# Lecture 10: More on Control
# Sarsa, Q-learning, DQN

# Recall: action-value functions

- They have a very strong relationship to $v_\pi$

- $v_\pi =$ np.einsum('sa, sa -> s', $\pi$, $q_\pi$)

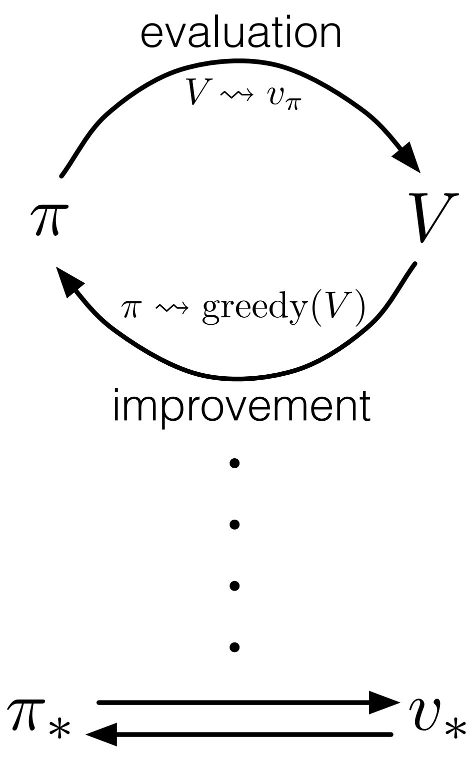
- (Or $v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$)

- $q_\pi = r + \gamma P v_\pi$

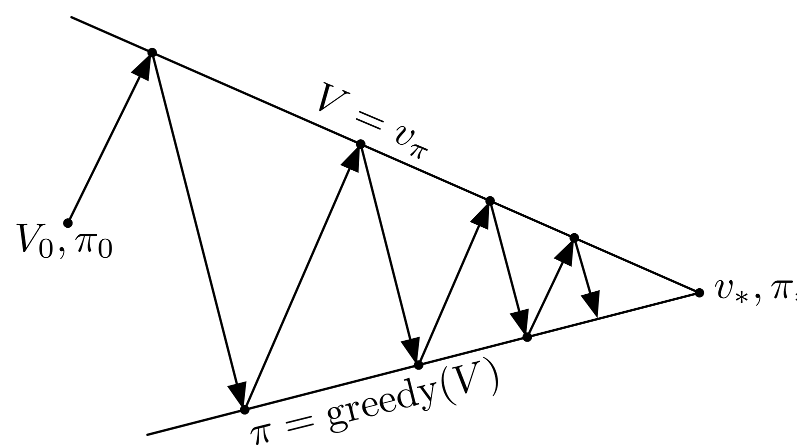
- All contraction arguments still apply

# Recall: Generalized Policy Iteration

**Generalized Policy Iteration** (GPI):
any interaction of policy evaluation and policy improvement, independent of their granularity.




A geometric metaphor for convergence of GPI:



# Recall: Policy Improvement

Suppose we have computed $v_\pi$ for a deterministic policy π.

For a given state $s$,
would it be better to do an action $a \neq \pi(s)$?

It is better to switch to action $a$ for state $s$ if

$$q_\pi(s,a) > v_\pi(s)$$

# Recall: On-policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \leftarrow$ arbitrary

$Returns(s,a) \leftarrow$ empty list

$\pi(a|s) \leftarrow$ an arbitrary $\varepsilon$-soft policy

Repeat forever:

(a) Generate an episode using $\pi$

(b) For each pair $s, a$ appearing in the episode:

$G \leftarrow$ return following the first occurrence of $s, a$

Append $G$ to $Returns(s,a)$

$Q(s,a) \leftarrow \text{average}(Returns(s,a))$

(c) For each $s$ in the episode:

$A^* \leftarrow \arg\max_a Q(s,a)$

For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

# Recall: Sarsa

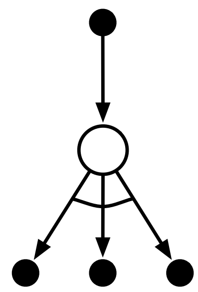
Turn this into a control method by always updating the policy to be greedy with respect to the current estimate:

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize $S$
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Repeat (for each step of episode):
        Take action $A$, observe $R$, $S'$
        Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until $S$ is terminal

# Q-Learning: Off-Policy TD Control

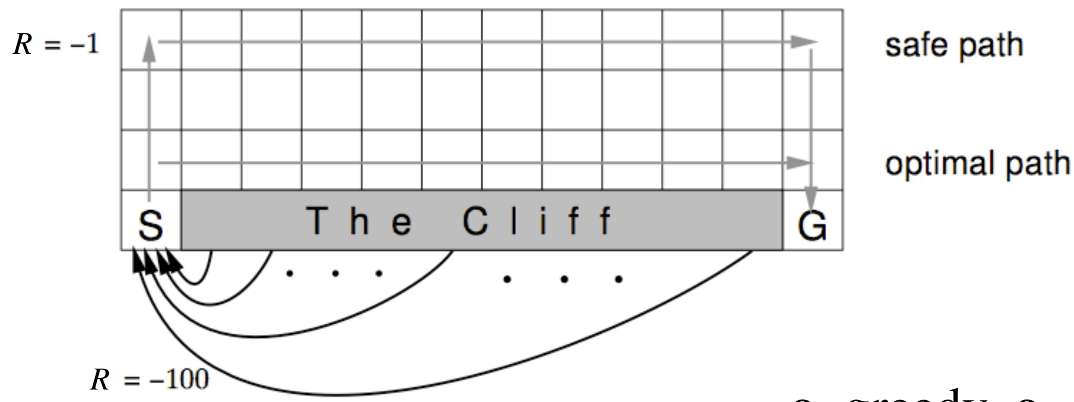
One-step Q-learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

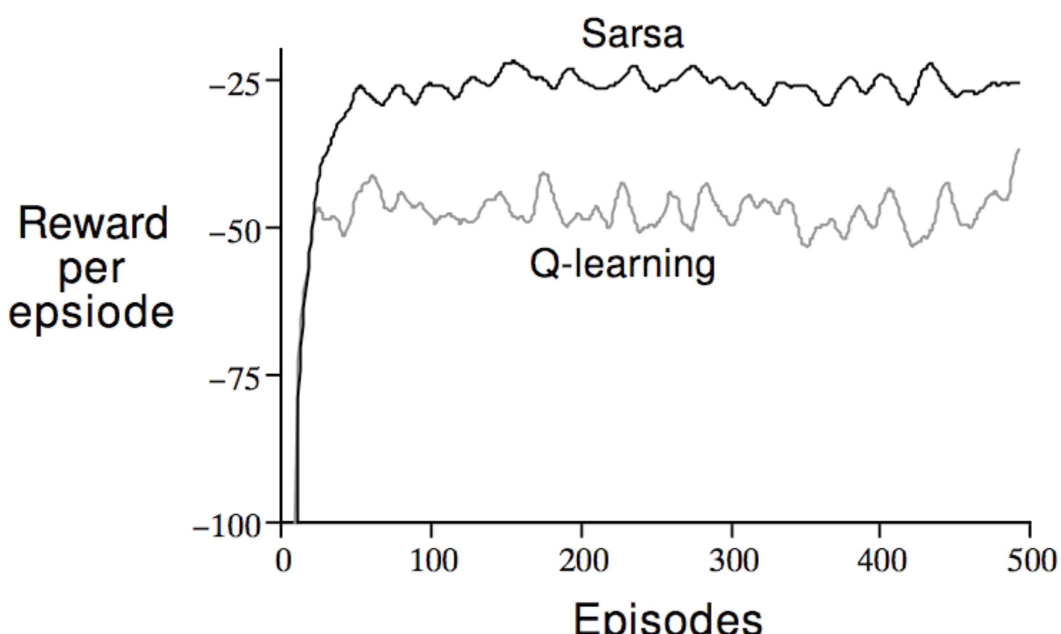


Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize $S$
  Repeat (for each step of episode):
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$;
  until $S$ is terminal

# Cliffwalking



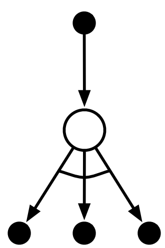

ε−greedy, ε = 0.1

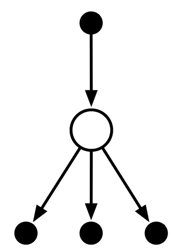


# Expected Sarsa

- Instead of the *sample* value-of-next-state, use the expectation!

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma\mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big]$$
$$\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma\sum_a \pi(a|S_{t+1})Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$


Q-learning

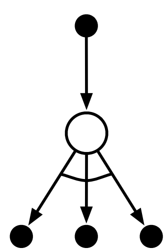

Expected Sarsa

- Expected Sarsa's performs better than Sarsa (but costs more)

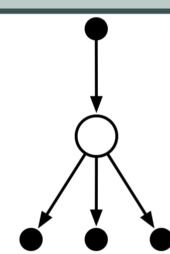
# *Off-policy* Expected Sarsa

- Expected Sarsa generalizes to arbitrary behaviour policies $\mu$
  - in which case it includes Q-learning as the special case in which π is the greedy policy

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[ R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&\leftarrow Q(S_t, A_t) + \alpha \Big[ R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big]
\end{aligned}
$$

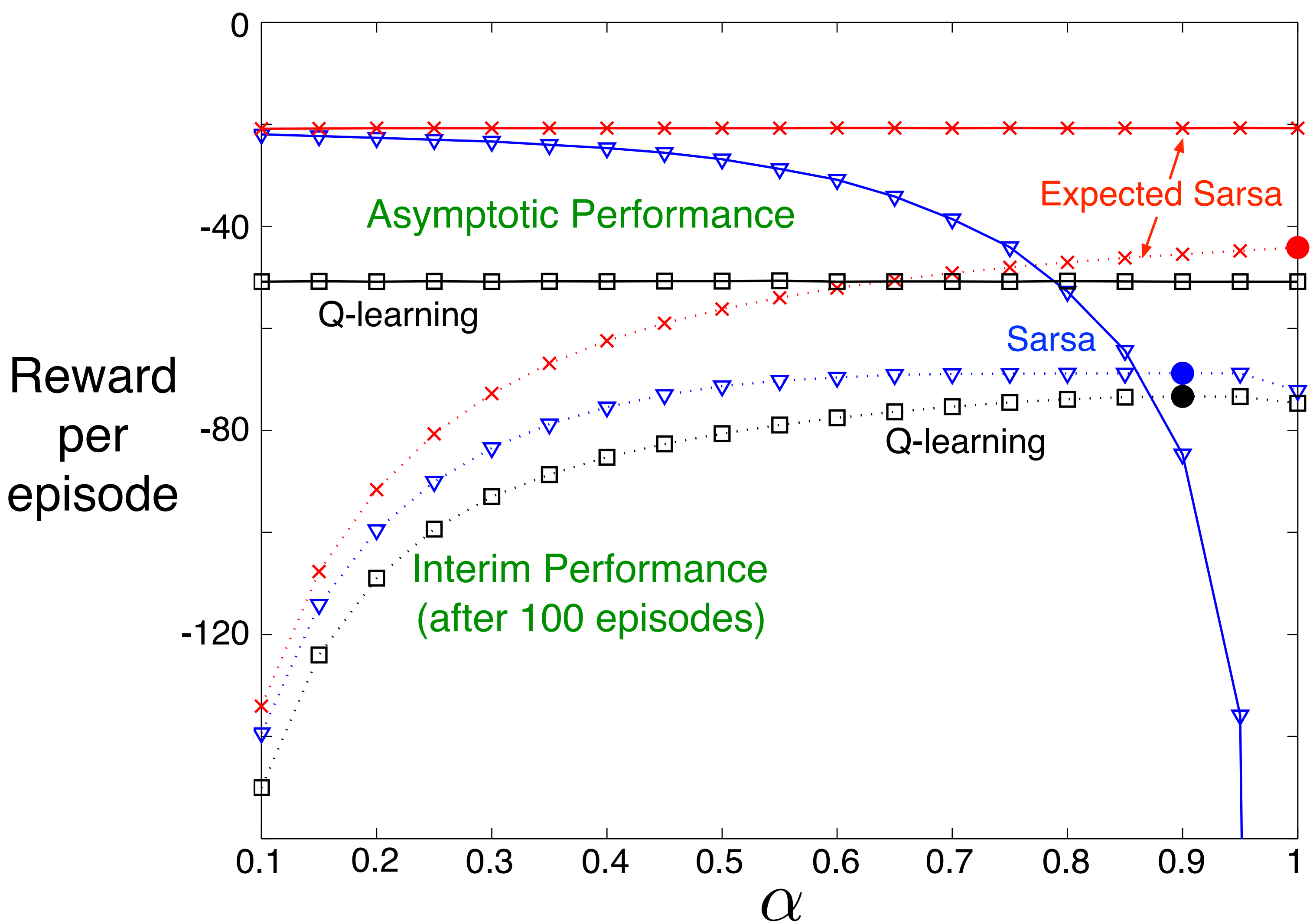


Q-learning

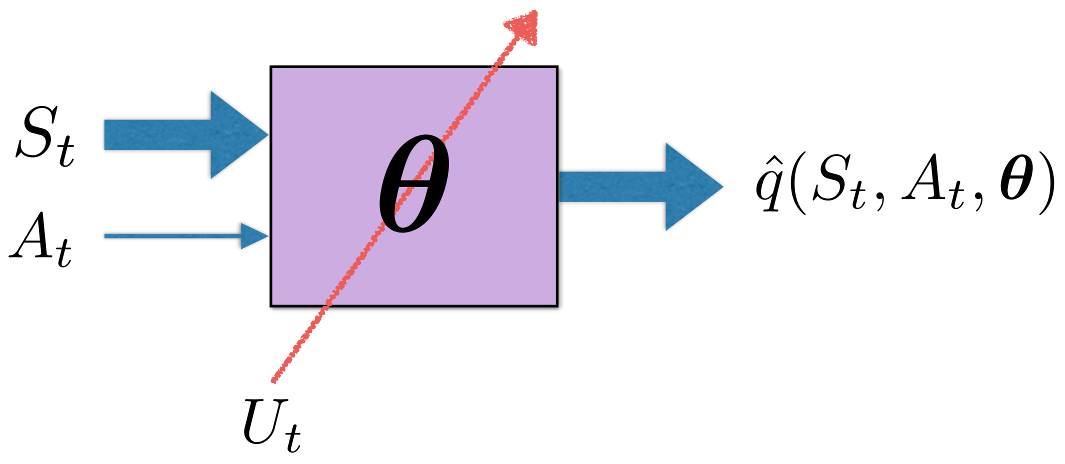


Expected Sarsa

# Performance on the Cliff-walking Task



# Value function approximation (VFA) for control



# On-Policy GPI

- Always learn the action-value function of the current policy
- Always act near-greedily wrt the current action-value estimates
- The learning rule is:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big[U_t - \hat{q}(S_t, A_t, \boldsymbol{\theta}_t)\Big]\nabla\hat{q}(S_t, A_t, \boldsymbol{\theta}_t)$$

update target, e.g. $U_t = G_t$ (MC)

$U_t = R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{\theta}_t)$ (Sarsa)

$U_t = R_{t+1} + \gamma\sum_a \pi(a|S_{t+1})\hat{q}(S_{t+1}, a, \boldsymbol{\theta}_t)$ (Expected Sarsa)

$U_t = \sum_{s',r} p(s', r|S_t, A_t)\Big[r + \gamma\sum_{a'} \pi(a'|s')\hat{q}(s', a', \boldsymbol{\theta}_t)\Big]$ (DP)

# Example: Sarsa

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big[U_t - \hat{q}(S_t, A_t, \boldsymbol{\theta}_t)\Big]\nabla\hat{q}(S_t, A_t, \boldsymbol{\theta}_t)$$

**Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$**

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^n \to \mathbb{R}$

Initialize value-function weights $\boldsymbol{\theta} \in \mathbb{R}^n$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
    $S, A \leftarrow$ initial state and action of episode (e.g., $\varepsilon$-greedy)
    Repeat (for each step of episode):
        Take action $A$, observe $R, S'$
        If $S'$ is terminal:
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R - \hat{q}(S, A, \boldsymbol{\theta})\big]\nabla\hat{q}(S, A, \boldsymbol{\theta})$
            Go to next episode
        Choose $A'$ as a function of $\hat{q}(S', \cdot, \boldsymbol{\theta})$ (e.g., $\varepsilon$-greedy)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R + \gamma\hat{q}(S', A', \boldsymbol{\theta}) - \hat{q}(S, A, \boldsymbol{\theta})\big]\nabla\hat{q}(S, A, \boldsymbol{\theta})$
        $S \leftarrow S'$
        $A \leftarrow A'$

# Example: n-step Sarsa

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}),$$

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \left[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})\right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}),$$

**Episodic semi-gradient $n$-step Sarsa for estimating $\hat{q} \approx q_*$ or $q_\pi$**

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy $\pi$ (if estimating $q_\pi$)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer $n$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations ($S_t$, $A_t$, and $R_t$) can take their index mod $n+1$

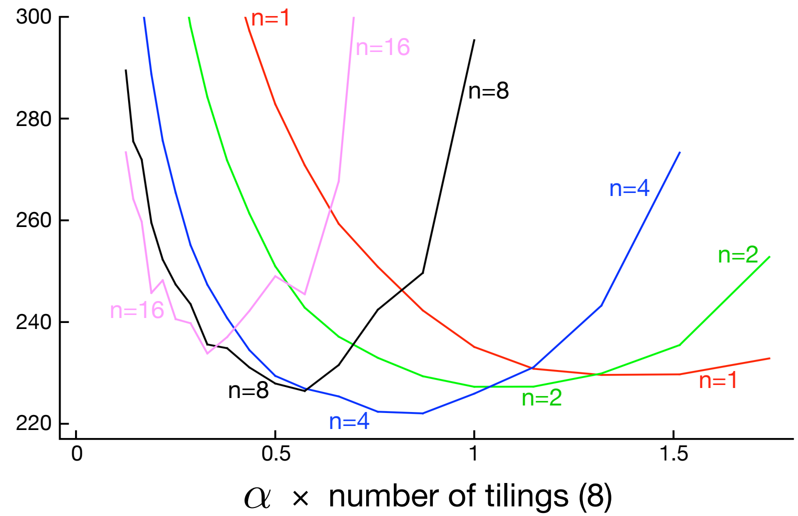
```
Loop for each episode:
    Initialize and store S_0 ≠ terminal
    Select and store an action A_0 ~ π(·|S_0) or ε-greedy wrt q̂(S_0, ·, w)
    T ← ∞
    Loop for t = 0, 1, 2, ... :
    |   If t < T, then:
    |       Take action A_t
    |       Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    |       If S_{t+1} is terminal, then:
    |           T ← t + 1
    |       else:
    |           Select and store A_{t+1} ~ π(·|S_{t+1}) or ε-greedy wrt q̂(S_{t+1}, ·, w)
    |   τ ← t − n + 1    (τ is the time whose estimate is being updated)
    |   If τ ≥ 0:
    |       G ← Σ_{i=τ+1}^{min(τ+n,T)} γ^{i−τ−1} R_i
    |       If τ + n < T, then G ← G + γ^n q̂(S_{τ+n}, A_{τ+n}, w)          (G_{τ:τ+n})
    |       w ← w + α [G − q̂(S_τ, A_τ, w)] ∇q̂(S_τ, A_τ, w)
    Until τ = T − 1
```

# n-step Sarsa is better for n>1

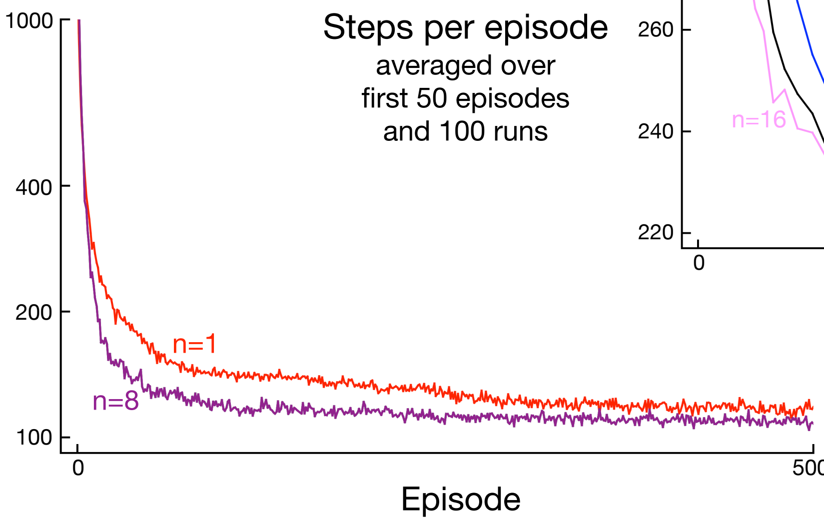
$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}),$$

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \left[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})\right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}),$$

Mountain Car
Steps per episode
averaged over
first 50 episodes
and 100 runs




Mountain Car
Steps per episode
log scale
averaged over 100 runs

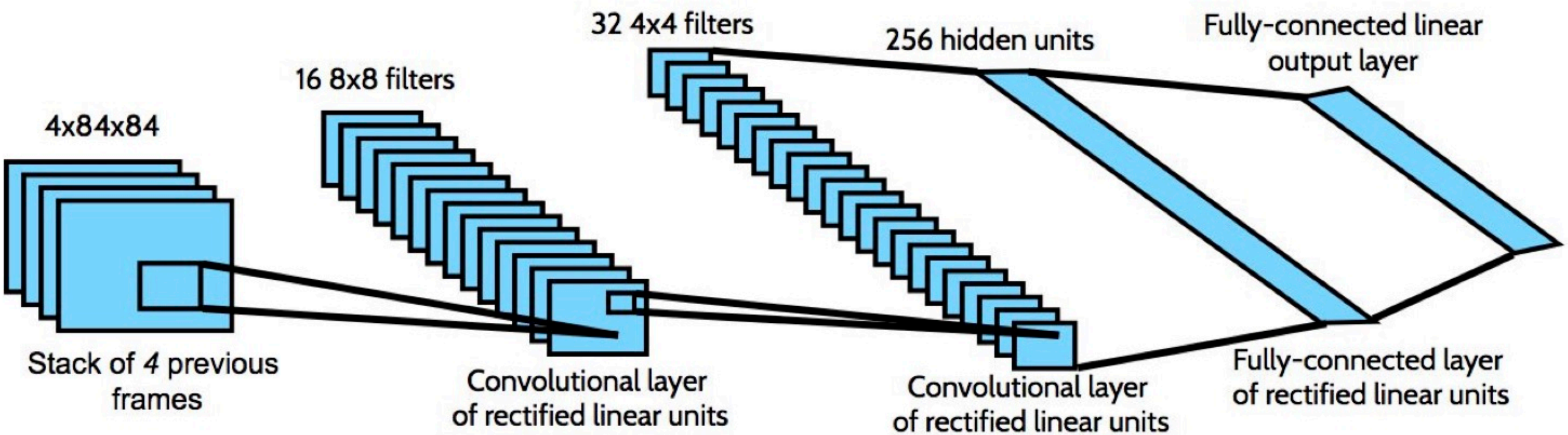


# DQN

(Mnih, Kavukcuoglu, Silver, et al., Nature 2015)

- Learns to play video games **from raw pixels**, simply by playing
- Can learn Q function by Q-learning

$$\Delta \boldsymbol{w} = \alpha \left( R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{w}) - Q(S_t, A_t; \boldsymbol{w}) \right) \nabla_{\boldsymbol{w}} Q(S_t, A_t; \boldsymbol{w})$$



# DQN

(Mnih, Kavukcuoglu, Silver, et al., Nature 2015)

- Learns to play video games **from raw pixels**, simply by playing
- Can learn Q function by Q-learning

$$\Delta \boldsymbol{w} = \alpha \left( R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{w}) - Q(S_t, A_t; \boldsymbol{w}) \right) \nabla_{\boldsymbol{w}} Q(S_t, A_t; \boldsymbol{w})$$

- Core components of DQN include:
  - Target networks (Mnih et al. 2015)

$$\Delta \boldsymbol{w} = \alpha \left( R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{w}^-) - Q(S_t, A_t; \boldsymbol{w}) \right) \nabla_{\boldsymbol{w}} Q(S_t, A_t; \boldsymbol{w})$$

  - Experience replay (Lin 1992): replay previous tuples (s, a, r, s')

# Target Network Intuition

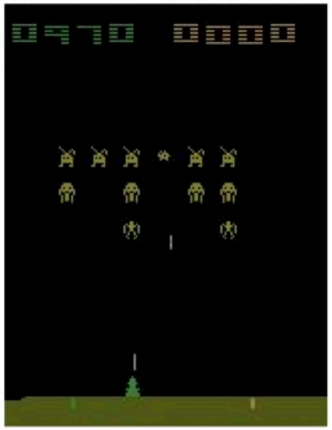
(Slide credit: Vlad Mnih)

- Changing the value of one action will change the value of other actions and similar states.
- The network can end up chasing its own tail because of bootstrapping.
- Somewhat surprising fact - bigger networks are less prone to this because they alias less.

$$L_i(\theta_i) = \mathbb{E}_{s,a,s',r \sim D}\left(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{target}} - Q(s, a; \theta_i)\right)^2$$

*s*

*s'*

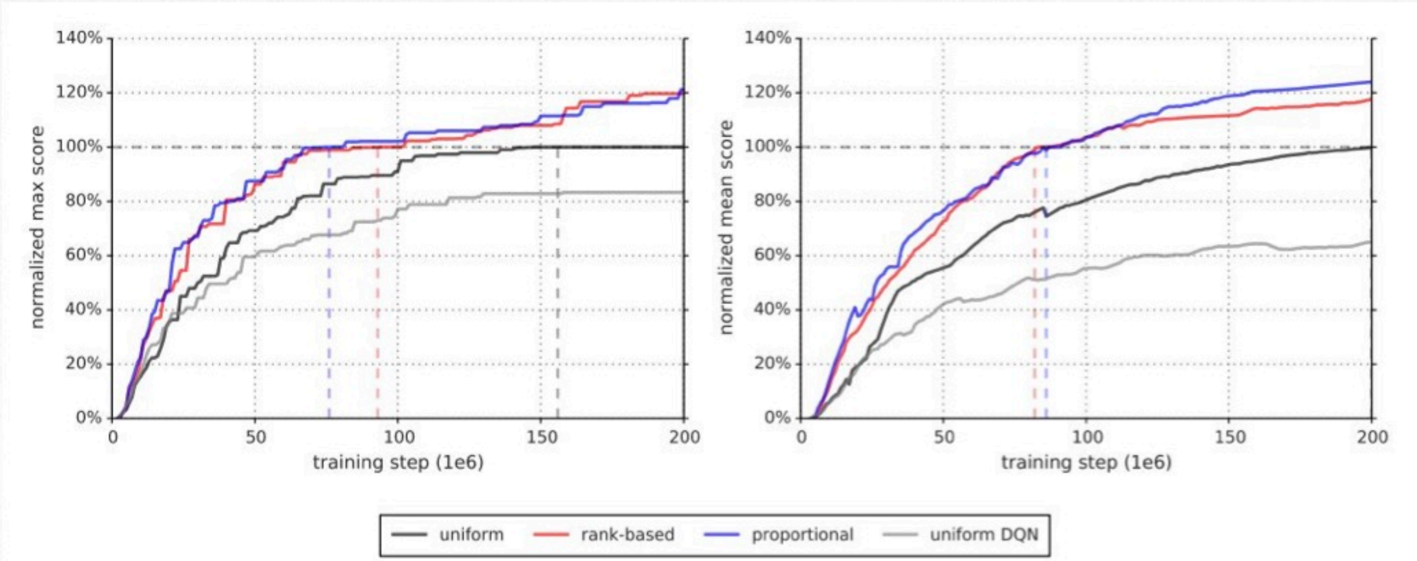


# Prioritized Experience Replay

"Prioritized Experience Replay", Schaul et al. (2016)

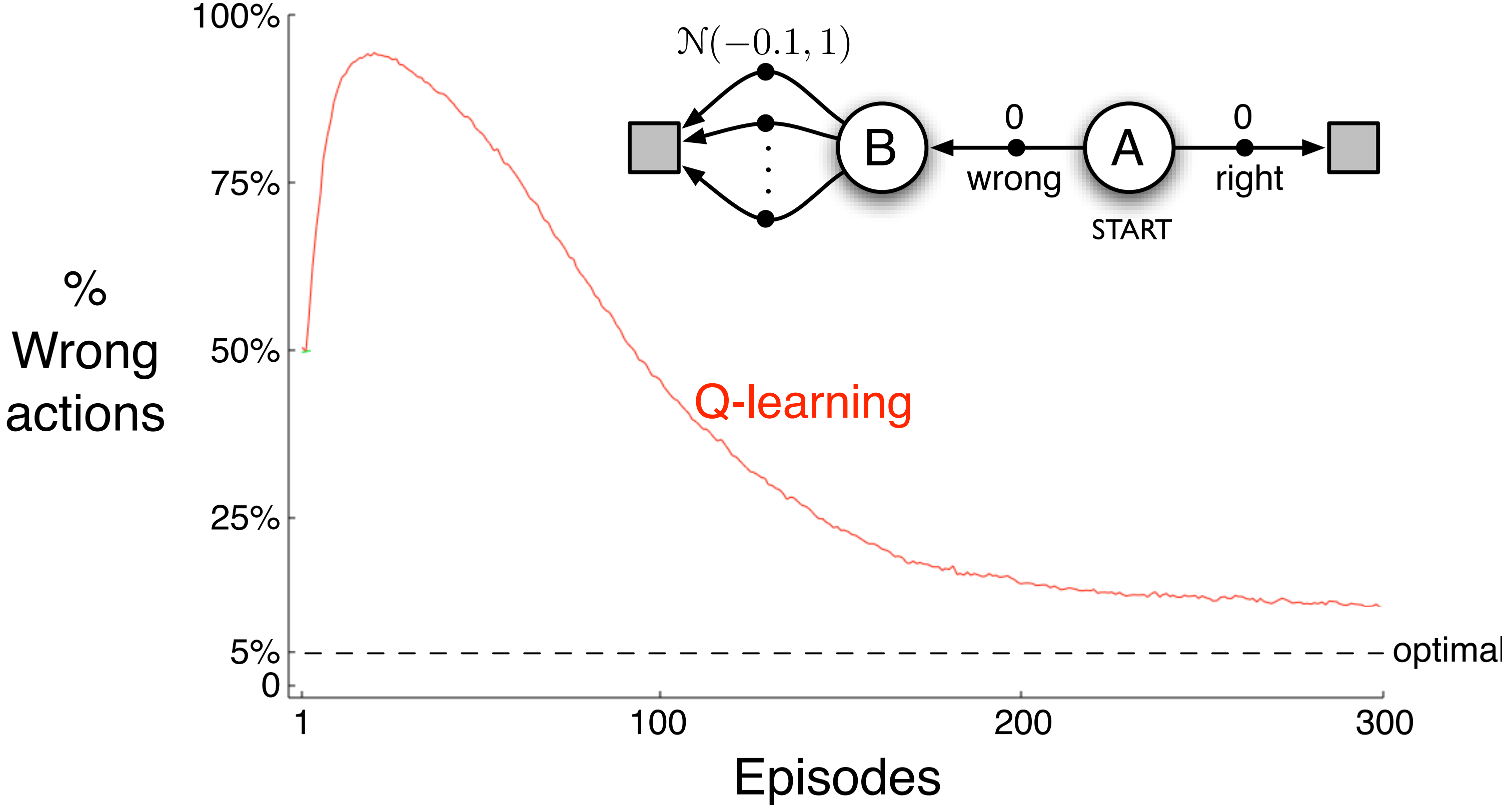
- **Idea**: Replay transitions in proportion to TD error:

$$\left| r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right|$$



# Maximization Bias Example

A maximum over estimated values is used implicitly as an estimate of the maximum value, which can lead to a significant positive bias.




Tabular Q-learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$

* with ε-greedy policy of ε=10%

Hado van Hasselt 2010

# Double Q-Learning

- Train 2 action-value functions, $Q_1$ and $Q_2$
- Do Q-learning on both, but
  - never on the same time steps ($Q_1$ and $Q_2$ are indep.)
  - pick $Q_1$ or $Q_2$ at random to be updated on each step
- If updating $Q_1$, use $Q_2$ for the value of the next state:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\Big(R_{t+1} + Q_2\big(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t)\Big)$$

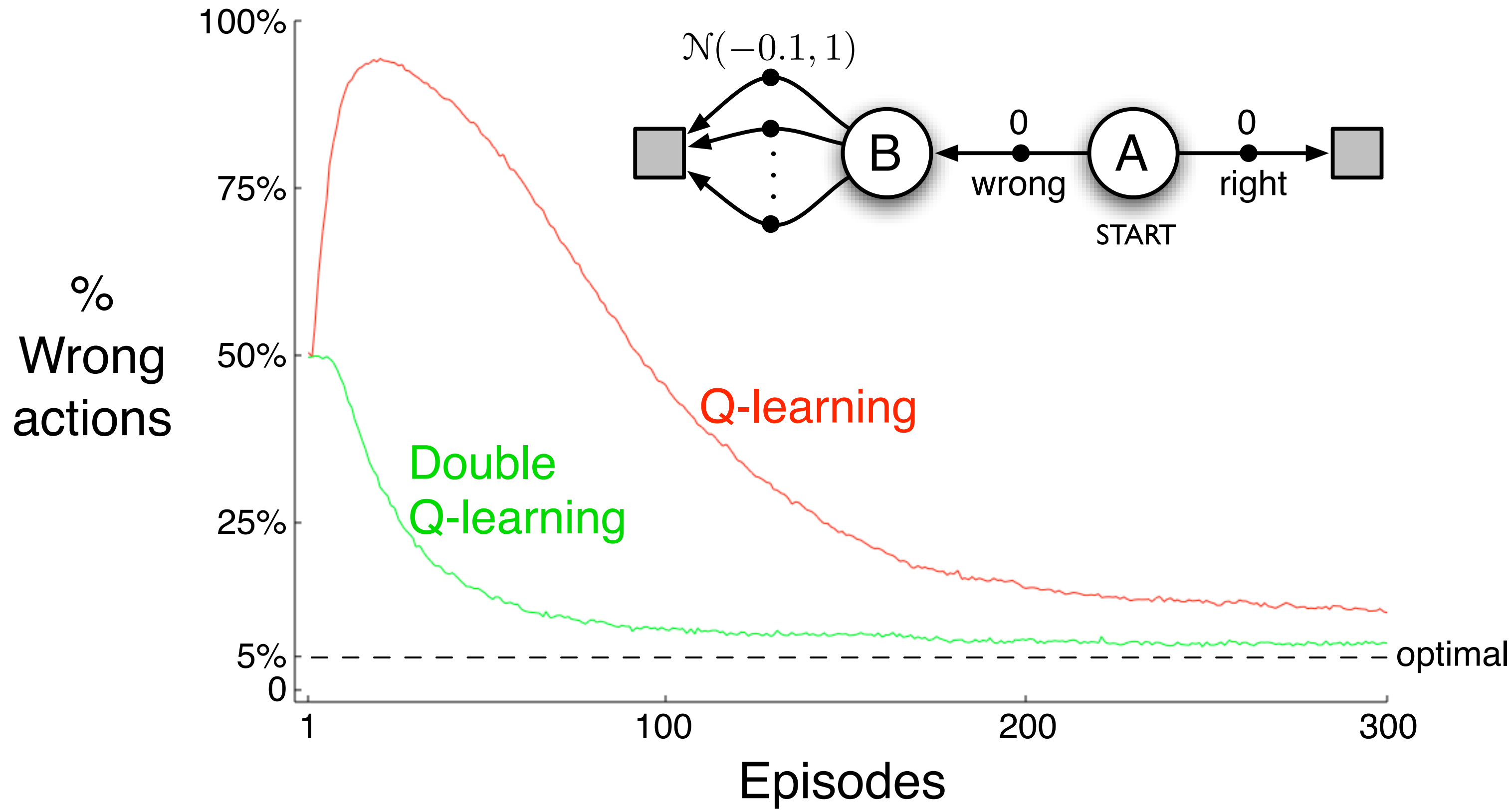
- Action selections are (say) ε-greedy with respect to the sum of $Q_1$ and $Q_2$

Hado van Hasselt 2010

# Double Q-Learning

Initialize $Q_1(s,a)$ and $Q_2(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily
Initialize $Q_1(\textit{terminal-state}, \cdot) = Q_2(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize $S$
    Repeat (for each step of episode):
        Choose $A$ from $S$ using policy derived from $Q_1$ and $Q_2$ (e.g., $\varepsilon$-greedy in $Q_1 + Q_2$)
        Take action $A$, observe $R$, $S'$
        With 0.5 probabilility:
            $Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S',a)\big) - Q_1(S,A)\Big)$
        else:
            $Q_2(S,A) \leftarrow Q_2(S,A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S',a)\big) - Q_2(S,A)\Big)$
        $S \leftarrow S'$;
    until $S$ is terminal

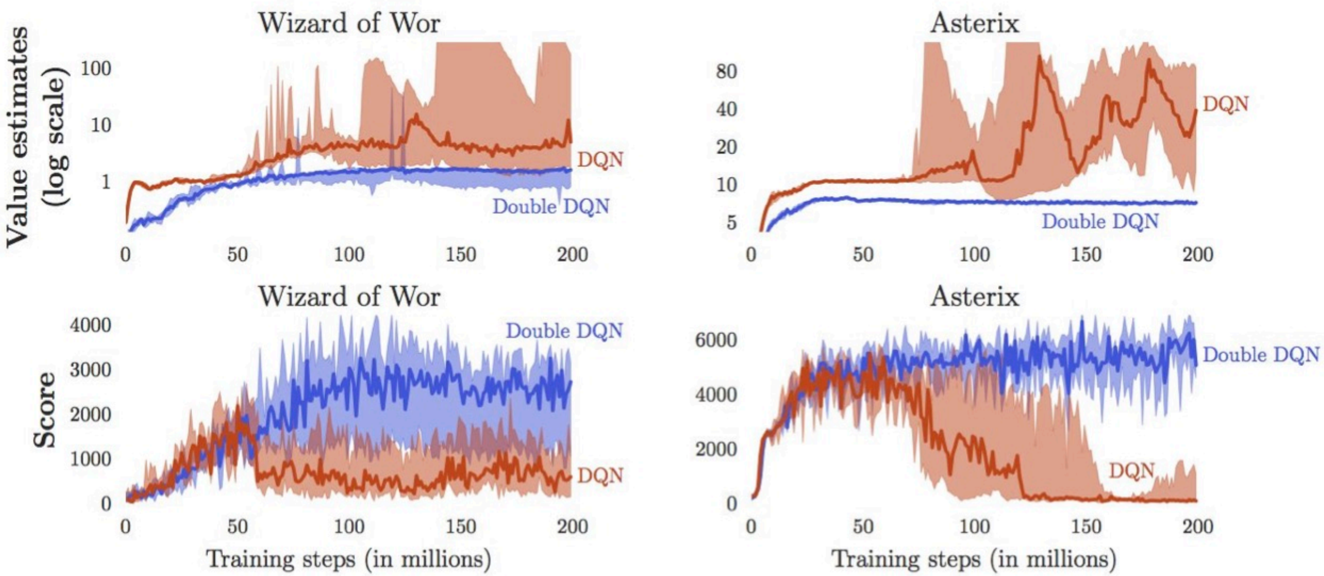
# Example of Maximization Bias




Double Q-learning:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha \Big[ R_{t+1} + \gamma Q_2\big(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t) \Big]$$

# Double DQN



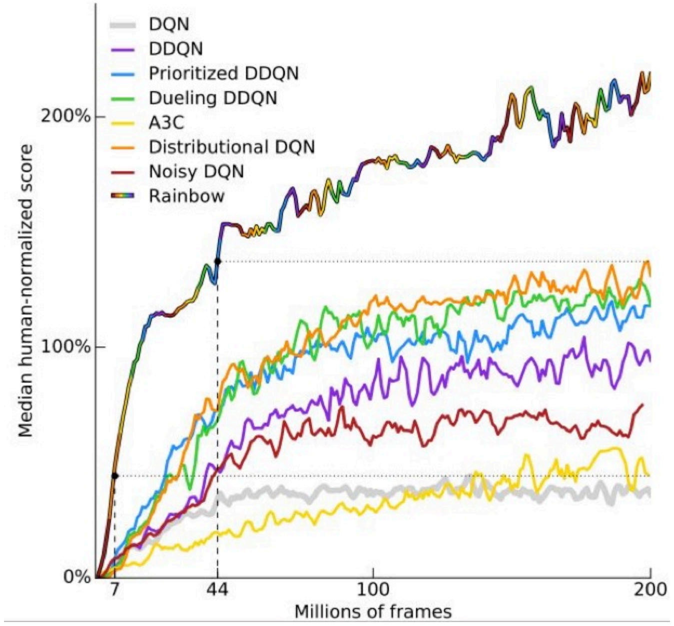

cf. van Hasselt et al, 2015)

# DQN

(Mnih, Kavukcuoglu, Silver, et al., Nature 2015)

- Many later improvements to DQN
  - Double Q-learning (van Hasselt 2010, van Hasselt et al. 2015)
  - Prioritized replay (Schaul et al. 2016)
  - Dueling networks (Wang et al. 2016)
  - Asynchronous learning (Mnih et al. 2016)
  - Adaptive normalization of values (van Hasselt et al. 2016)
  - Better exploration (Bellemare et al. 2016, Ostrovski et al., 2017, Fortunato, Azar, Piot et al. 2017)
  - Distributional losses (Bellemare et al. 2017)
  - Multi-step returns (Mnih et al. 2016, Hessel et al. 2017)
  - … many more …

# Which DQN improvements




Rainbow model, Hessel et al, 2017)

# Summary

- Extend prediction to control by employing some form of GPI
  - On-policy control: Sarsa, Expected Sarsa
  - Off-policy control: Q-learning, Expected Sarsa
- We can make these work with function approximation
- All ideas we talked about (n-step, eligibility traces) generalize to control