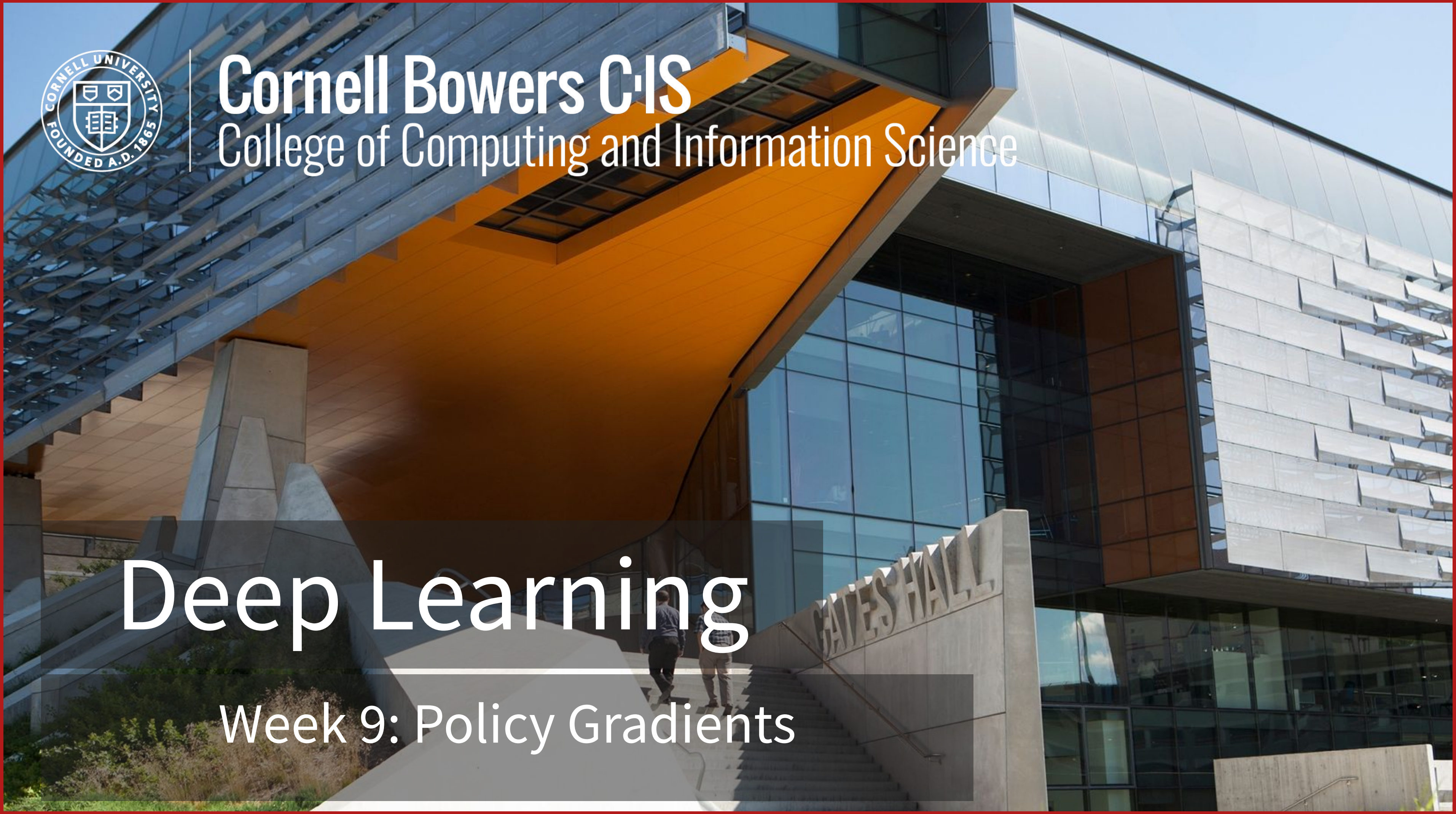


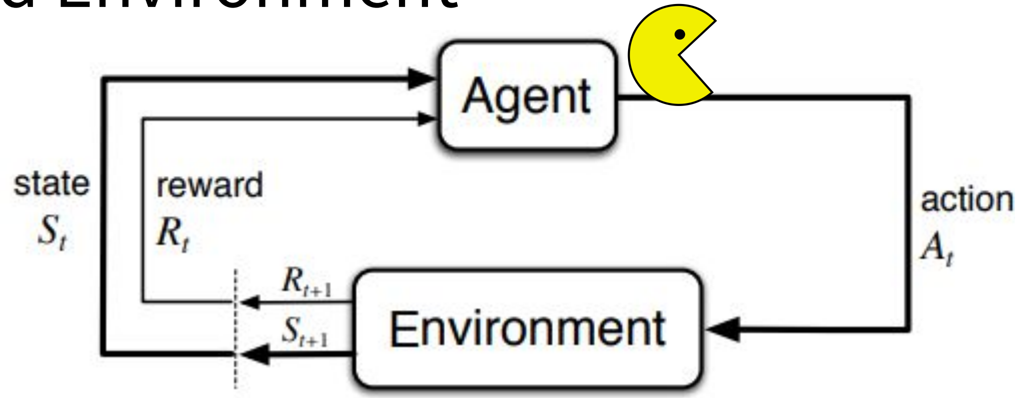
Cornell Bowers C·IS
College of Computing and Information Science

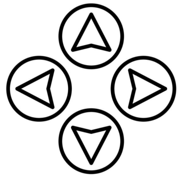
Deep Learning

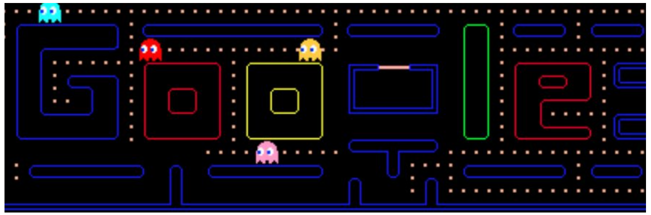
Week 9: Policy Gradients

Cornell Bowers C·IS

Agent and Environment







Agent:	Environment:
- Perceives environment. - Makes decisions. - Aims to maximize reward.	- The external context in which an agent operates and interacts with - Provides feedback to agent

Markov Decision Process (MDP)

- MDPs provide a framework for modeling sequential decision-making problems.
- An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$:
  - $\mathcal{S}$: Set of states representing the environment.
  - $\mathcal{A}$: Set of actions the agent can take.
  - $\mathcal{P}$: Transition probability function, $\mathcal{P}(s'|s, a)$.
  - $\mathcal{R}$: Reward function, $\mathcal{R}(s, a)$.
  - γ: Discount factor, $\gamma \in [0, 1]$.

Cornell Bowers C·IS

Q-Learning Update Rule

$$Q^*(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a')$$

- The Q-Learning update rule can be expanded as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

$$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \left[\mathcal{R}(s,a) + \gamma \max_{a'} Q(s',a') \right]$$

Cornell Bowers C·IS

Goal of Reinforcement Learning

Find the optimal policy π_θ^* that maximizes the expected discounted return $J(\theta)$:

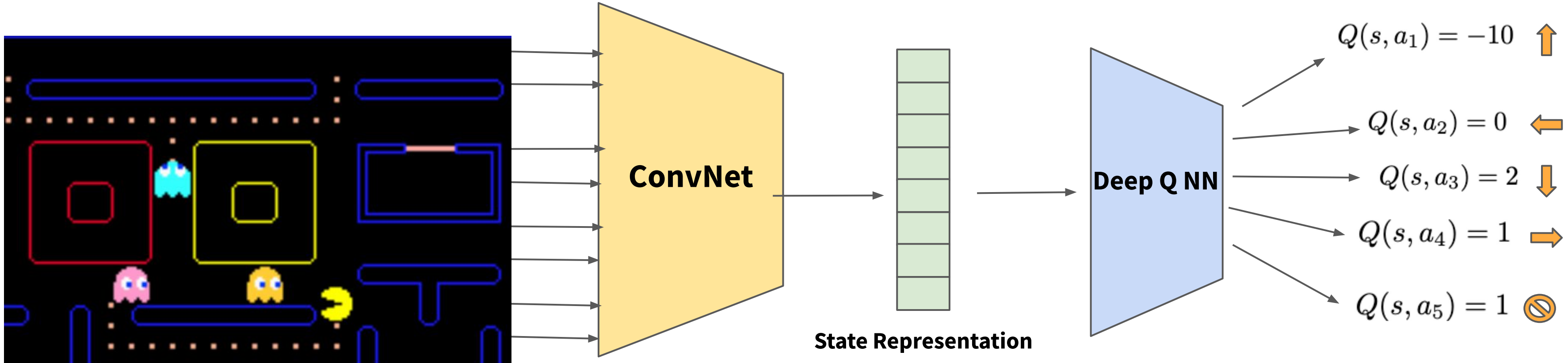
$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

$$\pi_\theta^* = \arg\max_\theta J(\theta)$$

where:

- $J(\theta)$ is the expected discounted return under the policy π_θ.
- τ represents a trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ sampled from the policy π_θ.
- $p_\theta(\tau)$ is the probability distribution over trajectories induced by the policy π_θ.

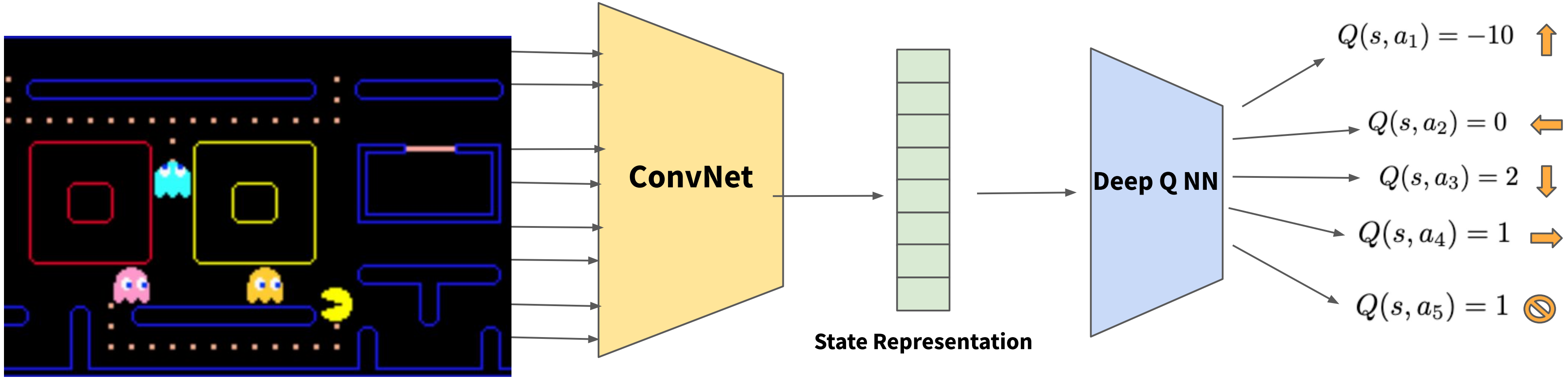
Deep Q-Learning:



$$\pi(s) = \underset{a}{argmax}(Q(s, a))$$

Cornell Bowers CIS

Deep Q-Learning:

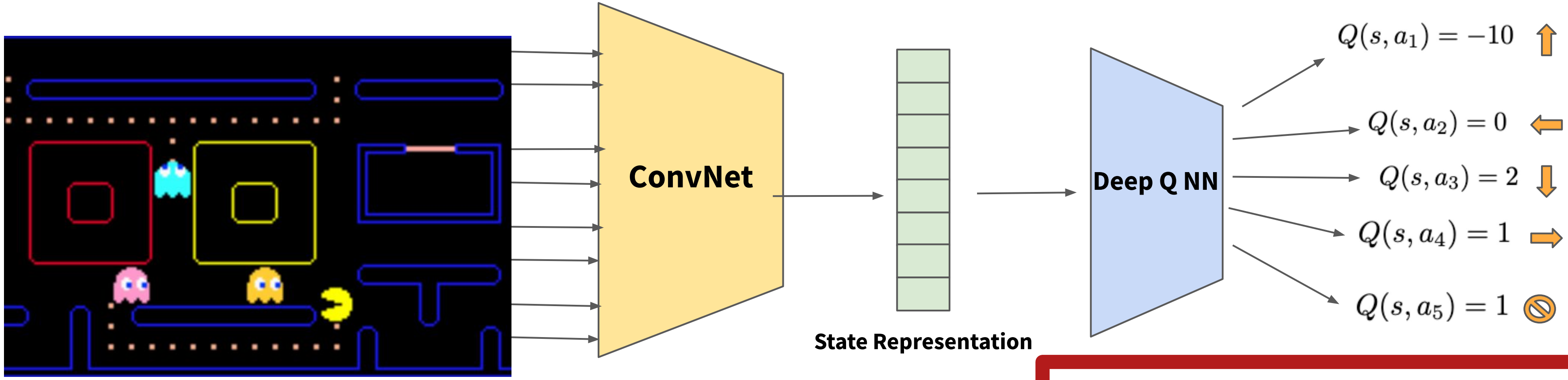


$$\pi(s) = \underset{a}{argmax}(Q(s,a))$$

This max function makes continuous and stochastic actions hard

Cornell Bowers C·IS

Deep Q-Learning:

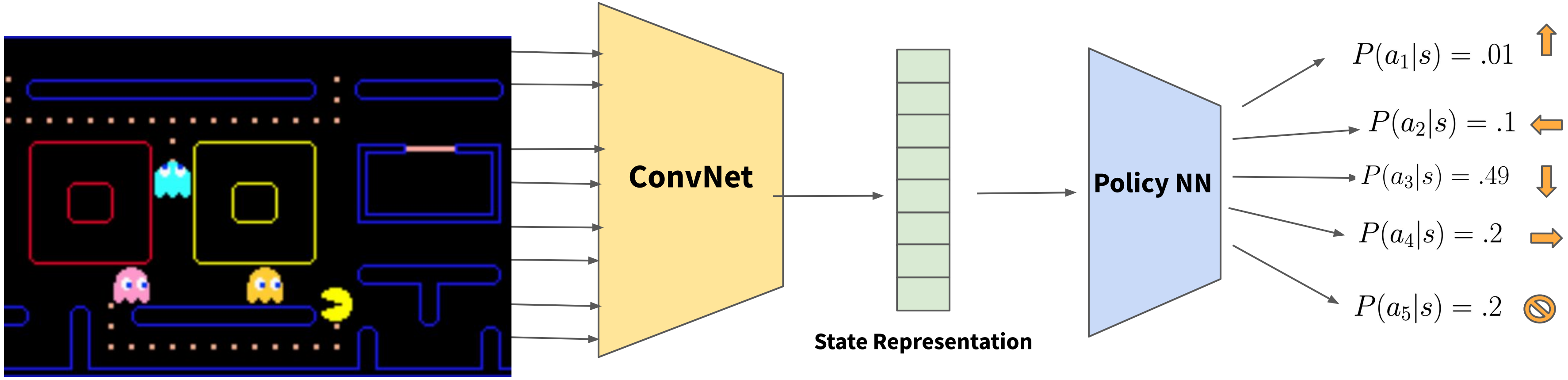


$$\boxed{\pi(s) = \underset{a}{argmax}(Q(s, a))}$$

What if we can learn the policy directly?

Cornell Bowers C·IS

Policy Gradient:



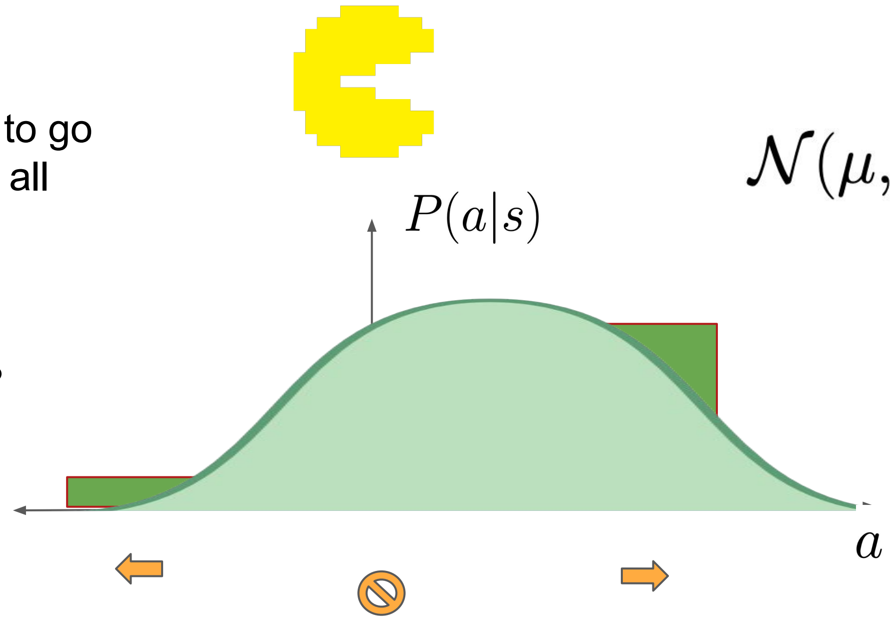
$$\pi(s) \sim P(a|s)$$

Cornell Bowers C·IS

Policy Distribution:

Pacman is trying to decide to go left, right, or not move at all

Pacman is now trying to decide how fast to move?

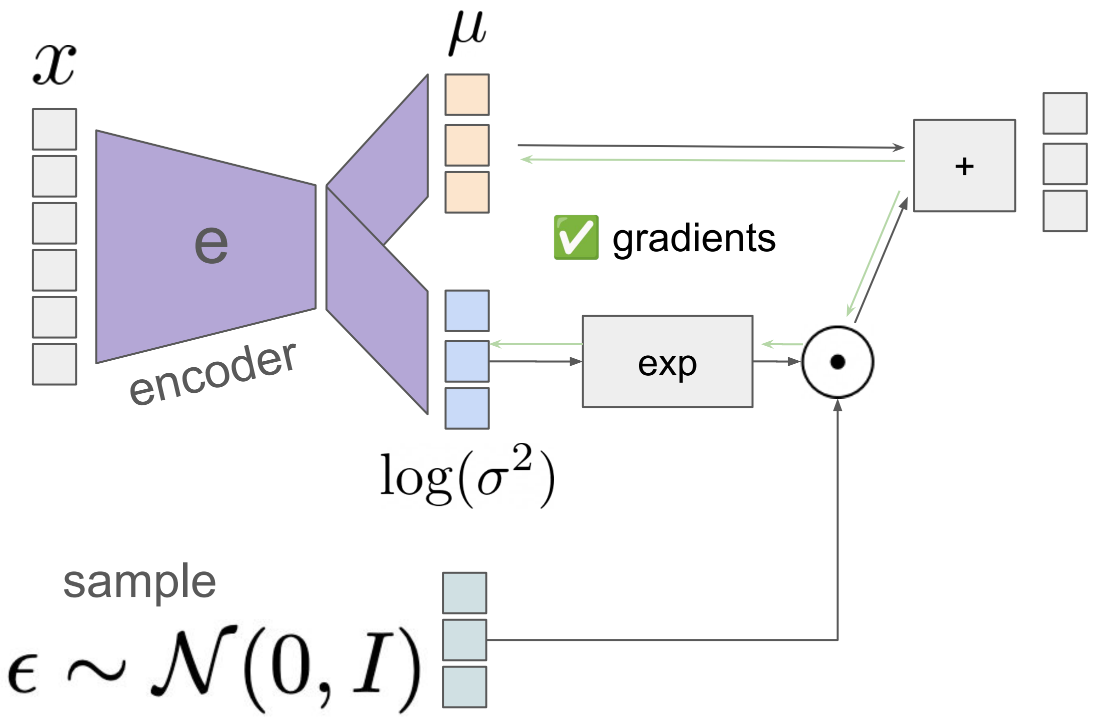


$\mathcal{N}(\mu, \sigma^2)$

$$\pi(s) \sim P(a|s)$$

Cornell Bowers C·IS

Review: The Reparameterization Trick

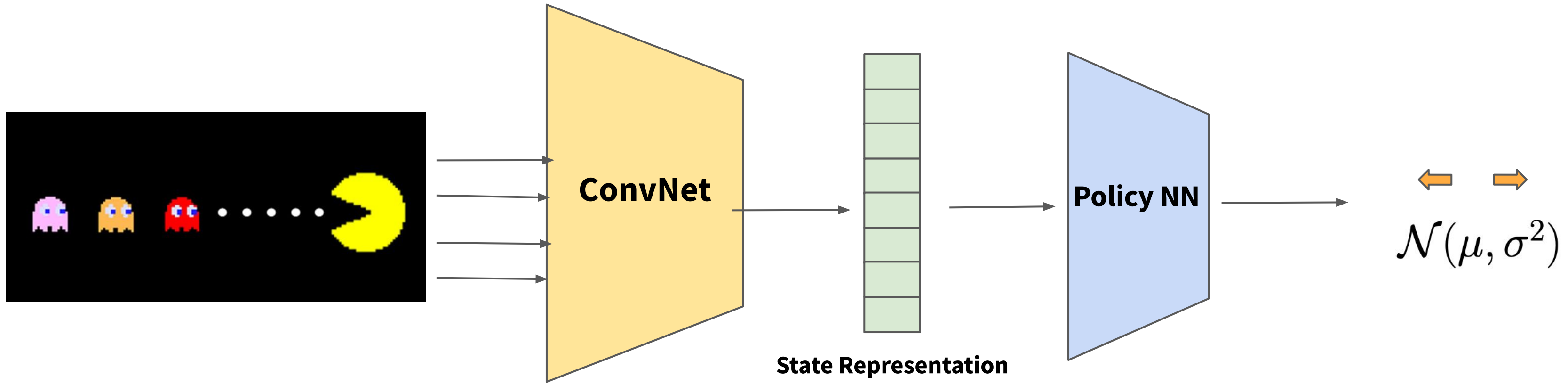


$$z = \mu + \sigma \odot \epsilon$$

$$z \sim \mathcal{N}(\mu, \sigma^2 I)$$

Cornell Bowers C·IS

Continuous Policy Gradient:



$$\pi(s) \sim \mathcal{N}(\mu, \sigma^2)$$

Cornell Bowers C·IS

Goal of Reinforcement Learning

Find the optimal policy π_θ^* that maximizes the expected discounted return $J(\theta)$:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

$$\pi_\theta^* = \arg\max_\theta J(\theta)$$

What is our loss function?

Gradient Descent:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta \mathcal{L}(\theta_k)$$

Objective function:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$$

Cornell Bowers C·IS

Policy Gradient Objective

Aim to update the policy parameters θ in the direction of the gradient $\nabla_\theta J(\theta)$ to maximize the expected discounted return:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$$

where:

- θ_k represents the policy parameters at iteration k.
- α is the learning rate that controls the step size of the parameter update.
- $\nabla_\theta J(\theta_k)$ is the gradient of the expected discounted return with respect to the policy parameters at iteration k.

Cornell Bowers C·IS

Policy Gradient Objective

To maximize the expected discounted return $J(\theta)$, we need to compute its gradient with respect to the policy parameters θ:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

where:

- $J(\theta)$ is the expected discounted return under the policy π_θ.
- τ represents a trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ sampled from the policy π_θ.
- $p_\theta(\tau)$ is the probability distribution over trajectories induced by the policy π_θ.

Cornell Bowers C·IS

Discuss

What are some of the challenges with computing this objective:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

where:

- $J(\theta)$ is the expected discounted return under the policy π_θ.
- τ represents a trajectory $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ sampled from the policy π_θ.
- $p_\theta(\tau)$ is the probability distribution over trajectories induced by the policy π_θ.

Cornell Bowers C·IS

Policy Gradient Theorem

- Will give us a way to estimate the gradient of our objective with respect to our policy network parameters

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

Cornell Bowers C·IS

Policy-Gradient Theorem (Part 1)

Express the expected discounted return $J(\theta)$ in terms of the state-value function $V^{\pi_\theta}(s)$:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right] \\ &= \mathbb{E}_{s_0 \sim p(s_0)} \left[V^{\pi_\theta}(s_0) \right] \\ &= \sum_{s_0} p(s_0) V^{\pi_\theta}(s_0) \end{aligned}$$

where:

- $V^{\pi_\theta}(s) = \mathbb{E}_{\tau \sim p_\theta(\tau | s_0 = s)} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \right]$ is the state-value function.
- $p(s_0)$ is the initial state distribution.

Recall: Action-Value (Q) Function

- Relationship between value function and Q-function:

$$V^\pi(s) = \sum_a \pi(a \mid s) Q^\pi(s, a)$$

- Can recover the value function from the Q-function by marginalizing over all possible actions.

Cornell Bowers C·IS

Policy-Gradient Theorem (Part 2)

Let's differentiate $J(\theta) = \sum_{s_0} p(s_0) V^{\pi_\theta}(s_0)$ with respect to θ:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{s_0} p(s_0) \nabla_\theta V^{\pi_\theta}(s_0) && \text{(Step 1: Differentiate w.r.t. } \theta) \\
&= \sum_{s_0} p(s_0) \nabla_\theta \sum_a \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) && \text{(Step 2: Expand } V^{\pi_\theta}(s_0)) \\
&= \sum_{s_0} p(s_0) \sum_a \nabla_\theta \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) && \text{(Step 3: Interchange } \nabla_\theta \text{ and } \textstyle\sum_a) \\
&= \sum_{s_0} p(s_0) \sum_a \pi_\theta(a|s_0) \frac{\nabla_\theta \pi_\theta(a|s_0)}{\pi_\theta(a|s_0)} Q^{\pi_\theta}(s_0, a) && \text{(Step 4: Multiply and divide by } \pi_\theta(a|s_0)) \\
&= \sum_{s_0} p(s_0) \sum_a \pi_\theta(a|s_0) \nabla_\theta \log \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) && \text{(Step 5: Log-trick)}
\end{aligned}
$$

Policy Gradient Theorem (Part 3)

Express the gradient $\nabla_\theta J(\theta)$ as an expectation over states and actions:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{s_0} p(s_0) \sum_a \pi_\theta(a|s_0) \nabla_\theta \log \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) \\
&= \sum_{s_0} \sum_a p(s_0) \pi_\theta(a|s_0) \nabla_\theta \log \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) && \text{(Step 1: Rearrange terms)} \\
&= \sum_{s_0} \sum_a p(s_0, a) \nabla_\theta \log \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) && \text{(Step 2: Define joint distribution } p(s_0, a)\text{)} \\
&= \mathbb{E}_{s_0 \sim p(s_0), a \sim \pi\theta(a|s_0)} \left[\nabla_\theta \log \pi_\theta(a|s_0) Q^{\pi_\theta}(s_0, a) \right] && \text{(Step 3: Express as an expectation)} \\
&= \mathbb{E}_{s \sim p^{\pi_\theta}(s), a \sim \pi_\theta(a|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a) \right] && \text{(Step 4: Generalize to state distribution)}
\end{aligned}$$

where $p^{\pi_\theta}(s)$ is the state distribution induced by the policy π_θ.

Cornell Bowers CIS

Policy Gradient Theorem

Policy gradient theorem expresses the gradient of the expected discounted return as an expectation over states and actions, weighted by the gradient of the log policy and the action-value function:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^{\pi_\theta}(s),a\sim\pi_\theta(a|s)}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right]$$

where $p^{\pi_\theta}(s)$ is the state distribution induced by the policy π_θ.

Cornell Bowers C·IS

REINFORCE Algorithm

Key ideas:

- Estimate the policy gradient:

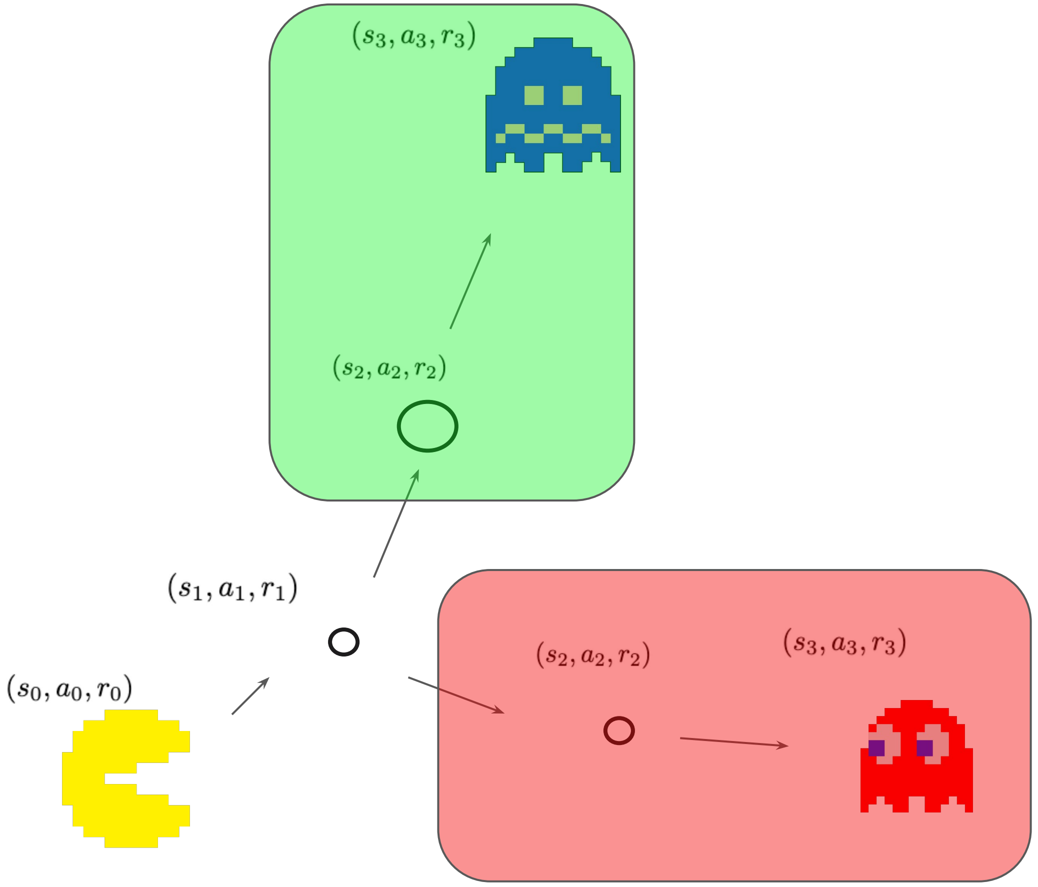
$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p^{\pi_\theta}(s), a \sim \pi_\theta(a|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right]$$

  using samples from the policy.
- Use the return $G_t = \sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{t+k+1}$ as an unbiased estimate of the action-value function $Q^{\pi_\theta}(s_t, a_t)$.
- Update the policy parameters θ in the direction of the estimated gradient.

Cornell Bowers C·IS

REINFORCE Algorithm:

1. Run a policy over the environment
2. Record actions, states, and rewards
3. Increase probability of good actions and decrease probability of bad actions



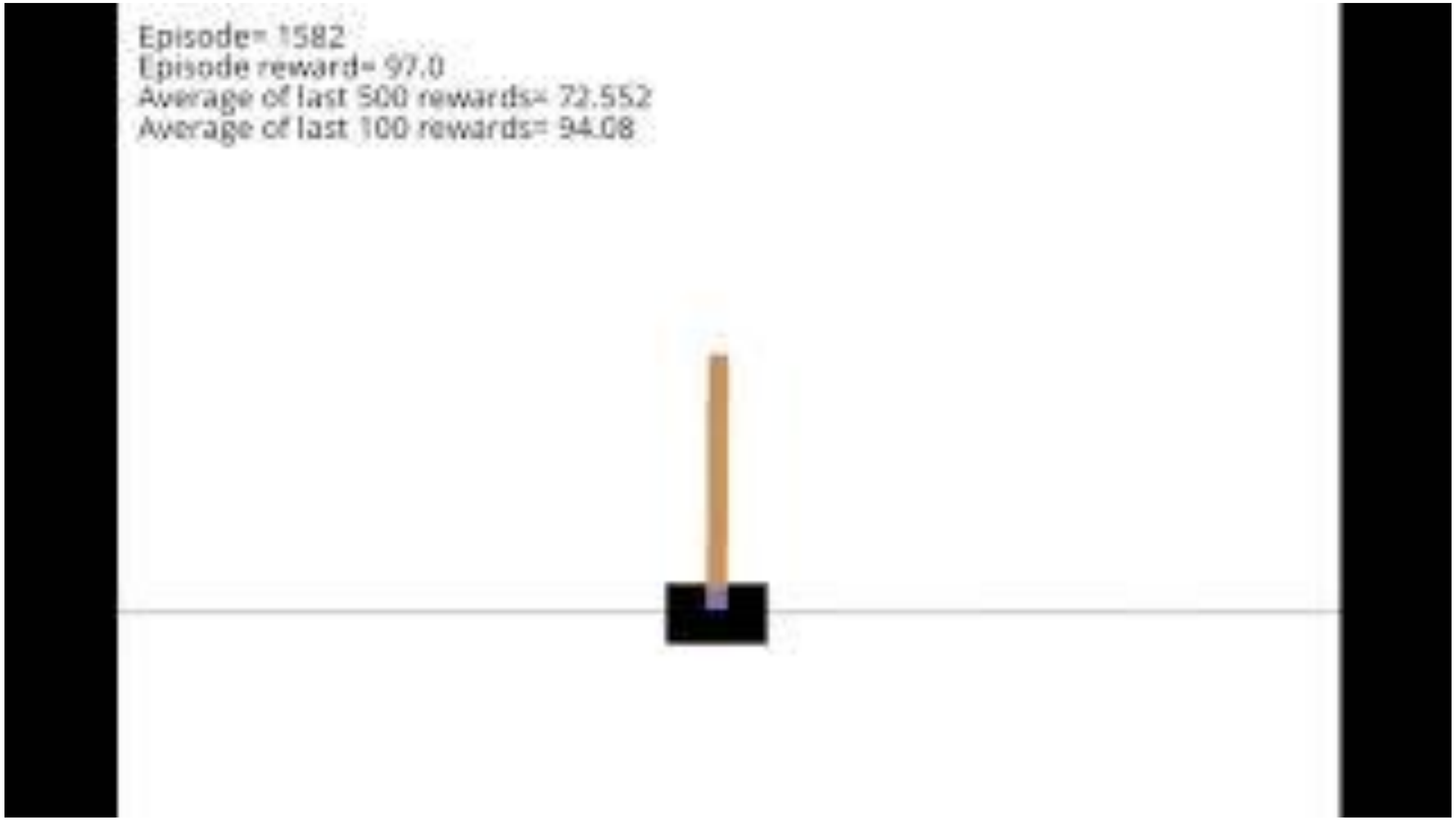
REINFORCE Algorithm

Algorithm 1 REINFORCE Algorithm

```
1: Initialize policy parameters $\theta$ randomly
2: for each episode do
3:     Sample a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ following the policy $\pi_\theta$
4:     for each timestep $t$ in the trajectory do
5:         Compute the return $G_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$
6:         Estimate the policy gradient: $\hat{g} = \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$
7:     end for
8:     Update the policy parameters: $\theta \leftarrow \theta + \alpha \frac{1}{T} \sum_{t=0}^{T} \hat{g}$, where $\alpha$ is the learning rate
9: end for
```

Cornell Bowers C·IS

REINFORCE: Demo



Episode= 1582
Episode reward= 97.0
Average of last 500 rewards= 72.552
Average of last 100 rewards= 94.08

Cornell Bowers C·IS

REINFORCE Algorithm: Advantages and Disadvantages

Advantages:

- Unbiased estimate of the policy gradient.
- Can be applied to both continuous and discrete action spaces.

Limitations:

- Return is a noisy estimate of the action-value function.
  - High variance in the gradient estimates.
- High variance leads to potential instabilities and slow convergence.

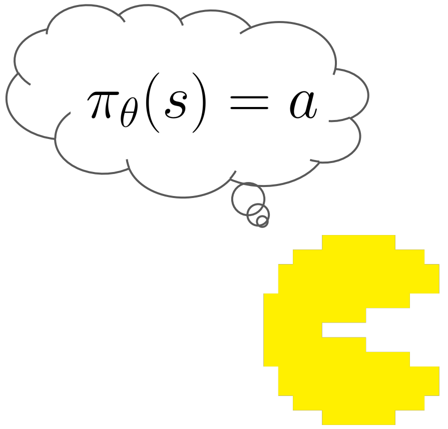
Cornell Bowers C·IS

On-Policy vs. Off-Policy

Reinforcement learning algorithms can be categorized as either on-policy or off-policy based on how they collect and use data for learning.

On-Policy Algorithms:

- Learn from experiences generated by the current policy being followed.
- The policy used for generating data is the same as the policy being improved.



Cornell Bowers C·IS

On-Policy vs. Off-Policy

Reinforcement learning algorithms can be categorized as either on-policy or off-policy based on how they collect and use data for learning.

Off-Policy Algorithms:

- Learn from experiences generated by a different policy than the one being improved.
- The behavior policy (used for generating data) is different from the target policy (being learned).
- Can learn from data generated by any policy and enables efficient use of data through replay buffers.

Discuss: Which of the following are on- vs. off-policy algorithms?

Q-Learning, Deep Q-Learning, REINFORCE

Cornell Bowers C·IS

Issues with Policy Gradient Methods

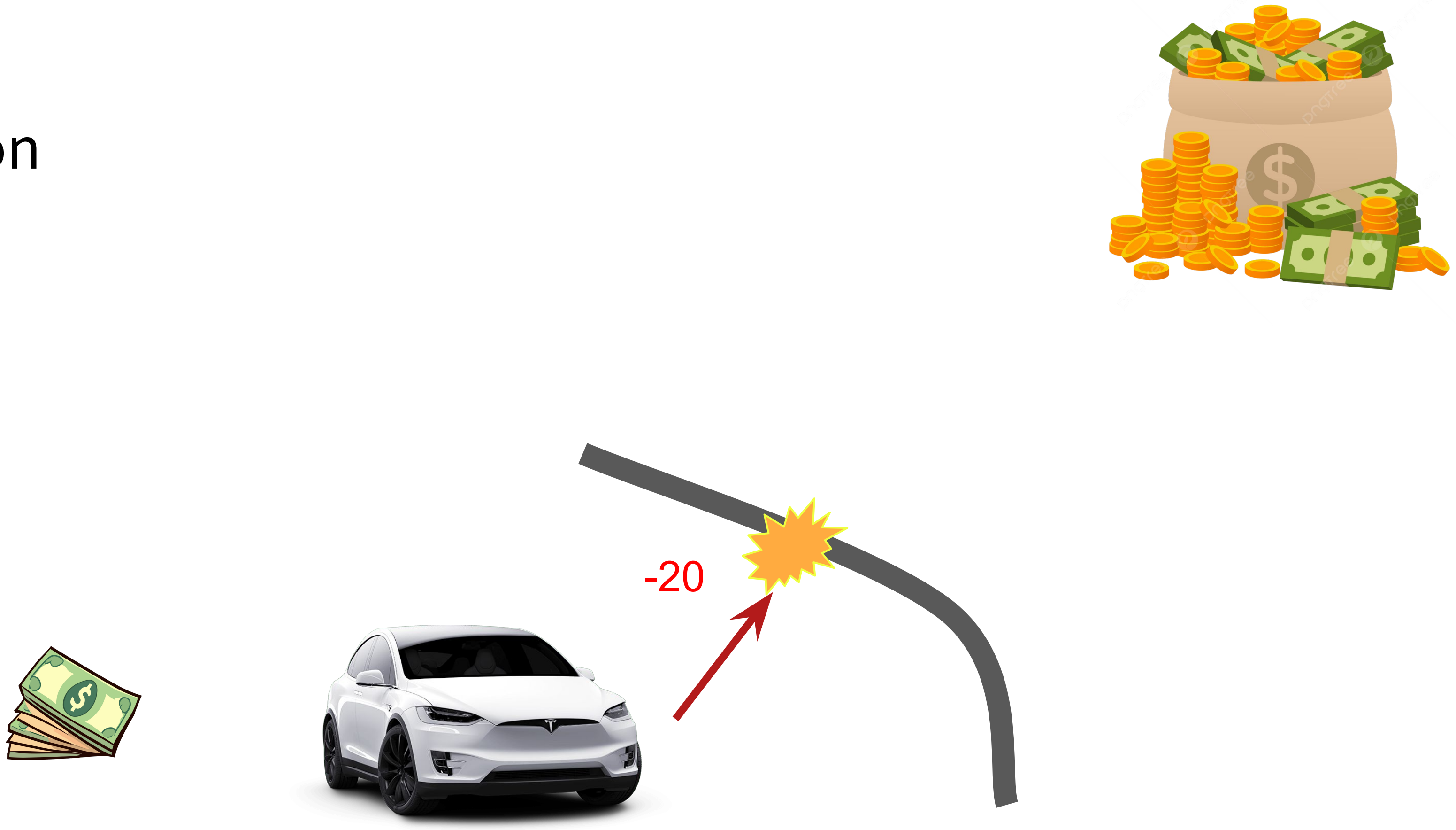
Cornell Bowers C·IS

Exploration

1. Might get stuck in Local Optima
2. Want to discover better policies
3. Adapt to changing environments
4. Generalize and handle uncertainty

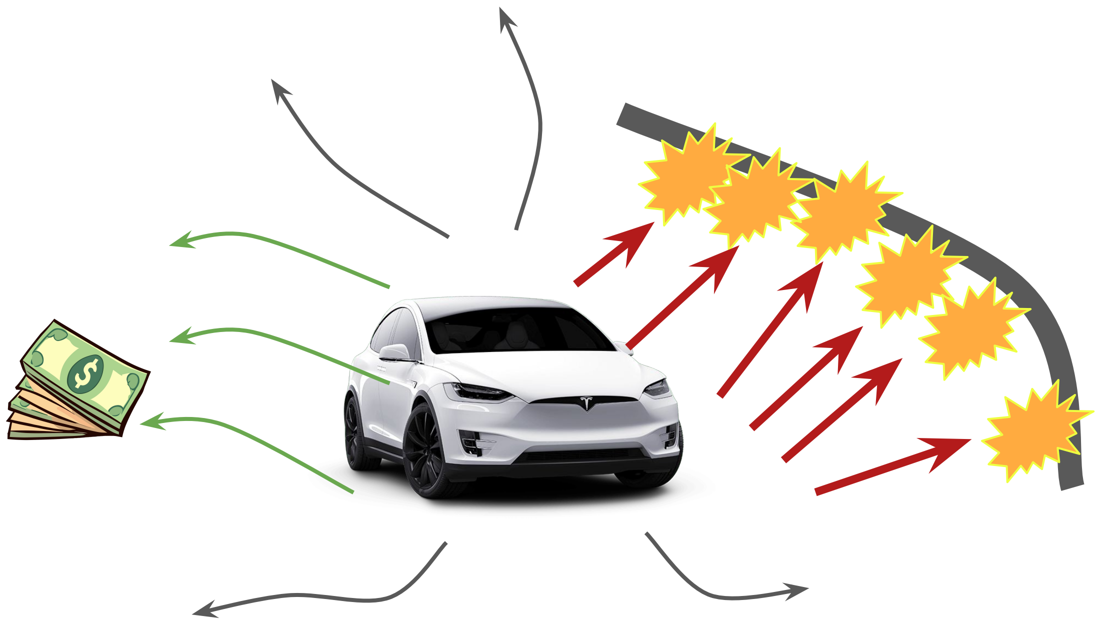
Cornell Bowers C·IS

Exploration



Cornell Bowers C·IS

Exploration

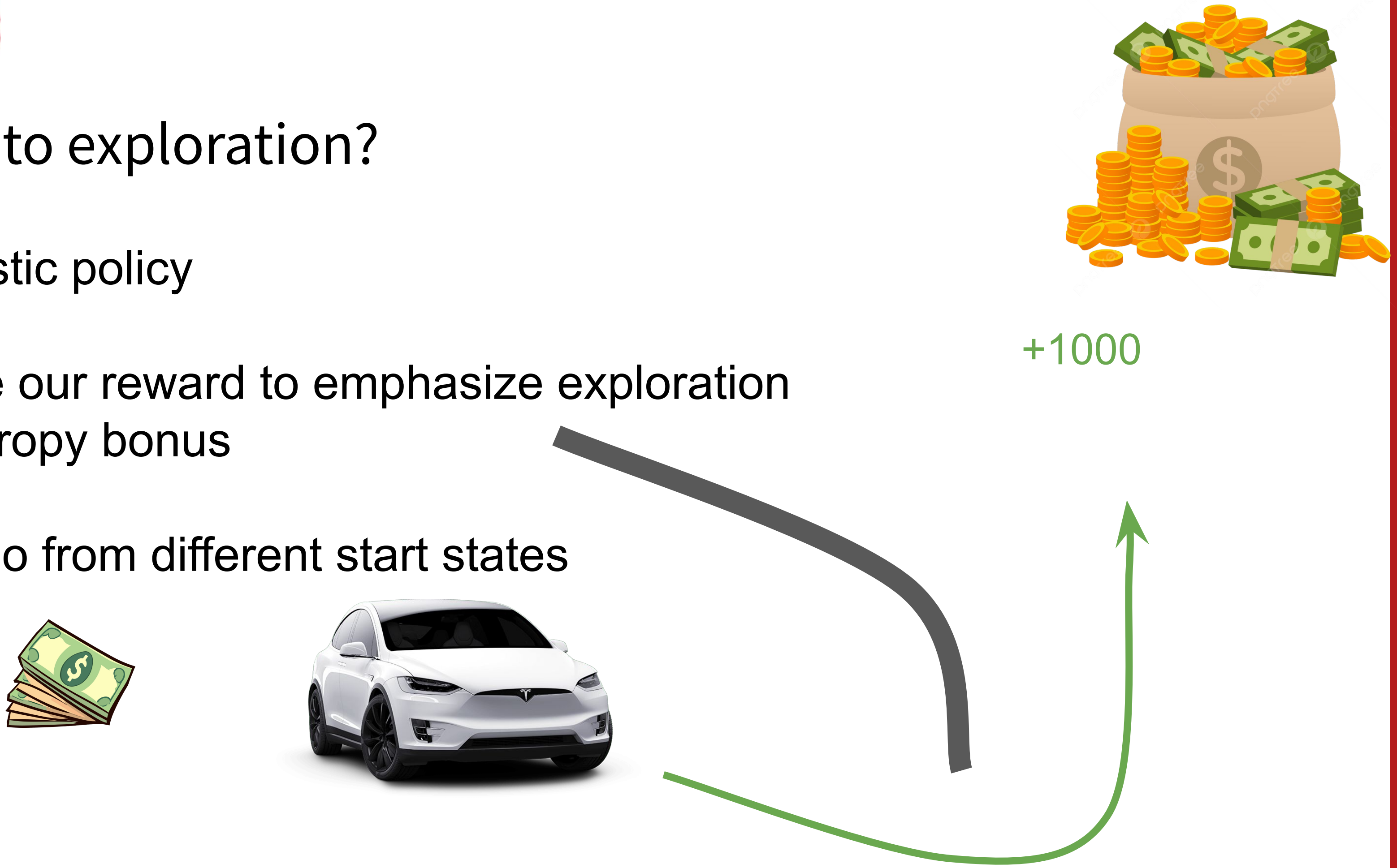


Cornell Bowers C·IS

Exploration

Solutions to exploration?

1. Stochastic policy
2. Change our reward to emphasize exploration
   a. Entropy bonus
3. Run algo from different start states



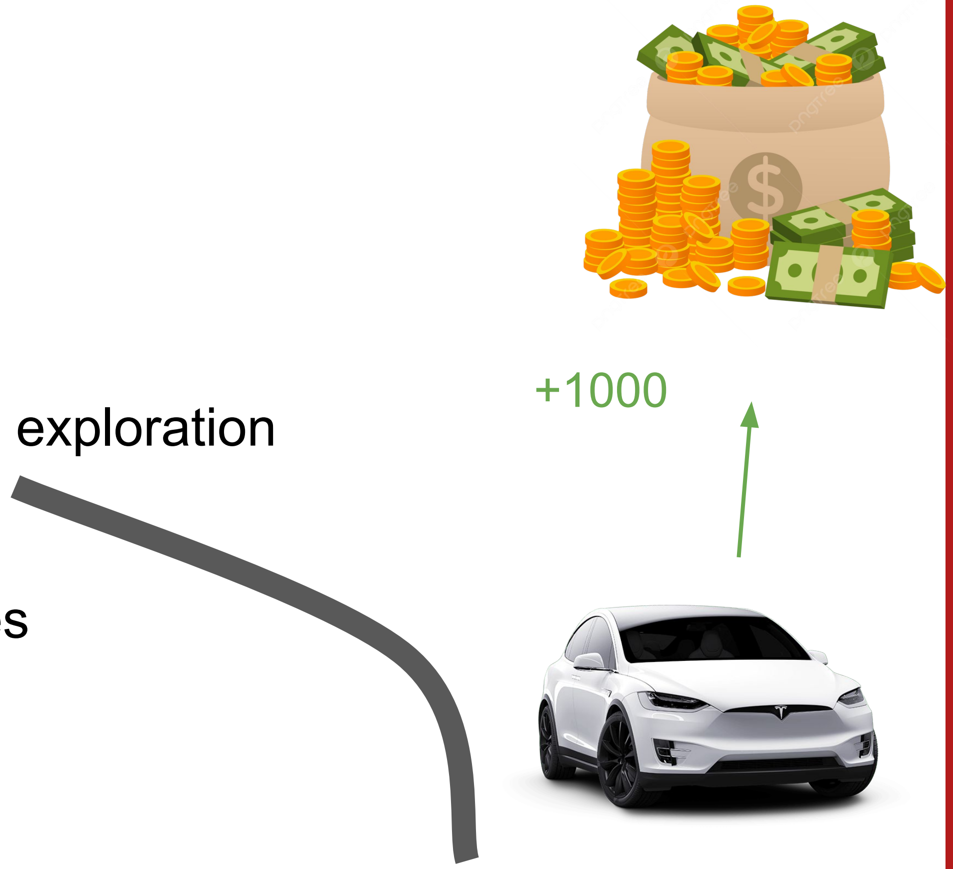
Cornell Bowers C·IS

Solutions to exploration?

1. Stochastic policy

2. Change our reward to emphasize exploration
   a. Entropy bonus

3. Run algo from different start states



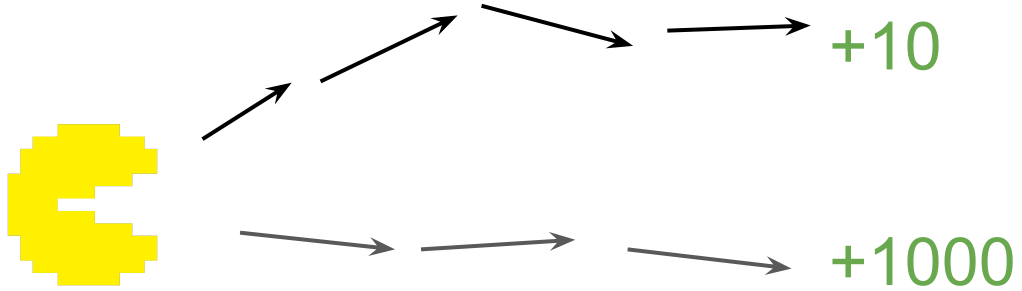
Cornell Bowers C·IS

Variance

- Your policy is dependent on randomness
- Noise = high variance in how you evaluate your policy
- Need relative information about how good a policy is

Is this a good sequence of actions?

$$\hat{g} = \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$



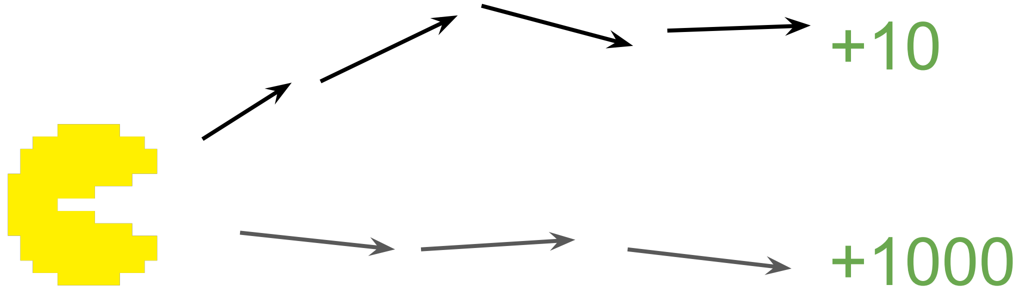
Cornell Bowers C·IS

Variance Reduction with a Baseline

How good was that sequence of actions compared to other sequences?

$$\hat{g} = \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))$$

Subtract a Baseline



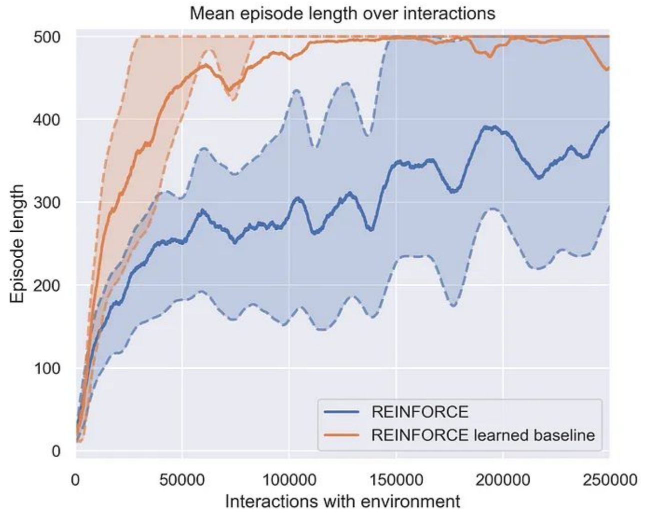
https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

Cornell Bowers C·IS

Variance Reduction with a Baseline

- Subtracting a baseline is a simple technique to reduce variance in the REINFORCE algorithm.
  - Improves learning stability and convergence speed.
- A baseline, $b(s_t)$, estimates the expected return from state s_t.
  - Can be an estimated state-value function $V^{\pi_\theta}(s_t)$ or a moving average of returns.
- The policy gradient estimate becomes:

$$\hat{g} = \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))$$

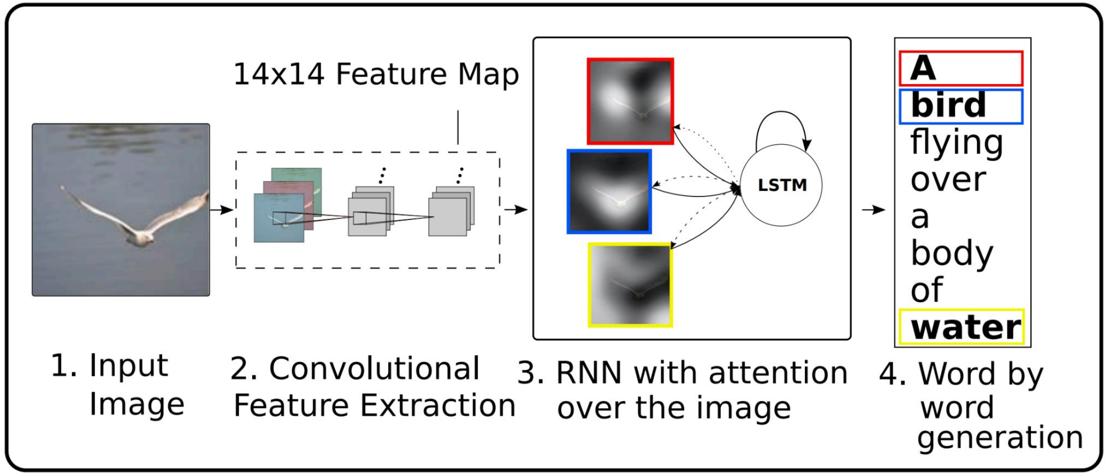


https://medium.com/@fork.tree.ai/understanding-baseline-techniques-for-reinforce-53a1e2279b57

Cornell Bowers C·IS

REINFORCE Application- Image Captioning

- Typically train image captioning models with the cross-entropy loss
  - Would be nice to directly optimize for downstream metrics of interest
    - Typically n-gram metrics of some form

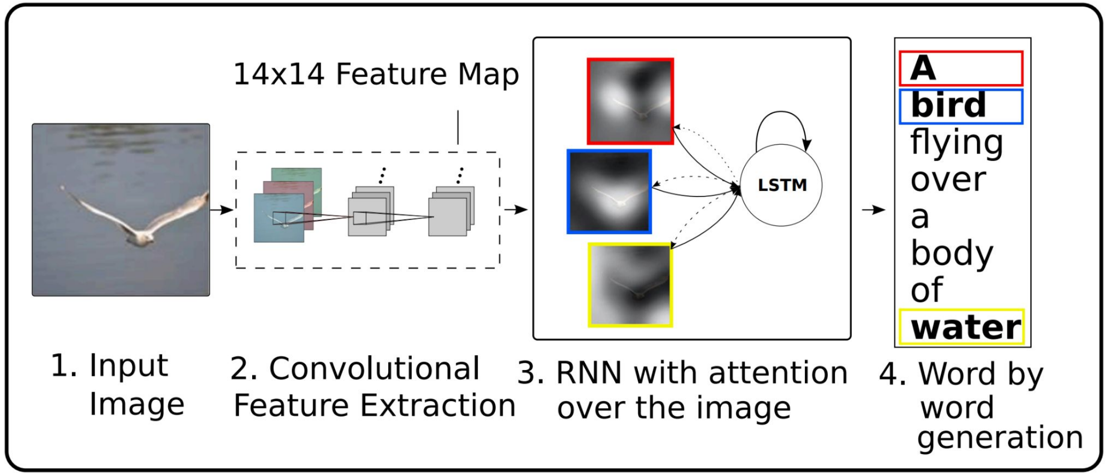

14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. PMLR, 2015.

Cornell Bowers C·IS

Self-Critical Sequence Training

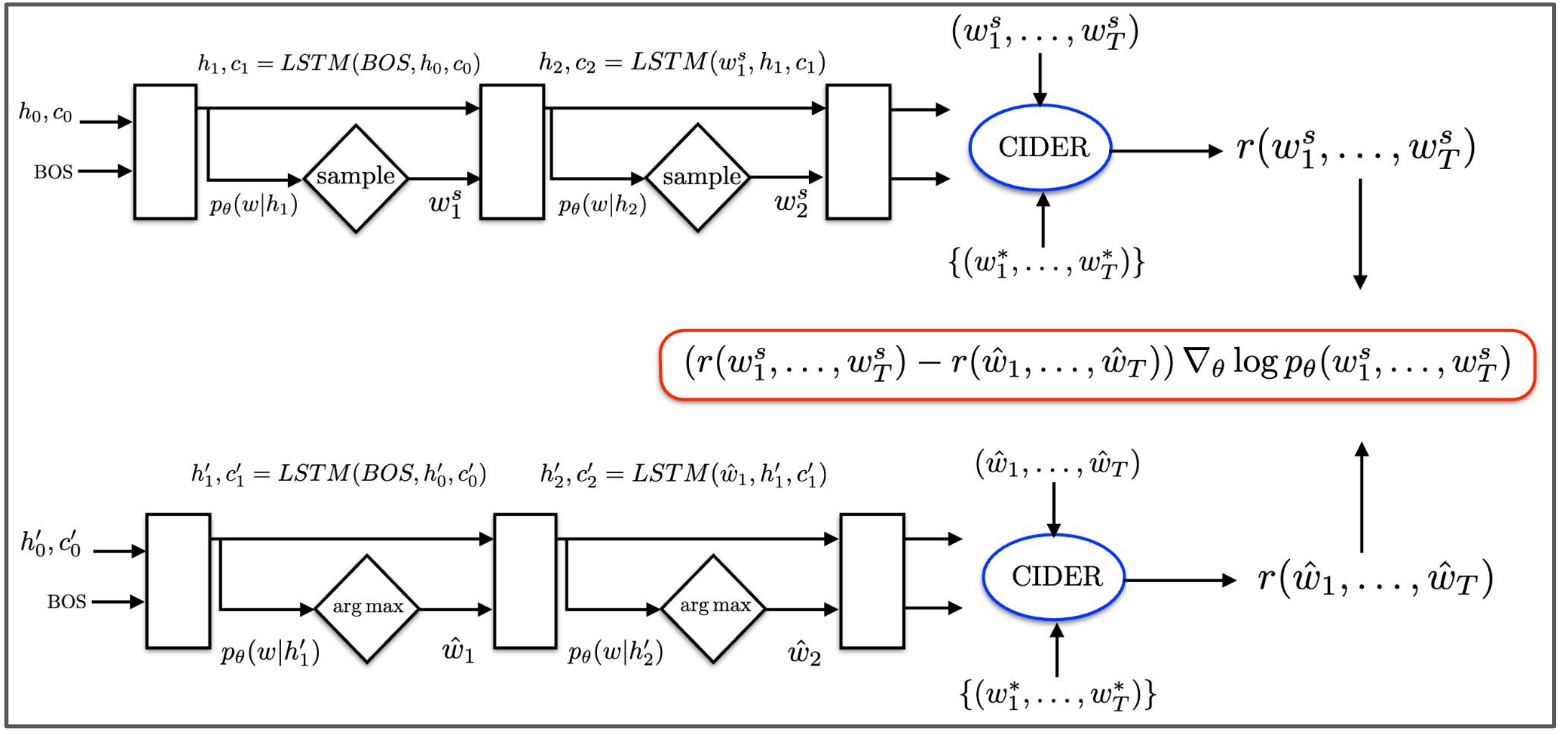
- Train image captioning model with the cross-entropy objective
- Use REINFORCE to optimize some non-differentiable metric
    - CIDEr: Average cosine similarity of n-gram feature vectors for the candidate and reference


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Rennie, Steven J., et al. "Self-critical sequence training for image captioning." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Cornell Bowers C·IS

Self-Critical Sequence Training

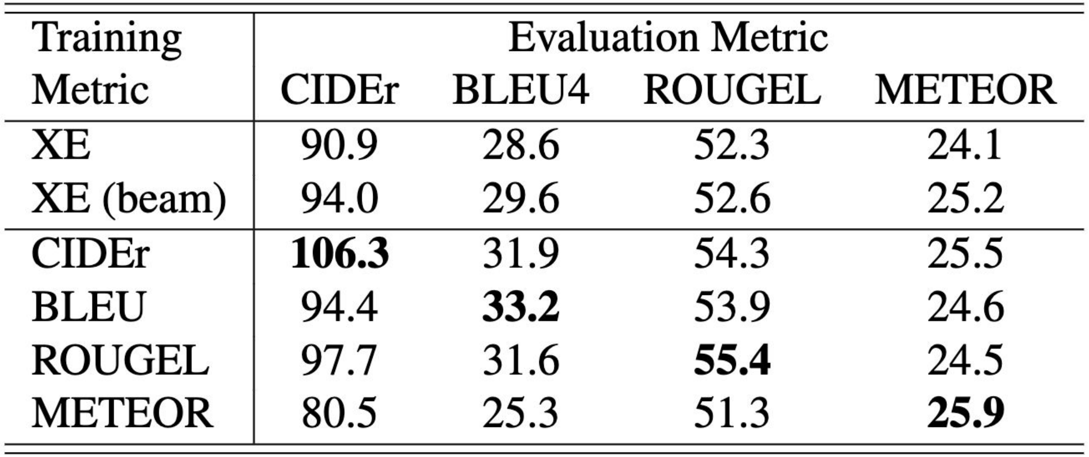


$h_1, c_1 = LSTM(BOS, h_0, c_0)$
$h_2, c_2 = LSTM(w_1^s, h_1, c_1)$
h_0, c_0
BOS
sample
$p_\theta(w|h_1)$
w_1^s
$p_\theta(w|h_2)$
w_2^s
$(w_1^s, \ldots, w_T^s)$
CIDER
$\{(w_1^*, \ldots, w_T^*)\}$
$r(w_1^s, \ldots, w_T^s)$
$(r(w_1^s, \ldots, w_T^s) - r(\hat{w}_1, \ldots, \hat{w}_T)) \nabla_\theta \log p_\theta(w_1^s, \ldots, w_T^s)$
$h_1', c_1' = LSTM(BOS, h_0', c_0')$
$h_2', c_2' = LSTM(\hat{w}_1, h_1', c_1')$
h_0', c_0'
arg max
$p_\theta(w|h_1')$
$\hat{w}_1$
$p_\theta(w|h_2')$
$\hat{w}_2$
$(\hat{w}_1, \ldots, \hat{w}_T)$
$r(\hat{w}_1, \ldots, \hat{w}_T)$

Rennie, Steven J., et al. "Self-critical sequence training for image captioning." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Self-Critical Sequence Training

- Can directly optimize non-differentiable downstream metrics with REINFORCE



Training Metric	Evaluation Metric			
	CIDEr	BLEU4	ROUGEL	METEOR
XE	90.9	28.6	52.3	24.1
XE (beam)	94.0	29.6	52.6	25.2
CIDEr	**106.3**	31.9	54.3	25.5
BLEU	94.4	**33.2**	53.9	24.6
ROUGEL	97.7	31.6	**55.4**	24.5
METEOR	80.5	25.3	51.3	**25.9**

Cornell Bowers C·IS

Actor-Critic Algorithms

Combine the benefits of both policy-based and value-based approaches.

Key Components:

- **Actor**: The actor is a policy network $\pi_\theta(a|s)$ that maps states to probability distributions over actions. It selects actions based on the current policy.
- **Critic**: The critic is a value network $V_\phi(s)$ or $Q_\phi(s, a)$ that estimates the expected return or action-value function. It evaluates the quality of the actor's decisions.

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(a|s) Q_\phi(s, a)$$

Cornell Bowers C·IS

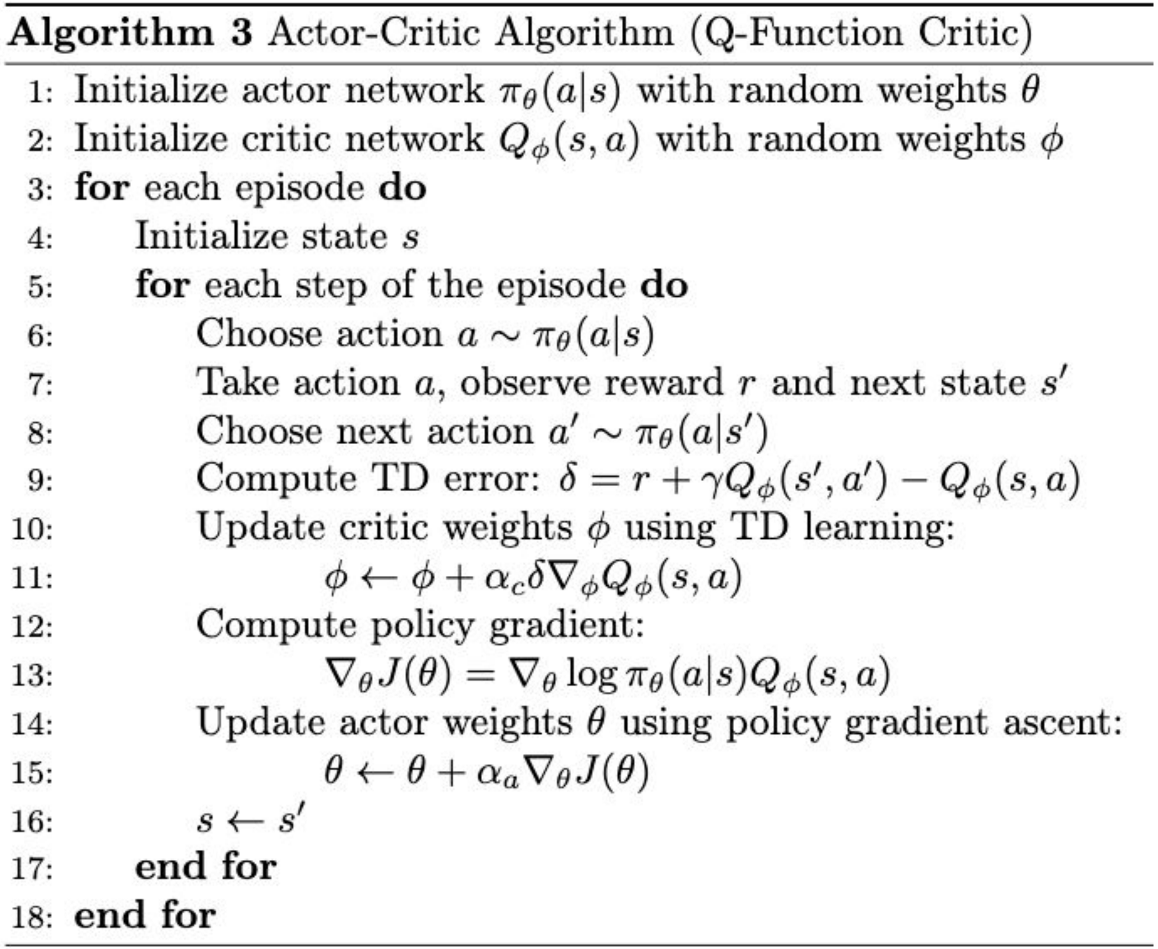
Actor-Critic Algorithms

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(a|s) Q_\phi(s,a)$$

Moving right sounds like a good choice

Don’t do that. Bad Idea!

Actor-Critic Algorithm



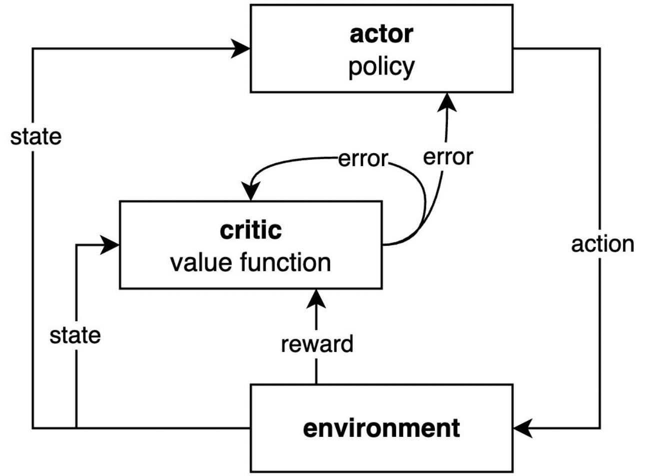
Algorithm 3 Actor-Critic Algorithm (Q-Function Critic)

```
 1: Initialize actor network $\pi_\theta(a|s)$ with random weights $\theta$
 2: Initialize critic network $Q_\phi(s, a)$ with random weights $\phi$
 3: for each episode do
 4:     Initialize state $s$
 5:     for each step of the episode do
 6:         Choose action $a \sim \pi_\theta(a|s)$
 7:         Take action $a$, observe reward $r$ and next state $s'$
 8:         Choose next action $a' \sim \pi_\theta(a|s')$
 9:         Compute TD error: $\delta = r + \gamma Q_\phi(s', a') - Q_\phi(s, a)$
10:         Update critic weights $\phi$ using TD learning:
11:                 $\phi \leftarrow \phi + \alpha_c \delta \nabla_\phi Q_\phi(s, a)$
12:         Compute policy gradient:
13:                 $\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(a|s) Q_\phi(s, a)$
14:         Update actor weights $\theta$ using policy gradient ascent:
15:                 $\theta \leftarrow \theta + \alpha_a \nabla_\theta J(\theta)$
16:         $s \leftarrow s'$
17:     end for
18: end for
```

Cornell Bowers C·IS

Actor-Critic: Advantages

- Combine the benefits of policy-based and value-based methods.
- The critic helps in reducing the variance of the policy gradient estimates.
- The actor allows for continuous and stochastic action spaces.
- Can be more sample-efficient compared to pure policy-based methods.



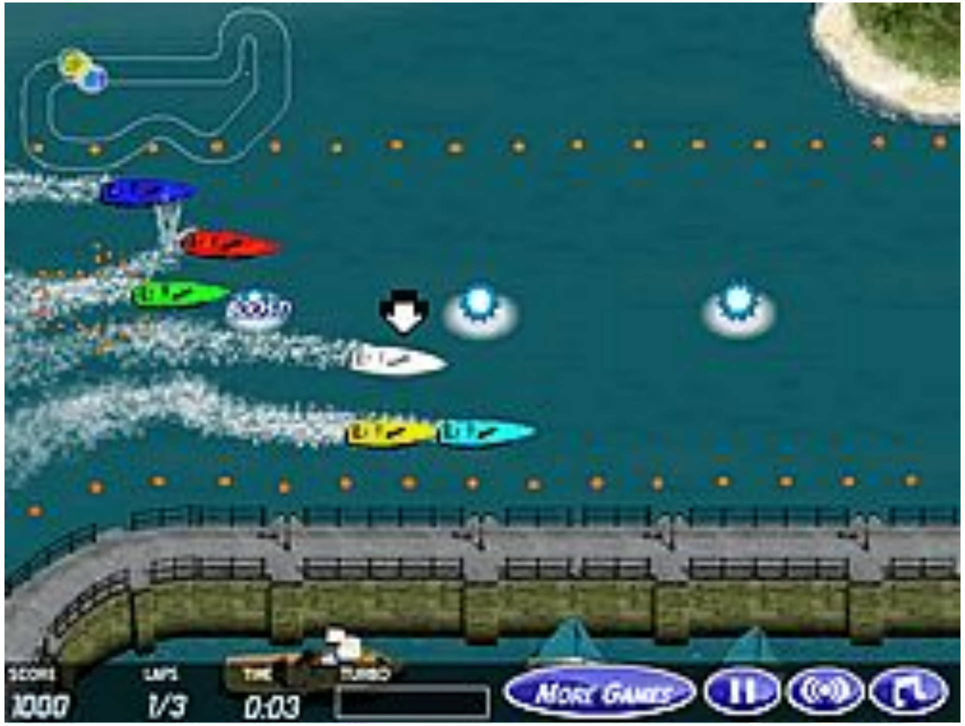
Challenges with RL in the Real World

Sample Inefficiency + Danger + Cost = Simulation

Cornell Bowers C·IS

Reward hacking

Learn to maximize the reward in unexpected ways



Cornell Bowers C·IS

How do we get what we want, NOT what we say we want?

It is important to have a good reward function

Recap

- Policy gradient methods directly learn the policy function
  - Policy Gradient Theorem lets us estimate the gradient of expected reward with respect to the policy parameters
- REINFORCE learns the policy network parameters with an unbiased estimate of the gradient
  - But can have high variance!
- Actor-Critic algorithms introduce a learnable critic to estimate the gradient
  - Reduces variance