

REINFORCEMENT LEARNING: THEORY AND PRACTICE

Exploration and Intrinsic Motivation II

Profs. Amy Zhang and Peter Stone



What's the problem?

this is easy (mostly)



this is impossible



Why?

Montezuma's revenge



- Getting key = reward
- Opening door = reward
- Getting killed by skull = nothing (is it good? bad?)
- Finishing the game only weakly correlates with rewarding events
- We know what to do because we **understand** what these sprites mean!

Three broad classes of exploration approaches:

1. Optimistic Exploration
2. Posterior Sampling
3. Information Gain

Go over the basic idea

How do we implement this for large environment/continuous state-action spaces/function approximation?

Optimistic exploration in RL

$$\text{UCB: } a = \arg \max \hat{\mu}_a + \sqrt{\frac{2 \ln T}{N(a)}}$$

“exploration bonus”

lots of functions work, so long as they decrease with $N(a)$

can we use this idea with MDPs?

count-based exploration: use $N(\mathbf{s}, \mathbf{a})$ or $N(\mathbf{s})$ to add *exploration bonus*

use $r^+(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \mathcal{B}(N(\mathbf{s}))$



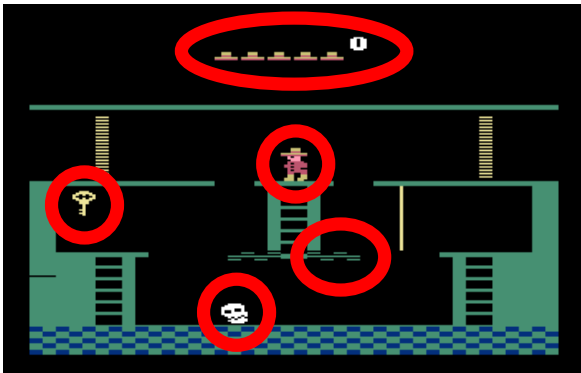
bonus that decreases with $N(\mathbf{s})$

use $r^+(\mathbf{s}, \mathbf{a})$ instead of $r(\mathbf{s}, \mathbf{a})$ with any model-free algorithm

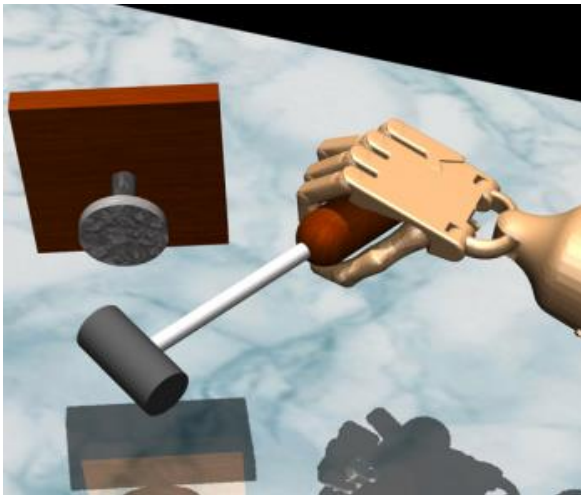
+ simple addition to any RL algorithm

- need to tune bonus weight

Exploring with pseudo-counts



fit model $p_{\theta}(\mathbf{s})$ to all states \mathcal{D} seen so far
 take a step i and observe \mathbf{s}_i
 fit new model $p_{\theta'}(\mathbf{s})$ to $\mathcal{D} \cup \mathbf{s}_i$
 use $p_{\theta}(\mathbf{s}_i)$ and $p_{\theta'}(\mathbf{s}_i)$ to estimate $\hat{N}(\mathbf{s})$
 set $r_i^+ = r_i + \mathcal{B}(\hat{N}(\mathbf{s}))$ ← “pseudo-count”



how to get $\hat{N}(\mathbf{s})$? use the equations

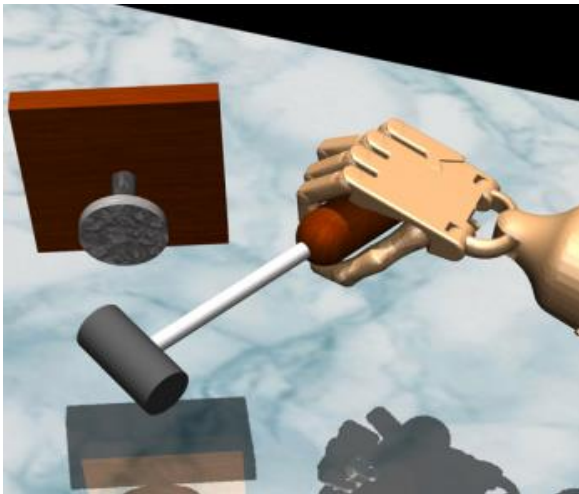
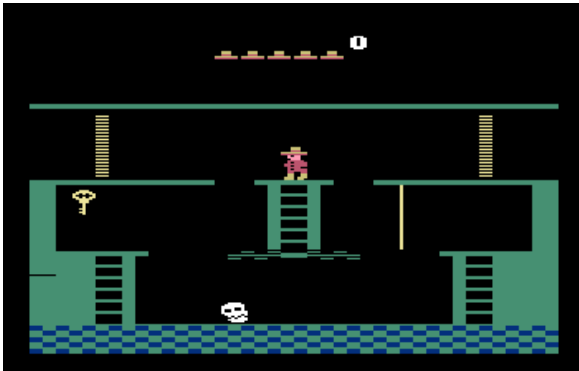
$$p_{\theta}(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i)}{\hat{n}} \qquad p_{\theta'}(\mathbf{s}_i) = \frac{\hat{N}(\mathbf{s}_i) + 1}{\hat{n} + 1}$$

two equations and two unknowns!

$$\hat{N}(\mathbf{s}_i) = \hat{n}p_{\theta}(\mathbf{s}_i) \qquad \hat{n} = \frac{1 - p_{\theta'}(\mathbf{s}_i)}{p_{\theta'}(\mathbf{s}_i) - p_{\theta}(\mathbf{s}_i)}p_{\theta}(\mathbf{s}_i)$$

Bellemare et al. “Unifying Count-Based Exploration...” Slide credit: Sergey Levine CS 285

What kind of model to use?

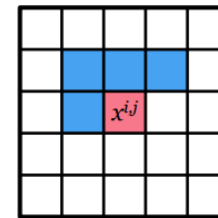


$$p_{\theta}(\mathbf{s})$$

need to be able to output densities, but doesn't necessarily need to produce great samples

opposite considerations from many popular generative models in the literature (e.g., GANs)

Bellemare et al.: "CTS" model:
condition each pixel on its top-left neighborhood



Other models: stochastic neural networks, compression length, EX2

Posterior sampling in deep RL

Thompson sampling:

$$\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$$


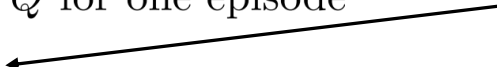
$$a = \arg \max_a E_{\theta_a} [r(a)]$$

What do we sample?

How do we represent the distribution?

bandit setting: $\hat{p}(\theta_1, \dots, \theta_n)$ is distribution over *rewards*

MDP analog is the Q -function!

- 
1. sample Q-function Q from $p(Q)$
 2. act according to Q for one episode
 3. update $p(Q)$
- since Q-learning is off-policy, we don't care which Q-function was used to collect data
- 

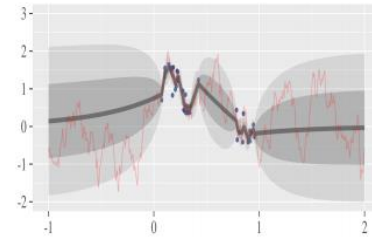
how can we represent a distribution over functions?

Bootstrap

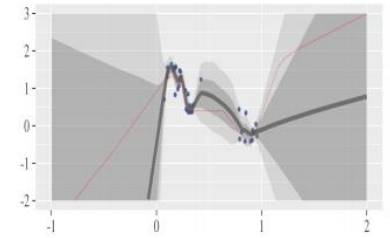
given a dataset \mathcal{D} , resample with replacement N times to get $\mathcal{D}_1, \dots, \mathcal{D}_N$

train each model f_{θ_i} on \mathcal{D}_i

to sample from $p(\theta)$, sample $i \in [1, \dots, N]$ and use f_{θ_i}

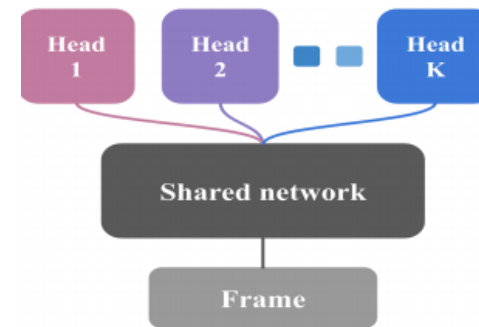


(b) Gaussian process posterior



(c) Bootstrapped neural nets

training N big neural nets is expensive, can we avoid it?



Information Gain in Deep RL

Reasoning about information gain (approximately)

Info gain: $IG(z, y|a)$

information gain about *what*?

Reasoning about information gain (approximately)

Info gain: $IG(z, y|a)$

information gain about *what*?

information gain about reward $r(\mathbf{s}, \mathbf{a})$?

state density $p(\mathbf{s})$?

information gain about dynamics $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$?

not very useful if reward is sparse

a bit strange, but somewhat makes sense!

good proxy for *learning* the MDP, though still heuristic

Generally intractable to use exactly, regardless of what is being estimated!

Reasoning about information gain (approximately)

Generally intractable to use exactly, regardless of what is being estimated

A few approximations:

prediction gain: $\log p_{\theta'}(\mathbf{s}) - \log p_{\theta}(\mathbf{s})$ (Schmidhuber '91, Bellemare '16)

intuition: if density changed a lot, the state was novel

Reasoning about information gain (approximately)

Generally intractable to use exactly, regardless of what is being estimated

A few approximations:

prediction gain: $\log p_{\theta'}(s) - \log p_{\theta}(s)$ (Schmidhuber '91, Bellemare '16)

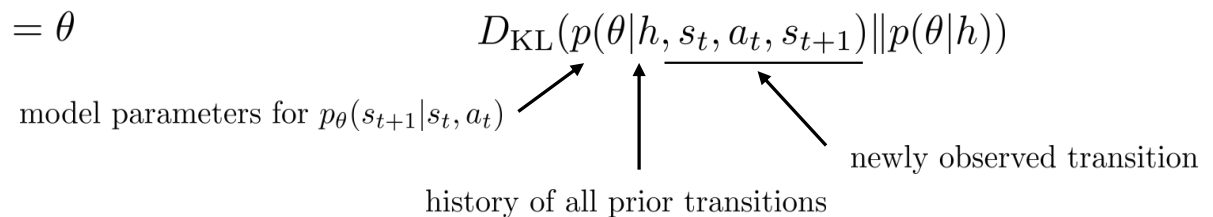
intuition: if density changed a lot, the state was novel

variational inference: (Houthoof et al. "VIME")

IG can be equivalently written as $D_{\text{KL}}(p(z|y)||p(z))$

learn about *transitions* $p_{\theta}(s_{t+1}|s_t, a_t): z = \theta$

$y = (s_t, a_t, s_{t+1})$



intuition: a transition is more informative if it causes belief over θ to change

idea: use variational inference to estimate $q(\theta|\phi) \approx p(\theta|h)$

given new transition (s, a, s') , update ϕ to get ϕ'

Reasoning about information gain (approximately)

VIME implementation:

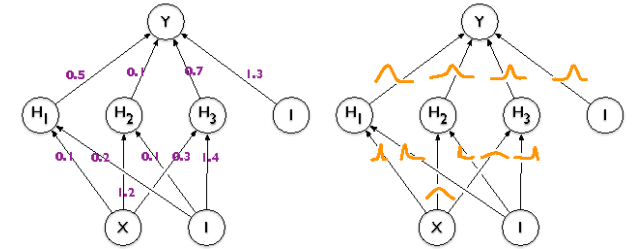
IG can be equivalently written as $D_{\text{KL}}(p(\theta|h, s_t, a_t, s_{t+1}) || p(\theta|h))$

model parameters for $p_{\theta}(s_{t+1}|s_t, a_t)$



history of all prior transitions

newly observed transition



$q(\theta|\phi) \approx p(\theta|h)$

specifically, optimize variational lower bound $D_{\text{KL}}(q(\theta|\phi) || p(h|\theta)p(\theta))$

represent $q(\theta|\phi)$ as product of independent Gaussian parameter distributions

with mean ϕ (see Blundell et al. “Weight uncertainty in neural networks”)

given new transition (s, a, s') , update ϕ to get ϕ'

i.e., update the network weight means and variances

use $D_{\text{KL}}(q(\theta|\phi') || q(\theta|\phi))$ as approximate bonus

$$p(\theta|\mathcal{D}) = \prod_i p(\theta_i|\mathcal{D})$$

$$p(\theta_i|\mathcal{D}) = \mathcal{N}(\mu_i, \sigma_i)$$



Houthoof et al. “VIME”

Slide credit: Sergey Levine CS 285

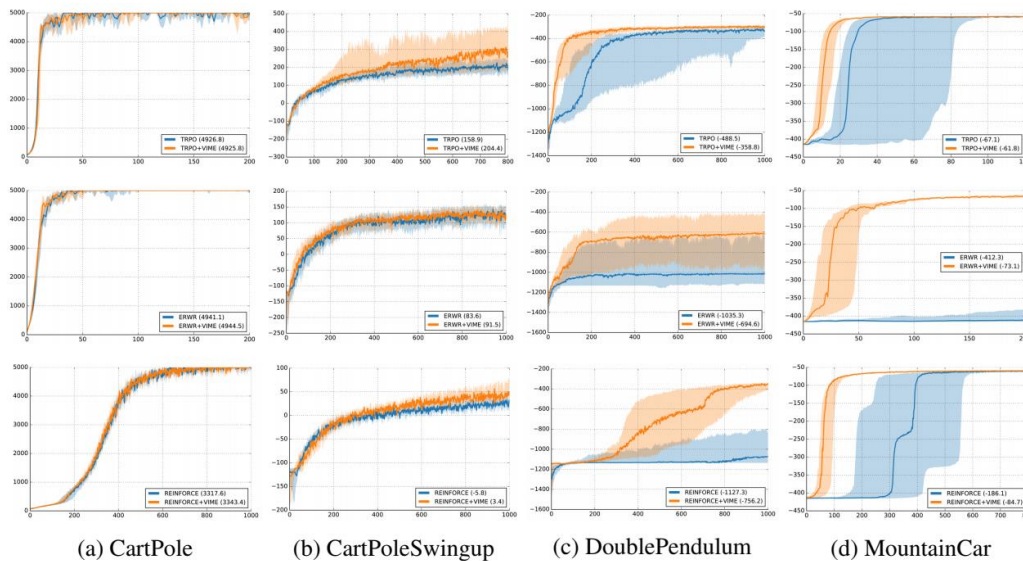
Reasoning about information gain (approximately)

VIME implementation:

IG can be equivalently written as $D_{\text{KL}}(p(\theta|h, s_t, a_t, s_{t+1})||p(\theta|h))$

$q(\theta|\phi) \approx p(\theta|h)$ specifically, optimize variational lower bound $D_{\text{KL}}(q(\theta|\phi)||p(h|\theta)p(\theta))$

use $D_{\text{KL}}(q(\theta|\phi')||q(\theta|\phi))$ as approximate bonus



Approximate IG:

- + appealing mathematical formalism
- models are more complex, generally harder to use effectively

Houthoof et al. "VIME"

Slide credit: Sergey Levine CS 285

Exploration with model errors

$D_{\text{KL}}(q(\theta|\phi')||q(\theta|\phi))$ can be seen as change in network (mean) parameters ϕ
if we forget about IG, there are many other ways to measure this

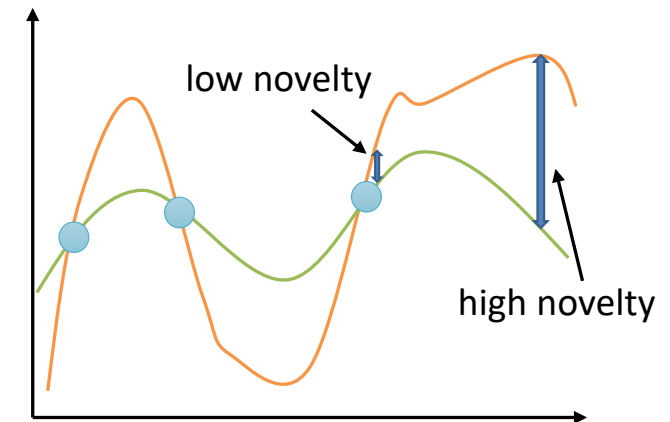
Stadie et al. 2015:

- encode image observations using auto-encoder
- build predictive model on auto-encoder latent states
- use model error as exploration bonus

Schmidhuber et al. (see, e.g. “Formal Theory of Creativity, Fun, and Intrinsic Motivation):

- exploration bonus for model error
- exploration bonus for model gradient
- many other variations

Many others!



General themes

UCB:

$$a = \arg \max \hat{\mu}_a + \sqrt{\frac{2 \ln T}{N(a)}}$$

Thompson sampling:

$$\theta_1, \dots, \theta_n \sim \hat{p}(\theta_1, \dots, \theta_n)$$
$$a = \arg \max_a E_{\theta_a}[r(a)]$$

Info gain:

$$\text{IG}(z, y|a)$$

- Most exploration strategies require some kind of uncertainty estimation (even if it's naïve)
- Usually assumes some value to new information
 - Assume unknown = good (optimism)
 - Assume sample = truth
 - Assume information gain = good

Discussion Exercise

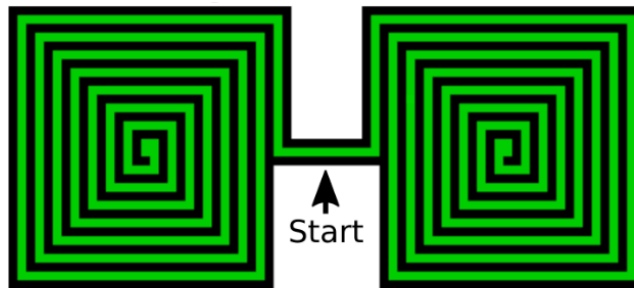
Go back to the robot navigating to the tower example.

Which of the three types of exploration would you pick?

How would you implement it?

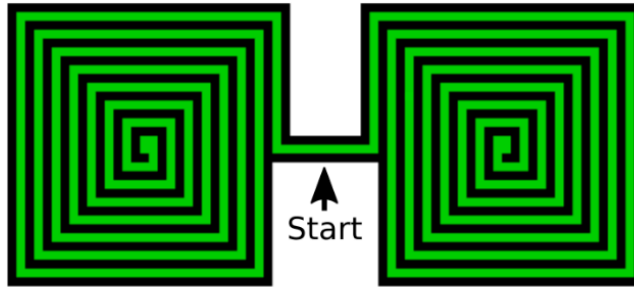
What are potential upsides and downsides compared to the other methods?

What's a possible failure mode of intrinsic motivation?

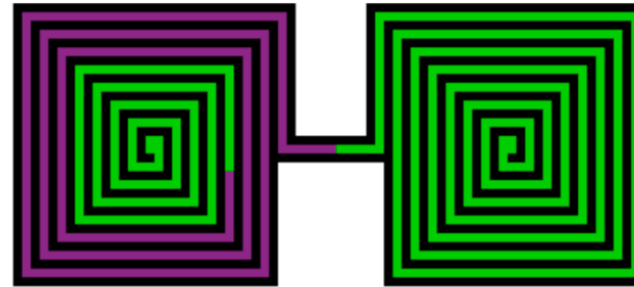


Go-Explore (Ecoffet et al. 2019)

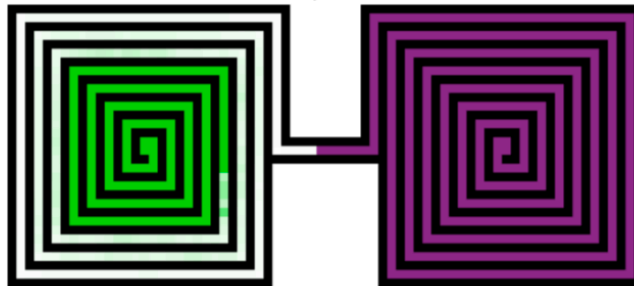
1. Intrinsic reward (green) is distributed throughout the environment



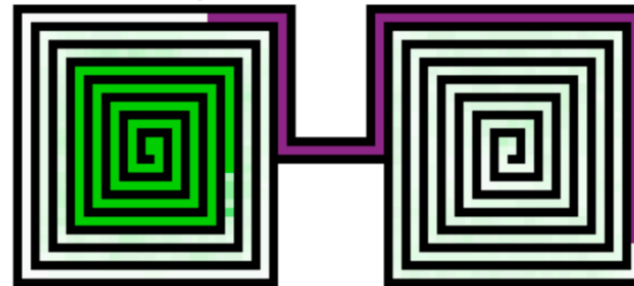
2. An IM algorithm might start by exploring (purple) a nearby area with intrinsic reward



3. By chance, it may explore another equally profitable area



4. Exploration fails to rediscover promising areas it has detached from



Go-Explore (Ecoffet et al. 2019)

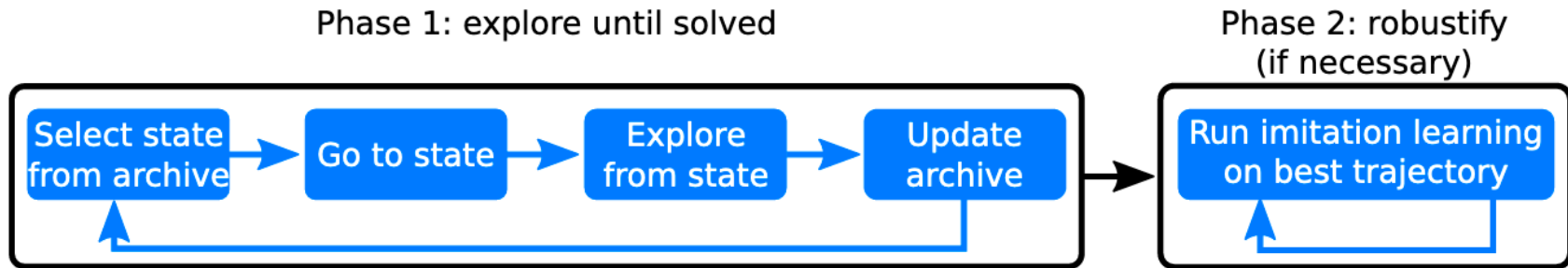


Figure 2: A **high-level overview of the Go-Explore algorithm.**

Cell representations

Go-Explore can be run directly in high-dimensional state space - but intractable in practice.

How to construct a good cell representation (state abstraction)?

Cell representations

Go-Explore can be run directly in high-dimensional state space - but intractable in practice.

How to construct a good cell representation (state abstraction)?

Turns out simple dimensionality reduction (downsample game frame) works!

Cell representations

Go-Explore can be run directly in high-dimensional state space - but intractable in practice.

How to construct a good cell representation (state abstraction)?

Turns out simple dimensionality reduction (downsample game frame) works!

How would you implement a state abstraction with domain knowledge?

Cell representations

Go-Explore can be run directly in high-dimensional state space - but intractable in practice.

How to construct a good cell representation (state abstraction)?

Turns out simple dimensionality reduction (downsample game frame) works!

How would you implement a state abstraction with domain knowledge?

X, y position of agent, room number, level number, which rooms currently-held keys were found, etc.

Go-Explore results on Montezuma's revenge

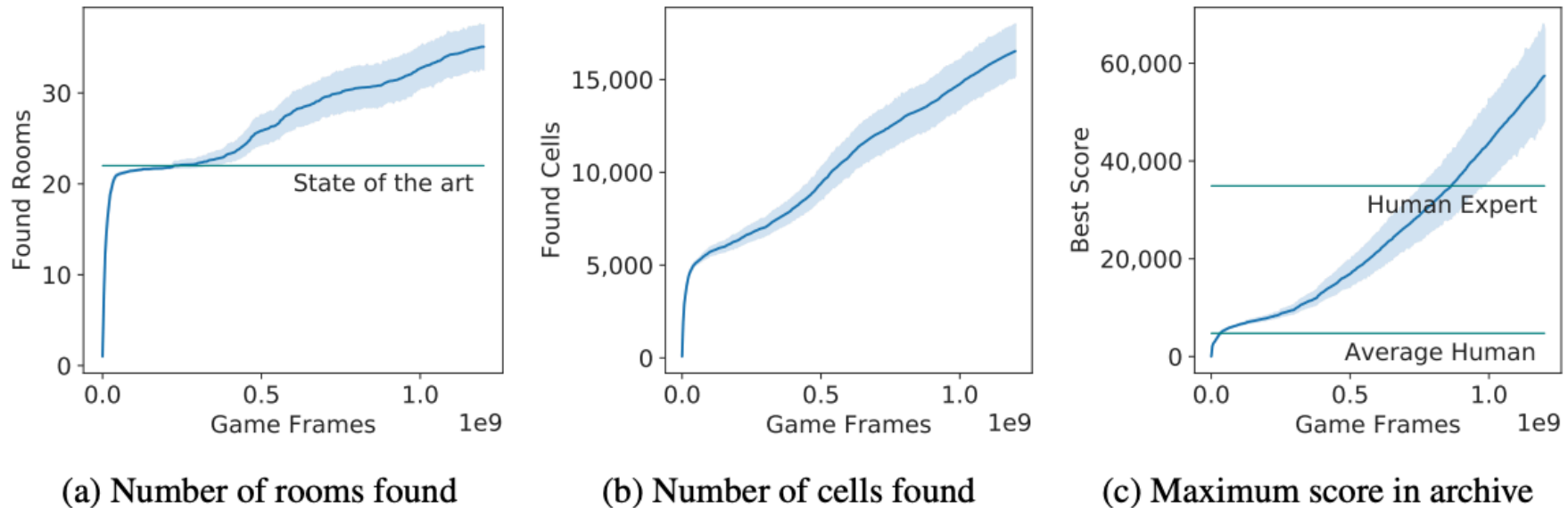


Figure 4: Performance of the exploration phase of Go-Explore with downscaled frames on Montezuma's Revenge. Lines indicating human and the algorithmic state of the art are for comparison, but recall that the Go-Explore scores in this plot are on a deterministic version of the game (unlike the post-Phase 2 scores presented in this section).

Go-Explore results on Montezuma's revenge

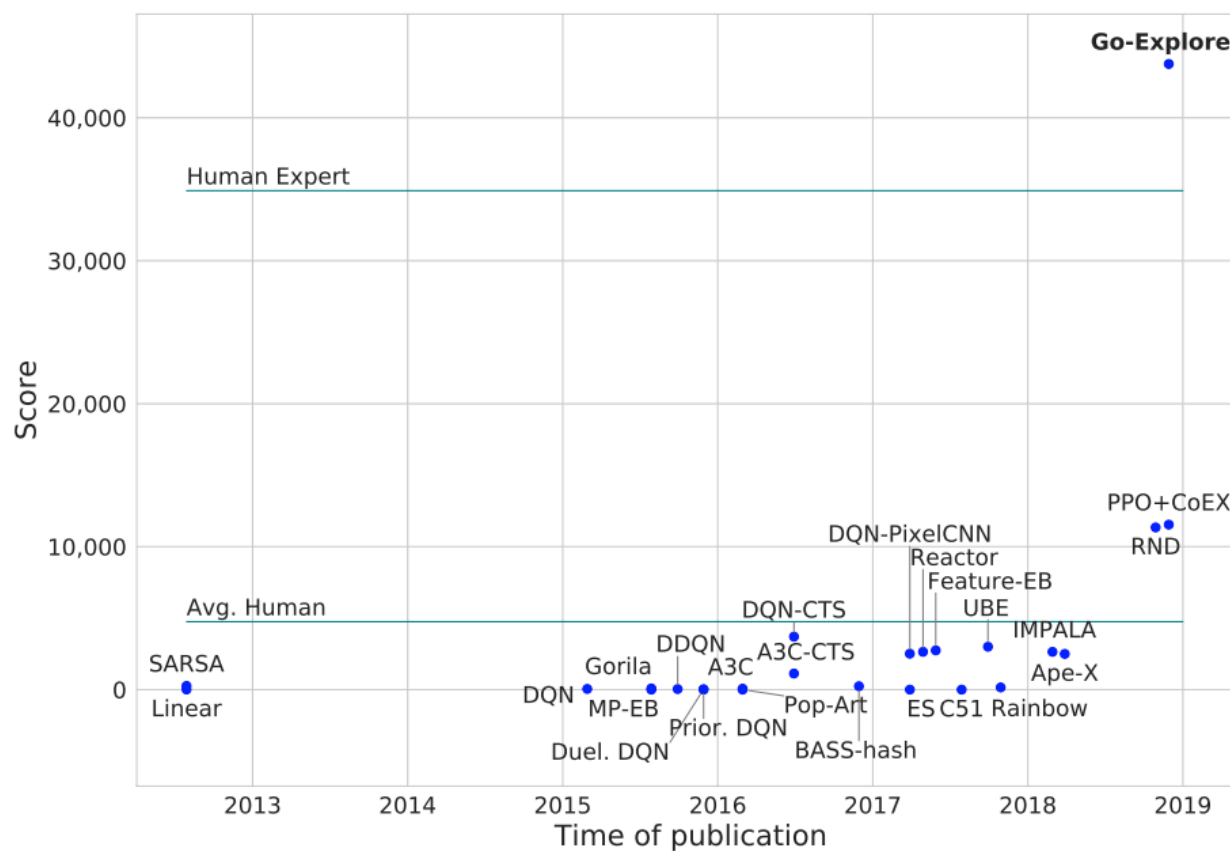


Figure 6: **History of progress on Montezuma's Revenge vs. the version of Go-Explore that does not harness domain knowledge.** Go-Explore significantly improves on the prior state of the art. These data are presented in tabular form in Appendix A.9.

Go-Explore results on Montezuma's revenge with domain knowledge in the cell representation

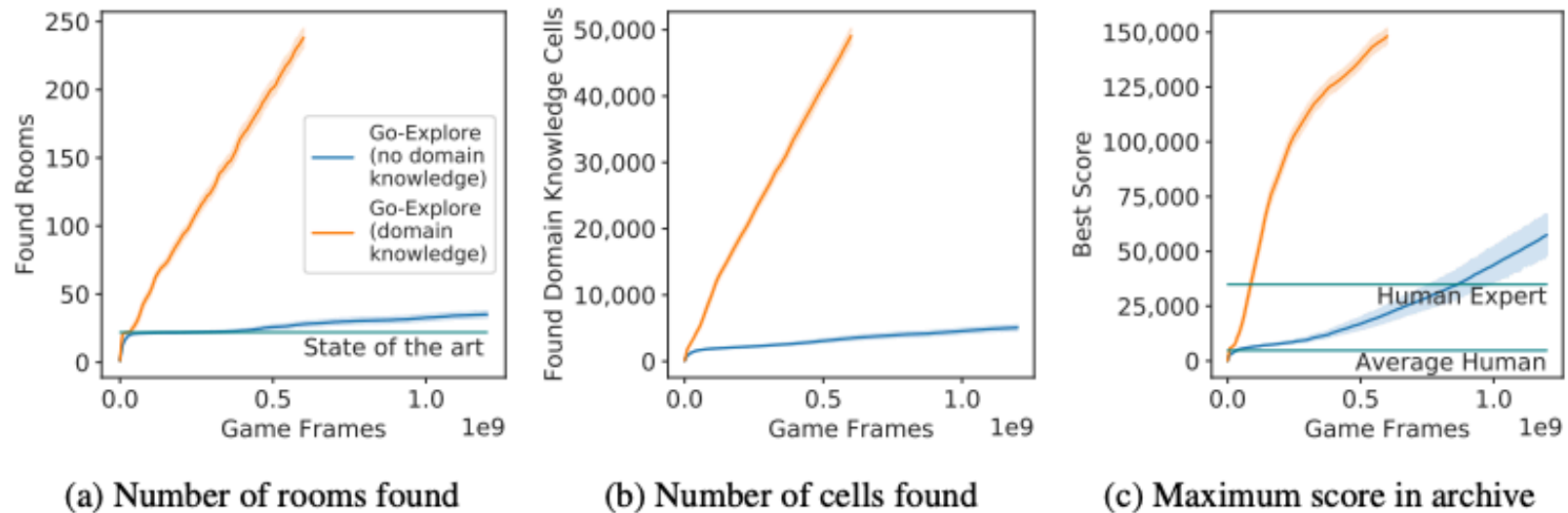


Figure 7: Performance on Montezuma's Revenge of Phase 1 of Go-Explore with and without domain knowledge. The algorithm finds more rooms, cells, and higher scores with the easily provided domain knowledge, and does so with a better sample complexity. For (b), we plot the number of cells found in the no-domain-knowledge runs according to the more intelligent cell representation from the domain-knowledge run to allow for an equal comparison.

Go-Explore results on Montezuma's revenge with domain knowledge in the cell representation

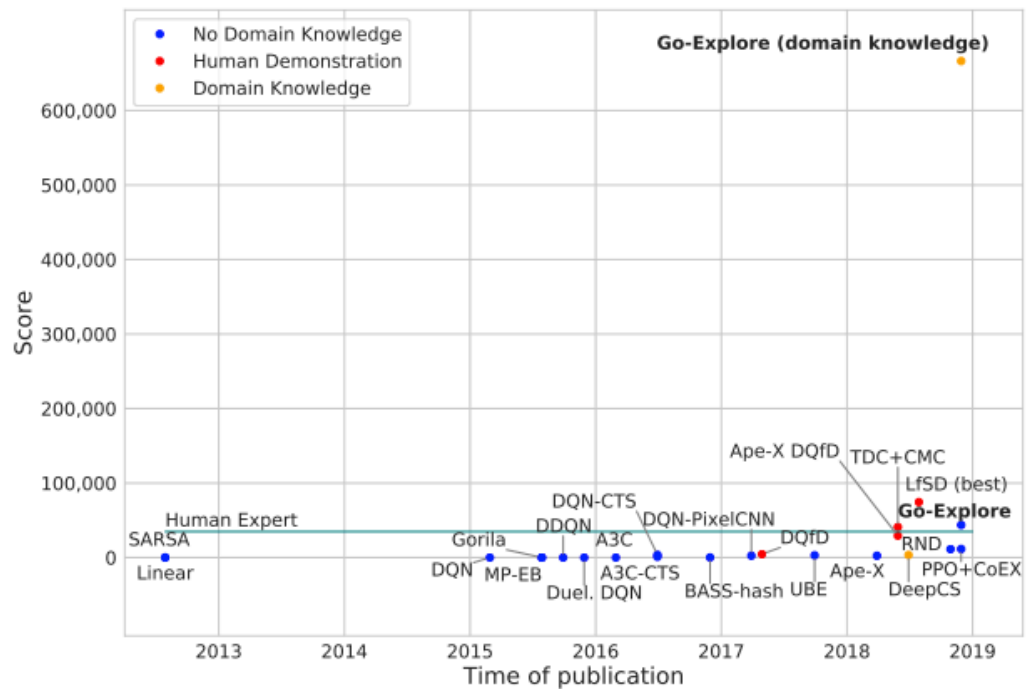


Figure 8: **Historical progress on Montezuma's Revenge vs. the version of Go-Explore that harnesses domain knowledge.** With domain knowledge, Go-Explore dramatically outperforms prior work, the no-domain-knowledge version of Go-Explore, and even prior work with imitation learning that was provided the solution in the form of human demonstrations. The data are presented in tabular form in Appendix A.9.

Discussion Exercise

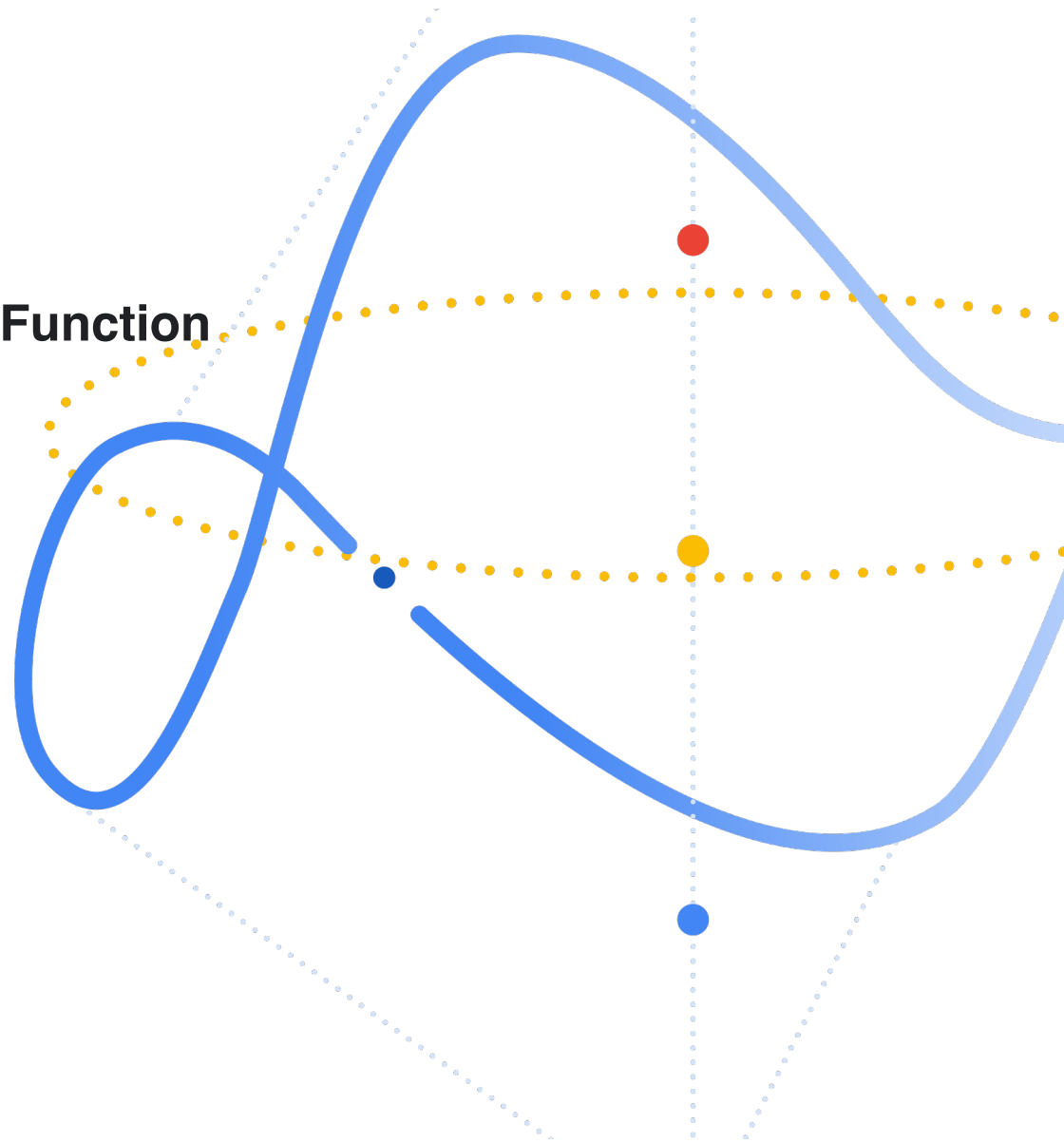
How can state abstractions or temporal abstractions be combined with exploration?

Go back to the robot navigating to the tower example. What type of state abstraction would make exploration more efficient?

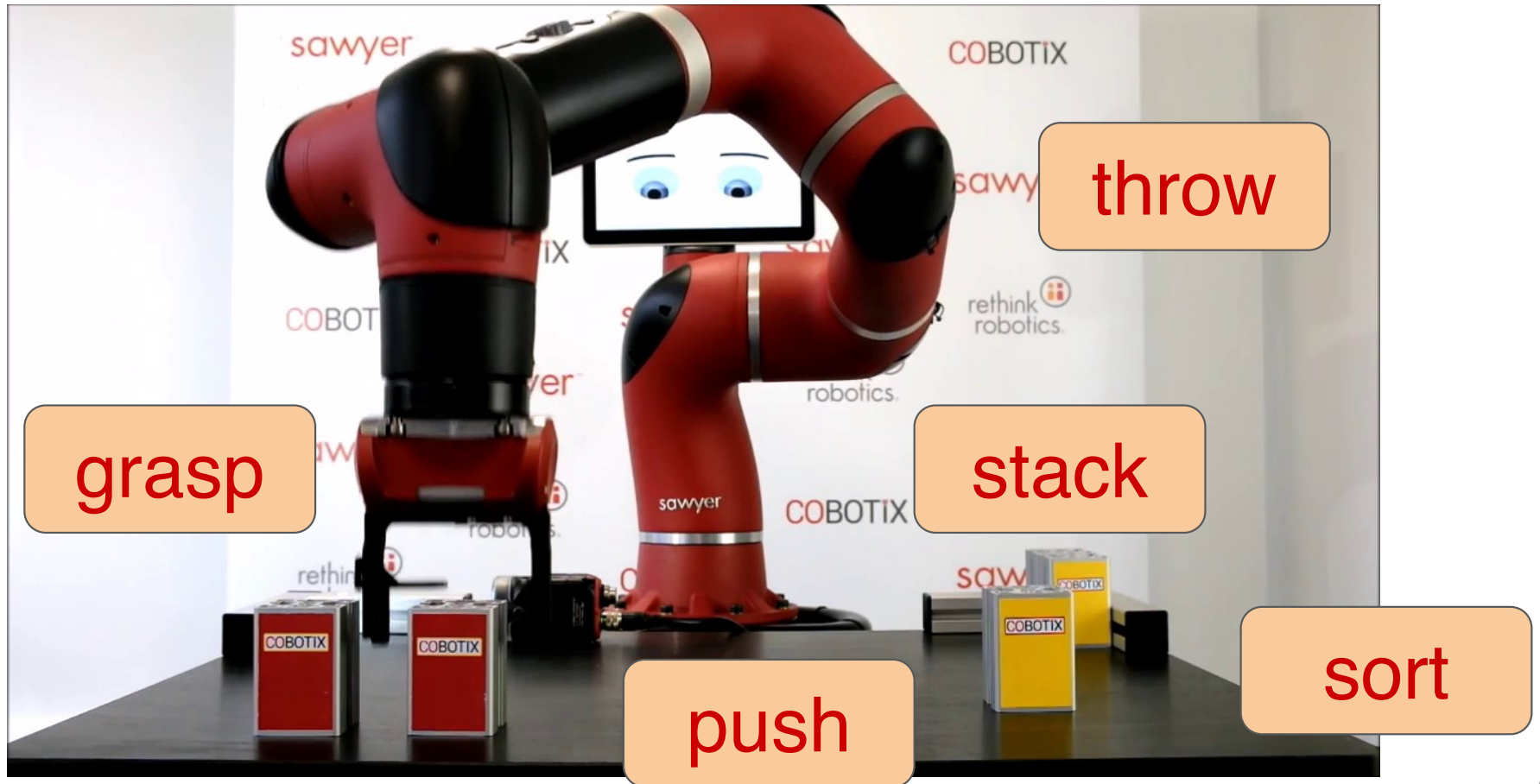
What type of temporal abstraction would make exploration more efficient?

Diversity is All You Need: Learning Skills without a Reward Function

Ben Eysenbach
PhD student at CMU



What is a Skill?



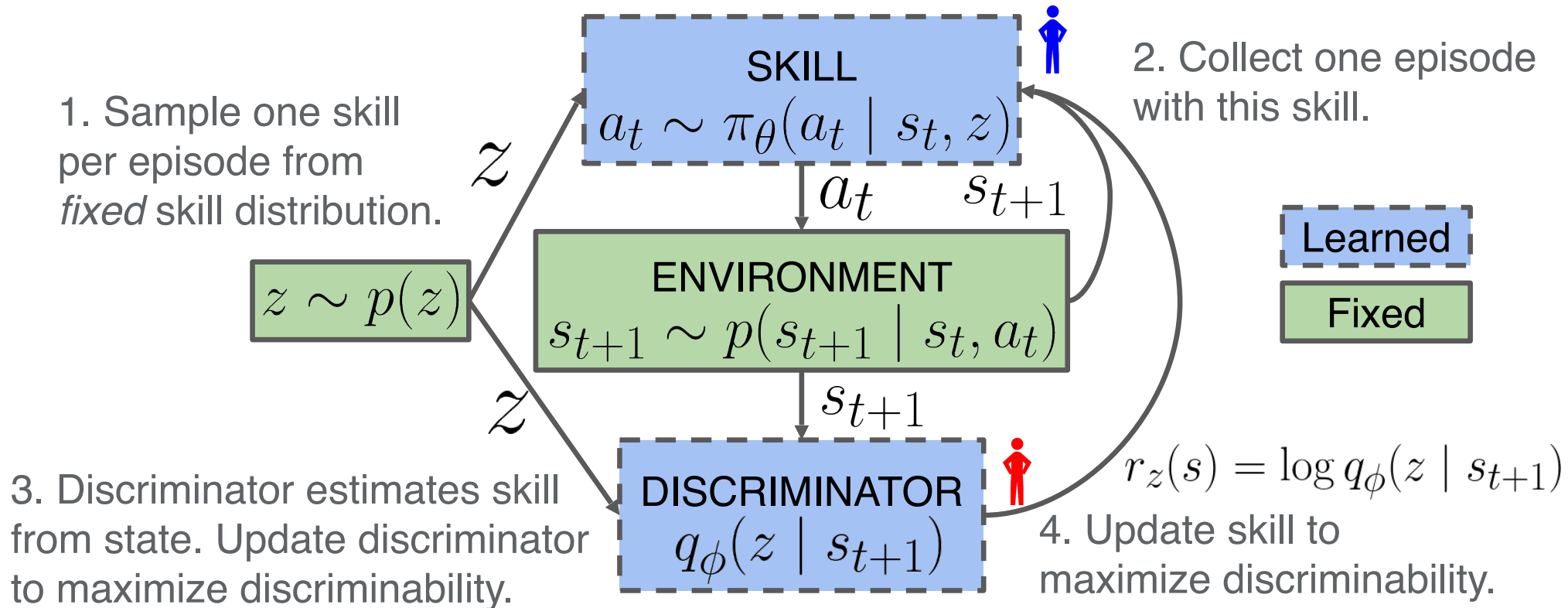
What is a Skill?

Properties of good skills:

- *Exploration* - at most one skill “dithers”; forces skills to explore large regions of the state space.
- *Predictability* - want to predict what a skill will do (important for hierarchical RL).
- *Interpretability* - easy to infer which skill is being executed at any given point in time.

Idea: Learn a **set** of skills that is **as diverse as possible**.

DIAYN: How does the algorithm work?



How to Learn Without a Reward Function?

$$\mathcal{F}(\theta) \triangleq \underbrace{MI(s, z)}_{\substack{\text{Skills control states visited:} \\ \bullet \text{ Infer skill from state.} \\ \bullet \text{ Predict where skill goes.}}} + \underbrace{\mathcal{H}[a | s]}_{\substack{\text{Act as randomly} \\ \text{as possible.}}} - \underbrace{MI(a, z | s)}_{\substack{\text{Ignore actions that} \\ \text{don't affect the state.}}}$$

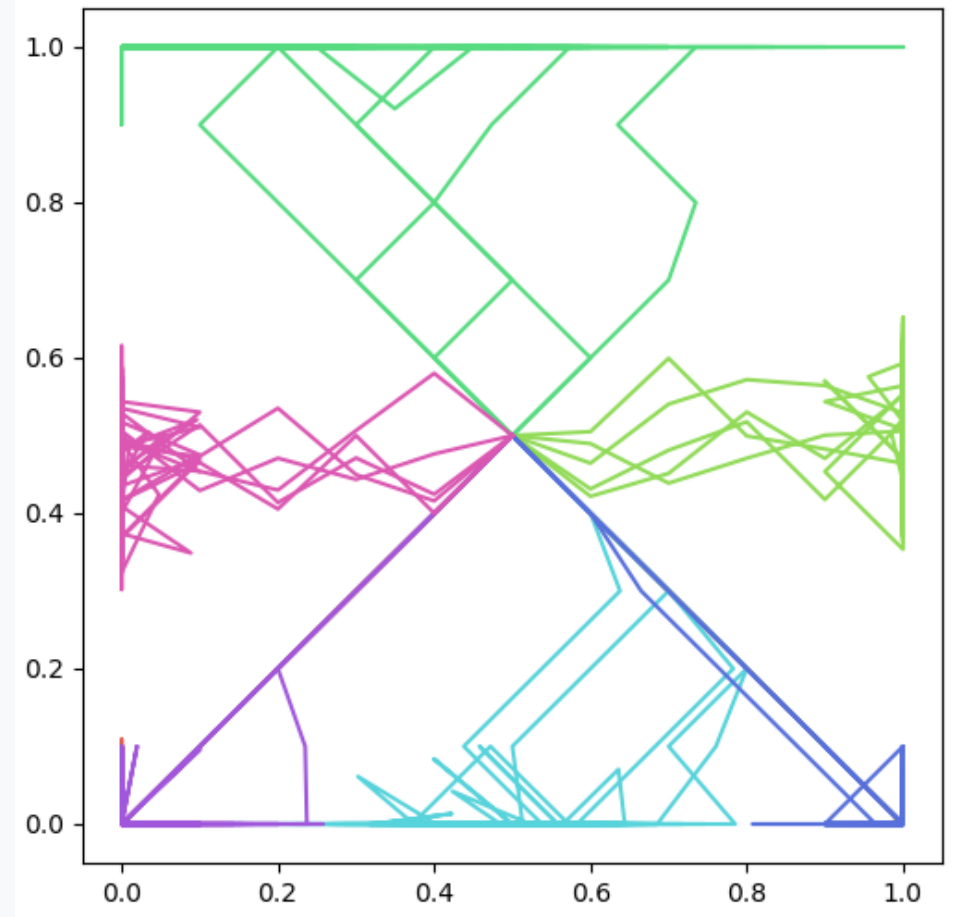


$$r_z(s, a) \triangleq \log q_\phi(z | s) - \log p(z)$$

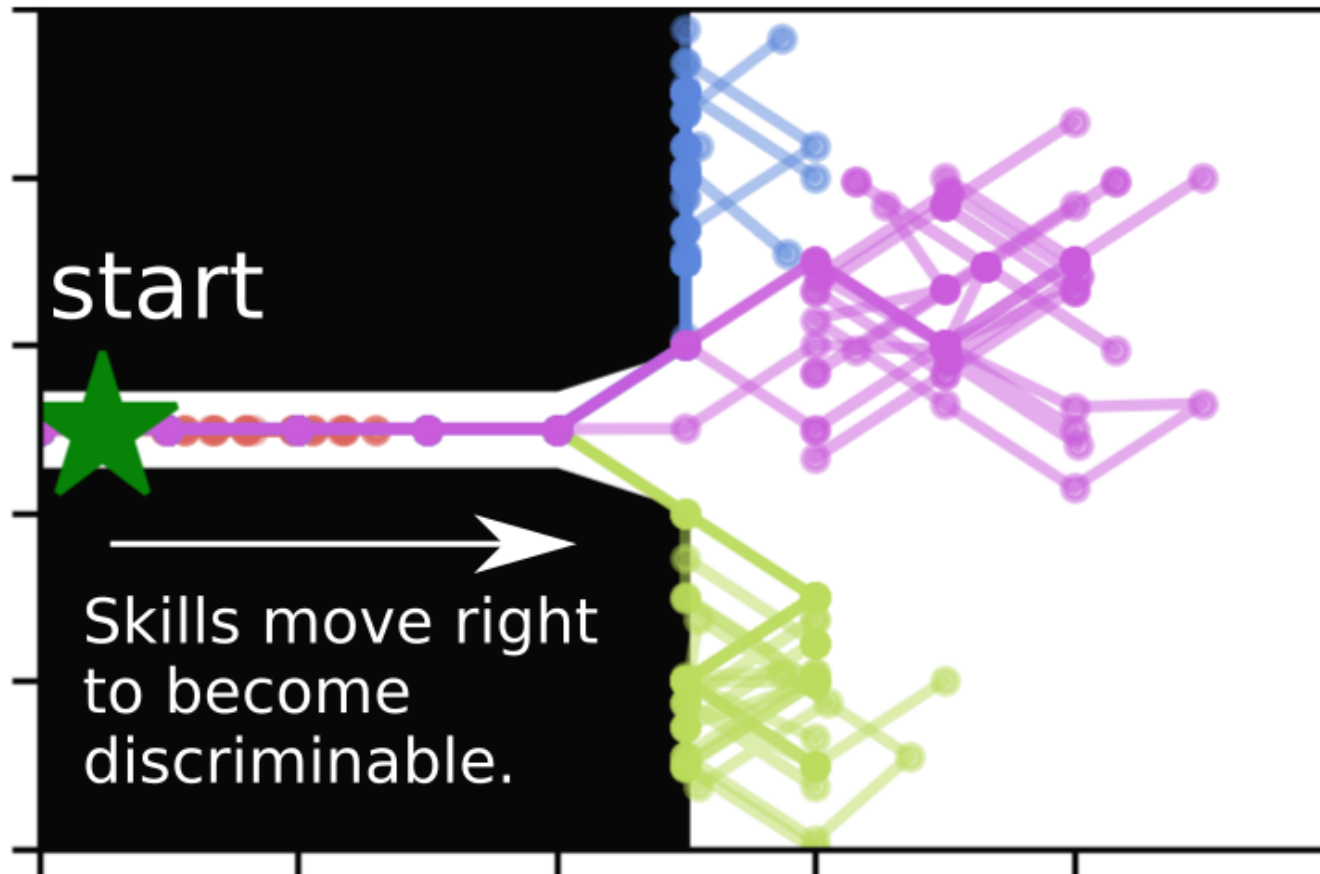
DIAYN is not the first to suggest objectives with this flavor, and there is much concurrent work [Gregor 16, Florensa 17, Co-Reyes 18, Achiam 18]. What differentiates our work is (1) that it's able to scale to complex environments, (2) the application to imitation learning and HRL.

Visualizing DIAYN

- *Exploration*
- *Predictability*
- *Interpretability*

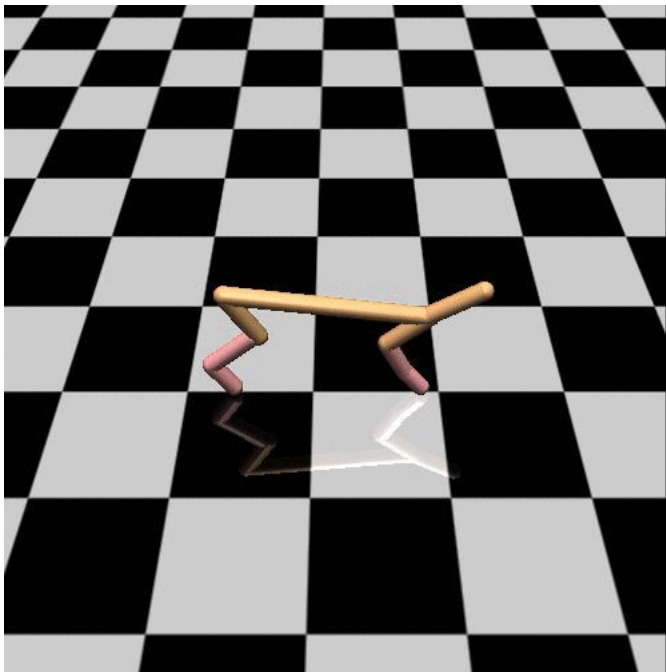


DIAYN maximizes *future* diversity.

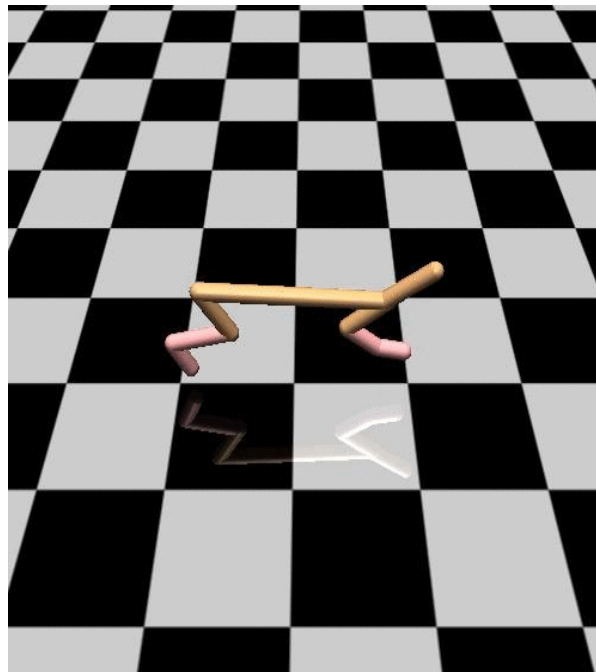


What Skills are Learned?

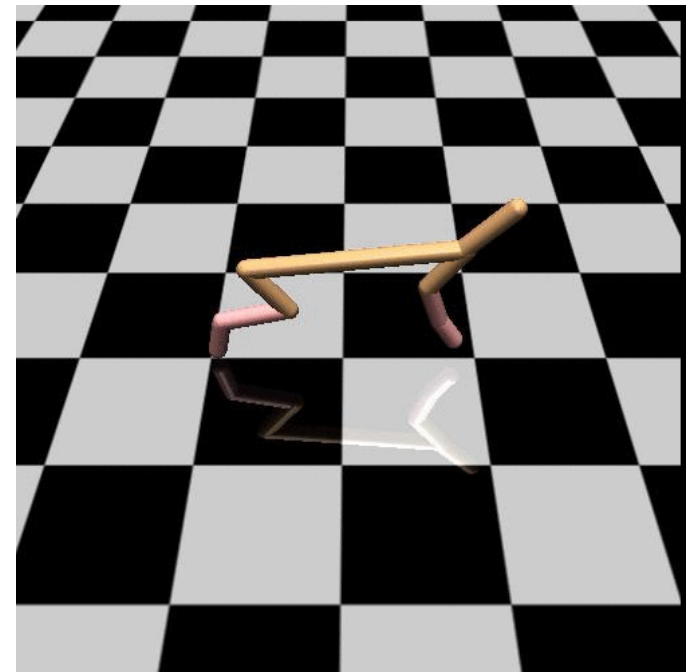
Walking forwards



Running Backwards

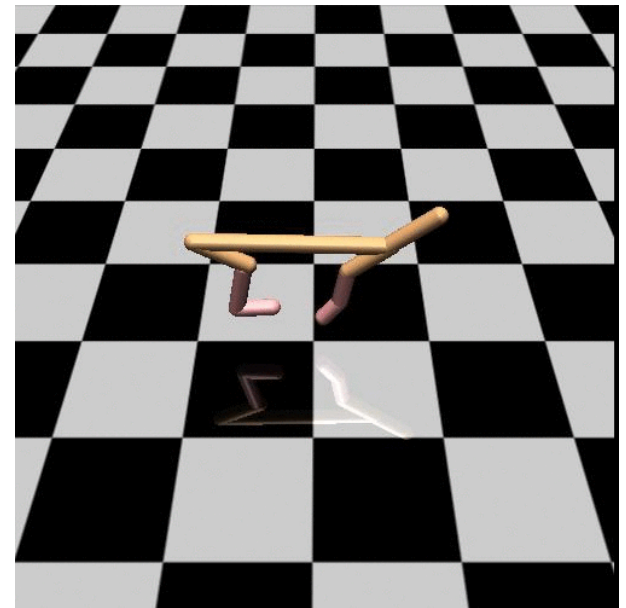
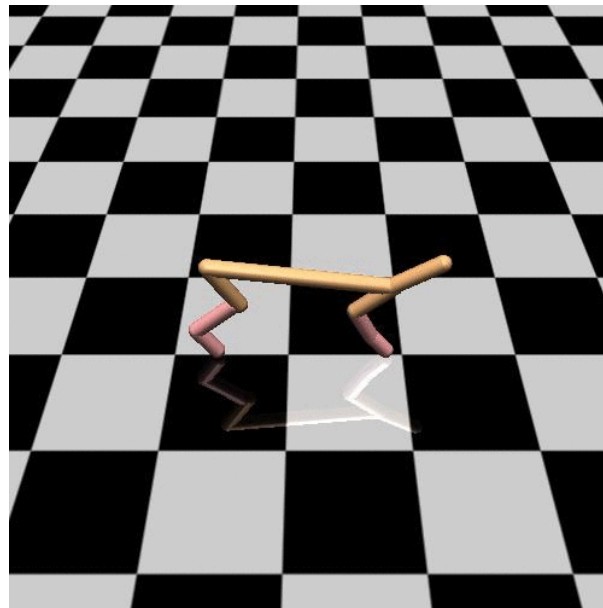
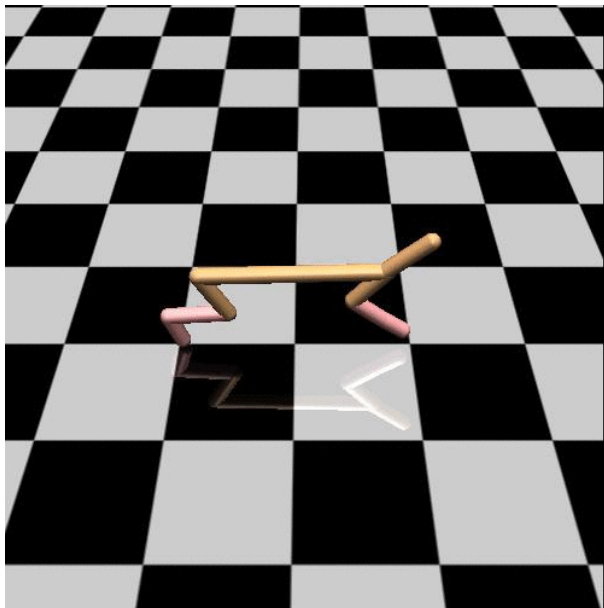


Front flips



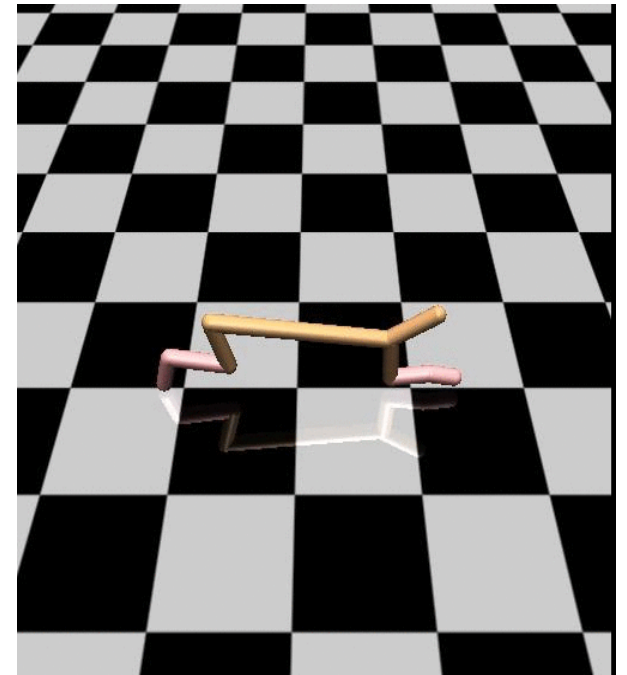
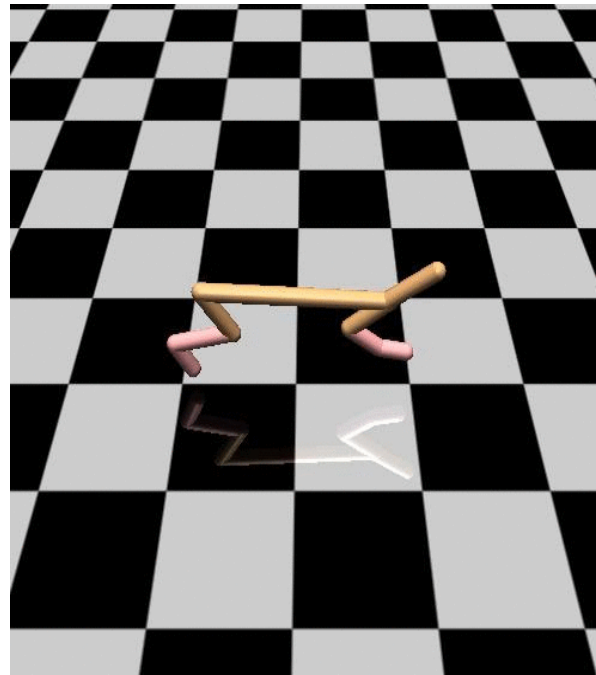
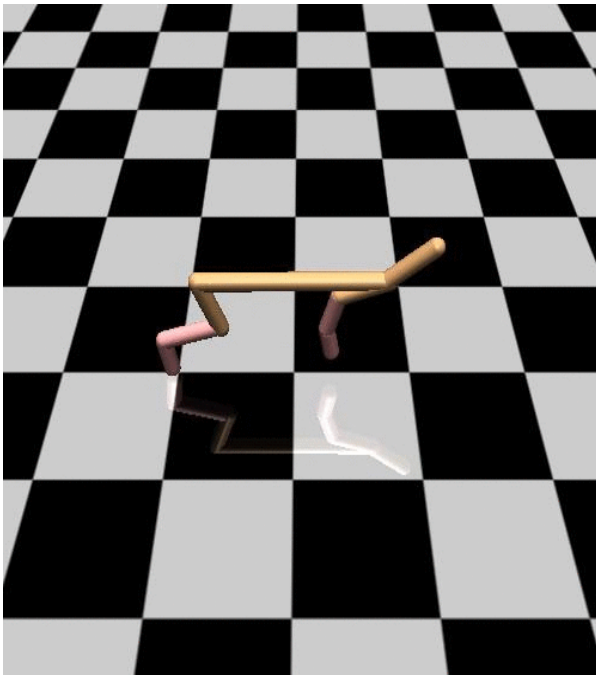
What Skills are Learned?

Skills for different forward gaits



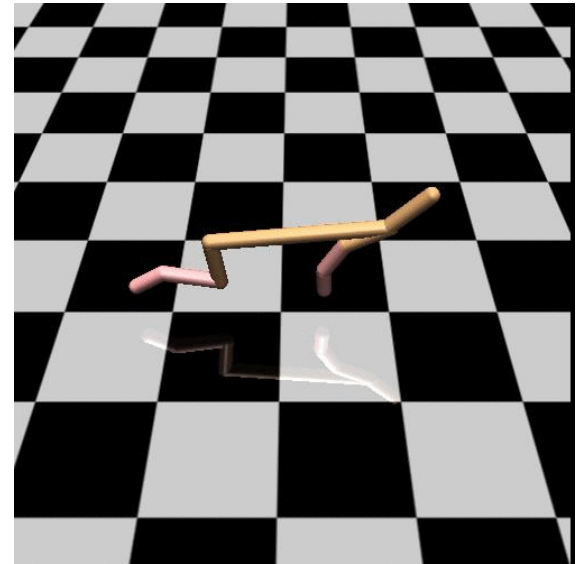
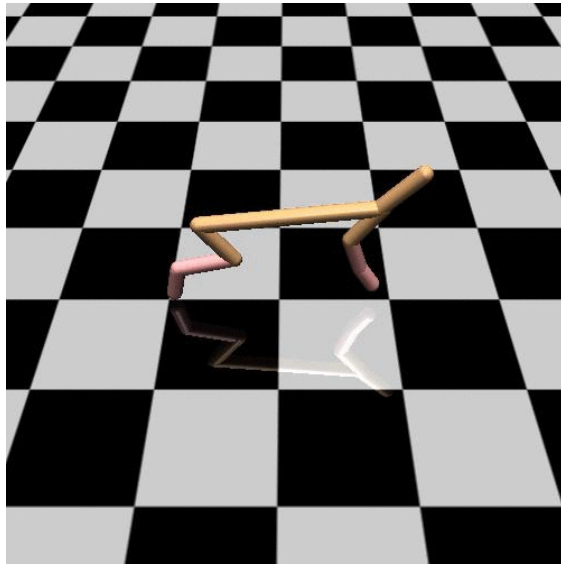
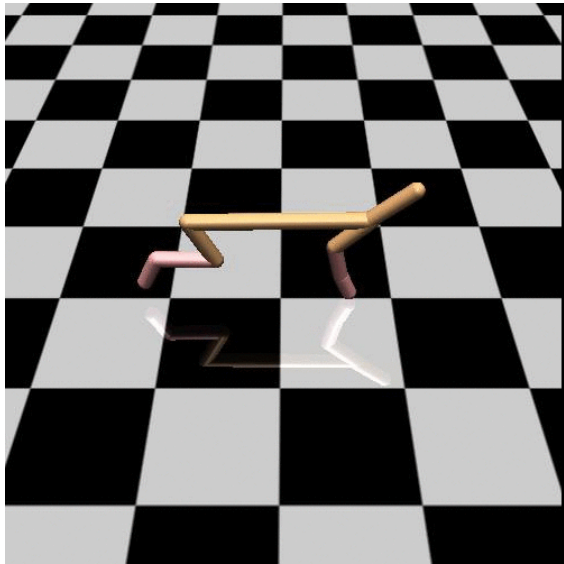
What Skills are Learned?

Skills for different backward gaits

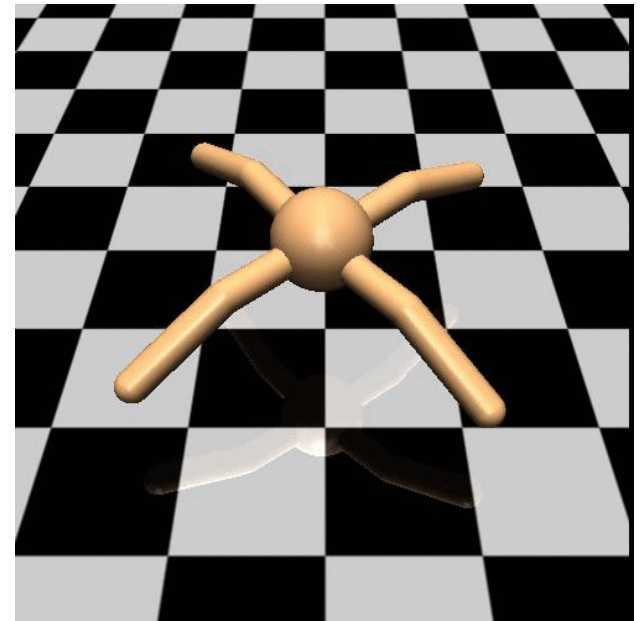
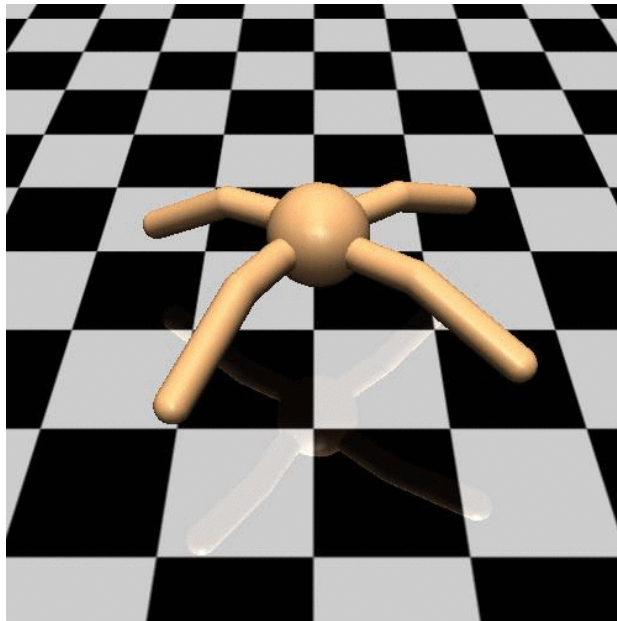


What Skills are Learned?

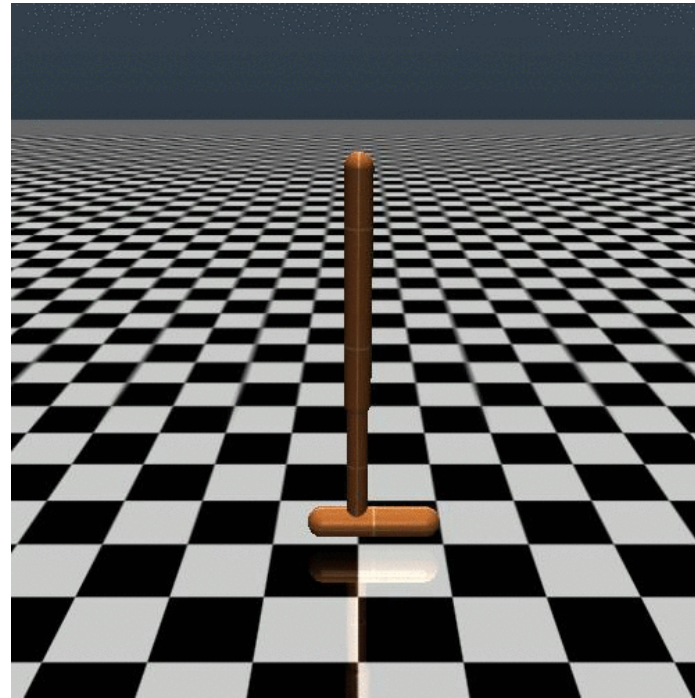
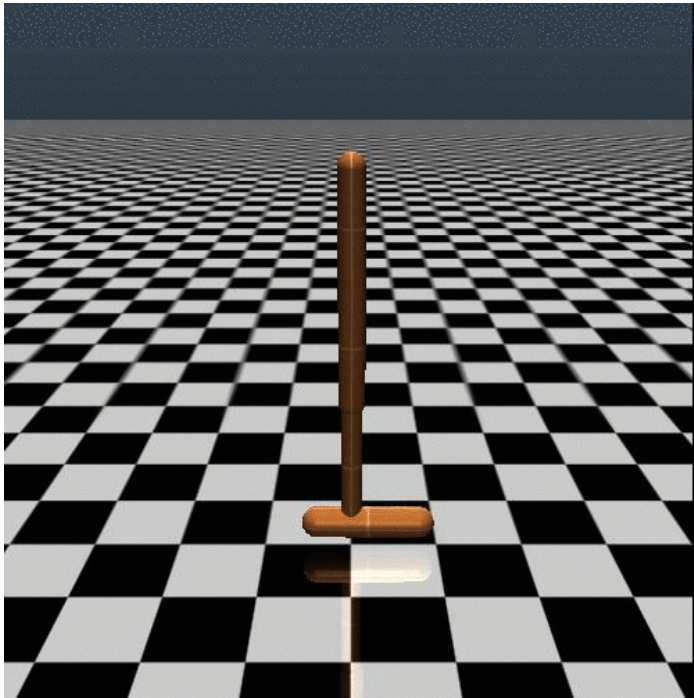
Skills for different front flips



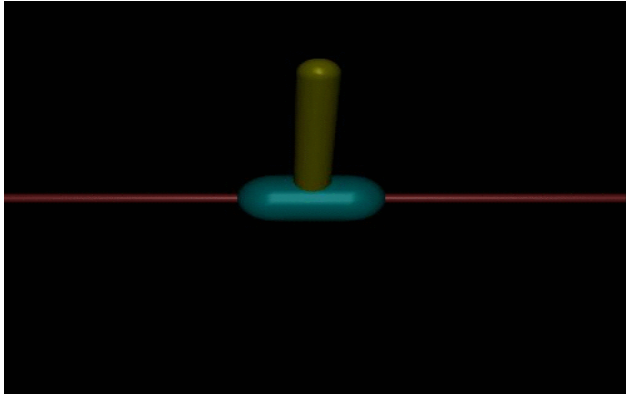
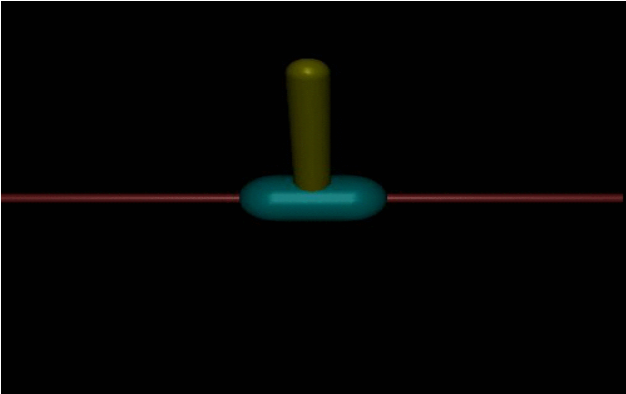
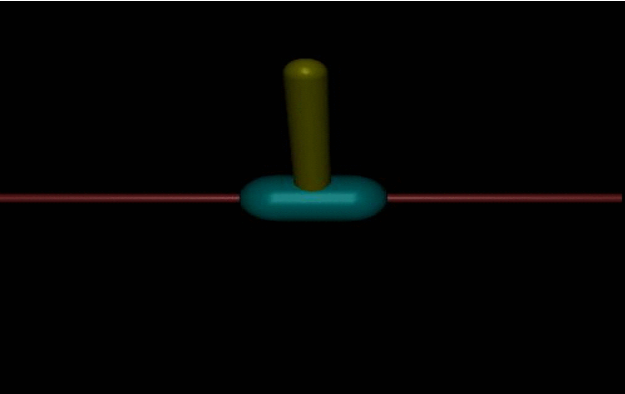
What Skills are Learned?



What Skills are Learned?



What Skills are Learned?



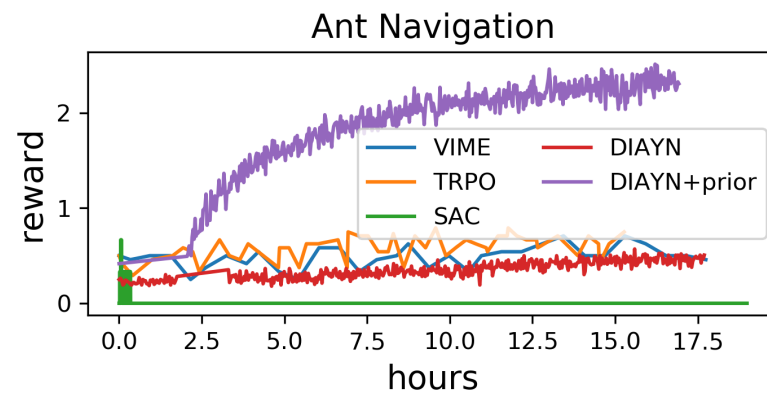
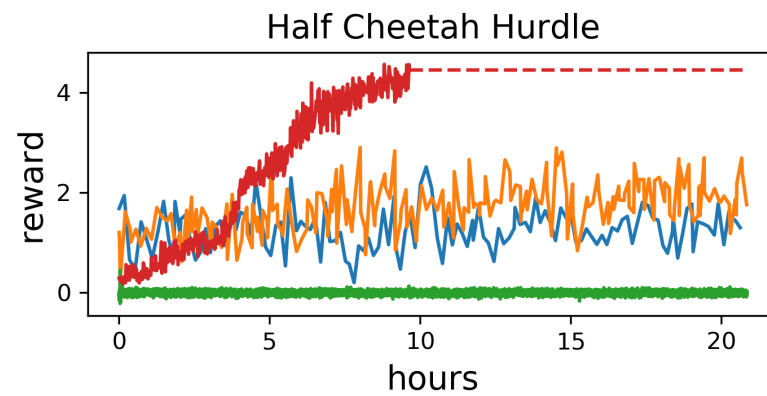
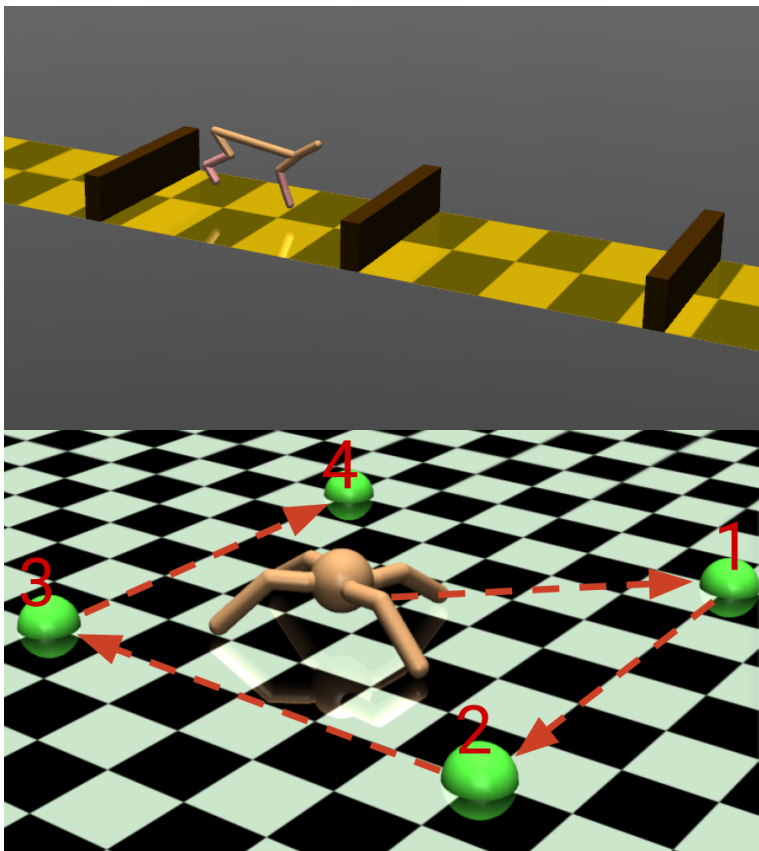
Why is DIAYN useful?

Returns a policy $\pi_{\theta}(a \mid s, z)$ with a low dimensional knob that spans a large set of behaviors.

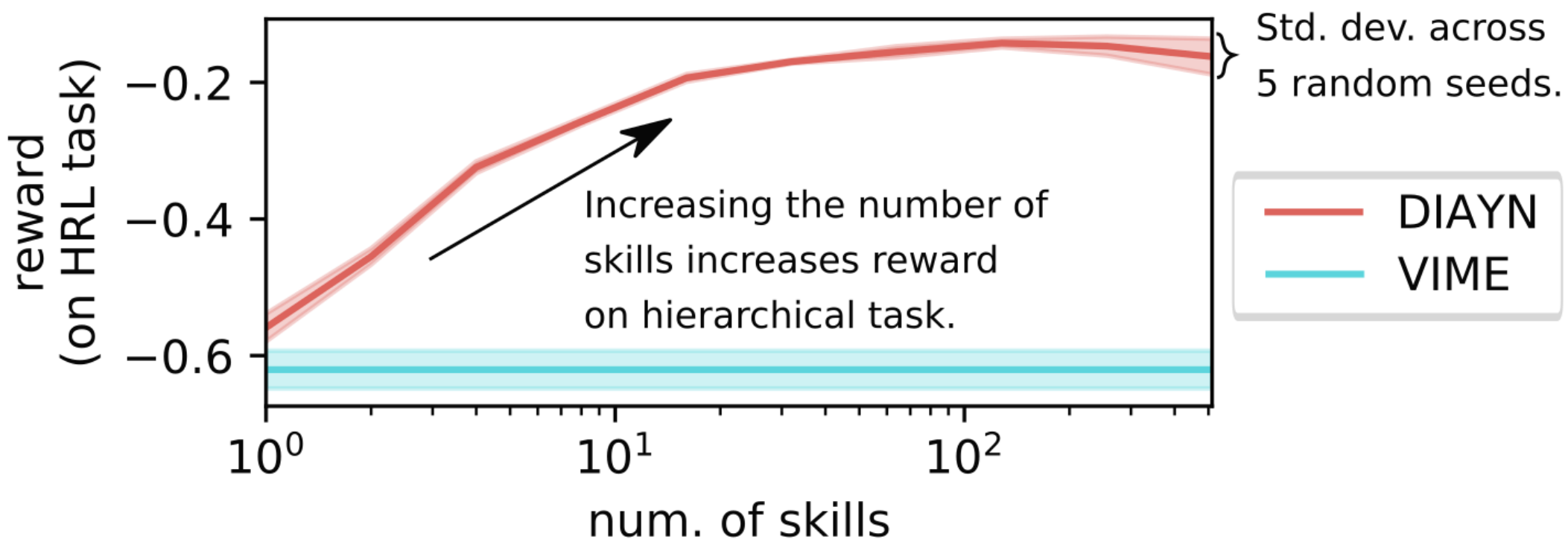
Applications of DIAYN:

- Hierarchical RL
- Imitation Learning
- Learn an environment-specific policy initialization
- Unsupervised Meta-Learning

DIAYN for Hierarchical RL



Effect of Number of Skills



Discussion Exercise

What are some potential downsides to using DIAYN? In robotics? In high-dimensional state-action spaces?

(Laith Altarabishi) How do the authors reason and explore the dimensionality of the skill latent space? How much does the dimensionality impact performance, and what is the relationship between the environment's complexity and the necessary dimensionality of the skill space?

Reading Responses

(William Avery) The authors suggest DIAYN can be used as an effective pre-training task for a supervised, sample-efficient finetuning of a target task. If domains are similar enough, could DIAYN potentially create opportunities for transfer learning?

Reading Responses

(Chloe Chen) In general how might extrinsic rewards play into these equations? Can an agent trained with this method then be put into a more traditional framework? What might that look at?

Final Logistics

Next week: Learning from Human Input
Reading assignments due **2PM Monday**

Final project due at **11:59pm on Monday, 4/29**