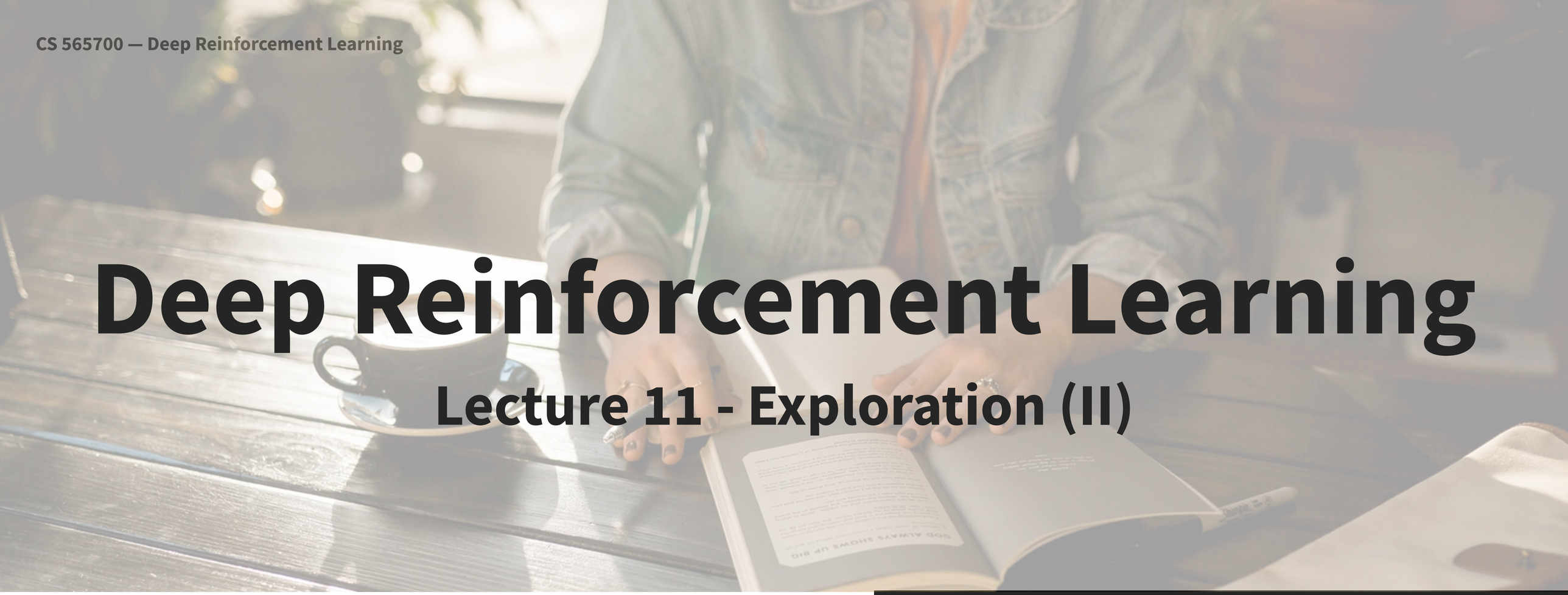


Deep Reinforcement Learning

Lecture 11 - Exploration (II)

國立清華大學
NATIONAL TSING HUA UNIVERSITY

National Tsing Hua University
Department of Computer Science

Prof. Chun-Yi Lee

Exploration Strategies

- **Classical strategies**
  - Optimistic Exploration
  - Posterior Sampling
- **Recent reinforcement learning strategies**
  - Epsilon-Greedy
  - Boltzmann Exploration
  - Entropy term
  - Noise-based exploration
  - Information Gain
- **More recent DRL exploration strategies (next time)**

Exploration Strategies

- **Diversity is All You Need (DIAYN)**
- Random Network Distilation (RND)
- Never Give Up (NGU)
- Go-Explore

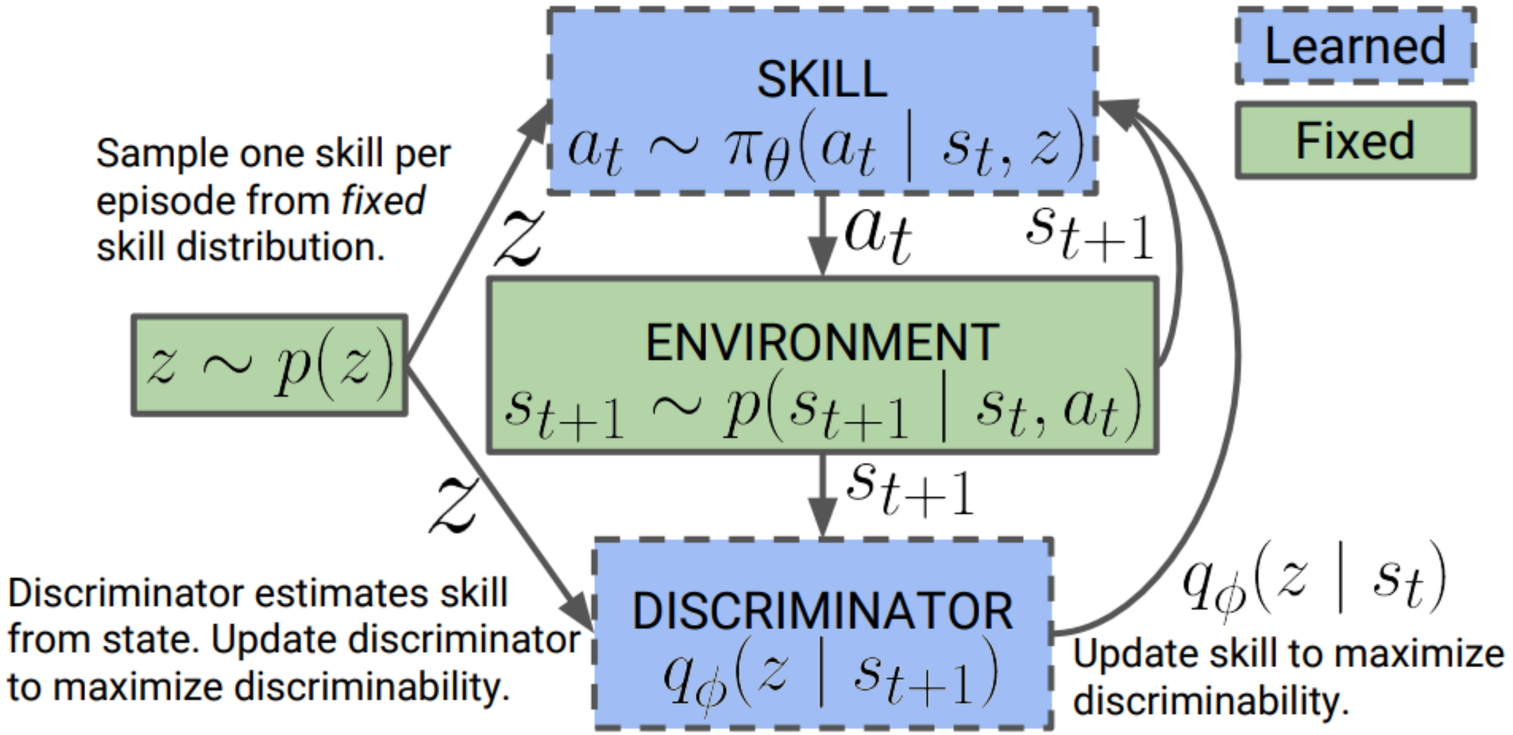
Diversity is All You Need (DIAYN)

Objectives

- Maximize an information-based objective
- Force the policy to learn different skills
- This method does not need the true reward signal
    - Learning skills without reward

Diversity is All You Need (DIAYN)

The architecture of DIAYN



Diversity is All You Need (DIAYN)

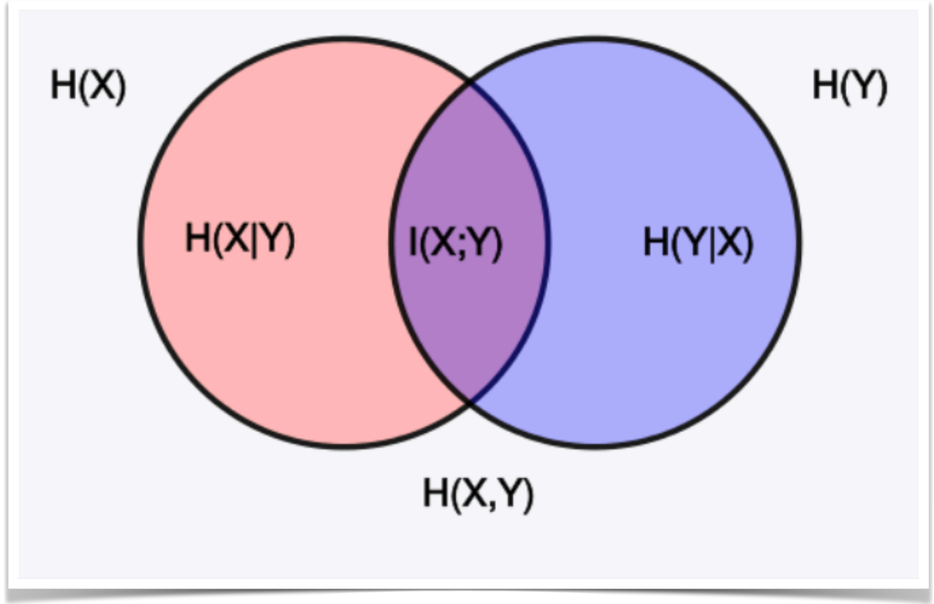
Notations

- Z : A latent variable as skill in system
- A : Action, S : State
- I : The mutual information
  - The information obtained about one random variable through observing the other random variable.
- H : The entropy function

Diversity is All You Need (DIAYN)

Background: Mutual information

- I : mutual information



- $I(X; Y) = H(Y) - H(Y \mid X)$

Diversity is All You Need (DIAYN)

The main motivations - learning to play with no rewards

- Three main ideas to design the skill,
- 1. Maximize the mutual information $I(S; Z)$, to encode the idea that the skill should control which states the agent visits

Diversity is All You Need (DIAYN)

The main motivations - learning to play with no rewards

- 2. Minimize the mutual information $I(A; Z \mid S)$, to ensure that states, not actions, are used to distinguish skills.
- 3. Maximize the entropy $H(A \mid S)$ of this mixture policy

Diversity is All You Need (DIAYN)

Learning to play with no rewards - motivation

- In summary, the objective function of DIAYN is represented as:

$$
\begin{aligned}
\mathcal{F}(\theta) &\triangleq I(S;Z) + \mathcal{H}[A \mid S] - I(A;Z \mid S) \\
&= (\mathcal{H}[Z] - \mathcal{H}[Z \mid S]) + \mathcal{H}[A \mid S] - (\mathcal{H}[A \mid S] - \mathcal{H}[A \mid S, Z]) \\
&= \mathcal{H}[Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[A \mid S, Z]
\end{aligned}
$$

Diversity is All You Need (DIAYN)

Learning to play with no rewards - motivation

- To replace $p(z \mid s)$, DIAYN approximates this posterior with a learning discriminator $q_\phi(z \mid s)$
- The final object function of DIAYN therefore becomes:

$$
\begin{aligned}
\mathcal{F}(\theta) &= \mathcal{H}[A \mid S, Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[Z] \\
&= \mathcal{H}[A \mid S, Z] + \mathbb{E}_{z \sim p(z), s \sim \pi(z)}[\log p(z \mid s)] - \mathbb{E}_{z \sim p(z)}[\log p(z)] \\
&\geq \mathcal{H}[A \mid S, Z] + \mathbb{E}_{z \sim p(z), s \sim \pi(z)}[\log q_\phi(z \mid s) - \log p(z)] \triangleq \mathcal{G}(\theta, \phi)
\end{aligned}
$$

Diversity is All You Need (DIAYN)

Implementation and the pseudo-code

$$\mathcal{H}[A \mid S, Z] + \mathbb{E}_{z \sim p(z), s \sim \pi(z)}[\log q_\phi(z \mid s) - \log p(z)]$$

An entropy term similar
in Soft Actor Critic (SAC) objective

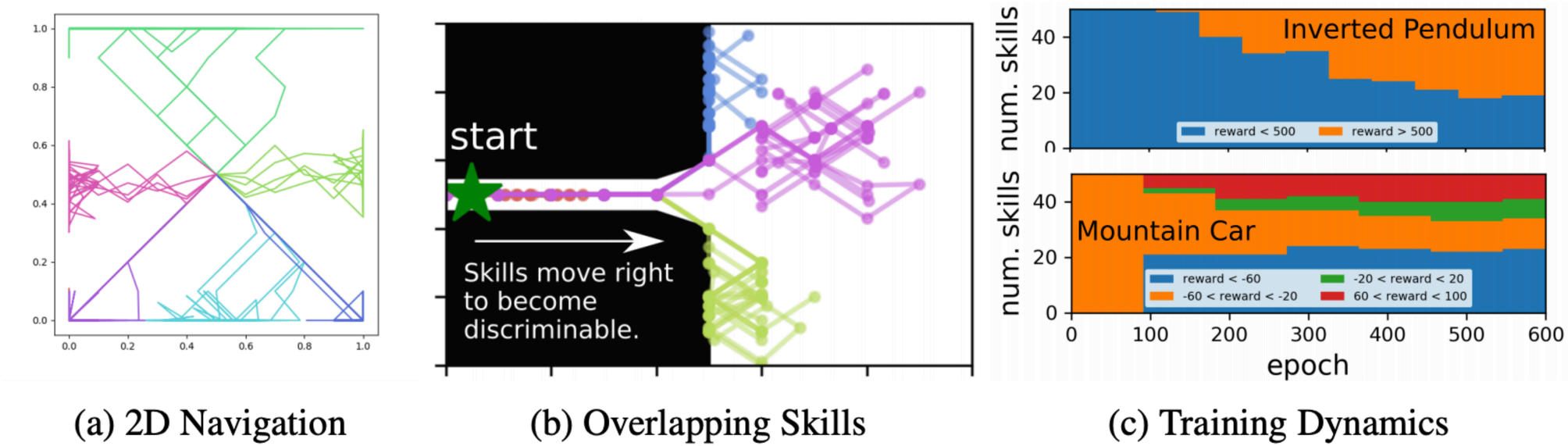
Taking this term as the pseudo-reward,
we can encode the objective into
Soft RL methods

Algorithm 1: DIAYN

while *not converged* **do**
  Sample skill $z \sim p(z)$ and initial state $s_0 \sim p_0(s)$
  for $t \leftarrow 1$ **to** *steps_per_episode* **do**
    Sample action $a_t \sim \pi_\theta(a_t \mid s_t, z)$ from skill.
    Step environment: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$.
    Compute $q_\phi(z \mid s_{t+1})$ with discriminator.
    Set skill reward $r_t = \log q_\phi(z \mid s_{t+1}) - \log p(z)$
    Update policy (θ) to maximize r_t with SAC.
    Update discriminator (ϕ) with SGD.

Diversity is All You Need

What skills have been learned?

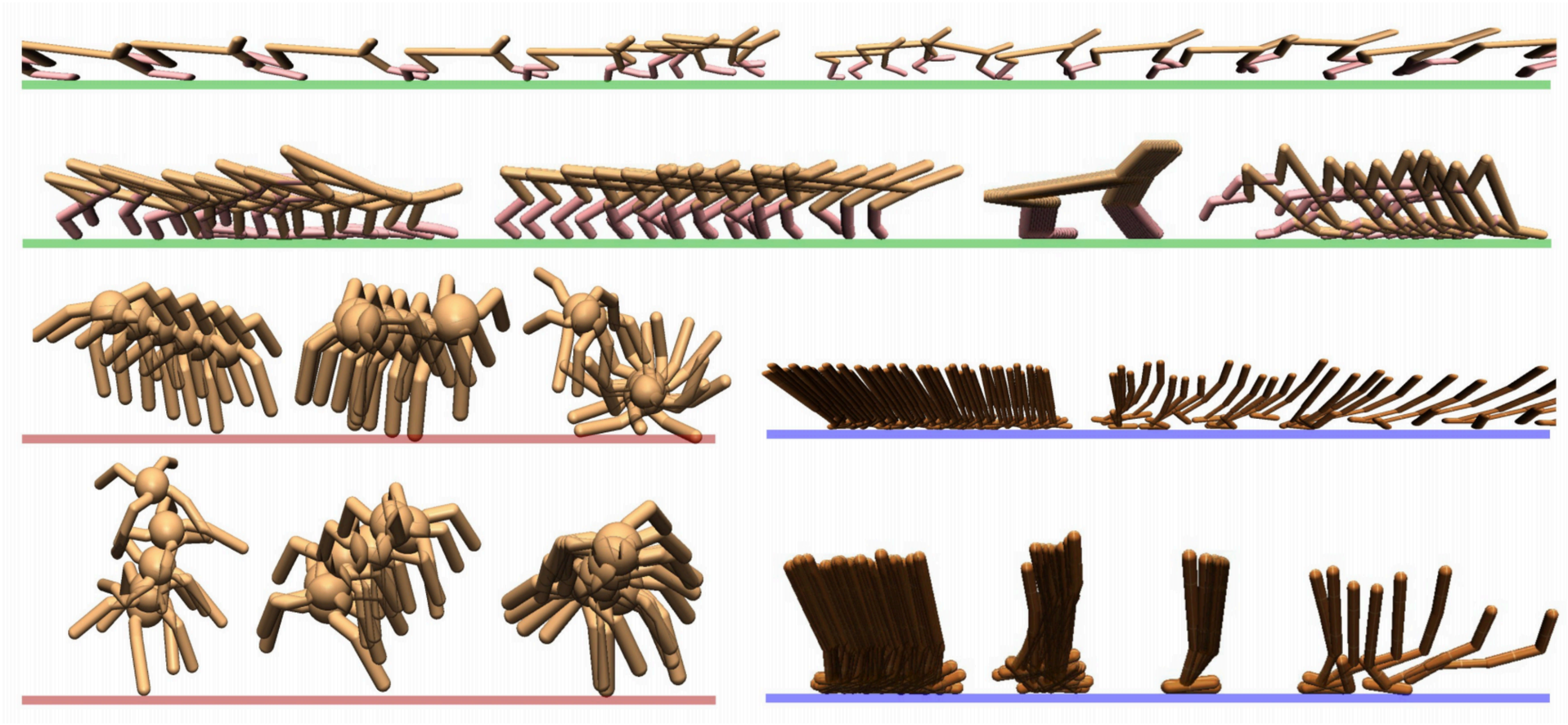


(a) 2D Navigation (b) Overlapping Skills (c) Training Dynamics

Figure 2: *(Left)* DIAYN skills in a simple navigation environment; *(Center)* skills can overlap if they eventually become distinguishable; *(Right)* diversity of the rewards increases throughout training.

Diversity is All You Need (DIAYN)

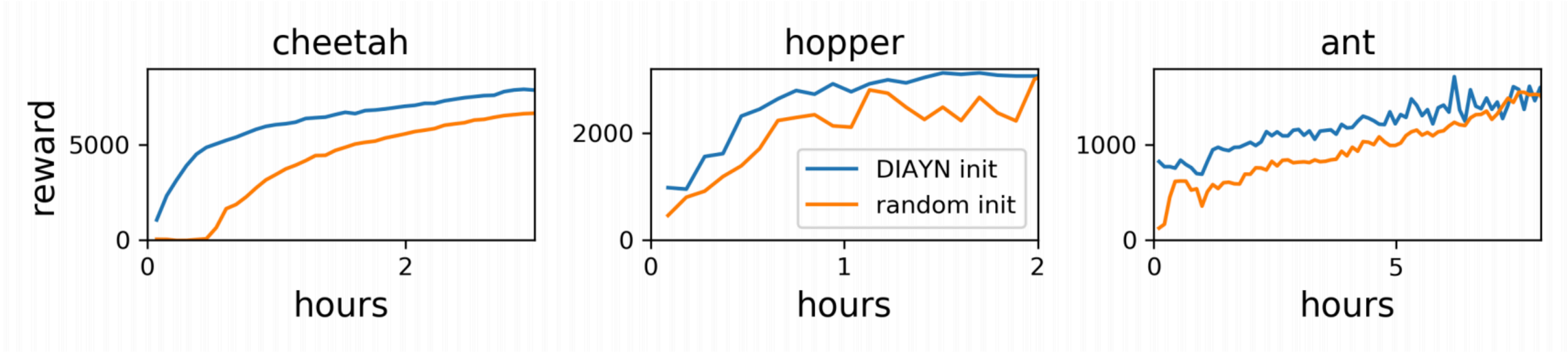
What skills have been learned?



Diversity is All You Need (DIAYN)

Help initialize?

- Use DIAYN pre-trained policy, then learn as the normal case

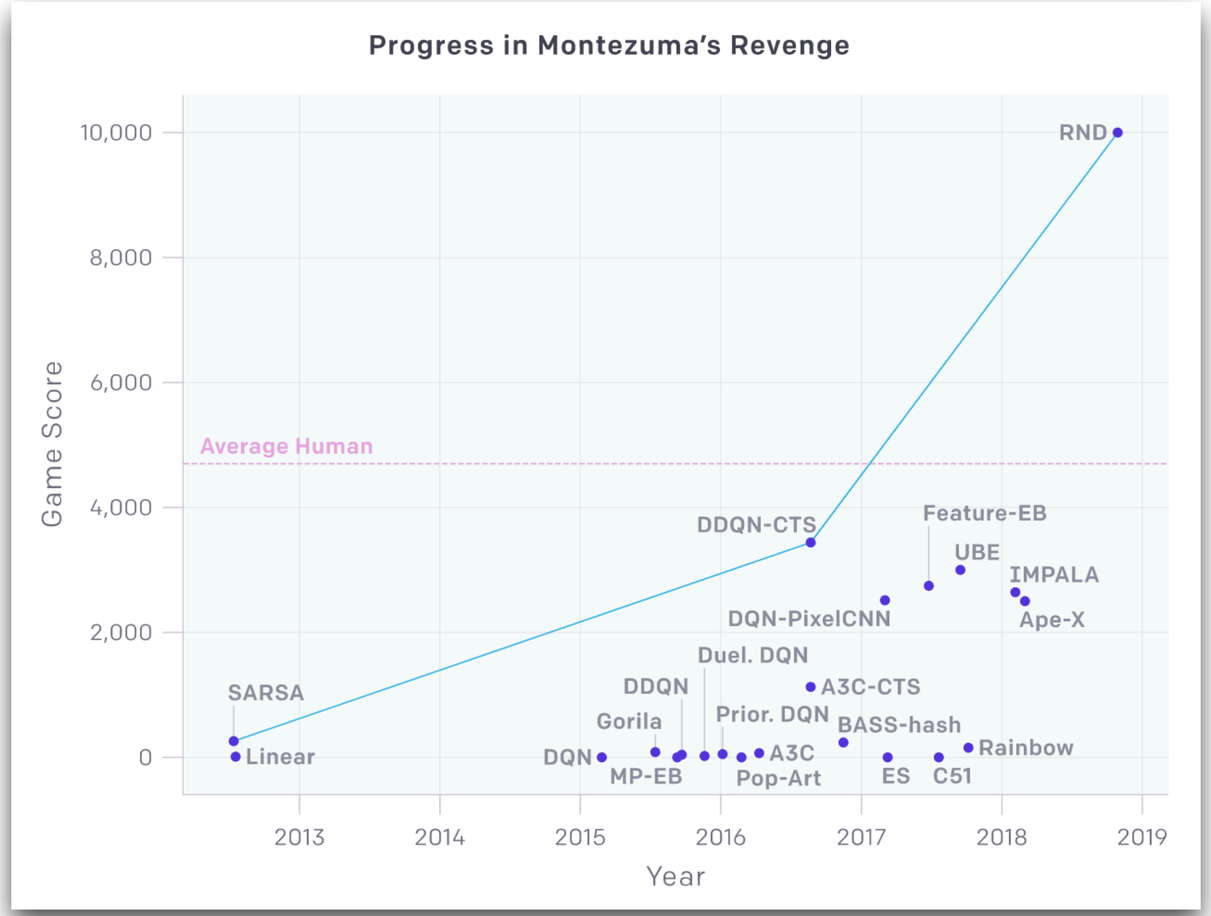


Exploration Strategies

- Diversity is All You Need (DIAYN)
- Random Network Distilation (RND)
- Never Give Up (NGU)
- Go-Explore

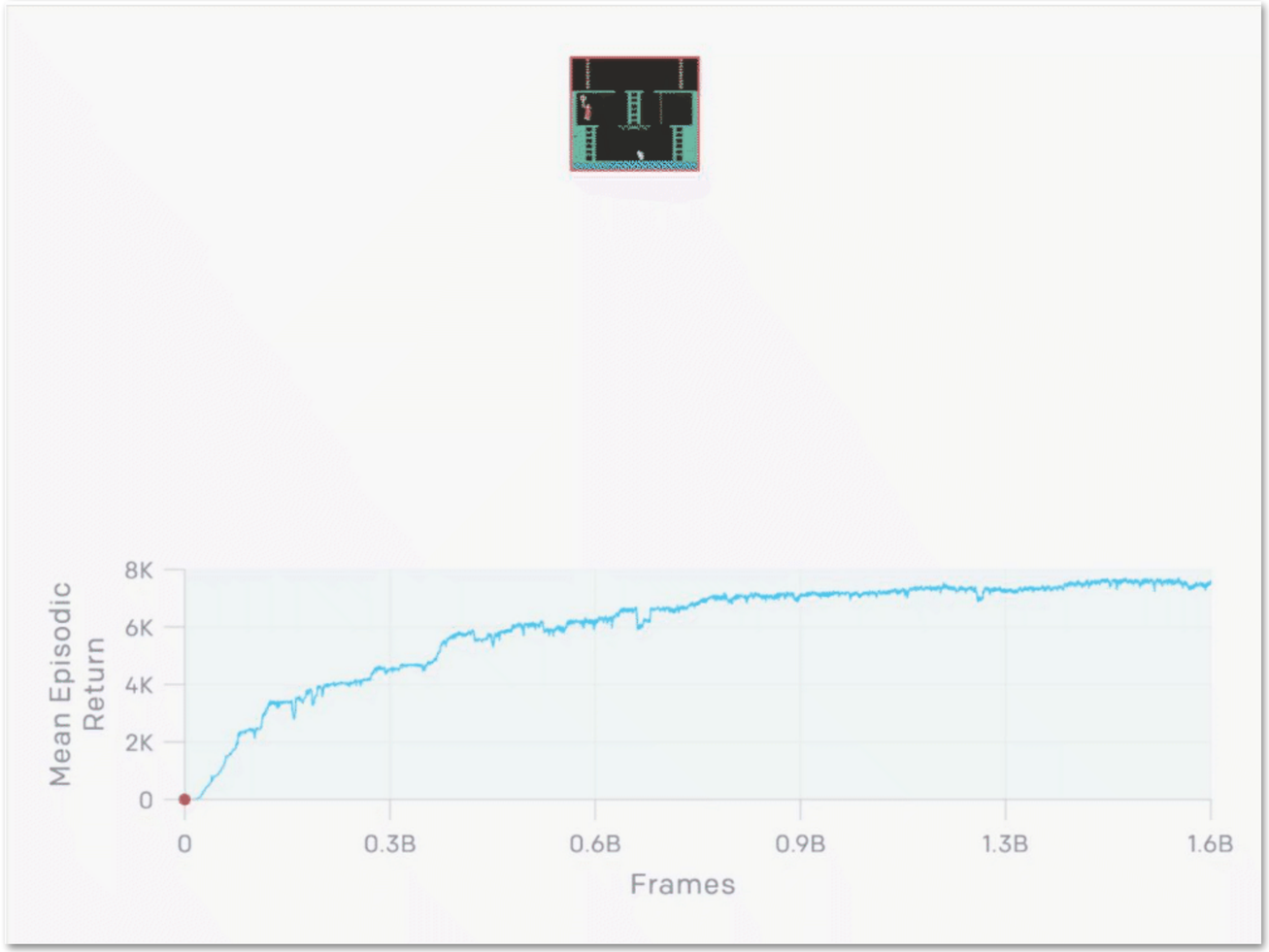
Random Network Distillation (RND)

A breakthrough in Montezuma's Revenge



Random Network Distillation (RND)

A breakthrough in Montezuma's Revenge



Random Network Distillation (RND)

Source of prediction errors

1. **Amount of training data** — Prediction error is high where few similar examples were seen by the predictor (epistemic uncertainty)
2. **Stochasticity** — Prediction error is high because the target function is stochastic (aleatoric uncertainty). Stochastic transitions are a source of such error for forward dynamics prediction
3. **Model misspecification** — Prediction error is high because necessary information is missing, or the model class is too limited to fit the complexity of the target function
4. **Learning dynamics** — Prediction error is high because the optimization process fails to find a predictor in the model class that best approximates the target function

Random Network Distillation (RND)

Source of prediction errors

Factor (1) ***Amount of training data*** is what allows one to use prediction error as an exploration bonus.

Factor (2) ***Stochasticity*** can result in the Noisy TV problem

RND obviates factors 2 and 3 since **the target network can be chosen to be deterministic** and inside the model-class of the predictor network.

Random Network Distillation (RND)

Random network distillation

A different approach where **the prediction problem is randomly generated**

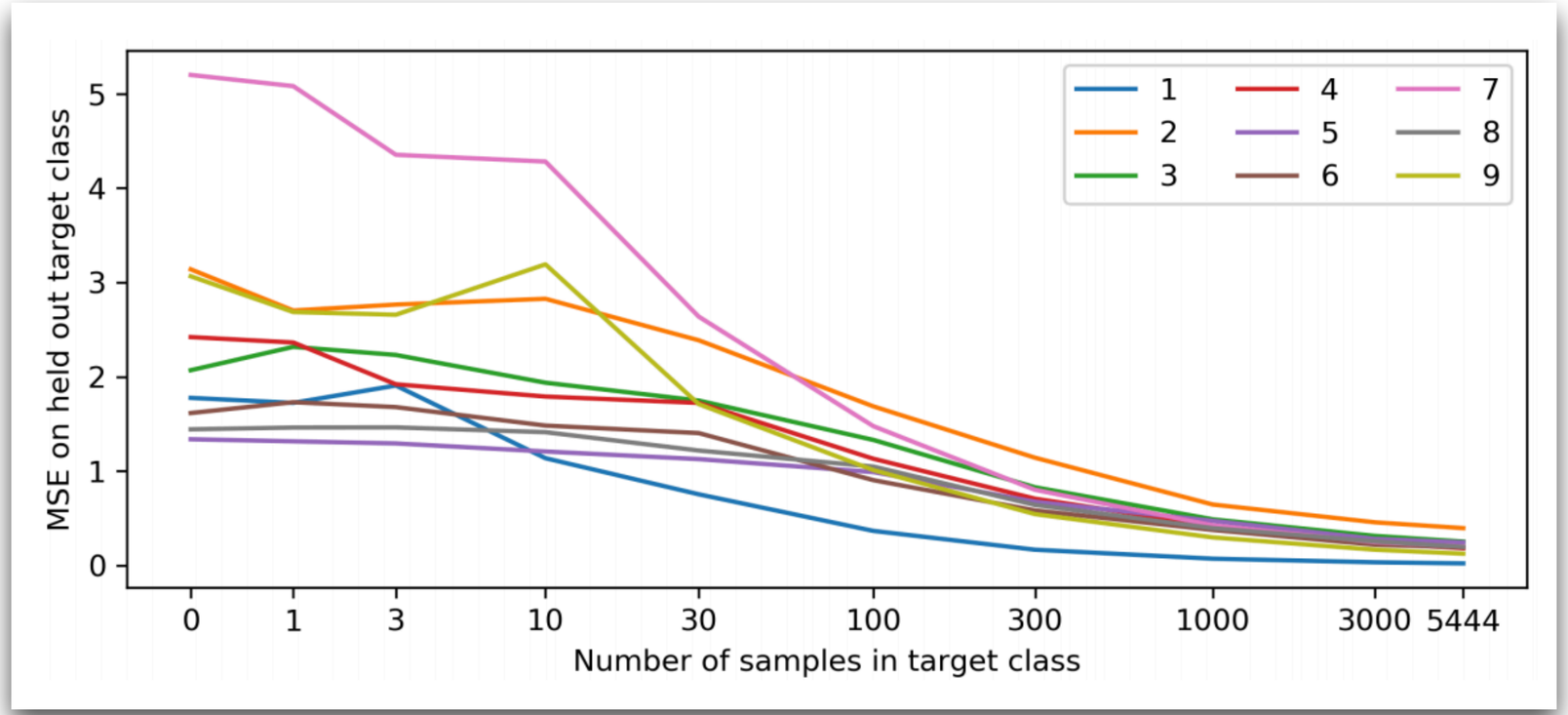
- This involves two neural networks:
  - A fixed and randomly initialized ***target*** network which sets the prediction problem
  - A ***predictor*** network trained on the data collected by the agent.

The ***target*** network takes an observation transforms it to an embedding $f : O \rightarrow \mathbb{R}^k$

The ***predictor*** network $\hat{f} : O \rightarrow \mathbb{R}^k$ is trained by gradient descent to minimize the expected MSE $||\hat{f}(x;\theta) - f(x)||^2$ with respect to its parameter $\theta_{\hat{f}}$.

Random Network Distillation (RND)

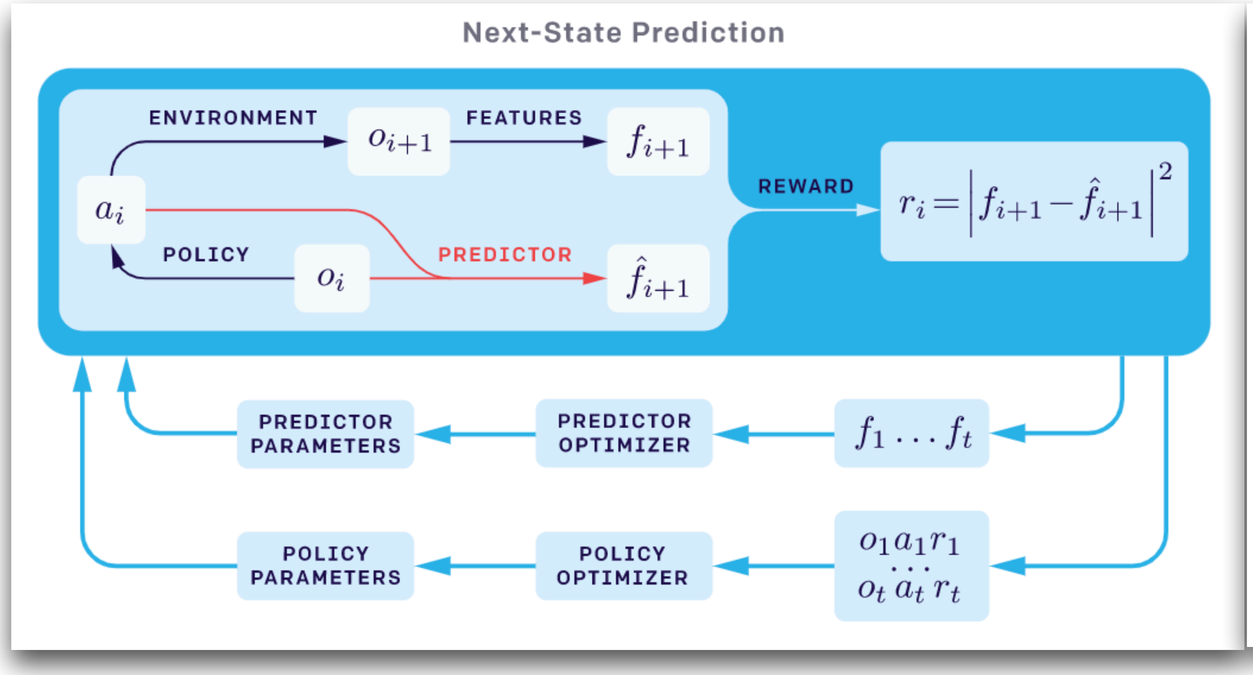
A toy model on MNSIT - novelty detection

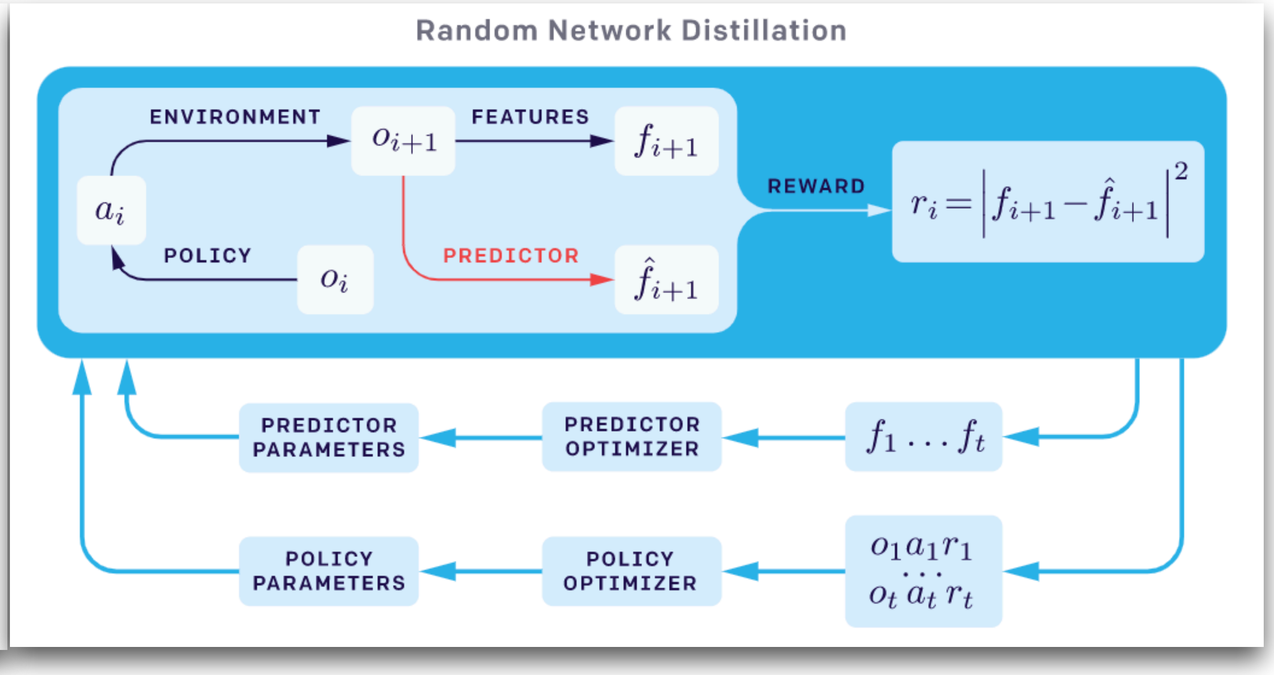


A predictor network is able to mimics a randomly initialized target network on MNIST dataset.

Random Network Distillation (RND)

Comparison of next-state prediction with RND





Random Network Distillation (RND)

Combining intrinsic and extrinsic returns

In preliminary experiments that used only intrinsic rewards

- Treating the problem as ***non-episodic*** resulted in better exploration
- The ***return is not truncated at "game over"***

It is also argued in [Burda et al., 2018] that **using episodic intrinsic rewards can leak information about the task** to the agent.

[Burda et al., 2018] Yuri Burda and *et al*. Large-scale study of curiosity-driven learning.

Random Network Distillation (RND)

The way how humans explore the games

Scenario — Fubuki is playing a video game and is attempting a tricky maneuver to reach a suspected secret room

- Because the maneuver is tricky the chance of a game over is high
- The payoff to Fubuki's curiosity will be high if she succeeds

If Fubuki is modeled as an ***episodic*** reinforcement learning agent, then her future return will be **exactly zero** if she gets a game over, which might make her **overly risk averse**

Random Network Distillation (RND)

The way how humans explore the games

Scenario — Fubuki is playing a video game and is attempting a tricky maneuver to reach a suspected secret room.

- Because the maneuver is tricky the chance of a game over is high
- The payoff to Fubuki's curiosity will be high if she succeeds.

The real cost of a game over to Fubuki is **the opportunity cost incurred by having to play through the game from the beginning** (which is presumably less interesting to Fubuki having played the game for some time)

Random Network Distillation (RND)

Combining intrinsic and extrinsic returns

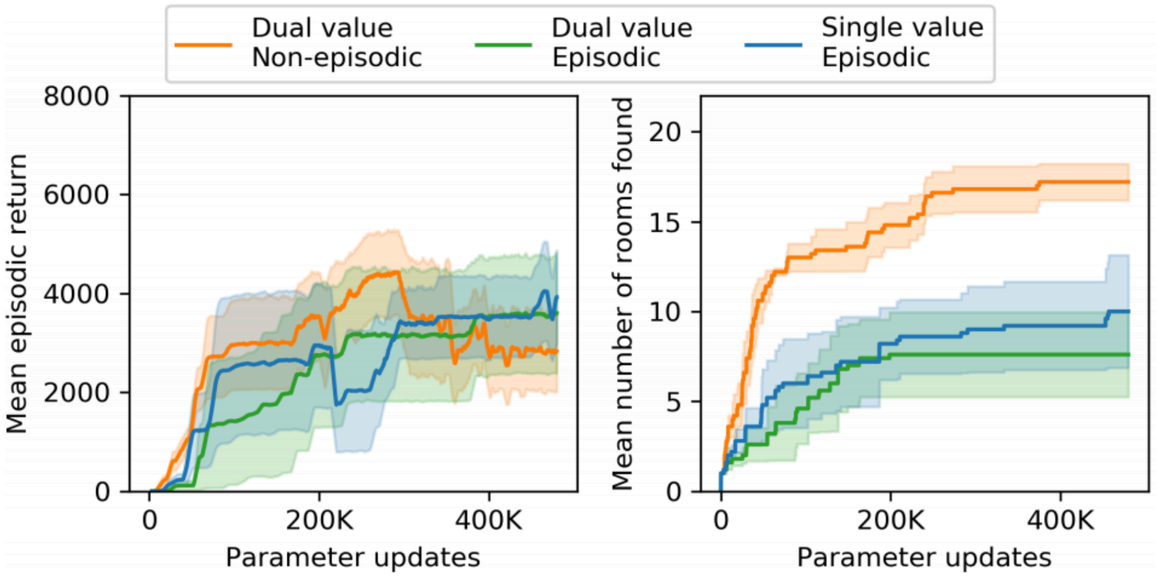
It is not obvious how to estimate the combined value of the non-episodic stream of intrinsic rewards i_t and the episodic stream of extrinsic rewards e_t

Solution — The total return R is linear and thus can be decomposed as a sum $R = R_i + R_e$ of intrinsic and extrinsic returns, respectively

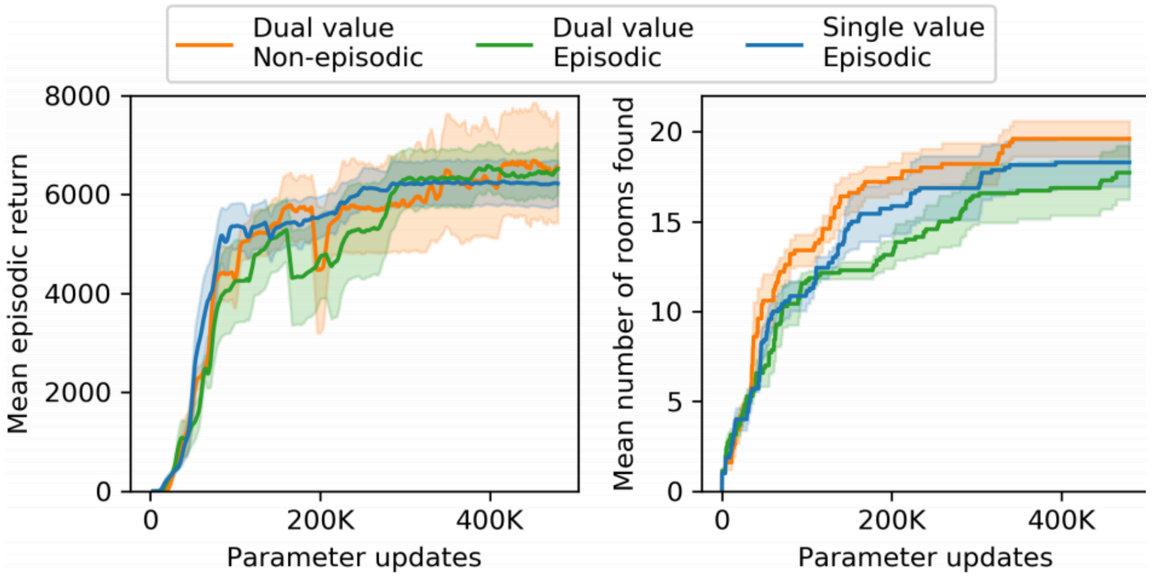
Hence, we can fit two value heads V_i and V_e separately using their respective return.

Random Network Distillation (RND)

Combining intrinsic and extrinsic returns



(a) RNN policies

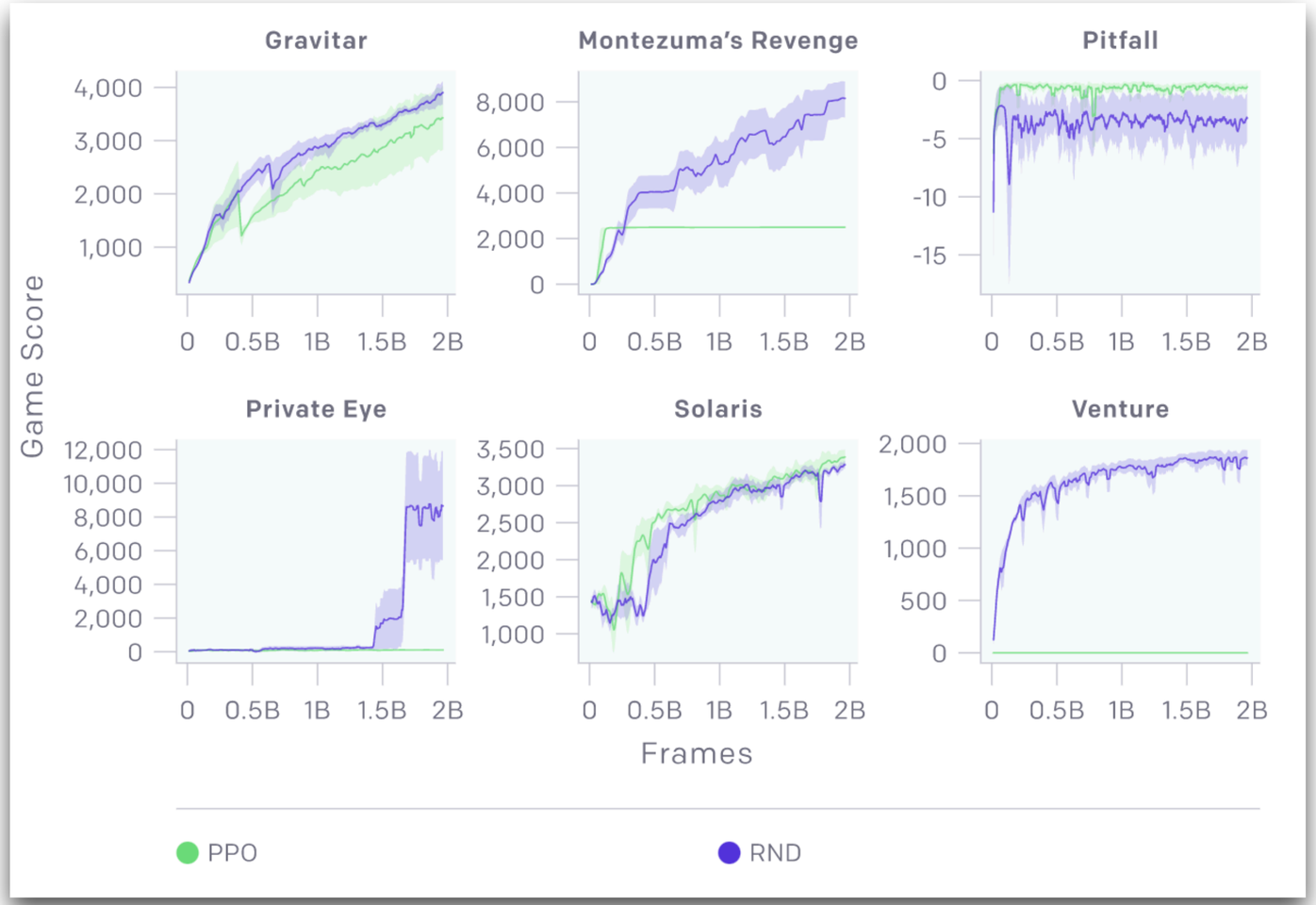


(b) CNN policies

Compare episodic intrinsic rewards to non-episodic intrinsic rewards combined with episodic extrinsic rewards.

Random Network Distillation (RND)

Experimental Results



Exploration Strategies

- Diversity is All You Need (DIAYN)
- Random Network Distilation (RND)
- **Never Give Up (NGU)**
- Go-Explore

Never Give Up: Learning Directed Exploration Strategies

Never Give Up: Learning Directed Exploration Strategies

Highlights

- First reinforcement learning agent able to solve hard exploration games by **learning a range of directed exploratory policies**
- First algorithm to achieve non-zero rewards (with a mean score of 8,400) in the game of **Pitfall!** without using demonstrations or hand-crafted features

Never Give Up: Learning Directed Exploration Strategies

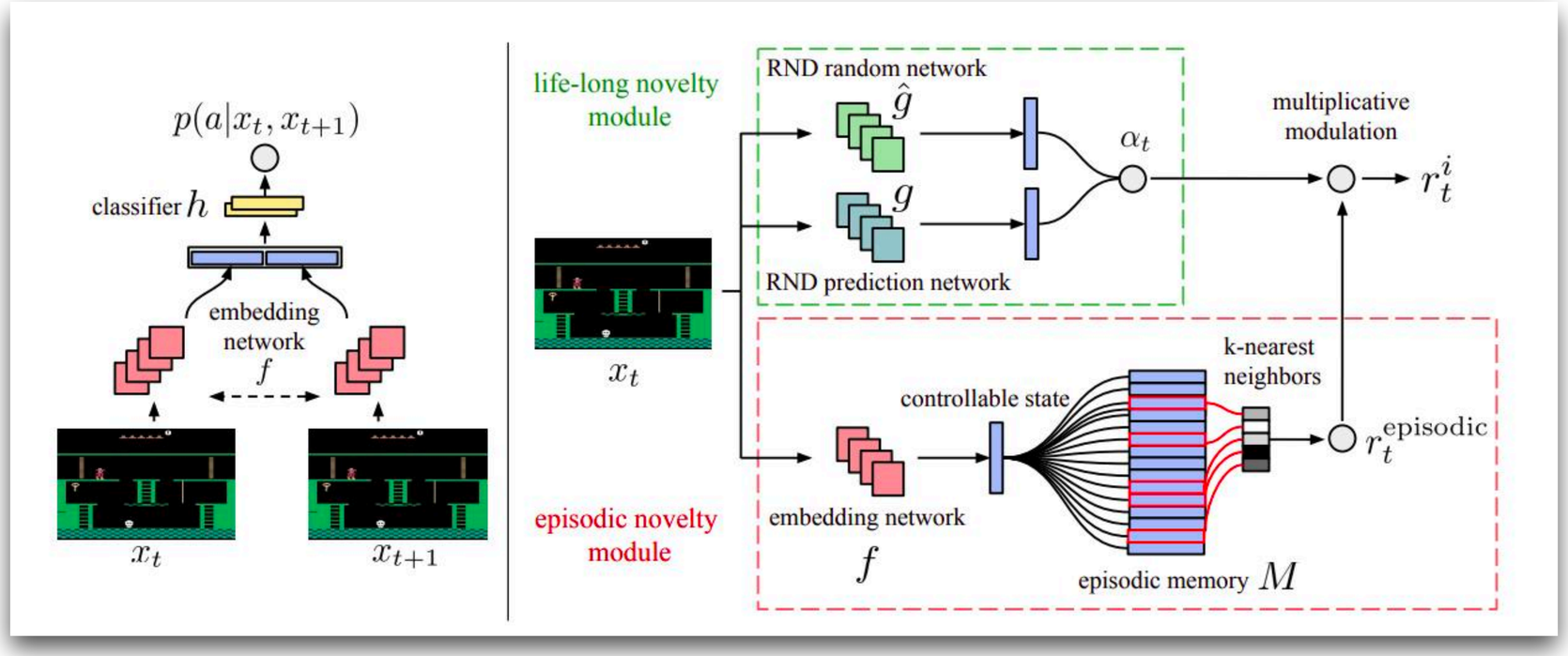
Contributions

The authors propose

- An exploration bonus combining ***life-long*** and ***episodic*** novelty to learn exploratory strategies that can **maintain exploration** throughout the agent's training process (to never give up)
- To learn a family of policies that separate exploration and exploitation using a conditional architecture with shared weights

Never Give Up: Learning Directed Exploration Strategies

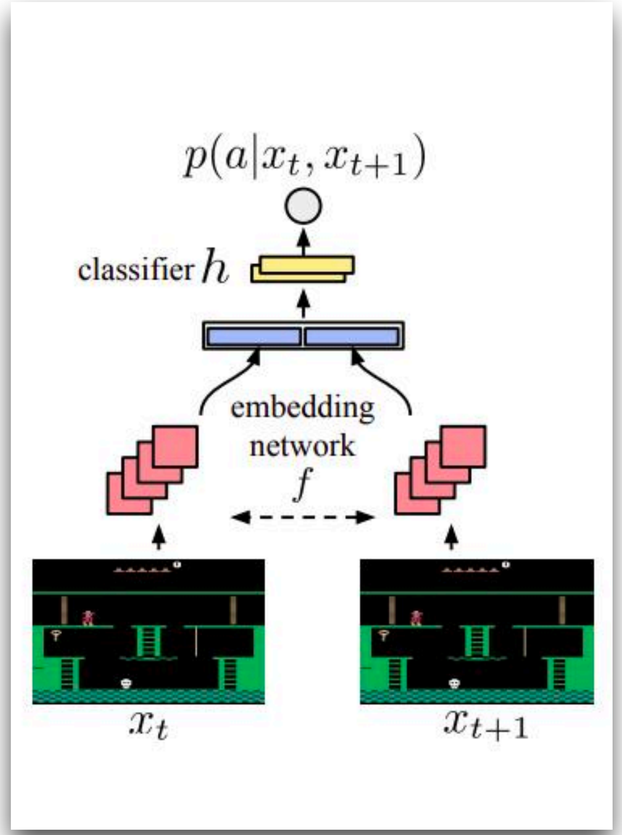
The never-give-up intrinsic reward generation architecture



The network is trained based on the augmented reward $r_t = r_t^e + \beta r_t^i$

Never Give Up: Learning Directed Exploration Strategies

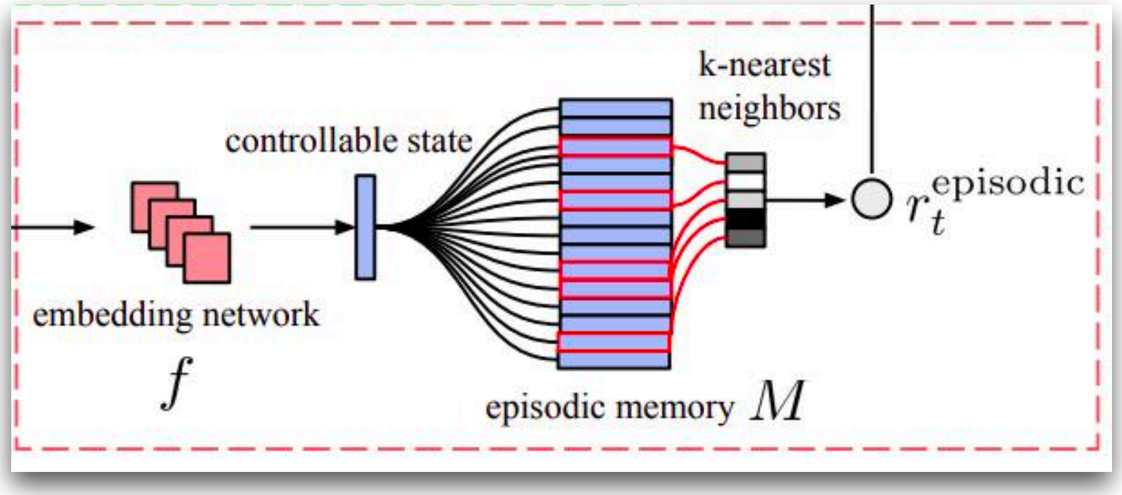
The never-give-up intrinsic reward — embedding network



- The embedding network can be thought of as a siamese network by maximizing $p(a \mid x_t, x_{t+1})$
- When used to embed states, the classifier and x_{t+1} are discarded.
- The network projects the states into a space where as much useless information as possible has been discarded.

Never Give Up: Learning Directed Exploration Strategies

The never-give-up intrinsic reward — episodic novelty module (1/3)



The embedding states are then saved in a buffer and clustered using k-NN so as to get a pseudo-count of the states.

Thus $r_t^{episodic}$ is defined by,

$$r_t^{episodic} = \frac{1}{\sqrt{n(f(x_t)}} \approx \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(x_t), f_i)} + c}$$

Never Give Up: Learning Directed Exploration Strategies

The never-give-up intrinsic reward — episodic novelty module (2/3)

$$r_t^{episodic} = \frac{1}{\sqrt{n(f(x_t)}} \approx \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(x_t), f_i) + c}}$$

- $n(f(x_t))$ is the counts for the visits to the abstract state $f(x_t)$
- Approximate the count $n(f(x_t))$ as the sum of the similarities given by a kernel function $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, over the content of episodic memory M

Never Give Up: Learning Directed Exploration Strategies

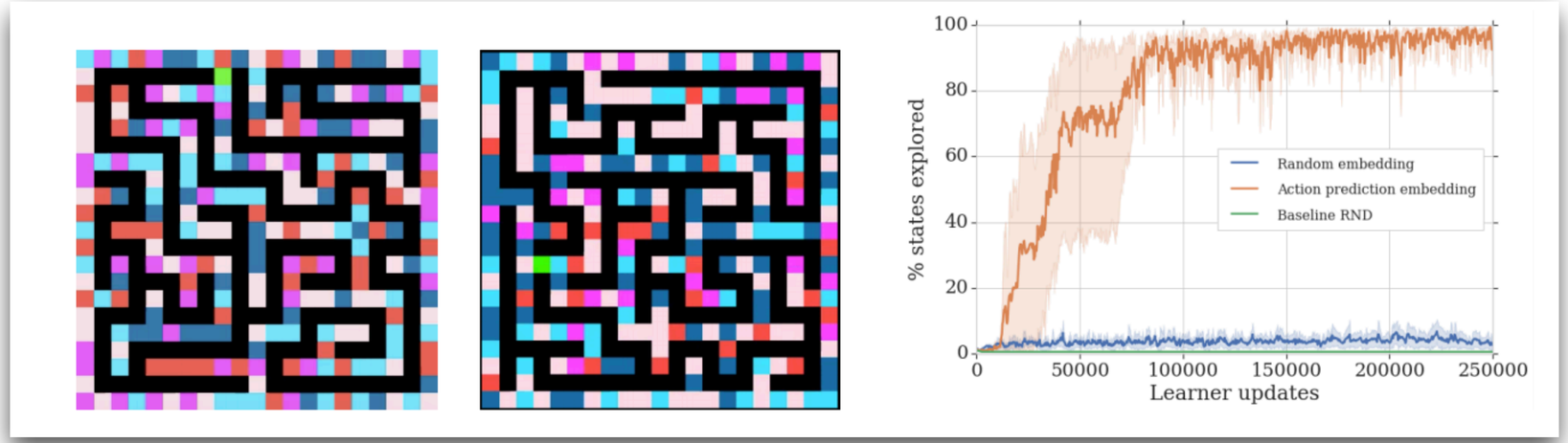
The never-give-up intrinsic reward — episodic novelty module (3/3)

$$r_t^{episodic} = \frac{1}{\sqrt{n(f(x_t)}} \approx \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(x_t), f_i)} + c}$$

- In practice, the pseudo-counts are computed using k-nearest neighbors of $f(x_t)$ in the memory M, denoted as $N_k = \{f_i\}_{i=1}^{k}$
- Use inverse kernel for K, where $K(x, y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon}$, d is the Euclidean distance and d_m is the moving average of d. (d_m is used to make the kernel more robust)

Never Give Up: Learning Directed Exploration Strategies

Experimental results



- Design an environment called ***Random Disco Maze***
  - The agent in green; the pathway in black. The colors of the wall change at every time step
  - There is no extrinsic reward
- The results show the importance of estimating the exploration bonus using a controllable state representation

Never Give Up: Learning Directed Exploration Strategies

Experimental results on hard exploration games

Algorithm	Gravitar	MR	Pitfall!	PrivateEye	Solaris	Venture
Human	3.4k	4.8k	6.5k	69.6k	**12.3k**	1.2k
Best baseline	**15.7k**	**11.6k**	0.0	11k	5.5k	2.0k
RND	3.9k	10.1k	-3	8.7k	3.3k	1.9k
R2D2+RND	15.6k±0.6k	10.4k±1.2k	-0.5±0.3	19.5k±3.5k	4.3k±0.6k	**2.7k±0.0k**
R2D2(Retrace)	13.3k±0.6k	2.3k±0.4k	-3.5±1.2	32.5k±4.7k	6.0k±1.1k	2.0k±0.0k
NGU(N=1)-RND	12.4k±0.8k	3.0k±0.0k	**15.2k±9.4k**	40.6k±0.0k	5.7k±1.8k	46.4±37.9
NGU(N=1)	11.0k±0.7k	8.7k±1.2k	9.4k±2.2k	60.6k±16.3k	5.9k±1.6k	876.3±114.5
NGU(N=32)	14.1k±0.5k	10.4k±1.6k	8.4k±4.5k	**100.0k±0.4k**	4.9k±0.3k	1.7k±0.1k

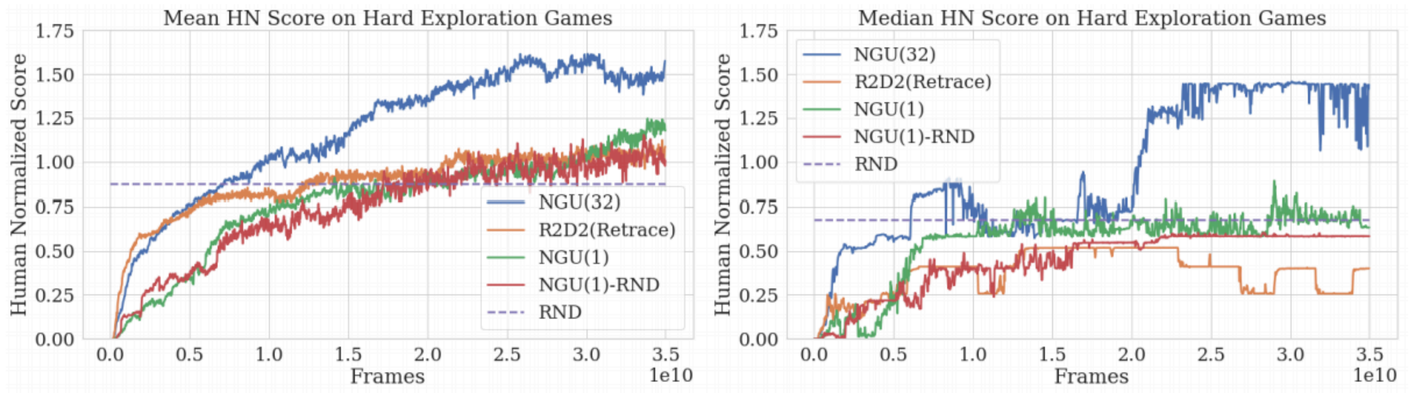
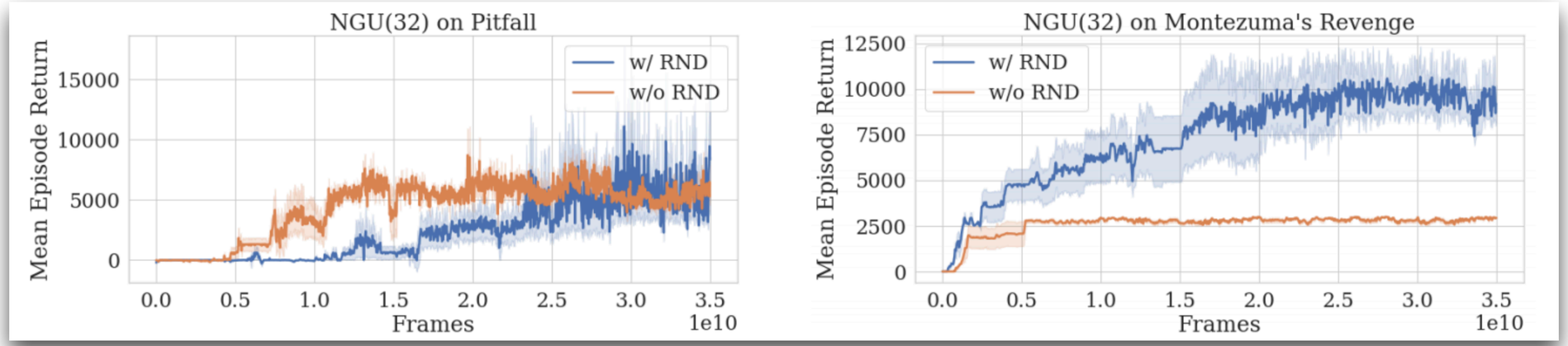


Figure 4: Human Normalized Scores on the 6 hard exploration games.

Never Give Up: Learning Directed Exploration Strategies

Experimental results on hard exploration games



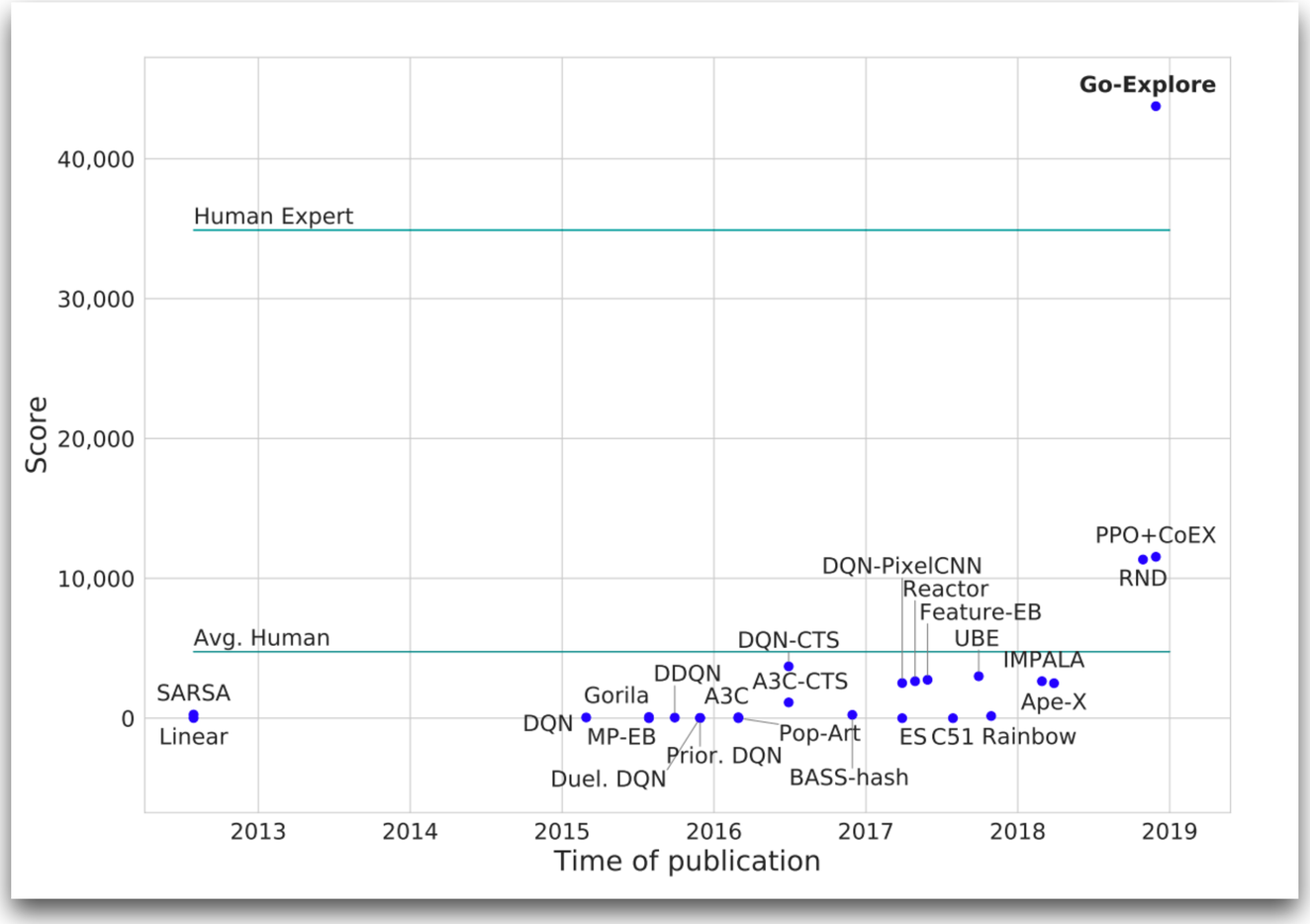
Algorithm	Pong	QBert	Breakout	Space Invaders	Beam Rider
Human	14.6	13.4k	30.5	1.6k	16.9k
R2D2	**21.0**	408.8k	837.7	43.2k	**188.2k**
R2D2+RND	20.7±0.0	353.5k±41.0k	815.8±5.3	**54.5k±2.8k**	85.7k±9.0k
R2D2(Retrace)	20.9±0.0	415.6k±55.8k	838.3±7.0	35.0k±13.0k	111.1k±5.0k
NGU(N=1)-RND	-8.1±1.7	647.1k±50.5k	**864.0±0.0**	45.3k±4.9k	166.5k±8.6k
NGU(N=1)	-9.4±2.6	**684.7k±8.8k**	**864.0±0.0**	43.0k±3.9k	114.6k±2.3k
NGU(N=32)	19.6±0.1	465.8k±84.9k	532.8±16.5	44.6k±1.2k	68.7k±11.1k

Exploration Strategies

- Diversity is All You Need (DIAYN)
- Random Network Distilation (RND)
- Never Give Up (NGU)
- Go-Explore

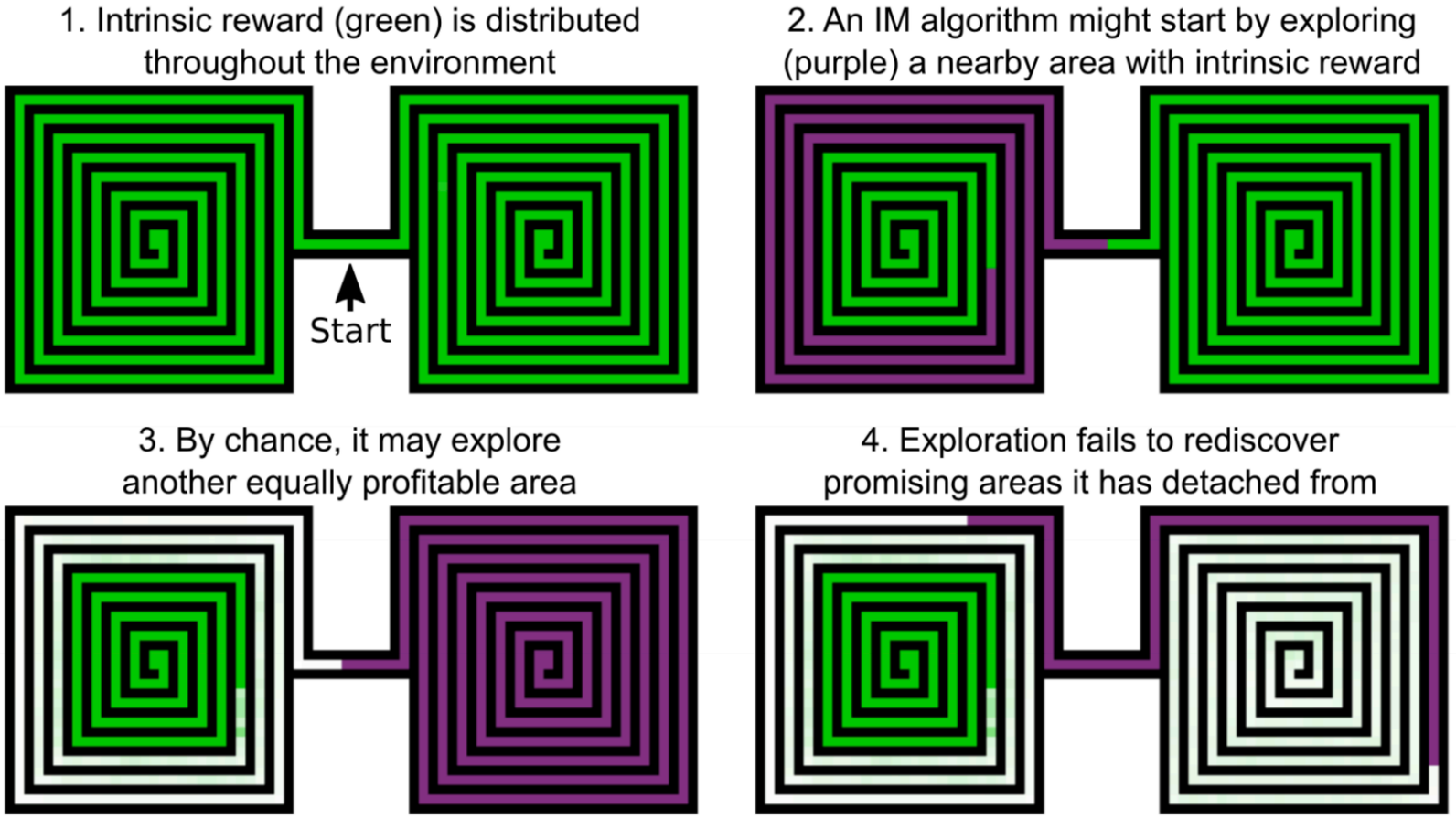
Go-Explore: A New Approach for Hard-Exploration Problems

A method that outperforms most methods in Atari games



Go-Explore: A New Approach for Hard-Exploration Problems

Problem in intrinsic reward: catastrophic forgetting



Go-Explore: A New Approach for Hard-Exploration Problems

The main idea of Go-Explore consists of two phases

1. **Phase 1**

   Go — Go to the selected state

   Explore — Start explore from that state

2. **Phase 2**

   Robustify — Use the trajectory collect from Phase 1.

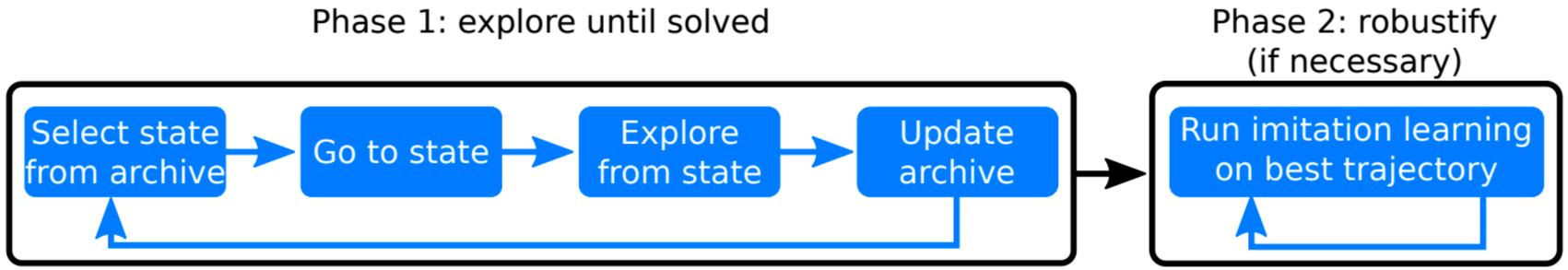


Figure 2: **A high-level overview of the Go-Explore algorithm.**

Go-Explore: A New Approach for Hard-Exploration Problems

Introduction of the concept of cells

- **Go** - Go to the selected state, How to represent the states ? Use the **cells**
  - The authors introduced the idea of **cells**. **Cells** are downscaled and grayscale images of the actual game frames

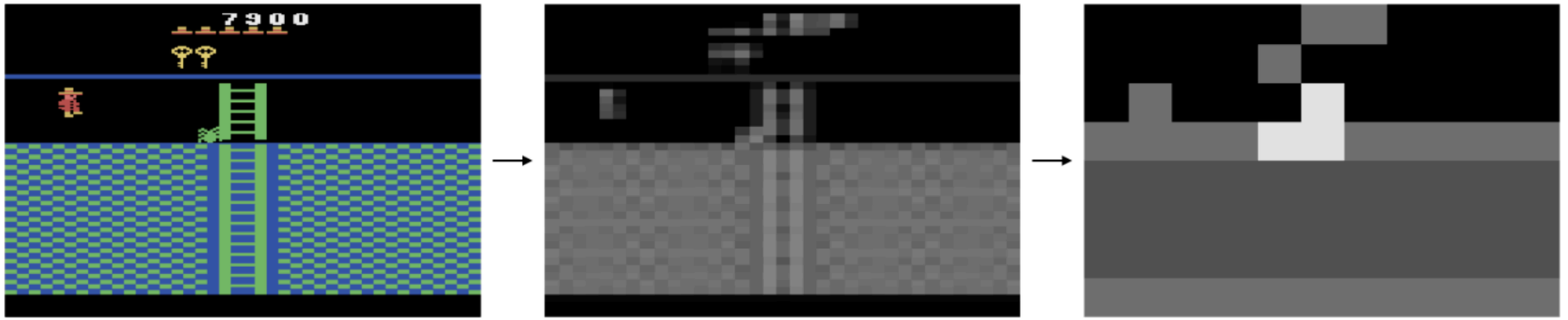


Figure 3: **Example cell representation without domain knowledge, which is simply to down-sample each game frame.** The full observable state, a color image, is converted to grayscale and downscaled to an 11×8 image with 8 possible pixel intensities.

Go-Explore: A New Approach for Hard-Exploration Problems

The objective of phase 1

- **Phase 1 objective**: Find interesting cells
  - Interesting cells that have not been found before and where the agent may collect a high reward
  - Whenever a new cell is discovered four values are added to an archive of cells
    - The full trajectory to this cell
    - State of the environment in this cell
    - Total reward achieved by this trajectory
    - Length of this trajectory
  - If a cell is encountered that is already in the archive the corresponding entry is updated in case it is ‘better’

Go-Explore: A New Approach for Hard-Exploration Problems

The **Go** operation

- **Go** — Find a cell in the archive, then use the actions of the saved trajectory to go to the specific states
  - Use a heuristic to choose a ‘good’ cell and go there
  - The heuristic might choose the cell with the highest total reward, the least-visited, etc.
  - The environment should be ‘deterministic’ and ‘resettable’

Go-Explore: A New Approach for Hard-Exploration Problems

The **Explore** operation

- **Explore** — Start exploring from a specific state
- Perform random actions to find new cells from this promising one
  - Any exploration method can be applied to find new cells

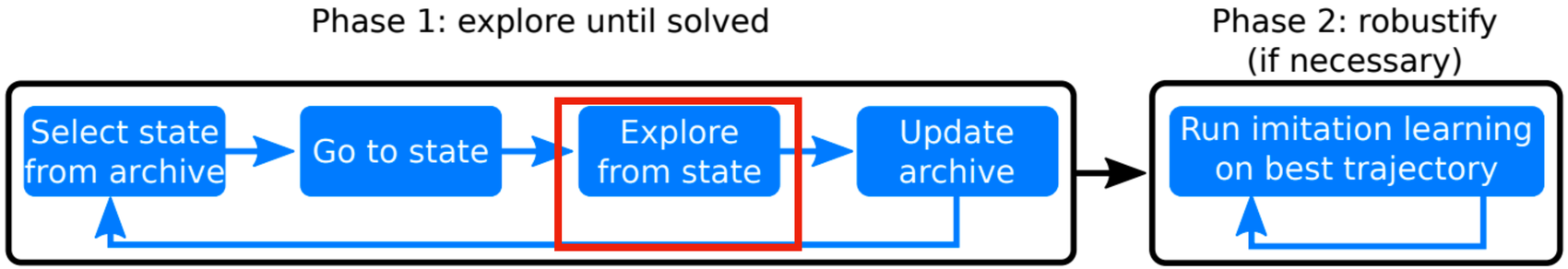


Figure 2: **A high-level overview of the Go-Explore algorithm.**

Go-Explore: A New Approach for Hard-Exploration Problems

The **Explore** operation

- **Explore -** Update the archive
- **The first condition** — when the agent visits a cell that was not yet in the archive? The cell is added into the archive with the following properties
    - The full trajectory to this cell
    - State of the environment in this cell
    - Total reward achieved by this trajectory
    - Length of this trajectory

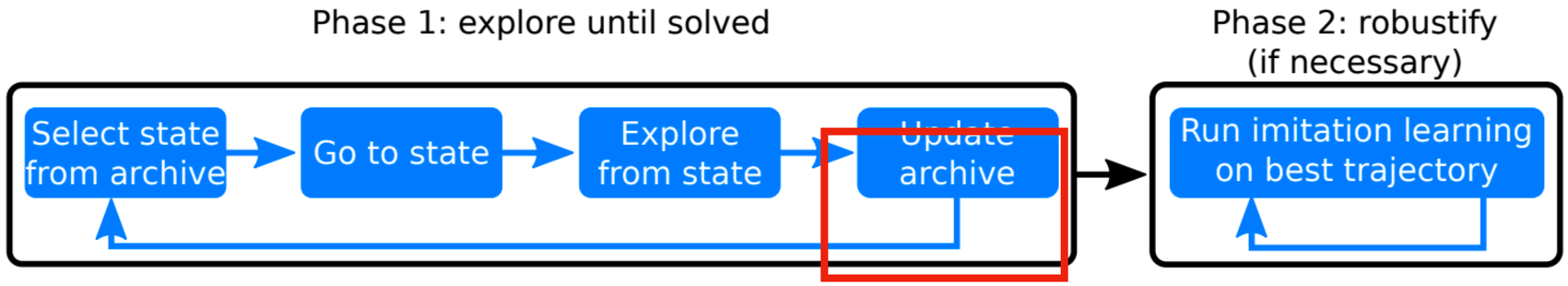


Figure 2: **A high-level overview of the Go-Explore algorithm.**

Go-Explore: A New Approach for Hard-Exploration Problems

The **Explore** operation

- **The second condition** - newly-encountered trajectory is “better” than that old to a cell already in the archive if it provides
  - Higher score than the old one
  - Shorter trajectory with the same score

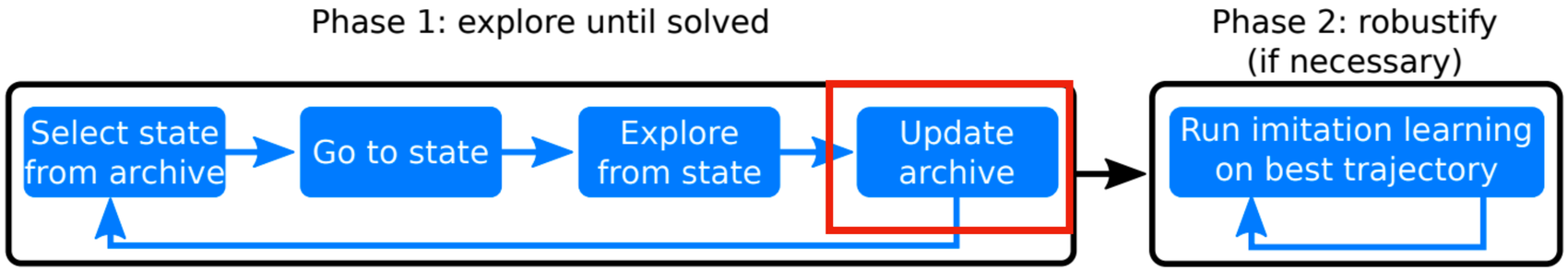


Figure 2: **A high-level overview of the Go-Explore algorithm.**

Go-Explore: A New Approach for Hard-Exploration Problems

The objective of phase 2

- **Robustify** — Use the trajectories collected from Phase 1
  - Improve the robustness of the agent
- Learning via imitation learning — use the better trajectory experience to update the policy
  - Stochasticity is added during phase 2
- Take this policy as the final policy

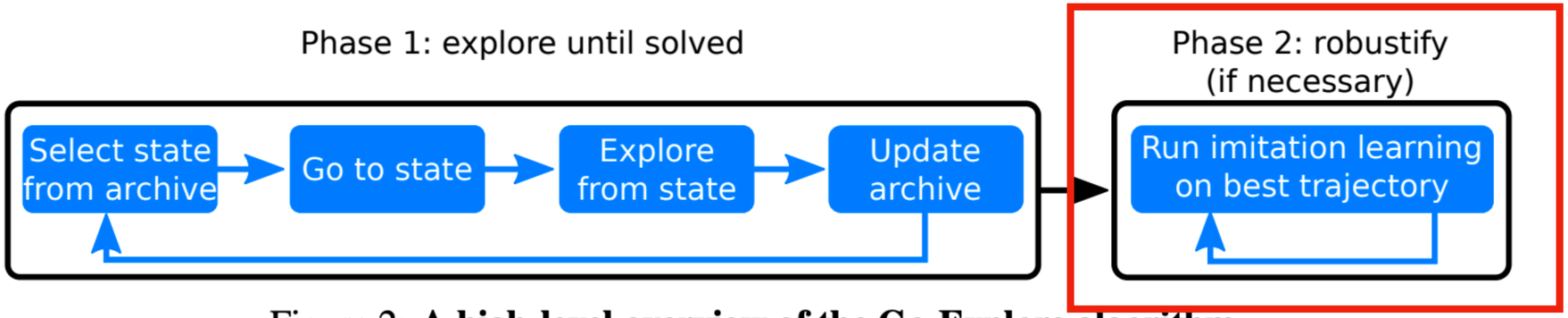
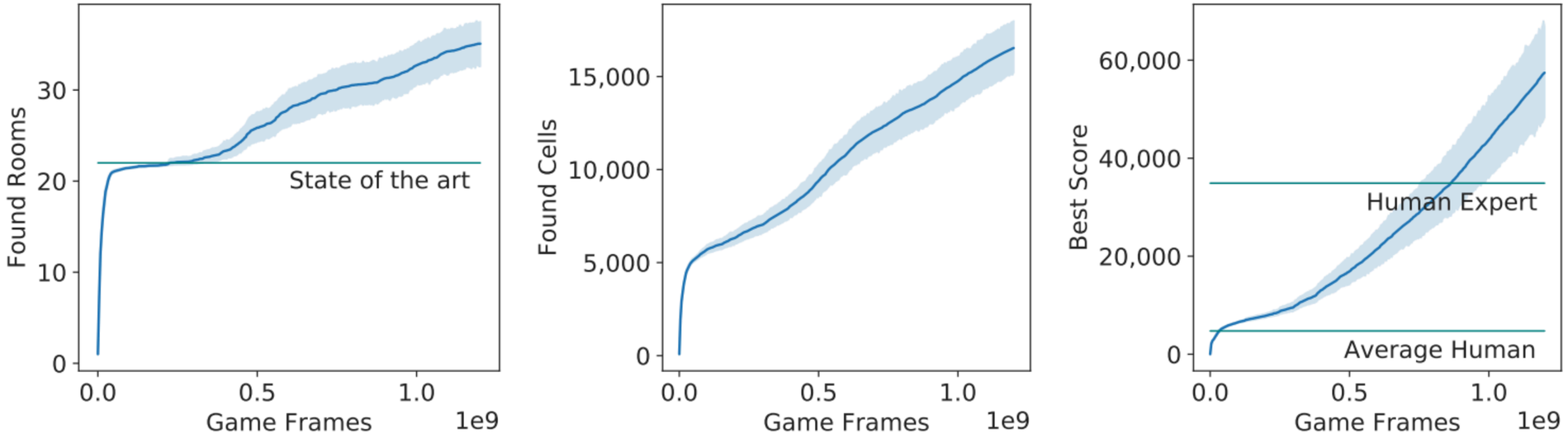


Figure 2: **A high-level overview of the Go-Explore algorithm.**

Go-Explore: A New Approach for Hard-Exploration Problems

Experimental results

- **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).

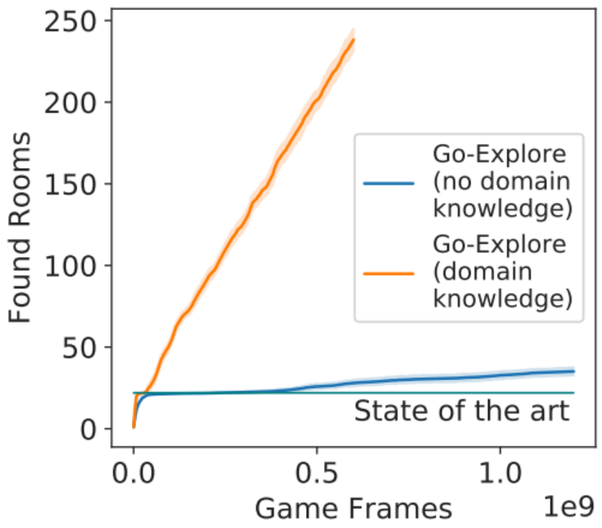


(a) Number of rooms found (b) Number of cells found (c) Maximum score in archive

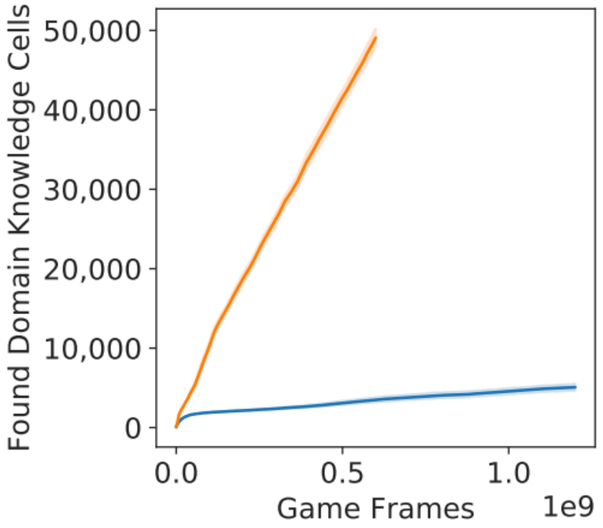
Go-Explore: A New Approach for Hard-Exploration Problems

Results — add domain knowledge

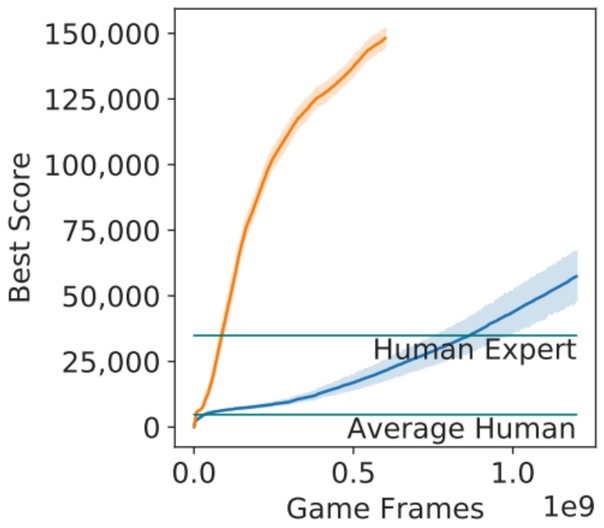
1. **What is domain knowledge** — Besides original cell, also add information such as character's location (e.g., x and y position), level number, etc.



(a) Number of rooms found



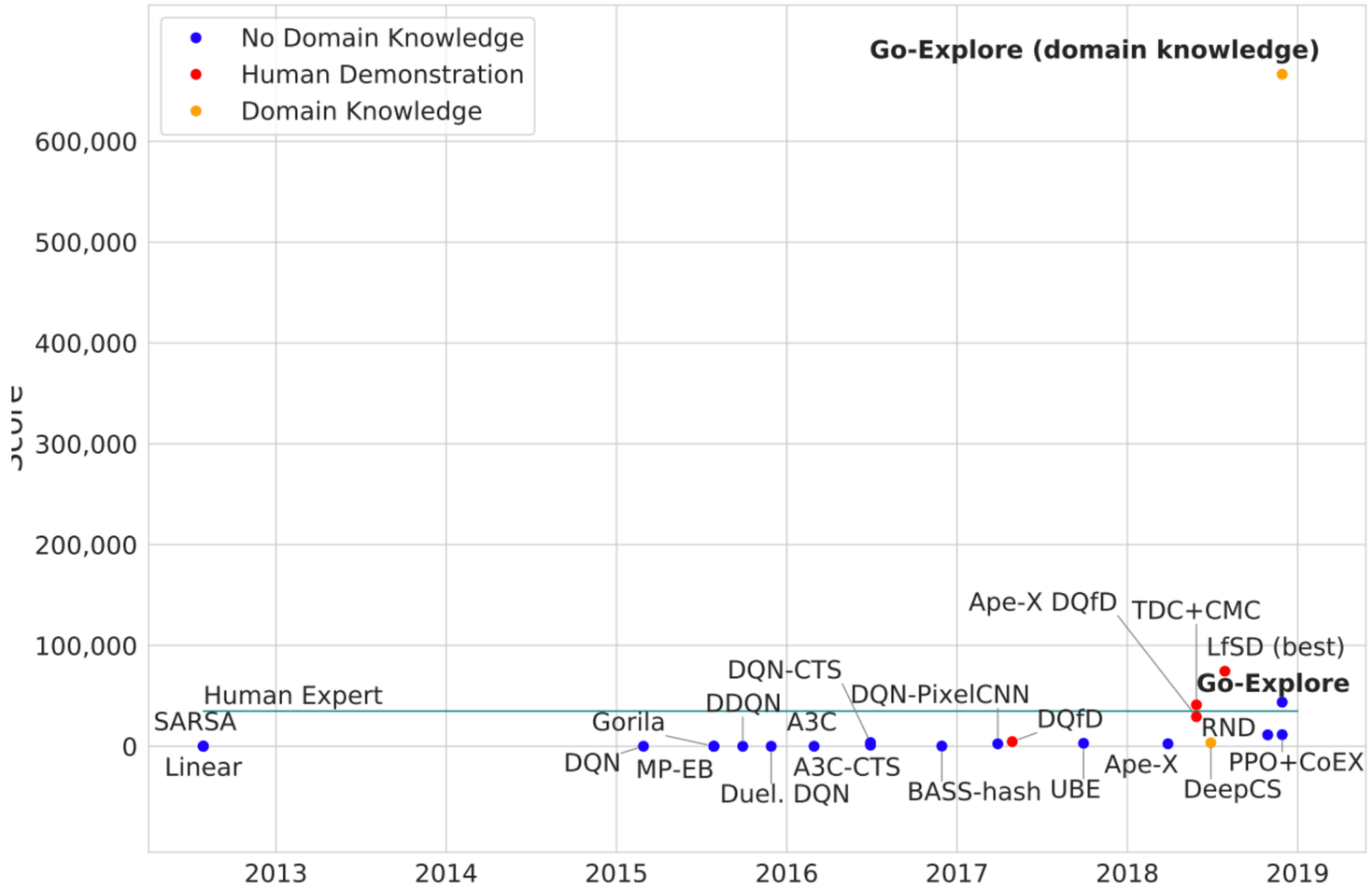
(b) Number of cells found



(c) Maximum score in archive

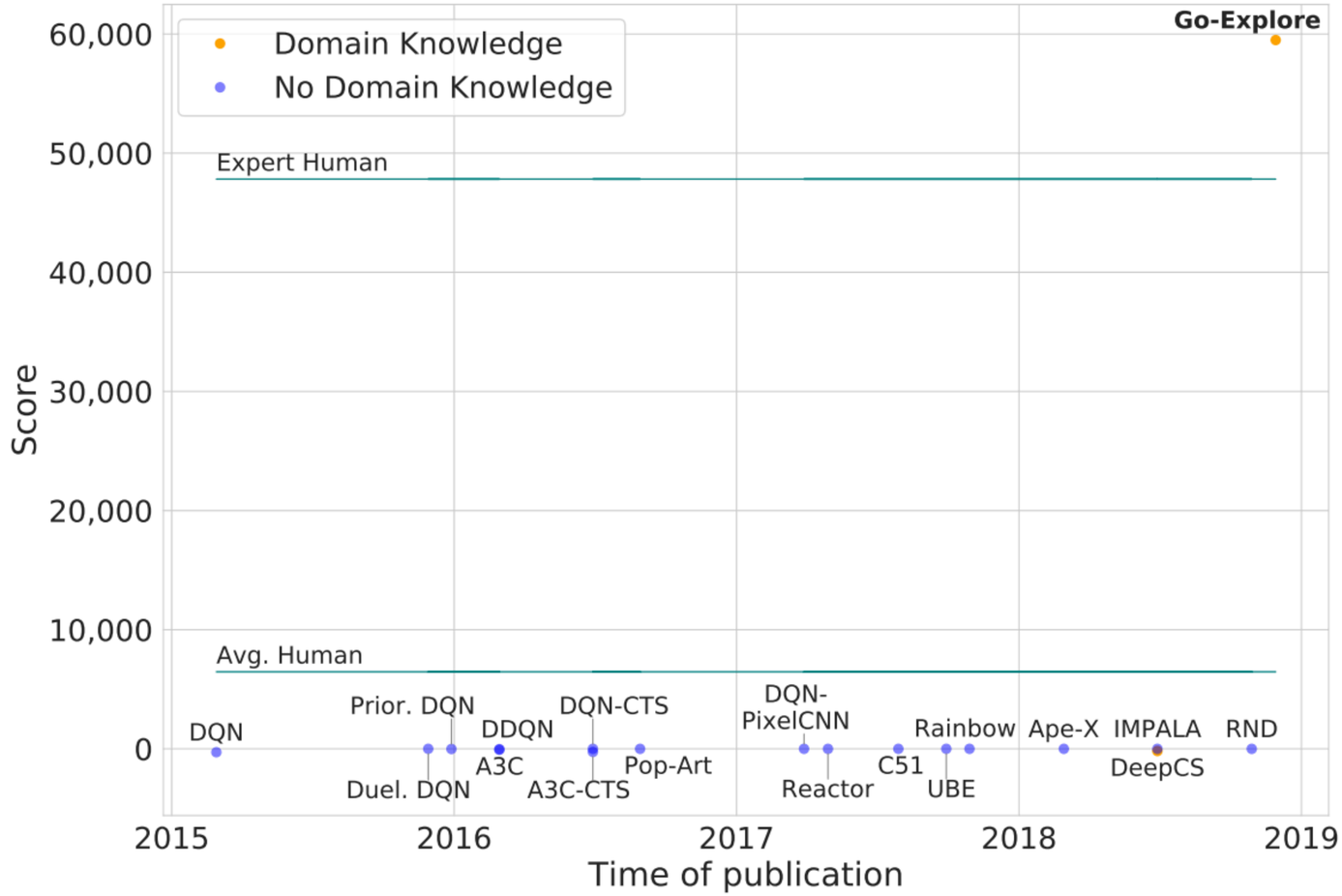
Go-Explore: A New Approach for Hard-Exploration Problems

Results — Montezuma's Revenge



Go-Explore: A New Approach for Hard-Exploration Problems

Experimental results —Pitfall!



ありがとうございます

ありがとうございます