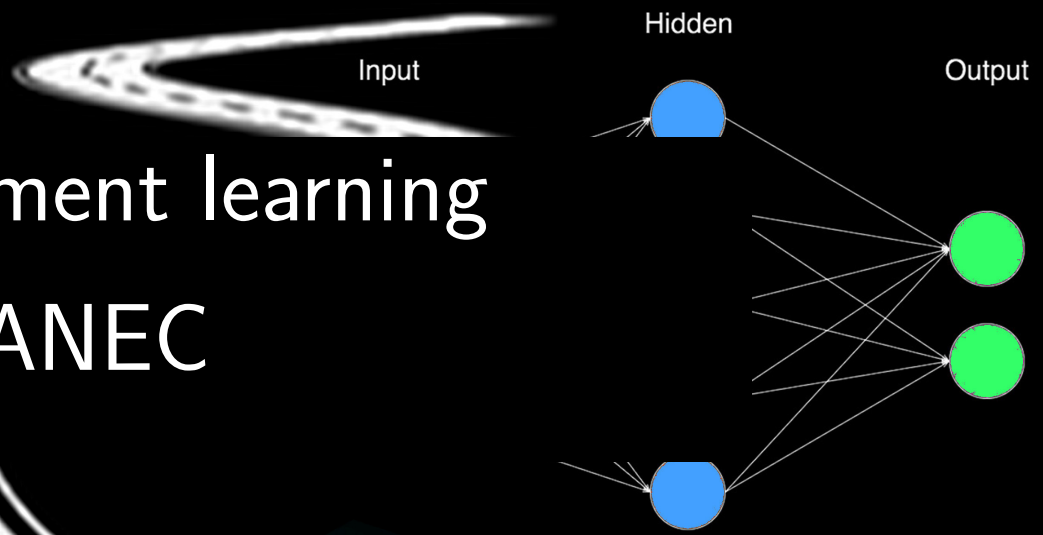
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

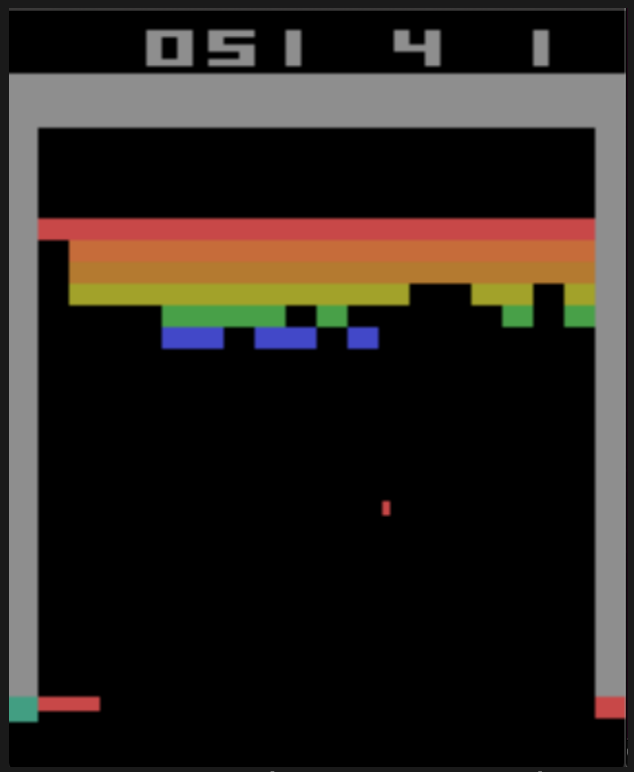
(The New Action Value = The Old Value) + The Learning Rate × (The New Information − the Old Information)

deep reinforcement learning - introduction

Michal CHOVANEC

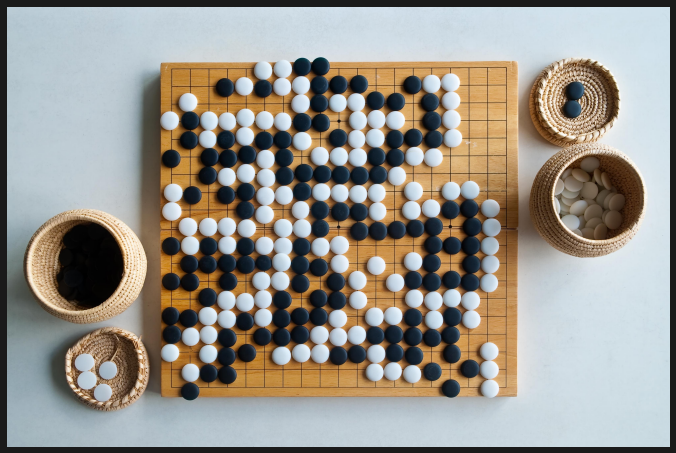


deep reinforcement learning - playing Atari



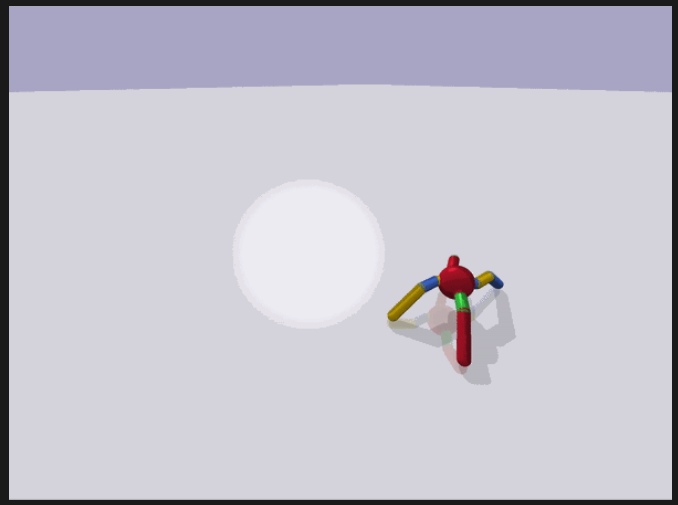
2013, Mnih (DeepMind) : Playing Atari with Deep Reinforcement Learning, `https://arxiv.org/pdf/1312.5602.pdf`

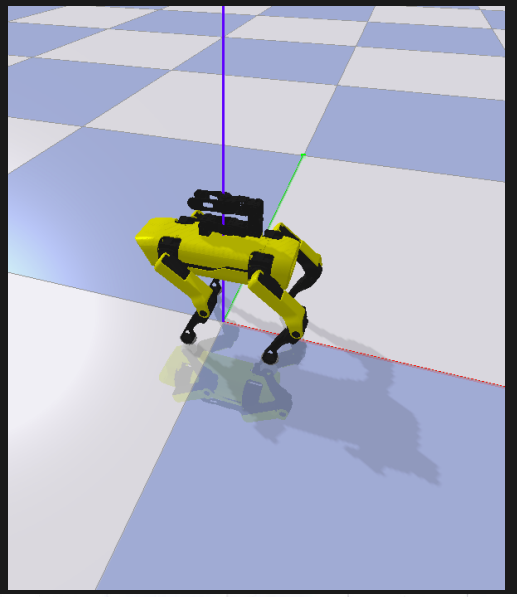
reinforcement learning - path to superhuman score

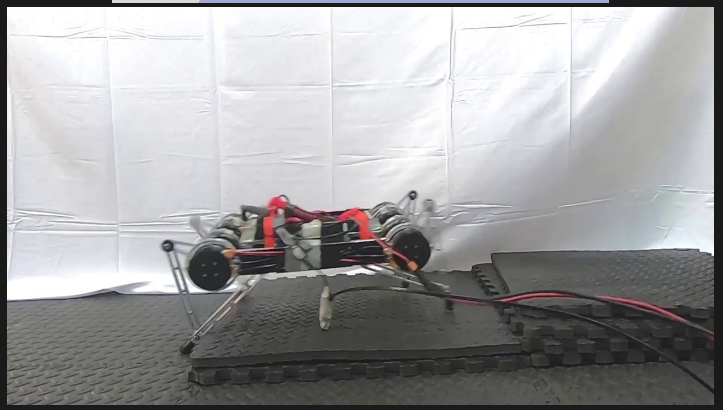


- 2016 AlphaGo, `https://www.nature.com/articles/nature16961`
- 2020 MuZero, `https://arxiv.org/pdf/1911.08265.pdf`
- AlphaZero, AlphaStar ...

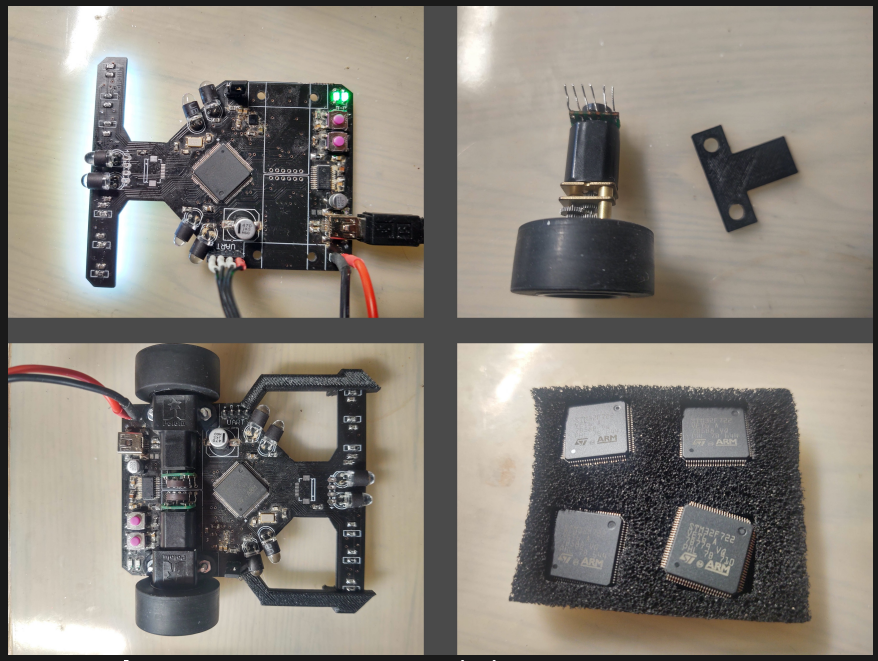
reinforcement learning - robotics

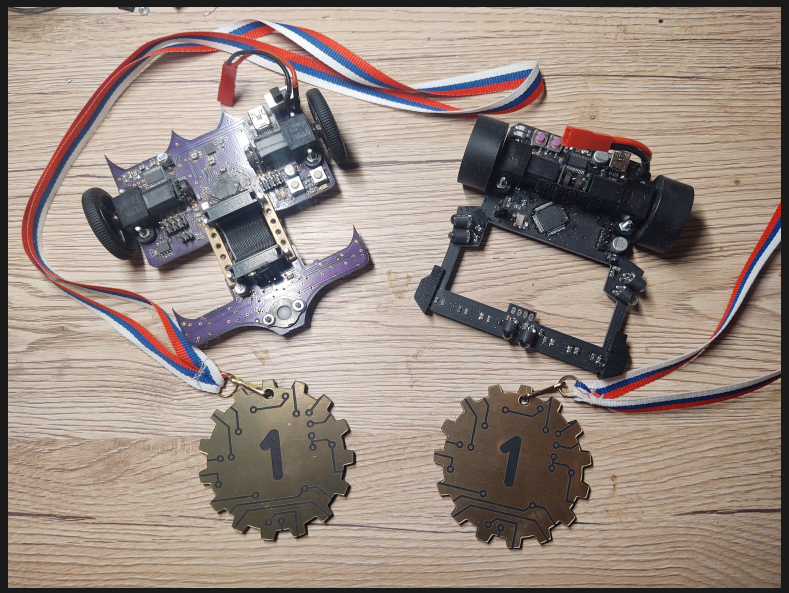






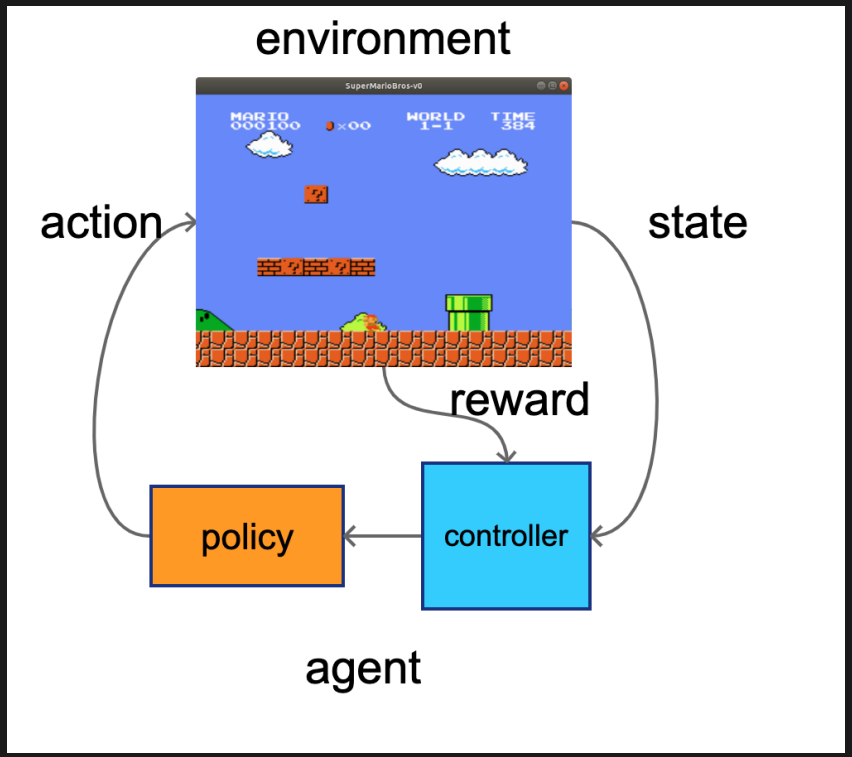
motoko uprising - line follower





video : https://www.youtube.com/watch?v=xUAJ1LA6Xwc

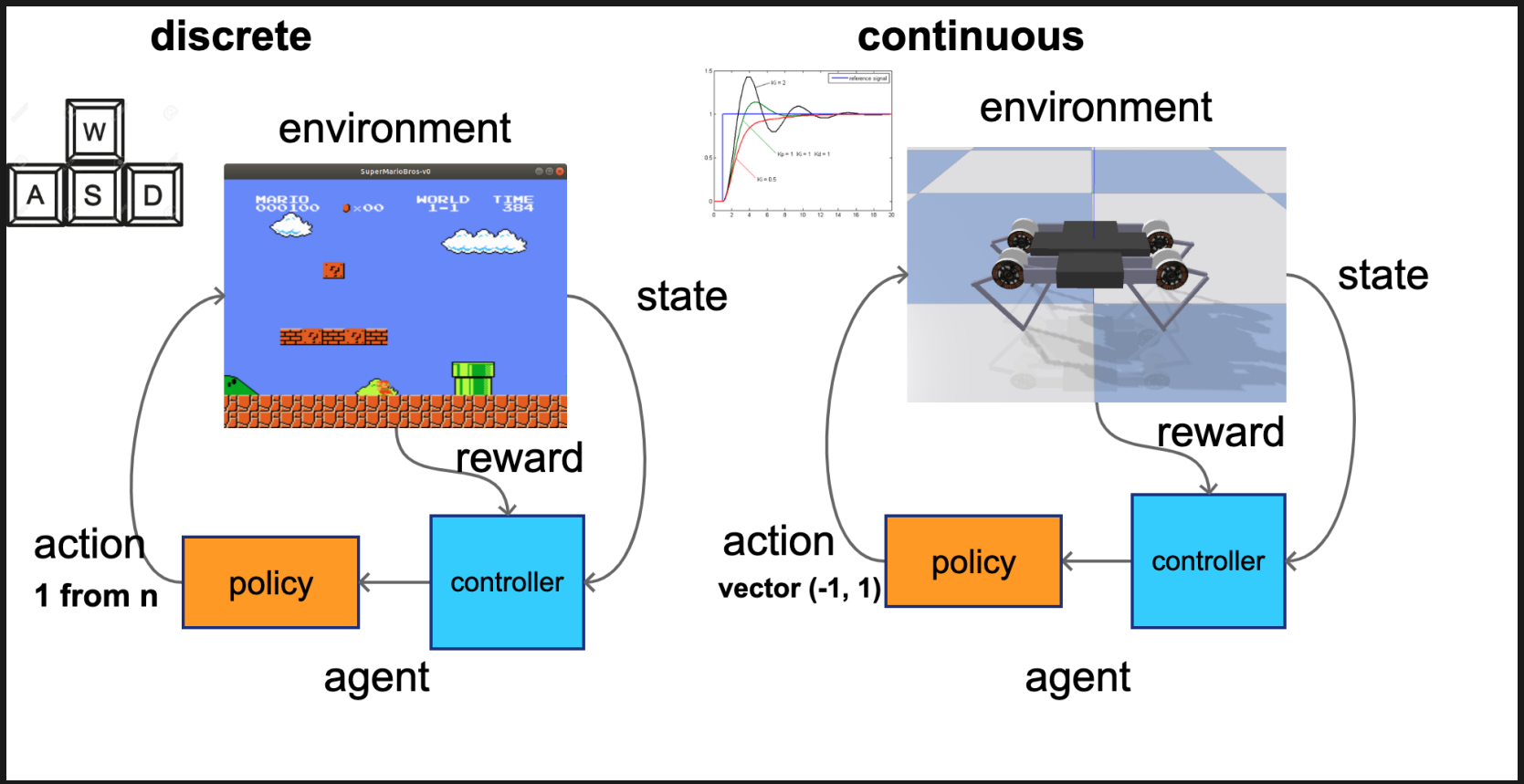
reinforcement learning



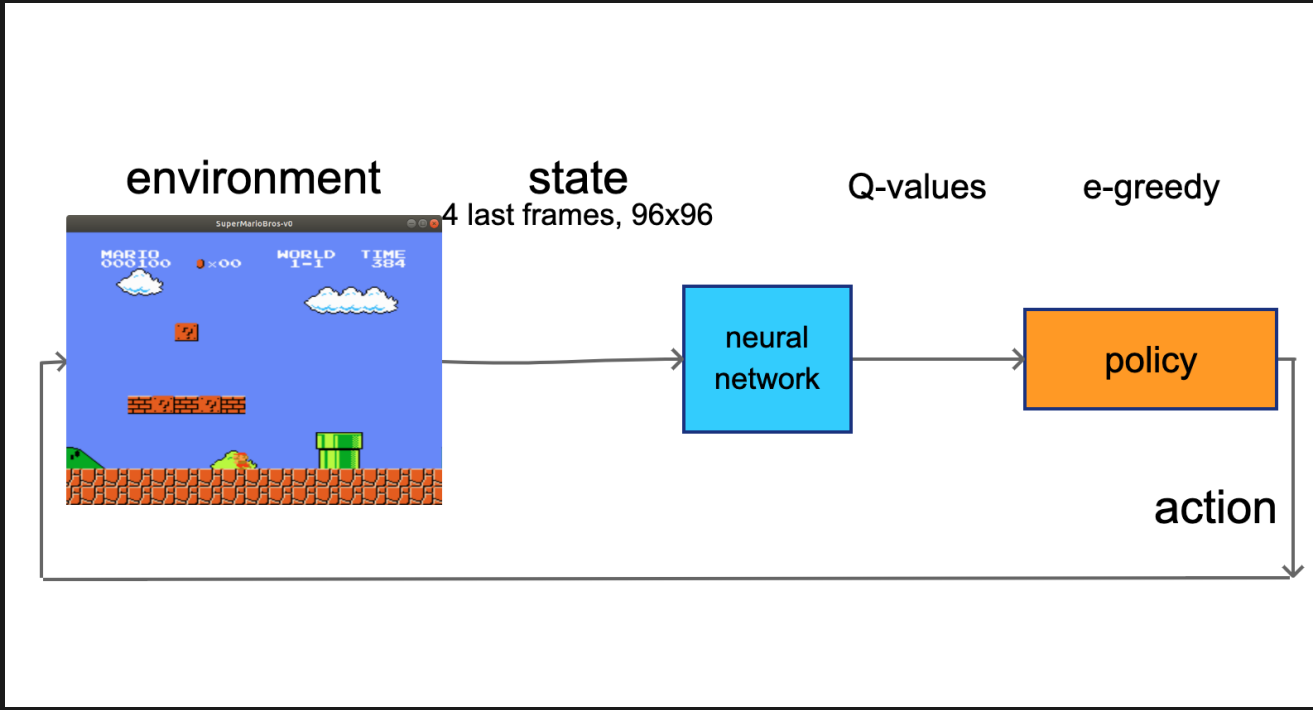
- obtain state
- select action
- exectute action
- learn from experiences

action space

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



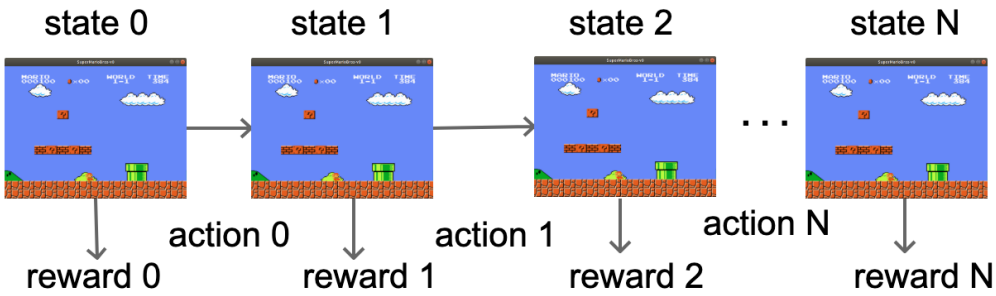
deep Q learning



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

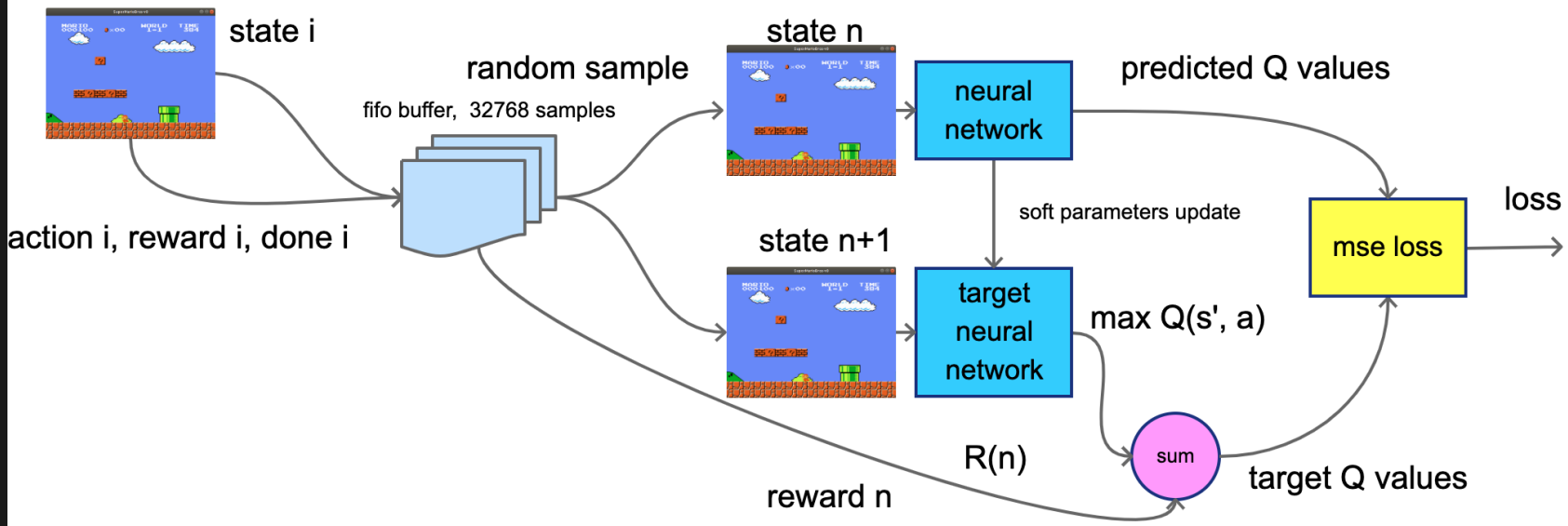
deep Q learning

1, game play

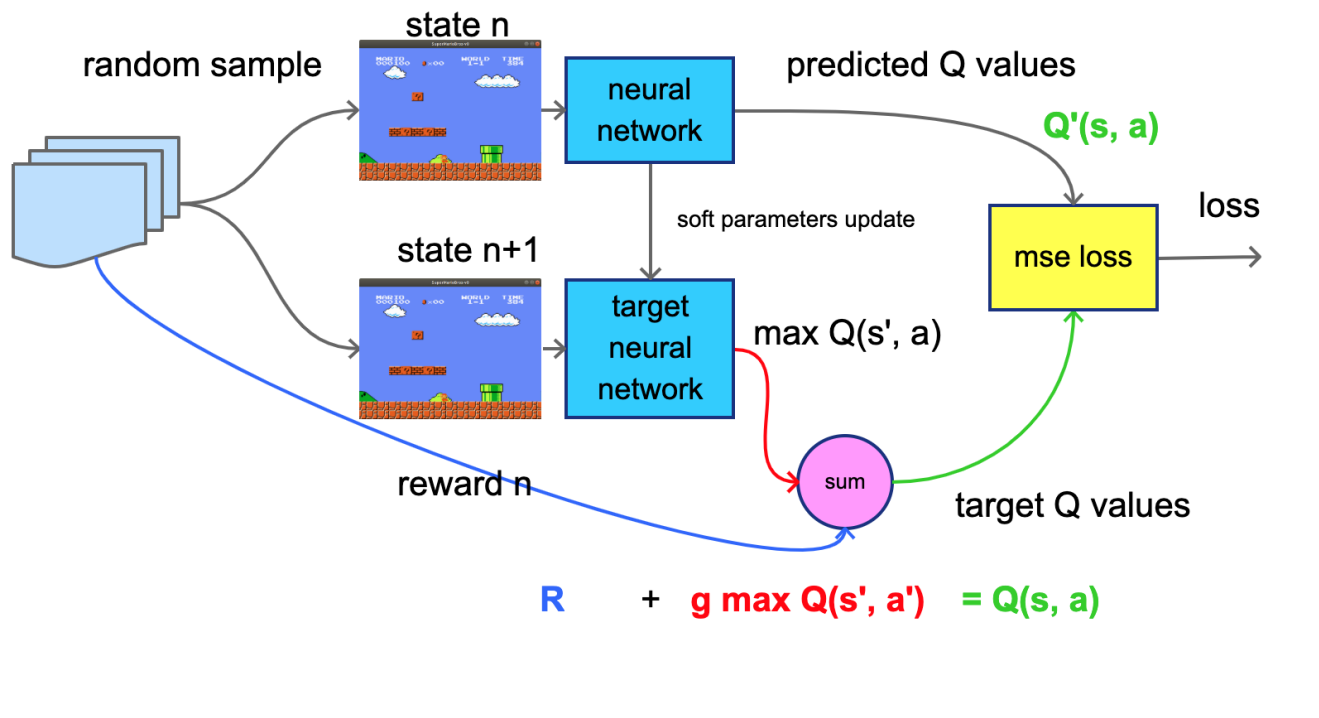


2, experience replay buffer

3, train network



deep Q learning

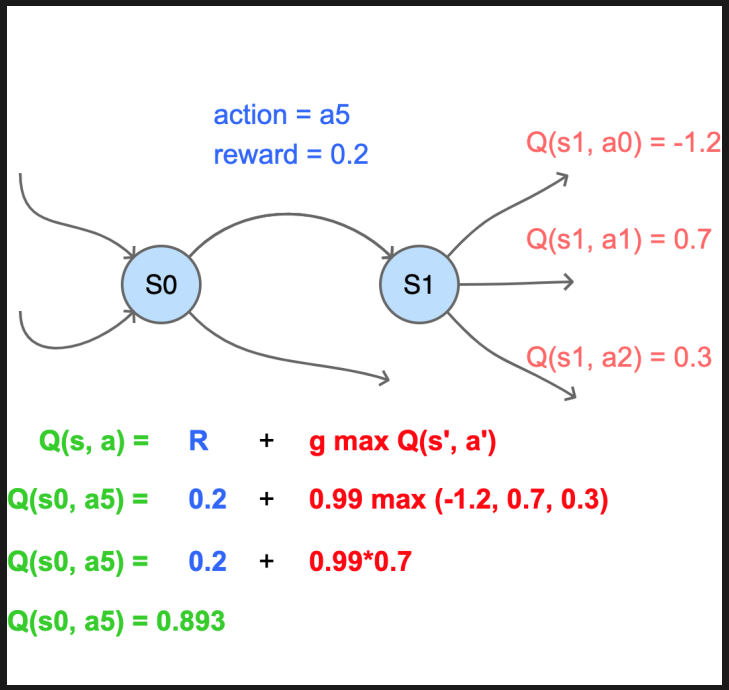


$$Q(s, a; \theta) = \underbrace{R}_{reward} + \gamma \max_{a'} Q(s', a'; \theta^-)$$

discounted future reward

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

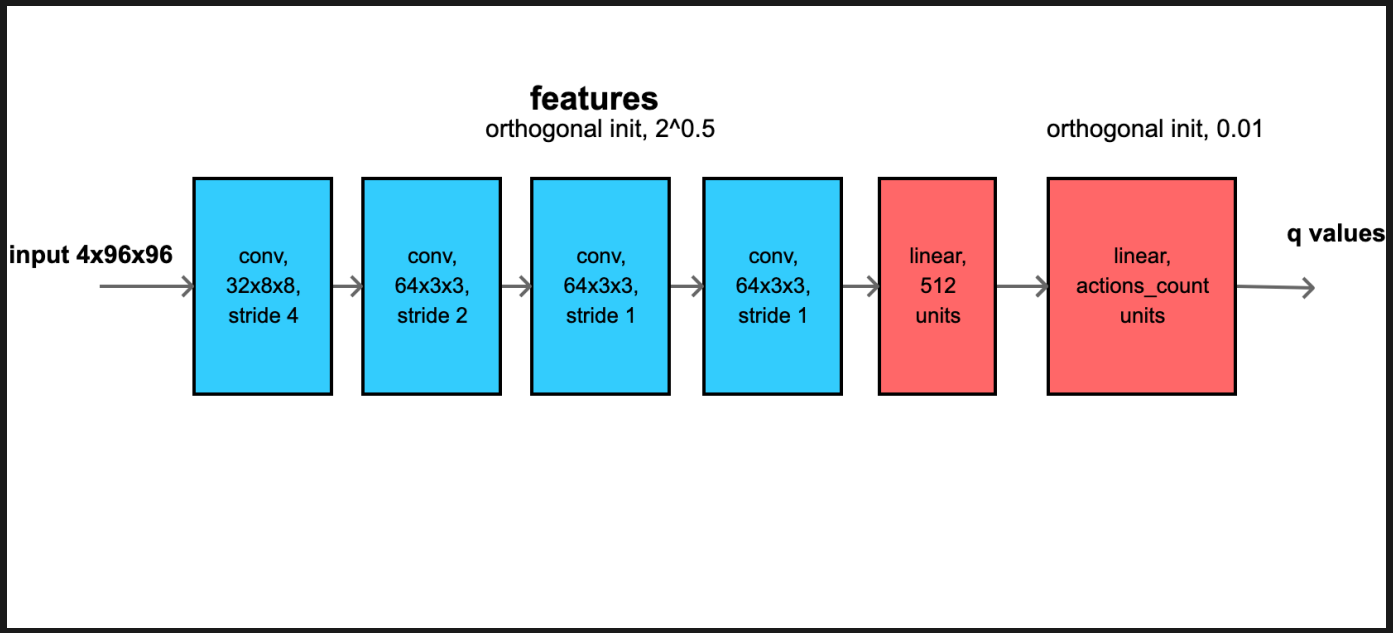
deep Q learning



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

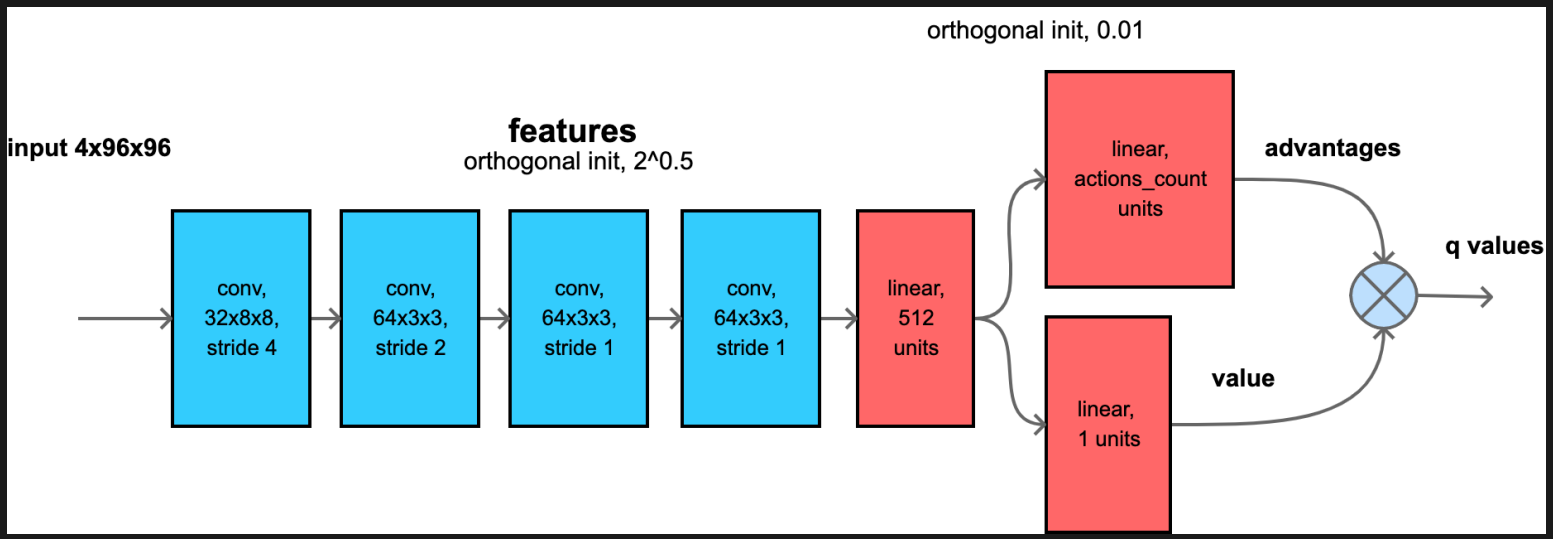
$$\mathcal{L}(\theta) = \left(R + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2$$

model architecture



- input 96x96 grayscale, 4 stacked frames
- 8x8 and 3x3 convs, with strides
- 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 8..16M samples

dueling DQN, model architecture



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

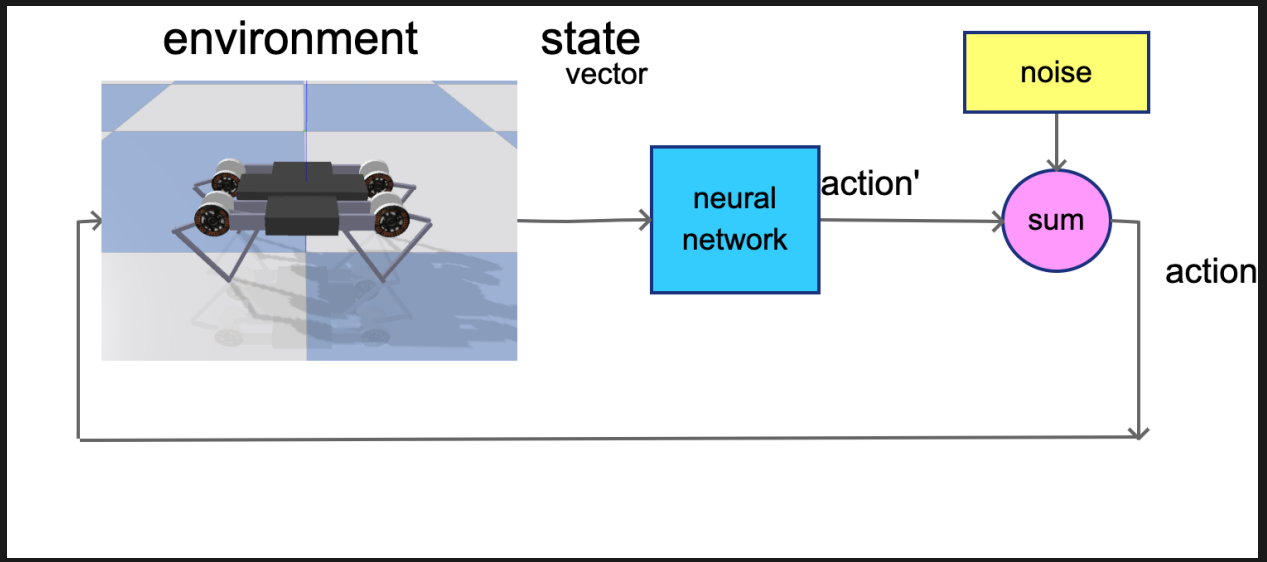
$$Q(s,a) = V(s) + A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} A(s,a')$$

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

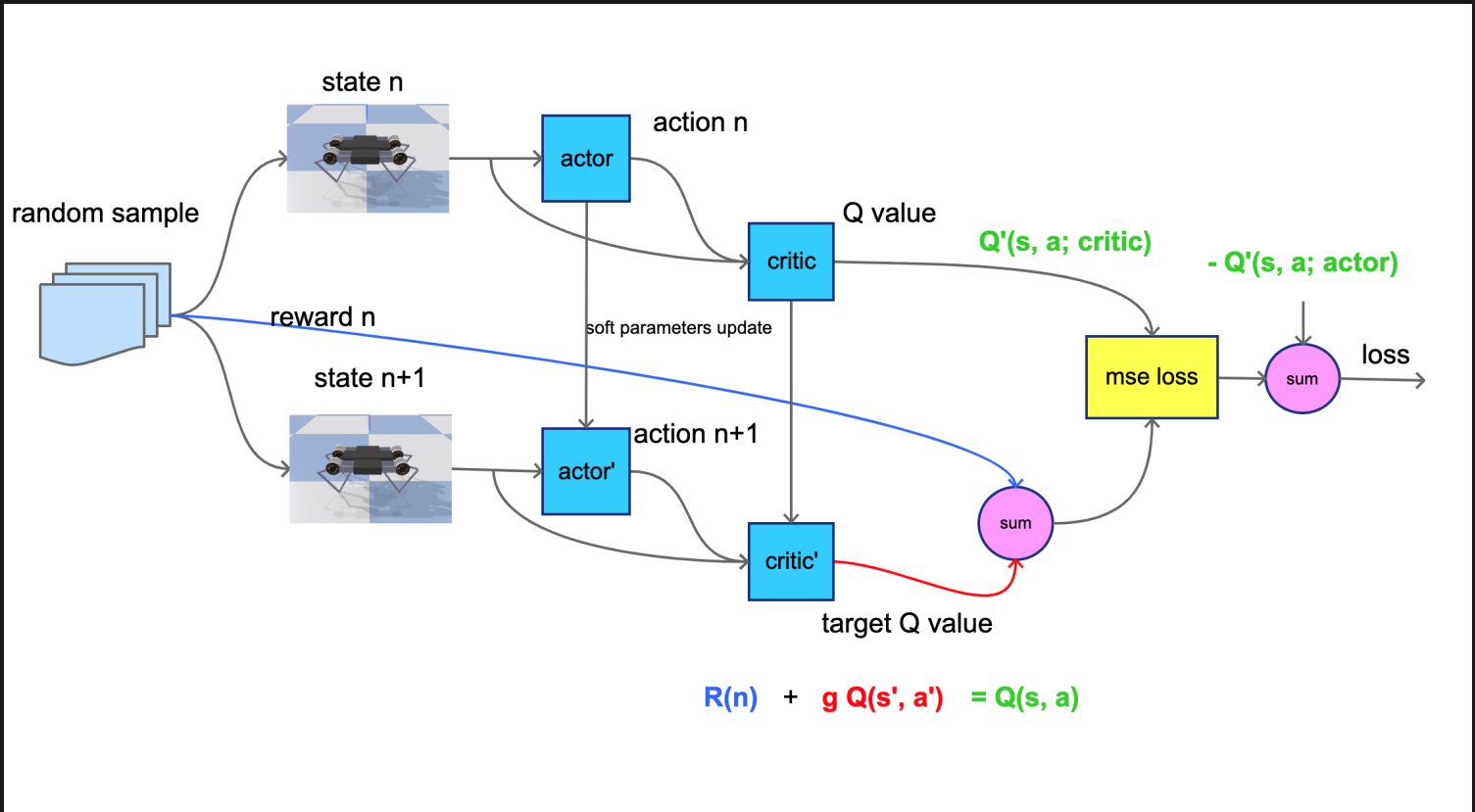
DDPG

deep deterministic policy gradietnt

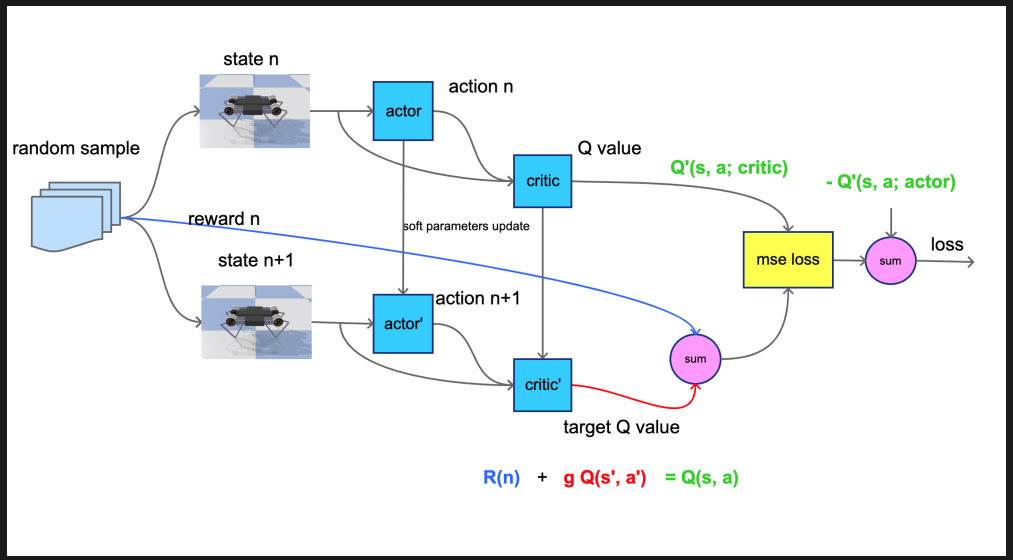
- continuous action space
- natural extension of DQN
- actor-critic structure



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

reinforcement learning - algorithms

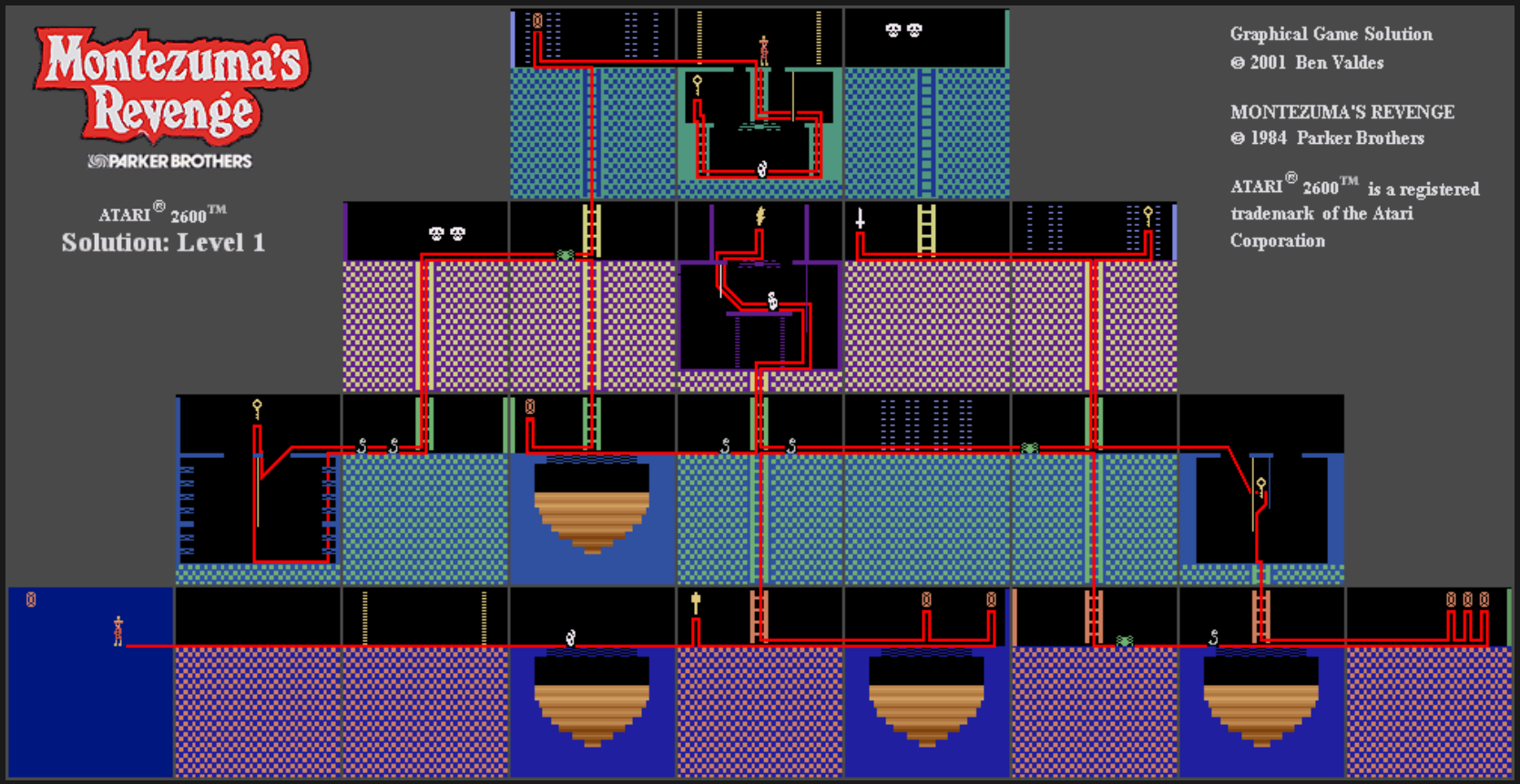
- discrete actions space
  - Deep Q-network, DQN
  - Dueling DQN
  - Reinbow DQN
- continuous actions space
  - Actor Critic (AC)
  - Advantage Actor Critic (A2C)
  - Proximal policy optimization (PPO)
  - Soft Actor critic (SAC)
  - Deep determinstc policy gradient (DDPG)
  - D4PG, SDDPG
- model based
  - curiosity
  - world models
  - imagination agents

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

unsolved problems

- RL sample efficiency, milions .. bilions frames
- sparse rewards
- detachment problem
- working model based agent

Montezuma's revenge



- extreme sparse rewards
- huge state space
- need to return hundrets of steps back
- still not solved - without game state save/load

state of the art score

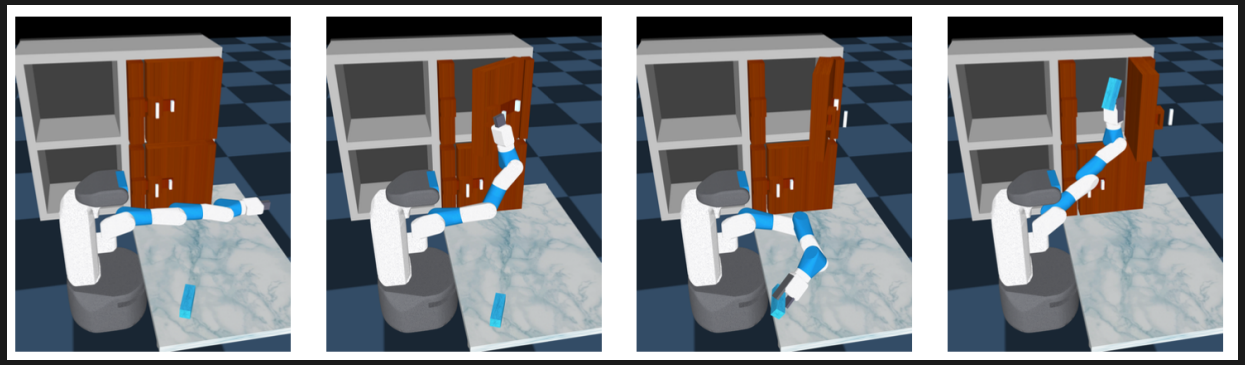
source : https://paperswithcode.com/sota/atari-games-on-atari-2600-montezumas-revenge

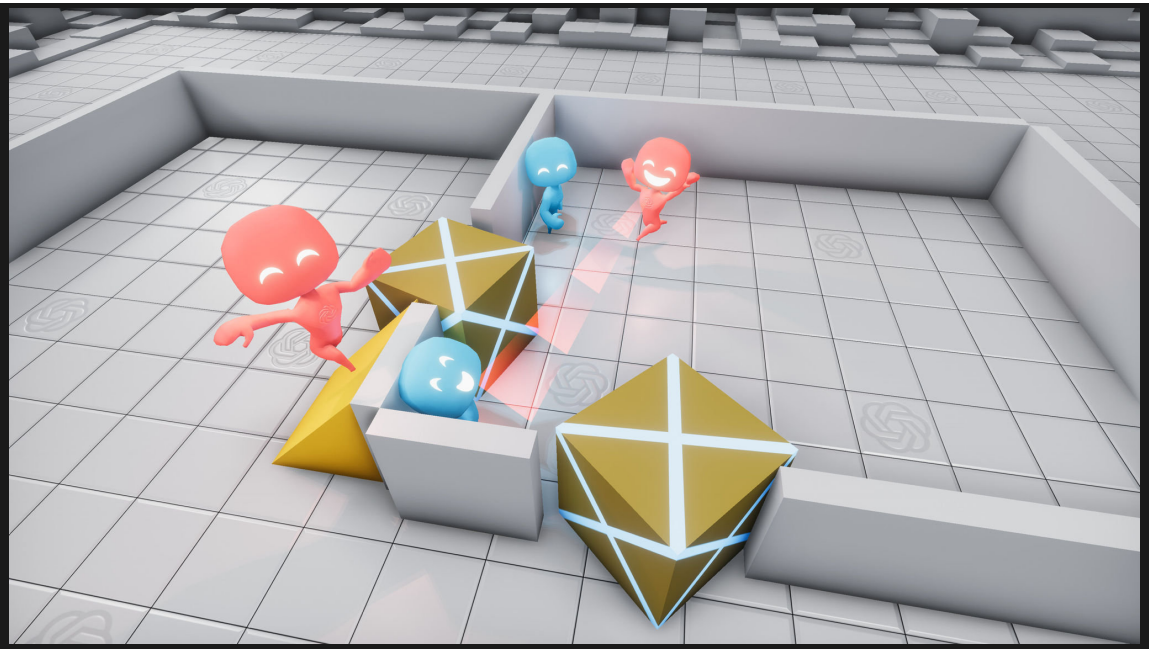
year	name	score
2015	Deep Reinforcement Learning with Double Q-learning	0
2017	Curiosity-driven Exploration by Self-supervised Prediction [a]	0
2021	MuZero	2500
2018	Count-Based Exploration with Neural Density Models [b]	3705
2019	**Exploration by Random Network Distillation** [c]	**8152**
2021	GoExplore* [d]	43 000

*** requires environment state saving/loading**

[a] https://arxiv.org/abs/1705.05363
[b] https://arxiv.org/abs/1703.01310
[c] https://arxiv.org/abs/1810.12894
[d] https://arxiv.org/abs/2004.12919

hard exploration robotics envs





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) - GTX1080ti, RTX3060, RTX3080ti ...

where to start

- simple discrete actions envs : LunarLander, Atari Pong
- algorithms : DQN or Dueling DQN
- continuous actions envs : LunarLander, Pybullet And + HalfCheetah
- algorithms : DDPG

books and blogs

- Maxim Lapan, 2020, Deep Reinforcement Learning Hands-On second edition
- Enes Bilgin, 2020, Mastering Reinforcement Learning with Python
- Andrej Karpathy, Pong from pixels `http://karpathy.github.io/2016/05/31/rl/`
- JordiTorresAI, Deep Q-Network, `https://towardsdatascience.com/deep-q-network-dqn-i-bce08bdf2af`

books to read

- Maxim Lapan, 2020, Deep Reinforcement Learning Hands-On second edition
- Maxim Lapan, 2018, Deep Reinforcement Learning Hands-On
- Enes Bilgin, 2020, Mastering Reinforcement Learning with Python
- 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