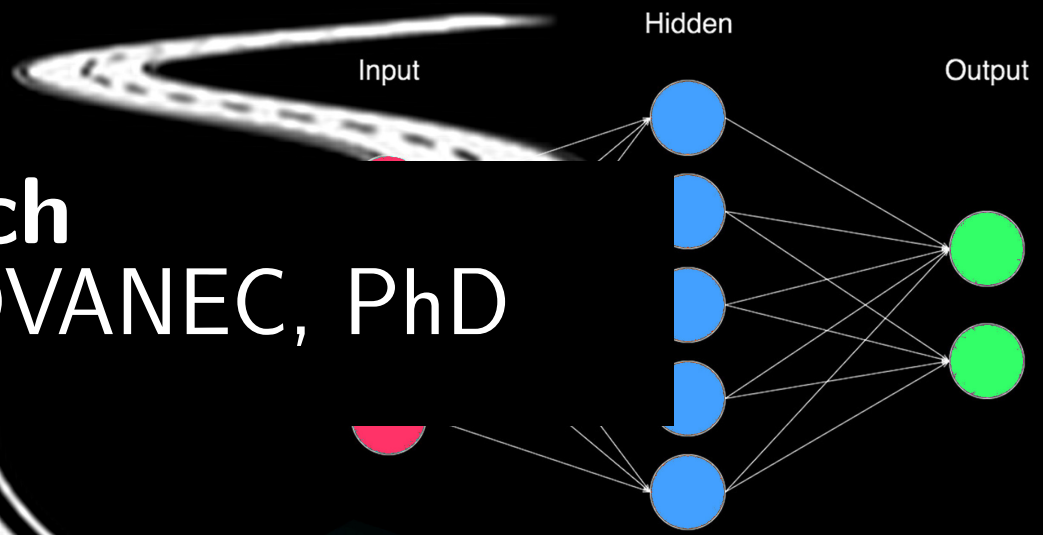
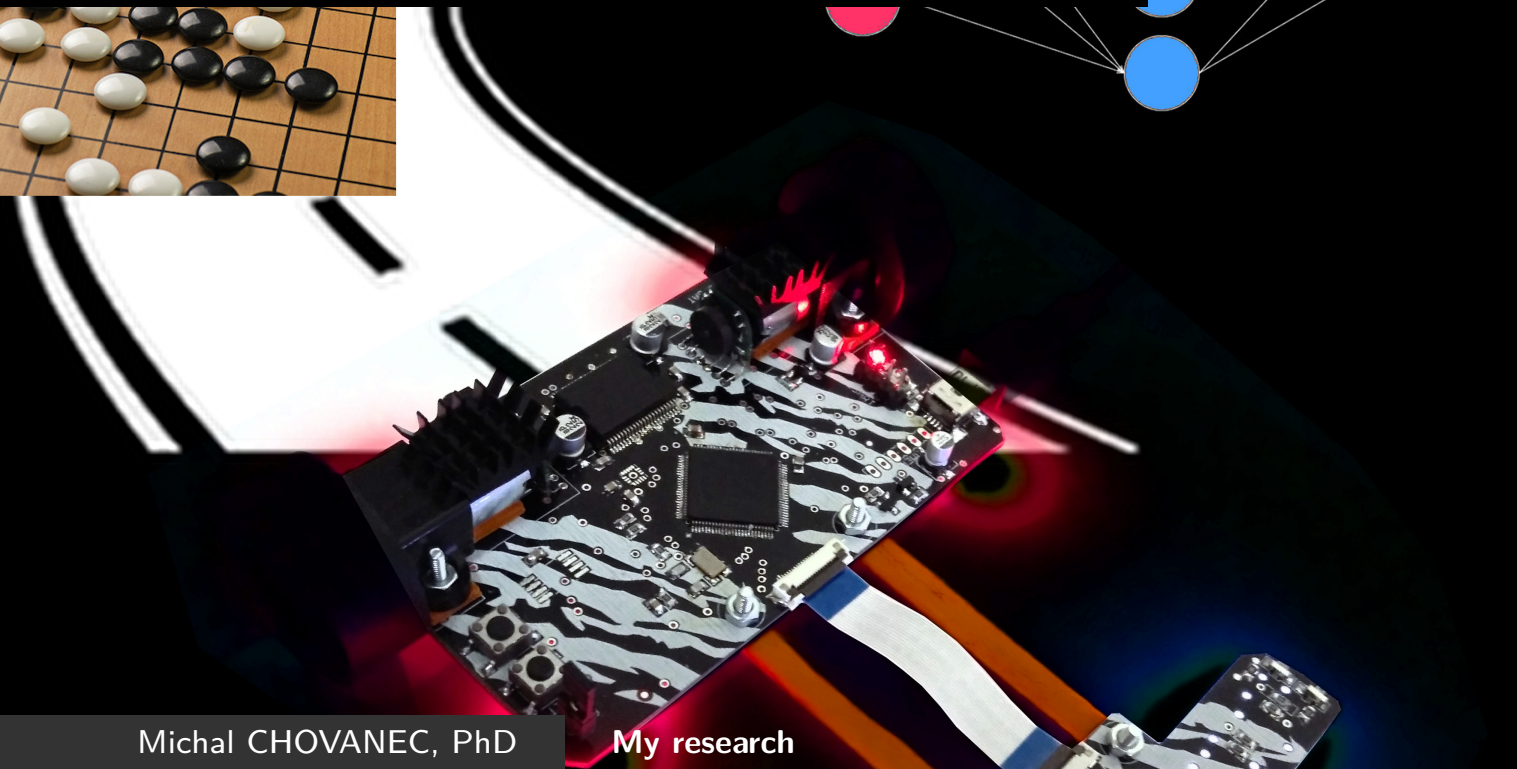
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

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



My research

Michal CHOVANEC, PhD



Overview

my GitHub `https://github.com/michalnand/`

- Robotics (hobby)
- Red blood cells trajectory prediction
- Reinforcement learning
- education (Learning systems - methods and applications, 5II140)

Rysy - my own CNN framework written from scratch

`https://github.com/michalnand/rysy`

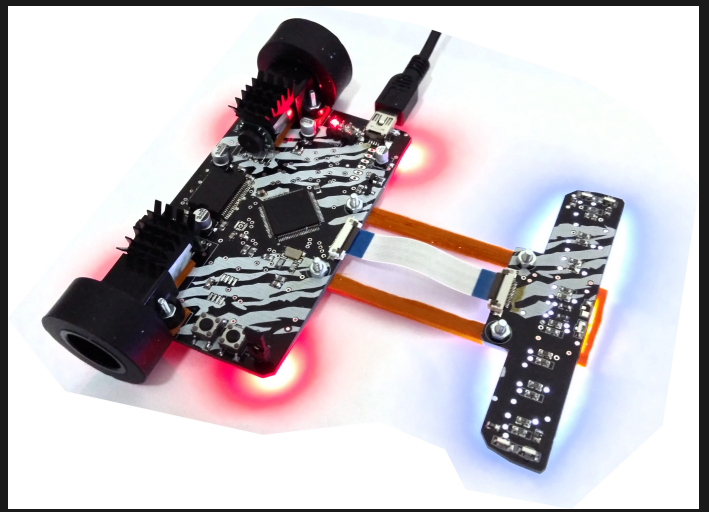
- technologies C++17, Cuda, Python

  - and many others : openCV, openGL, MPI, JsonCPP, cIMG, Swig, numpy, scipy, gym, ARM Cortex ...
- 38 000 lines of code
- CNN, DenseNet
- deep Q networks, dueling Q networks
- experiments automatization (classification, regression, RL)

Robotics - line follower

Curve shape classification - go faster on straight line

- network architecture
  IN8x8x1 - C3x3x4 - P2x2 -
  DC3x3x4 - DC3x3x4 - FC5
- **first** portable embedded DenseNet implementation
- running **more than 200FPS** on ARM Cortex M4 stm32f303 (72MHz)
- response 4..5ms
- network input : 8 last line sensors results (8x8 matrix)
- **istrobot 2016** L2 first price winner

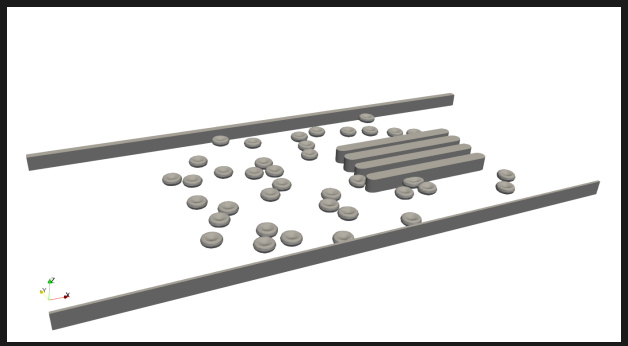


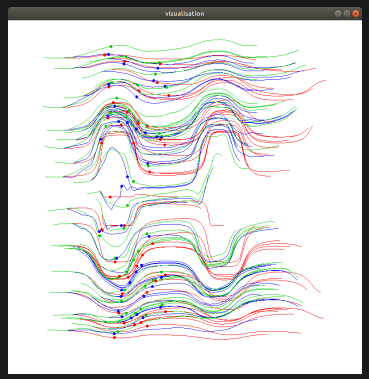
Red blood cells trajectory prediction

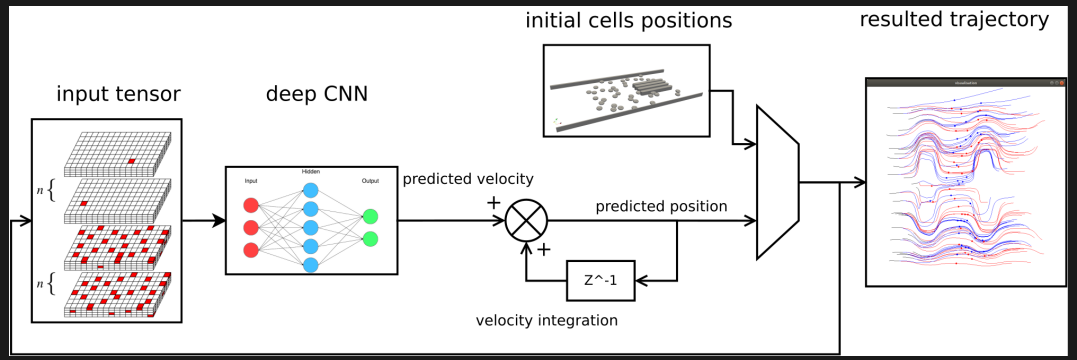
Research group **Cell in fluid**

Mgr. Katarína Jasenčáková, PhD thesis

- 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



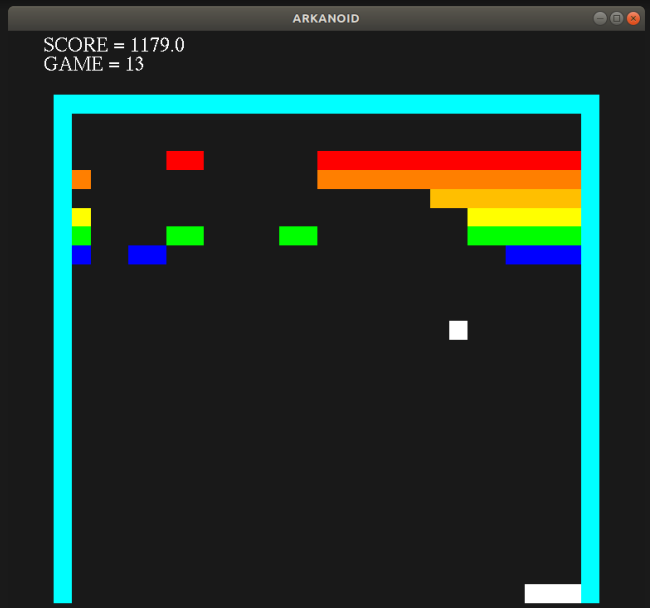


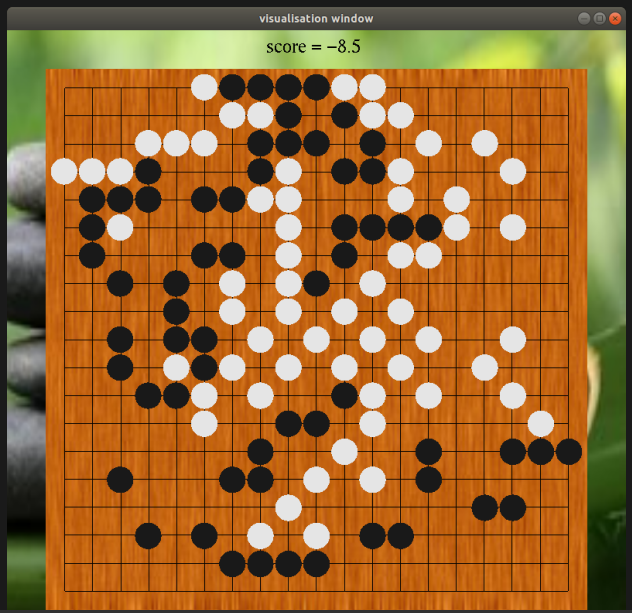


Networks

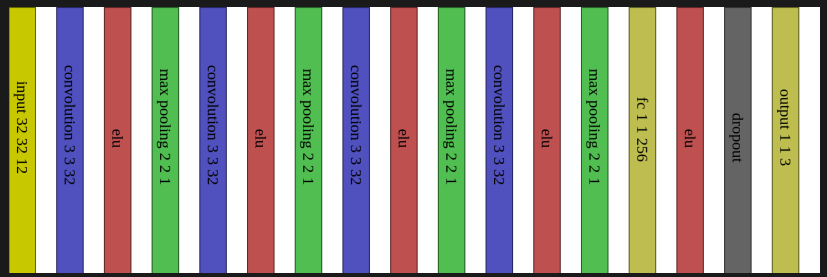
layer	net 0	net 1	net 2	net 3	net 4	net 5	net 6	net 7
0	fc 256	conv 3x3x32	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
1	fc 64	fc 64	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
2	fc 32	fc 32	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8
3	fc 3	fc 3	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	dense conv 3x3x8	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

Reinforcement learning





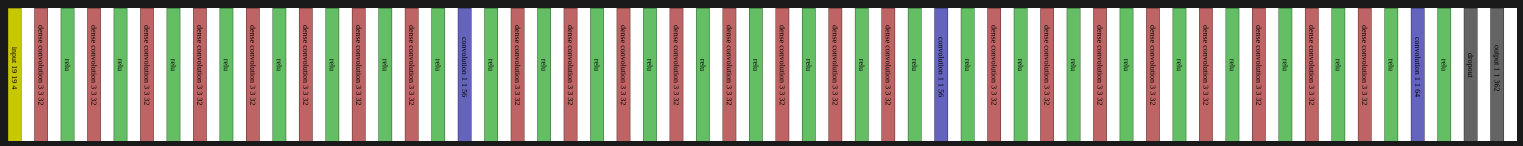
- **playing Atari**
- **playing Doom** -> scheduled on next month
- **playing GO**
  - **supervised training** - train game using Masters games
  - **reinforcement learning** - let play two networks against each other

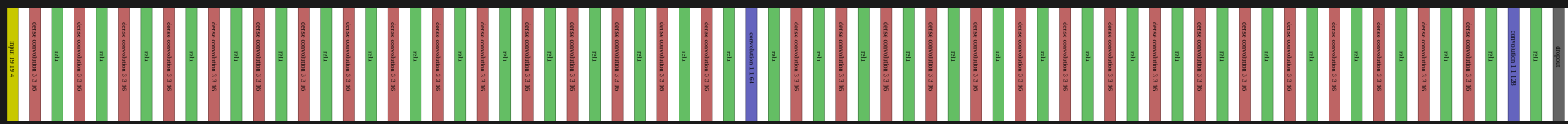


GO Network architecture

we need to go much deeper for GO

- **28, 35 layers**
  dense blocks + feature pooling layer
- **input**
  4 matrices 19x19: black stones, white stones, empty fields, active player
- **output**
  recommended moves 19x19 + 1 for pass = 362 outputs





Other stuff

- hiking, running
- archery, aikido
- caving, climbing