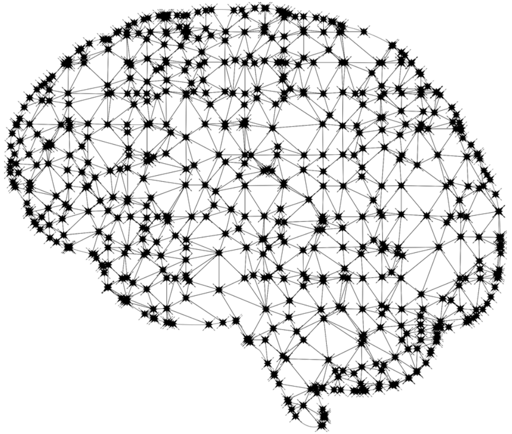
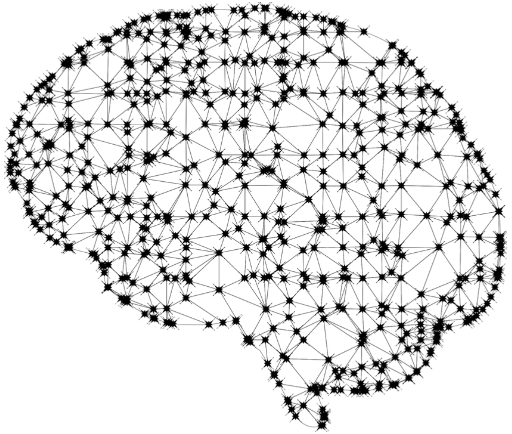
Reinforcement Learning with Pytorch



Introduction



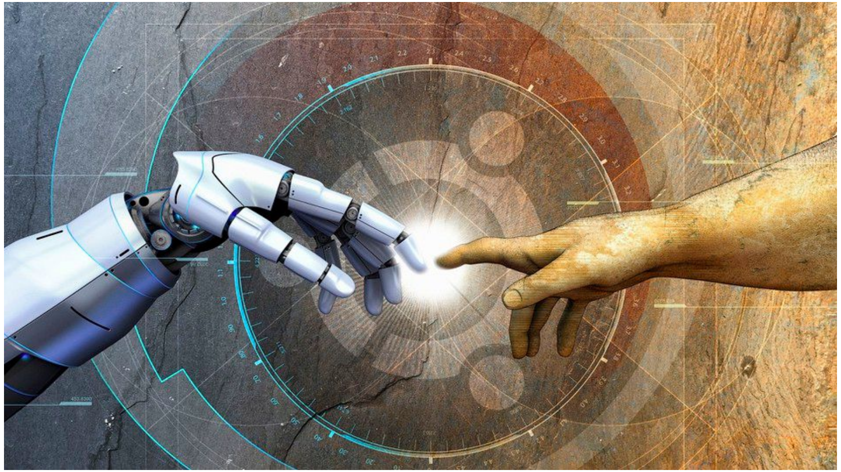
Before we start

What is this course about ?

Is this course for me ?

What are requirements?

Contact : marcin@atamai.biz

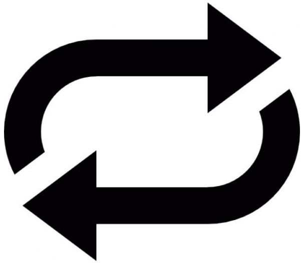


Source: freedomandsafety.com

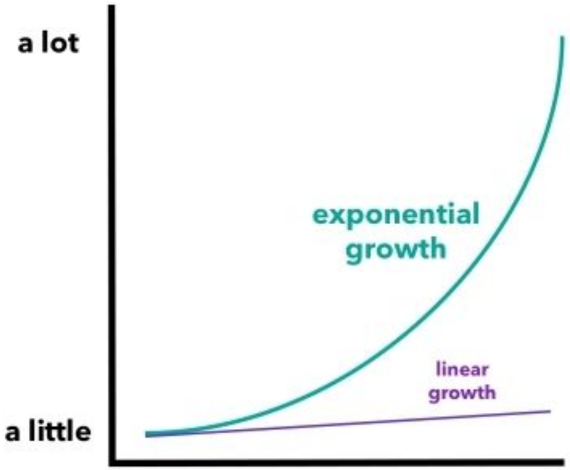
Introduction

Artificial Intelligence. Why getting so popular ?

- Moore law
- Reinforcement learning + Deep Learning
- Brain power
- More learning resources
- More people involved
- More projects
- Better results



Source: freepik.com



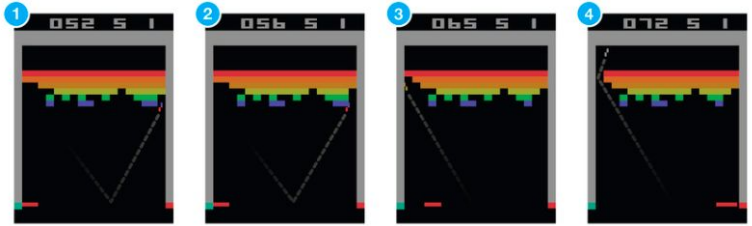
Source: charlesngo.com

Introduction

Example of Deepmind

2013 paper

https://arxiv.org/pdf/1312.5602.pdf



Source: nature.com

Google + Deepmind

https://techcrunch.com/2014/01/26/google-deepmind/

2015 paper

https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

AlphaGo, AlphaGo Zero

https://deepmind.com/blog/alphago-zero-learning-scratch/

Introduction

What will we do ...

Source: python.org

Source: pytorch.org

Source: openai.com

Introduction

Recommended resources:

http://www0.cs.ucl ac.uk/staff/d.silver/web/Teaching.html

http://incompleteideas.net/book/the-book-2nd.html

Introduction

What is Reinforcement learning?

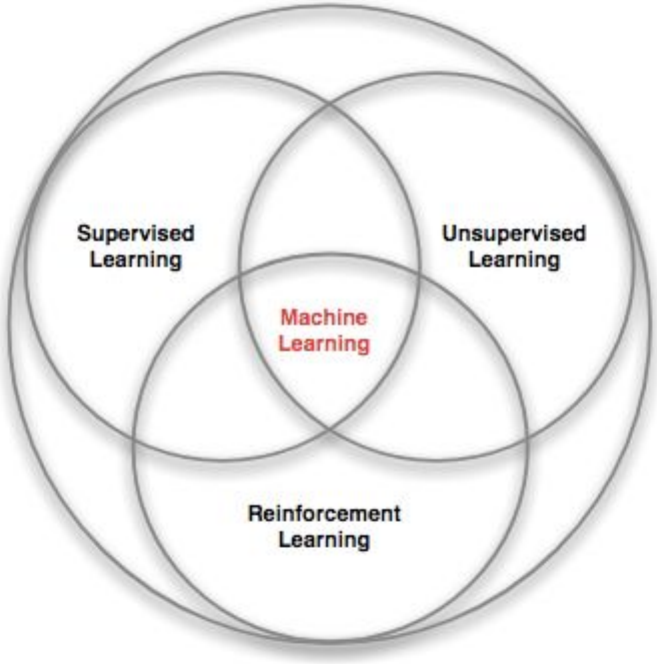
Supervised vs unsupervised learning

No supervisor

Rewards

Feedback

Time

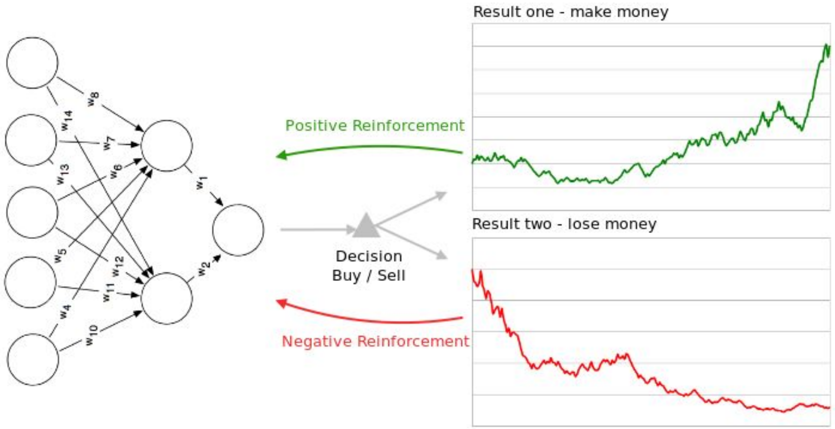


Source: David Silver

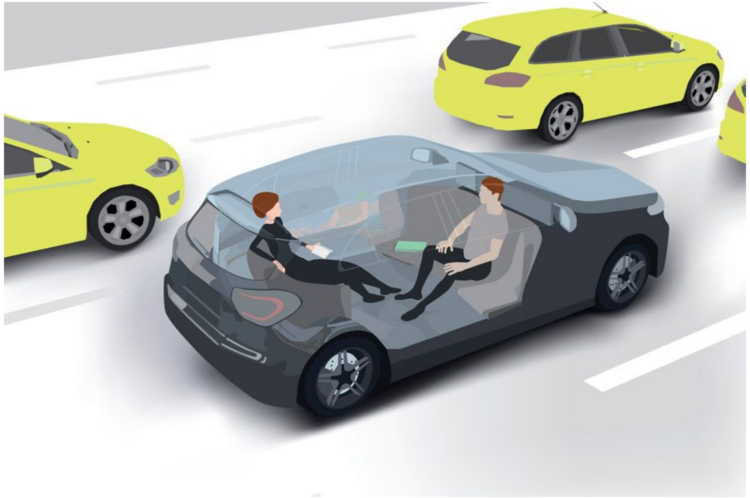
Introduction

What is Reinforcement learning?

Examples



Source: turingfinance.com



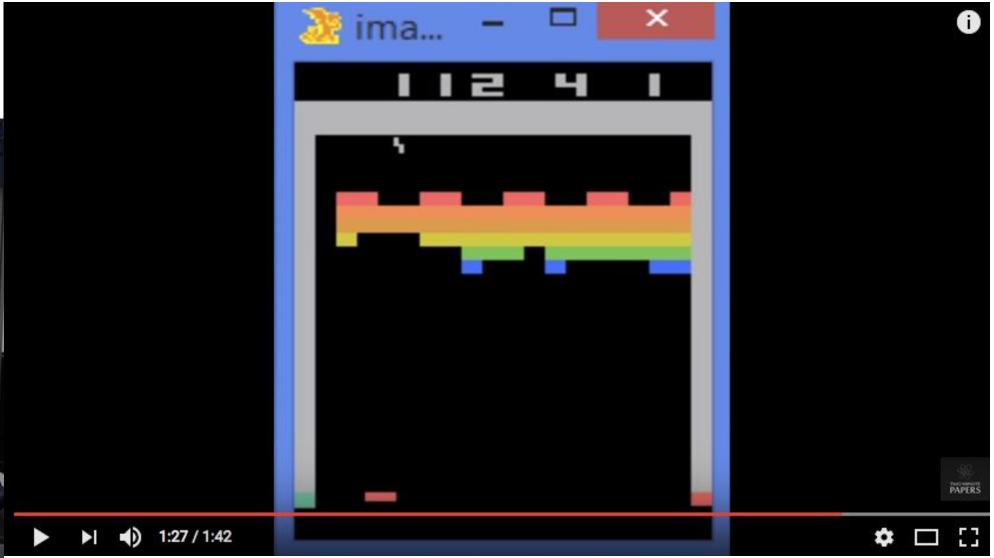
Source: stanford.edu

Introduction

What is Reinforcement learning?

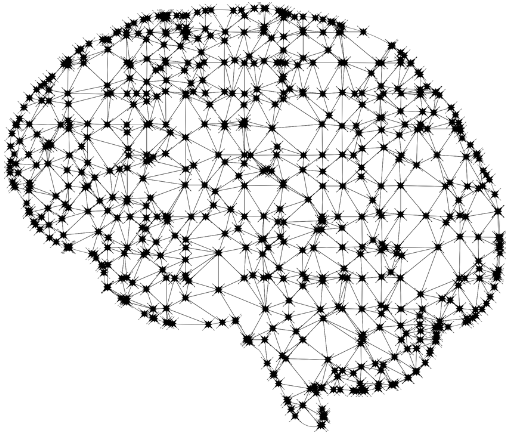
Examples

Source: google

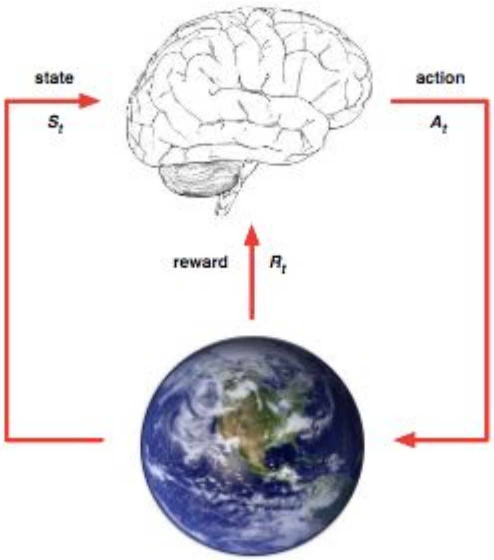


[Deepmind Atari]

Tabular methods



Some theory



Source: David Silver

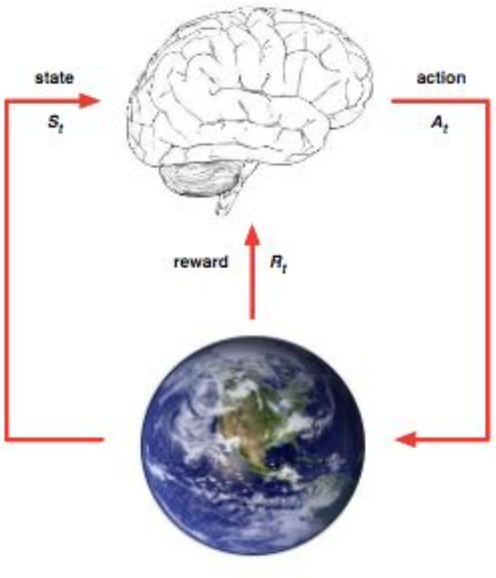
History:

$$H = O1, A1, R1, O2, A2, R2, ..., On, An, Rn$$

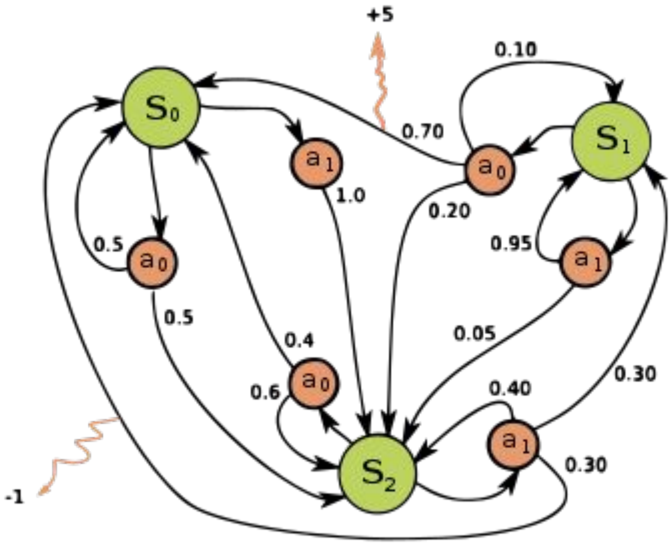
State:

$$state = f(H)$$

Markov Decision Process



Source: David Silver



Source: wikipedia

Openai gym

https://gym.openai.com/docs/

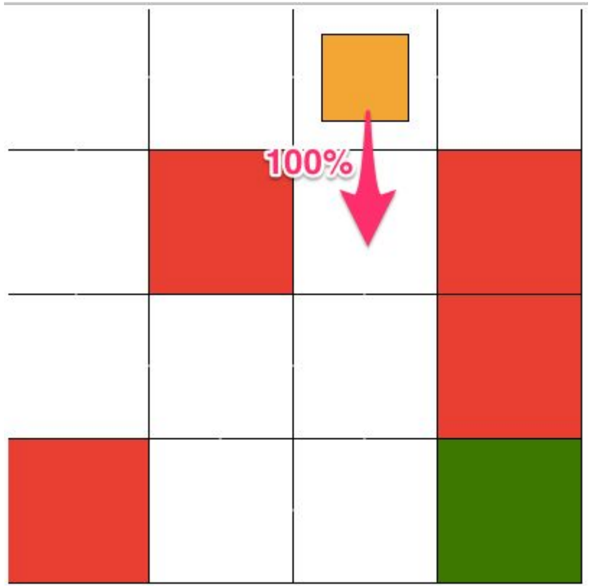
https://gym.openai.com/envs/

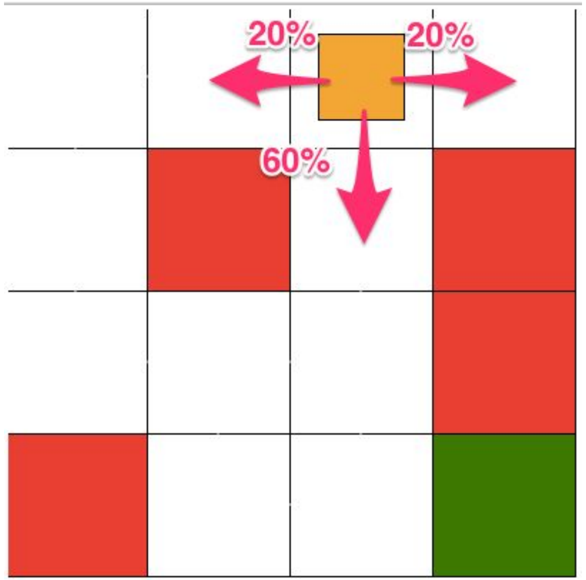
https://github.com/openai/gym

https://github.com/openai/gym/wiki

Source: openai.com

Deterministic vs Stochastic





Rewards

Rewards

$$R_t = r_t + r_{t+1} + r_{t+2} + ... + r_n$$

Discount:

$$\gamma \in \{0..1\}$$

Source: pixabay.com

$$R_t = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + ... + \gamma^{n-t} * r_n$$

Solution

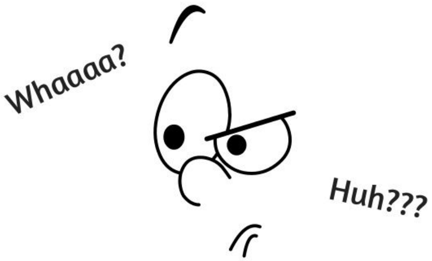
https://en.wikipedia.org/wiki/Markov_decision_process

$$V(s) := \sum_{s'} P_{\pi(s)}(s, s') \left(R_{\pi(s)}(s, s') + \gamma V(s')\right)$$

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P_a(s, s') \left(R_a(s, s') + \gamma V(s')\right) \right\}$$

$$V_{i+1}(s) := \max_a \left\{ \sum_{s'} P_a(s, s') \left(R_a(s, s') + \gamma V_i(s')\right) \right\},$$

$$Q(s, a) = \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V(s')).$$



Source: kmazing.net

Solution

Source for equations: http://cs231n.stanford.edu

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Q-value function at state s - taking action a
How good is a state-action pair

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Optimal Q value

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s, a\right]$$

Solution for deterministic environment

Bellman equation:

$$Q(s,a) = r + \gamma * max_{a'} Q(s',a')$$

Source: wikipedia

Algorithm for deterministic environment

initialize Q[num_states, num_actions]

observe initial state s

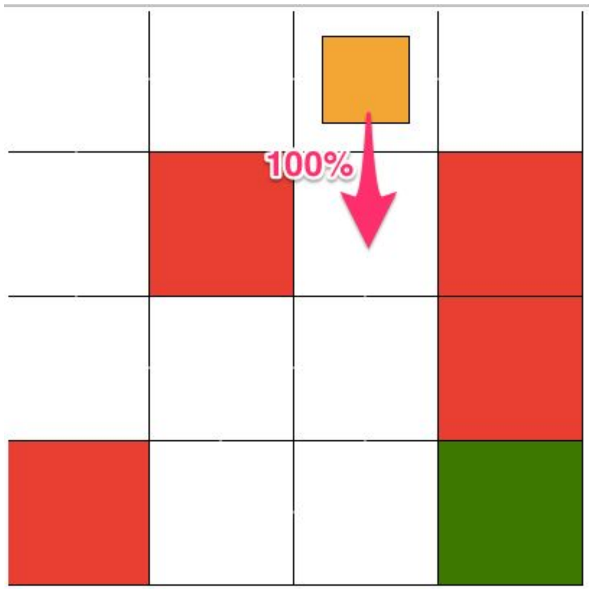
<u>repeat until terminated</u>:

    select and perform action a

    observe reward r and new state s'

    $Q(s,a) = r + \gamma * \max Q(s',a')$

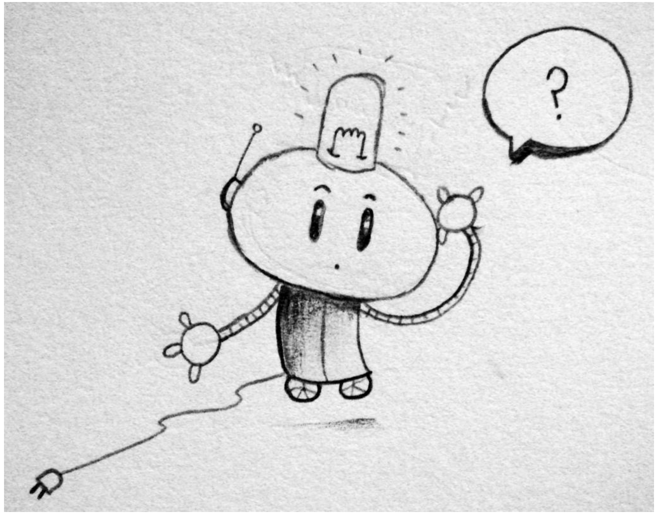
    s = s'



Stochastic environment

What if we don’t know everything about environment?

What if we only know set of states and actions?

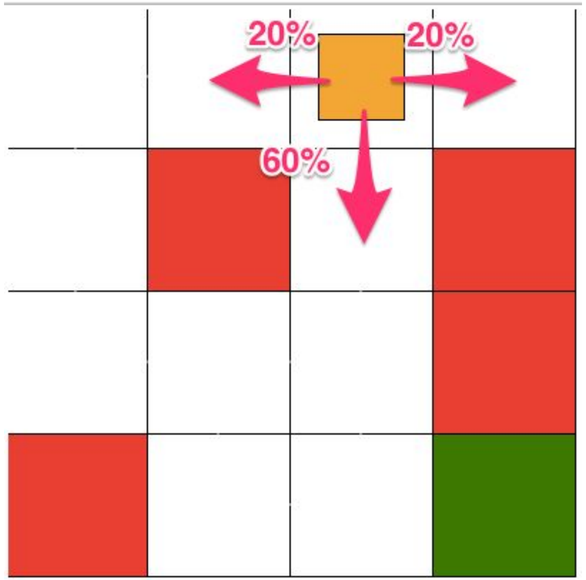


Source: flickr.com

It’s about learning!

Q learning as an answer

Temporal difference



observations before

vs

observation now

$$[r + \gamma * max_{a'} Q(s', a')] - [Q(s, a)]$$

Q learning

Source: Max Pixel

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha * TD$$

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma * max_{a'} Q(s',a') - Q(s,a)]$$

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma * max_{a'} Q(s',a')]$$

Algorithm for stochastic environment

initialize Q[num_states, num_actions]

observe initial state s

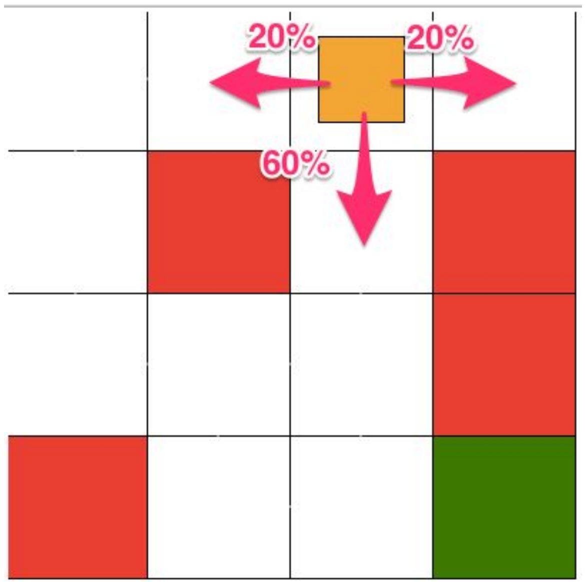
Repeat until terminated:

select and perform action a

observe reward r and new state s’

$Q(s,a) = (1-\alpha)\, Q(s,a) + \alpha\, [\, r + \gamma * \max Q(s',a')\,]$

$s = s'$

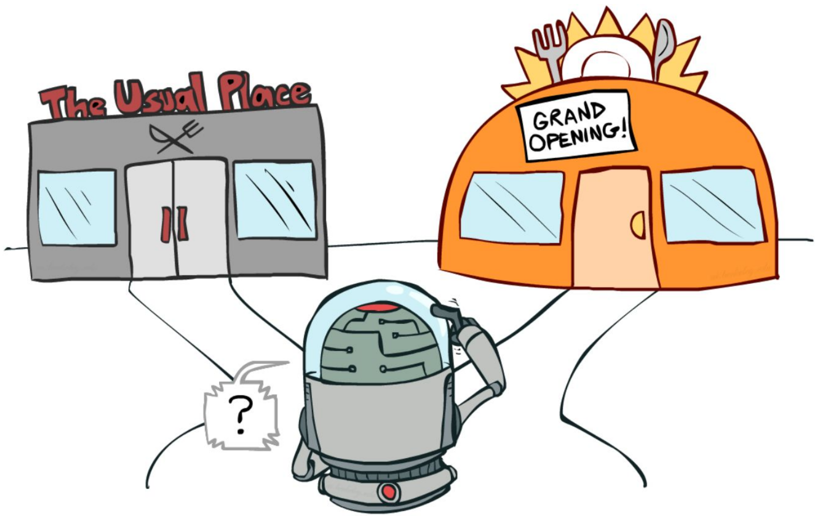


Exploitation vs Exploration

best decision vs more information

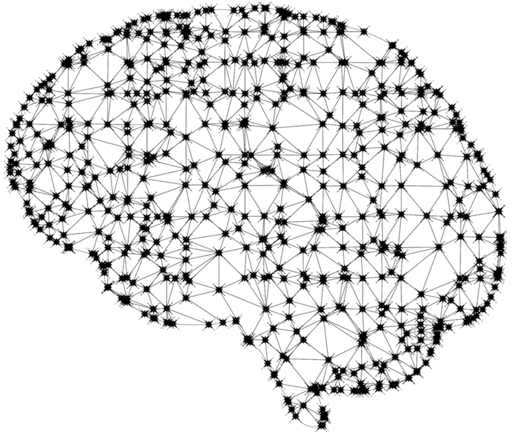
e - greedy:

$$a = \begin{cases} optimal\ a^* & 1 - \varepsilon \\ random & \varepsilon \end{cases}$$



Source: berkeley.edu

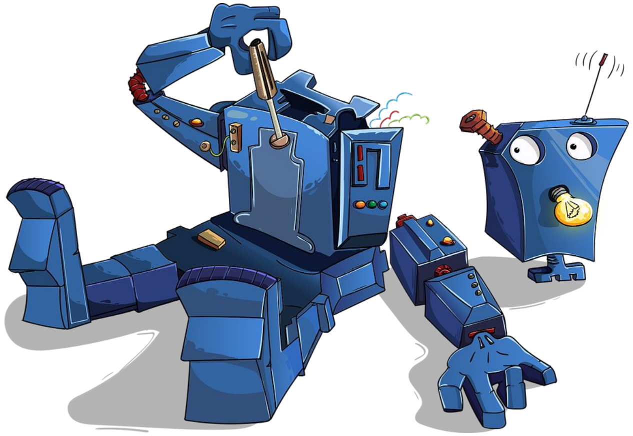
Scaling up



Scaling up

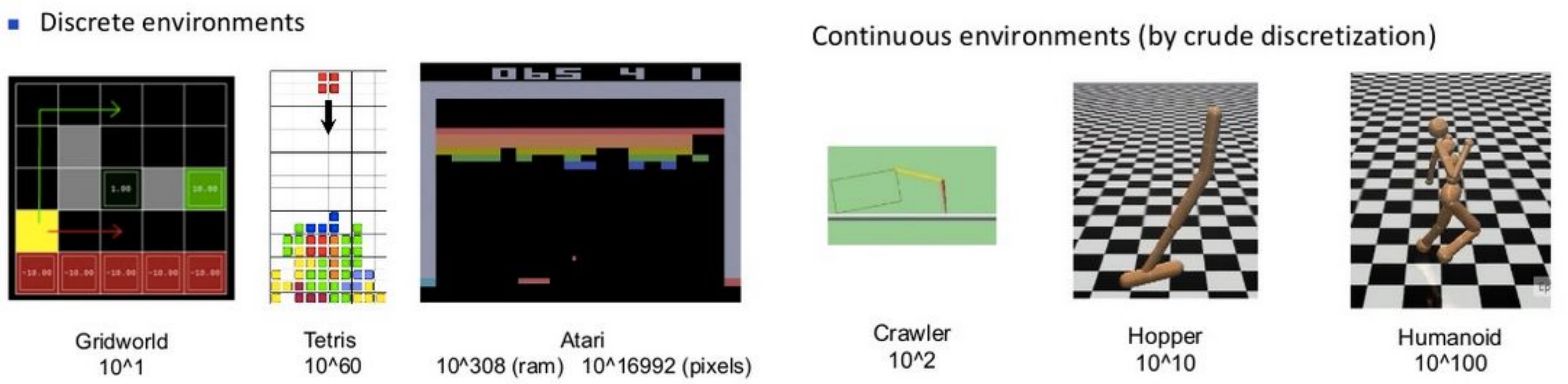
Problems with current approach ?

➔ Set of states
➔ Resources needs
➔ Performance



Source: pixabay

Scaling up



Source: Berkeley AI Research Lab

Scaling up

<u>Solution:</u>

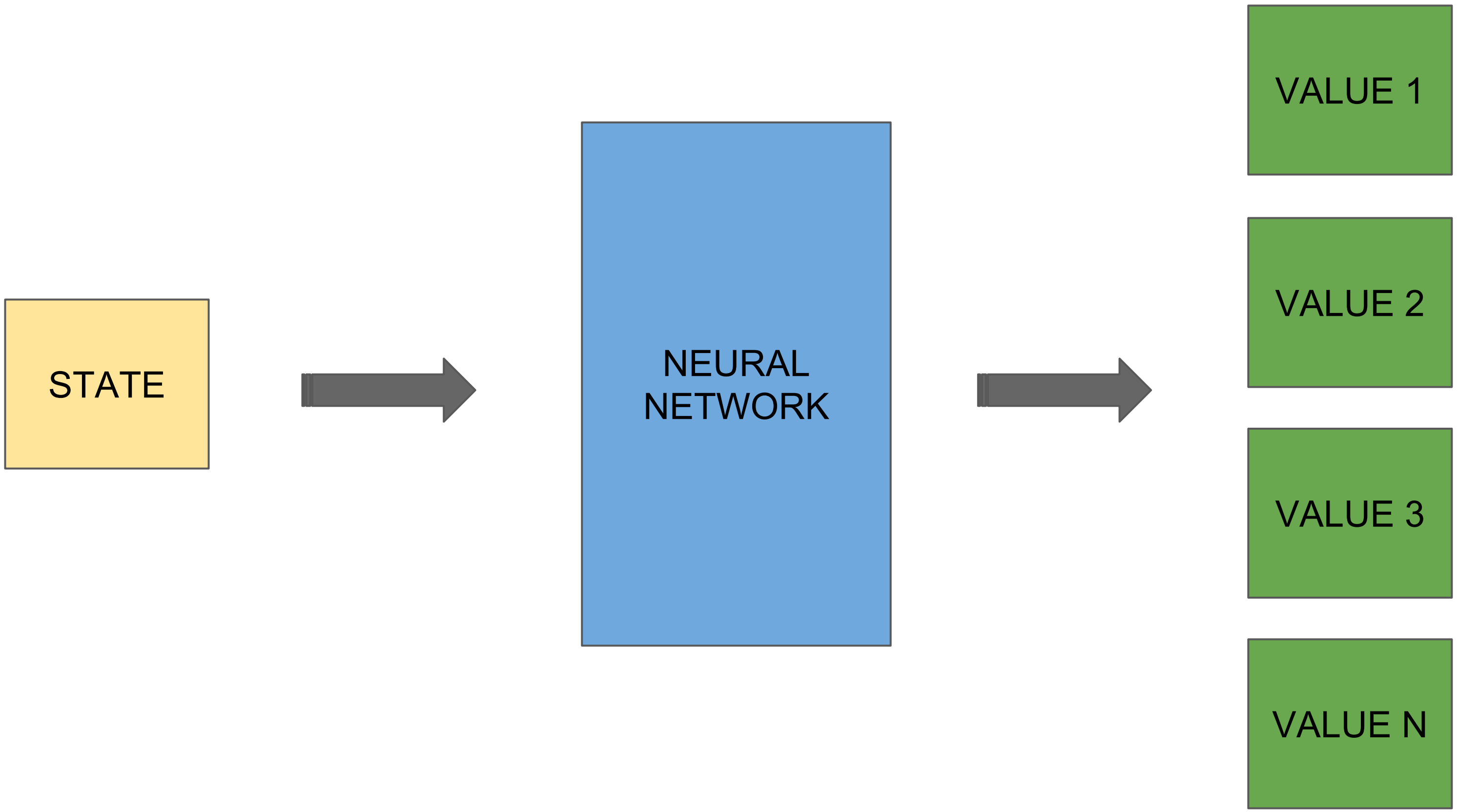
Generalize

Function approximation

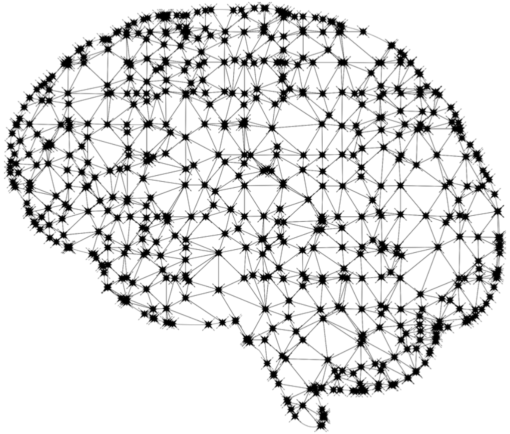
Understand the rules and apply it to future, similar situations

Source: pixabay

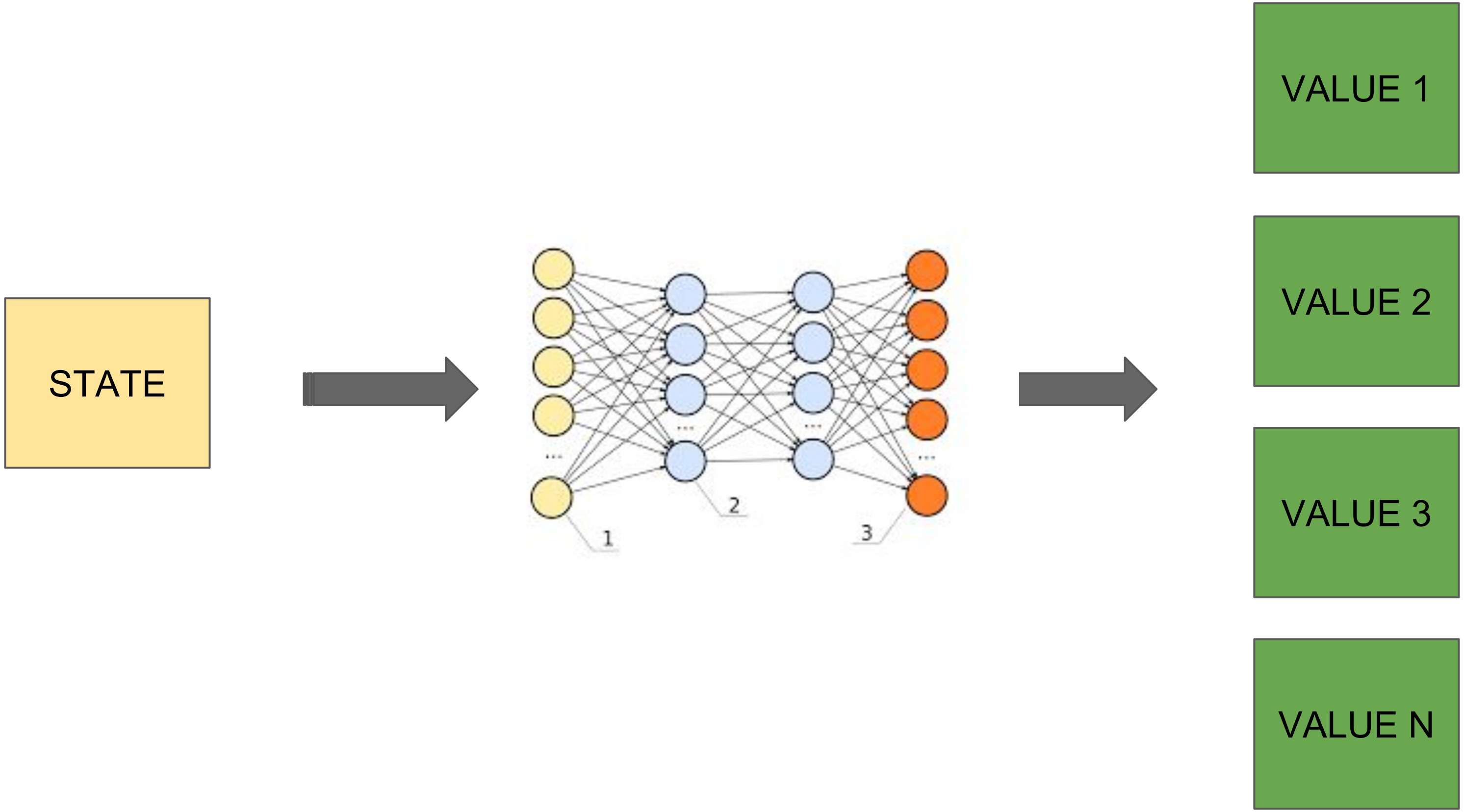
Scaling up



Neural Networks revisited

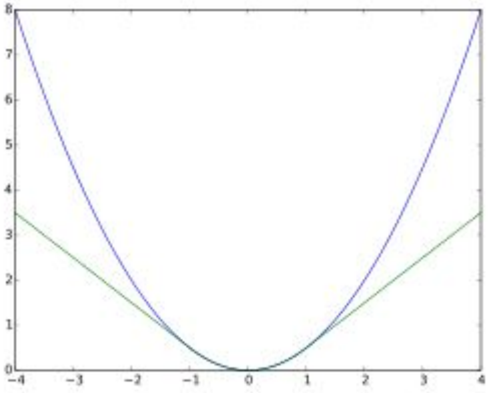


Neural Networks



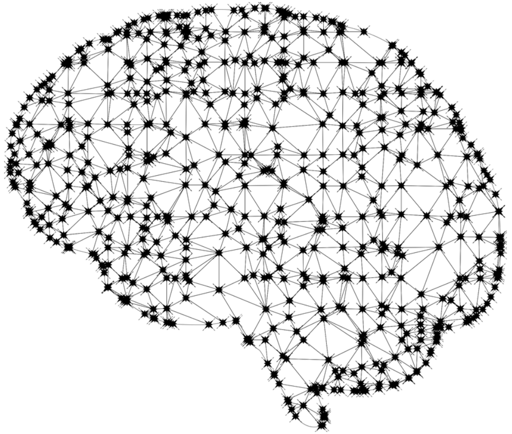
Neural Networks

- Activation functions
  - ReLU
  - Tanh
- Loss functions
  - MSE
  - Huber loss
- Optimizers
  - Adam
  - RMSProp



Huber loss vs MSE
Source: wikipedia

DQN



Deep Q learning

Paper “Playing Atari with Deep Reinforcement Learning”

https://arxiv.org/pdf/1312.5602v1.pdf

Source: Deepmind

Deep Q learning

Problems listed in Deepmind's paper:

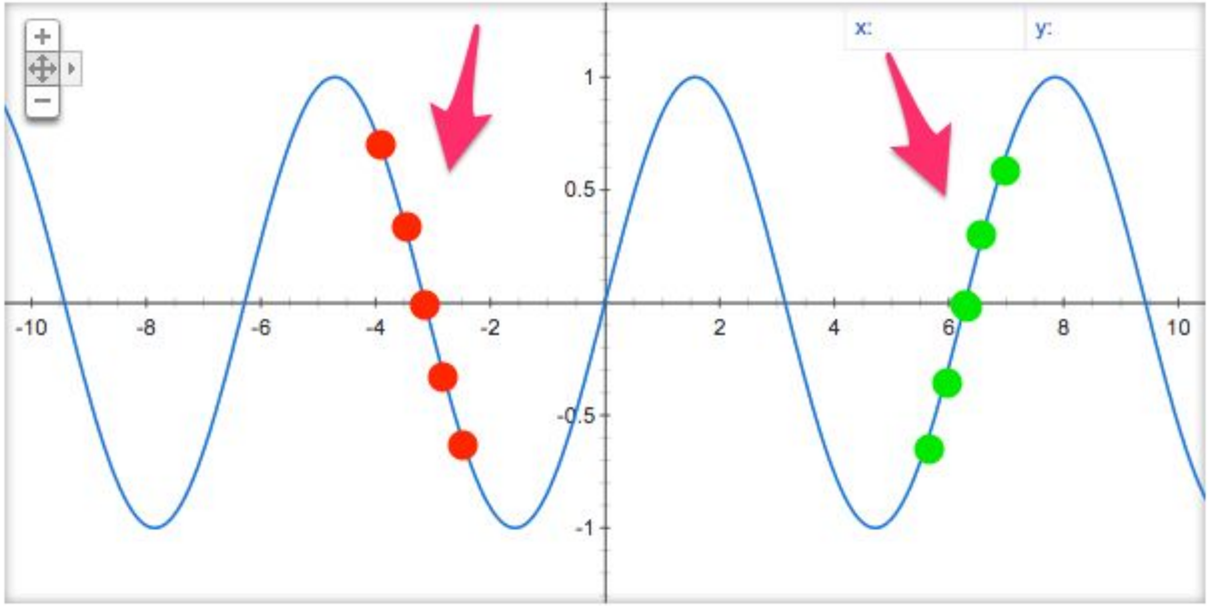
➔ highly correlated data

➔ non-stationary distributions

Deep Q learning

Problems listed in Deepmind's paper:

➔ Highly correlated data



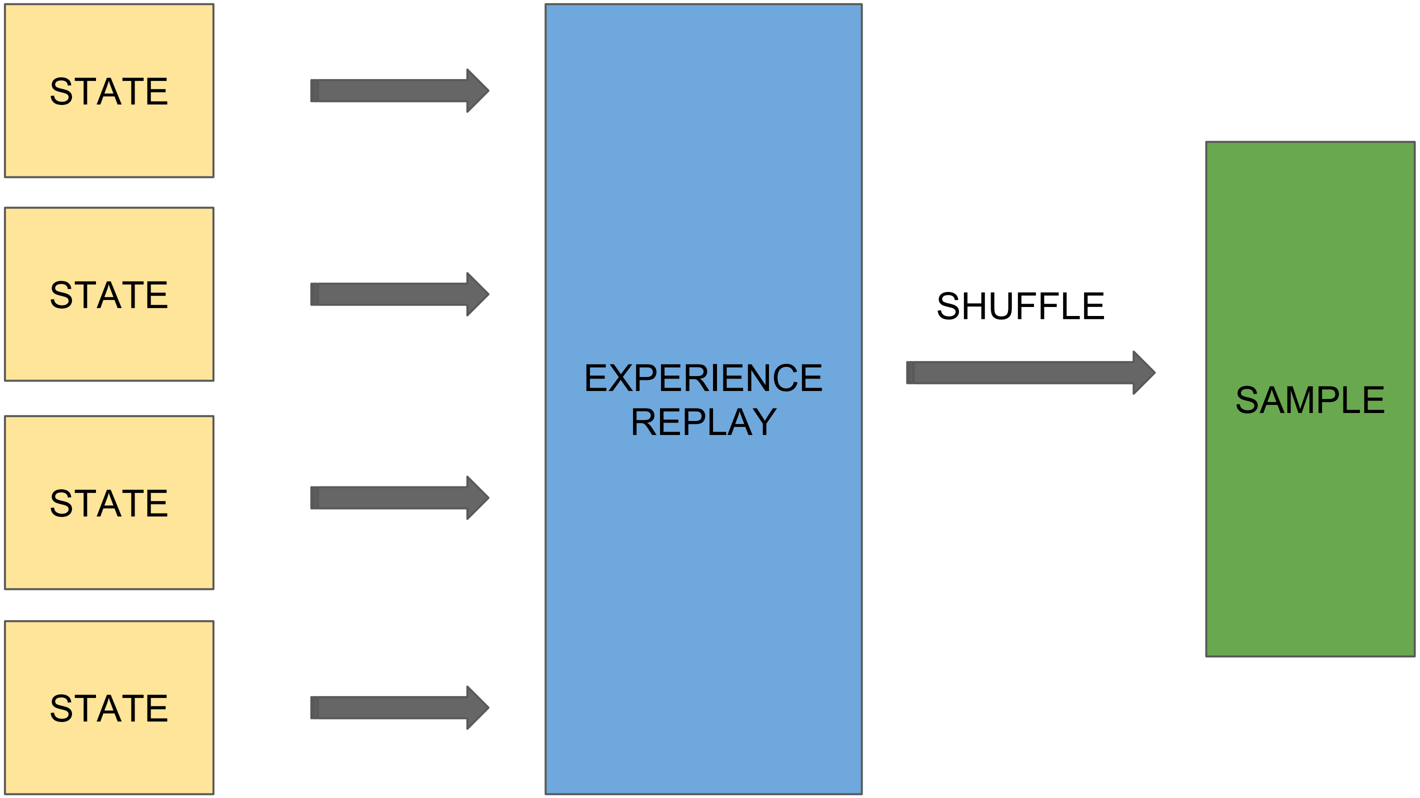
Deep Q learning

Problems listed in Deepmind’s paper:

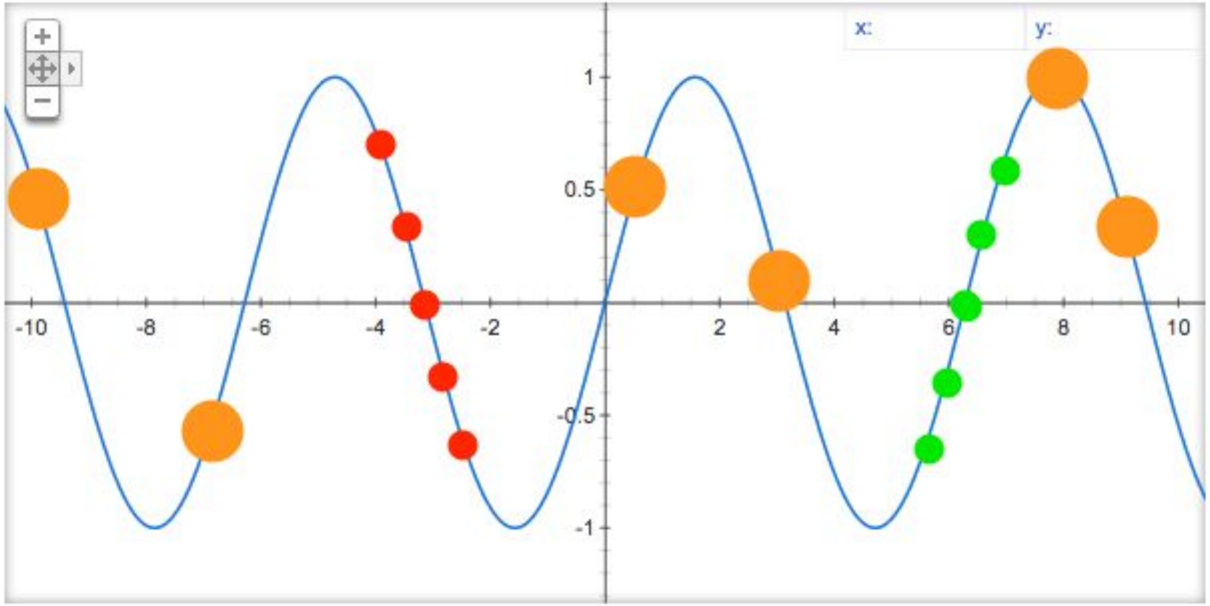
➔ Non stationary distributions

$$Q(s, a) = r + \gamma * max_{a'} Q(s', a')$$

Deep Q learning



Deep Q learning



Deep Q learning

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Source: DeepMind's "Playing Atari with Deep Reinforcement Learning"

DQN

Paper “Human-level control through deep reinforcement learning”

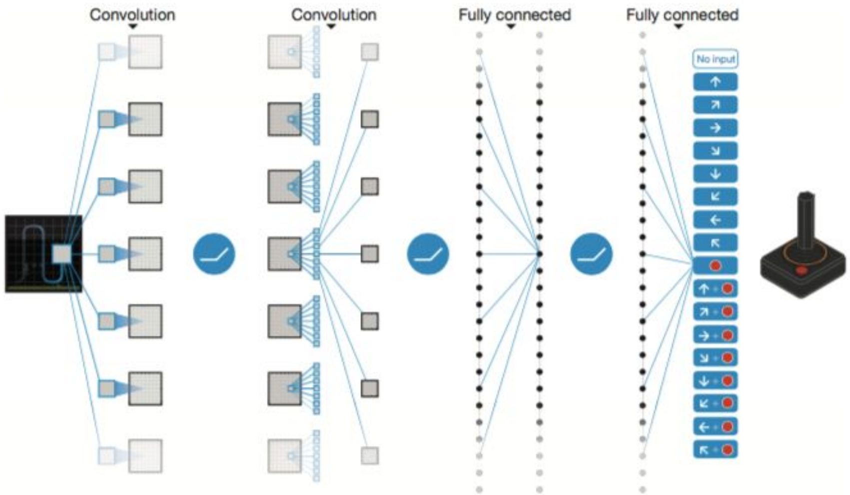
https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

Code:

https://github.com/deepmind/dqn

Patent:

https://patents.google.com/patent/US20150100530A1/en



DQN

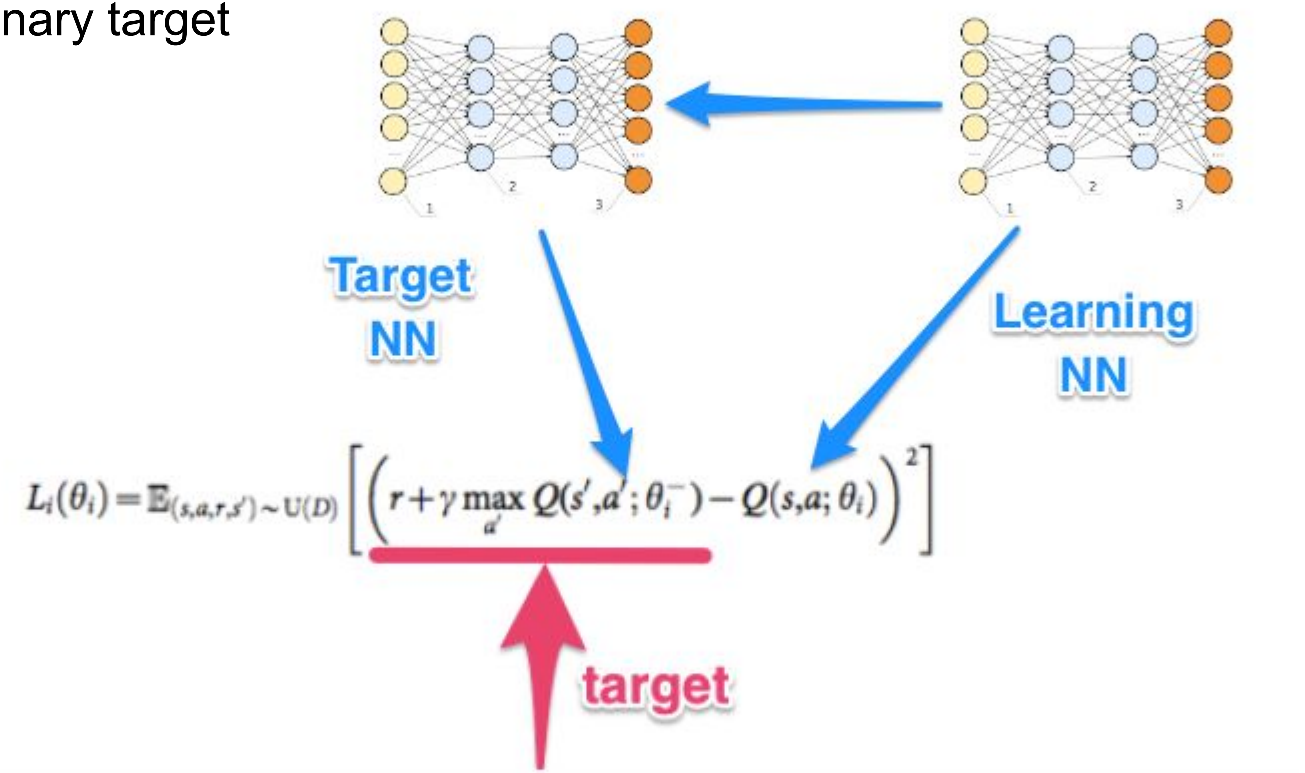
Problems listed in Deepmind's paper:

➔ Non stationary target

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(\underbrace{r + \gamma \max_{a'} Q(s',a';\theta_i^-)}_{\text{target}} - Q(s,a;\theta_i) \right)^2 \right]$$

DQN

➔ Non stationary target



DQN

Additional “trick”:

➔ Error clipping

Pytorch implementation:

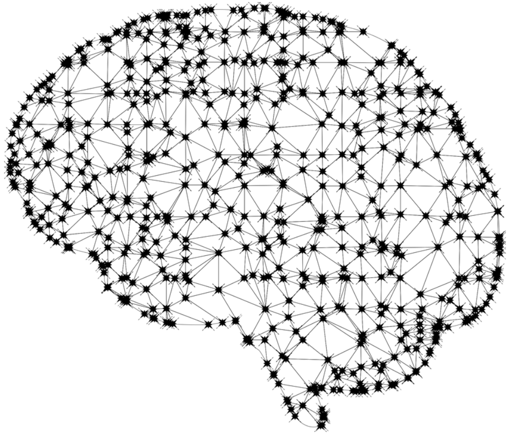
```
loss.backward()

for param in self.agent.parameters():
    param.grad.data.clamp_(-1, 1)

self.optimizer.step()
```

DQN

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Source: DeepMind's "Human-level control through deep reinforcement learning"

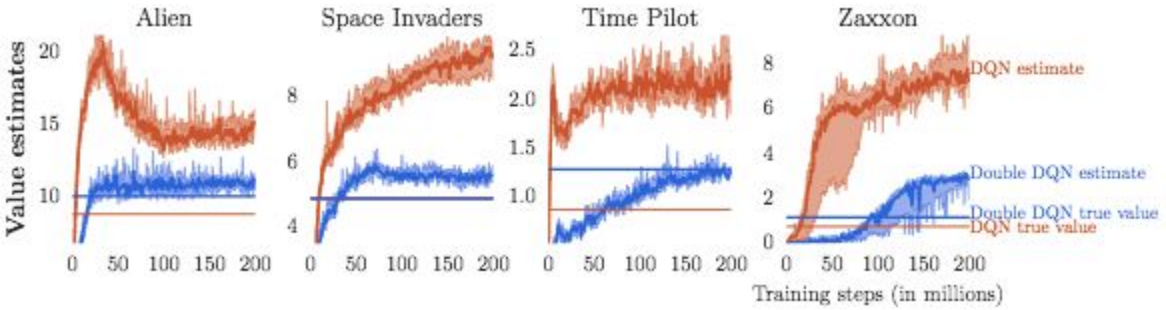
DQN Improvements



DQN improvements

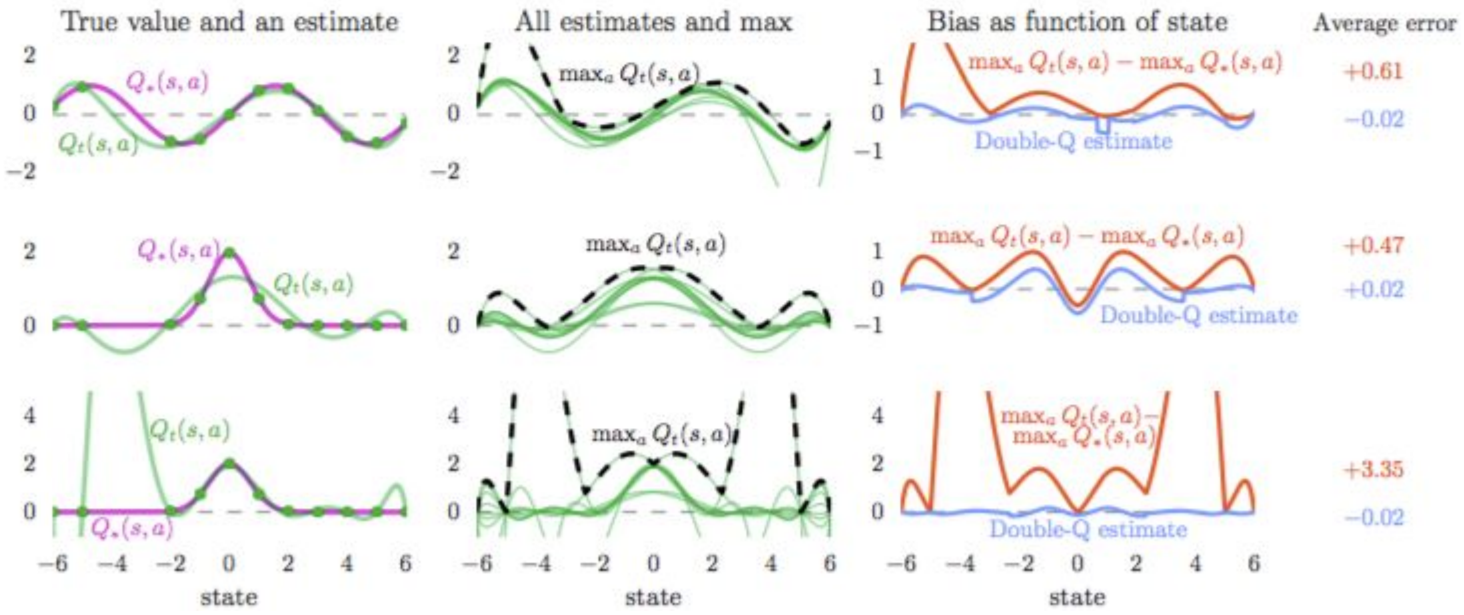
Paper “Deep Reinforcement Learning with Double Q-learning”

https://arxiv.org/pdf/1509.06461.pdf



DQN improvements

Double Q Learning



DQN improvements

Double Q Learning

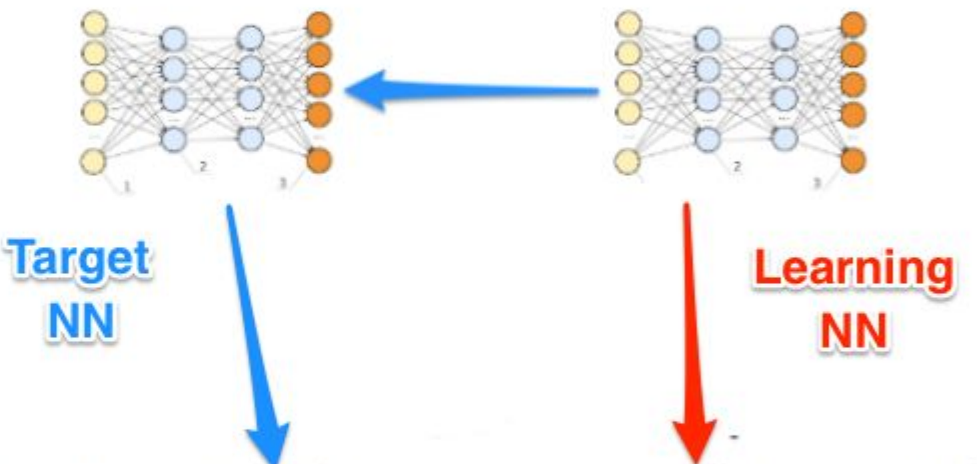
$$Y_t^{\text{DoubleDQN}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \boldsymbol{\theta}_t), \boldsymbol{\theta}_t^-)$$

Explanation:

$$Y_t^{\text{DoubleDQN}} \equiv R_{t+1} + \gamma \underbrace{Q}_{\text{NN for calculating value}}(S_{t+1}, \underset{a}{\operatorname{argmax}}\, \underbrace{Q(S_{t+1}, a; \boldsymbol{\theta}_t)}_{\text{NN for choosing best action}}, \boldsymbol{\theta}_t^-)$$

DQN improvements

Double Q Learning

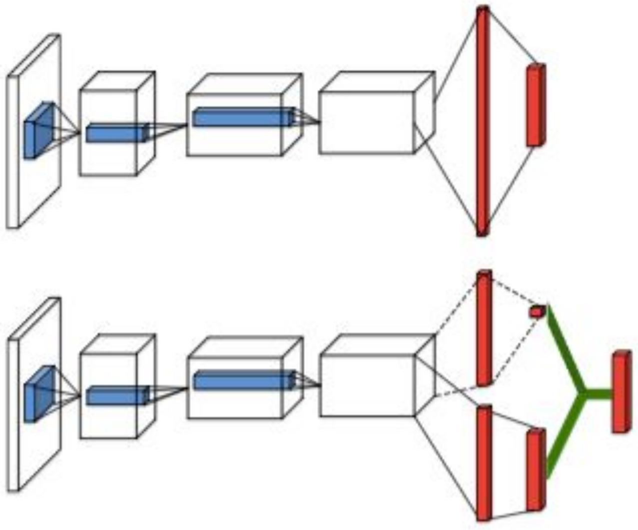


$$Y_t^{\text{DoubleDQN}} \equiv R_{t+1} + \gamma \underline{Q}(S_{t+1}, \operatorname*{argmax}_a \underline{Q(S_{t+1}, a; \boldsymbol{\theta}_t)}, \boldsymbol{\theta}_t^-)$$

DQN improvements

Paper “Dueling Network Architectures for Deep Reinforcement Learning”

https://arxiv.org/pdf/1511.06581.pdf



DQN improvements

Dueling DQN

Advantage function:

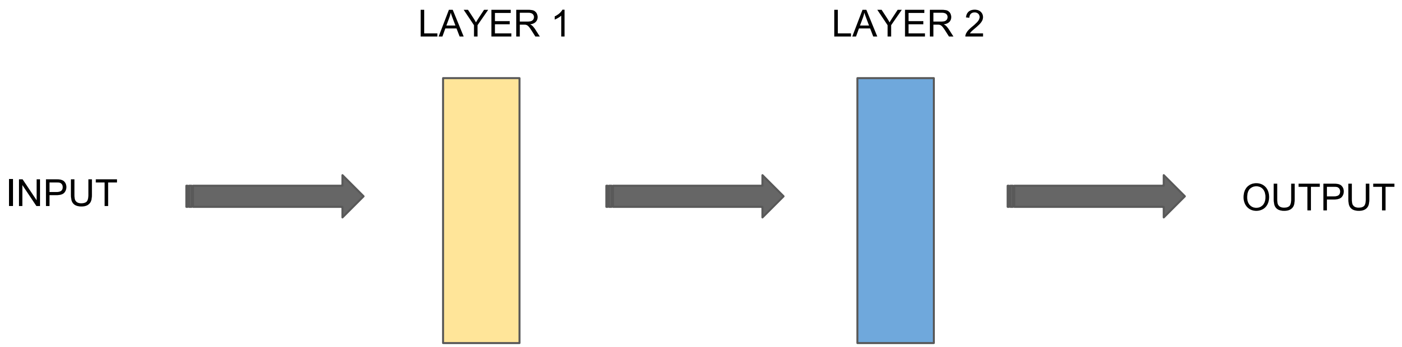
$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

Q function:

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

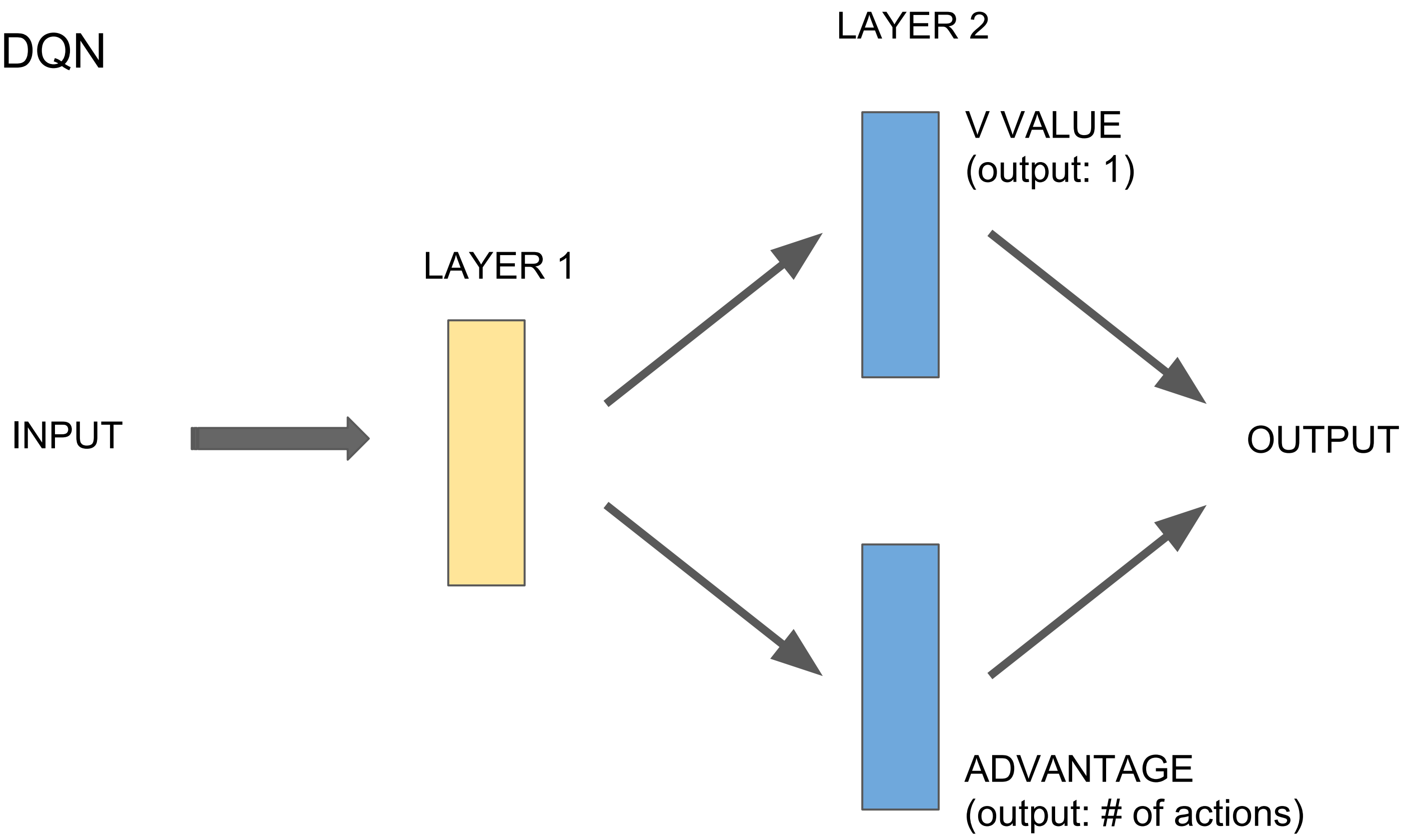
Deep Q learning

Dueling DQN

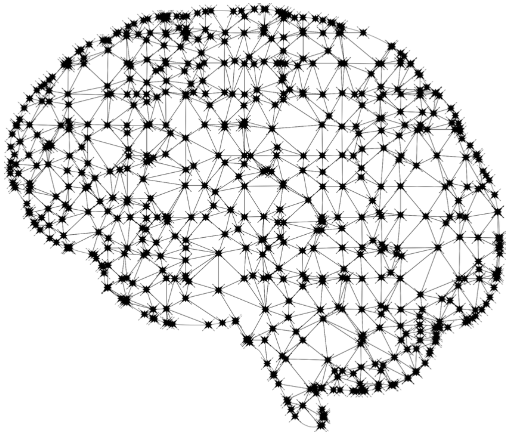


Deep Q learning

Dueling DQN

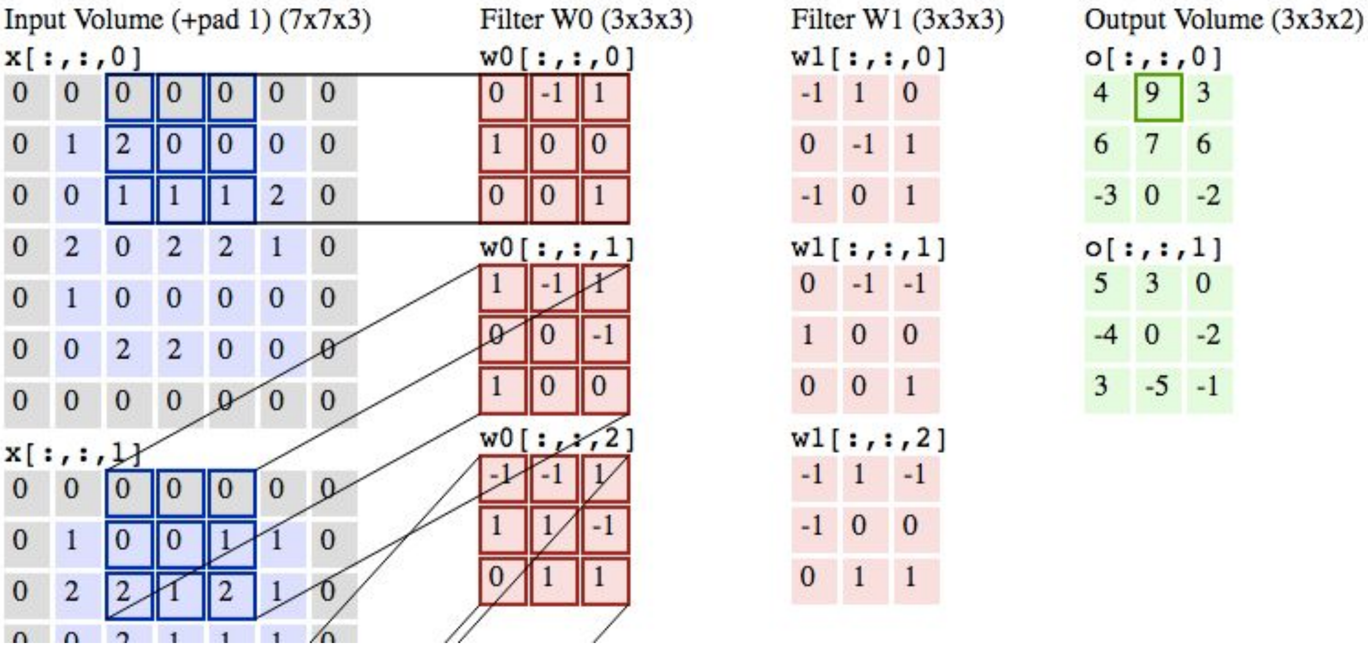


DQN + CNN



Deep Reinforcement learning

CNN



Source: cs231n.github.io

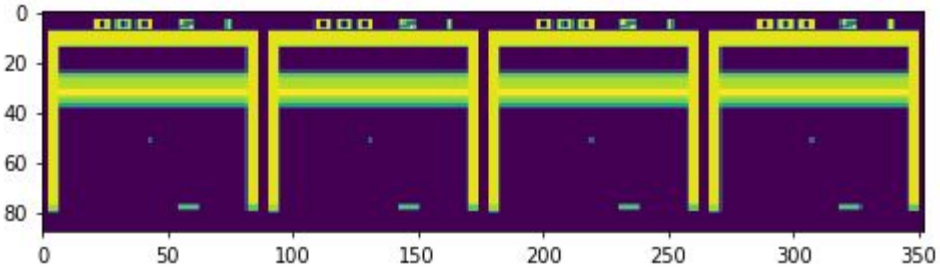
Deep Reinforcement learning

Deepmind’s CNN

Layer	Input	Filter size	Stride	# of filters	Activation	Output
conv1	84x84x4	8	4	32	ReLU	20x20x32
conv2	20x20x32	4	2	64	ReLU	9x9x64
conv3	9x9x64	3	1	64	ReLU	7x7x64
fc1	7x7x64			512	ReLU	512
fc2	512				Linear	# of actions

DQN potential improvements

➔ Stack 4 screens



➔ Play around with number of layers / settings of Neural Networks

➔ Play around with number of neurons in hidden layer(s)

➔ Play around with hyperparameters, optimizer and loss function

➔ Get more / different kind of experience (egreedy)

➔ More DQN improvements (for example PER, Rainbow)

➔ Different methods

Example - Pong

```
learning_rate = 0.0001
gamma = 0.99
replay_mem_size = 100000
batch_size = 32
update_target_frequency = 5000
double_dqn = True
egreedy = 0.9
egreedy_final = 0.01
egreedy_decay = 10000
clip_error = True
normalize_image = True
```

Possible results

(GeForce GTX 1080)

considering solved as of average > 18 points over last 100 episodes

```
*** Episode 290 ***
Av.reward: [last 10]: 19.50, [last 100]: 17.67, [all]: -0.09
epsilon: 0.01, frames_total: 633567
Elapsed time:  01:14:30
SOLVED! After 300 episodes

*** Episode 300 ***
Av.reward: [last 10]: 19.20, [last 100]: 18.03, [all]: 0.55
epsilon: 0.01, frames_total: 652434
Elapsed time:  01:16:44
```

What’s next?