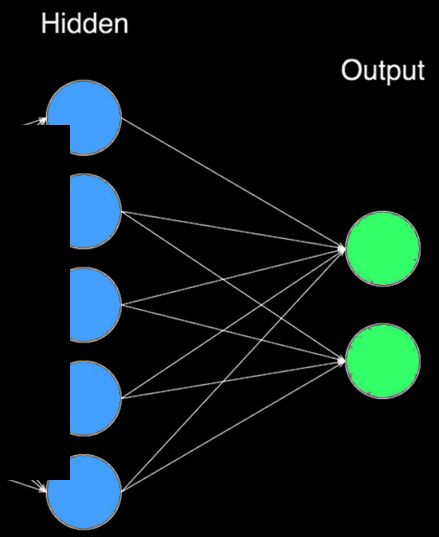
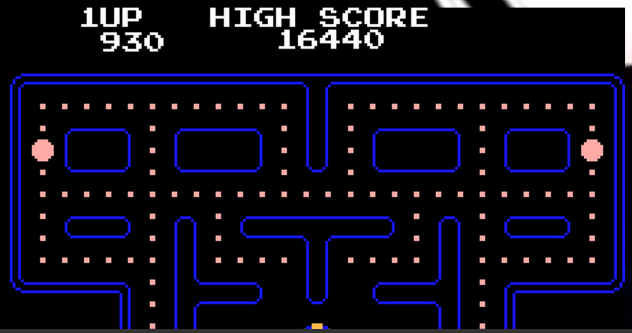
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

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

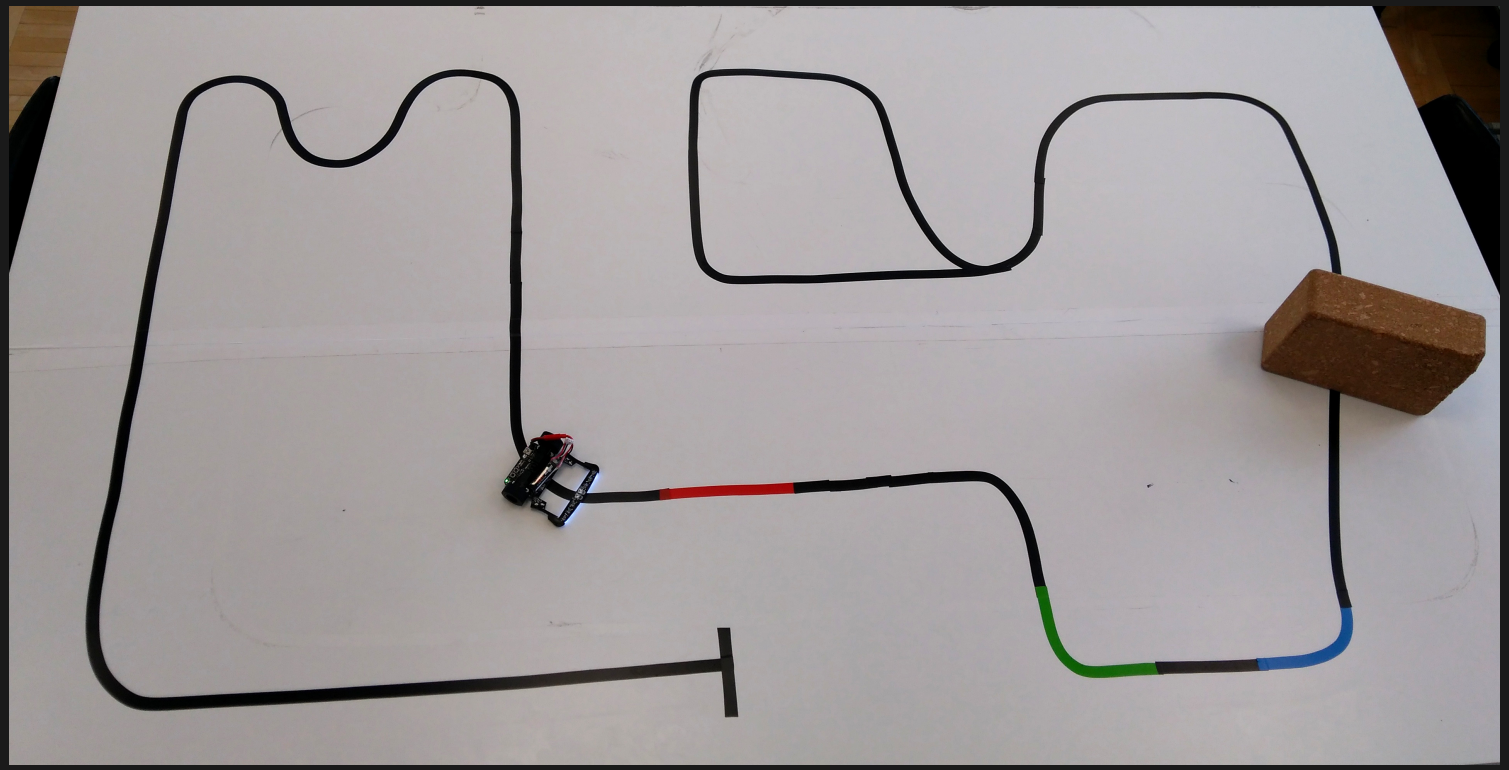
deep learning for line following robot

Michal CHOVANEC





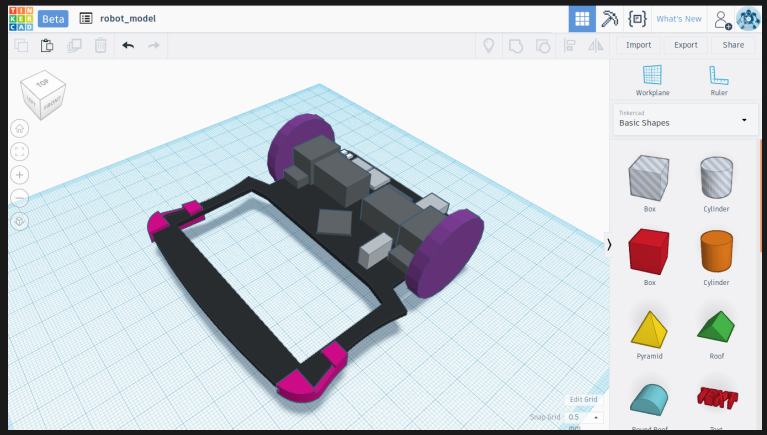
line follower competition

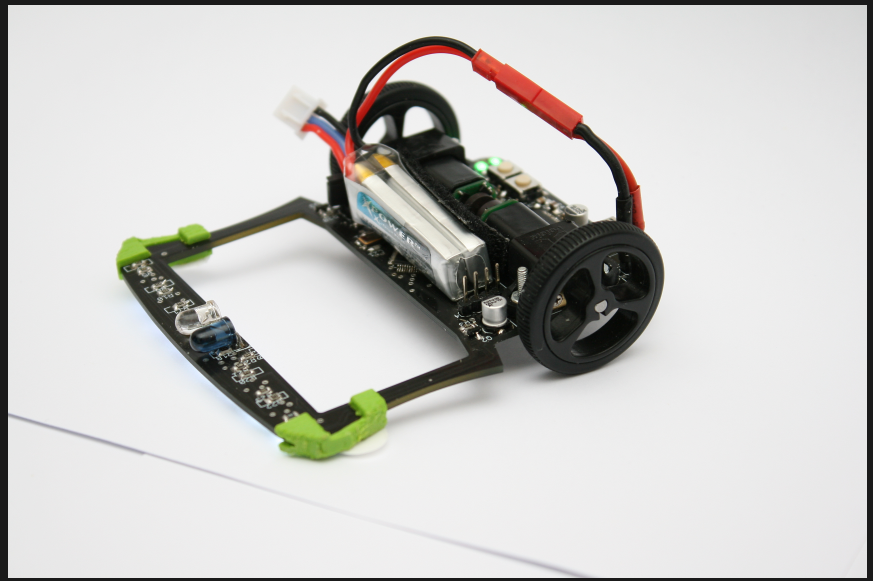


- SR - Istrobot
- CR - Roboticky den
- AU - Robot Challenge
- PL - Zawody robotow

what does it take

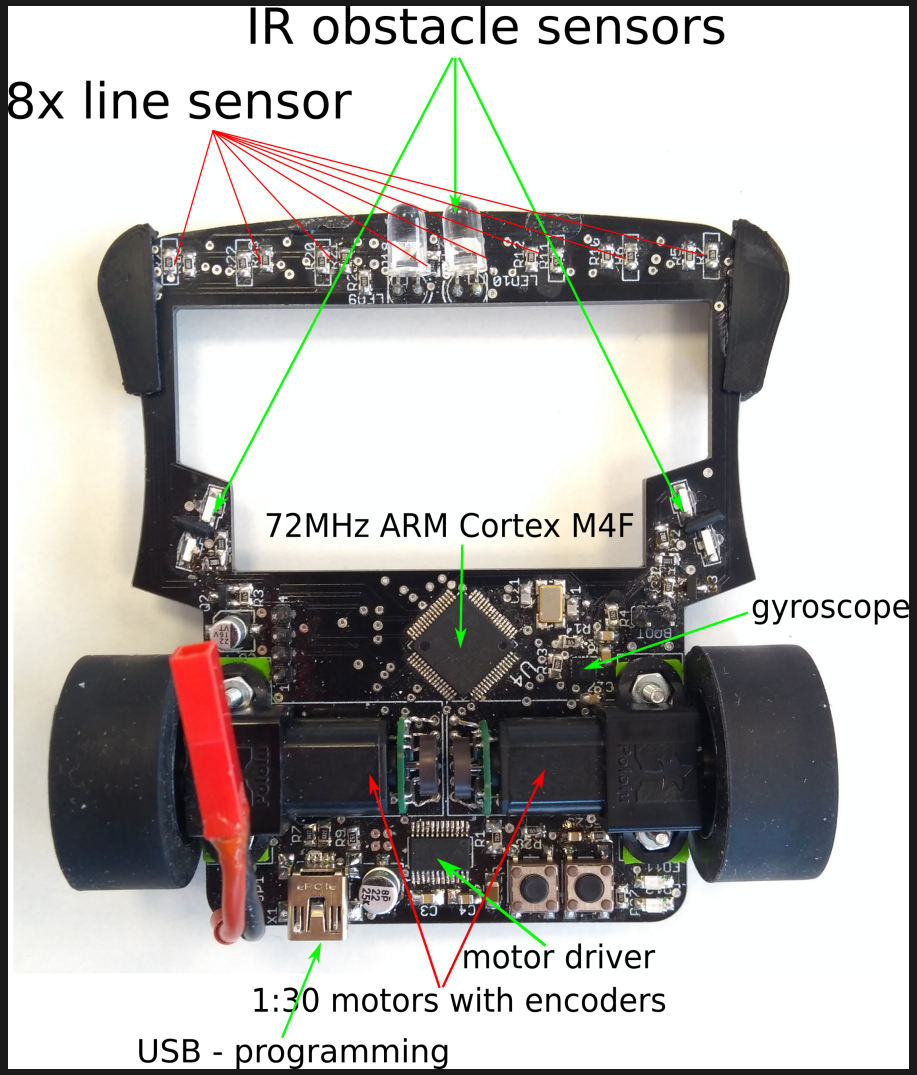
- Hardware
  - strong light motors
  - high adhesion tyres
  - light accumulator
  - fast CPU
  - dozen of sensors
- Software
  - hard real time OS
  - tunned PIDs
  - predictive controll
  - mapping





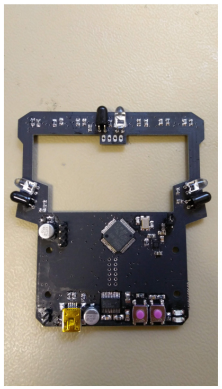
DL linefollower

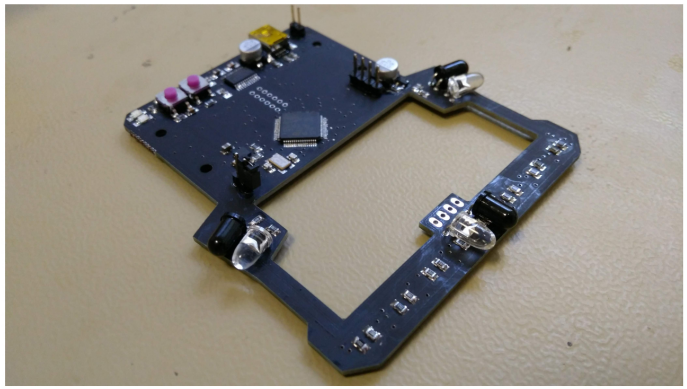
motoko uprising - hardware

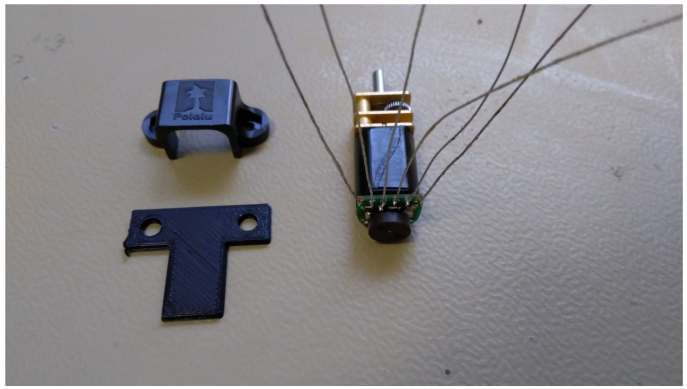


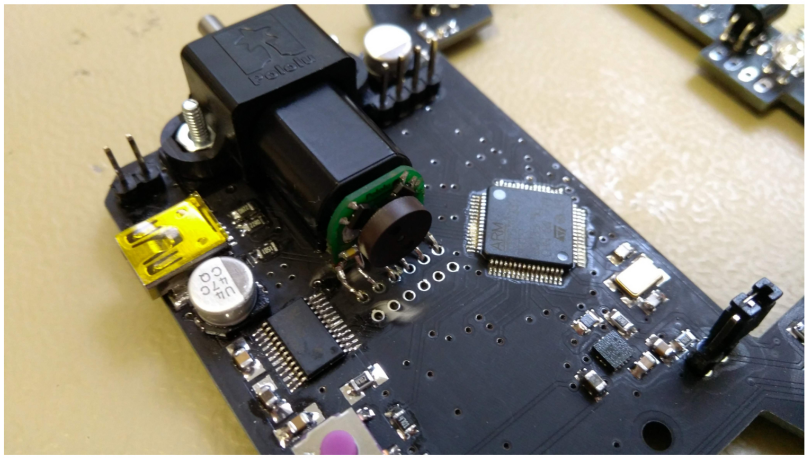
- **mcu** : STM32F303, 72MHz ARM Cortex M4F
- **motor driver** : TI DRV8834
- **motors** : 1:30 HP Pololu, with magnetic encoder
- **tyres** : Pololu 28mm diameter
- **line sensor** : 8x 540nm phototransistor + white LED
- **obstacle sensor** : SMD IR phototransistor + IR LED
- **imu** : LSM6DS0, gyroscope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

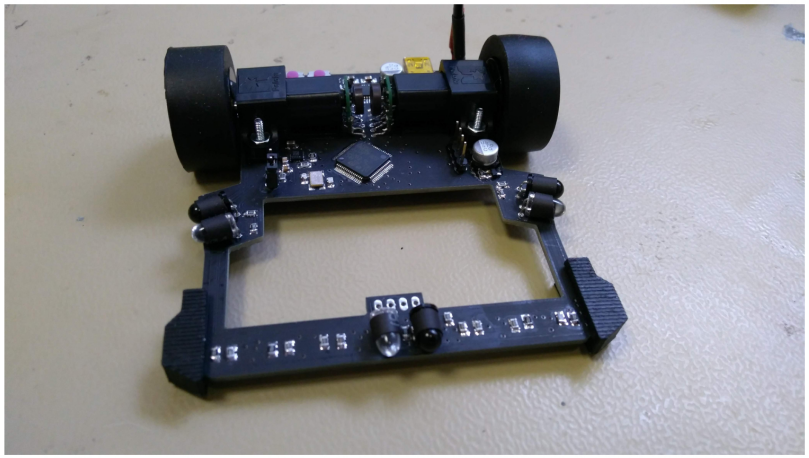
motoko uprising - hardware





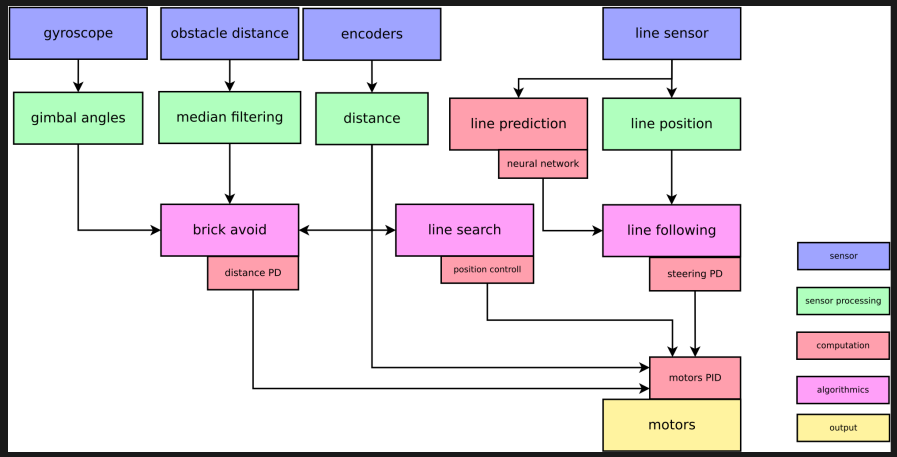






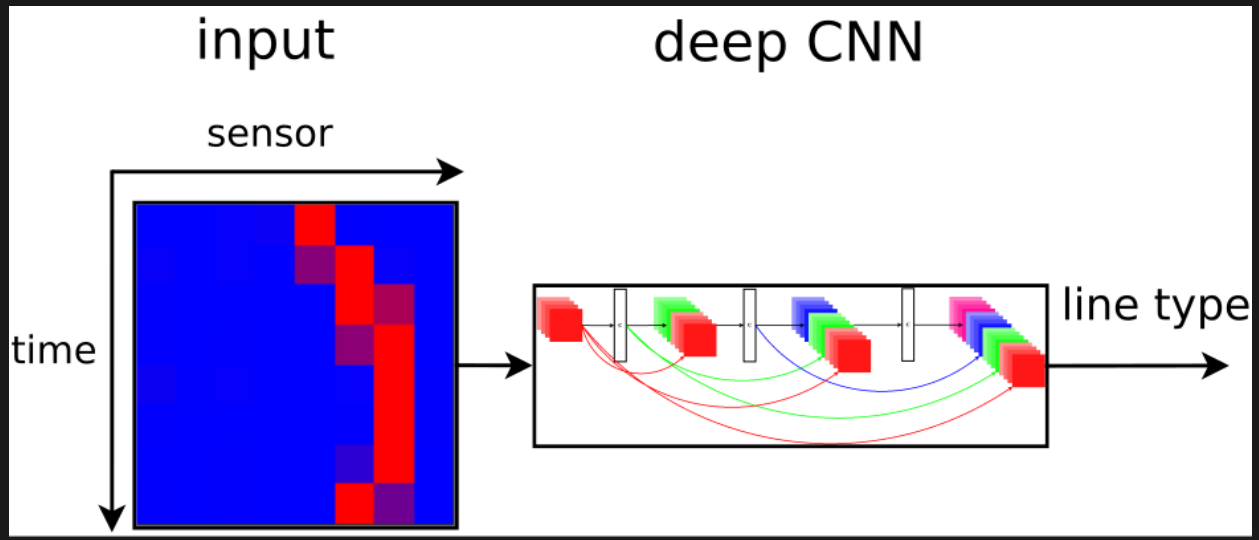
software

- **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

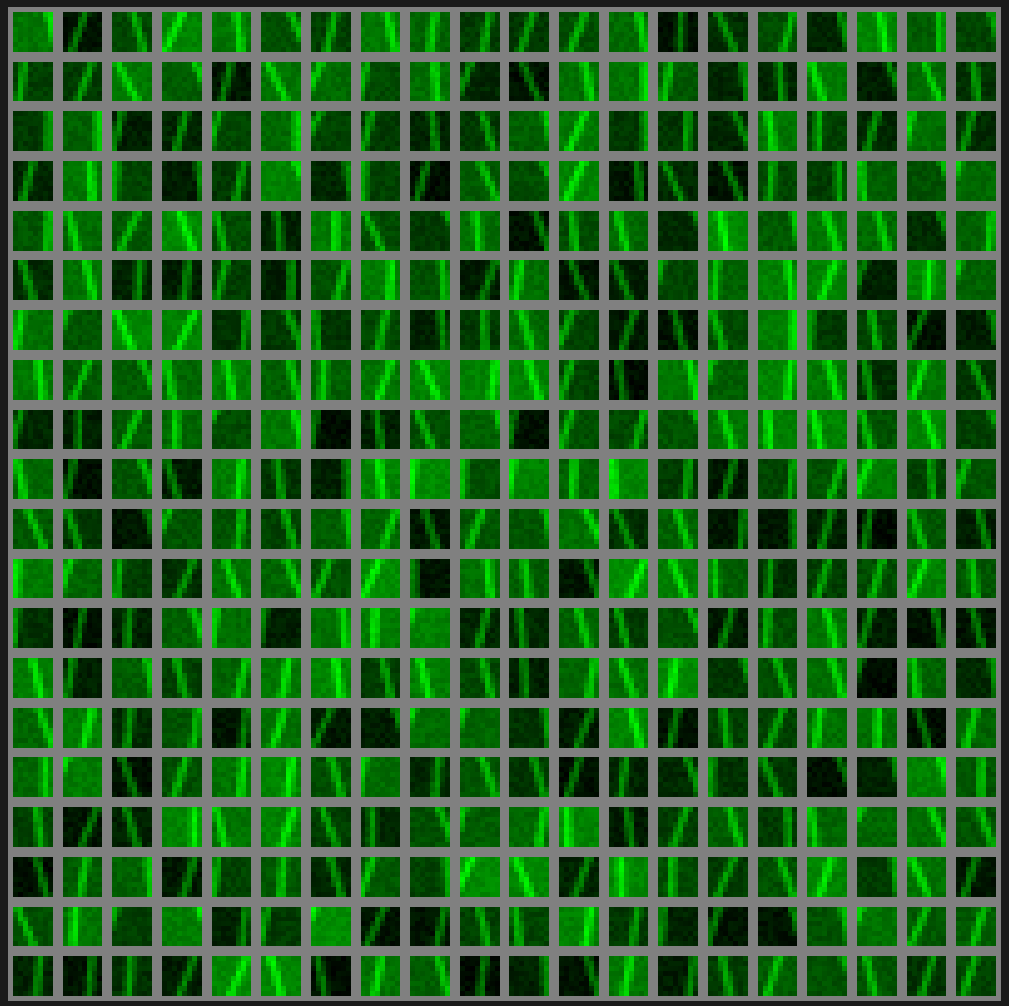


line shape prediction

- **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

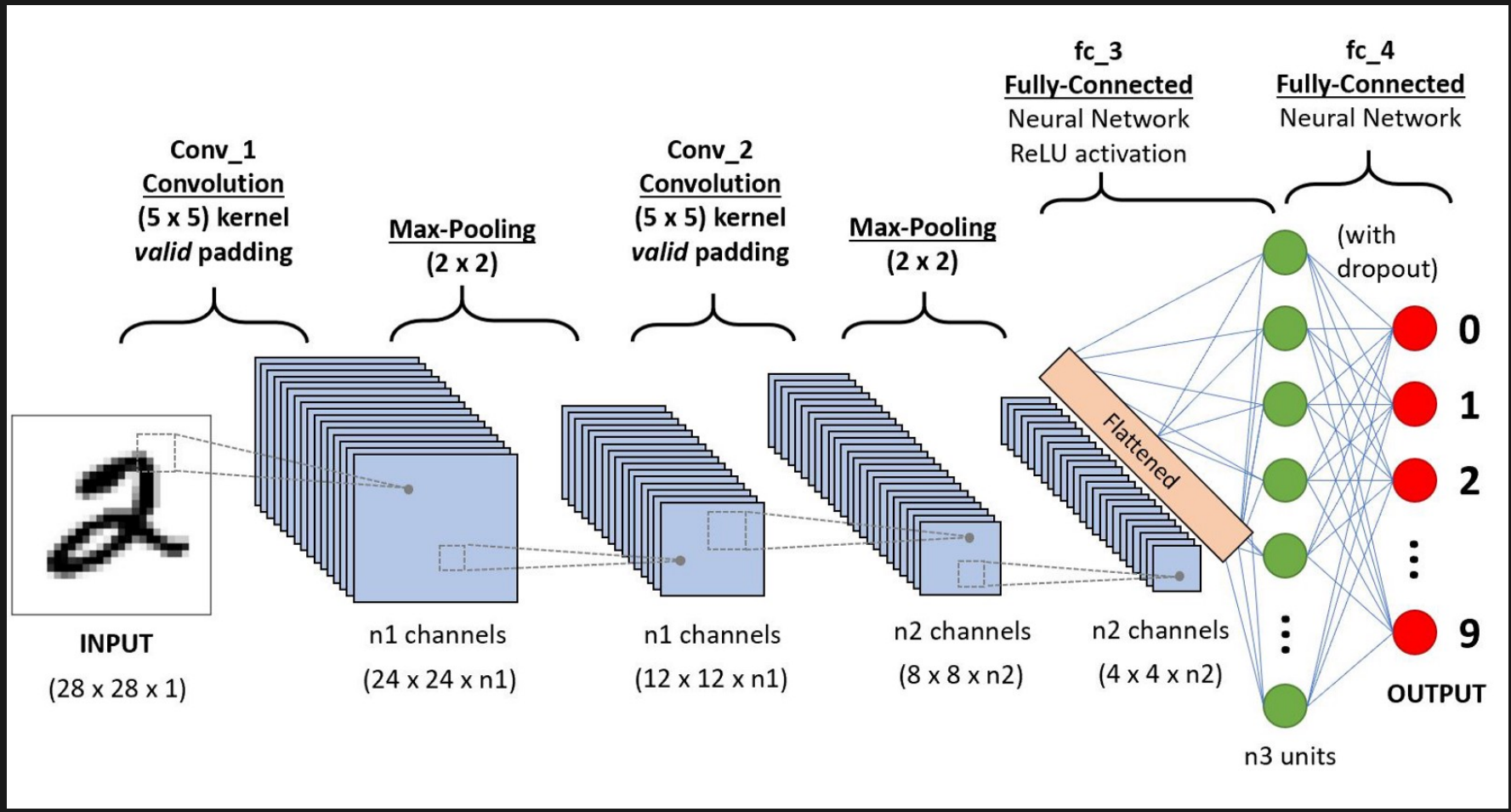


line dataset

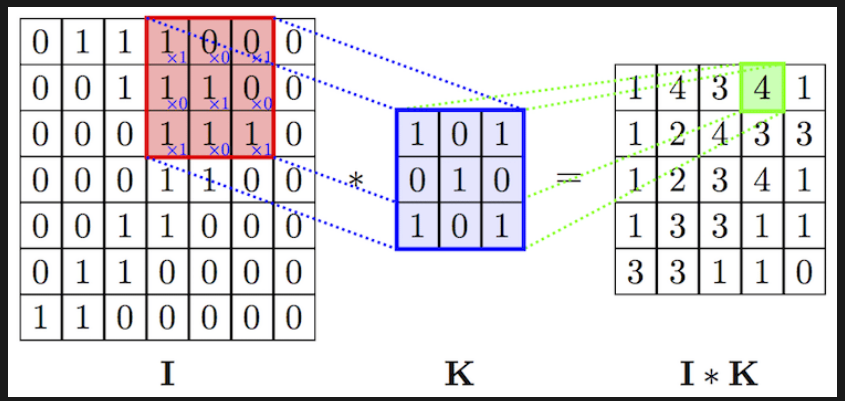


- 20000 for training
- 2500 for testing
- 8x8 inputs
- 5 outputs (5 curves types)
  - two left
  - one straight
  - two right
- augmentation - luma noise, white noise

convolutional neural network - CNN



discrete convolution



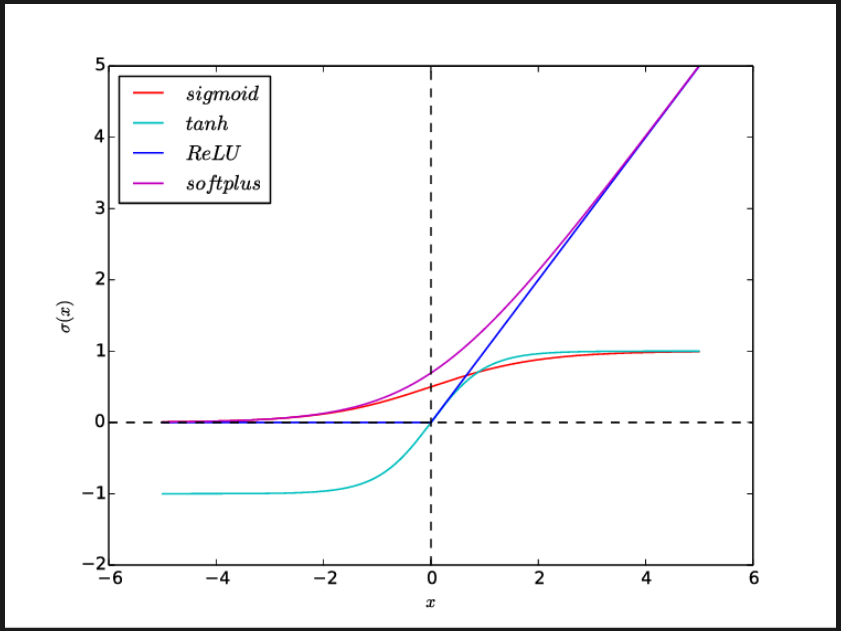

```
for (unsigned y = 0; y < input_height; y++)
for (unsigned x = 0; x < input_width; x++)
{
    float sum = 0.0;

    for (unsigned ky = 0; ky < kernel_height; ky++)
    for (unsigned kx = 0; kx < kernel_width; kx++)
    {
      sum+= kernel[ky][kx]*input[y + ky][x + kx];
    }

    output[y + kernel_height/2][x + kernel_width/2] = sum;
  }
}
```

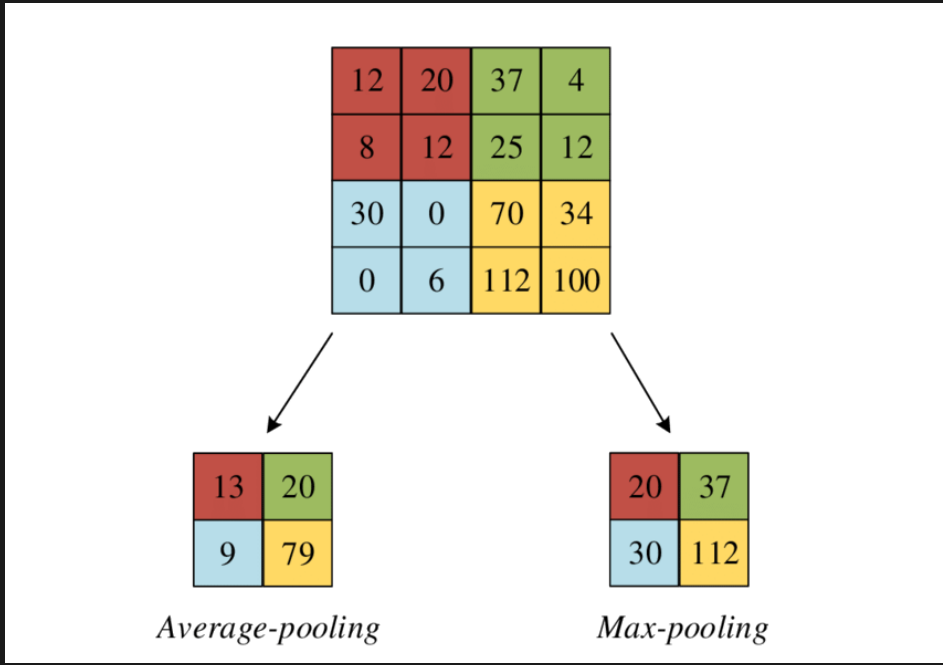
activation function

tanh, sigmoid, gaussian, **RELU**, leaky RELU ...



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \frac{df(x)}{dx} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

pooling


12 20 37 4
8 12 25 12
30 0 70 34
0 6 112 100
13 20
9 79
Average-pooling
20 37
30 112
Max-pooling

target hardware

target	bits	features	frequency
AVR Atmega 328	8	single cycle ADD, MUL	20MHz
ARM Cortex M0	32	single cycle ADD, MUL	48MHz
ARM Cortex M4, M7	32	SIMD DUAL 16bit MAC , FPU ...	72MHz 216MHz

embedded network implementation

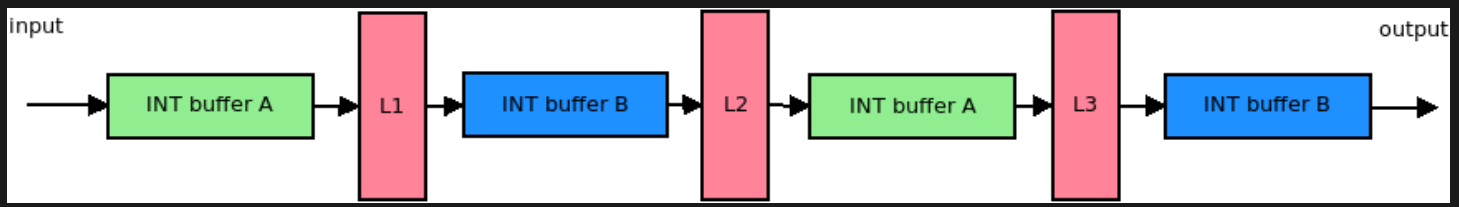
convert float weights to int8_t

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

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

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);



optimize kernel - templates

```
templete<unsigned int kernel_size>
void convolution()
{
  for (unsigned y = 0; y < input_height; y++)
  for (unsigned x = 0; x < input_width; x++)
  {
      int sum = 0;

      for (unsigned ky = 0; ky < kernel_size; ky++)
      for (unsigned kx = 0; kx < kernel_size; kx++)
      {
        sum+= kernel[ky][kx]*input[y + ky][x + kx];
      }

      output[y + kernel_size/2][x + kernel_size/2] = (sum*scale)/127;
    }
  }
}
```

optimize kernel - unrolling

```
templete<unsigned int kernel_size>
void convolution()
{
  for (unsigned y = 0; y < input_height; y++)
  for (unsigned x = 0; x < input_width; x++)
  {
      int sum = 0;

      if (kernel_size == 3)
      {
        sum+= kernel[0][0]*input[y + 0][x + 0];
        sum+= kernel[0][1]*input[y + 0][x + 1];
        sum+= kernel[0][2]*input[y + 0][x + 2];

        sum+= kernel[1][0]*input[y + 1][x + 0];
        sum+= kernel[1][1]*input[y + 1][x + 1];
        sum+= kernel[1][2]*input[y + 1][x + 2];

        sum+= kernel[2][0]*input[y + 2][x + 0];
        sum+= kernel[2][1]*input[y + 2][x + 1];
        sum+= kernel[2][2]*input[y + 2][x + 2];
      }

      output[y + kernel_size/2][x + kernel_size/2] = (sum*scale)/127;
    }
  }
}
```

3.6x speed up

optimize kernel - SIMD instructions

```
sum+= kernel[0][0]*input[y + 0][x + 0];
sum+= kernel[0][1]*input[y + 0][x + 1];
sum+= kernel[0][2]*input[y + 0][x + 2];
```

```
smlabb  r2, fp, sl, r2
ldrsb.w sl, [r8, #1]
ldrsb.w fp, [r0, #-24]

smlabb  r2, fp, sl, r2
ldrsb.w sl, [r8, #2]
ldrsb.w fp, [r0, #-23]

smlabb  r2, fp, sl, r2
ldrsb.w sl, [r8, #3]
ldrsb.w fp, [r0, #-22]
```

network implementation

architecture : IN8x8x1 - C3x3x4 - P2x2 - C3x3x8 - P2x2 - FC5

```
layers[0] = new EmbeddedNetConvolutionLayer(
          layer_0_shape, layer_0_input_shape, layer_0_output_shape,
          layer_0_weights, layer_0_bias, layer_0_weights_range, layer_0_bias_range);

layers[1] = new EmbeddedNetMaxPoolingLayer(
          layer_1_shape, layer_1_input_shape, layer_1_output_shape);

layers[2] = new EmbeddedNetConvolutionLayer(
          layer_2_shape, layer_2_input_shape, layer_2_output_shape,
          layer_2_weights, layer_2_bias, layer_2_weights_range, layer_2_bias_range);

layers[3] = new EmbeddedNetMaxPoolingLayer(
          layer_3_shape, layer_3_input_shape, layer_3_output_shape);

layers[4] = new EmbeddedNetFcLayer(
          layer_4_shape, layer_4_input_shape, layer_4_output_shape,
          layer_4_weights, layer_4_bias, layer_4_weights_range, layer_4_bias_range);
```

Q&A

michal chovanec (michal.nand@gmail.com)

www.youtube.com/channel/UCzVvP2ou8v3afNiVrPAHQGg

github https://github.com/michalnand