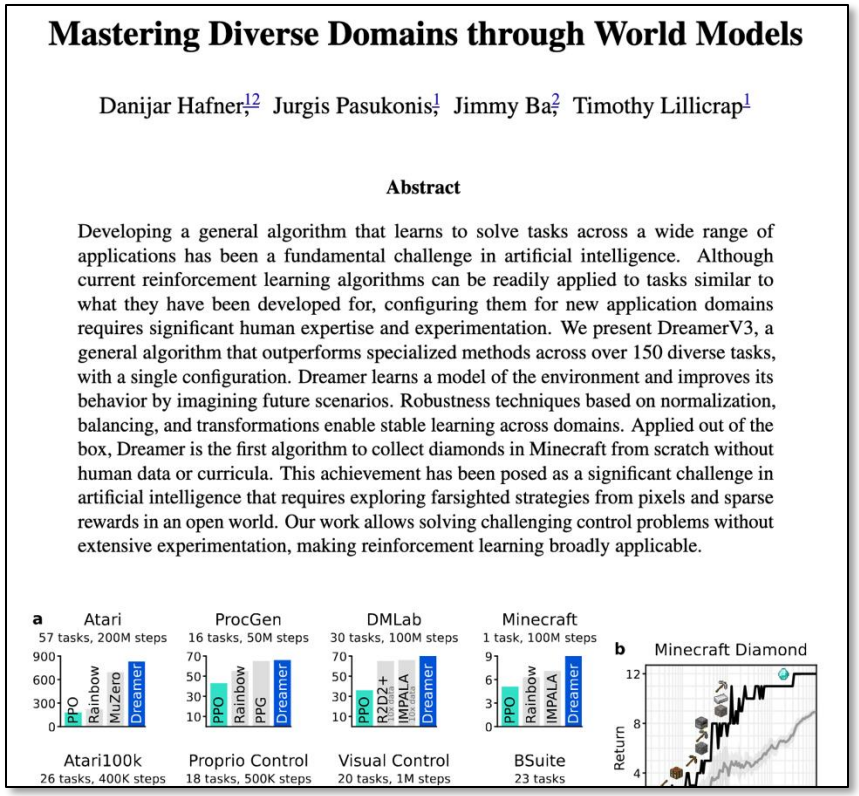
Danijar Hafner * Google Research
Timothy Lillicrap DeepMind
Mohammad Norouzi Google Research
Jimmy Ba University of Toronto

ABSTRACT

Intelligent agents need to generalize from past experience to achieve goals in complex environments. World models facilitate such generalization and allow learning behaviors from imagined outcomes to increase sample-efficiency. While learning world models from image inputs has recently become feasible for some tasks, modeling Atari games accurately enough to derive successful behaviors has remained an open challenge for many years. We introduce DreamerV2, a reinforcement learning agent that learns behaviors purely from predictions in the compact latent space of a powerful world model. The world model uses discrete representations and is trained separately from the policy. DreamerV2 constitutes the first agent that achieves human-level performance on the Atari benchmark of 55 tasks by learning behaviors inside a separately trained world model. With the same computational budget and wall-clock time, Dreamer V2 reaches 200M frames and surpasses the final performance of the top single-GPU agents IQN and Rainbow. DreamerV2 is also applicable to tasks with continuous actions, where it learns an accurate world model of a complex humanoid robot and solves stand-up and walking from only pixel inputs.

1 INTRODUCTION

To successfully operate in unknown environments, reinforcement learning agents need to learn about their environments over time. World models are an explicit way to represent an agent's knowledge about its environment. Compared to model-free reinforcement learning that learns through trial and error, world models facilitate



Mastering Diverse Domains through World Models

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

Abstract

Developing a general algorithm that learns to solve tasks across a wide range of applications has been a fundamental challenge in artificial intelligence. Although current reinforcement learning algorithms can be readily applied to tasks similar to what they have been developed for, configuring them for new application domains requires significant human expertise and experimentation. We present DreamerV3, a general algorithm that outperforms specialized methods across over 150 diverse tasks, with a single configuration. Dreamer learns a model of the environment and improves its behavior by imagining future scenarios. Robustness techniques based on normalization, balancing, and transformations enable stable learning across domains. Applied out of the box, Dreamer is the first algorithm to collect diamonds in Minecraft from scratch without human data or curricula. This achievement has been posed as a significant challenge in artificial intelligence that requires exploring farsighted strategies from pixels and sparse rewards in an open world. Our work allows solving challenging control problems without extensive experimentation, making reinforcement learning broadly applicable.

Dreamer v1
(continuous control)

Dreamer v2
(discrete control)

Dreamer v3
(better performance)

We introduce Dreamer

1. Scalable reinforcement learning from pixels using a world model
2. Learn actor and value in imagination for long-sighted behaviors
3. Efficiently update actor by backprop through imagined sequences

https://youtu.be/BDxRNnhPTlU?si=NjMJI_0fv2vJPXRY&t=40