Foundations of Machine Learning (ECE 5984)

- Convolutional Neural Networks -

Eunbyung Park

Assistant Professor

School of Electronic and Electrical Engineering

Eunbyung Park (silverbottlep.github.io)

Convolution

1D Convolution

- Convolution is a mathematical operation on two functions (f , g) that produces a third function $f * g$

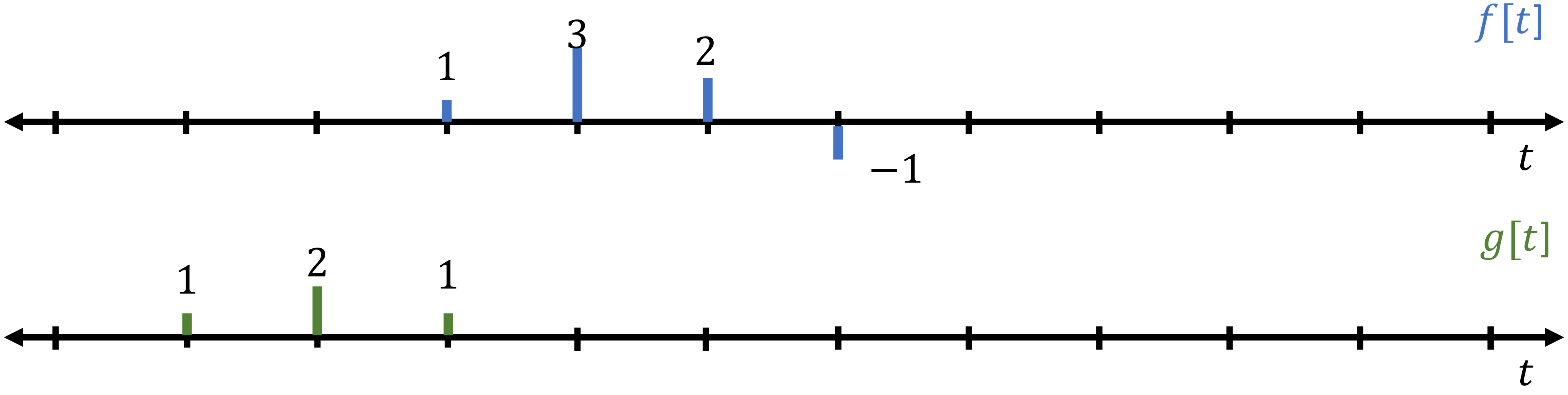
$$(f * g)(t) \coloneqq \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

$$(f * g)[t] \coloneqq \sum_{\tau} f[t-\tau]g[\tau]$$

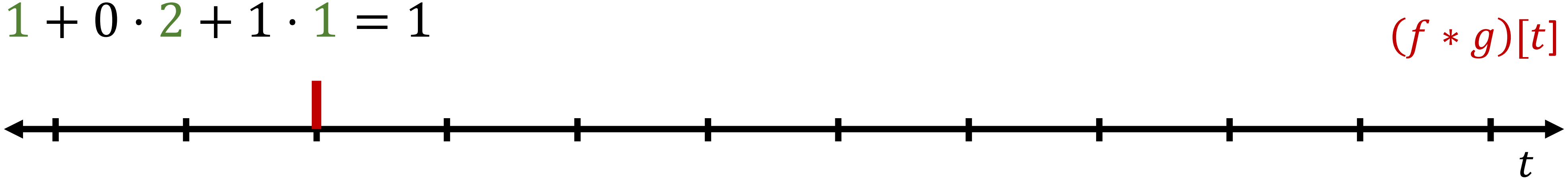
1D Convolution

$$(f * g)[t] \coloneqq \sum_{\tau} f[t-\tau]g[\tau]$$

- Flip the filter and sliding



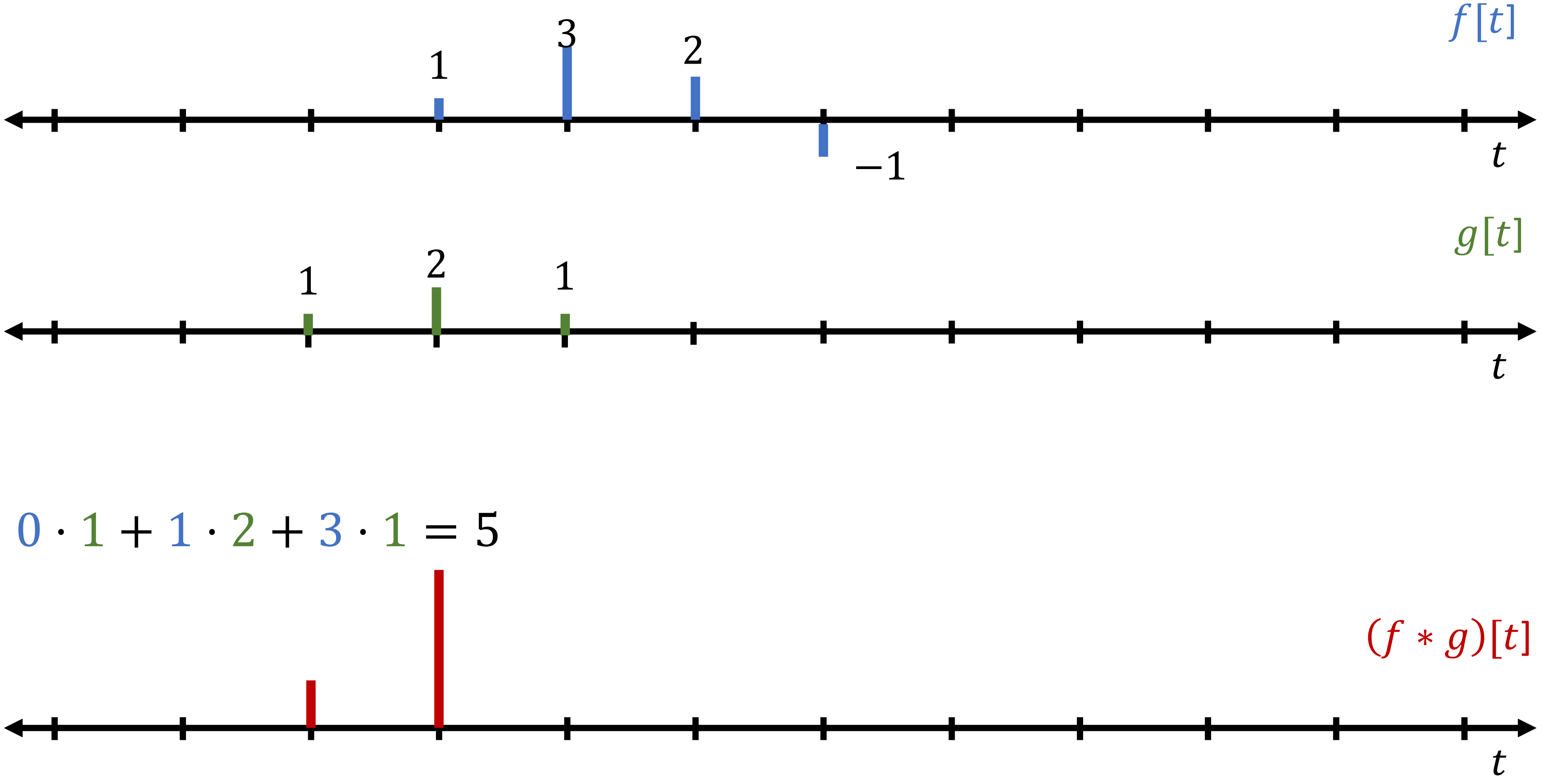
$0 \cdot 1 + 0 \cdot 2 + 1 \cdot 1 = 1$



1D Convolution

$$(f * g)[t] \coloneqq \sum_{\tau} f[t-\tau]g[\tau]$$

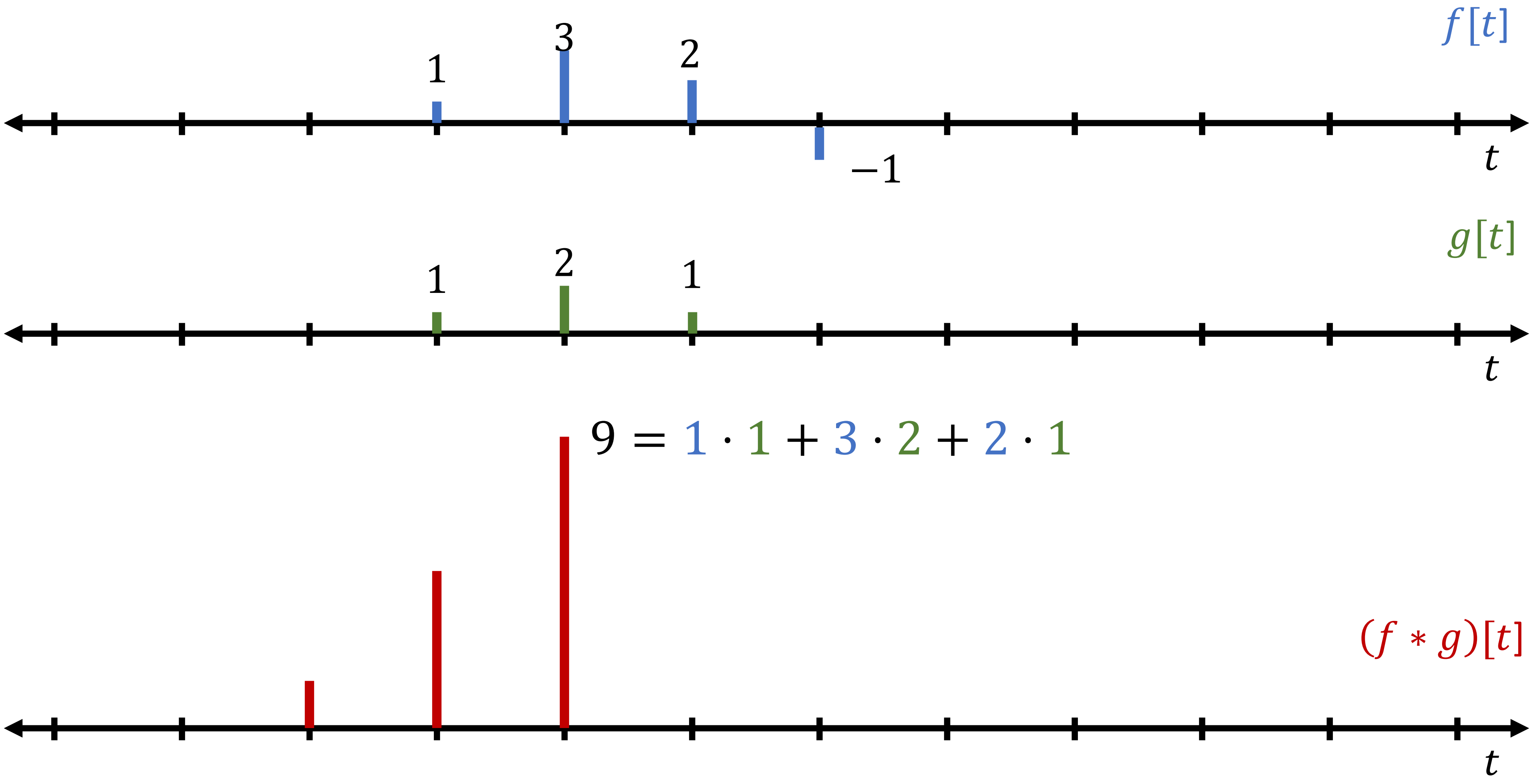
- Flip the filter and sliding



1D Convolution

$$(f * g)[t] \coloneqq \sum_{\tau} f[t-\tau]g[\tau]$$

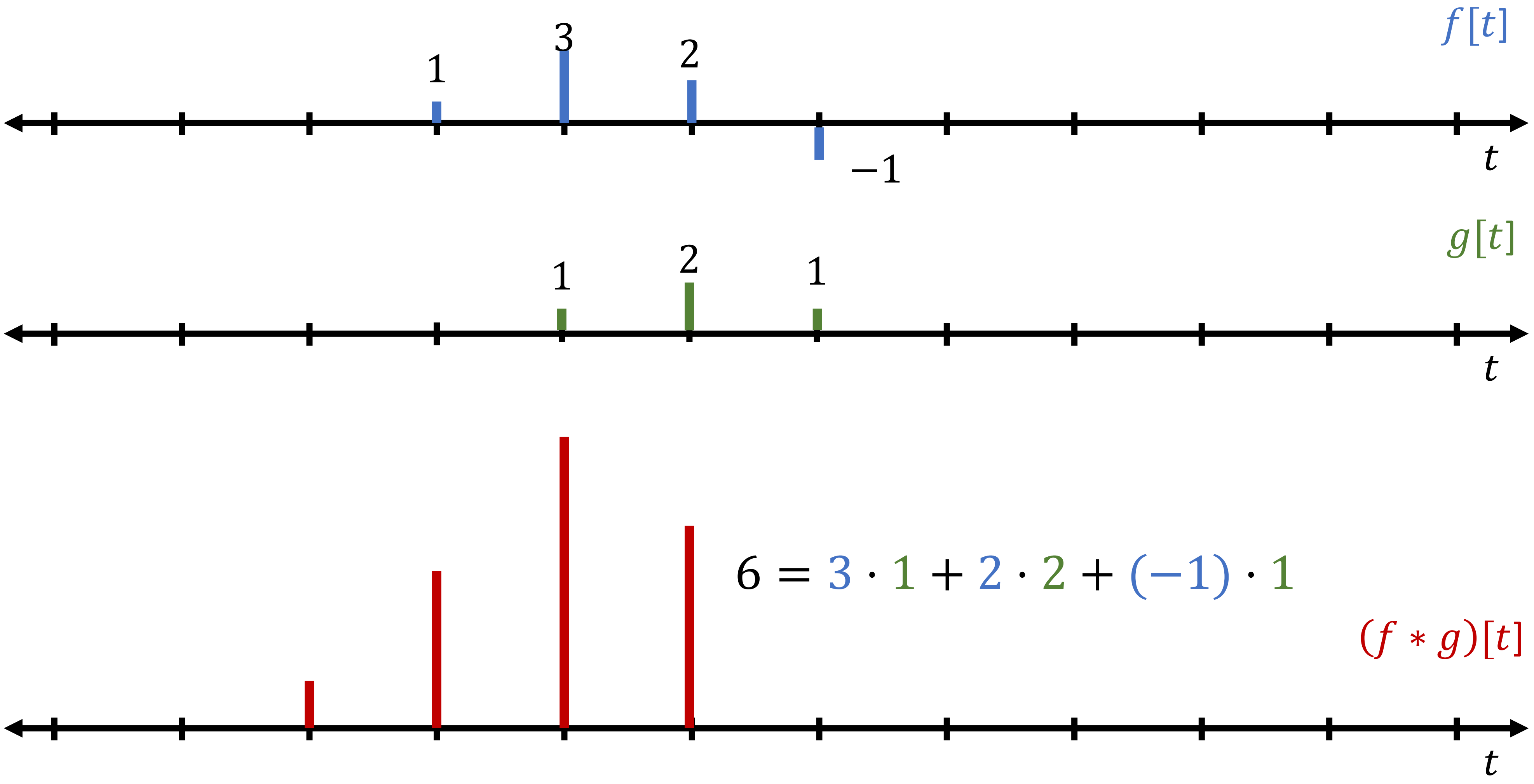
- Flip the filter and sliding



1D Convolution

$$(f * g)[t] \coloneqq \sum_{\tau} f[t-\tau]g[\tau]$$

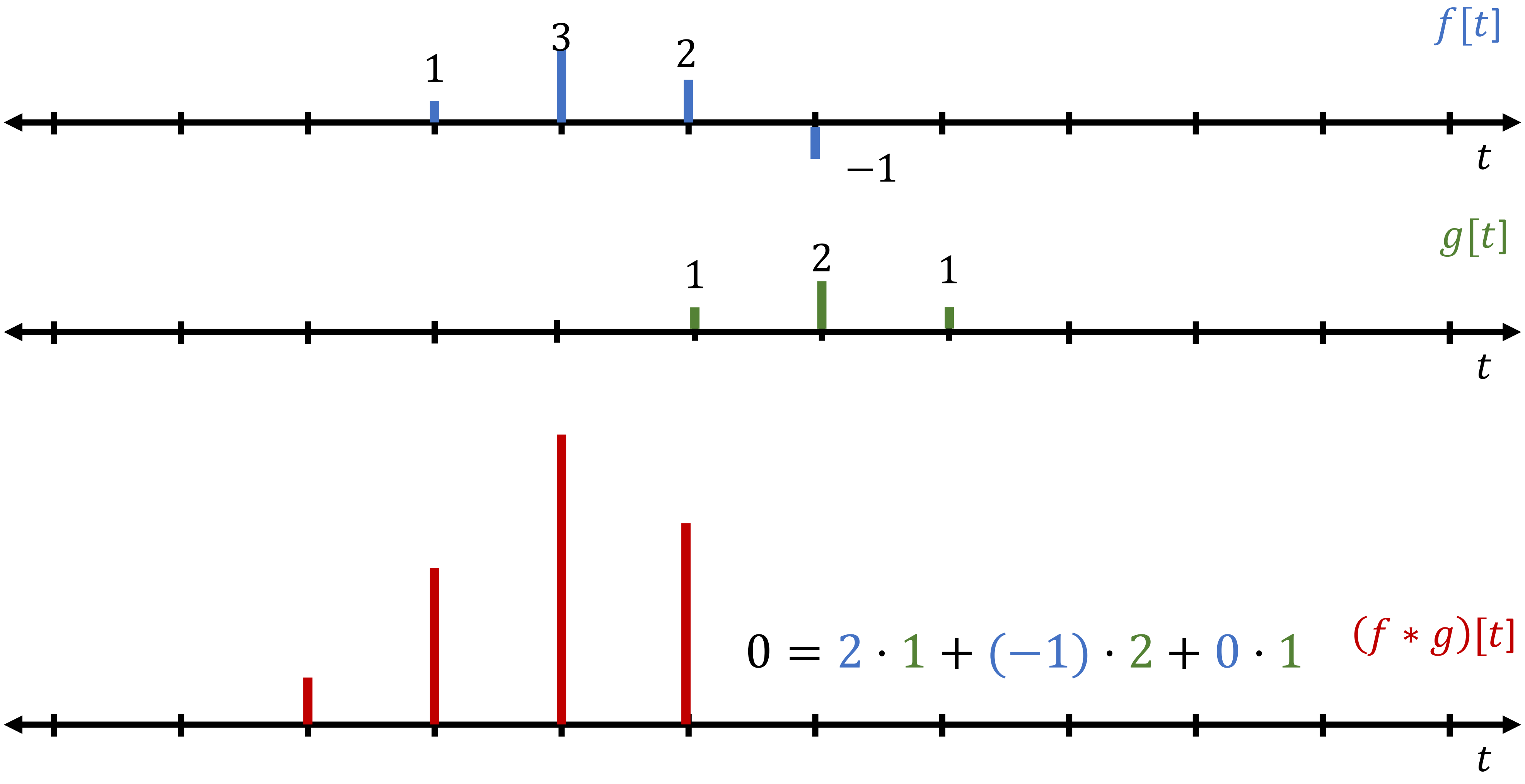
- Flip the filter and sliding



1D Convolution

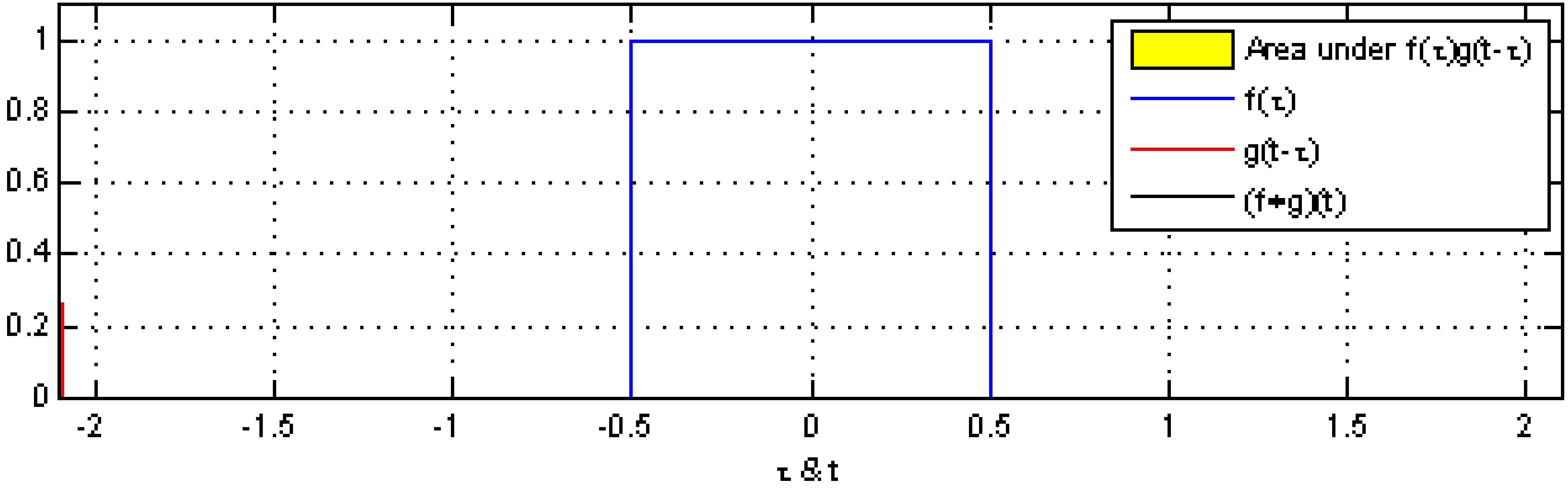
$$(f * g)[t] \coloneqq \sum_{\tau} f[t-\tau]g[\tau]$$

- Flip the filter and sliding



1D Convolution

- Example

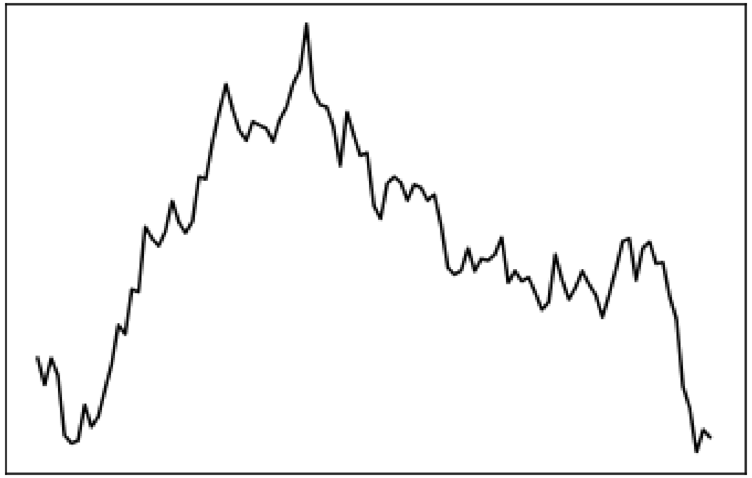


[Convolution - Wikipedia](#)

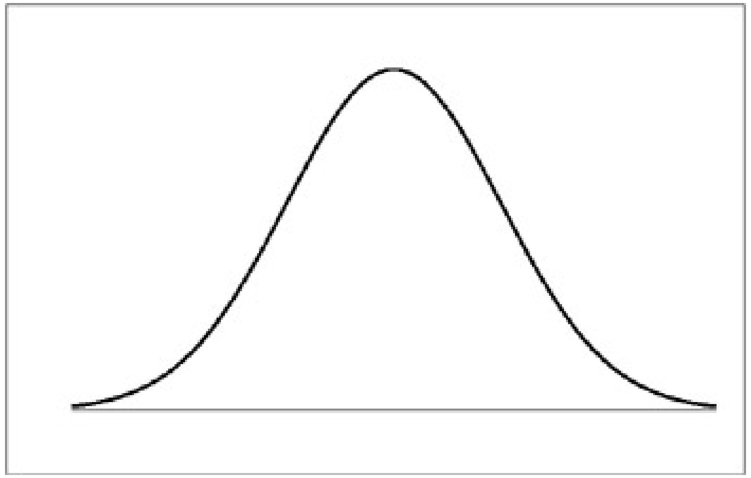
1D Convolution

- Gaussian filter

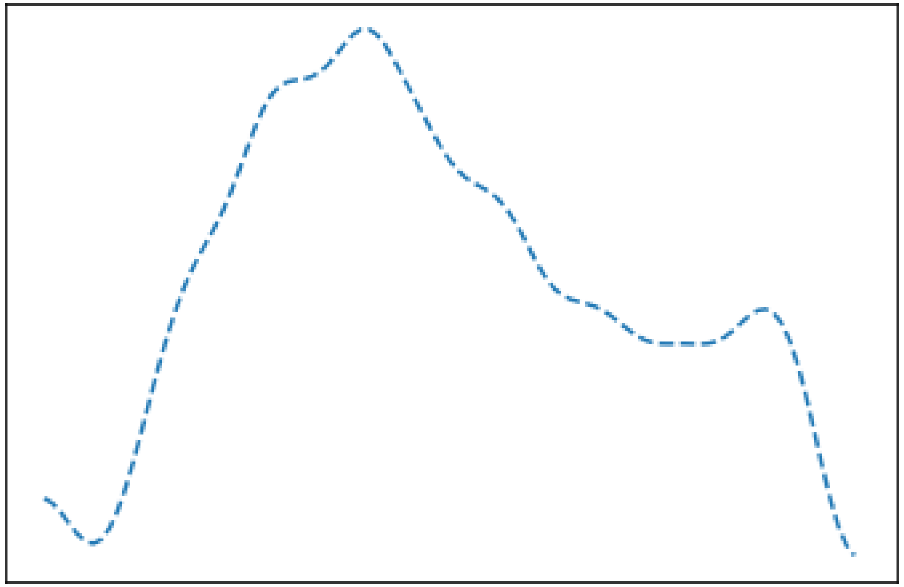
$f(t)$



$g(t)$



$f(t) * g(t)$



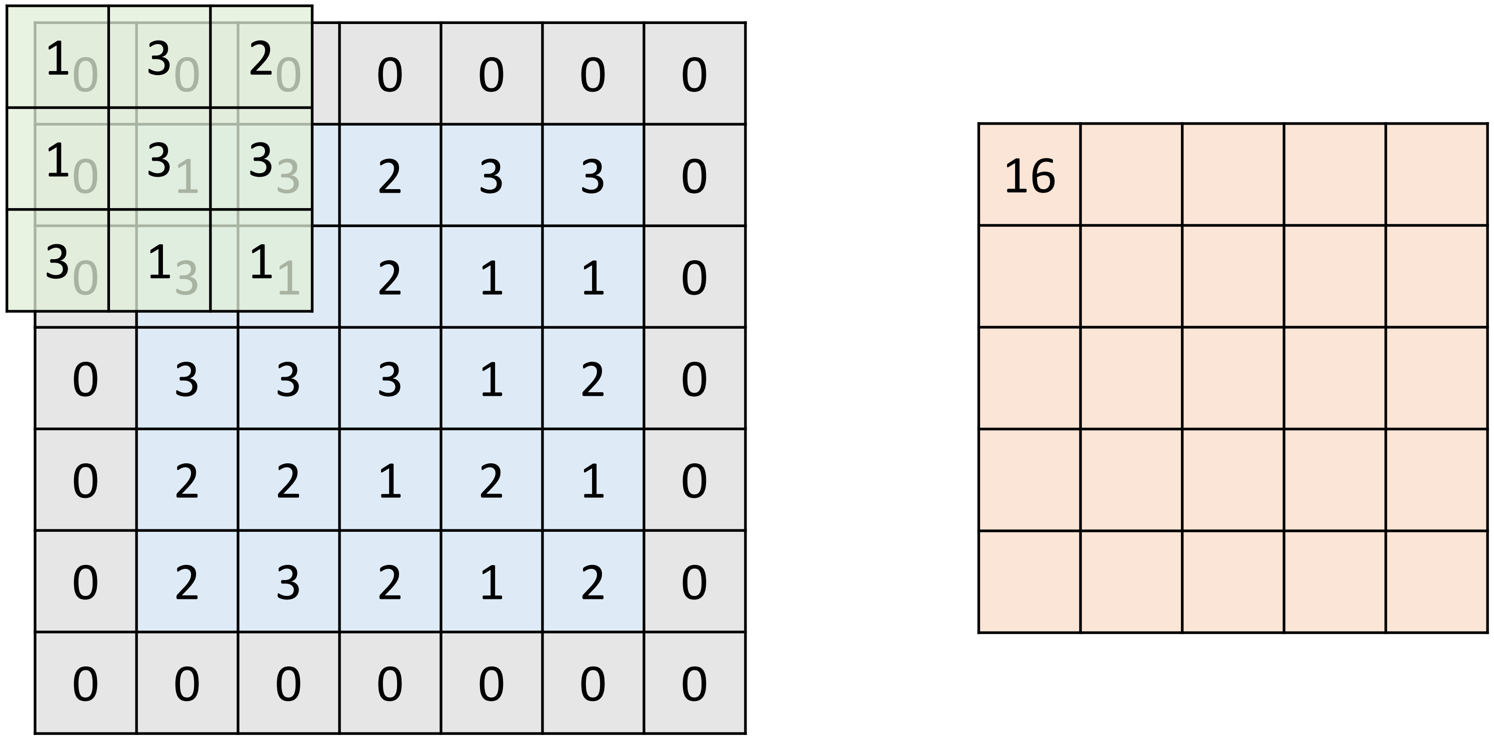
2D Convolution

$$(f * g)(s,t) \coloneqq \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(s-\tau_1, t-\tau_2)g(\tau_1,\tau_2)d\tau_1 d\tau_2$$

$$(f * g)[s,t] \coloneqq \sum_{\tau_1}\sum_{\tau_2} f[s-\tau_1, t-\tau_2]g[\tau_1,\tau_2]$$

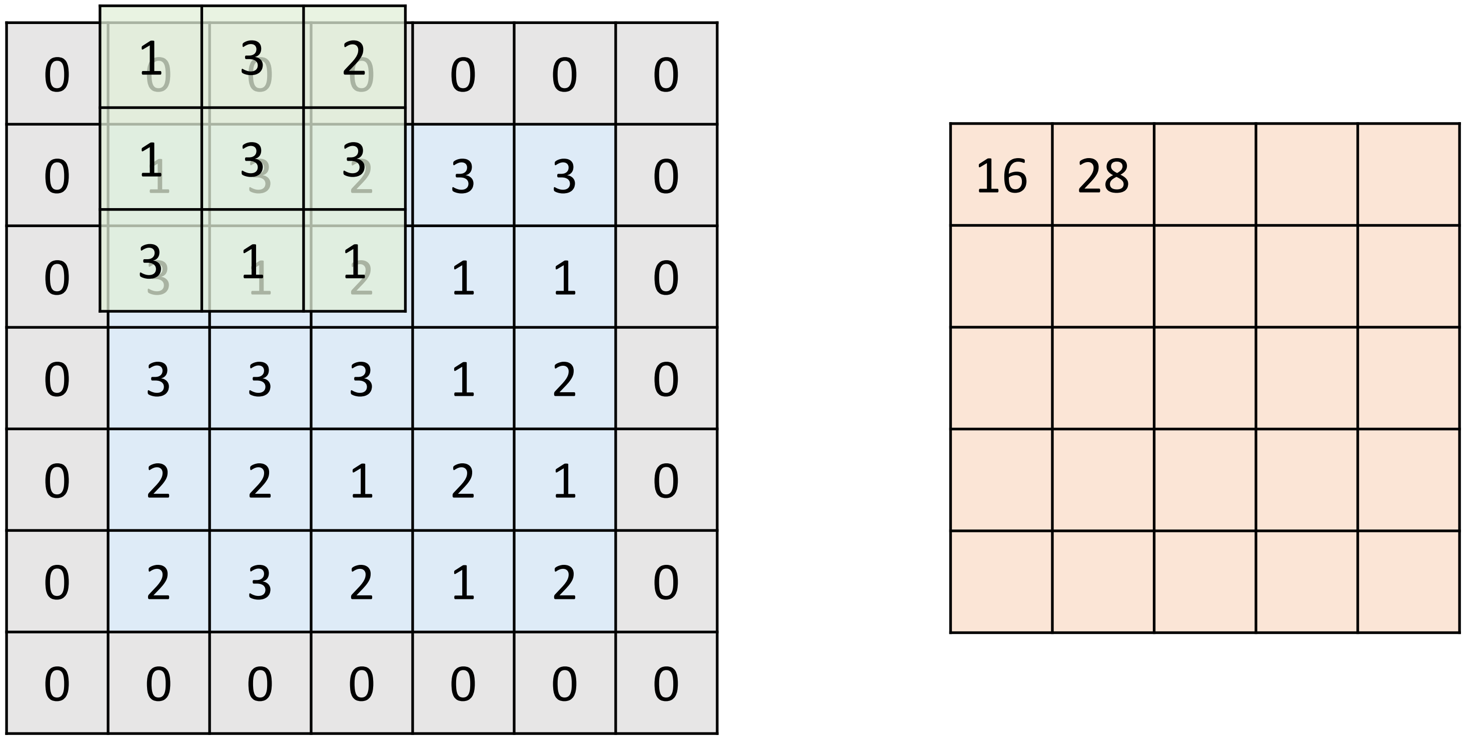
2D Convolution

- One input channel, e.g. gray color image
  - Padding=1, stride=1



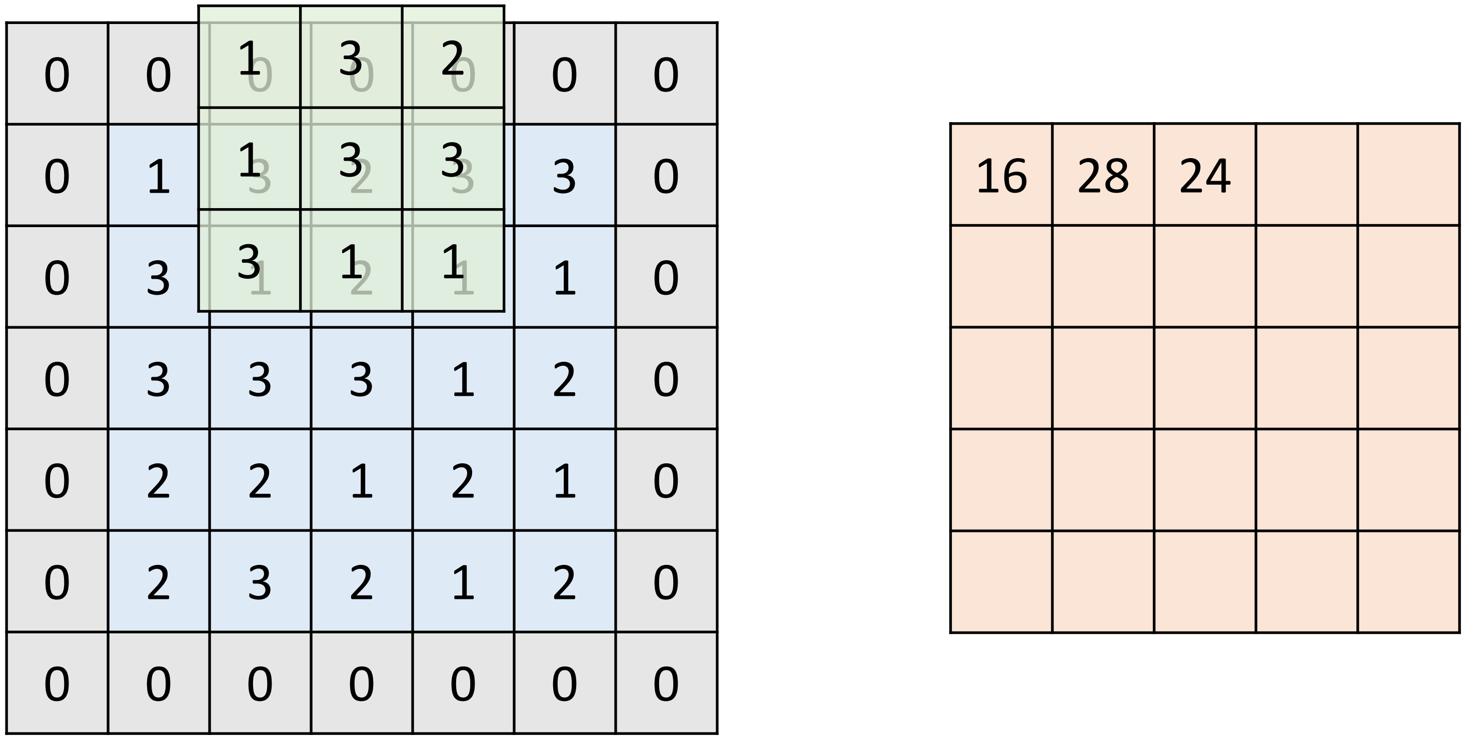
2D Convolution

- One input channel, e.g. gray color image
  - Padding=1, stride=1

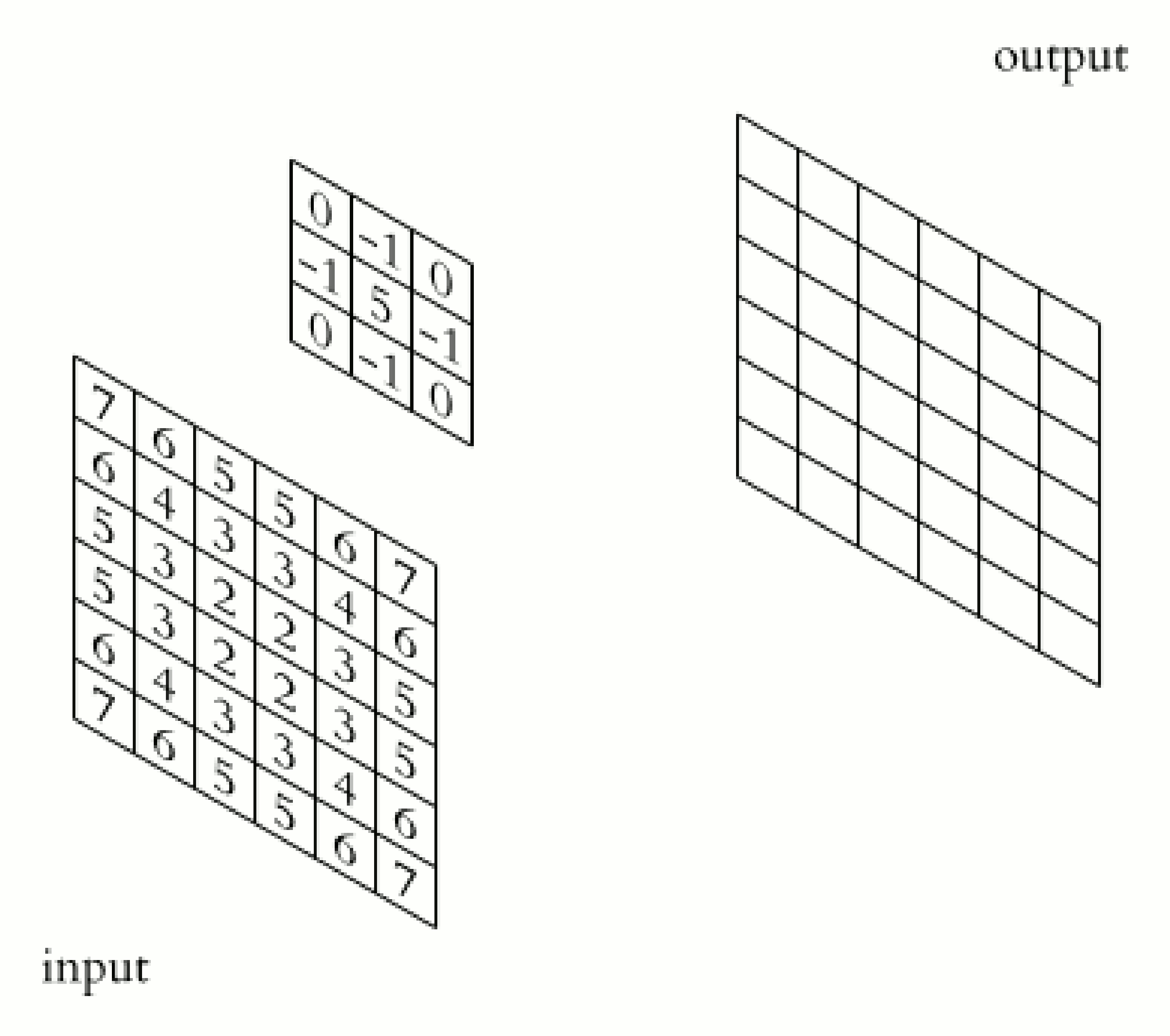


2D Convolution

- One input channel, e.g. gray color image
  - Padding=1, stride=1



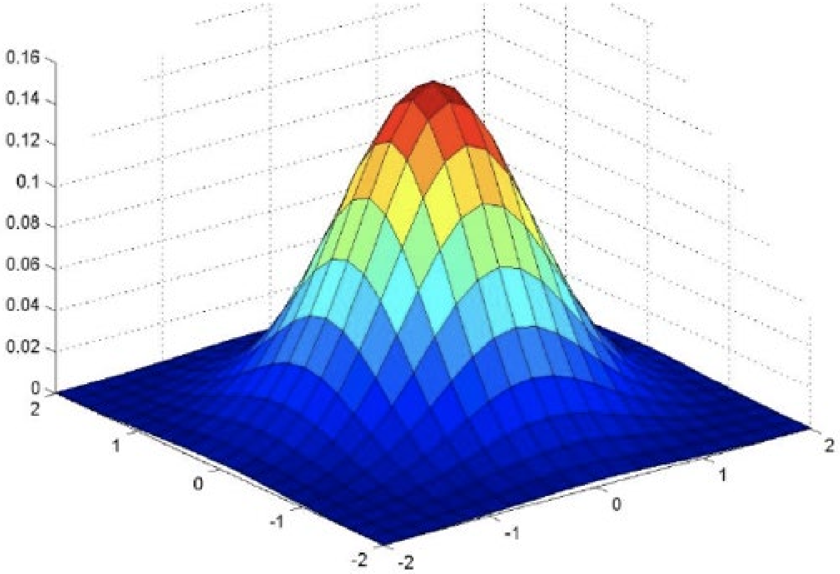
2D Convolution

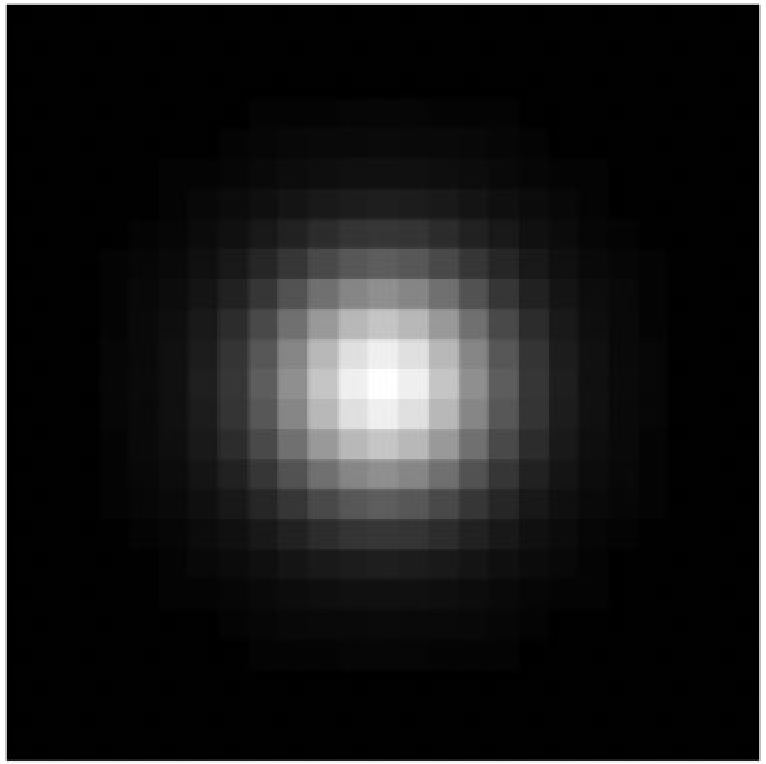


File:2D Convolution Animation.gif - Wikimedia Commons

2D Convolution

- 2D gaussian filter





[File:2D Convolution Animation.gif - Wikimedia Commons]

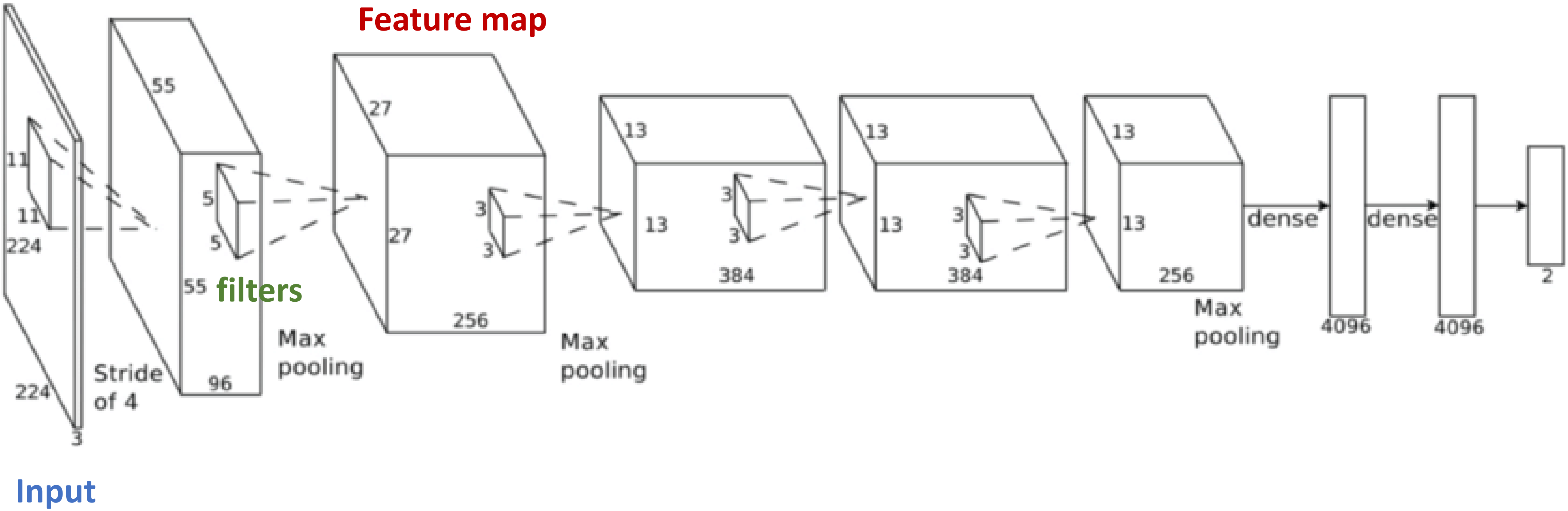
2D Convolution

- 2D gaussian filter

[2-D Gaussian filtering of images - MATLAB imgaussfilt (mathworks.com)]

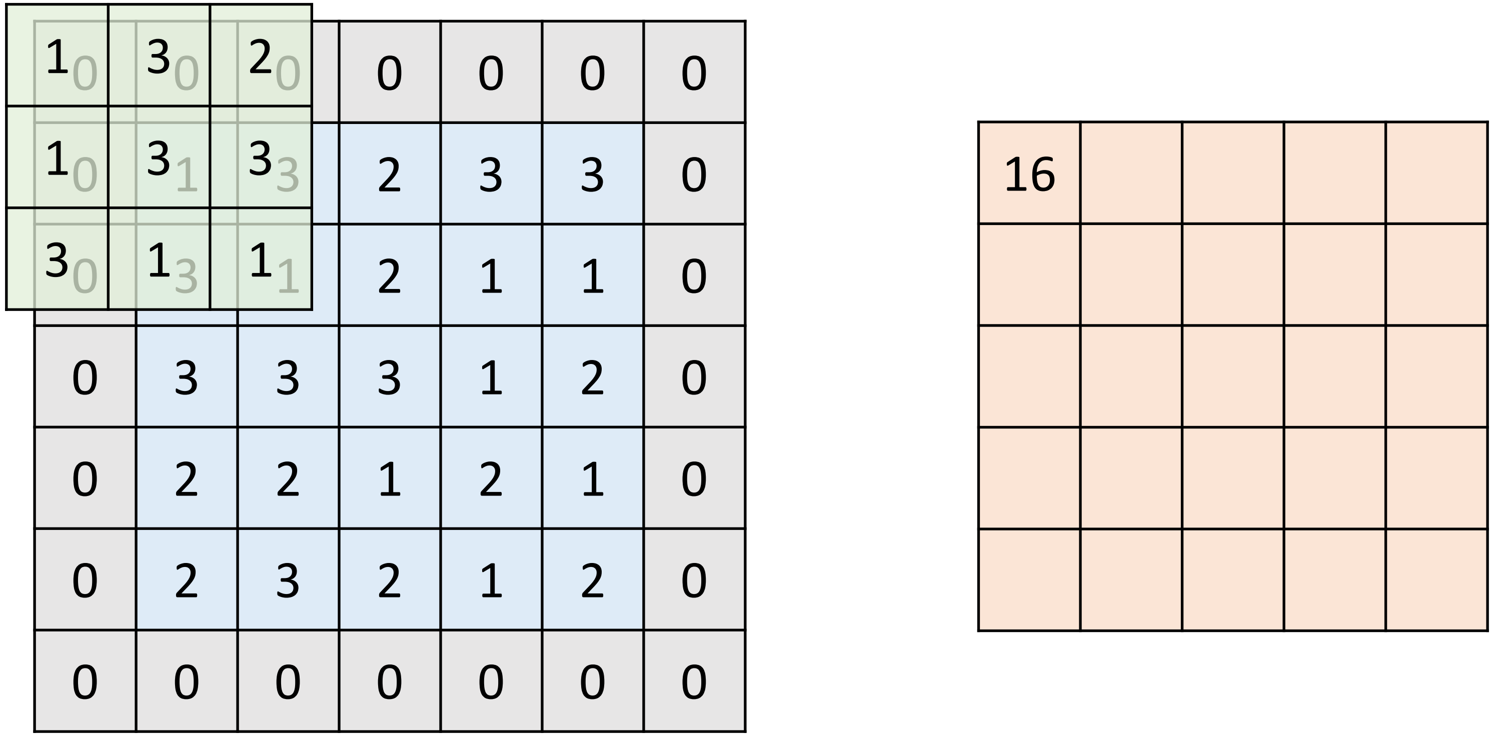
Convolutional Neural Networks

Convolutional Neural Network



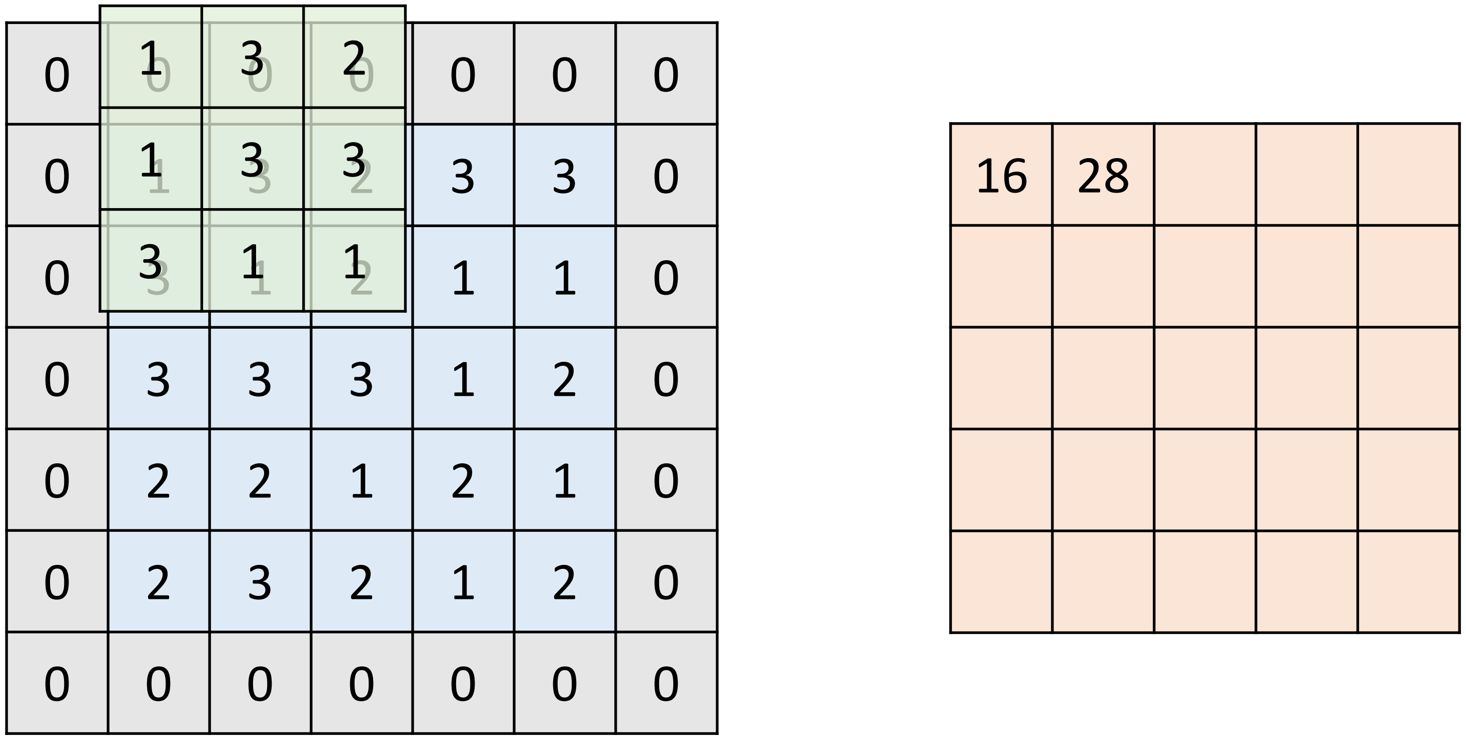
2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=1



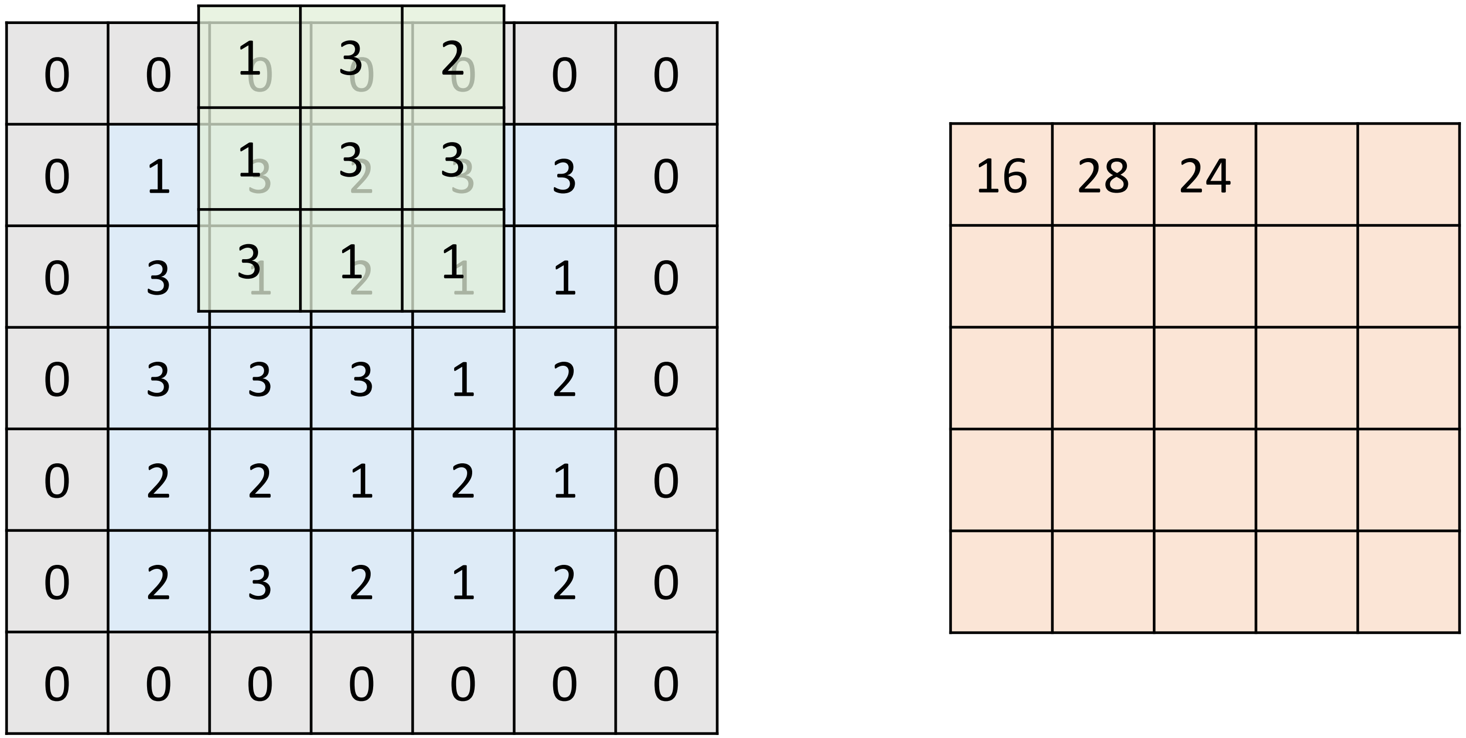
2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=1



2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=1



2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=1

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 2 & 3 & 3 & 0 \\ 0 & 3 & 1 & 2 & 1 & 1 & 0 \\ 0 & 3 & 3 & 3 & 1 & 2 & 0 \\ 0 & 2 & 2 & 1 & 2 & 1 & 0 \\ 0 & 2 & 3 & 2 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 2 \\ 1 & 3 & 3 \\ 3 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 16 & 28 & 24 & 28 & 16 \\ 27 & 41 & 38 & 37 & 21 \\ 33 & 40 & 33 & 25 & 18 \\ 32 & 40 & 37 & 29 & 17 \\ 25 & 27 & 21 & 20 & 12 \end{bmatrix}$$

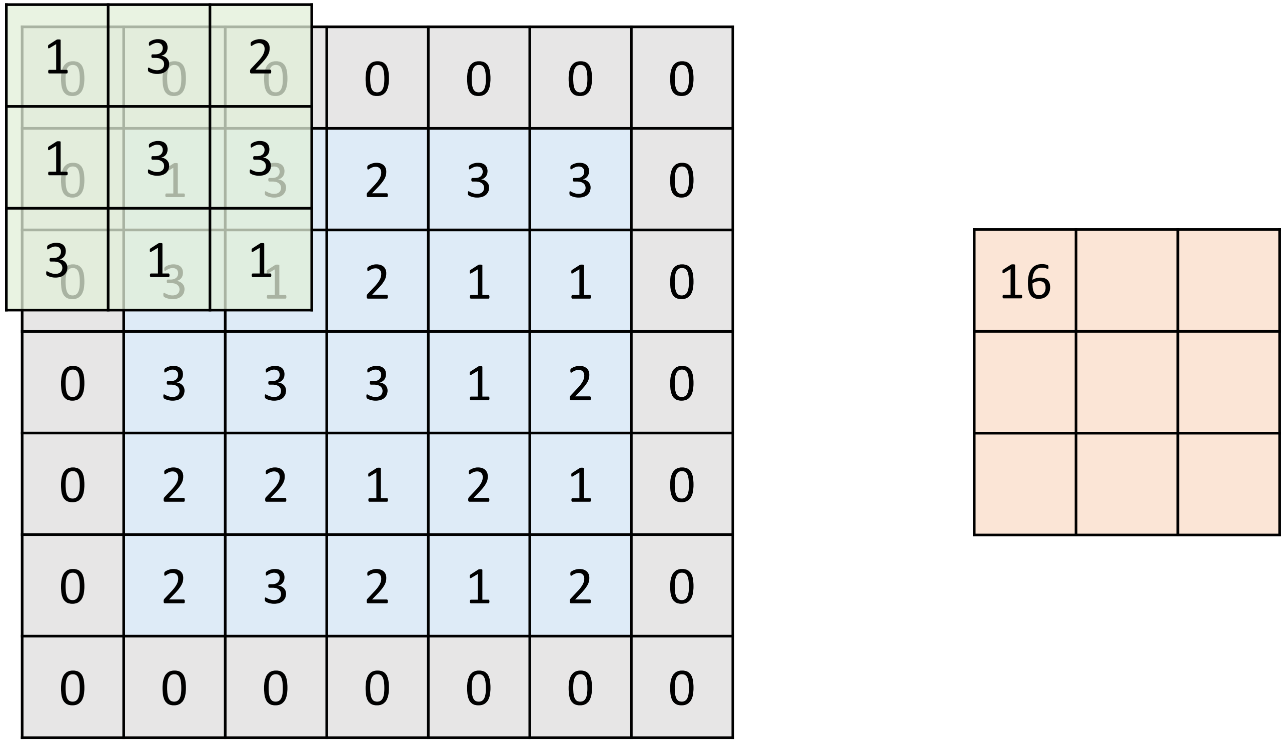
Input

filter

Output

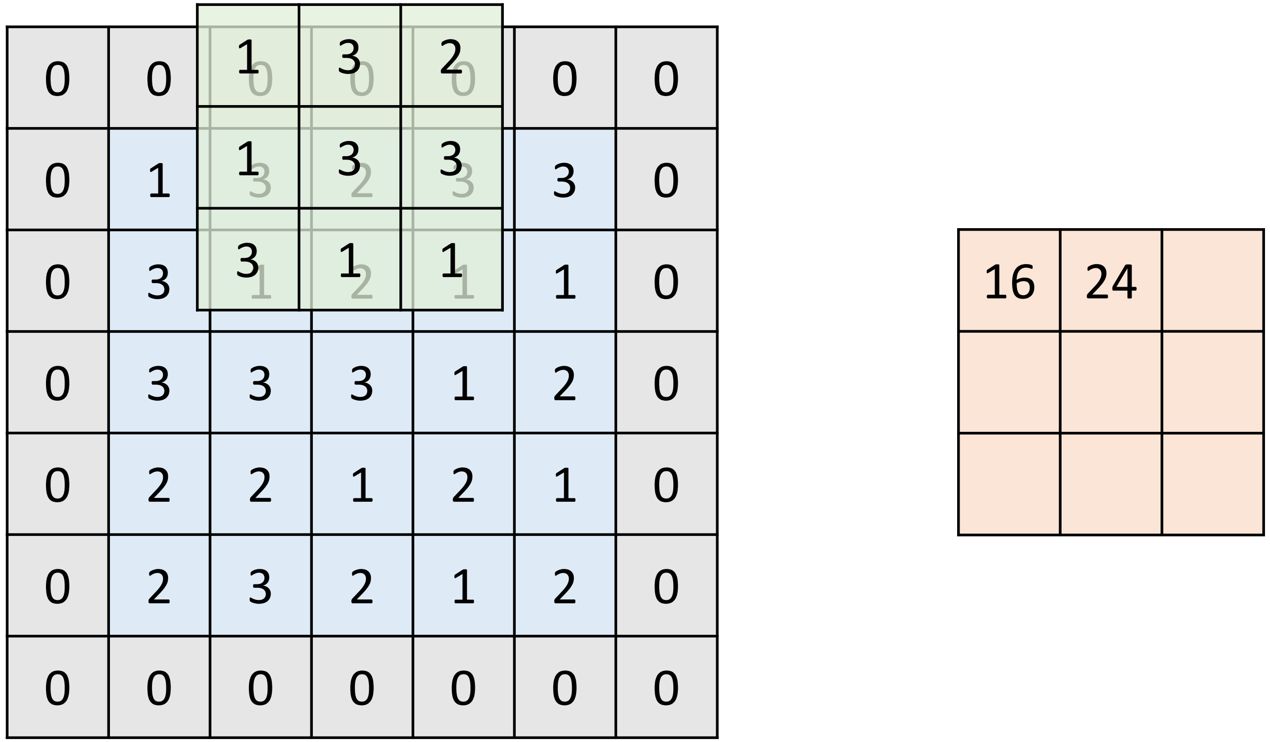
2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=2



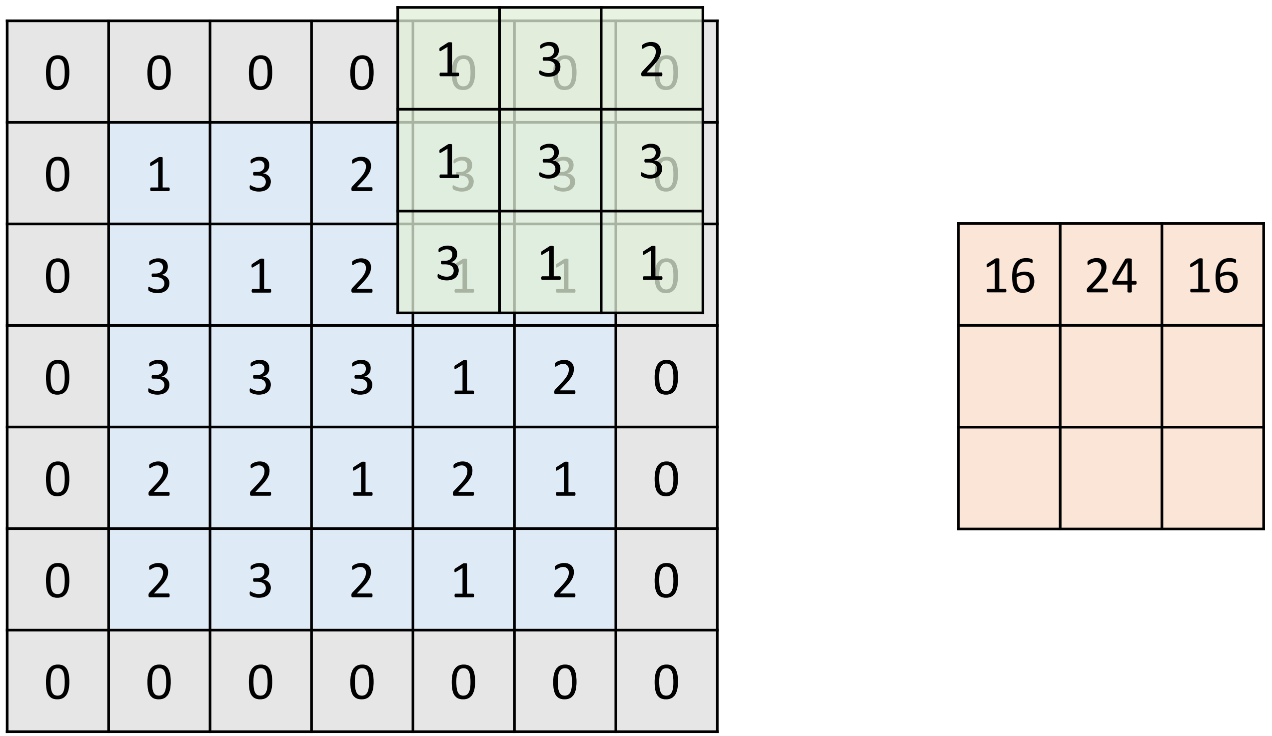
2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=2



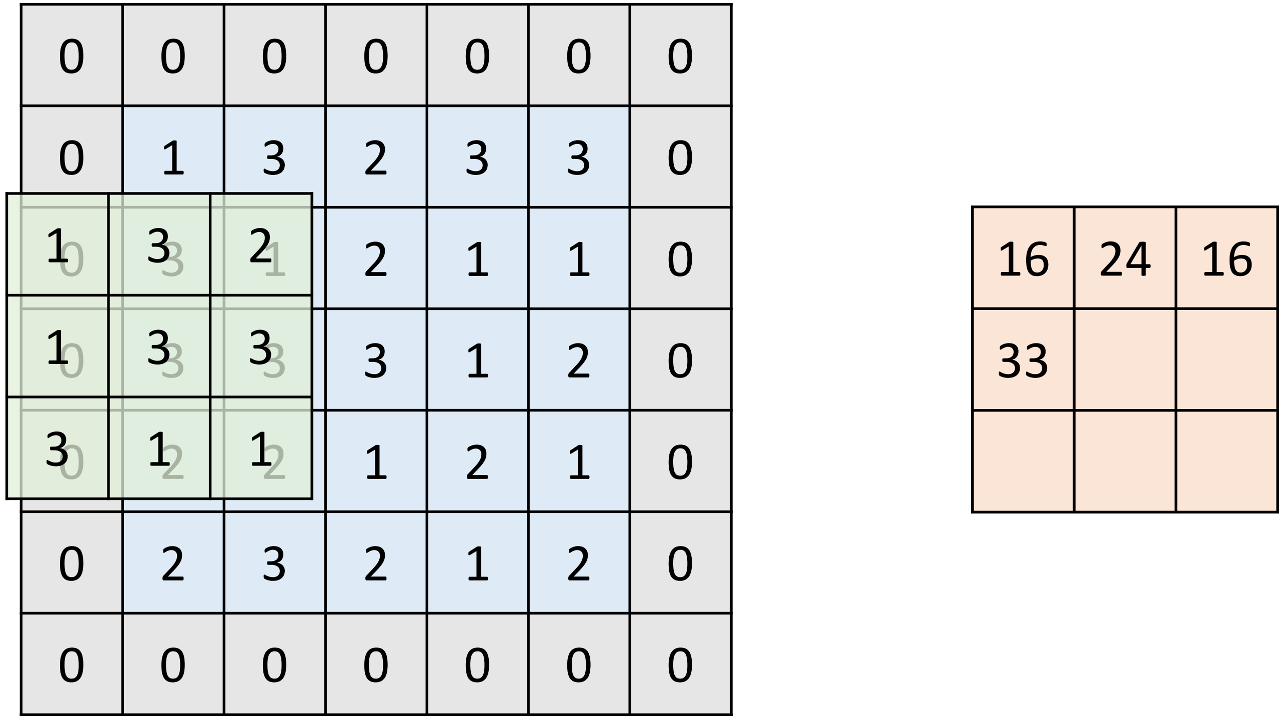
2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=2



2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=2



2D Convolution

- Input_channel=1, output_channel=1, padding=1, stride=2

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

Input

*

1	3	2
1	3	3
3	1	1

filter

=

16	24	16
33	33	18
25	21	12

Output

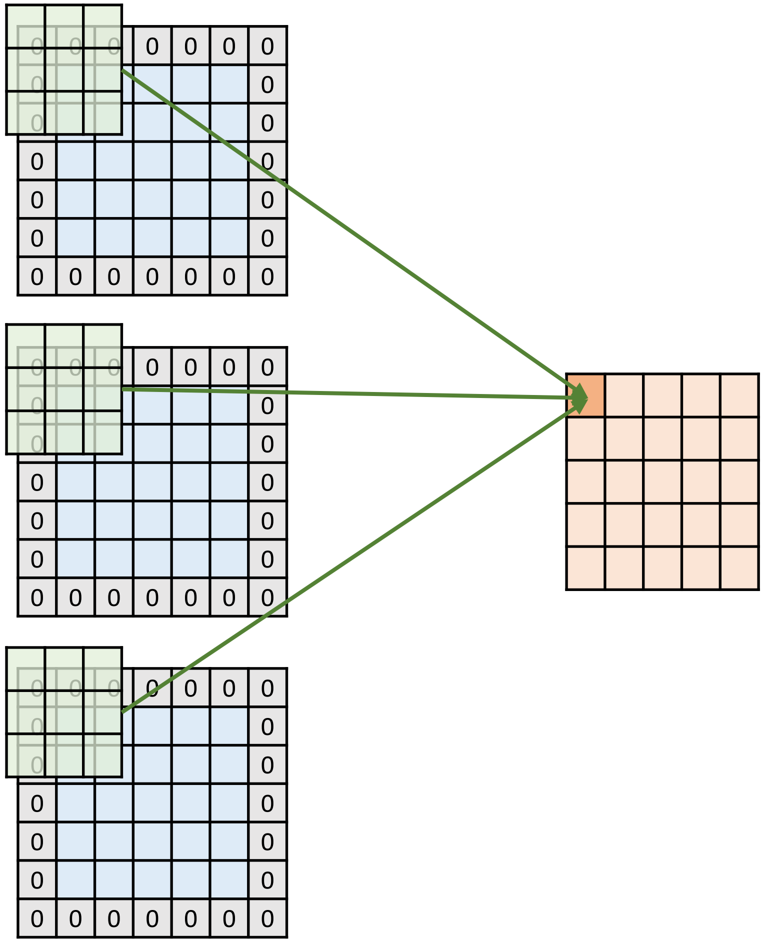
2D Convolution

- Input_channel=1, output_channel=1, padding=2, stride=1

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 3 & 2 & 3 & 3 & 0 & 0 \\
0 & 0 & 3 & 1 & 2 & 1 & 1 & 0 & 0 \\
0 & 0 & 3 & 3 & 3 & 1 & 2 & 0 & 0 \\
0 & 0 & 2 & 2 & 1 & 2 & 1 & 0 & 0 \\
0 & 0 & 2 & 3 & 2 & 1 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 3 & 2 \\
1 & 3 & 3 \\
3 & 1 & 1
\end{bmatrix}
=
\begin{bmatrix}
1 & 4 & 8 & 14 & 12 & 12 & 9 \\
6 & 16 & 28 & 24 & 28 & 16 & 6 \\
14 & 27 & 41 & 38 & 37 & 21 & 10 \\
17 & 33 & 40 & 33 & 25 & 18 & 6 \\
14 & 32 & 40 & 37 & 29 & 17 & 9 \\
10 & 25 & 27 & 21 & 20 & 12 & 3 \\
4 & 12 & 15 & 11 & 9 & 7 & 2
\end{bmatrix}
$$

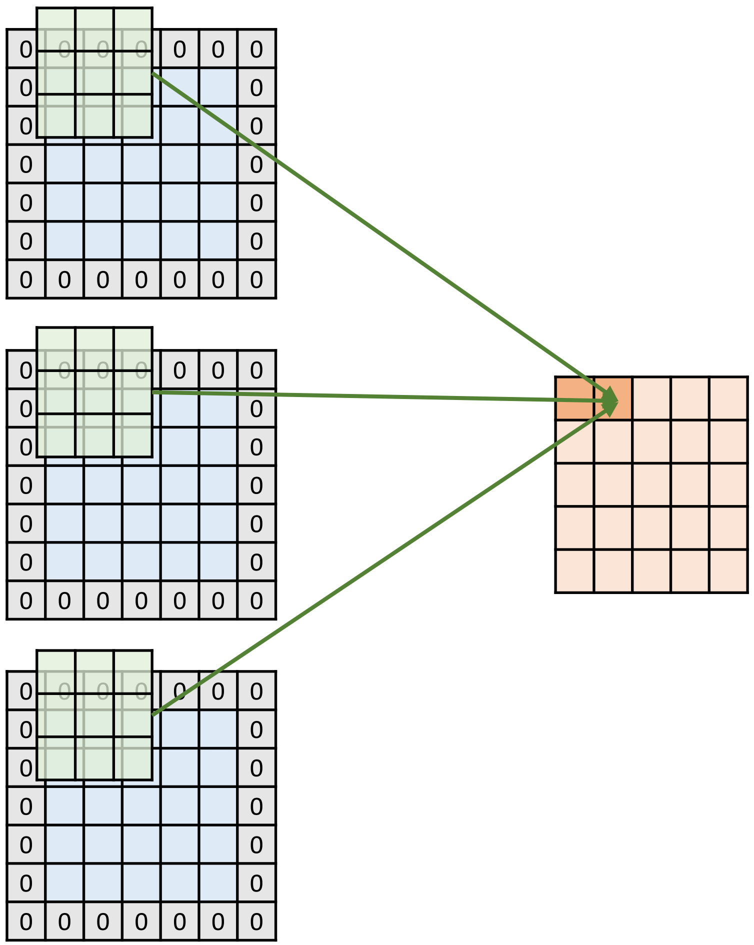
2D Convolution

- Input_channel=3, output_channel=1, padding=1, stride=1



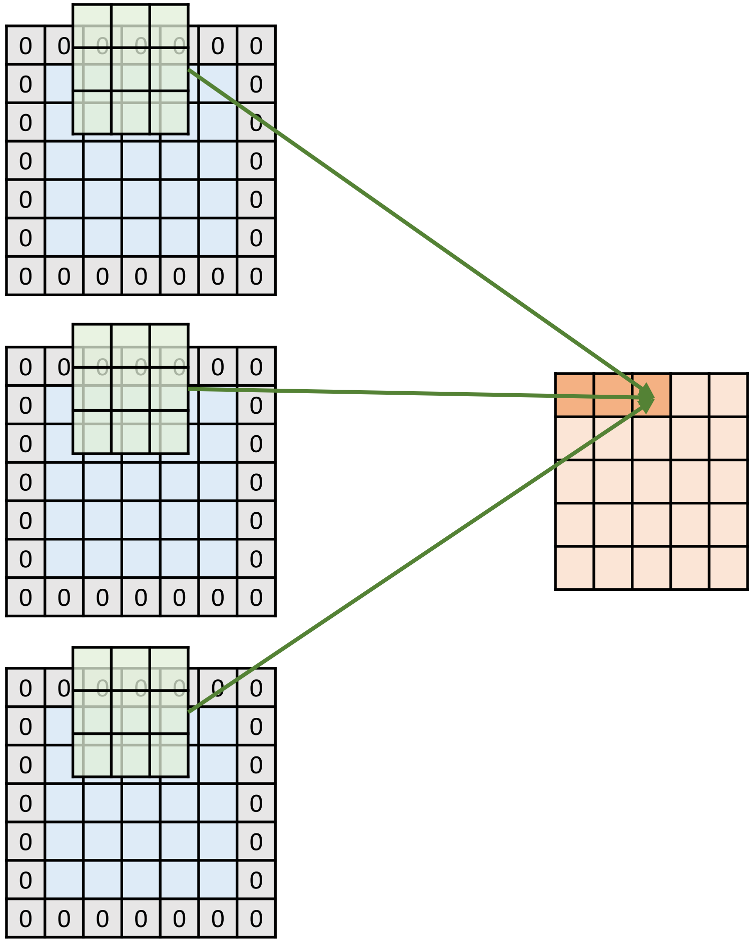
2D Convolution

- Input_channel=3, output_channel=1, padding=1, stride=1



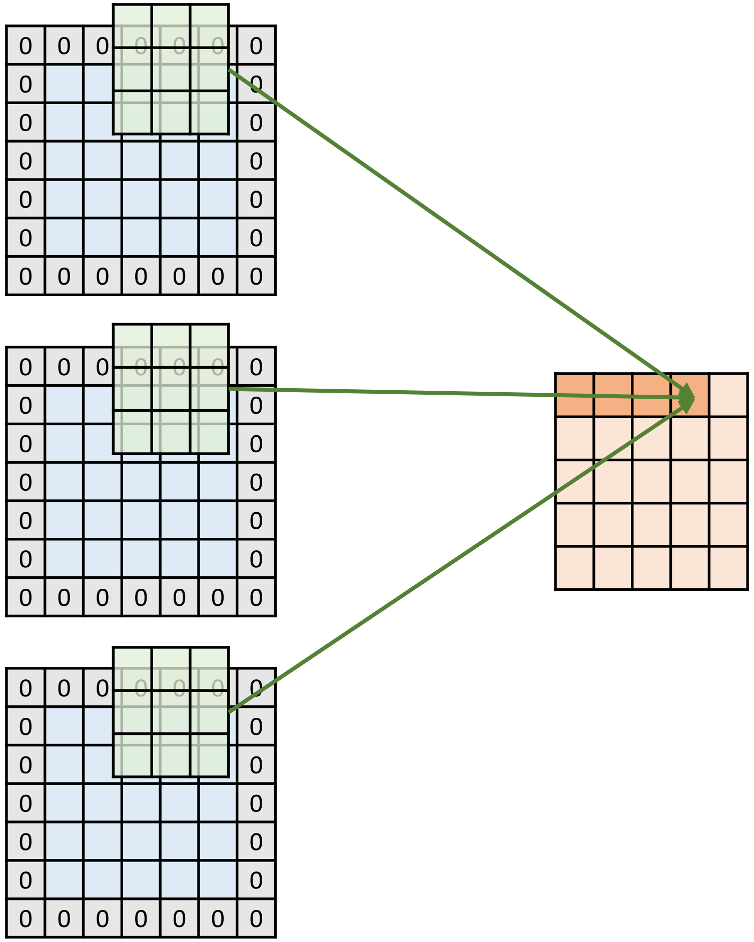
2D Convolution

- Input_channel=3, output_channel=1, padding=1, stride=1



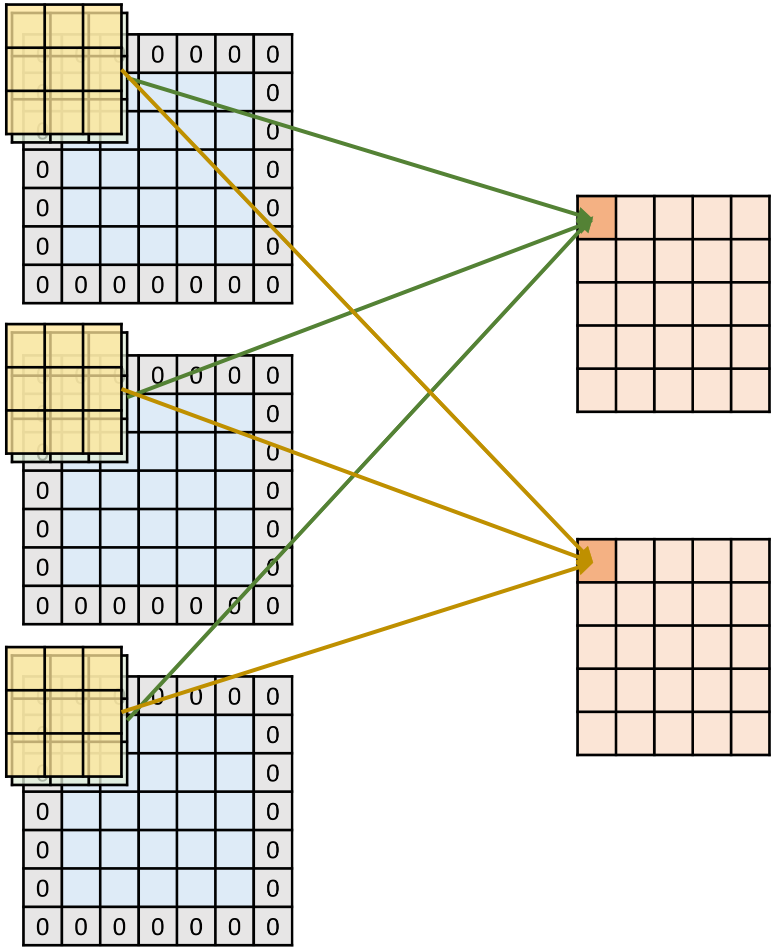
2D Convolution

- Input_channel=3, output_channel=1, padding=1, stride=1



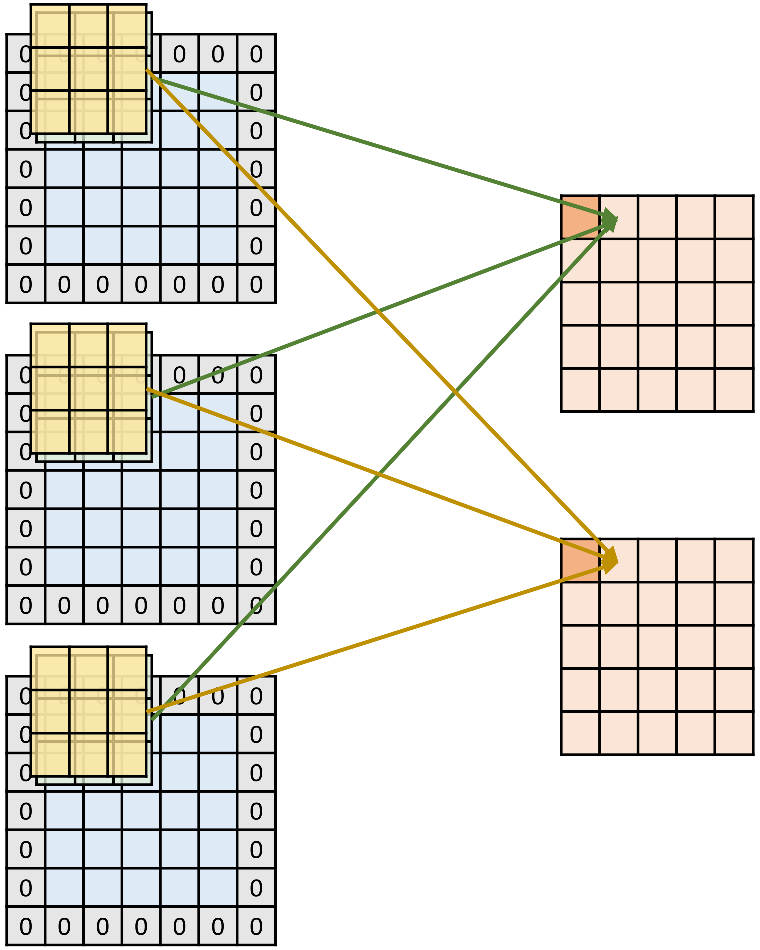
2D Convolution

- Input_channel=3, output_channel=2, padding=1, stride=1



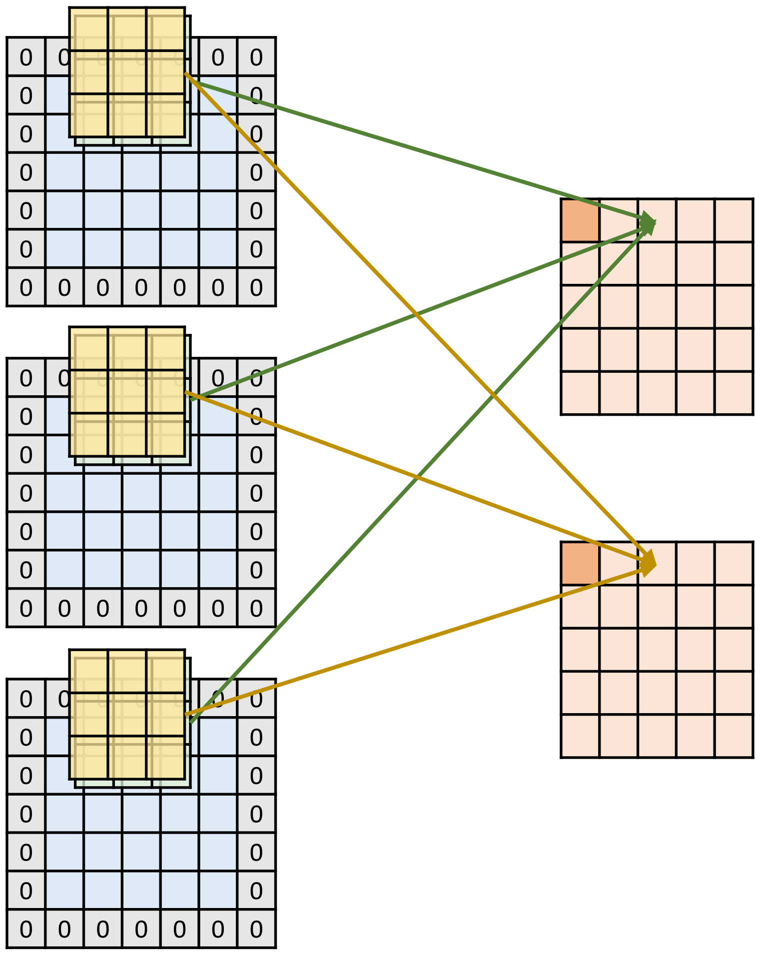
2D Convolution

- Input_channel=3, output_channel=2, padding=1, stride=1



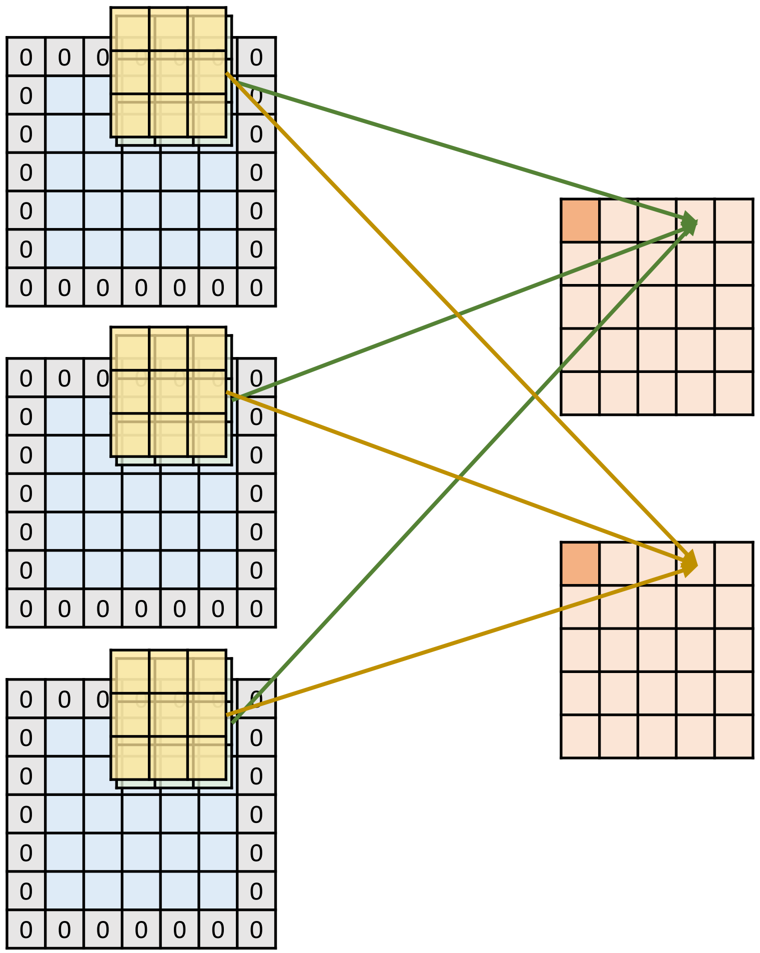
2D Convolution

- Input_channel=3, output_channel=2, padding=1, stride=1



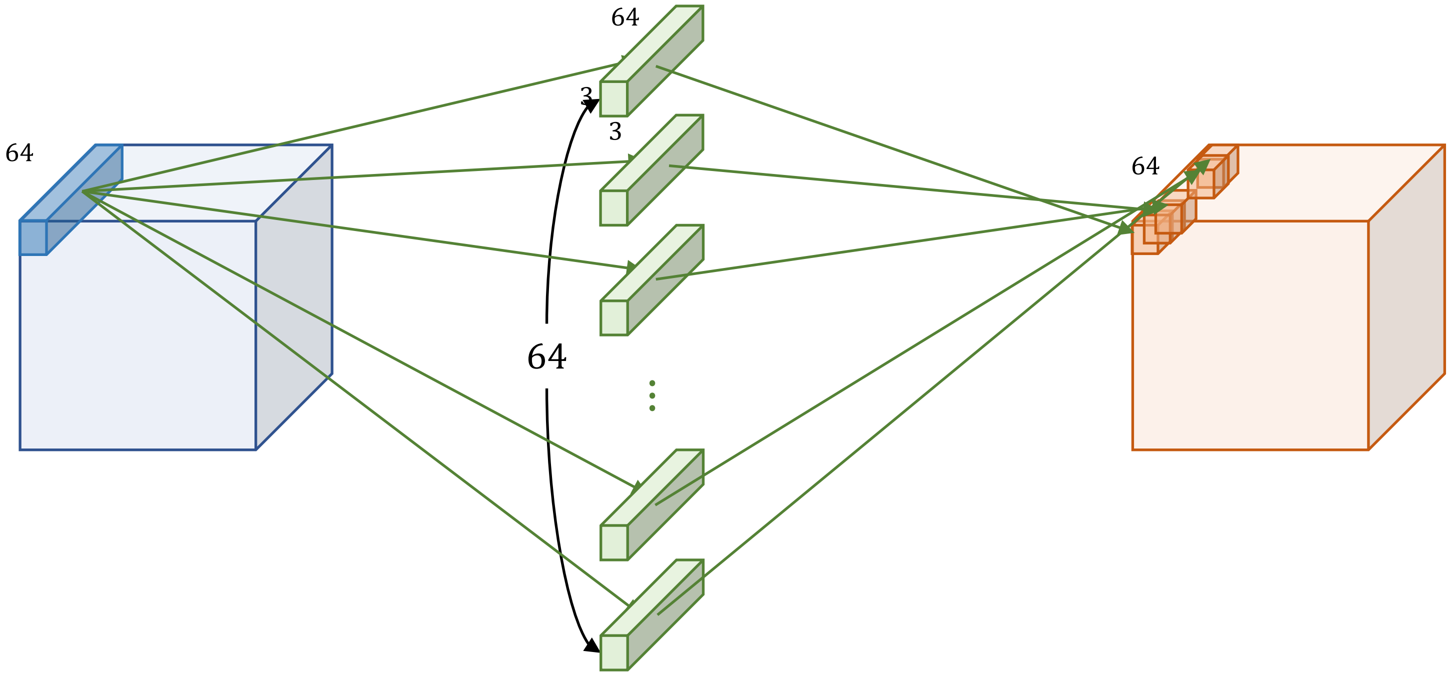
2D Convolution

- Input_channel=3, output_channel=2, padding=1, stride=1



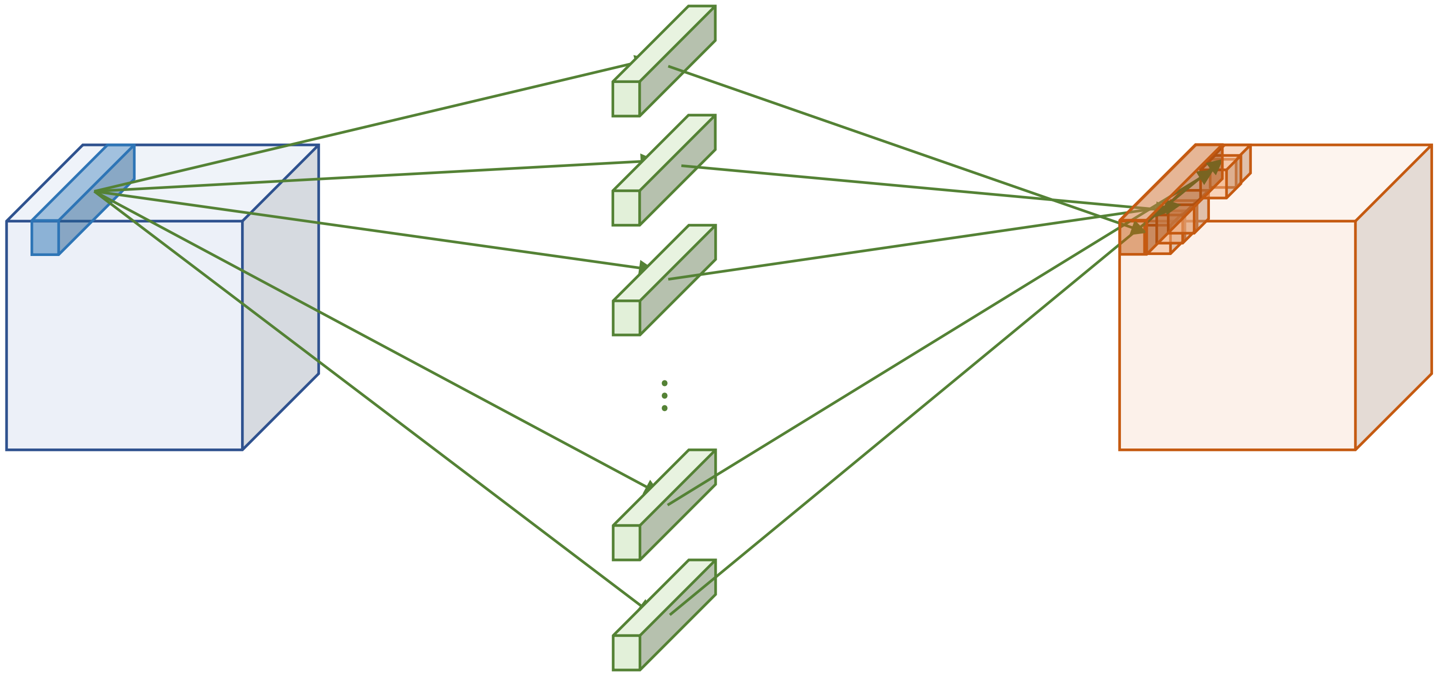
2D Convolution

- Input_channel=64, output_channel=64, kernel_size=3, padding=1, stride=1



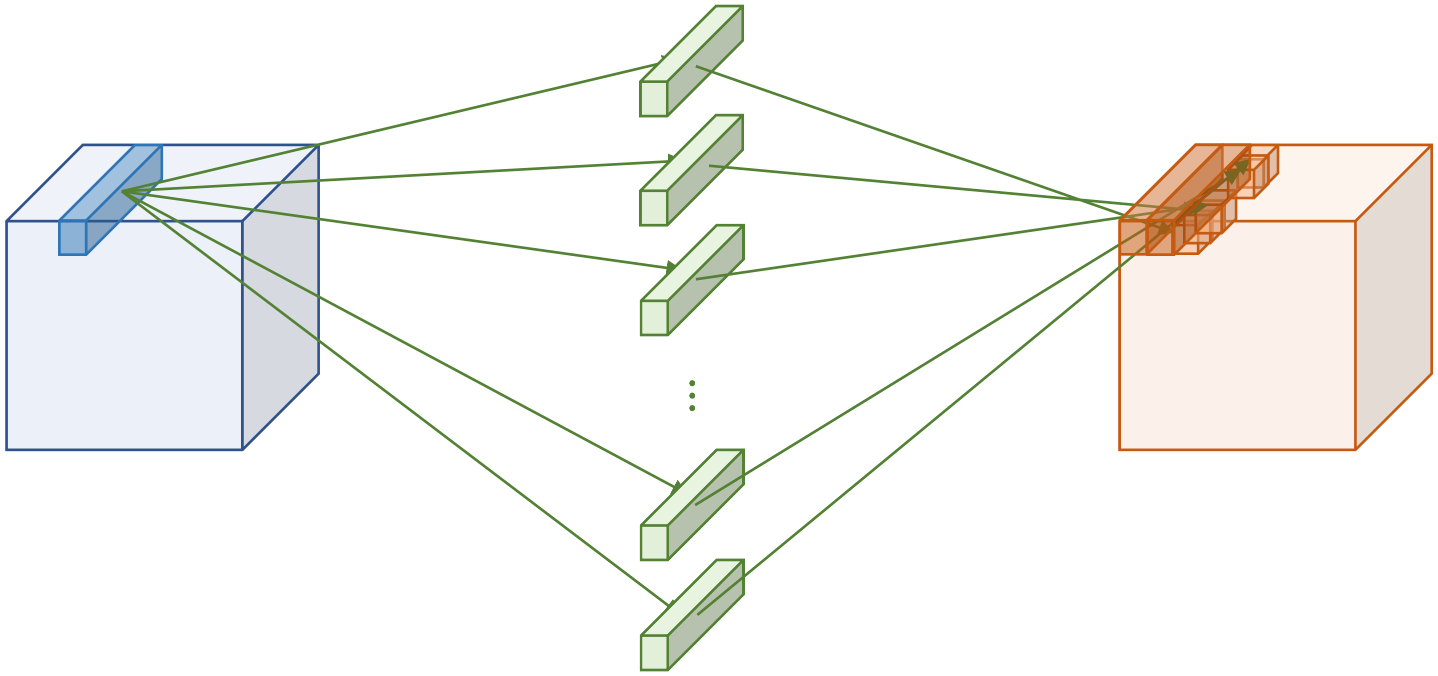
2D Convolution

- Input_channel=64, output_channel=64, kernel_size=3, padding=1, stride=1



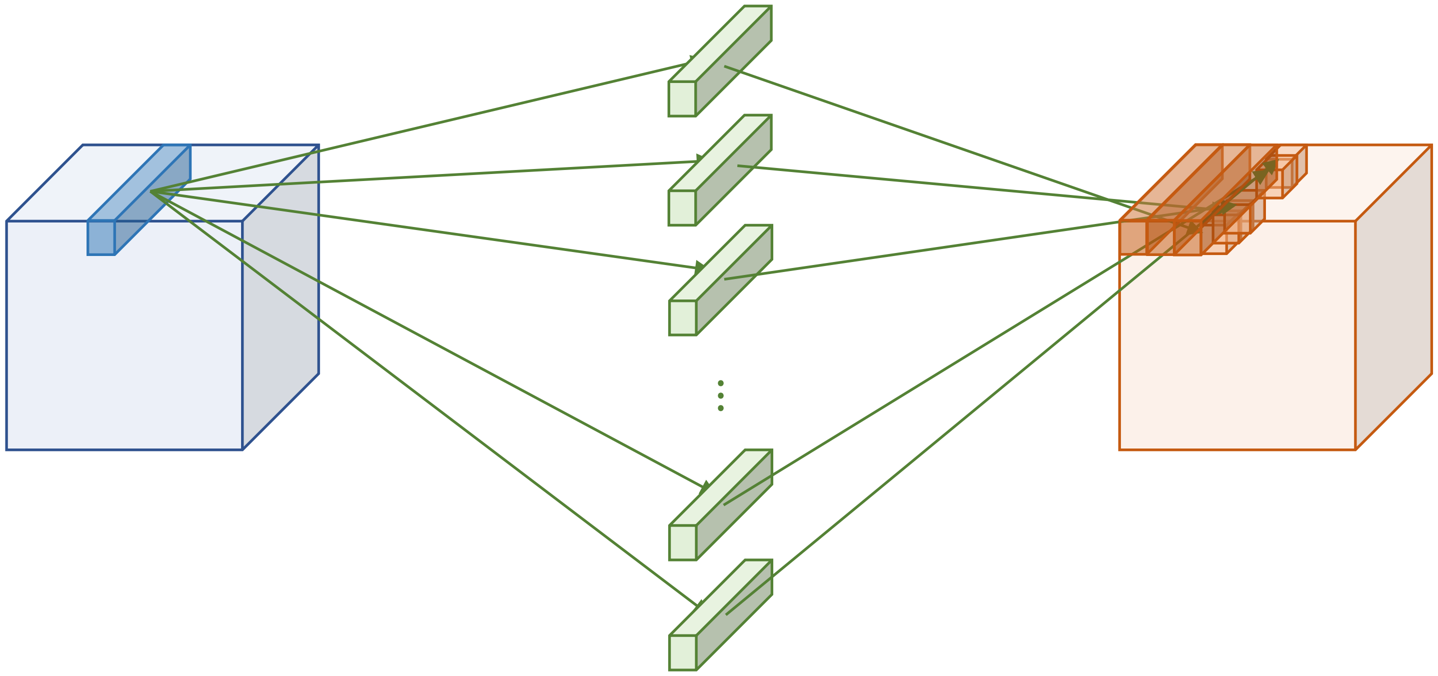
2D Convolution

- Input_channel=64, output_channel=64, kernel_size=3, padding=1, stride=1



2D Convolution

- Input_channel=64, output_channel=64, kernel_size=3, padding=1, stride=1



Convolutions in PyTorch

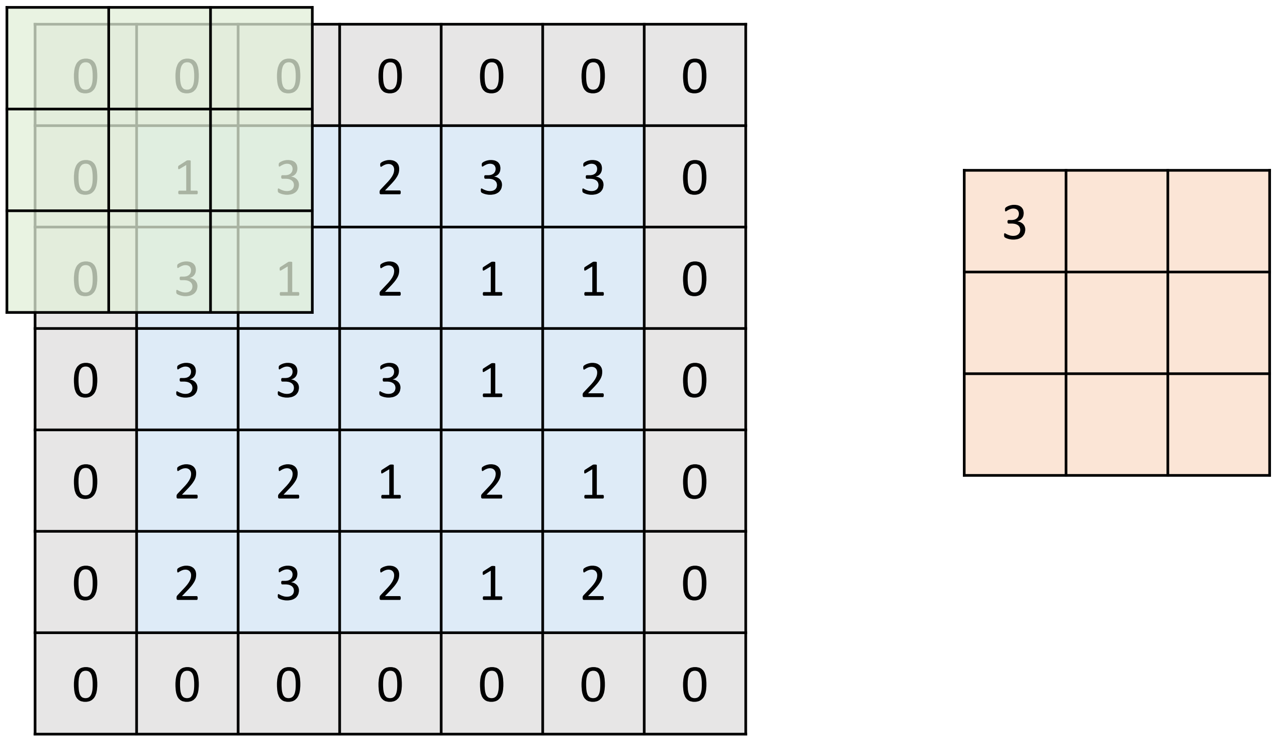
CLASS `torch.nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

CLASS `torch.nn.Conv3d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

Max Pooling

- Pooling a maximum value given the window
- Used to reduce the size of feature maps
- Example) stride=2, padding=1



Max Pooling

- Pooling a maximum value given the window
- Used to reduce the size of feature maps
- Example) stride=2, padding=1

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

3	3	

Max Pooling

- Pooling a maximum value given the window
- Used to reduce the size of feature maps
- Example) stride=2, padding=1

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

3	3	3

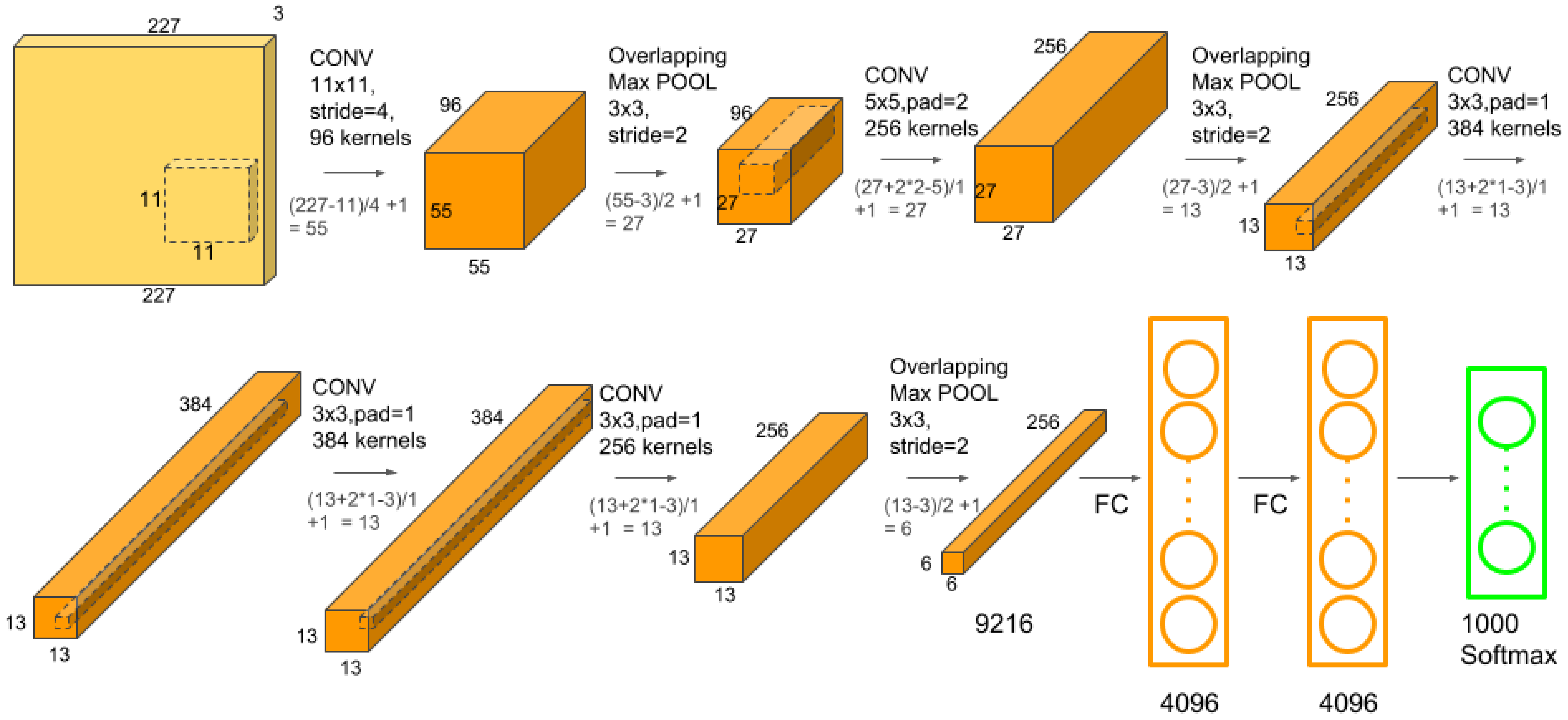
Max Pooling in PyTorch

CLASS `torch.nn.MaxPool1d(`*`kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False`*`)` [SOURCE]

CLASS `torch.nn.MaxPool2d(`*`kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False`*`)` [SOURCE]

CLASS `torch.nn.MaxPool3d(`*`kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False`*`)` [SOURCE]

AlexNet

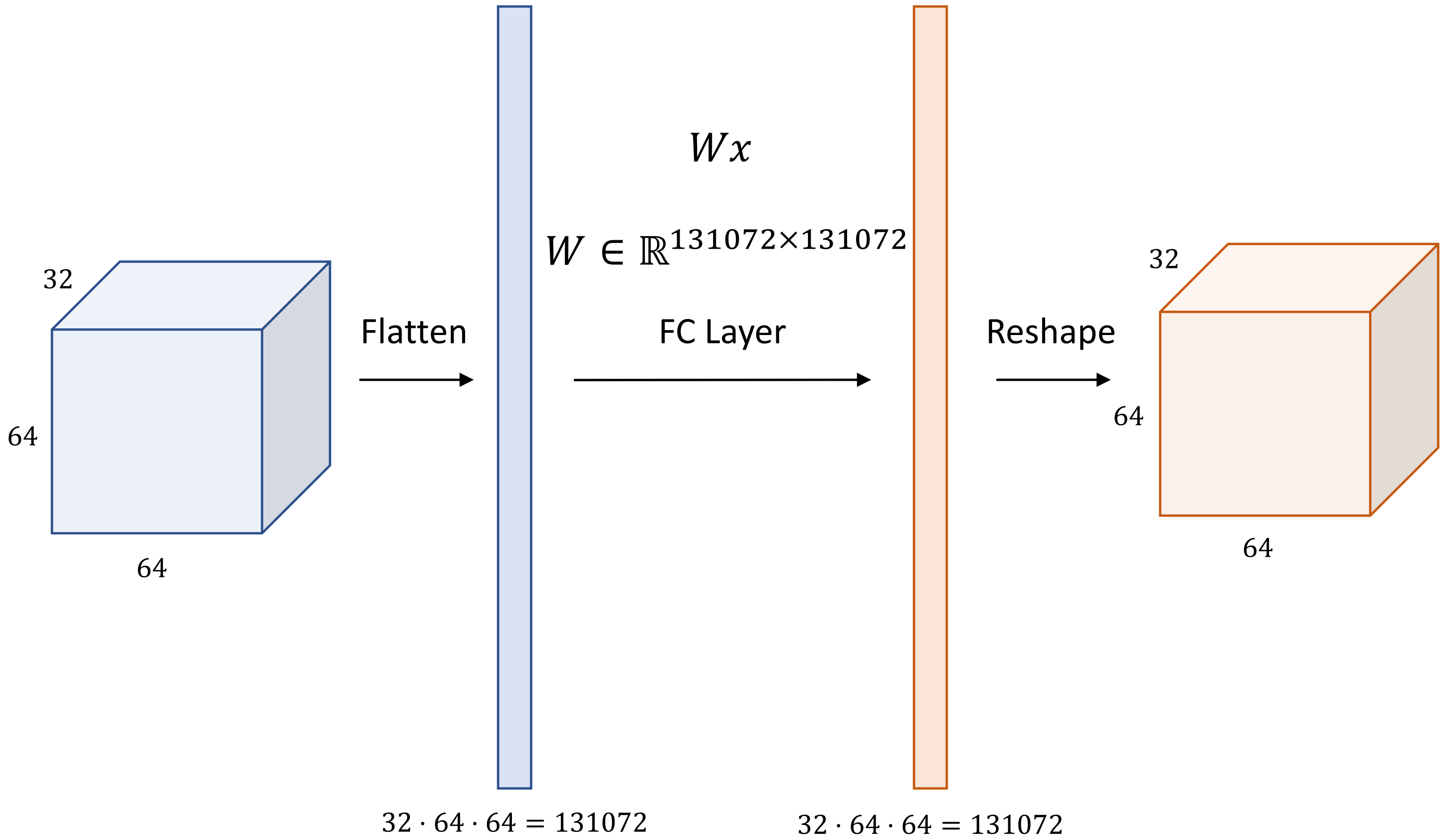


Dropout Dropout

Understanding AlexNet | LearnOpenCV #

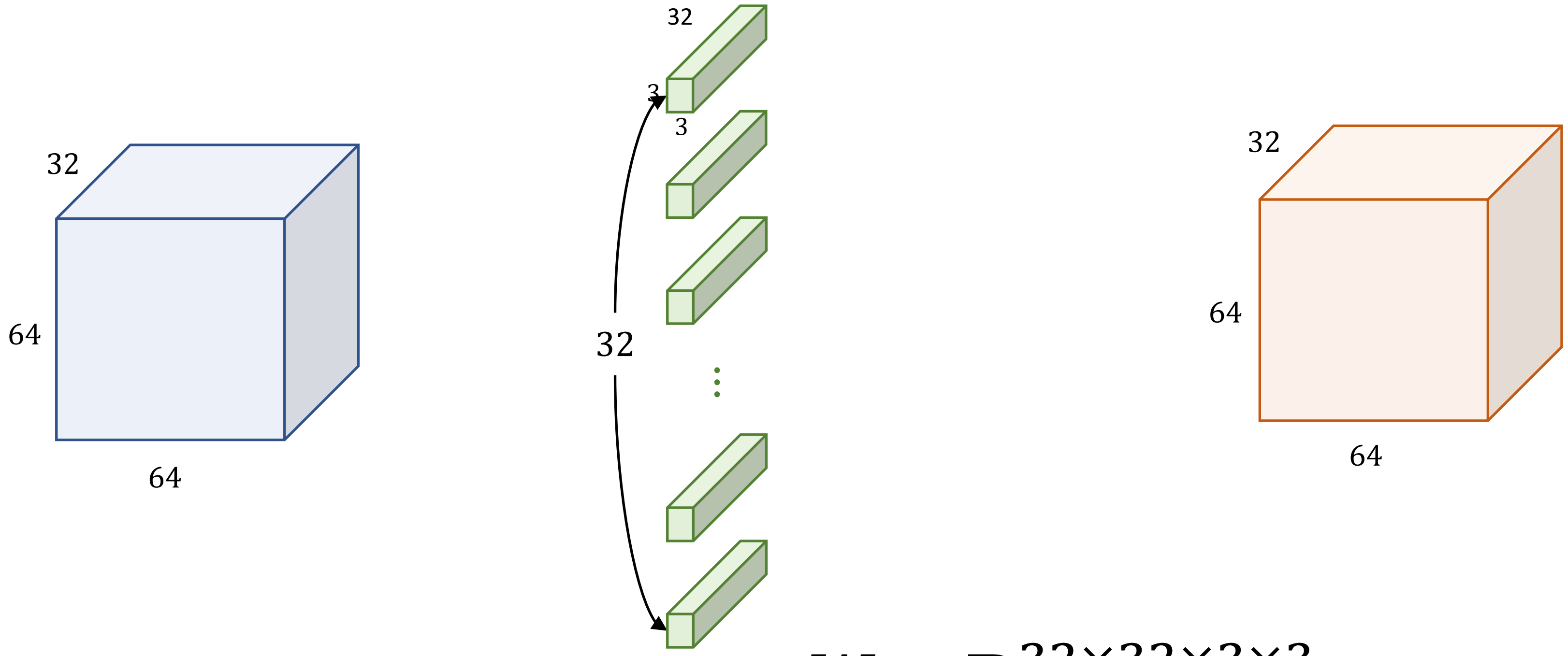
Fully Connected Layer vs Convolutional Layer

- Translation equivariance and parameter sharing



Fully Connected Layer vs Convolutional Layer

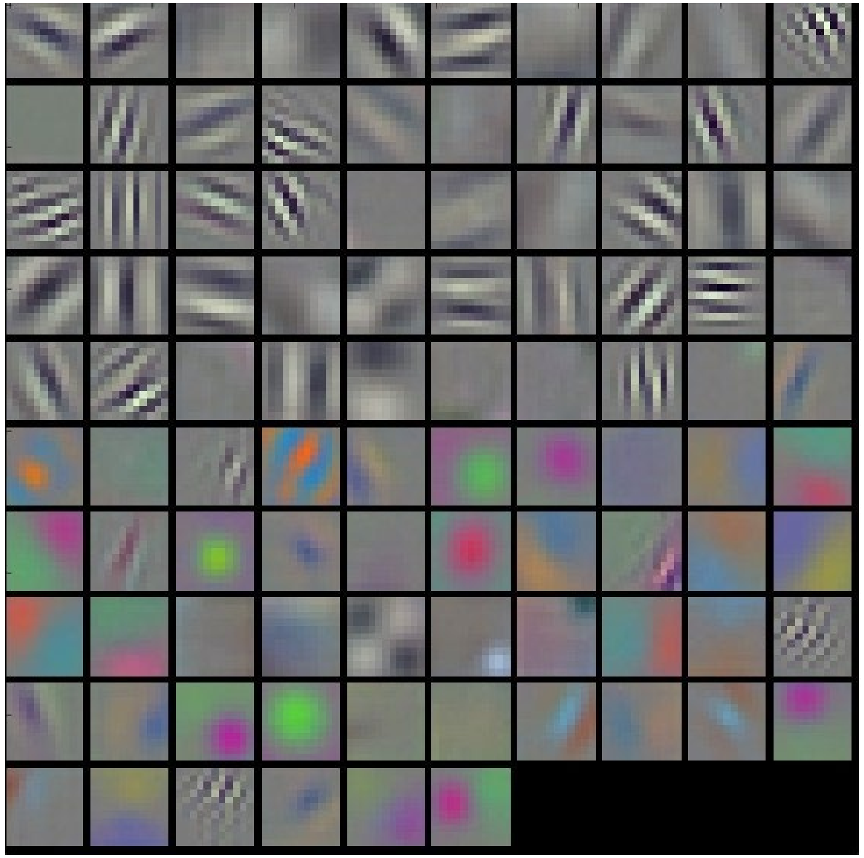
- Translation equivariance and parameter sharing



$$W \in \mathbb{R}^{32\times32\times3\times3}$$

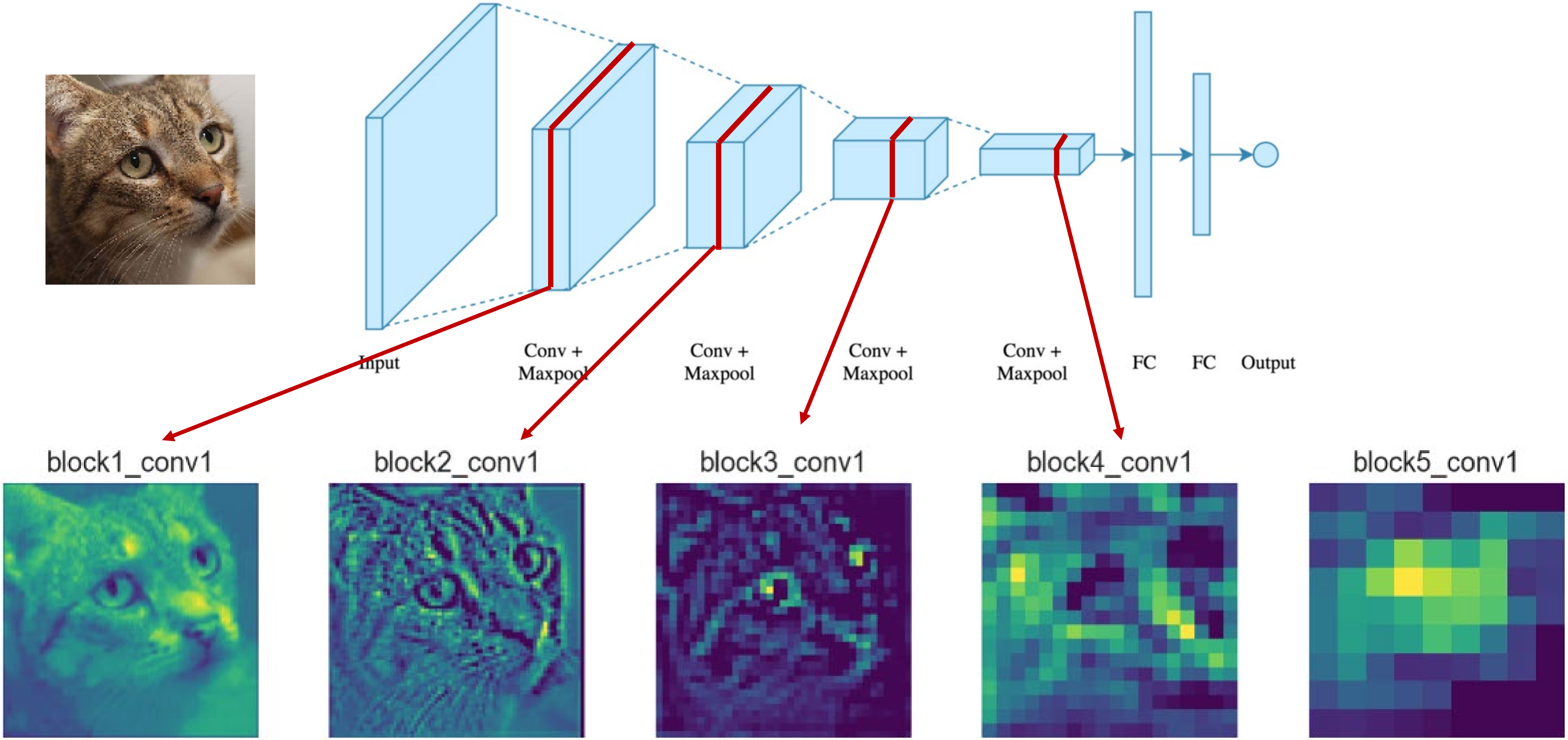
Visualization of Learned Filter

- First layer conv filters



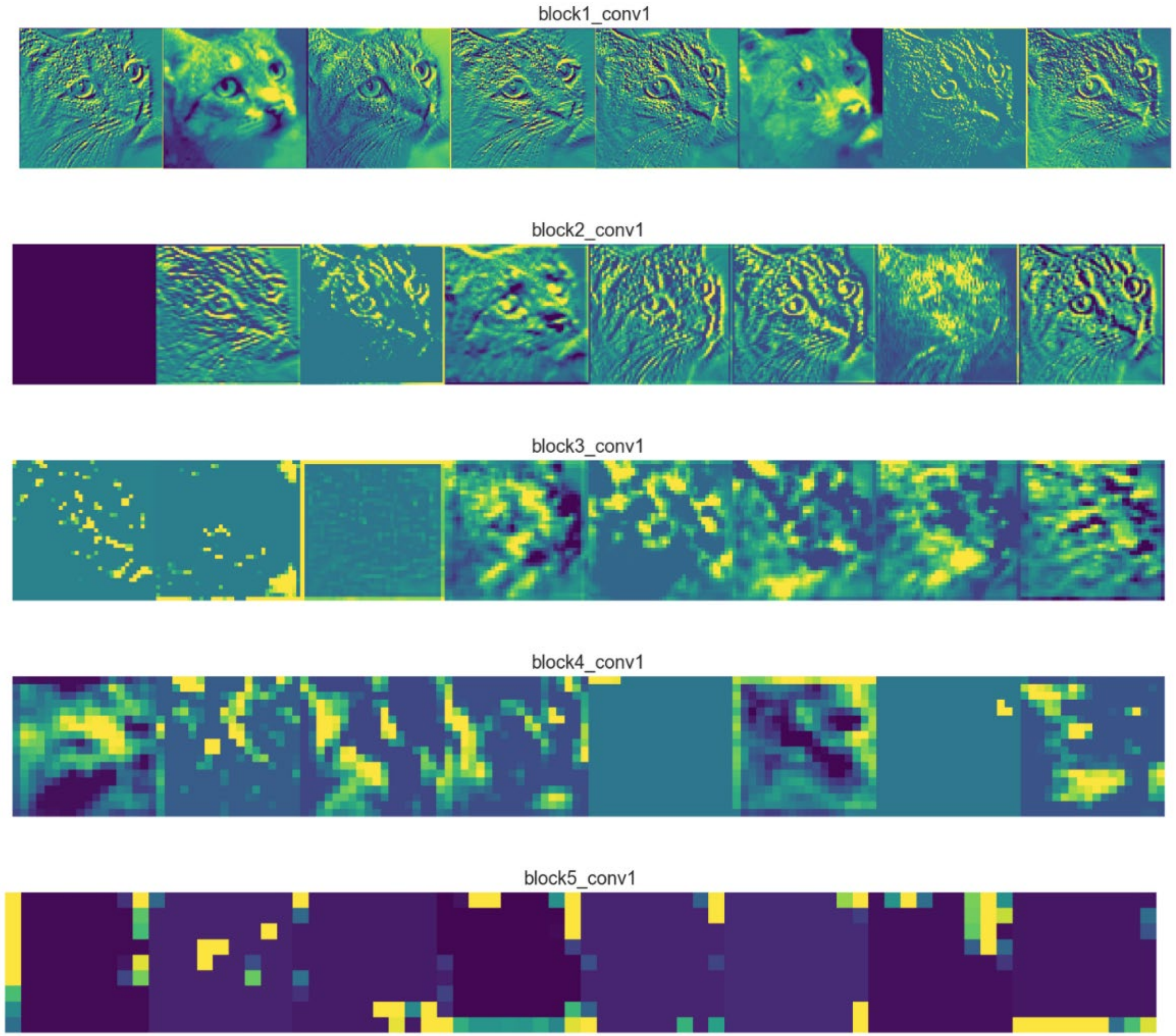
CS231n Convolutional Neural Networks for Visual Recognition

Visualization of Learned Feature Maps



[Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat | Towards Data Science]

Visualization of Learned Feature Maps



[Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat | Towards Data Science]

ImageNet Large Scale Visual Recognition Challenge

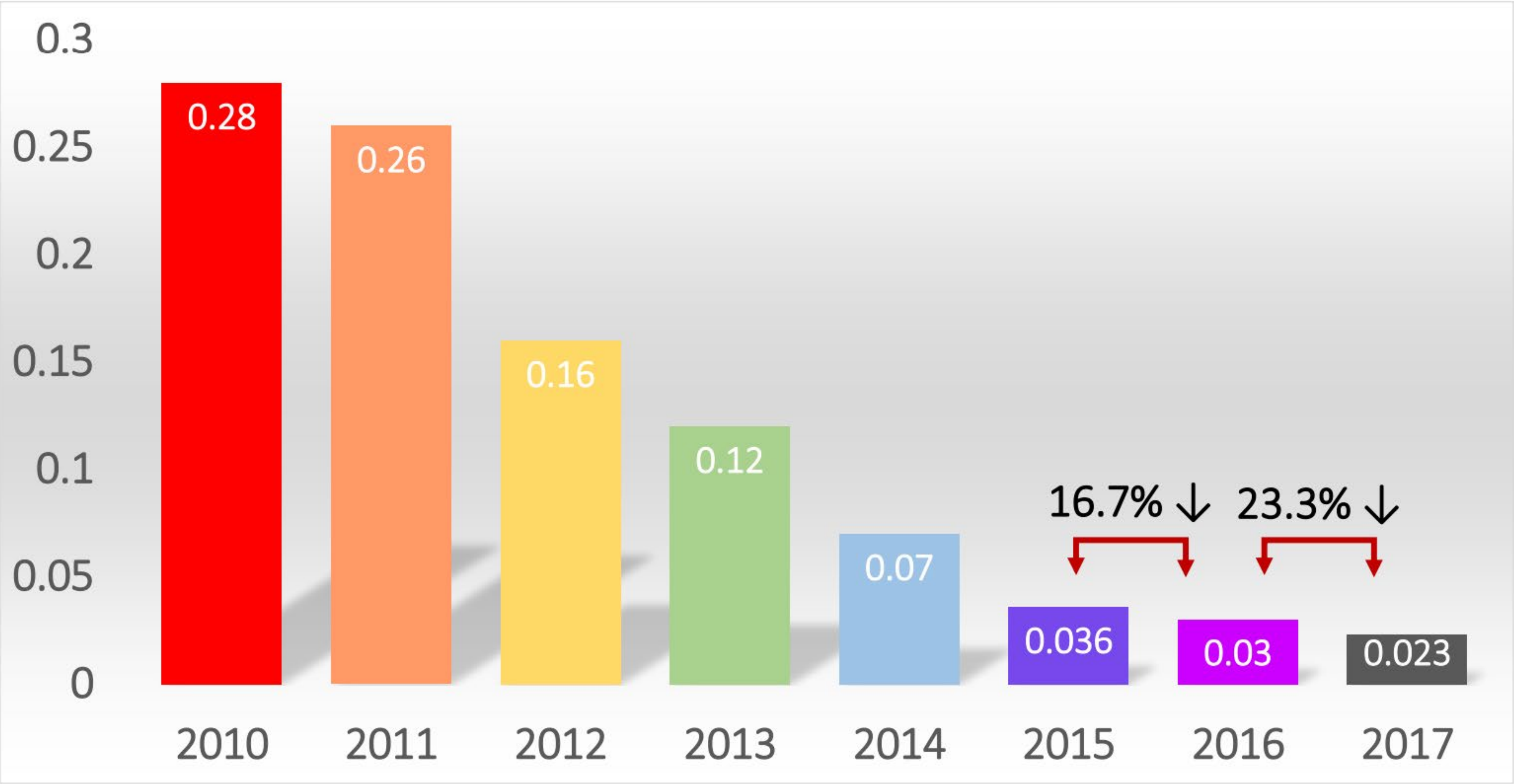
(ILSVRC)

ILSVRC

- ImageNet is an image database organized according to the WordNet hierarchy (nouns)
  - 1000 object classes
  - About 1.2M training images, 50K validation images, 100K test images

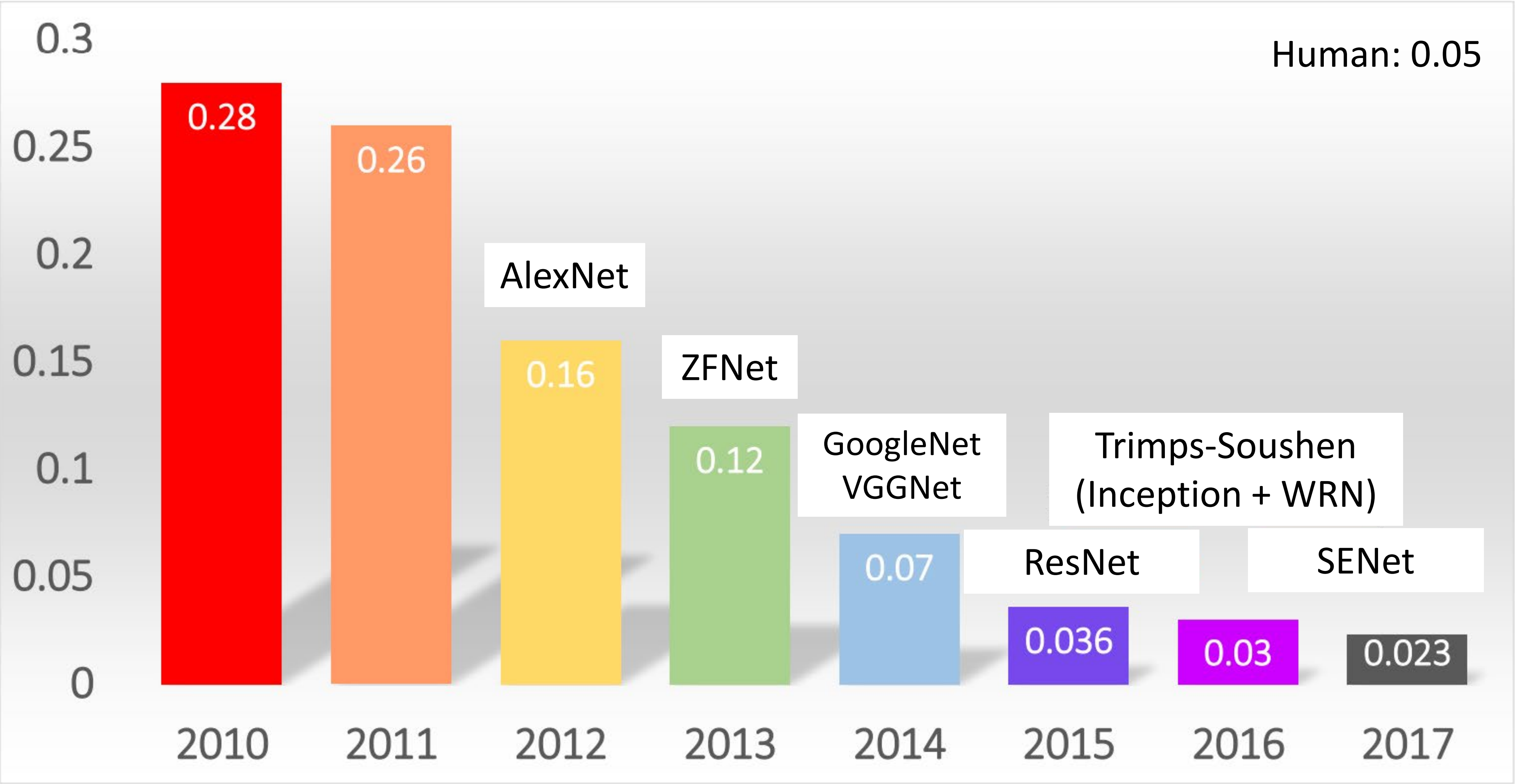
- The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
  - 8 years history (2010 – 2017)
  - It was the most powerful driving force to facilitate deep learning research

Classification Results



[Beyond ILSVRC workshop 2017 (image-net.org)](image-net.org)

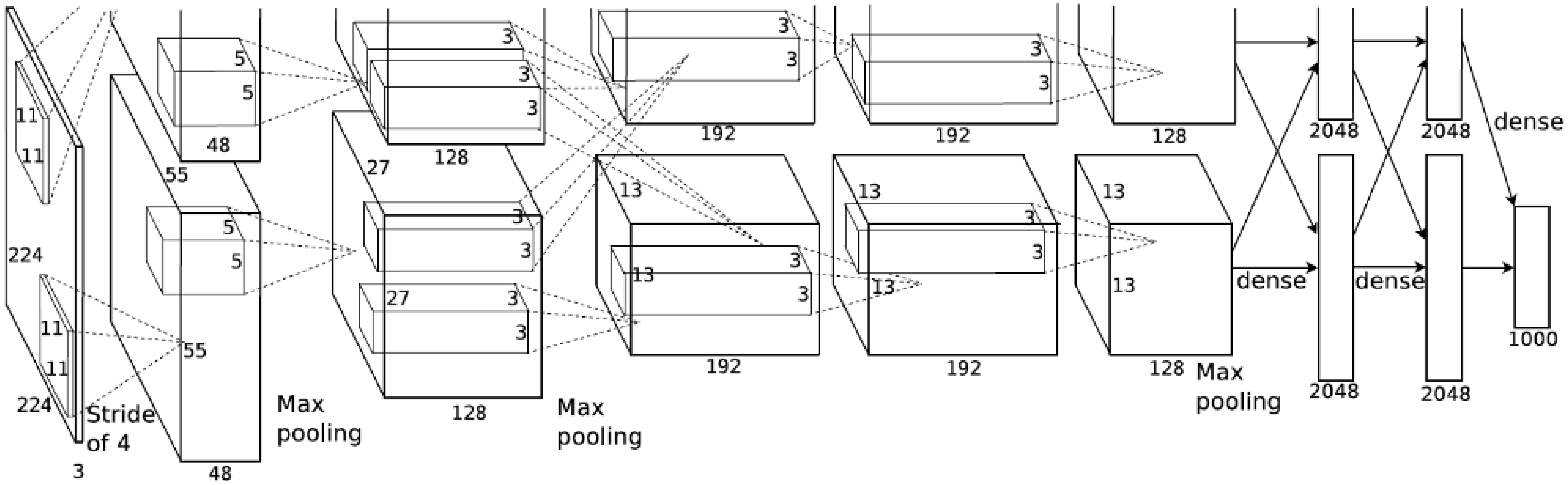
Classification Results



Beyond ILSVRC workshop 2017 (image-net.org)

AlexNet

- The winner of ILSVRC 2012
- It changed the entire computer vision research

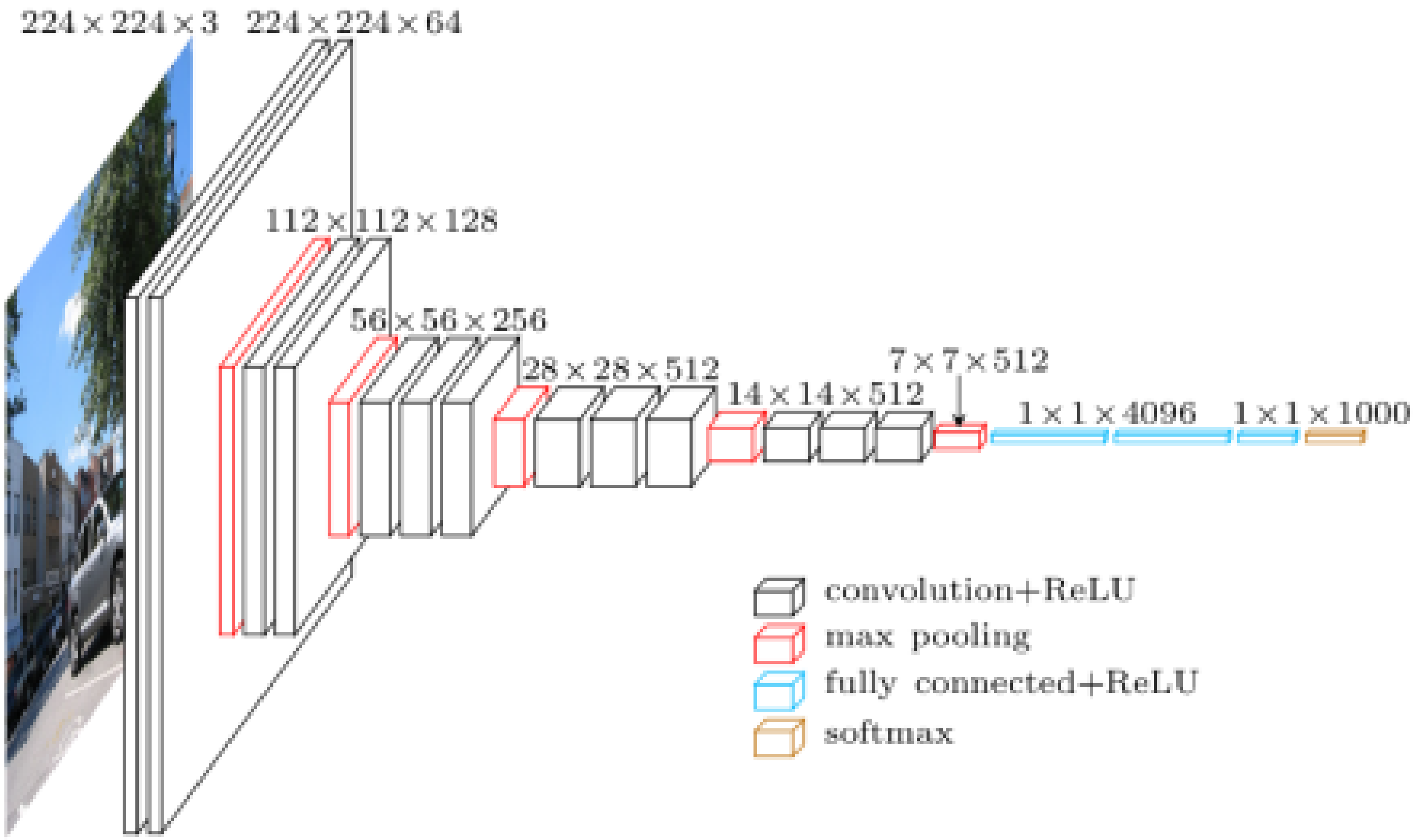


ZFNet

- The winner of ILSVRC 2013
- The network architectures were developed by using the visualization techniques
  - Visualizing and Understanding Convolutional Networks, Zeiler et al, ECCV 2014

- Reduced the 1st layer filter size from 11x11 to 7x7
- 1st layer stride from 4 -> 2

VGGNet



[Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet | by Khush Patel | Towards Data Science]

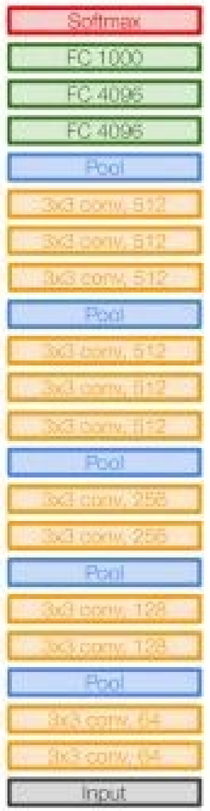
VGGNet

[Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet | by Khush Patel | Towards Data Science]

VGGNet

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters



VGG16

[Architecture comparison of AlexNet, VGGNet, ResNet, Inception, DenseNet | by Khush Patel | Towards Data Science]

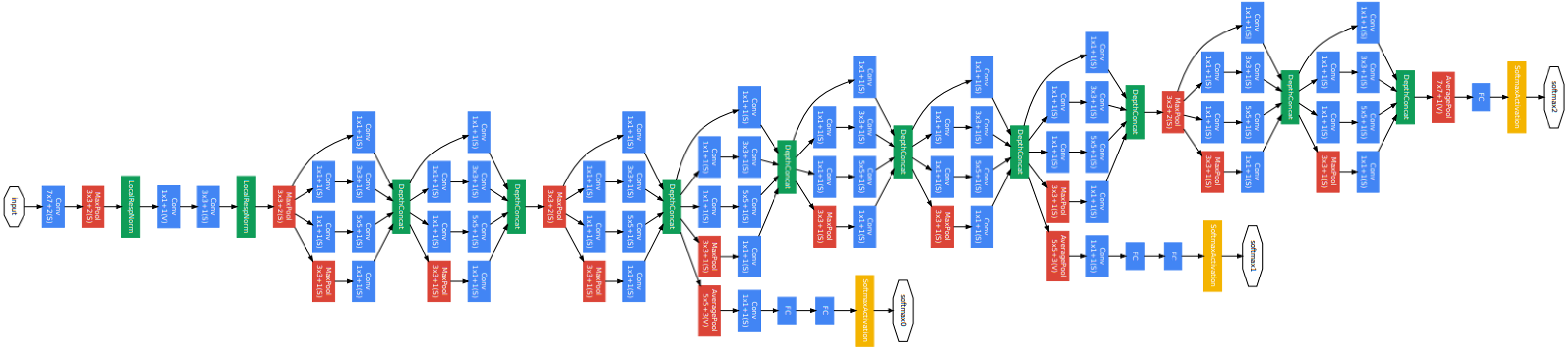
GoogLeNet

- Winner of ISLVRC 2014
- Also called ‘Inception’

Max pooling

Concatenation

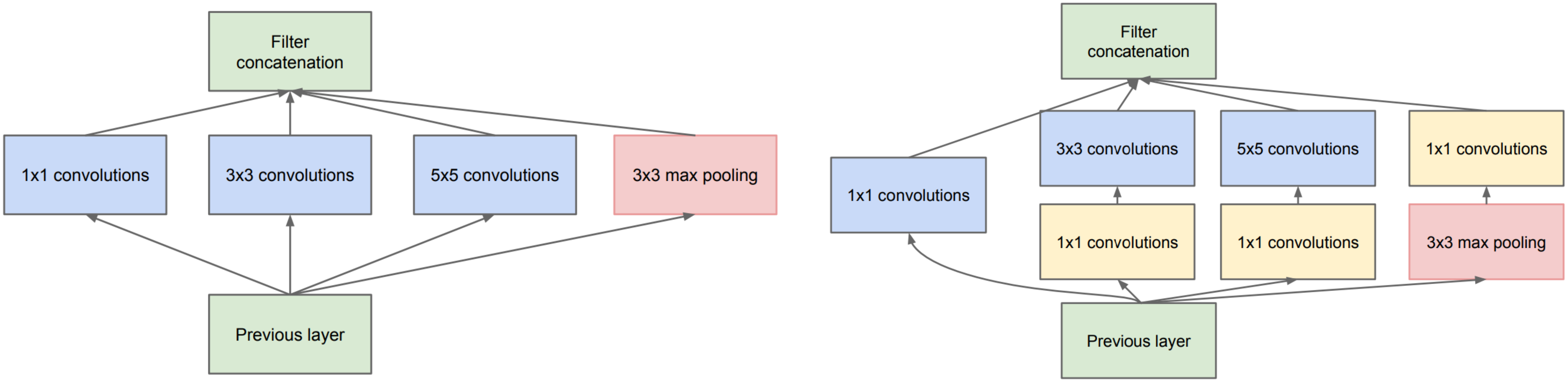
Convolution



Going deeper with convolutions, Szegedy et al, CVPR 2015

GoogLeNet

- Inception module



(a) Inception module, naïve version

(b) Inception module with dimensionality reduction

Going deeper with convolutions, Szegedy et al, CVPR 2015

GoogLeNet

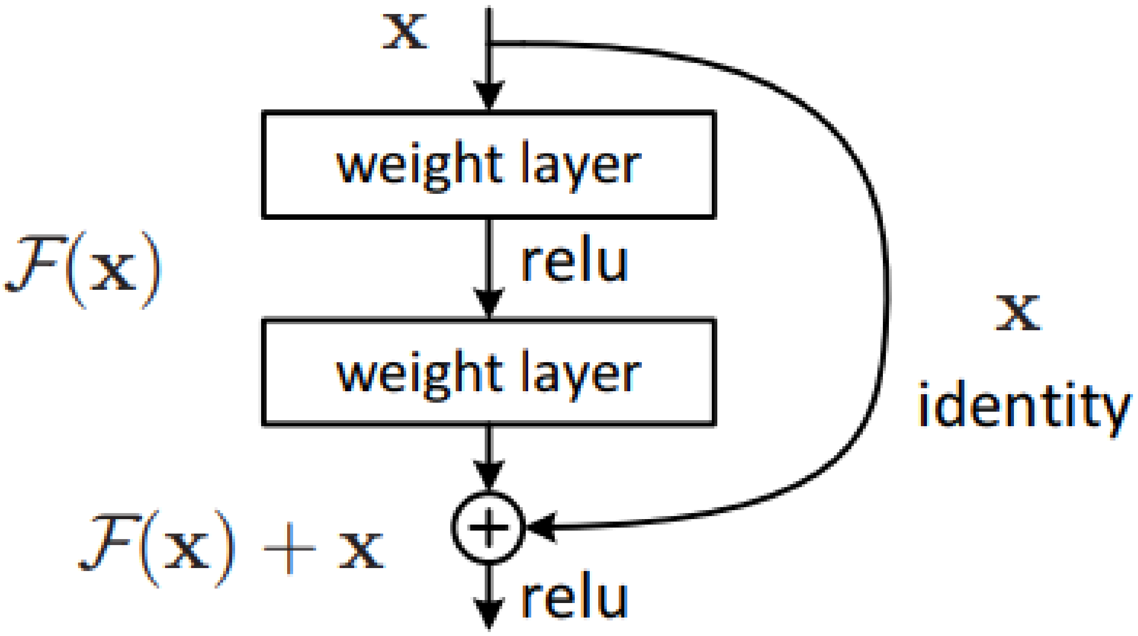
- ILSVRC 2014 classification results

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Going deeper with convolutions, Szegedy et al, CVPR 2015

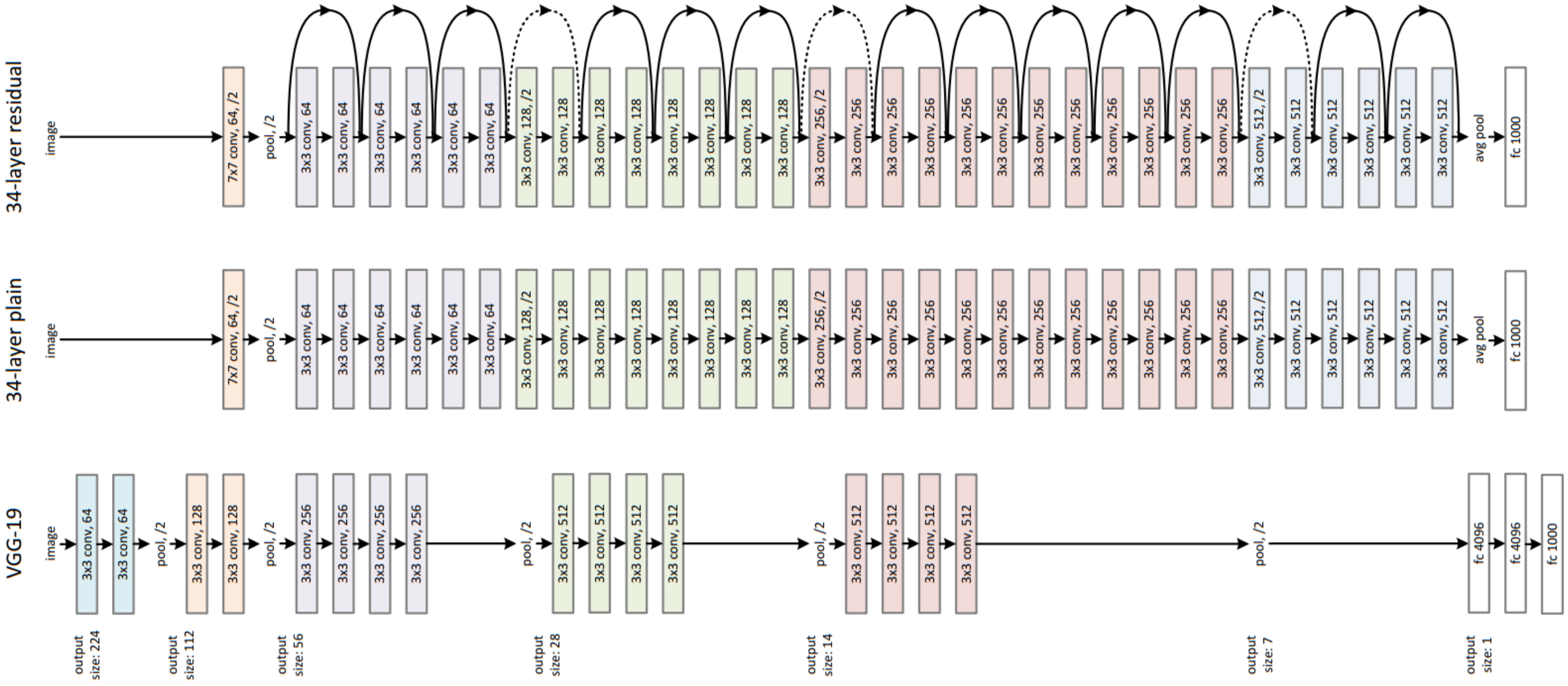
ResNet

- The winner of ILSVRC 2015
- Residual building block



Deep residual learning for image recognition, He et al, CVPR 2016

ResNet



Deep residual learning for image recognition, He et al, CVPR 2016

ResNet

- Training on ImageNet

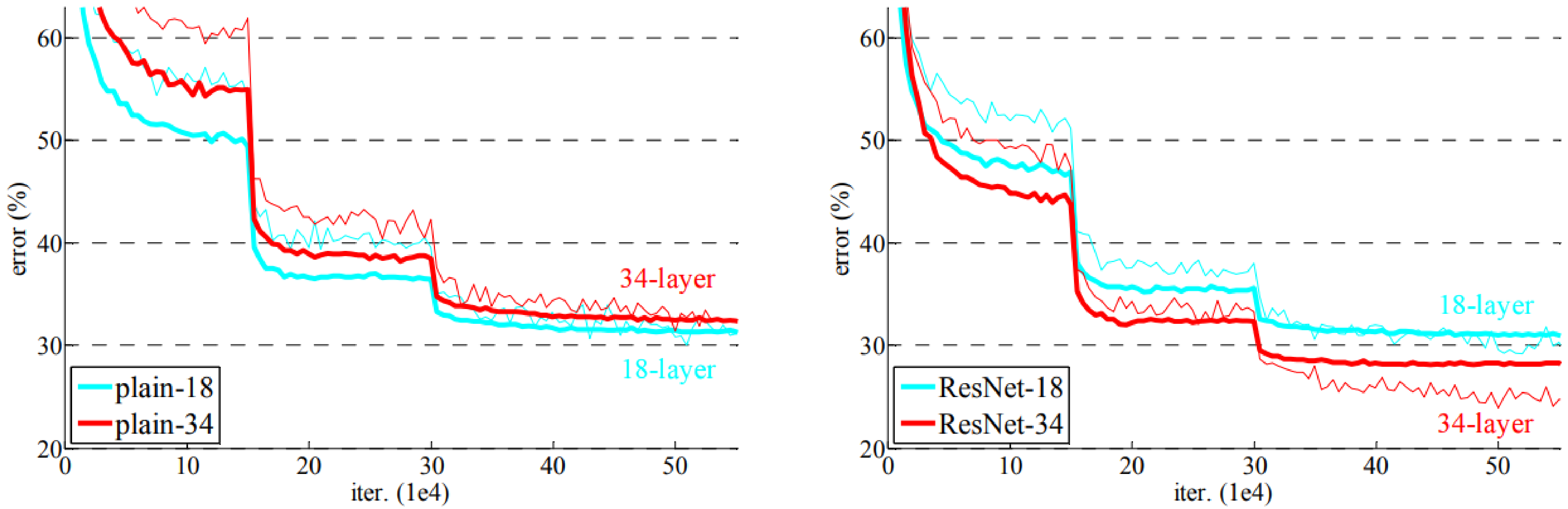


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Deep residual learning for image recognition, He et al, CVPR 2016

ResNet

- ILSVRC 2015 classification results

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

Deep residual learning for image recognition, He et al, CVPR 2016