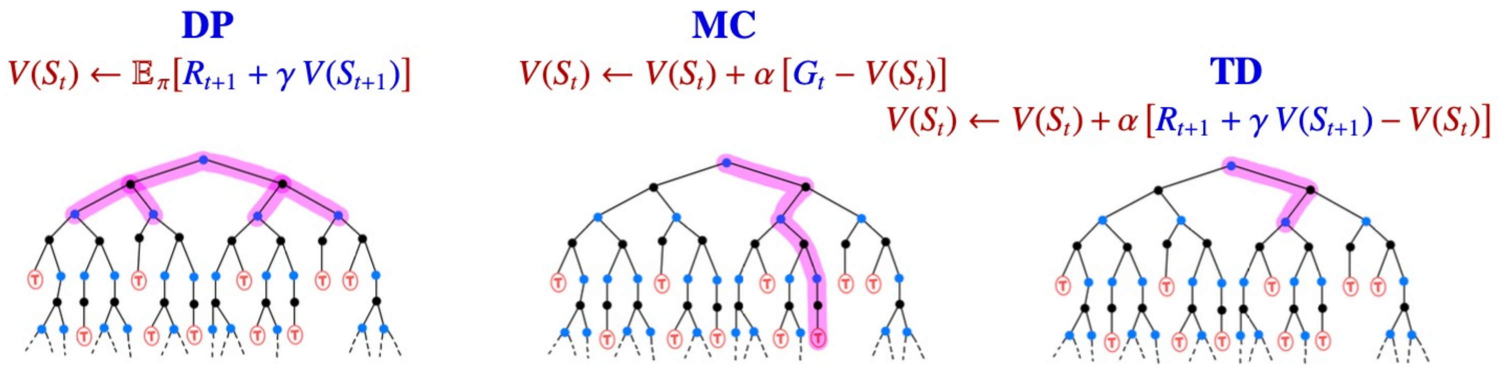
DQN Review
Reinforcement Learning Review

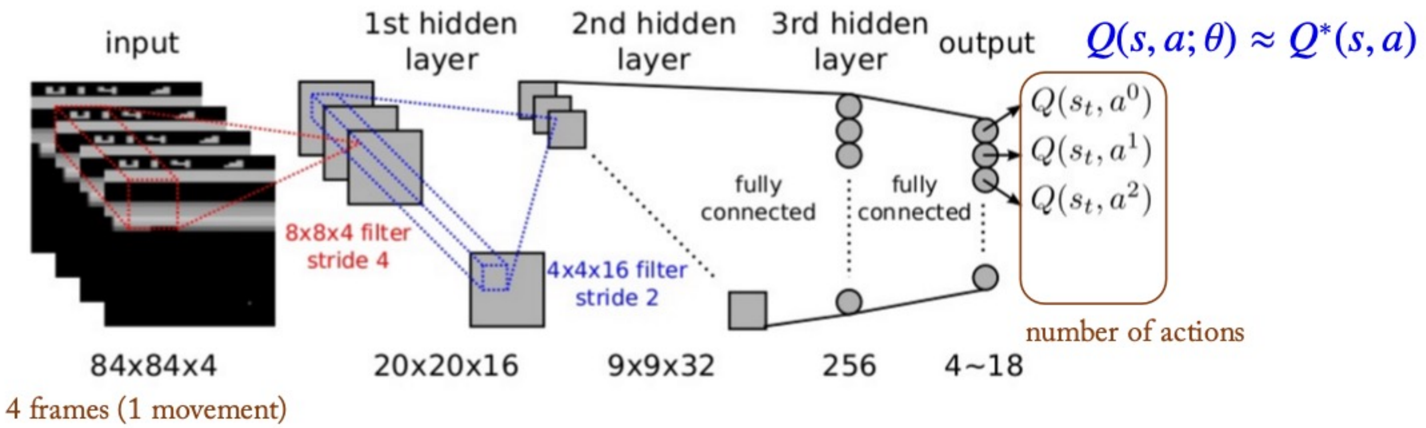
Based on Prof. Oh's Reinforcement Learning Lectures

Suinne Lee

Deep Q Network (DQN)

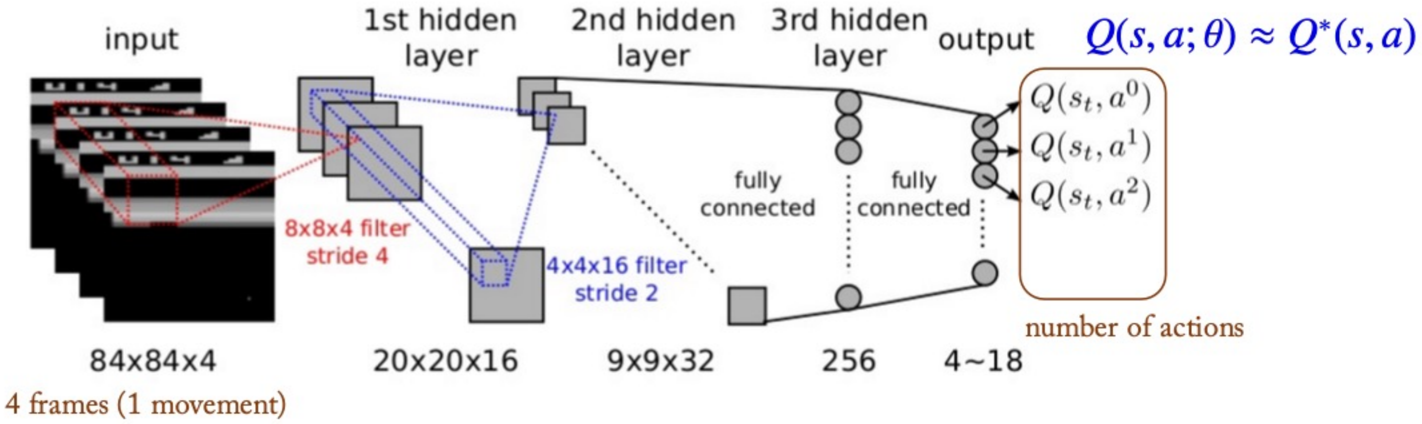
- CNN
- Sampling!





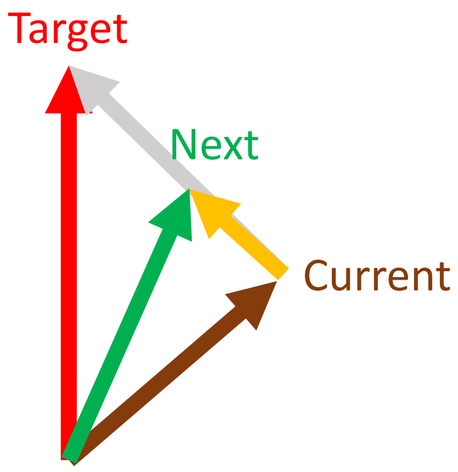
Deep Q Network (DQN)

- CNN
- Sampling!
- One (to a few) states are input, and we do not have to consider the whole state space. → good when the state space is large or continuous.
- Once trained,
  one forward pass
  is all that is needed.



The Maths

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



Cf. Bellman Optimality Eq

$$q_*(s,a) = \sum_{s',r} p(s',r \mid s,a)[r + \gamma \max_{a'} q_*(s',a')]$$

- We use MSE loss and perform SGD:

$$L(\theta) = [r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta) - Q(s_t, a_t; \theta)]^2$$

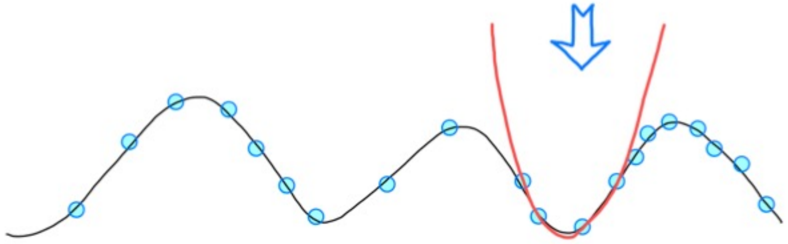
Problems

- Forgetting useful experiences
- Temporal correlation
- Non-stationary target

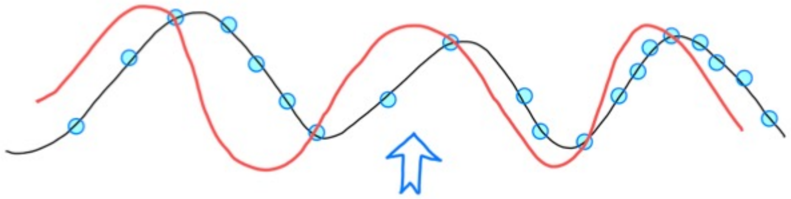
- **Solutions:**
- **Experience Replay**
- **Target Network**

Experience Replay (Replay Buffer)

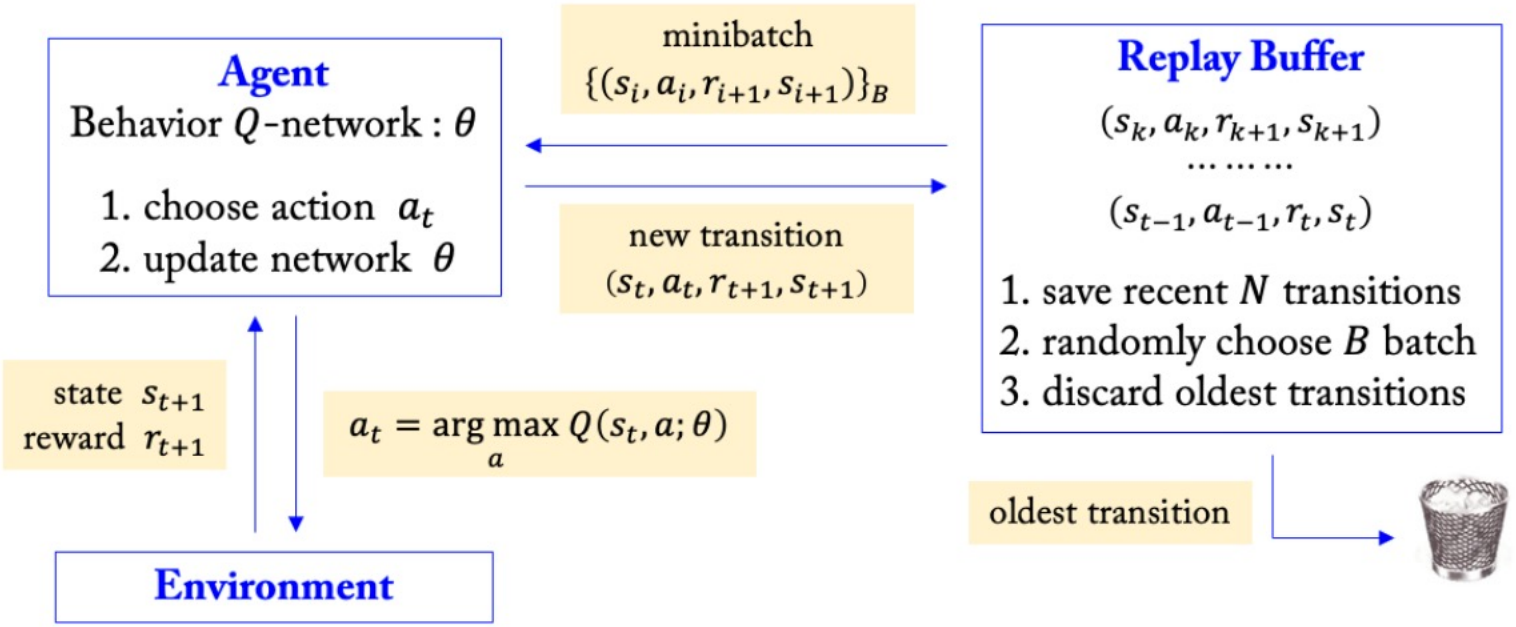
- Stores past experiences ~ 10^5 (s,a,r,s’) 4-tuples
- To resolve issues where we forget temporally correlated or useful experiences.



☺



- Reduces temporal correlation (reduces bias due to temporal correlation)
- Data efficiency
- Learning speed



Agent
Behavior Q-network : θ
1. choose action a_t
2. update network θ
minibatch
$\{(s_i, a_i, r_{i+1}, s_{i+1})\}_B$
new transition
$(s_t, a_t, r_{t+1}, s_{t+1})$
Replay Buffer
$(s_k, a_k, r_{k+1}, s_{k+1})$
$\cdots\cdots\cdots$
$(s_{t-1}, a_{t-1}, r_t, s_t)$
1. save recent N transitions
2. randomly choose B batch
3. discard oldest transitions
state s_{t+1}
reward r_{t+1}
$a_t = \arg\max_a Q(s_t, a; \theta)$
Environment
oldest transition

Target Network

- We introduce a second network (target network) where we fix the parameters $\hat{\theta}$ for a number of steps. (E.g. 1000) (whereas the parameters θ are updated every B (minimbatch size.)

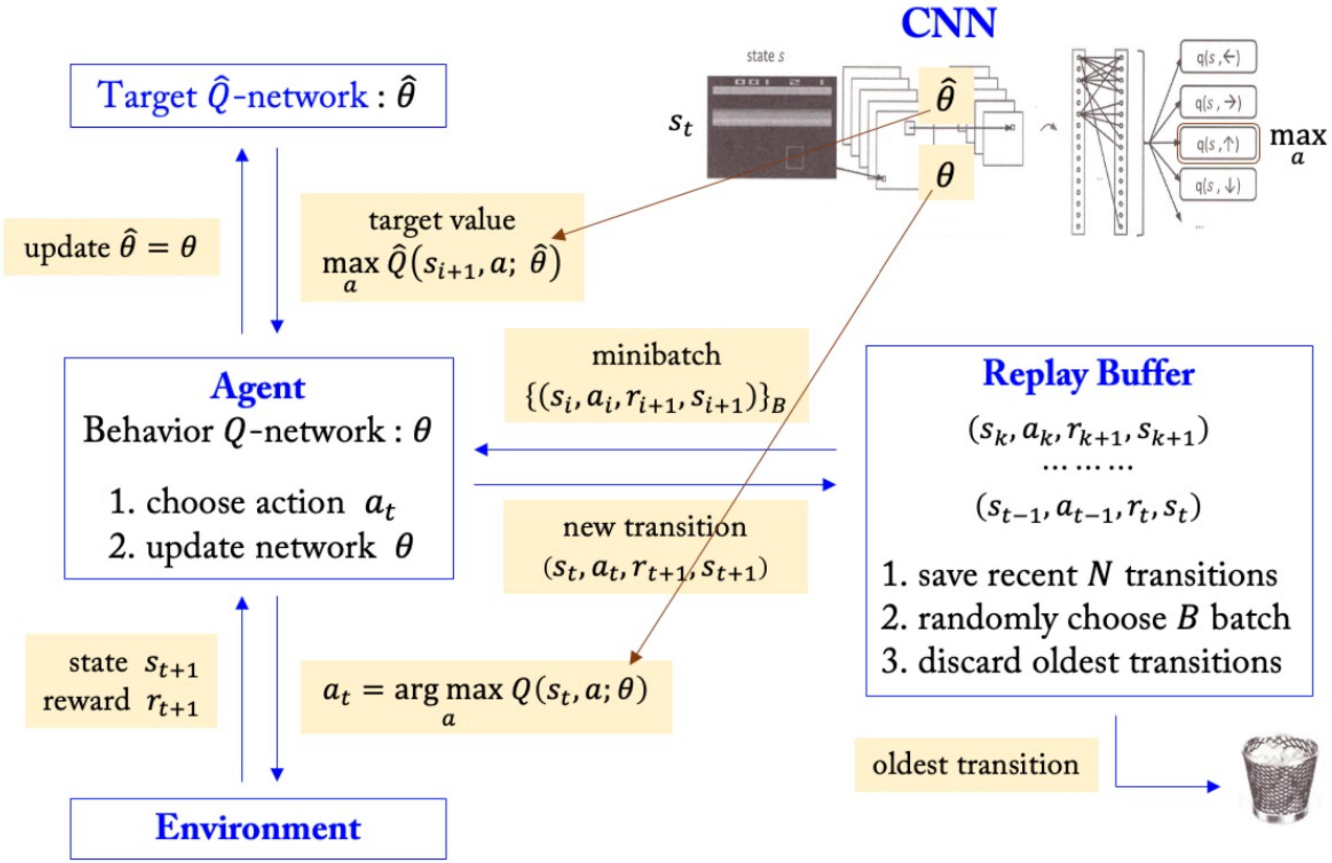
- Forward pass (loss function)

$$L(\theta) = \frac{1}{B} \sum_{|\{i\}|=B} \left[r_{i+1} + \gamma \max_a \hat{Q}(s_{i+1}, a; \hat{\theta}) - Q(s_i, a_i; \theta)\right]^2$$

- Backward pass (gradient descent w.r.t. parameters θ)

$$-\nabla_\theta L(\theta) = \frac{1}{B} \sum_{|\{i\}|=B} \left[r_{i+1} + \gamma \max_a \hat{Q}(s_{i+1}, a; \hat{\theta}) - Q(s_i, a_i; \theta)\right] \nabla_\theta Q(s_i, a_i; \theta)$$

- DQN weight update $\theta := \theta - \alpha \nabla_\theta L(\theta)$



CNN
state s
s_t
$\hat{\theta}$
θ
$q(s,\leftarrow)$
$q(s,\rightarrow)$
$q(s,\uparrow)$
$q(s,\downarrow)$
$\max_a$
Target $\hat{Q}$-network : $\hat{\theta}$
update $\hat{\theta} = \theta$
target value
$\max_a \hat{Q}(s_{i+1}, a;\ \hat{\theta})$
minibatch
$\{(s_i, a_i, r_{i+1}, s_{i+1})\}_B$
Agent
Behavior Q-network : θ
1. choose action a_t
2. update network θ
new transition
$(s_t, a_t, r_{t+1}, s_{t+1})$
Replay Buffer
$(s_k, a_k, r_{k+1}, s_{k+1})$
$\cdots\cdots\cdots$
$(s_{t-1}, a_{t-1}, r_t, s_t)$
1. save recent N transitions
2. randomly choose B batch
3. discard oldest transitions
state s_{t+1}
reward r_{t+1}
$a_t = \arg\max_a Q(s_t, a; \theta)$
oldest transition
Environment

DQN Variants

- Multi-step learning
- Double DQN
- Prioritized Replay
- Dueling DQN
- Dueling Double DQN

Multi-step learning

- Looking ahead more steps

→ Instead of using the TD target, we use:

$$G_{t:t+n} = \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} + \gamma^n \max_a Q(s_{t+n}, a; \theta^-)$$

- Truncated n-step return $r_{t+1}^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k+1}$

- Loss: $L(\theta) = [r_{t+1}^{(n)} + \gamma^n \max_a \hat{Q}(s_{t+n}, a; \hat{\theta}) - Q(s_t, a_t; \theta)]^2$

Double DQN

- Resolves overestimation of Q (due to max operation)
- Splits max operation to **action selection** and **action evaluation**.

$$\text{DQN:}\ \ L(\theta) = \left[r_{t+1} + \gamma \max_a \hat{Q}(s_{t+1}, a; \hat{\theta}) - Q(s_t, a_t; \theta)\right]^2$$

$$\text{Double DQN:}\ \ L(\theta) = \left[r_{t+1} + \gamma\, \hat{Q}(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \hat{\theta}) - Q(s_t, a_t; \theta)\right]^2$$

- Make use of double (behavior / target) networks for this!

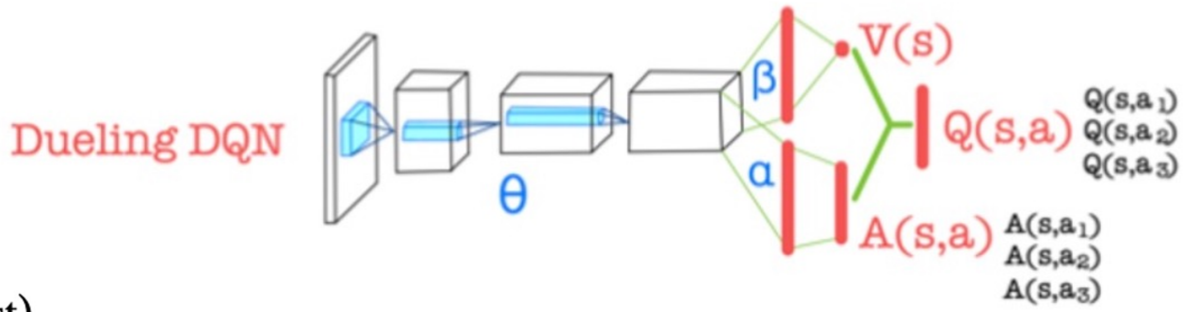
Prioritized Replay

- Prioritize important experiences from the replay buffer!
- Priority is based on TD error:

- But several problems persist, such as the TD error continuing to be high if initially high / biases occuring

Dueling DQN

- Networks splitting and giving features V(s) and A(s,a) (state value and advantage)
- Useful in contexts where state value function is noteworthy.
- Identifiability issue:



$$Q(s,a) = (V(s) + \text{const}) + (A(s,a) - \text{const})$$

→ $$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left[A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\alpha)\right]$$

Dueling Double DQN

- https://dnai-deny.tistory.com/93
- https://towardsdatascience.com/dueling-double-deep-q-learning-using-tensorflow-2-x-7bbbcec06a2a