WB-XIC, Lab5:

Konwolucyjne sieci neuronowe w praktyce: ResNet & DenseNet

Hubert Baniecki

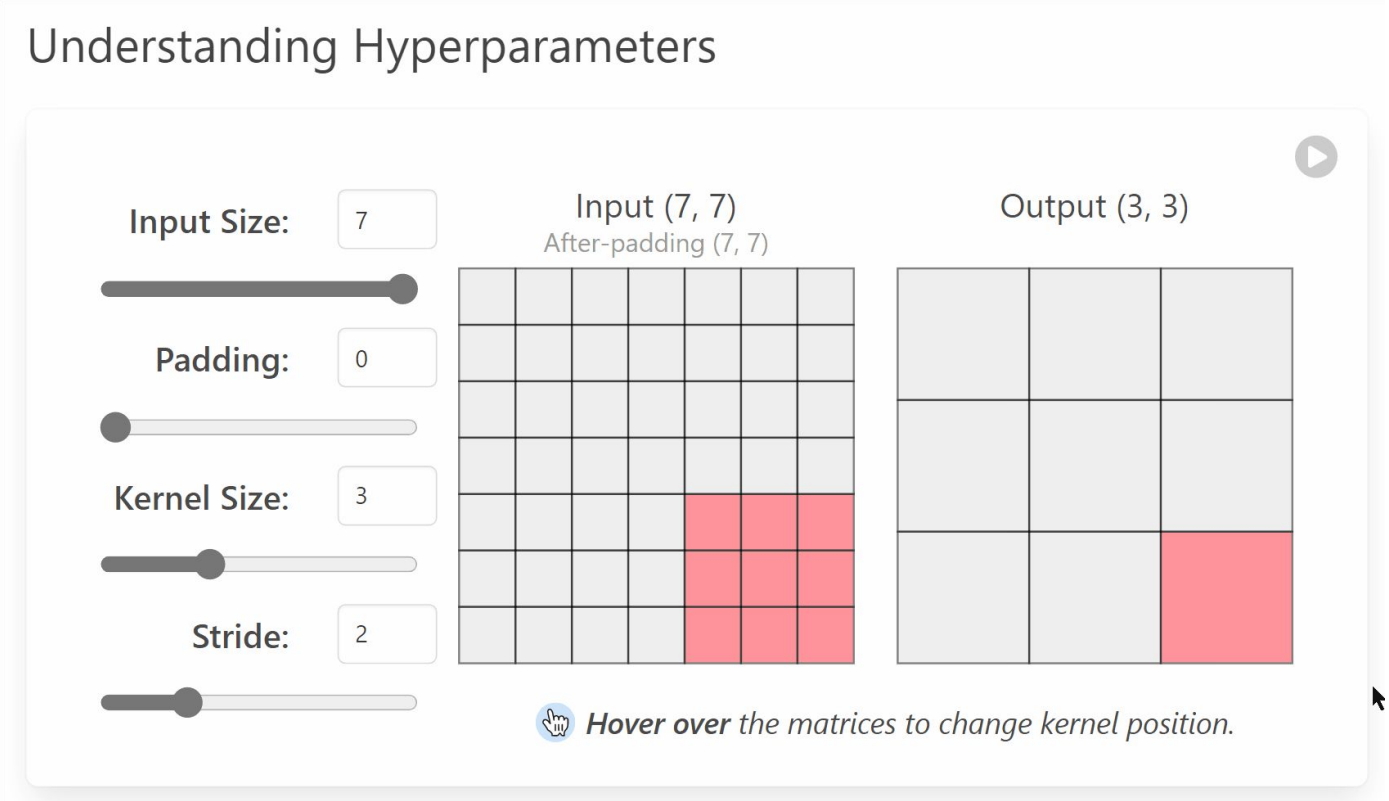
hbaniecki@gmail.com | http://hbaniecki.com

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

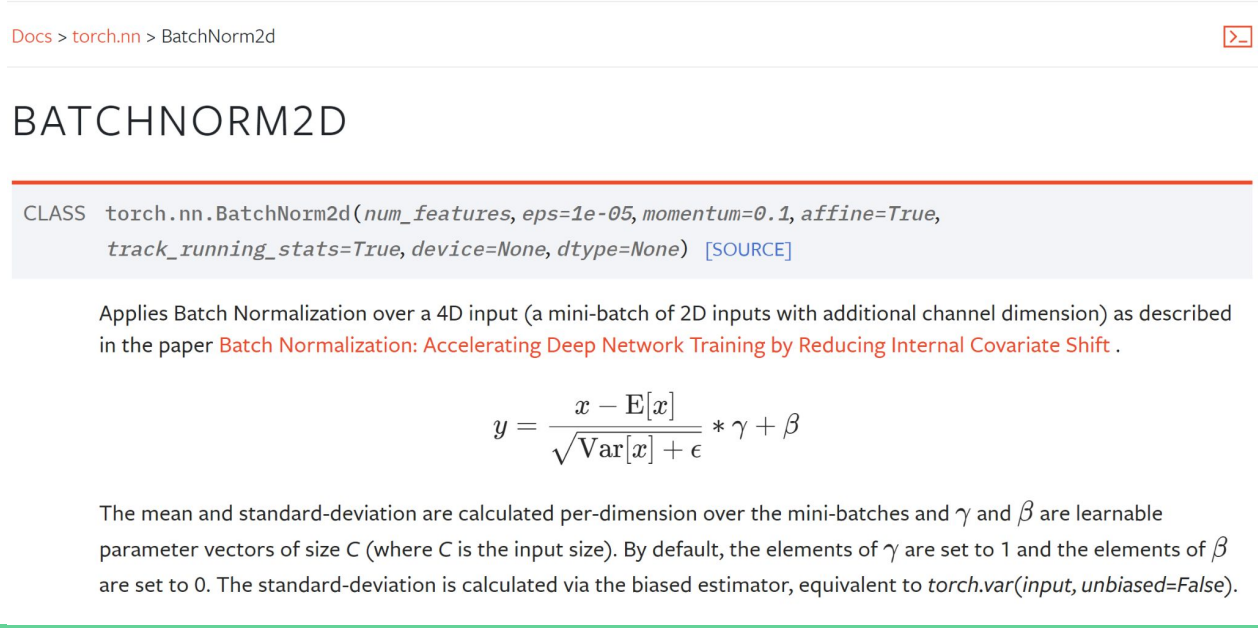


Google Colab (powtórka)

Batch Normalization

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

https://arxiv.org/abs/1502.03167



Docs > torch.nn > BatchNorm2d

BATCHNORM2D

CLASS `torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True, device=None, dtype=None)` [SOURCE]

Applies Batch Normalization over a 4D input (a mini-batch of 2D inputs with additional channel dimension) as described in the paper Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift .

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

The mean and standard-deviation are calculated per-dimension over the mini-batches and γ and β are learnable parameter vectors of size C (where C is the input size). By default, the elements of γ are set to 1 and the elements of β are set to 0. The standard-deviation is calculated via the biased estimator, equivalent to *torch.var(input, unbiased=False)*.

ResNet

https://arxiv.org/abs/1512.03385

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

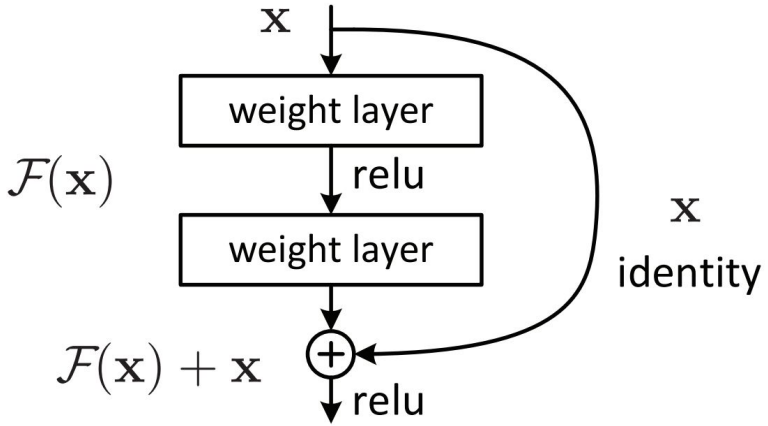


Figure 2. Residual learning: a building block.

```
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

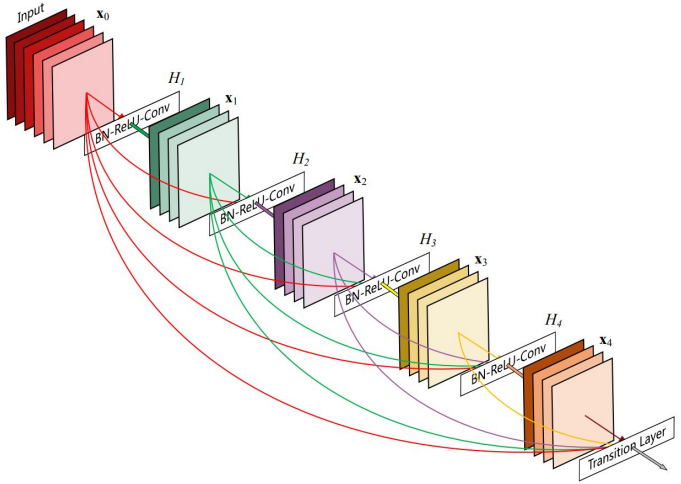


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

```
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