Lecture 23- April 11, 2023

- Deep Q-Network (DQN) → Finite action
  - Deep Q-Network (DQN)
  - Double DQN
  - Prioritized DQN
  - Dueling DQN
- Deep Policy Gradients (DPG) → large/continus Action
  - REINFORCE
  - REINFORCE with Baseline
  - Advantage Actor Critic (A2C)
  - Deep Deterministic Policy Gradiat (DDPG)

Project 3 → Due April 17

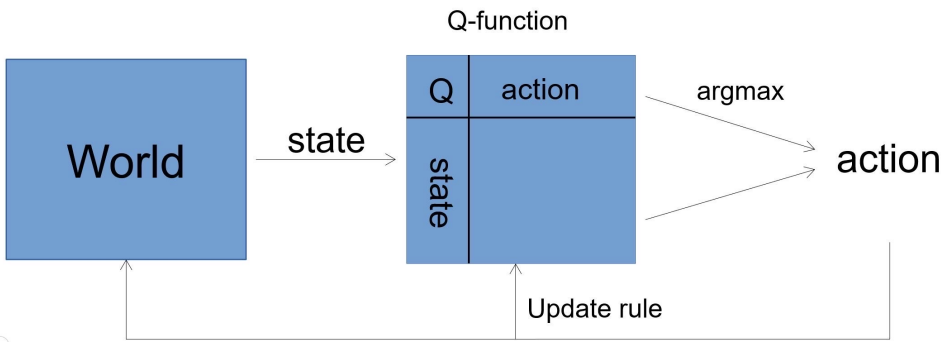
HW5 → Due April 18

TA's office hour: Wendsdays, 2pm-3pm (in-person)

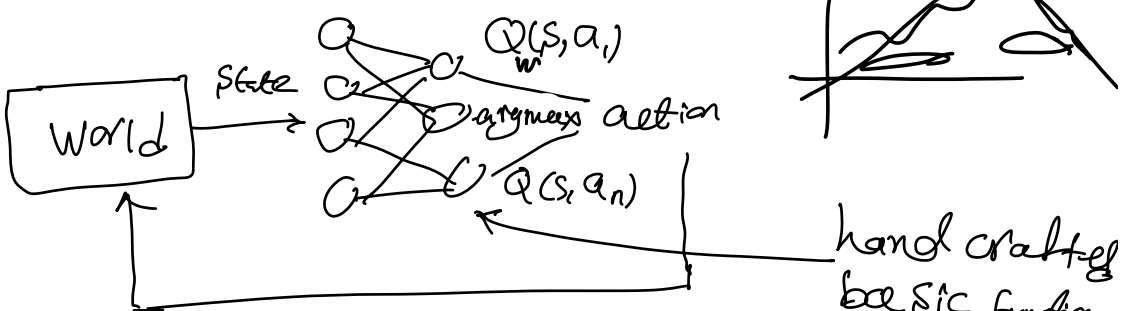
Fridays, 2pm-3pm (virtual)

Q-learning : Tabular

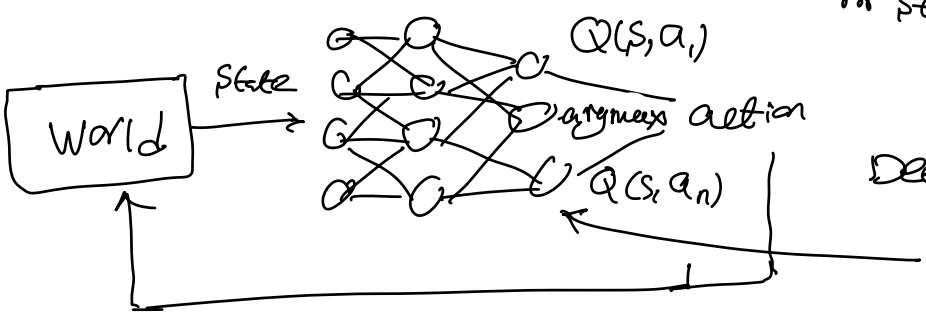
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R,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



Q-Learning with Linear function Approximation

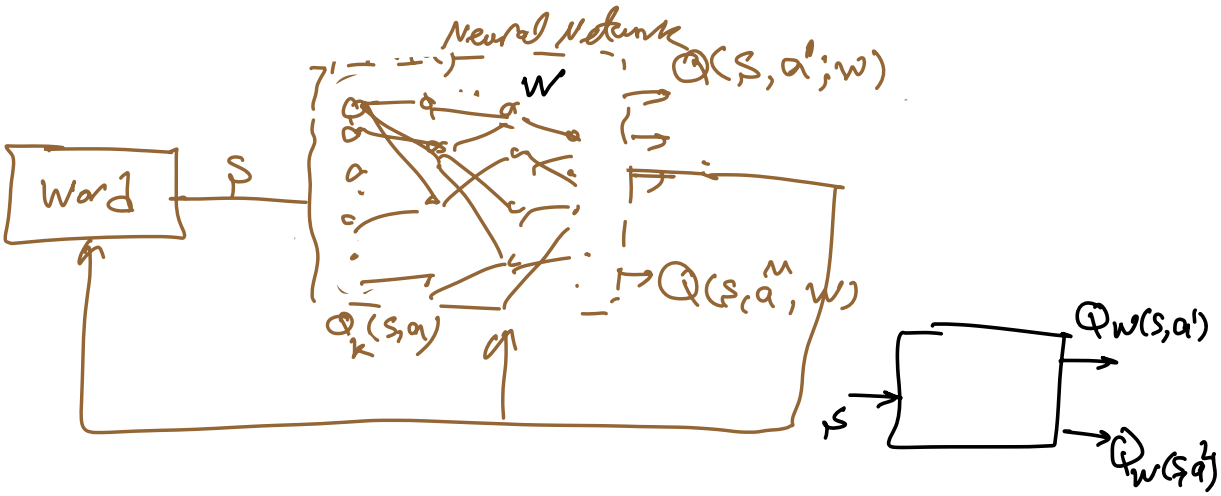


hand crafted basis function for state-action pair



Deep Q Network

State ← big & Complex ← Raw data (Image)



Q-Learning Update:

Q-Learning Target

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

Q-Learning error

① model $Q_w(s,a)$ with NN with weights w.

② Define Loss function

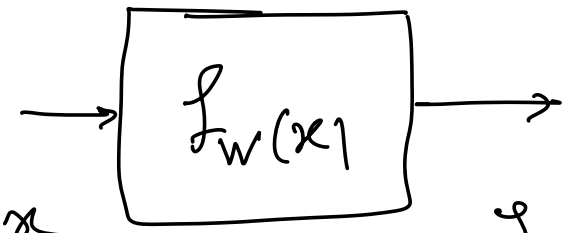
$$L(s,a,s';w) = \frac{1}{2}\left(\underbrace{r + \gamma \max_{a'} Q_w(s',a')}_{y} - Q_w(s,a)\right)^2$$

Q is parameterized by weights w

$$\nabla_w L(s,a,s';w) \approx -\left(r + \gamma \max Q_w(s',a') - Q_w(s,a)\right) \nabla_w Q_w(s,a)$$

$$w \leftarrow w - \alpha \nabla_w L(s,a,s';w)$$

s, a, r, s'



x
feature

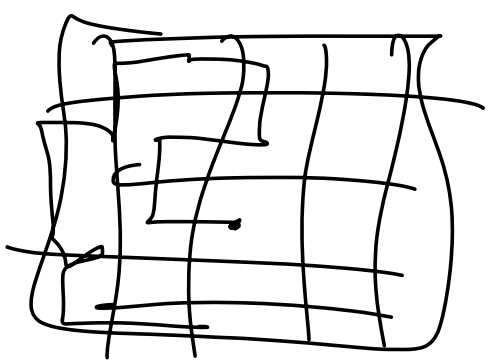
y
regressor

$(f_w(x) - y)^2$

s_0, a_0 $Q_w(s_0, a_0)$ $y_0 = r_T \gamma \max_{a \in A} Q_w(s', a)$

s_1, a_1 y_1

s_2 a_2 y_2



Bare bones DQN (Do Not Converge)

Initialize Q(*s,a;w*) with random weights
Repeat (for each episode):
  Initialize *s*
  Repeat (for each step of the episode):
    Choose *a* from *s* using policy derived from Q (e.g. e-greedy)
    Take action *a*, observe *r, s'*
    $w \leftarrow w - \alpha \nabla_w L(s, a, s'; w)$
    $s \leftarrow s'$
  Until *s* is terminal

Where:

$$\nabla_w L(s, a, s'; w) \approx -\left(r + \gamma \max_{a'} Q_w(s', a') - Q_w(s, a)\right) \nabla_w Q_w(s, a)$$

Deep Learning (and most Supervised learning) typically assume independent, identically distributed (iid) training data.

How about in the above algorithm? (general Deep RL)

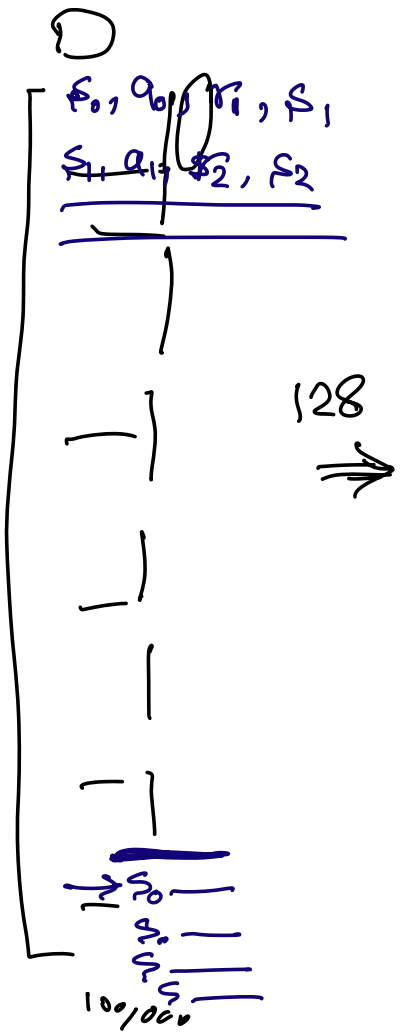
⇒ Solution: buffer experiences and then replay randomly them during training

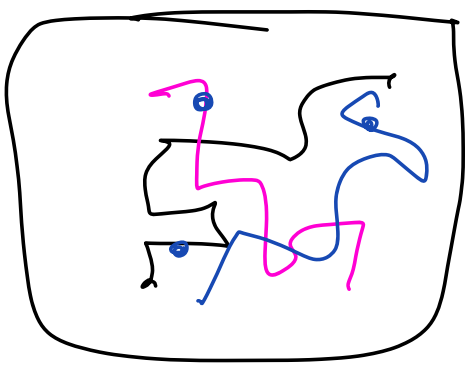
LETTER

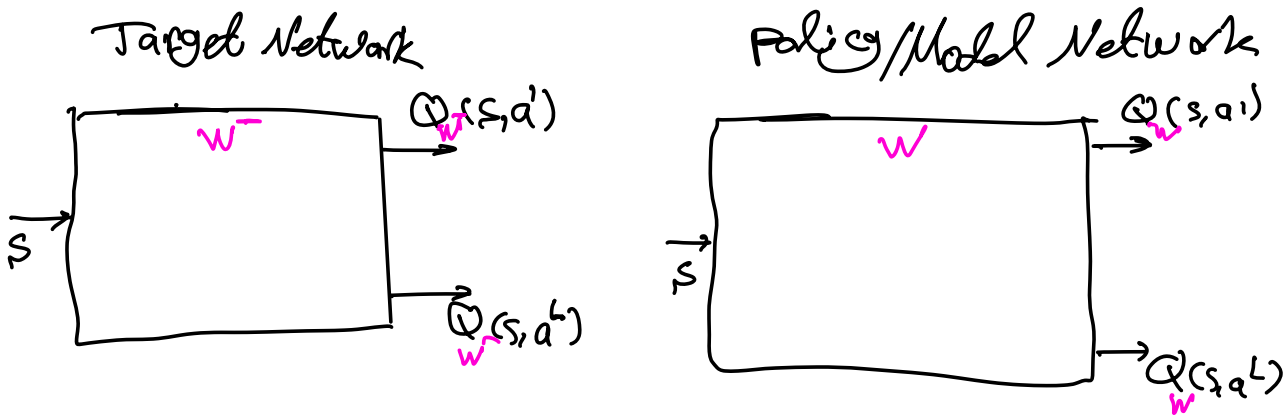
doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]





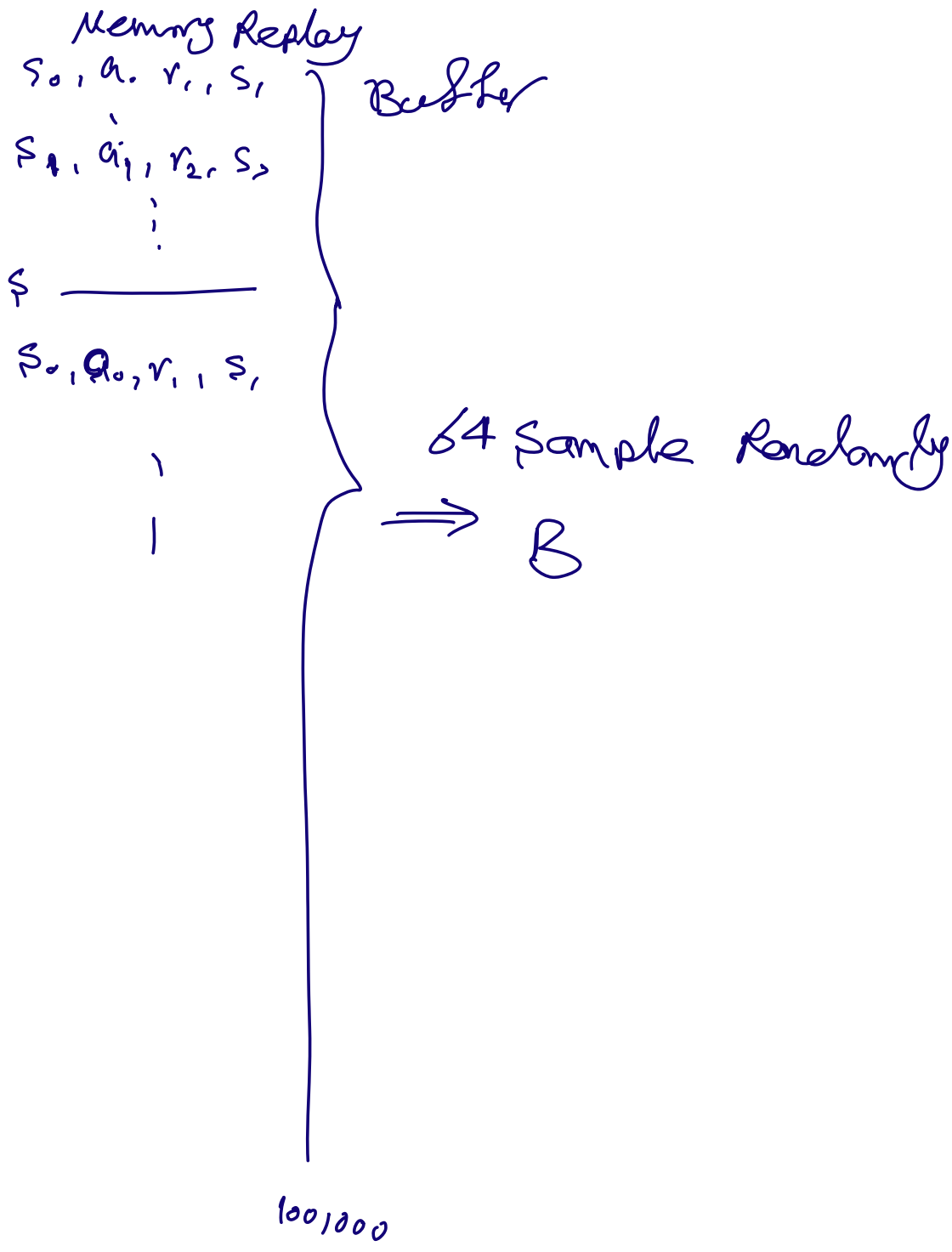


The same structure

$$a_0 \sim \pi(s_0) = \begin{cases} \arg\max_a Q_w(s_0, a) & 1-\varepsilon \\ \text{Random} & \varepsilon \end{cases}$$

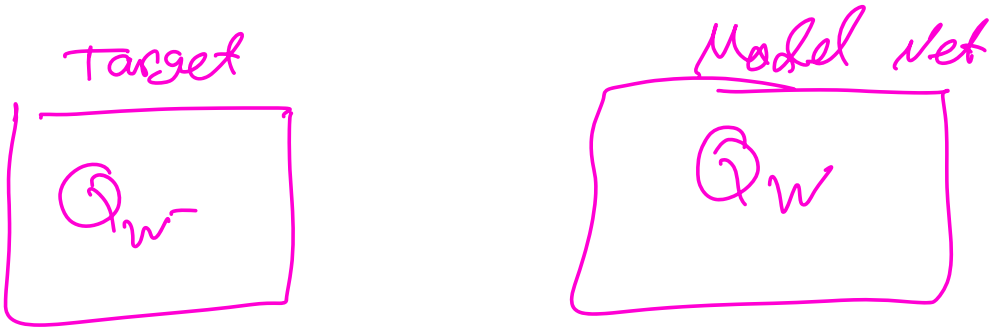
e.g.

s_0 — r_1 — s_1 — ⋯



$$L(B; w, w^-) = \frac{1}{B} \sum_{i} \left(r_{i+1} + \gamma \max_{a'} Q_{w^-}(s_{i+1}, a') - Q_w(s_i, a_i) \right)^2$$

$$w \leftarrow w + \alpha \frac{1}{B} \sum_{i=1}^{|B|} \left(y_i - Q_w(s_i, a_i) \right) \nabla Q_w(s_i, a_i)$$



$$r + \gamma \max_{a'} Q_{w^-}(s', a') = Q_w(s, a)$$

(B)

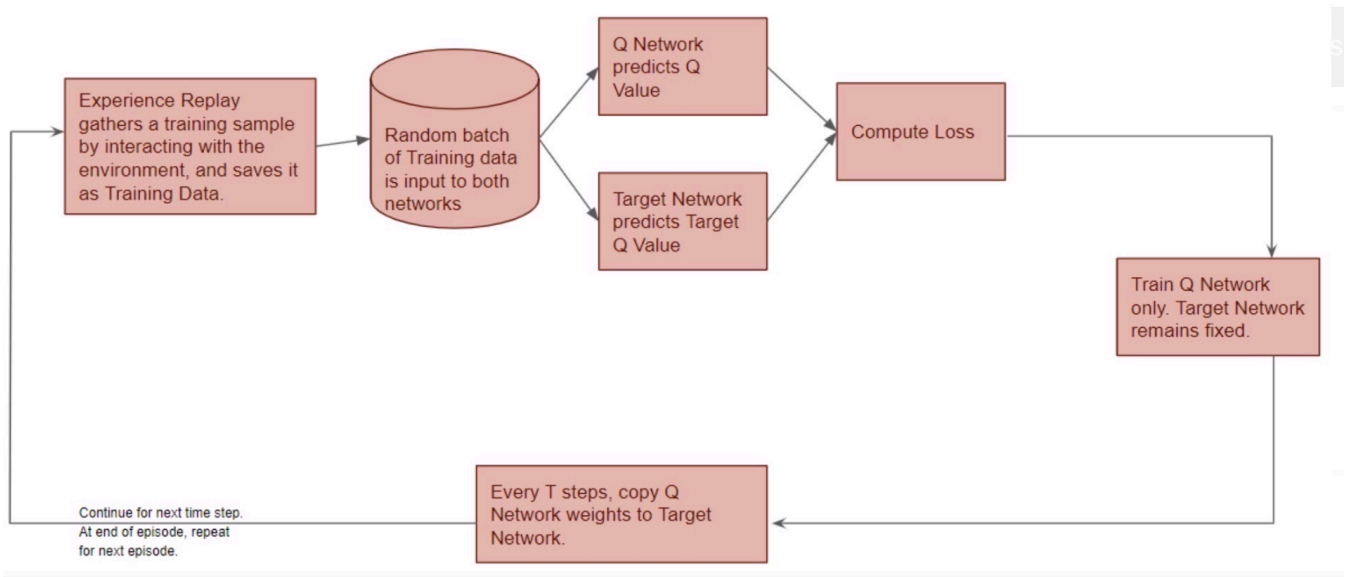
$s_1, a_1 \qquad y_1 = r_2 + \gamma \max Q_w(s_2', a')$

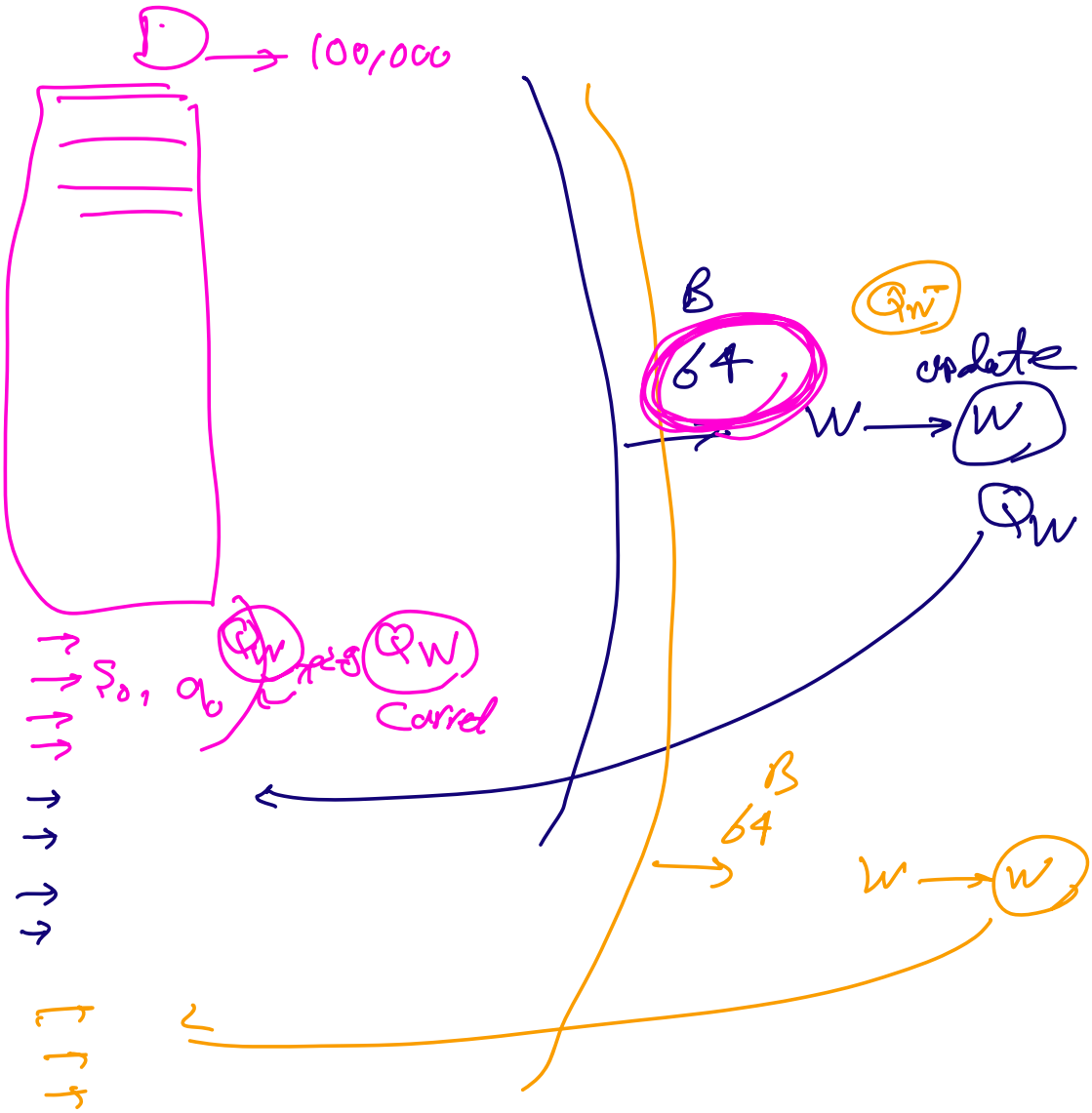
→

→

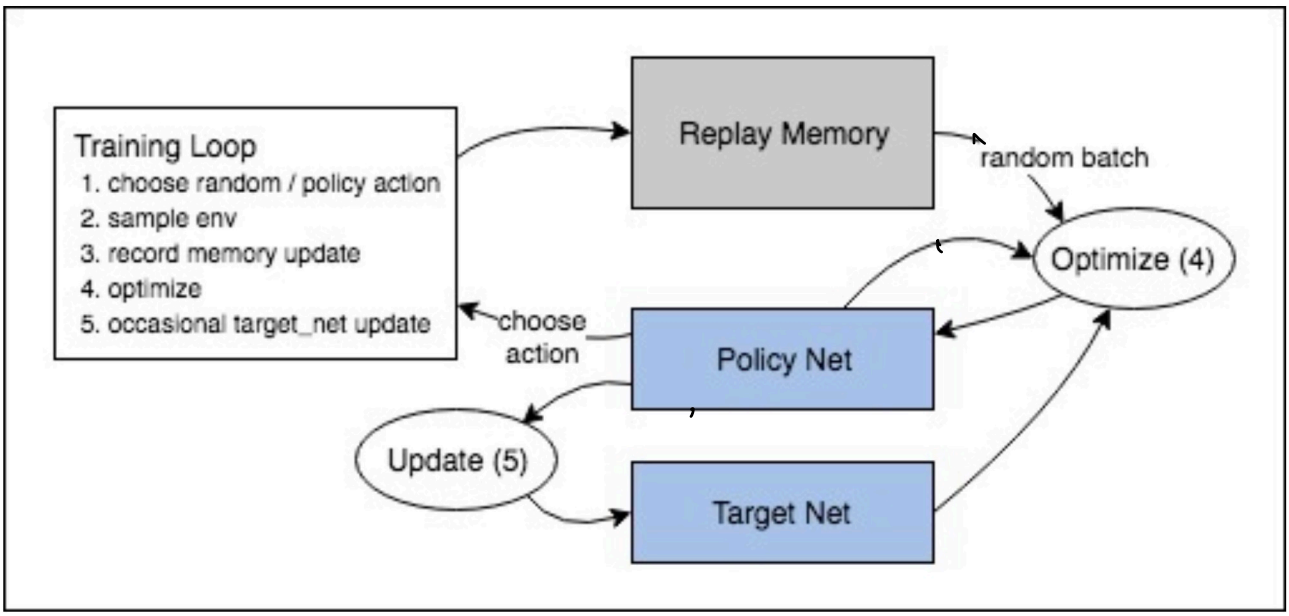
→

→


Experience Replay gathers a training sample by interacting with the environment, and saves it as Training Data.
Random batch of Training data is input to both networks
Q Network predicts Q Value
Target Network predicts Target Q Value
Compute Loss
Train Q Network only. Target Network remains fixed.
Every T steps, copy Q Network weights to Target Network.
Continue for next time step.
At end of episode, repeat for next episode.



$$\left(r + \gamma \max Q_{W^-}(s', a') - Q_W(s, a) \right)^2$$

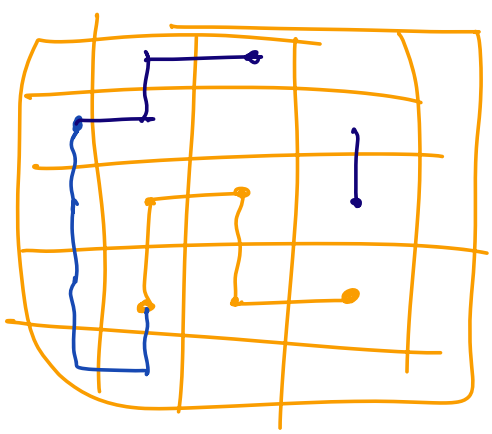

Training Loop
1. choose random / policy action
2. sample env
3. record memory update
4. optimize
5. occasional target_net update
Replay Memory
random batch
Optimize (4)
choose
action
Policy Net
Update (5)
Target Net

DQN

Initialize Q_w, Q_{w^-} with random weights
$D \leftarrow \emptyset$
Repeat (for each episode):
  Initialize *s*
  Repeat (for each step of the episode):
    Choose *a* from *s* using policy derived from Q (e.g., e-greedy)
    Take action *a*, observe *r, s'*
    $D \leftarrow D \cup (s, a, s', r)$
    $s \leftarrow s'$
    If mod(*step,trainfreq*) == 0:
      sample batch *B* from *D*
      $w \leftarrow w - \alpha \nabla_w L(B; w, w^-)$
      if mod(*step,copyfreq*) == 0:
        $w^- \leftarrow w$

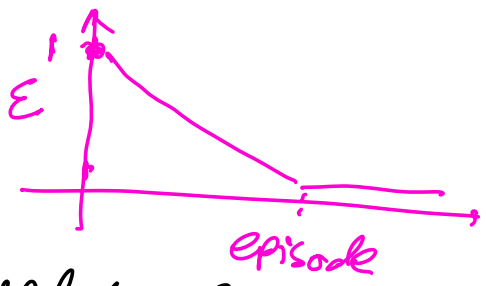
Where: $\nabla_w L(B; w, w^-) \approx -\frac{1}{|B|} \sum_{(s,a,s',r) \in B} \left(target(s'; w^-) - Q_w(s,a)\right) \nabla_w Q_w(s,a)$

$$target(s'; w^-) = r + \gamma \max_{a'} Q_{w^-}(s', a')$$



Two Tricks

→ Start $\varepsilon = 1$ and reduce it to 0.01



→ Soft update of $Q_{\bar{W}}$

$$\bar{W} \leftarrow W\tau + \bar{W}(1-\tau) \qquad \tau = 0.001$$

DQN

Initialize Q_w, Q_{w^-} with random weights
$D \leftarrow \emptyset$
Repeat (for each episode):
  Initialize *s*
  Repeat (for each step of the episode):
    Choose *a* from *s* using policy derived from Q (e.g., e-greedy)
    Take action *a*, observe *r, s'*
    $D \leftarrow D \cup (s, a, s', r)$
    $s \leftarrow s'$
    If mod(*step,trainfreq*) == 0:
      sample batch *B* from *D*
      $w \leftarrow w - \alpha \nabla_w L(B; w, w^-)$
      if mod(*step,copyfreq*) == 0:
        $w^- \leftarrow w$

Where: $\nabla_w L(B; w, w^-) \approx -\frac{1}{|B|} \sum_{(s,a,s',r) \in B} \left(target(s'; w^-) - Q_w(s, a)\right) \nabla_w Q_w(s, a)$

$$target(s'; w^-) = r + \gamma \max_{a'} Q_{w^-}(s', a')$$

Q_1: Large Buffer vs. Small Buffer

✗ old Data

✓ less likely correlated samples

✓ newer Data

✗ correlated

Q_2: Large Batch Size (B)

Slower

Stability

Small

Faster

Instability

Extended Data Table 1 | List of hyperparameters and their values

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which each stochastic gradient descent (SGD) update is computed.
replay memory size	1000000	SGD updates are sampled from this number of most recent frames.
agent history length	4	The number of most recent frames experienced by the agent that are given as input to the Q network.
target network update frequency	10000	The frequency (measured in the number of parameter updates) with which the target network is updated (this corresponds to the parameter *C* from Algorithm 1).
discount factor	0.99	Discount factor gamma used in the Q-learning update.
action repeat	4	Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame.
update frequency	4	The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates.
learning rate	0.00025	The learning rate used by RMSProp.
gradient momentum	0.95	Gradient momentum used by RMSProp.
squared gradient momentum	0.95	Squared gradient (denominator) momentum used by RMSProp.
min squared gradient	0.01	Constant added to the squared gradient in the denominator of the RMSProp update.
initial exploration	1	Initial value of ε in ε-greedy exploration.
final exploration	0.1	Final value of ε in ε-greedy exploration.
final exploration frame	1000000	The number of frames over which the initial value of ε is linearly annealed to its final value.
replay start size	50000	A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory.
no-op max	30	Maximum number of "do nothing" actions to be performed by the agent at the start of an episode.

The values of all the hyperparameters were selected by performing an informal search on the games Pong, Breakout, Seaquest, Space Invaders and Beam Rider. We did not perform a systematic grid search owing to the high computational cost, although it is conceivable that even better results could be obtained by systematically tuning the hyperparameter values.

Double DQN

Double Q-Learning → Tabular

Initialize $Q_1(s,a)$ and $Q_2(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, 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 DQN

→ DQN $\quad w \leftarrow w - \alpha \nabla_w L(B; w, \bar{w})$; $\bar{w}$

$$\nabla_w L(B;w,\bar{w}) = \frac{1}{B} \sum_{s,a,r,s' \in B} \Big(\text{target}(s',\bar{w}) - Q_w(s,a) \Big) \nabla_w Q_w(s,a)$$

$$\text{target}(s',\bar{w}) = r + \gamma \max_{a'} Q_{\bar{w}}(s',a')$$

↳ $\text{argmax}_{a'} Q_{\bar{w}}(s',a')$

Double DQN:

$$\nabla_w L(B;w,\bar{w}) = -\frac{1}{B} \sum_{s,a,r,s' \in B} \Big(\text{target}(s',a',w,\bar{w}) - Q_w(s,a) \Big) \nabla_w Q_w(s,a)$$

$$\text{target}(s',a',w,\bar{w}) = r + \gamma Q_{\bar{w}}\big(s', \underset{a'}{\text{argmax}}\, Q_w(s,a')\big)$$

Double DQN

Initialize Q_w, Q_{w^-} with random weights
$D \leftarrow \emptyset$
Repeat (for each episode):
    Initialize *s*
    Repeat (for each step of the episode):
        Choose *a* from *s* using policy derived from Q (e.g., e-greedy)
        Take action *a*, observe *r, s'*
        $D \leftarrow D \cup (s, a, s', r)$
        $s \leftarrow s'$
        If mod(*step,trainfreq*) == 0:
            sample batch *B* from *D*
            $w \leftarrow w - \alpha \nabla_w L(B; w, w^-)$
            if mod(*step,copyfreq*) == 0:
                $w^- \leftarrow w$

Where: $\nabla_w L(B; w, w^-) \approx -\frac{1}{|B|} \sum_{(s,a,s',r) \in B} \left(target(s', a'; w, w^-) - Q_w(s, a) \right) \nabla_w Q_w(s, a)$

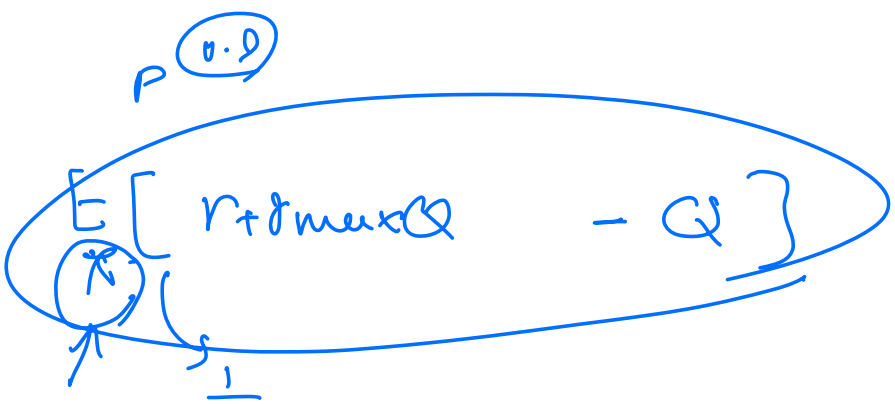
$$target(s', a'; w, w^-) = r + \gamma Q_{w^-}(s', \arg\max_{a'} Q_w(s', a'))$$

Prioritized DQN

D Buffer

s_0, a_0, r_1, s_1	$\rightarrow \delta_1 = \lvert r_1 + \gamma \max_{a'} Q_{w^-}(s_1, a') - Q_w(s_0, a_0) \rvert$
$s_1\ a_1\ r_2\ s_2$	$\rightarrow \delta_2 = \lvert r_2 + \gamma \max_{a' \in A} Q_{w^-}(s_2, a') - Q_w(s_1, a_1) \rvert$
$\vdots$	
$s_D\ a_D\ r_{D+1} s_{D+1}$	$\delta_D = \lvert r_{D+1} + \gamma \max_{a' \in A} Q_{w^-}(s_{D+1}, a') - Q_w(s_D, a_D) \rvert$

$$w \leftarrow w + \frac{1}{B} \sum_{B} \left(r + \gamma \max_{a'} Q_{w^-}(s', a') - Q_w(s, a) \right) \nabla_w Q_w(s, a)$$



D Buffer

s_0, a_0, r_1, s_1 → $\delta_1 = | r_1 + \gamma \max_{a'} Q_{\bar{w}}(s_1, a') - Q_w(s_0, a_0) |$

$s_1\ a_1\ r_2\ s_2$ → $\delta_2 = | r_2 + \gamma \max_{a' \in A} Q_{\bar{w}}(s_2, a') - Q_w(s_1, a_1) |$

⋮

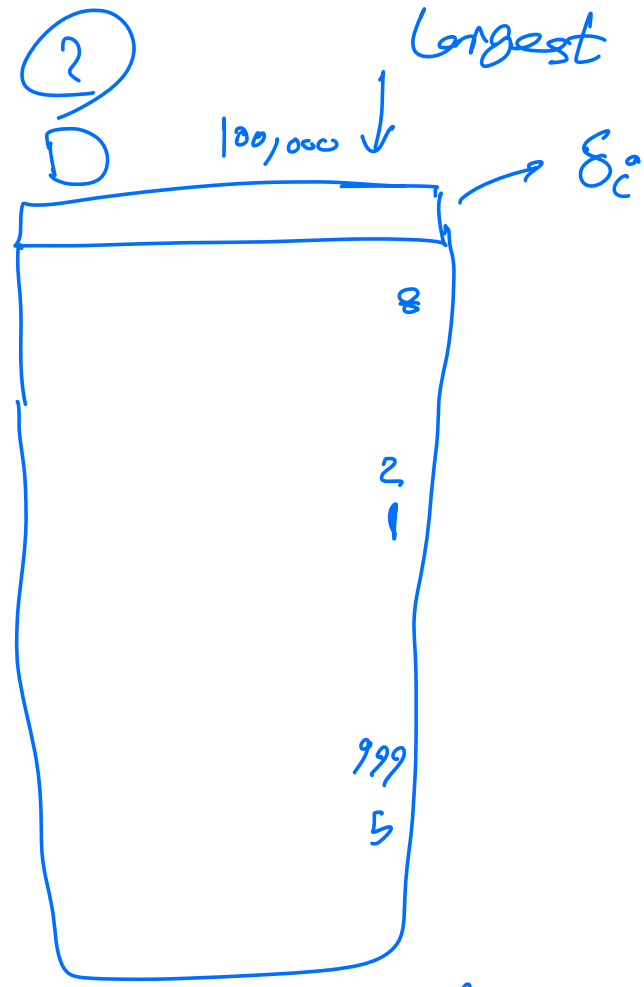
$s_D\ a_D\ r_{D+1} s_{D+1}$ | $\delta_D = | r_{D+1} + \gamma \max_{a' \in A} Q_{\bar{w}}(s_{D+1}, a') - Q_w(s_D, a_D) |$

$P \propto [\ \delta_1 + \epsilon \qquad \delta_{|D|} + \epsilon\ ]$

$$P(i) = \frac{\delta_i + \epsilon}{\sum_{j=1}^{|D|} \delta_j + \epsilon}$$

0.9
0.01
0.01 ⟹
⋮

Ranked-Based



$$P(i) \propto \frac{1}{Rank(i)}$$

③

$$P_i = \frac{P(i)^{\alpha}}{\sum_{j=1}^{D} P(i)^{\alpha}}$$

Rank → α ; δ_i

α: Level of Prioritization

$\alpha = 0 \Rightarrow$ DQN

$\alpha = 1$: Full Prioritized

$$\min_w \mathbb{E}_{s,a,s',r \in D}\left[\left(r + \gamma \max_{a'} Q_{w^-}(s',a') - Q_w(s,a)\right)^2\right]$$

$$\nabla L(B; w, w^-) = -\frac{1}{|B|} \sum_{s,a,r,s' \in B} \frac{1}{p_{s,a,r,s'}} \left(r + \gamma \max_{a'} Q_{w^-}(s',a') - Q_w(s,a)\right) \nabla_w Q_w$$