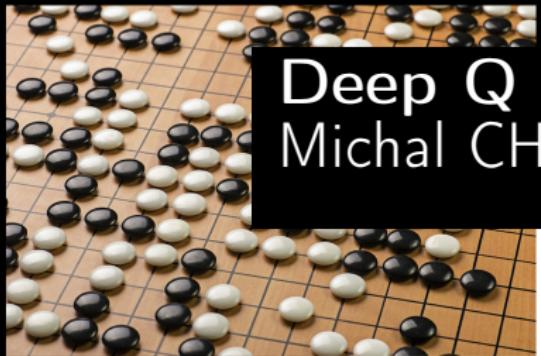


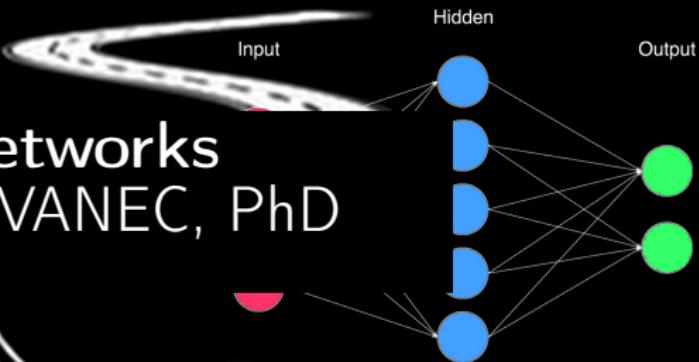
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

(The New Action Value = The Old Value) + The Learning Rate  $\times$  (The New Information — the Old Information)



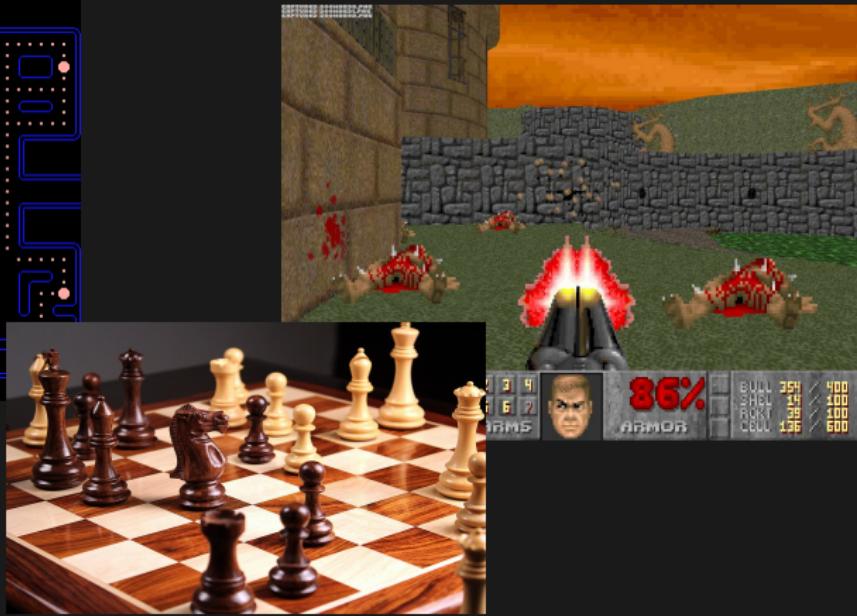
# Deep Q networks

Michal CHOVANEC, PhD



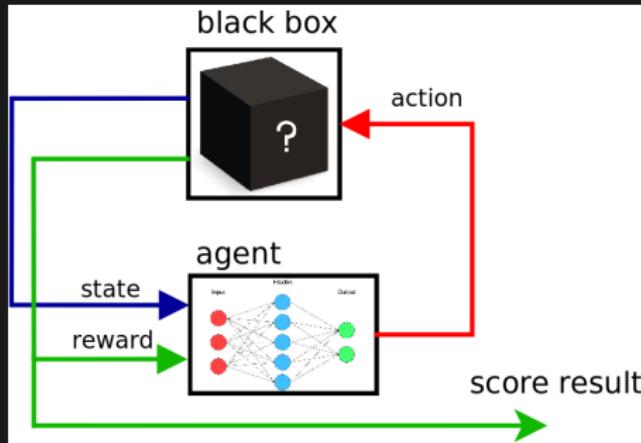
# Reinforcement learning

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



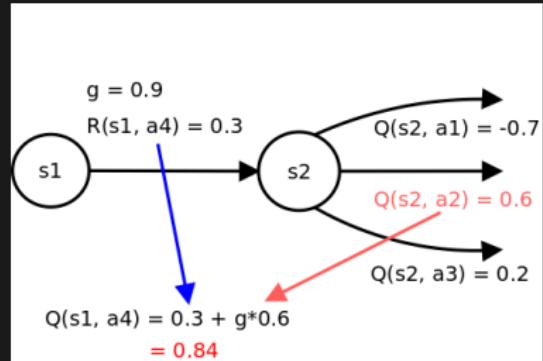
# Reinforcement learning

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



# Q learning (1989)

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



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

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

weights gradient

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

# Deep Q network - DQN

- **unstable training :**

non-stationary target value  $\hat{Q}(s, a; w)$ , depends on  $w$

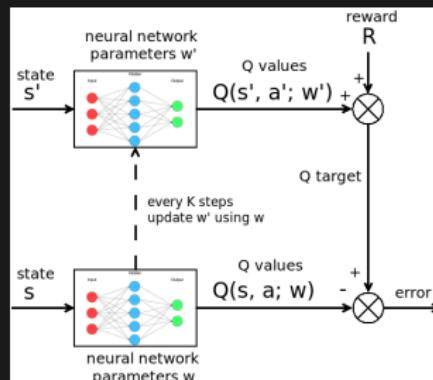
- **solution - fix  $w$  :**

using temporary network with  $w'$  weights.

## DQN equation

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

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



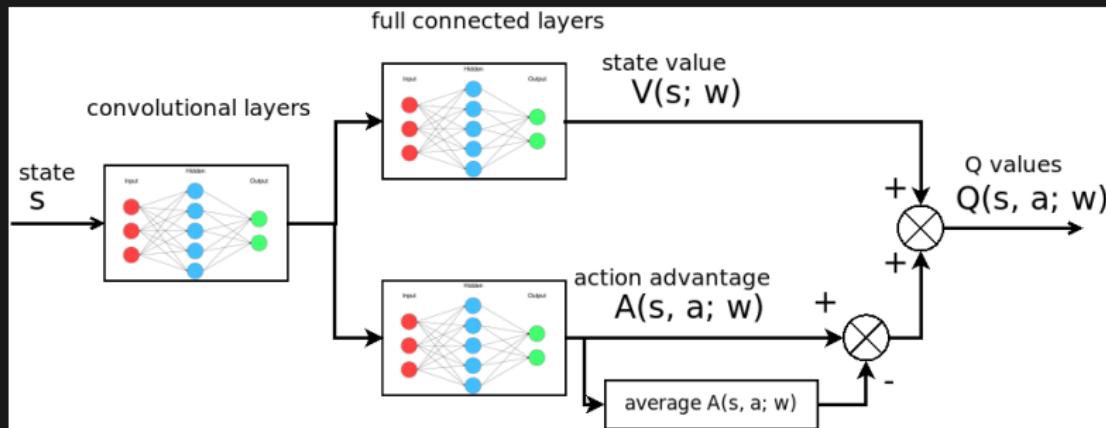
# Dueling deep Q network - DDQN (2016)

$$\hat{Q}(s, a, w) = \hat{V}(s, w) + \hat{A}(s, a, w)$$

**value for being in state s**      **advantage of taking action a at state s**

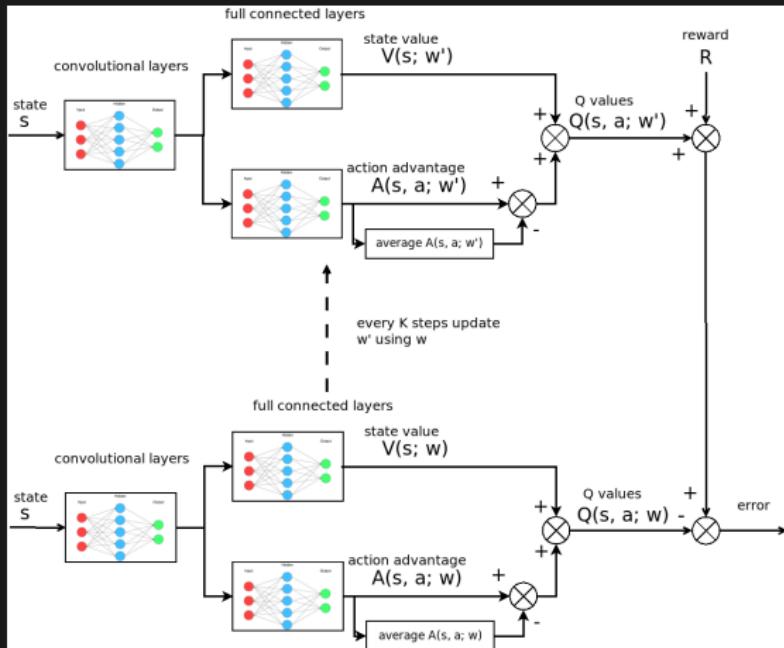
to avoid identifiability we subtract average value of A truth all actions

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



# 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

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

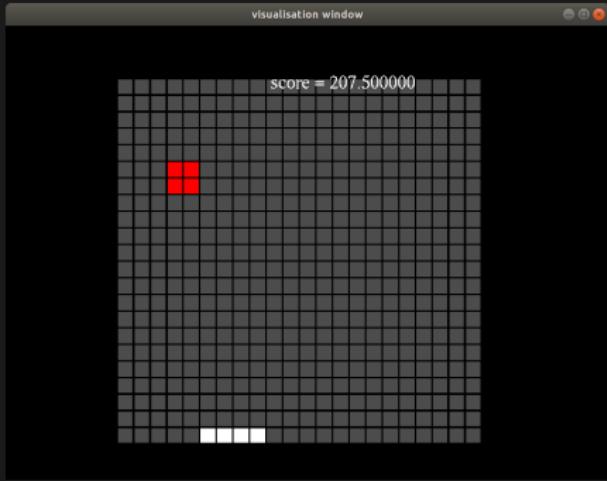
we obtain dueling deep Q network equation

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

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

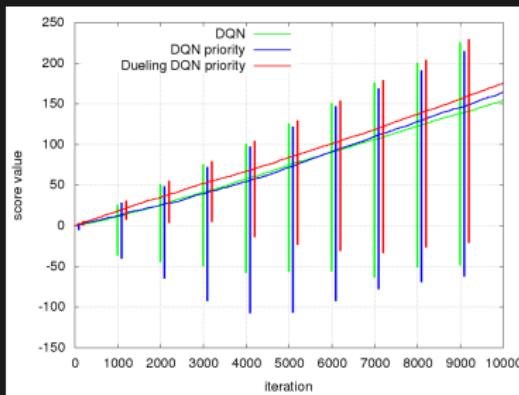
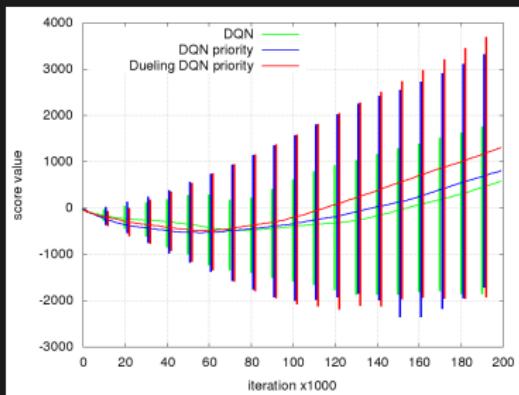
- init weights : XAVIER
- learning rate : 0.0005
- L1 regularization : 0.0000001
- L2 regularization : 0.0000001
- dropout : 0.2
- minibatch size : 32

## RL hyperparameters

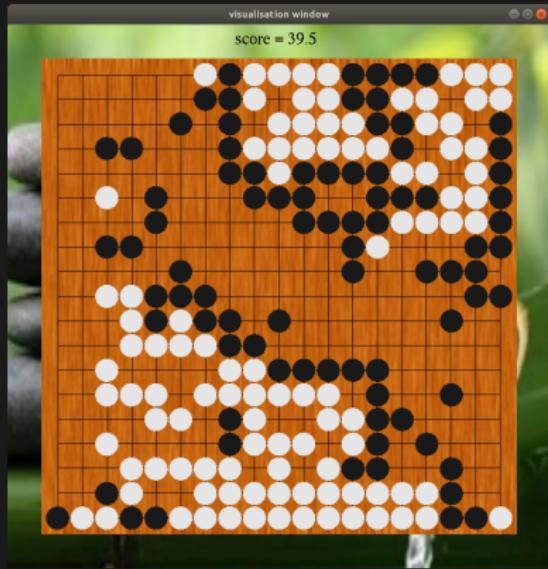
- experience buffer size : 1024
- gamma : 0.95
- epsilon training : 0.1
- epsilon testing : 0.05

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



# Playing GO

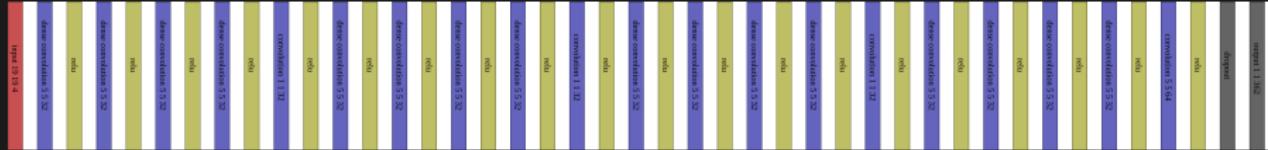


- **supervised training** - train game using Masters games
- **reinforcement learning** - let play two networks against each other

# Network architecture

we need 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

FD6864 34CDF9

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

77	33	40	45	43	43	49	48	50	54	57	50	56	47	40	40	31	2	88
27	45	52	56	50	49	46	53	54	45	47	53	54	57	52	59	54	3	0
34	44	53	80	54	58	45	45	45	50	42	44	47	62	64	62	61	17	3
37	48	64	83	55	51	46	52	47	40	43	42	48	55	50	64	51	4	3
50	44	64	61	50	43	46	55	54	52	50	49	50	47	49	61	48	2	7
40	46	56	57	45	45	46	50	46	54	47	50	43	45	51	58	57	3	12
61	48	50	43	47	43	48	48	47	47	50	53	49	46	47	54	45	5	16
50	43	44	51	53	47	54	45	55	51	49	50	47	43	54	50	42	3	18
57	56	39	47	47	48	46	49	48	52	50	58	53	51	54	43	52	3	19
51	51	44	42	47	53	49	48	51	53	54	54	48	53	54	39	42	2	22
53	50	43	45	45	46	47	48	51	49	52	52	50	48	55	44	46	3	32
52	51	47	47	48	48	47	49	53	50	51	51	49	49	52	44	47	4	21
46	46	43	51	50	42	50	48	45	46	51	49	51	54	49	59	51	7	10
42	48	65	54	43	43	43	47	45	44	50	54	49	47	52	58	53	4	12
45	44	63	57	50	45	45	45	47	51	52	48	47	41	48	57	58	5	15
45	52	65	81	53	58	45	53	48	32	51	49	45	55	47	72	72	5	3
46	57	61	70	50	62	38	44	42	46	39	36	39	46	58	72	59	18	2
35	2	10	2	2	1	0	4	1	2	1	4	1	1	3	0	13	3	8
86	1	1	1	14	15	15	12	21	34	16	24	24	29	11	4	8	3	81
																		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>
-  **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>
-  **Mastering the Game of Go without Human Knowledge**  
[https://deepmind.com/documents/119/agz\\_unformatted\\_nature.pdf](https://deepmind.com/documents/119/agz_unformatted_nature.pdf)

# Q&A



michal chovanec (michal.nand@gmail.com)

[www.youtube.com/channel/UCzVvP2ou8v3afNiVrPAHQGg](https://www.youtube.com/channel/UCzVvP2ou8v3afNiVrPAHQGg)

github <https://github.com/michalnand>