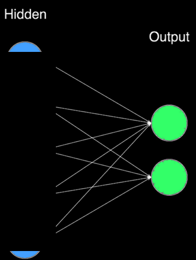
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

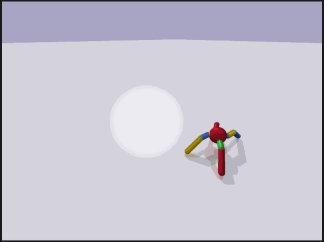
(The New Action Value = The Old Value) + The Learning Rate × (The New Information − the Old Information)

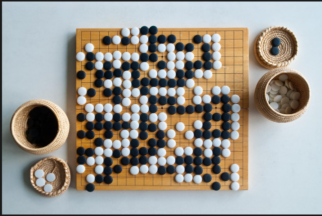
deep reinforcement learning 1
- deep Q networks
- deep deterministic policy gradient

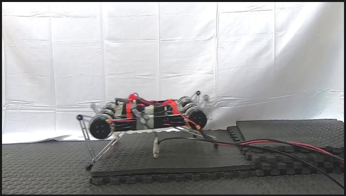
Michal CHOVANEC



reinforcement learning

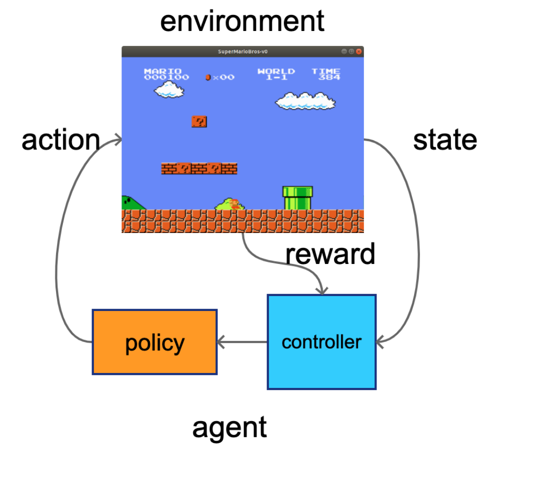




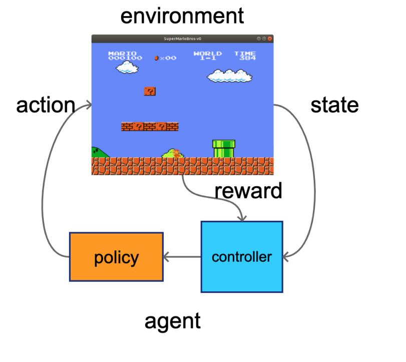


reinforcement learning

learning from punishments and rewards



reinforcement learning



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

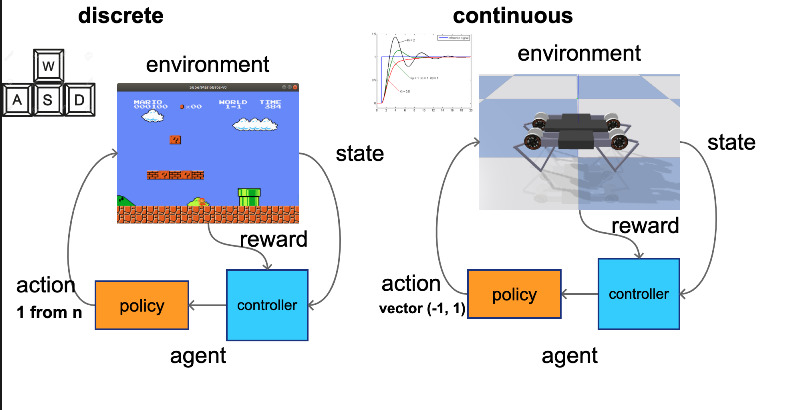
reinforcement learning - algorithms

- discrete actions space
  - Deep Q-network, DQN
  - Dueling DQN
  - Reinbow DQN
- continuous actions space
  - Actor Critic
  - Advantage Actor Critic
  - Proximal policy optimization
  - Soft Actor critic
  - Deep deterministc policy gradient
  - D4PG, SDDPG
- model based
  - curiosity
  - world models
  - imagination agents

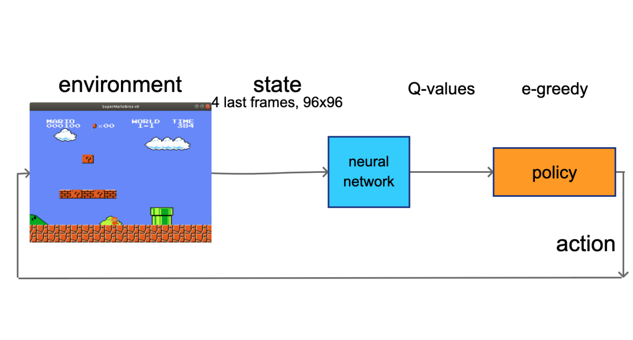
f.e. SDDPG - sampled DDPG, based on Wasserstein loss : Optimal transport, Cédric Villani, 600+ pages

action space

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

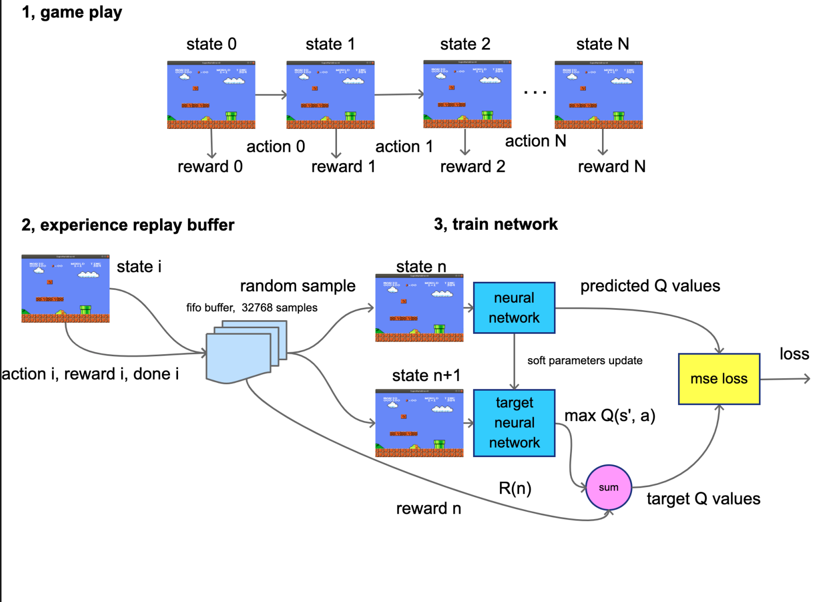


deep Q learning

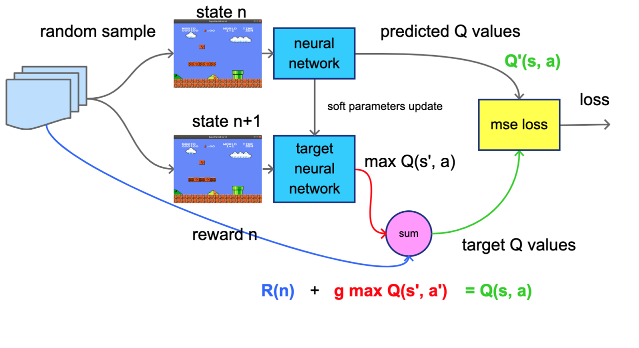


1. play games
2. store transitions into buffer
   - state, action, reward, done
3. learn from buffer

deep Q learning



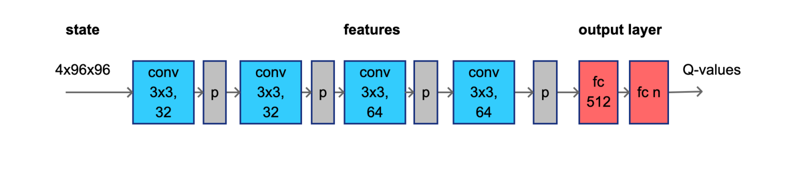
deep Q learning



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

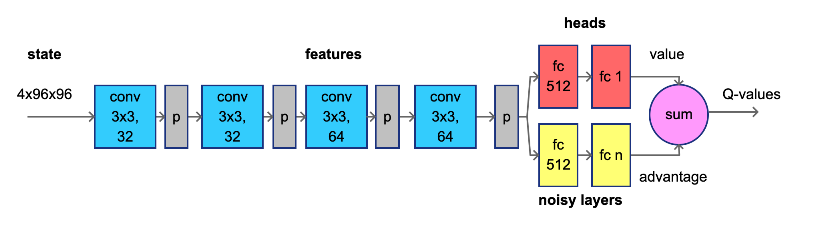
$$\mathcal{L}(\theta) = \left(R + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s,a;\theta) \right)^2$$

model architecture



- input 96x96 grayscale, 4 stacked frames
- 3x3 convs + pooling
- two fully connected layers
- small learning rate $\eta = 0.0001$, batch size = 32
- $\gamma = 0.99$
- exploration ϵ-greedy, 1M samples linear decay from 1 to 0.05
- total training 10M samples

dueling DQN, model architecture



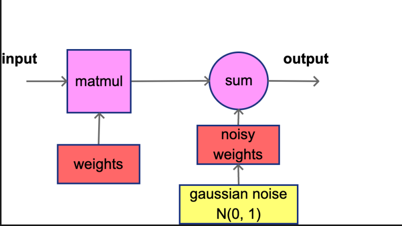
$$Q(s, a) = V(s) + A(s, a)$$

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} A(s, a')$$

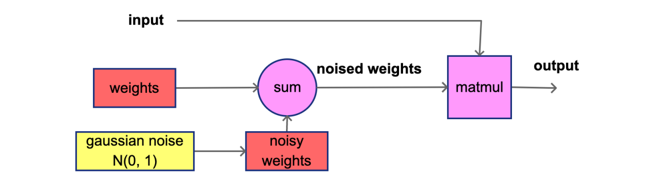
```
WRONG   : q = value + advantage - advantage.mean()
CORRECT : a = value + advantage - advantage.mean(dim=1, keepdim=True)
```

noisy layers for exploration

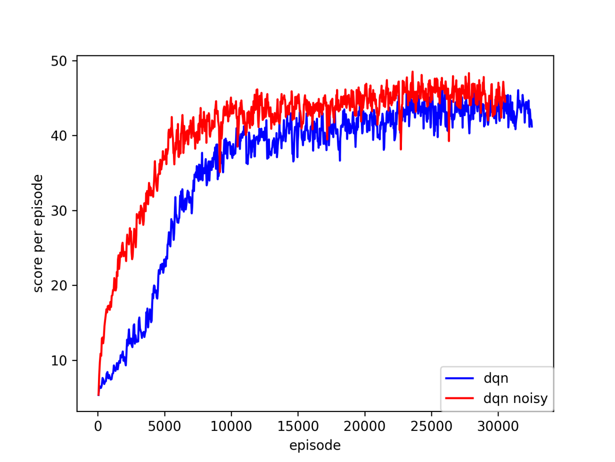
action space noise



parameter space noise



results on MsPacman



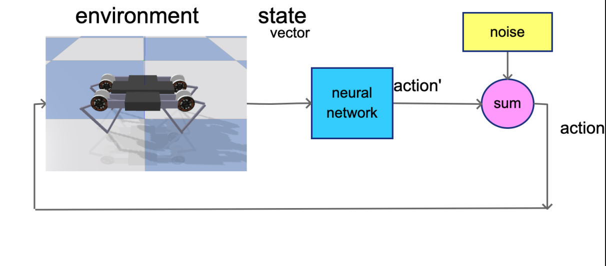
wise Wizard's DQN spell chart

- use **input** with nice size $k2^j$, 64, 96, 128 ...
- **conv layers** use 3x3 convs, poolings or strides > 1
- for **weight init** use Xavier
- **dont use** batchnorm (use SkipInit - not tested yet),
- **dont use** weight regularisation
- use **soft** target network update, $\tau = 0.001$
- slow learning rate $\eta = 0.0001$ works better
- for exploration use ϵ-greedy decay from 1.0 to 0.05 in 1M steps, or use noisy layers

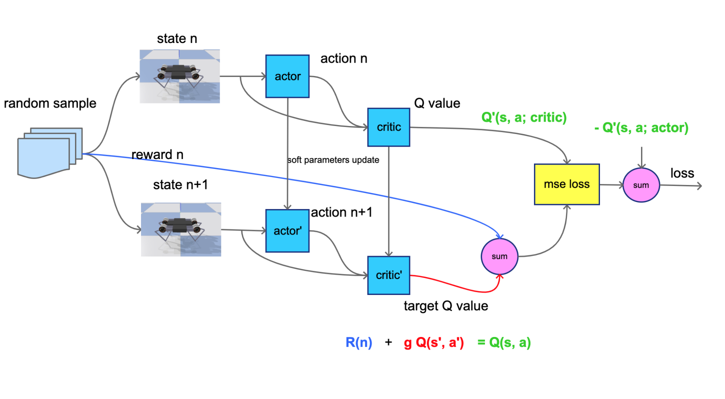
DDPG

deep deterministic policy gradietnt

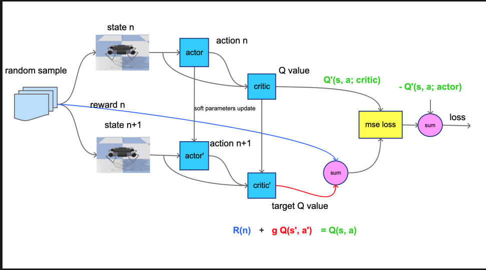
- continuous action space
- natural extension of DQN
- actor-critic structure



DDPG


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

DDPG

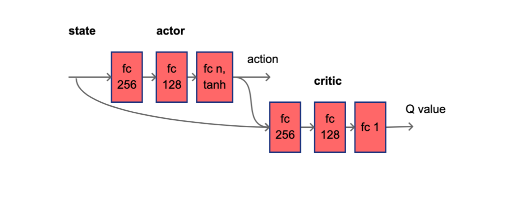


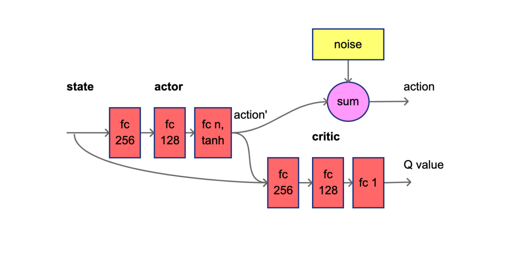
$$\mathcal{L}(\theta) = \left(R + \gamma Q(s', A(s'; \phi^{-}); \theta^{-}) - Q(s, A(s; \phi); \theta)\right)^2$$
$$\mathcal{L}(\phi) = -Q(s, A(s; \phi); \theta)$$

where

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

model architecture





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

michal.nand@gmail.com
https://github.com/michalnand/imagination_reinforcement_learning