Lecture 9 Recap
What are Convolutions?

$$f * g = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

- $f = \text{red}$
- $g = \text{blue}$
- $f * g = \text{green}$

Convolution of two box functions

Convolution of two Gaussians

Application of a filter to a function

the ‘smaller’ one is typically called the filter kernel
What are Convolutions?

Discrete case: box filter

<table>
<thead>
<tr>
<th>4</th>
<th>3</th>
<th>2</th>
<th>-5</th>
<th>3</th>
<th>5</th>
<th>2</th>
<th>5</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What to do at boundaries?

1) Shrink

| 3 | 0 | 0 | 1 | 10/3 | 4 | 4 | 16/3 |

2) Pad often '0'

| 7/3 | 3 | 0 | 0 | 1 | 10/3 | 4 | 4 | 16/3 | 11/3 |
## Convolutions on Images

### Image 5x5

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>3</td>
<td>2</td>
<td>-5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

### Kernel 3x3

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Output 3x3

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>-7</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>-5</td>
<td>-9</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
5 \cdot 4 + (-1) \cdot 3 + (-1) \cdot 4 + (-1) \cdot 9 + (-1) \cdot 1 = 20 - 17 = 3
\]
Image Filters

- Each kernel gives us a different image filter

Input

<table>
<thead>
<tr>
<th>Edge detection</th>
<th>Box mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-1\ -1\ -1|</td>
<td></td>
</tr>
<tr>
<td>(-1\ 8\ -1)</td>
<td></td>
</tr>
<tr>
<td>(-1\ -1\ -1)</td>
<td></td>
</tr>
<tr>
<td>(1/9)</td>
<td>(1/1)</td>
</tr>
<tr>
<td>(1)</td>
<td>(1)</td>
</tr>
<tr>
<td>(1)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sharpen</th>
<th>Gaussian blur</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0\ -1\ 0)</td>
<td></td>
</tr>
<tr>
<td>(-1\ 5\ -1)</td>
<td></td>
</tr>
<tr>
<td>(0\ -1\ 0)</td>
<td></td>
</tr>
<tr>
<td>(1/16)</td>
<td>(1/1)</td>
</tr>
<tr>
<td>(1)</td>
<td>(1)</td>
</tr>
<tr>
<td>(2)</td>
<td>(2)</td>
</tr>
<tr>
<td>(1)</td>
<td>(1)</td>
</tr>
</tbody>
</table>

LET'S LEARN THESE FILTERS!
Convolutions on RGB Images

32×32×3 image (pixels $x$)

5×5×3 filter (weights $w$)

slide over all spatial locations and compute all output $z_i$.

w/o padding, there are 28×28 locations

activation map (also feature map)
Let's apply a different filter with different weights!
Let's apply **five** filters, each with different weights!
## Convolution Layers: Dimensions

<table>
<thead>
<tr>
<th>Input width of N</th>
<th>Filter width of F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N = 7, F = 3, S = 1: ( \frac{7-3}{1} + 1 = 5 )</td>
</tr>
<tr>
<td></td>
<td>N = 7, F = 3, S = 2: ( \frac{7-3}{2} + 1 = 3 )</td>
</tr>
<tr>
<td></td>
<td>N = 7, F = 3, S = 3: ( \frac{7-3}{3} + 1 = 2.3333 )</td>
</tr>
</tbody>
</table>

Input: \( N \times N \)  
Filter: \( F \times F \)  
Stride: \( S \)  
Output: \( \left( \frac{N-F}{S} + 1 \right) \times \left( \frac{N-F}{S} + 1 \right) \)

Fractions are illegal.
Convolution Layers: Padding

Types of convolutions:

- **Valid** convolution: using no padding
  
- **Same** convolution: output = input size

Set padding to $P = \frac{F-1}{2}$
Convolution Layers: Dimensions

Remember: \[ \text{Output} = \left( \frac{N + 2 \cdot P - F}{S} + 1 \right) \times \left( \frac{N + 2 \cdot P - F}{S} + 1 \right) \]

**REMARK:** in practice, typically **integer division** is used (i.e., apply the floor-operator!)

Example: 3x3 conv with same padding and strides of 2 on an 64x64 RGB image \( \rightarrow N = 64, F = 3, P = 1, S = 2 \)

\[
\text{Output: } \left( \frac{64 + 2 \cdot 1 - 3}{2} + 1 \right) \times \left( \frac{64 + 2 \cdot 1 - 3}{2} + 1 \right) = \text{floor}(32.5) \times \text{floor}(32.5) = 32 \times 32
\]
CNN Learned Filters

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
CNN Prototype

Slide by Karpathy
Pooling Layer: Max Pooling

Single depth slice of input

Max pool with 2x2 filters and stride 2

‘Pooled’ output

I2DL: Prof. Niessner, Prof. Leal-Taixé
Receptive Field

• Spatial extent of the connectivity of a convolutional filter

7x7 input

5x5 receptive field on the original input: one output value is connected to 25 input pixels

3x3 output
Lecture 10 – CNNs (part 2)
Classic Architectures
LeNet

- Digit recognition: 10 classes

Input: 32×32 grayscale images
This one: Labeled as class “7”
LeNet

• Digit recognition: 10 classes

• Valid convolution: size shrinks
• How many conv filters are there in the first layer? 6
LeNet

- Digit recognition: 10 classes

- At that time average pooling was used, now max pooling is much more common
LeNet

- Digit recognition: 10 classes

• Again valid convolutions, how many filters?
LeNet

• Digit recognition: 10 classes

• Use of tanh/sigmoid activations → not common now!
LeNet

- Digit recognition: 10 classes

- Conv -> Pool -> Conv -> Pool -> Conv -> FC
LeNet

- Digit recognition: 10 classes

- Conv -> Pool -> Conv -> Pool -> Conv -> FC

- As we go deeper: Width, Height ↓ Number of Filters ↑

60k parameters
Test Benchmarks

- **ImageNet Dataset:**
  ImageNet Large Scale Visual Recognition Competition (ILSVRC)


I2DL: Prof. Niessner, Prof. Leal-Taixé
Common Performance Metrics

• **Top-1 score**: check if a sample’s top class (i.e. the one with highest probability) is the same as its target label

• **Top-5 score**: check if your label is in your 5 first predictions (i.e. predictions with 5 highest probabilities)

• → **Top-5 error**: percentage of test samples for which the correct class was not in the top 5 predicted classes
AlexNet

- Cut ImageNet error down in half

ILSVRC top-5 error on ImageNet

I2DL: Prof. Niessner, Prof. Leal-Taixé
AlexNet

- First filter with stride 4 to reduce size significantly
- 96 filters

[Krizhevsky et al. NIPS’12] AlexNet
AlexNet

- Use of same convolutions
- As with LeNet: Width, Height ↓ Number of Filters ↑

[Krizhevsky et al. NIPS’12] AlexNet
AlexNet

[Krizhevsky et al. NIPS’12] AlexNet
AlexNet

- Softmax for 1000 classes

[Krizhevsky et al. NIPS’12] AlexNet
AlexNet

• Similar to LeNet but much bigger (~1000 times)

• Use of ReLU instead of tanh/sigmoid

60M parameters

[Krizhevsky et al. NIPS'12] AlexNet
VGGNet

• Striving for simplicity

• CONV = 3x3 filters with stride 1, same convolutions

• MAXPOOL = 2x2 filters with stride 2

[Simonyan and Zisserman ICLR’15] VGGNet
VGGNet

Conv=3x3, s=1, same
Maxpool=2x2, s=2

[Simonyan and Zisserman ICLR'15] VGGNet
VGGNet

Conv=3x3, s=1, same
Maxpool=2x2, s=2

[Simonyan and Zisserman ICLR’15] VGGNet
VGGNet

Conv=3x3, s=1, same
Maxpool=2x2, s=2

[Simonyan and Zisserman ICLR’15] VGGNet
VGGNet

Conv=3x3, s=1, same
Maxpool=2x2, s=2

[Simonyan and Zisserman ICLR’15] VGGNet
VGGNet

Conv=3x3, s=1, same
Maxpool=2x2, s=2

[Simonyan and Zisserman ICLR'15] VGGNet
VGGNet

- Conv -> Pool -> Conv -> Pool -> Conv -> FC
- As we go deeper: Width, Height ↓ Number of Filters ↑

- Called VGG-16: 16 layers that have weights

138M parameters

- Large but simplicity makes it appealing

[Simonyan and Zisserman ICLR’15] VGGNet
VGGNet

- A lot of architectures were analyzed

[Simonyan and Zisserman 2014]

I2DL: Prof. Niessner, Prof. Leal-Taixé
Skip Connections
The Problem of Depth

• As we add more and more layers, training becomes harder

• Vanishing and exploding gradients

• How can we train very deep nets?
Residual Block

- Two layers

\[ x^{L-1} \xrightarrow{\text{Linear}} x^L \xrightarrow{\text{Non-linearity}} x^{L+1} \]

Input: \( W^L x^{L-1} + b^L \)

\[ x^L = f(W^L x^{L-1} + b^L) \]

\[ x^{L+1} = f(W^{L+1} x^L + b^{L+1}) \]
Residual Block

- Two layers

\[ x^{L-1} \rightarrow x^L \rightarrow x^{L+1} \]

Skip connection

Input \[\rightarrow\] Linear \[\rightarrow\] Linear

Main path
Residual Block

- Two layers

\[ x_{L-1} \xrightarrow{\text{Linear}} x^L \xrightarrow{\text{Linear}} x_{L+1} \]

\[ x_{L+1} = f(W^{L+1}x^L + b^{L+1} + x_{L-1}) \]

\[ x_{L+1} = f(W^{L+1}x^L + b^{L+1}) \]
Residual Block

- Two layers

- Usually use a same convolution since we need same dimensions

- Otherwise we need to convert the dimensions with a matrix of learned weights or zero padding
ResNet Block

Plain Net

Any two stacked layers

Residual Net

Identity

[He et al. CVPR’16] ResNet

I2DL: Prof. Niessner, Prof. Leal-Taixé
ResNet

- Xavier/2 initialization
- SGD + Momentum (0.9)
- Learning rate 0.1, divided by 10 when plateau
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout

ResNet-152: 60M parameters

[He et al. CVPR'16] ResNet
ResNet

- If we make the network deeper, at some point performance starts to degrade
ResNet

- If we make the network deeper, at some point performance starts to degrade
Why do ResNets Work?

- How is this block really affecting me?
Why do ResNets Work?

\[ x^{L+1} = f(W^{L+1}x^L + b^{L+1} + x^{L-1}) \]

\[ \sim \text{zero} \]

\[ x^{L+1} = f(x^{L-1}) \]
Why do ResNets Work?

- We kept the same values and added a non-linearity

\[ x^{L+1} = f(x^{L-1}) \]
Why do ResNets Work?

- The identity is easy for the residual block to learn
- Guaranteed it will not hurt performance, can only improve
1x1 Convolutions
Recall: Convolutions on Images

Image 5x5:

<table>
<thead>
<tr>
<th>-5</th>
<th>3</th>
<th>2</th>
<th>-5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>-5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

Kernel 3x3:

<table>
<thead>
<tr>
<th>0</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>

Output 3x3:

\[
5 \cdot 3 + (-1) \cdot 3 + (-1) \cdot 2 + (-1) \cdot 0 + (-1) \cdot 4 = 15 - 9 = 6
\]
### 1x1 Convolution

What is the output size?

<table>
<thead>
<tr>
<th>Image 5x5</th>
<th>5</th>
<th>3</th>
<th>2</th>
<th>-5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

**Kernel 1x1**

2
1x1 Convolution

Image 5x5

Kernel 1x1

\[-5 \times 2 = -10\]
1x1 Convolution

Image 5x5

<table>
<thead>
<tr>
<th></th>
<th>-5</th>
<th>3</th>
<th>2</th>
<th>-5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

Kernel 1x1

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

\[-1 \times 2 = -2\]
### 1x1 Convolution

#### Image 5x5

<table>
<thead>
<tr>
<th></th>
<th>-5</th>
<th>3</th>
<th>2</th>
<th>-5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>-10</th>
<th>6</th>
<th>4</th>
<th>-10</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>-6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>14</td>
<td>18</td>
<td>-2</td>
<td></td>
</tr>
</tbody>
</table>

- 1x1 kernel: keeps the dimensions and scales input
1x1 Convolution

- Same as having a 3 neuron fully connected layer
• As always we use more convolutional filters
Using 1x1 Convolutions

- Use it to shrink the number of channels
- Further adds a non-linearity → one can learn more complex functions
Inception Layer
Inception Layer

• Tired of choosing filter sizes?
• Use them all!
• All same convolutions
• 3x3 max pooling is with stride 1
Inception Layer

• Possible size of the output

• Not sustainable!
Inception Layer: Computational Cost

92 Conv 5x5x200 + ReLU

Multiplications: 5x5x200 \times 32x32x92 \approx 470 \text{ million}

1 value of the output volume
Inception Layer: Computational Cost

Multiplications: $1 \times 1 \times 200 \times 32 \times 32 \times 16$

$5 \times 5 \times 16 \times 32 \times 32 \times 92$

$\sim 40$ million

Reduction of multiplications by $1/10$
Inception Layer

(a) Inception module, naive version

(b) Inception module with dimensionality reduction

[Szegedy et al. CVPR'15] GoogLeNet

I2DL: Prof. Niessner, Prof. Leal-Taixé
Inception Layer: Dimensions

We do not want max pool result to take up almost all the output

<table>
<thead>
<tr>
<th>Layer</th>
<th>Dimensions</th>
<th>Convolution Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous layer</td>
<td>28x28x192</td>
<td></td>
</tr>
<tr>
<td>1x1 convolutions</td>
<td>28x28x192</td>
<td>3x3 max pooling</td>
</tr>
<tr>
<td>1x1 convolutions</td>
<td>28x28x16</td>
<td>1x1 convolutions</td>
</tr>
<tr>
<td>3x3 convolutions</td>
<td>28x28x96</td>
<td>1x1 convolutions</td>
</tr>
<tr>
<td>3x3 convolutions</td>
<td>28x28x128</td>
<td>3x3 convolutions</td>
</tr>
<tr>
<td>Filter concatenation</td>
<td>28x28x256</td>
<td></td>
</tr>
</tbody>
</table>

[GoogLeNet] Szegedy et al. CVPR'15
GoogLeNet: Using the Inception Layer

Extra max pool layers to reduce dimensionality

[Szegedy et al CVPR’15] GoogLeNet
Xception Net

• „Extreme version of Inception“: applying (modified) Depthwise Separable Convolutions instead of normal convolutions
• 36 conv layers, structured into several modules with skip connections
• outperforms Inception Net V3
Depth-wise separable convolutions

Normal convolutions act on all channels.

I2DL: Prof. Niessner, Prof. Leal-Taixé
Depth-wise separable convolutions

Filters are applied only at certain depths of the features. Normal convolutions have groups set to 1, the convolutions used in this image have groups set to 3.

```python
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, groups=3)

class torch.nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride=1, padding=0, groups=3)
```
Depth-wise separable convolutions

```
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, groups=3)
class torch.nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride=1, padding=0, groups=3)
```

But the depth size is always the same!
Depth-wise separable convolutions

Solution: 1x1 convs!
But why?

Original convolution
256 kernels of size 5x5x3

Multiplications:
256x5x5x3 x (8x8 locations) = 1.228.800
But why?

**Original convolution**
256 kernels of size 5x5x3

Multiplications:
256x5x5x3 x (8x8 locations) = 1,228,800

**Depth-wise convolution**
3 kernels of size 5x5x1

Multiplications:
5x5x3 x (8x8 locations) = 4,800

**1x1 convolution**
256 kernels of size 1x1x3

Multiplications:
256x1x1x3 x (8x8 locations) = 49,152

I2DL: Prof. Niessner, Prof. Leal-Taixé
ImageNet Benchmark

ImageNet Classification top-5-error (%)

ILSVRC 2010
ILSVRC 2011
AlexNet (ILSVRC 2012)
ZF (ILSVRC 2013)
VGG (ILSVRC 2014)
GoogLeNet (ILSVRC 2014)
Xception (2016)
ResNet (ILSVRC 2015)
Trimp-Soushen (ILSVRC 2016)
SENet (ILSVRC 2017)

28.2
25.8
16.4
11.7
7.3
6.66
5.5
3.57
2.99
2.25

Shallow
8 Layers
8 Layers
19 Layers
22 Layers
36 Layers
Revolution of Depth

I2DL: Prof. Niessner, Prof. Leal-Taixé
Fully Convolutional Network
Classification Network

convolution

fully connected

227 × 227  55 × 55  27 × 27  13 × 13

“tabby cat”
FCN: Becoming Fully Convolutional

Convert fully connected layers to convolutional layers!
FCN: Becoming Fully Convolutional

convolution

H × W  H/4 × W/4  H/8 × W/8  H/16 × W/16  H/32 × W/32
FCN: Upsampling Output

convolution

H × W  H/4 × W/4  H/8 × W/8  H/16 × W/16  H/32 × W/32  H × W
Semantic Segmentation (FCN)

How do we go back to the input size?

[Long and Shelhamer, 15] FCN
Types of Upsampling

• 1. Interpolation

I2DL: Prof. Niessner, Prof. Leal-Taixé
Types of Upsampling

- 1. Interpolation

Original image  x 10

Nearest neighbor interpolation  Bilinear interpolation  Bicubic interpolation

Image: Michael Guerzhoy

I2DL: Prof. Niessner, Prof. Leal-Taixé
Types of Upsampling

• 1. Interpolation

  Few artifacts
Types of Upsampling

• 2. Transposed conv

[A. Dosovitskiy, TPAMI 2017] “Learning to Generate Chairs, Tables and Cars with Convolutional Networks”
Types of Upsampling

• 2. Transposed convolution
  – Unpooling
  – Convolution filter (learned)
  – Also called up-convolution
    (never deconvolution)
Refined Outputs

• If one does a cascade of unpooling + conv operations, we get to the encoder-decoder architecture

• Even more refined: Autoencoders with skip connections (aka U-Net)
U-Net architecture: Each blue box is a multichannel feature map. Number of channels denoted at the top of the box. Dimensions at the top of the box. White boxes are the copied feature maps.
U-Net: Encoder

Left side: **Contraction Path** (Encoder)

- Captures context of the image
- Follows typical architecture of a CNN:
  - Repeated application of 2 unpadded 3x3 convolutions
  - Each followed by ReLU activation
  - 2x2 maxpooling operation with stride 2 for downsampling
  - At each downsampling step, # of channels is doubled

→ as before: Height, Width ↓  Depth: ↑

[Ronneberger et al. MICCAI’15] U-Net
U-Net: Decoder

Right Side: **Expansion Path** (Decoder):

- Upsampling to recover spatial locations for assigning class labels to each pixel
  - 2x2 up-convolution that halves number of input channels
  - **Skip Connections**: outputs of up-convolutions are concatenated with feature maps from encoder
  - Followed by 2 ordinary 3x3 convs
  - Final layer: 1x1 conv to map 64 channels to # classes

- Height, Width: ↑  Depth: ↓

[Ronneberger et al. MICCAI'15] U-Net
See you next time!
We highly recommend to read through these papers!

- **AlexNet** [Krizhevsky et al. 2012]
- **VGGNet** [Simonyan & Zisserman 2014]
- **ResNet** [He et al. 2015]
- **GoogLeNet** [Szegedy et al. 2014]
- **Xception** [Chollet 2016]
- **Fast R-CNN** [Girshick 2015]
- **U-Net** [Ronneberger et al. 2015]
- **EfficientNet** [Tan & Le 2019]