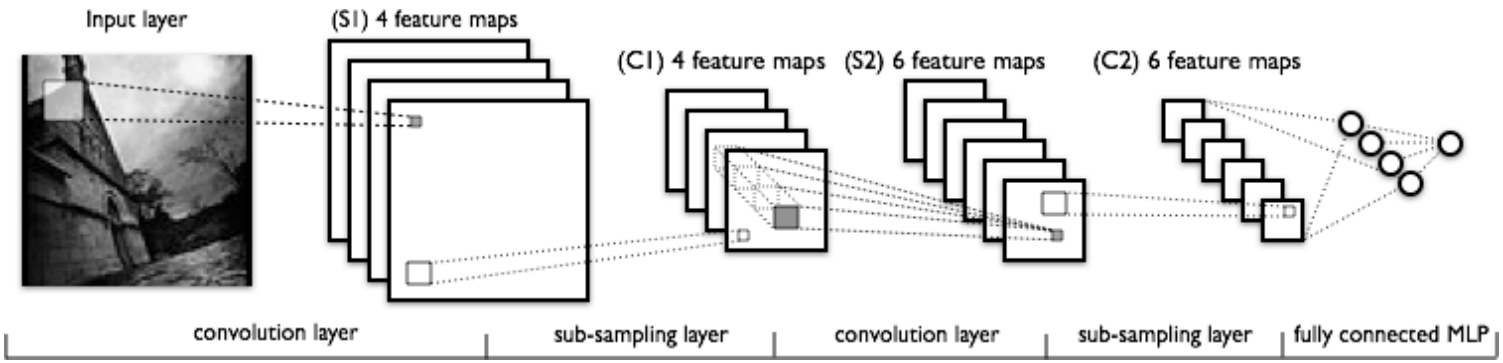
Convolutional neural network

Michal CHOVANEC, PhD.

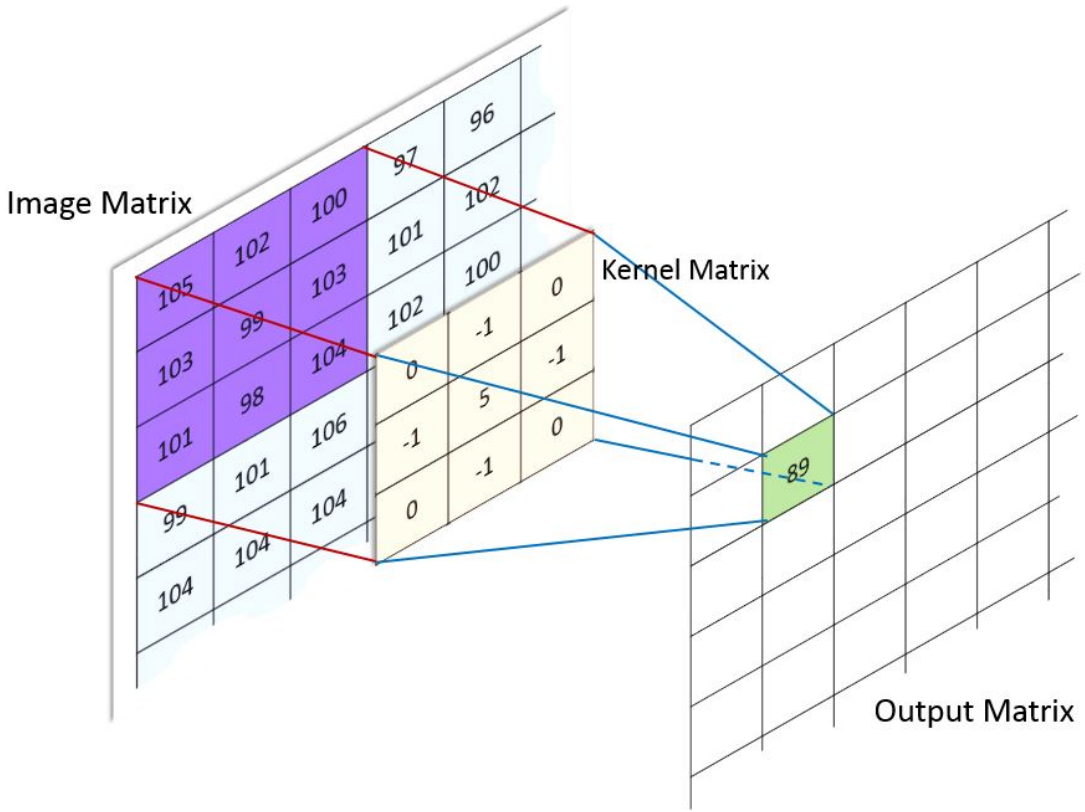
April 2018

Faculty of Management Science and Informatics

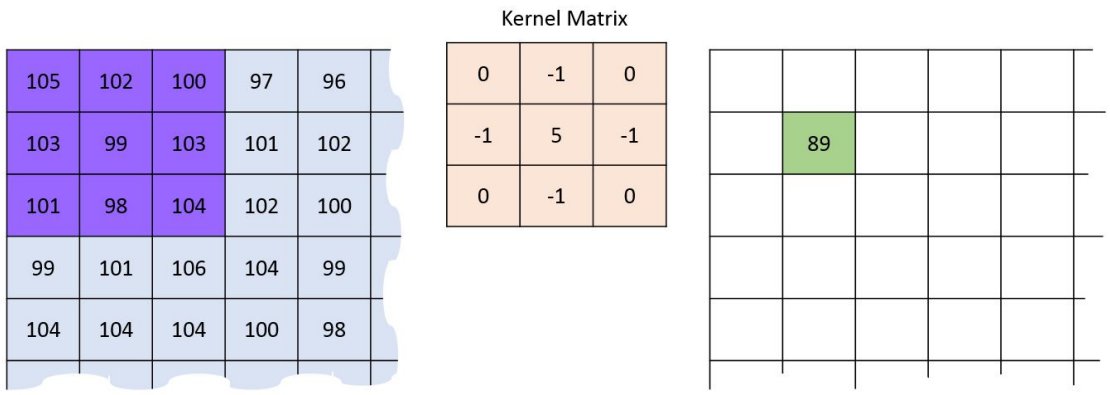
CNN net architecture



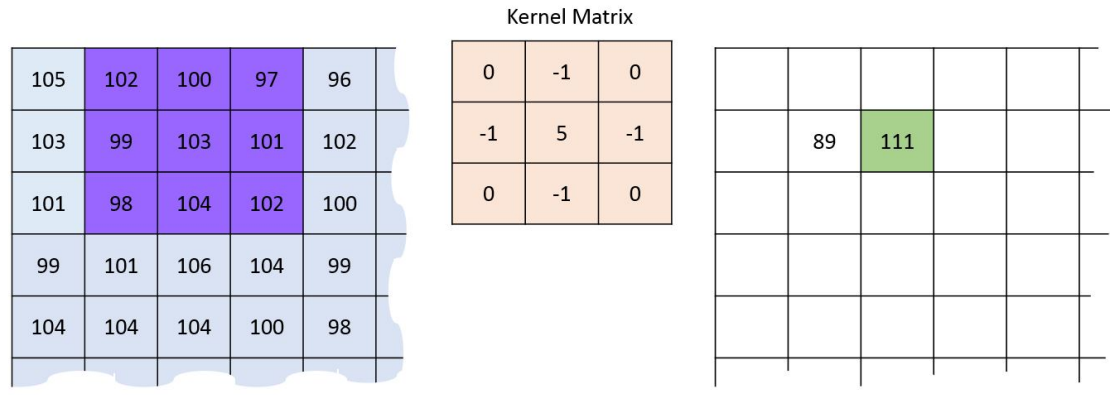
2D convolution



2D convolution



$$105 * 0 + 102 * -1 + 100 * 0 + 103 * -1 + 99 * 5 + 103 * -1 + 101 * 0 + 98 * -1 + 104 * 0 = 89$$



$$102 * 0 + 100 * -1 + 97 * 0 + 99 * -1 + 103 * 5 + 101 * -1 + 98 * 0 + 104 * -1 + 102 * 0 = 111$$

2D 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+= w[ky][kx]*input[y + ky][x + kx];
    }

    output[y][x] = sum;
  }
```

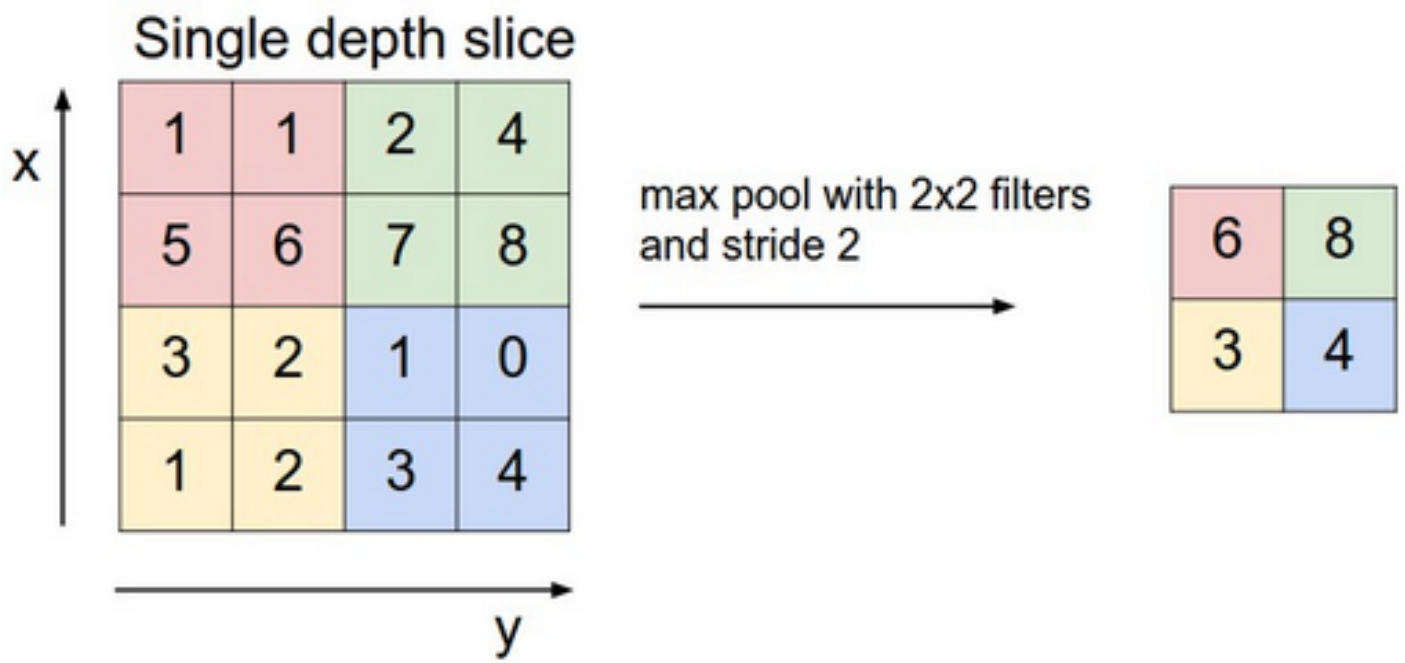
2D convolution - multiple filters and channels

```
for (unsigned filter = 0; filter < filters_count; filter++)
for (unsigned y = 0; y <= input_height; y++)
for (unsigned x = 0; x <= input_width; x++)
{
    float sum = 0.0;
    for (unsigned ch = 0; ch < channels_count; ch++)
    for (unsigned ky = 0; ky < kernel_height; ky++)
    for (unsigned kx = 0; kx < kernel_width; kx++)
    {
      sum+= w[filter][ch][ky][kx]*input[ch][y + ky][x + kx];
    }

  output[filter][y][x] = sum;
}
```

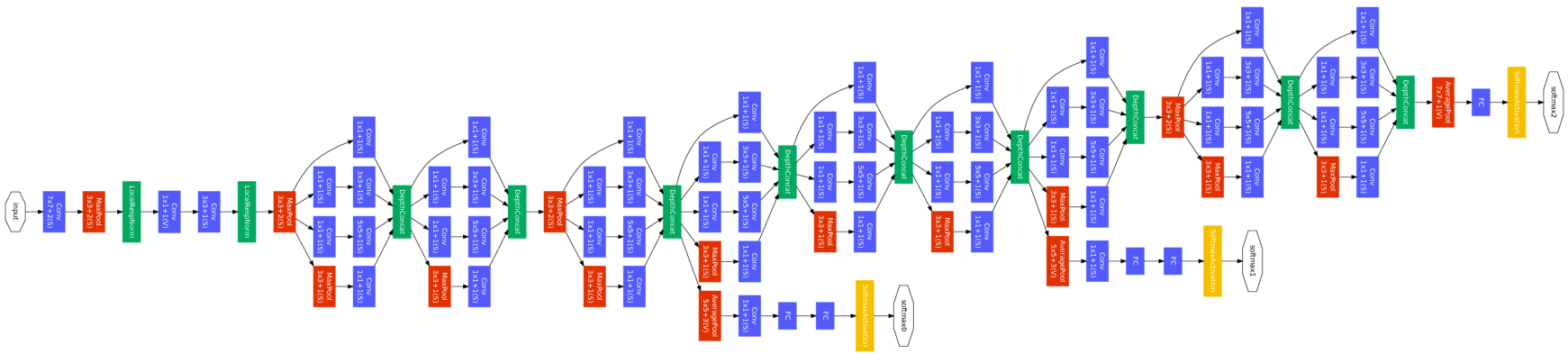
Learning weights

```
for (unsigned filter = 0; filter < filters_count; filter++)
for (unsigned y = 0; y <= input_height; y++)
for (unsigned x = 0; x <= input_width; x++)
{
    float err = error[filter][y][x];

    for (unsigned ch = 0; ch < channels_count; ch++)
    for (unsigned ky = 0; ky < kernel_height; ky++)
    for (unsigned kx = 0; kx < kernel_width; kx++)
    {
        float dif = err*input[ch][y + ky][x + kx];
        w[filter][ch][ky][kx]+= dif*learning_rate;
    }
}
```

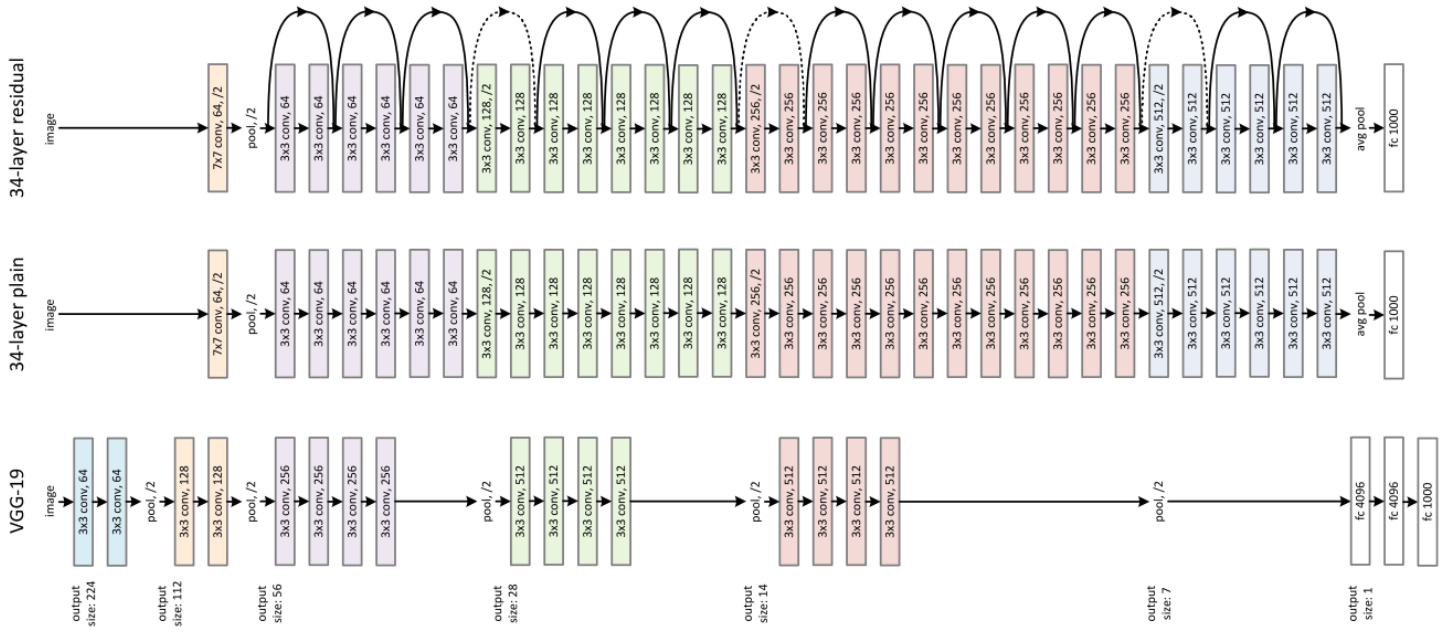
Pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

GoogleNet - inception



Resnet


VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 224
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 112
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 56
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

Q&A

https://github.com/michalnand/robotics
https://github.com/michalnand/machine_learning

michal.nand@gmail.com