TEXAS TECH UNIVERSITY SYSTEM™

Reinforcement Learning Algorithms

Code Implementation

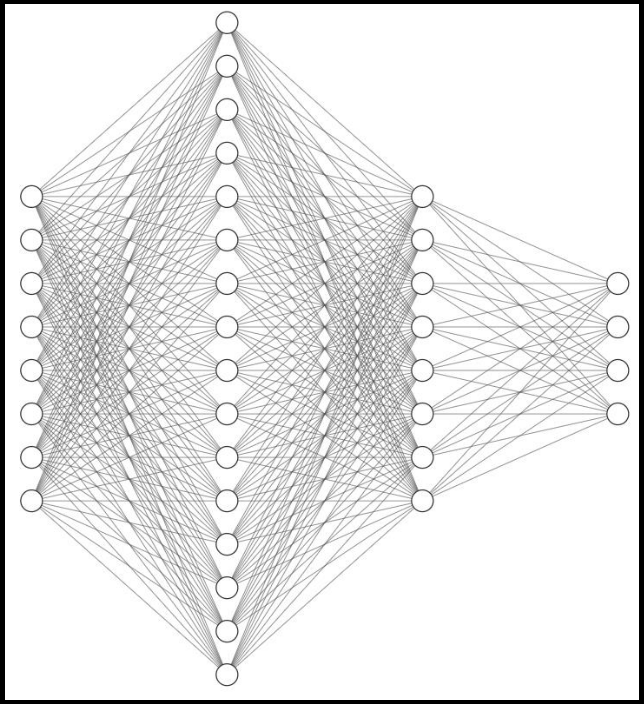
Saif Al Wahaibi
PhD Candidate

July 19th, 2023

Q-Learning

- Intuition:
  - *Estimate $Q_{\boldsymbol{\pi}}(s, a)$ via function approximation*

$s \rightarrow$



$\rightarrow \hat{Q}_{\boldsymbol{\theta}}(s, a_1)$

$\vdots$

$\rightarrow \hat{Q}_{\boldsymbol{\theta}}(s, a_n)$

- Math:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\left(\delta_{TD} - \hat{Q}_{\boldsymbol{\theta}}(s, a)\right)^2\right] = \mathbb{E}_\pi\left[\left(r + \gamma \max_{a'} \hat{Q}_{\boldsymbol{\theta}}(s', a') - \hat{Q}_{\boldsymbol{\theta}}(s, a)\right)^2\right]$$

Q-Learning Algorithm

- Pseudocode:

Initialize $\hat{Q}_{\boldsymbol{\theta}}(s, a)$ *with random weights*
for *episode = 1, 2, 3, …, E* **do**
  Initialize environment $s_{\mathbf{0}}$
  for *t = 0, 1, 2, …, T* **do**
    Select action $a_{\boldsymbol{t}}$ *randomly with probability* ϵ, *otherwise*

$$a_{\boldsymbol{t}} = \operatorname*{argmax}_{a_{\boldsymbol{t}}} \hat{Q}_{\boldsymbol{\theta}}(s_{\boldsymbol{t}}, a_{\boldsymbol{t}})$$

    Execute action $a_{\boldsymbol{t}}$ *in environment and observe* $r_{\boldsymbol{t+1}}$, $s_{\boldsymbol{t+1}}$, *and terminal or truncate flags*
    Set TD target $\delta_{TD} = r_{\boldsymbol{t+1}}$ *if terminal or truncate, otherwise*

$$\delta_{TD} = r_{\boldsymbol{t+1}} + \gamma \max_{a_{\boldsymbol{t+1}}} \hat{Q}_{\boldsymbol{\theta}}(s_{\boldsymbol{t+1}}, a_{\boldsymbol{t+1}})$$

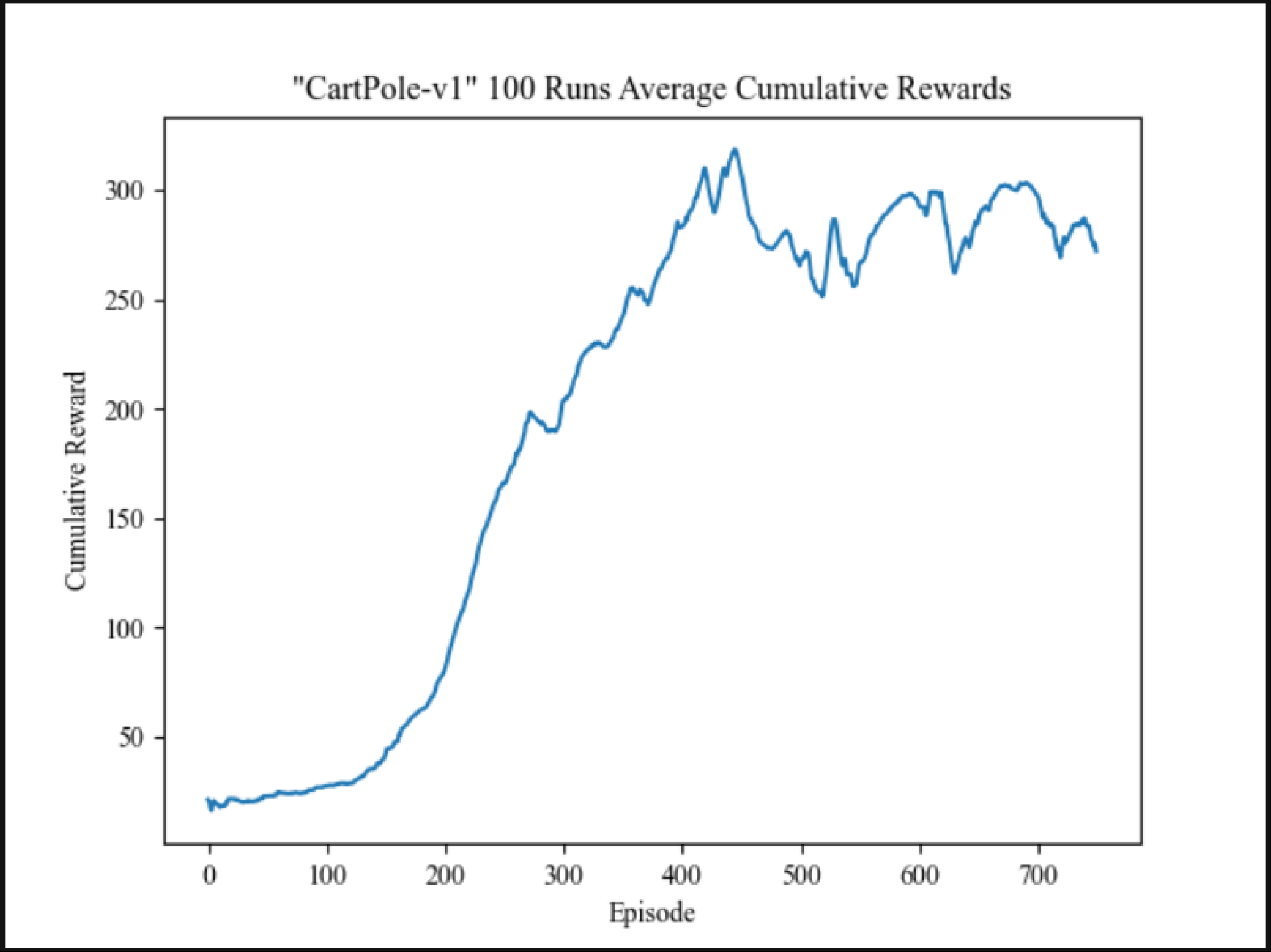
    Perform a gradient descent step on

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[\left(\delta_{TD} - \hat{Q}_{\boldsymbol{\theta}}(s_{\boldsymbol{t}}, a_{\boldsymbol{t}})\right)^2\right]$$

    Set $s_{\boldsymbol{t+1}}$ *as current state*
  end for
end for

Q-Learning Example

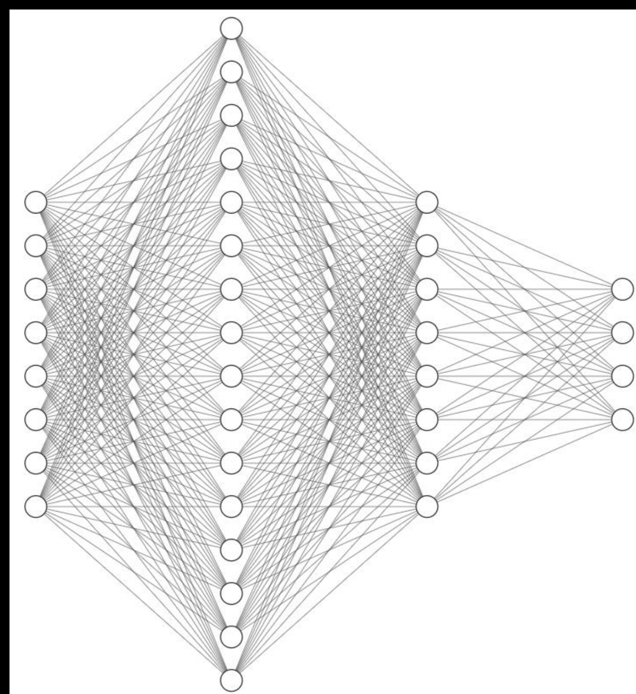
- "CartPole-v1" Gym Environment:



Double Q-Learning

- Intuition:
  - *Estimate $Q_{\boldsymbol{\pi}}(s, a)$ via function approximation*
  - *Use two Q-networks to handle maximization bias*

$s \rightarrow$



$$\left(\rightarrow \hat{Q}_{\boldsymbol{\theta}_x}(s, a_1) \quad \vdots \quad \rightarrow \hat{Q}_{\boldsymbol{\theta}_x}(s, a_n) \right) \times 2$$

Double Q-Learning

- Example of Maximization Bias:
  - *Assume* $\hat{Q} = 150$
  - *10 noisy estimates of* $\hat{Q}$*:*

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
150	151	151	150	151	151	151	152	150	149

  - *Another 10 noisy estimates of* $\hat{Q}$ *generated similarly but independently:*

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
150	151	147	151	151	150	150	150	151	148

- Math:

$$\mathcal{J}(\boldsymbol{\theta}_i) = \mathbb{E}_\pi\left[\left(r + \gamma \hat{Q}_{\boldsymbol{\theta}_j}\left(s', \underset{a'}{\operatorname{argmax}} \hat{Q}_{\boldsymbol{\theta}_i}(s', a')\right) - \hat{Q}_{\boldsymbol{\theta}_i}(s, a)\right)^2\right]$$

Double Q-Learning Algorithm

- Pseudocode:

Initialize both $\hat{Q}_{\boldsymbol{\theta}_x}(s, a)$ *with random weights*
for *episode = 1, 2, 3, …, E* ***do***
Initialize environment $s_{\mathbf{0}}$
for *t = 0, 1, 2, …, T* ***do***
Select action $a_{\boldsymbol{t}}$ *randomly with probability* ϵ*, otherwise*

$$a_{\boldsymbol{t}} = \underset{a_{\boldsymbol{t}}}{\operatorname{argmax}} \left(\frac{\hat{Q}_{\boldsymbol{\theta}_1}(s_{\boldsymbol{t}}, a_{\boldsymbol{t}}) + \hat{Q}_{\boldsymbol{\theta}_2}(s_{\boldsymbol{t}}, a_{\boldsymbol{t}})}{\mathbf{2}} \right)$$

Execute action $a_{\boldsymbol{t}}$ *in environment and observe* $r_{\boldsymbol{t+1}}$*,* $s_{\boldsymbol{t+1}}$*,*
and terminal or truncate flags
Choose at random either to update 1 or 2
if *i = 1 or 2* ***then***
Set TD target $\delta_{TD} = r_{\boldsymbol{t+1}}$ *if terminal or truncate,*
otherwise

$$a^*_{\boldsymbol{t+1}} = \underset{a_{\boldsymbol{t+1}}}{\operatorname{argmax}} \hat{Q}_{\boldsymbol{\theta}_i}(s_{\boldsymbol{t+1}}, a_{\boldsymbol{t+1}})$$

$$\delta_{TD} = r_{\boldsymbol{t+1}} + \gamma \hat{Q}_{\boldsymbol{\theta}_{3-i}}(s_{\boldsymbol{t+1}}, a^*_{\boldsymbol{t+1}})$$

Double Q-Learning Algorithm

$$a^*_{\boldsymbol{t+1}} = \underset{a_{\boldsymbol{t+1}}}{\operatorname{argmax}} \hat{Q}_{\boldsymbol{\theta}_i}(s_{\boldsymbol{t+1}}, a_{\boldsymbol{t+1}})$$

$$\delta_{TD} = r_{\boldsymbol{t+1}} + \gamma \hat{Q}_{\boldsymbol{\theta}_{3-i}}(s_{\boldsymbol{t+1}}, a^*_{\boldsymbol{t+1}})$$

Perform a gradient descent step on

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\left(\delta_{TD} - \hat{Q}_{\boldsymbol{\theta}_i}(s_{\boldsymbol{t}}, a_{\boldsymbol{t}})\right)^2\right]$$

end if

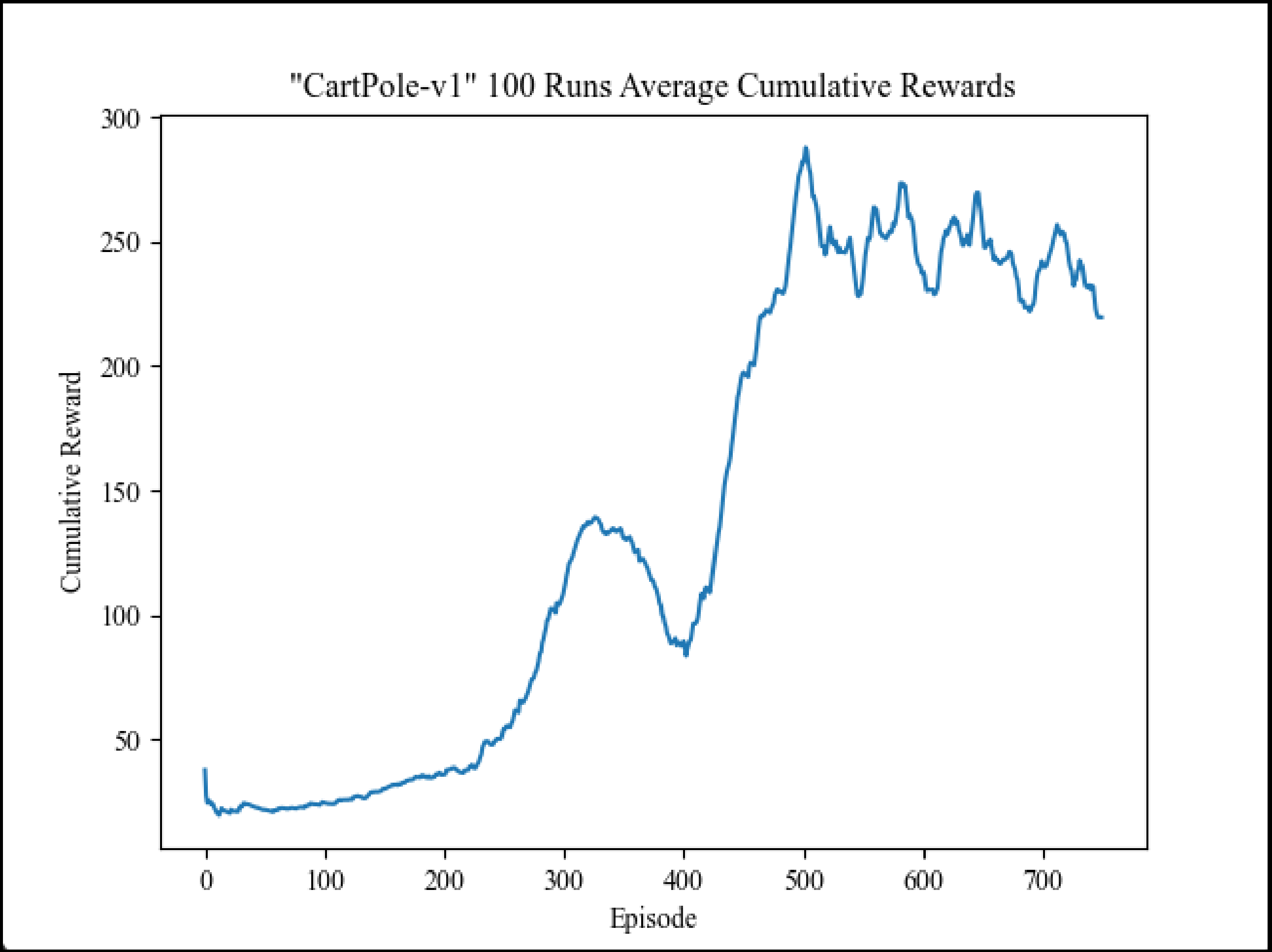
Set $s_{\boldsymbol{t+1}}$ *as current state*

end for

end for

Double Q-Learning Example

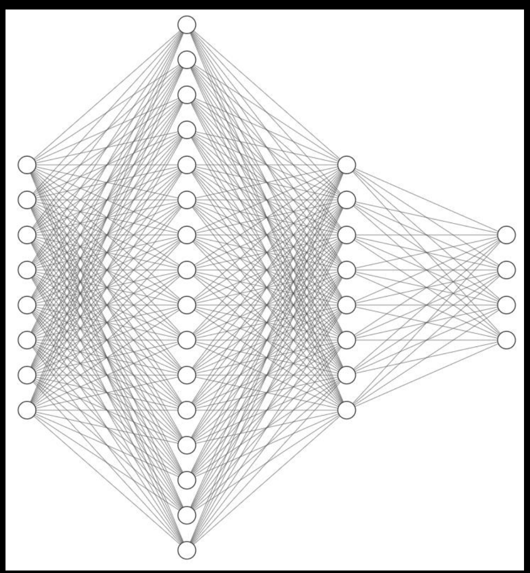
- "CartPole-v1" Gym Environment:



Deep Q-Learning Network

- Intuition:
  - *Estimate $Q_{\boldsymbol{\pi}}(s,a)$ via function approximation*
  - *Use two Q-networks, one frozen and the other online, to fix the TD-target*
  - *Utilize a replay buffer to store experience and train networks in randomized batches*

$s \rightarrow$

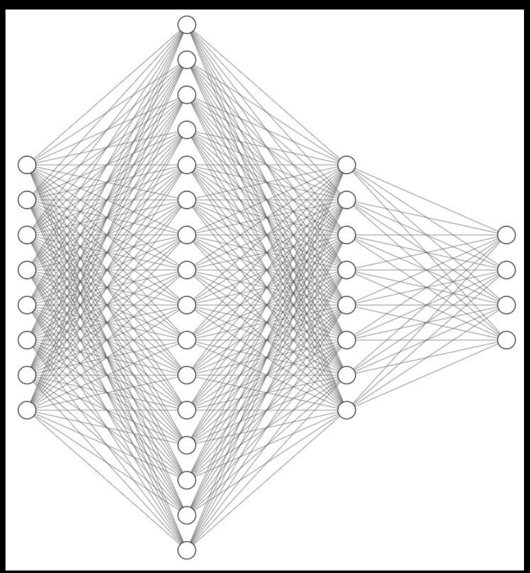


$\rightarrow \hat{Q}_{\boldsymbol{\theta}'}(s,a_1)$

$\vdots$

$\rightarrow \hat{Q}_{\boldsymbol{\theta}'}(s,a_n)$

$s \rightarrow$



$\rightarrow \hat{Q}_{\boldsymbol{\theta}}(s,a_1)$

$\vdots$

$\rightarrow \hat{Q}_{\boldsymbol{\theta}}(s,a_n)$

- Math:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[\left(r + \gamma \max_{a'} \hat{Q}_{\boldsymbol{\theta}'}(s',a') - \hat{Q}_{\boldsymbol{\theta}}(s,a)\right)^2\right]$$

Deep Q-Learning Network Algorithm

- Pseudocode:

Initialize both $\hat{Q}_{\boldsymbol{\theta}}(s,a)$ *&* $\hat{Q}_{\boldsymbol{\theta}'}(s,a)$ *with random weights*
Initialize replay buffer
for *episode = 1, 2, 3, …, E* **do**
Initialize environment $s_{\mathbf{0}}$
for *t = 0, 1, 2, …, T* **do**
Select action $a_{\boldsymbol{t}}$ *randomly with probability* ϵ*, otherwise*

$$a_{\boldsymbol{t}} = \underset{a_{\boldsymbol{t}}}{\operatorname{argmax}}\, \hat{Q}_{\boldsymbol{\theta}}(s_{\boldsymbol{t}}, a_{\boldsymbol{t}})$$

Execute action $a_{\boldsymbol{t}}$ *in environment and observe* $r_{\boldsymbol{t+1}}$*,* $s_{\boldsymbol{t+1}}$*, and terminal or truncate flags*
Save $s_{\boldsymbol{t}}$*,* $a_{\boldsymbol{t}}$*,* $r_{\boldsymbol{t+1}}$*,* $s_{\boldsymbol{t+1}}$*, and terminal or truncate flags in replay buffer*
Sample a random batch of ${s^{\mathcal{R}}}_{\boldsymbol{t}}$*,* ${a^{\mathcal{R}}}_{\boldsymbol{t}}$*,* ${r^{\mathcal{R}}}_{\boldsymbol{t+1}}$*,* ${s^{\mathcal{R}}}_{\boldsymbol{t+1}}$*, and terminal or truncate flags from replay buffer*
Set TD target $\delta_{TD} = {r^{\mathcal{R}}}_{\boldsymbol{t+1}}$ *if terminal or truncate, otherwise*

$$\delta_{TD} = {r^{\mathcal{R}}}_{\boldsymbol{t+1}} + \gamma \max_{{a^{\mathcal{R}}}_{\boldsymbol{t+1}}} \hat{Q}_{\boldsymbol{\theta}'}\left({s^{\mathcal{R}}}_{\boldsymbol{t+1}}, {a^{\mathcal{R}}}_{\boldsymbol{t+1}}\right)$$

Deep Q-Learning Network Algorithm

Set TD target $\delta_{TD} = r^{\mathcal{R}}{}_{\boldsymbol{t+1}}$ *if terminal or truncate, otherwise*

$$\delta_{TD} = r^{\mathcal{R}}{}_{\boldsymbol{t+1}} + \gamma \max_{a^{\mathcal{R}}{}_{\boldsymbol{t+1}}} \hat{Q}_{\boldsymbol{\theta}'}\left(s^{\mathcal{R}}{}_{\boldsymbol{t+1}}, a^{\mathcal{R}}{}_{\boldsymbol{t+1}}\right)$$

Perform a gradient descent step on

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[\left(\delta_{TD} - \hat{Q}_{\boldsymbol{\theta}}\left(s^{\mathcal{R}}{}_{\boldsymbol{t}}, a^{\mathcal{R}}{}_{\boldsymbol{t}}\right)\right)^2\right]$$

Count learning steps:

$i = i + 1$

if *i = replace* ***then***

Update target network:

$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$

end if

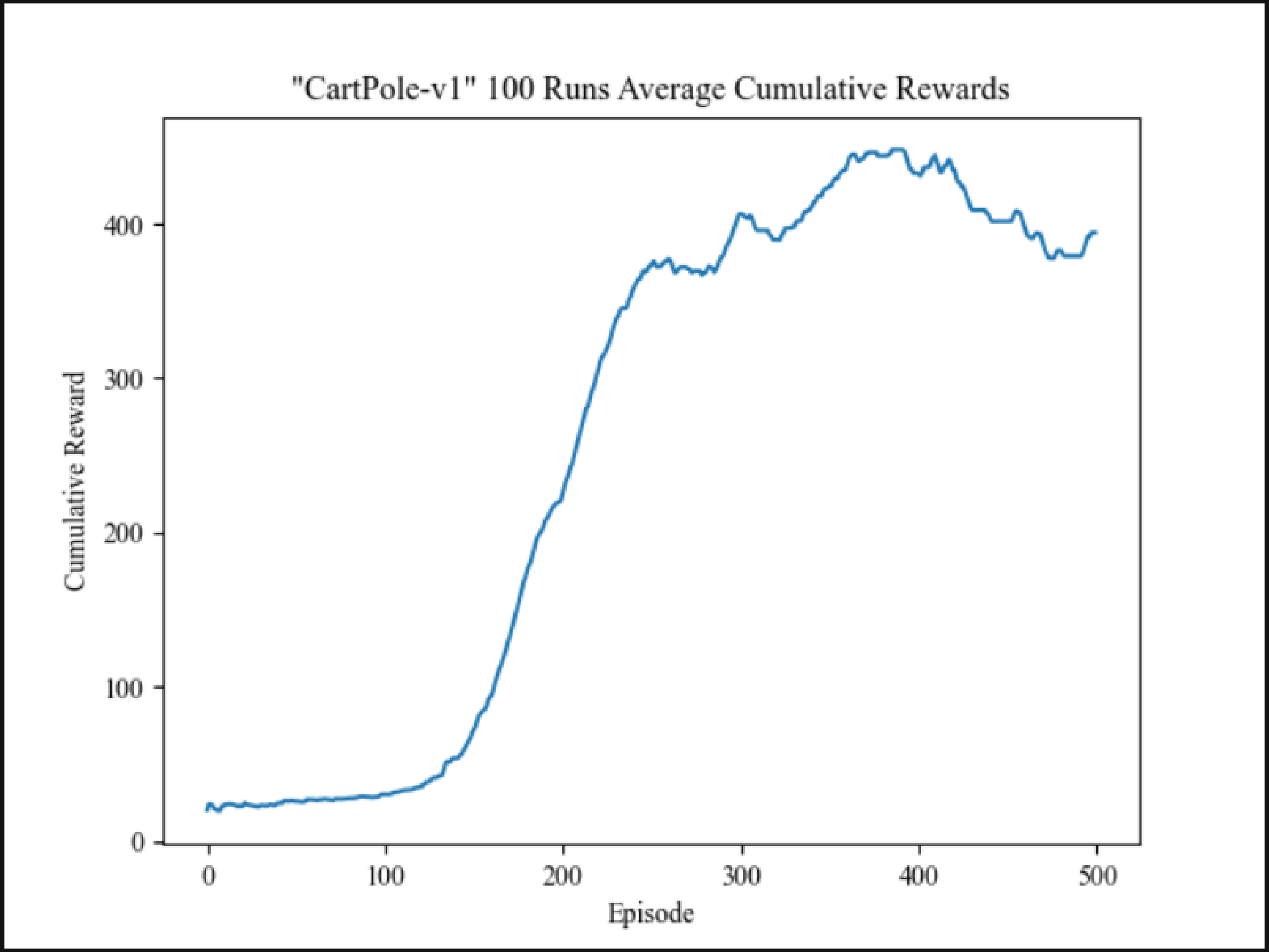
Set $s_{\boldsymbol{t+1}}$ *as current state*

end for

end for

Deep Q-Learning Network Example

- "CartPole-v1" Gym Environment:



Double Deep Q-Learning Network

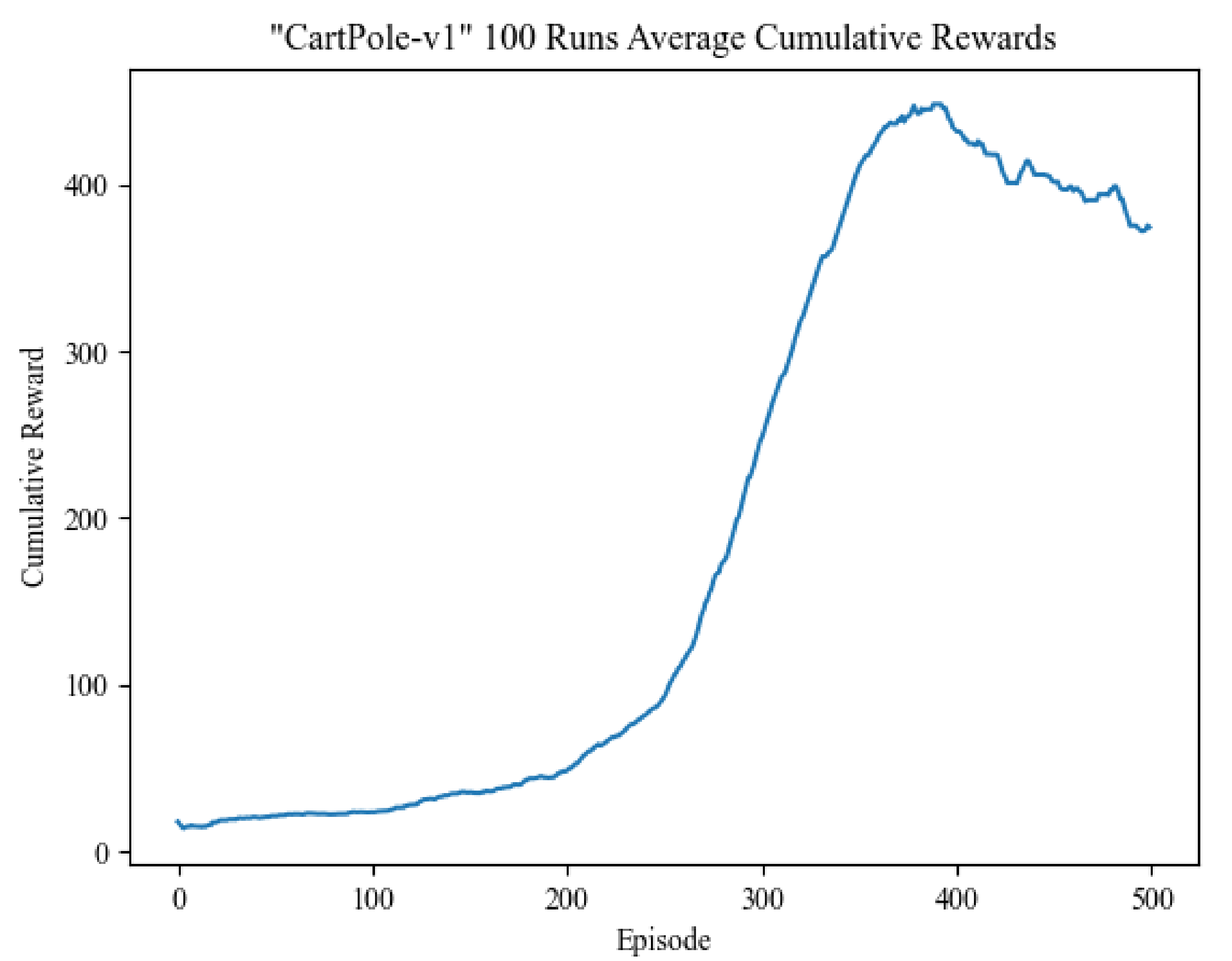
- Intuition:
  - *Estimate $Q_{\boldsymbol{\pi}}(s, a)$ via function approximation*
  - *Use two Q-networks, one frozen and the other online, to fix the TD-target*
  - *Utilize a replay buffer to store experience and train networks in randomized batches*
  - *Use the online and target Q-networks to handle maximization bias*

- Math:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[\left(r + \gamma \hat{Q}_{\boldsymbol{\theta}'}\left(s', \underset{a'}{\operatorname{argmax}} \hat{Q}_{\boldsymbol{\theta}}(s', a')\right) - \hat{Q}_{\boldsymbol{\theta}}(s, a)\right)^2\right]$$

Double Deep Q-Learning Network Example

- "CartPole-v1" Gym Environment:



TEXAS TECH UNIVERSITY SYSTEM™