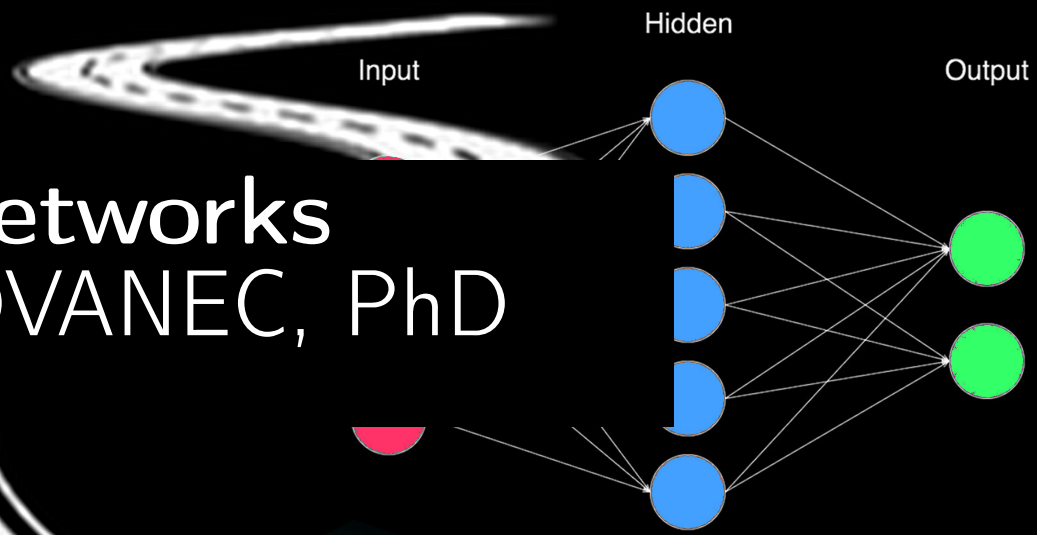
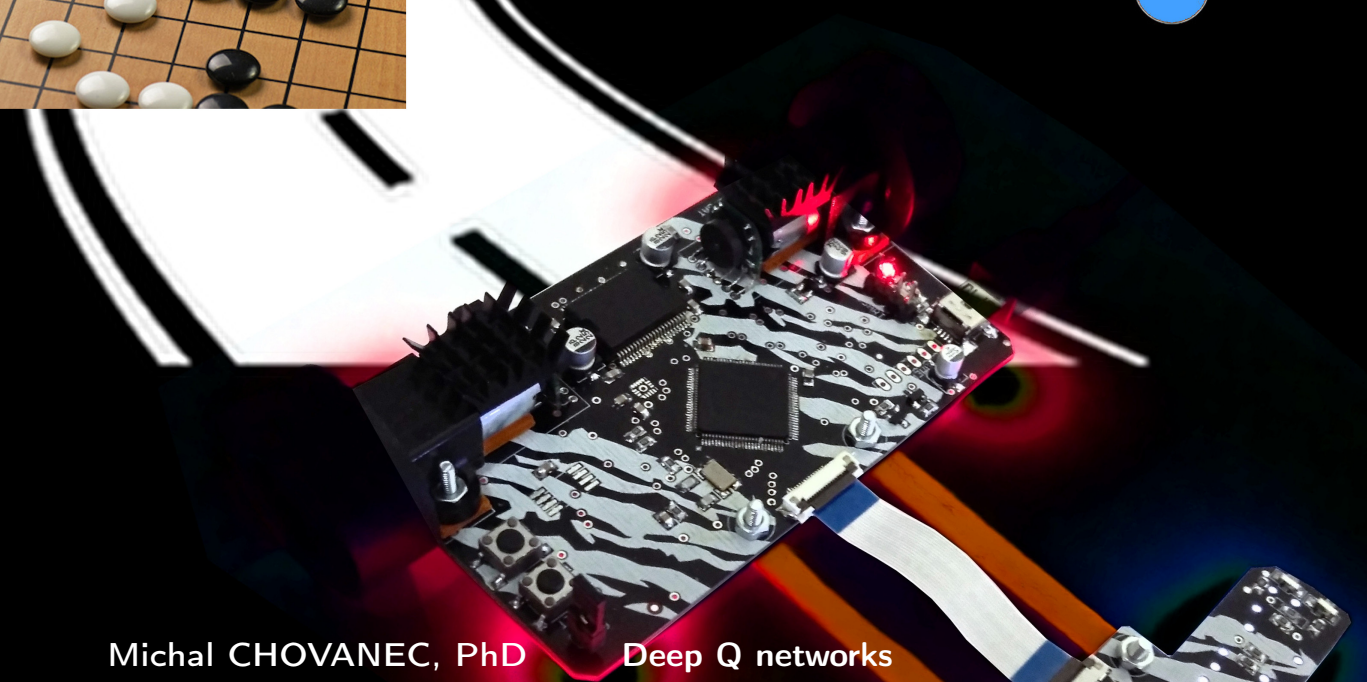
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

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



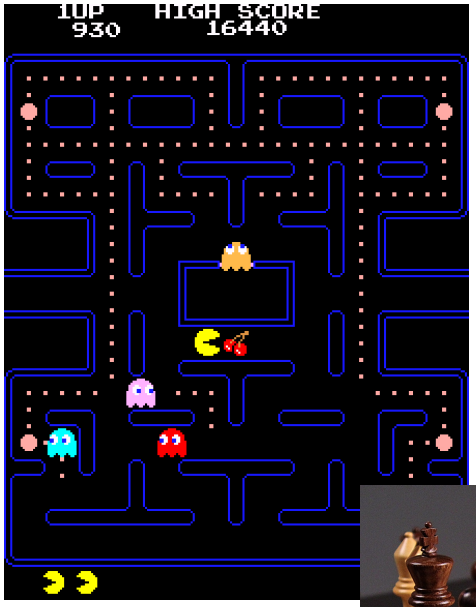
Deep Q networks

Michal CHOVANEC, PhD



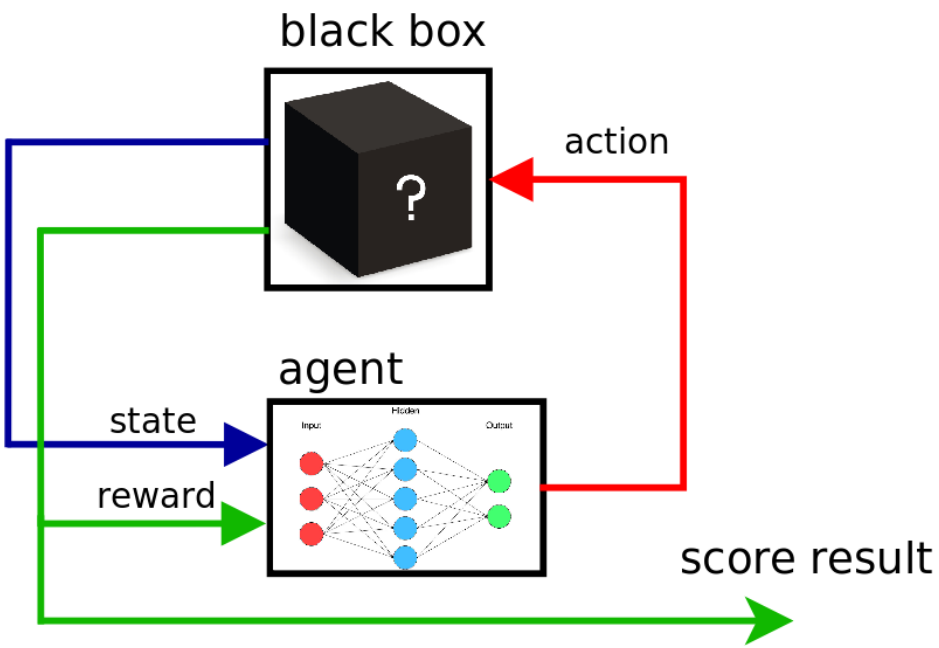
Reinforcement learning

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



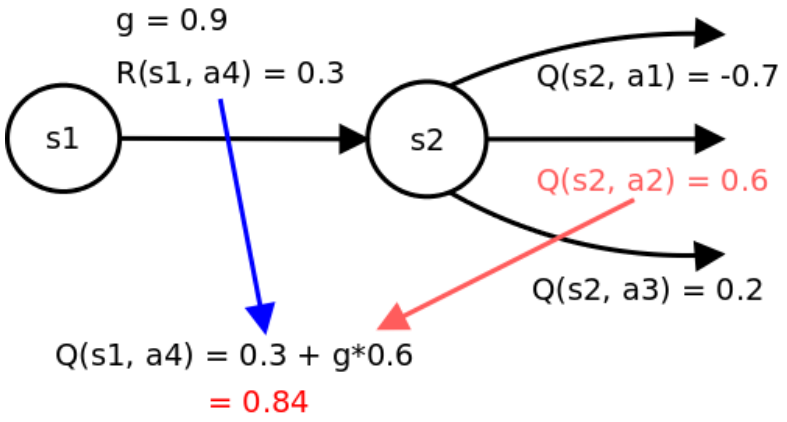
Reinforcement learning

- 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

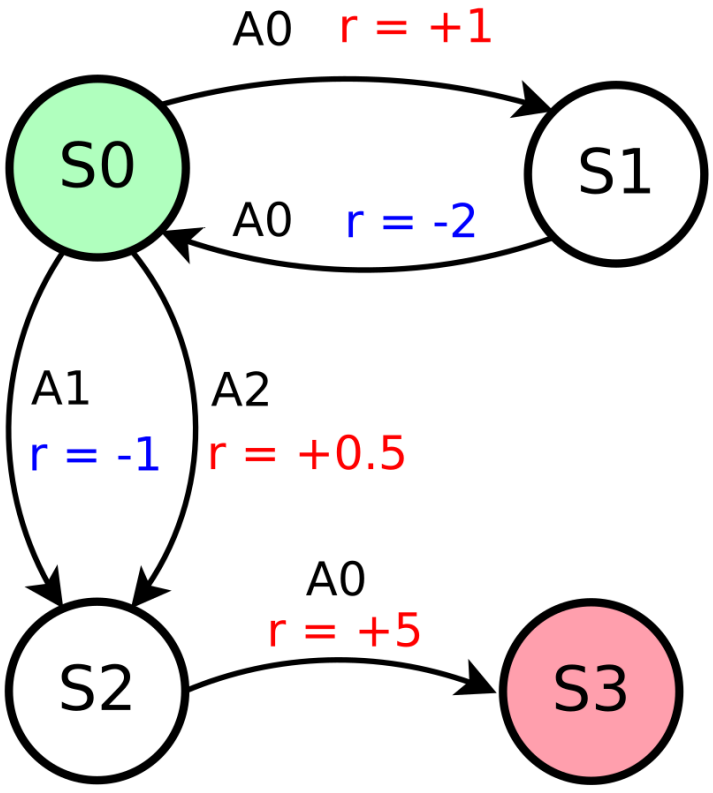
s' is next state

a' is best action in next state

$R(s, a)$ is reward

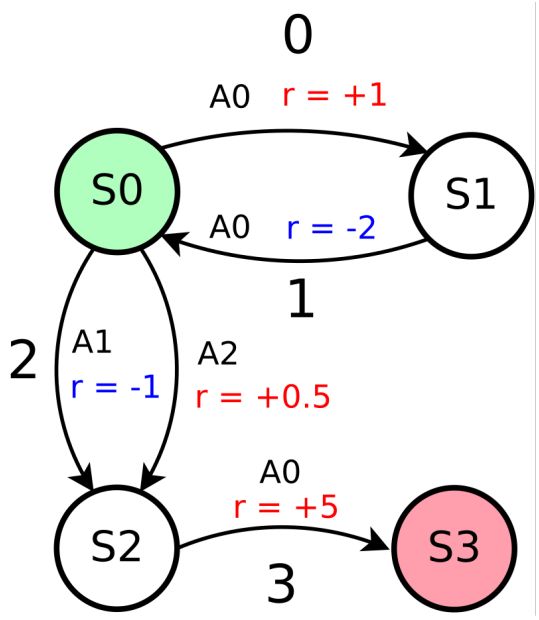
$\gamma \in \langle 0, 1 \rangle$ is discount factor

Q learning - example



S	reward			Q(s,a)		
	A0	**A1**	**A2**	**A0**	**A1**	**A2**
0	1	-1	0.5	0	0	0
1	-2	x	x	0	0	0
2	5	x	x	0	0	0
3	x	x	x	0	0	0

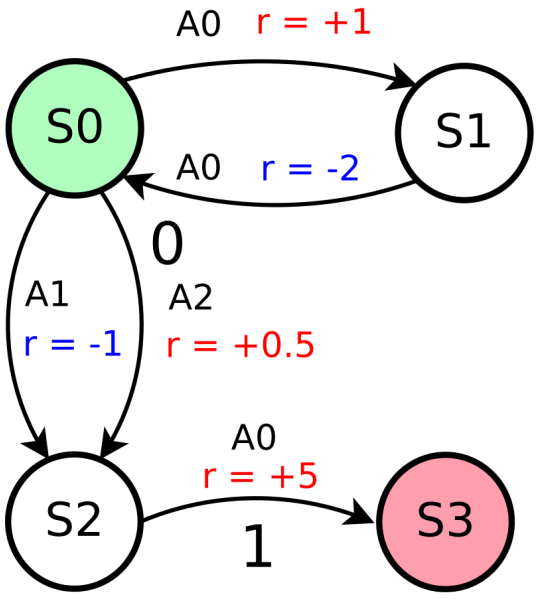
Q learning - example



S	reward			Q(s,a)		
	A0	A1	A2	A0	A1	A2
0	1	-1	0.5	**2.035**	**3.5**	0
1	-2	x	x	**1.15**	0	0
2	5	x	x	**5**	0	0
3	x	x	x	0	0	0

state	S0	S1	S0	S2	S3
action	A0	A0	A1	A0	
reward	+1	-2	-1	+5	
Q	1 + 0.9*1.15 = 2.035	-2 + 0.9*3.5 = 1.15	-1+0.9*5 = 3.5	5	

Q learning - example



S	reward			Q(s,a)		
	A0	**A1**	**A2**	**A0**	**A1**	**A2**
0	1	-1	0.5	**2.035**	**3.5**	**5**
1	-2	x	x	**1.15**	0	0
2	5	x	x	**5**	0	0
3	x	x	x	0	0	0

state	**S0**	**S2**	**S3**
action	A2	A0	
reward	0.5	+5	
Q	0.5 + 0.9*5 = 5	5	

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

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

weights gradient

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

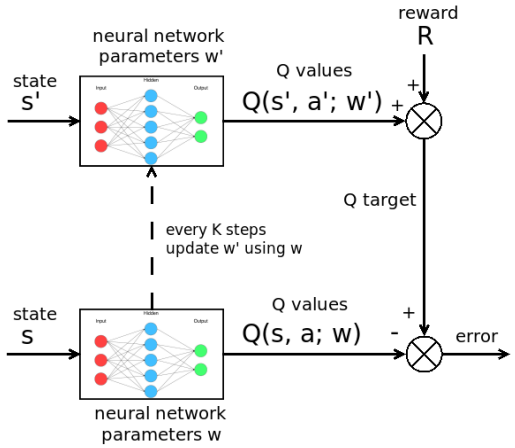
This naive NN doesn't works - solution DQN

Deep Q network - DQN

- **correlated states** :
  experience replay buffer
- **unstable training** :
  non-stationary target value $\hat{Q}(s, a; w)$, depends on w, use temporary fixed weights w'
- **unknow gradients values** :
  clip or normalise gradients into $\langle -1, 1 \rangle$

DQN equation

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

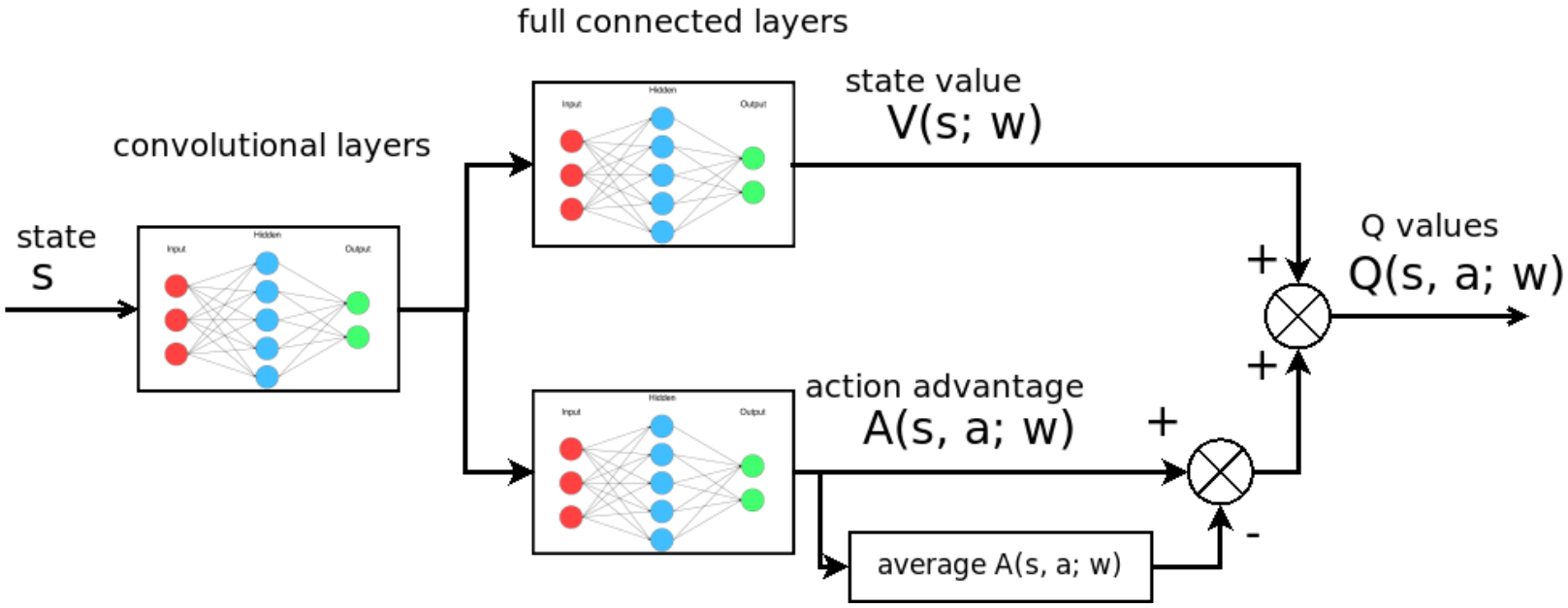


Dueling deep Q network - DDQN (2016)

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

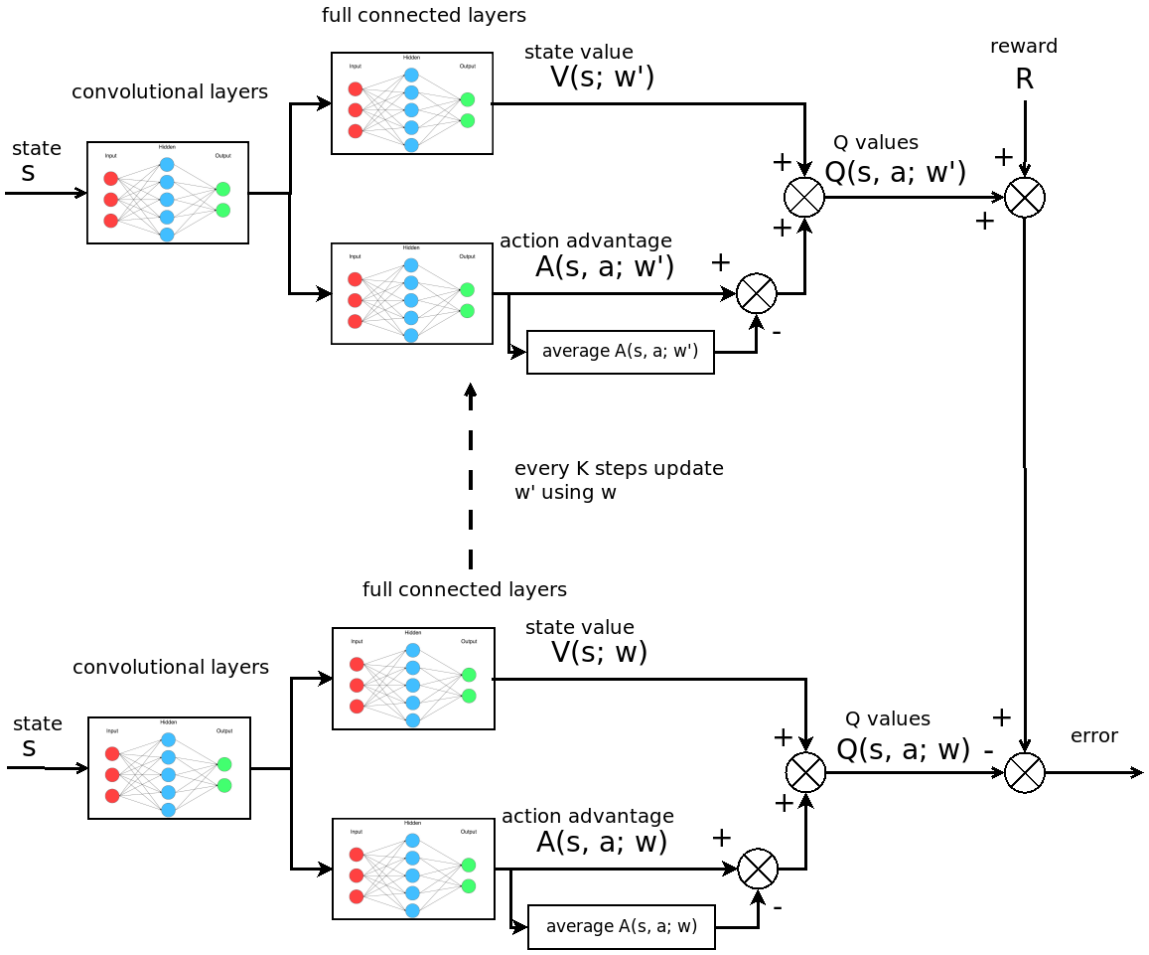
to avoid identifiability we substract average value of A truth all actions

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



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

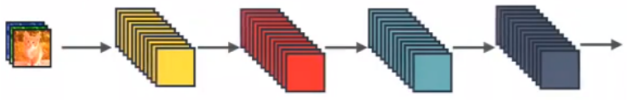
$$\hat{Q}(s, a; w) = R + \gamma \left(\hat{V}(s'; w') + \max_{\alpha'} \hat{A}(s', \alpha'; w') - \frac{1}{N_{\alpha'}} \sum_{\alpha'} \hat{A}(s', \alpha'; w') \right)$$

and finally the weights update rule

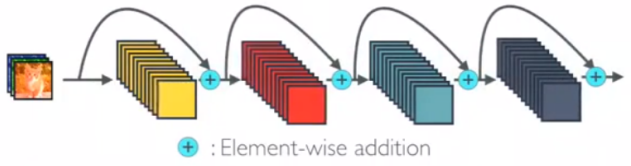
$$\Delta w = \eta \left(R + \gamma \left(\hat{V}(s'; w') + \max_{\alpha'} \hat{A}(s', \alpha'; w') - \frac{1}{N_{\alpha'}} \sum_{\alpha'} \hat{A}(s', \alpha'; w') \right) - \hat{Q}(s, a; w) \right) \nabla_w \hat{Q}(s, a; w)$$

DenseNet

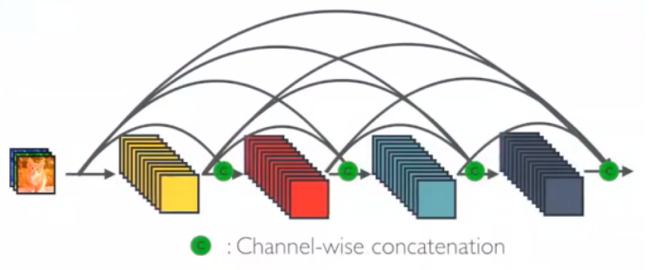
- CNN



- ResNET

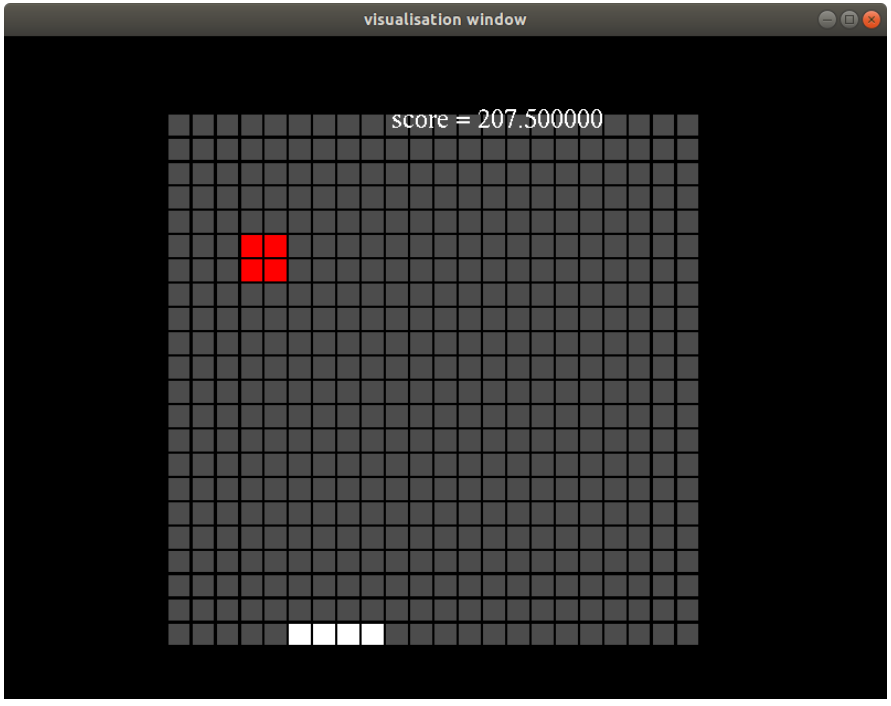


- DenseNet



layer type	input dimensions	kernel dimensions	output dimensions
dense convolution	256x256x4	3x3x8	256x256x12
dense convolution	256x256x12	3x3x8	256x256x20
dense convolution	256x256x20	3x3x8	256x256x28
dense convolution	256x256x28	3x3x8	256x256x36
convolution	256x256x36	1x1x16	256x256x16
pooling	256x256x16	2x2	128x128x16

Experiments



Network hyperparameters

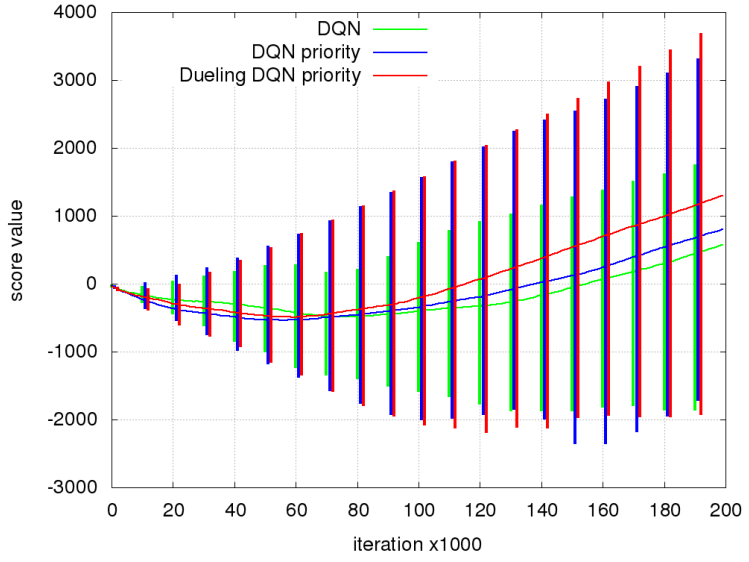
- init weights : XAVIER
- learning rate : 0.0005
- L1 regularization : 0.0000001
- L2 regularization : 0.0000001
- dropout : 0.2
- minibatch size : 32

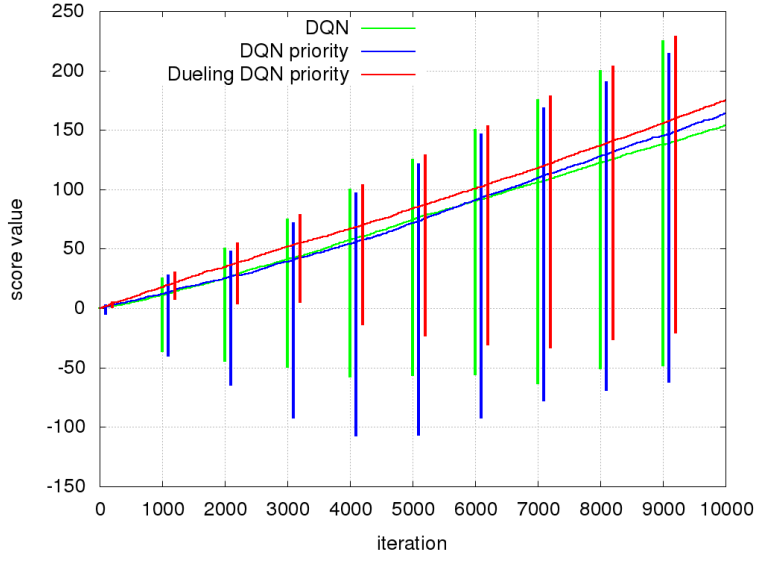
RL hyperparameters

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

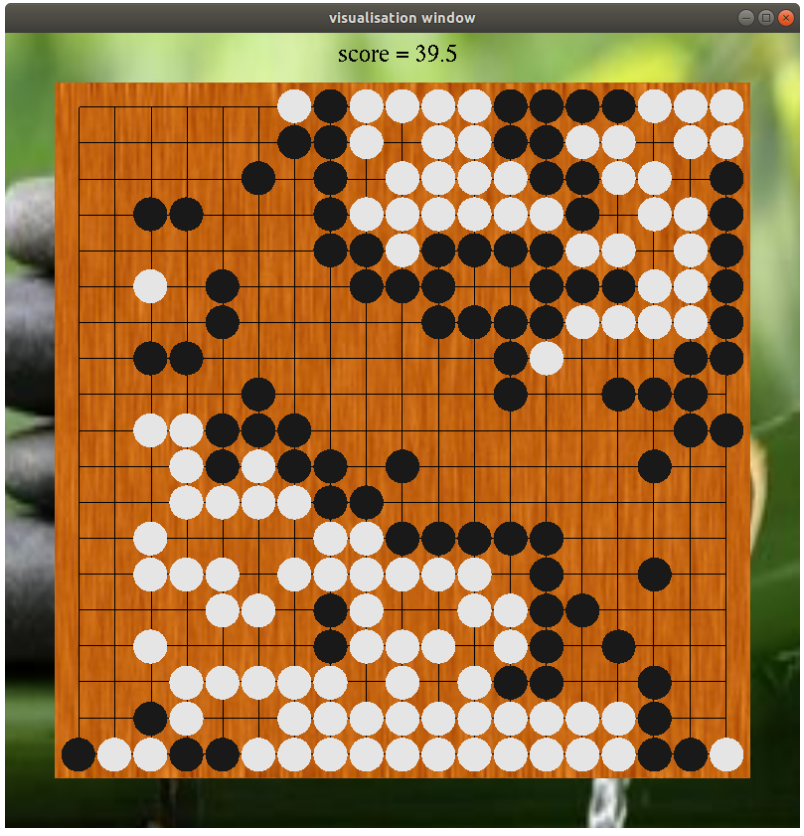
layer type	input dimensions	kernel dimensions
dense convolution	22x22x1	3x3x8
dense convolution	22x22x9	3x3x8
dense convolution	22x22x17	3x3x8
dense convolution	22x22x25	3x3x8
convolution	22x22x33	3x3x32
full connected	22x22x32	actions_count

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)

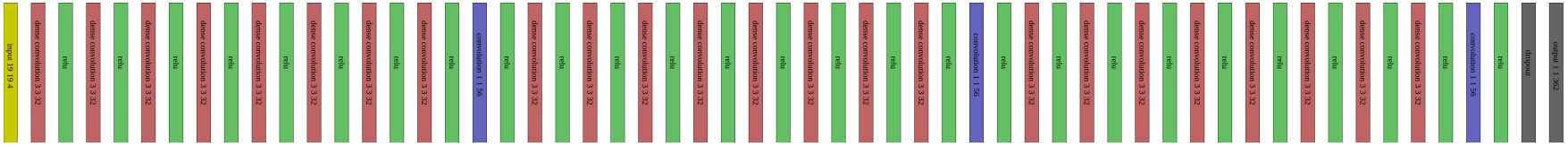


- **supervised training** - train game using Masters games
- **reinforcement learning** - let play two networks against each other

Network architecture

we need to go much deeper for GO

- **27 convolutional layers**
  3 blocks with 8 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	net 3	net 5
0	dense conv 5x5x32	dense conv 3x3x32
1	dense conv 5x5x32	dense conv 3x3x32
2	dense conv 5x5x32	dense conv 3x3x32
3	dense conv 5x5x32	dense conv 3x3x32
4	conv 1x1x32	dense conv 3x3x32
5	dense conv 5x5x32	dense conv 3x3x32
6	dense conv 5x5x32	dense conv 3x3x32
7	dense conv 5x5x32	dense conv 3x3x32
8	dense conv 5x5x32	conv 1x1x56
9	conv 1x1x32	dense conv 3x3x32
10	dense conv 5x5x32	dense conv 3x3x32
11	dense conv 5x5x32	dense conv 3x3x32
12	dense conv 5x5x32	dense conv 3x3x32
13	dense conv 5x5x32	dense conv 3x3x32
14	conv 1x1x32	dense conv 3x3x32
15	dense conv 5x5x32	dense conv 3x3x32
16	dense conv 5x5x32	dense conv 3x3x32
17	dense conv 5x5x32	conv 1x1x56
18	dense conv 5x5x32	dense conv 3x3x32
19	conv 5x5x64	dense conv 3x3x32
20	fc 362	dense conv 3x3x32
21		dense conv 3x3x32
22		dense conv 3x3x32
23		dense conv 3x3x32
24		dense conv 3x3x32
25		dense conv 3x3x32
26		conv 1x1x64
27		fc 362

Supervised results

89	60	56	61	62	60	61	60	57	66	67	62	64	60	61	57	55	40	82
46	55	60	66	59	54	56	60	61	51	52	55	56	63	53	60	59	58	38
59	52	62	81	55	60	49	49	52	54	48	49	52	64	70	64	63	62	56
61	57	66	83	62	55	50	53	47	41	46	48	49	56	57	63	51	60	52
70	55	68	57	55	53	56	56	54	55	54	53	57	53	53	60	51	57	62
53	57	62	62	55	53	56	59	50	60	49	57	49	54	60	60	65	55	58
69	51	59	57	55	49	55	52	55	48	51	56	51	52	53	58	47	55	57
66	52	55	58	58	54	54	51	56	54	53	56	53	58	56	51	55	60	48
68	62	51	50	56	55	51	55	58	57	57	55	54	55	57	48	55	56	51
61	53	57	48	51	57	57	54	56	60	59	55	56	57	57	46	59	55	61
66	60	52	51	54	55	53	52	53	53	55	55	56	53	59	51	52	52	61
67	59	54	55	52	57	52	58	59	56	56	59	56	48	56	52	52	55	61
67	52	50	55	57	54	56	55	51	53	53	56	56	54	57	64	59	56	54
61	56	68	65	53	52	52	59	55	50	58	57	57	56	56	62	70	61	58
63	57	67	63	58	53	53	55	55	62	59	54	51	51	54	63	60	52	62
56	60	69	82	60	60	55	55	49	38	53	54	54	55	52	73	78	60	58
61	63	65	76	58	70	55	55	50	57	50	51	51	62	69	78	65	66	57
69	58	65	71	57	59	59	56	56	52	57	58	58	57	60	65	57	63	50
90	55	69	57	64	58	56	59	58	69	56	55	67	57	52	65	58	58	86
																		0

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`