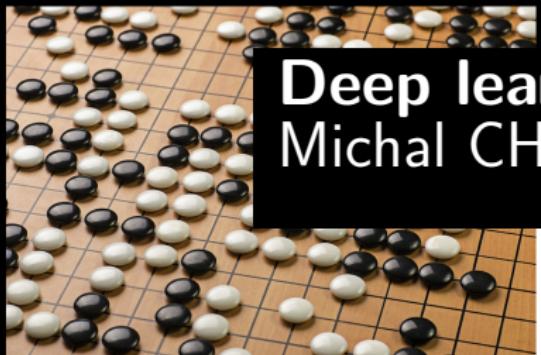


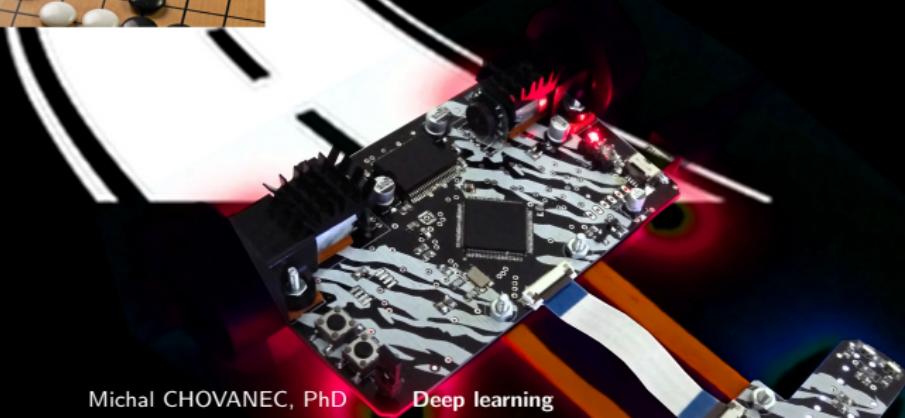
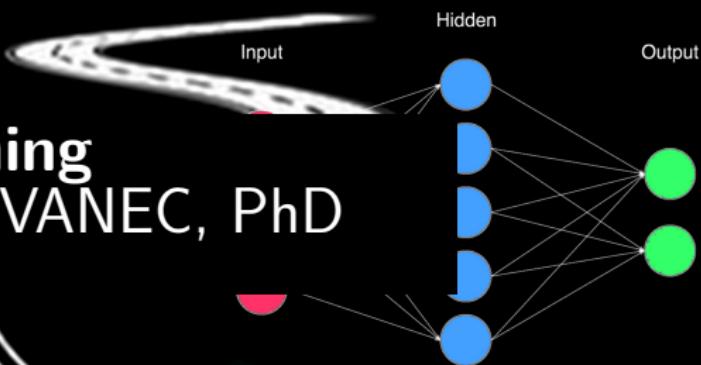
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

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



Deep learning

Michal CHOVANEC, PhD



Applications

- self driving cars - Tesla model S
- healthcare, biomedical engineering - Cell in fluid
- voice search, control - Google, Amazon Echo
- machine translation - Google translator
- image recognition - Huawei Mate 10
- game bots, robotics - DeepMind, Boston Dynamics



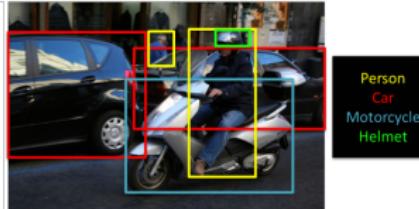
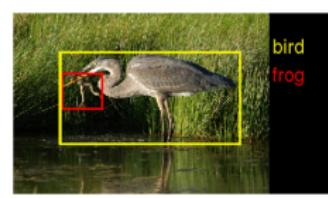
Speech Recognition



Reduced word errors by more than 30%

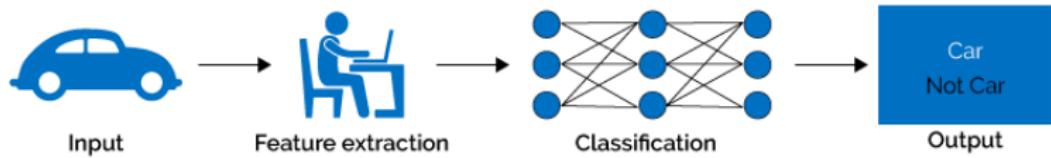
Google Research Blog - August 2012, August 2015

Research at Google

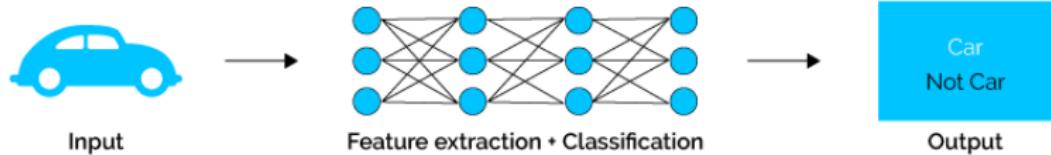


Deep learning¹

Machine Learning



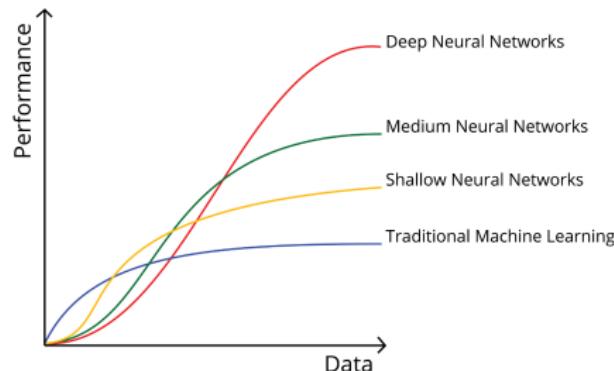
Deep Learning



¹<https://medium.com/datamob/clearing-the-buzzwords-in-machine-learning-e395ad73178b>

Deep learning

- classification
- object detection
- regression
- unsupervised features extraction
- reinforcement learning



Milestones

- Specters of past
 - vanishing gradient, **solution** : ReLU, Inception, Resnet, DenseNet
 - overfitting, **solution** : Xavier, L1, L2 regularization, dropout, batch normalisation
 - computation cost, **solution** : NVIDIA Cuda, sharing weights
- ImageNet Classification with Deep Convolutional Neural Networks ²
 - 2012, Alex Krizhevsky, 34887 citations
- Adam
 - 2014, Method for Stochastic Optimization ³
 - 2014, VGG net, 3x3 kernels instead of 5x5 ⁴
- AlphaGO, AlphaZero, AlphaStar ⁵
 - march 2016, december 2017, januray 2019

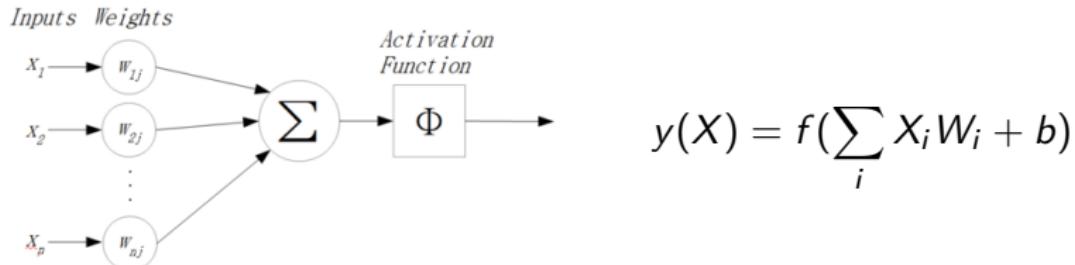
²<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

³<https://arxiv.org/abs/1412.6980>

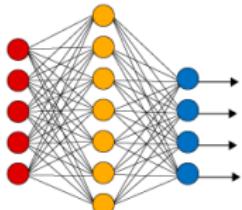
⁴<https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvrc-2014-image-classification-55333a2a2a2c>

⁵<https://deepmind.com/>

Neural network



Simple Neural Network

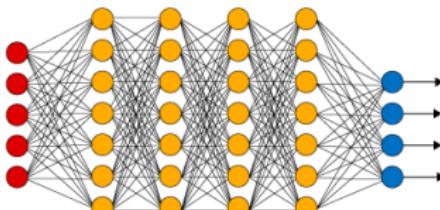


● Input Layer

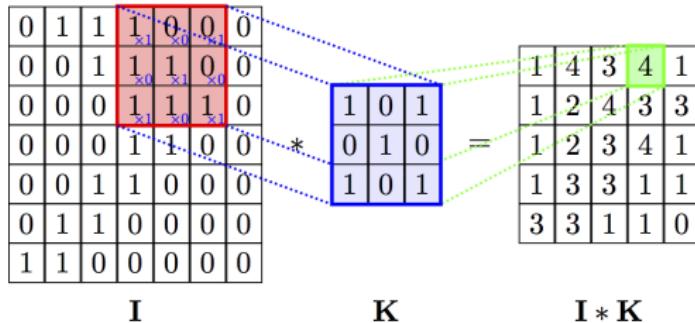
● Hidden Layer

● Output Layer

Deep Learning Neural Network



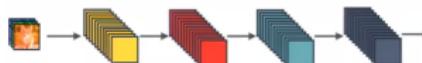
Convolutional layer



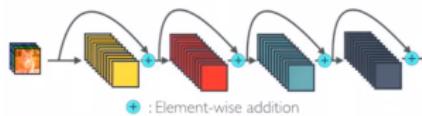
$$Y_{yxI} = \sum_k^{in_depth} \sum_j^{kh} \sum_i^{kw} X_{k,j+y,i+x} W_{lkji} + b_I$$

Deep neural network

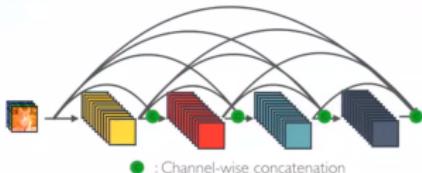
- Old methods (before NN) - 2011, 25.8%
- Convolutional, AlexNet - 2012, 16.4%
- Google inception - 2013, 6.7%
- Microsoft ResNet - 2015, 6.1%
- DenseNet - 2018, 5.17%
- CNN



- ResNET



- DenseNet



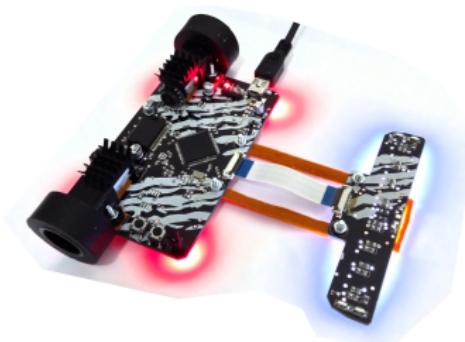
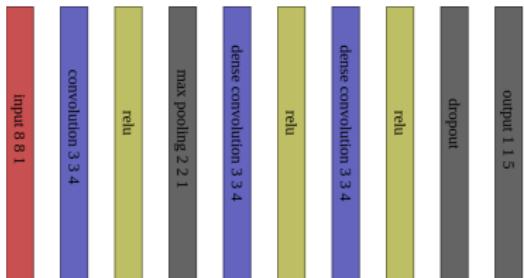
My research

- Robotics - hobby
- Red blood cells trajectory prediction
- Deep reinforcement learning

Robotics - line follower

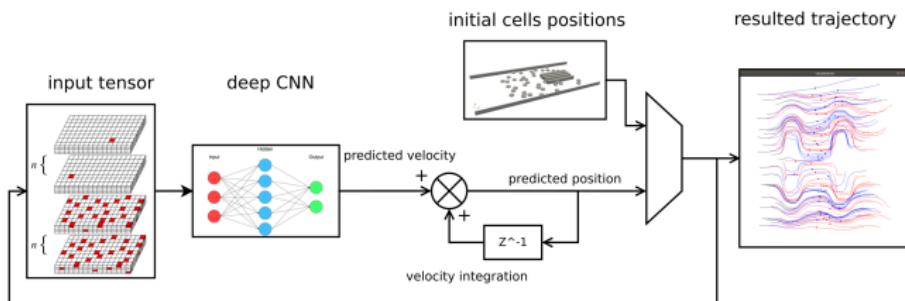
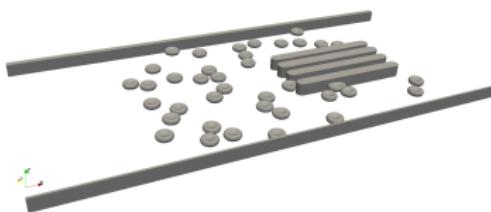
Curve shape classification

- stm32f303 (72MHz),
stm32f746 (216MHz)
- 8x 500nm line sensors
- pololu motors, 1:30
- network input : 8 last line
sensors results (8x8 matrix)
- response 4..5ms



Red blood cells trajectory prediction

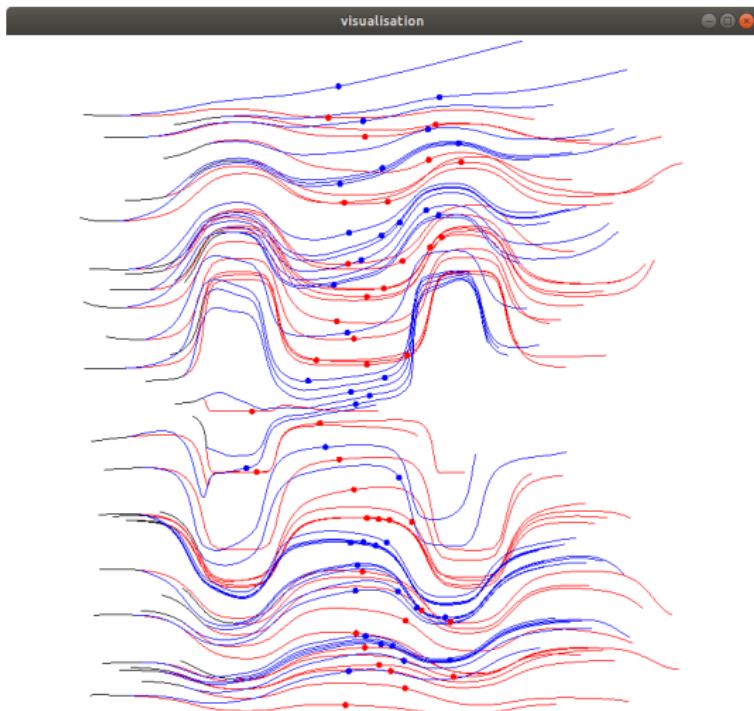
- train DNN to predict RBC trajectory from past
- 15 conv layers network (6hours training on GTX1080ti)
- input : RBC position + 7 past frames + other cells position
- output: RBC predicted velocity



Networks

layer	net 0	net 1	net 2	net 3	net 4	net 5	net 6	net 7
0	fc 256	conv 3x3x32	dense conv 3x3x8					
1	fc 64	fc 64	dense conv 3x3x8					
2	fc 32	fc 32	dense conv 3x3x8					
3	fc 3	fc 3	dense conv 3x3x8					
4			conv 1x1x32	conv 1x1x16	conv 1x1x32	conv 1x1x16	conv 1x1x16	conv 1x1x32
5			fc 3	dense conv 3x3x8	fc 3	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
6				dense conv 3x3x8		dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
7				dense conv 3x3x8		dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
8				dense conv 3x3x8		dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
9				conv 1x1x32		conv 1x1x32	conv 1x1x16	conv 1x1x32
10				fc 3		fc 3	dense conv 3x3x8	dense conv 3x3x8
11							dense conv 3x3x8	dense conv 3x3x8
12							dense conv 3x3x8	dense conv 3x3x8
13							dense conv 3x3x8	dense conv 3x3x8
14							conv 1x1x32	conv 1x1x64
15							fc 3	fc 3

Results



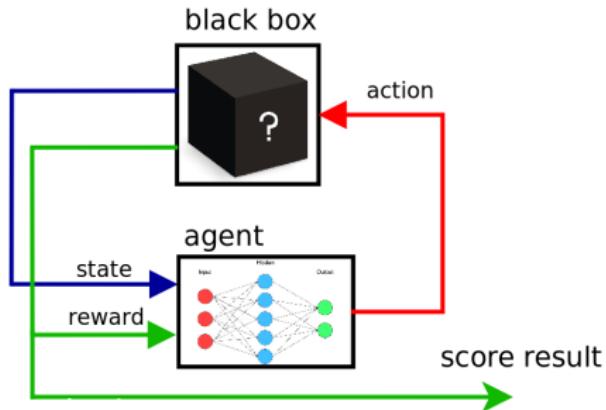
Results

ID	total error			
	mean [um]	sigma [um]	rms [um]	rms relative [%]
0	-9.8	38.88	40.1	7.23
1	-36.99	215.02	218.18	39.32
2	-3.85	23.11	23.43	4.22
3	-0.32	16.09	16.09	2.9
4	-1.26	11.44	11.51	2.07
5	-0.25	11.27	11.28	2.03
6	-0.85	12.26	12.29	2.22
7	-1.67	11.9	12.02	2.17

Reinforcement learning

- learn from punishment and rewards
- learn to play a game with unknown rules
- obtain **state**, s
- choose **action**, a
- **execute** action
- obtain **reward**, R
- learn from **experiences**, $Q(s, a) = R + \gamma \max_{\alpha'} Q(s', \alpha')$

$$Q(s, a) \approx \hat{Q}(s, a; w)$$



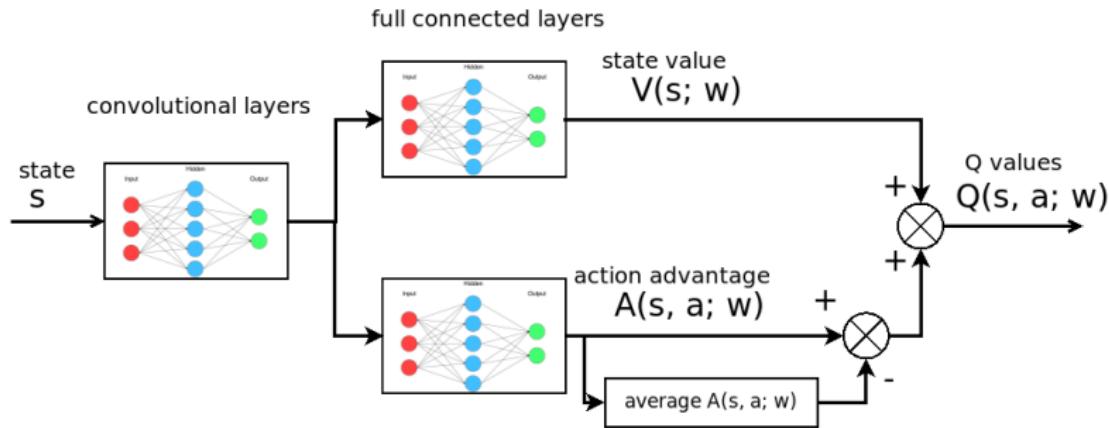
Dueling deep Q network - DDQN (2016)

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

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

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

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

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

- **28 convolutional layers**

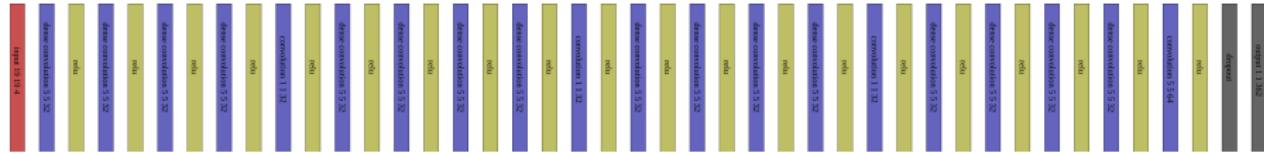
3 blocks with 8 dense conv + 1 conv layer

- **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 1x1x32
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 1x1x32
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 1x1x32
27		conv 3x3x64
28		fc 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/r1/>
-  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>