Learning Atari with a Dueling Double DQN

By William Fraher

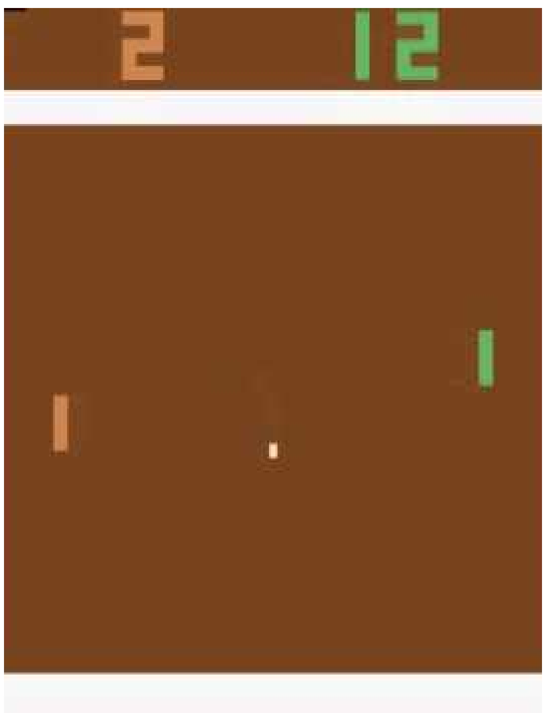
Goals of this project

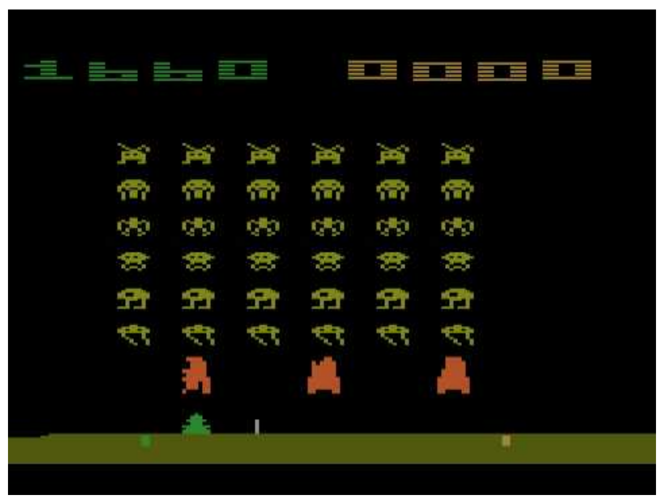
- AI and Atari

Goals of this project

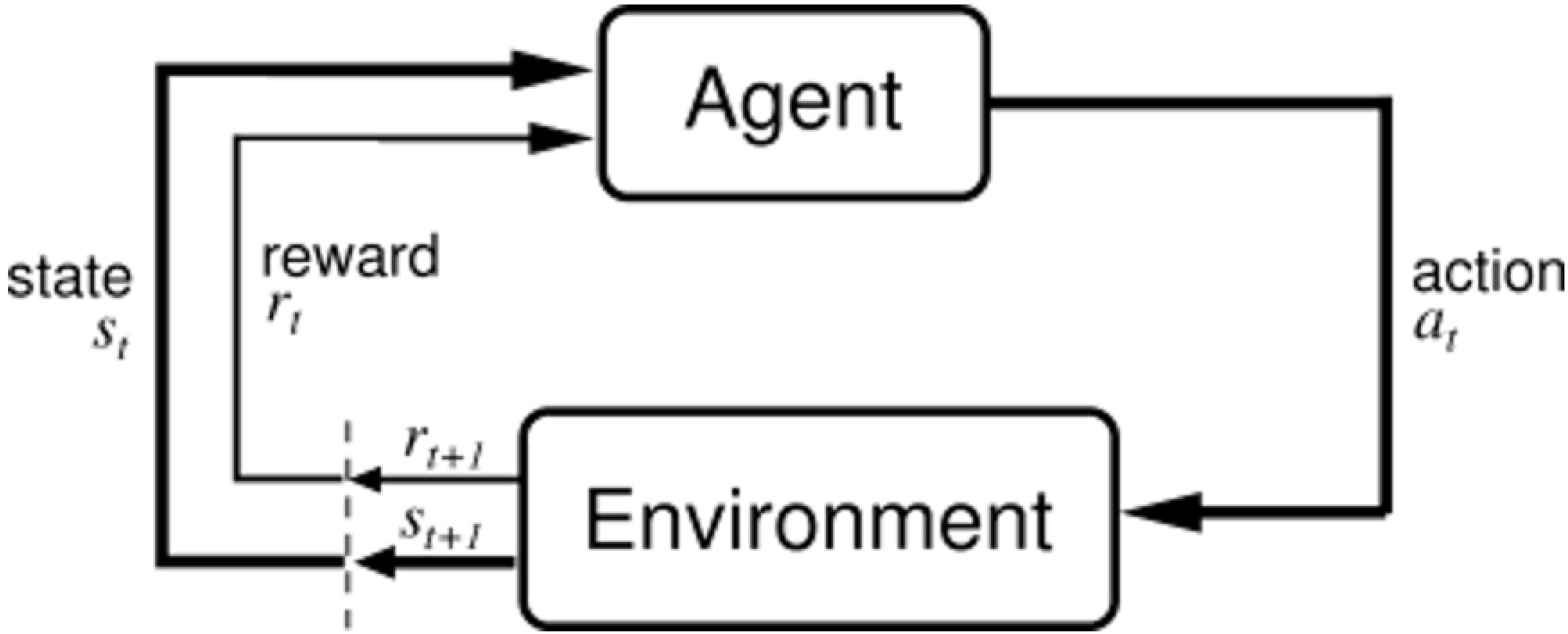
- AI and Atari

- Could one AI learn how to play any Atari game?





Reinforcement Learning



(Sutton and Barto, Reinforcement Learning: An Introduction, 1998)

We want to find a policy π_θ which maps states to actions. These actions ideally maximize future rewards.

Reinforcement Learning

- π_θ maps from states to actions given parameters θ.

- In reinforcement learning, our goal is to find the best θ.

Reinforcement Learning

- π_θ maps from states to actions given parameters θ.

- In reinforcement learning, our goal is to find the best θ.

- We collect rollouts, or episodes in our environment, of S_t, a_t, r_t, S_{t+1} transitions.

Reinforcement Learning

- How do we actually find θ?

Reinforcement Learning

- How do we actually find θ?

- Each S_t, a_t, r_t, S_{t+1} transition contains a reward. We want to maximize

$$\sum_{t'=t}^{\infty} r^{t'}$$

Reinforcement Learning

- In some cases, we can just use the reward.

Reinforcement Learning

- In some cases, we can just use the reward.
- One such algorithm is CEM.

<u>CEM:</u>

Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^d_{>0}$

for iteration = 1, 2, ...

Sample n parameters $\theta_i \sim N(\mu, \mathrm{diag}(\sigma^2))$

For each θ_i, perform one rollout to get return $R(\tau_i)$

Select the top k% of θ, and fit a new diagonal Gaussian to those samples. Update μ, σ

endfor

(image from Berkeley Deep RL Bootcamp, Peter Chen, John Schulman, Pieter Abbeel)

Reinforcement Learning

- We could use supervised learning.

Reinforcement Learning

- We could use supervised learning.
- Supervised learning problem: Map states to actions.
- Supervised learning techniques: Linear regression, neural nets.

Neural Nets

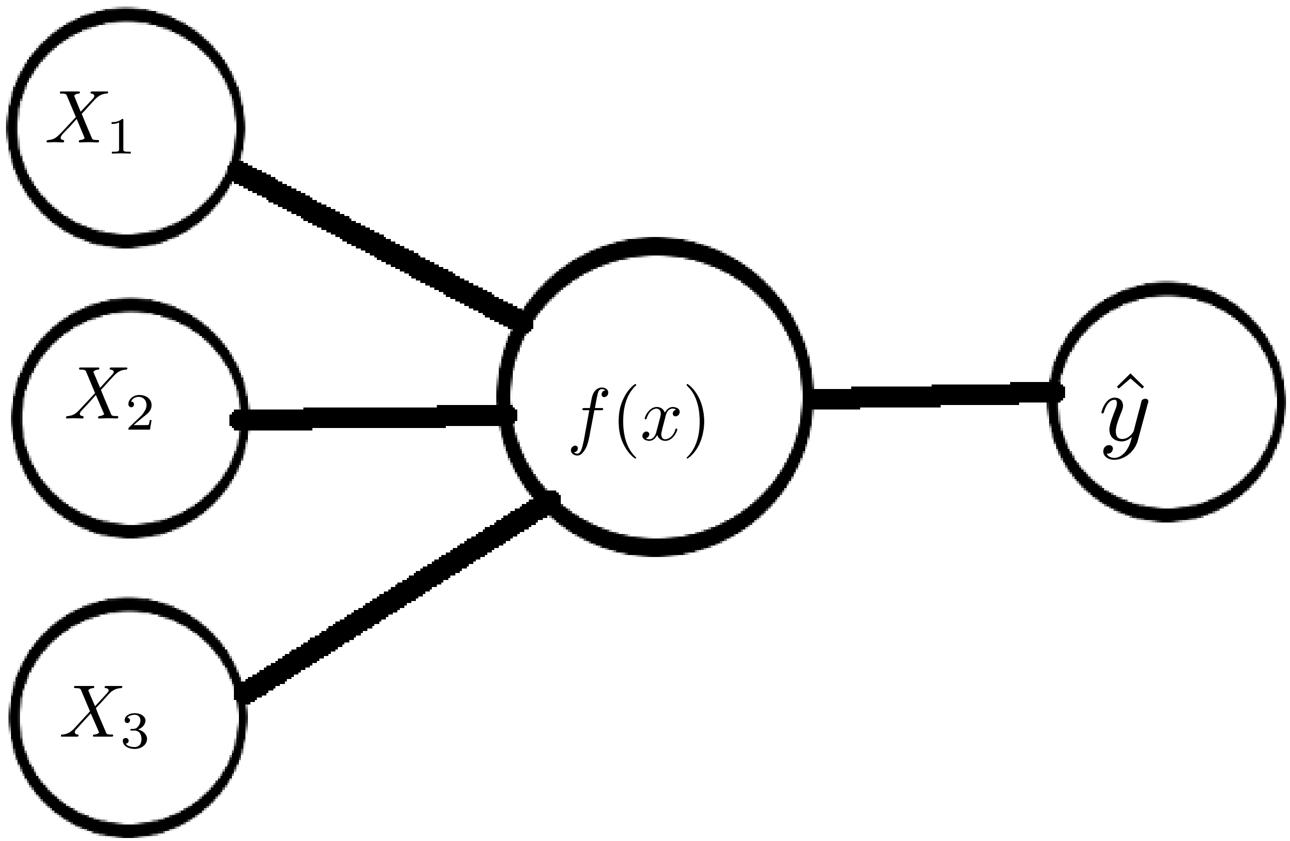
- Extremely sophisticated supervised learning technique.

Neural Nets

- Extremely sophisticated supervised learning technique.

- Uses training data of $(x_{(i)}, y_{(i)})$ pairs. Attempts to find a function that maps new x values to their y values.

Neural Nets



Neural Nets

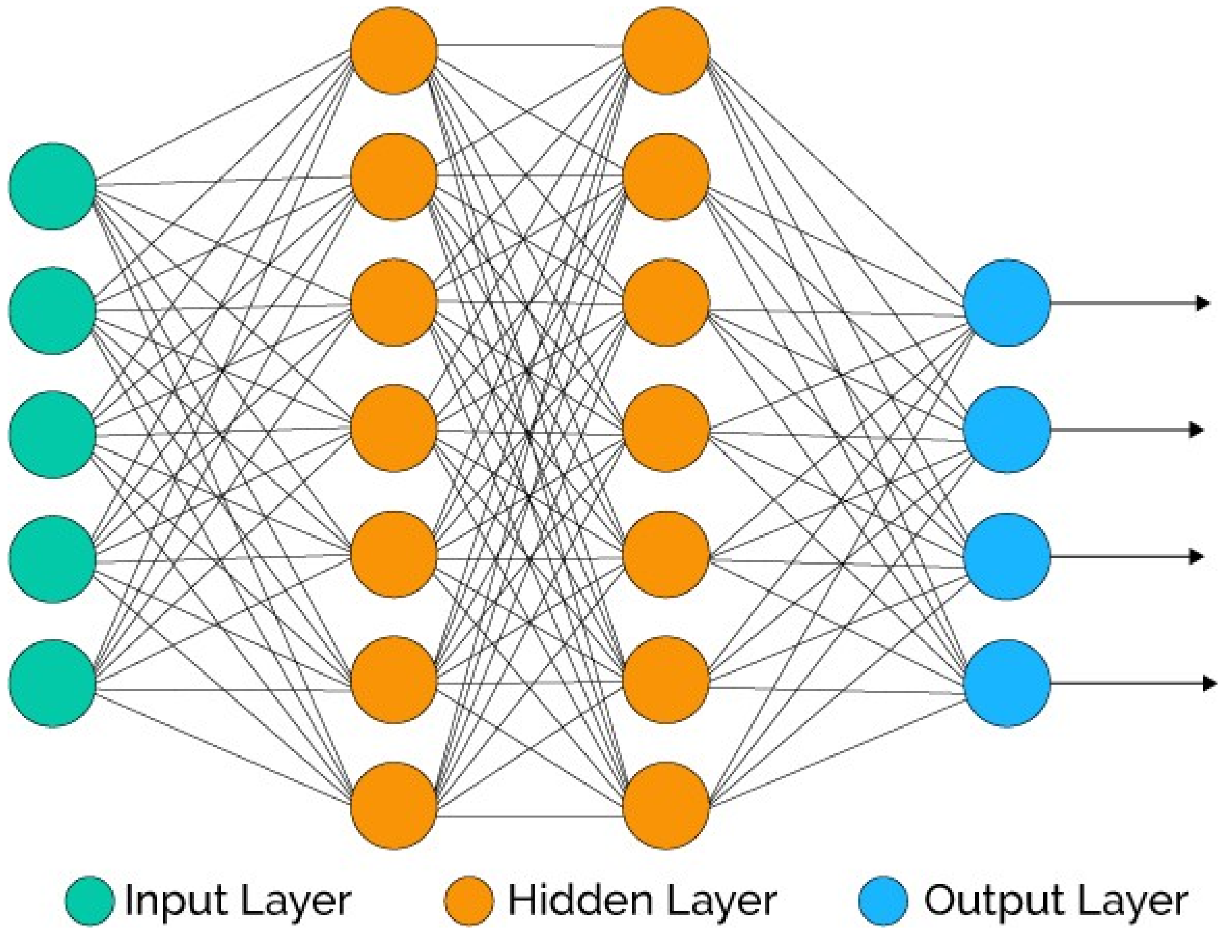
- $f(x)$ computed with a weight vector.

Neural Nets

- $f(x)$ computed with a weight vector.
- $\hat{y}$: Take a linear combination of weights and inputs.

$$[w_1, w_2, w_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \quad f(x)$$

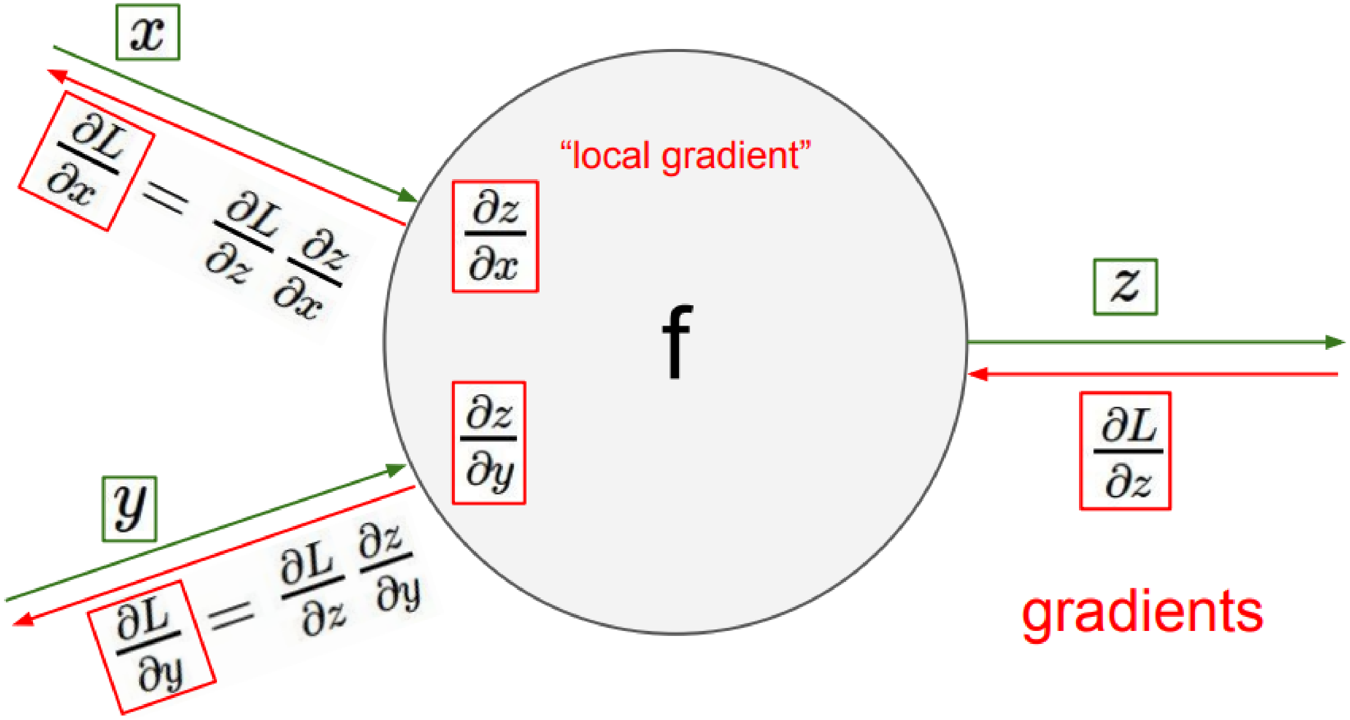
Neural Nets



(image from medium.com)

Neural Nets

- Uses a procedure called backpropagation to tune itself. This is why we need outputs.



(image from Stanford CS231 course)

Neural Nets

- Computes a hypothesis, we need an error to tune it accordingly.

Neural Nets

- Computes a hypothesis, we need an error to tune it accordingly.

$$(\hat{y}_i - y_i)^2$$

- Mean squared error is one way to do this. We tune the weights of our neural network with respect to this error.

Neural Nets

- How do we create training examples?

Neural Nets

- How do we create training examples?

- In reinforcement learning, we try to estimate rewards.

Deep Q-Learning

- Q-Learning: An algorithm which approximates the value of state-action pairs (Watkins, 1989). Works by building a table of state-action pairs.

Deep Q-Learning

- Q-Learning: An algorithm which discovers the value of state-action pairs (Watkins, 1989). Works by building a table of state-action pairs.

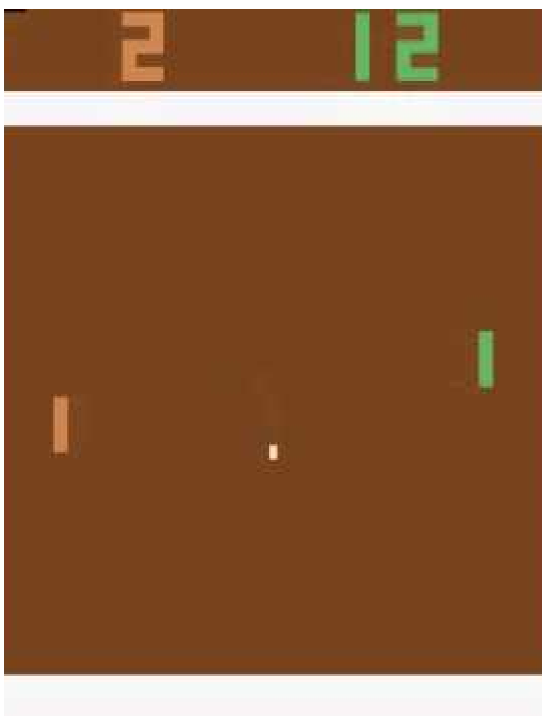
- Approximate Q-Learning: Uses a function approximation to estimate the value of state-action pairs.

Deep Q-Learning

- Q-Learning: An algorithm which discovers the value of state-action pairs (Watkins, 1989). Works by building a table of state-action pairs.
- Approximate Q-Learning: Uses a function approximation to estimate the value of state-action pairs.
- Deep Q-Learning: Uses a multi-layer neural network to approximate the value of state-action pairs.

Deep Q-Learning

- Given a state, we approximate the value of actions.



$$\begin{bmatrix} 0.1 \\ 0.3 \\ 0.6 \end{bmatrix}$$

Deep Q-Learning

- The values of state-action pairs are called Q-values.

- This is written as $Q(S, a)$, the Q-value of action a at state S.

Deep Q-Learning

- Instead of having a given y_i as in supervised learning, we need to specify a target.

Deep Q-Learning

- Instead of having a given y_i as in supervised learning, we need to specify a target.

- In Deep Q-learning, we tune the network with respect to the following target:

$$r + \gamma * max_a Q(S, a)$$

- Where γ is a discount factor.

Deep Q-Learning

- When training the network, we use the following loss:

$$(r_t + \gamma * max_{a_{t+1}} Q(S_{t+1}, a_{t+1}) - Q(S_t, a_t))^2$$

- This operates over a single transition, and gives us everything we need to test it.

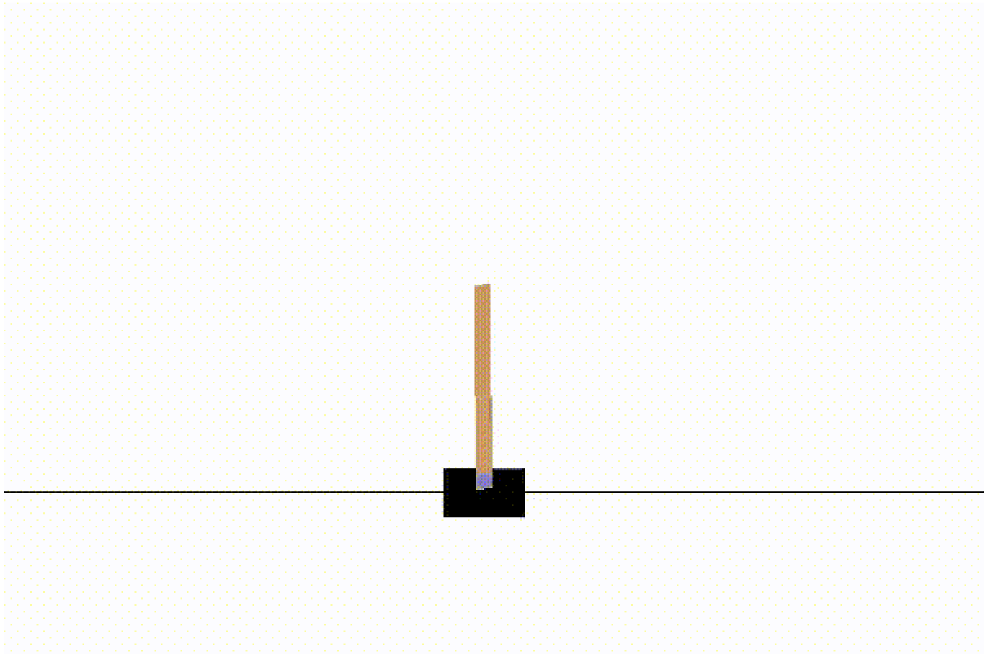
OpenAI Gym

- OpenAI Gym: Used to benchmark RL algorithms.
- We will experiment with the pole balancing problem.

OpenAI Gym

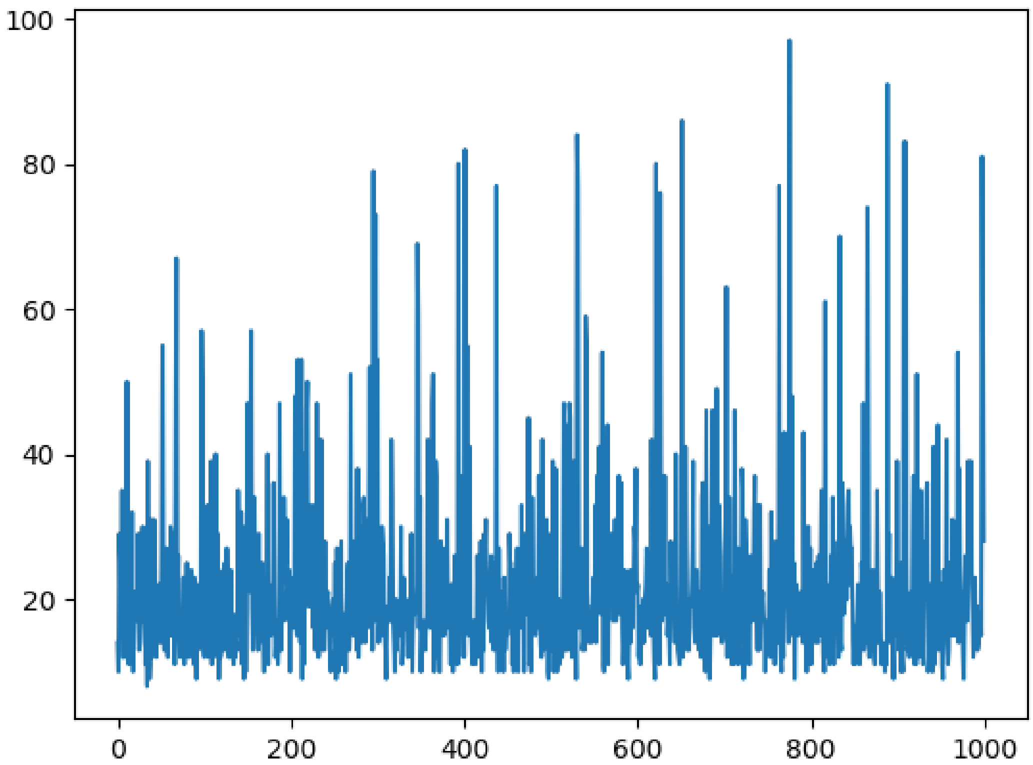
- OpenAI Gym: Used to benchmark RL algorithms.

- We will experiment with the pole balancing problem.



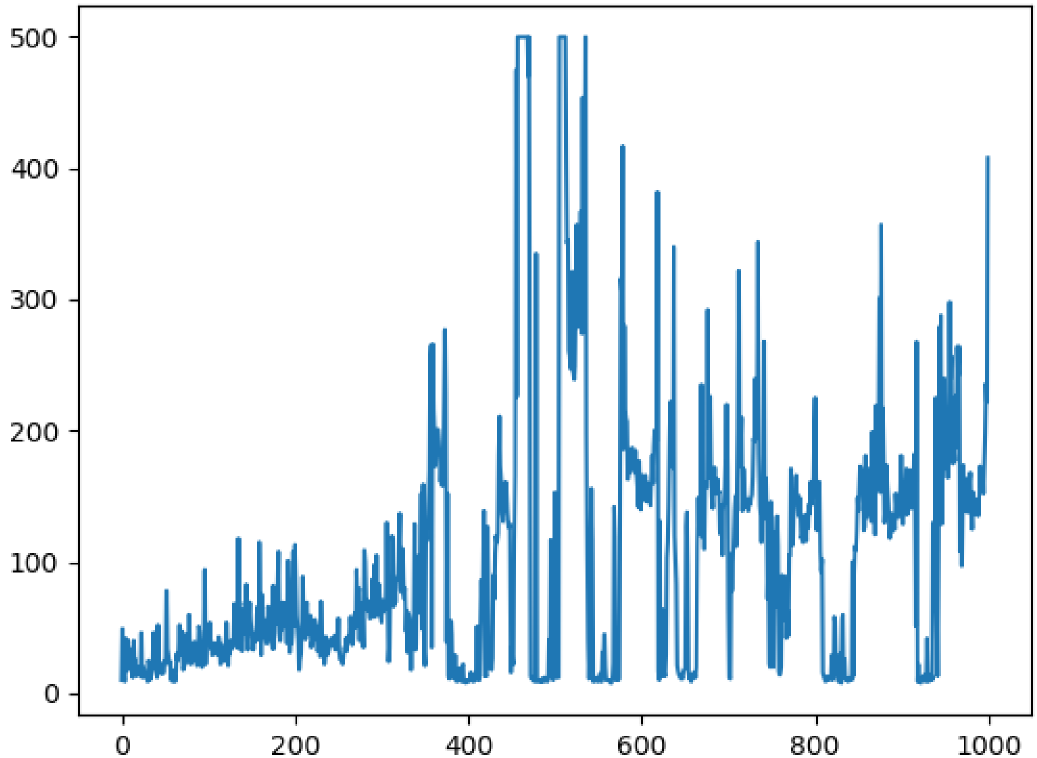
OpenAI Gym

- An agent that acts randomly gets the following scores:



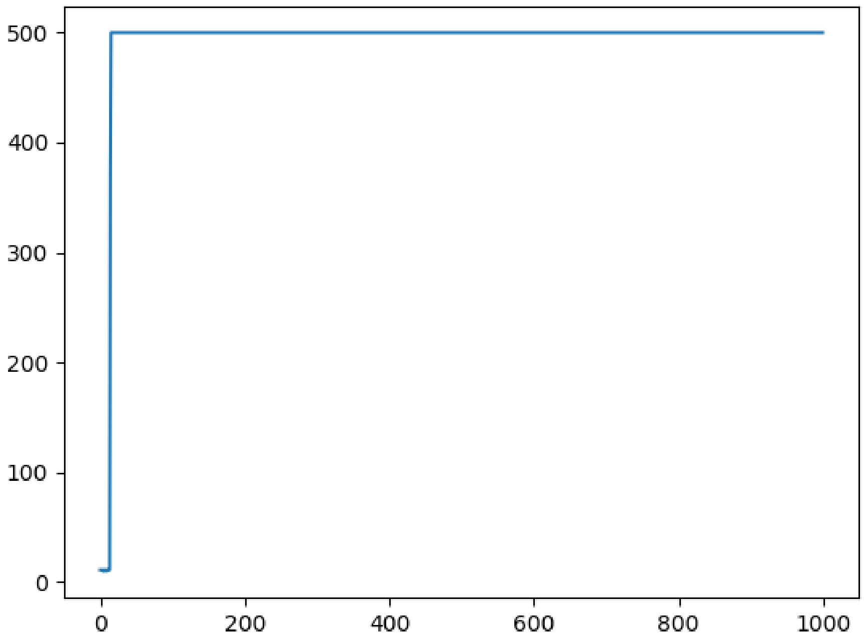
OpenAI Gym

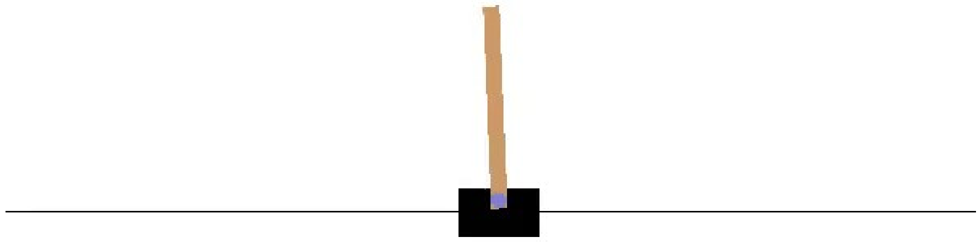
- A neural network using Q-Learning gets the following scores:



OpenAI Gym

- CEM, while simpler, is better here.





OpenAI Gym

- Why use Q-Learning?

OpenAI Gym

- Why use Q-Learning?

- Deep Q-Learning might be sub-optimal here, but CEM cannot handle large parameter vectors.

OpenAI Gym

- Why use Q-Learning?
- Deep Q-Learning might be sub-optimal here, but CEM cannot handle large parameter vectors.
- To learn Atari, we will need to use Deep Q-Learning with several optimizations.

Experience Replay

- Each S_t, a_t, r_t, S_{t+1} transition is stored in a memory buffer and sampled at random.

Experience Replay

- Each S_t, a_t, r_t, S_{t+1} transition is stored in a memory buffer and sampled at random.

- A variation of this is called Prioritized Experience Replay, which selects transitions by error.

Target Network

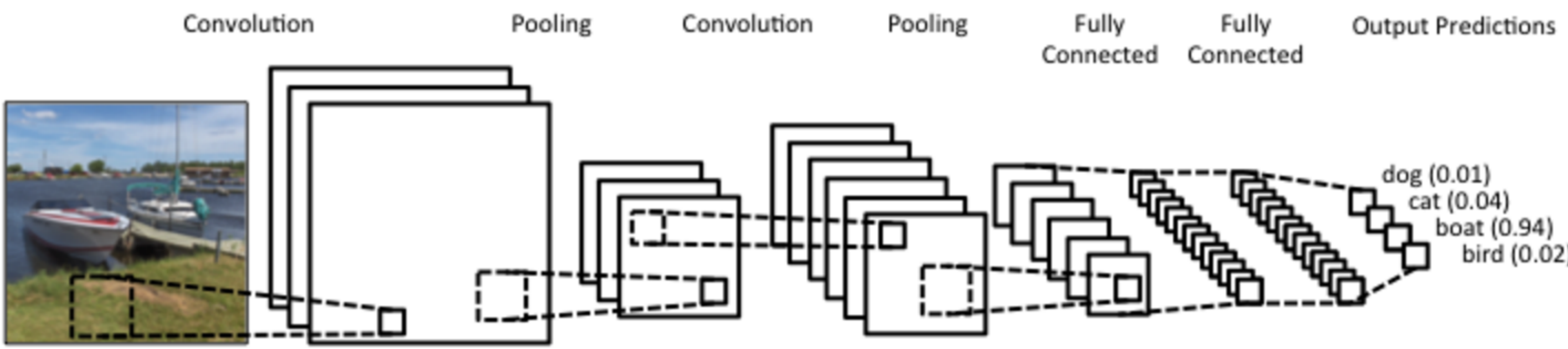
- We use the following loss

$$(r_t + max_{a_{t+1}} Q(S_{t+1}, a_{t+1}; \theta^-) - Q(S_t, a_t; \theta))^2$$

- θ^- is a second network used to evaluate the next state.
- We update the second network every $\mathcal{T}$ training steps.

Convolutional Neural Networks

- Instead of multiplying matrices, we could convolve over them to generalize spatial information.



(image from WildML.com)

Double DQN

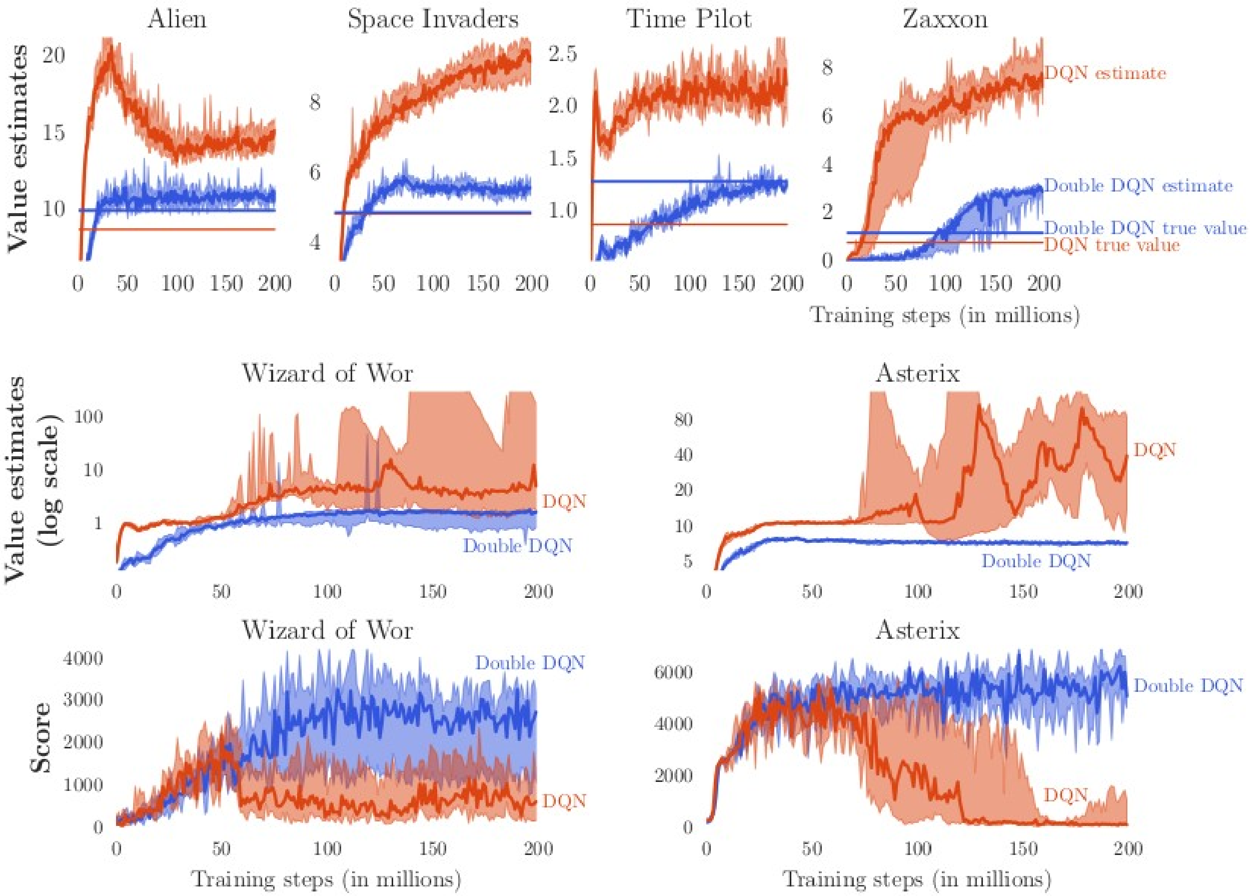
- Instead of taking the max Q-value from our second network, we decouple the $\max\limits_{a_{t+1}}$ step.

$$(r_t + Q(S_{t+1}, argmax_{a_{t+1}} Q(S_{t+1}, a_{t+1}; \theta); \theta^-) - Q(S_t, a_t))^2$$

- This has been proven to increase stability of DQN (van Hasselt et al. 2015).

Double DQN

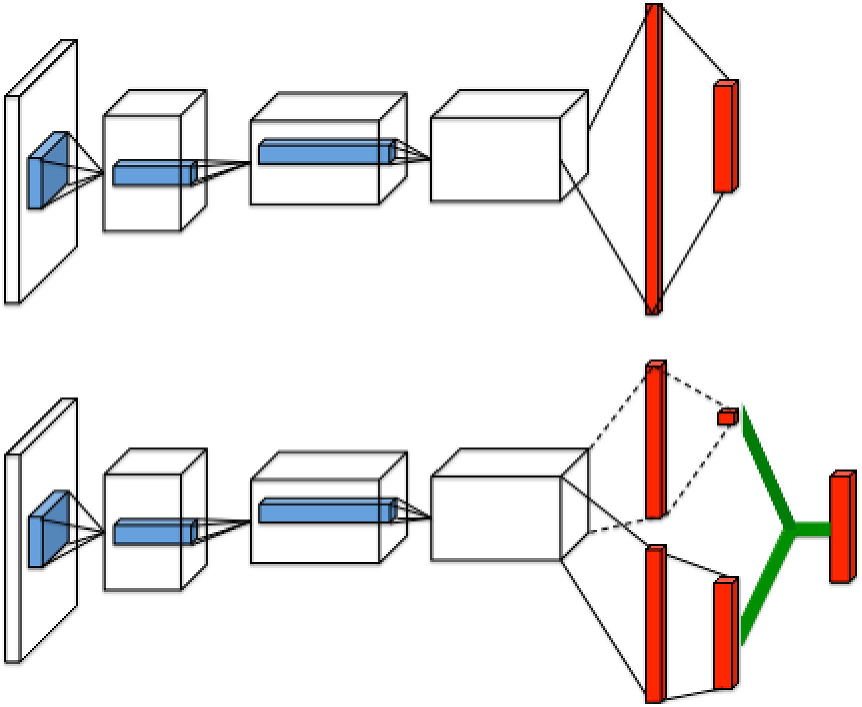
- Why does Double DQN work?



(image from van Hasselt et al, 2015)

Dueling DQN

- Splits the network's output into two streams.



(image from Wang et al. 2015)

Dueling DQN

- By splitting the network into two streams, we estimate the value of the state and the action separately.

$$Q^{\pi}(S_t, a_t) = V^{\pi}(S_t) + A^{\pi}(S_t, a_t)$$

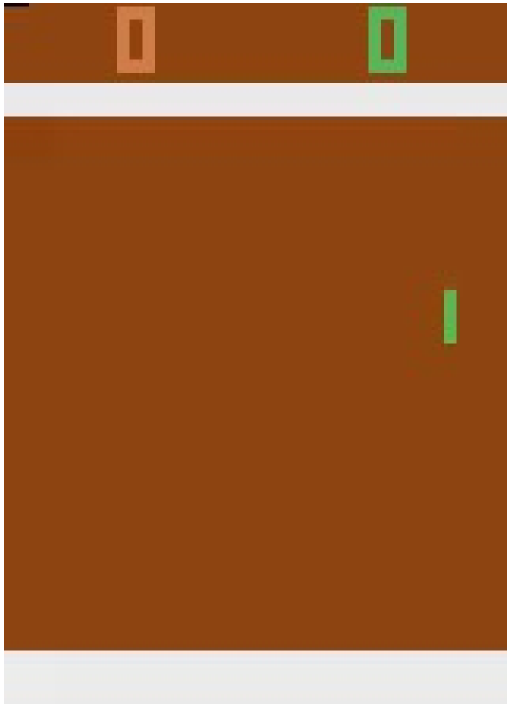
- Thus the new Q-value is the sum of the state value and action value for that state-action pair.

Dueling Double DQN

- Combining these techniques allows us to learn Atari games.

Dueling Double DQN

- Combining these techniques allows us to learn Atari games.



(DQN agent is on the right)

It works!

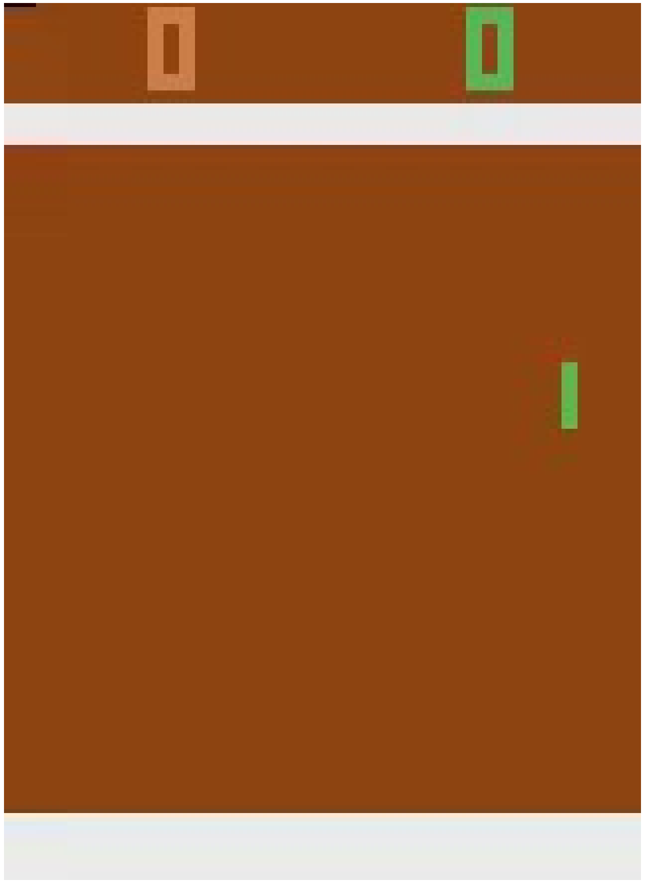
- I claim this works for other Atari games.

It works!

- I claim this works for other Atari games.
- Proof:

It works!

- I claim this works for other Atari games.
- Proof:



Possible Enhancements

- Prioritized experience replay: Selects transitions based of TD error

$$(r_t + Q(S_{t+1}, argmax_{a_{t+1}} Q(S_{t+1}, a_{t+1}; \theta); \theta^-) - Q(S_t, a_t))^2$$

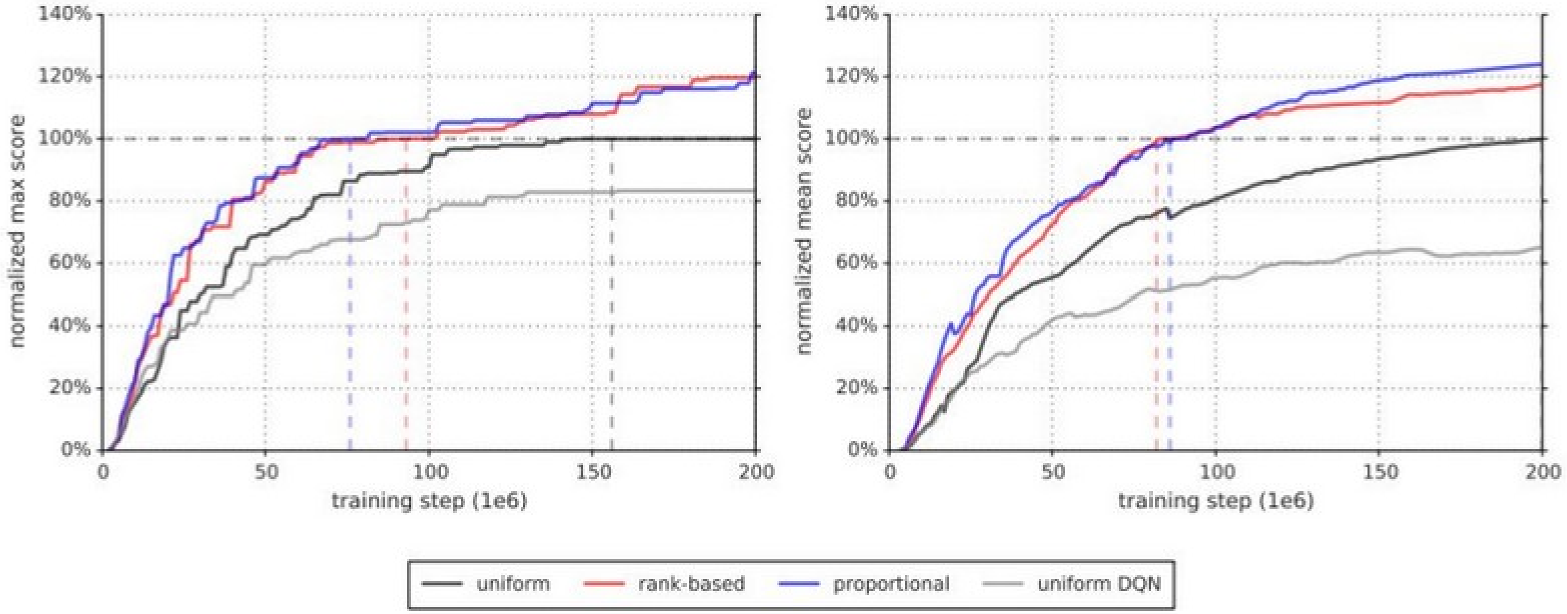
Possible Enhancements

- Prioritized experience replay: Selects transitions based of TD error

$$(r_t + Q(S_{t+1}, argmax_{a_{t+1}} Q(S_{t+1}, a_{t+1}; \theta); \theta^-) - Q(S_t, a_t))^2$$

- Dramatically increases learning speed.

Possible Enhancements



	DQN		Double DQN (tuned)		
	baseline	rank-based	baseline	rank-based	proportional
Median	48%	106%	111%	113%	128%
Mean	122%	355%	418%	454%	551%
> baseline	–	41	–	38	42
> human	15	25	30	33	33
# games	49	49	57	57	57

(images from Schaul et al. 2016)

Possible Enhancements

- A3C: Uses Q-Learning to tune an Actor-Critic network.

- Policy Gradient + Generalized Advantage Estimation:
  - Init π_{θ_0} $V^{\pi}_{\phi_0}$
  - Collect roll-outs {s, u, s', r} and $\hat{Q}_i(s, u)$
  - Update: $\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s,u,s',r)} \|\hat{Q}_i(s,u) - V^{\pi}_{\phi}(s)\|_2^2 + \kappa\|\phi - \phi_i\|_2^2$

$$\theta_{i+1} \leftarrow \theta_i + \alpha \frac{1}{m} \sum_{k=1}^{m} \sum_{t=0}^{H-1} \nabla_\theta \log \pi_{\theta_i}(u_t^{(k)}|s_t^{(k)}) \left(\hat{Q}_i(s_t^{(k)}, u_t^{(k)}) - V^{\pi}_{\phi_i}(s_t^{(k)})\right)$$

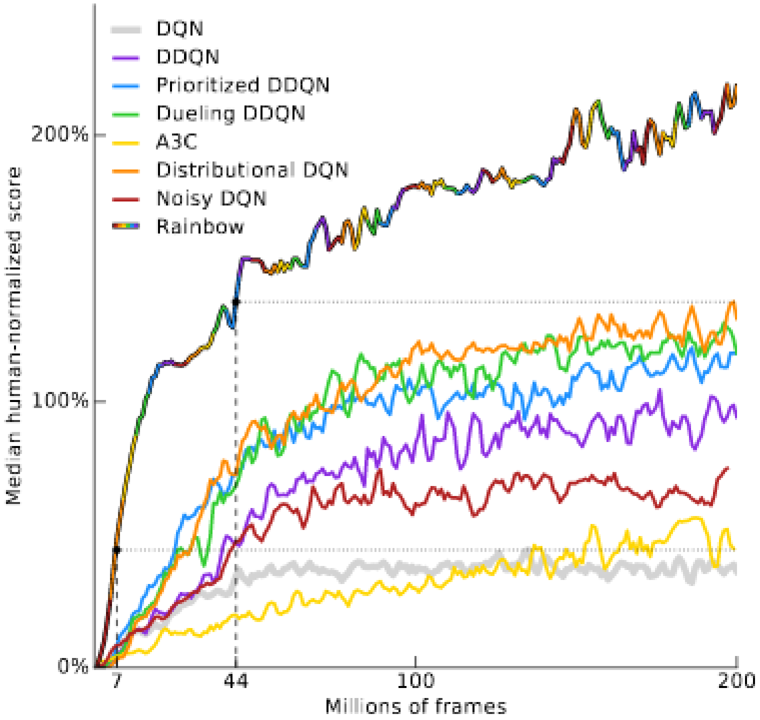
(image from Berkeley Deep RL Bootcamp, Pieter Abbeel, John Schulman, Peter Chen)

Possible Enhancements

(video from Google Deepmind)

Possible Enhancements

- Rainbow DQN

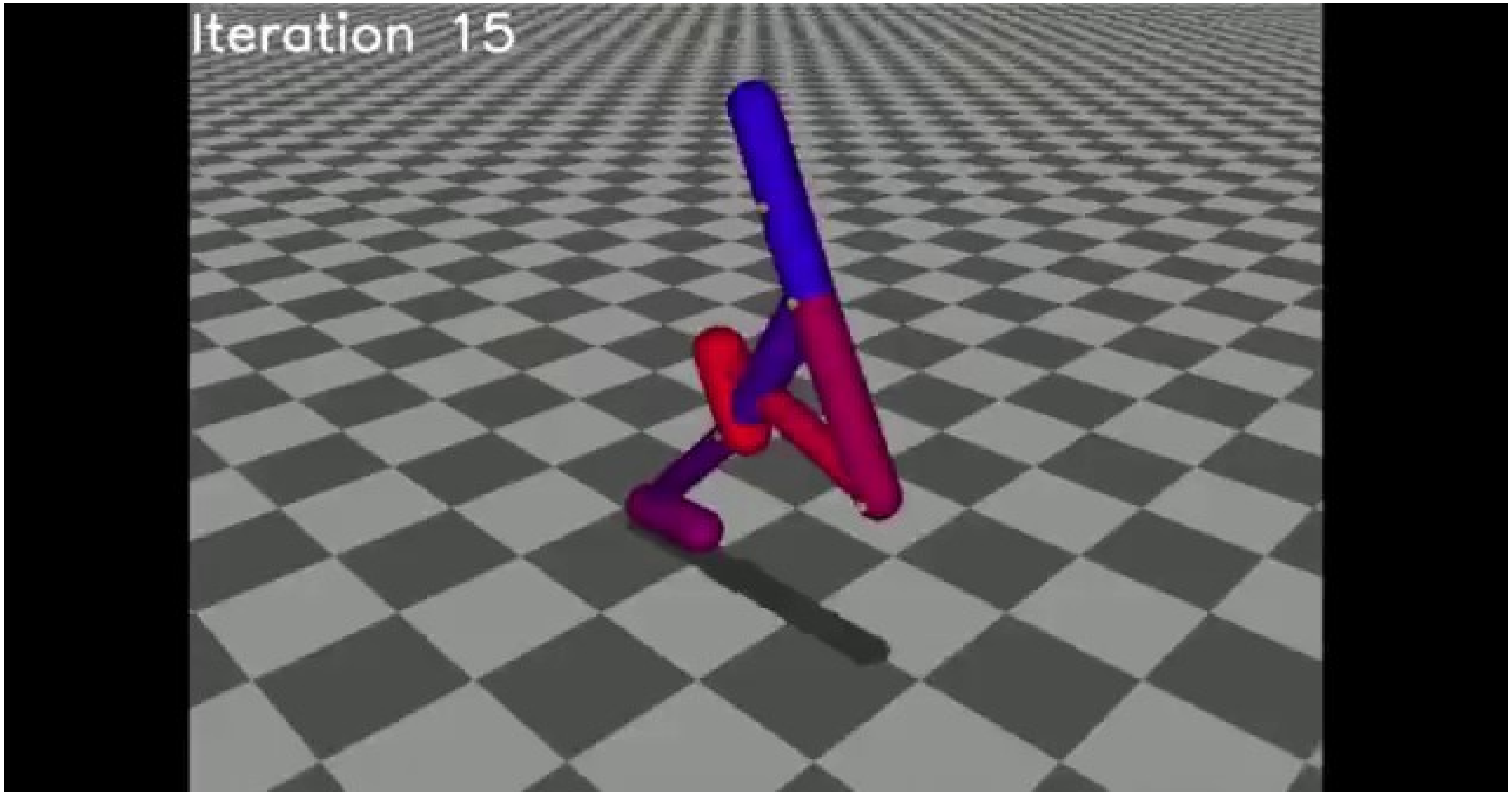


(image from Hessel et al. 2017)

Why do I care?

Why do I care?

- ROBOTS!!!



(Trust region policy optimization)

Questions?

Resources

- https://wfraher.github.io/
- UC Berkeley – CS188
- David Silver – Reinforcement Learning
- Andrew Ng - Coursera