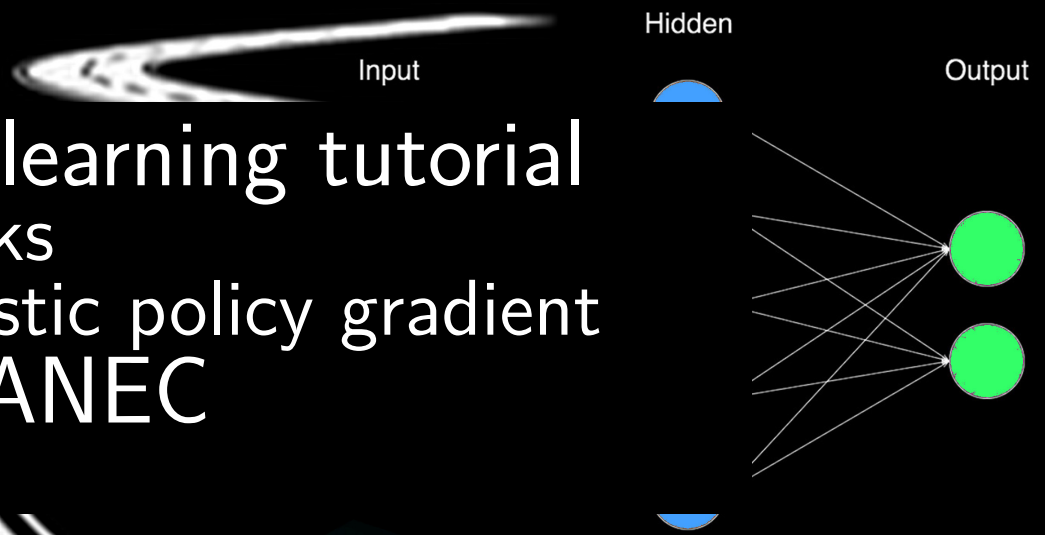
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

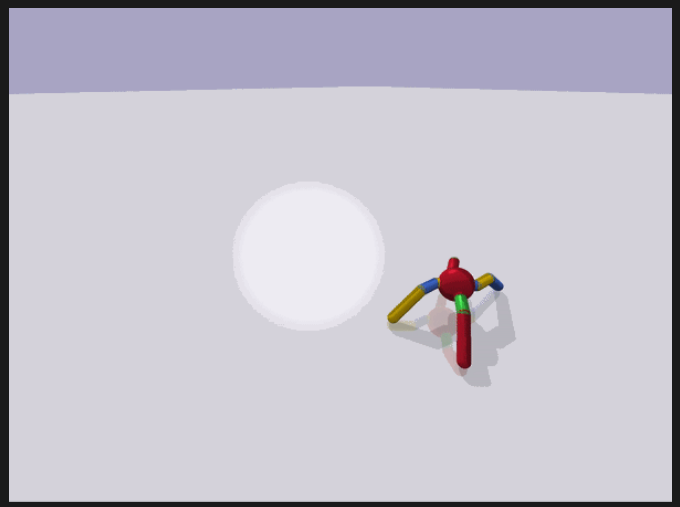
(The New Action Value = The Old Value) + The Learning Rate × (The New Information − the Old Information)

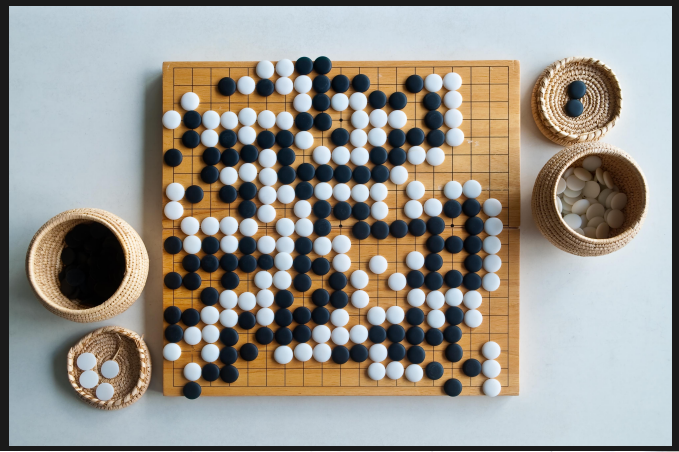
reinforcement learning tutorial
- deep Q networks
- deep deterministic policy gradient

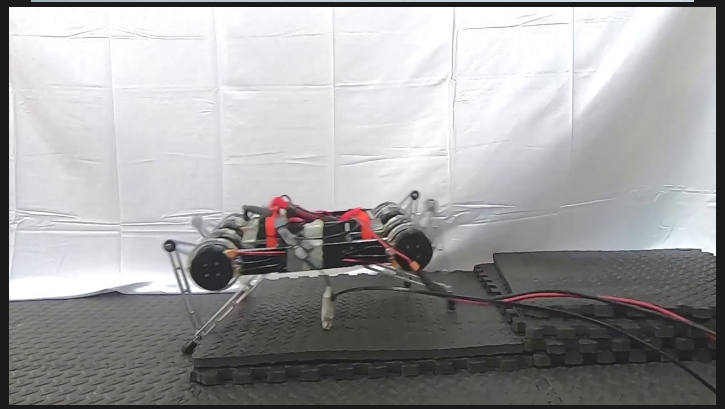
Michal CHOVANEC



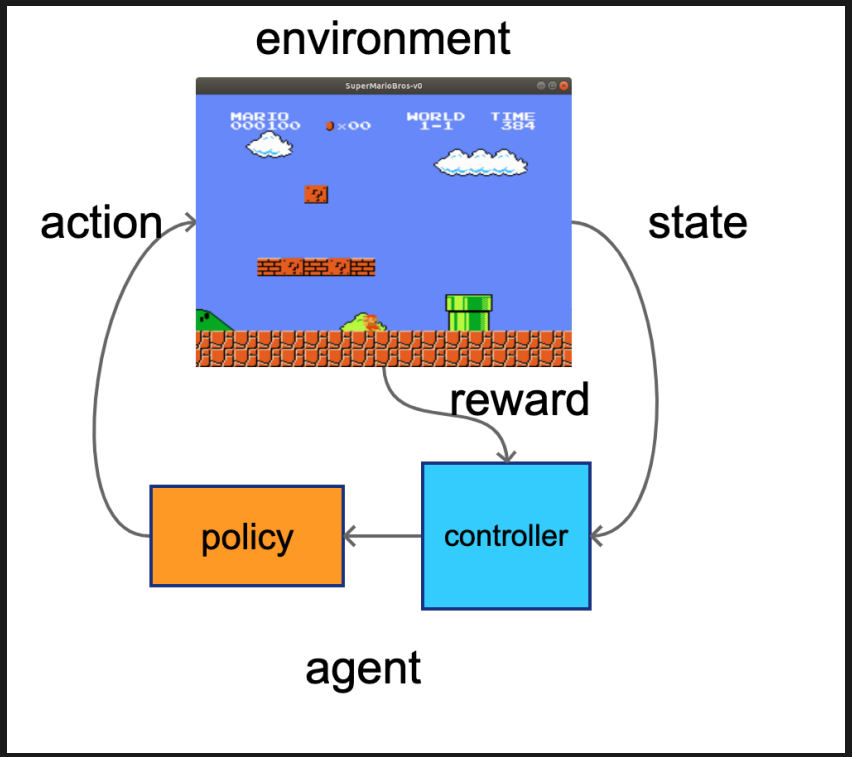
reinforcement learning







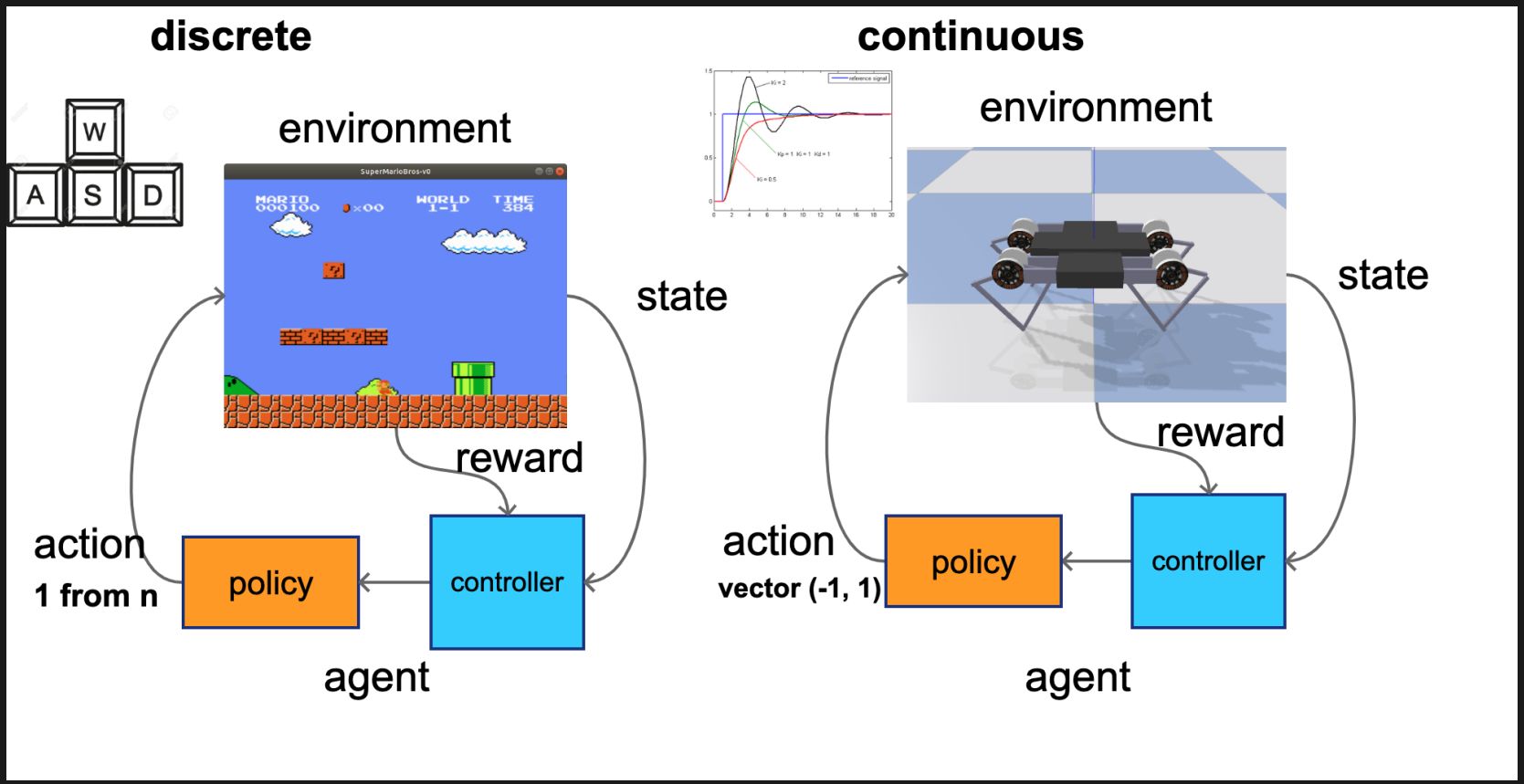
reinforcement learning



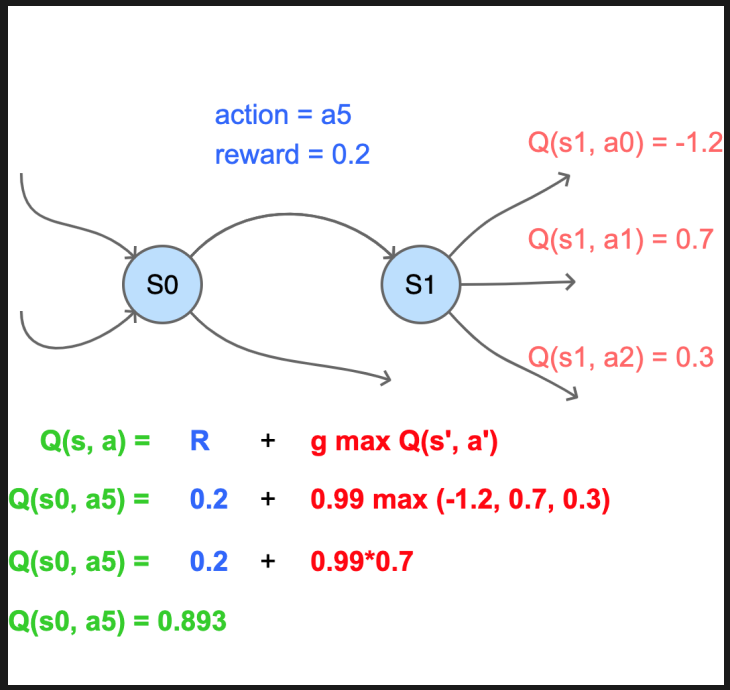
- obtain state
- select action
- exectute action
- learn from experiences

action space

- discrete action space
  - keys, keypad
- continuous action space
  - motors, PWMs, steering, force controll



deep Q learning



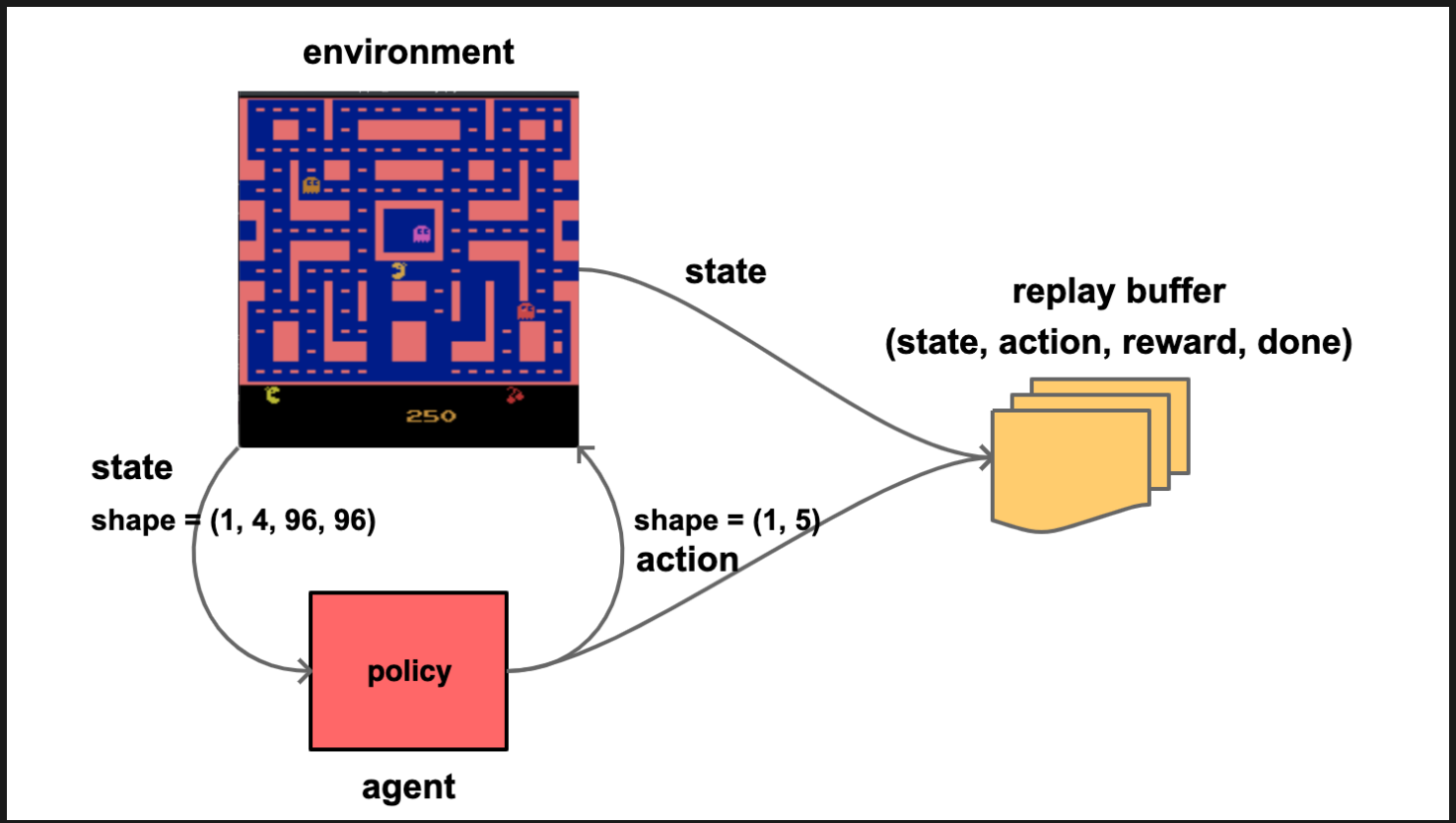
- play games with any policy
- store to buffer : **state, action, reward, done**
- sample random batch
- obtain q-values and next q-values
- use q-learning to update q-values
- train model

$$Q(s,a;\theta) = \underbrace{R}_{reward} + \underbrace{\gamma \max_{a'} Q(s',a';\theta^-)}_{discounted\ future\ reward}$$

$$\mathcal{L}(\theta) = \left(R + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta) \right)^2$$

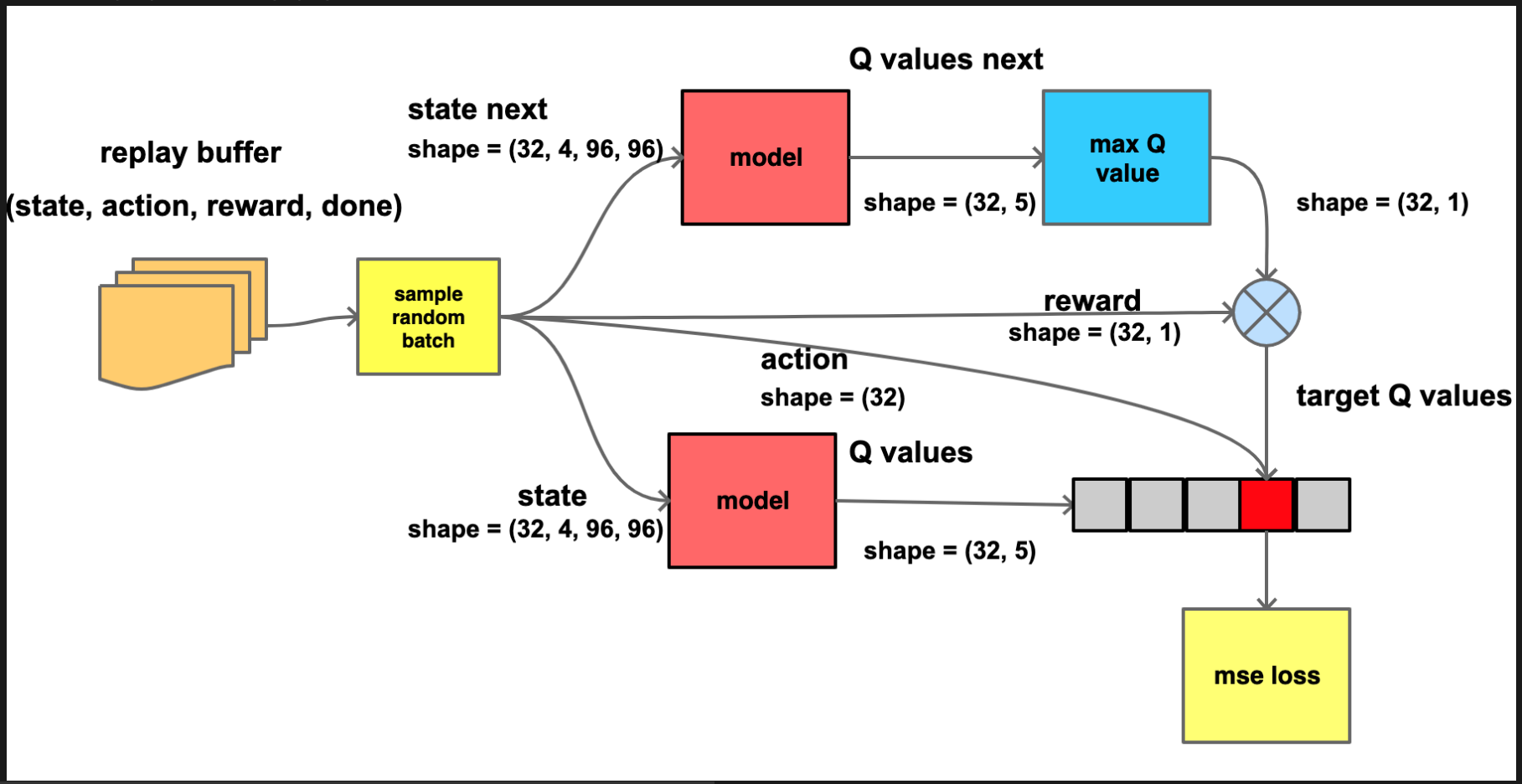
deep Q learning

- play games with any policy
- store to buffer : **state, action, reward, done**



deep Q learning

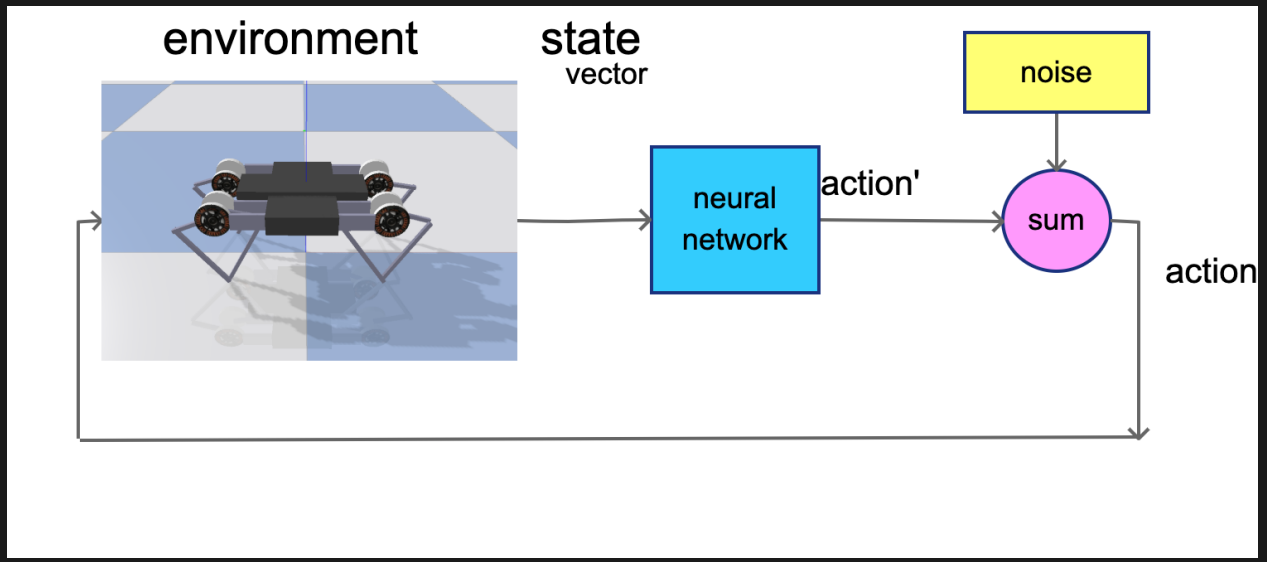
- sample random batch **state, state next, action, reward, done**
- obtain q-values and next q-values
- use q-learning to update q-values
- train model



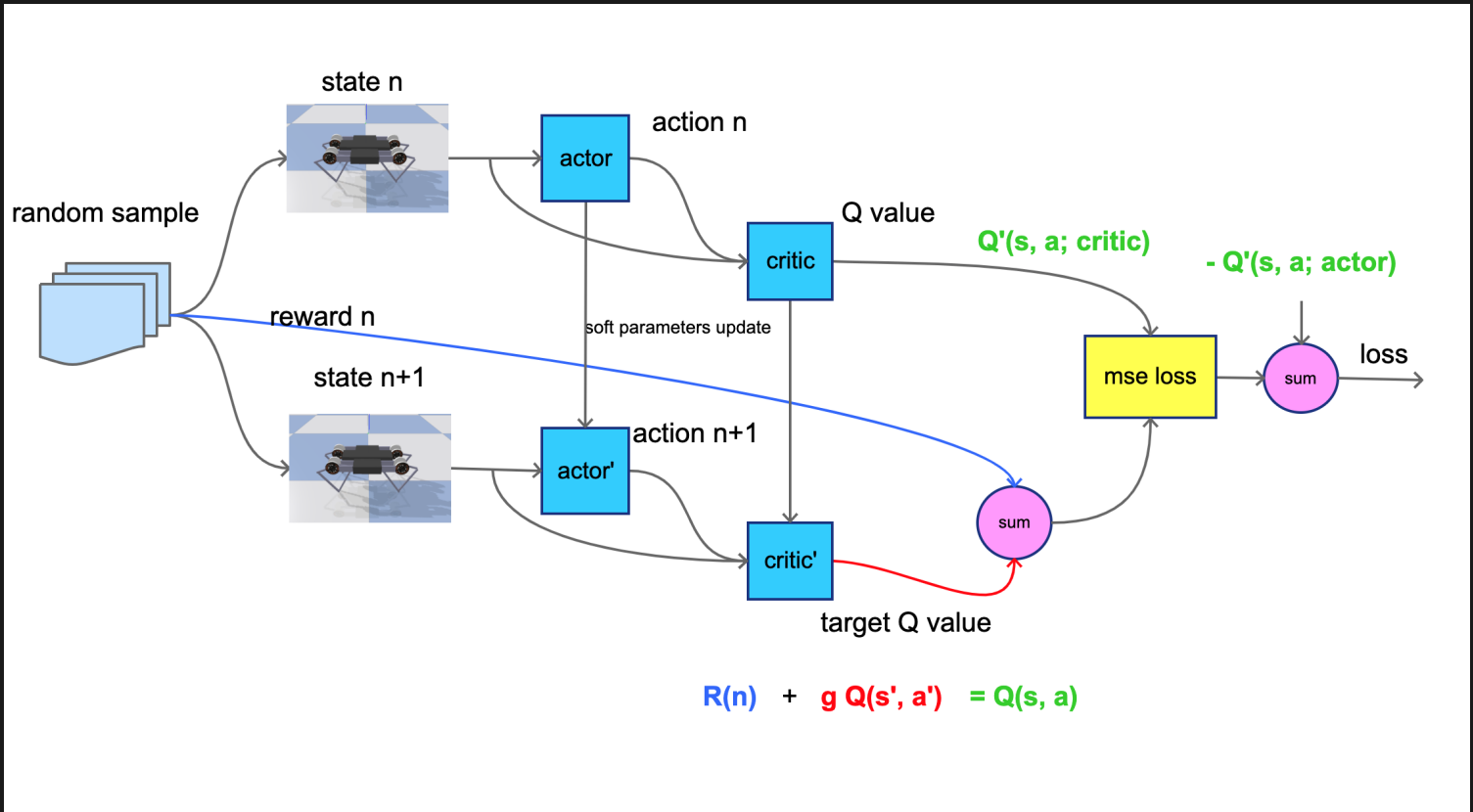
deep deterministic policy gradietnt

DDPG

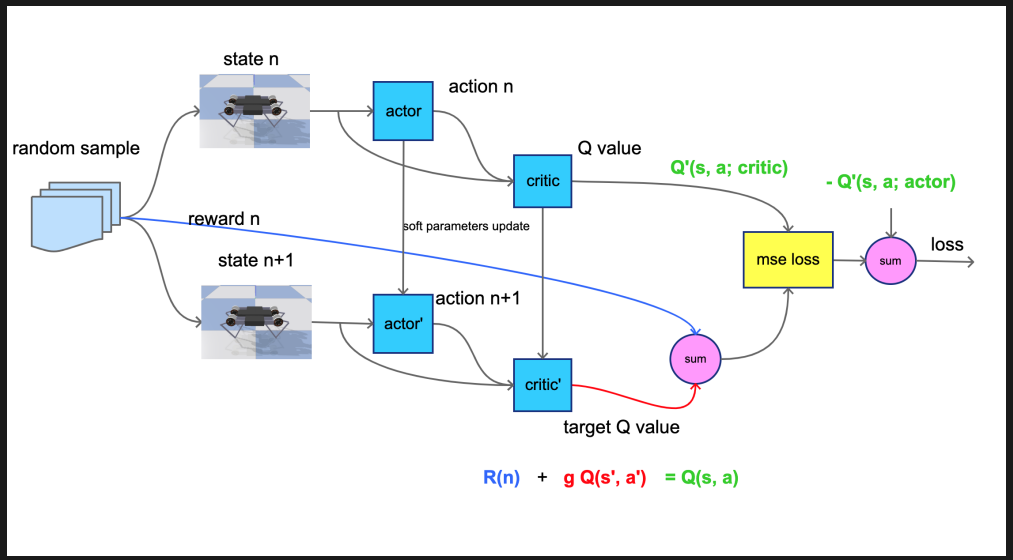
- continuous action space
- natural extension of DQN
- actor-critic structure



DDPG


state n
action n
actor
random sample
Q value
critic
Q'(s, a; critic)
- Q'(s, a; actor)
reward n
soft parameters update
mse loss
sum
loss
state n+1
action n+1
actor'
sum
critic'
target Q value
R(n) + g Q(s', a') = Q(s, a)

DDPG



$$\mathcal{L}(\theta) = \left(R + \gamma Q(s', A(s'; \phi^-); \theta^-) - Q(s, A(s; \phi); \theta)\right)^2$$
$$\mathcal{L}(\phi) = -Q(s, A(s; \phi); \theta)$$

where

- Q is critic network with parameters θ
- A is actor network with parameters ϕ

wise Wizard's DDPG spell chart

- **neurons count** on 1st layer = 10x state vector size
- **neurons count** on 2nd layer = 0.5x neurons on 1st layer
- **weight init** for hidden layers : use Xavier
- **weight init** actor output : use uniform $\langle -0.3, 0.3 \rangle$
- **weight init** critic output : use uniform $\langle -0.003, 0.003 \rangle$
- **gaussian noise** : linear decay variance, from 0.5 to 0.1, for 1M steps, or noisy layers
- use **soft** target network update, $\tau = 0.001$
- actor learning rate $\eta_a = 0.0001$
- critic learning rate $\eta_c = 0.0002$

wise Wizard's magic staff

- fully connected nets (robotic envs) **train on CPU** - AMD Ryzen
- convolutional nets (visual inputs envs) **train on GPU**
- use fast CPU - envs are slow
- 32GB of RAM is enough
- for small visual envs (Atari, DOOM, Nec) - GTX1060, GTX1080ti, RTX2080 ...

books to read

- Maxim Lapan, 2020, Deep Reinforcement Learning Hands-On second edition
- Maxim Lapan, 2018, Deep Reinforcement Learning Hands-On
- Praveen Palanisamy, 2018, Hands-On Intelligent Agents with OpenAI Gym
- Andrea Lonza, 2019, Reinforcement Learning Algorithms with Python
- Rajalingappaa Shanmugamani, 2019, Python Reinforcement Learning
- Micheal Lanham, 2019, Hands-On Deep Learning for Games

Q&A