Deep Reinforcement Learning

Lecture 10 —Exploration (I)

國立清華大學
NATIONAL TSING HUA UNIVERSITY

National Tsing Hua University
Department of Computer Science

Prof. Chun-Yi Lee

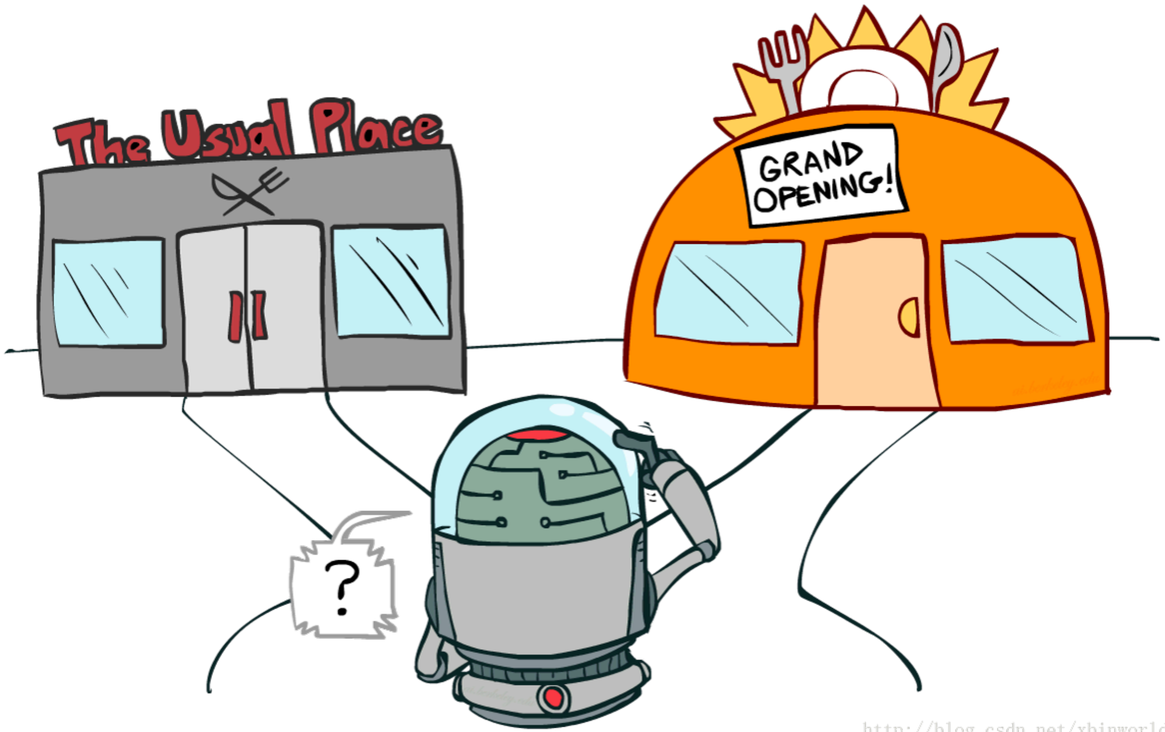
Outline

- **Exploration and Exploitation**
- **Exploration Strategies**

Exploration and Exploitation

Why exploration is necessary

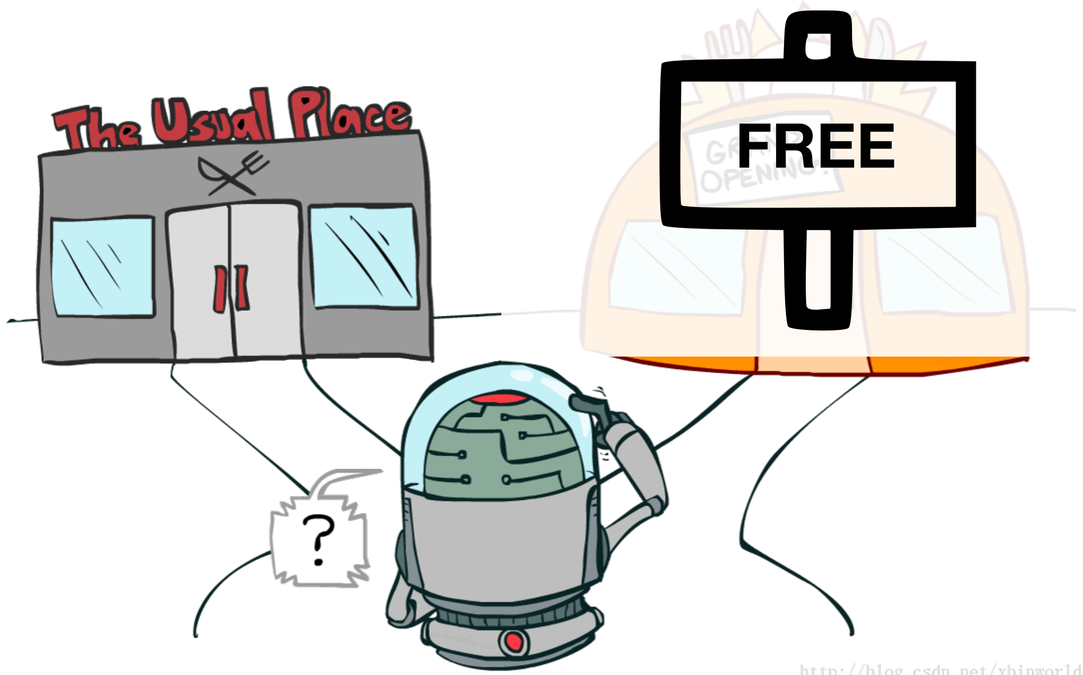
- Choose the usual one or try the new one ?



Exploration and Exploitation

Why exploration is necessary

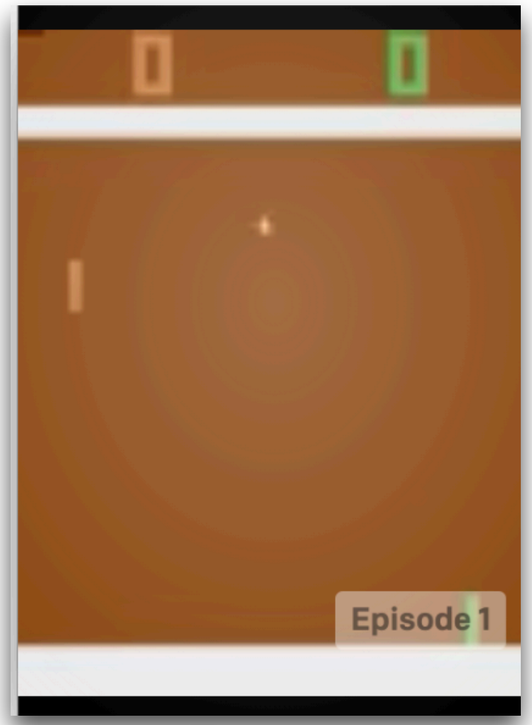
- If an agent always tries the usual one, it may potentially lose the chances to receive high rewards



Exploration and Exploitation

An example of easy environments: Pong

- In the Pong environment:



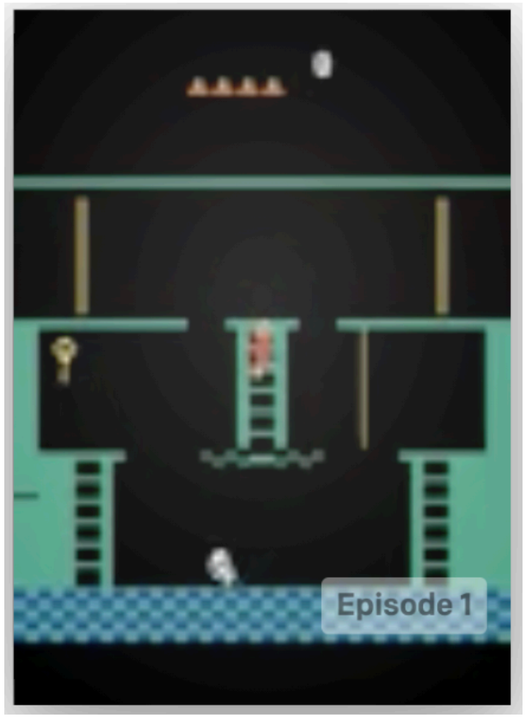
Pong

- The objective of Pong is easy
  - The agent receives rewards if it wins
  - The agent loses rewards if it fails
- The observation space is simple

Exploration and Exploitation

An example of spare-reward environments: Montezuma's Revenge

- In the Montezuma's Revenge environment:



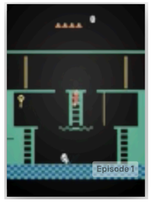
MontezumaRevenge

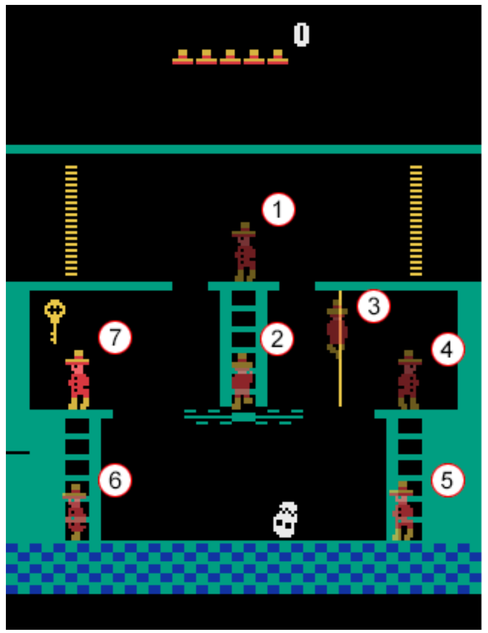
- The agent controls a character to move from room to room in a labyrinthine
- The objective of this environment is complex
  - Gather jewels
  - Kill enemies on the way
  - Collect and use equipments
  - Collect keys and open doors
  - Avoid pits, traps, obstacles, etc.
- The environment is complicated
- For most policies, the agent may be easily killed, making it to prefer to stay in the same location

Exploration and Exploitation

An example of spare-reward environments: Montezuma's Revenge

- The environment is nine floors deep
- 99 rooms to explore in total

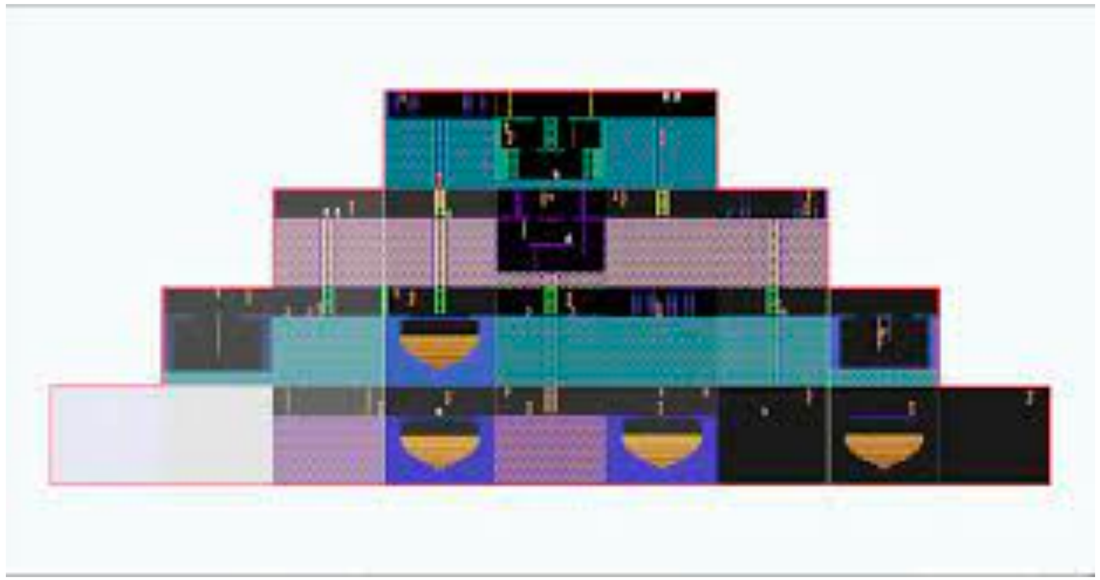




Instructions to RL Agent

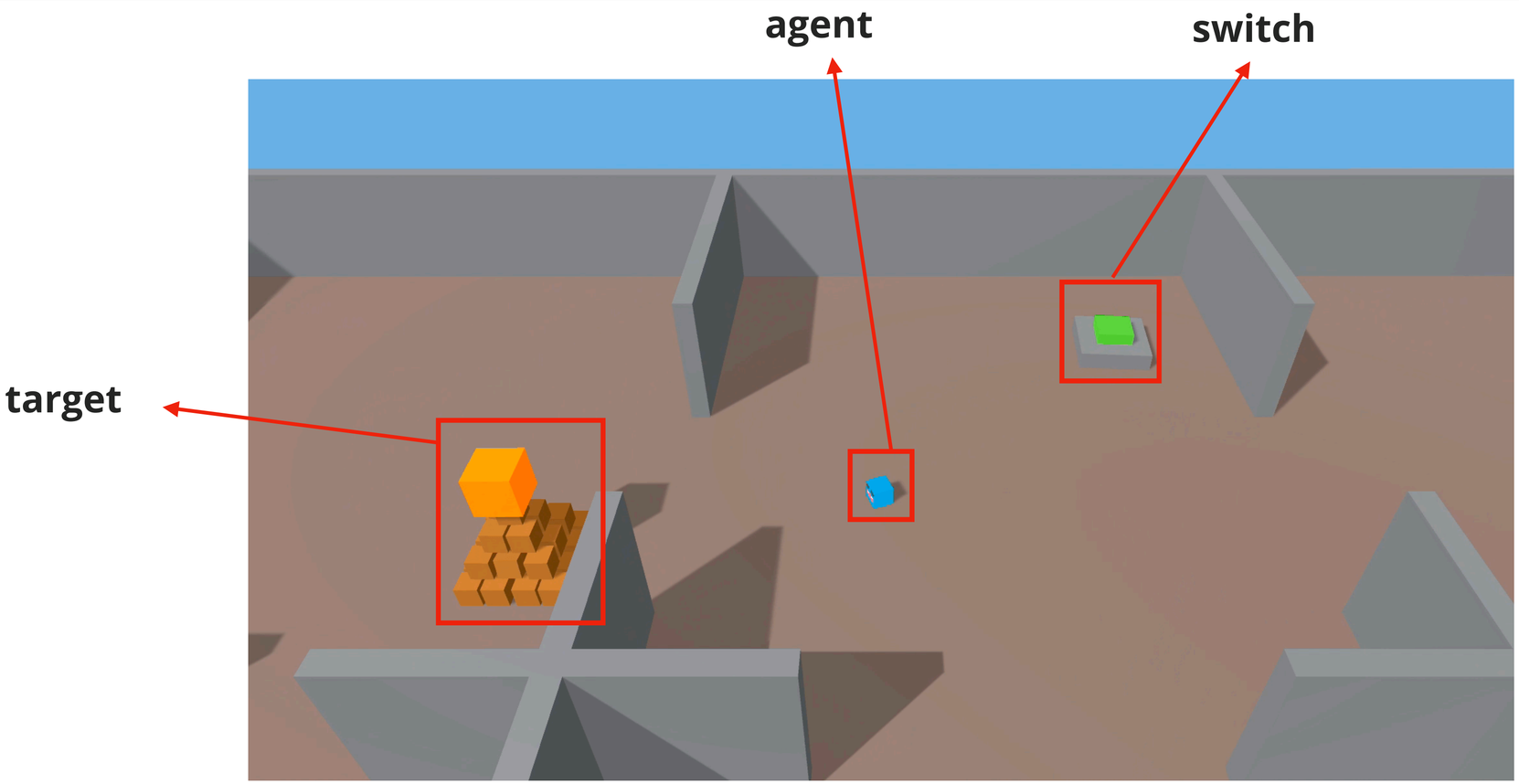
1: “Climb down the ladder”
2: “Jump to the rope”
3: “Go to the right side of the room”
4: “Climb down the ladder”
5: “Go to the left side of the room”
6: “Climb up the ladder”
7: “Get the key”

Step #: Current instruction



Exploration and Exploitation

Another example of a sparse-reward environment

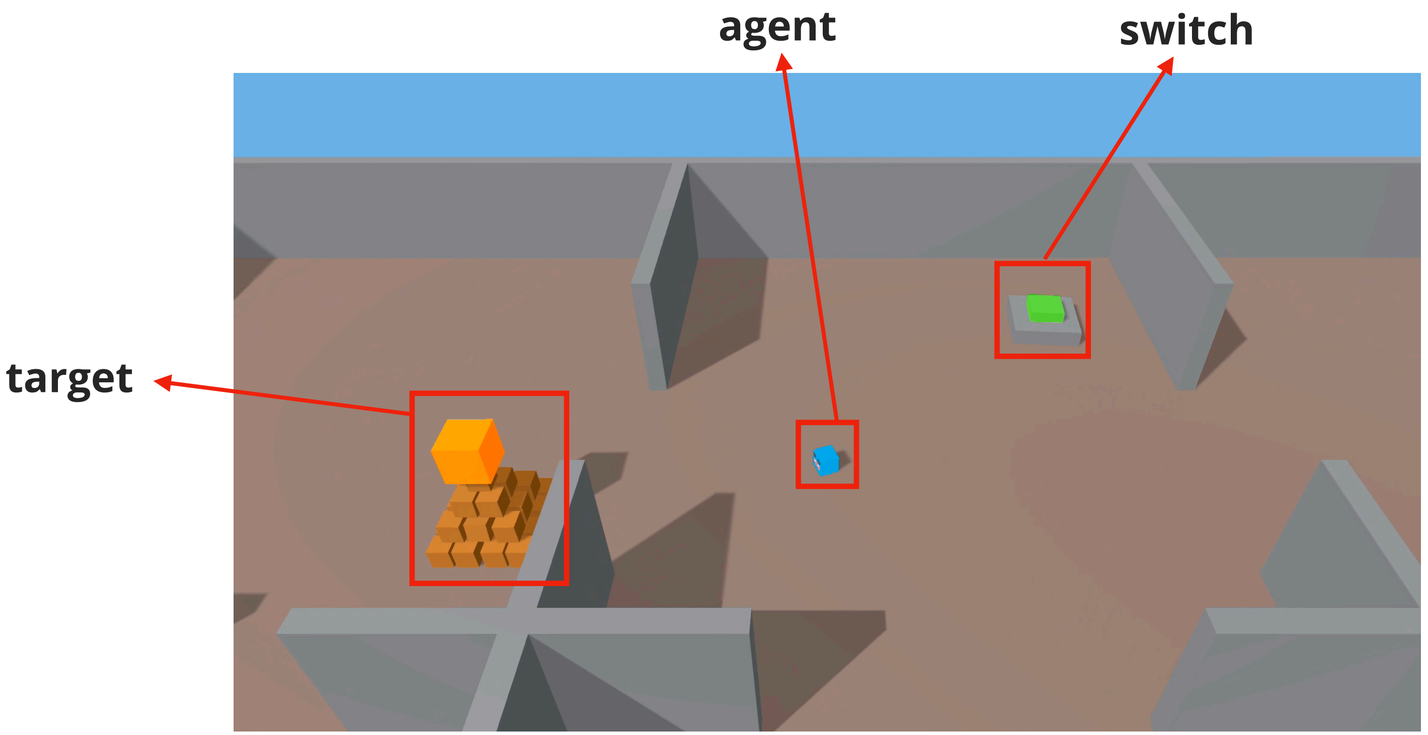


An illustration of sparse-reward environment

Demo link: https://blogs.unity3d.com/pt/2018/06/26/solving-sparse-reward-tasks-with-curiosity/

Exploration and Exploitation

Another example of a sparse-reward environment



An illustration of sparse-reward environment

The goal of this game is to

1. Interact with the switch
2. Find the target and collide with it
3. Touch the gold brick

No intermediate reward for

Moving to new room

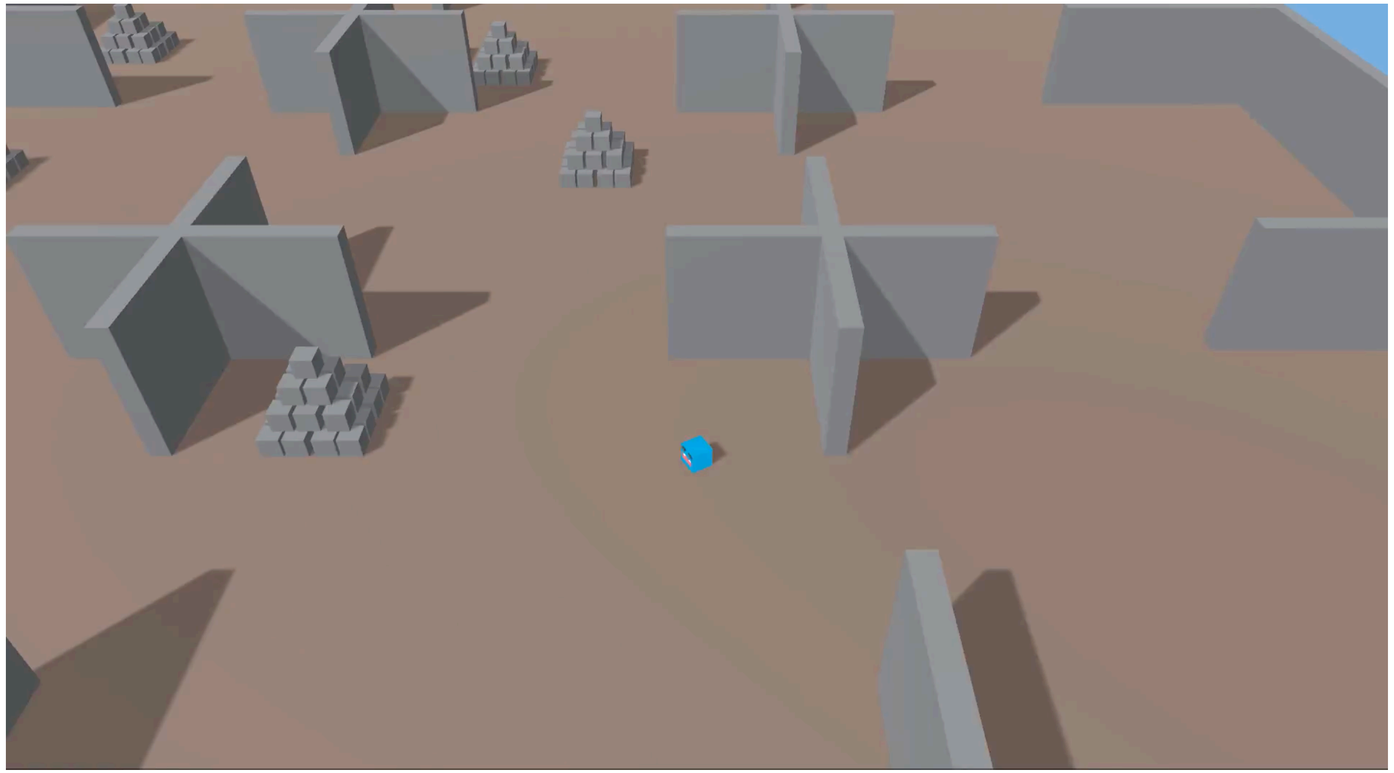
Flipping the switch

Knocking over the tower

Exploration and Exploitation

Another example of a sparse-reward environment

- Simply using PPO is not sufficient to train an effective agent in this environment

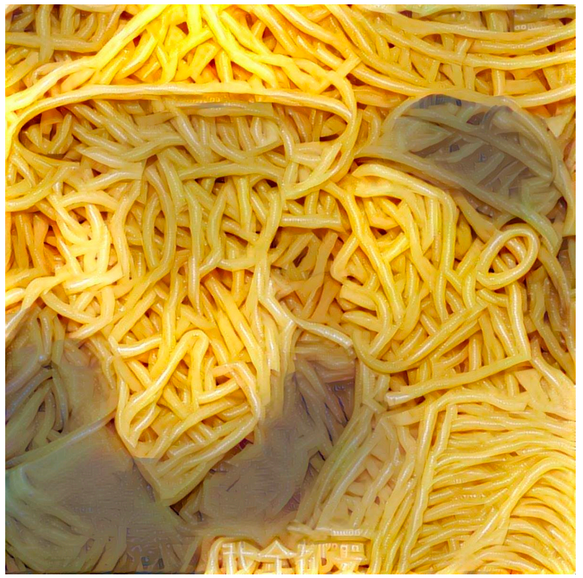


Agent trained with Proximal Policy Optimization (PPO) only

Exploration and Exploitation

Which one is important, exploration or exploitation?

- For a good reinforcement learning agent, both exploitation and exploration are important



Exploration and Exploitation

The importance of exploitation

- Exploitation
  - Leverage the current knowledge to Improve the agent itself
  - The current knowledge can come from the data samples collected by the agent
    - Either on-policy or off-policy
    - More data efficient
  - Can be completed by the approaches introduced in the previous lectures (value-based, policy-based, actor-critic)

Exploration and Exploitation

The importance of exploration

- Exploration
  - Obtain more knowledge, in the hope to get more rewards
    - Encourages the agent to attempt different behaviors
    - Motivate the agent to go to different states in an environment
- Diffidulties
  - How can an agent discover high-reward strategies that require a temporally extended sequence of complex behaviors that, individually, are not rewarding?
  - How can an agent decide whether to attempt new behaviors (to discover ones with higher reward) or continue to do the best thing it knows so far?

Outline

- Exploration and Exploitation
- **Exploration Strategies**

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)**

Optimistic Exploration

Concepts

- **Assumption**
  - New states = Good states
- **Methodology**
  - Add bonus to rewards
  - Requires estimating state visitation frequencies or novelty
  - Typically realized by means of exploration bonuses

Optimistic Exploration

The upper confidence bound (UCB) exploration

- UCB :

$$A_t = \arg\max_a [Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}]$$

- c : A confidence value that controls the level of exploration
- $N(a)$: the number of times that action '***a***' has been selected, prior to time '***t***'
- Usually used in Monte Carlo Tree Search (MCTS)
- However, the UCB algorithm focuses on a current state
- It is also limited to simpler, tabular based tasks

Optimistic Exploration

The upper confidence bound (UCB) exploration

- UCB :

$$A_t = \arg\max_a [Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}]$$

- Exploitation part: $Q_t(a)$
  - **Optimism in the fact of uncertainty**
  - If you don’t know which action is best then choose the one that currently looks to be the best
  - the action that currently has the highest estimated reward will be the chosen action

Optimistic Exploration

The upper confidence bound (UCB) exploration

- Expand the idea : add a state bonus into rewards

$$r_{new} = r + B(N(s))$$

- $B(N(s))$ defined in different types
  - UCB $\sqrt{\frac{\log n}{N(s)}}$
  - MBIE-EB (Strehl & Littman, 2008) $\sqrt{\frac{1}{N(s)}}$
  - BEB (Kolter & Ng, 2009) $\frac{1}{N(s)}$

Optimistic Exploration

The upper confidence bound (UCB) exploration

- UCB :

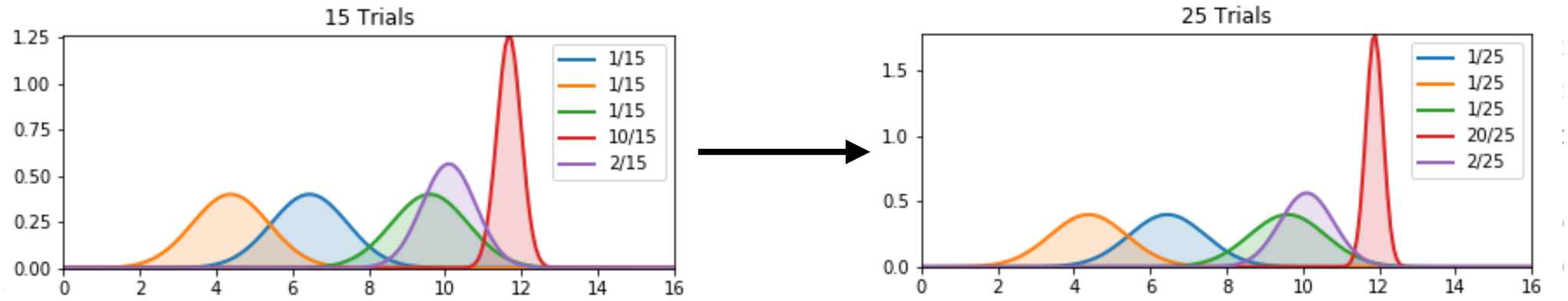
$$A_t = \arg\max_a [Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}]$$

- Exploration part: $c\sqrt{\frac{\log t}{N_t(a)}}$
  - The degree of exploration is controlled by the hyper-parameter ‘c’
  - $N_t(a)$ is small if an action hasn’t been tried very often

Posterior Sampling

Exploration

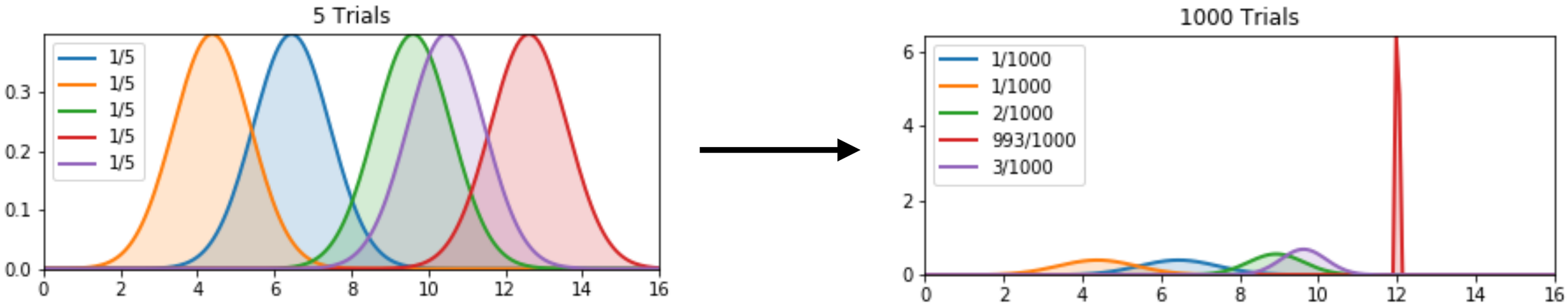
- After each sample, it will modify its strategy



Posterior Sampling

Exploration

- If it use uniform sampling, it has 1/5 chance to take each action.
- After each sampling, it will know that the red one always get higher rewards
- Thus, it will modify its strategy to take that more frequently



Posterior Sampling

Exploration

- Thompson sampling
  - Sample a distribution
  - Take this distribution to select action
  - Update the distribution

Algorithm 4.2

Thompson$(\mathcal{X}, p, q, r)$

for $t = 1, 2, \ldots$ **do**
  #sample model:
  Sample $\hat{\theta} \sim p$

  #select and apply action:
  $x_t \leftarrow \text{argmax}_{x \in \mathcal{X}} \mathbb{E}_{q_{\hat{\theta}}}[r(y_t)|x_t = x]$
  Apply x_t and observe y_t

  #update distribution:
  $p \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot | x_t, y_t)$
end for

Posterior Sampling

Exploration

- Thompson Sampling
  - Take each action reward as Beta distribution
  - When take action and receive reward, use this information to update the distribution.

$$(\alpha_k, \beta_k) \leftarrow \begin{cases} (\alpha_k, \beta_k) & \text{if } x_t \neq k \\ (\alpha_k, \beta_k) + (r_t, 1 - r_t) & \text{if } x_t = k. \end{cases}$$

Exploration Strategies

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

ϵ-Greedy Exploration

Sometimes random

- Choose the greedy action a^* with probability $1 - \epsilon$
- Choose an random action a with probability ϵ

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon & \text{if } a^* = \text{greedy}(Q(s, a)) \\ \epsilon & \text{otherwise} \end{cases}$$

ϵ-Greedy Exploration

Policy Improvement

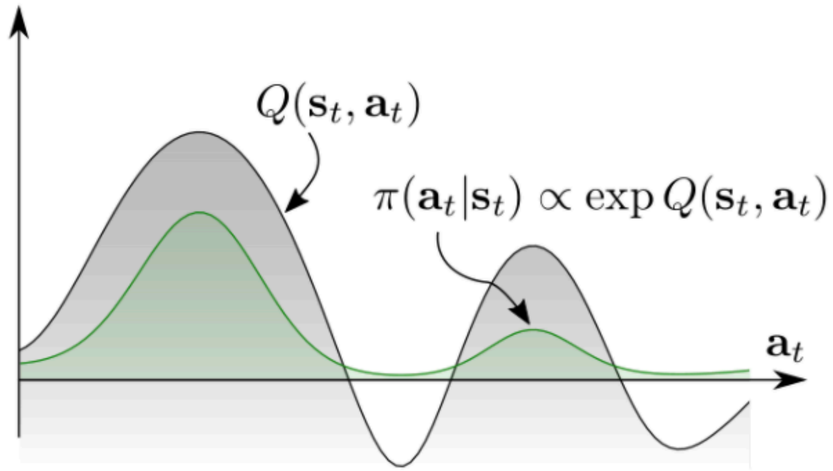
- For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to Q_π will lead to an improvement, $V_{\pi'}(s) \geq V_\pi(s)$

$$
\begin{aligned}
V_{\pi'}(s) &= \sum_{a \in A} \pi'(a \mid s) Q_\pi(s, a) \\
&= \frac{\epsilon}{m} \sum_{a \in A} Q_\pi(s, a) + (1 - \epsilon) max_{a \in A} Q_\pi(s, a) \text{ , m is the number of actions} \\
&\geq \frac{\epsilon}{m} \sum_{a \in A} Q_\pi(s, a) + (1 - \epsilon) \sum_{a \in A} \frac{\pi(a \mid s) - \frac{\epsilon}{m}}{1 - \epsilon} Q_\pi(s, a) \quad \text{(LINK)} \\
&= \sum_{a \in A} \pi(a \mid s) Q_\pi(s, a) = V_\pi(s)
\end{aligned}
$$

Boltzmann Exploration

Recap: energy-based policy

- An energy-based policy assumes that the policy $\pi(a_t \mid s_t) \propto \exp(Q(s_t, a_t))$
- The policy can easily learn multimodal behavior



Boltzmann Exploration

Policy Improvement

- The agent draws actions from a Boltzmann distribution (softmax) over the learned Q values (energy-based policy)
  - The equation can be represented as:

$$\text{The probability of action } a_1 \text{ at state } s = \frac{exp(Q(s, a_1)/kT)}{\sum_{a'} exp(Q(s, a')/kT)}$$

  - k is a constant, while T is the temperature term
  - This formulation enables different actions to be selected based on their learned Q values

Entropy Term

A widely used technique adopted by DRL methods

- The entropy term is an effective technique used by a lot of DRL methods to encourage exploration, expressed as:

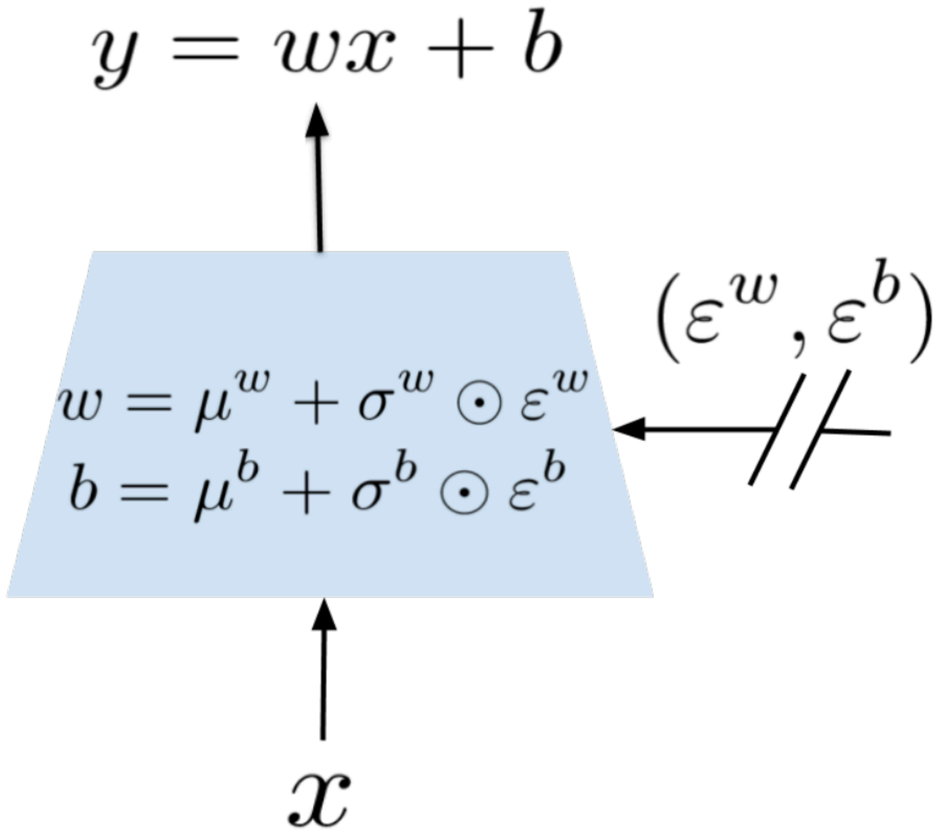
$$H(\pi(a \mid s))$$

  - H is the entropy function
- The entropy term H can be used in different ways:
  - Loss function (e.g., A3C, TRPO, and PPO)
  - Reward function (e.g., SAC)

Noise-Based Exploration

Noisy nets: An exploration technique

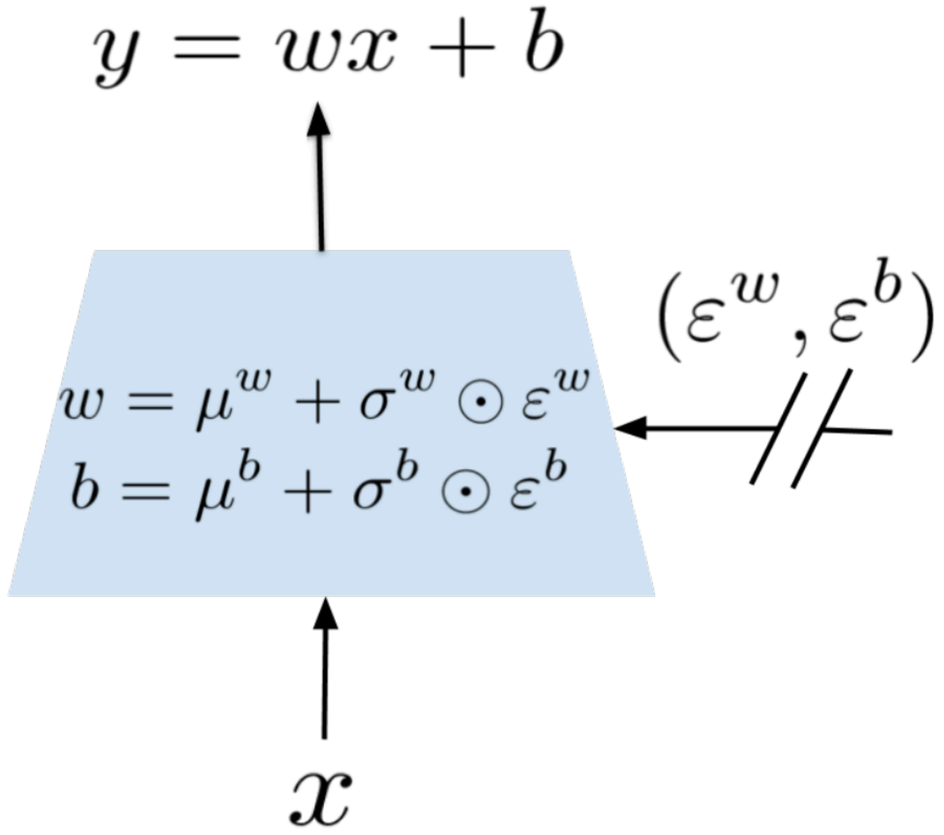
- **Noisy nets**
  - Add a noise layer to introduce stochastic behaviors for the policy
    - ϵ^w, ϵ^b are random variable
    - $\mu^w, \mu^b, \sigma^w, \sigma^b$ are training Variables



Noise-Based Exploration

Noisy nets: An exploration technique

- **Noisy nets**
  - Does not require the ϵ-greedy exploration strategy
  - Can be applied to other DRL algorithms



Noise-Based Exploration

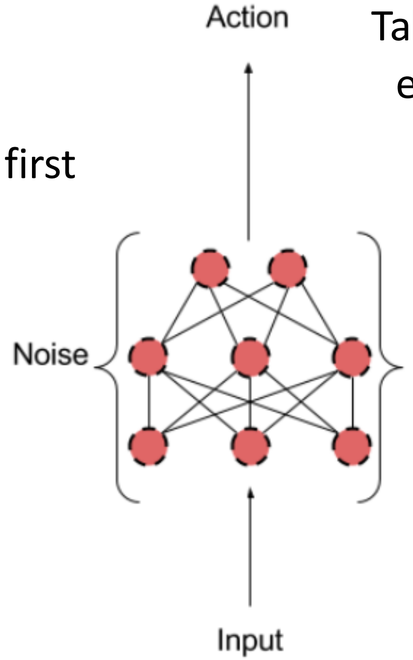
Parameter space exploration

- Similar to the concept of noisy nets, but does not learn the mean and variance
- Adaptive approach: It samples random Gaussian noise values at the beginning of each episode, and scales it according to the variation between the actions with and without noise
- The variance is updated according to the following:

$$\sigma_{k+1} = \begin{cases} \alpha\sigma_k & \text{if } d(\pi, \widetilde{\pi}) \leq \delta, \\ \frac{1}{\alpha}\sigma_k & \text{otherwise,} \end{cases}$$

Injected at the first timestep

Injected at the beginning of an episode and fixed during that episode



Take an action at each timestep

Compute action with the perturbed network, during that episode

$$a_t = \pi_{\widetilde{\theta^\pi}}(s_t)$$

Information Gain

Information-theoretic exploration

- Information Gain : $D_{KL}(P \mid Q)$

- In MDP, Gain what information?
  - —Gain about reward r ?
  - —Gain about state density $p(s)$?
  - —Gain about dynamics transition $p(s' \mid s, a)$

Information Gain

Information-theoretic exploration

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

- Information Gain : $\log p'_\theta(s) - \log p_\theta(s)$

- If density changed a lot, the state was novel

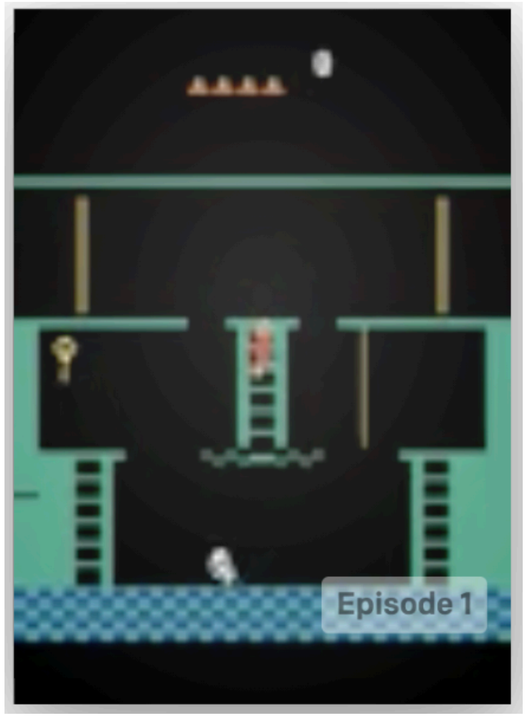
Exploration Strategies

- **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)**

Optimistic Exploration

Pseudo-counts exploration

- However, how do we count the image-type states?
- Difficulties:
  - Too many pixels to count for tabular settings
  - Require hand-crafted features for each game
  - May need to hire human beings to count (X)



Optimistic Exploration

Pseudo-counts exploration

- Pseudo-counts : Some states are more similar than others
- Find a density model $p_\theta(s)$.
  - If s is similar to an old state, $p_\theta(s)$ would output a higher value
- Take a small MDP for example:

$$p(s) = \frac{N(s)}{n}$$

  - $N(s)$ is the number of visits to state s
  - n is total count, $p(s)$ is density function

Optimistic Exploration

Pseudo-counts exploration

- In the real density function, if we see one more state s, density function can be updated as:

$$p_{new}(s) = \frac{N(s) + 1}{n + 1}$$

- This is real density function and count, can be replaced by the pseudo-ones in the formula

Optimistic Exploration

Pseudo-counts exploration

- A pseudo-count function $\hat{N}(s)$ and a pseudo-count total $\hat{n}$, are adopted to replace the corresponding counterparts
- This allows us to derive the pseudo density function as follows:

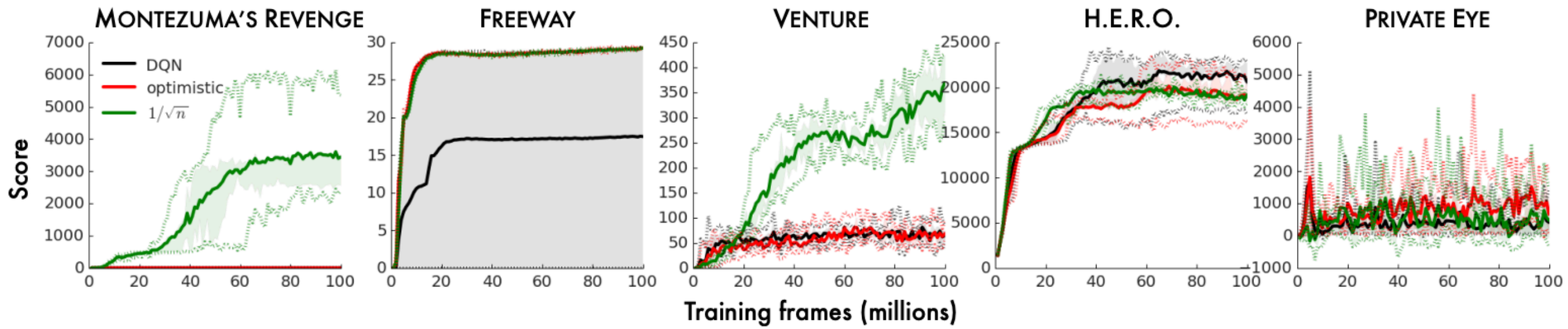
$$p_\theta(s) = \frac{\hat{N}(s)}{\hat{n}} \qquad p'_\theta(s) = \frac{\hat{N}(s)+1}{\hat{n}+1}$$

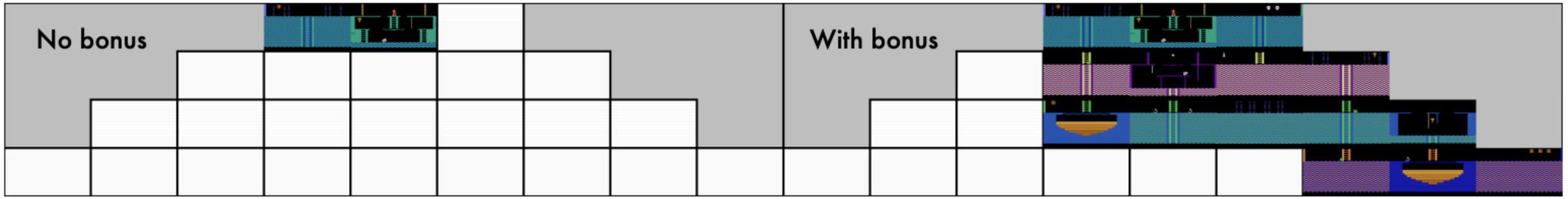
- If we can find a method to estimate $p_\theta(s)$ (Dirichlet estimator), we can know use them to estimate the pseudo-counts

Optimistic Exploration

Pseudo-counts exploration

- Unifying count-based exploration and intrinsic motivation





Optimistic Exploration

PixelCNN

- However, $p_\theta(s)$ is not easy to estimate
- Researchers came up with a new CNN approach to estimate it
  - Count-Based Exploration with Neural Density Models
  - The method is based on PixelCNN
- PixelCNN provides an advanced neural density model for images
- As a result, it is able to provide a more accurate pseudo density model $p_\theta(s)$ and pseudo-count $\hat{N}(s)$

Optimistic Exploration

PixelCNN

- PixelCNN — A variant of Gated PixelCNN (van den Oord, A note on the evaluation of generative models.)
- It is a method to calculate the maximum likelihood
- PixelCNN rapidly learns a sensible distribution over state space.

Optimistic Exploration

PixelCNN

- For hard exploration games, both outperform the baseline DQN
- For easy ones, the reward bonuses do not provide large improvements and may have a negative effect by skewing the reward landscape

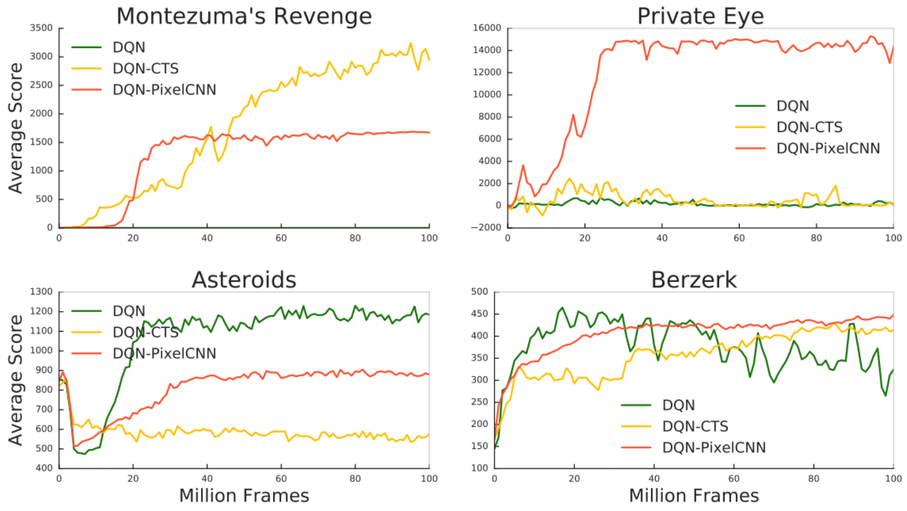
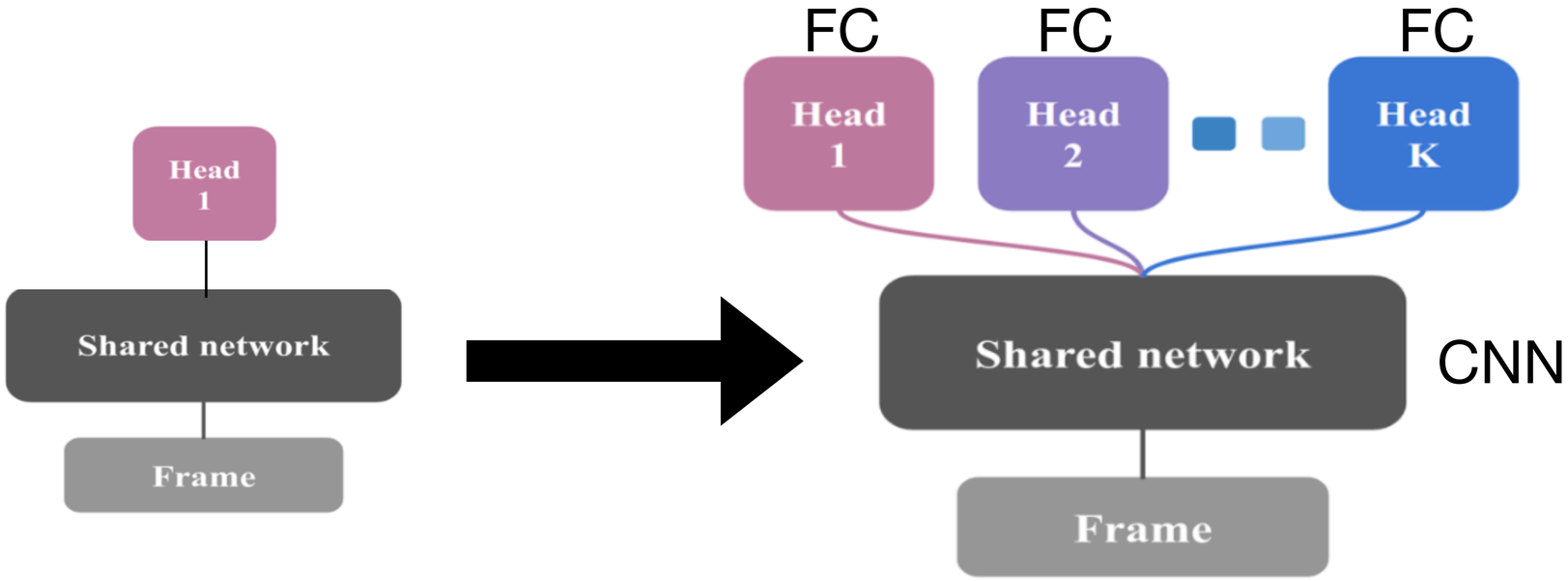


Figure 5. DQN, DQN-CTS and DQN-PixelCNN on hard exploration games (**top**) and easier ones (**bottom**)

Posterior Sampling

Bootstrapped DQN

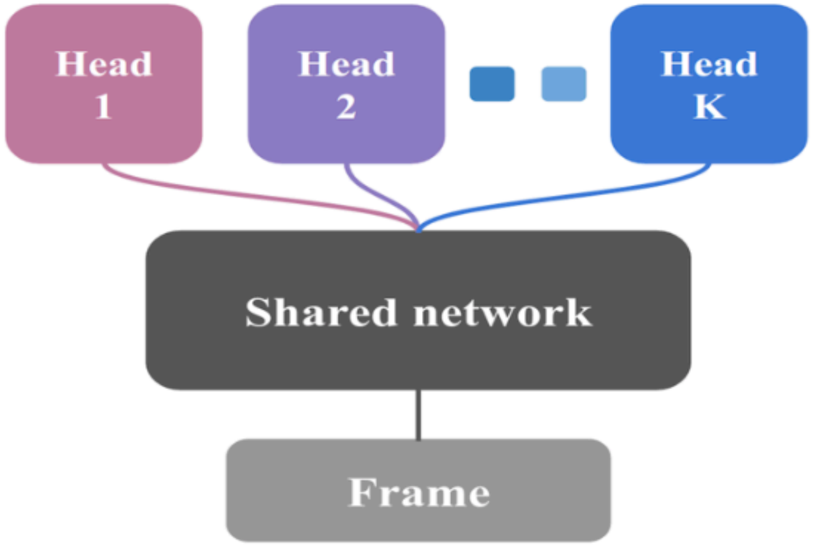
- Bootstrapped DQN



Posterior Sampling

Bootstrapped DQN

- K bootstrapped Q-value function heads
- Each episode re-samples a Q-value function head to use
- Collect data from different bootstrapped head makes agent explore well



Posterior Sampling

Bootstrapped DQN

Algorithm 4.2

Thompson$(\mathcal{X}, p, q, r)$

for $t = 1, 2, \ldots$ **do**
  #sample model:
  Sample $\hat{\theta} \sim p$

  #select and apply action:
  $x_t \leftarrow \text{argmax}_{x \in \mathcal{X}} \mathbb{E}_{q_{\hat{\theta}}}[r(y_t)|x_t = x]$
  Apply x_t and observe y_t

  #update distribution:
  $p \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot | x_t, y_t)$
end for

Algorithm 1 Bootstrapped DQN

Input: Value function networks Q with K outputs $\{Q_k\}_{k=1}^K$. Masking distribution M.
Let B be a replay buffer storing experience for training.
for each episode **do**
  Obtain initial state from environment s_0
  Pick a value function to act using $k \sim \text{Uniform}\{1, \ldots, K\}$
  for step $t = 1, \ldots$ until end of episode **do**
    Pick an action according to $a_t \in \arg\max_a Q_k(s_t, a)$
    Receive state s_{t+1} and reward r_t from environment, having taking action a_t
    Sample bootstrap mask $m_t \sim M$
    Add $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$ to replay buffer B
  end for
end for

Posterior Sampling

Bootstrapped DQN

- Deep Exploration —Chain environment

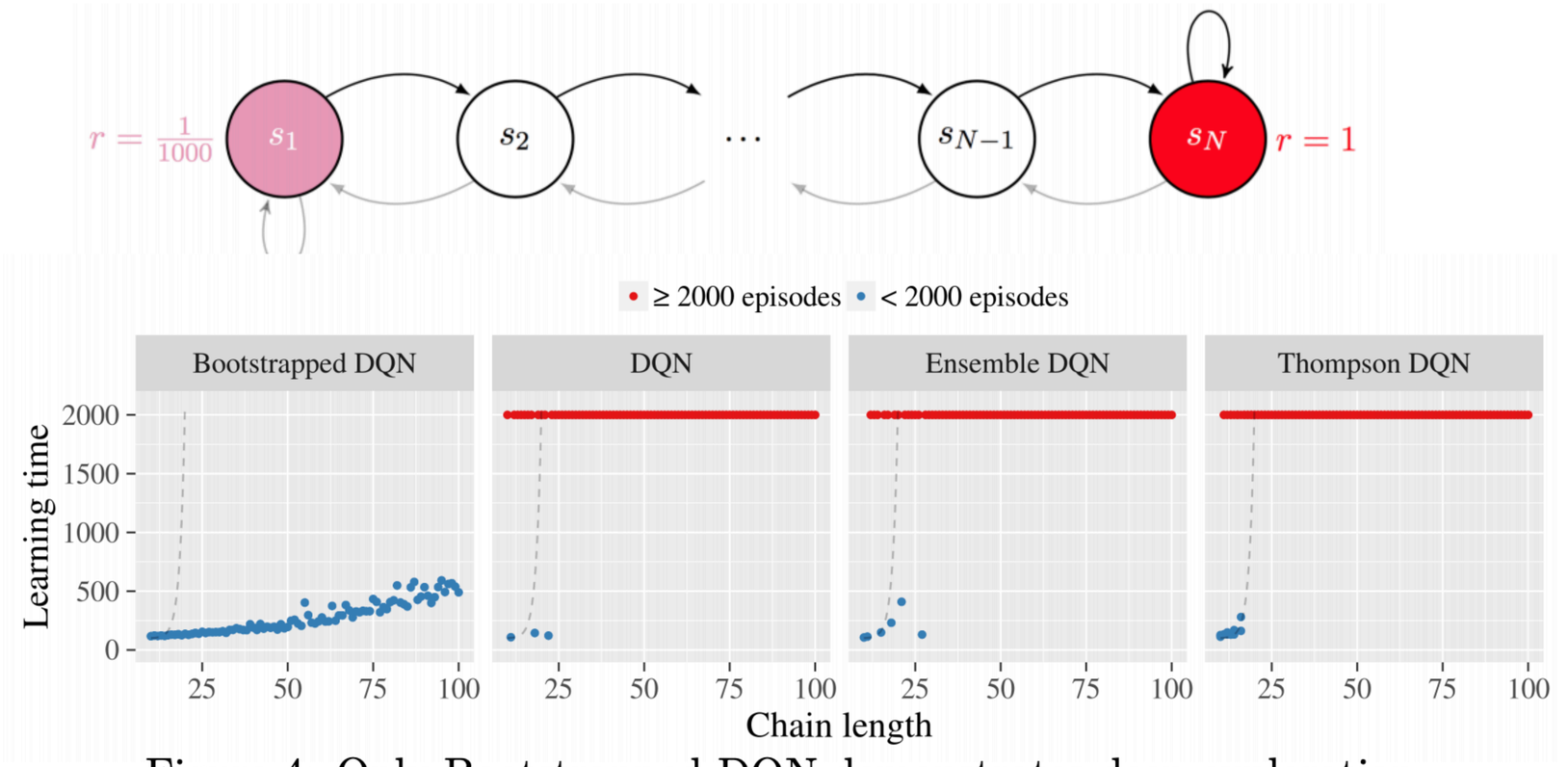


Figure 4: Only Bootstrapped DQN demonstrates deep exploration.

Posterior Sampling

Bootstrapped DQN — Experimental results

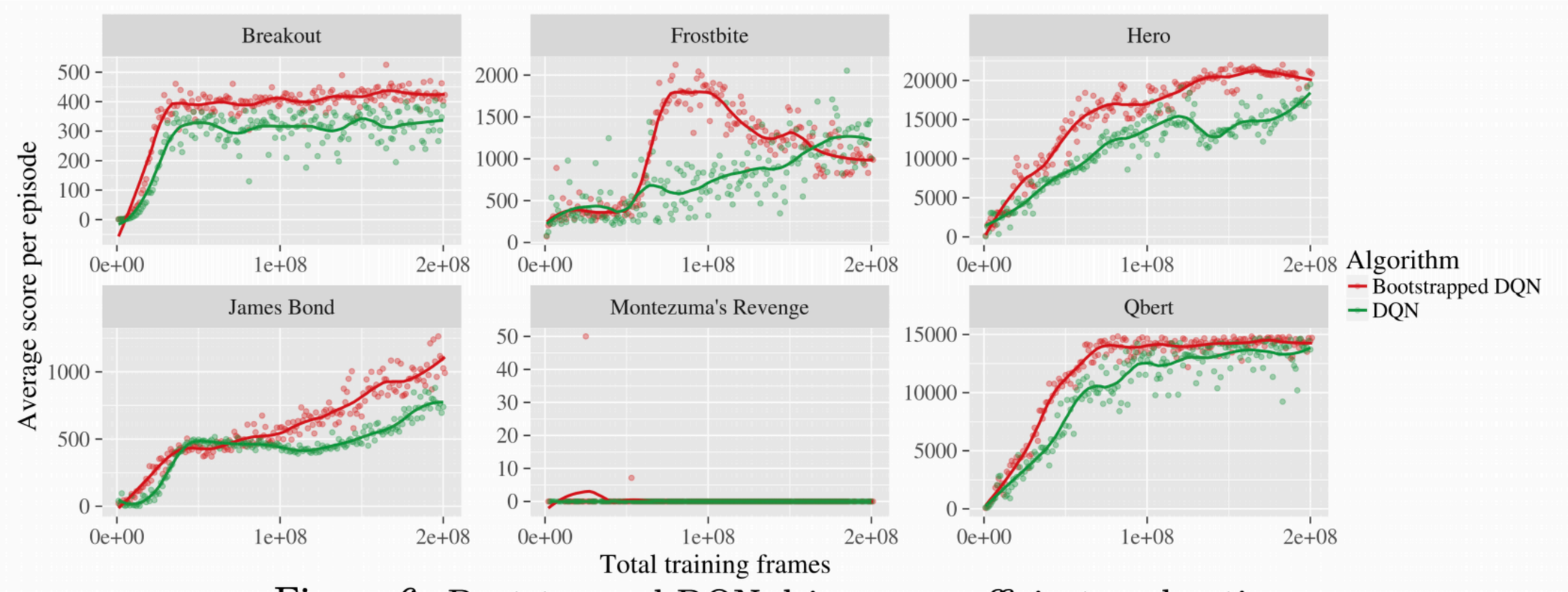


Figure 6: Bootstrapped DQN drives more efficient exploration.

Intrinsic Curiosity Module (ICM)

Learning to play with no rewards - motivation

- Endow the agent with a sense of curiosity and to **reward it based on how surprised it is by the environment** around it.
- The policy sub-system is trained to maximize the sum of there two rewards:

$$R_t = R_t^i + R_t^e$$

Intrinsic Curiosity Module (ICM)

Prediction error provides a measure of curiosity

We can divide all sources that can modify agent's observations into three cases:

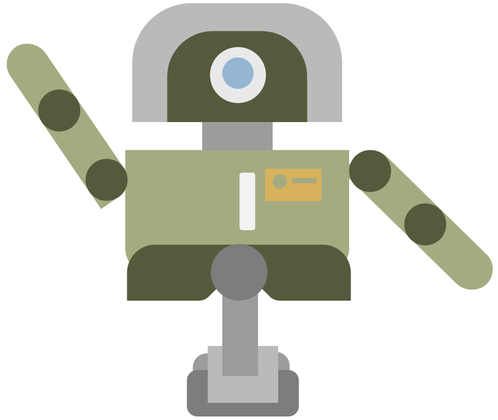
(1) Things that can be controlled by the agent

(2) Things that the agent cannot control but can affect the agent (e.g., vehicle driven by another agent)

(3) Things out of the agent's control and not affecting the agent (e.g., moving leaves)

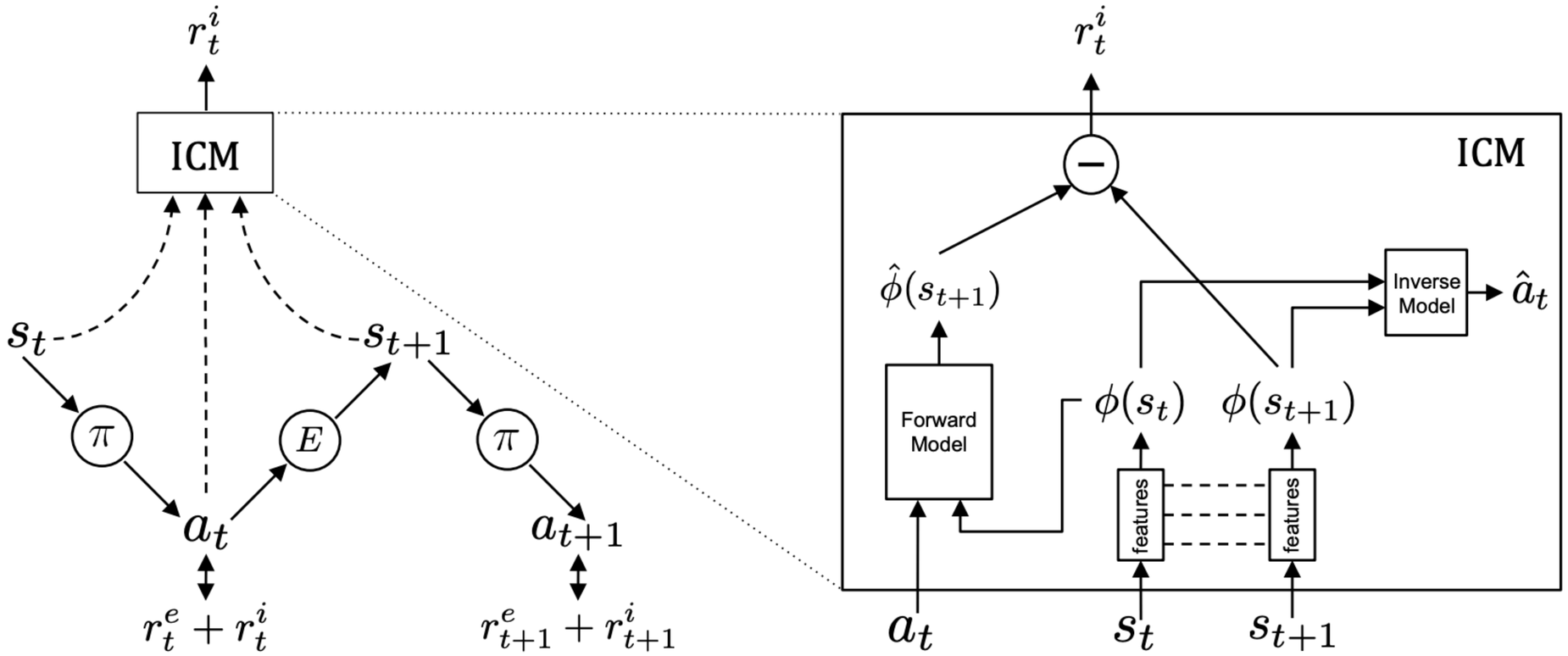
Intrinsic Curiosity Module (ICM)

Prediction error provides a measure of curiosity



Intrinsic Curiosity Module (ICM)

Prediction error as the curiosity reward



Intrinsic Curiosity Module (ICM)

Inverse dynamics module

- Take feature encoding $(\phi(s_t),\ \phi(s_{t+1}))$ as inputs and predict the action a_t taken by the agent to move from s_t to s_{t+1}

- Formulated as: $\hat{a}_t = g(\phi(s_t), \phi(s_{t+1}); \theta_I)$

- θ_I is trained to minimize: $\min_{\theta_I} L_1(\hat{a}_t, a_t)$

Intrinsic Curiosity Module (ICM)

Forward dynamics module

- Take feature encoding $(\phi(s_t),\ a_t)$ as inputs and predict the feature encoding $\phi(s_{t+1})$ of next state
- Formulated as: $\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F)$
- θ_I is trained to minimize:

$$L_F = \frac{1}{2} || \hat{\phi}(s_{t+1}) - \phi(S_{t+1}) ||_2^2$$

Intrinsic Curiosity Module (ICM)

Overall optimization objective

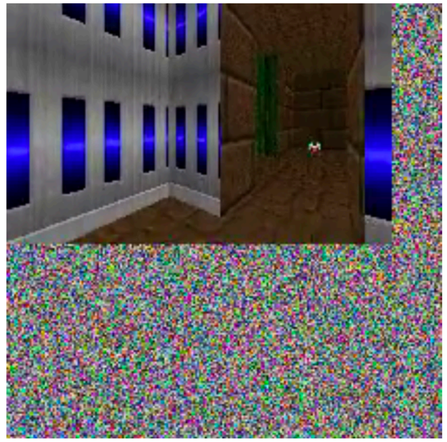
- The overall optimization problem that is solved for learning the agent is a composition of equations we mentioned above,

$$\min_{\theta_P,\theta_I,\theta_F} [-\lambda \mathbb{E}_{\pi(s_t;\theta_P)}[\sum_t (r_t)] + (1-\beta)L_I + \beta L_F]$$

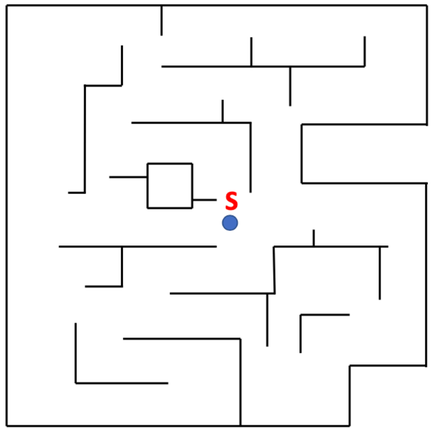
Intrinsic Curiosity Module (ICM)

Experimental setup

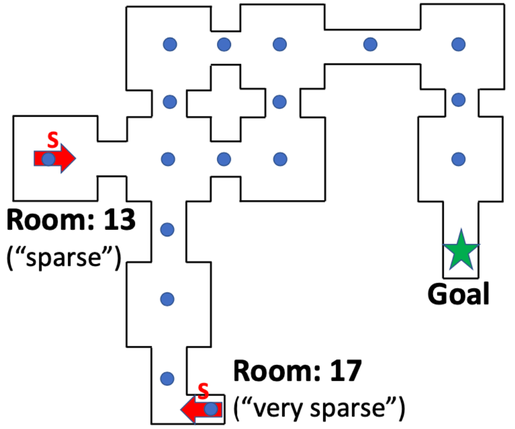
(a) Input snapshot in VizDoom



(b) Input w/ noise



(a) Train Map Scenario

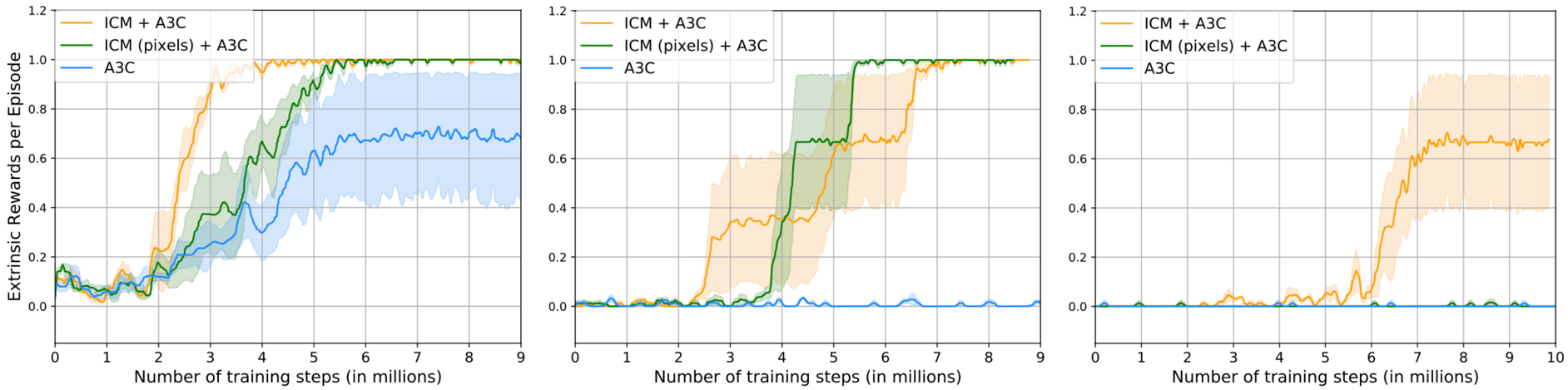


(b) Test Map Scenario

Intrinsic Curiosity Module (ICM)

Experimental results

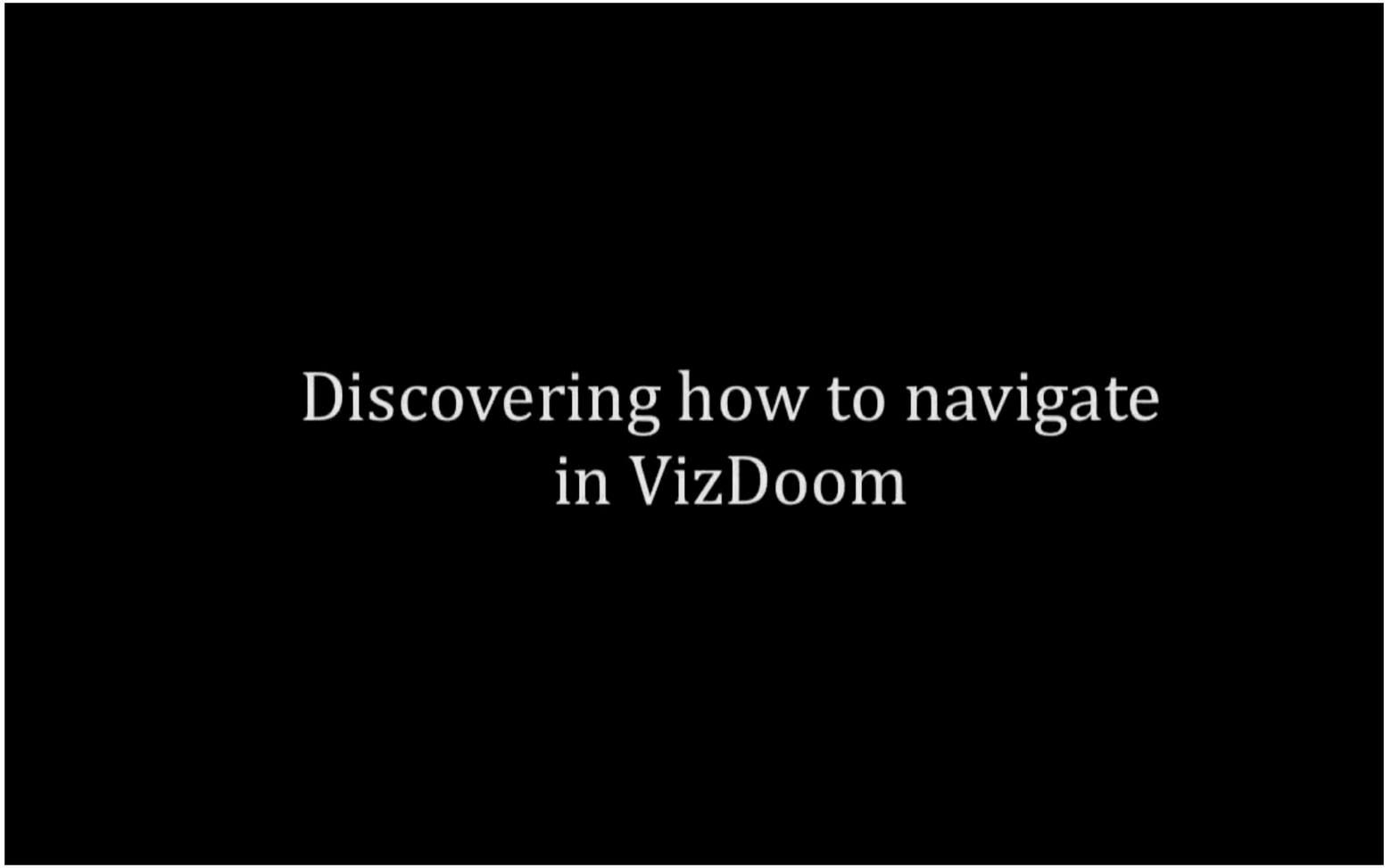
- Exploration becomes harder with larger distance between the initial and goal locations: “dense”, “sparse” and “very sparse”.



(a) “dense reward” setting (b) “sparse reward” setting (c) “very sparse reward” setting

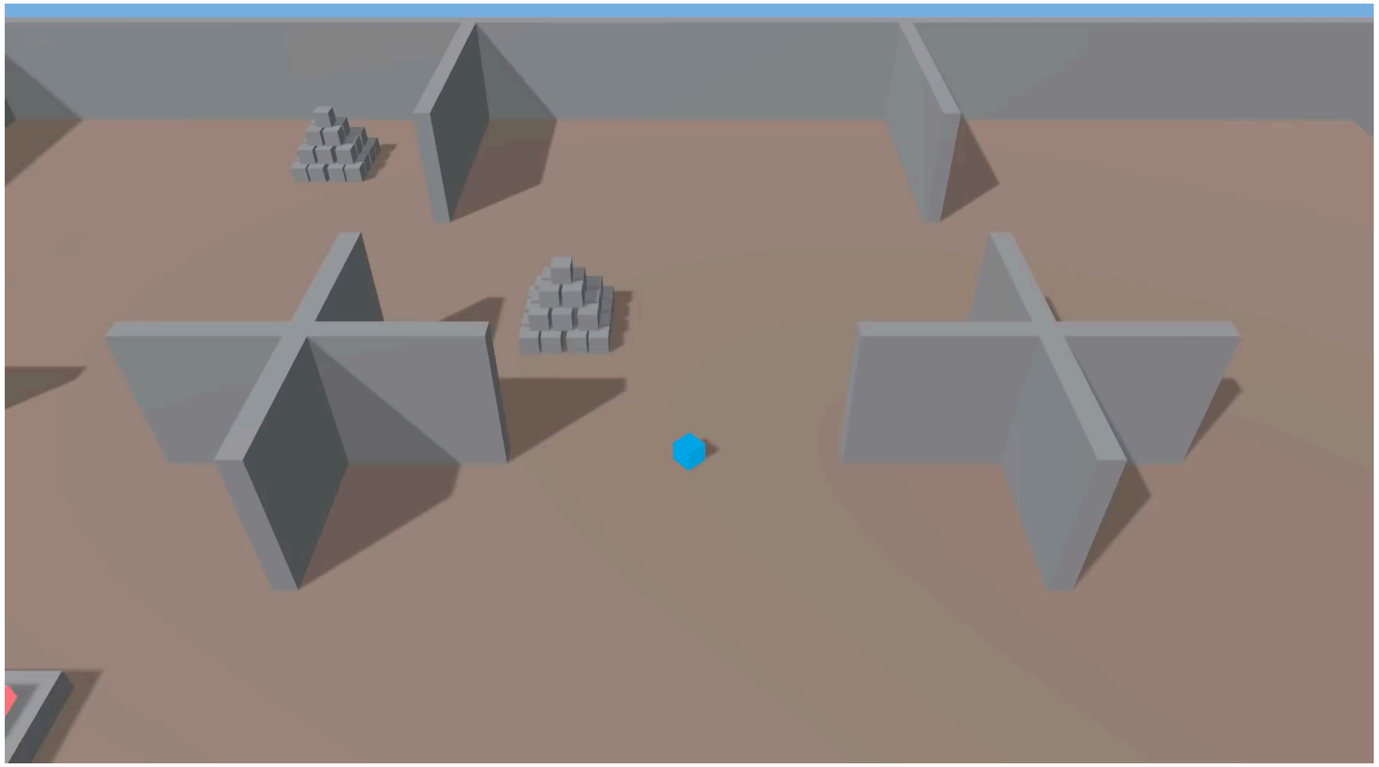
Intrinsic Curiosity Module (ICM)

Experimental setup



Intrinsic Curiosity Module (ICM)

Solve the sparse-reward environment with ICM



Agent trained with PPO + curiosity-driven intrinsic rewards.

ありがとうございます