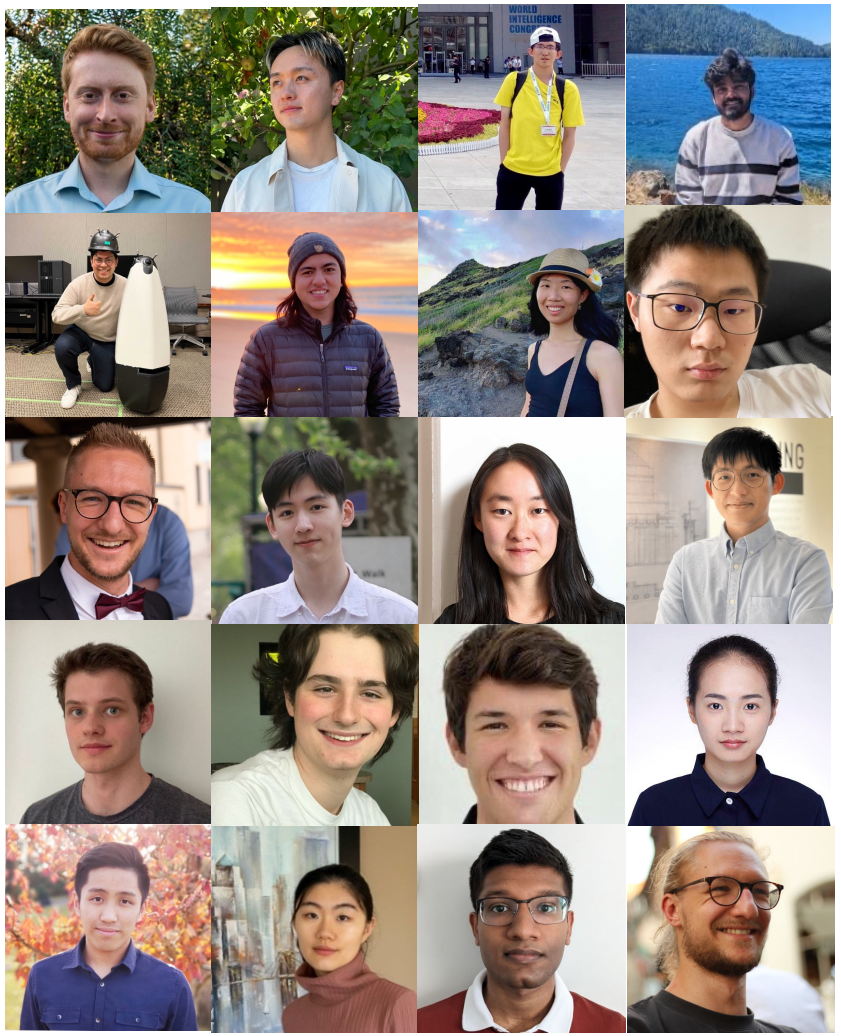
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

World models beyond autoregressive next state reconstruction

Abhishek Gupta

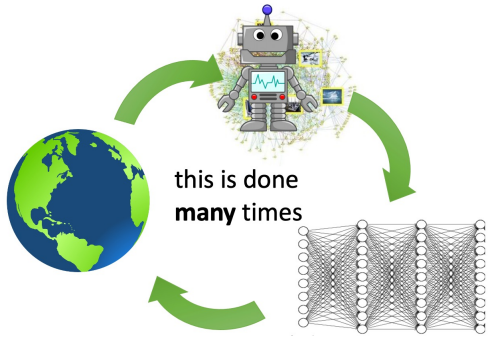
University of Washington

Everything discussed today is done by the **WEIRD Lab**

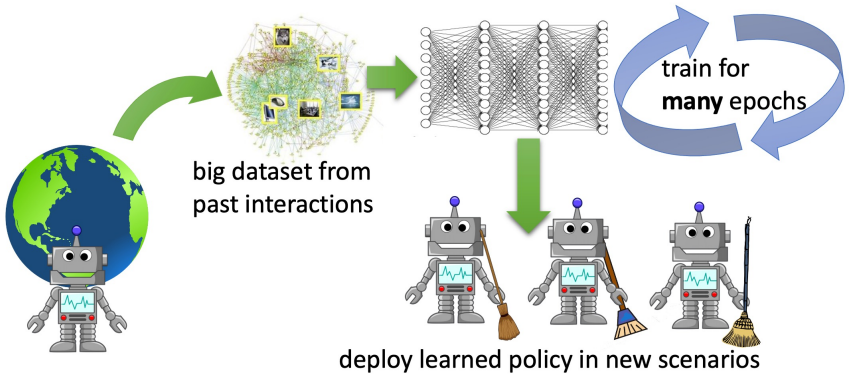


Offline RL – A Taxonomy

reinforcement learning



offline reinforcement learning

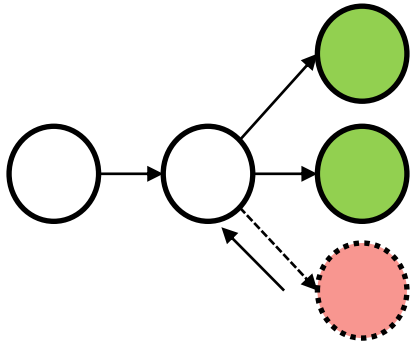


Kumar et al

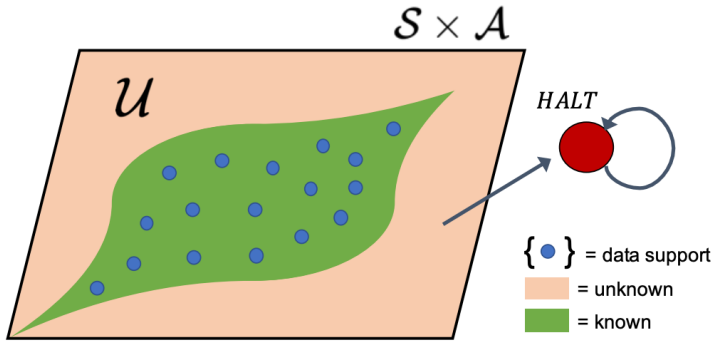
Importance Sampling

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right]$$
$$= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\left(\prod_{t=0}^{H}\frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\beta(\mathbf{a}_t|\mathbf{s}_t)}\right)\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \approx \sum_{i=1}^{n} w_H^i \sum_{t=0}^{H}\gamma^t r_t^i,$$

Model-Free

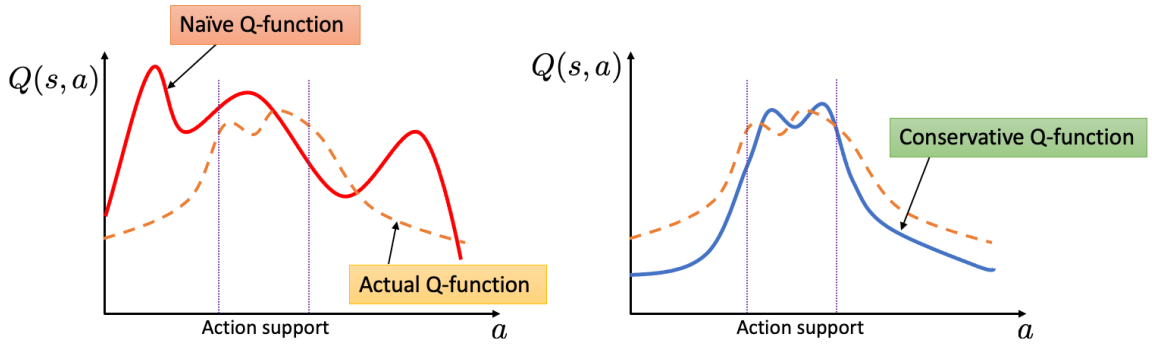


Model-Based

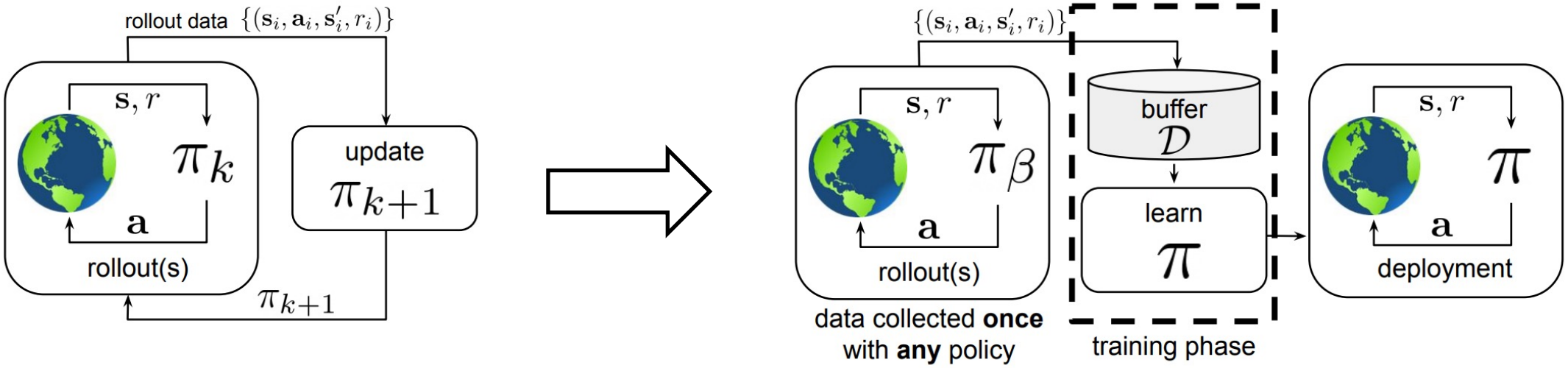


Why even study offline RL?

We've been talking a lot about pessimism, and different ways of enforcing it

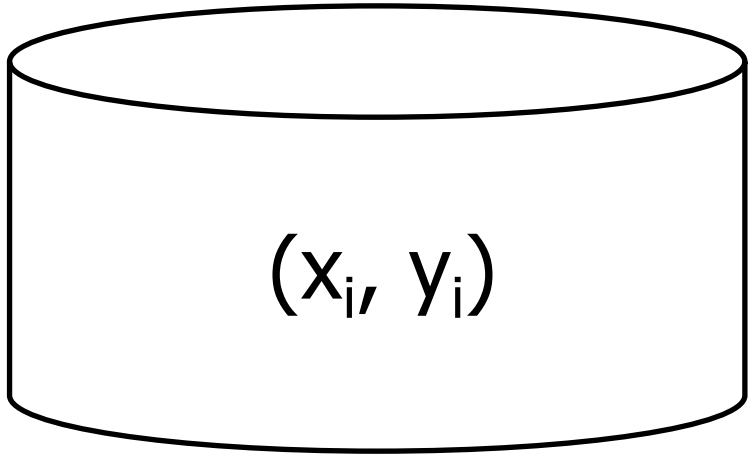


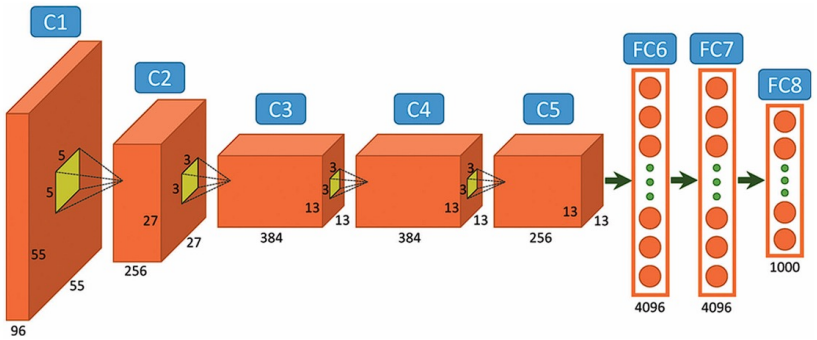
Offline RL is all about studying RL as a data-driven problem



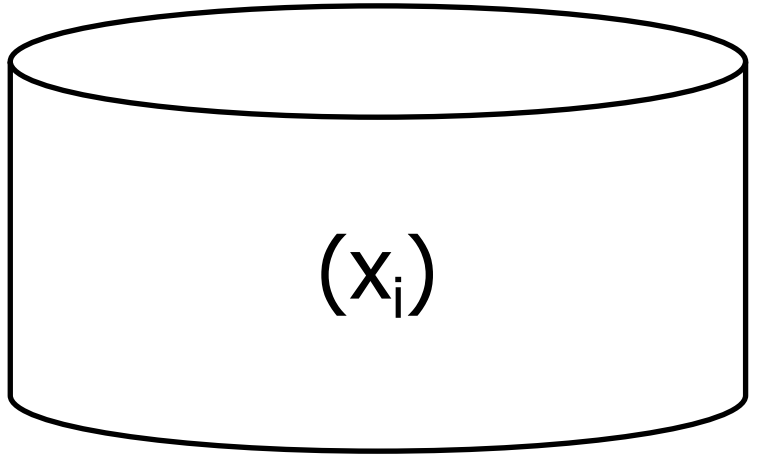
How has data driven machine learning evolved?

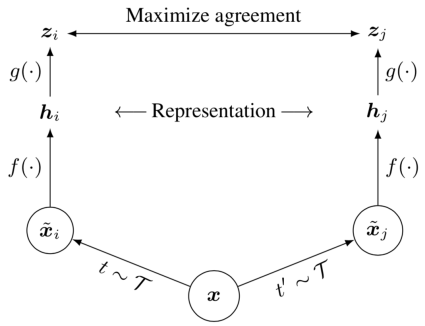
Supervised

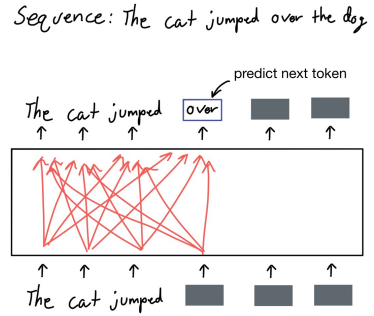




Unsupervised





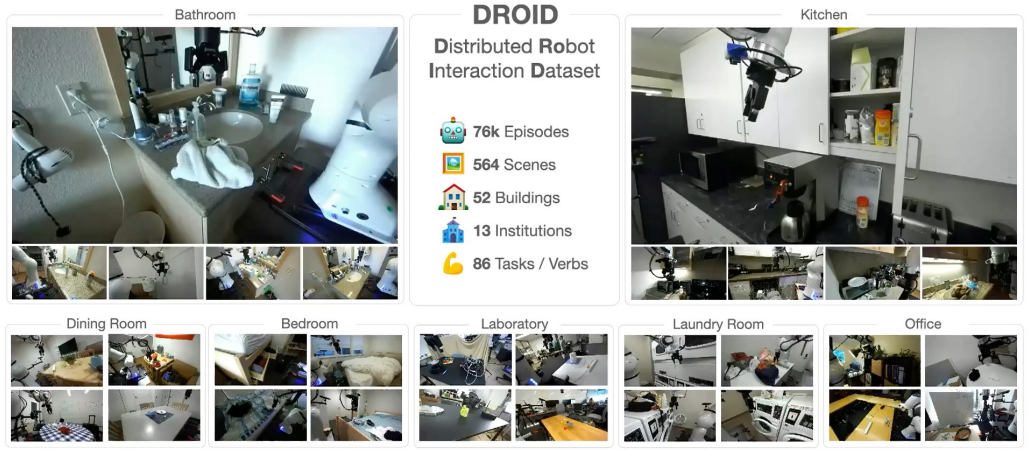


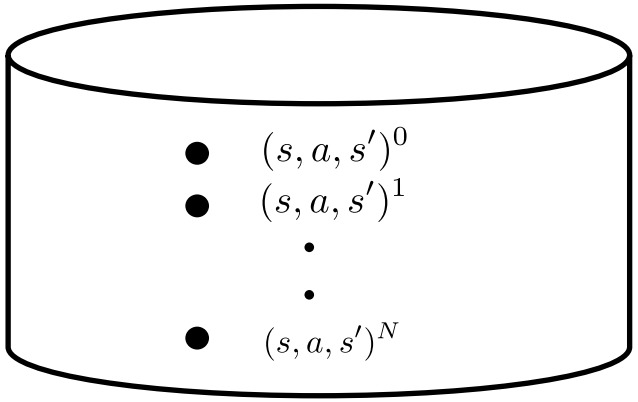
What about RL?

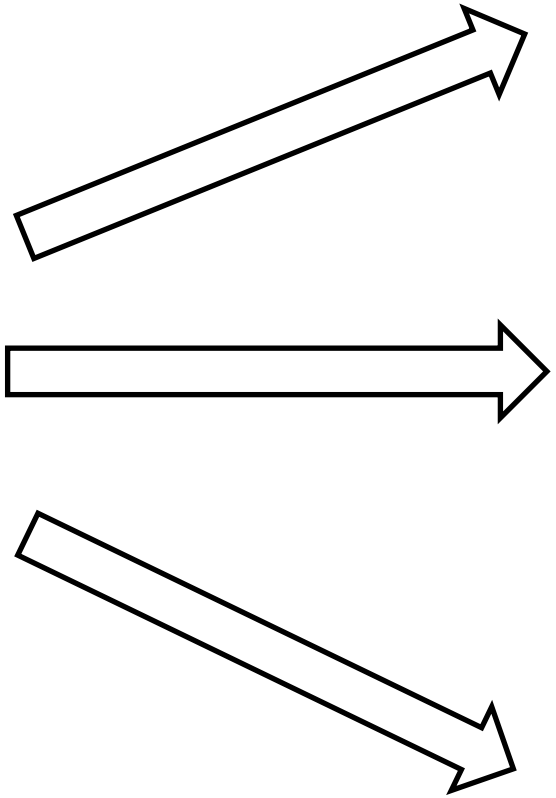
What is a concrete instantiation of unsupervised RL?

Training-time

Large corpus of transitions







Quick adaptation / planning

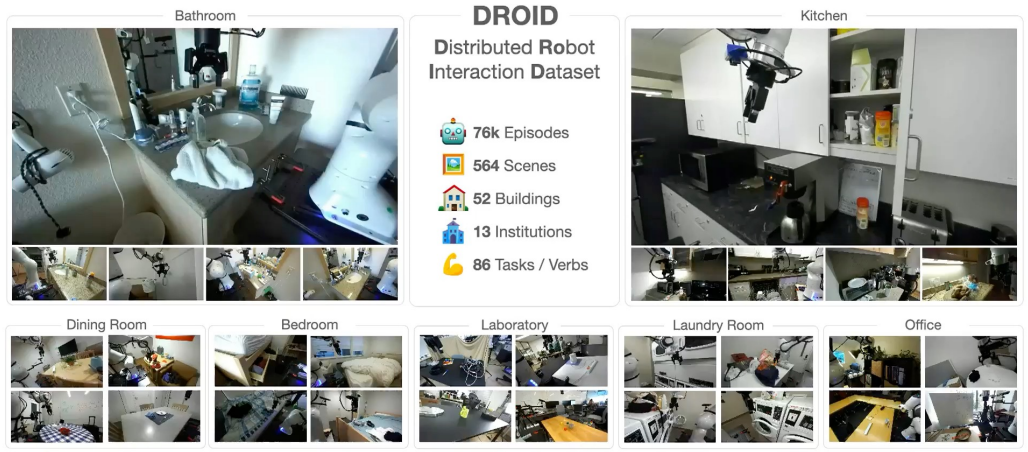
Deployment-time

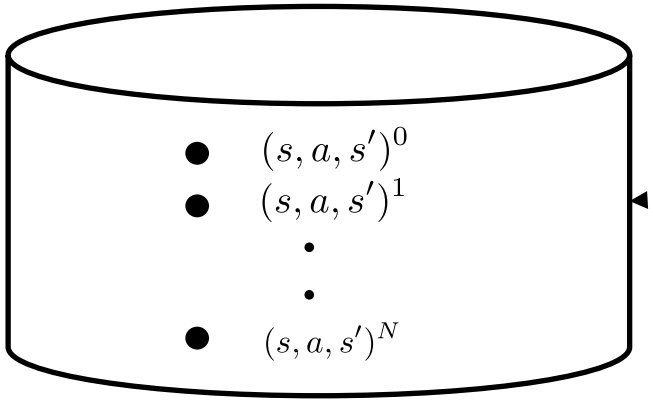
....

What is a concrete instantiation of unsupervised RL?

What can we even learn from this data?

Large corpus of transitions

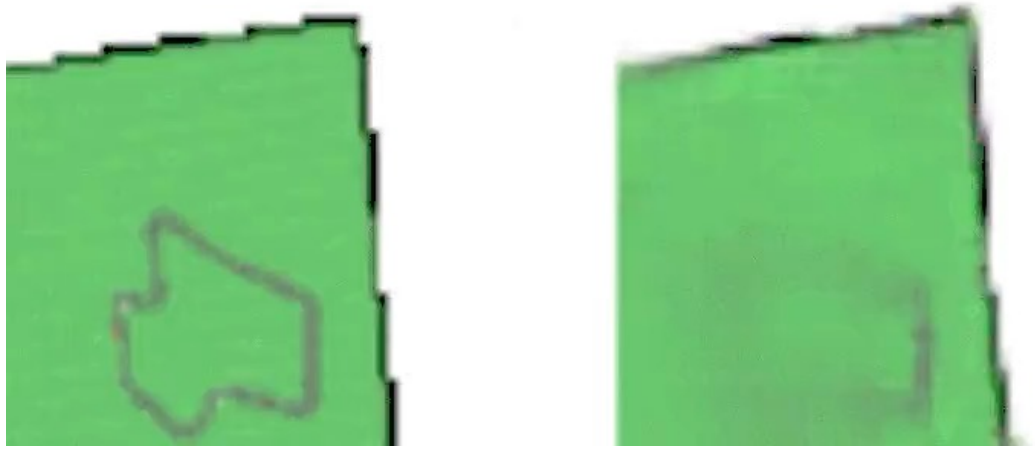




Directly model dynamics: Model-based RL / World Modeling

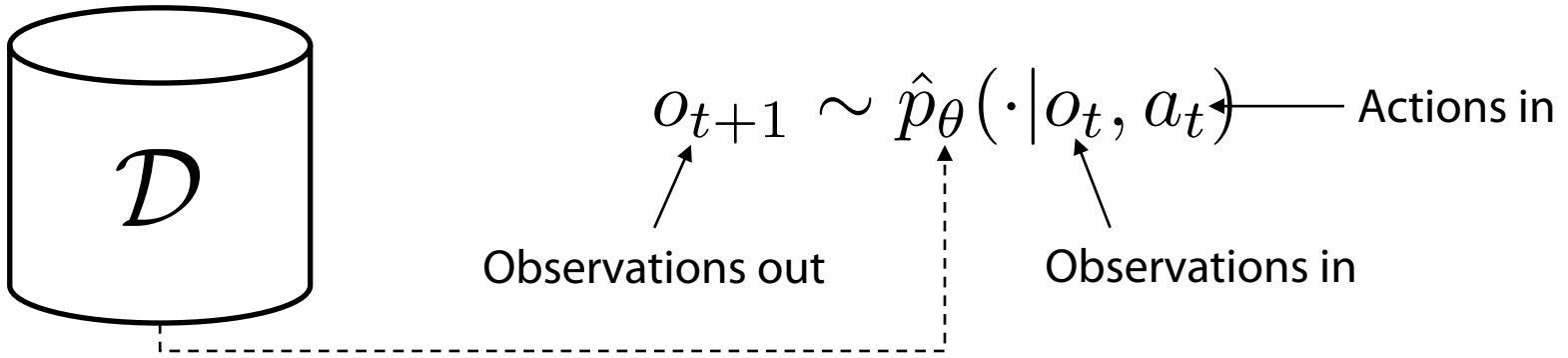
Let’s talk about world models!

Ok so what do I mean by a world model?



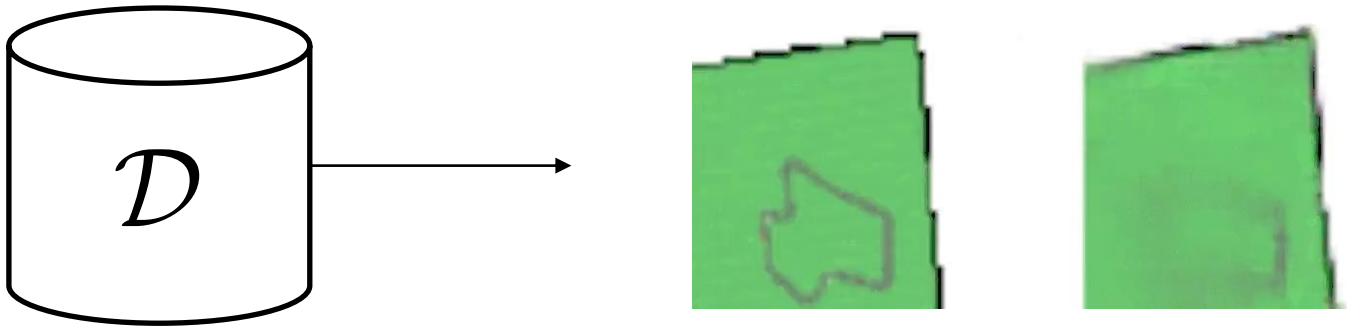
Ha et al '18

A learned predictive model of future observations operating directly from perceptual inputs



Often referred to as action-conditioned video prediction

Why do we care about learned dynamics models?

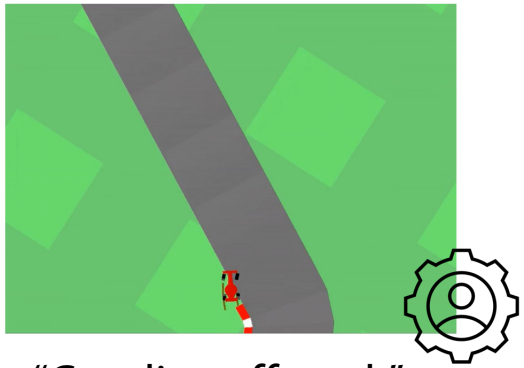


Planning and decision making

Deciding what actions to take

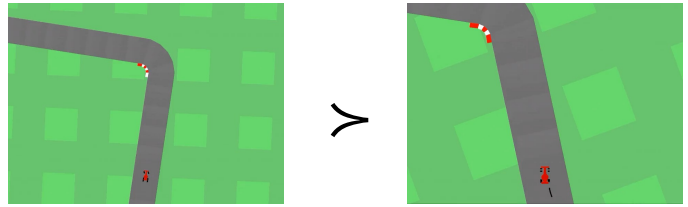
Synthesizing controllers

Outcome prediction



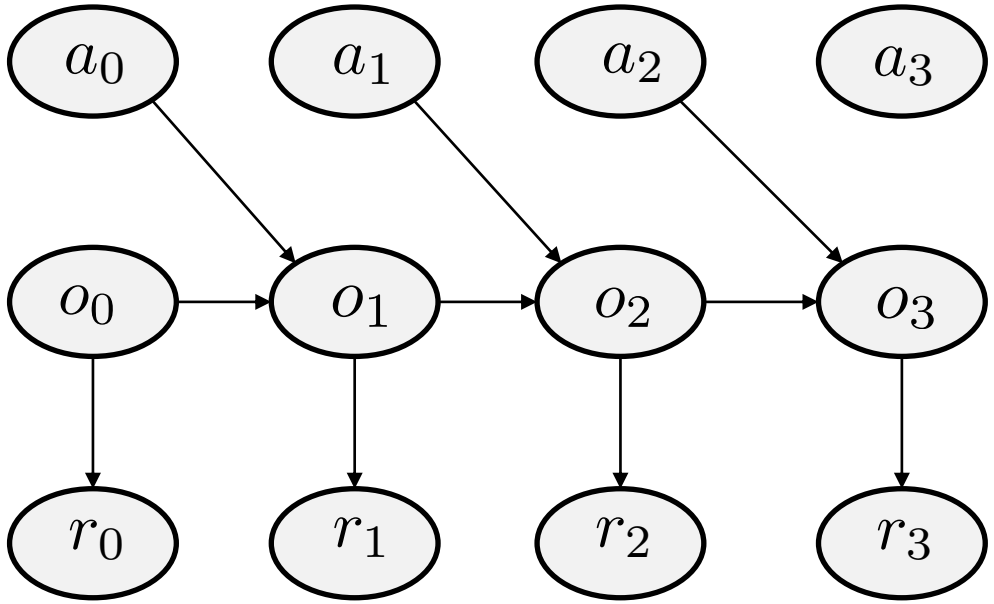
"Car slips off track"

Policy evaluation



Model selection pre-deployment

Preliminaries: Sequential Decision Making



Observations: $\mathcal{O}$ Initial state dist: $\rho_0(o)$

Actions: $\mathcal{A}$ Discount: γ

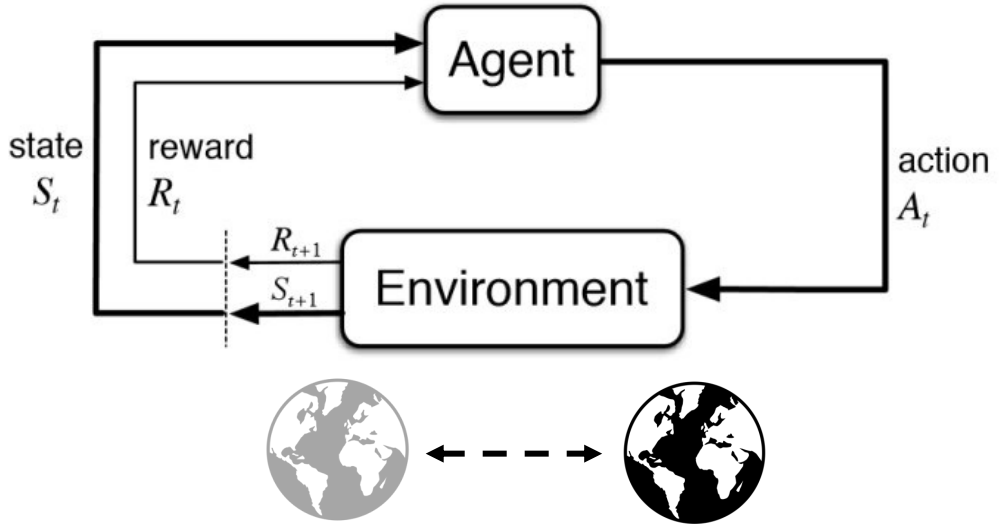
Rewards: $\mathcal{R}$

Transition Dynamics - $p(o_{t+1}|o_t, a_t)$

$$\pi^*(a_t|o_t) \leftarrow \max_{\pi} \mathbb{E}_{p,\pi} \left[\sum_t \gamma^t r(o_t, a_t) \right]$$

Learning Dynamics Models from Data

Learn an approximate (one-step*) model of the dynamics via supervised learning



Maximize one-step transition likelihood

$$\max_{\theta} \mathbb{E}_{(o,a,o')\sim\mathcal{D}} \left[\log \hat{p}_\theta(o'|o,a)\right]$$

Choice of distribution determines the loss function:

1. Gaussian → L_2
2. Energy Based Model → Contrastive Divergence
3. Diffusion Model → Score Matching

+ Simple, easy to optimize
+ Independent of reward, easy to transfer

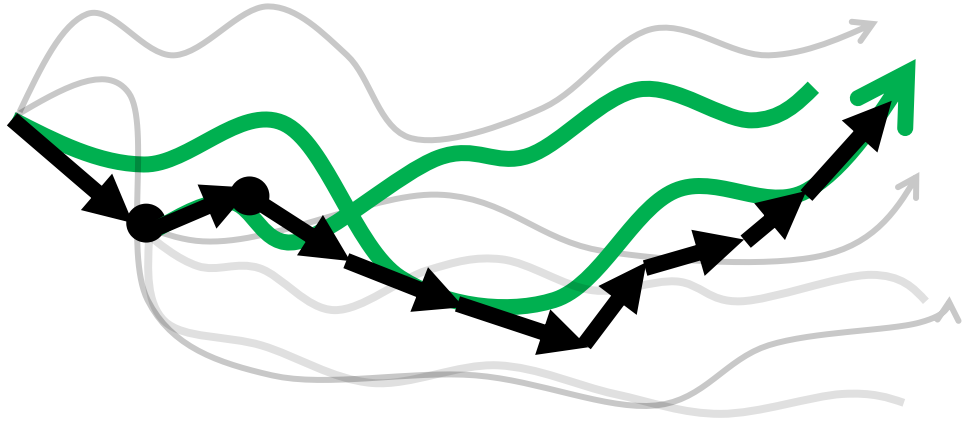
Planning with Learned Dynamics Models

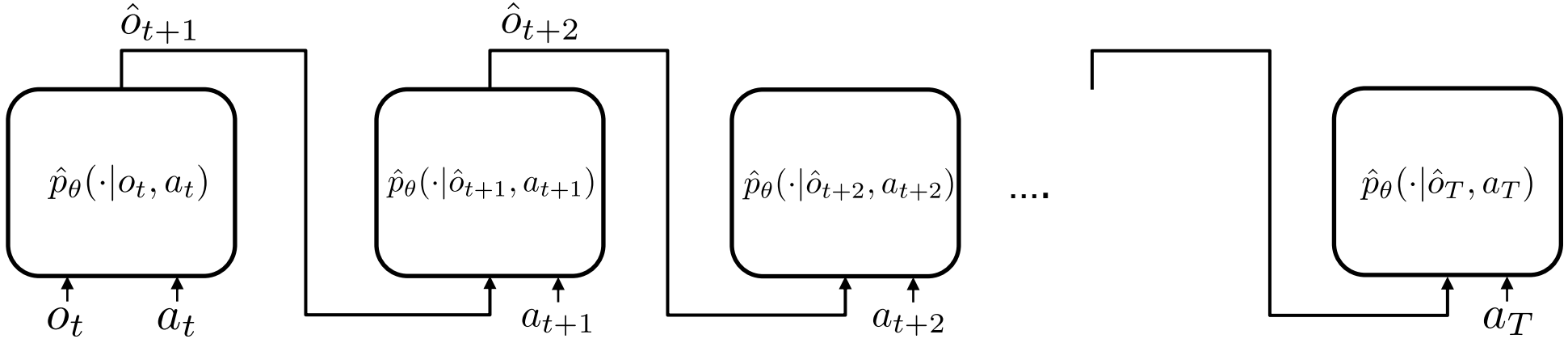
Model is treated as true environment, optimal actions found in this model

$$\arg \max_{a_0, a_1, \ldots, a_T} \sum_{t=0}^{T} r(\hat{o}_t, a_t)$$

$$\hat{o}_{t+1} \sim \hat{p}_\theta(\cdot | \hat{o}_t, a_t)$$

$$\hat{o}_1 \sim \hat{p}_\theta(\cdot | o_0, a_0)$$

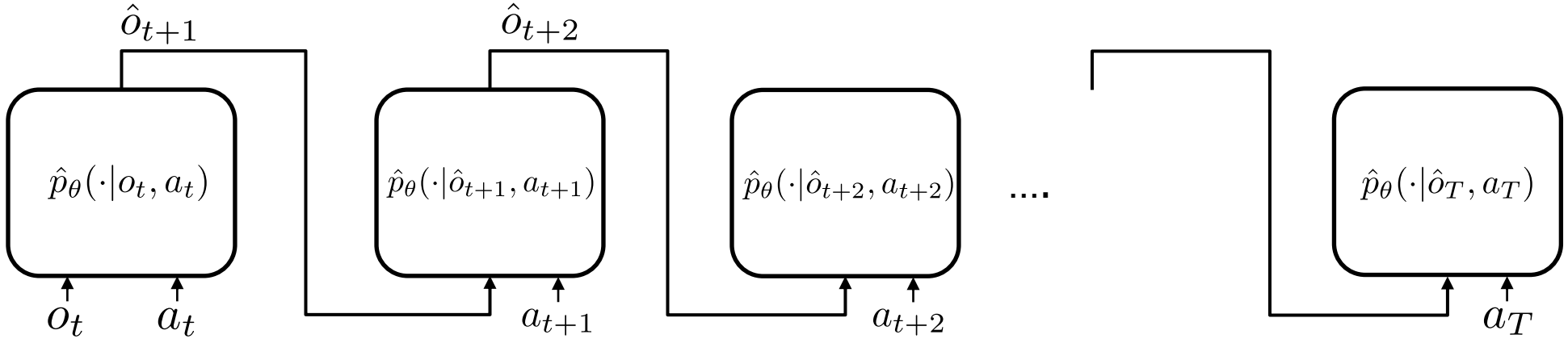


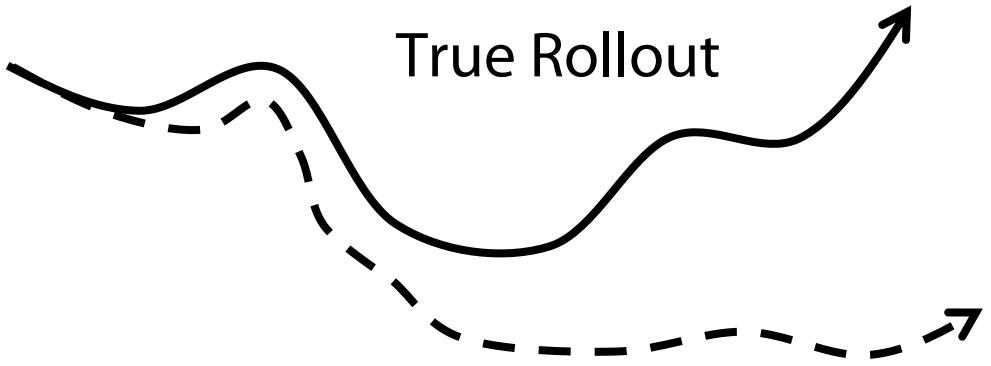


Sample autoregressively through the model, optimize for **optimal** actions

Where does this break?

Compounding error with increasing observation size and horizon

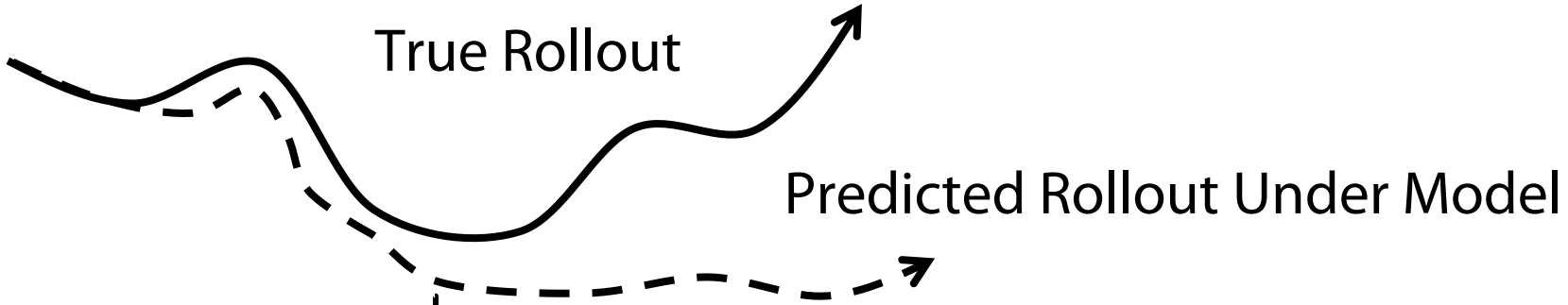




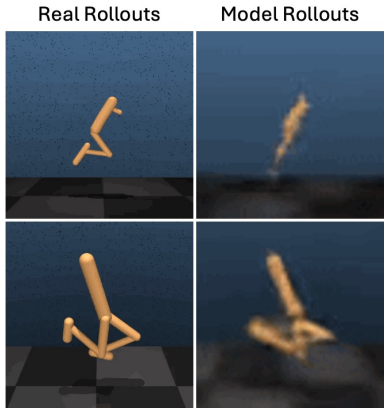
Predicted Rollout Under Model

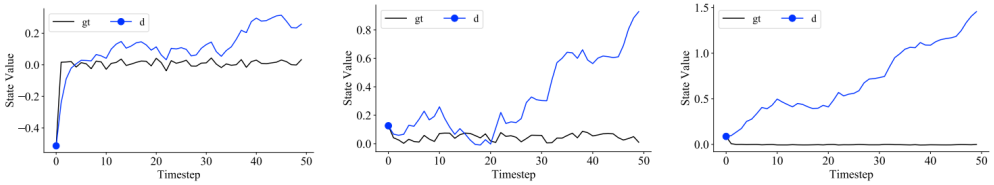
Small model errors lead to large compounding errors over time via autoregressive prediction

Where does this break?

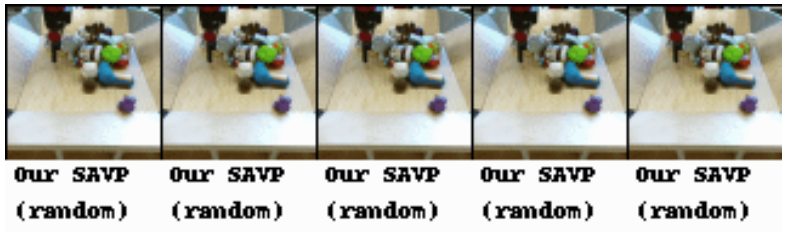


Scales poorly with horizon

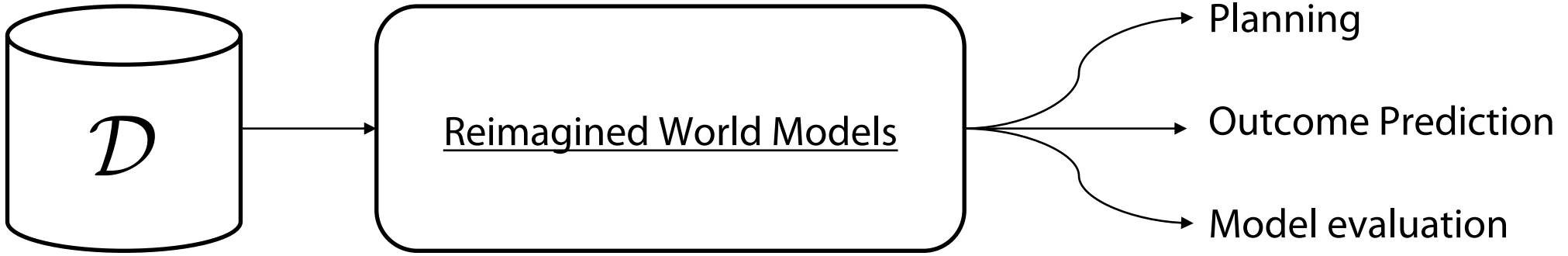


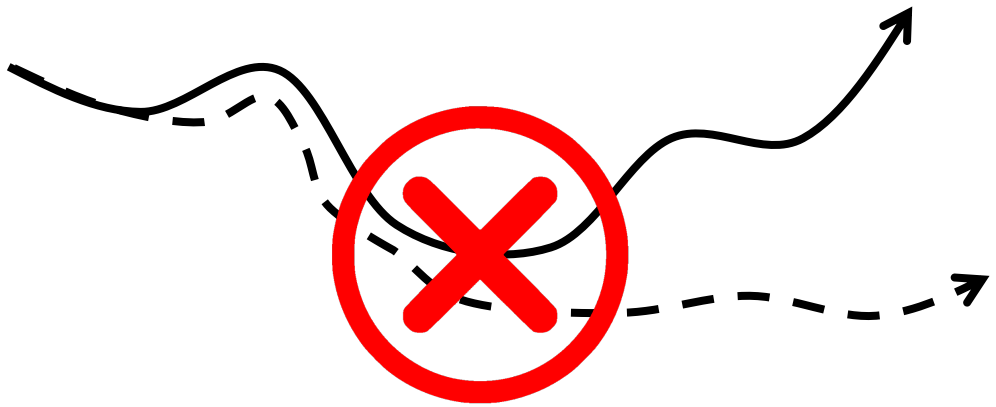


Scales poorly with observation size



What is our goal today?





Develop model-based RL algorithms that avoid compounding error and scale with:

1. Observation dimensionality
2. Horizon

⇩ How?

Build models that go beyond autoregressive, next-state prediction

What do I mean by a model?

Typical definition of a model $o_{t+1} \sim \hat{p}_\theta(\cdot|o_t, a_t)$

$\hat{p}_\theta(\cdot|o_t, a_t)$

Generative | Predicts things about the future given control | Autoregressive | 1-step | Predict Observation

Retain key elements of a model

Observation

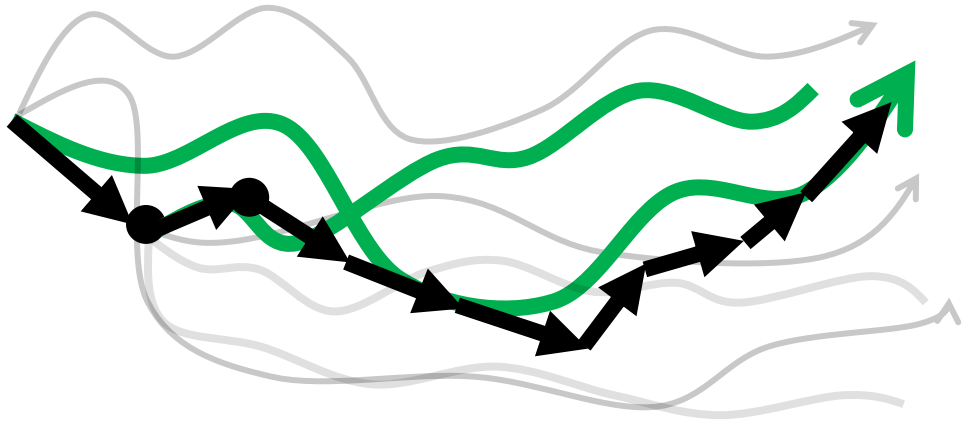
Control

(Need not be action)

$\mathcal{M}$

(Need not be next state)

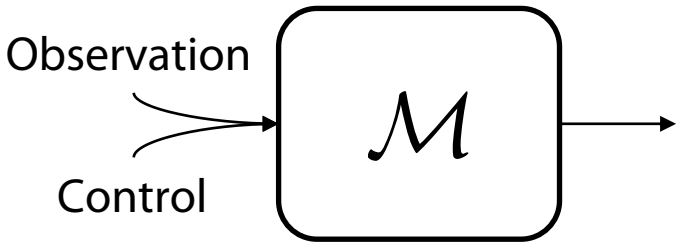
Something about the future



Planning procedure

(Need not be autoregressive sampling)

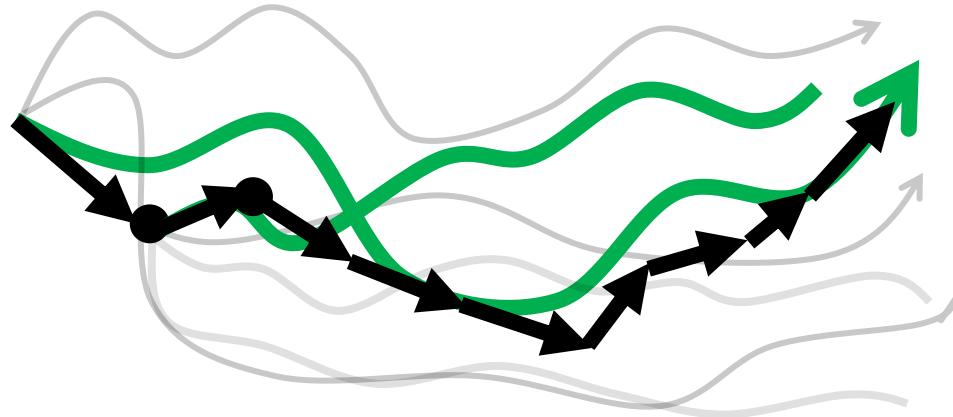
What is this talk about?



(Need not be action)

(Need not be next state)

Something about the future



Planning procedure

(Need not be autoregressive sampling)

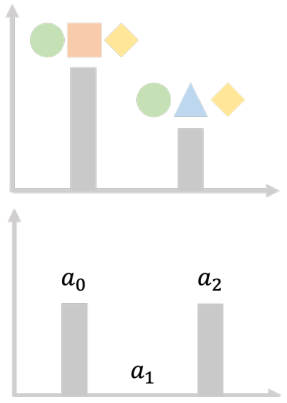
1. Change what we predict?

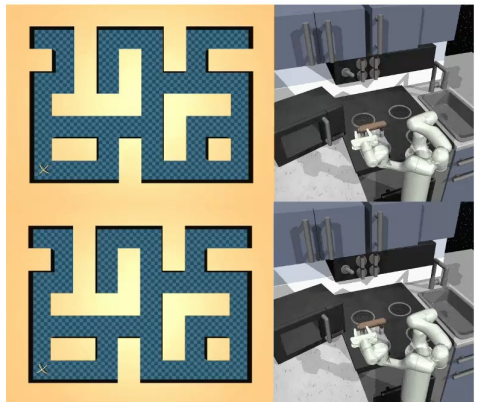
2. Change how we predict

Talk outline

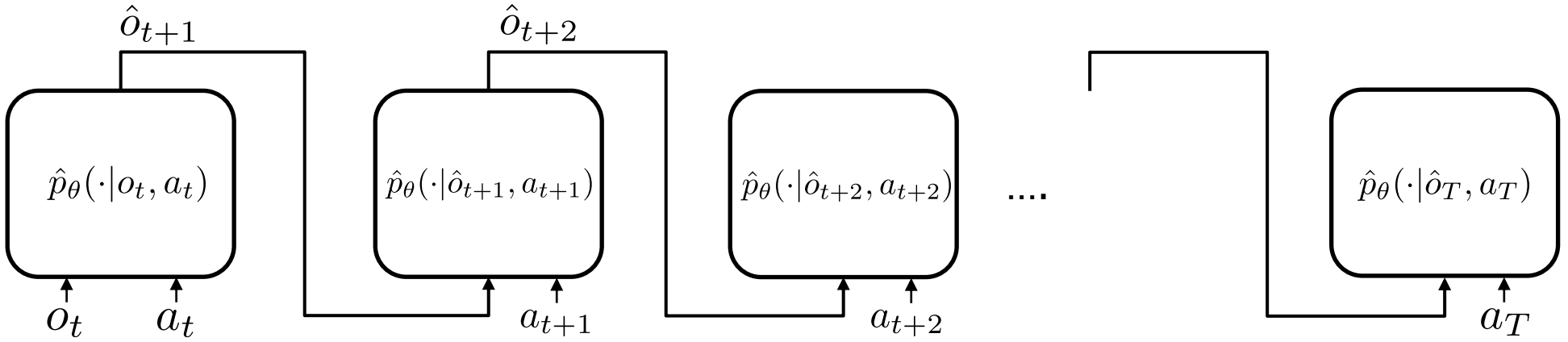
Learning from Task-Relevant Objectives (Observation Size)

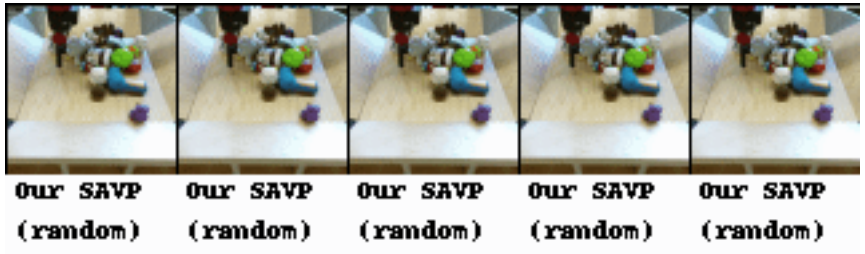
Learning Models of Cumulative Outcomes (Horizon)





Challenges with Model-Based RL via Video Prediction

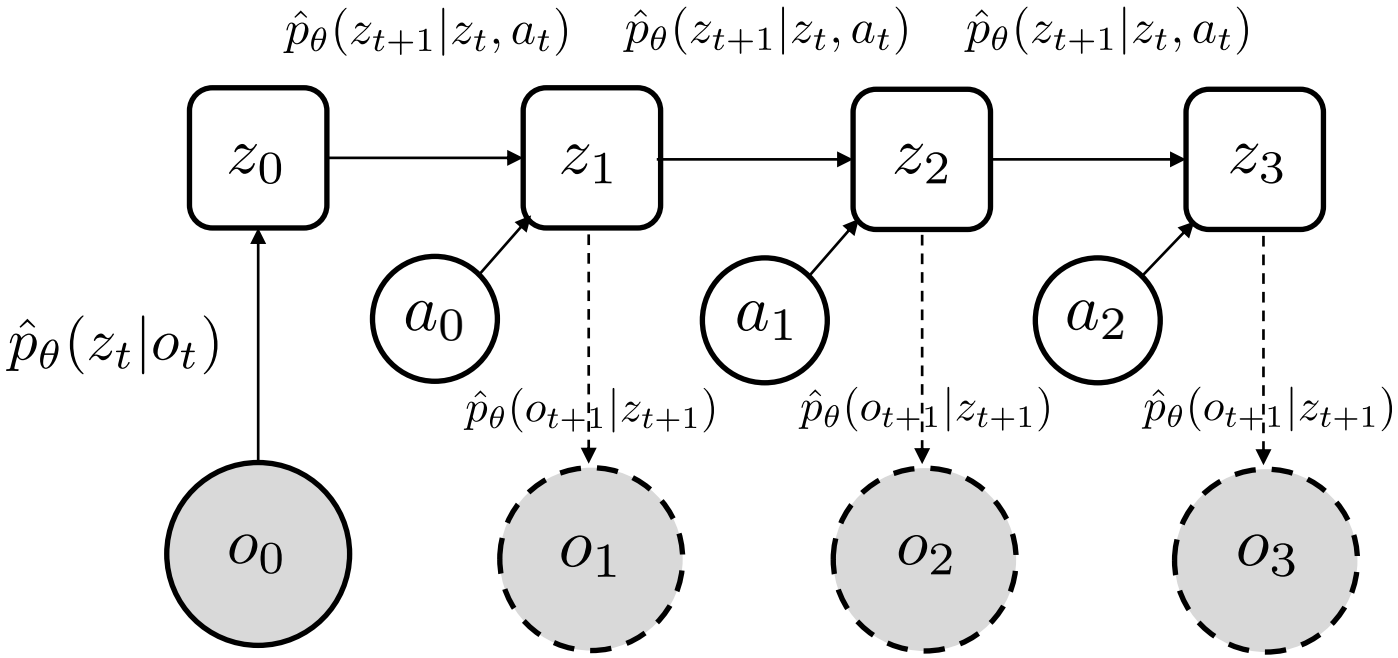




Model predictions often get blurry or hallcuinate

Do we even need to predict next observation?

What if we just made predictions in a "latent" space?

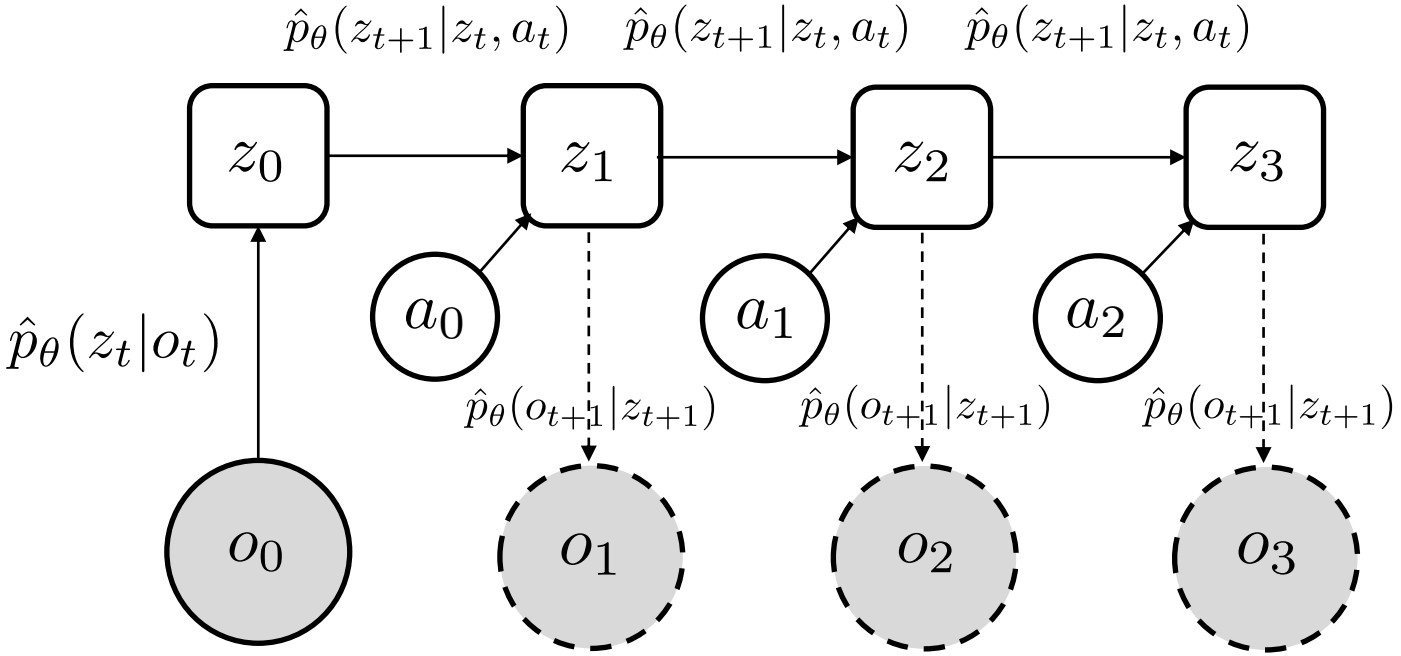


Can do multistep prediction in latent space rather than going back to images

What latent space is a good latent space?

Option 1: lossless compression of state ⟶ Reconstruction based objective

Where does this fail?



Struggles with cluttered, dynamic environments

Struggles with visual distribution shift

Real

Predicted

Real

Predicted

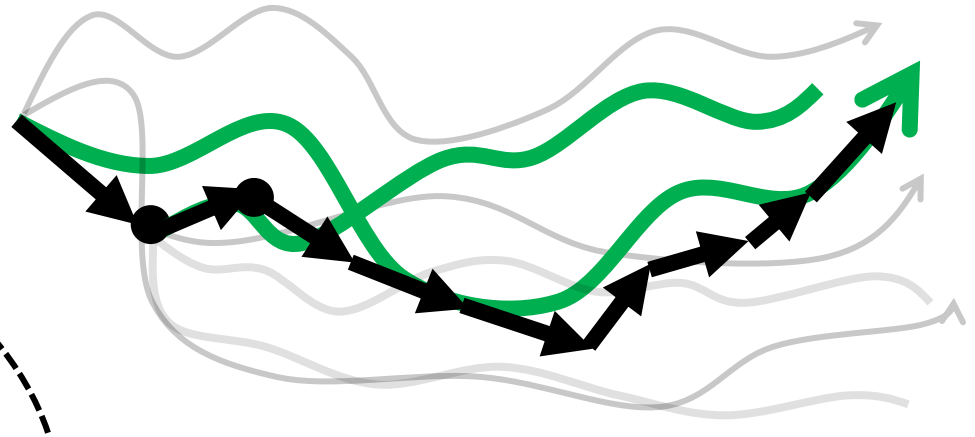
Model Learning as a Surrogate for Control

$$\arg \max_{a_0, a_1, \ldots, a_T} \sum_{t=0}^{T} r(\hat{z}_t, a_t)$$

$$\hat{z}_{t+1} \sim \hat{p}_\theta(\cdot | \hat{z}_t, a_t)$$

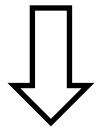
$$\hat{z}_1 \sim \hat{p}_\theta(\cdot | z_0, a_0)$$

$$z_0 \sim \hat{p}_\theta(\cdot | o_0)$$



The primary reason to predict future observations is to predict current and future rewards

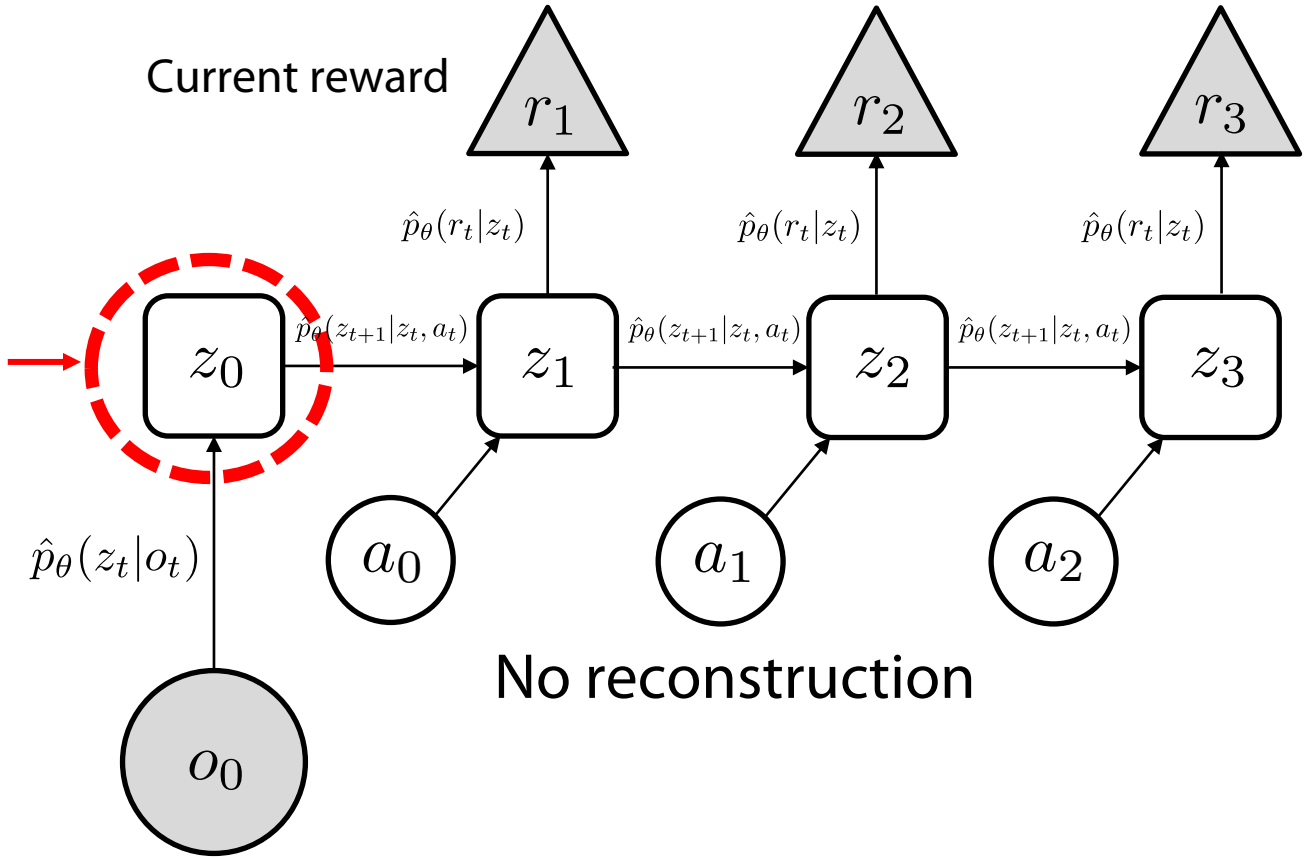
Proposition: a good latent to predict is one that predicts next latent and current reward



Option 2: lossy compression of state ⟶ Reward prediction objective

Key Insight: Model Latents Predictive of Future Reward

Option 2: lossy compression of state ⟶ Reward prediction objective

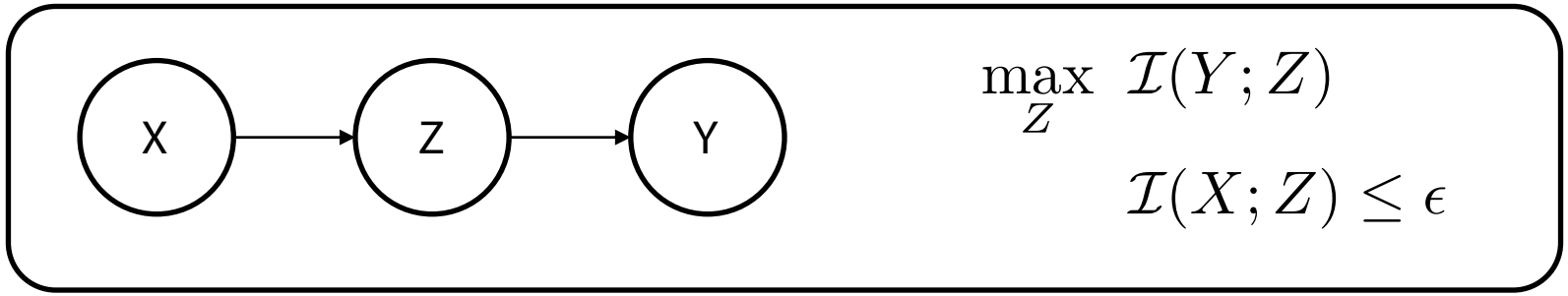


No reconstruction

Like a model, but only predict what "matters"

Model of future reward predictive latents, rather than observations

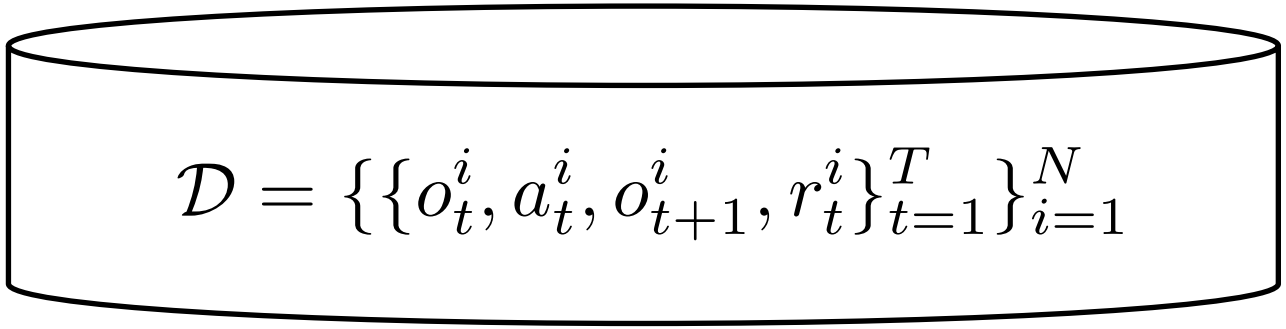
Let's resolve: Information Bottleneck



$$\max_Z \ \mathcal{I}(Y;Z)$$

$$\mathcal{I}(X;Z) \leq \epsilon$$

Apply to sequential dataset



$$\mathcal{D} = \{\{o_t^i, a_t^i, o_{t+1}^i, r_t^i\}_{t=1}^T\}_{i=1}^N$$

$$\max_Z \ \mathcal{I}_p(z_{1:T}; r_{1:T} | a_{1:T})$$

Be predictive of future reward

$$\mathcal{I}_p(z_{1:T}; o_{1:T} | a_{1:T}) \leq \epsilon$$

Be minimal

RePo: Resilient Model-Based Reinforcement Learning by Regularizing Posterior Predictability, Zhu et al, NeurIPS 2023 (Spotlight)

Let's derive: Variational Relaxation

$$\max_{Z} \ \mathcal{I}_p(z_{1:T}; r_{1:T} | a_{1:T})$$

Be predictive of future reward

$$\mathcal{I}_p(z_{1:T}; o_{1:T} | a_{1:T}) \leq \epsilon$$

Be minimal

Variational Relaxation

$$\mathcal{I}_p(z_{1:T}; r_{1:T} | a_{1:T}) \geq \mathbb{E}_p \left[\sum_{t=1}^{T} \log q_r(r_t | z_t) \right]$$

Reward predictor

$$\mathcal{I}_p(z_{1:T}; o_{1:T} | a_{1:T}) \leq \mathbb{E}_p \left[\sum_{t=1}^{T-1} D_{\mathrm{KL}}(p(\cdot | z_t, a_t, o_{t+1}), q_z(\cdot | z_t, a_t) \right]$$

Posterior

Prior

RePo: Resilient Model-Based Reinforcement Learning by Regularizing Posterior Predictability, Zhu et al, NeurIPS 2023 (Spotlight)

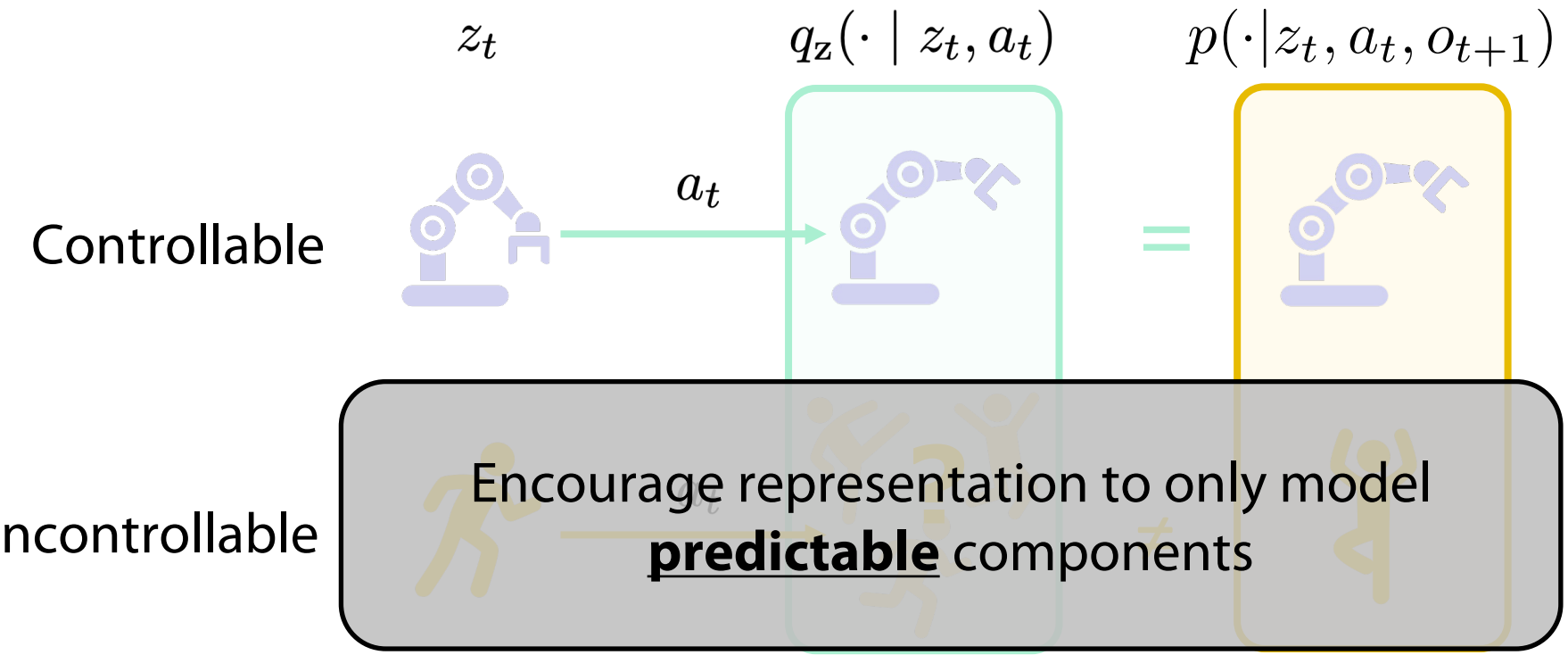
Models that Predict only the Predictable

$$\max_{Z} \; \mathcal{I}_p(z_{1:T}; r_{1:T} | a_{1:T}) \;\; \text{s.t} \;\; \mathcal{I}_p(z_{1:T}; o_{1:T} | a_{1:T}) \leq \epsilon$$

reward prediction

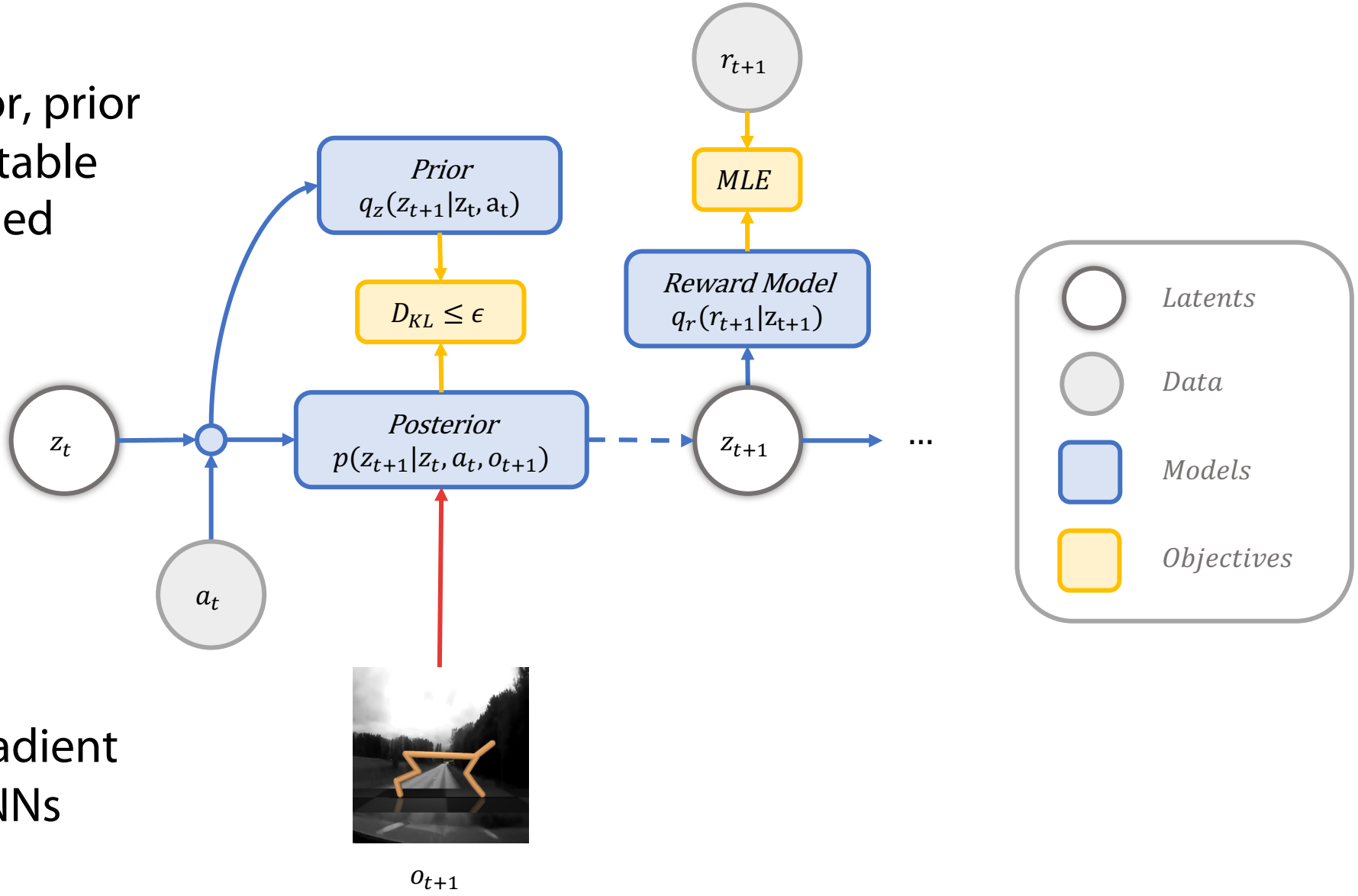
dynamics consistency + predictability

$$\max_{p, q_r, q_z} \min_{\beta} \mathbb{E}_p \left[\sum_{t=1}^{T} \log q_r(r_t | z_t) \right] + \beta \left(\mathbb{E}_p \left[\sum_{t=1}^{T-1} D_{\mathrm{KL}}(p(\cdot | z_t, a_t, o_{t+1}), q_z(\cdot | z_t, a_t) \right] - \epsilon \right)$$



RePO: Model Architecture

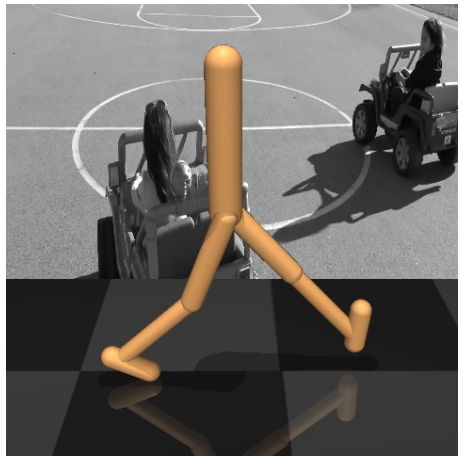
Learn encoder, posterior, prior such that only predictable elements are retained

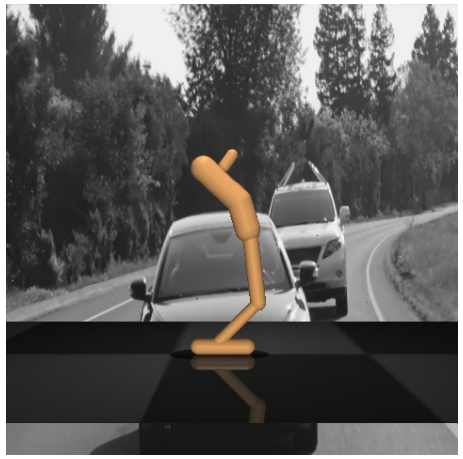


Optimize with dual gradient descent and deep NNs

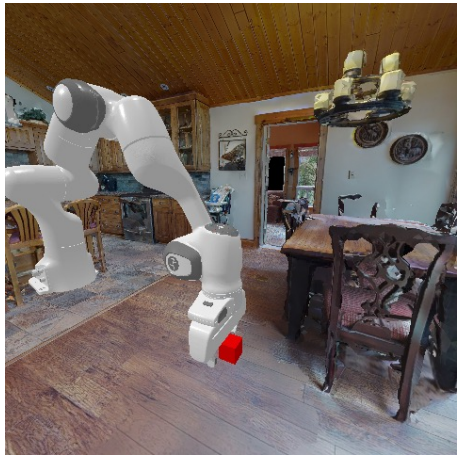
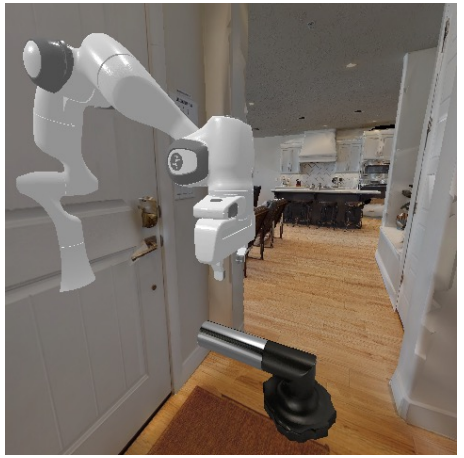
Does this work?

Distracted DeepMind Control

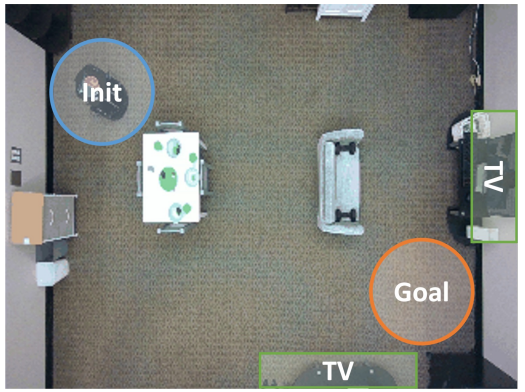




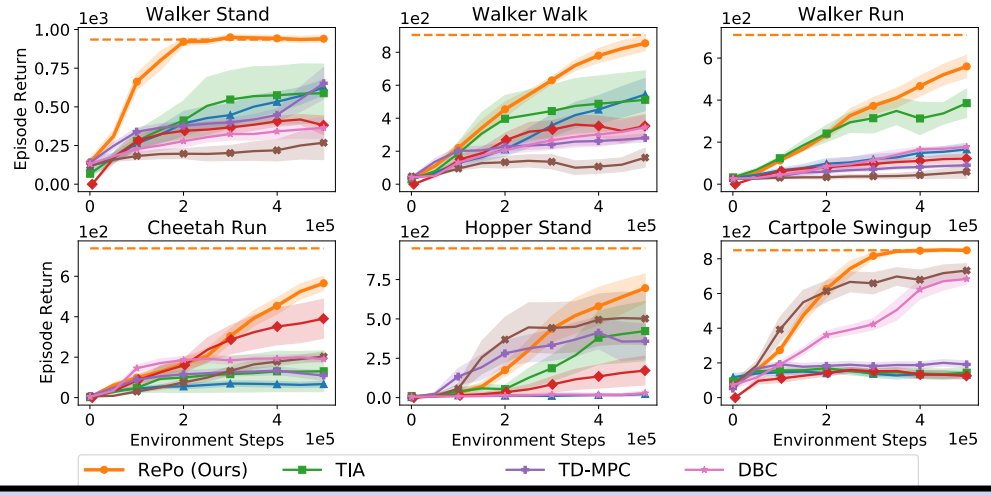

Realistic Maniskill

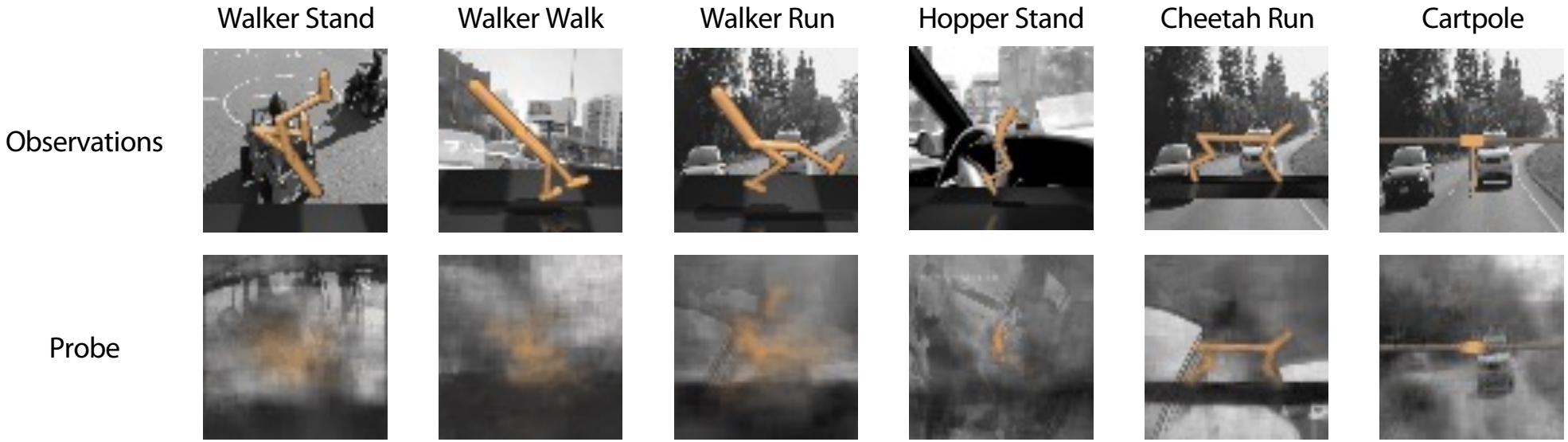
Lazy TurtleBot



Does RePo learn in dynamic environments?

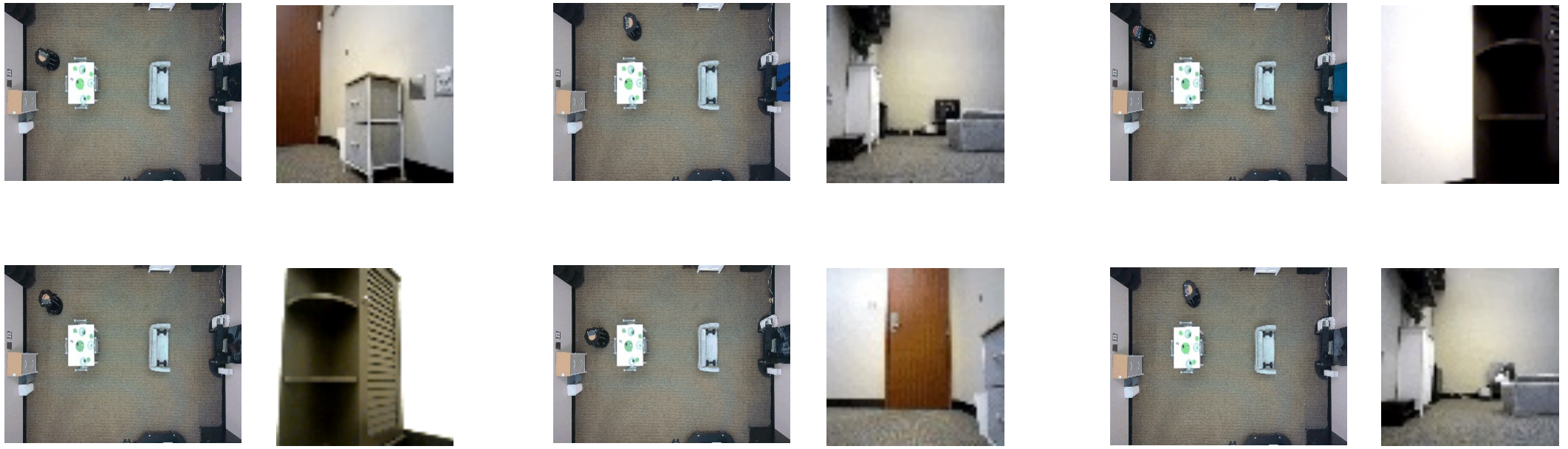


RePo learns a task-relevant representation and ignores the distractors.



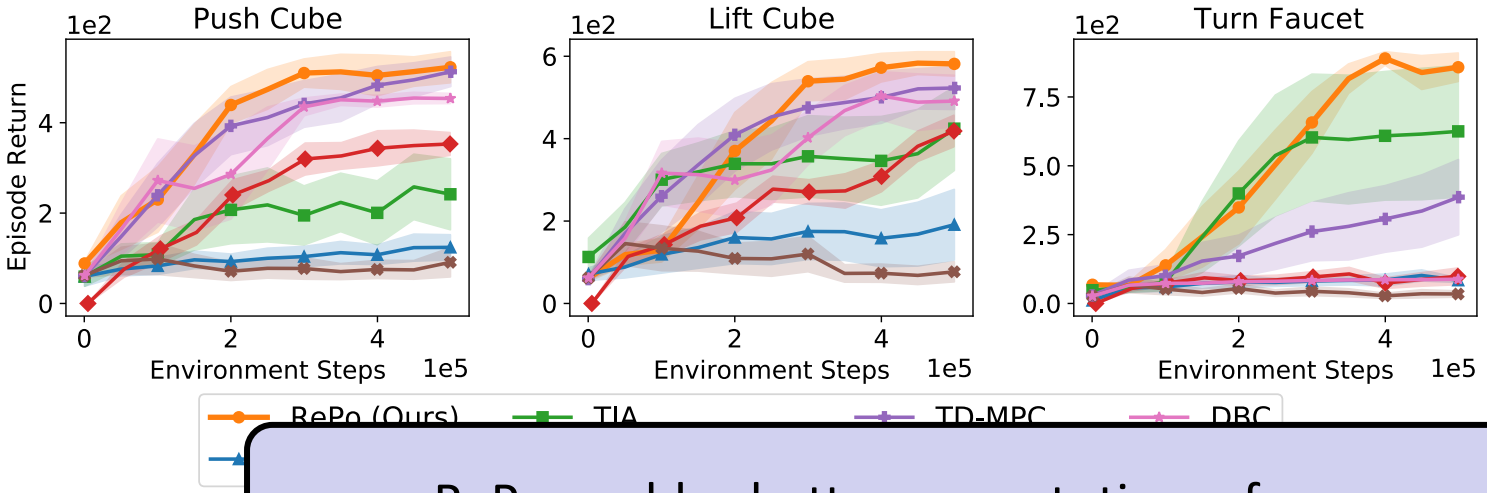
Does RePo learn in dynamic environments?

	Success	Return
RePo (Ours)	**62.5%**	**-24.3**
Dreamer [15]	0.0%	-61.7

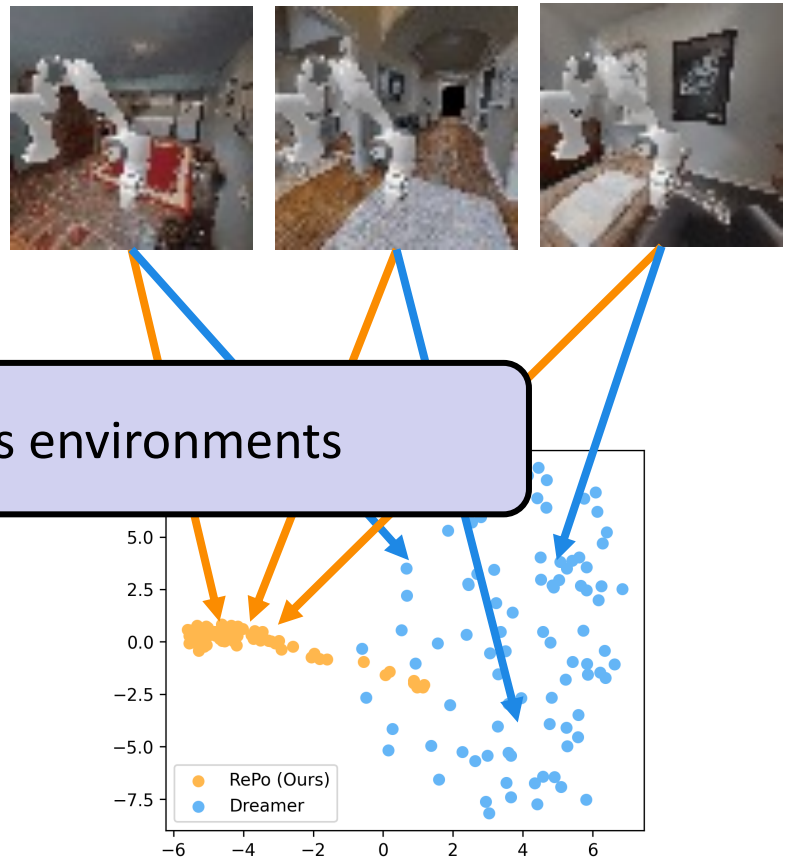


RePo achieves nontrivial success rate in 15K environment steps!

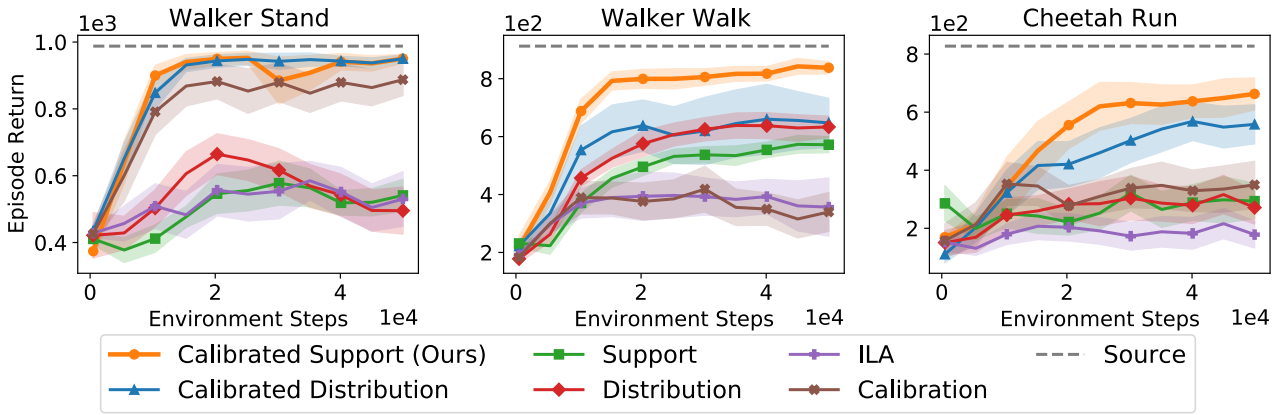
Does RePo learn across diverse environments?



RePo enables better asymptotic performance across environments



Does RePo representation transfer in distribution shifts?



RePo enables quick adaptation across distribution shifts

Pre-Adaptation



Post-Adaptation



Takeaways

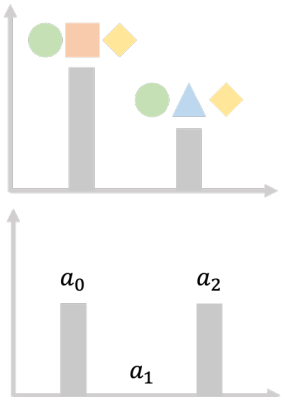
Models don’t need to model future observations,
→ predict task relevant quantities such as minimal, reward relevant latents

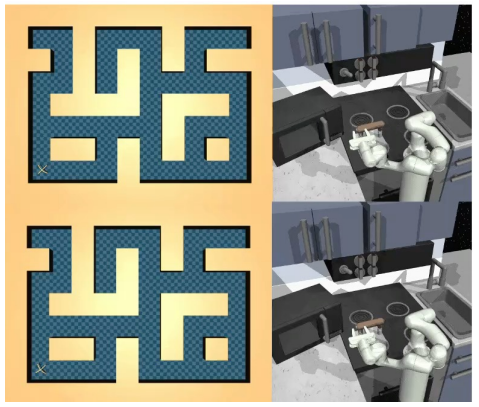
Change *what* to predict

Talk outline

Learning from Task-Relevant Objectives (Observation Size)

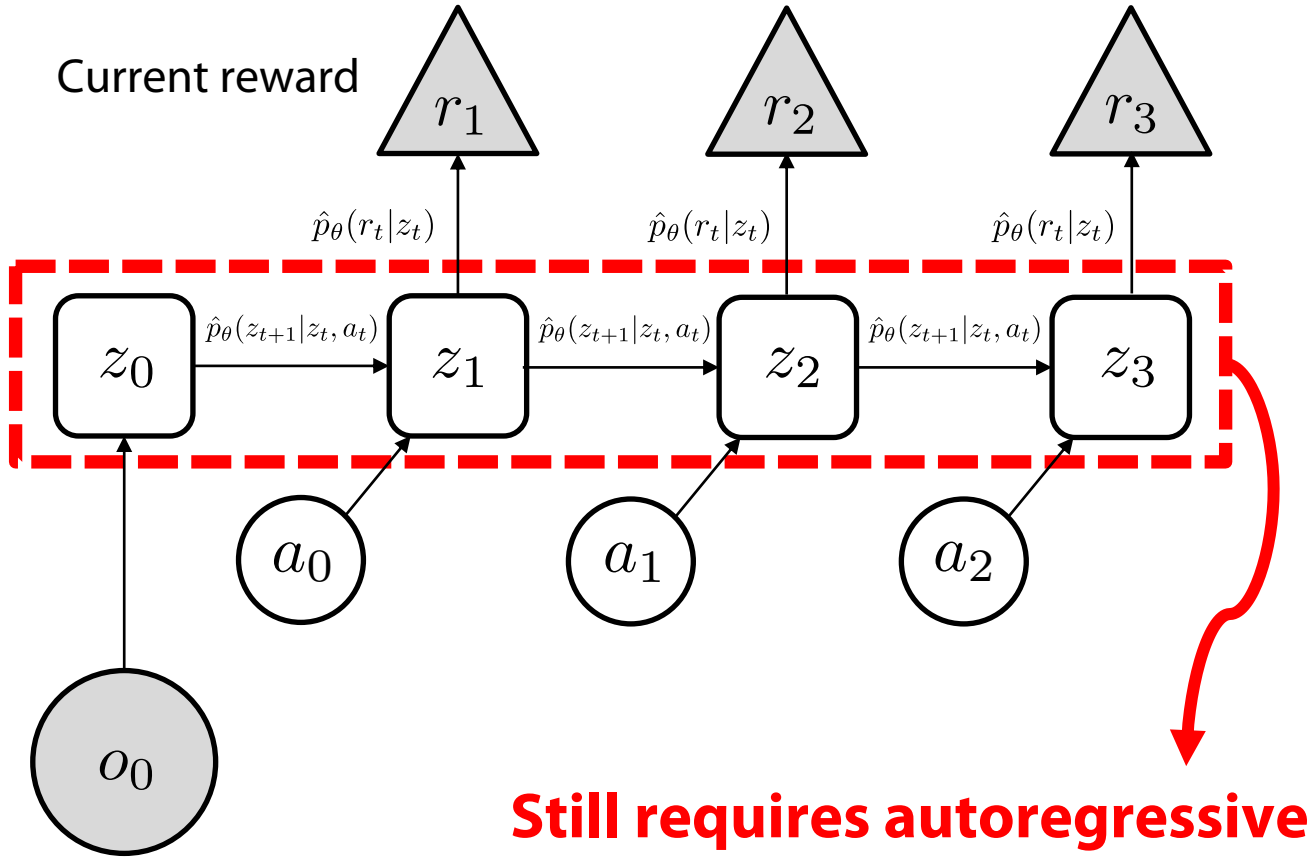
Learning Models of Cumulative Outcomes (Horizon)





Compounding Error with Autoregressive Prediction

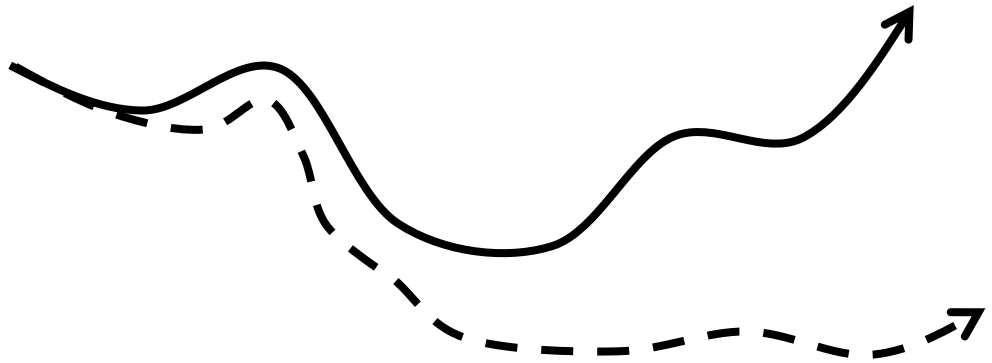
Does latent prediction alleviate the problem with long-horizon modeling?



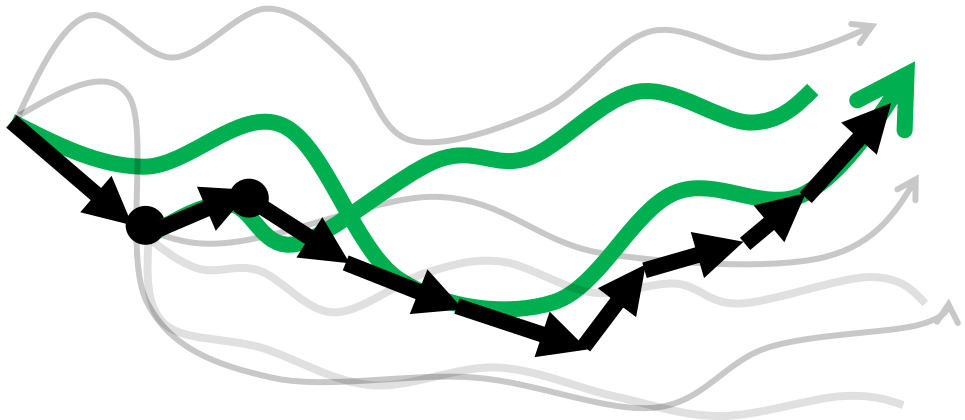
Still requires autoregressive prediction, doesn't eliminate the problem of compounding error

How might we avoid compounding error?

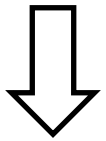
Predictions quickly deviate from ground truth



Learned dynamics models are prone to compounding error



What if we changed the ***class*** of models themselves to avoid autoregressive generation?



What if we just directly modeled cumulative, long-term outcomes?

Model-free off-policy RL

What do we mean by model-free off-policy RL?

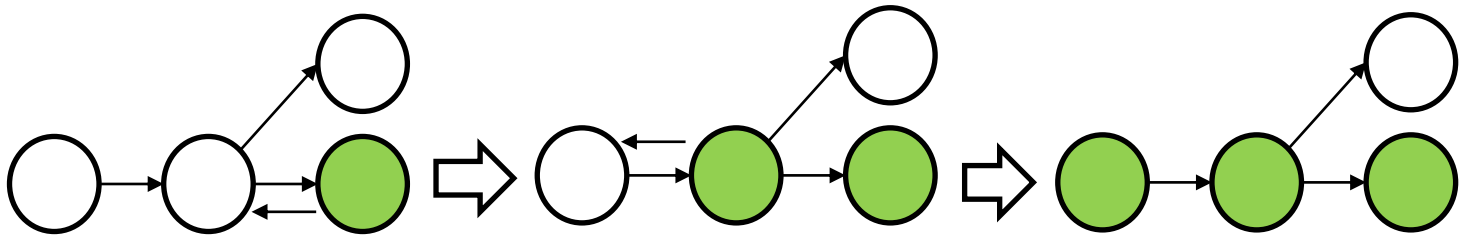
Key idea:

1. Use dynamic programming to model the future **sum** of rewards under a policy **(policy evaluation)**
2. Use this estimate to improve the policy **(policy improvement)**

$$Q_r^\pi(s,a) = \mathbb{E}_{\pi, p(\cdot|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \mid s_0 = s, a_0 = a\right]$$

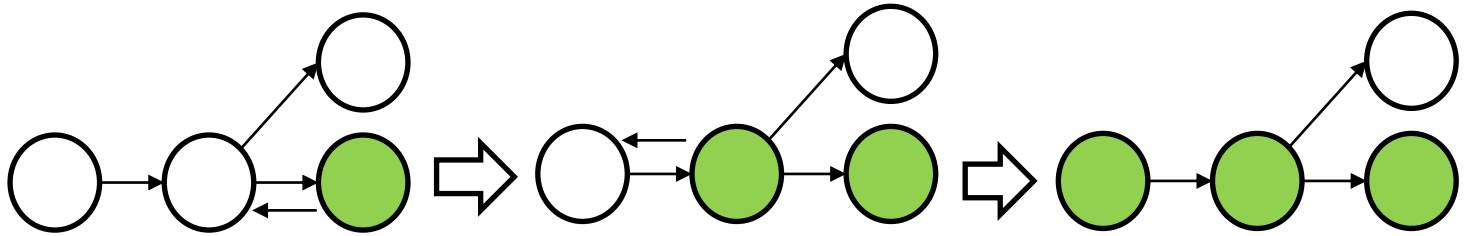
$$Q_r^\pi(s_t,a_t) \leftarrow r(s_t) + \gamma \mathbb{E}_{p(s_t'|s_t,a_t)}\left[\mathbb{E}_{\pi(a_t'|s_t')}\left[Q_r^\pi(s_t',a_t')\right]\right] \quad \text{Policy evaluation}$$

$$\pi_r \leftarrow \arg\max_\pi \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi(a|s)}\left[Q_r^\pi(s,a)\right] \quad \text{Policy improvement}$$



+ Avoids autoregressive generation
+ Performant over long horizons

Does this satisfy the needs of long-horizon modeling?



Q-functions are expected sum of **rewards** under a **particular policy**

$$Q^{\pi}_{r}(s,a) = \mathbb{E}_{\pi, p(\cdot|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \mid s_0 = s, a_0 = a\right]$$

Q-functions are like models, but inherently **reward** and **policy** dependent

Need to resolve!

How do we build models without autoregressive next-step prediction?

How can we develop off-policy RL methods without reward and policy dependence?

Distributional Successor Features for Zero-Shot Policy Optimization

Distributional Successor Features for Zero-Shot Policy Optimization
(DiSPOs)

Key idea: Use TD learning to model cumulative outcomes w/ reward/policy independence:

Reward Independence: Rewards → Modeling random cumulants ←
Policy Independence: single policy → Modeling every possible outcome

New class of world models that enable transfer without compounding error

Distributional Successor Features Enable Zero-Shot Policy Optimization, Zhu et al NeurIPS 2024

Generalized Value Functions: From Rewards to Cumulants

Let's revisit the definition of value functions

$$Q_r^\pi(s,a) = \mathbb{E}_{\pi, p(\cdot|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \mid s_0 = s, a_0 = a\right]$$

This depends on knowledge of a particular r

What if we replaced r with an arbitrary, vector-valued function g(s)?

$$\psi^\pi(s,a) = \mathbb{E}_{\pi, p(\cdot|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t) \mid s_0 = s, a_0 = a\right] \quad g \in \mathcal{R}^k$$ Cumulant

Everything remains the same, just vector-valued regression rather than scalar regression

Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction, Sutton et al

Successor Features: Linear cumulants allow for transfer

What if we replaced r with an arbitrary, vector-valued function g(s)?

$$\psi^{\pi}(s,a) = \mathbb{E}_{\pi, p(\cdot|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t) \mid s_0 = s, a_0 = a\right]$$

But why? → Not specific to a task, allows us to transfer across tasks!

Successor Features: if rewards are linear in cumulants, (Q) values are linear in summed cumulants

(Through linearity of expectation)

$$r(s) = w_r^T g(s)$$

$$Q_r^{\pi}(s,a) = w_r^T \psi^{\pi}(s,a)$$

Task-specific linear regression

Task-agnostic Dynamic Programming

Transfer has been reduced to linear regression!

Successor Features for Transfer in RL, Barreto et al

DiSPOs: Linearity Through Random Features

Key idea: designing features is hard, just use high-dimensional random features

$$r(s) = w_r^T g(s)$$ ⟵ Set to random Fourier features

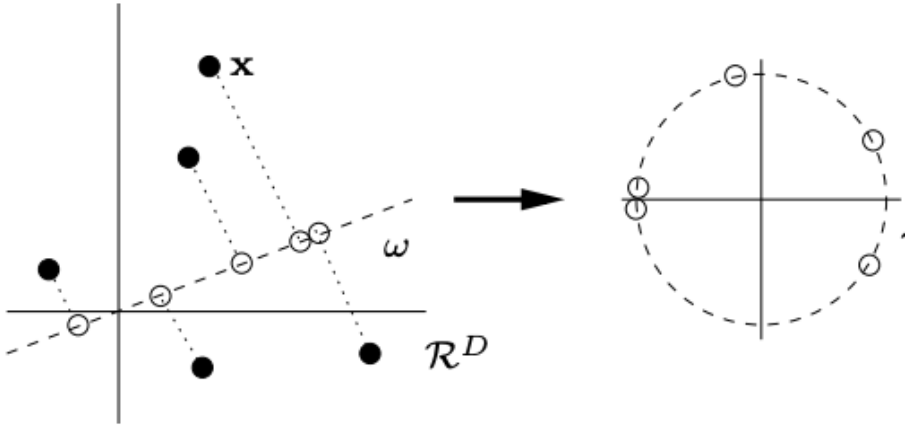
$$Q_r^\pi(s,a) = w_r^T \psi^\pi(s,a)$$

Algorithm Learned Fourier Features (LFF)

```
class LFF(nn.Linear):
  def __init__(self, in, out, b_scale):
    super().__init__(in, out)
    init.normal_(self.weight, std=b_scale/in)
    init.uniform_(self.bias, -1.0, 1.0)

  def forward(self, x):
    x = np.pi * super().forward(x)
    return torch.sin(x)
```

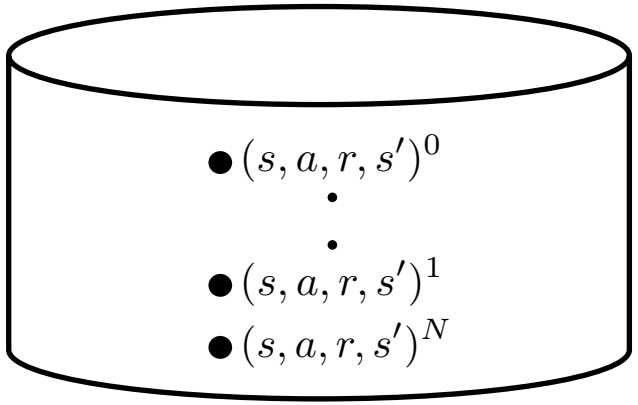
Random kitchen sink features
make linear regression easy!

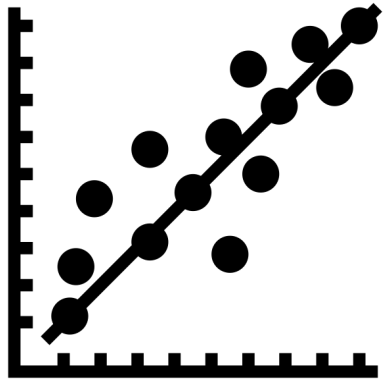


What does this mean for transfer?

$$r(s) = w_r^T g(s) \qquad Q_r^\pi(s,a) = w_r^T \psi^\pi(s,a)$$

Reward transfer has been reduced to linear regression!





$$Q_r^\pi(s,a) = w_r^T \psi^\pi(s,a)$$

Small dataset of (s, r) pairs

Simple linear regression

Q-values in an instant

Insight: Model accumulated random features rather than rewards to be reward agnostic

Insights for Reward Independence

Model Q-functions of random features rather than rewards

Transfer becomes linear regression

Distributional Successor Features for Zero-Shot Policy Optimization

Distributional Successor Features for Zero-Shot Policy Optimization
(DiSPOs)

Key idea: Use TD learning to model cumulative outcomes w/ reward/policy independence:

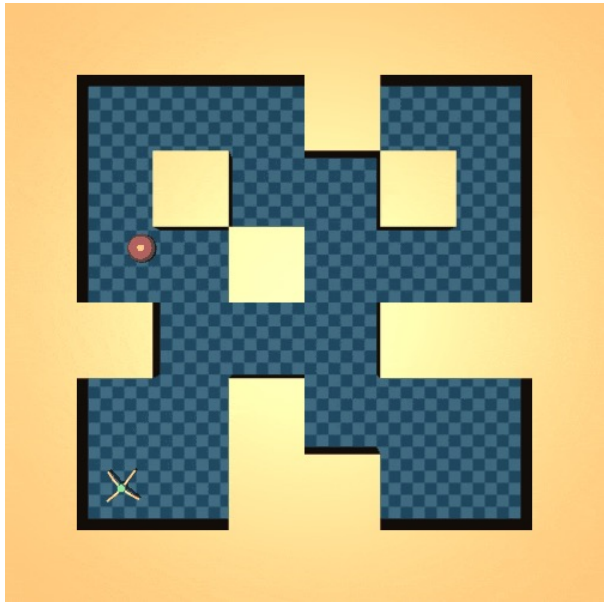
Reward Independence: Rewards → Modeling random cumulants
Policy Independence: single policy → Modeling every possible outcome ⟵

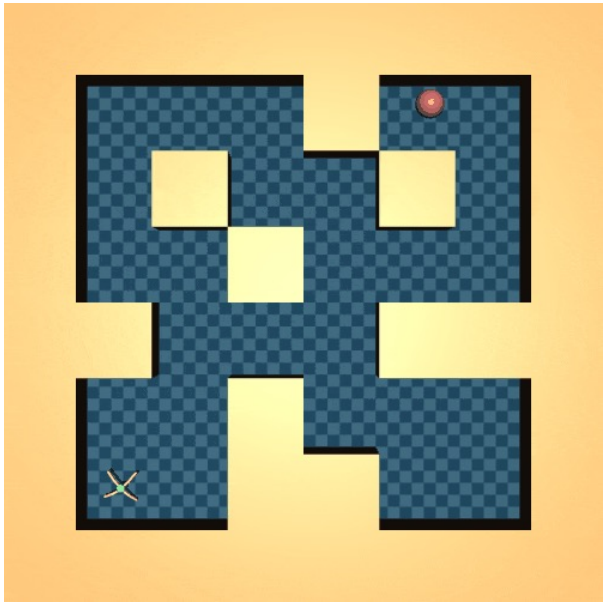
Why is policy dependence problematic?

Let's take a closer look at value functions

$$\psi^{\pi}(s,a) = \mathbb{E}_{\pi, p(\cdot|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t) \mid s_0 = s, a_0 = a\right]$$

With the previous linearity-based tool, we can evaluate new r, but we cannot evaluate new π





New rewards need new policies!

But we don't know what policy to evaluate apriori!

How can we eliminate policy dependence?

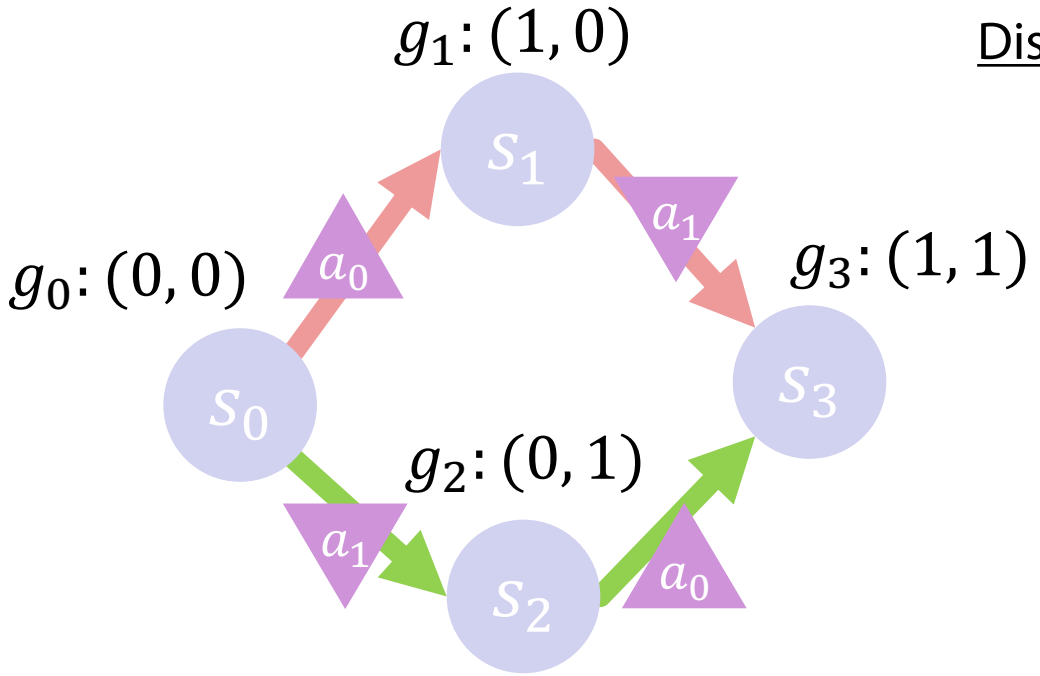
Key idea: if we don't know what policy needs to be evaluated → model them all!

Model a distribution over all possible outcomes $p(\psi|s)$

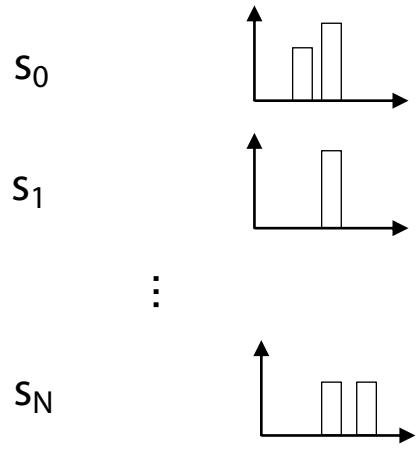
Current value functions

s_0	$\psi(s_1)$
s_1	$\psi(s_2)$
⋮	
s_N	$\psi(s_N)$

Expected cumulant sum under **a** policy π



Distributional cumulative outcomes



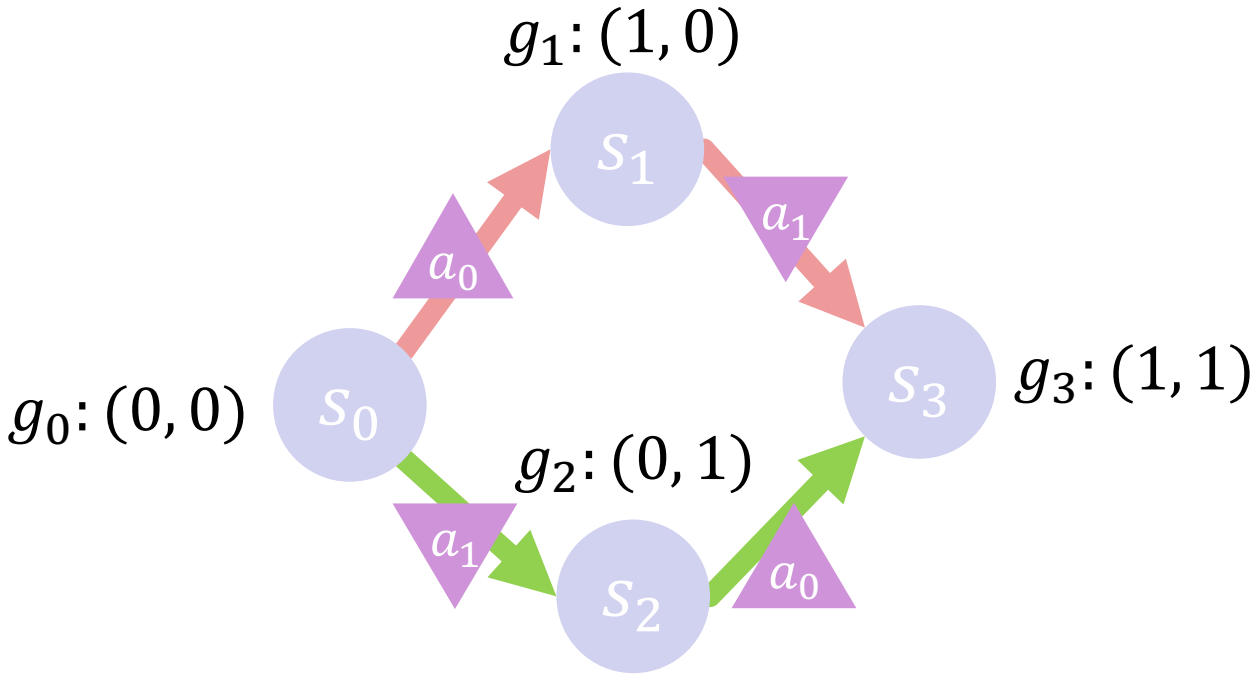
Distribution over all possible cumulant sums under dynamics

Intuition: Distribution is over all **possible** cumulative outcomes ψ from a particular state

What does the distribution over ψ correspond to?

ψ corresponds to a single cumulative outcome in the environment

p(ψ|s) corresponds to the distribution of all the things that could happen in an environment

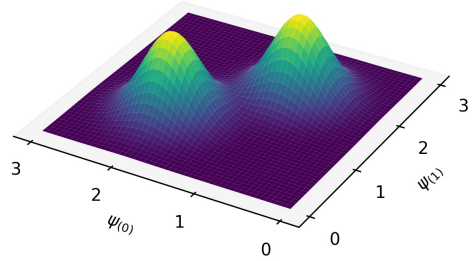


Let's consider a state s

Path 1: $\psi^0 = g_0 + g_1 + g_3 = (2, 1)$

Path 2: $\psi^1 = g_0 + g_2 + g_3 = (1, 2)$

$p(\psi|s)$



Why is this good? → if you represent all possible outcomes, then you can pick the best one for a task

How can we learn the distribution $\mathbf{p}(\psi|s)$?

p(ψ|s) corresponds to the distribution of all the things that <u>could</u> happen in an environment

(informal) p(ψ|s) = ∪ g(s) + p(ψ|s′) over all (s, a, s′)

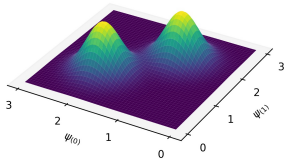
Current cumulant

Sample from future distribution over accumulated ψ

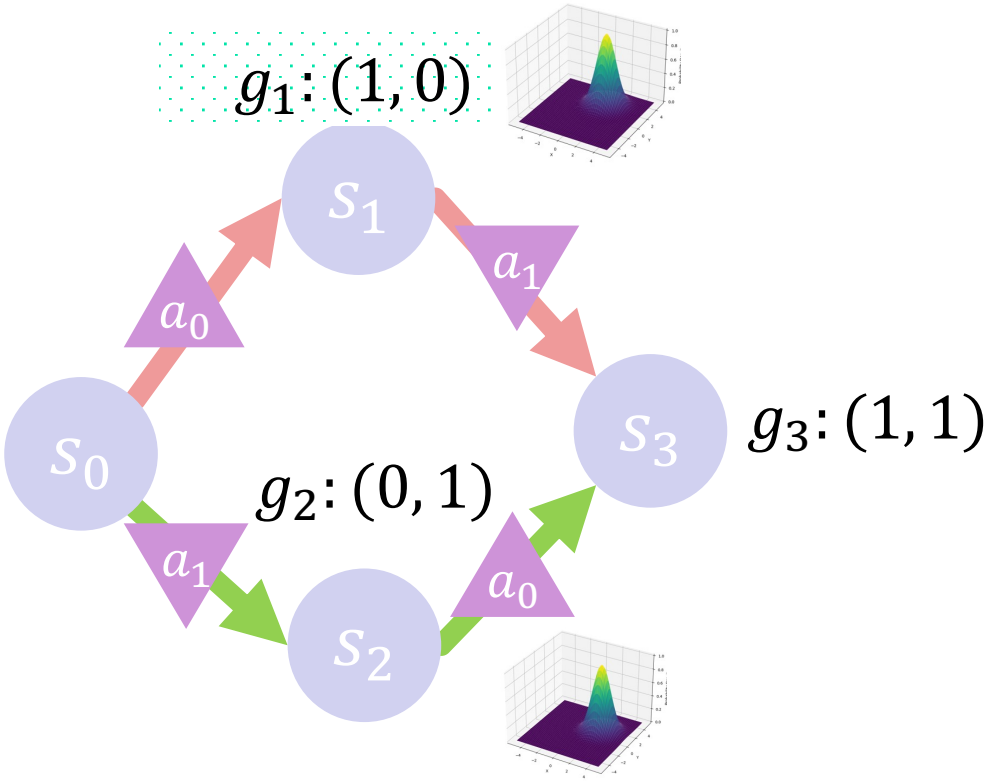
$$\max_{p_\theta(\psi|s)} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\log p_\theta(g(s) + \gamma\psi_{s'}|s)\right]$$

$$\text{s.t} \quad \psi_{s'} \sim p_\theta(\cdot|s')$$

<u>Distributional Bellman</u>

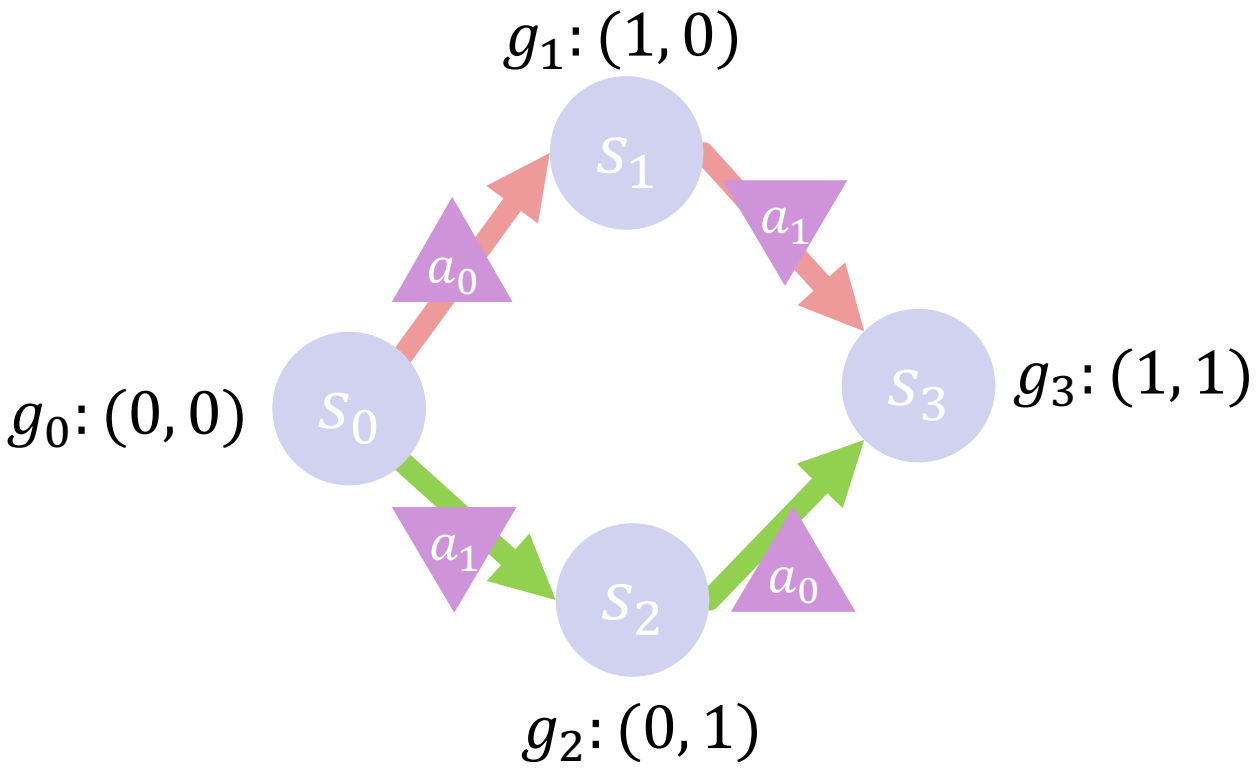


$g_0: (0, 0)$



Every ψ can be realized via an “inverse” model $\pi(a|s, \psi)$

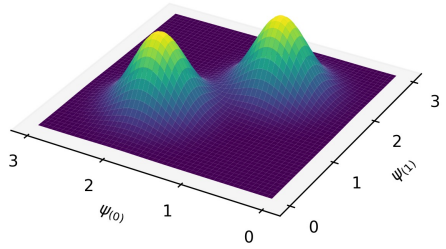
Let's run through an example



Computing $p(\psi|s)$:

$$\psi^0 = g_0 + g_1 + g_3 = (2, 1)$$

$$\psi^1 = g_0 + g_2 + g_3 = (1, 2)$$



Computing $\pi(a|s, \psi)$:

$$\pi(.|s_0, \psi = (1,2)) = a_1$$

$$\pi(.|s_0, \psi = (2,1)) = a_0$$

What can happen — How to accomplish it?

A DiSPO is a combination of $p(\psi|s)$and $\pi(a|s, \psi)$

Insight: Model distributions of possible outcomes to be policy agnostic

Insights for Policy Independence

Model distributions of cumulative outcomes

Avoids committing to any one policy, represent them all

DiSPOs – Putting it together

Distributional Successor Features for Zero-Shot Policy Optimization
(DiSPOs)

Key idea: Use TD learning to model cumulative outcomes w/ reward/policy independence:

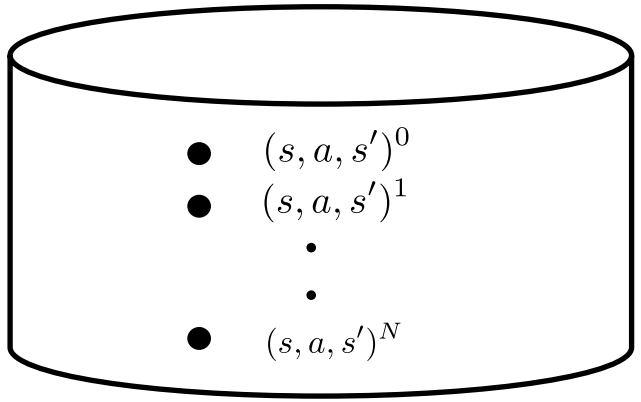
Reward Independence: Rewards → Modeling random cumulants
Policy Independence: single policy → Modeling every possible outcome

Learn $p(\psi|s)$and $\pi(a|s,\psi)$ using distributional TD learning, setting cumulant to random features g(s)

Ok, why does it matter for planning?

Allows for easy transfer to new problems!

Large corpus of unsupervised transitions



Learn DiSPOs

$$p(\psi|s)$$
$$\pi(a|s, \psi)$$

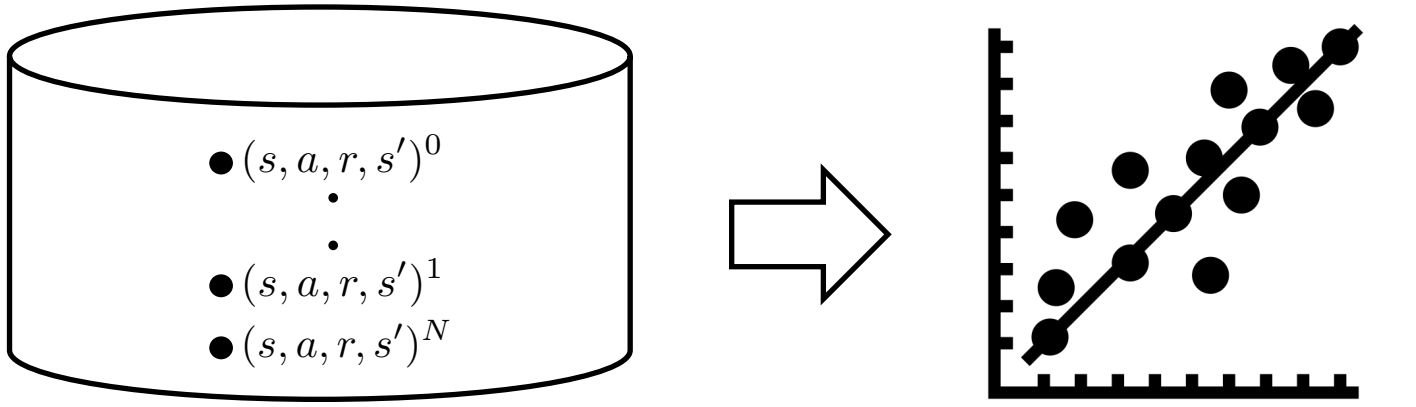
Quickly find optimal policies for new tasks

Can easily transfer to new tasks, without:

1. Compounding error
2. Redoing policy learning

Only need linear regression!

Using DiSPOs to transfer to new tasks: Linear Regression



Small dataset of (s, r) pairs

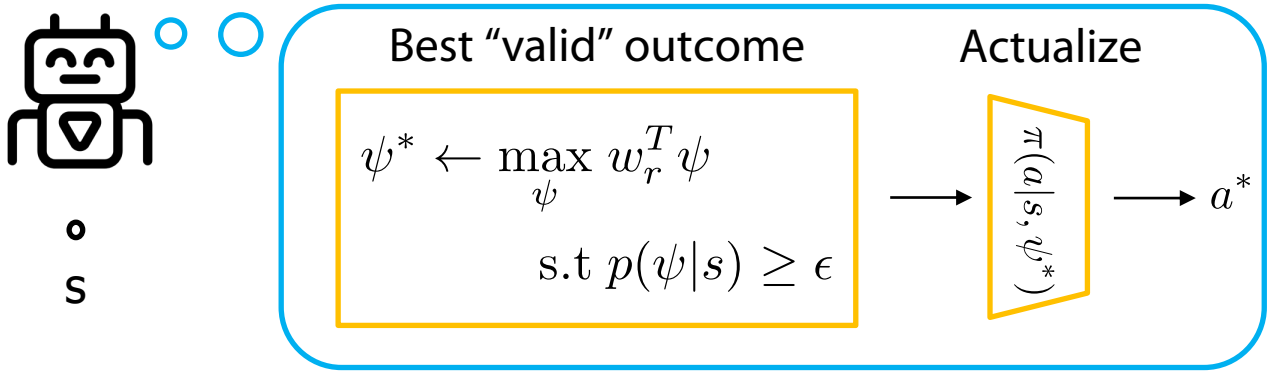
Simple linear regression

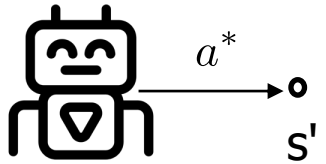
Step 1: Find linear embedding of reward function

$$\min_{w_r} \mathbb{E}_{(s,r)} \left[(r - w_r^T g(s))^2 \right]$$

Task-specific w_r that characterizes test-time rewards

Using DiSPOs to transfer to new tasks: Planning





Repeat!

Step 2: Find best “valid” outcome ψ and execute a

⇩ Gradient of Lagrangian

$$\nabla_\psi \mathcal{L}(\psi, \alpha) = w_r + \alpha \nabla_\psi \log p(\psi|s)$$

Very easy when $p(\psi|s)$ is a diffusion policy → guided diffusion

Models without Compounding Errors

Can easily transfer to **arbitrary** new tasks, without:

1. Compounding error
2. Redoing TD learning

✔ Avoids compounding error

Directly modeling cumulative outcomes

✔ Reward-independent

Model random features

✔ Policy-independent

Model all possible outcomes

Is this still a model?

Typical definition of a model

$$o_{t+1} \sim \hat{p}_\theta(\cdot|o_t, a_t)$$

$\hat{p}_\theta(\cdot|o_t, a_t)$

Generative

Predicts things about the future given control

Autoregressive

1-step

Predict Observation

Predict future "outcomes" rather than observation

Distributional Successor Features

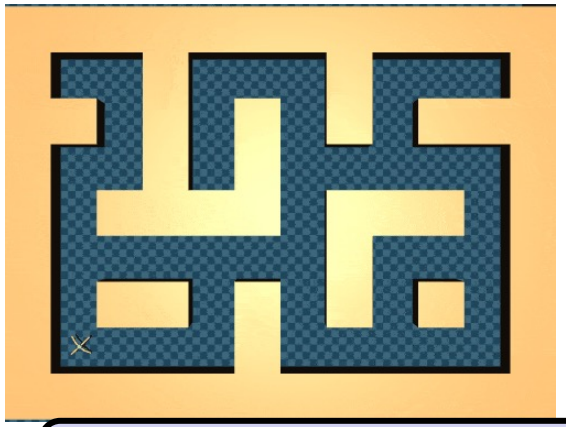
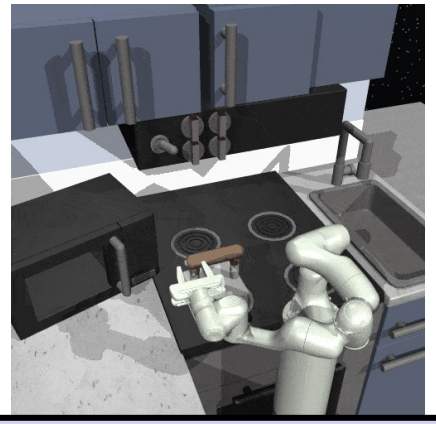
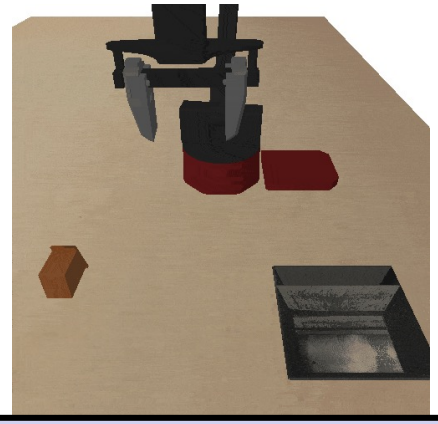
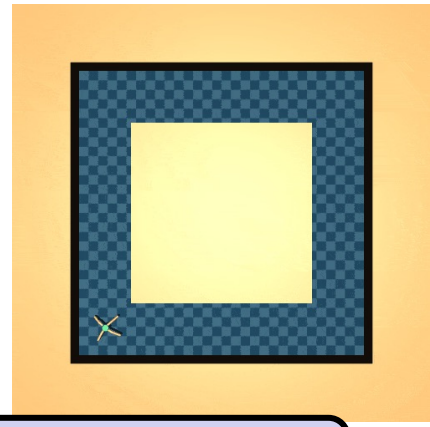
$p(\psi|s)$ $\pi(a|s, \psi)$

Remove 1-step prediction

Remove autoregression

Does it work?

Measure the ability to transfer behaviors across rewards

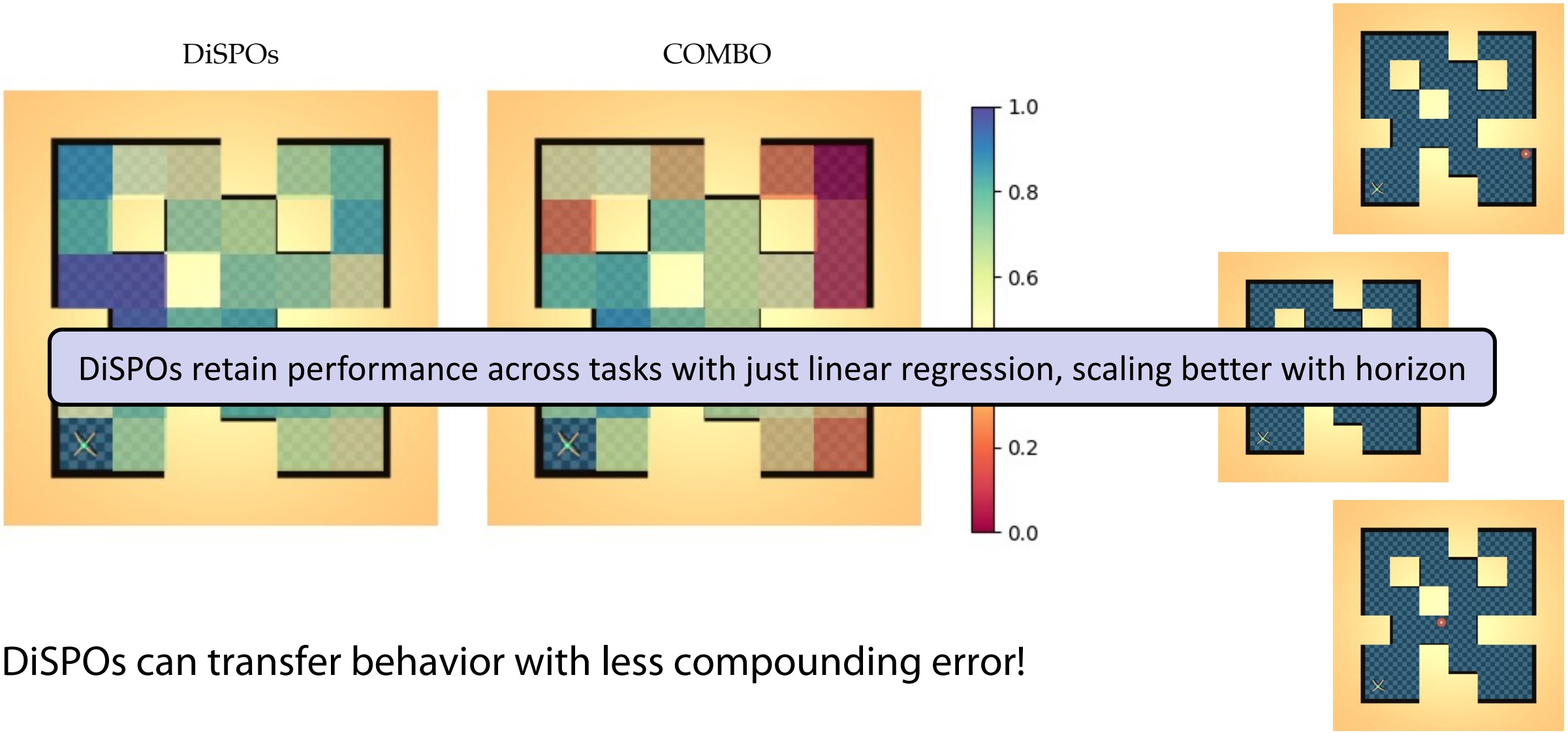





DiSPOs outperforms model-based RL with horizon and transfers better than model-free RL

umaze-diverse	**568** ± 12	447 ± 3	474 ± 2	460 ± 7	467 ± 5	547 ± 11	577 ± 7
medium-diverse	**631** ± 67	394 ± 52	294 ± 61	266 ± 2	236 ± 4	418 ± 16	403 ± 10
medium-play	**624** ± 58	370 ± 31	264 ± 29	271 ± 5	232 ± 4	397 ± 12	390 ± 33
large-diverse	**359** ± 59	215 ± 20	181 ± 46	132 ± 1	128 ± 1	244 ± 19	226 ± 9
large-play	**306** ± 18	250 ± 41	165 ± 12	134 ± 3	128 ± 2	248 ± 4	229 ± 5
kitchen-partial	**43** ± 6	0 ± 0	4 ± 4	0 ± 0	8 ± 7	11 ± 9	-
kitchen-mixed	**46** ± 5	10 ± 10	5 ± 5	0 ± 0	0 ± 0	0 ± 0	-
hopper-forward	566 ± 63	487 ±110	452 ± 59	470 ± 16	493 ±114	**982** ±157	-
hopper-backward	367 ± 15	261 ± 68	269 ± 77	220 ± 15	**596** ±211	194 ± 74	-
hopper-stand	**800** ± 0	685 ±130	670 ±120	255 ± 15	**800** ± 0	600 ±111	-
hopper-jump	**832** ± 22	746 ±112	726 ± 35	652 ± 28	753 ± 51	670 ±109	-

Model-based RL Model-free RL

Do DiSPOs transfer behavior?

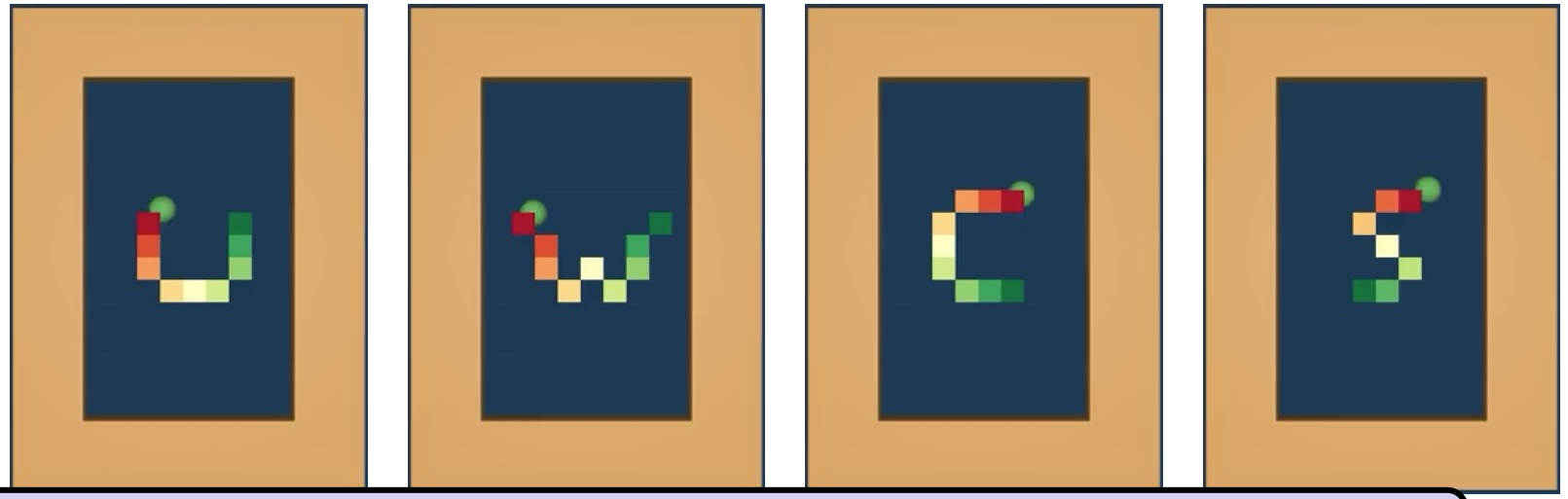


DiSPOs retain performance across tasks with just linear regression, scaling better with horizon

DiSPOs can transfer behavior with less compounding error!

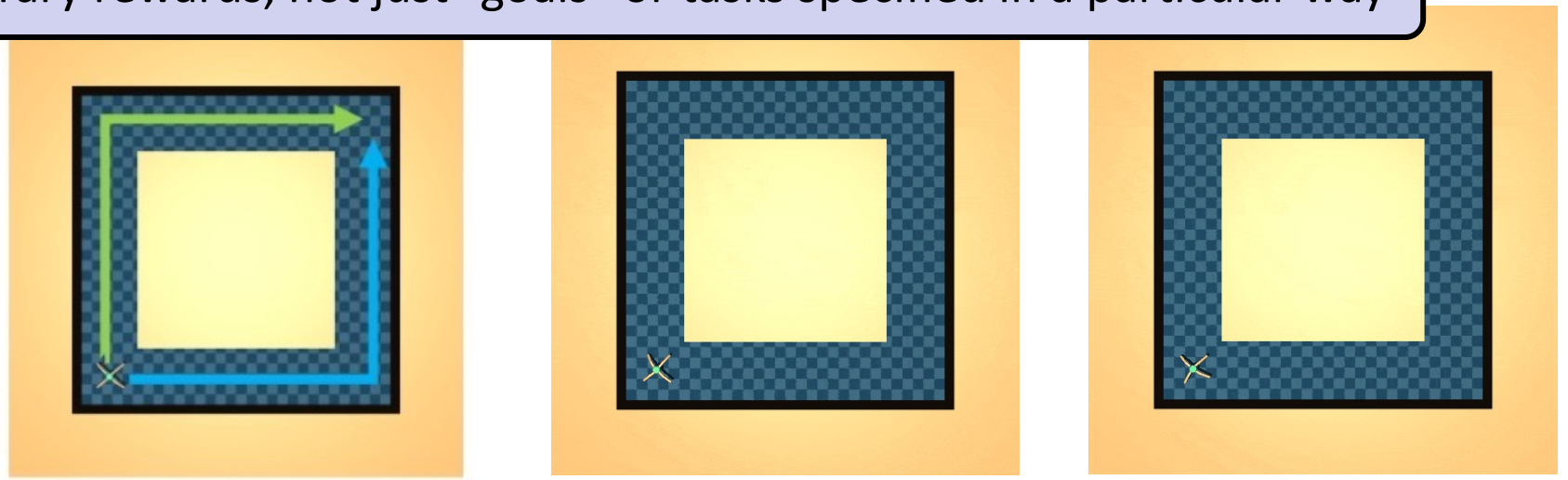
Can DiSPOs learn any task?

Drawing arbitrary shapes

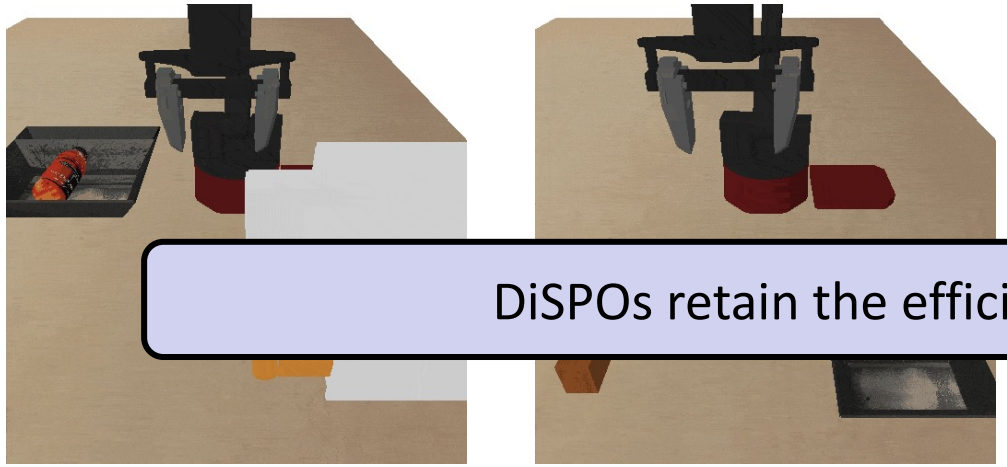


DiSPOs can optimize arbitrary rewards, not just “goals” or tasks specified in a particular way

Can represent non-goal, preference-based tasks



Can DiSPOs retain benefits of dynamic programming?



Monte-carlo methods

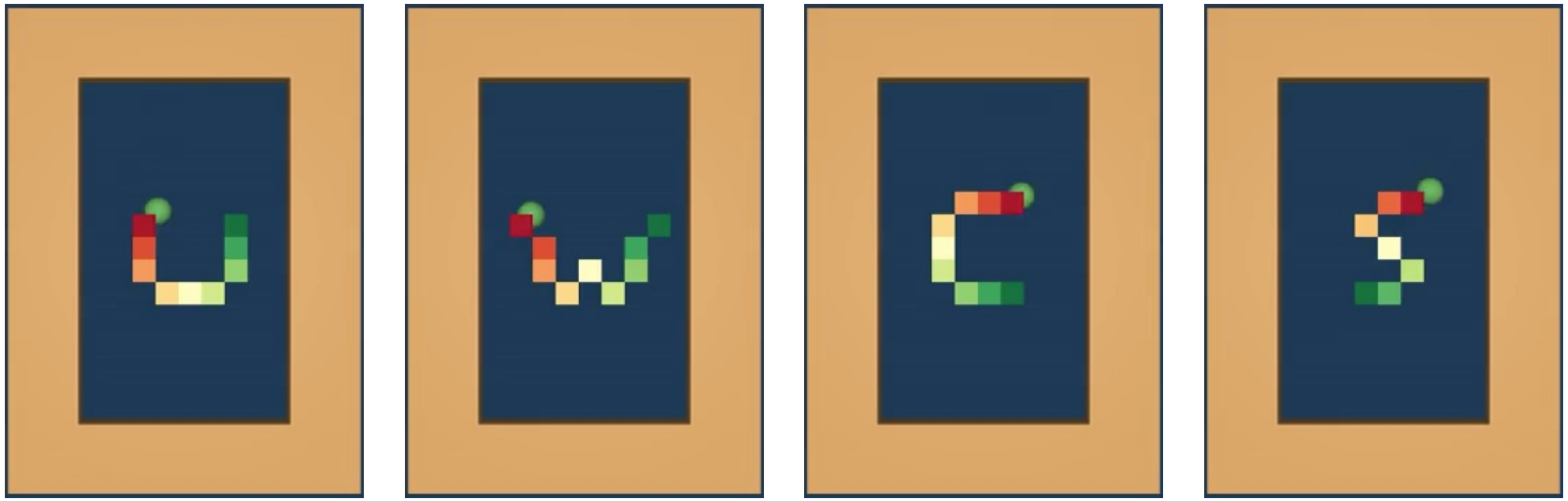
	DiSPO (Ours)	RaMP	DT
PickPlace	**49** ± 8	0 ± 0	0 ± 0
			0
			0

DiSPOs retain the efficiency benefits of dynamic programming

Distributional Bellman backups in DiSPOs retain stitching properties

What should you take away?

- Off-policy RL is model-based RL in disguise
- DiSPOs make off-policy learning reward and policy independent
  - Reward independence → modeling random features
  - Policy independence → modeling all paths

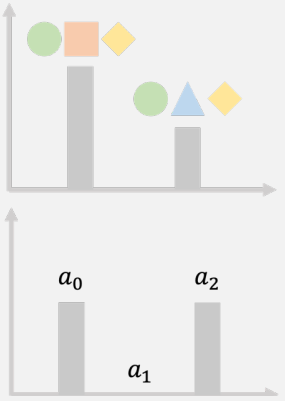


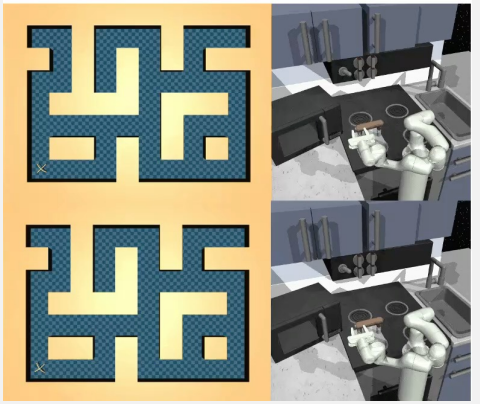
Change *how* to predict

Talk outline

Learning from Task-Relevant Objectives (Observation Size)

Learning Models of Cumulative Outcomes (Horizon)



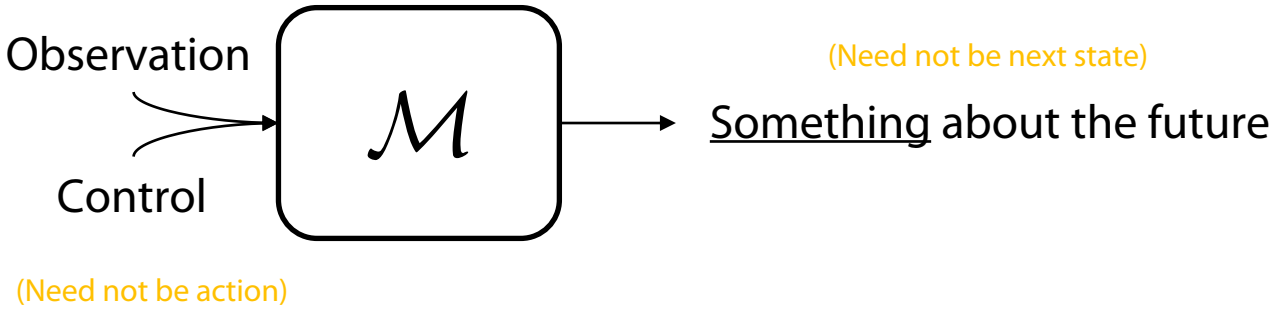


What does world modeling look like going forward?

1-step autoregressive modeling

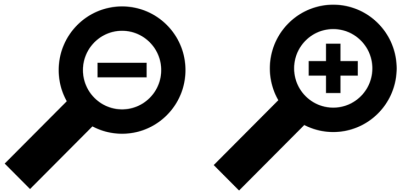
$$o_{t+1} \sim \hat{p}_\theta(\cdot|o_t, a_t) \qquad \boxed{\hat{p}_\theta(\cdot|o_t, a_t)} \qquad \max_\theta \mathbb{E}_{(o,a,o')\sim\mathcal{D}} \left[\log \hat{p}_\theta(o'|o, a)\right]$$

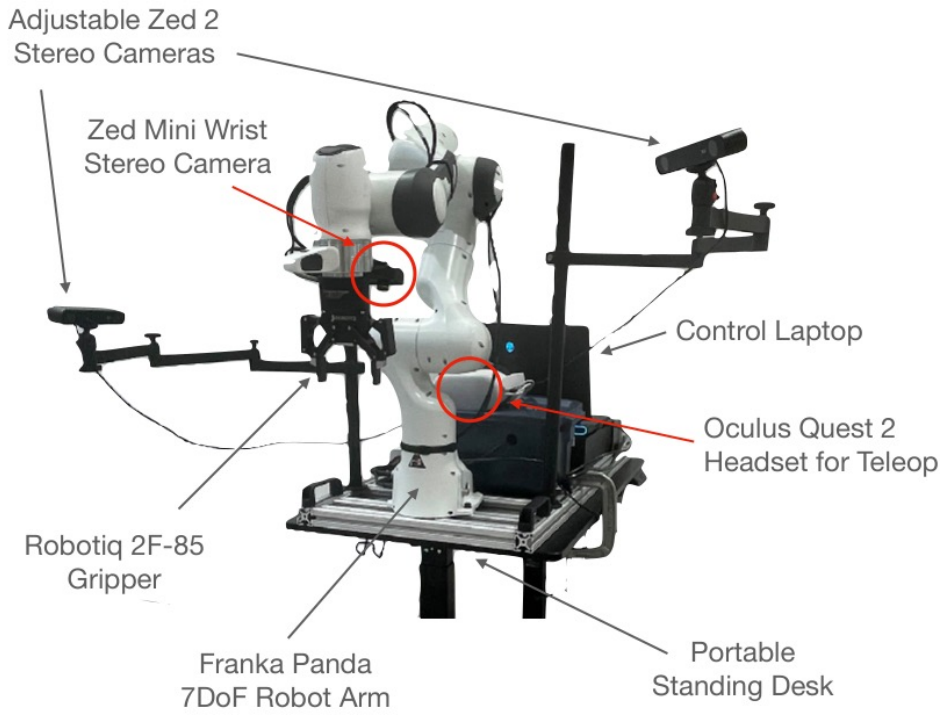
My take: models only need to predict what is needed for control, prediction, evaluation



Will scale better given reconstruction and autoregression is avoided

What is the big open question?





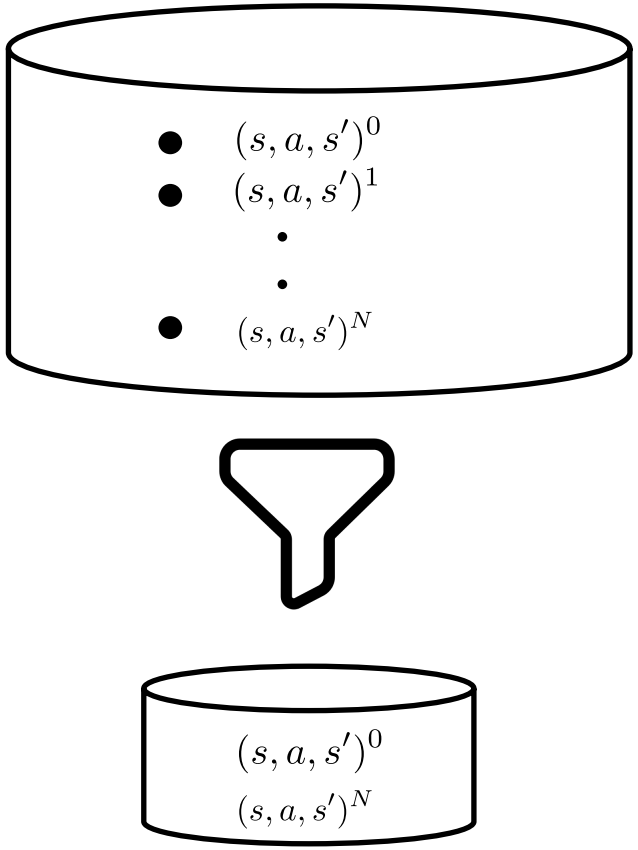
Expensive

Where is the data?

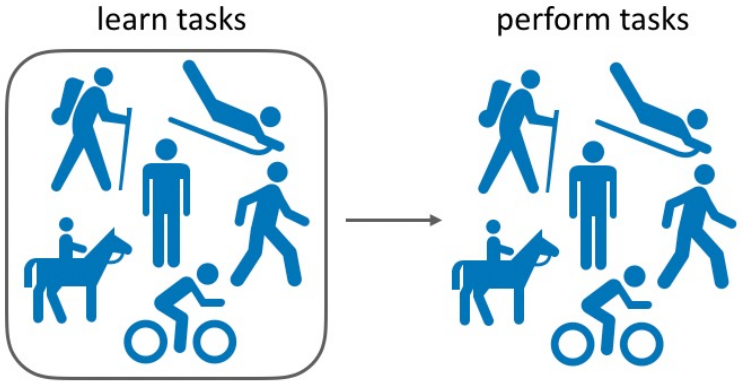
Some Exciting Directions for Future Work

Learning from large-scale action-free data

Data Curation and Selection

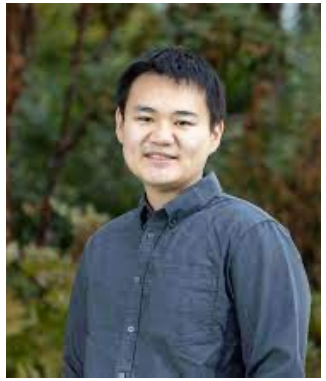
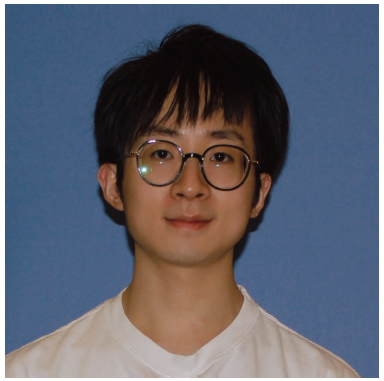


Quick Adaptation and Transfer



Thanks! Questions?




Papers covered:

1. **RePo: Resilient Model-Based Reinforcement Learning by Regularizing Posterior Predictability**, Chuning Zhu, Max Simchowitz, Siri Gadipudi, Abhishek Gupta, NeurIPS 2023 (Spotlight)
2. **Distributional Successor Features Enable Zero-Shot Policy Optimization,** Chuning Zhu, Xinqi Wang, Tyler Han, Simon Shaolei Du, Abhishek Gupta, NeurIPS 2024