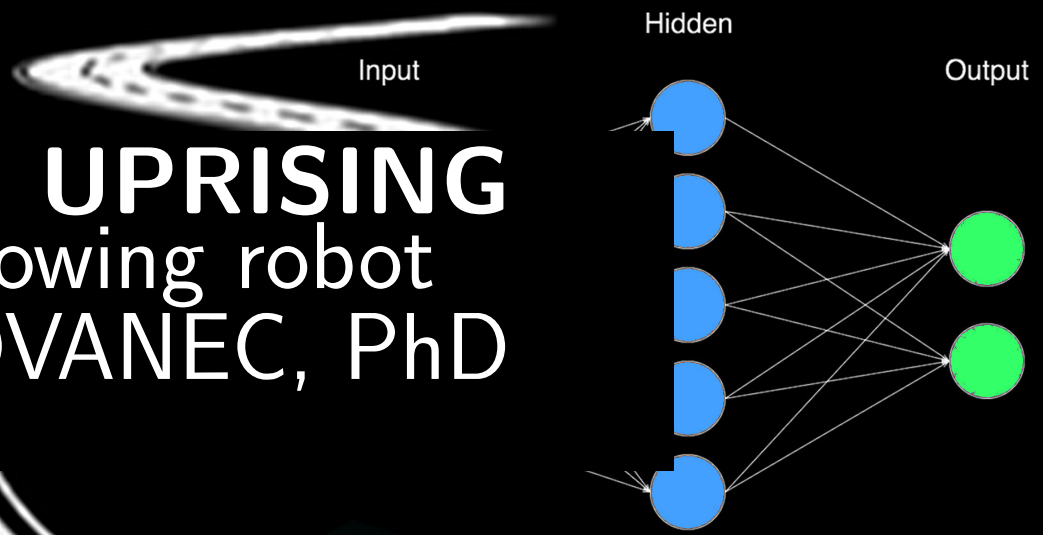
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

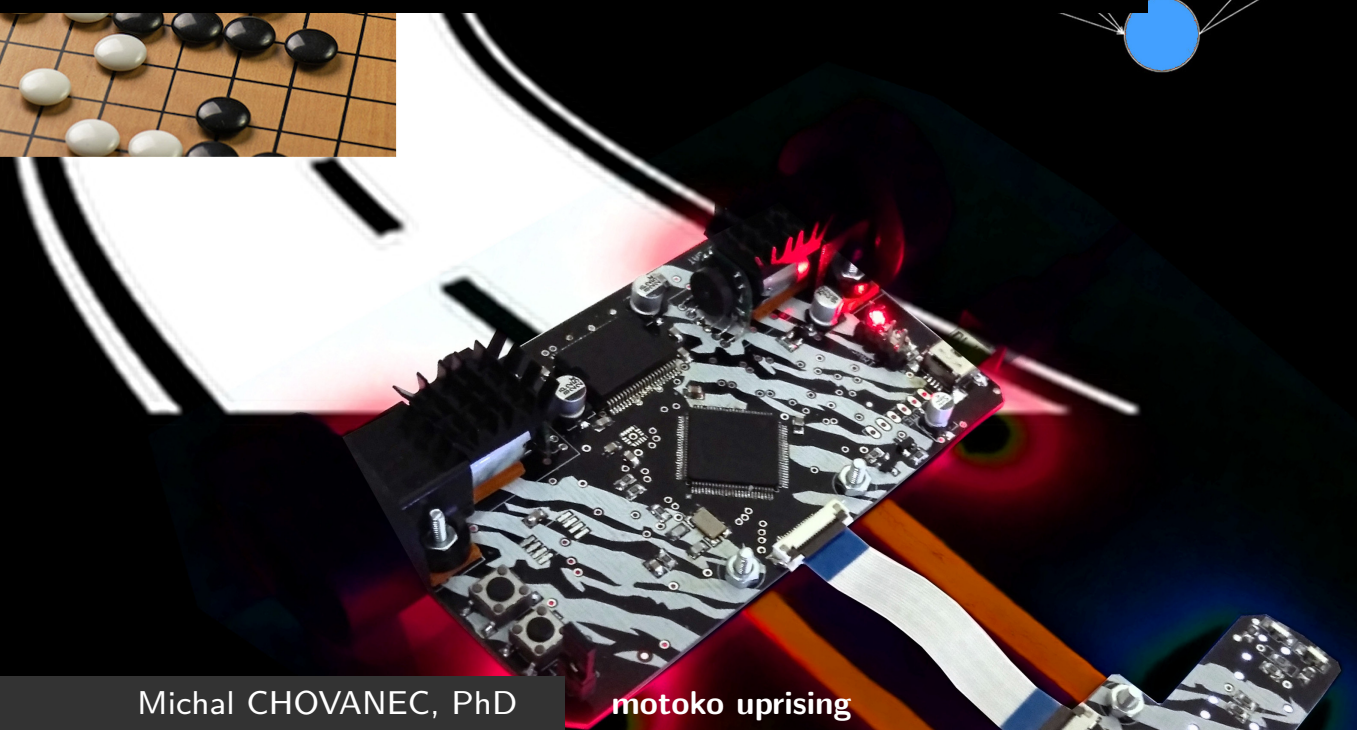
(The New Action Value = The Old Value) + The Learning Rate × (The New Information − the Old Information)

MOTOKO UPRISING

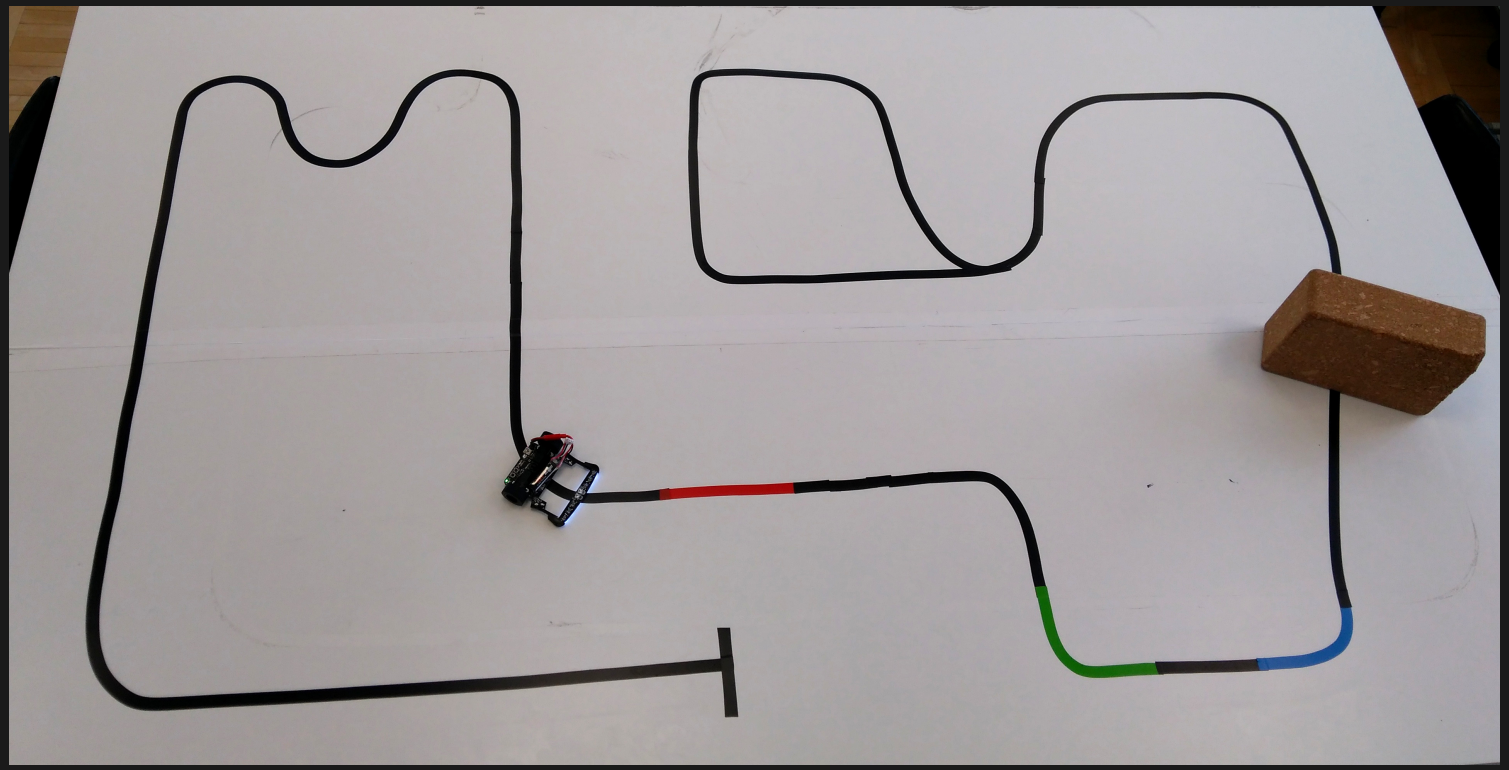
The line following robot

Michal CHOVANEC, PhD





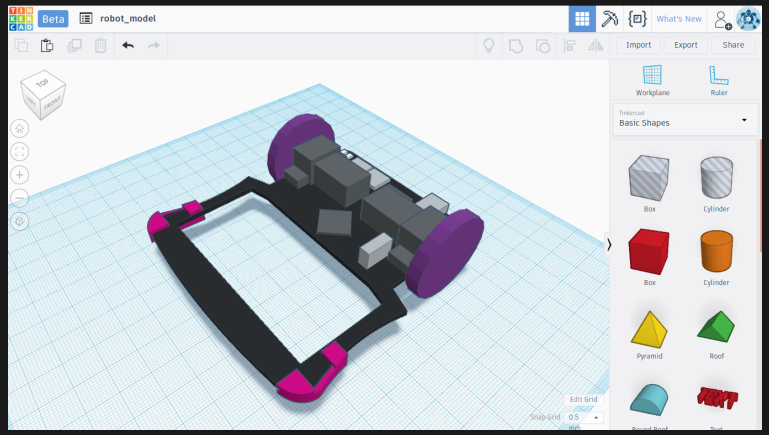
Line follower competition

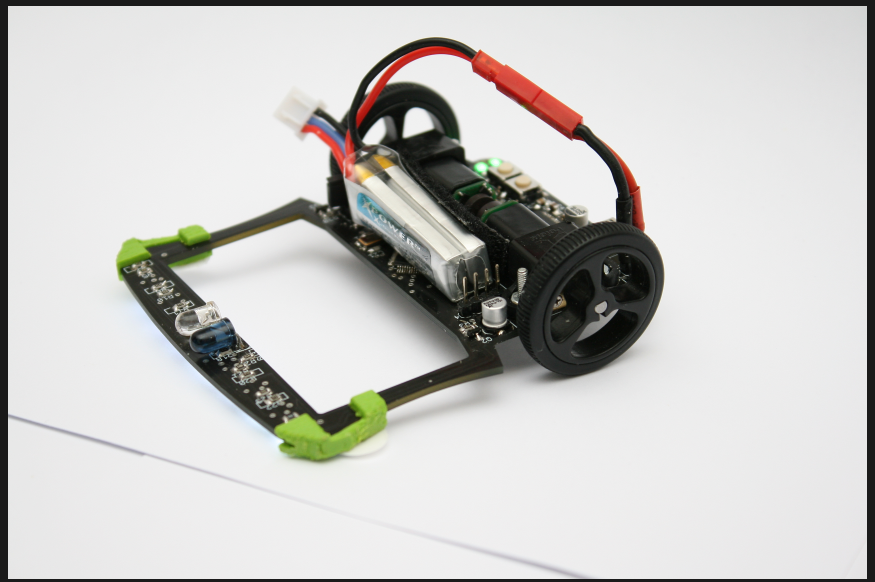


- SR - Istrobot
- CR - Roboticky den
- AU - Robot Challenge
- PL - Zawody robotow

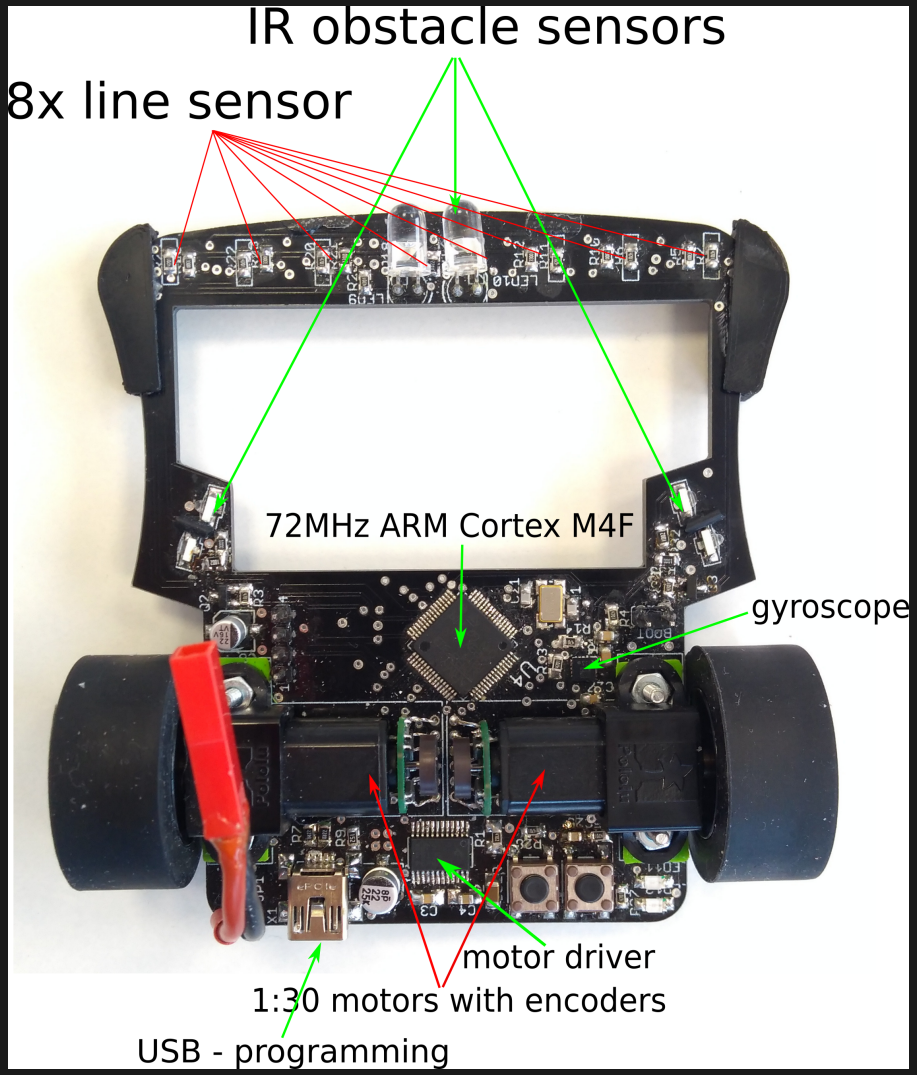
What does it take

- Hardware
  - strong light motors
  - high adhesion tyres
  - light accumulator
  - fast CPU
  - dozen of sensors
- Software
  - hard real time OS
  - tunned PIDs
  - predictive controll
  - mapping





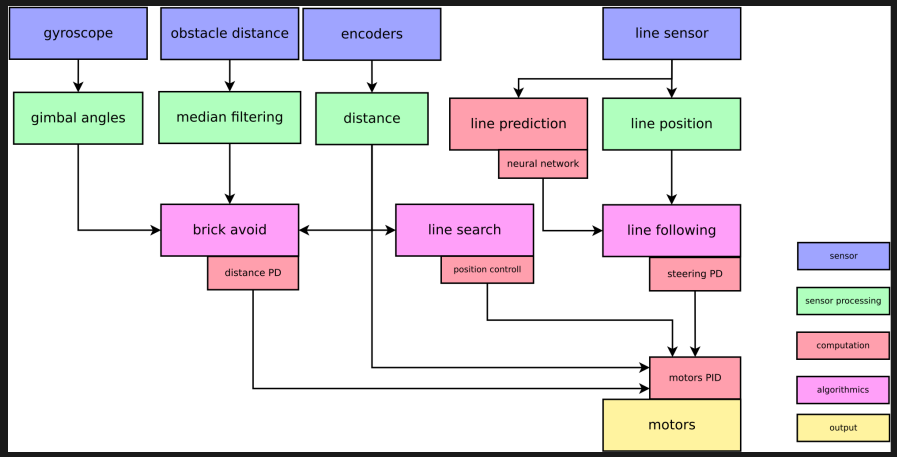
Motoko uprising - hardware



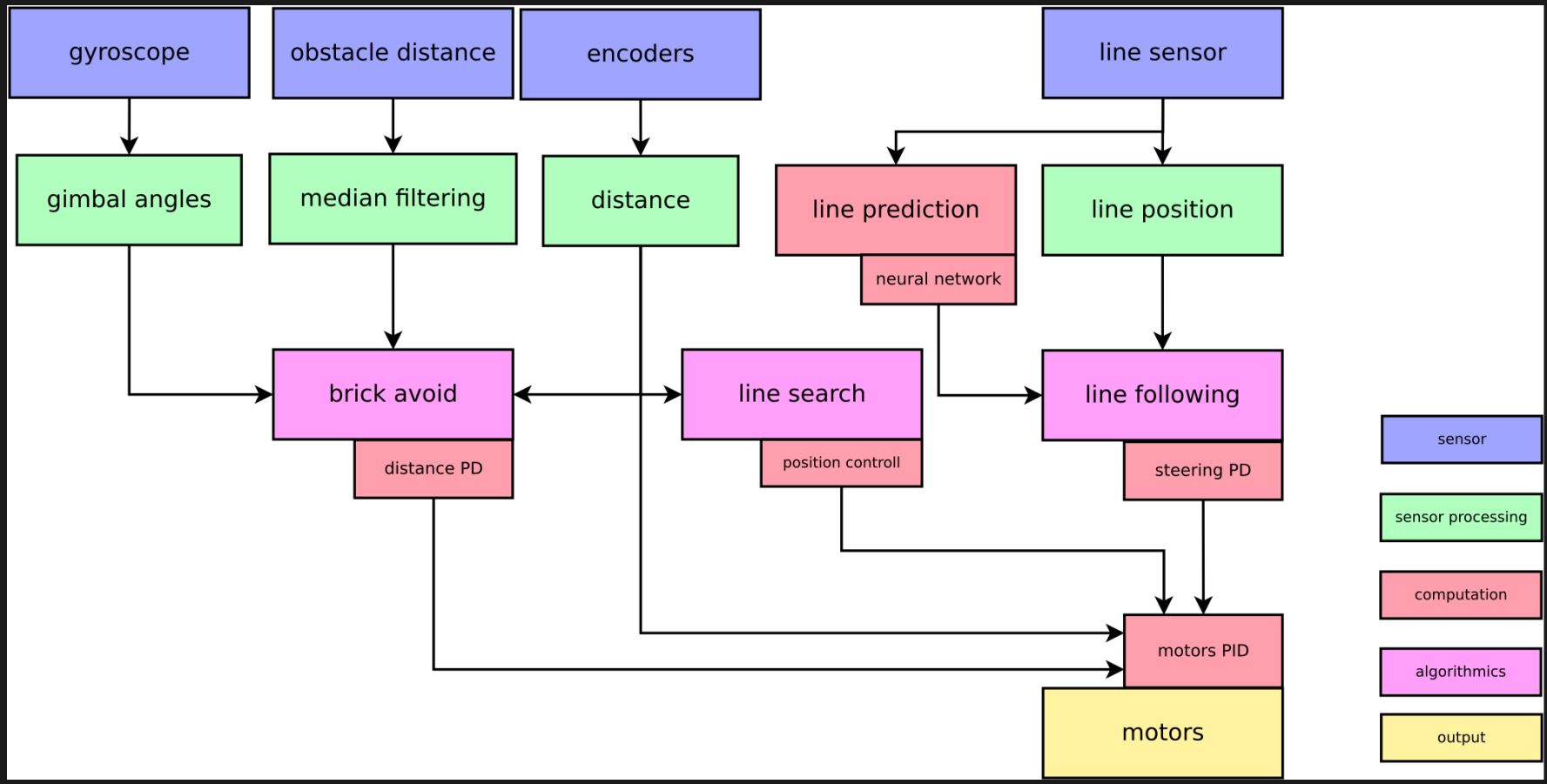
- **mcu** : STM32F303, 72MHz ARM Cortex M4F
- **motor driver** : TI DRV8834
- **motors** : 1:30 HP Pololu, with magnetic encoder
- **tyres** : Pololu 28mm diameter
- **line sensor** : 8x 540nm phototransistor + white LED
- **obstacle sensor** : SMD IR phototransistor + IR LED
- **imu** : LSM6DS0, gyroscope + accelerometer

Software

- **quadratic interpolation** : high precission line position computing
- **steering PID** : PD controller for steering
- **motor PID** : two PIDs for motors speed controll
- **curve shape prediction**
  - **fast** run on straight line, **brake** on curve
  - **deep neural network**
- written in **C++**
- **network training** on GPU - own framework for CNN

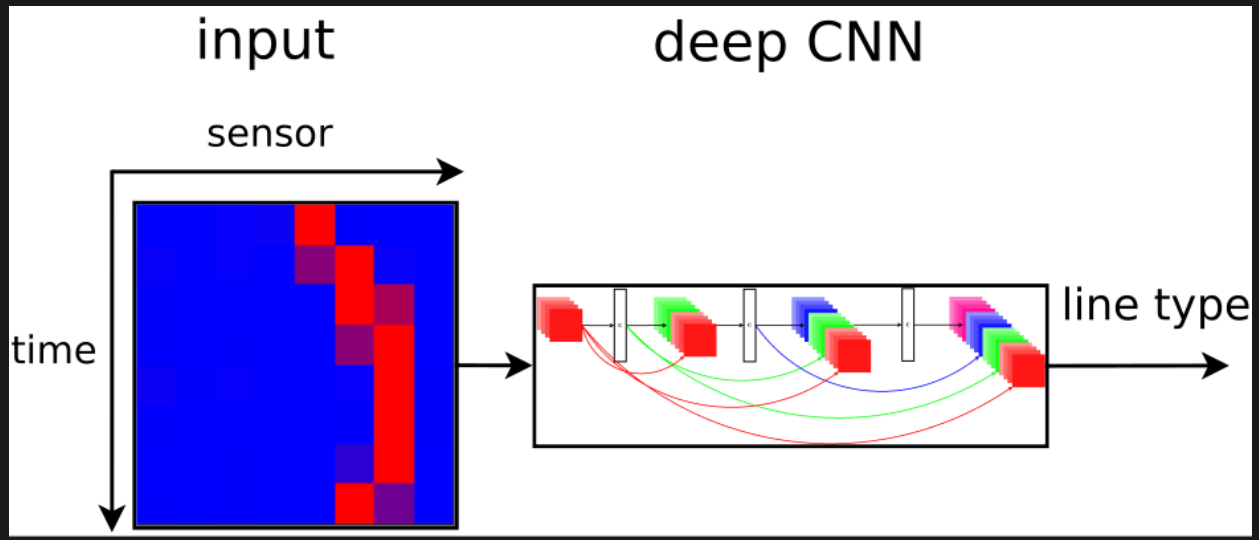


Software

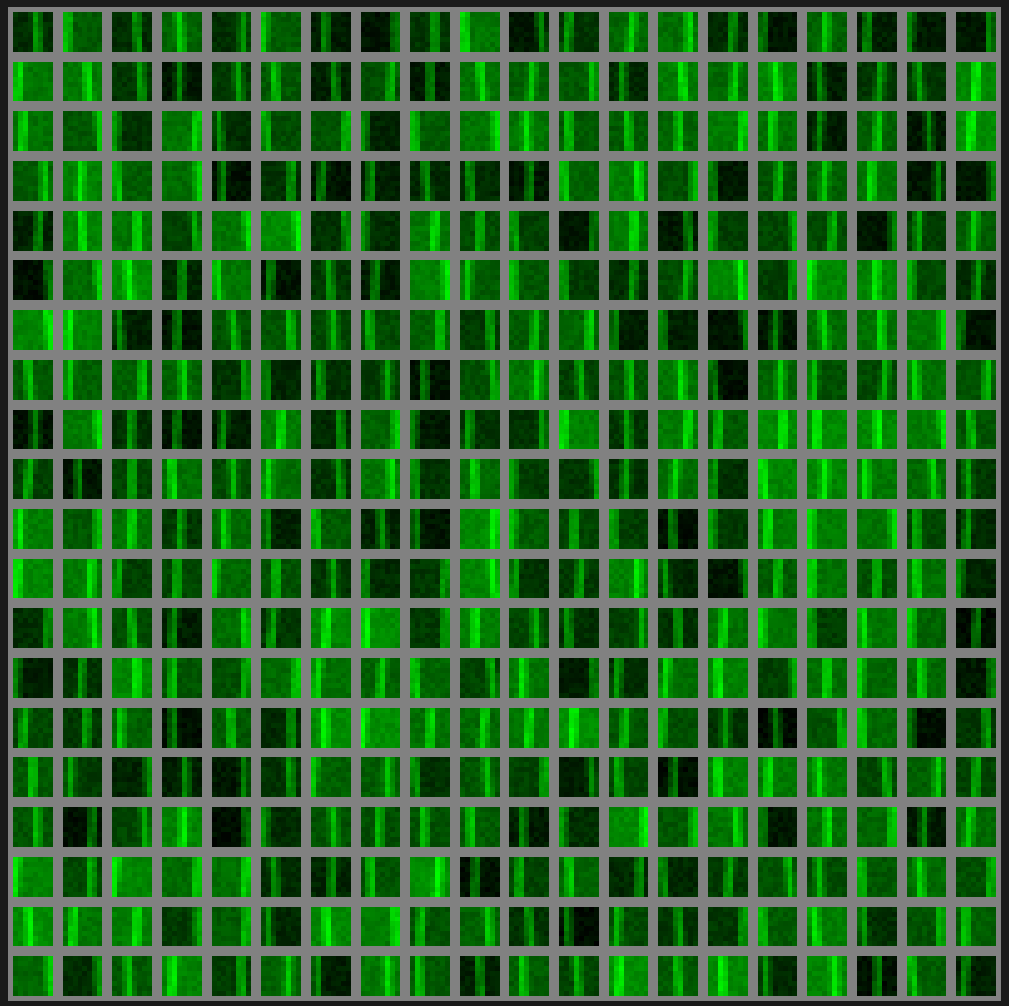

gyroscope
obstacle distance
encoders
line sensor
gimbal angles
median filtering
distance
line prediction
neural network
line position
brick avoid
distance PD
line search
position controll
line following
steering PD
motors PID
motors
sensor
sensor processing
computation
algorithmics
output

Line shape prediction

- **fast** run on straight line, **brake** on curve
- **neural network** for line type classification
  - DenseNet - densely connected convolutional neural network
- **input** 8x8 matrix raw data from line sensors
  - 8 past line positions from 8 sensors
- **output** 5 curves types



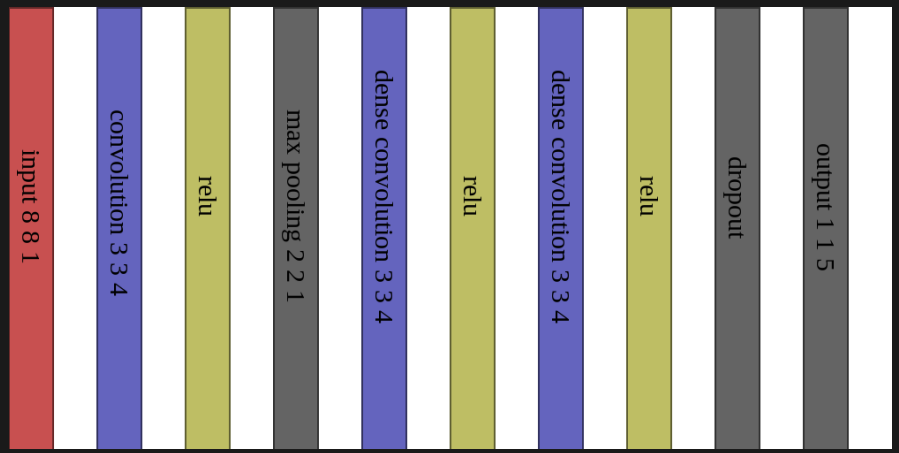
Line dataset



- 20000 for training
- 2500 for testing
- 8x8 inputs
- 5 outputs (5 curves types)
  - two left
  - one straight
  - two right
- augmentation - luma noise, white noise

Neural networks architecture

layer	net 0	net 1	net 2	net 3
0	conv 3x3x4	conv 3x3x6	conv 3x3x4	conv 3x3x6
1	max pooling 2x2	max pooling 2x2	max pooling 2x2	max pooling 2x2
2	dense conv 3x3x4	dense conv 3x3x6	dense conv 3x3x4	dense conv 3x3x6
3	fc 5	fc 5	dense conv 3x3x4	dense conv 3x3x6
4			fc 5	fc 5



Networks results

layer	net 0	net 1	net 2	net 3
success [%]	94.04	93.92	94.96	96.32
FLOPS	7754	13354	13578	25546
success / FLOPS	0.01213	0.00703	0.00699	0.00377

net 2 confusion matrix

class	0	1	2	3	4
0	**520**	8	0	0	0
1	12	**453**	15	0	0
2	0	12	**483**	10	0
3	0	0	15	**461**	53
4	0	0	0	1	**457**
class success [%]	97.744	95.772	94.152	97.669	89.608
total success [%]	**94.96**				

Fitting network into embedded

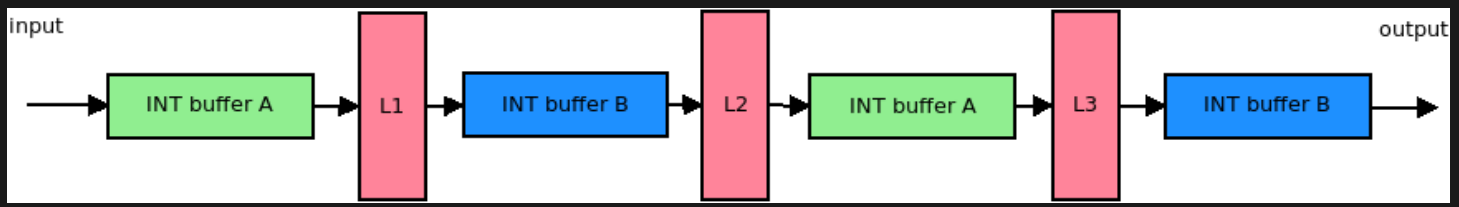
convert float weights to int8_t

$$scale = max(|\vec{w}|_1)$$

$$\vec{w}' = \vec{w}\frac{127}{scale}$$

use double buffer memory trick

- unsigned buffer_size = max_i(layers[i].input_size());
- buffer_a = new int8_t(buffer_size);
- buffer_b = new int8_t(buffer_size);



Q&A

michal chovanec (michal.nand@gmail.com)

www.youtube.com/channel/UCzVvP2ou8v3afNiVrPAHQGg

github https://github.com/michalnand