Reinforcement Learning: Basics and DQN

Changyou Chen

Department of Computer Science and Engineering
Universitpy at Buffalo, SUNY
changyou@buffalo.edu

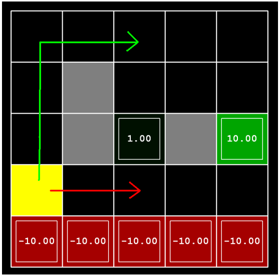
April 25, 2019

Limitation

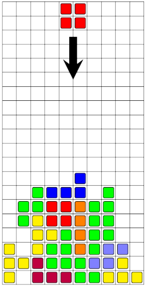
Limitations:

- Iteration over / storage for all states and actions: requires small, discrete state-action space:
  - sampling-based approximations
- Update equations require access to dynamics mode:
  - Q/V function fitting.

Can tabular methods scale?



Gridworld
10^1



Tetris
10^60

Atari
10^308 (ram) 10^16992 (pixels)

Approximate Q-Learning

- Instead of a table, we have a parametrized Q-function $Q_\theta(s, a)$:
  - Can be a linear function in features:

$$Q_\theta(s, a) = \theta_0 f_0(s, a) + \theta_1 f_1(s, a) + \cdots + \theta_n f_n(s, a)$$

  - Or a complicated neural network.
- Learning rule:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta \left[\frac{1}{2} \left(Q_\theta(s, a) - \text{target}(s') \right)^2 \right] |_{\theta=\theta_k}$$

$$\text{target}(s') \triangleq R(s, a, s') + \gamma \max_{a'} Q_{\theta_k}(s', a')$$

$$Q_{k+1}(s, a) = \sum_{s'} P(s'|s, a) \left(R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right)$$

$$\approx R(s, a, s') + \gamma \max_{a'} Q_k(s', a'), \quad \text{with } s' \sim P(\cdot|s, a)$$

Connection to Tabular Q-Learning

- Suppose $\theta \in \mathbb{R}^{|S|\times|A|}, \quad Q_\theta(s,a) \equiv \theta_{sa}$

$$\begin{aligned}
&\nabla_{\theta_{sa}} \left[\frac{1}{2}(Q_\theta(s,a) - \mathrm{target}(s'))^2\right] \\
&= \nabla_{\theta_{sa}} \left[\frac{1}{2}(\theta_{sa} - \mathrm{target}(s'))^2\right] \\
&= \theta_{sa} - \mathrm{target}(s')
\end{aligned}$$

- Plug into update: $$\begin{aligned}\theta_{sa} &\leftarrow \theta_{sa} - \alpha(\theta_{sa} - \mathrm{target}(s')) \\ &= (1-\alpha)\theta_{sa} + \alpha[\mathrm{target}(s')]\end{aligned}$$

- Compare with Tabular Q-Learning update:

$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha\left[\mathrm{target}(s')\right]$$

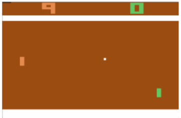
Convergence of Approximate Q-Learning

- It is not guaranteed to converge, even if the function approximation is expressive enough to represent the true Q-function:
  - The approximation is sequential, so if each time there induces a large enough error, the aggregate error might be exploded.

Deep Q-Learning[1]

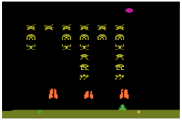
[1] Adapted from http://www.teach.cs.toronto.edu/∼csc2542h/fall/material/csc2542f16_dqn.pdf

Artari Games



Pong

Breakout



Space Invaders

Seaquest

Beam Rider

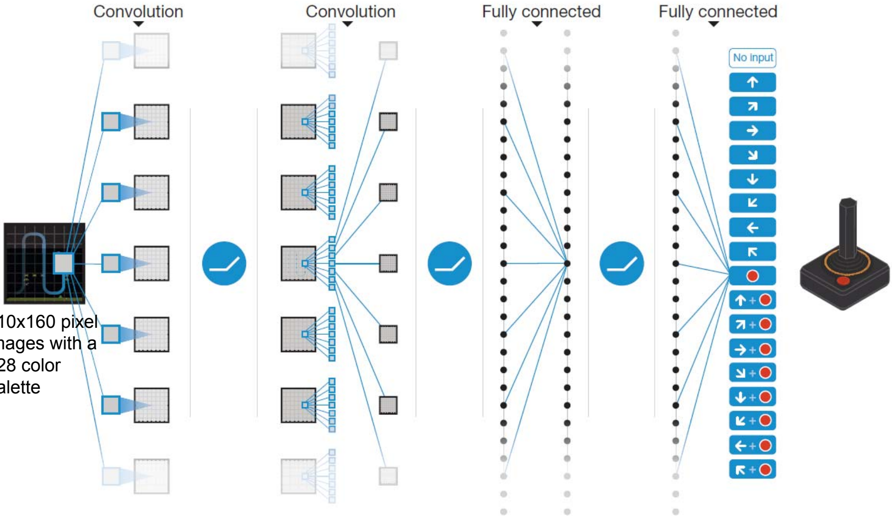
Breakout game

How to define a state?

- Location of the paddle
- Location/direction of the ball
- Presence/absence of each individual brick

Use screen pixels!

DQN in Atari



210x160 pixel images with a 128 color palette

Deep Q-Learning

- Naive Q-learning oscillates or diverges with neural nets:
  - Data is sequential:
    - Successive samples are correlated, non-i.i.d.
  - Policy changes rapidly with slight changes to Q-values:
    - Policy may oscillate
    - Distribution of data can swing from one extreme to another
  - Scale of rewards and Q-values is unknown:
    - Naive Q-learning gradients can be large unstable when backpropagated

Approximate Q-Learning

- Instead of a table, we have a parametrized Q-function $Q_\theta(s, a)$:
  - Can be a linear function in features:

$$Q_\theta(s, a) = \theta_0 f_0(s, a) + \theta_1 f_1(s, a) + \cdots + \theta_n f_n(s, a)$$

  - Or a complicated neural network.
- Learning rule:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta \left[\frac{1}{2} \left(Q_\theta(s, a) - \text{target}(s') \right)^2 \right] |_{\theta=\theta_k}$$

$$\text{target}(s') \triangleq R(s, a, s') + \gamma \max_{a'} Q_{\theta_k}(s', a')$$

Deep Q-Learning

- Deep Q-Network provides a stable solution to deep value-based RL:
  - Use experience replay:
    - Break correlations in data, bring us back to i.i.d. setting
    - Learn from all past policies
    - Using off-policy Q-learning
  - Freeze target Q-network:
    - Avoid oscillations
    - Break correlations between Q-network and target
  - Clip rewards or normalize network adaptively to sensible range:
    - Robust gradients

Algorithm:

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, … till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s'

else:

$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) \leftarrow (1 - \alpha) Q_k(s, a) + \alpha\,[\text{target}]$$

$$s \leftarrow s'$$

Deep Q-Learning: Experience Replay

- To remove correlations, build data-set from agent's own experience:
  - Take action a_t according to ϵ-greedy policy
  - Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory $\mathcal{D}$ (Huge data base to store historical samples)
  - Sample random mini-batch of transitions (s, a, r, s') from $\mathcal{D}$
  - Optimize MSE between Q-network and Q-learning targets, *e.g.*,

$$L_k(\theta_k) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\theta_k}(s, a) - \text{target}(s') \right)^2 \right]$$

$$\text{target}(s') \triangleq R(s, a, s') + \gamma \max_{a'} Q_{\theta_k}(s', a')$$

Deep Q-Learning: Fixed Target Q-Network

- To avoid oscillations, fix parameters used in Q-learning target:
  - Compute Q-learning targets w.r.t. old, fixed parameters θ_k^-:

$$\text{target}(s'; \theta_k^-) \triangleq R(s, a, s') + \gamma \max_{a'} Q_{\theta_k^-}(s', a')$$

  - Optimize MSE between Q-network and Q-learning targets, *e.g.*,

$$L_k(\theta_k) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\theta_k}(s, a) - \text{target}(s'; \theta_k^-) \right)^2 \right]$$

  - Periodically update fixed parameters $\theta_k^- = \theta_k$.

Deep Q-Learning: Reward / Value Range

- DQN clips the reward to [-1, +1].
- This prevents Q-values from becoming too large.
- Ensures gradients are well-conditioned.

Stable DQN

DQN

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

Algorithm

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

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

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ

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

  End For

End For

Training

During Training

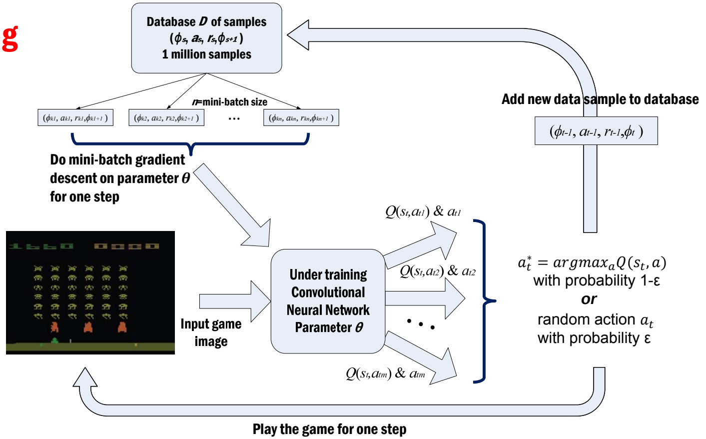


image at time t : x_t

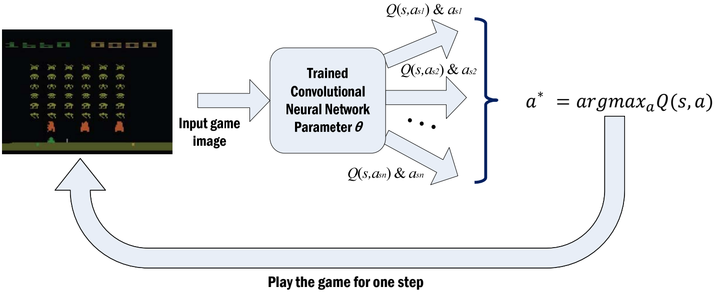
$s_t = s_{t-1}, a_{t-1}, x_t$

preprocessed sequence

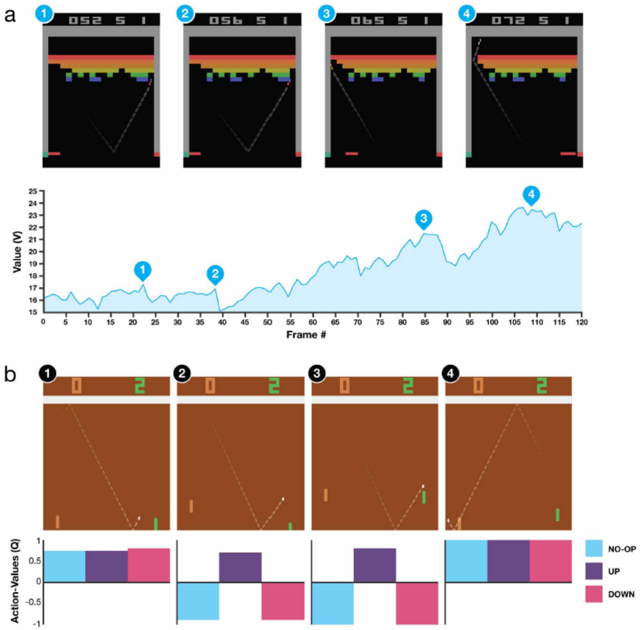
$\phi_t = \phi(s_t)$

Testing

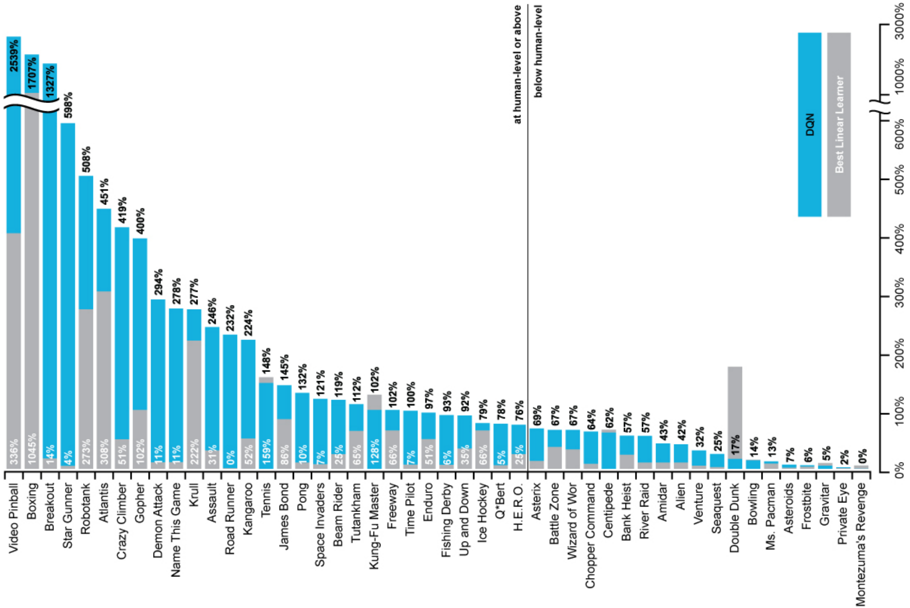
After Training



Results


a
052 5 1
056 5 1
065 5 1
072 5 1
Value (V)
Frame #
b
NO-OP
UP
DOWN
Action-Values (Q)

Results


DQN
Best Linear Learner
at human-level or above
below human-level
0%
100%
200%
300%
400%
500%
600%
1000%
3000%
Video Pinball 2539% 336%
Boxing 1707% 1045%
Breakout 1327% 14%
Star Gunner 598% 4%
Robotank 508% 273%
Atlantis 451% 308%
Crazy Climber 419% 51%
Gopher 400% 102%
Demon Attack 294% 11%
Name This Game 278% 11%
Krull 277% 222%
Assault 246% 31%
Road Runner 232% 0%
Kangaroo 224% 52%
Tennis 148% 159%
James Bond 145% 86%
Pong 132% 10%
Space Invaders 121% 7%
Beam Rider 119% 25%
Tutankham 112% 65%
Kung-Fu Master 102% 128%
Freeway 102% 66%
Time Pilot 100% 7%
Enduro 97% 51%
Fishing Derby 93% 6%
Up and Down 92% 35%
Ice Hockey 79% 66%
Q*Bert 78% 5%
H.E.R.O. 76% 25%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pacman 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%

Double Q-Learning

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 independent)
  - pick Q_1 or Q_2 at random to be updated on each step
- If updating Q_1, use Q_2 to evaluate the target:

$$Q_1(s_t, a_t) = Q_1(s_t, a_t) + \alpha \left(R_{t+1} + Q_2(s_{t+1}, \operatorname*{argmax}_a Q_1(s_{t+1}, a)) - Q_1(s_t, a_t) \right)$$

- Action selections are ϵ-greedy with respect to the sum of Q_1 and Q_2.

Deep Q-Learning

$$\text{target}(s'; \theta_k^-) \triangleq R(s, a, s') + \gamma \max_{a'} Q_{\theta_k^-}(s', a')$$

$$L_k(\theta_k) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\theta_k}(s, a) - \text{target}(s'; \theta_k^-) \right)^2 \right]$$

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., ε-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:

$$Q_2(S,A) \leftarrow Q_2(S,A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S',a)\big) - Q_2(S,A)\Big)$$

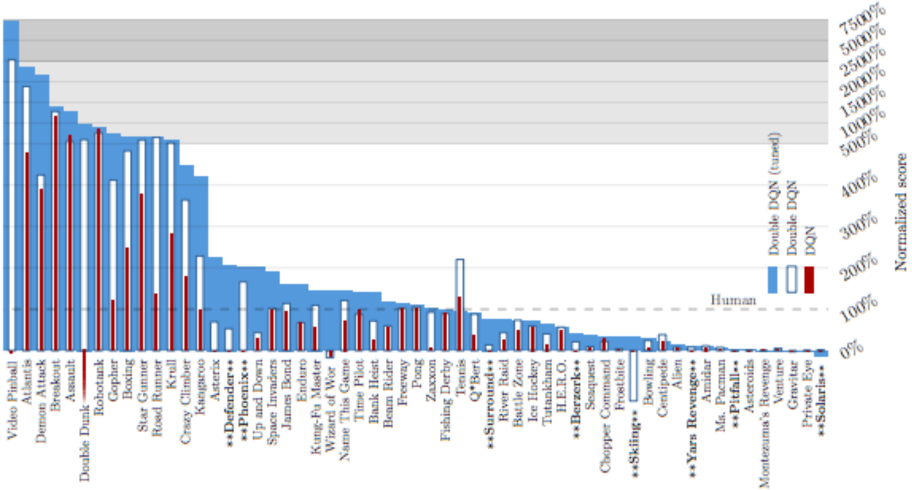
    $S \leftarrow S'$;
  until S is terminal

Double Q-Learning

- Current Q-network θ is used to select actions.
- Older Q-network θ^- is used to evaluate actions.

$$L = \left(r + \gamma \overbrace{Q(s', \underbrace{\operatorname{argmax}_{a'} Q(s', a', \mathbf{w})}_{\text{Action selection: } \theta}, \theta^-)}^{\text{Action evaluation: } \theta^-} - Q(s, a, \theta) \right)^2$$

Double Q-Learning



Dueling Network Architecture

Motivation

Q function should be designed more wisely: containing an action-independent component

For many states:

- unnecessary to estimate the value of each action choice, for example, move left or right only matters when a collision is eminent.
- In most of states, the choice of action has no affect on what happens

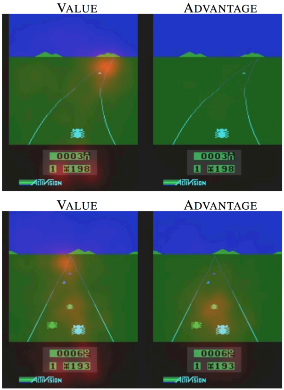
Decompose Q

$$
\begin{aligned}
& Q^\pi(s,a) = V^\pi(s) + A^\pi(s,a) \\
\Rightarrow & A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) = Q^\pi(s,a) - \mathbb{E}_{a\sim\pi}[Q^\pi(s,a)] \\
& \text{(advantage function)}
\end{aligned}
$$

Saliency map on the Atari game Enduro

1. Focus on **horizon**, where new **cars appear**

2. Focus on the **score**

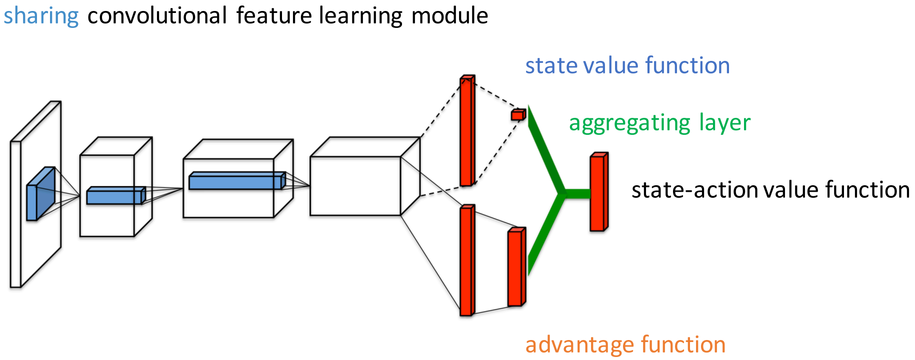


Not pay much attention when there are **no cars** in front

Attention on car immediately in front making **its choice of action very relevant**

Dueling Network

- Single Q-network with two streams.
- Produce separate estimations of state value function V and advantage function A.



- outputs a scalar value $V(s)$ and a $|\mathcal{A}|$-dimensional vector $A(s, a)$.

Aggregation Modules

- Add: $Q(s, a) = V(s) + A(s, a)$.
- Subtract max: $Q(s, a) = V(s) + (A(s, a) - \max_{a'} A(s, a'))$:
  - force A to have zero at the chosen optimal action.
  - Equivalent to shifting V.
- Subtract mean: $Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a')\right)$:
  - loses the original semantics of V and A.
  - but increases the stability of the optimization.
  - often works best.

Table 1. Mean and median scores across all 57 Atari games, measured in percentages of human performance.

	30 no-ops		Human Starts	
	Mean	**Median**	**Mean**	**Median**
Prior. Duel Clip	**591.9%**	**172.1%**	**567.0%**	**115.3%**
Prior. Single	434.6%	123.7%	386.7%	112.9%
Duel Clip	**373.1%**	**151.5%**	**343.8%**	**117.1%**
Single Clip	341.2%	132.6%	302.8%	114.1%
Single	307.3%	117.8%	332.9%	110.9%
Nature DQN	227.9%	79.1%	219.6%	68.5%