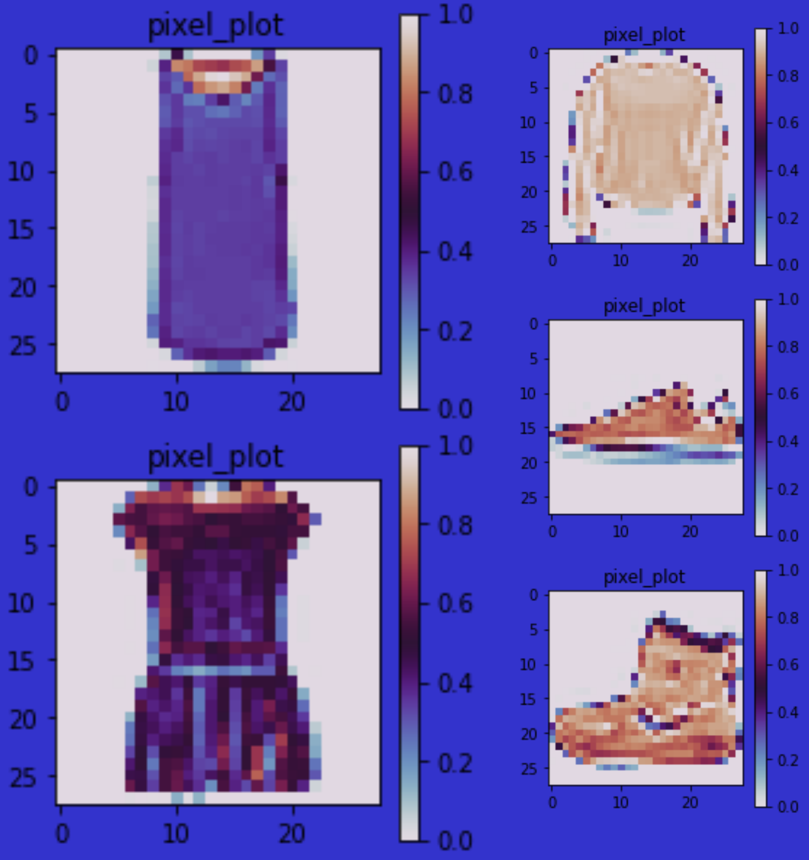
CS-E4890: Deep Learning

Convolutional neural networks

Q/A Session

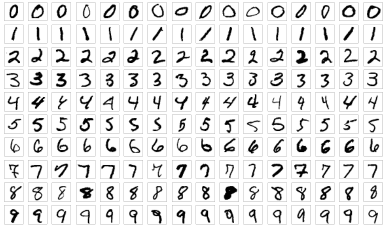
Taha Heidari
10.03.2023



Aalto University

Motivation

- Human eye, evolution
- CNNs inspired by visual cortex in 1981
- CNNs introduced in 1998 by Yann LeCun
- More Accuracy than MLPs
- With Less parameters
- Space invariant
- Keep 2D information
- IMAGENET



Hubel and Weisel after winning their Nobel Prize, 1981. Courtesy of *Harvard University Archives*.

Ilya Sutskever, Alex Krizhevsky and University Professor Geoffrey Hinton of the University of Toronto's Department of Computer Science (photo by John Guatto)

Motivation

- IMAGENET

ImageNet Large Scale Recognition Challenge 2012

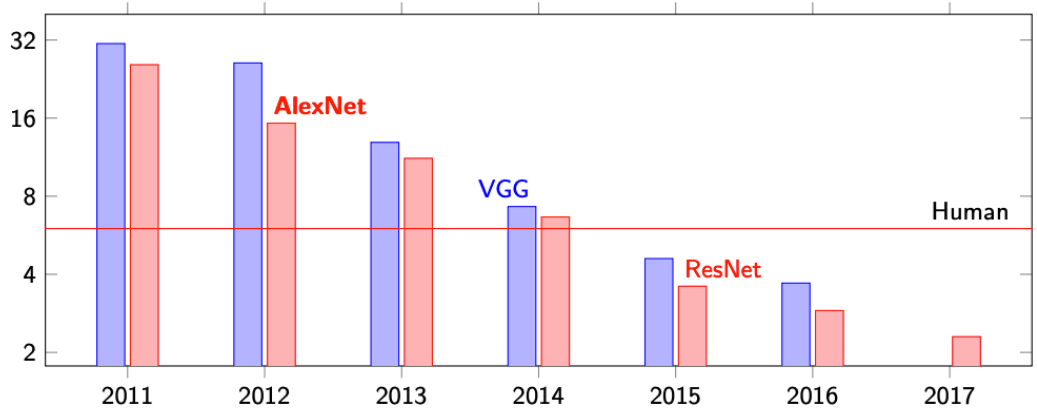
1.2 M images in 1000 classes

- AlexNet **16.4%** error with 8 layers
- Oxford VGG-19 **~8%**
- Microsoft ResNet less than **4%** error in 2015 with much more layers

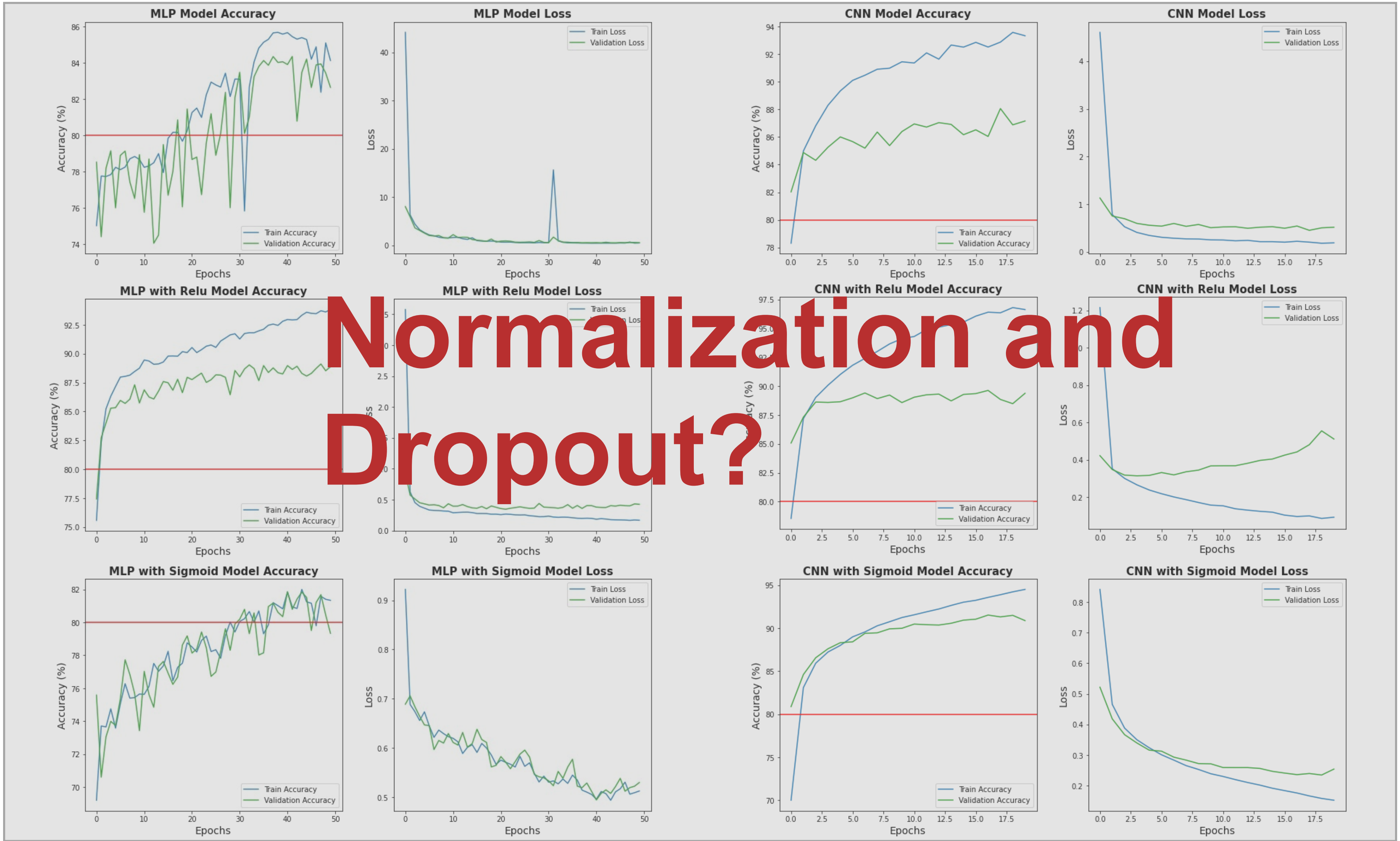
LifeClef

BirdCLEF 2023
Identify bird calls in soundscapes
$50,000 Prize

Classification error % (top-5)



Comparison



MLP Model Accuracy

MLP Model Loss

CNN Model Accuracy

CNN Model Loss

MLP with Relu Model Accuracy

MLP with Relu Model Loss

CNN with Relu Model Accuracy

CNN with Relu Model Loss

MLP with Sigmoid Model Accuracy

MLP with Sigmoid Model Loss

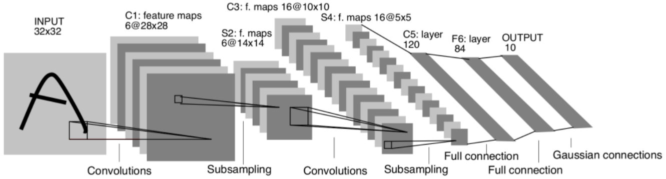
CNN with Sigmoid Model Accuracy

CNN with Sigmoid Model Loss

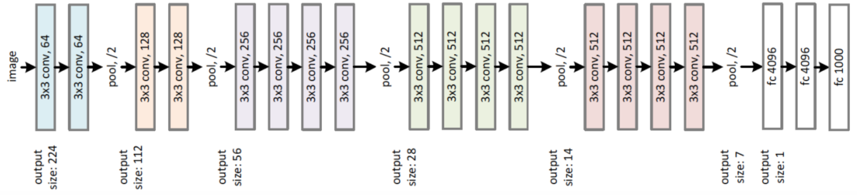
Normalization and Dropout?

Assignment 02_cnn

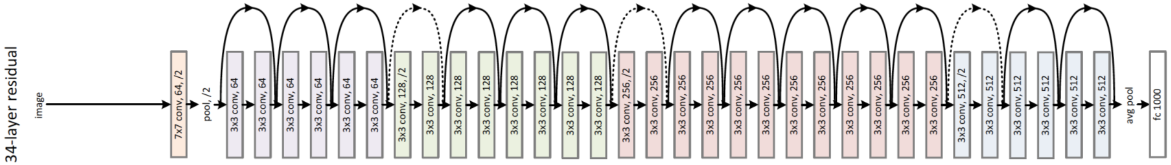
- Implement and train three convolutional networks
  1. CNN inspired by classical LeNet-5



  2. VGG-style network

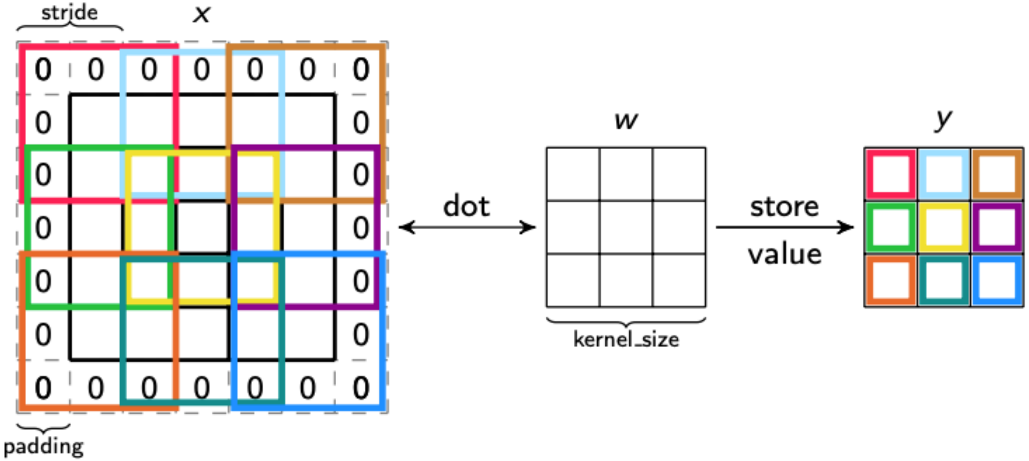


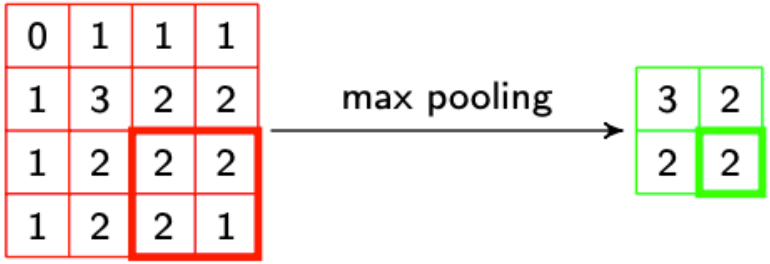
  3. ResNet



Common Terms

- `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding)`



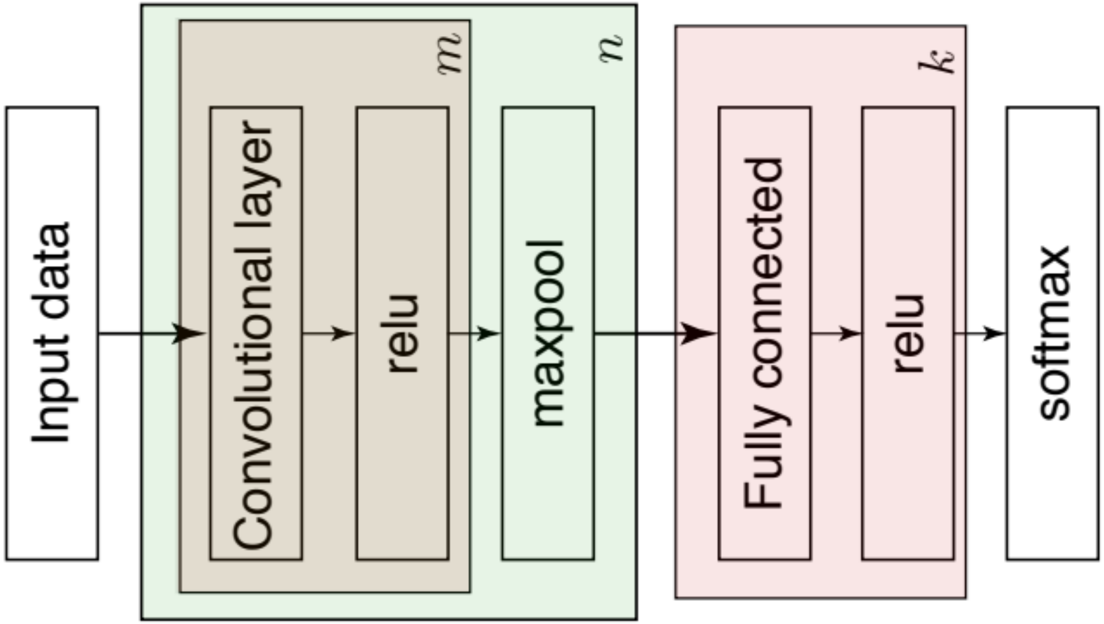


0	1	1	1
1	3	2	2
1	2	2	2
1	2	2	1

max pooling

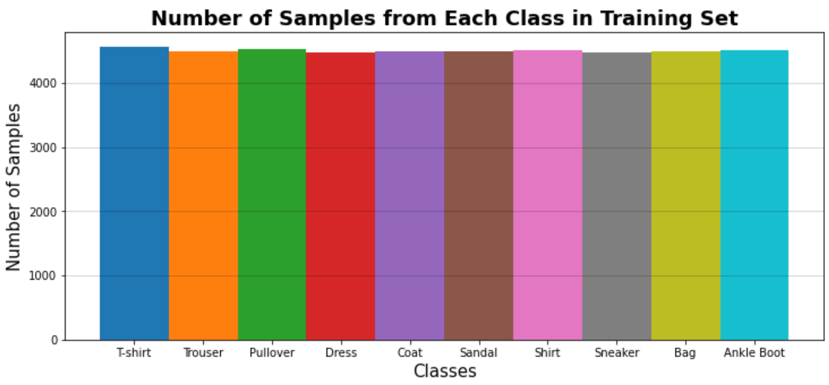
3	2
2	2

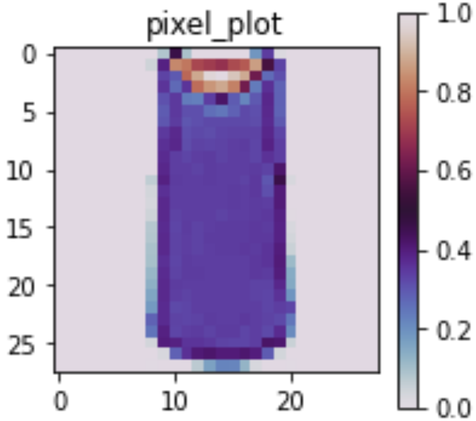
Common Architecture

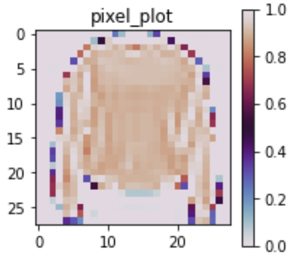


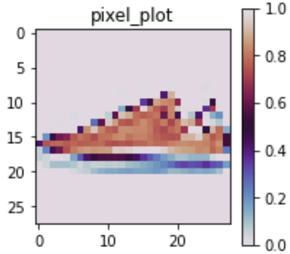
http://sharif.edu/~beigy/courses/14011/40719/Lect-8to10.pdf

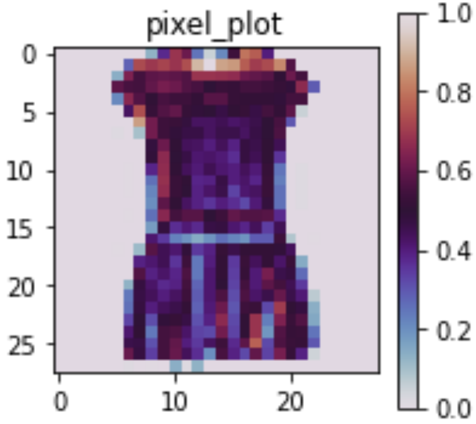
FashionMNIST

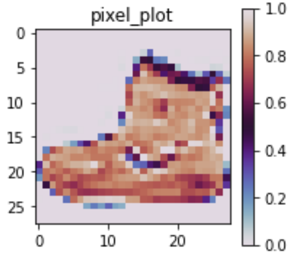




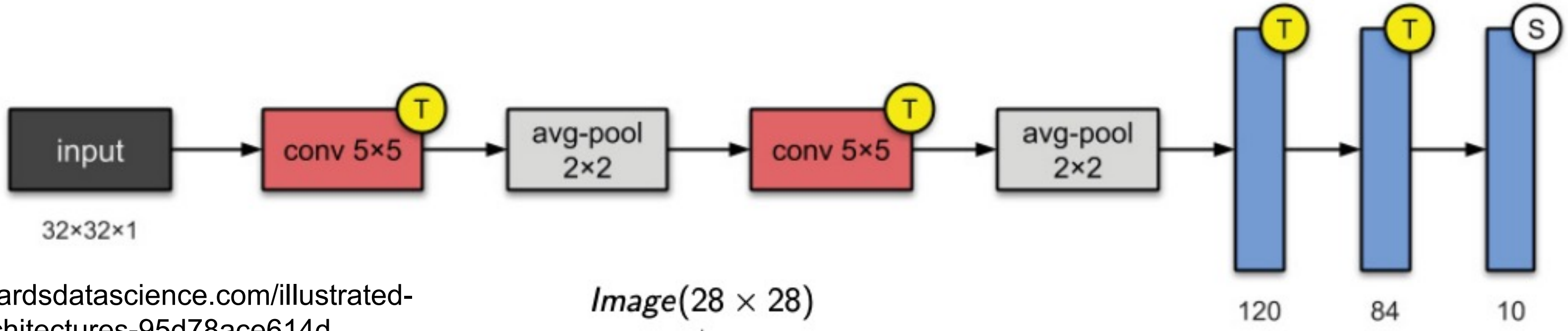




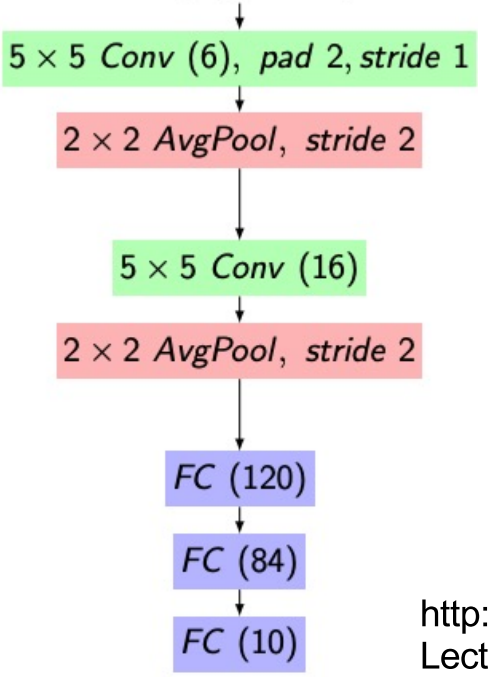




LeNet-5

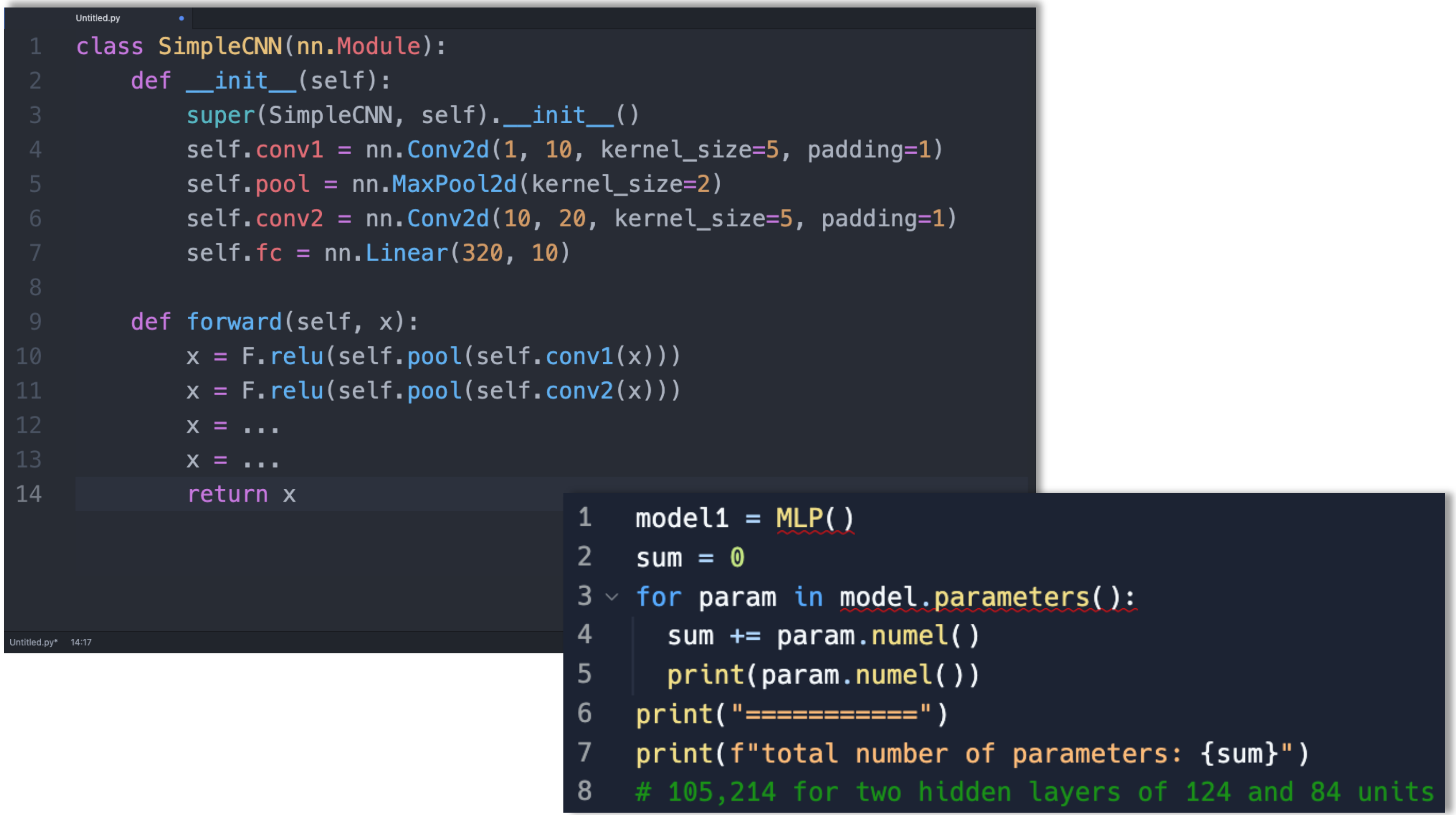


https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d


$Image(28 \times 28)$
5×5 $Conv$ (6), pad 2, $stride$ 1
2×2 $AvgPool$, $stride$ 2
5×5 $Conv$ (16)
2×2 $AvgPool$, $stride$ 2
FC (120)
FC (84)
FC (10)

http://sharif.edu/~beigy/courses/14011/40719/Lect-8to10.pdf

LeNet-5



```python
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5, padding=1)
        self.fc = nn.Linear(320, 10)

    def forward(self, x):
        x = F.relu(self.pool(self.conv1(x)))
        x = F.relu(self.pool(self.conv2(x)))
        x = ...
        x = ...
        return x
```

```python
model1 = MLP()
sum = 0
for param in model.parameters():
  sum += param.numel()
  print(param.numel())
print("===========")
print(f"total number of parameters: {sum}")
# 105,214 for two hidden layers of 124 and 84 units
```

Aalto-yliopisto
Aalto-universitetet
Aalto University

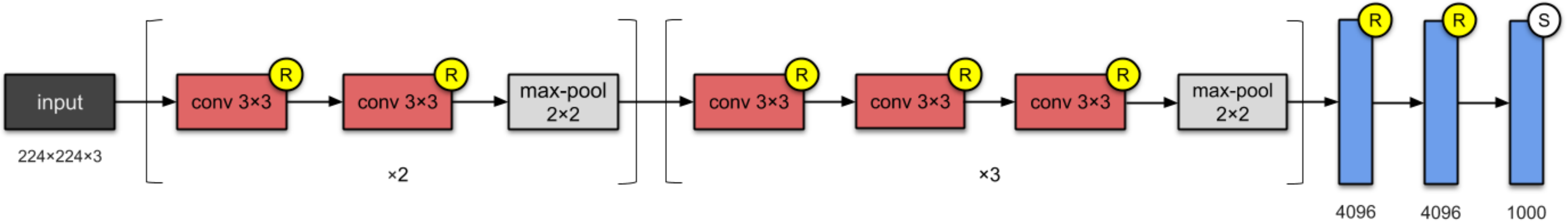
LeNet-5

In the first exercise, your task is to create a convolutional neural network with the architecture inspired by the classical LeNet-5 (LeCun et al., 1998).

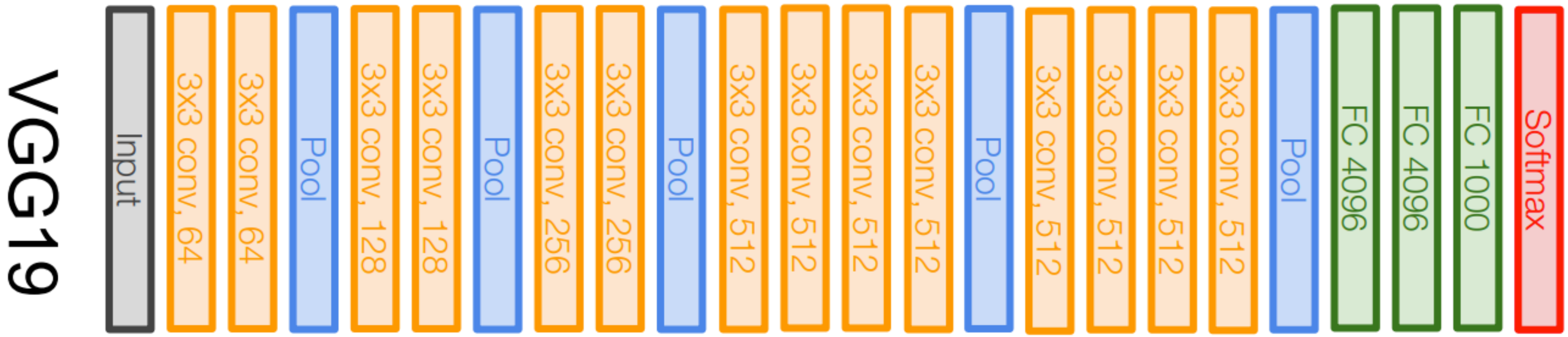
The architecture of the convolutional network that you need to create:

- 2d convolutional layer with:
  - one input channel
  - 6 output channels
  - kernel size 5 (no padding)
  - followed by ReLU
- Max-pooling layer with kernel size 2 and stride 2
- 2d convolutional layer with:
  - 16 output channels
  - kernel size 5 (no padding)
  - followed by ReLU
- Max-pooling layer with kernel size 2 and stride 2
- A fully-connected layer with:
  - 120 outputs
  - followed by ReLU
- A fully-connected layer with:
  - 84 outputs
  - followed by ReLU
- A fully-connected layer with 10 outputs and without nonlinearity.

VGG style network

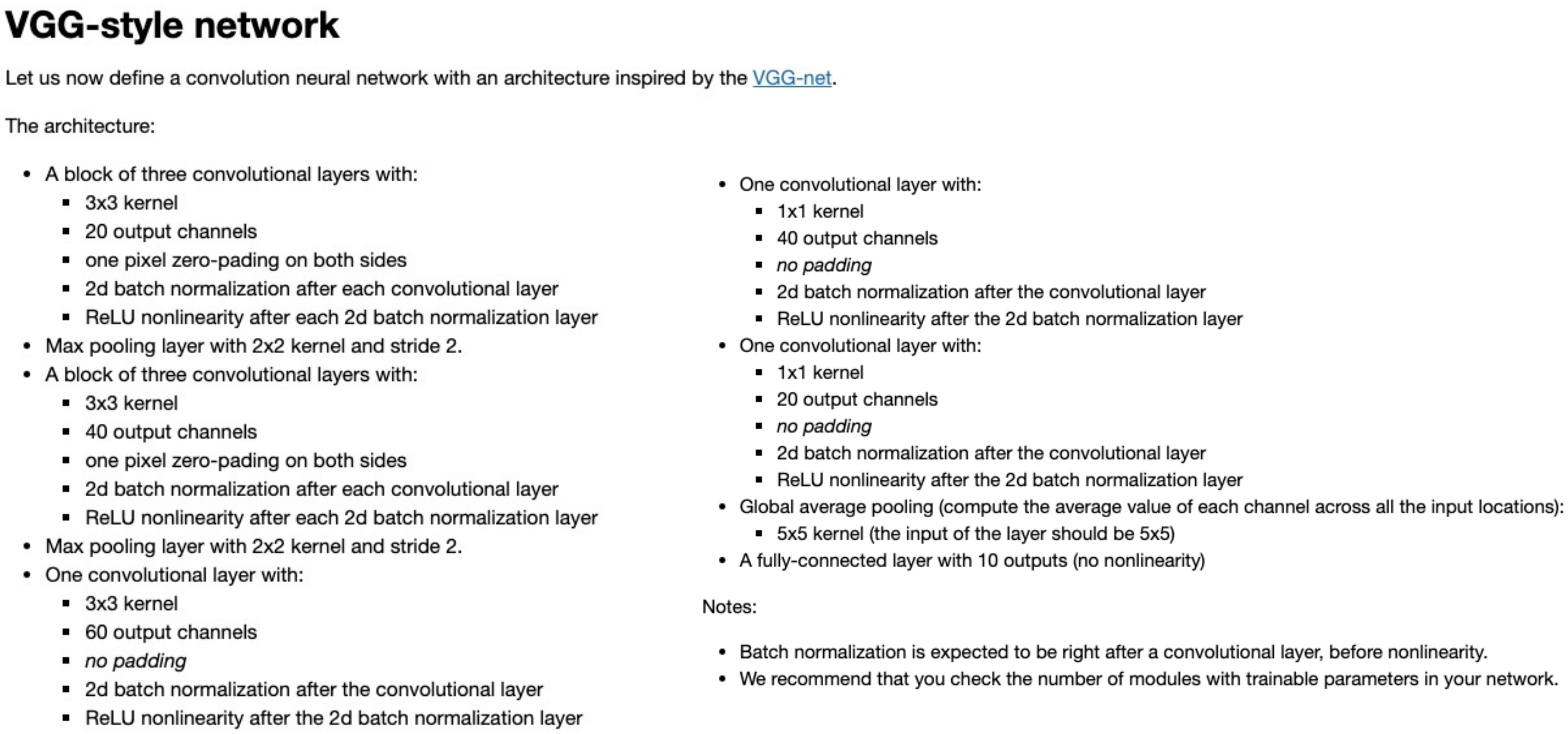


https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d



http://sharif.edu/~beigy/courses/14011/40719/Lect-8to10.pdf

VGG style network



VGG-style network

Let us now define a convolution neural network with an architecture inspired by the VGG-net.

The architecture:

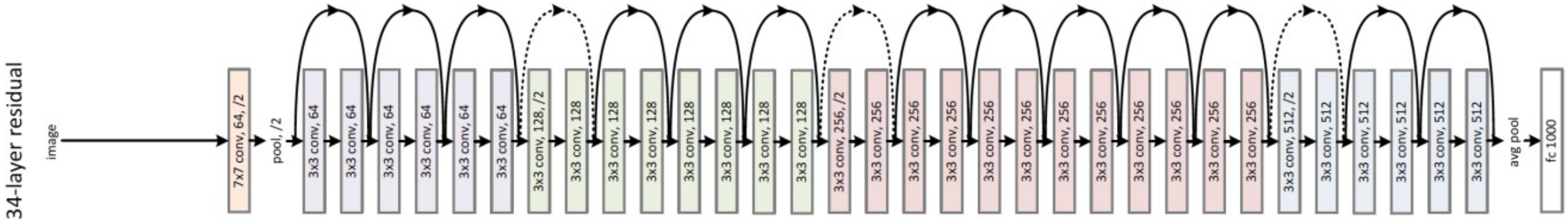
- A block of three convolutional layers with:
  - 3x3 kernel
  - 20 output channels
  - one pixel zero-pading on both sides
  - 2d batch normalization after each convolutional layer
  - ReLU nonlinearity after each 2d batch normalization layer
- Max pooling layer with 2x2 kernel and stride 2.
- A block of three convolutional layers with:
  - 3x3 kernel
  - 40 output channels
  - one pixel zero-pading on both sides
  - 2d batch normalization after each convolutional layer
  - ReLU nonlinearity after each 2d batch normalization layer
- Max pooling layer with 2x2 kernel and stride 2.
- One convolutional layer with:
  - 3x3 kernel
  - 60 output channels
  - *no padding*
  - 2d batch normalization after the convolutional layer
  - ReLU nonlinearity after the 2d batch normalization layer
- One convolutional layer with:
  - 1x1 kernel
  - 40 output channels
  - *no padding*
  - 2d batch normalization after the convolutional layer
  - ReLU nonlinearity after the 2d batch normalization layer
- One convolutional layer with:
  - 1x1 kernel
  - 20 output channels
  - *no padding*
  - 2d batch normalization after the convolutional layer
  - ReLU nonlinearity after the 2d batch normalization layer
- Global average pooling (compute the average value of each channel across all the input locations):
  - 5x5 kernel (the input of the layer should be 5x5)
- A fully-connected layer with 10 outputs (no nonlinearity)

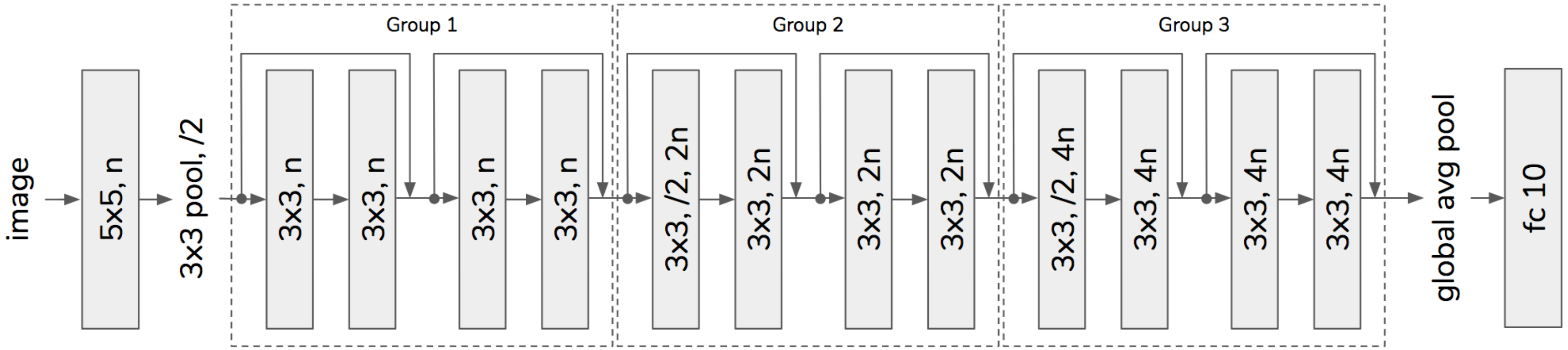
Notes:

- Batch normalization is expected to be right after a convolutional layer, before nonlinearity.
- We recommend that you check the number of modules with trainable parameters in your network.

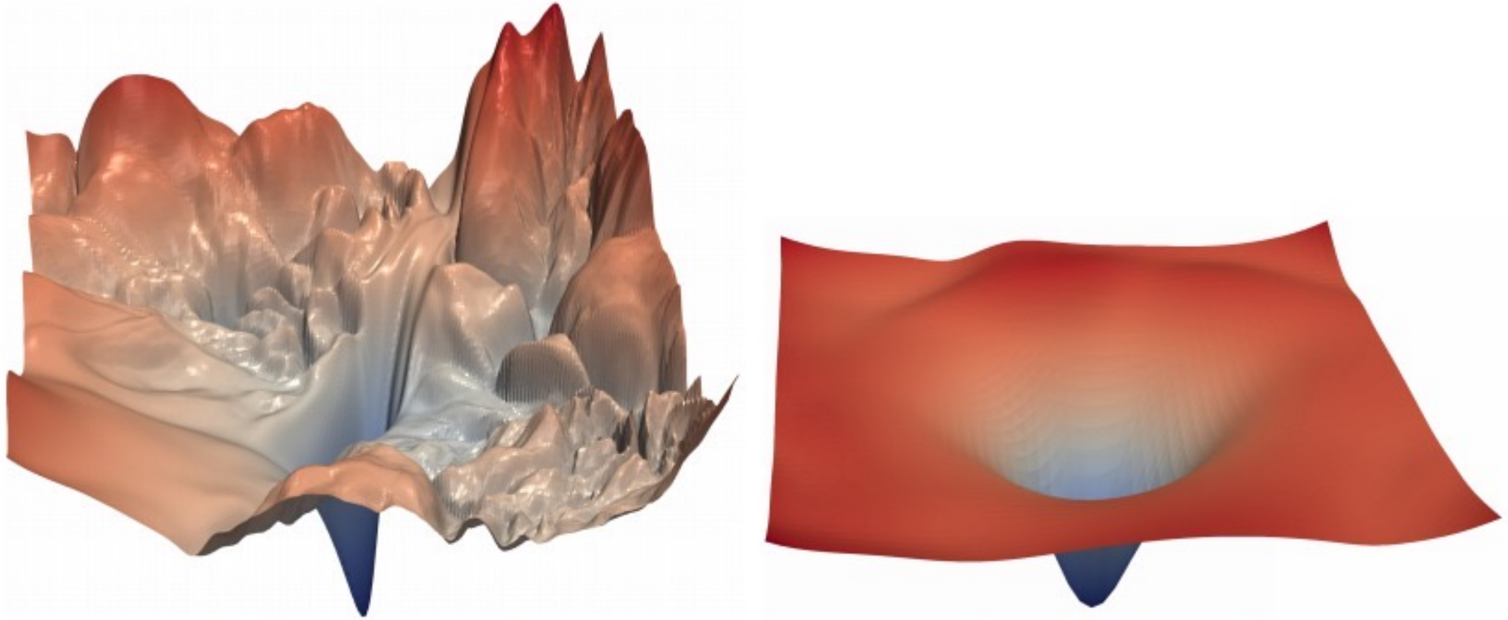
ResNet

- ResNet:
  - Instead of learning $f(\mathbf{x})$, layers learn $\mathbf{x} + h(\mathbf{x})$.
  - He et al., (2016): If an identity mapping is optimal, it might be easier to push residual $h(\mathbf{x})$ to zero than to learn an identity mapping with $f(\mathbf{x})$.





ResNet



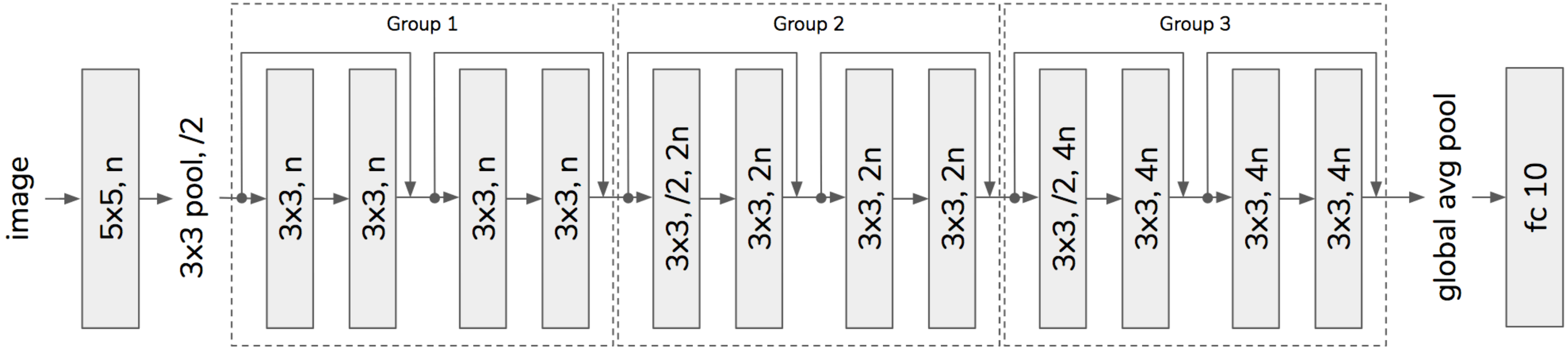
(a) without skip connections

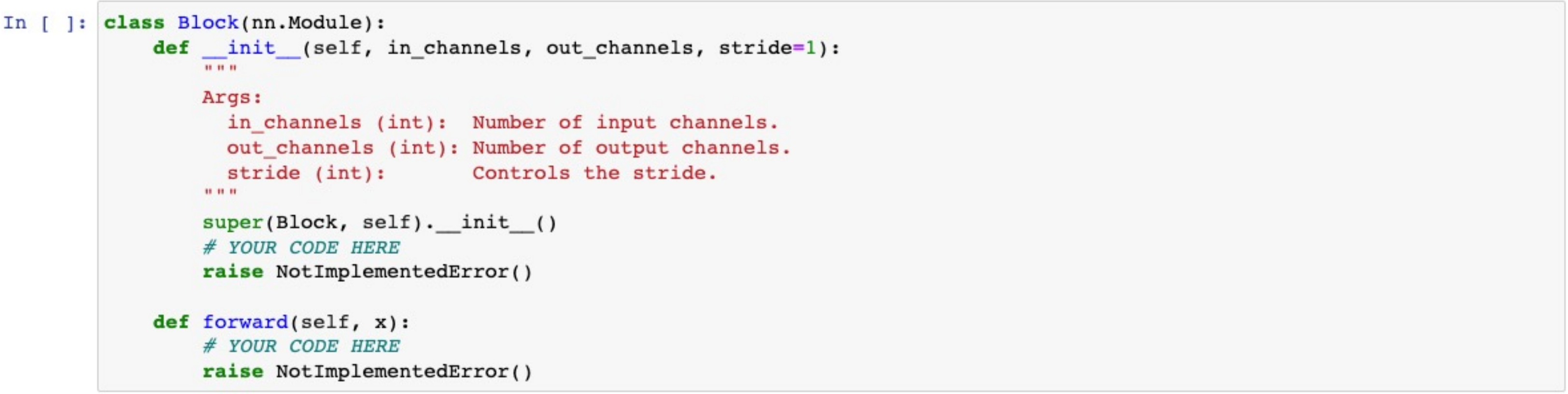
(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

https://doi.org/10.48550/arXiv.1712.09913

ResNet





```
In [ ]: class Block(nn.Module):
            def __init__(self, in_channels, out_channels, stride=1):
                """
                Args:
                  in_channels (int):  Number of input channels.
                  out_channels (int): Number of output channels.
                  stride (int):       Controls the stride.
                """
                super(Block, self).__init__()
                # YOUR CODE HERE
                raise NotImplementedError()

            def forward(self, x):
                # YOUR CODE HERE
                raise NotImplementedError()
```

Thank You

Aalto University