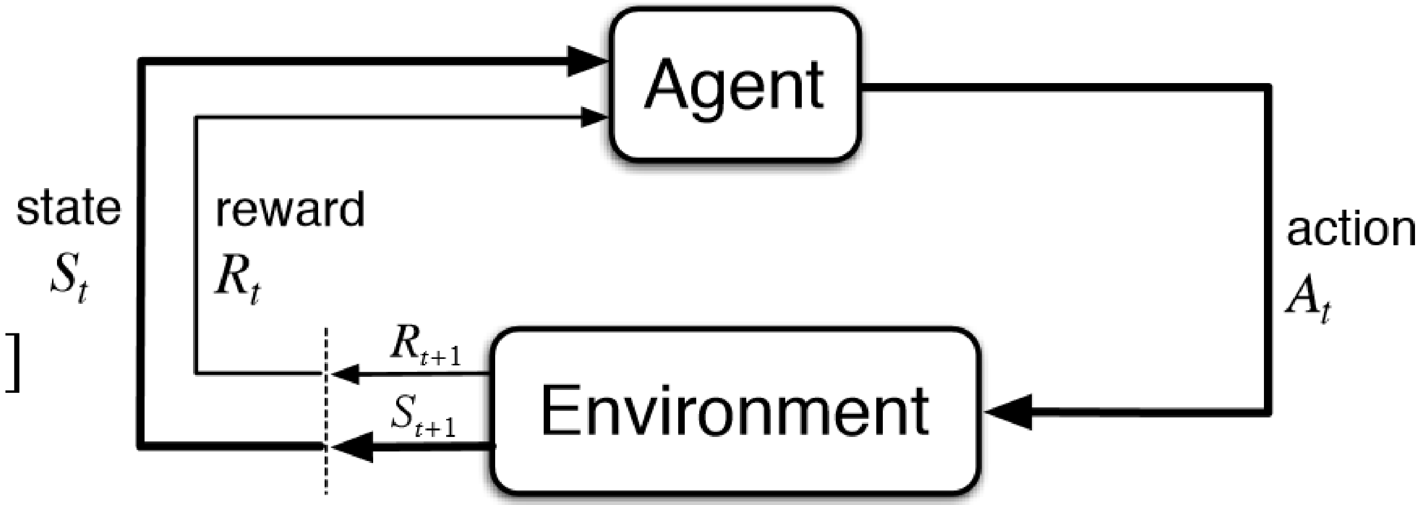
Review – RL formalized as MDP



Markov property

$$p[s_{t+1}|s_t] = p[s_{t+1}|s_1, s_2, \dots s_t]$$

Value function

$$V_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

Markov process (chain)

$$s_1, s_2, \dots s_t$$

https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da

Review – MDP is solved by Bellman Equation

$$V_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s]$$

$$= R_{\pi,s} + \gamma \sum_{s'} P(s'|s) V_\pi(s')$$

$$= \sum_a \pi(a|s) Q_\pi(s,a)$$

$$Q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]$$

$$= R_s^a + \gamma \sum_{s'} P(s'|s,a) V_\pi(s')$$

https://towardsdatascience.com/introduction-to-reinforcement-learning-markov-decision-process-44c533ebf8da

Bellman Optimal Equation

$$V^*(s) = \max_a \sum P(s'|s,a)(R_s^a + \gamma V^*(s'))$$

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)\,(R_s^a + \gamma \max_{a'} Q^*(s',a'))$$

https://markus-x-buchholz.medium.com/deep-reinforcement-learning-introduction-deep-q-network-dqn-algorithm-fb74bf4d6862

Q-learning algorithm to find $Q^*(s,a)$

1. Initialize Table Q(S,A) with random values.
2. Take a action (A) with epsilon — greedy policy and move to next state S’
3. Update the Q value of a previous state by following the update equation:

$$Q(s,a) = Q(s,a) + \alpha(R_s^a + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

https://markus-x-buchholz.medium.com/deep-reinforcement-learning-introduction-deep-q-network-dqn-algorithm-fb74bf4d6862

Value iteration converges

Theorem. Value iteration converges. At convergence, we have found the optimal value function V^* for the discounted infinite horizon problem, which satisfies the Bellman equations

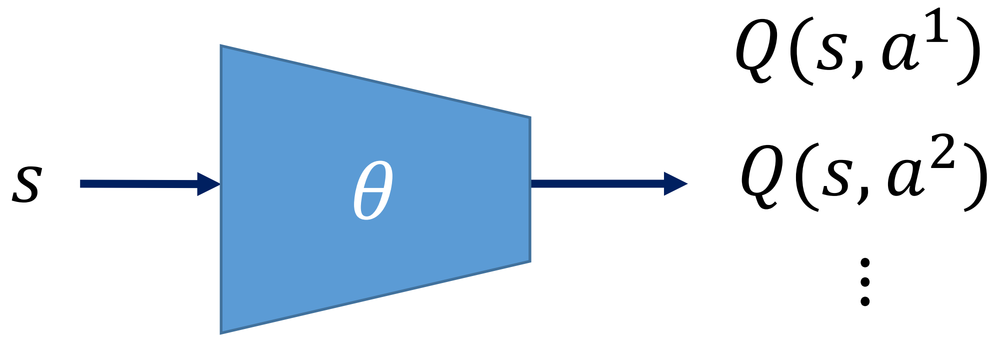
$$\forall s \in S \quad V^*(\mathrm{s}) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')]$$

V^*, in turn, tells us how to act:

$$\pi^*(\mathrm{s}) = arg \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

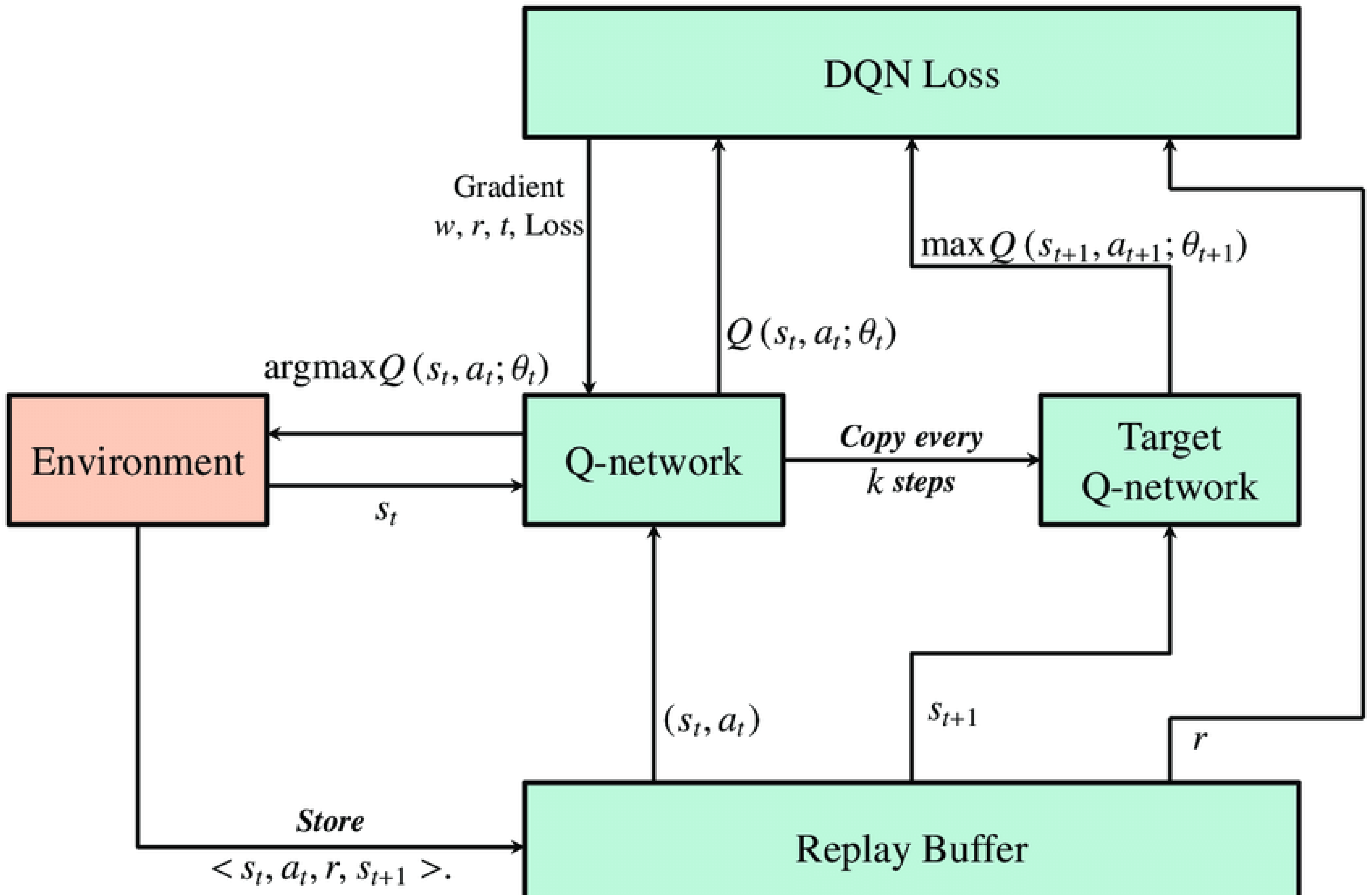
Deep Q-Network (DQN)

Q-table is replaced by Deep Neural Network (Q-Network) which maps (non-linear approximation) environment states to Agent actions.



Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529.

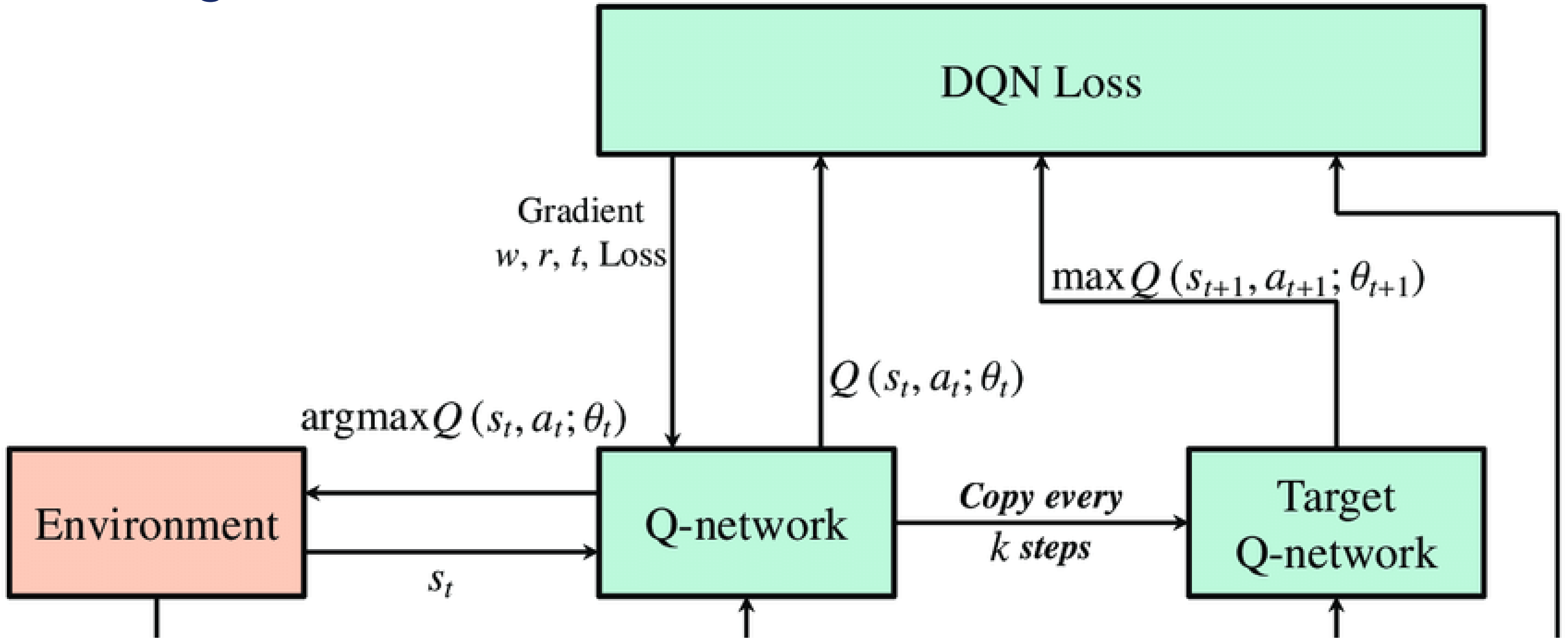
DQN architecture



https://www.researchgate.net/figure/A-Concept-of-DQN-Algorithm7_fig4_348444080

Stabilize DQN training

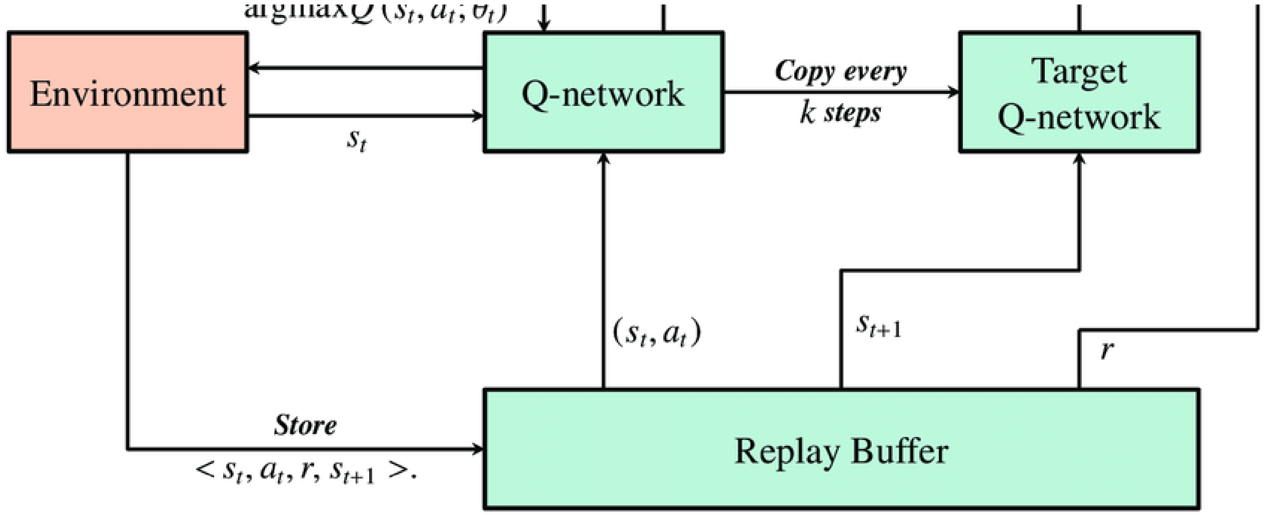
The target network is frozen for several time steps and then the target network weights are updated by copying the weights from the actual Q network. Freezing the target Q-network for a while and then updating its weights with the actual Q network weights stabilizes the training.



https://markus-x-buchholz.medium.com/deep-reinforcement-learning-introduction-deep-q-network-dqn-algorithm-fb74bf4d6862

Stabilize DQN training (2)

In order to make training process more stable (we would like to avoid learning network on data which is relatively correlated, which can happen if we perform learning on consecutive updates - last transition) we apply replay buffer which memorizes experiences of the Agent behavior. Then, training is performed on random samples from the replay buffer (this reduces the correlation between the agent's experience and helps the agent to learn better from a wide range of experiences).



https://markus-x-buchholz.medium.com/deep-reinforcement-learning-introduction-deep-q-network-dqn-algorithm-fb74bf4d6862

Use DQN to learn push block

8. DQN to learn Push block (1) (MLAgent_10).ipynb

DQN algorithm

1. Initialize replay buffer.
2. Send s to DQN and select an action using the epsilon-greedy policy. Select an action that has a maximum Q value.
3. Agent performs chosen action and move to a new state S' and receive a reward R.
4. Store transition in replay buffer.
5. Sample batches of transitions from the replay buffer and calculate the loss.
6. Perform gradient descent to minimize the loss.
8. After every k steps, copy our actual network weights to the target network weights.
9. Repeat these steps for M number of episodes.

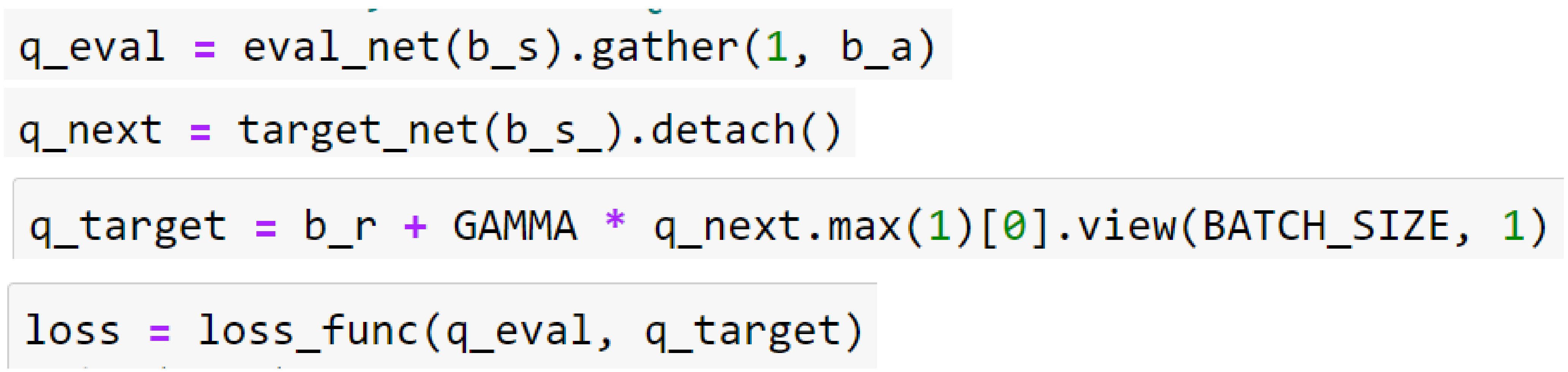
$$Loss = \left(R_s^a + \gamma \max_{a'} Q(s', a') - Q(s, a)\right)^2$$

https://markus-x-buchholz.medium.com/deep-reinforcement-learning-introduction-deep-q-network-dqn-algorithm-fb74bf4d6862

Use DQN to learn push block

8. DQN to learn Push block (2) (MLAgent_10).ipynb

5. Sample batches of transitions from the replay buffer and calculate the loss.



```
q_eval = eval_net(b_s).gather(1, b_a)
```

```
q_next = target_net(b_s_).detach()
```

```
q_target = b_r + GAMMA * q_next.max(1)[0].view(BATCH_SIZE, 1)
```

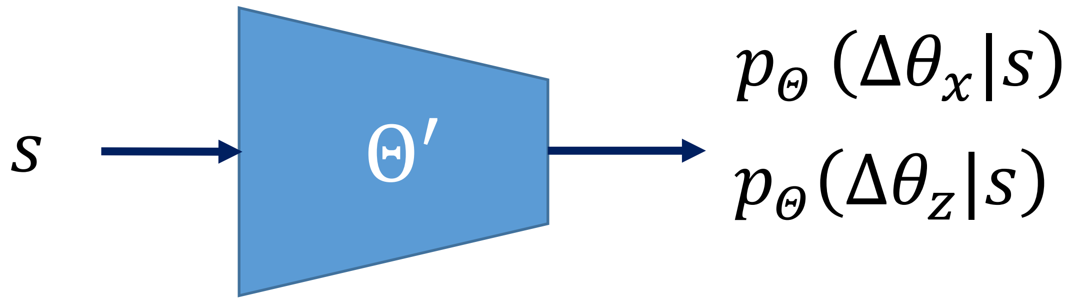
```
loss = loss_func(q_eval, q_target)
```

$$Loss = \left(R_s^a + \gamma \max_{a'} Q(s', a') - Q(s, a)\right)^2$$

Comparison - policy based RL

$$\nabla \bar{R}_{\Theta} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'}^{T_n}\gamma^{t'-t}r_{t'}^{n} - b\right)\nabla \log p_{\Theta}(a_t^n|s_t^n)$$

$$\Theta' \leftarrow \Theta + \eta\nabla \bar{R}_{\Theta}$$


s
Θ'
$p_{\Theta}(\Delta\theta_x|s)$
$p_{\Theta}(\Delta\theta_z|s)$

$$\nabla \bar{R}_{\Theta'} \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'}^{T_n}\gamma^{t'-t}r_{t'}^{n} - b\right)\nabla \log p_{\Theta'}(a_t^n|s_t^n)$$

Review: Use expected value to reduce sampling variance

$\boxed{V^{\pi_\theta}(s_t^n)}$ Expected value of b

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underbrace{\boxed{\sum_{t'}^{T_n}\gamma^{t'-t}r_{t'}^{n}}}_{\boxed{E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)}} - \overbrace{\boxed{b}}^{\boxed{V^{\pi_\theta}(s_t^n)}}\right)\nabla \log p_\theta(a_t^n|s_t^n)$$

$\boxed{E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)}$ Expected value of G_t^n

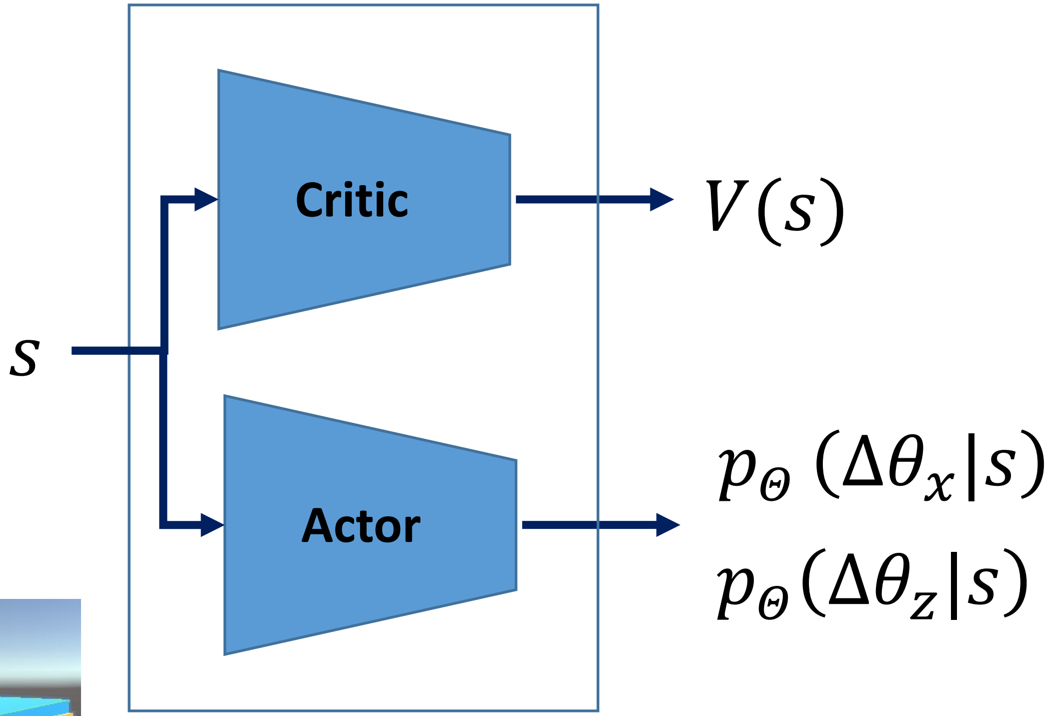
$$G_t^n = \sum_{t'}^{T_n}\gamma^{t'-t}r_{t'}^{n}$$

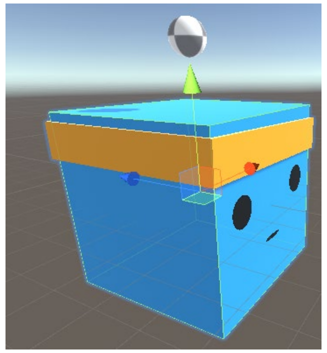
unstable when sampling amount is not large enough

Ref: 李弘毅 https://youtu.be/j82QLgfhFiY

Review: A2C (Advantage Actor Critic)

Actor – Learns the best actions (that can have maximum long-term rewards)
Critic – Learns the expected value of the long-term reward.


s
Critic
Actor
$V(s)$
$p_{\Theta}(\Delta\theta_x|s)$
$p_{\Theta}(\Delta\theta_z|s)$



8-128-128-2