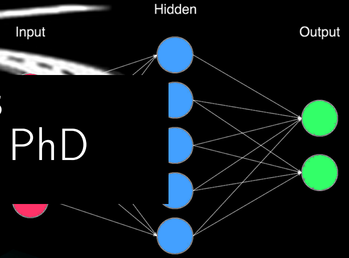
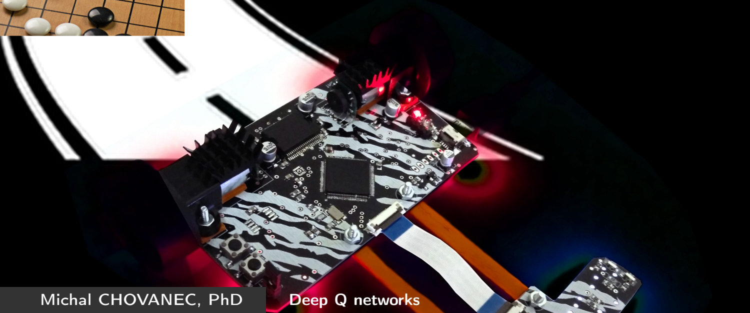
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

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



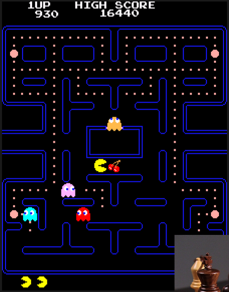


Deep Q networks

Michal CHOVANEC, PhD

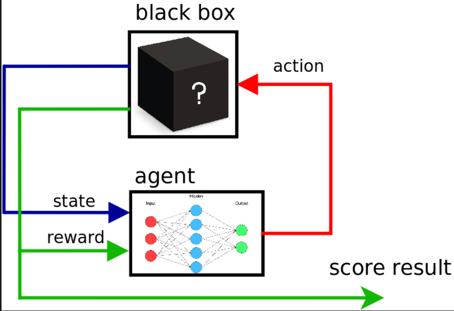
Reinforcement learning

- learn from punishment and rewards
- learn to play a game with unknown rules



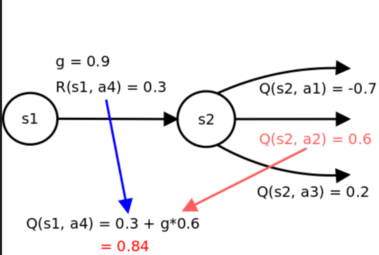
Reinforcement learning

- obtain **state**
- choose **action**
- **execute** action
- obtain **reward**
- learn from **experiences**



Q learning (1989)

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



where
s is state
a is action
s' is next state
a' is best action in next state
$R(s, a)$ is reward
$\gamma \in \langle 0, 1 \rangle$ is discount factor

Deep Q network - DQN (2013)

Approximate $Q(s,a)$ using deep neural network as $\hat{Q}(s,a;w)$, where w are learnable network parameters

$$Q(s,a) = R + \gamma \max_{\alpha'} Q(s',\alpha')$$
$$\hat{Q}(s,a;w) = R + \gamma \max_{\alpha'} \hat{Q}(s',\alpha';w)$$

error to minimize

$$E = \underbrace{R + \gamma \max_{\alpha'} \hat{Q}(s',\alpha';w)}_{\text{target value}} - \underbrace{\hat{Q}(s,a;w)}_{\text{predicted value}}$$

weights gradient

$$\Delta w = \eta E \nabla_w \hat{Q}(s,\alpha,w)$$

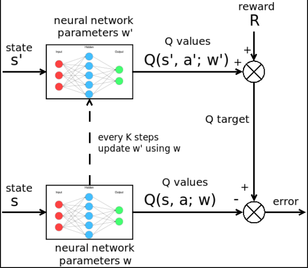
Deep Q network - DQN

- **unstable training** :
  non-stationary target value $\hat{Q}(s, a; w)$, depends on w
- **solution - fix w** :
  using temporary network with w' weights.

DQN equation

$$\hat{Q}(s, a; w) = R + \gamma \max_{\alpha'} \hat{Q}(s', \alpha'; w')$$

$$E = R + \gamma \max_{\alpha'} \hat{Q}(s', \alpha'; w') - \hat{Q}(s, a; w)$$

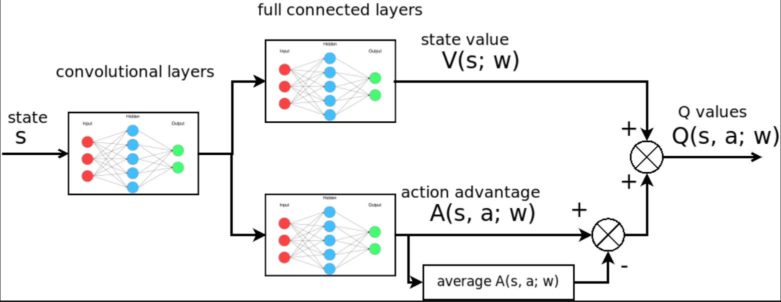


Dueling deep Q network - DDQN (2016)

$$\hat{Q}(s,a,w) = \underbrace{\hat{V}(s,w)}_{\text{value for being in state s}} + \underbrace{\hat{A}(s,a,w)}_{\text{advantage of taking action a at state s}}$$

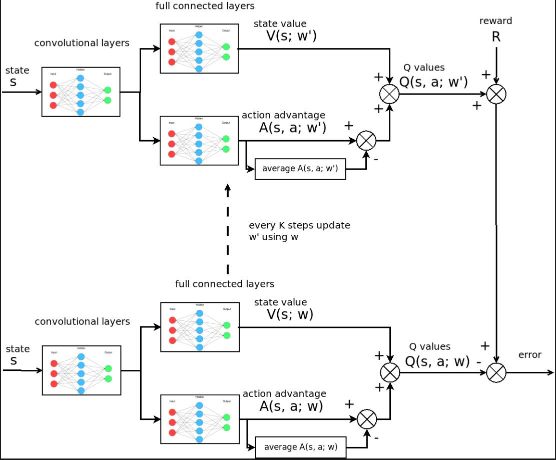
to avoid identifiability we substract average value of A truth all actions

$$\hat{Q}(s,a,w) = \hat{V}(s,w) + \hat{A}(s,a,w) - \frac{1}{N_{\alpha'}} \sum_{\alpha'} \hat{A}(s,\alpha',w)$$



Dueling deep Q network - DDQN

$$\hat{Q}(s,a,w) = \hat{V}(s,w) + \hat{A}(s,a,w) - \frac{1}{N_{\alpha'}}\sum_{\alpha'}\hat{A}(s,\alpha',w)$$



Dueling deep Q network - DDQN

using DQN equation

$$\hat{Q}(s, a; w) = R + \gamma \max_{\alpha'} \hat{Q}(s', \alpha'; w')$$

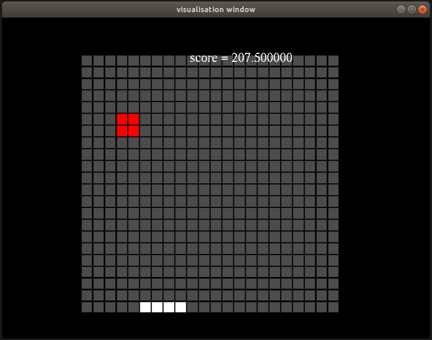
we obtain dueling deep Q network equation

$$\hat{Q}(s, a; w) = R + \gamma \left(\hat{V}(s', w') + \max_{\alpha'} \hat{A}(s', \alpha', w') - \frac{1}{N_{\alpha'}} \sum_{\alpha'} \hat{A}(s', \alpha', w') \right)$$

and finally the weights learning rule

$$\Delta w = \eta \left(R + \gamma \left(\hat{V}(s', w') + \max_{\alpha'} \hat{A}(s', \alpha', w') - \frac{1}{N_{\alpha'}} \sum_{\alpha'} \hat{A}(s', \alpha', w') \right) - \hat{Q}(s, a; w) \right) \nabla_w \hat{Q}(s, a; w)$$

Experiments



Network hyperparameters

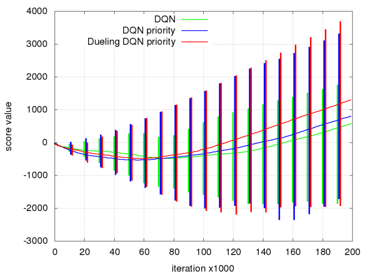
- init weights : XAVIER
- learning rate : 0.0005
- L1 regularization : 0.0000001
- L2 regularization : 0.0000001
- dropout : 0.2
- minibatch size : 32

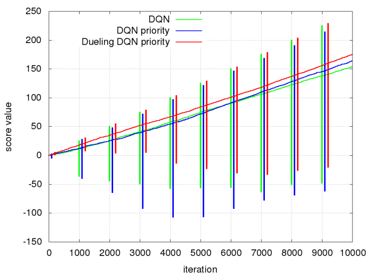
RL hyperparameters

- experience buffer size : 1024
- gamma : 0.95
- epsilon training : 0.1
- epsilon testing : 0.05

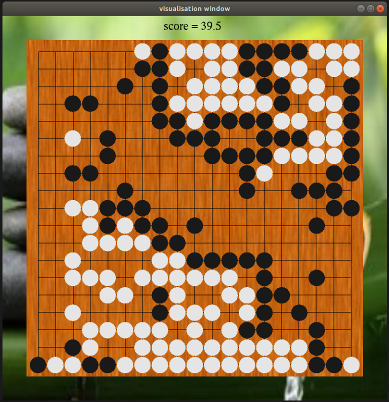
layer type	input dimensions	kernel dimensions
dense convolution	22x22x1	3x3x8
dense convolution	22x22x9	3x3x8
dense convolution	22x22x17	3x3x8
dense convolution	22x22x25	3x3x8
convolution	22x22x33	3x3x32
full connected	22x22x32	actions_count

Results


DQN
DQN priority
Dueling DQN priority
score value
iteration x1000
4000
3000
2000
1000
0
-1000
-2000
-3000
0
20
40
60
80
100
120
140
160
180
200


DQN
DQN priority
Dueling DQN priority
score value
iteration
250
200
150
100
50
0
-50
-100
-150
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Playing GO (October 2017)

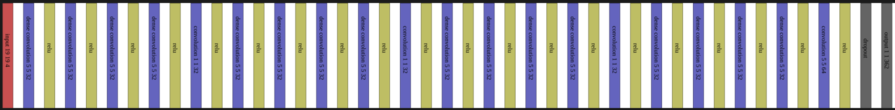


- **supervised training** - train game using Masters games
- **reinforcement learning** - let play two networks against each other

Network architecture

we need to go much deeper for GO

- **20 convolutional layers**
  4 blocks with 4 dense conv + 1 conv layer
- **input**
  4 matrices 19x19: black stones, white stones, empty fields, active player
- **output**
  recommended moves 19x19 + 1 for pass = 362 outputs



Network architecture

layer type	input size	kernel size	output size
dense convolution	19x19x4	5x5x32	19x19x36
dense convolution	19x19x36	5x5x32	19x19x68
dense convolution	19x19x68	5x5x32	19x19x100
dense convolution	19x19x100	5x5x32	19x19x132
convolution	19x19x132	1x1x32	19x19x32
dense convolution	19x19x32	5x5x32	19x19x64
dense convolution	19x19x64	5x5x32	19x19x96
dense convolution	19x19x96	5x5x32	19x19x128
dense convolution	19x19x128	5x5x32	19x19x160
convolution	19x19x160	1x1x32	19x19x32
dense convolution	19x19x32	5x5x32	19x19x64
dense convolution	19x19x64	5x5x32	19x19x96
dense convolution	19x19x96	5x5x32	19x19x128
dense convolution	19x19x128	5x5x32	19x19x160
convolution	19x19x160	1x1x32	19x19x32
dense convolution	19x19x32	5x5x32	19x19x64
dense convolution	19x19x64	5x5x32	19x19x96
dense convolution	19x19x96	5x5x32	19x19x128
dense convolution	19x19x128	5x5x32	19x19x160
convolution	19x19x160	5x5x64	19x19x64
full connected	19x19x64	19x19x362	362

Supervised results

74	5	6	6	13	12	26	25	23	19	21	20	19	22	14	12	12	15	82
12	26	43	37	31	35	32	33	35	30	30	35	35	36	31	30	43	26	7
25	39	50	73	42	50	34	35	37	43	34	32	38	50	48	58	48	52	15
15	29	55	82	43	46	41	41	41	31	38	37	38	44	35	55	48	40	16
28	37	62	50	47	43	43	50	51	50	47	46	46	45	44	56	49	48	20
31	34	64	53	44	43	44	48	43	50	46	47	43	44	44	52	54	43	17
35	35	45	43	44	44	48	49	45	45	51	51	44	42	46	49	36	47	21
31	34	46	46	49	45	51	46	50	53	48	48	47	46	50	45	42	49	27
29	33	41	41	48	49	46	50	50	52	47	50	48	46	47	41	46	49	25
27	41	39	34	48	47	48	49	49	54	50	51	48	49	52	35	49	45	29
31	37	42	42	47	49	47	47	49	48	49	50	48	47	47	40	43	45	34
28	39	44	46	45	46	45	49	51	49	51	45	46	43	46	41	44	49	22
29	38	38	45	50	44	48	50	45	43	50	49	51	50	41	49	43	49	20
31	34	73	49	40	41	42	45	42	45	47	47	47	45	48	53	53	49	25
32	30	57	52	44	41	42	44	47	50	51	46	46	43	47	52	49	40	18
24	34	64	77	47	53	41	45	43	30	45	46	41	48	40	61	71	39	8
24	51	56	65	46	58	45	42	41	46	38	35	38	45	63	76	51	52	9
32	26	52	47	44	46	47	39	40	44	43	42	43	37	45	44	46	46	18
77	3	4	3	7	13	13	17	29	34	19	25	20	23	13	9	8	13	66
																		100

Usefull links

- CHRISTOPHER J.C.H. WATKINS : Q-learning
  http://www.gatsby.ucl.ac.uk/~dayan/papers/cjch.pdf
- Richard S. Sutton : Reinforcement Learning: An Introduction
  https://www.amazon.com/Reinforcement-Learning-Introduction-Adaptive-Computation/dp/0262193981
- Google DeepMind : Playing Atari with Deep Reinforcement Learning
  https://arxiv.org/pdf/1312.5602.pdf
- Google DeepMind : Dueling Network Architectures for Deep Reinforcement Learning
  https://arxiv.org/pdf/1511.06581.pdf
- Google DeepMind :Mastering the Game of Go without Human Knowledge
  https://deepmind.com/documents/119/agz_unformatted_nature.pdf
- Andrej Karpathy : Pong from pixels
  http://karpathy.github.io/2016/05/31/rl/
- Maxim Lapan : Deep reinforcement learning
  https://www.amazon.com/Practical-Reinforcement-Learning-Maxim-Lapan/dp/1788834240
- Mohit Sewak : Practical Convolutional Neural Networks
  https://www.amazon.com/Practical-Convolutional-Neural-Networks-Implement/dp/1788392302
- Densely Connected Convolutional Networks
  https://arxiv.org/pdf/1608.06993.pdf

Q&A

michal chovanec (michal.nand@gmail.com)
www.youtube.com/channel/UCzVvP2ou8v3afNiVrPAHQGg
github https://github.com/michalnand