Deep Reinforcement Learning for Discrete Action Space

Group 6

Wei Li   Ming-Xu Huang

Reference

- Mnih, Volodymyr, et al. "**Human-level control through deep reinforcement learning**." Nature 518.7540 (2015): 529-533.
- Van Hasselt, Hado, Arthur Guez, and David Silver. "**Deep Reinforcement Learning with Double Q-Learning**." AAAI. 2016.
- Wang, Ziyu, et al. "**Dueling network architectures for deep reinforcement learning**." arXiv preprint arXiv:1511.06581 (2016).

- All the papers are published by **Google DeepMind**

Outline

- Introduction
- Deep Q Network
- Double DQN
- Dueling Network
- Experiment
- Conclusion

Outline

- Introduction
- Deep Q Network
- Double DQN
- Dueling Network
- Experiment
- Conclusion

Introduction

- In this slide we will discuss the traditional DQN and various improvements to Deep Q Network
- We will discuss the results of each methods and compare their performance.

Outline

- Introduction
- Deep Q Network
  - Target-Q
  - Experience replay
- Double DQN
- Dueling Network
- Experiment
- Conclusion

Deep Q Network

- State value: $V(s)$
- Action value: $Q(s, a)$
- Approximate action value by a neural network parameterized by θ
  - $Q(s, a; \theta)$
- Objective function
  - $L(\theta) = \mathbb{E}_{\mathrm{s,a,r,s'}}\left[\left(y^{DQN} - Q(s, a; \theta)\right)^2\right]$
  - $\mathrm{y}^{\mathrm{DQN}} = \mathrm{r} + \gamma \max_{a'} Q(s', a'; \theta)$
- Gradient
  - $\nabla L(\theta) = \mathbb{E}_{\mathrm{s,a,r,s'}}[(y^{DQN} - Q(s, a; \theta))\nabla Q(s, a; \theta)]$

Deep Q Network(cont.)

- Target-Q Network
  - Small updates to Q
    - Significantly change the policy
    - Changing correlations between the action-values and the target values
  - Neural networks is to use a separate network for generating the targets y
    - Every C step, clone weights θ of behavior Q network to target Q network weights θ^-

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

$$\text{Set } y_j = \begin{cases} r_j \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) \end{cases}$$

Every C steps reset $\hat{Q} = Q$

Deep Q Network(cont.)

- Experience replay
  - Store experiences e_t in a fixed size buffer D
    - $e_t = (s_t, a_t, r_t, s_{t+1})$
    - $D = \{e_1, e_2, \dots, e_n\}$
  - Trained by randomly sampling mini-batch of experiences from buffer uniformly
  - Decreasing the correlations present in the sequence of observations
  - Updating at iteration i uses the following loss function
    - $$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

Outline

- Introduction
- Deep Q Network
- Double DQN
  - Double Q-Learning
  - Overestimation of Q-Learning
  - Double DQN
- Dueling Network
- Experiment
- Conclusion

Double DQN

- The max operator in standard Q-learning and DQN, uses the same values both to select and to evaluate an action.
- This makes it more likely to select overestimated values, resulting in overoptimistic value estimates.
- To prevent this, we can decouple the selection from the evaluation.

Q-learning: $$\mathrm{Y}_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a;\ \theta_t)$$

Decouple

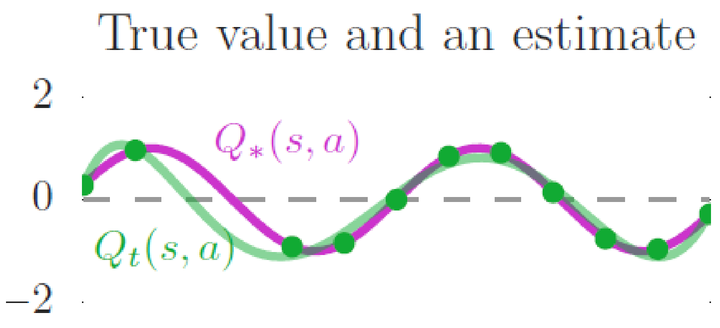
$$\mathrm{Y}_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a;\ \theta_t);\ \theta_t\right)$$

Double Q-learning: $$\mathrm{Y}_t^{Double\mathrm{Q}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a;\ \theta_t);\ \theta_t'\right)$$

Double DQN

- Overestimation of Q-Learning
  - Consider a real-valued continuous state space with 10 discrete actions in each state.
  - For simplicity, the true optimal action values in this example depend only on state so that in each state all actions have the same true value.

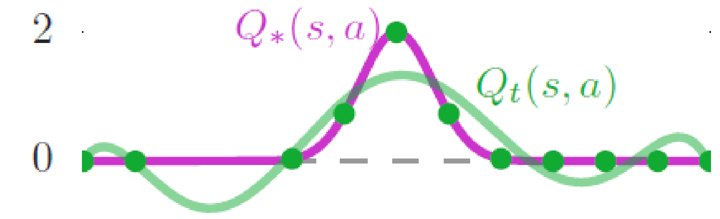
Double DQN(cont.)



$Q_*(s,a) = \sin(s)$

Degree of polynomial: 6

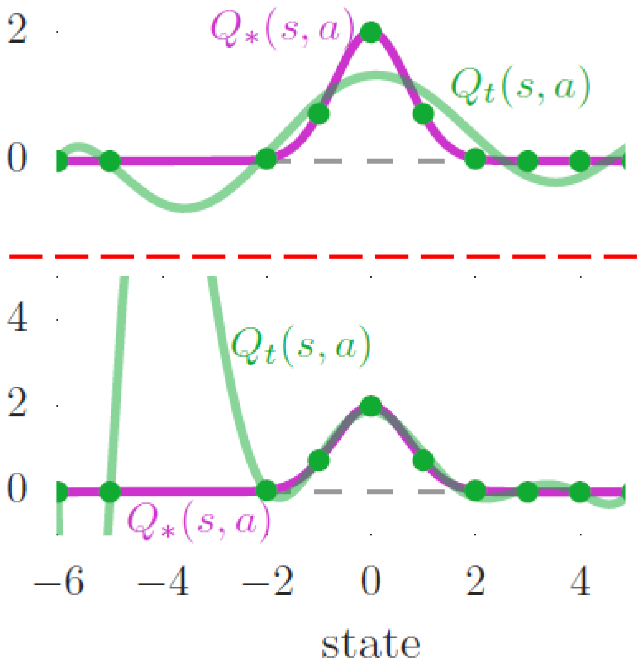
Different True Value Function



$Q_*(s,a) = 2\, exp(-s^2)$

Degree of polynomial: 6

Double DQN(cont.)



$Q_*(s,a) = 2\,exp(-s^2)$

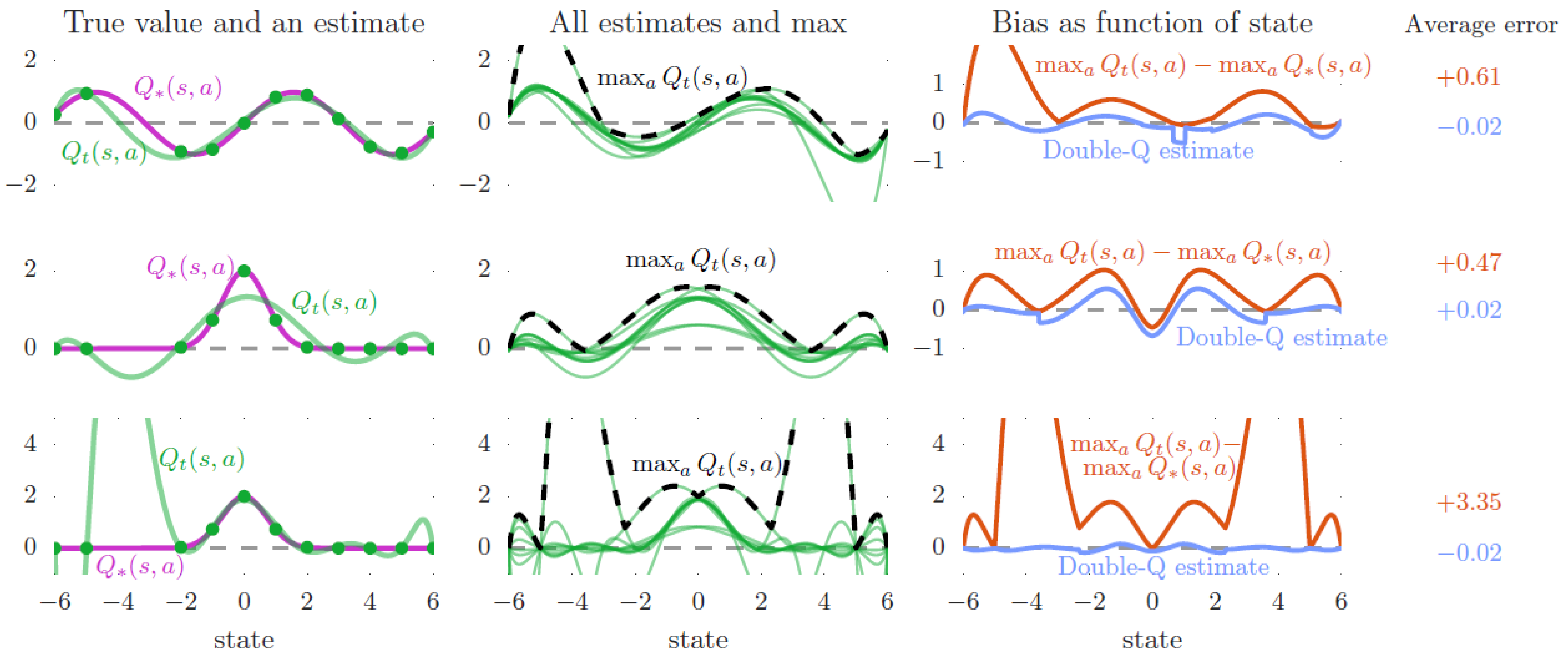
Degree of polynomial: 6

Different Degree of Polynomial

$Q_*(s,a) = 2\,exp(-s^2)$

Degree of polynomial: 9

Double DQN(cont.)



Double DQN(cont.)

- The idea of Double Q-learning is to reduce overestimations by decomposing the max operation in the target into action selection and action evaluation.
- Evaluate the greedy policy according to the online network.
- Using the target network to estimate its value.

$$\mathrm{Y}_t^{DQN} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a;\ \boldsymbol{\theta}_t^-)$$

$$\mathrm{Y}_t^{DQN} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-); \boldsymbol{\theta}_t^-\right)$$

$$\mathrm{Y}_t^{DoubleDQN} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a;\ \boldsymbol{\theta}_t);\ \boldsymbol{\theta}_t^-\right)$$

Double DQN(cont.)

Algorithm 1: Double DQN Algorithm.

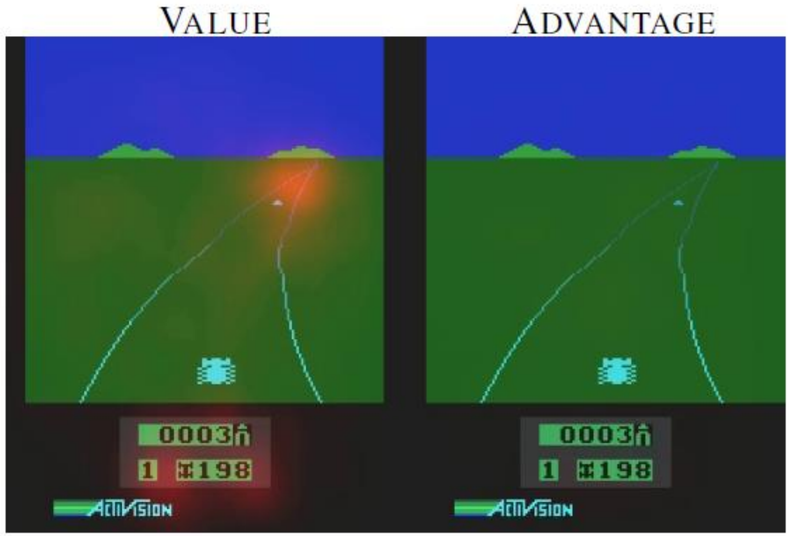
input : $\mathcal{D}$ – empty replay buffer; θ – initial network parameters, θ^- – copy of θ
input : N_r – replay buffer maximum size; N_b – training batch size; N^- – target network replacement freq.
for *episode* $e \in \{1, 2, \ldots, M\}$ **do**
  Initialize frame sequence $\mathbf{x} \leftarrow ()$
  for $t \in \{0, 1, \ldots\}$ **do**
    Set state $s \leftarrow \mathbf{x}$, sample action $a \sim \pi_{\mathcal{B}}$
    Sample next frame x^t from environment $\mathcal{E}$ given (s, a) and receive reward r, and append x^t to $\mathbf{x}$
    if $|\mathbf{x}| > N_f$ **then** delete oldest frame $x_{t_{min}}$ from $\mathbf{x}$ **end**
    Set $s' \leftarrow \mathbf{x}$, and add transition tuple (s, a, r, s') to $\mathcal{D}$,
      replacing the oldest tuple if $|\mathcal{D}| \geq N_r$
    Sample a minibatch of N_b tuples $(s, a, r, s') \sim \mathrm{Unif}(\mathcal{D})$
    Construct target values, one for each of the N_b tuples:
    $\boxed{\text{Define } a^{\max}(s'; \theta) = \arg\max_{a'} Q(s', a'; \theta)}$ $\boldsymbol{\theta}$
    $y_j = \begin{cases} r & \text{if } s' \text{ is terminal} \\ r + \gamma \boxed{Q(s', a^{\max}(s'; \theta); \theta^-)} & \text{otherwise.} \end{cases}$ $\boldsymbol{\theta^-}$
    Do a gradient descent step with loss $\|y_j - Q(s, a; \theta)\|^2$
    Replace target parameters $\theta^- \leftarrow \theta$ every N^- steps
  end
end

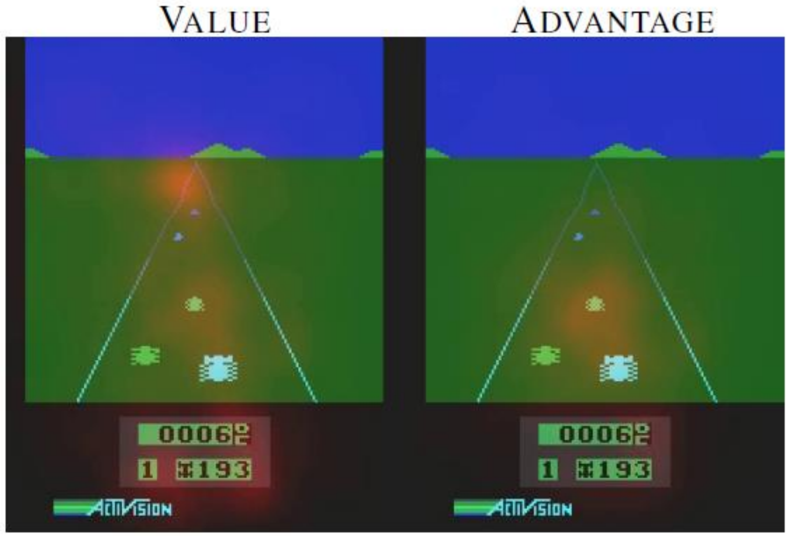
Outline

- Introduction
- Deep Q Network
- Double DQN
- Dueling Network
  - Advantage Function
  - Dueling network architecture
  - Combine methods
- Experiment
- Conclusion

Dueling Network

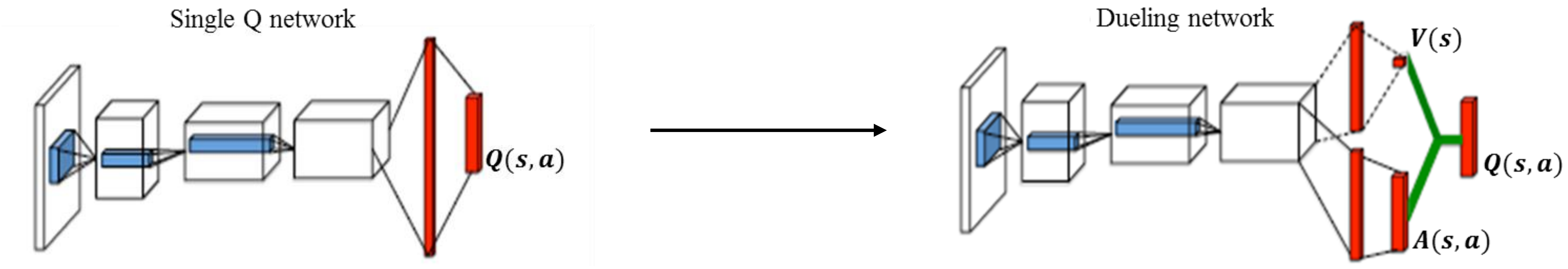
- Advantage Function
  - $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$
- The value stream learns to pay attention to the road.
- The advantage stream learns to pay attention only when there are cars immediately in front, so as to avoid collisions





Dueling Network(cont.)

- Produce a state value and advantage functions via a single network
  - state value is a scalar
  - advantage functions is a vector of size $|A|$
- Combine state value and advantage functions to generate action values
  - $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \alpha) + A(s, a; \theta, \beta)$



Dueling Network(cont.)

- Unidentifiable
  - Q(s, a) = $V(s)$ + $A(s,a)$
  - However, $V(s)$ + $A(s,a)$ = ($V(s)$ $-$ C) + ($A(s,a)$ + C) (Poor performance)
  - Improvement
    - $Q(s,a) = V(s) + \left(A(s,a) - \mathbf{\max_{a'}} A(s,a')\right)$
      - $a^* = \max_{a'} Q(s,a')\,,\ Q(s,a^*) = V(s)$
    - $Q(s,a) = V(s) + \left(A(s,a) - \frac{\mathbf{1}}{|A|} \sum_{\boldsymbol{a'}} \boldsymbol{A(s,a')}\right)$
      - $a^* = \max_{a'} Q(s,a')\,,\ Q(s,a^*) \neq V(s)$
      - Increase stability of the optimization

Dueling Network(cont.)

- Strengths
  - Compatibility
    - Easily combined with existing and future algorithms for RL

  - Better approximation of the state values
    - More frequent updating of the value stream
    - Only the value for one of the actions is updated in traditional deep Q network

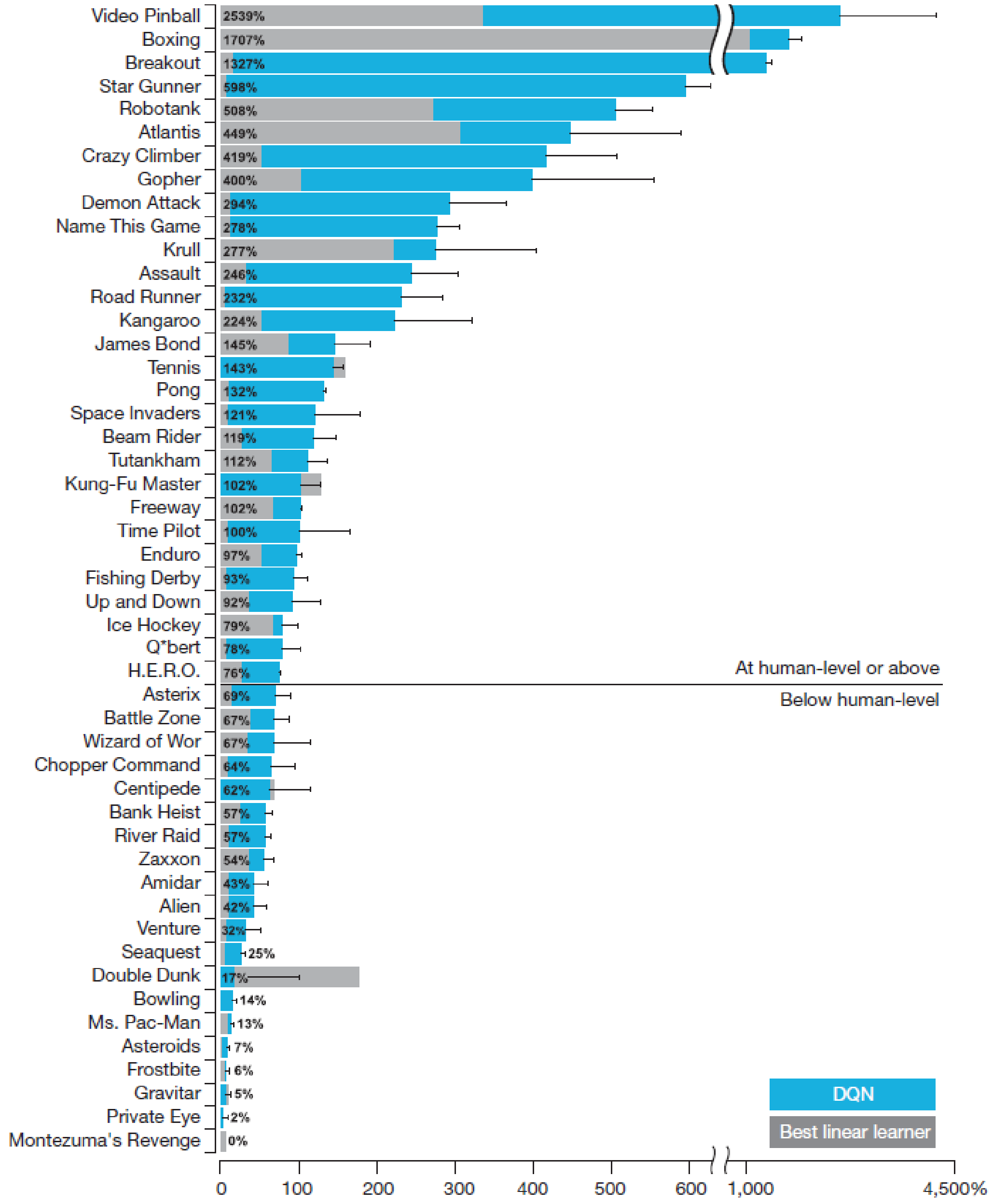
Outline

- Introduction
- Deep Q Network
- Double DQN
- Dueling Network
- Experiment
- Conclusion

Experiment

- Compare with Marc et al. (2012)
- Achieving more than 75% of the human score on 29 games.

$$\text{score}_{\text{normalized}} = \frac{\text{score}_{\text{agent}} - \text{score}_{\text{random}}}{\text{score}_{\text{human}} - \text{score}_{\text{random}}}$$



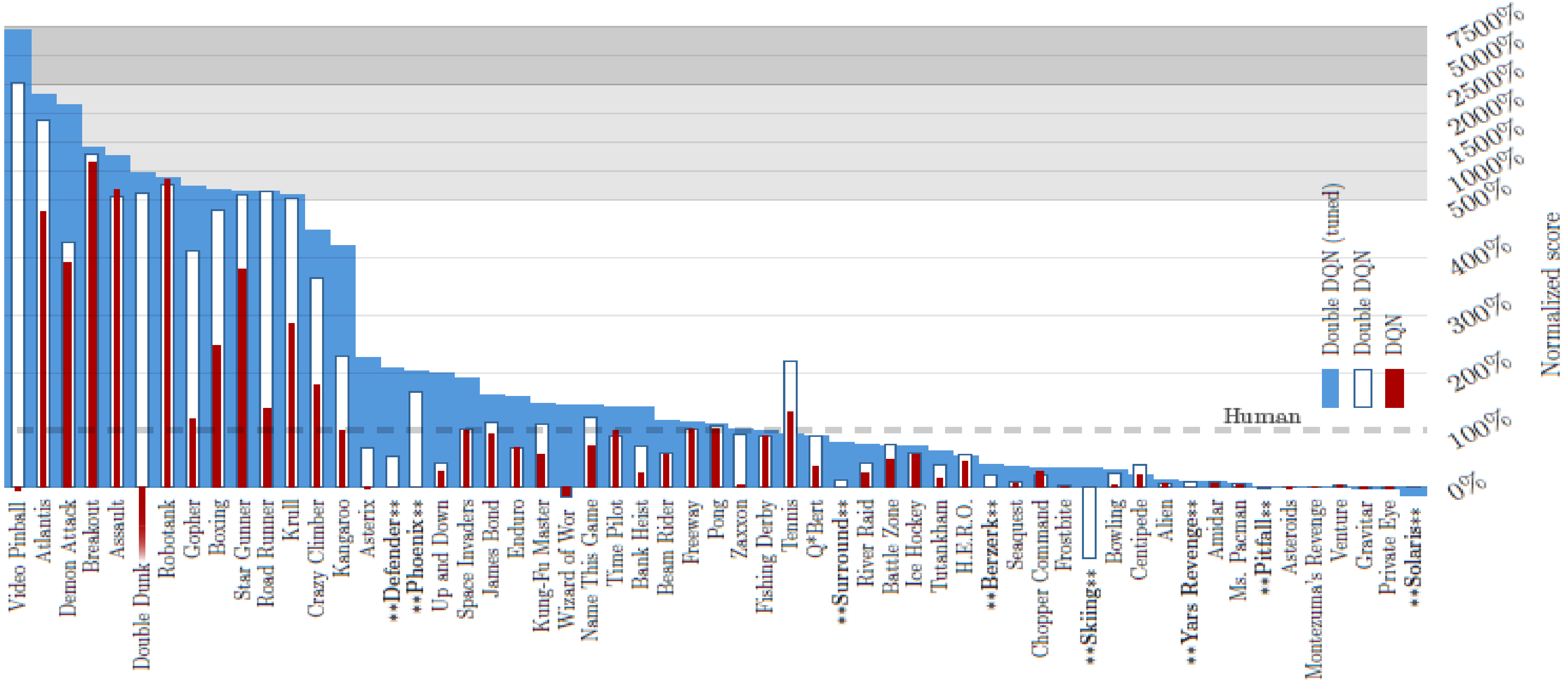
Experiment

- Replay memory and Target Q help to increase score.

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

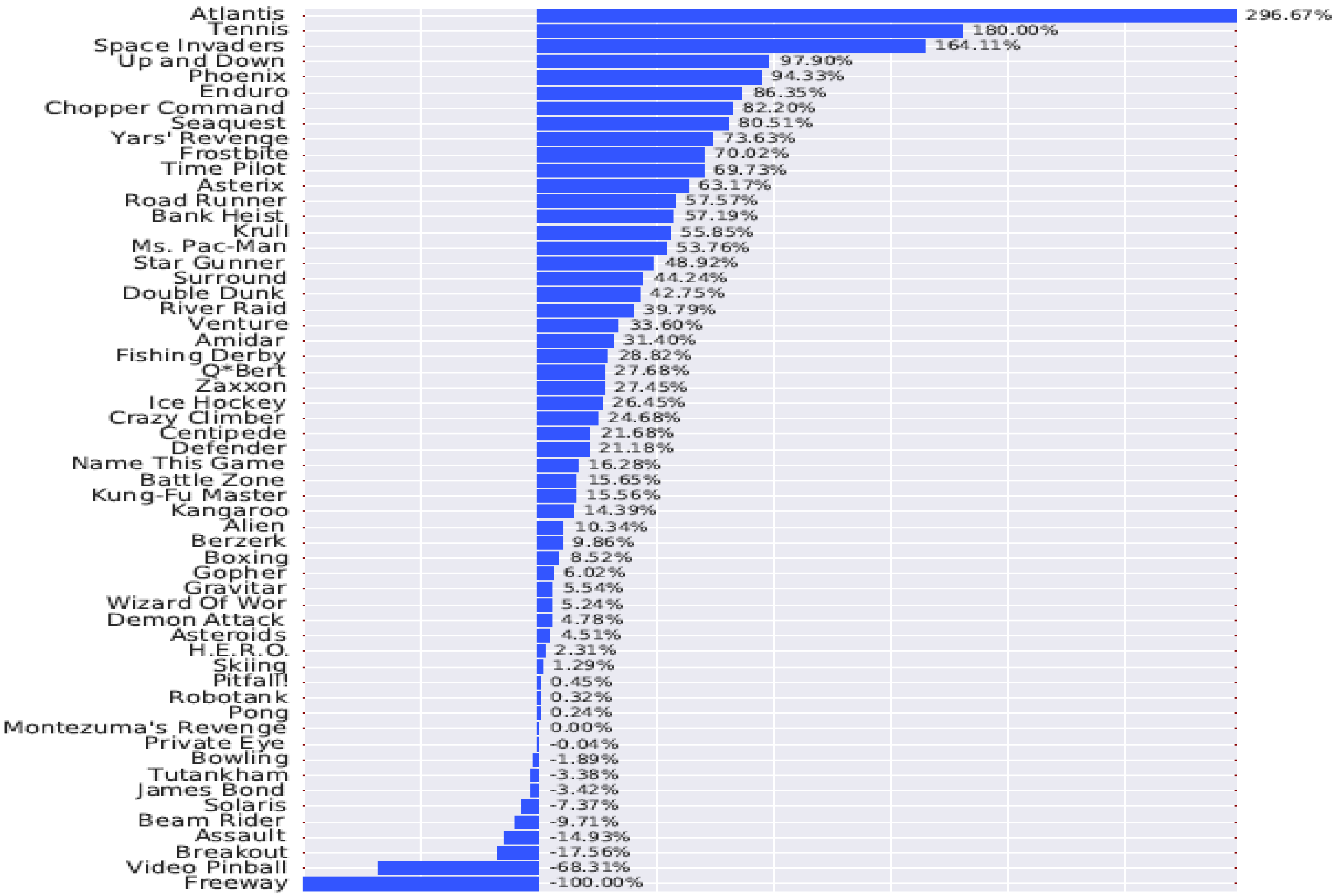
Experiment

- Double DQN vs DQN



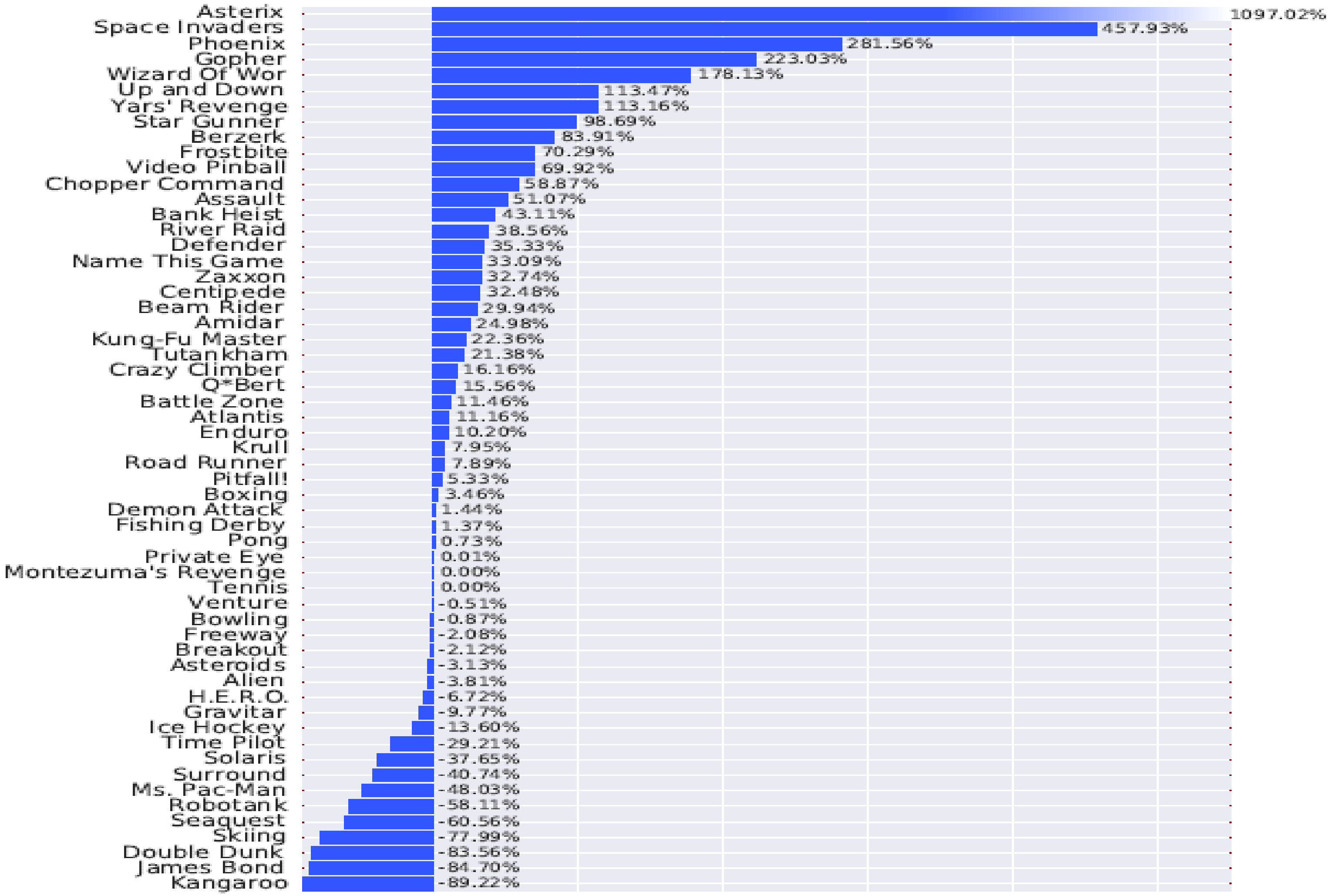
Experiment

- Dueling Network vs Hasselt et al. (2015)



Experiment

- Dueling Network vs DDQN



Outline

- Introduction
- Deep Q Network
- Double DQN
- Dueling Network
- Experiment
- Conclusion

Conclusion

- DQN is the first successful Deep Reinforcement Learning Algorithm
  - Comparable level with human on 49 games of Atari2600
  - Publish on Nature 2015
- Double DQN solve the overestimation problem of DQN
  - Publish AAAI 2016
- Dueling Network is a new network architecture for RL
  - Compatibility for existed RL algorithm
  - Better performance
  - ICML 2016 Best Paper