WB-XIC, Lab5:

Konwolucyjne sieci neuronowe w praktyce: ResNet & DenseNet

Hubert Baniecki

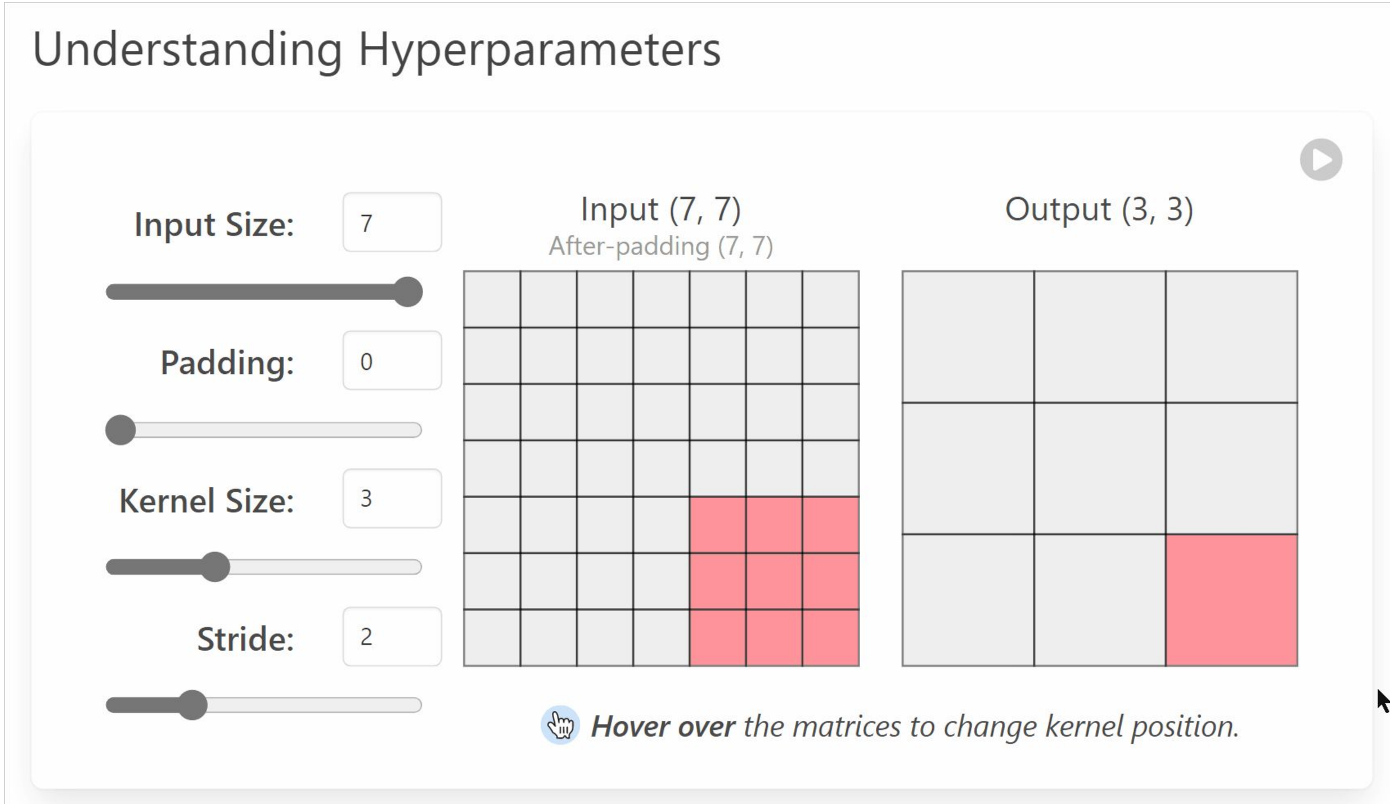
hbaniecki@gmail.com | http://hbaniecki.com

Cel:

1. Stride
2. Google Colab
3. Batch Normalization
4. ResNet
5. DenseNet
6. Zadanie domowe

Stride: Conv2d parameter

https://poloclub.github.io/cnn-explainer



Understanding Hyperparameters

Input Size: 7

Padding: 0

Kernel Size: 3

Stride: 2

Hover over the matrices to change kernel position.

Google Colab (powtórka)

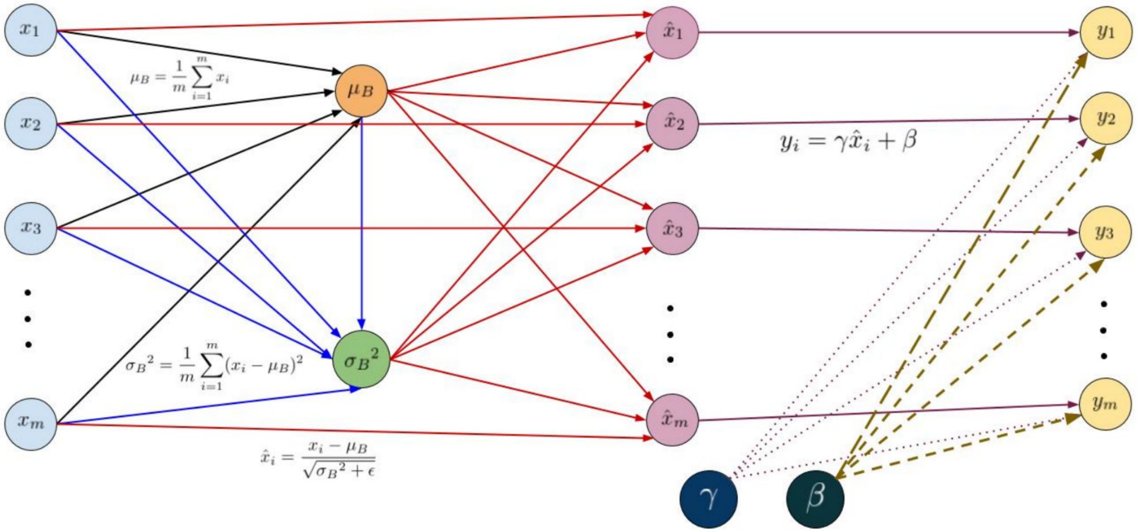
Batch Normalization

https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html

https://zaffnet.github.io/batch-normalization

*https://arxiv.org/abs/1502.03167

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$



x_1
x_2
x_3
x_m
$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$
μ_B
$\sigma_B{}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
$\sigma_B{}^2$
$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B{}^2 + \epsilon}}$
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_m$
$y_i = \gamma\hat{x}_i + \beta$
y_1
y_2
y_3
y_m
γ
β

ResNet

https://arxiv.org/abs/1512.03385

https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py

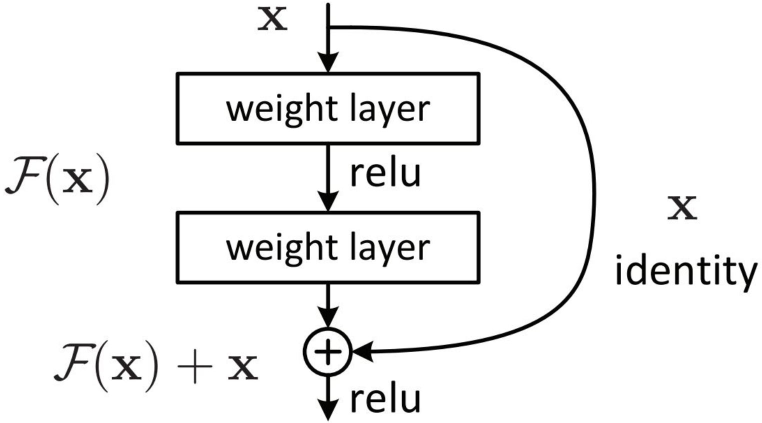
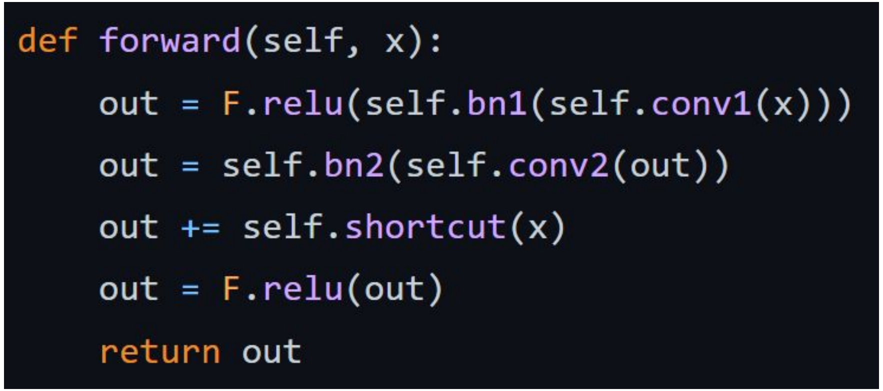


Figure 2. Residual learning: a building block.



```
def forward(self, x):
    out = F.relu(self.bn1(self.conv1(x)))
    out = self.bn2(self.conv2(out))
    out += self.shortcut(x)
    out = F.relu(out)
    return out
```

DenseNet

https://arxiv.org/abs/1608.06993

https://github.com/kuangliu/pytorch-cifar/blob/master/models/densenet.py

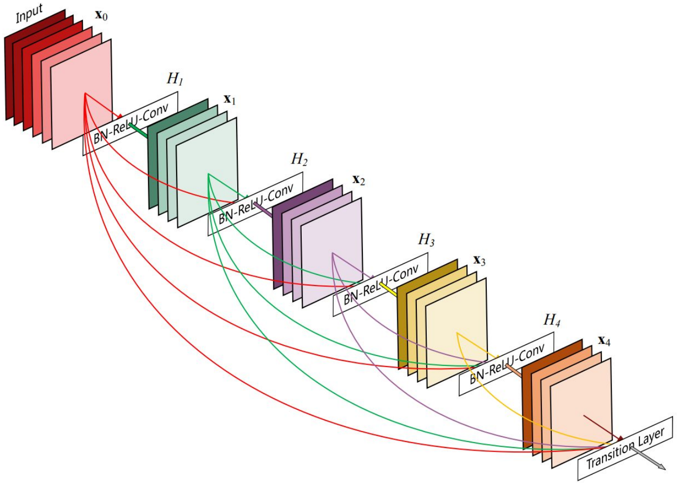
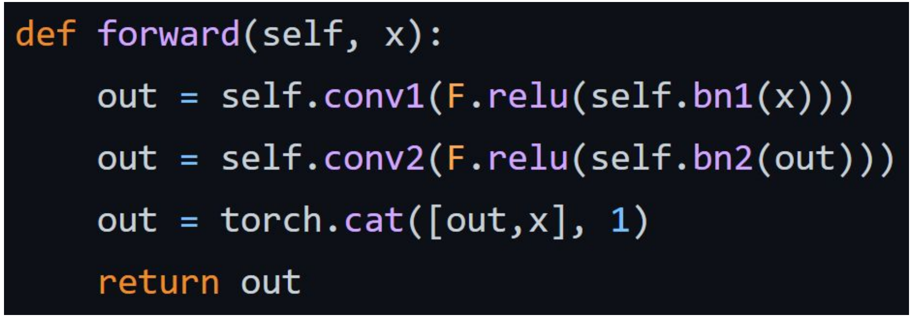


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.



```
def forward(self, x):
    out = self.conv1(F.relu(self.bn1(x)))
    out = self.conv2(F.relu(self.bn2(out)))
    out = torch.cat([out,x], 1)
    return out
```

Pytania, wnioski, zadanie domowe