MIT · 6.036 | Introduction to Machine Learning (2020)

MIT 6.036(2020)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1y44y187wN

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/mit-6.036

机器学习 循环神经 神经网络 感知器
特征构建 聚类 马尔可夫决策过程 网络
随机森林 决策树 逻辑回归 卷积神经网络 状态机

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]

6.036/6.862: Introduction to Machine Learning

Lecture: starts Tuesdays 9:35am (Boston time zone)

Course website: introml.odl.mit.edu

Who's talking? Prof. Tamara Broderick

Questions? discourse.odl.mit.edu ("Lecture 8" category)

Materials: Will all be available at course website

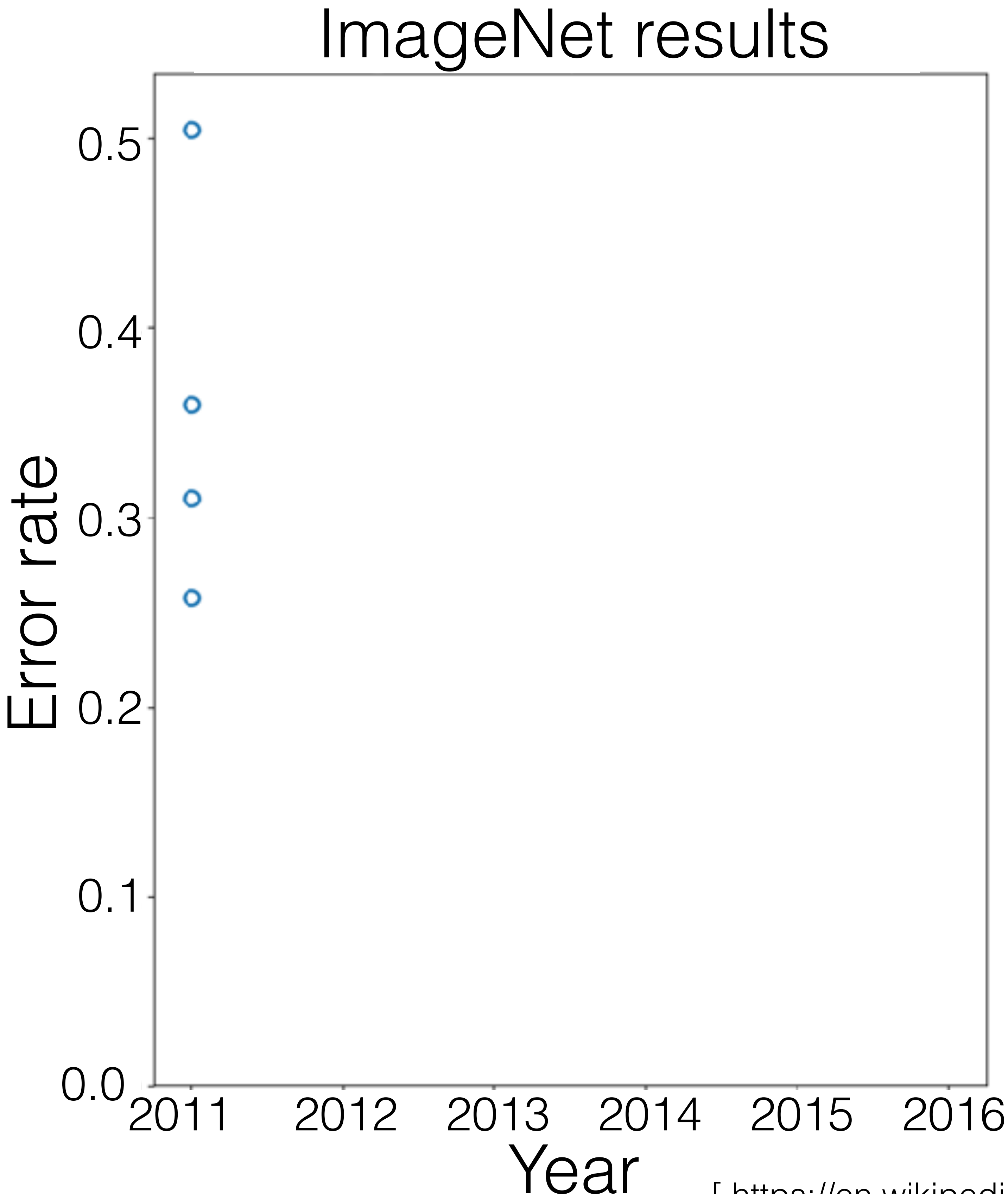
Last Time

I. Neural networks
   - 2 layers
   - Fully connected
   - Learning

Today's Plan

I. CNNs/ConvNets: hypothesis class
II. Filters & max pooling
III. Learning

Impact of CNNs

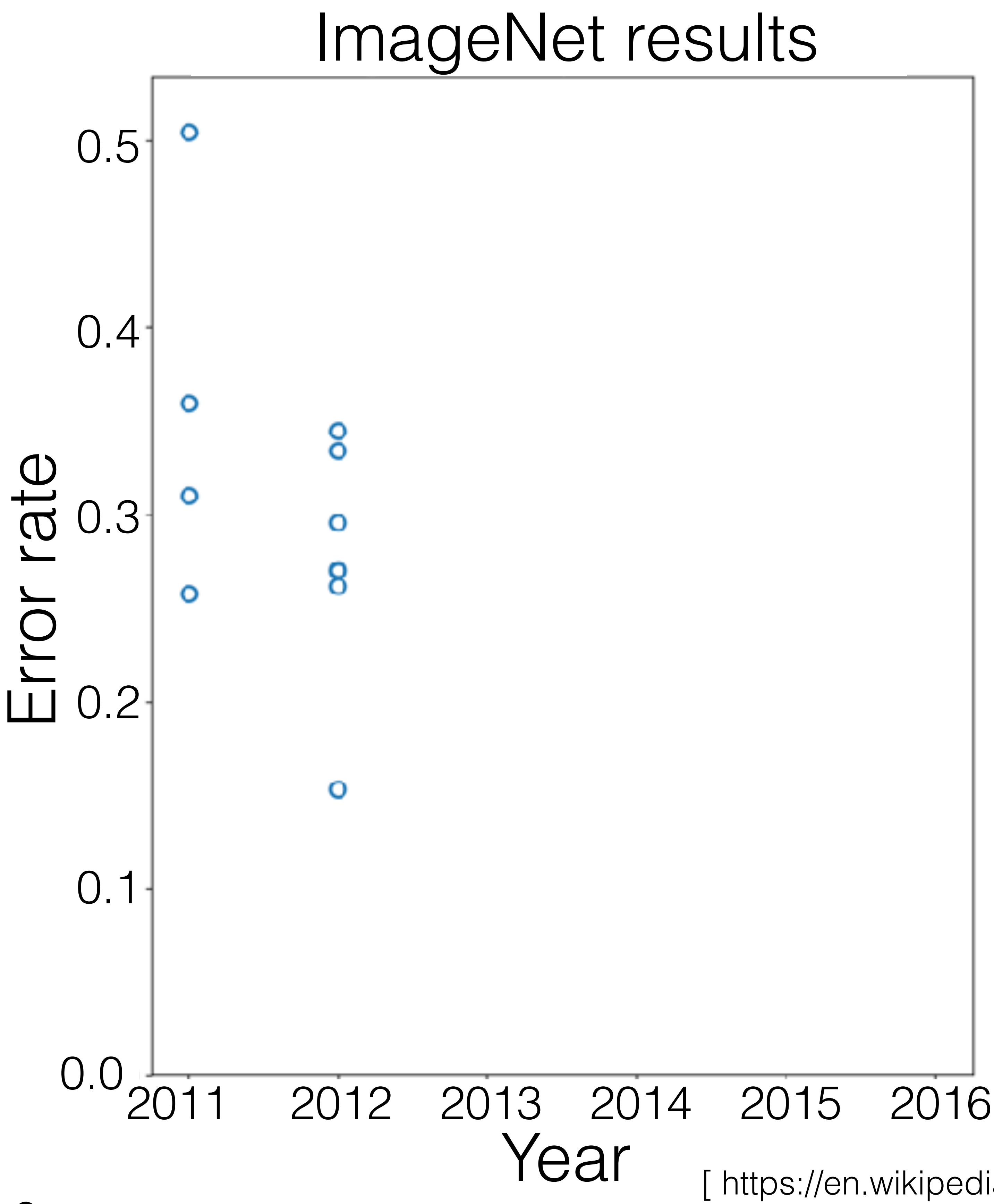


- Since 2010: large-scale image classification challenge

[https://en.wikipedia.org/wiki/ImageNet#History_of_the_ImageNet_Challenge]

[Russakovsky et al, "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015]

Impact of CNNs

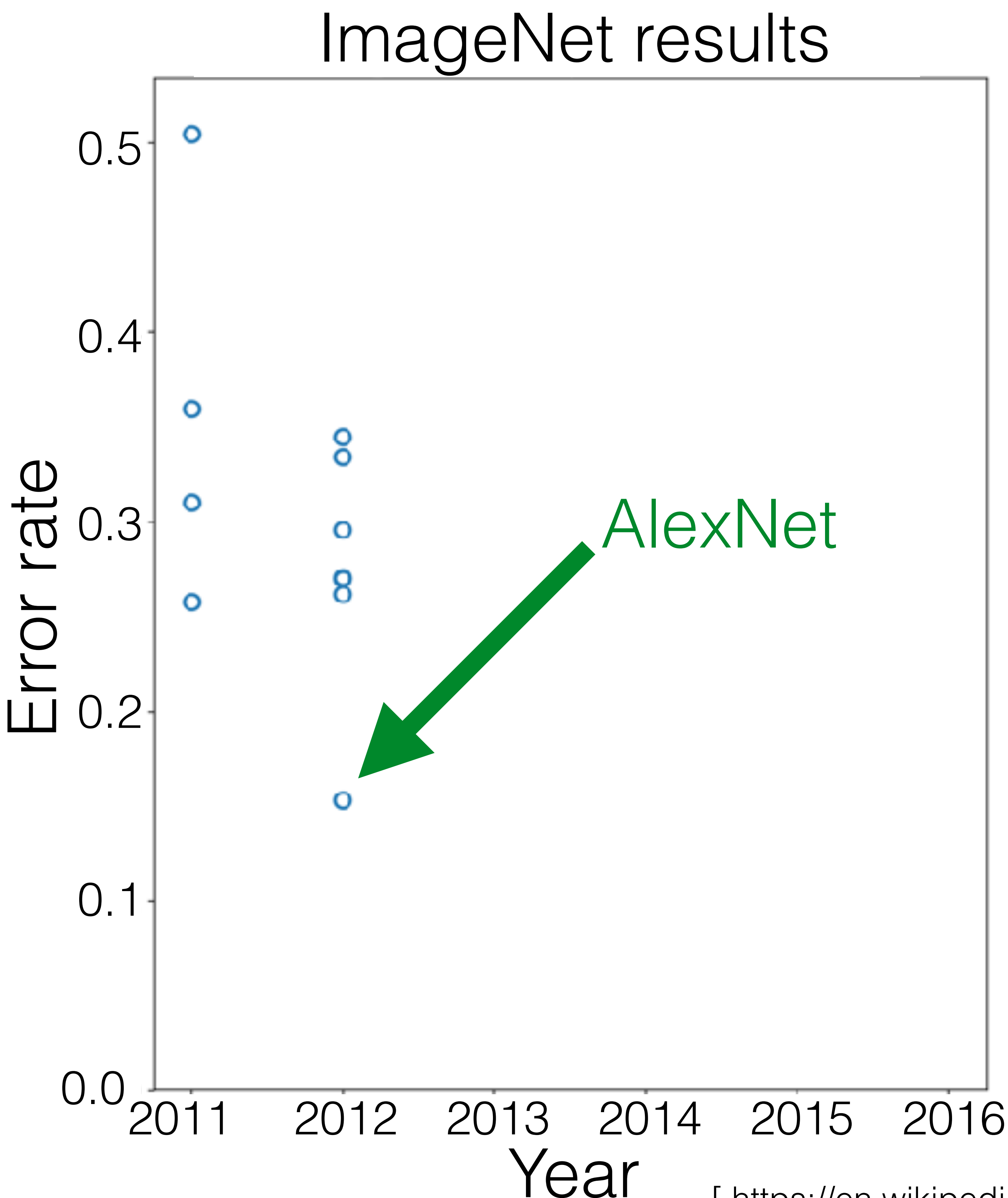


- Since 2010: large-scale image classification challenge

[https://en.wikipedia.org/wiki/ImageNet#History_of_the_ImageNet_Challenge]

[Russakovsky et al, "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015]

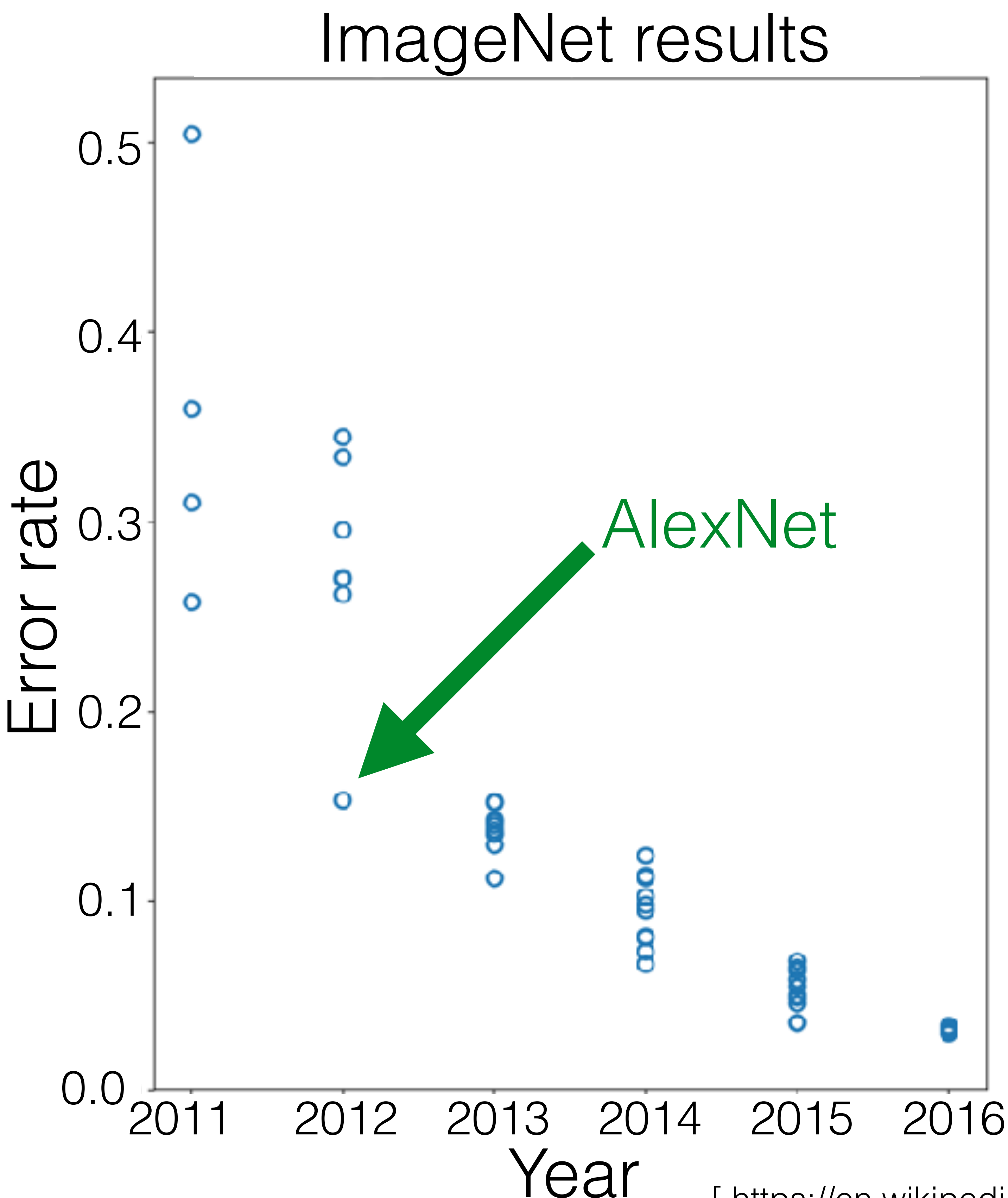
Impact of CNNs



- Since 2010: large-scale image classification challenge

[https://en.wikipedia.org/wiki/ImageNet#History_of_the_ImageNet_Challenge]
[Russakovsky et al, "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015]

Impact of CNNs

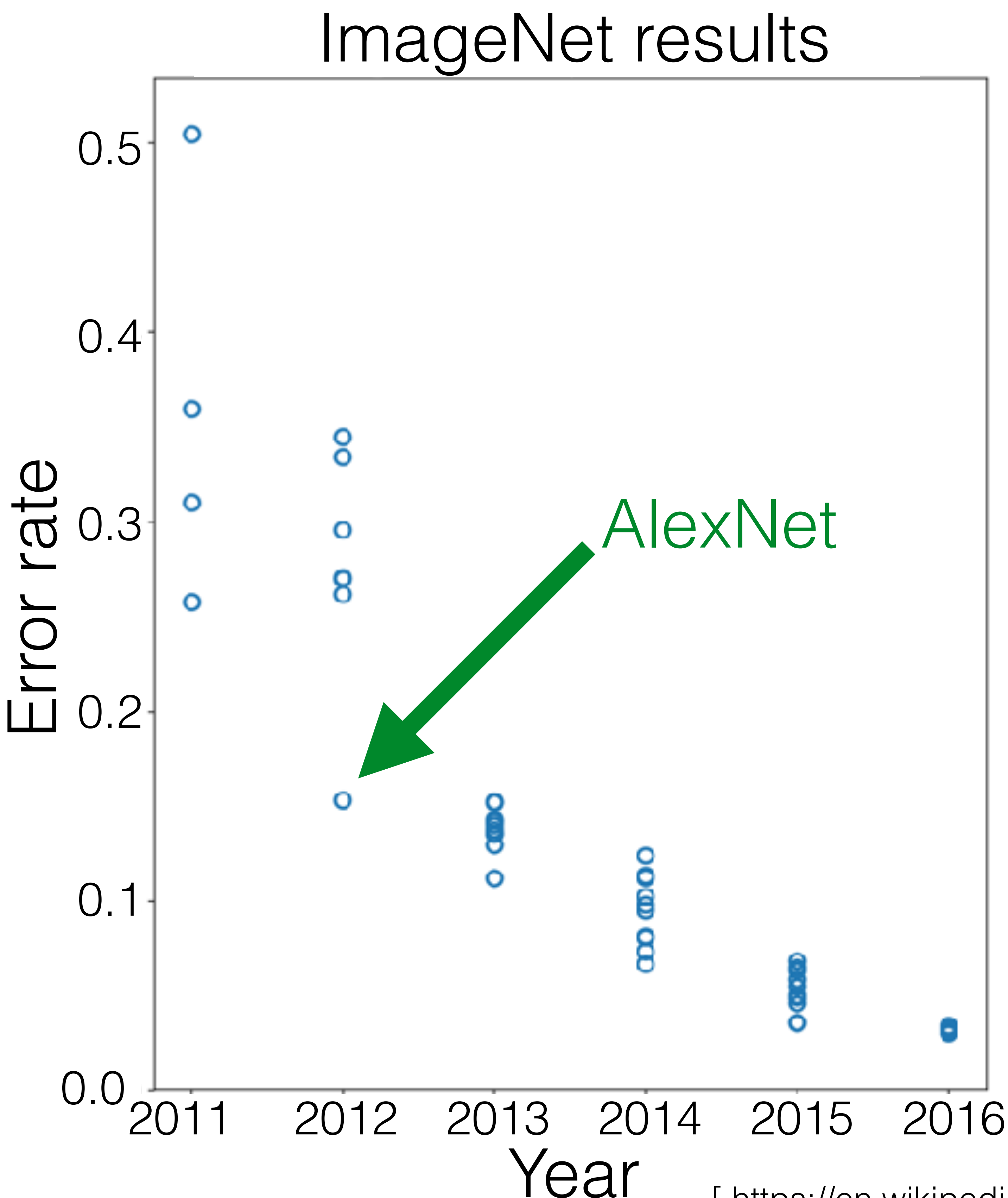


- Since 2010: large-scale image classification challenge

[https://en.wikipedia.org/wiki/ImageNet#History_of_the_ImageNet_Challenge]

[Russakovsky et al, "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015]

Impact of CNNs

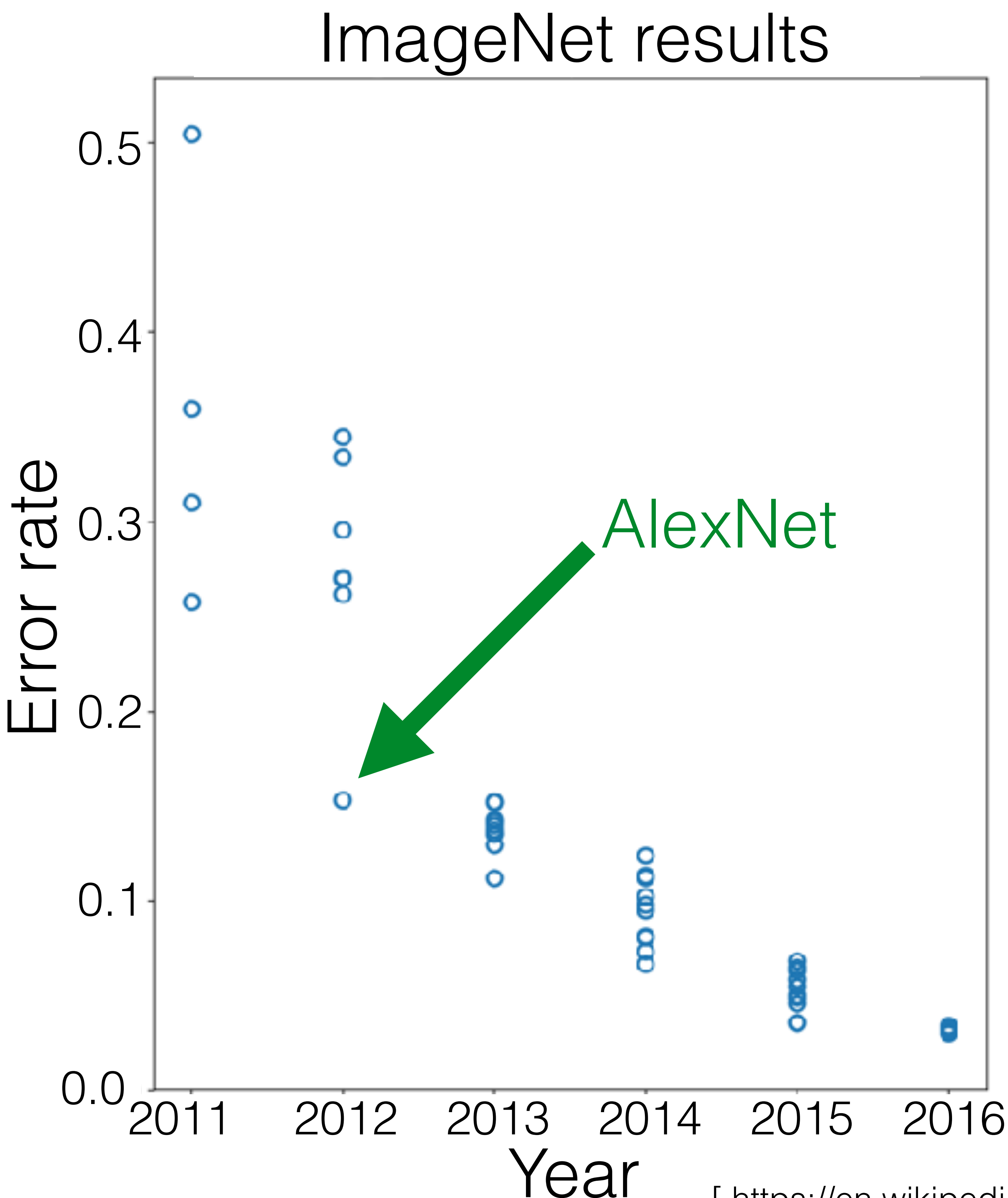


- Since 2010: large-scale image classification challenge
- Recent AI boom

[https://en.wikipedia.org/wiki/ImageNet#History_of_the_ImageNet_Challenge]

[Russakovsky et al, "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015]

Impact of CNNs



- Since 2010: large-scale image classification challenge
- Recent AI boom
- 1960s, 1980s, today: neural networks
- Since 1980s: CNNs

[https://en.wikipedia.org/wiki/ImageNet#History_of_the_ImageNet_Challenge]
[Russakovsky et al, "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015]

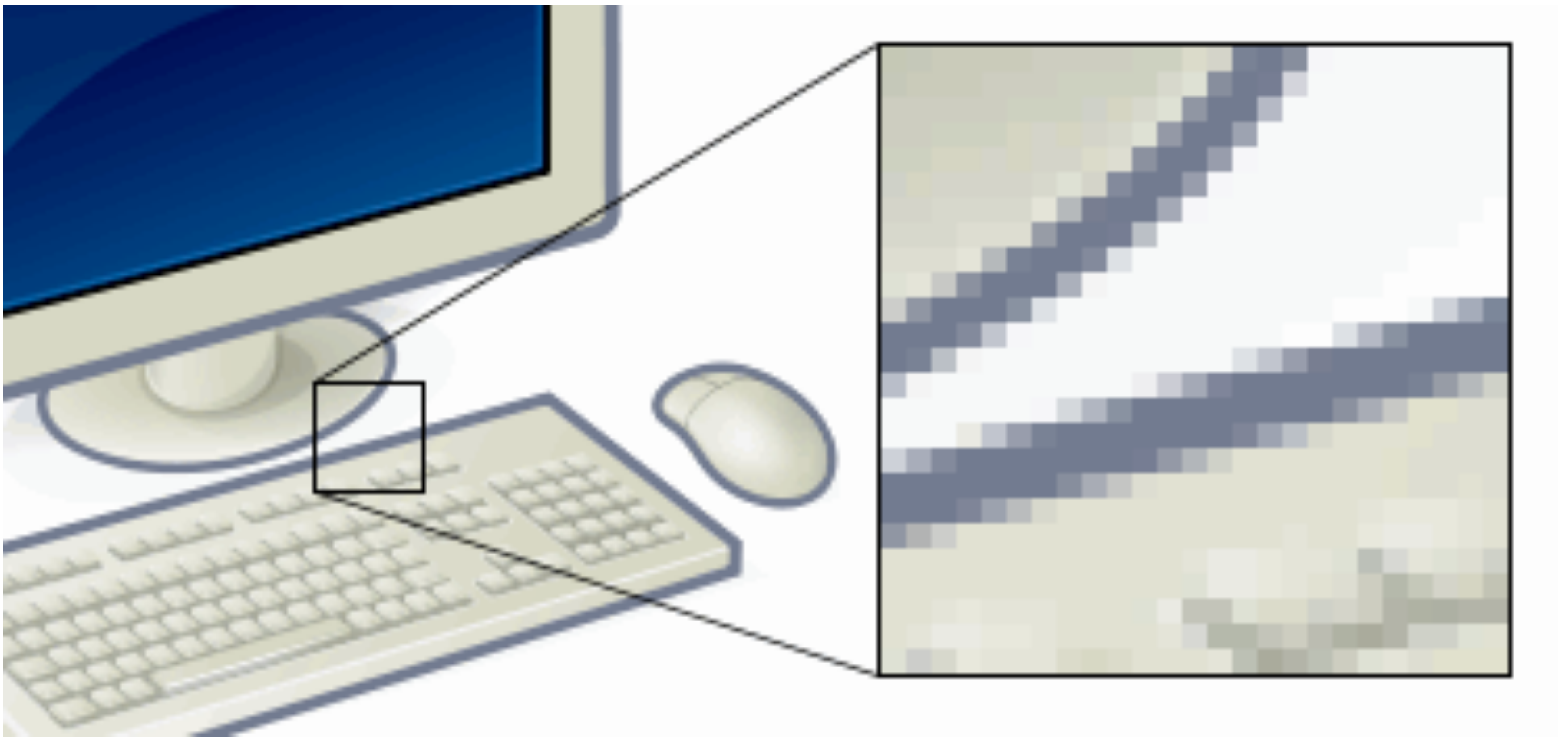
Images

- Potential uses of image classification: Detect tumor (type) from medical scans, image search online, autonomous driving

[https://en.wikipedia.org/wiki/Pixel#/media/File:Pixel-example.png]

Images

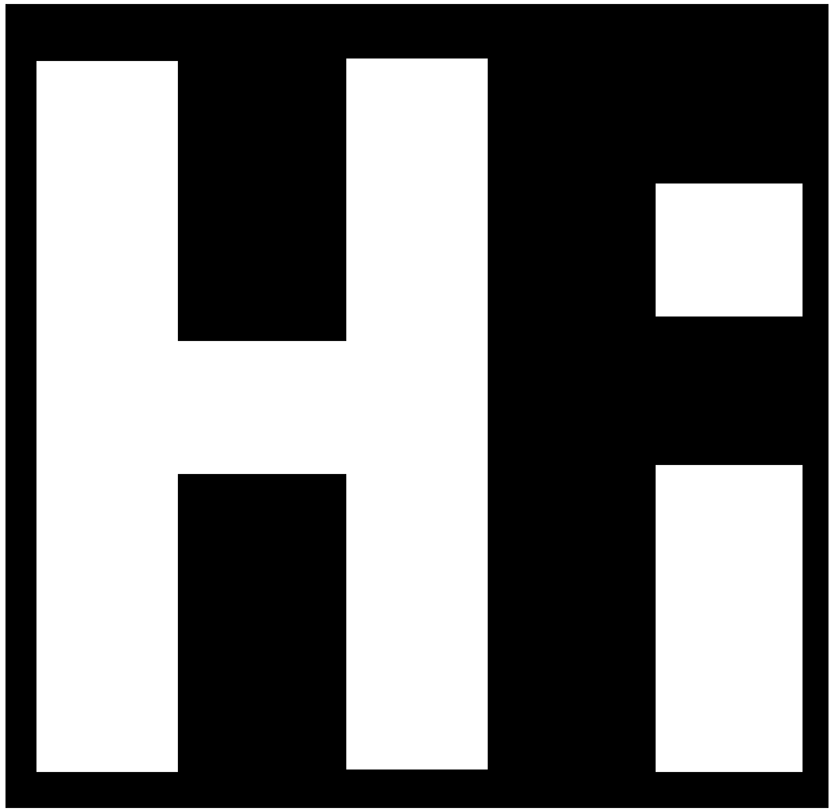
- Potential uses of image classification: Detect tumor (type) from medical scans, image search online, autonomous driving



- Recall: images are made of pixels

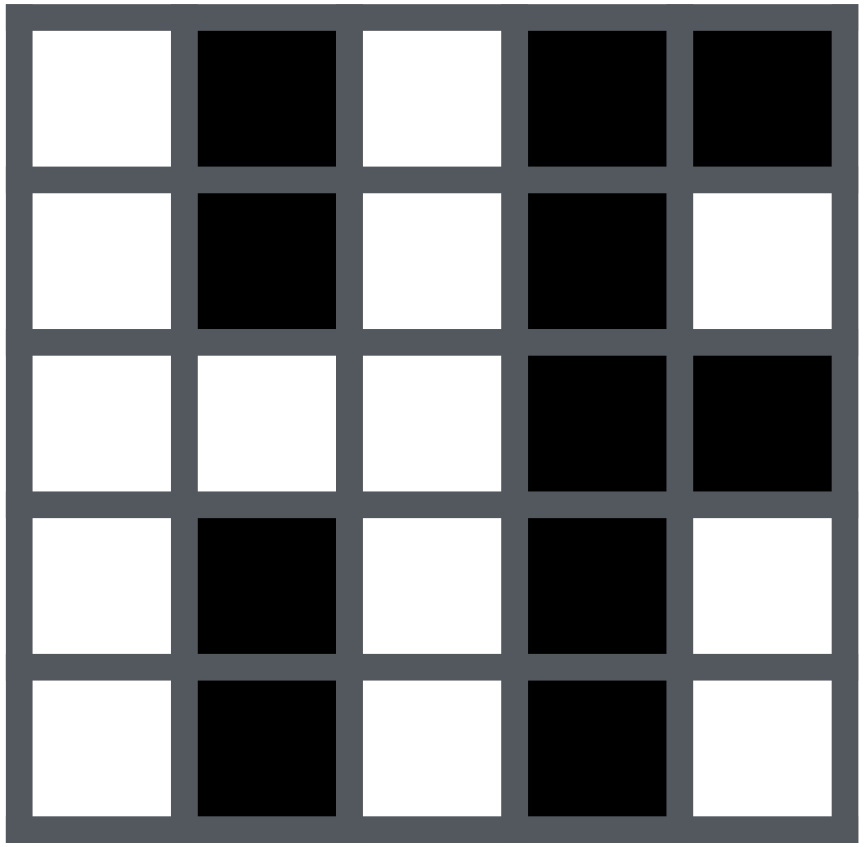
[https://en.wikipedia.org/wiki/Pixel#/media/File:Pixel-example.png]

Images



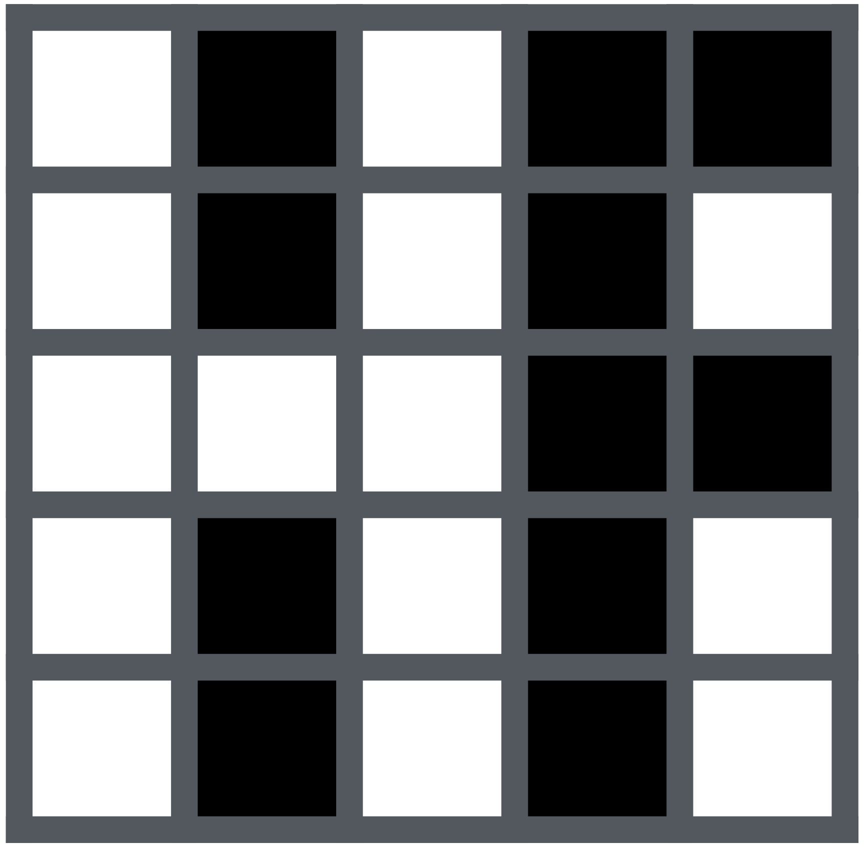
- We'll focus on grayscale images

Images



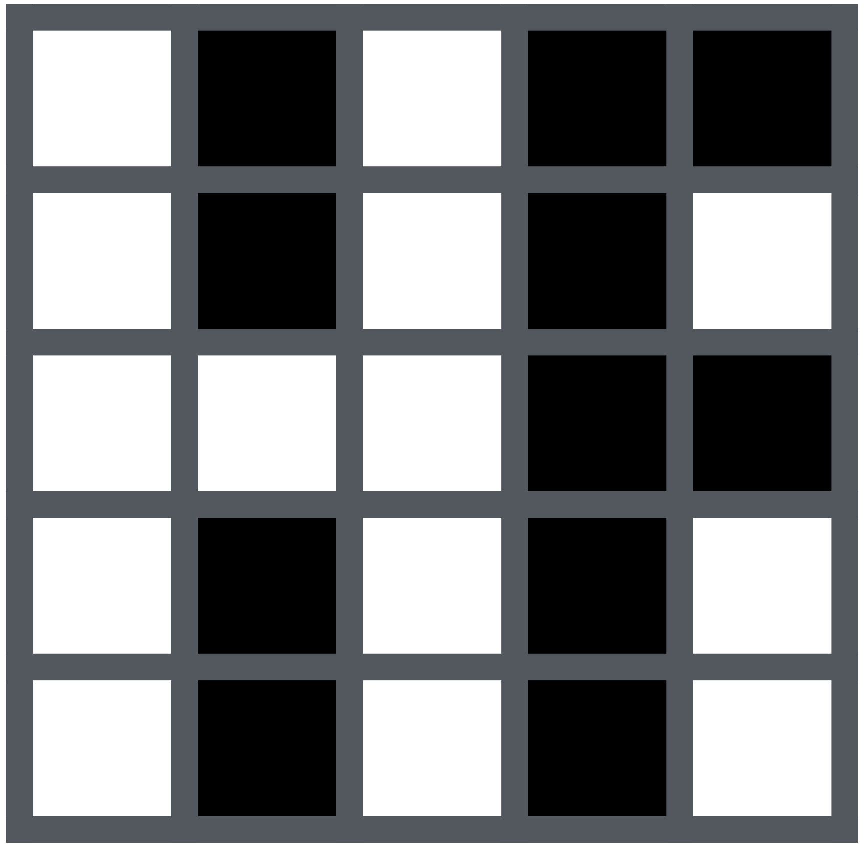
- We'll focus on grayscale images

Images



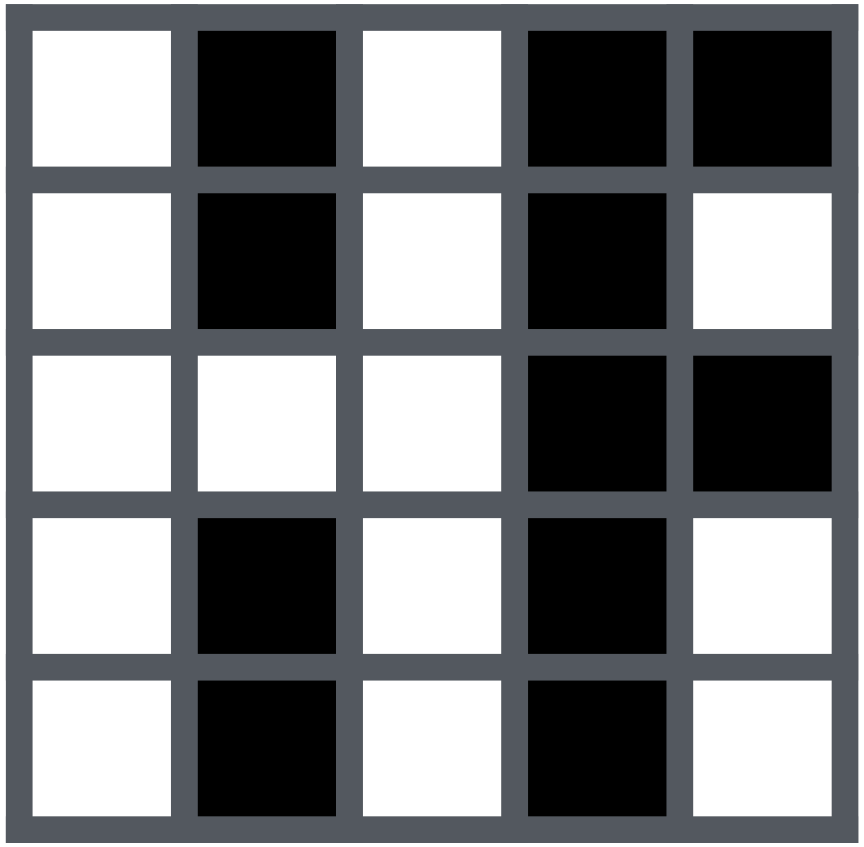
- We'll focus on grayscale images
  - Each pixel takes a value between 0 and *P*

Images



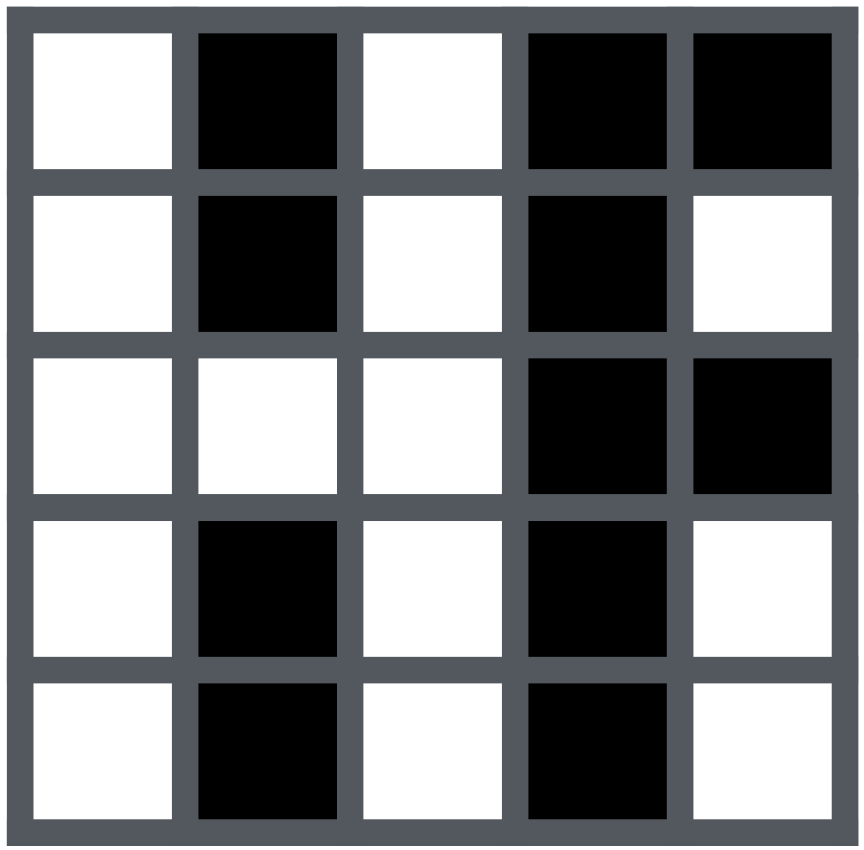
- We'll focus on grayscale images
  - Each pixel takes a value between 0 and P
  - Here, 0: black, 1: white

Images



- We’ll focus on grayscale images
  - Each pixel takes a value between 0 and P
  - Here, 0: black, 1: white
  - Larger P in Lab Week 08

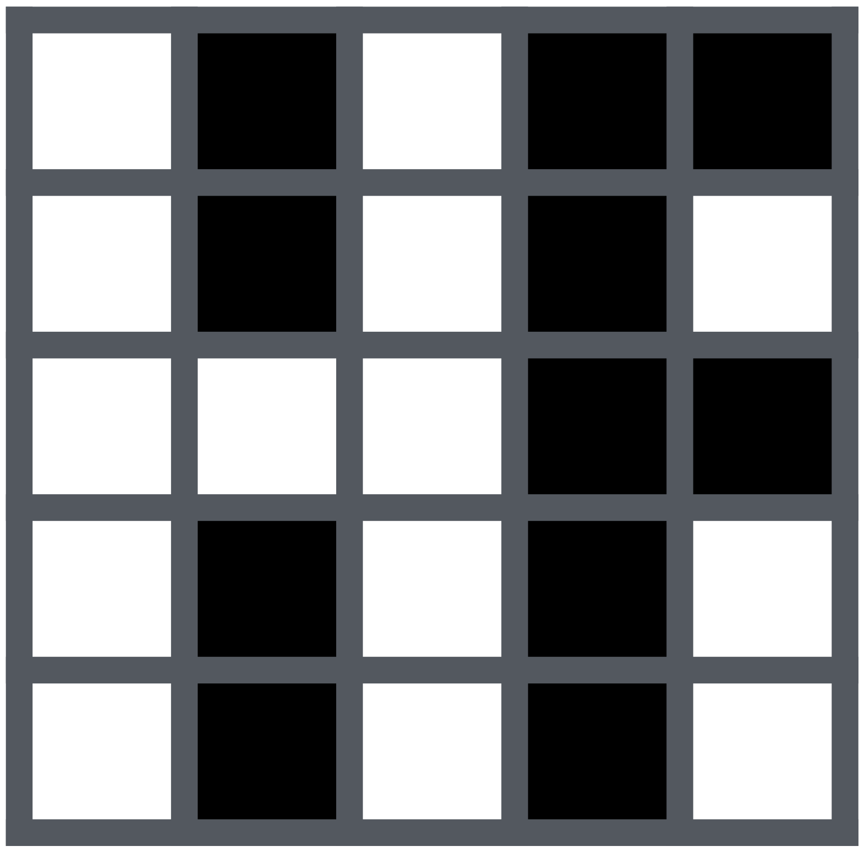
Images



1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

- We'll focus on grayscale images
    - Each pixel takes a value between 0 and *P*
    - Here, 0: black, 1: white
    - Larger *P* in Lab Week 08

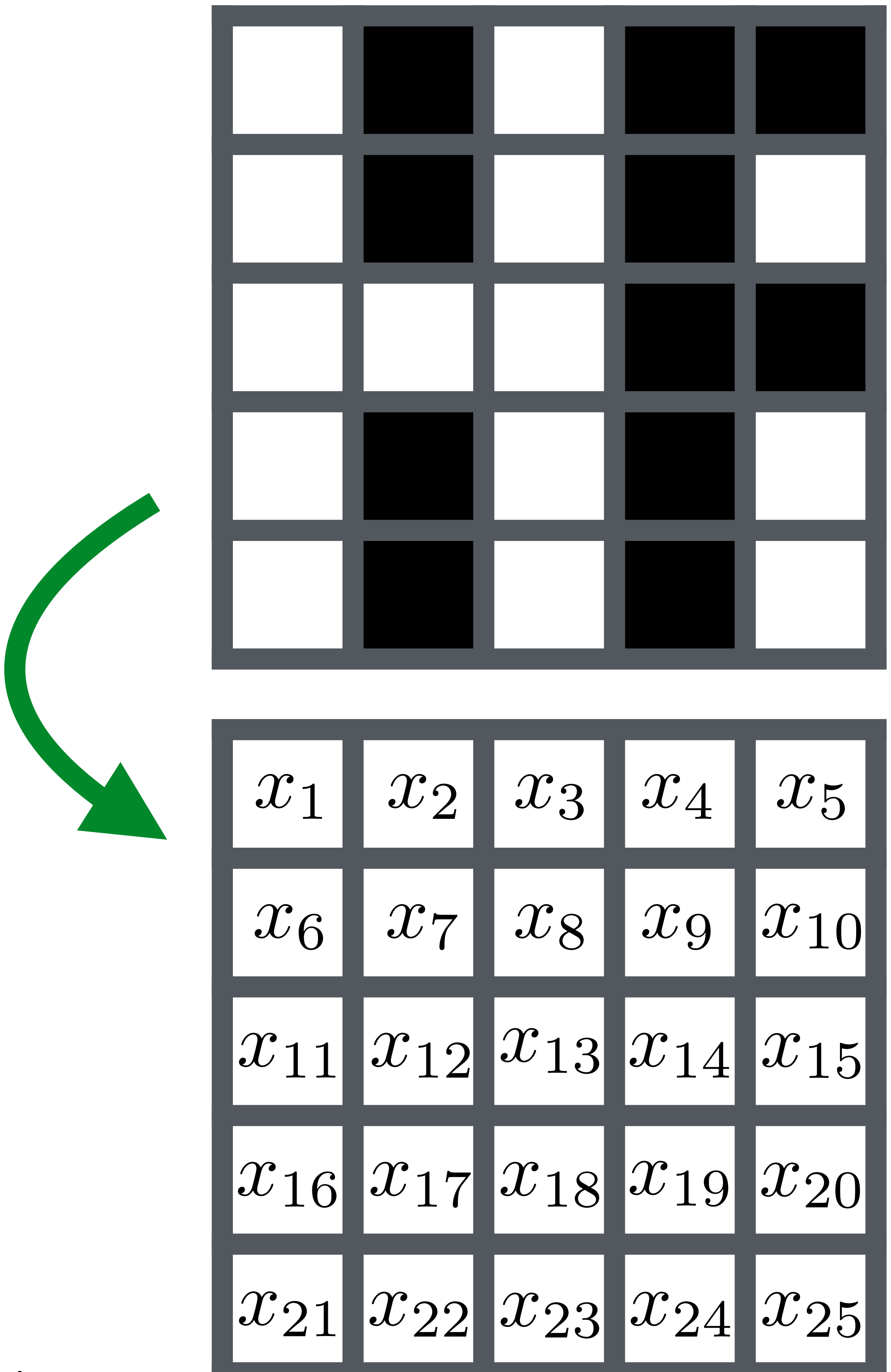
Images



1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

- We'll focus on grayscale images
  - Each pixel takes a value between 0 and *P*
  - Here, 0: black, 1: white
  - Larger *P* in Lab Week 08
- How do we use an image as an input for a neural net?

Images



- We'll focus on grayscale images
  - Each pixel takes a value between 0 and P
  - Here, 0: black, 1: white
  - Larger P in Lab Week 08

- How do we use an image as an input for a neural net?

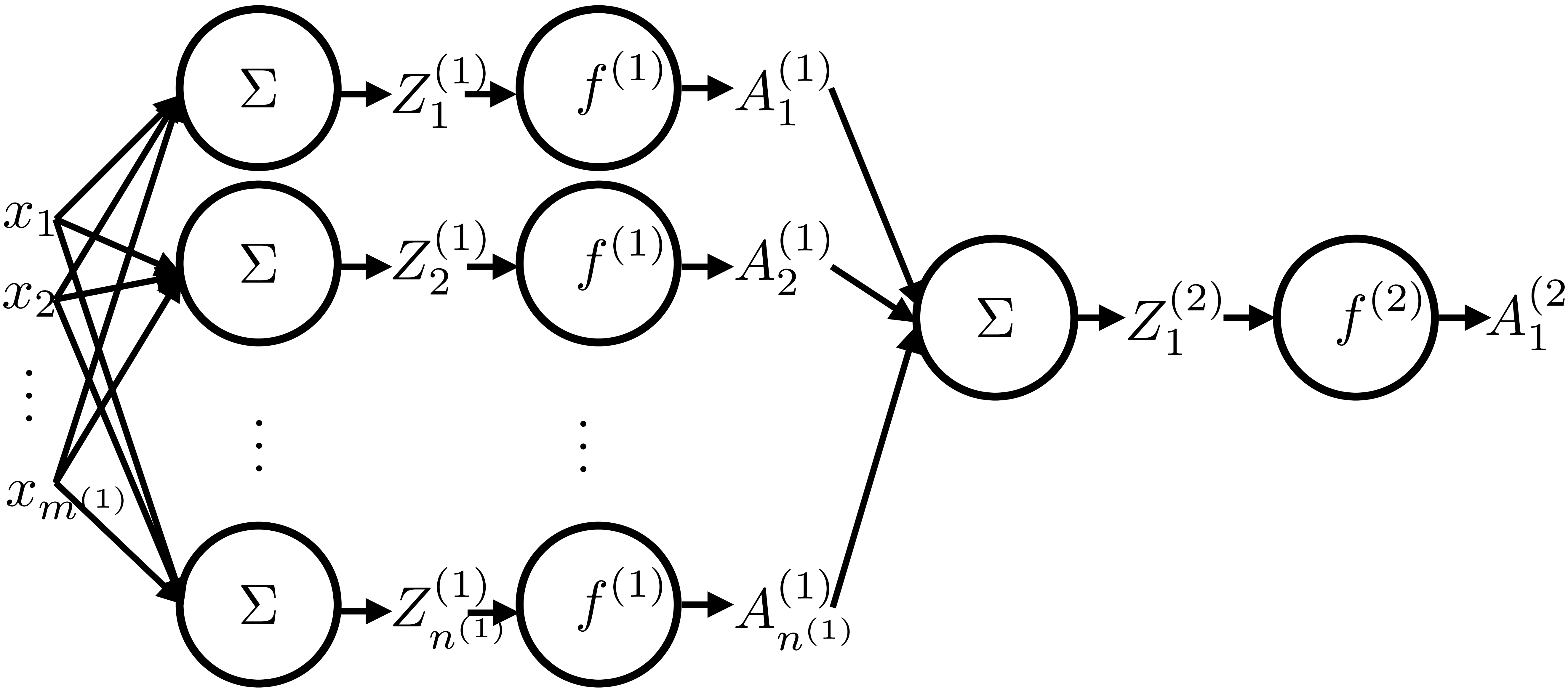
Previous neural nets in this class

Previous neural nets in this class

- Recall:

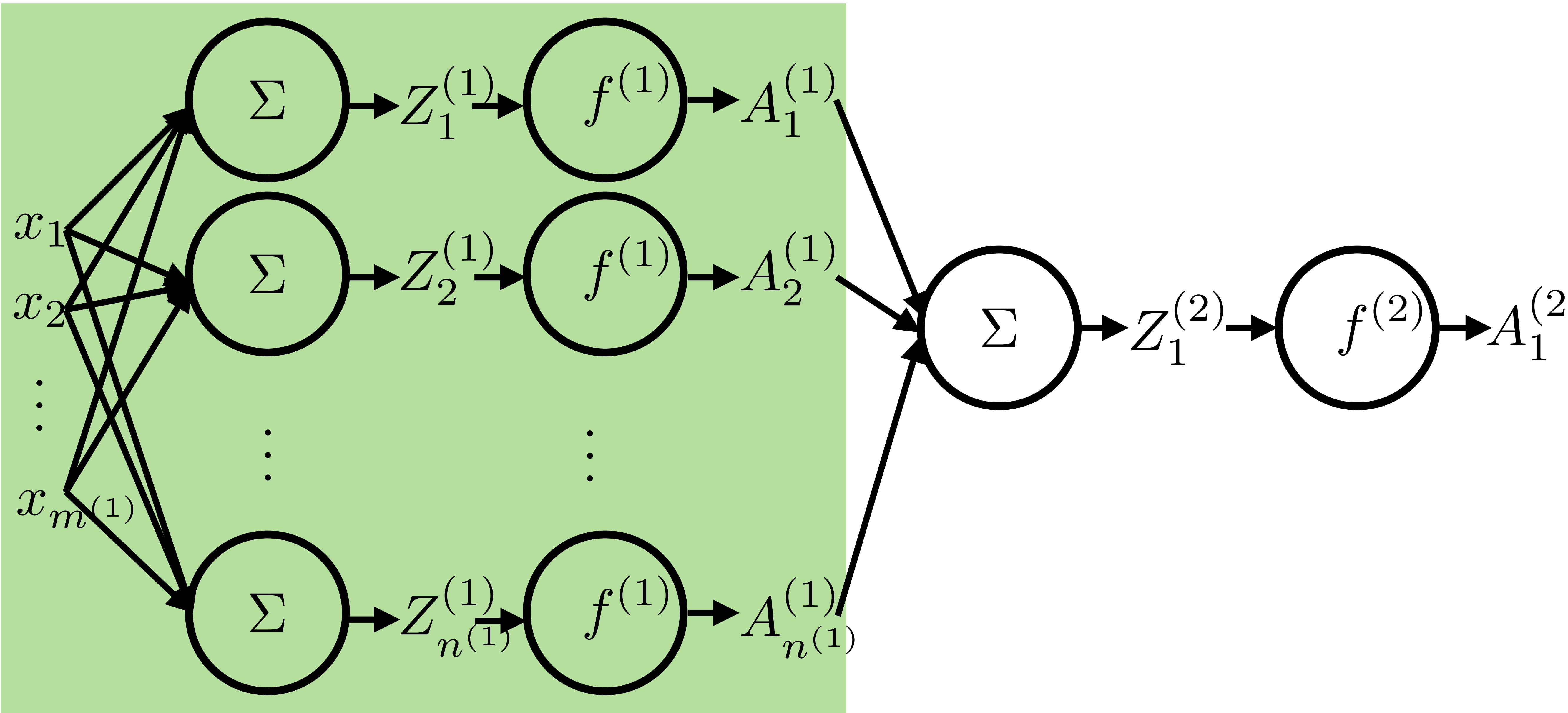
Previous neural nets in this class

- Recall:



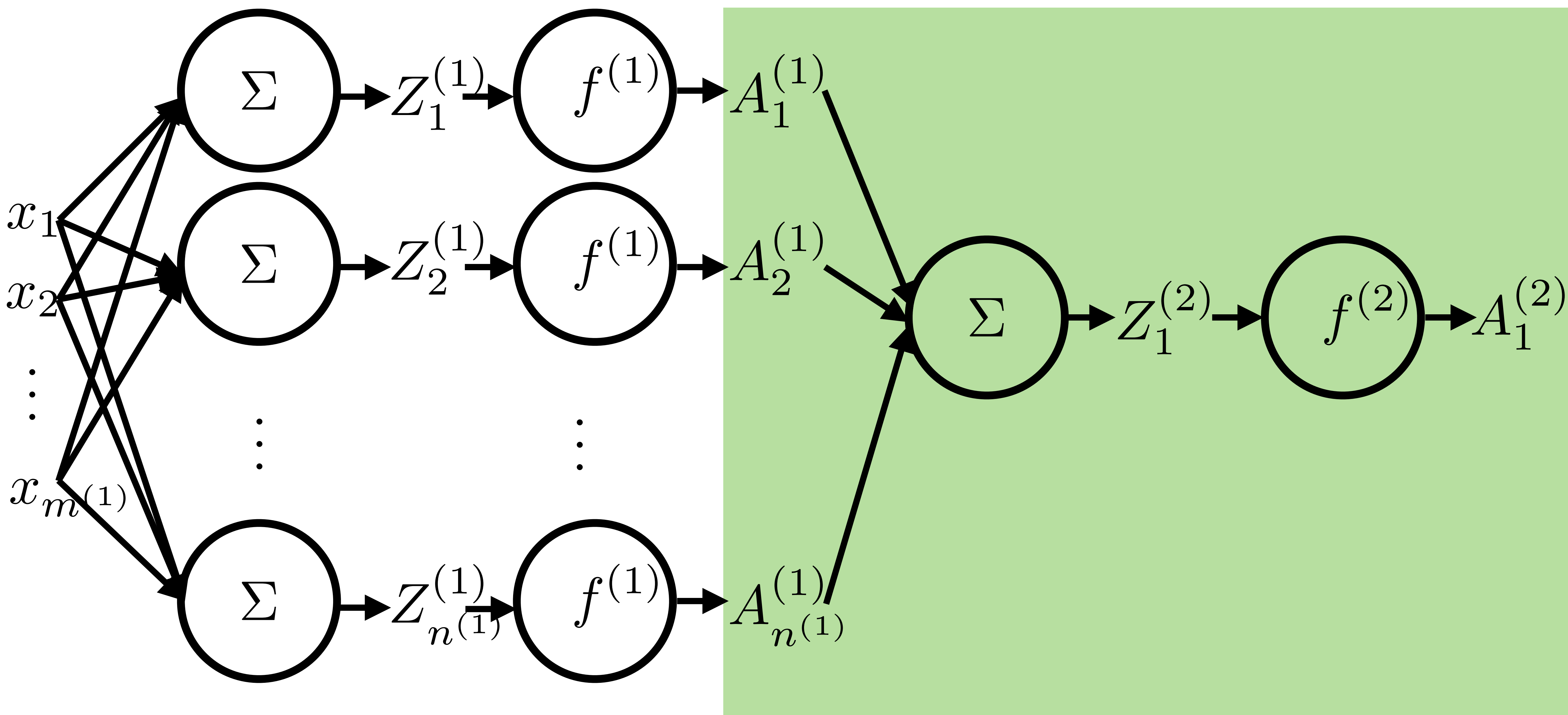
Previous neural nets in this class

- Recall:



Previous neural nets in this class

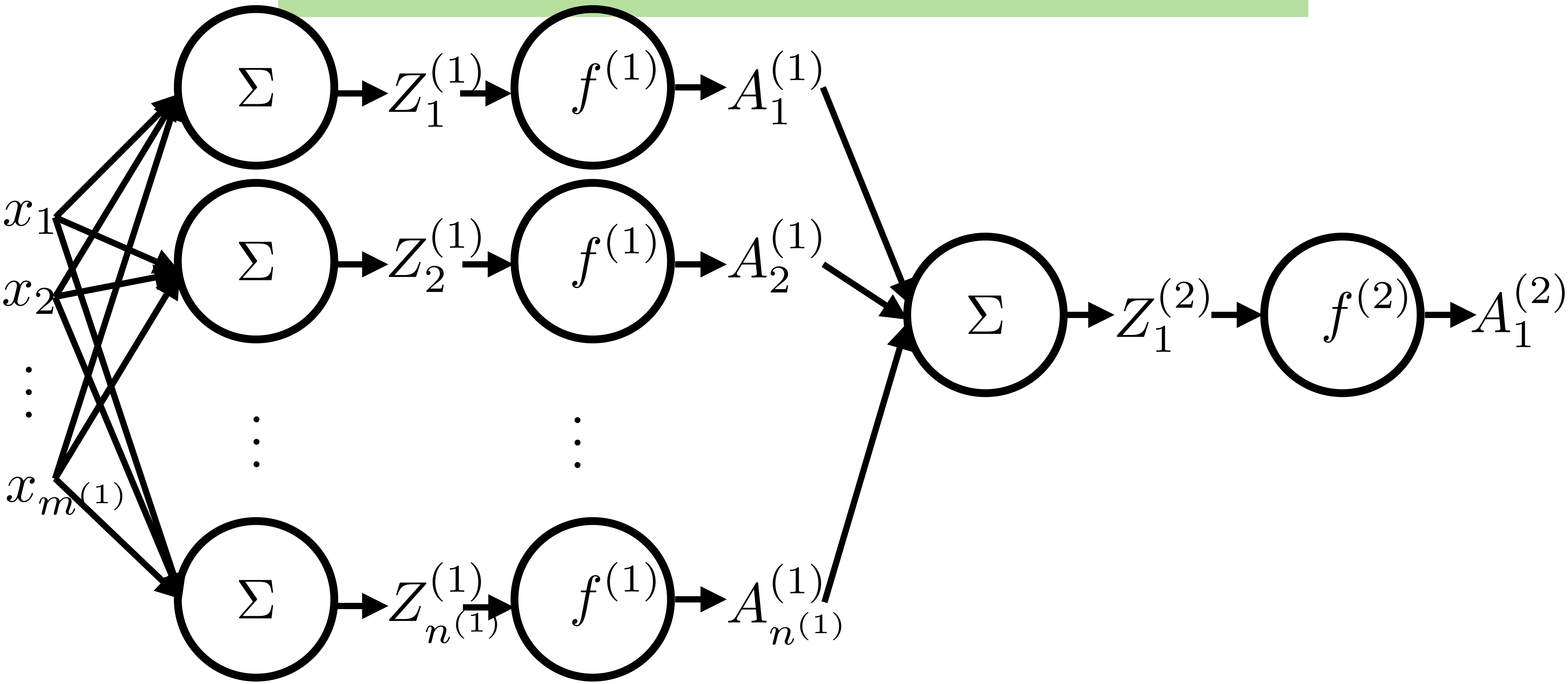
- Recall:



Previous neural nets in this class

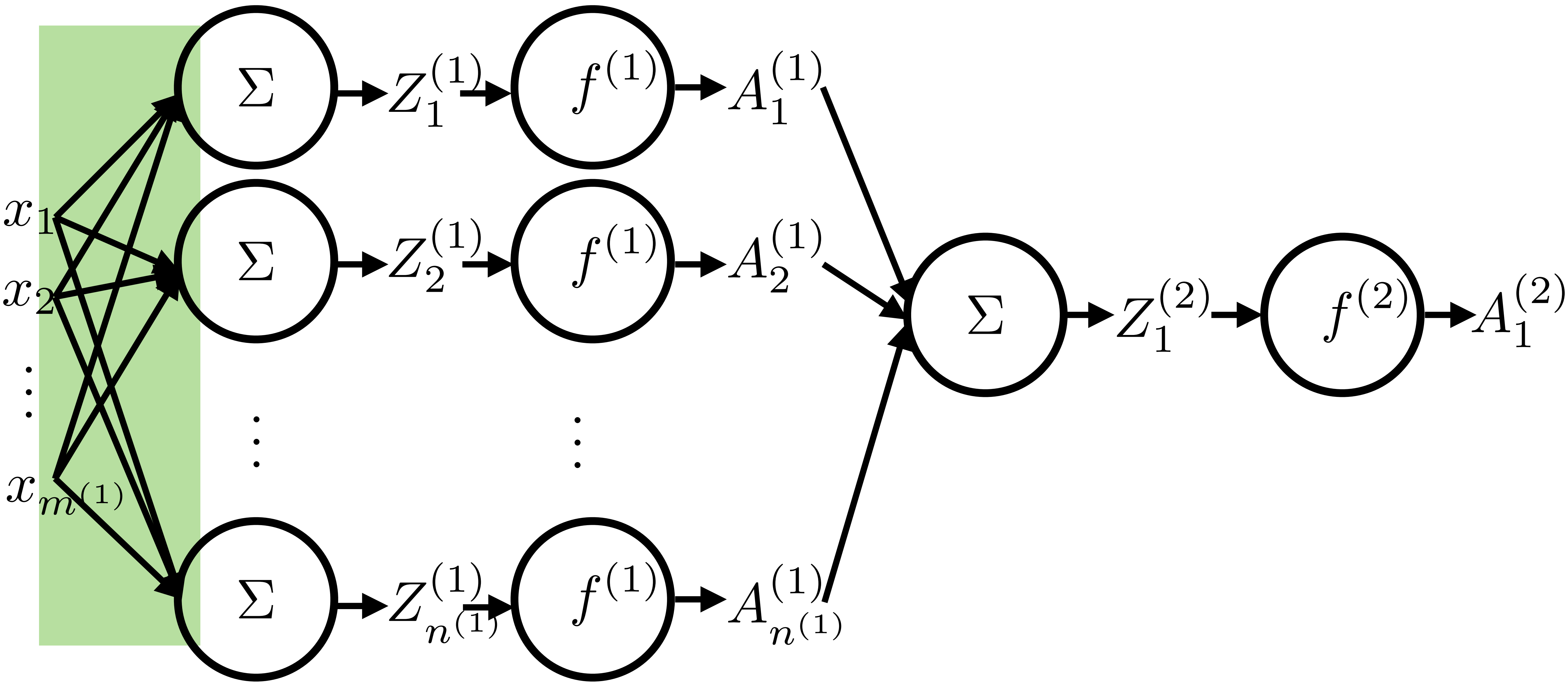
- Recall:

Fully connected layer: every input is connected to every output by a weight



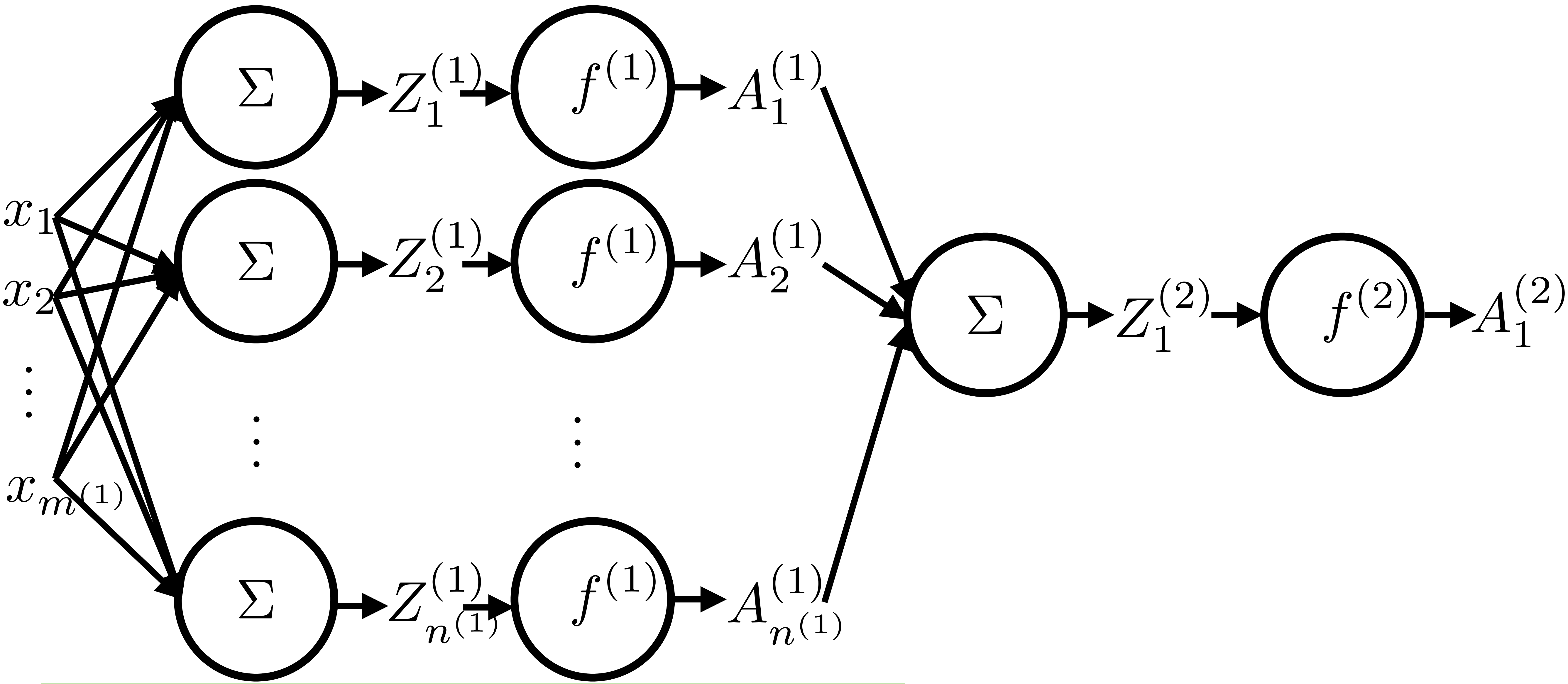
Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight



Previous neural nets in this class

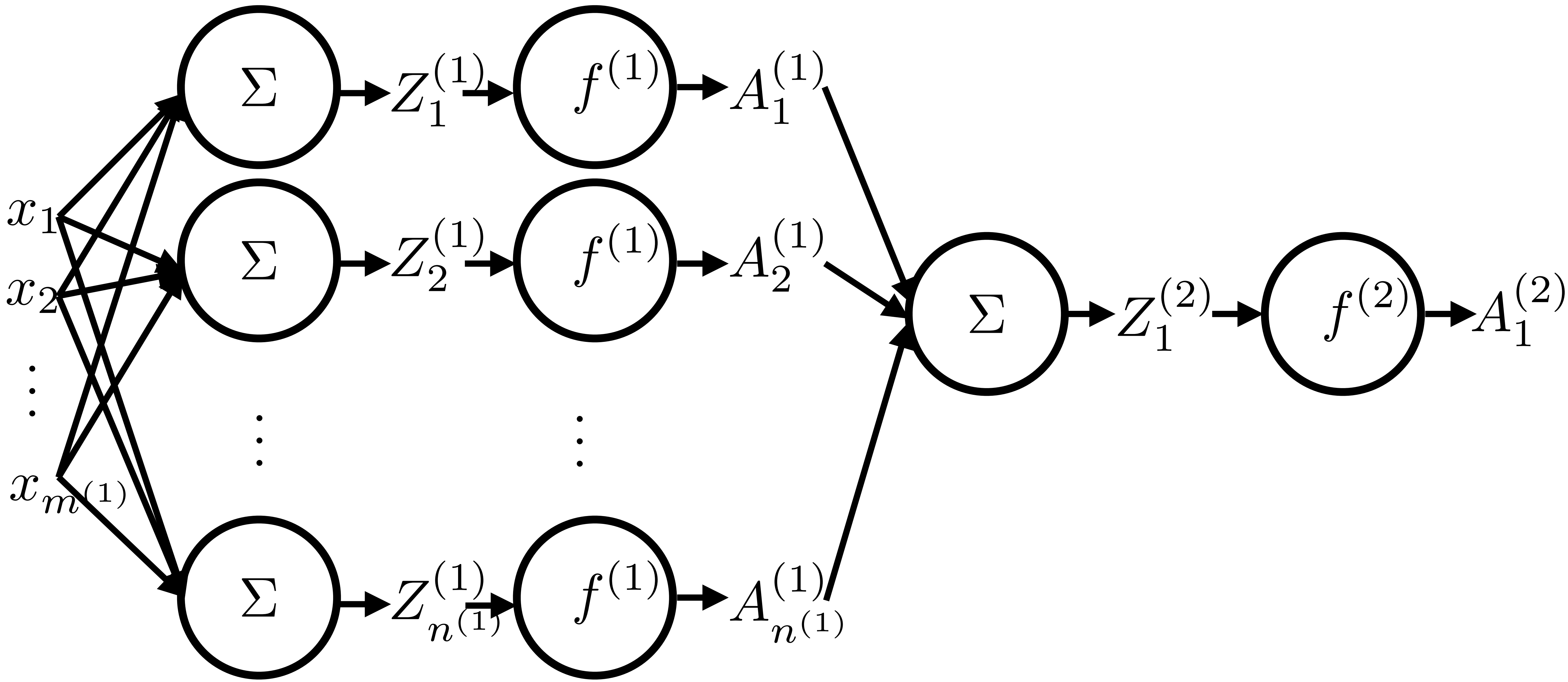
- Recall: *Fully connected* layer: every input is connected to every output by a weight



But we know more about images:

Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight

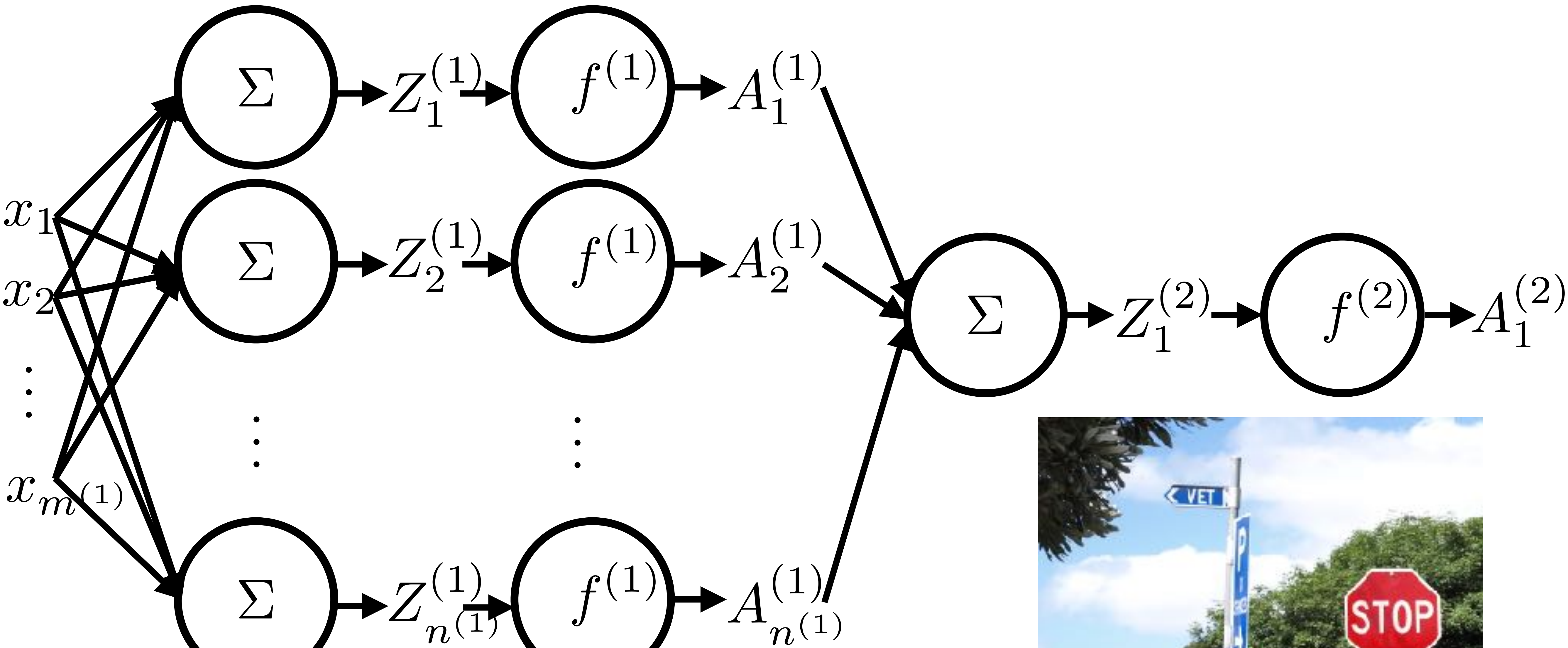


But we know more about images:

- Spatial locality

Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight

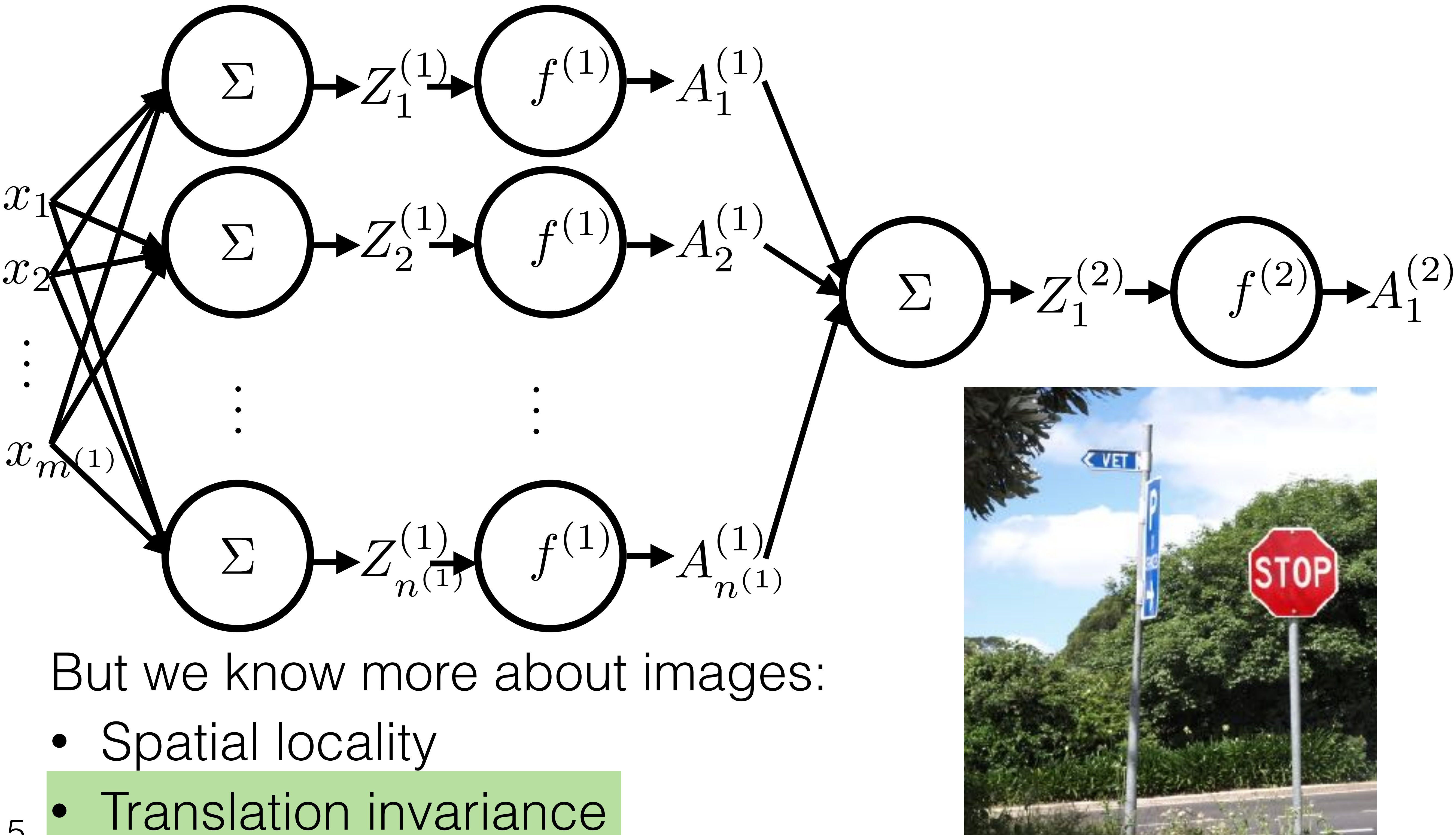


But we know more about images:

- Spatial locality

Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight

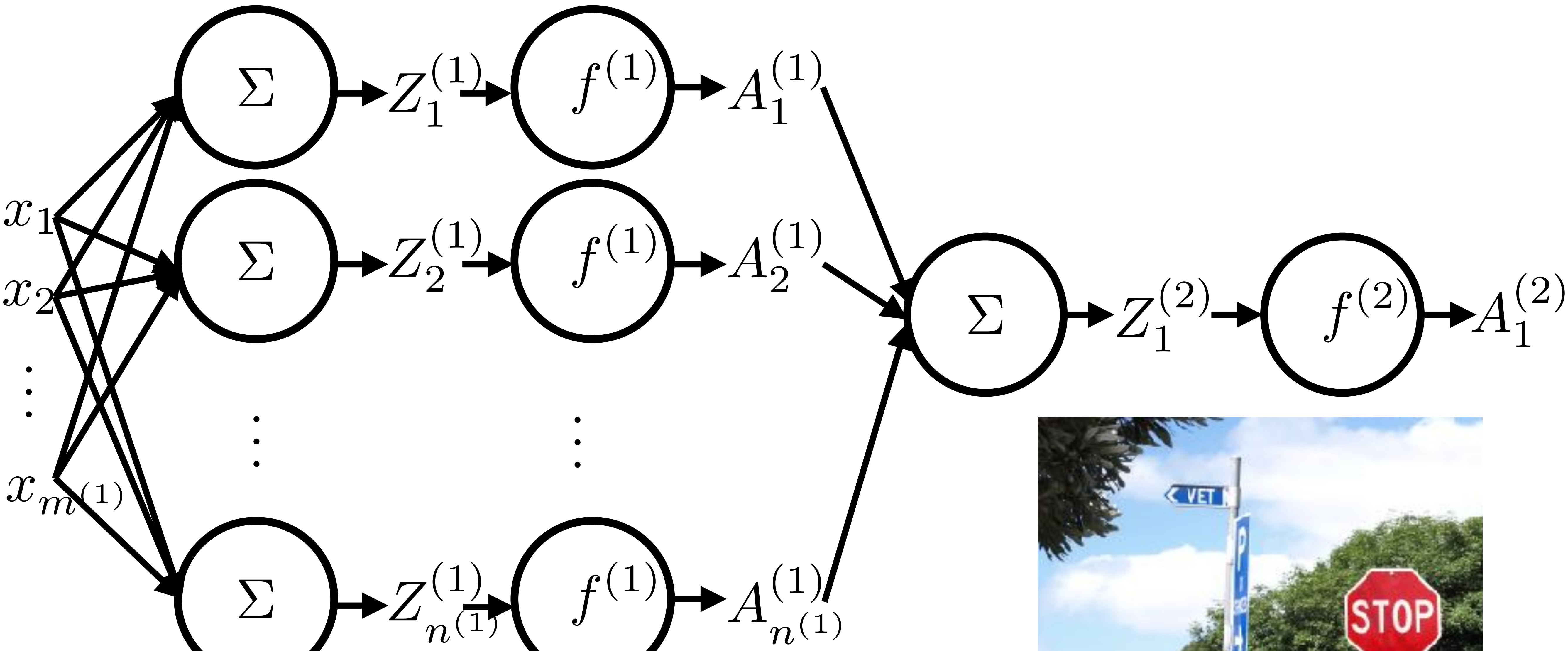


But we know more about images:

- Spatial locality
- Translation invariance

Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight

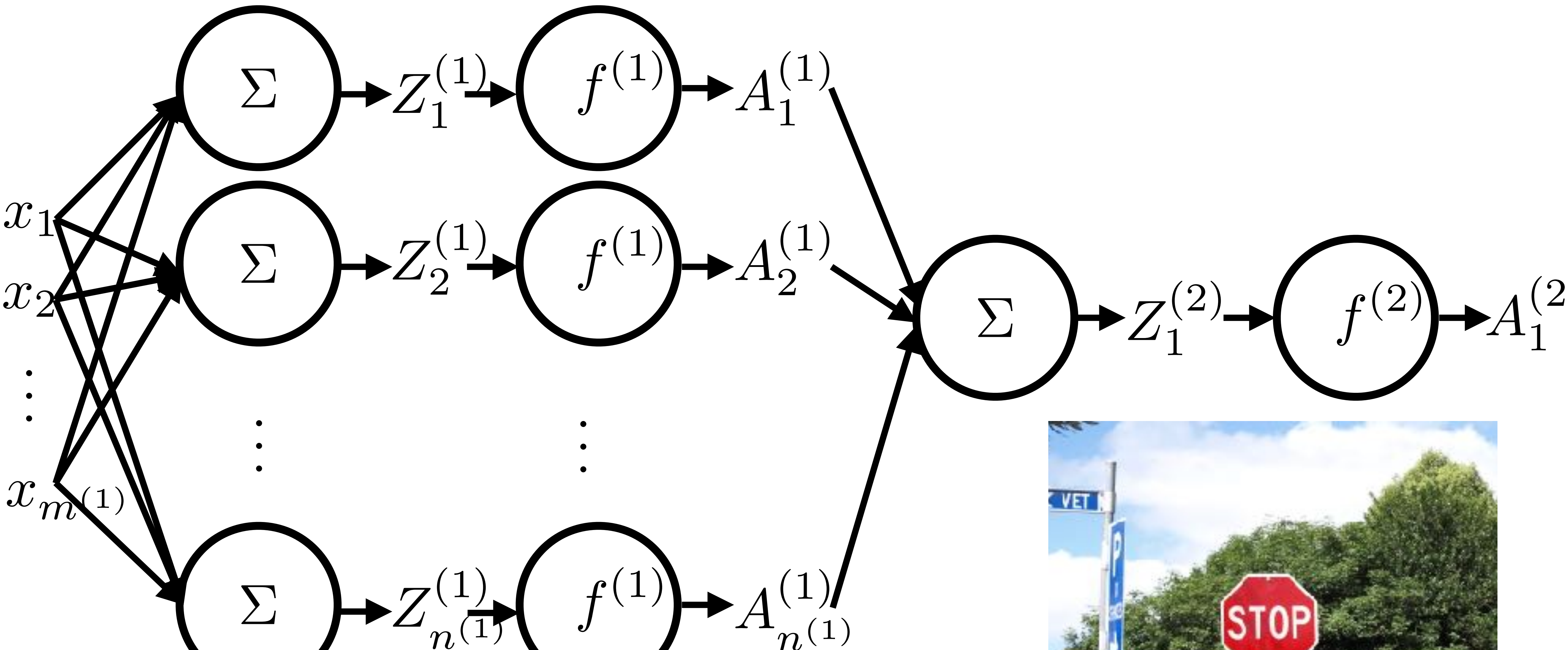


But we know more about images:

- Spatial locality
- Translation invariance

Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight

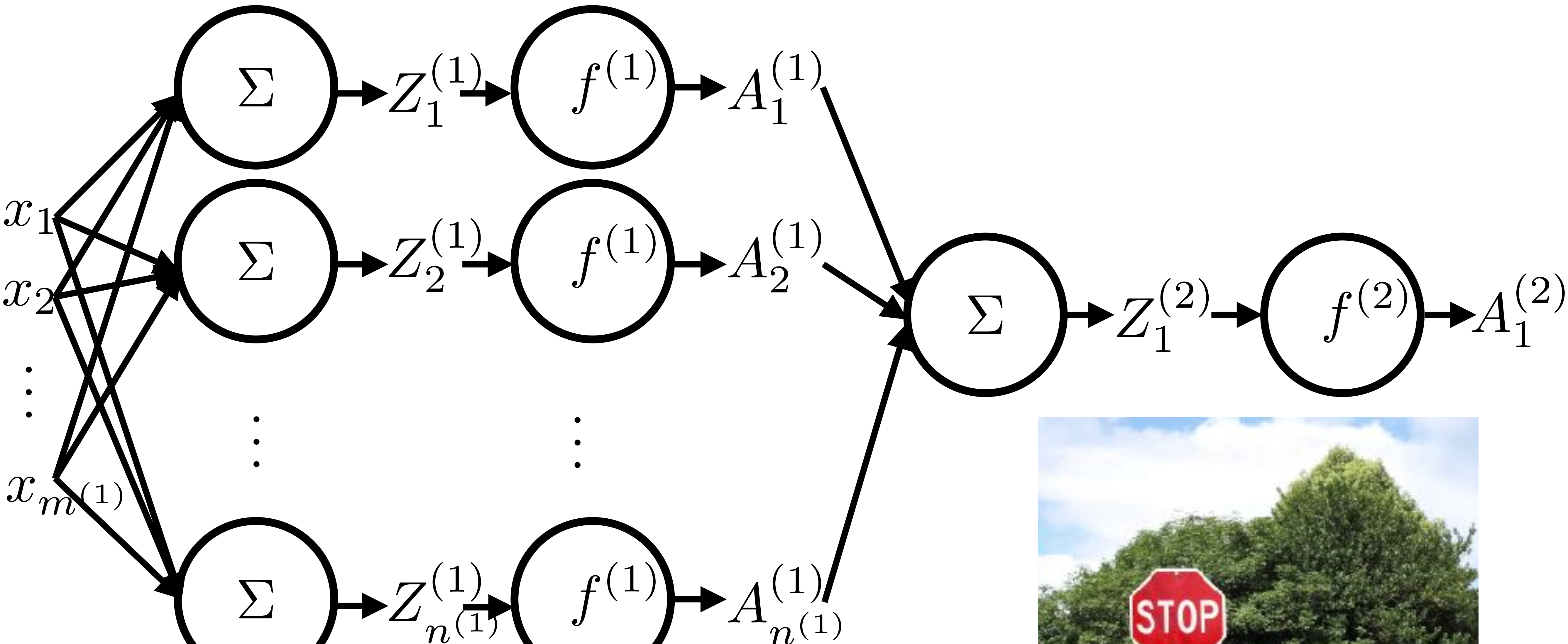


But we know more about images:

- Spatial locality
- Translation invariance

Previous neural nets in this class

- Recall: *Fully connected* layer: every input is connected to every output by a weight



But we know more about images:

- Spatial locality
- Translation invariance

Convolutional Layer: 1D example

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

Letter | Published: 07 January 2019

Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network

Awni Y. Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H. Tison, Codie Bourn, Mintu P. Turakhia & Andrew Y. Ng

[https://www.nature.com/articles/s41591-018-0268-3]

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

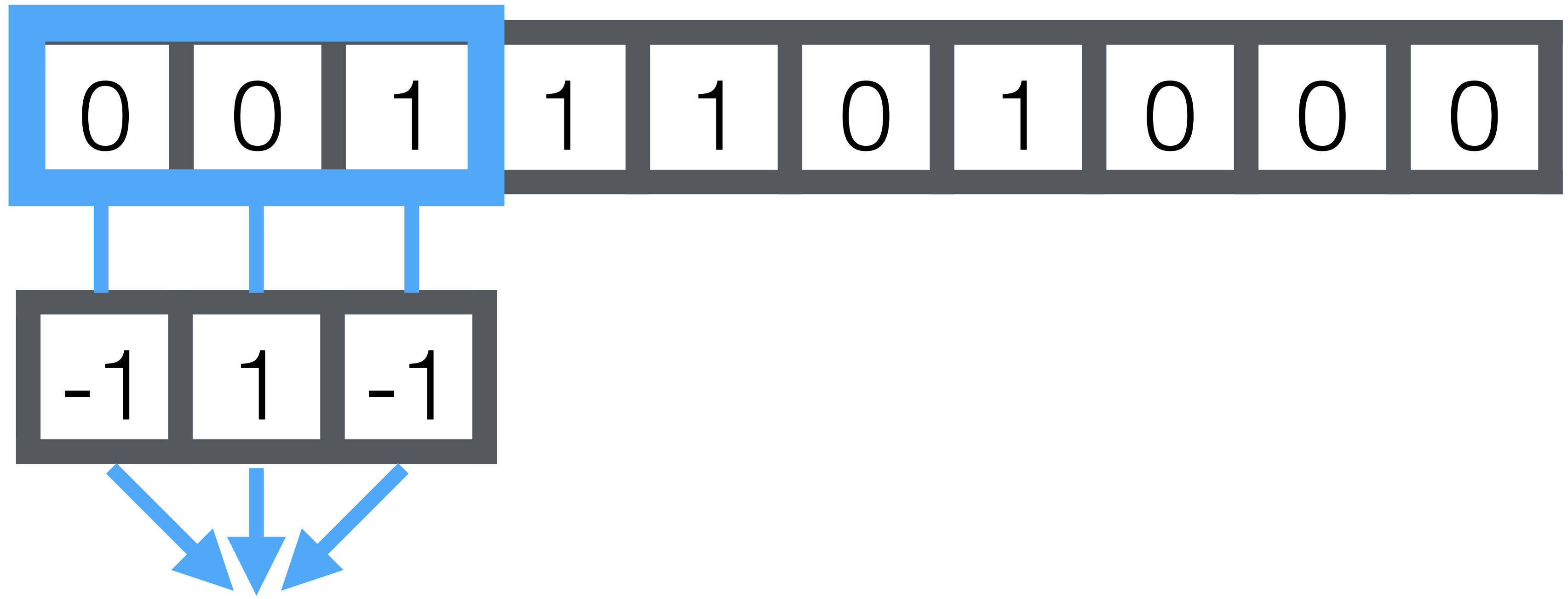
-1	1	-1

After convolution*:

Convolutional Layer: 1D example

A 1D image:

A filter:



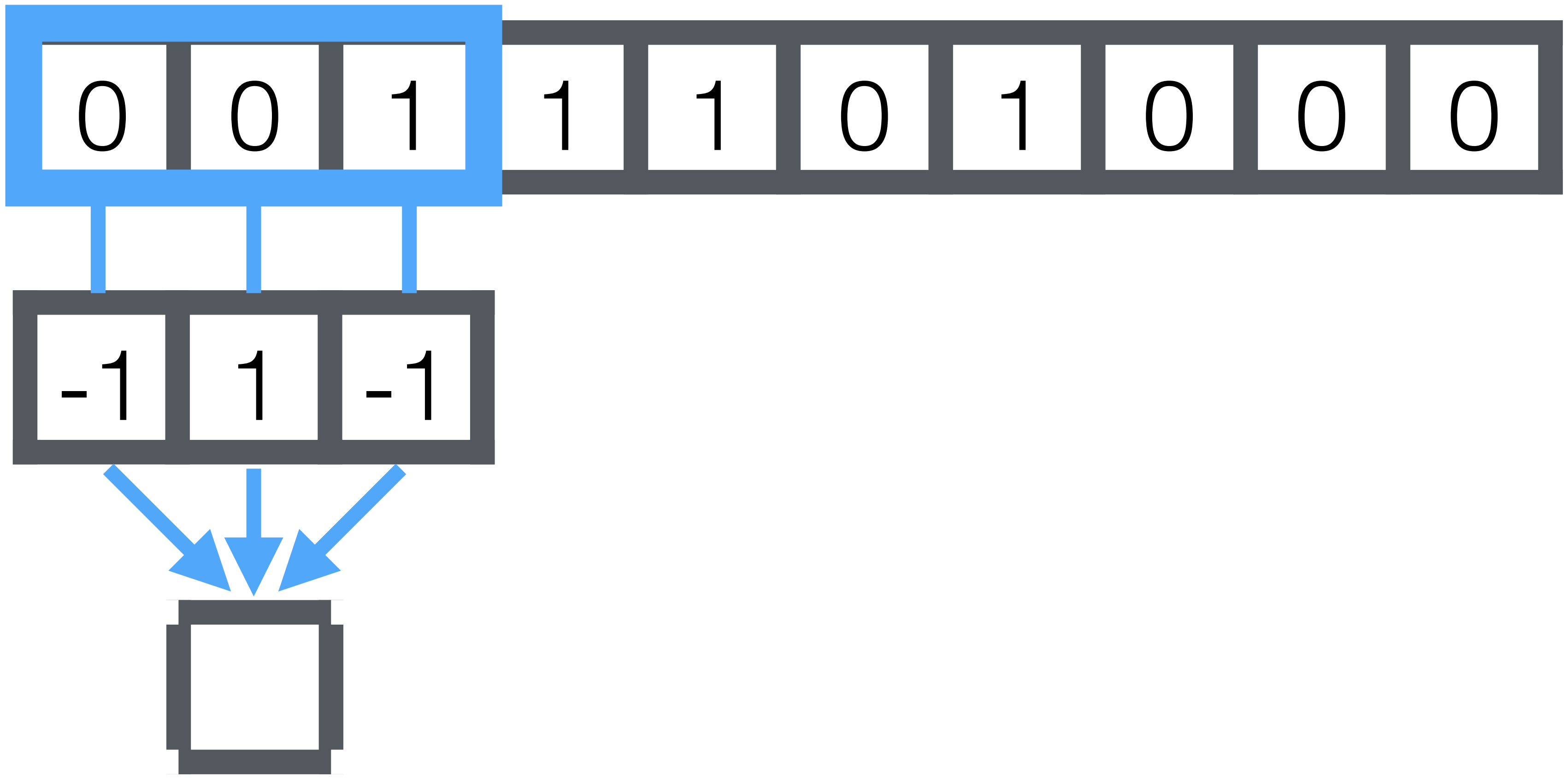
After convolution*:

Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

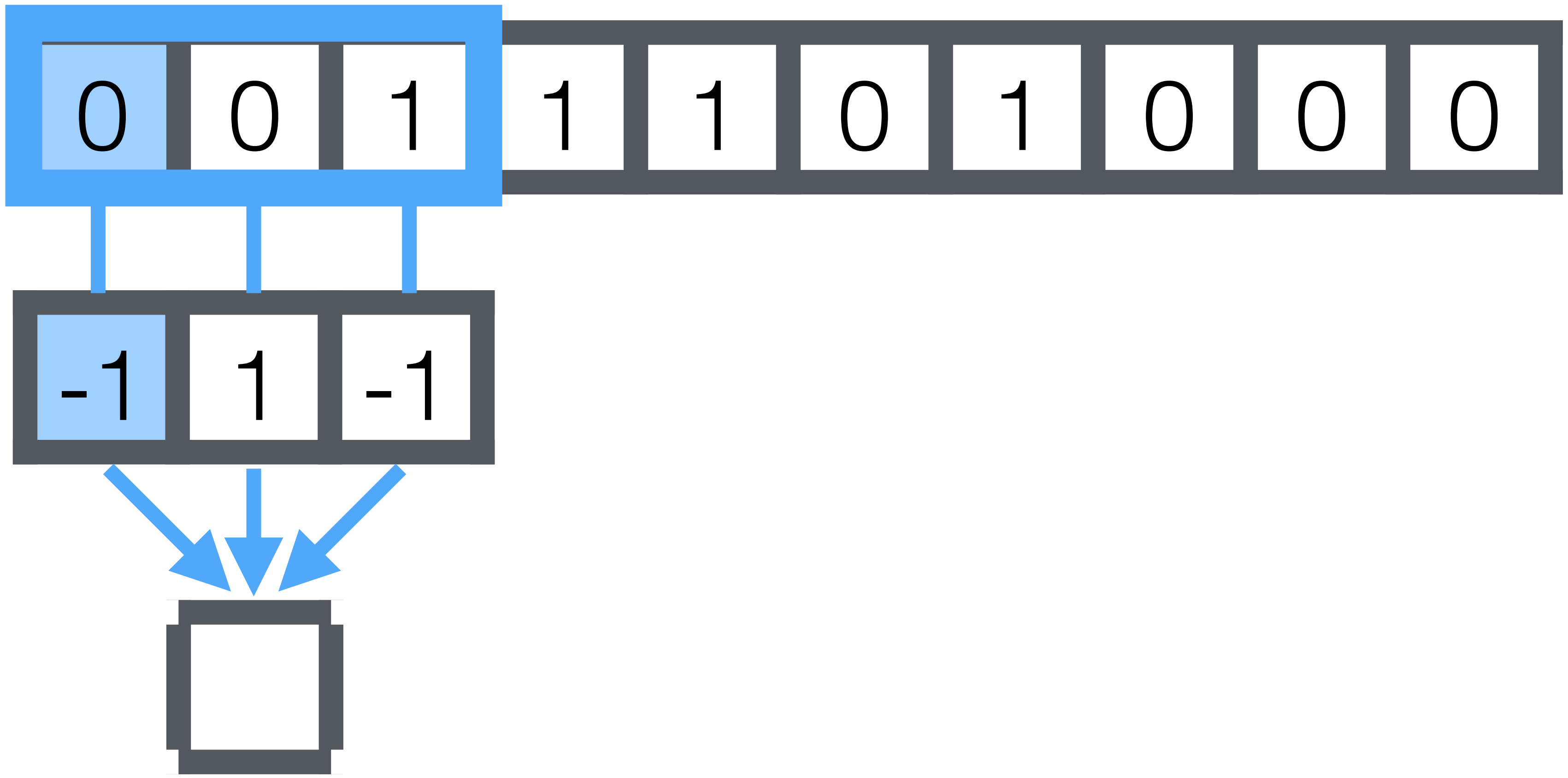


Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

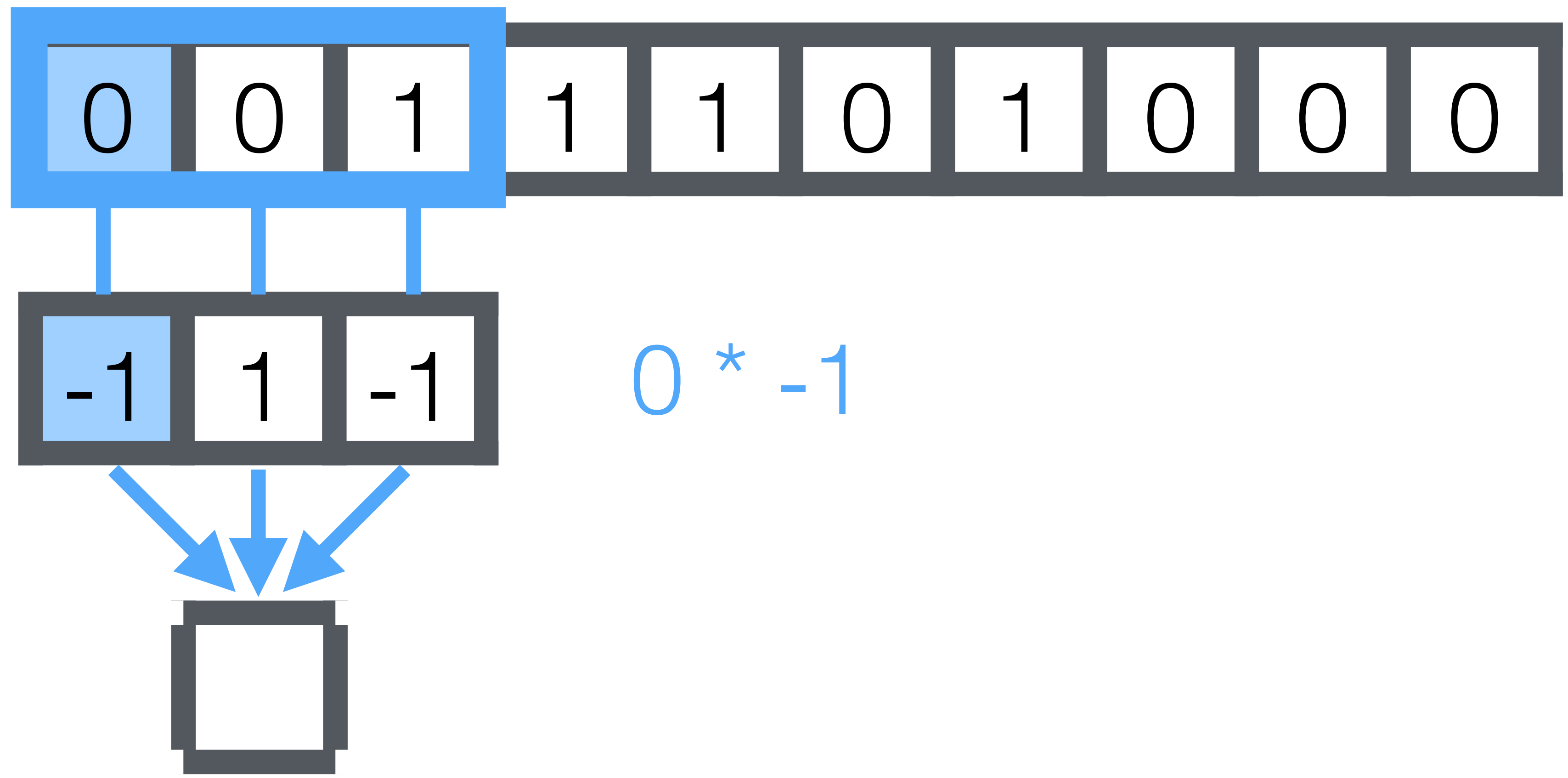


Convolutional Layer: 1D example

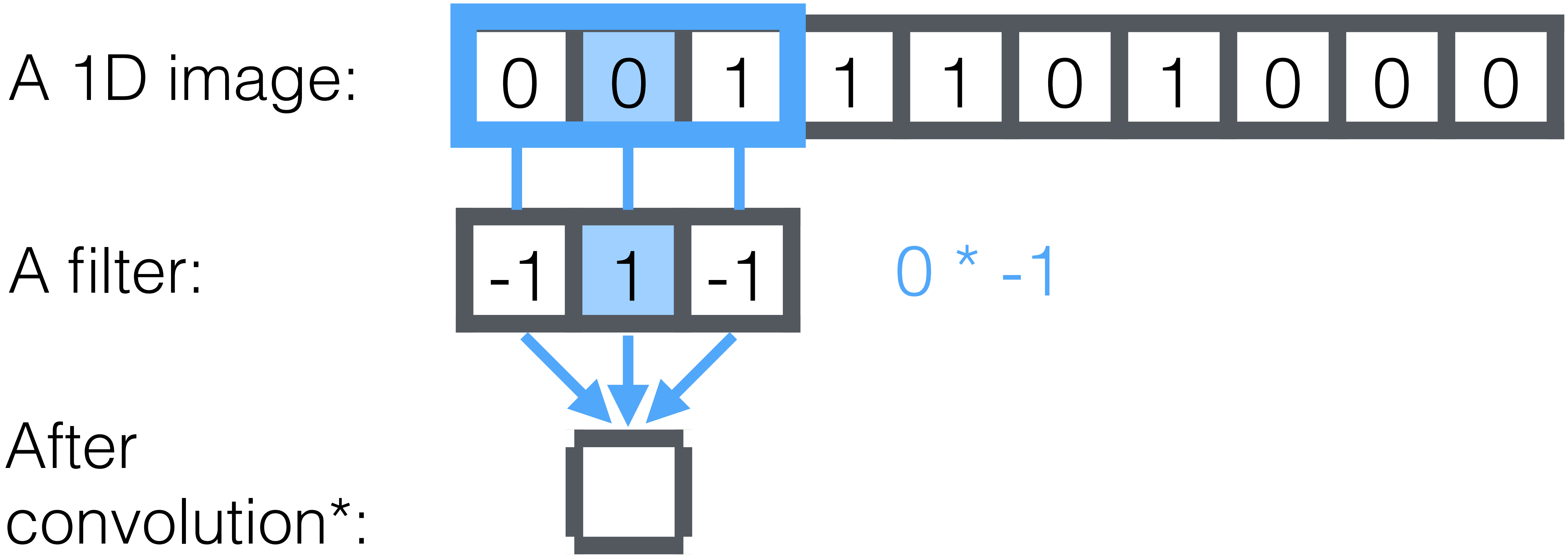
A 1D image:

A filter:

After convolution*:



Convolutional Layer: 1D example

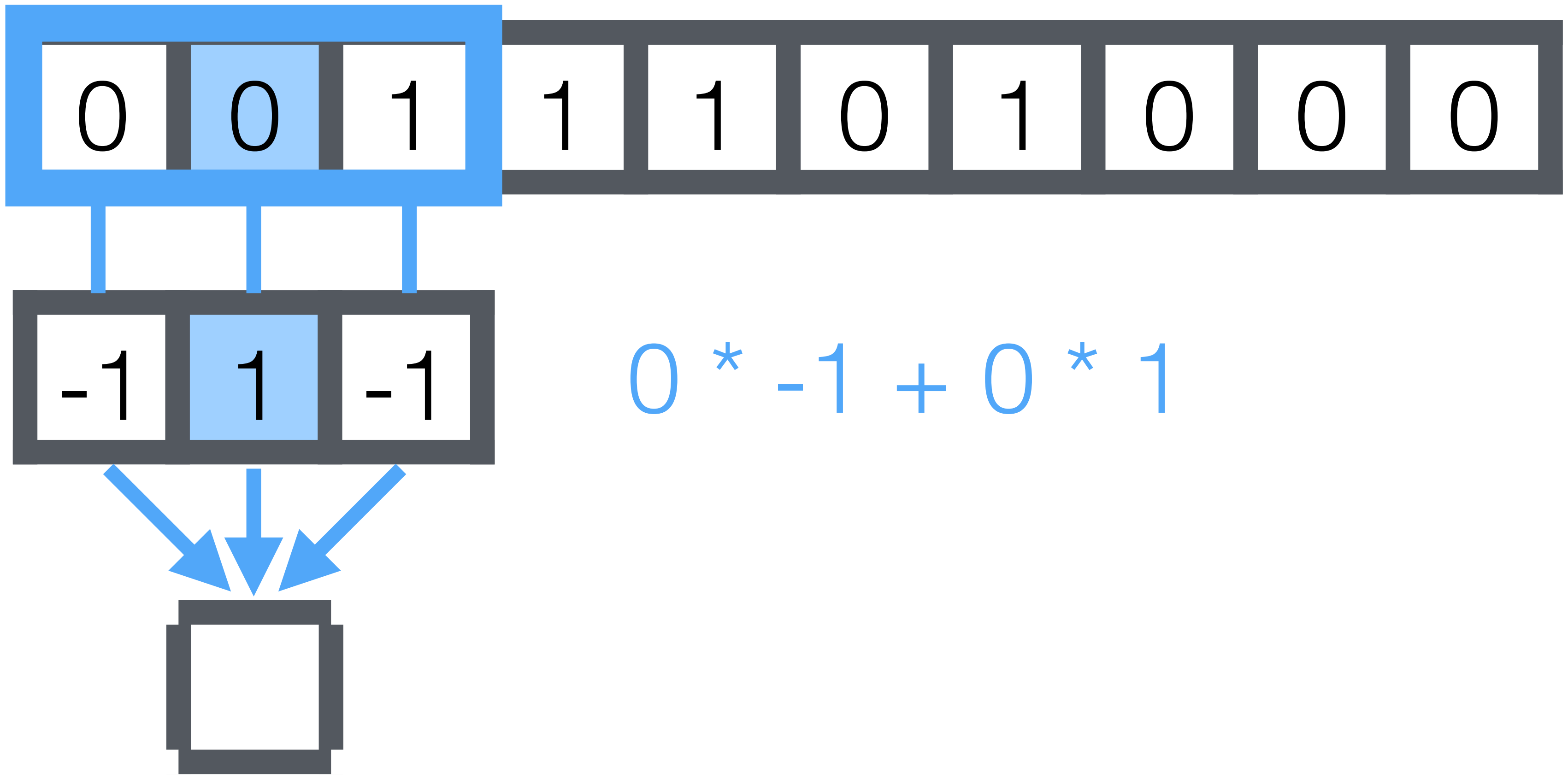


Convolutional Layer: 1D example

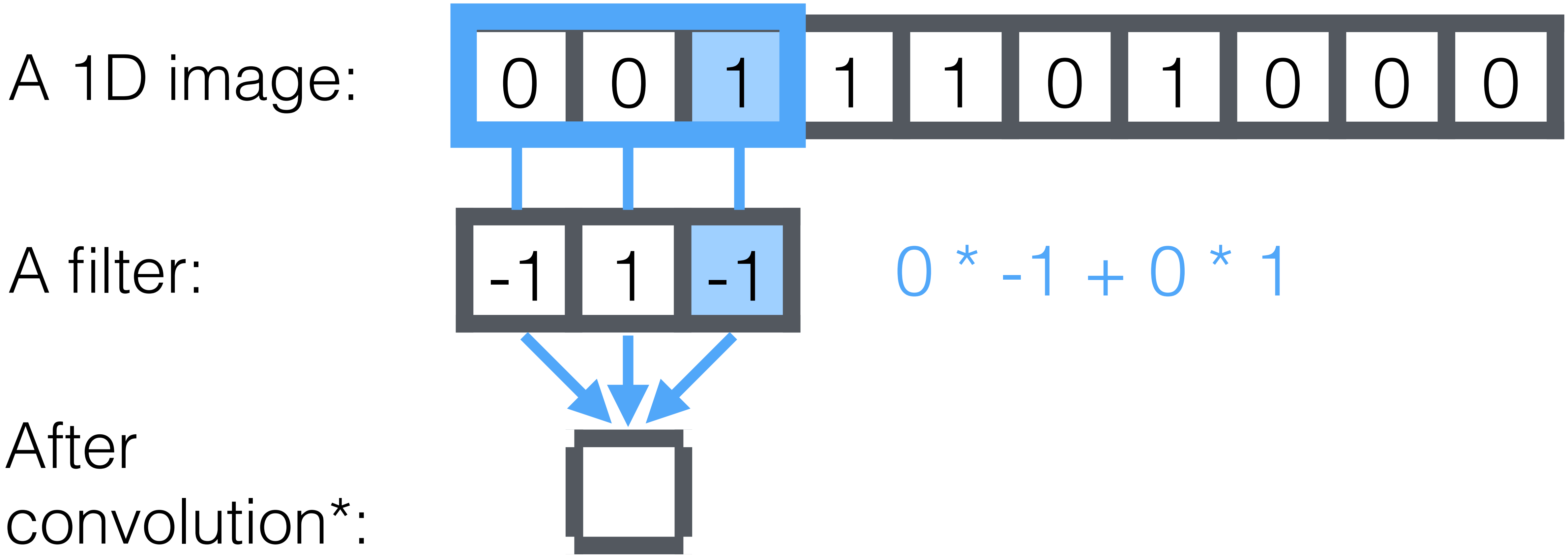
A 1D image:

A filter:

After convolution*:



Convolutional Layer: 1D example

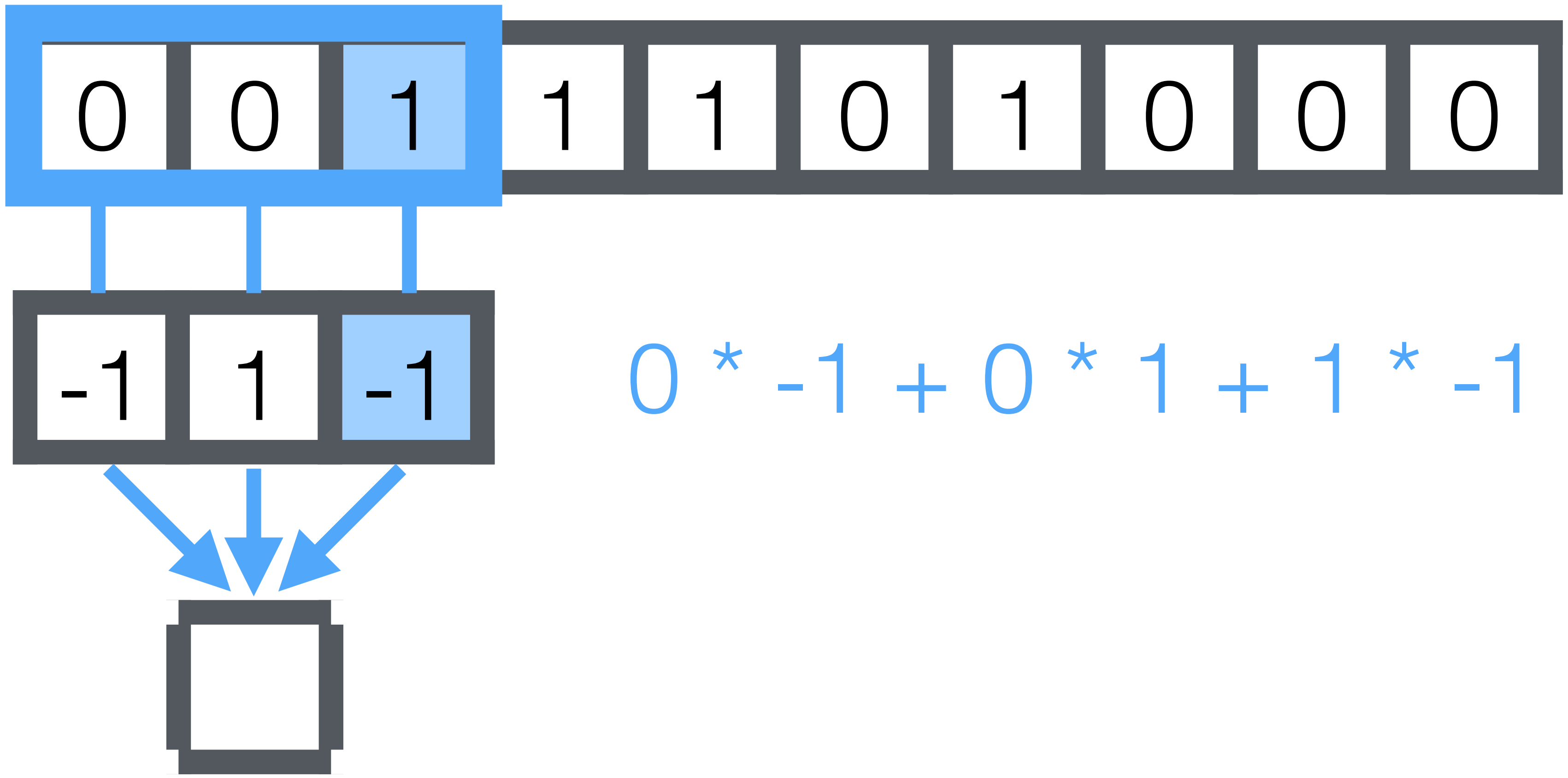


Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

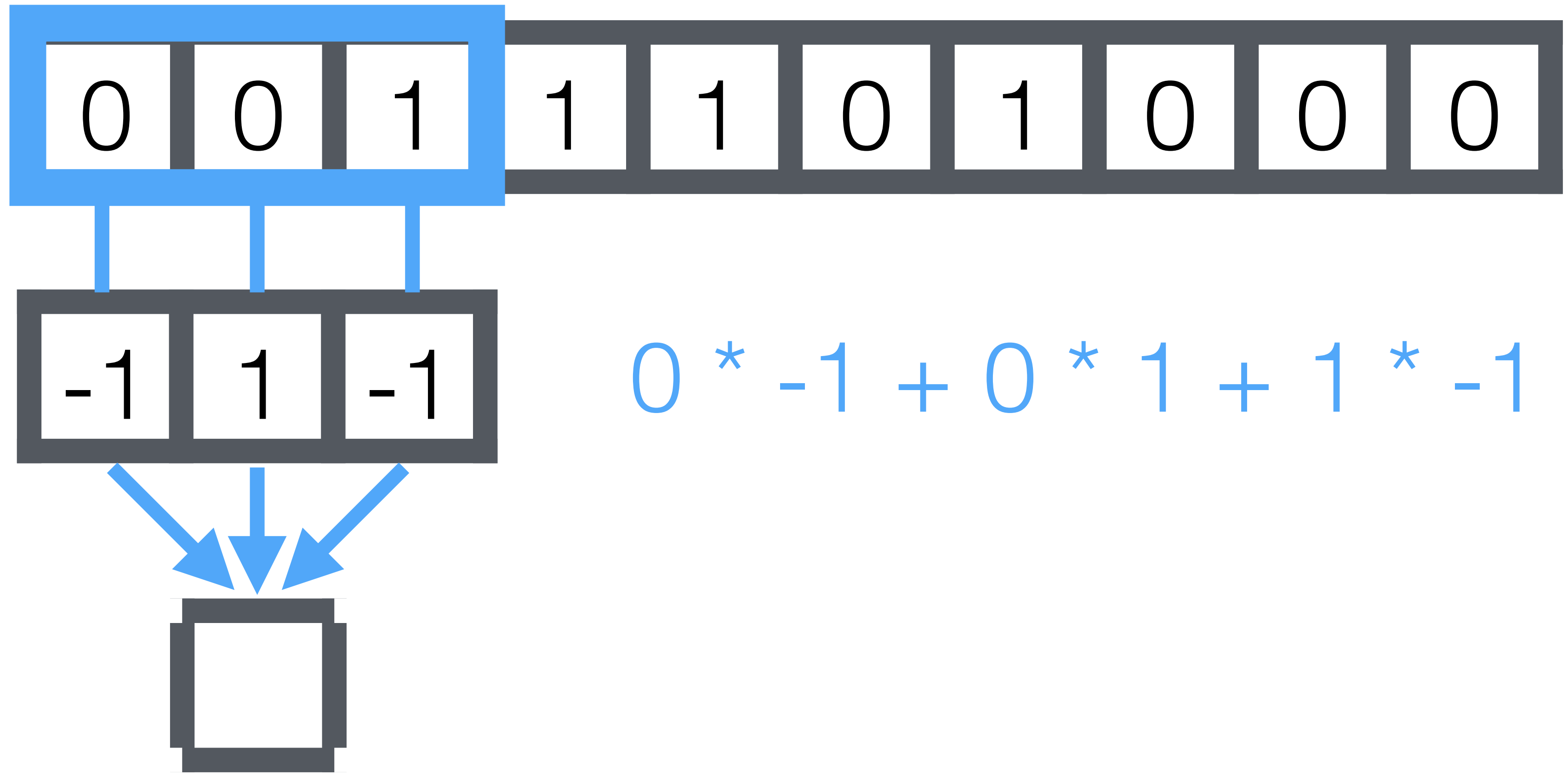


Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

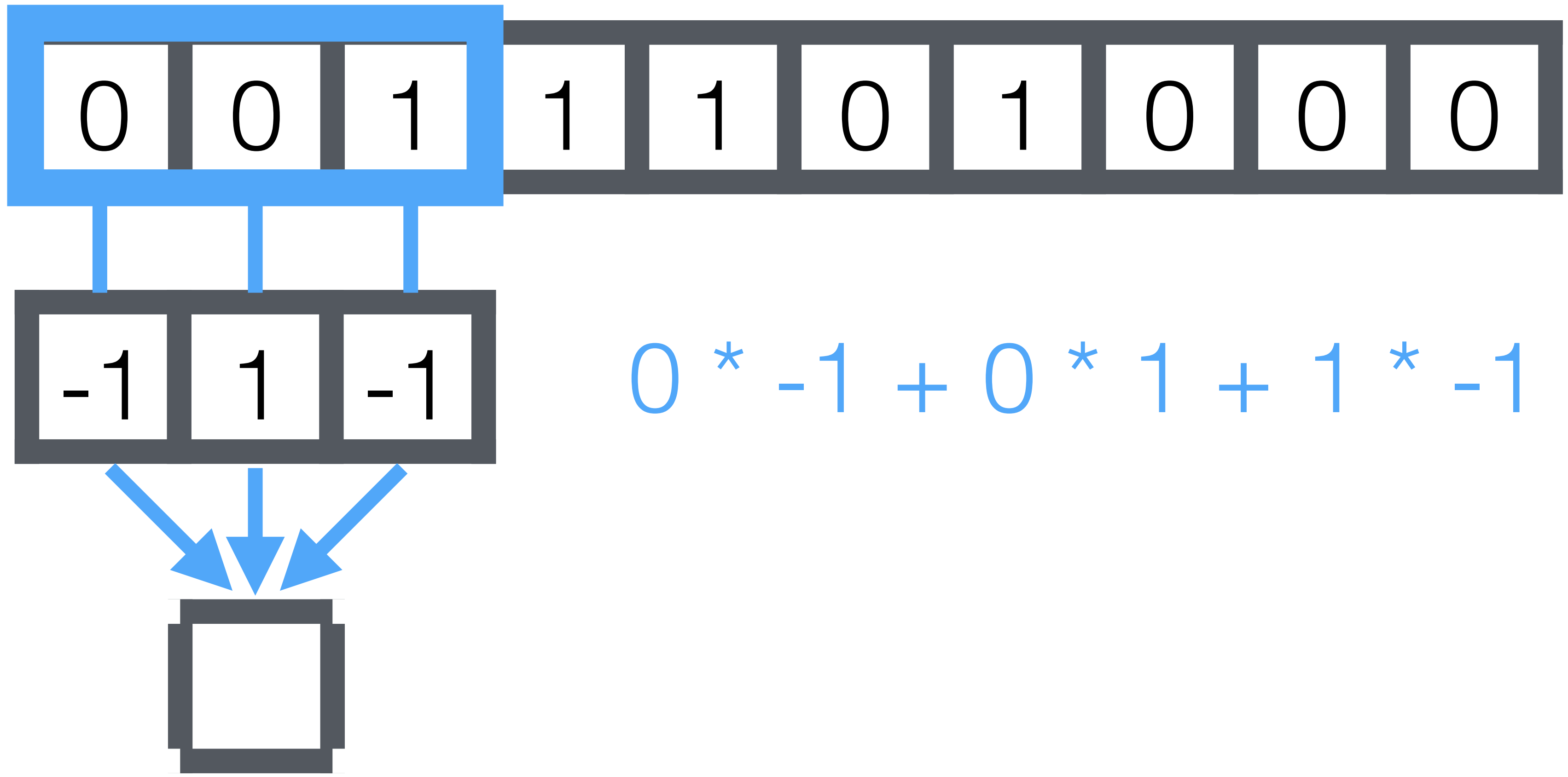


Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:



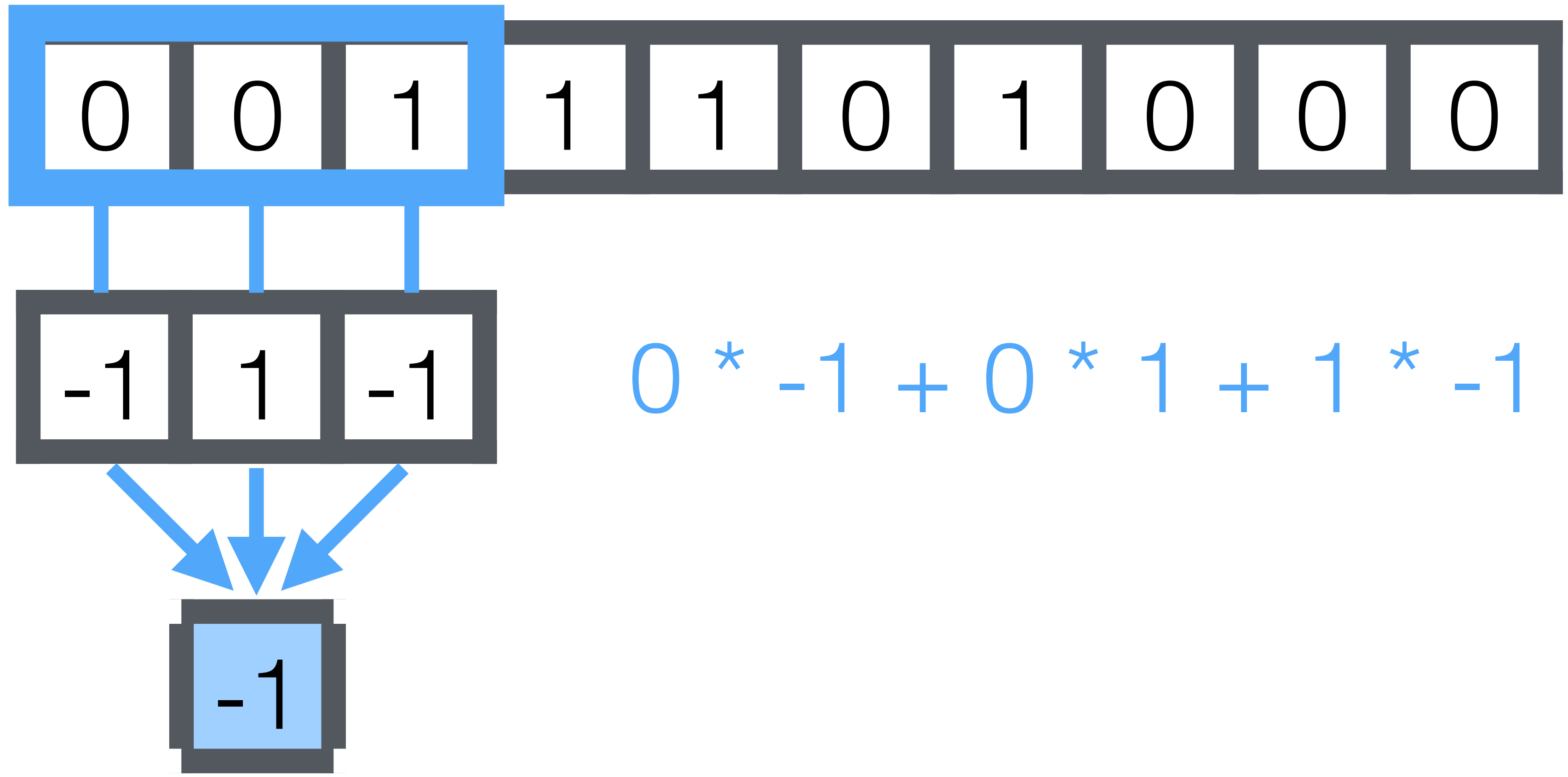
$0 * -1 + 0 * 1 + 1 * -1 = -1$

Convolutional Layer: 1D example

A 1D image:

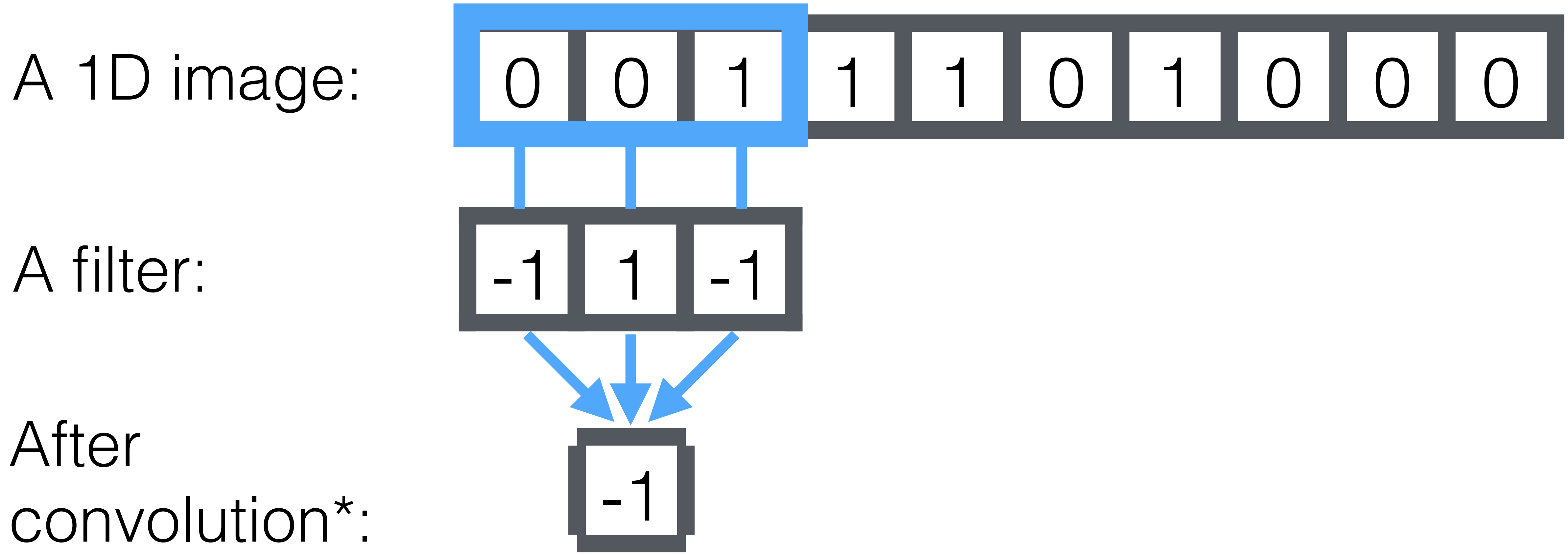
A filter:

After convolution*:

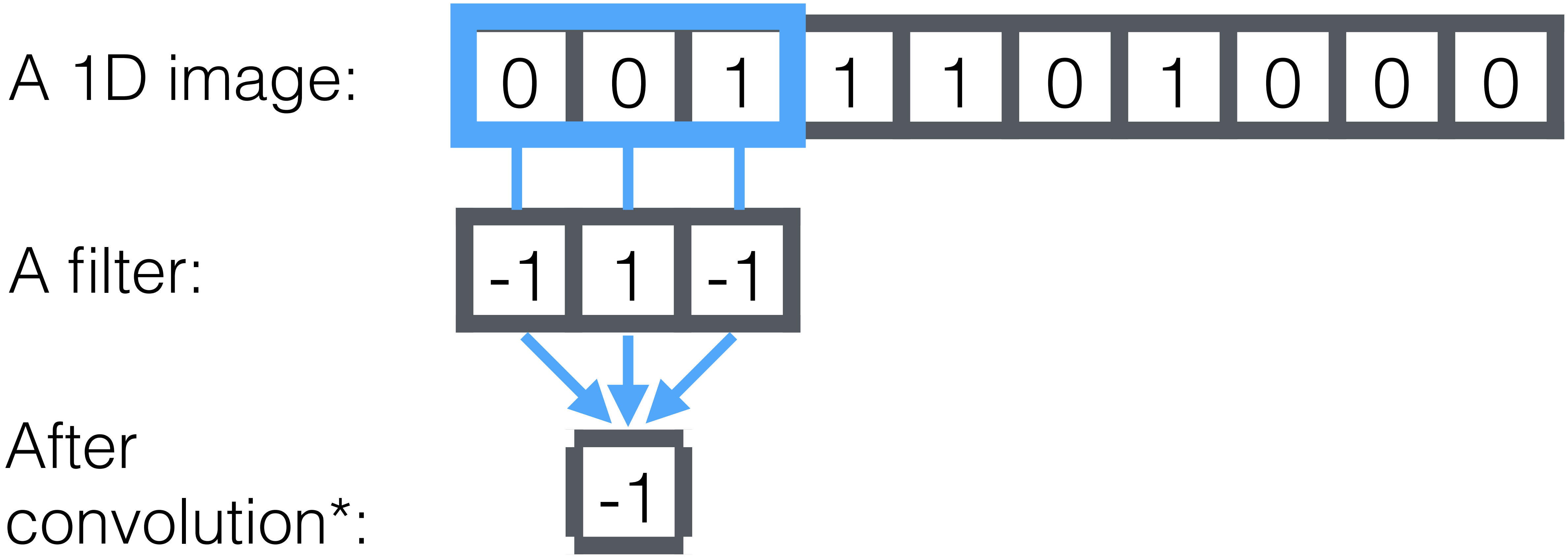


$0 * -1 + 0 * 1 + 1 * -1 = -1$

Convolutional Layer: 1D example



Convolutional Layer: 1D example



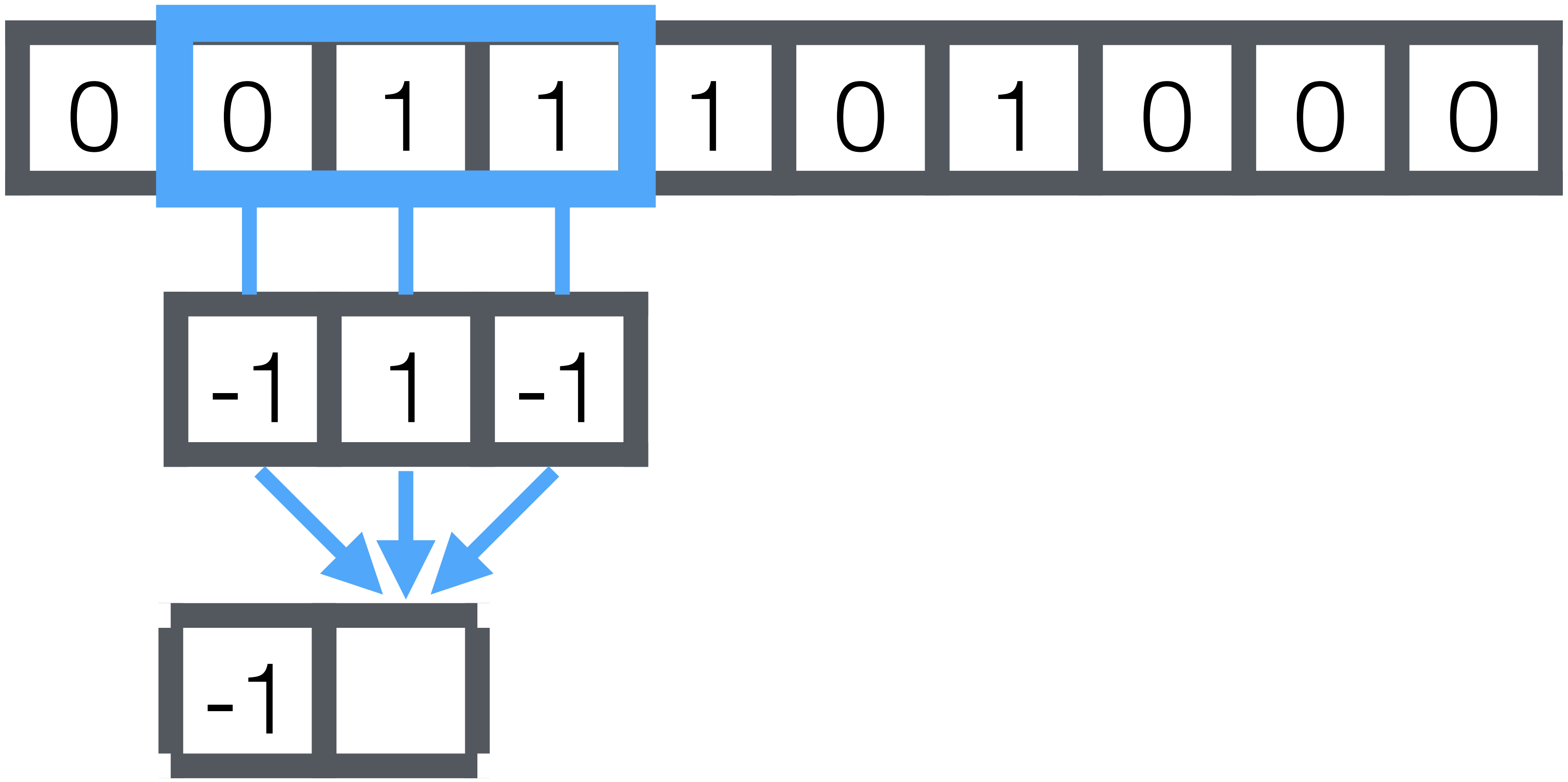
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:



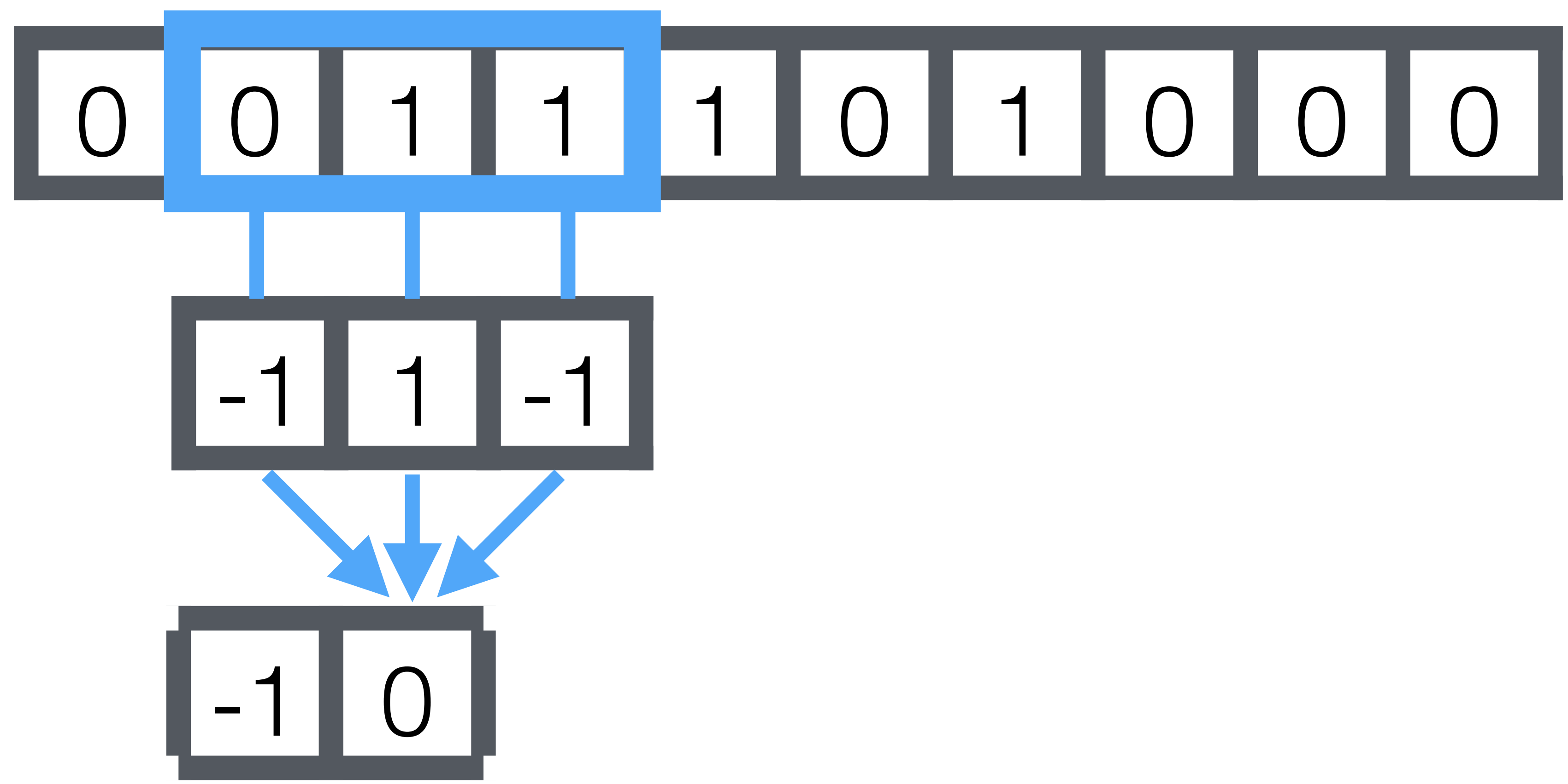
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:



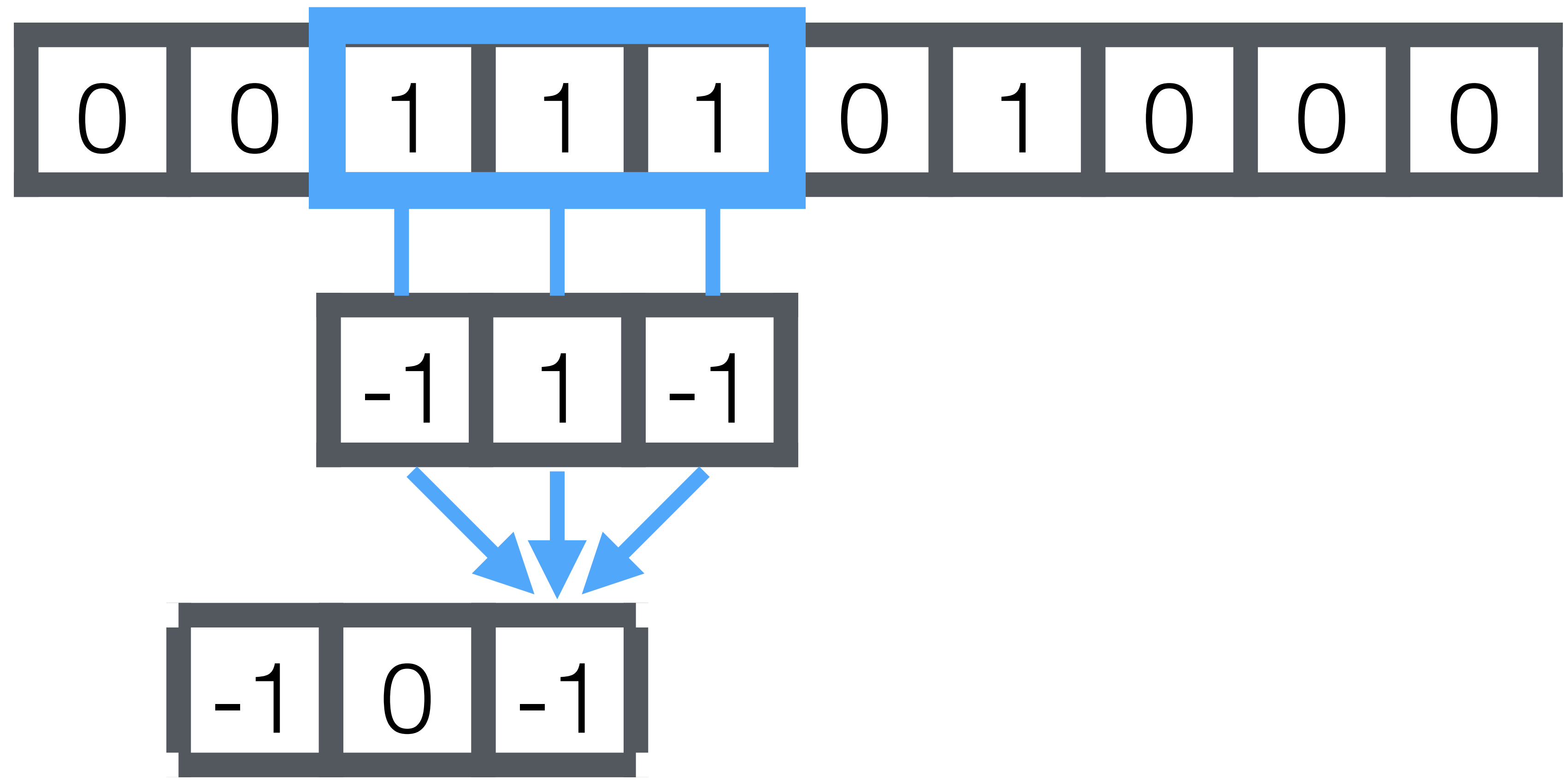
*correlation

Convolutional Layer: 1D example

A 1D image:

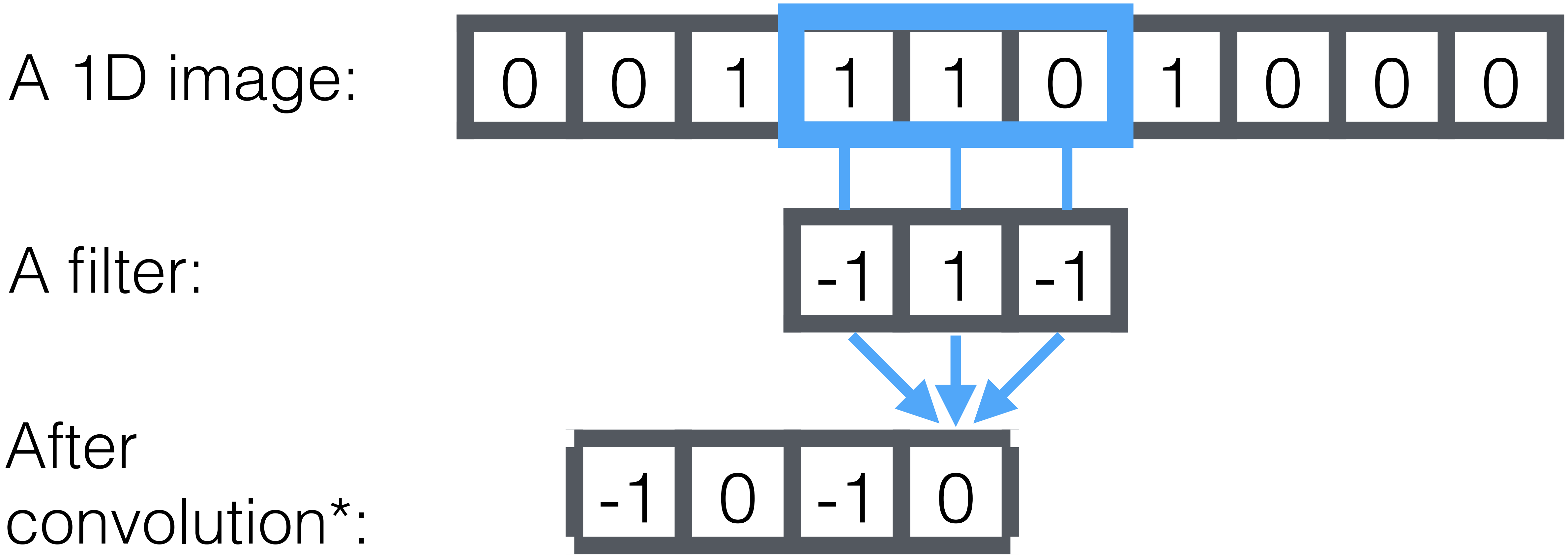
A filter:

After convolution*:



*correlation

Convolutional Layer: 1D example



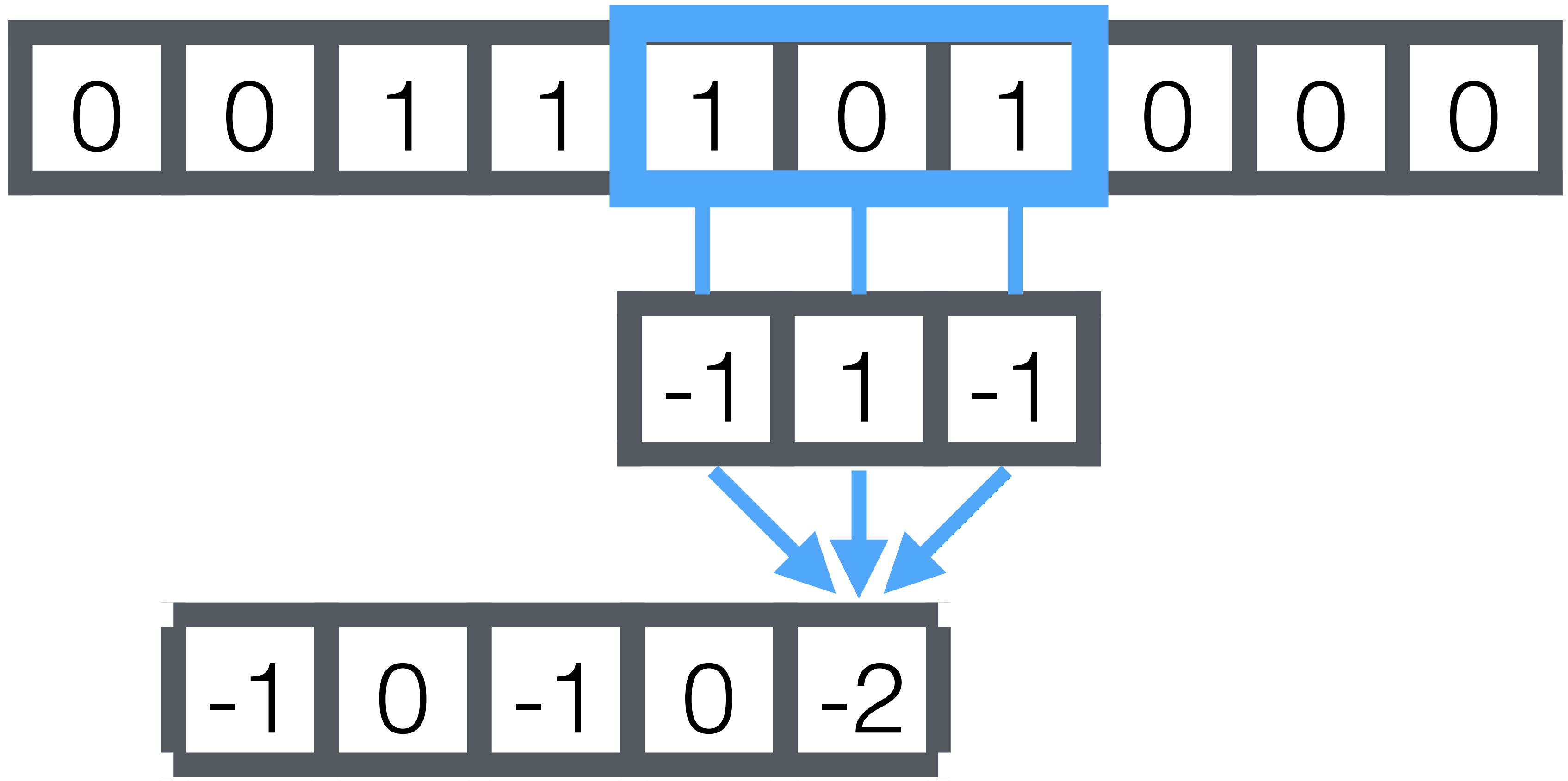
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:



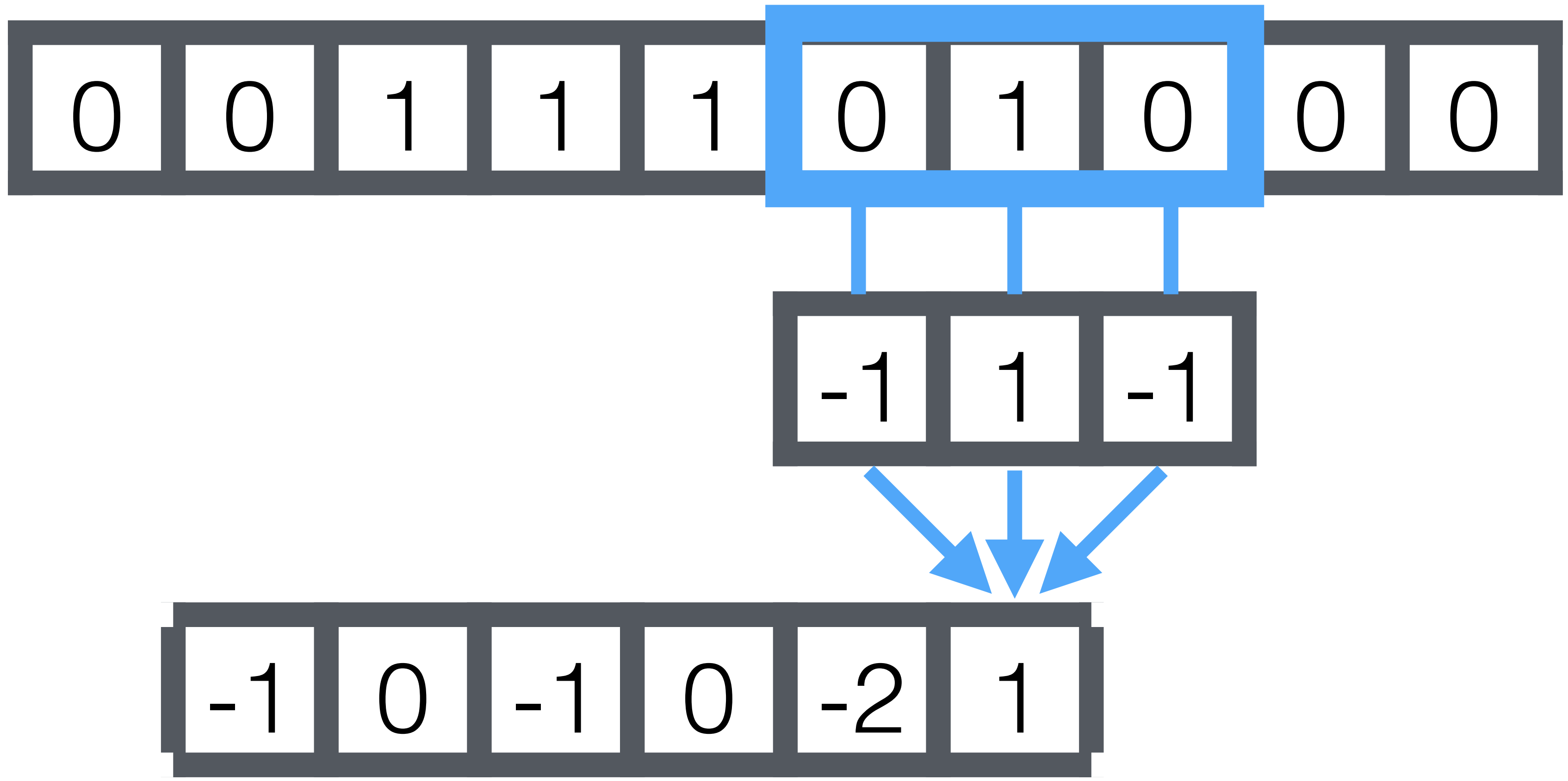
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:



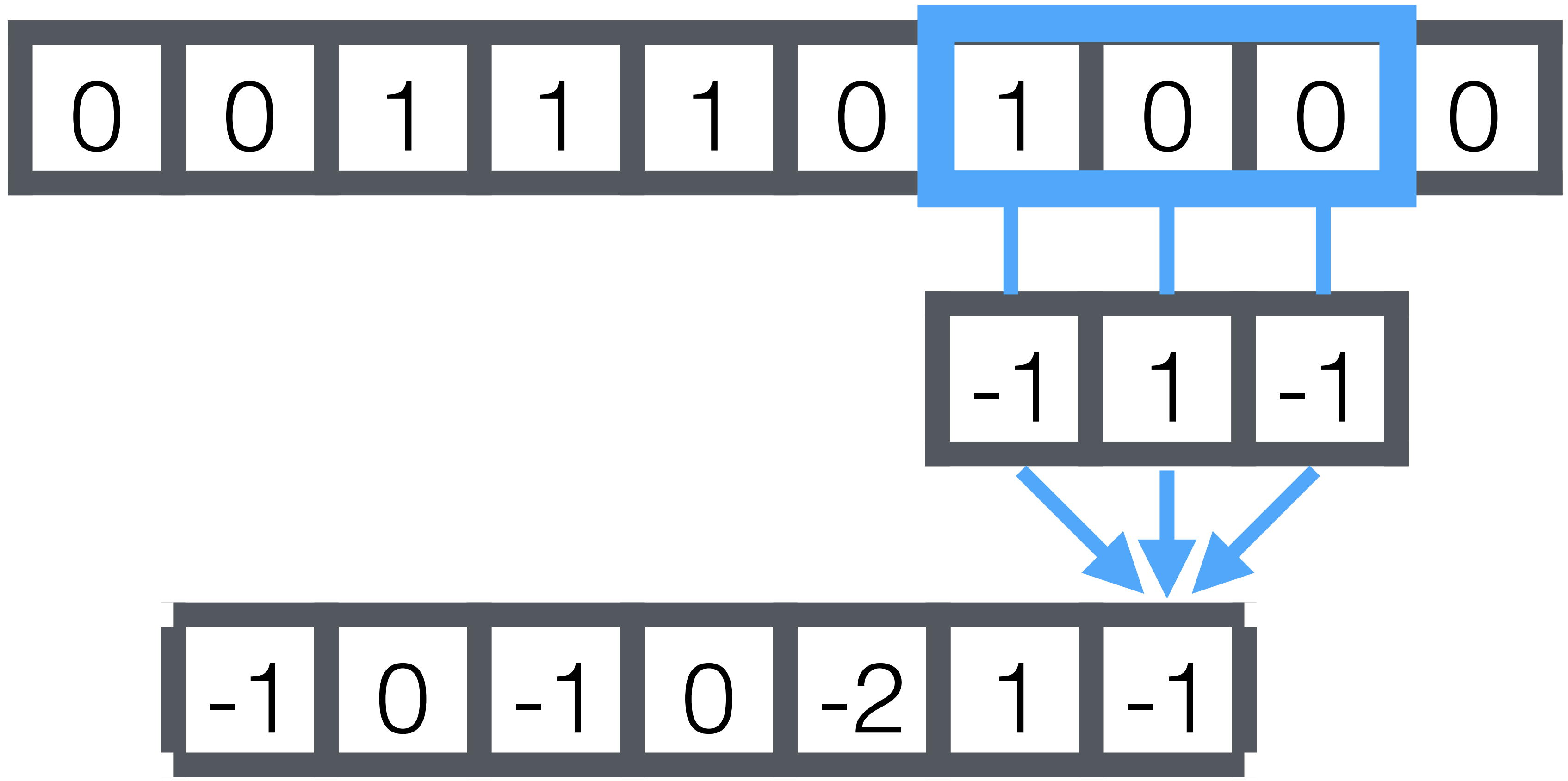
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

After
convolution*:



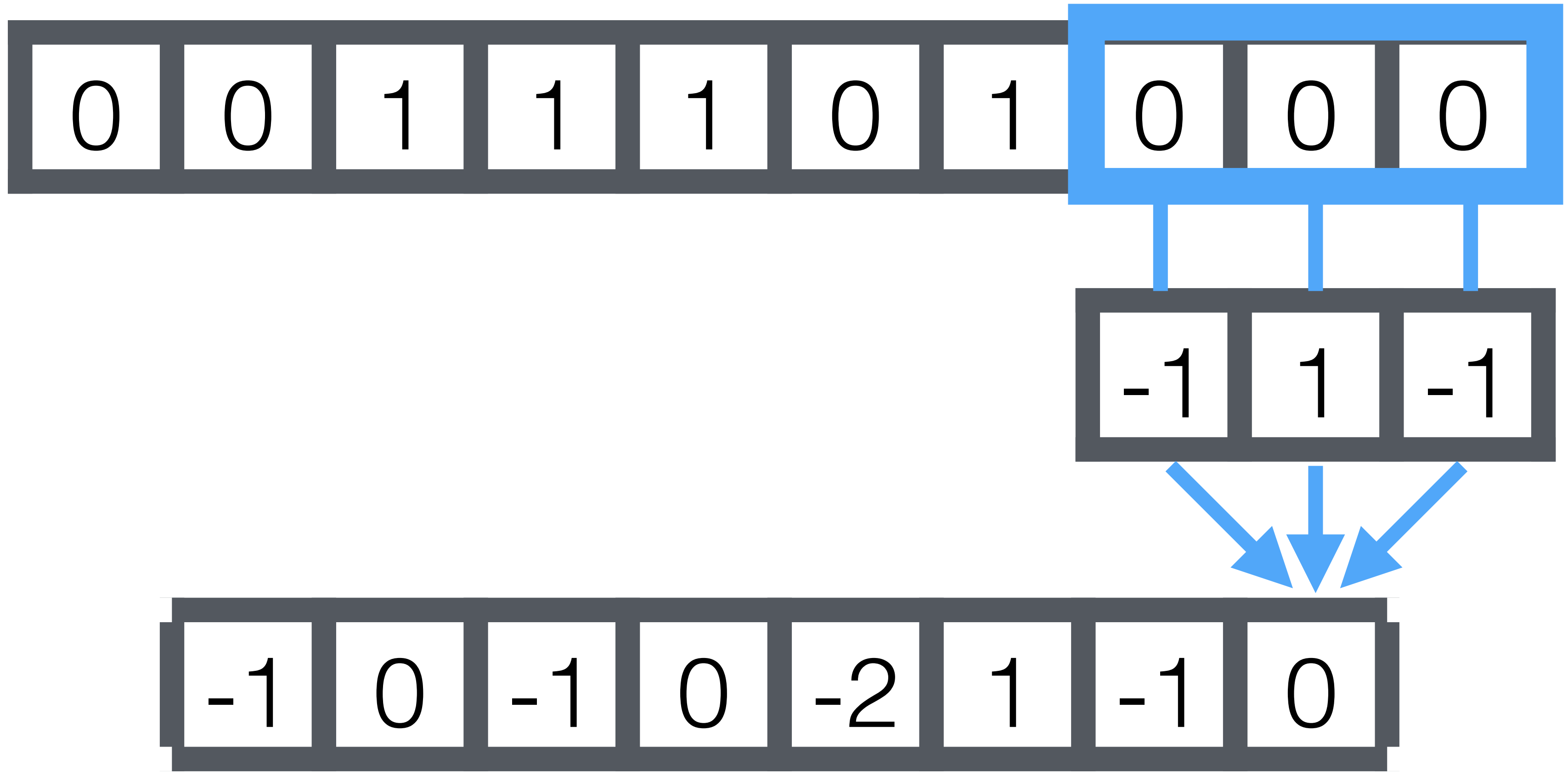
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:



*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image:	0	0	1	1	1	0	1	0	0	0
A filter:	-1	1	-1							
After convolution*:		-1	0	-1	0	-2	1	-1	0	
After ReLU:		0								

*correlation

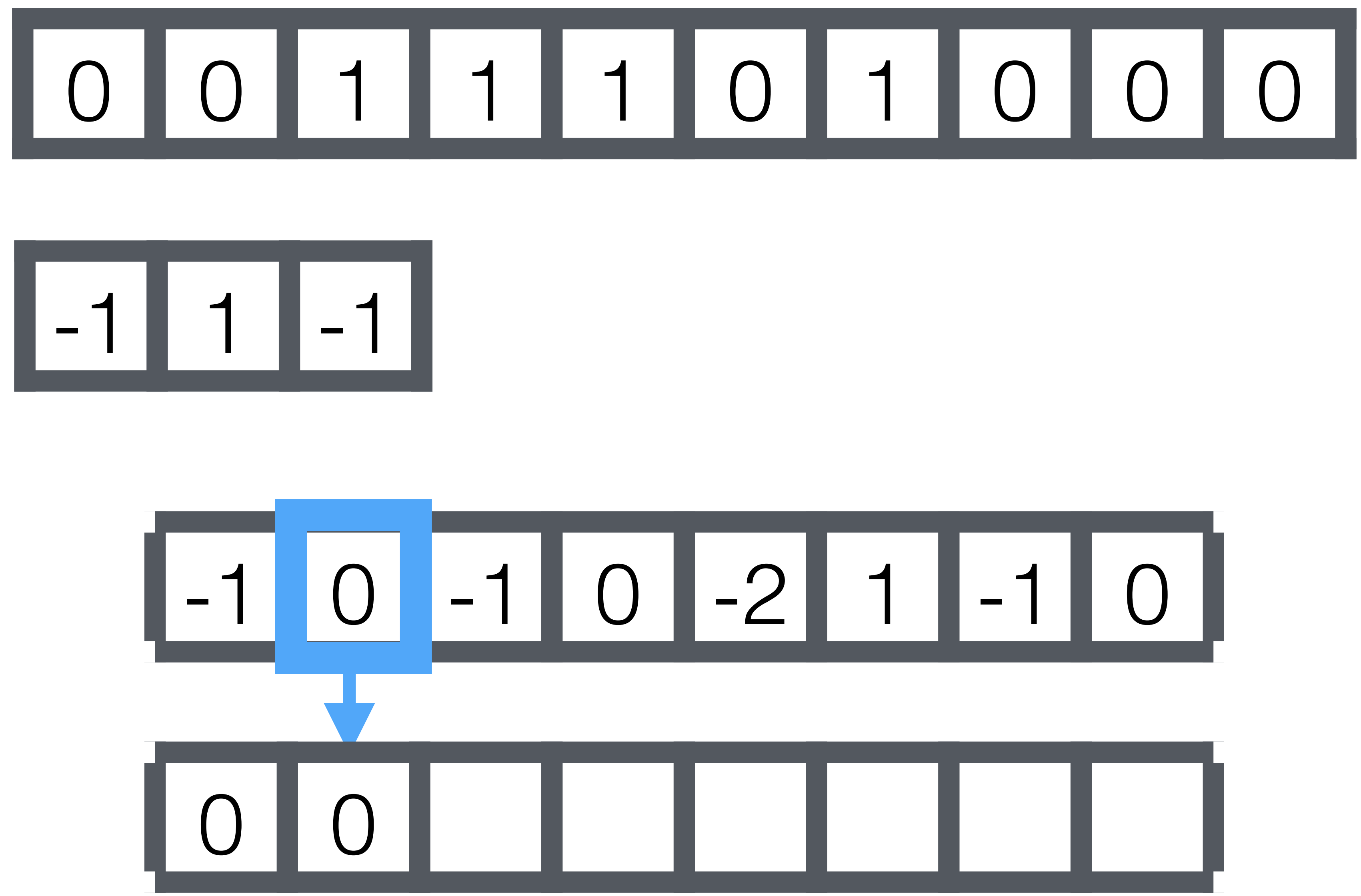
Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

After ReLU:



*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0					

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	$\boxed{0}$	-2	1	-1	0

After ReLU:

0	0	0	0				

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0			

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1		

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1	0	

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1	0	0

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1	0	0

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1	0	0

What does the filter do?

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	1	1	1	0	1	0	0	0

A filter:

-1	1	-1

After convolution*:

-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1	0	0

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

After convolution*:

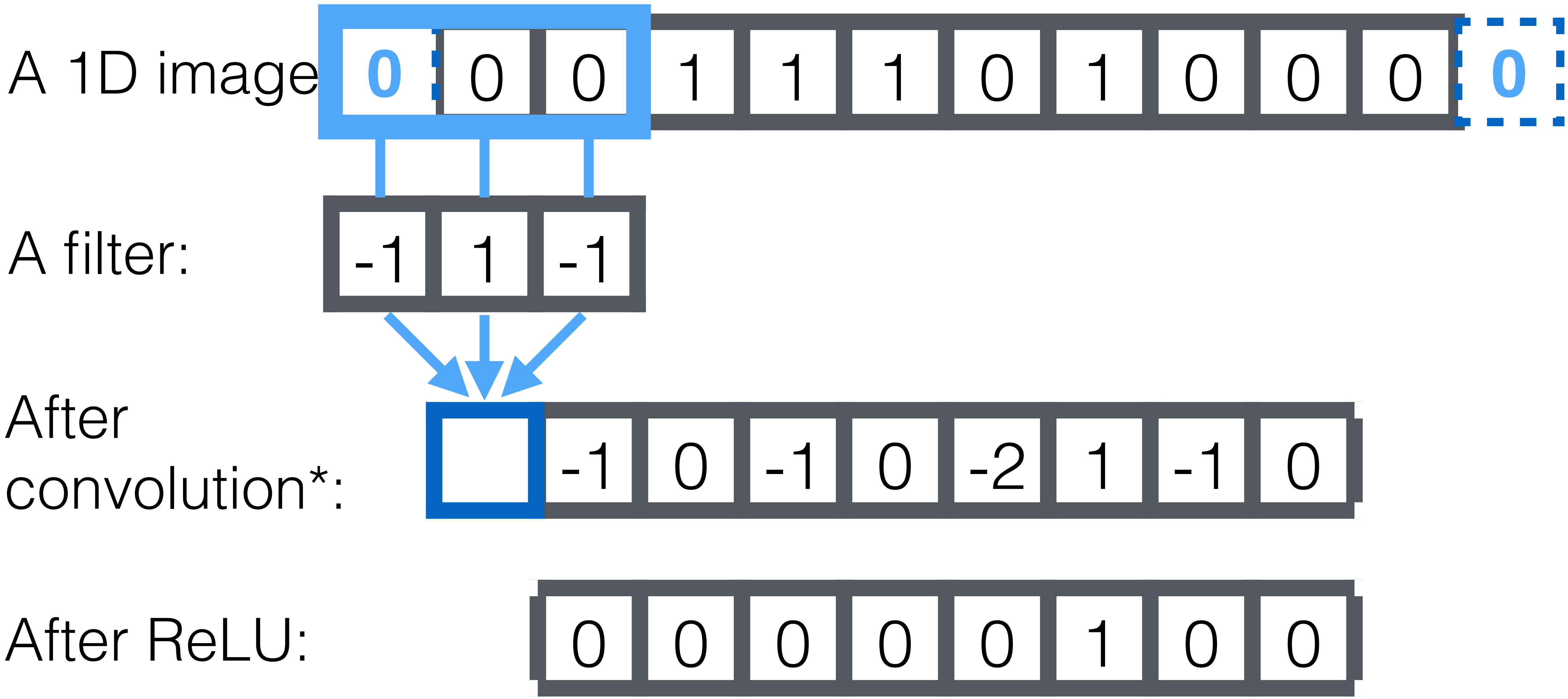
-1	0	-1	0	-2	1	-1	0

After ReLU:

0	0	0	0	0	1	0	0

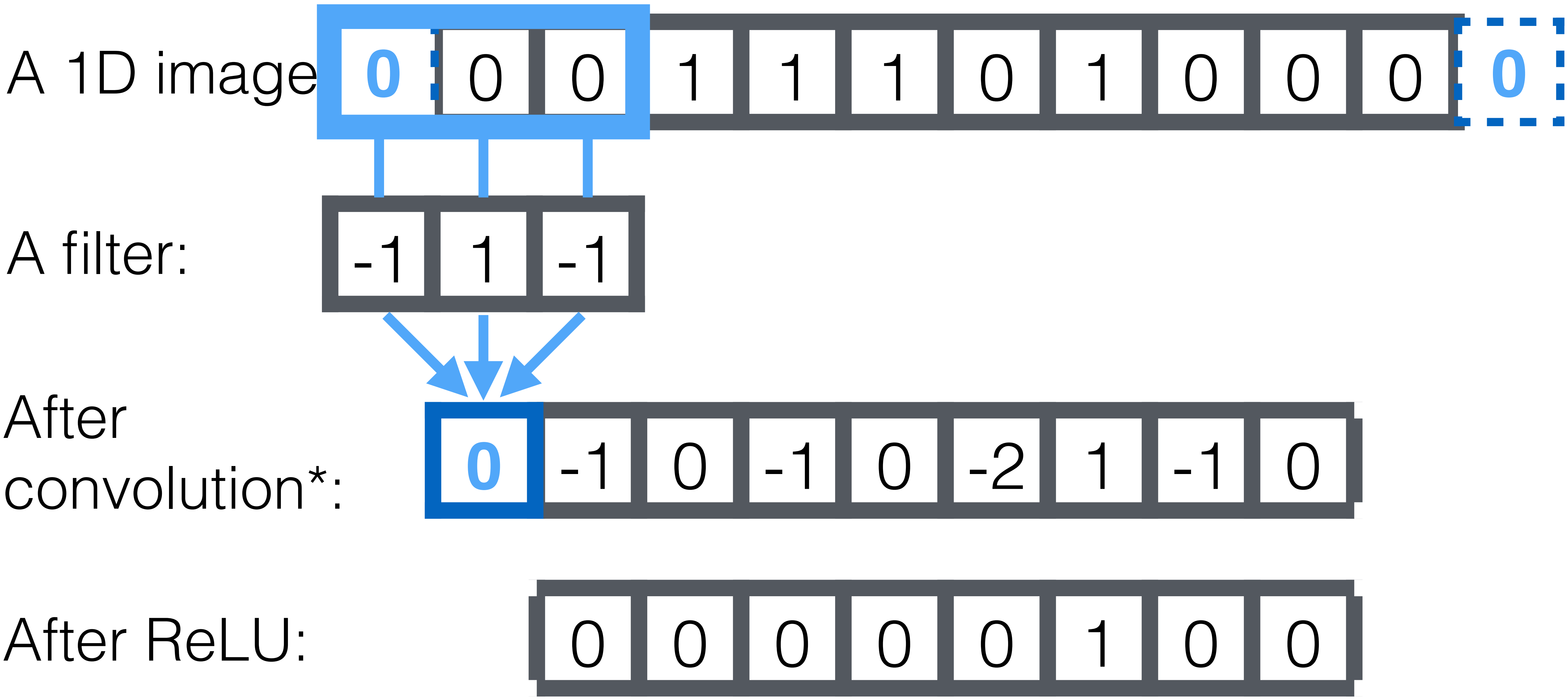
*correlation

Convolutional Layer: 1D example



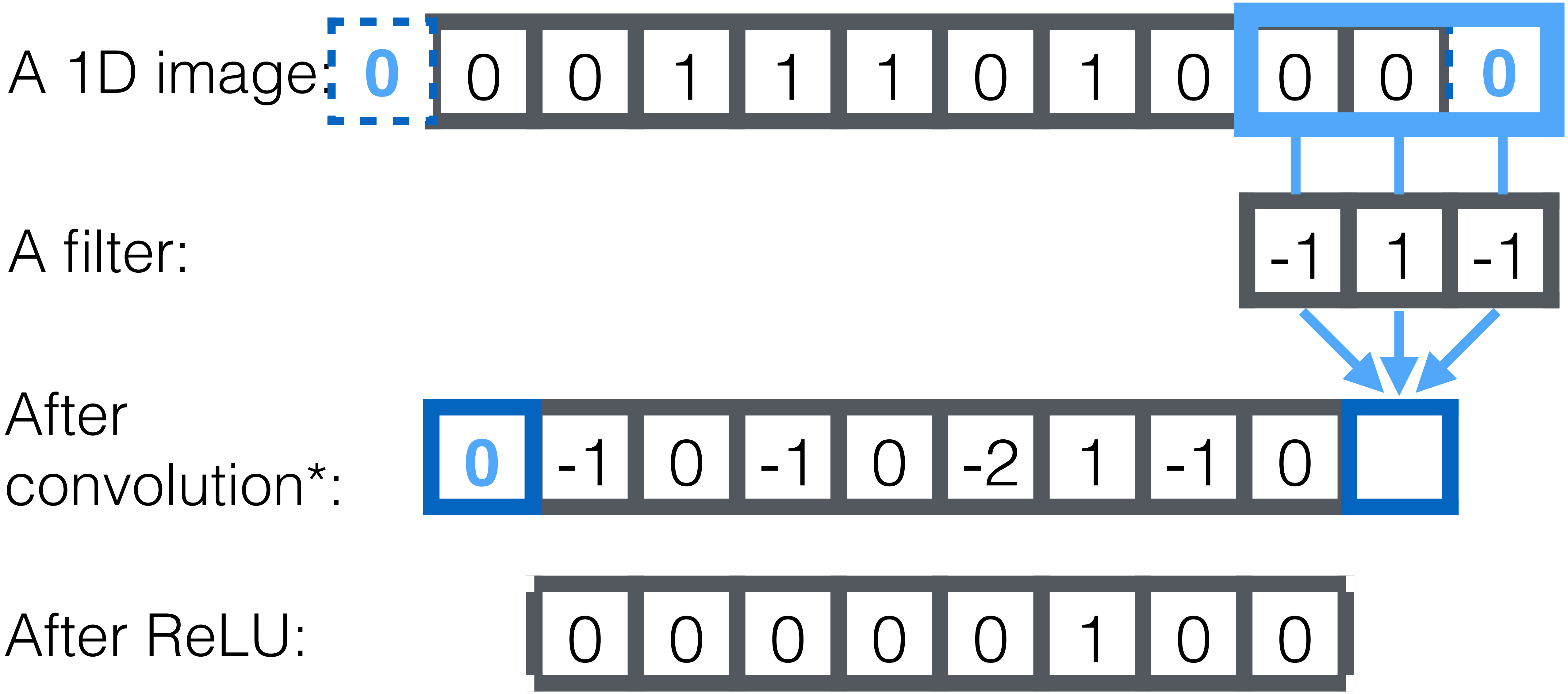
*correlation

Convolutional Layer: 1D example



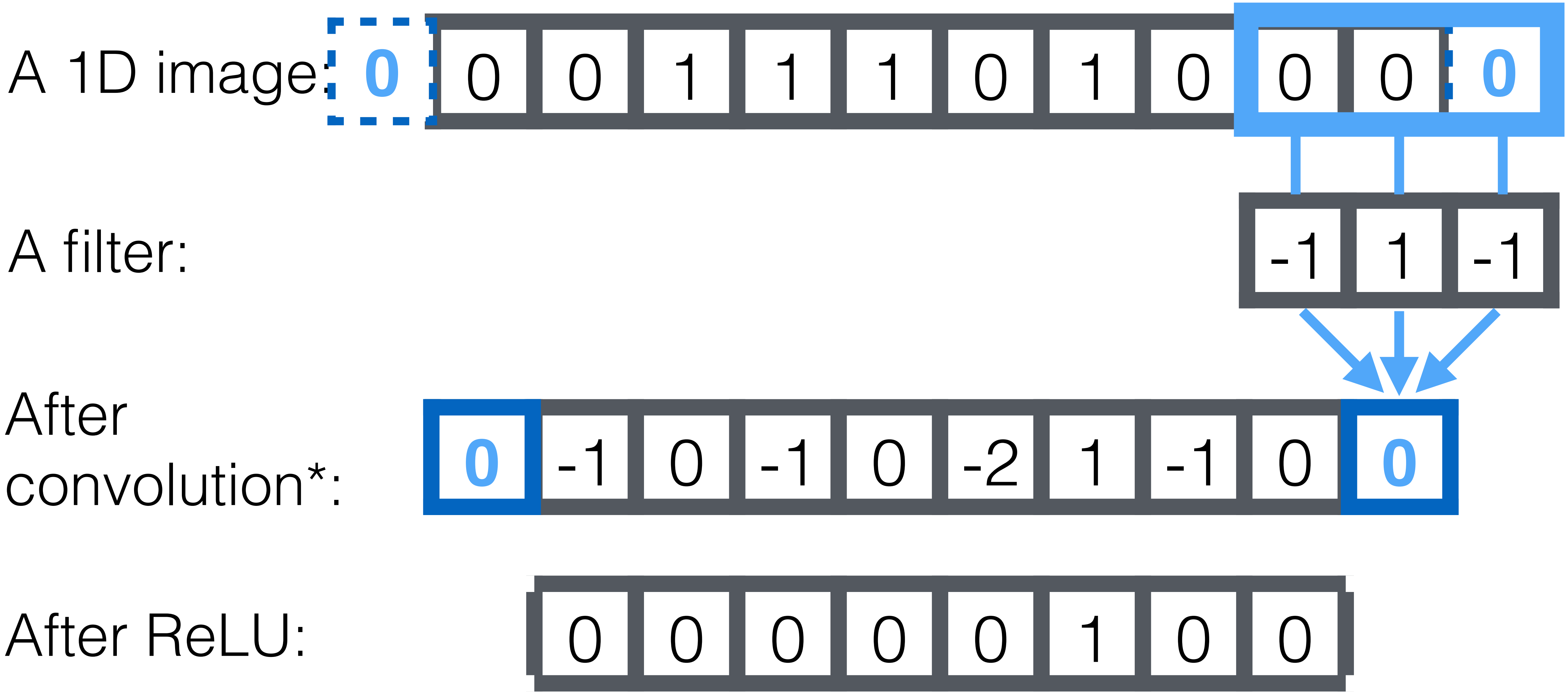
*correlation

Convolutional Layer: 1D example



*correlation

Convolutional Layer: 1D example



*correlation

Convolutional Layer: 1D example

A 1D image:	0	0	0	1	1	1	0	1	0	0	0	0
A filter:	-1	1	-1									
After convolution*:	0	-1	0	-1	0	-2	1	-1	0	0		
After ReLU:	0	0	0	0	0	1	0	0				

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

After convolution*:

0	-1	0	-1	0	-2	1	-1	0	0

After ReLU:

0	0	0	0	0	0	1	0	0	0

*correlation

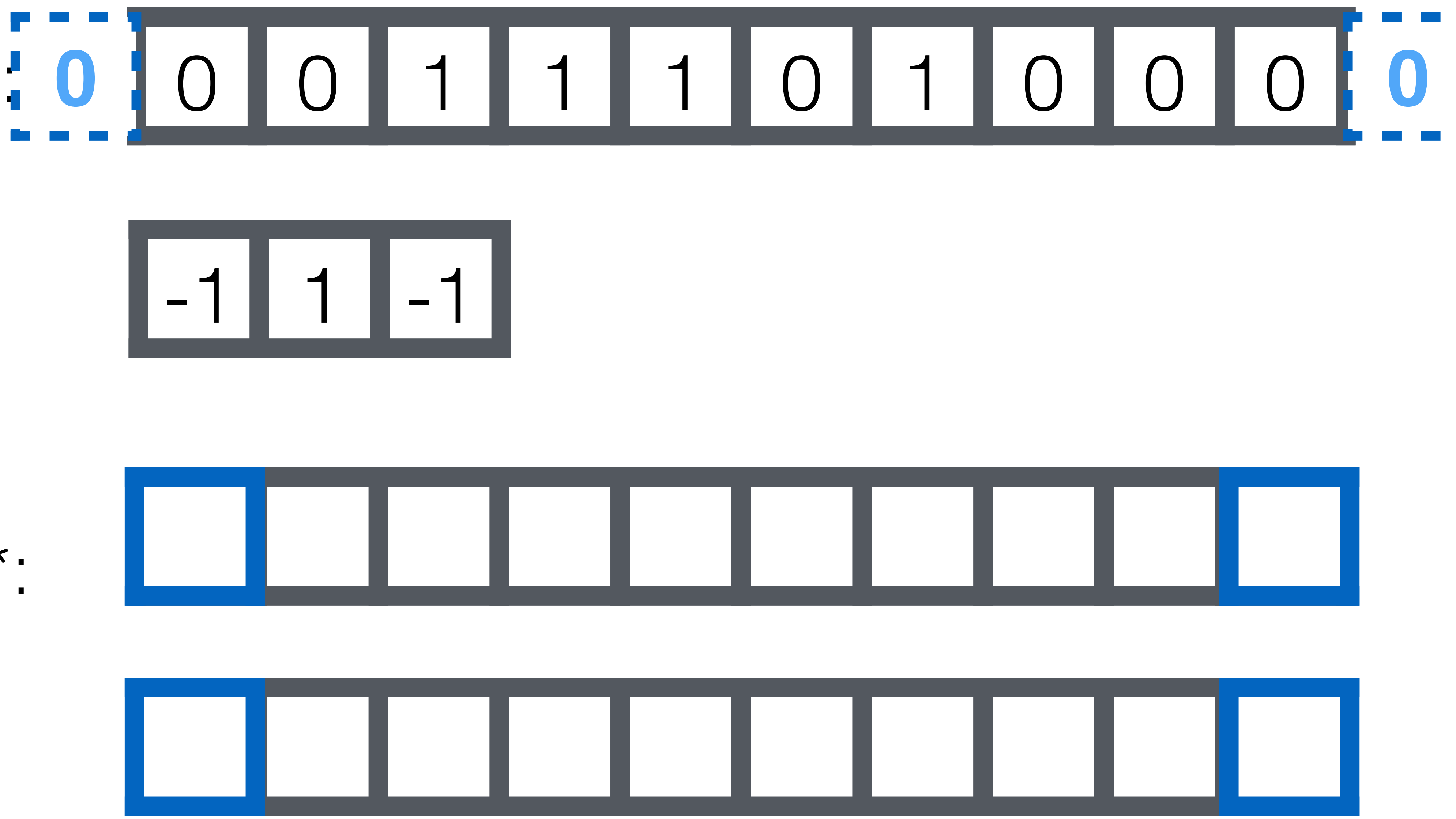
Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

After ReLU:



*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

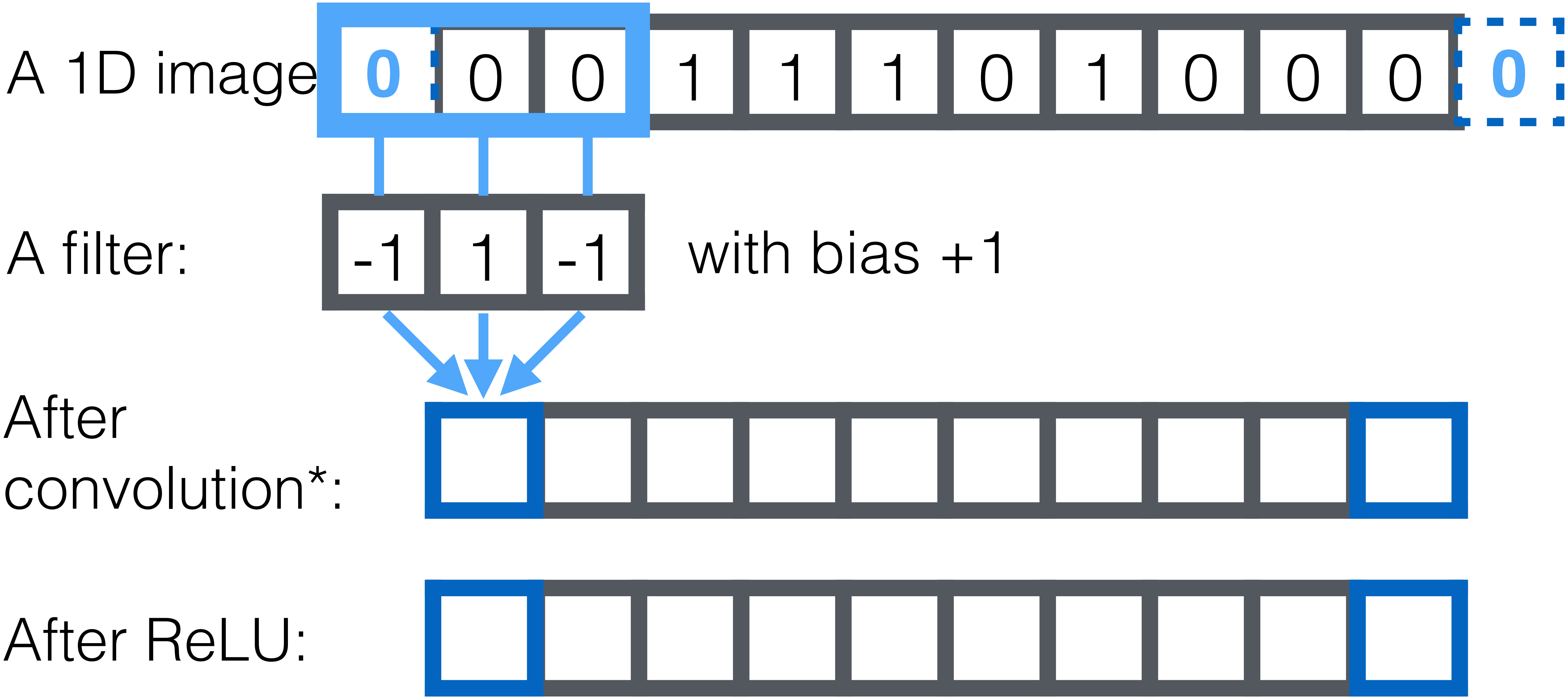
with bias +1

After convolution*:

After ReLU:

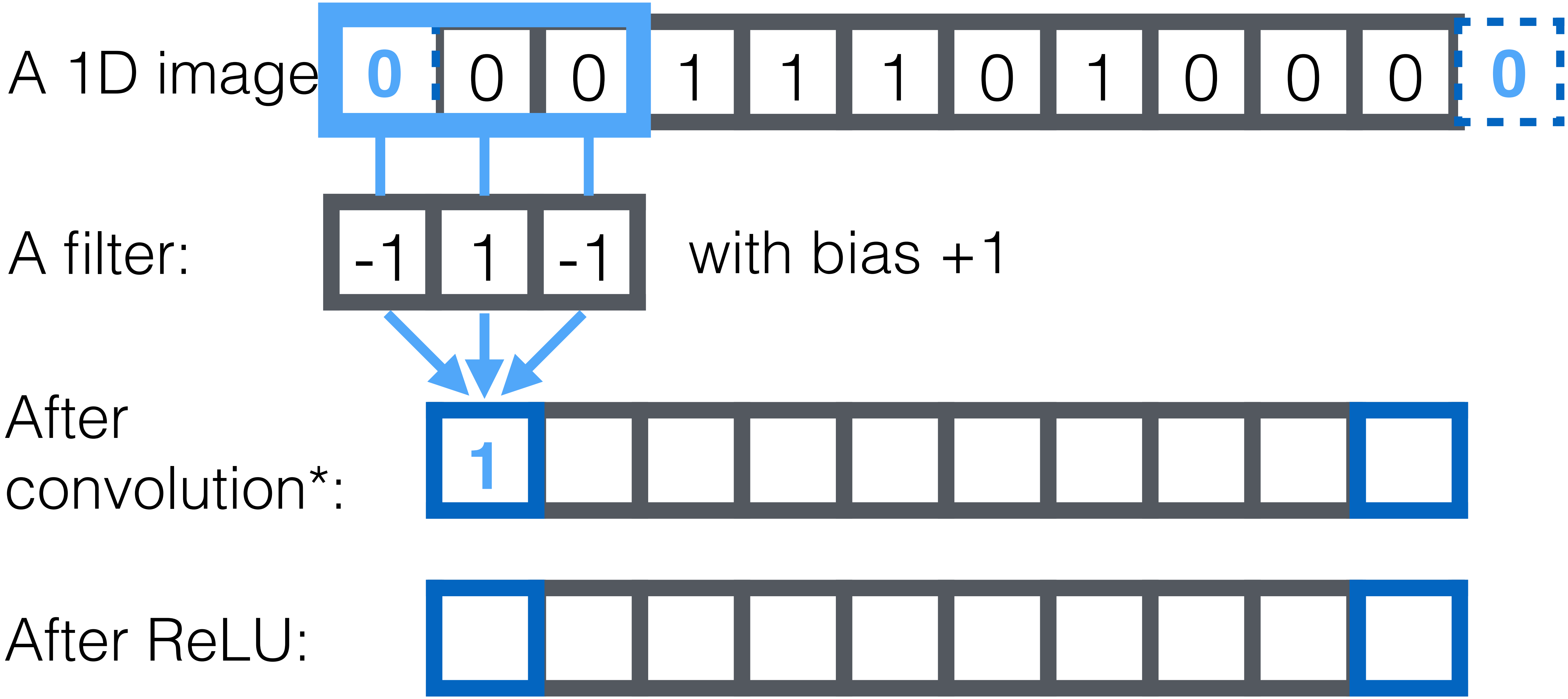
*correlation

Convolutional Layer: 1D example



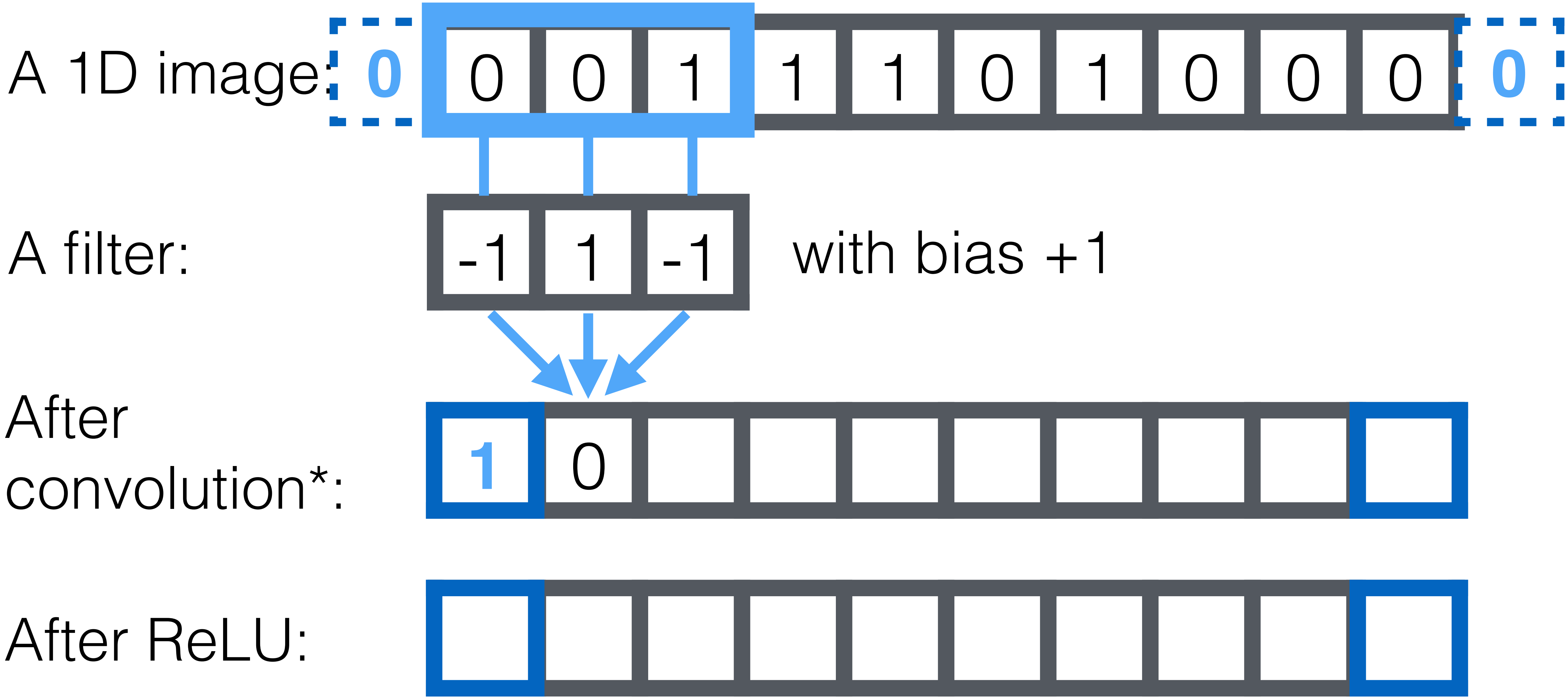
*correlation

Convolutional Layer: 1D example



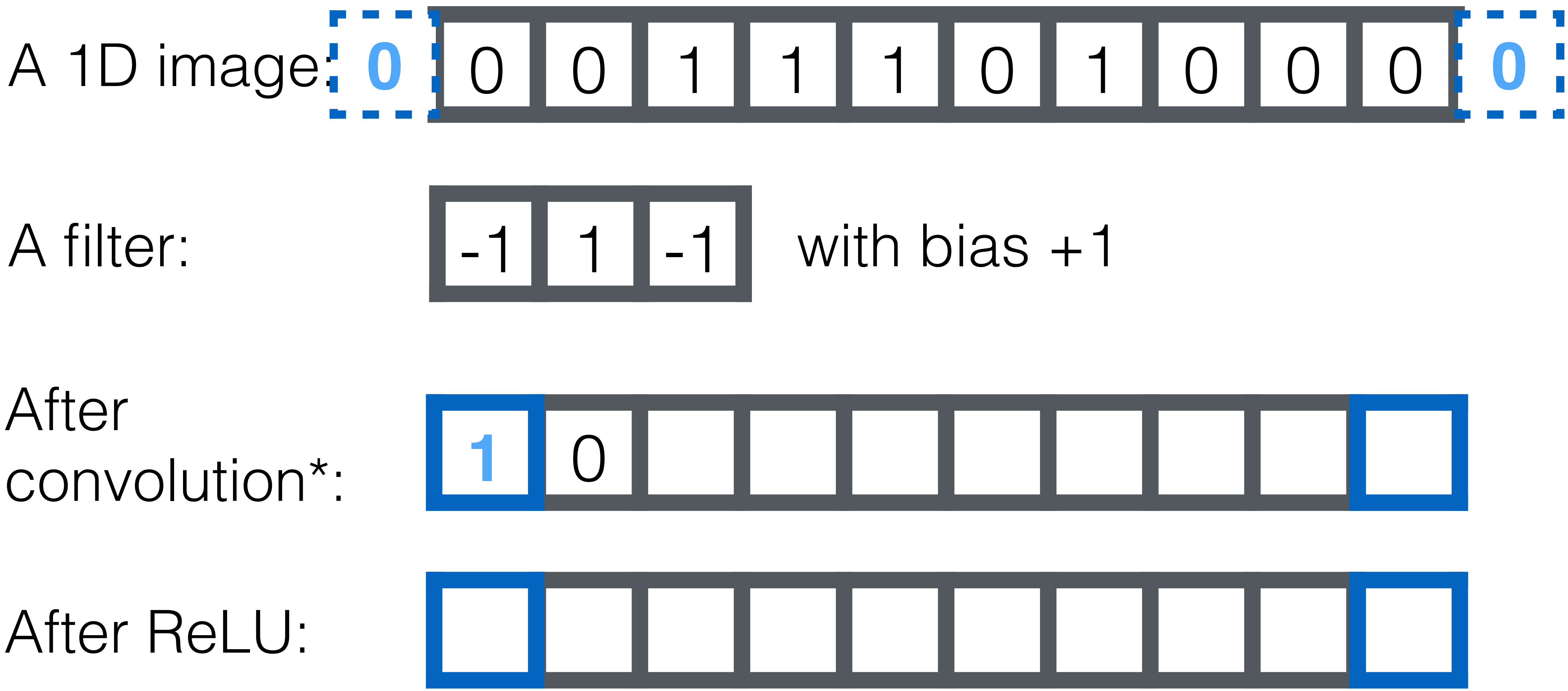
*correlation

Convolutional Layer: 1D example



*correlation

Convolutional Layer: 1D example



*correlation

Convolutional Layer: 1D example

A 1D image:	**0**	0	0	1	1	1	0	1	0	0	0	**0**
A filter:		-1	1	-1	with bias +1							
After convolution*:		1	0	1	0	1	-1	2	0	1	1	
After ReLU:												

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

with bias +1

After convolution*:

1	0	1	0	1	-1	2	0	1	1

After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image: 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0

A filter: -1 | 1 | -1 with bias +1

After convolution*: 1 | 0 | 1 | 0 | 1 | -1 | 2 | 0 | 1 | 1

After ReLU: 1

*correlation

Convolutional Layer: 1D example

A 1D image: 0 | 0 0 1 1 1 0 1 0 0 0 | 0

A filter: -1 1 -1 with bias +1

After convolution*: 1 0 1 0 1 -1 2 0 1 1

After ReLU: 1 0

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

with bias +1

After convolution*:

1	0	1	0	1	-1	2	0	1	1

After ReLU:

1	0								

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

with bias +1

After convolution*:

1	0	1	0	1	-1	2	0	1	1

After ReLU:

1	0	1	0	1	0	2	0	1	1

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

-1	1	-1

with bias +1

After convolution*:

After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image: 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0

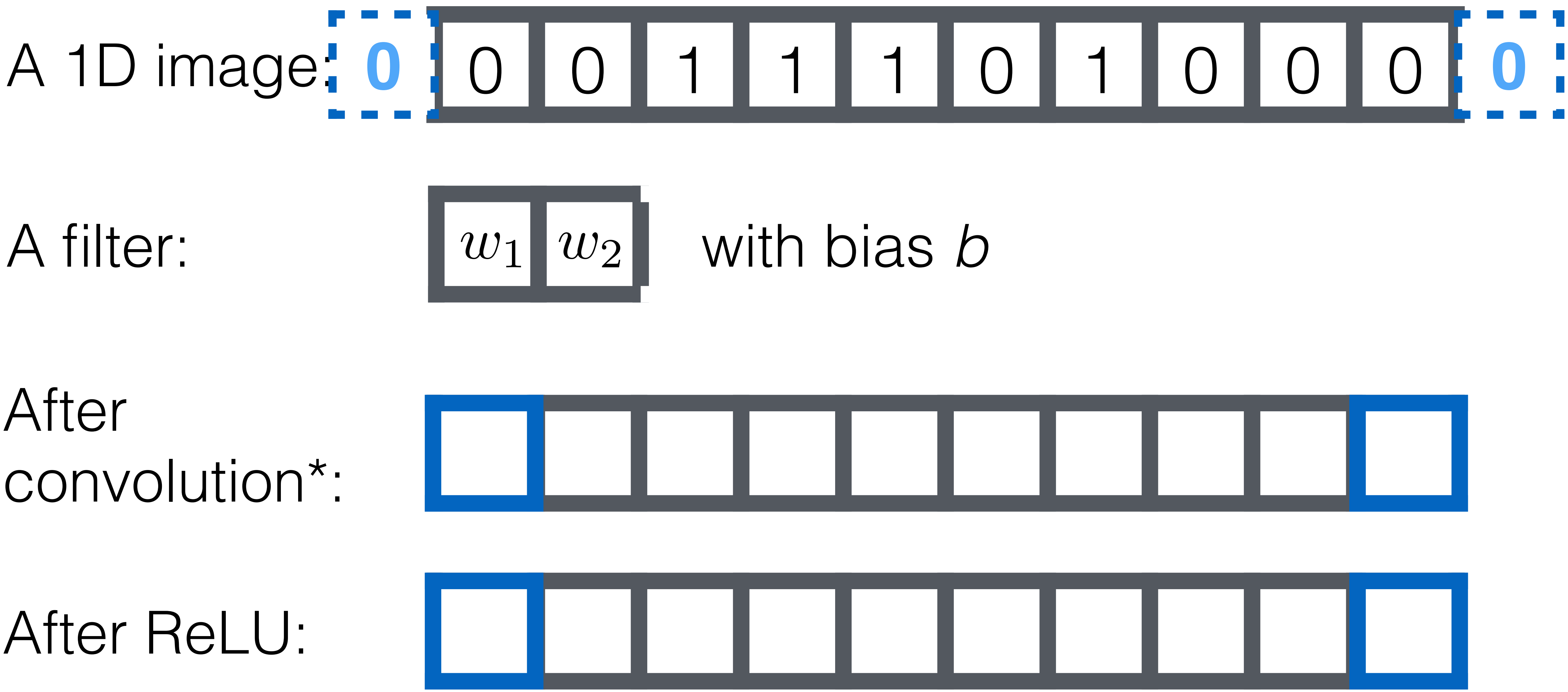
A filter: w_1 | w_2 | w_3 with bias *b*

After convolution*:

After ReLU:

*correlation

Convolutional Layer: 1D example



*correlation

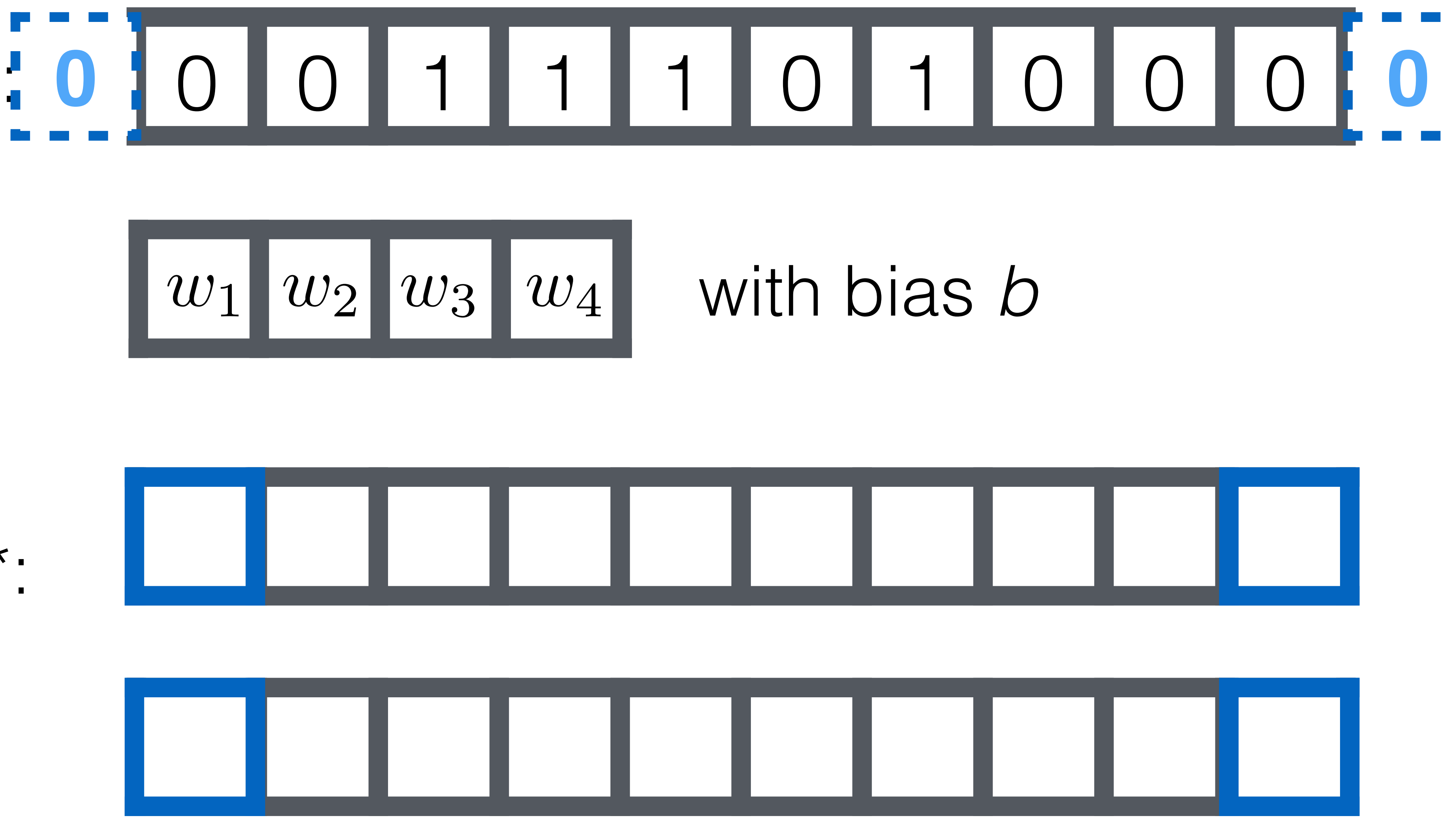
Convolutional Layer: 1D example

A 1D image:

A filter:

After convolution*:

After ReLU:



*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

w_1	w_2	w_3

with bias *b*

After convolution*:

After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

w_1	w_2	w_3

with bias b

After convolution*:

After ReLU:

- How many weights (including bias)?

*correlation

Convolutional Layer: 1D example

A 1D image: 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0

A filter: w_1 | w_2 | w_3 with bias *b*

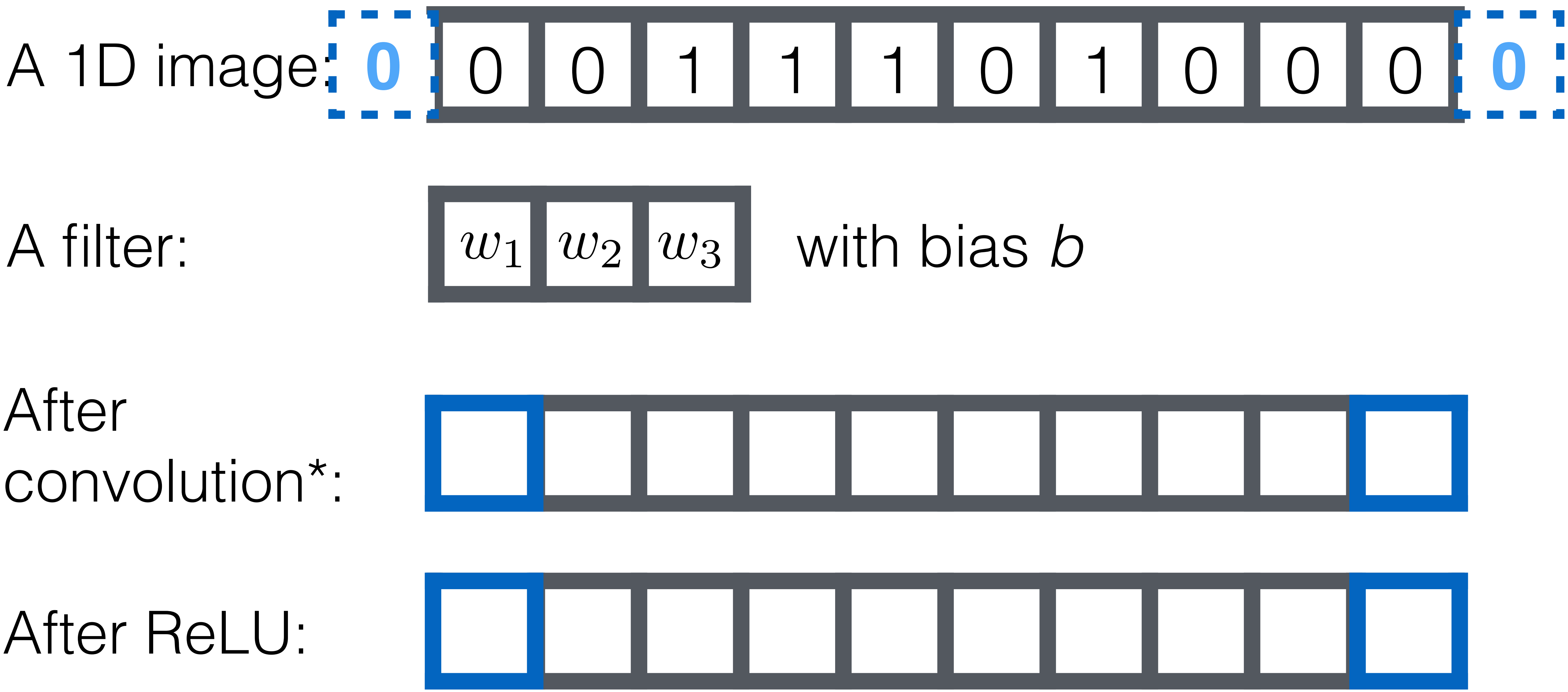
After convolution*:

After ReLU:

- How many weights (including bias)? 4

*correlation

Convolutional Layer: 1D example



- How many weights (including bias)? 4
- How many weights (including biases) for fully connected layer with 10 inputs & 10 outputs?

*correlation

Convolutional Layer: 1D example

A 1D image:

0	0	0	1	1	1	0	1	0	0	0	0

A filter:

w_1	w_2	w_3

with bias b

After convolution*:

After ReLU:

- How many weights (including bias)? 4
- How many weights (including biases) for fully connected layer with 10 inputs & 10 outputs? 10 x 11 =

*correlation

Convolutional Layer: 1D example

A 1D image: 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0

A filter: w_1 | w_2 | w_3 with bias *b*

After convolution*:

After ReLU:

- How many weights (including bias)? 4
- How many weights (including biases) for fully connected layer with 10 inputs & 10 outputs? 10 x 11 = 110

*correlation

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

Convolutional Layer: 2D example

A 2D image:

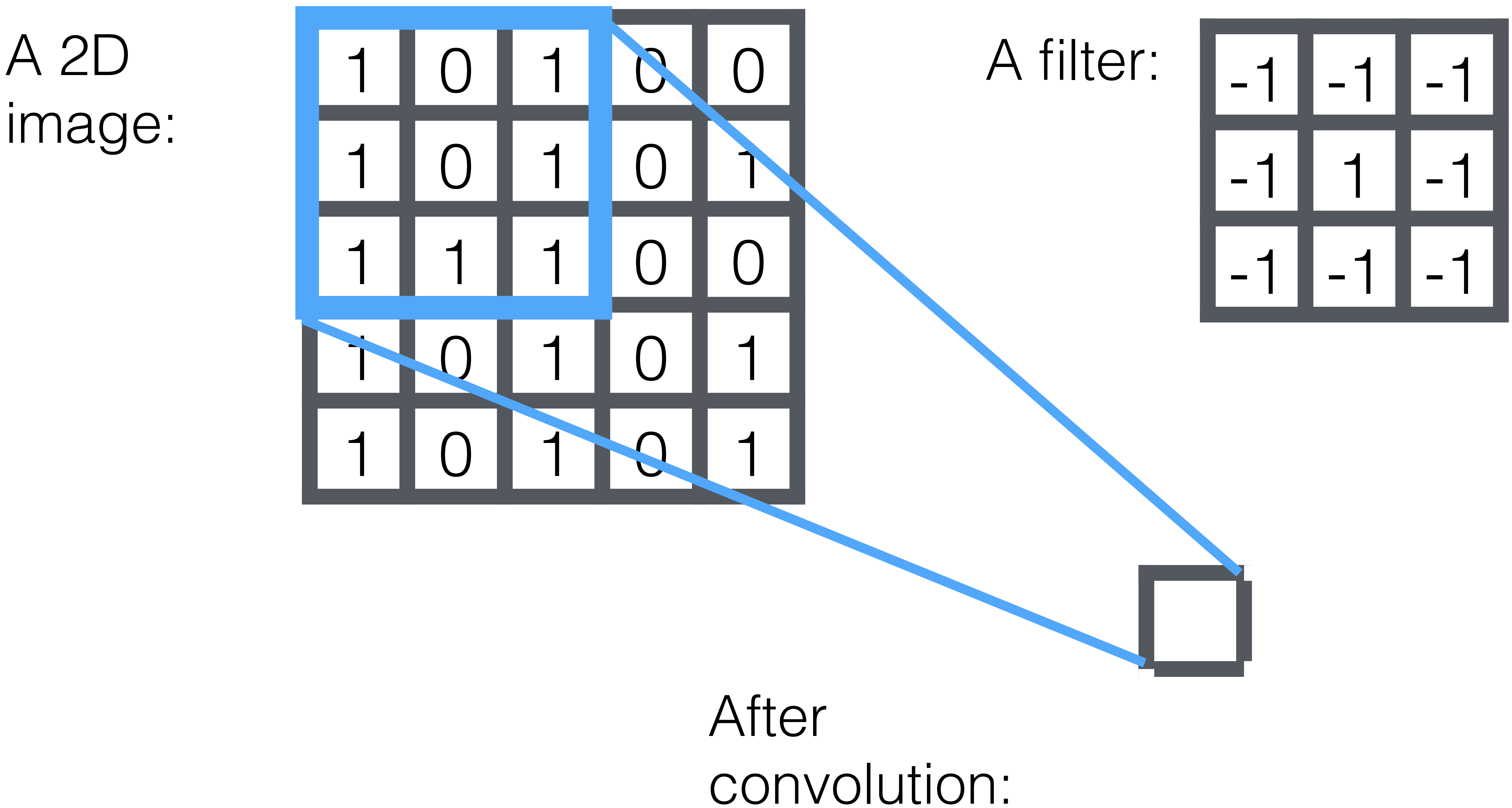
1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

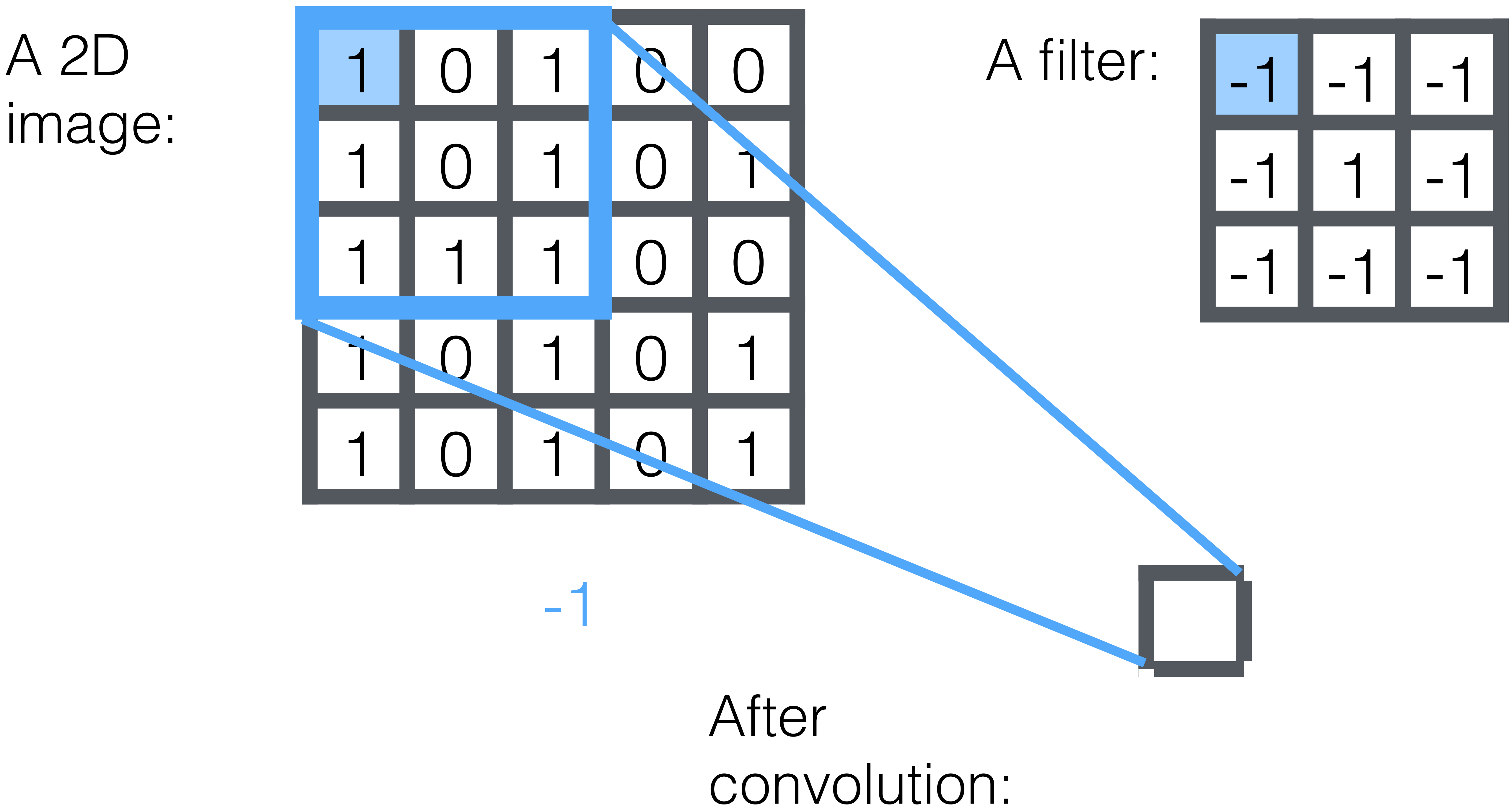
1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

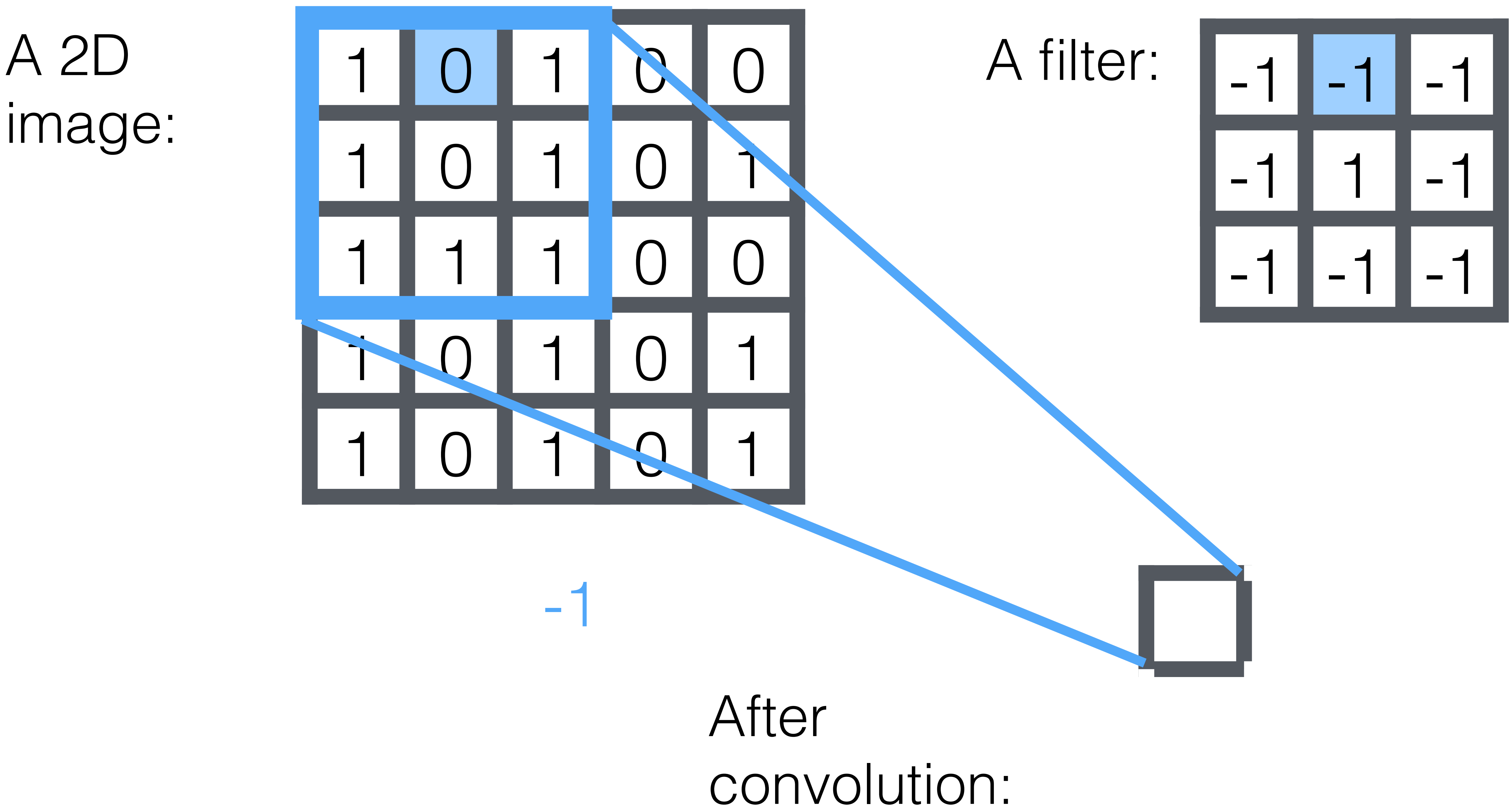
-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

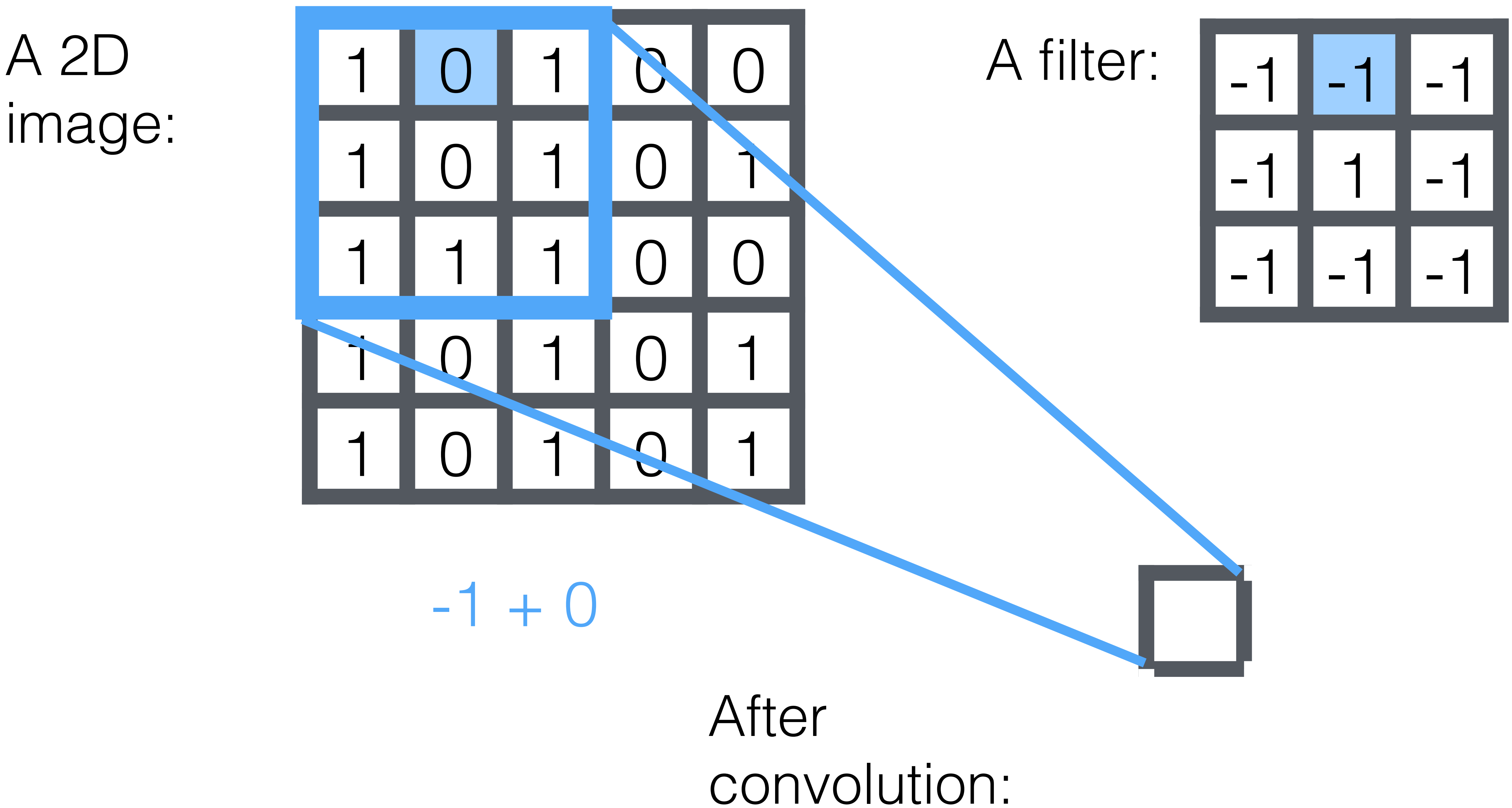
Convolutional Layer: 2D example



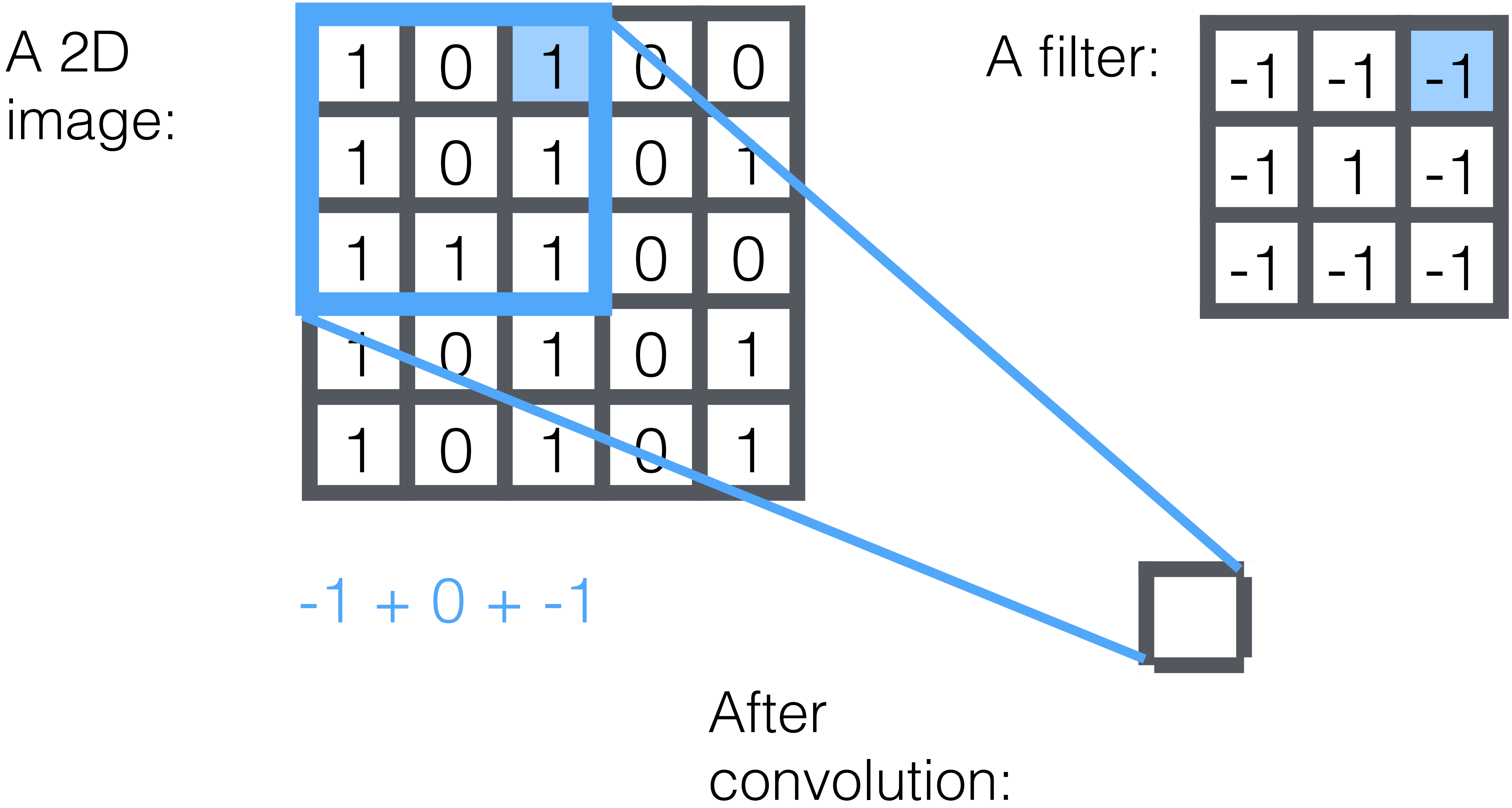
Convolutional Layer: 2D example



Convolutional Layer: 2D example



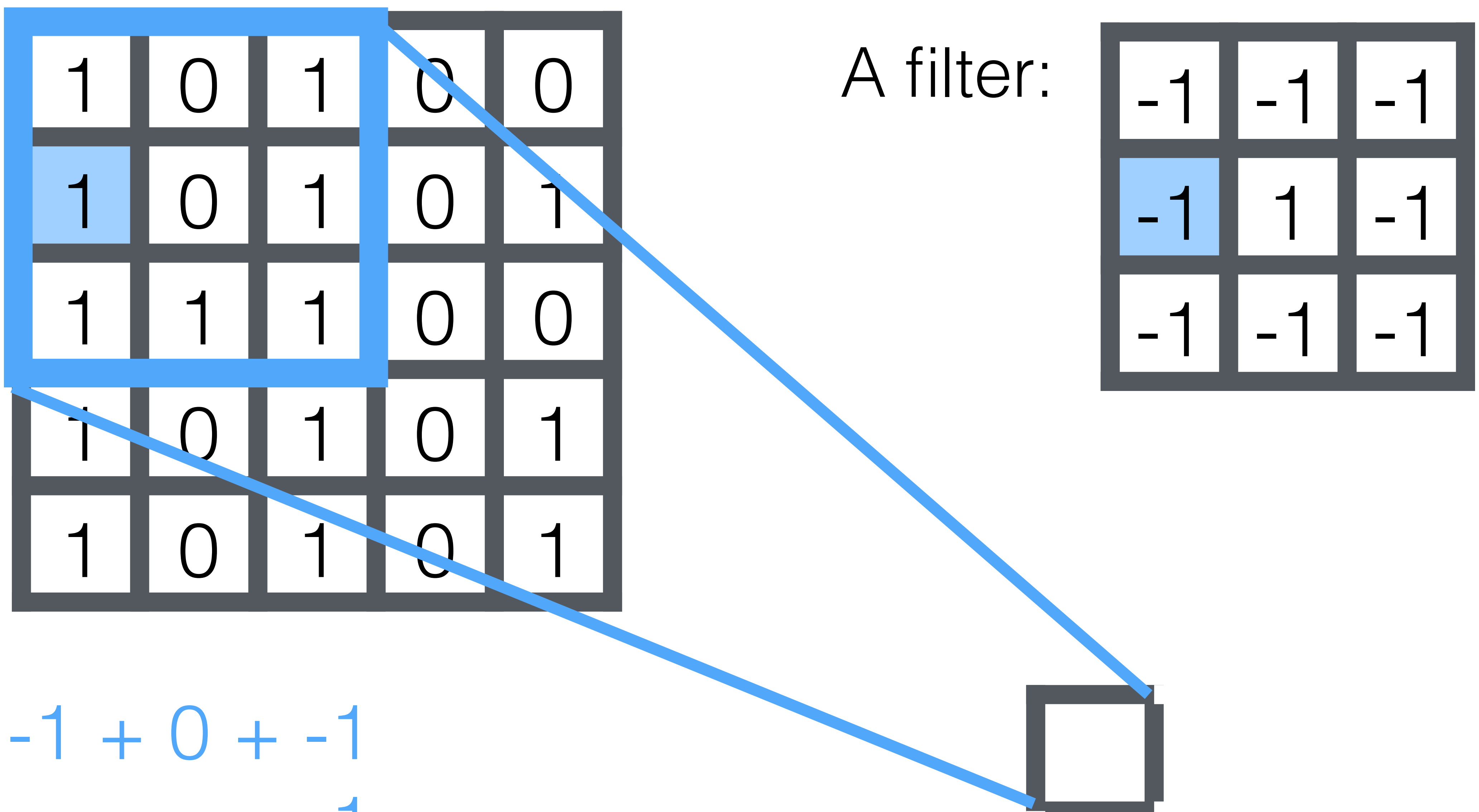
Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

A filter:



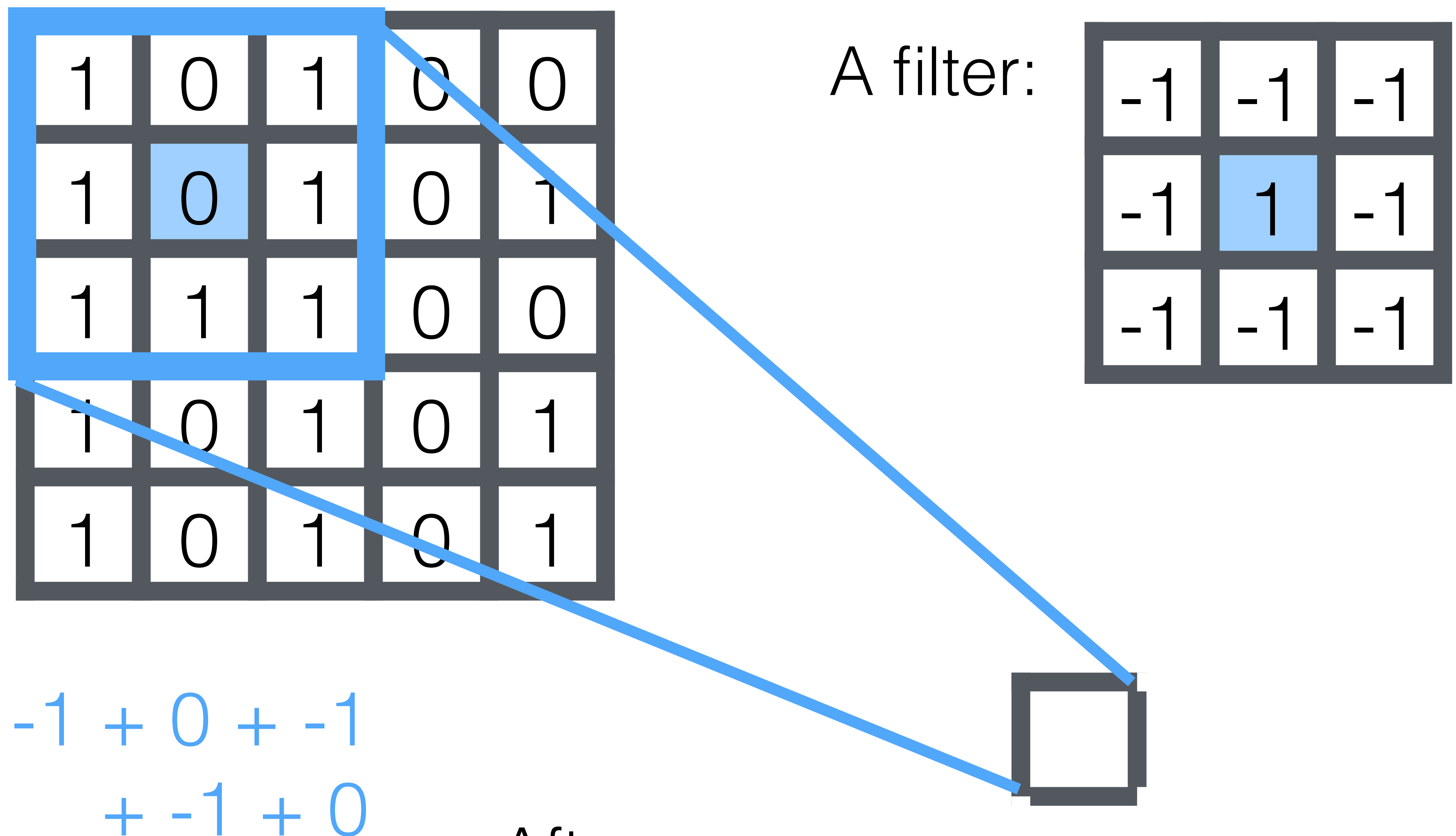
-1 + 0 + -1
+ -1

After convolution:

Convolutional Layer: 2D example

A 2D image:

A filter:

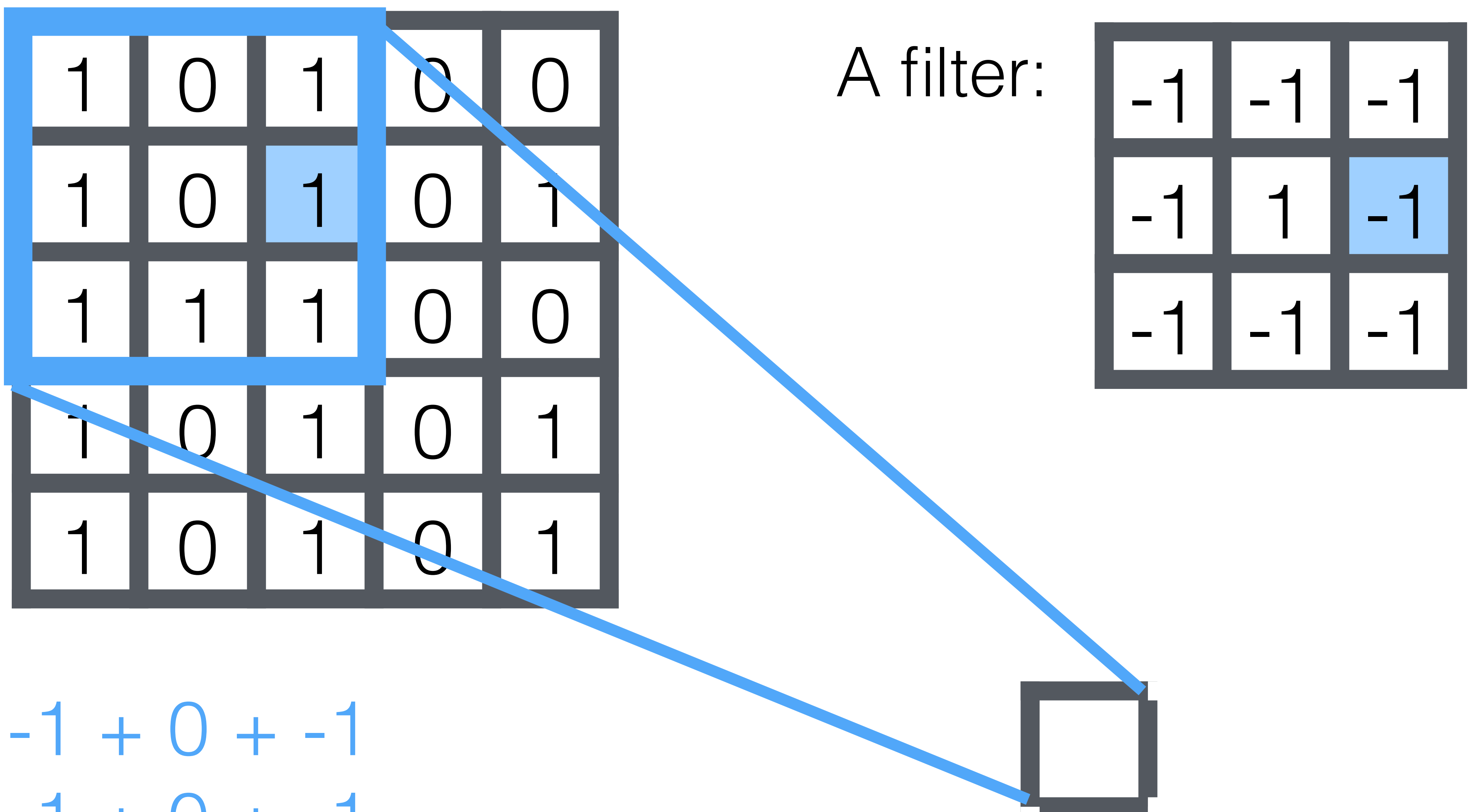


After convolution:

Convolutional Layer: 2D example

A 2D image:

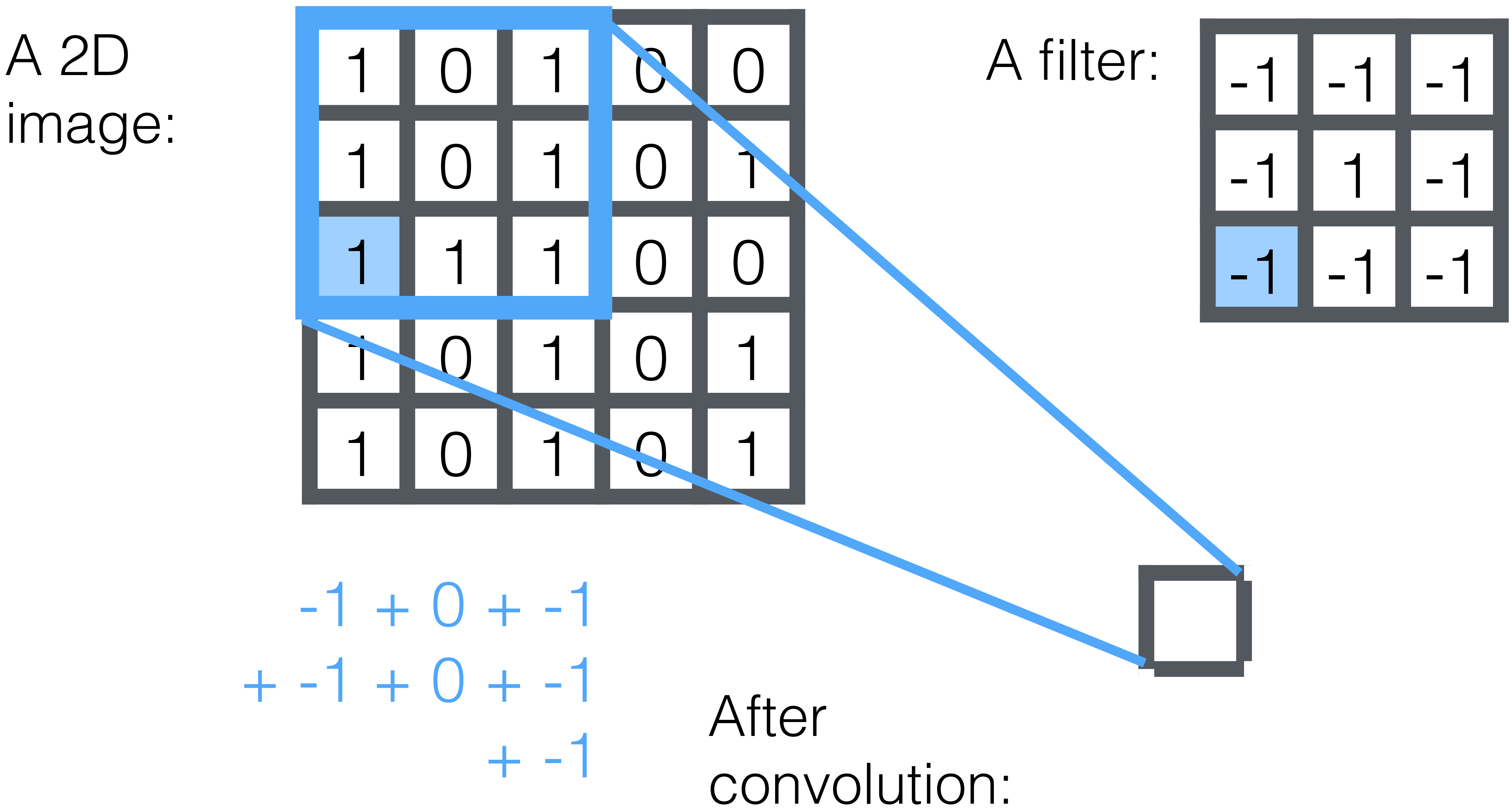
A filter:



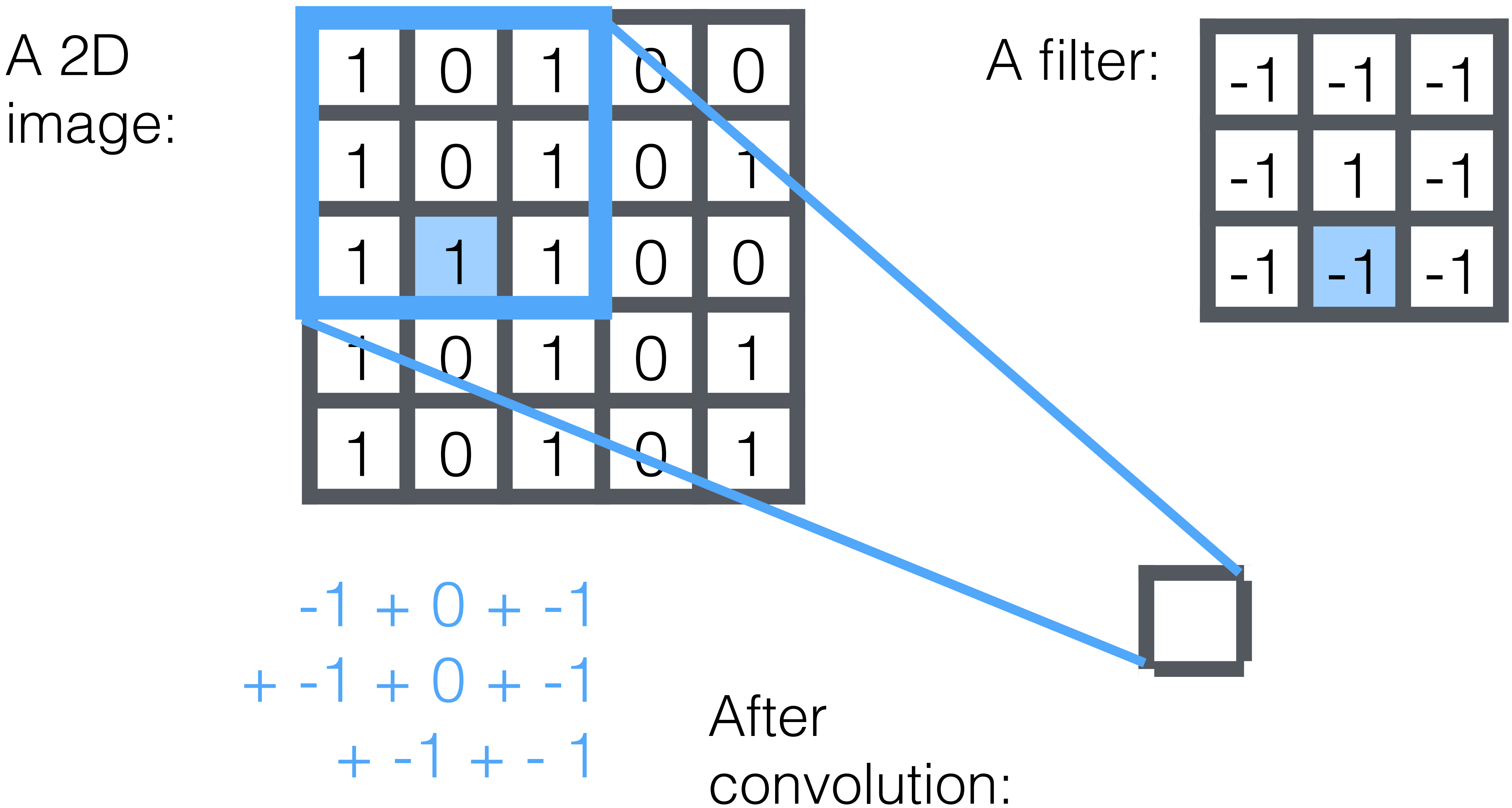
-1 + 0 + -1
+ -1 + 0 + -1

After convolution:

Convolutional Layer: 2D example



Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

A filter:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

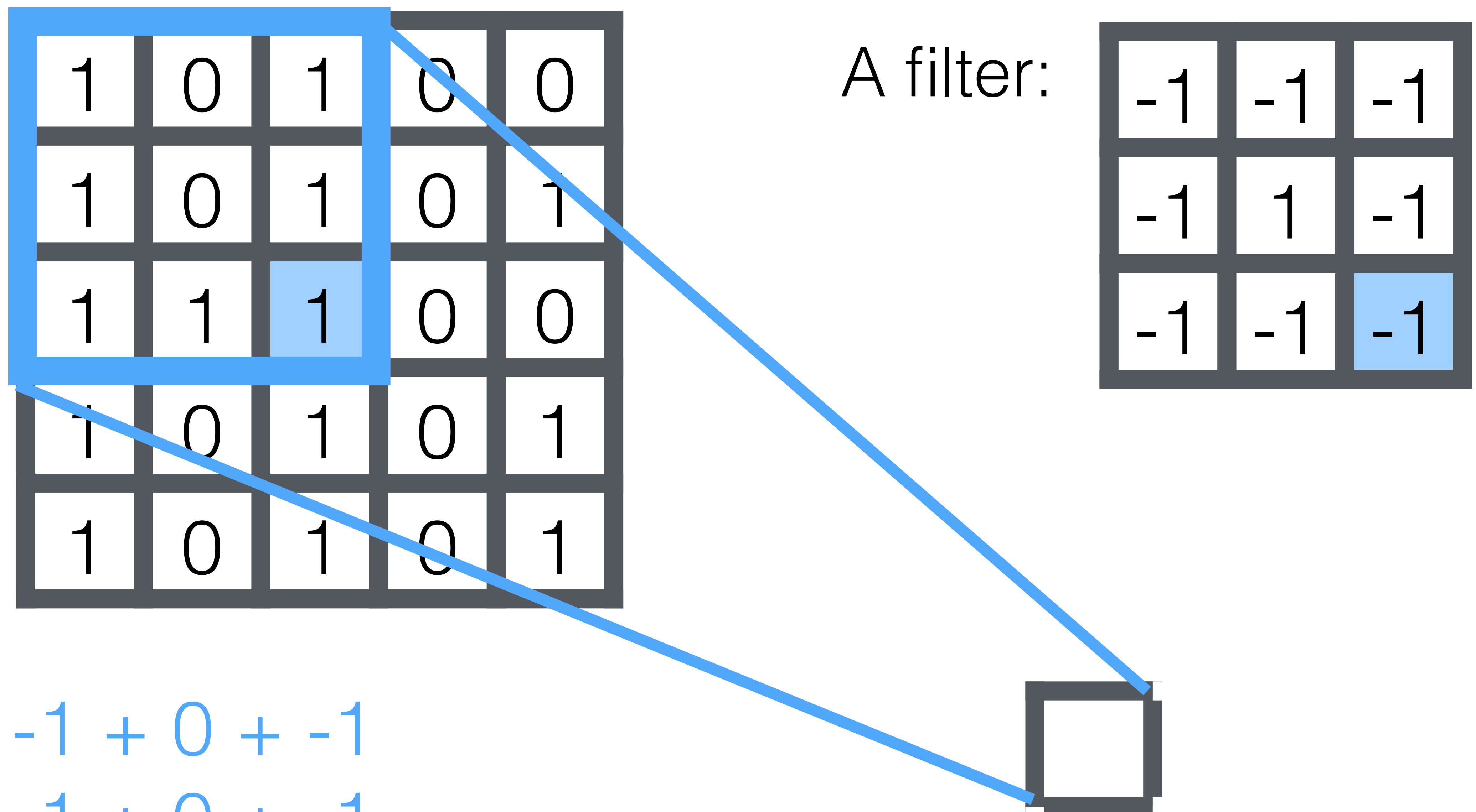
$$\begin{aligned} & -1 + 0 + -1 \\ & + -1 + 0 + -1 \\ & + -1 + -1 + -1 \end{aligned}$$

After convolution:

Convolutional Layer: 2D example

A 2D image:

A filter:



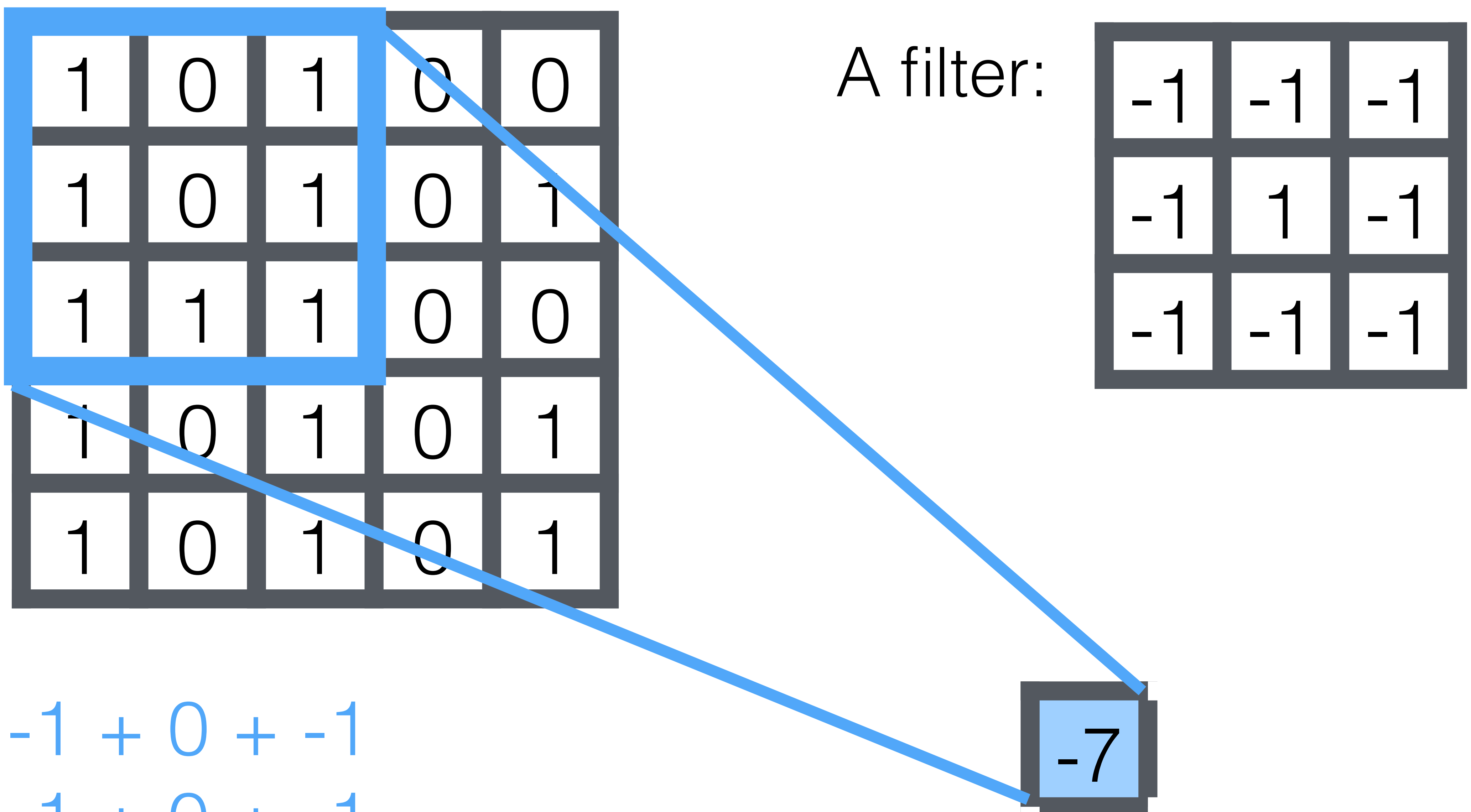
$$\begin{aligned} &-1 + 0 + -1 \\ &+ -1 + 0 + -1 \\ &+ -1 + -1 + -1 \\ &= -7 \end{aligned}$$

After convolution:

Convolutional Layer: 2D example

A 2D image:

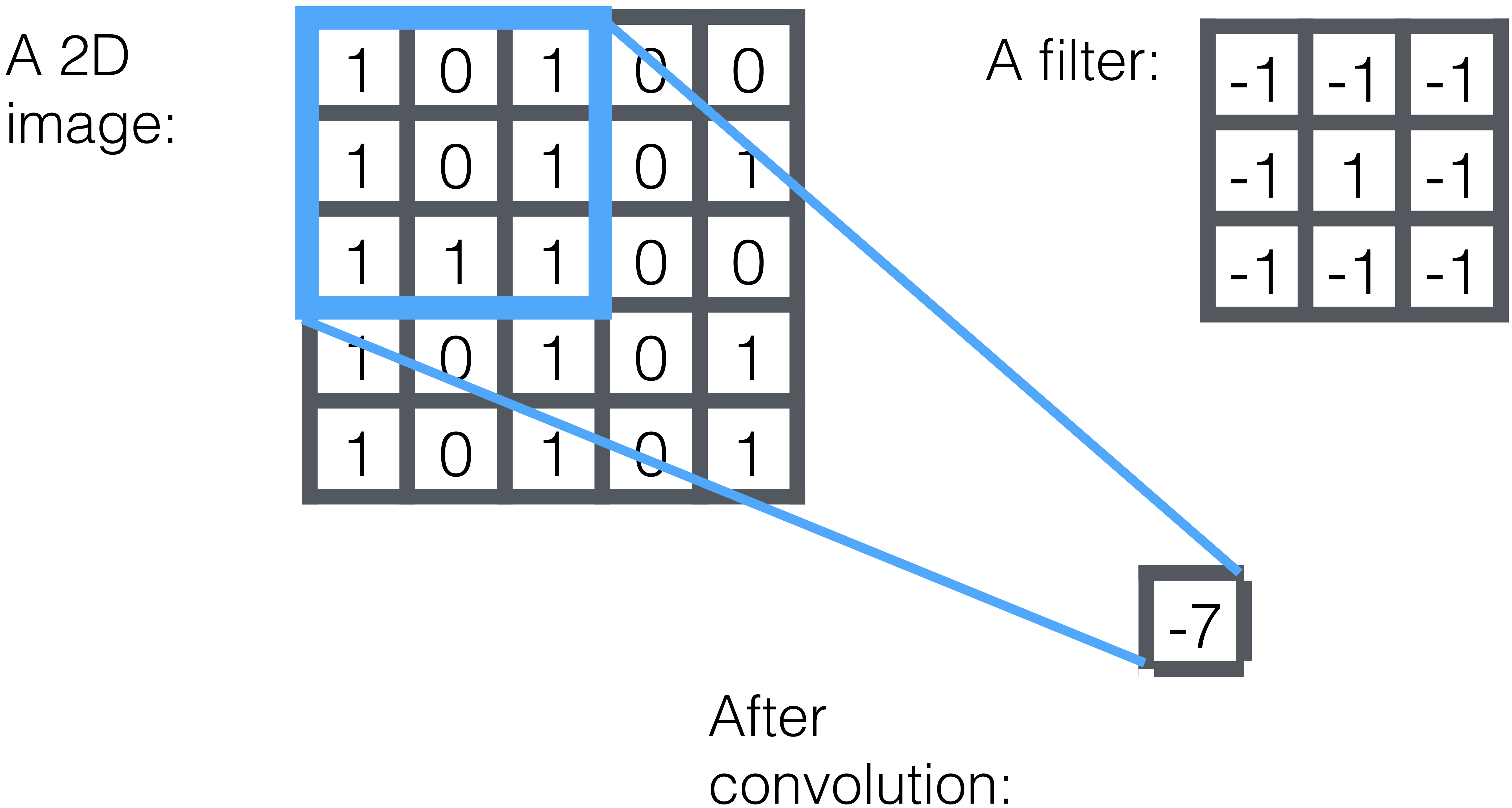
A filter:



$$
\begin{aligned}
&-1 + 0 + -1 \\
&+ -1 + 0 + -1 \\
&+ -1 + -1 + -1 \\
&= -7
\end{aligned}
$$

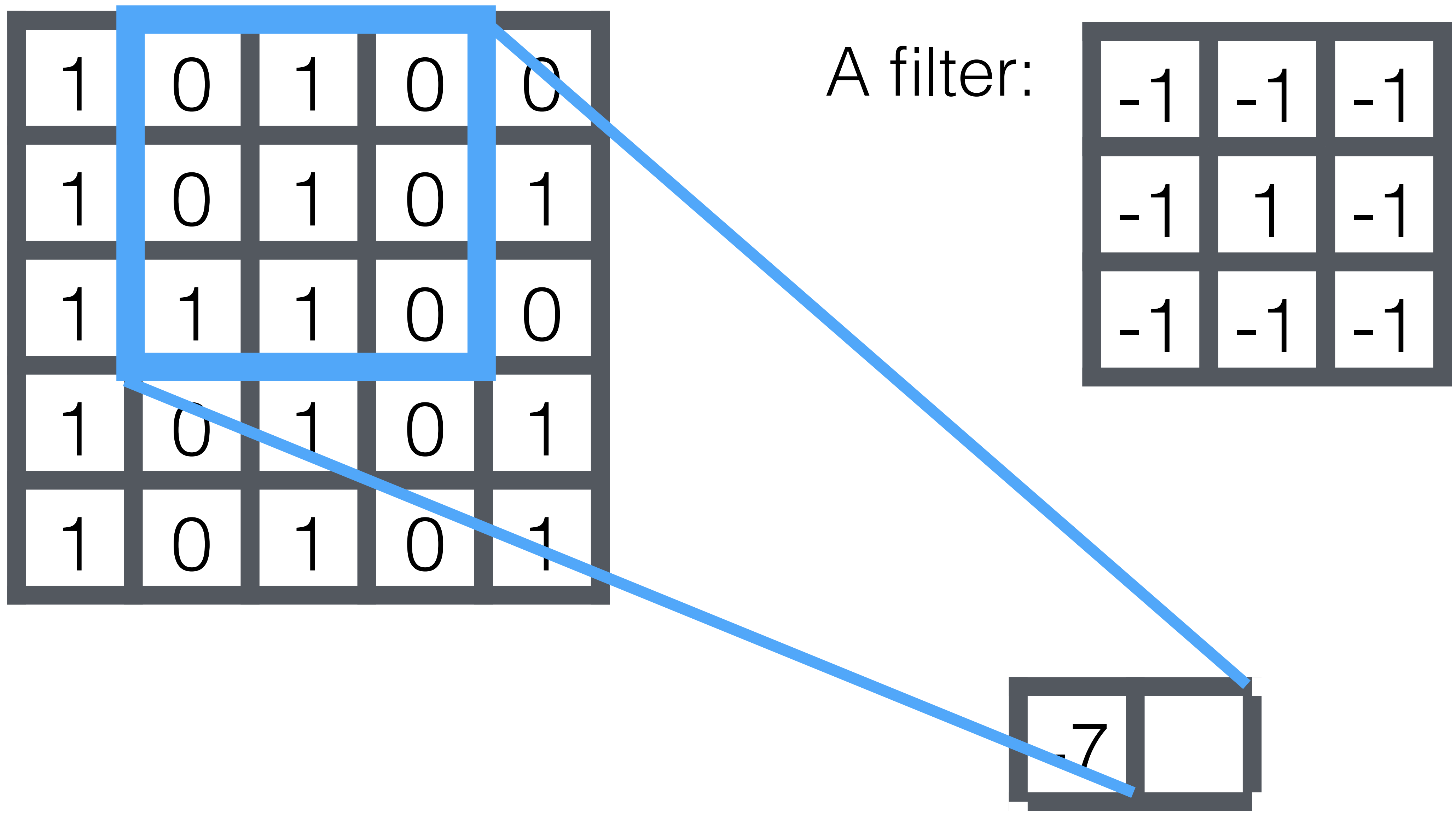
After convolution:

Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

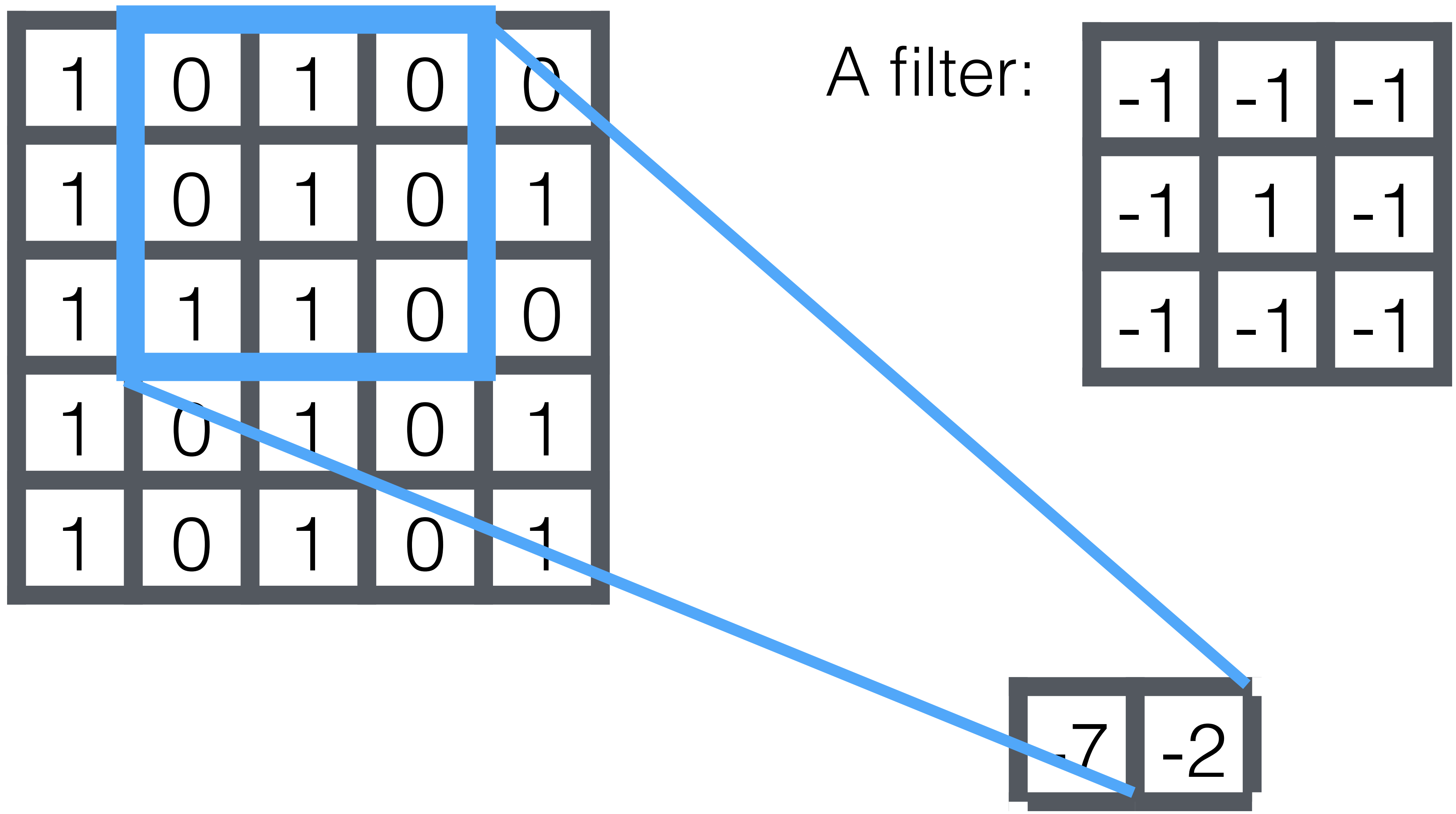


After convolution:

Convolutional Layer: 2D example

A 2D image:

A filter:

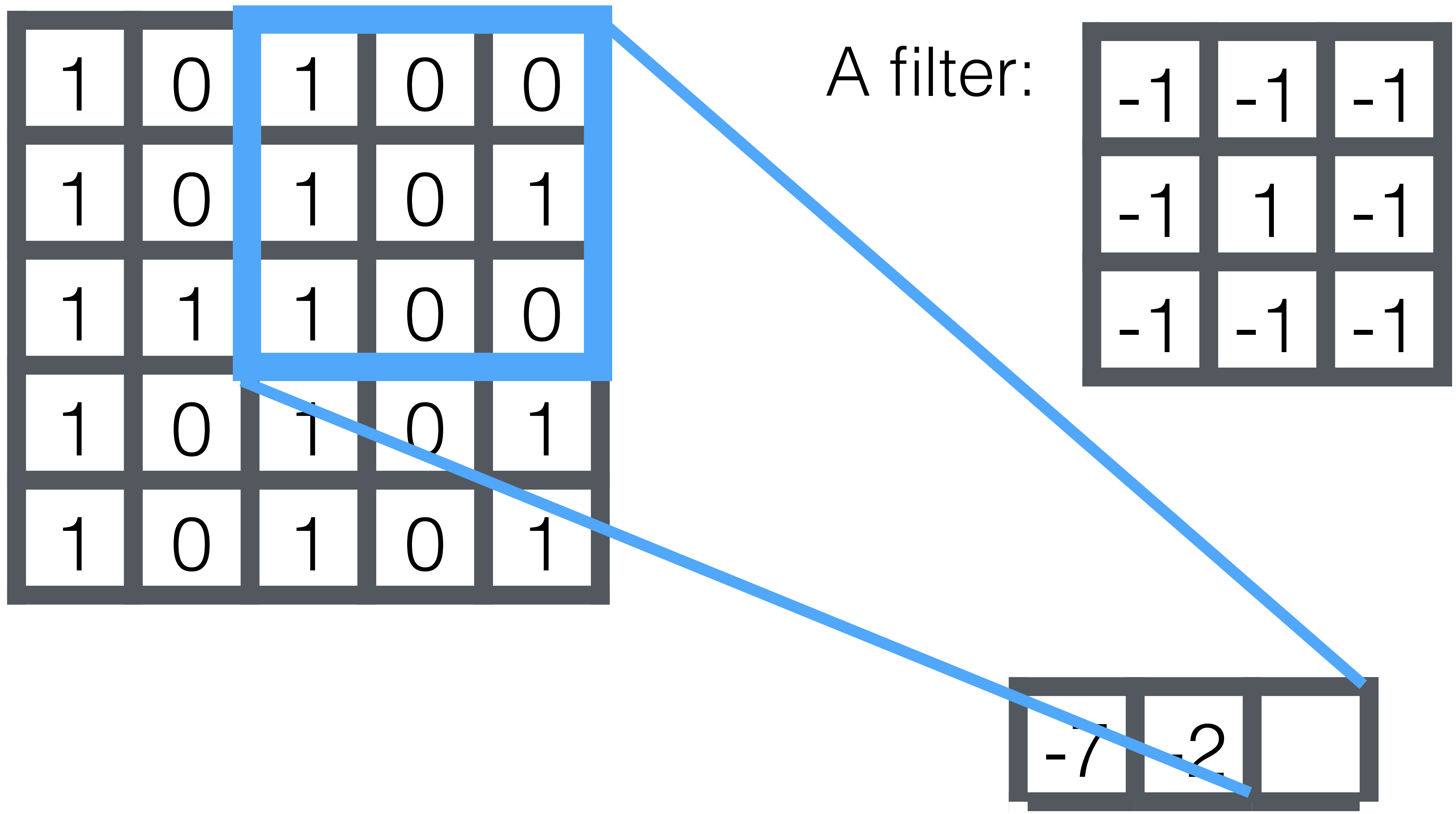


After convolution:

Convolutional Layer: 2D example

A 2D image:

A filter:



After convolution:

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

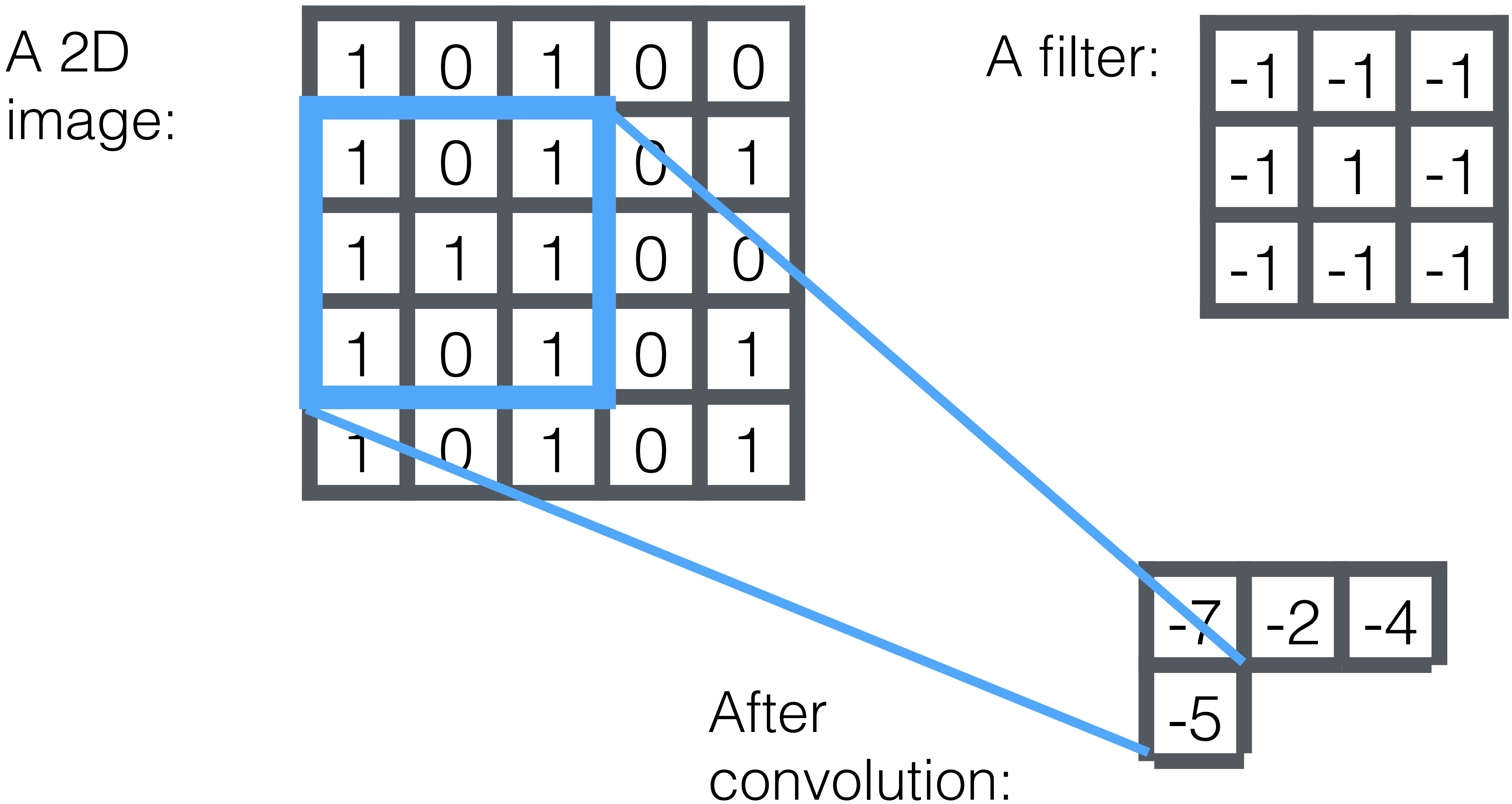
A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

-7	-2	-4

Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

-7	-2	-4
-5	-2	

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

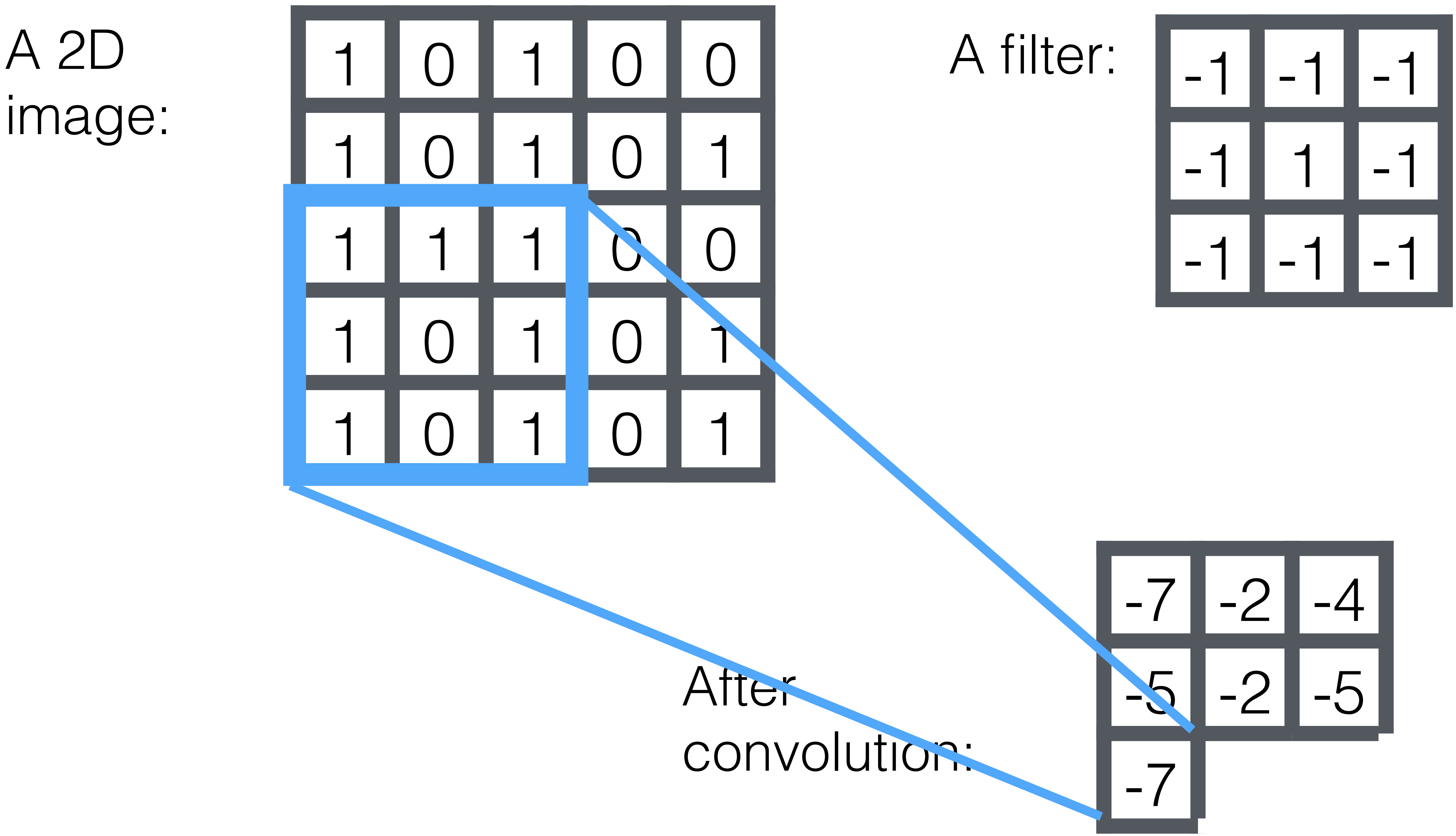
A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

-7	-2	-4
-5	-2	-5

Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

-7	-2	-4
-5	-2	-5
-7	-2	

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

-7	-2	-4
-5	-2	-5
-7	-2	-5

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

-7	-2	-4
-5	-2	-5
-7	-2	-5

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

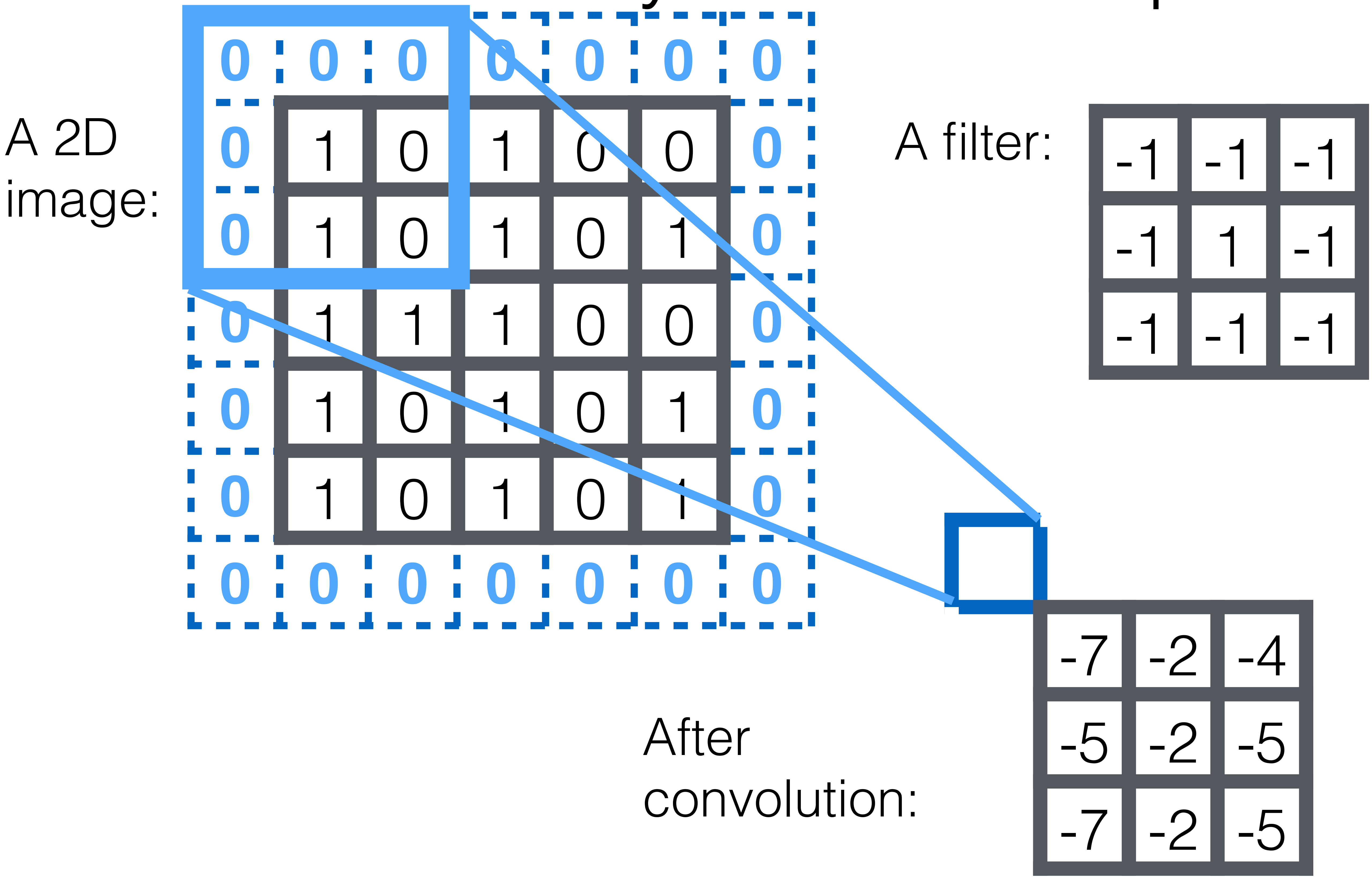
A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

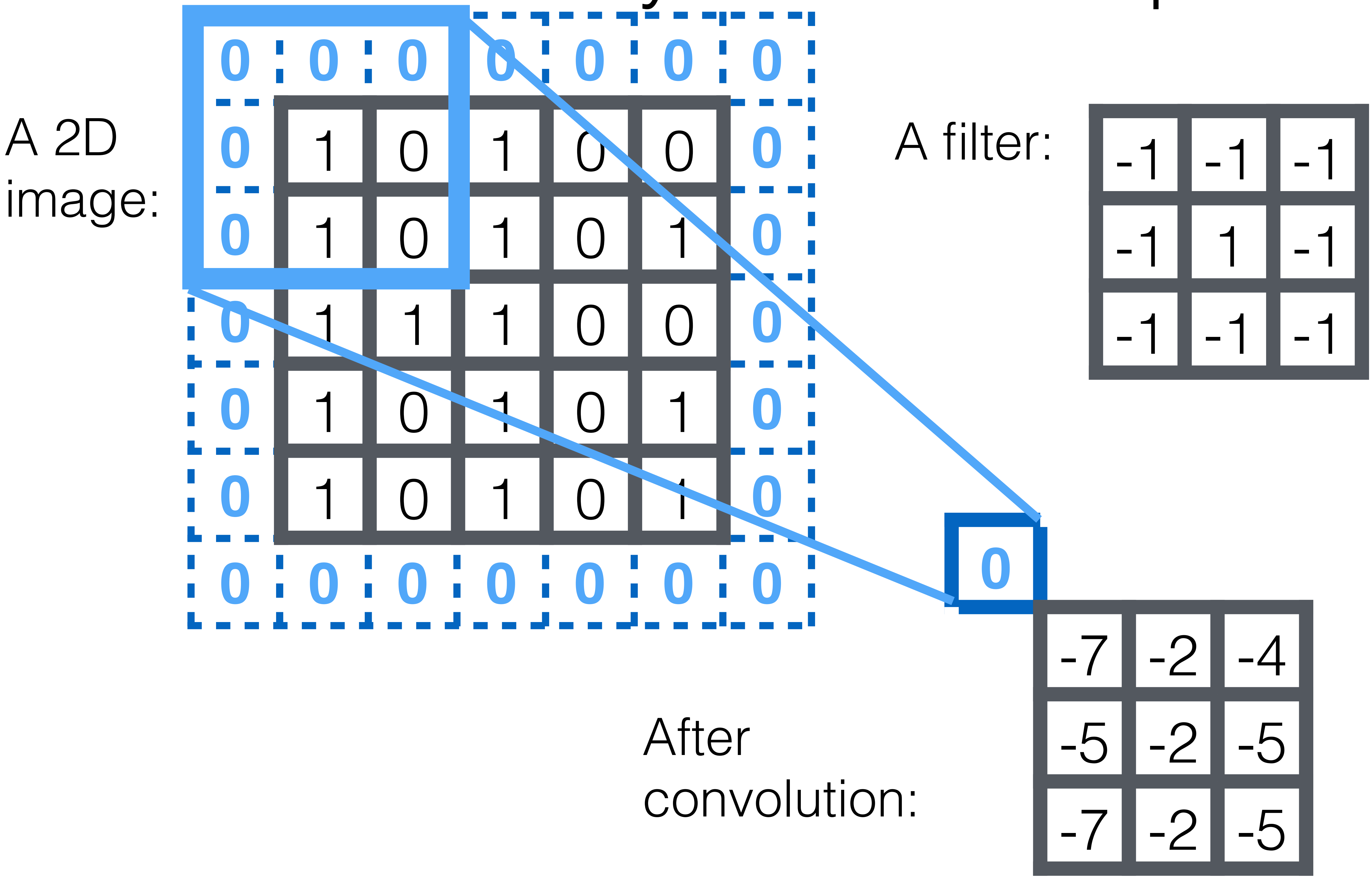
After convolution:

-7	-2	-4
-5	-2	-5
-7	-2	-5

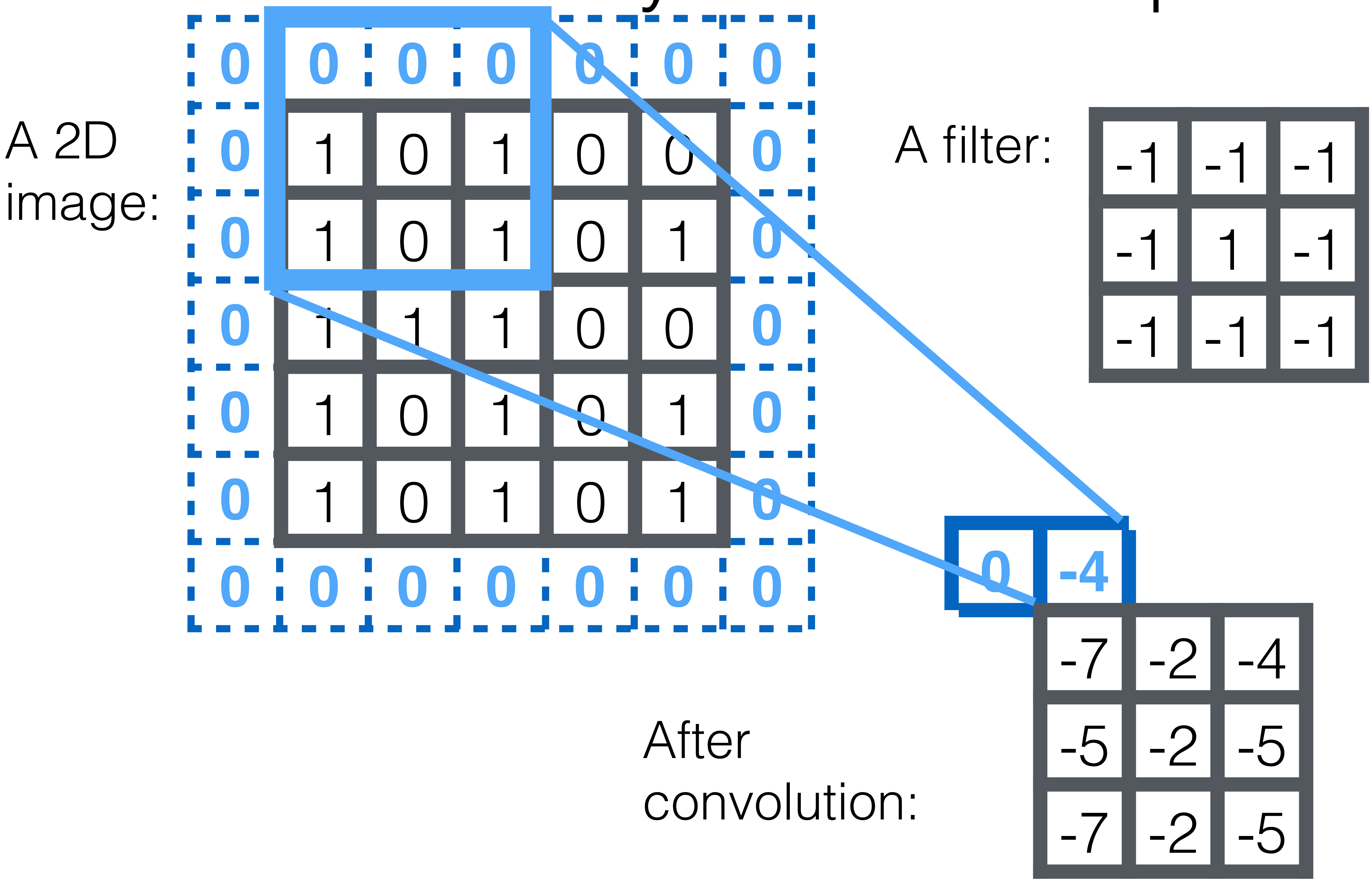
Convolutional Layer: 2D example



Convolutional Layer: 2D example



Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

0	-4		
	-7	-2	-4
	-5	-2	-5
	-7	-2	-5

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

0	-4	0	-3	-1
-2	-7	-2	-4	1
-2	-5	-2	-5	-2
-2	-7	-2	-5	0
0	-4	0	-4	0

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution & ReLU:

0	-4	0	-3	-1
-2	-7	-2	-4	1
-2	-5	-2	-5	-2
-2	-7	-2	-5	0
0	-4	0	-4	0

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution & ReLU:

0	-4	0	-3	-1
-2	-7	-2	-4	1
-2	-5	-2	-5	-2
-2	-7	-2	-5	0
0	-4	0	-4	0

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution & ReLU:

0	-4	0	-3	-1
-2	-7	-2	-4	1
-2	-5	-2	-5	-2
-2	-7	-2	-5	0
0	-4	0	-4	0

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution & ReLU:

0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution & ReLU:

0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Convolutional Layer: 2D example

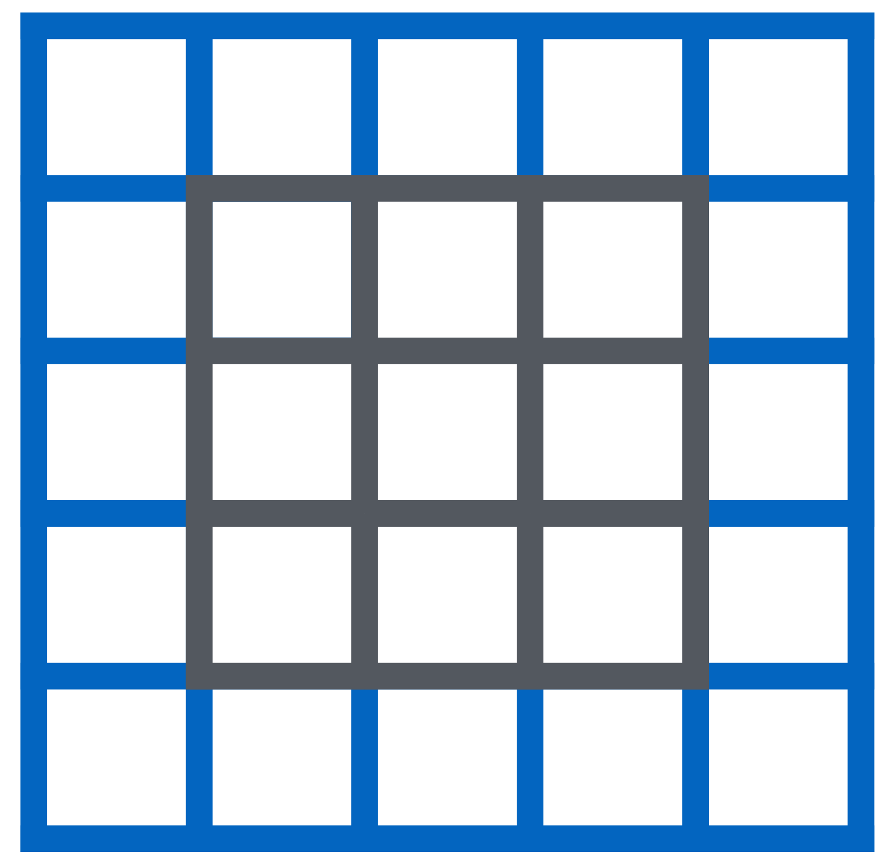
A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:



Convolutional Layer: 2D example

A 2D image:

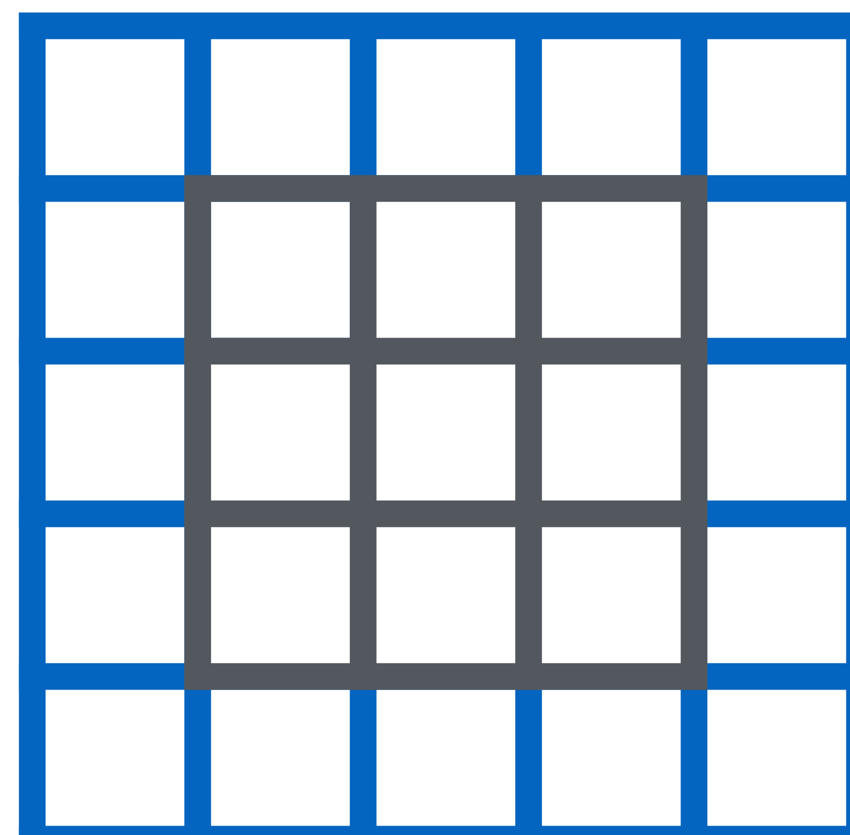
0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

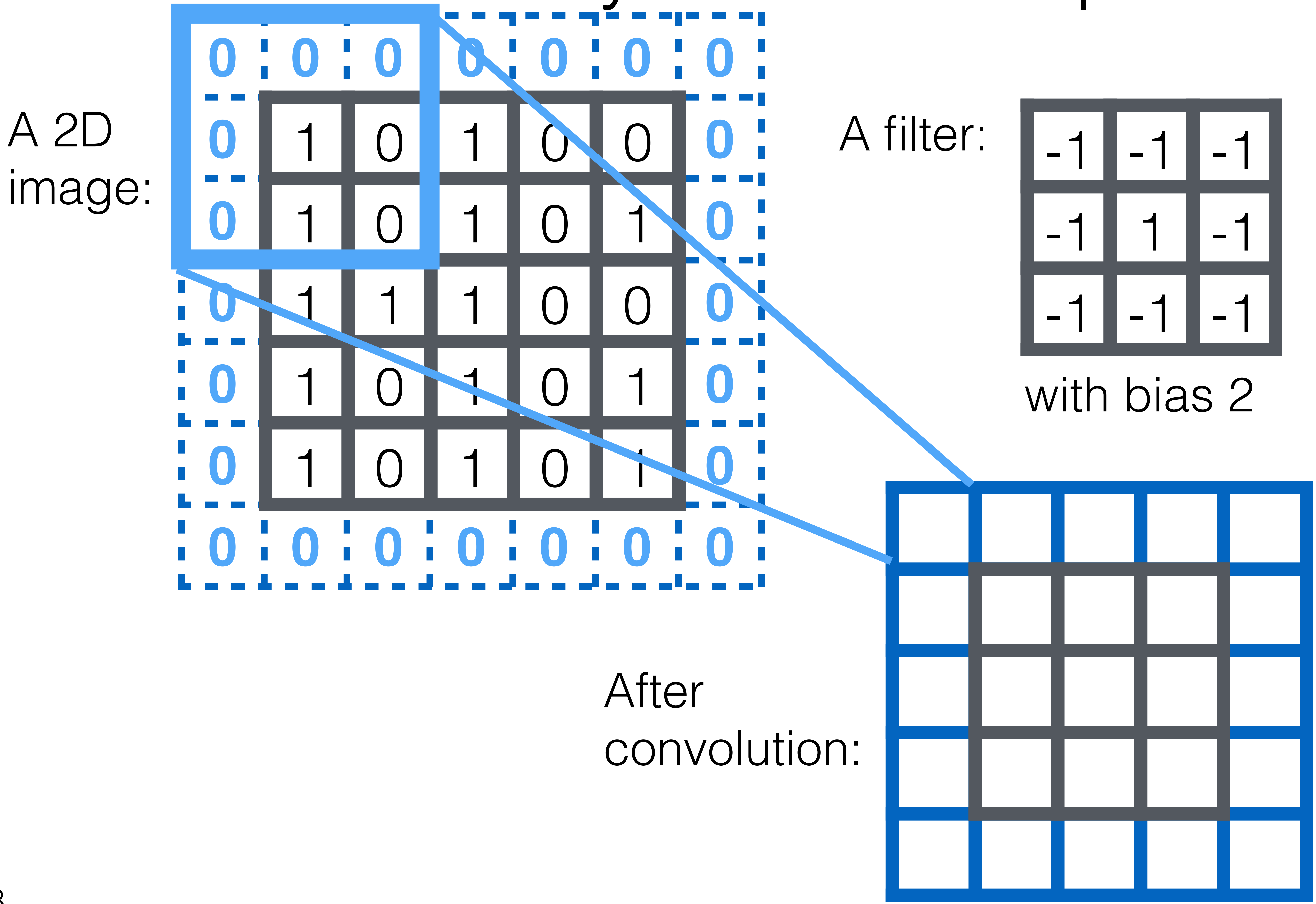
-1	-1	-1
-1	1	-1
-1	-1	-1

with bias 2

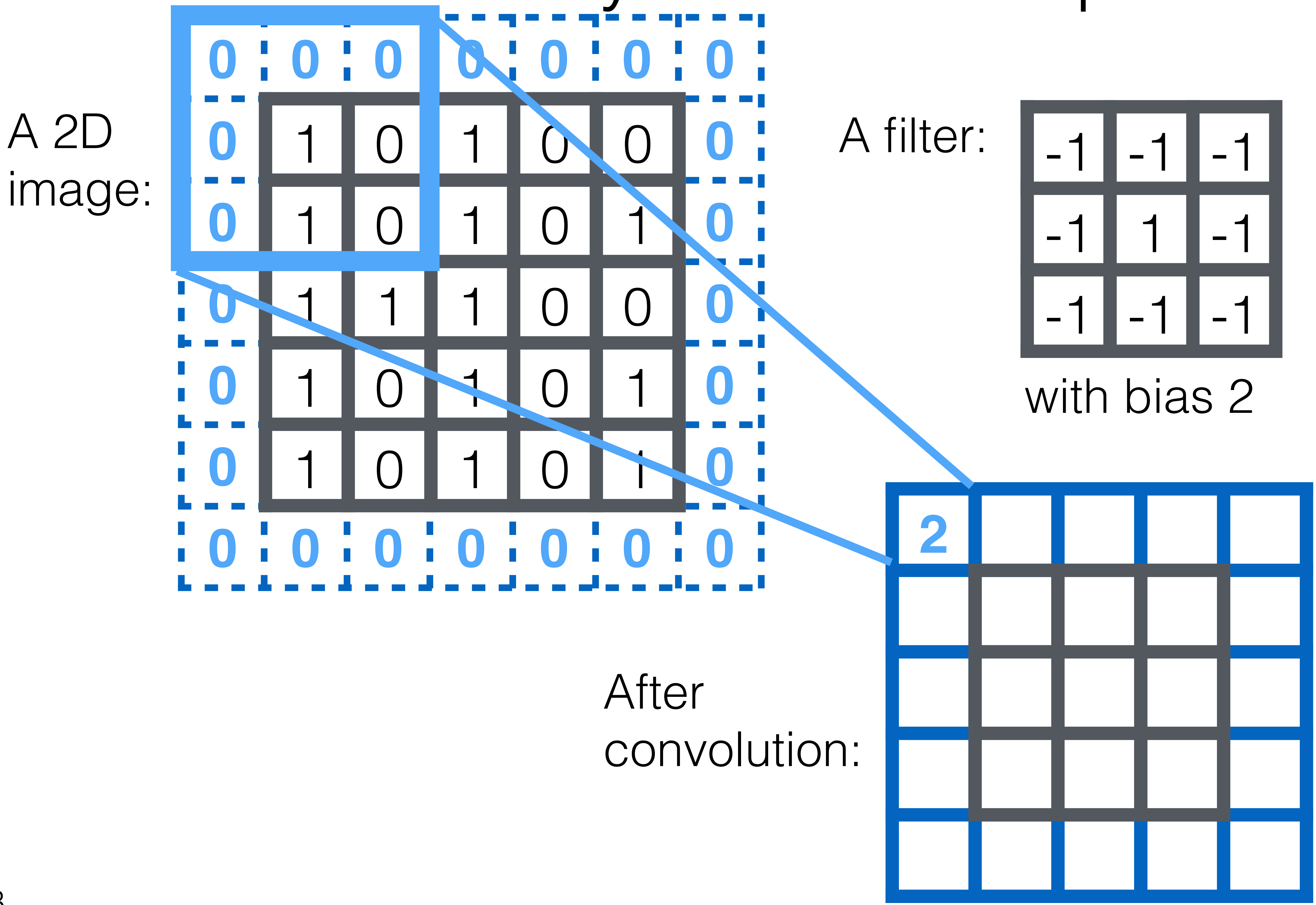
After convolution:



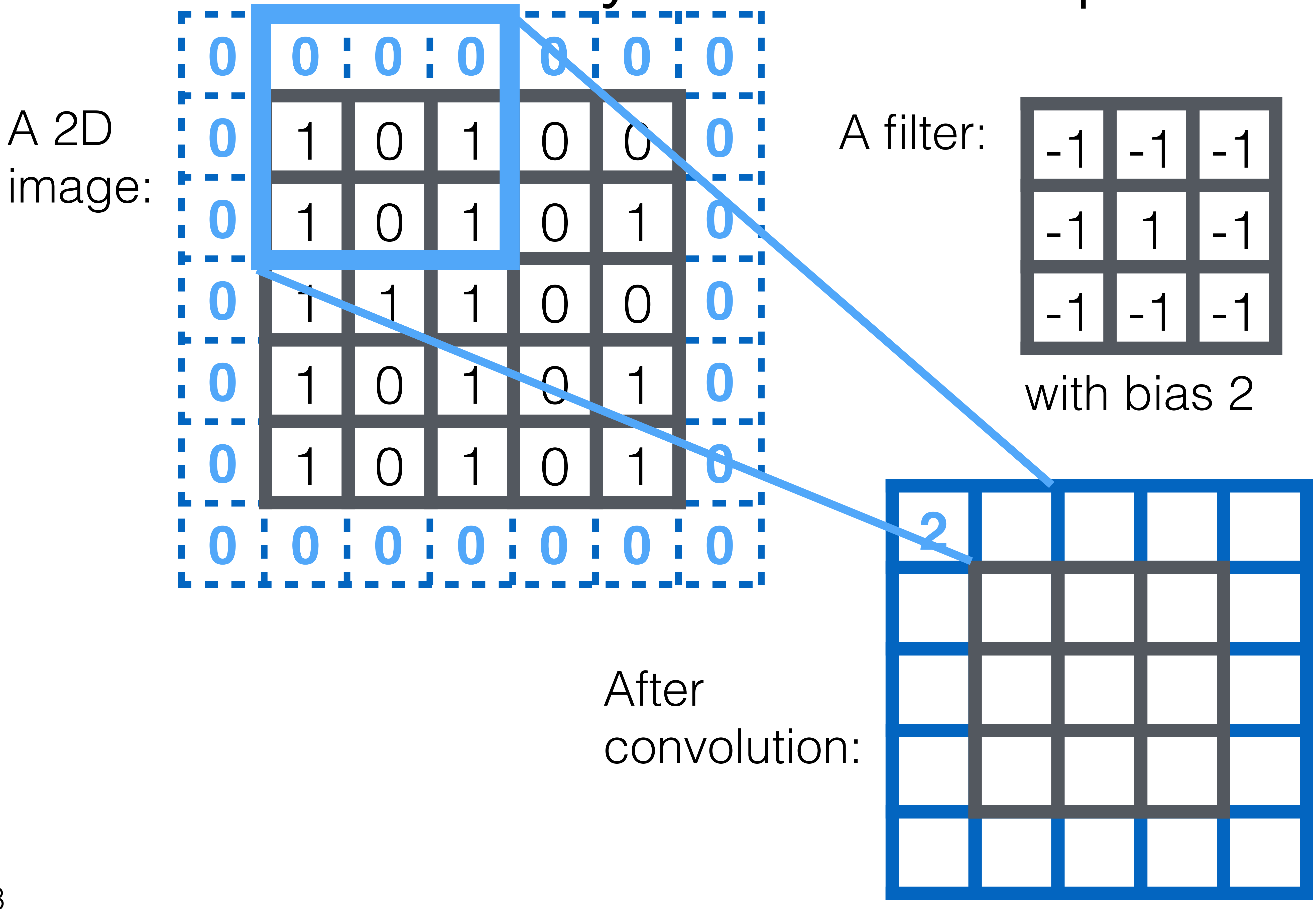
Convolutional Layer: 2D example



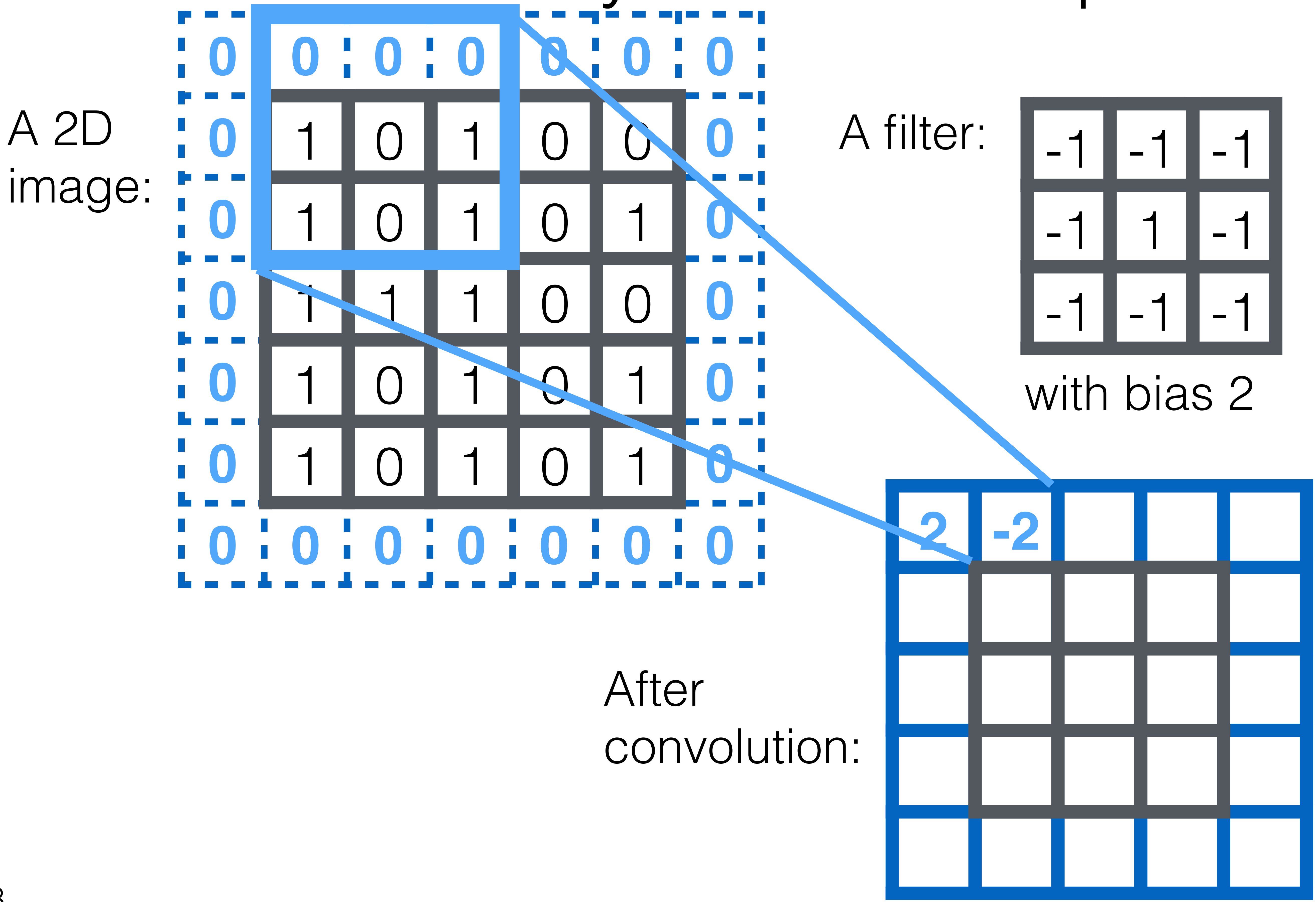
Convolutional Layer: 2D example



Convolutional Layer: 2D example



Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

with bias 2

After convolution:

2	-2			

Convolutional Layer: 2D example

A 2D image:

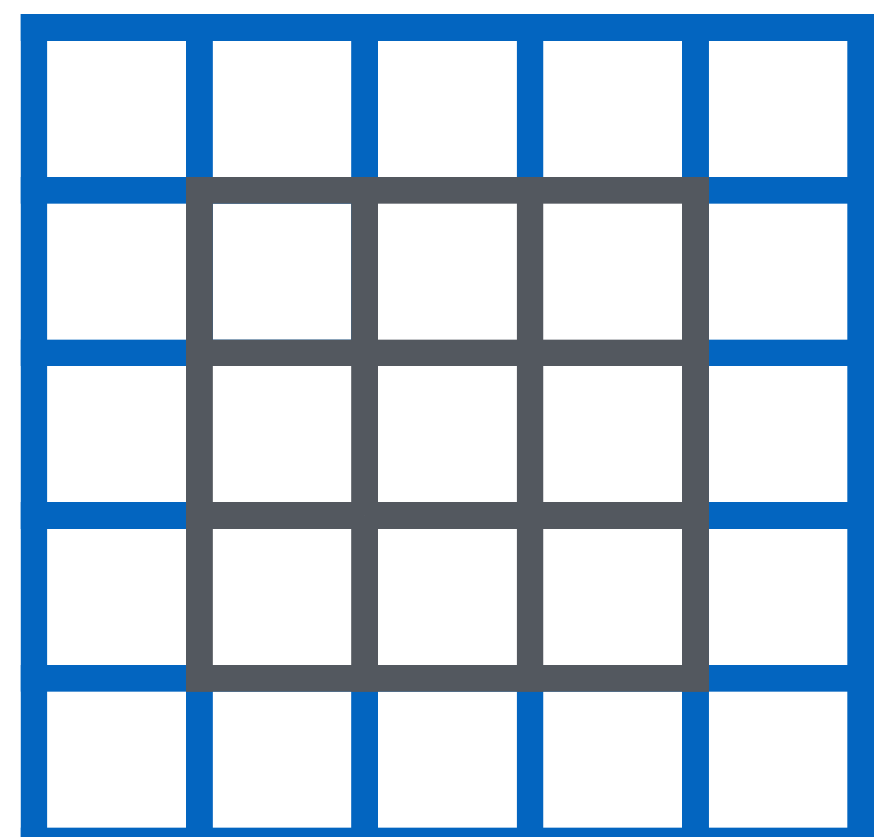
0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

-1	-1	-1
-1	1	-1
-1	-1	-1

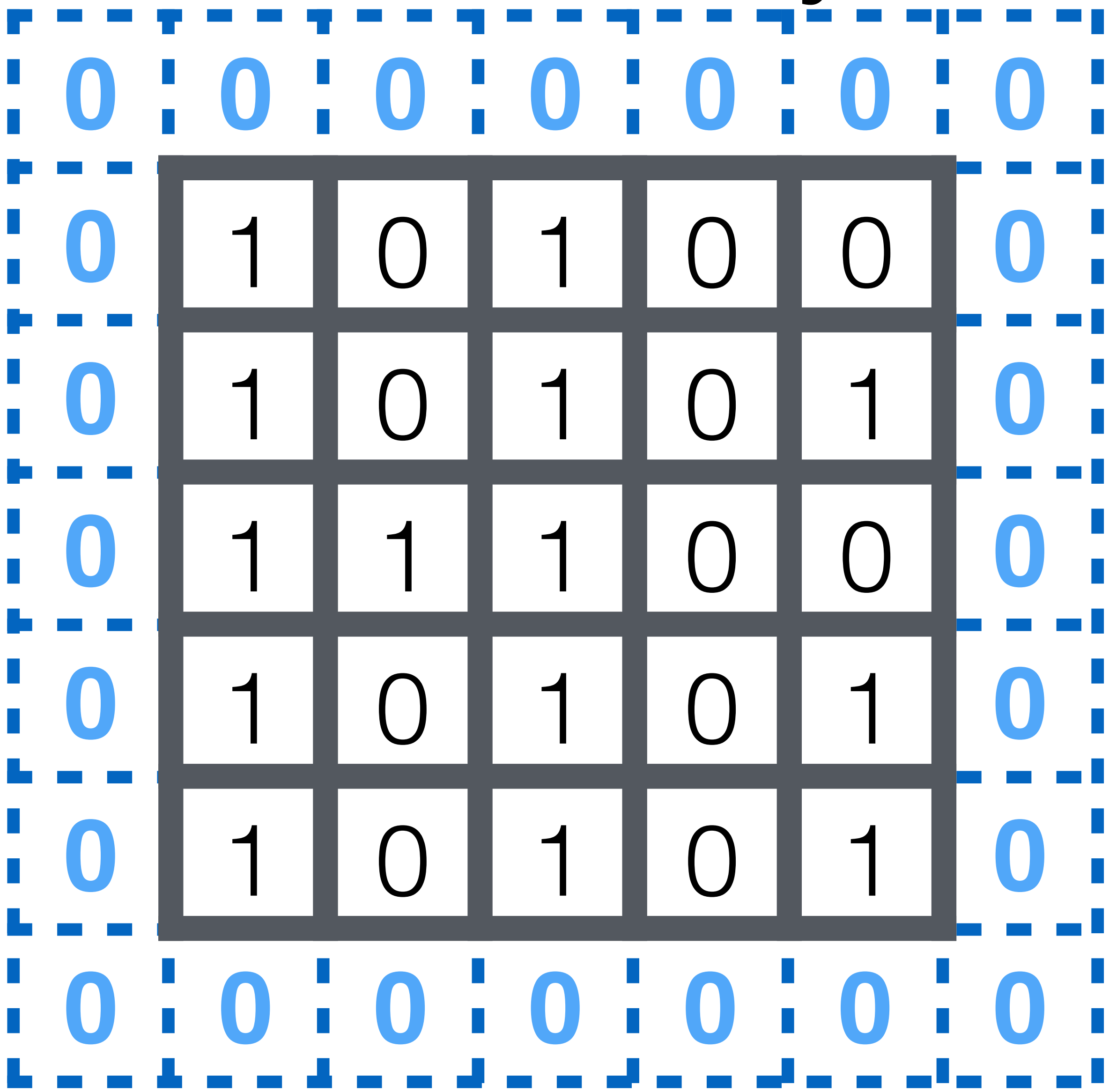
with bias 2

After convolution:

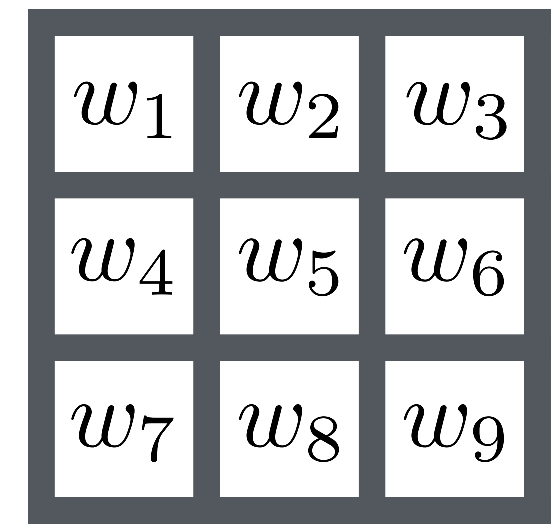


Convolutional Layer: 2D example

A 2D image:

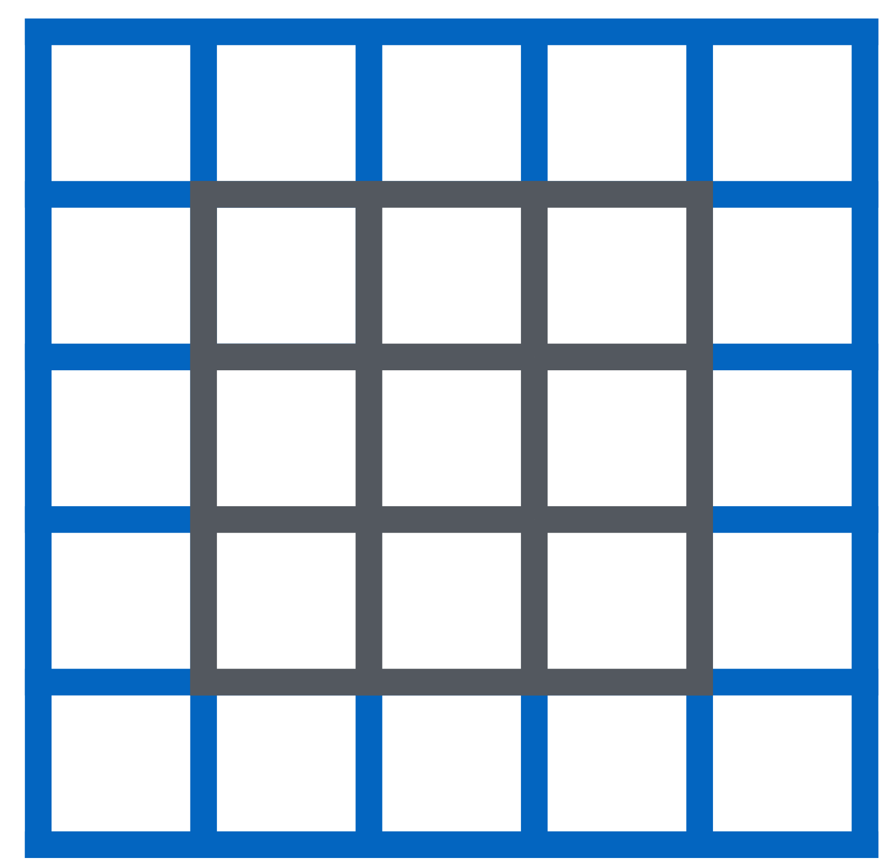


A filter:

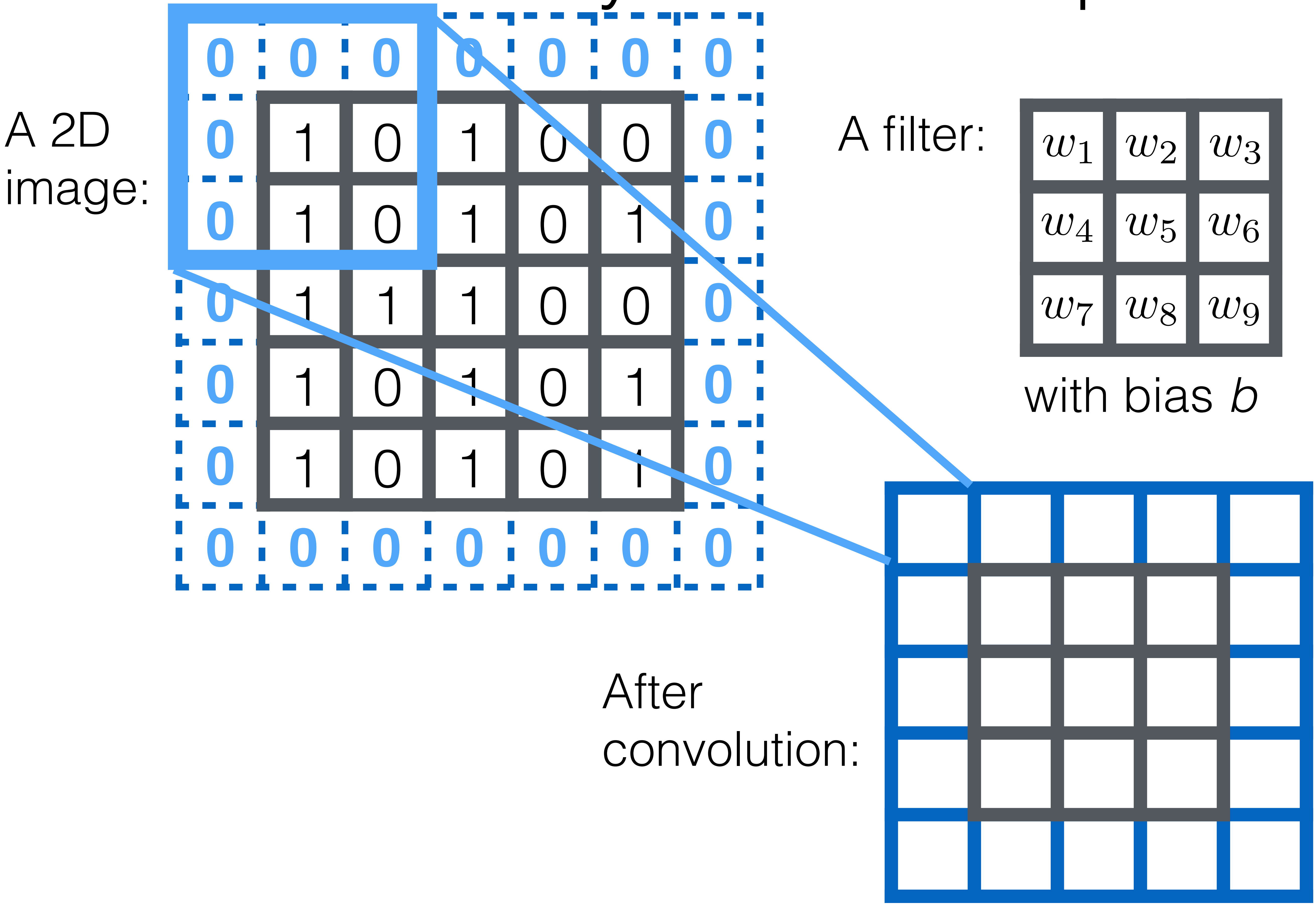


with bias *b*

After convolution:



Convolutional Layer: 2D example



Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

with bias *b*

Convolutional Layer: 2D example

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

w_1	w_2
w_3	w_4

with bias *b*

Convolutional Layer: 3D example

A 3D image:

[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

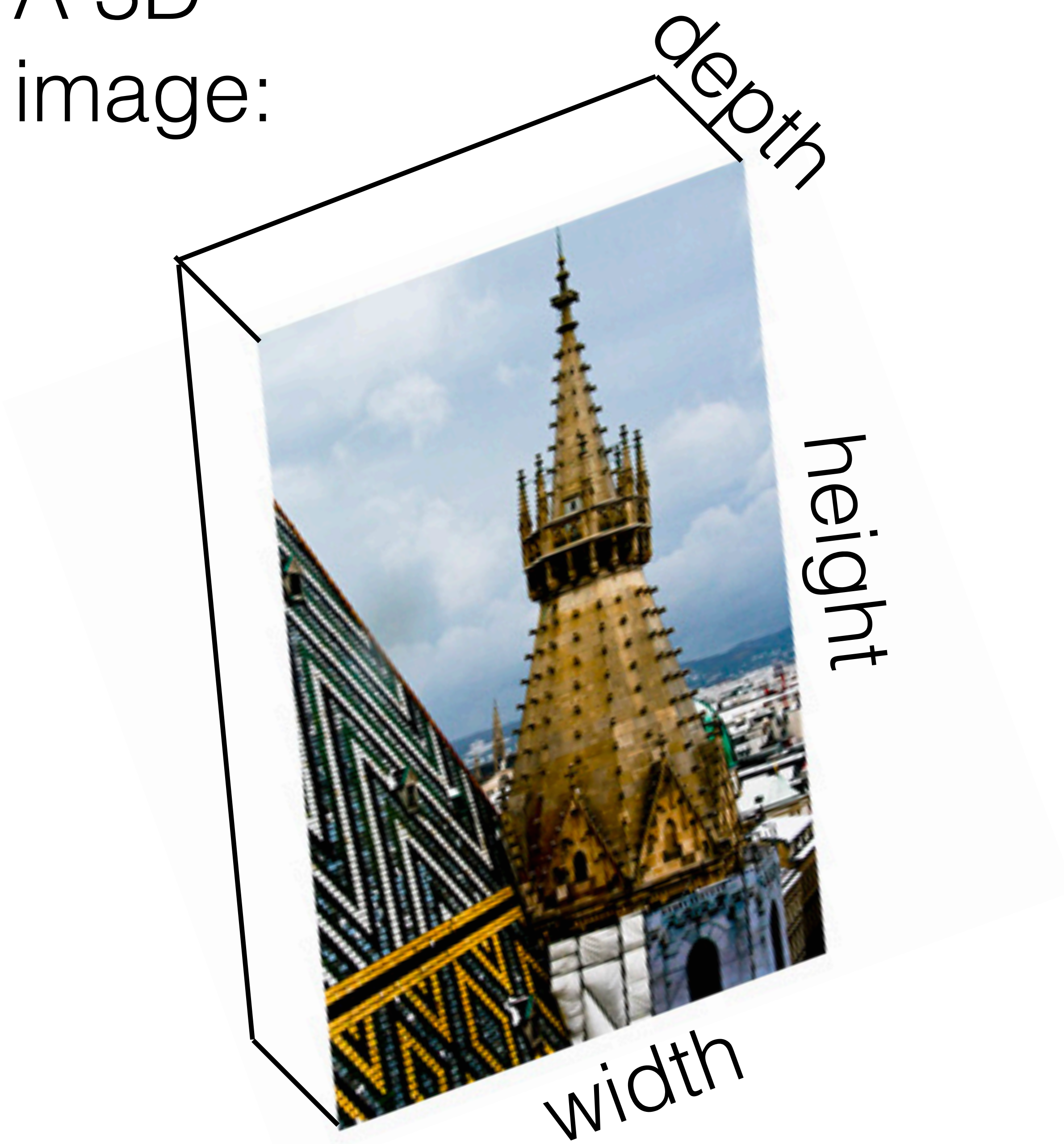
Convolutional Layer: 3D example

A 3D image:

[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

Convolutional Layer: 3D example

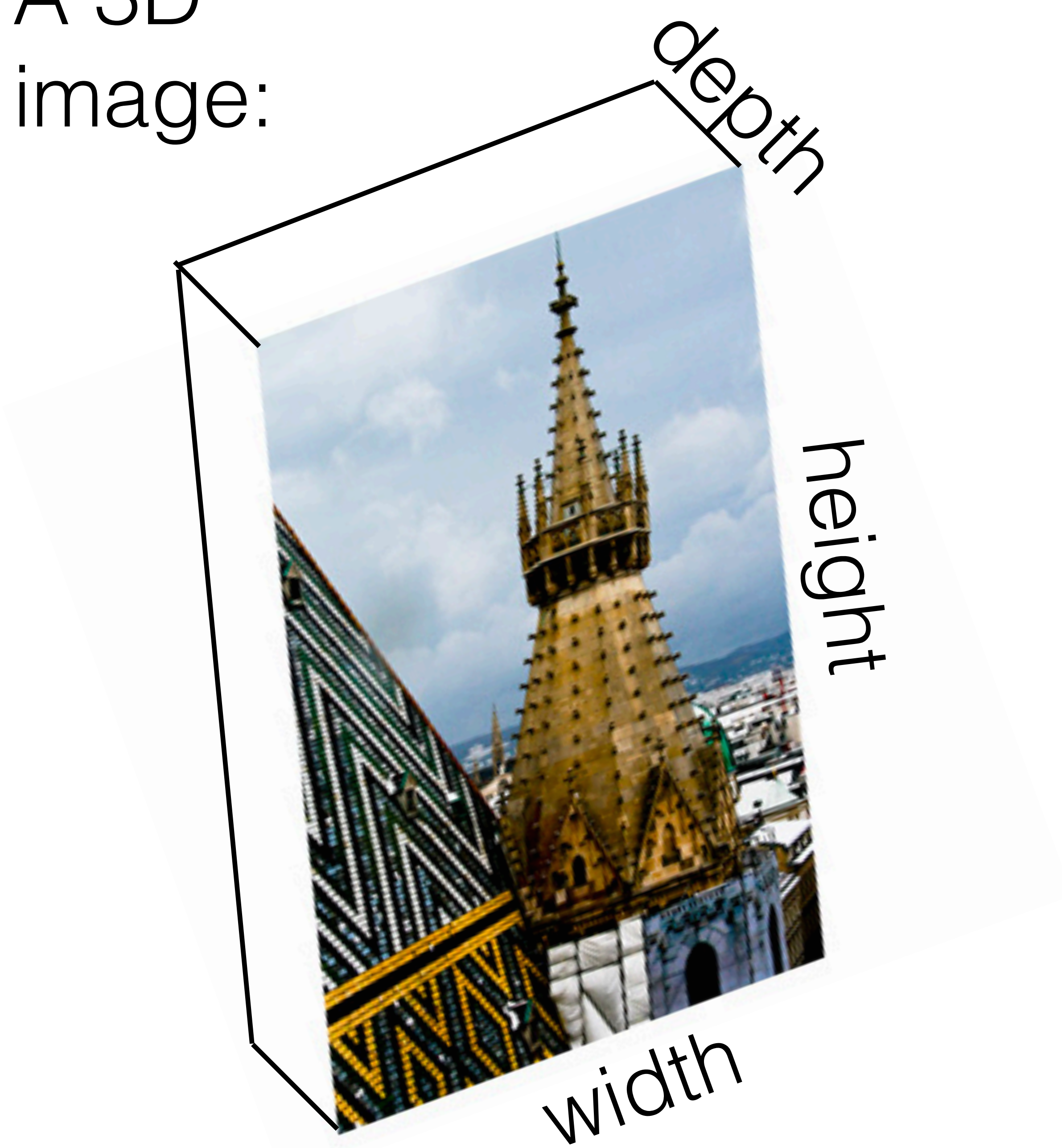
A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

Convolutional Layer: 3D example

A 3D image:

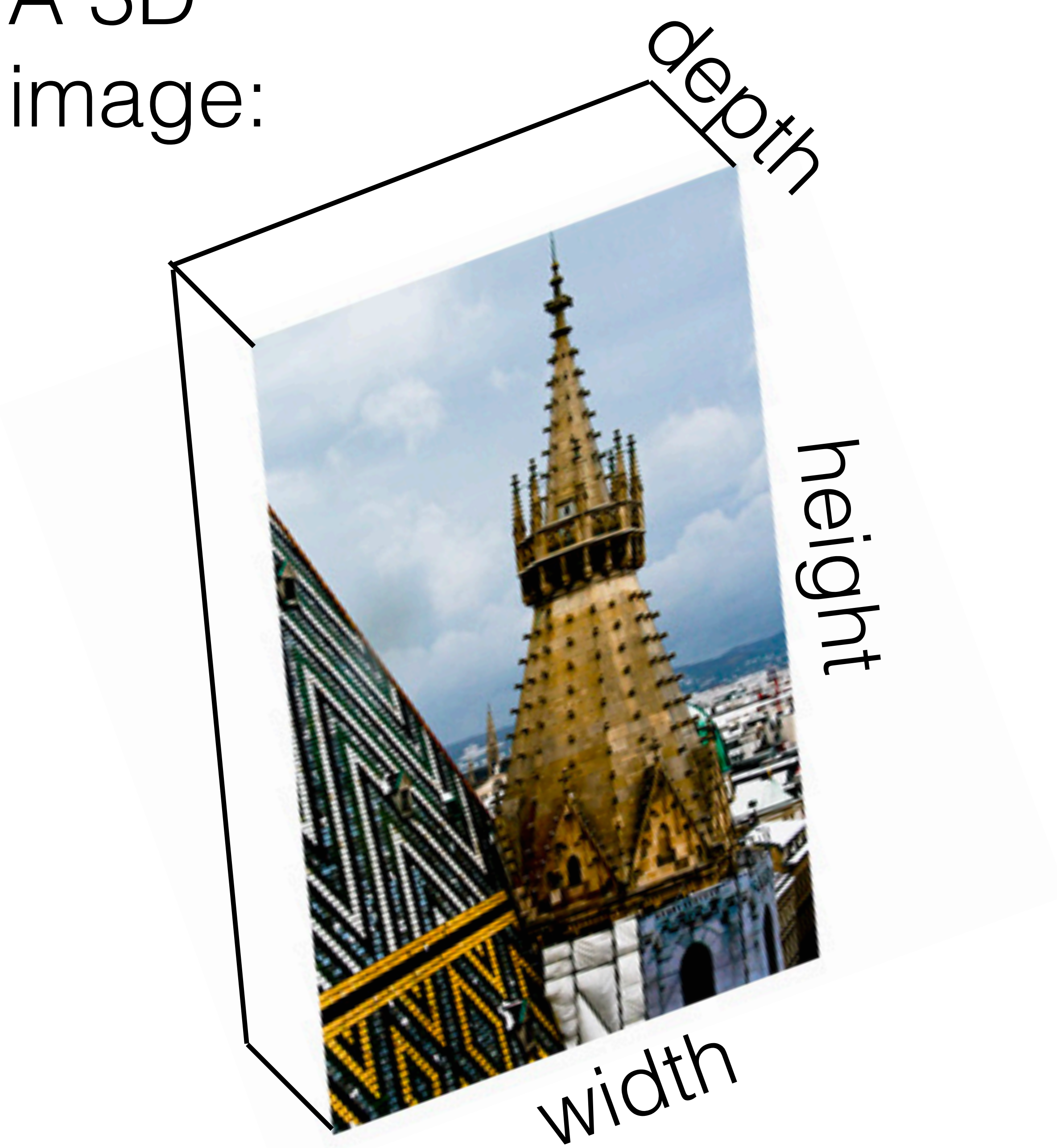


[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

Convolutional Layer: 3D example

A 3D image:



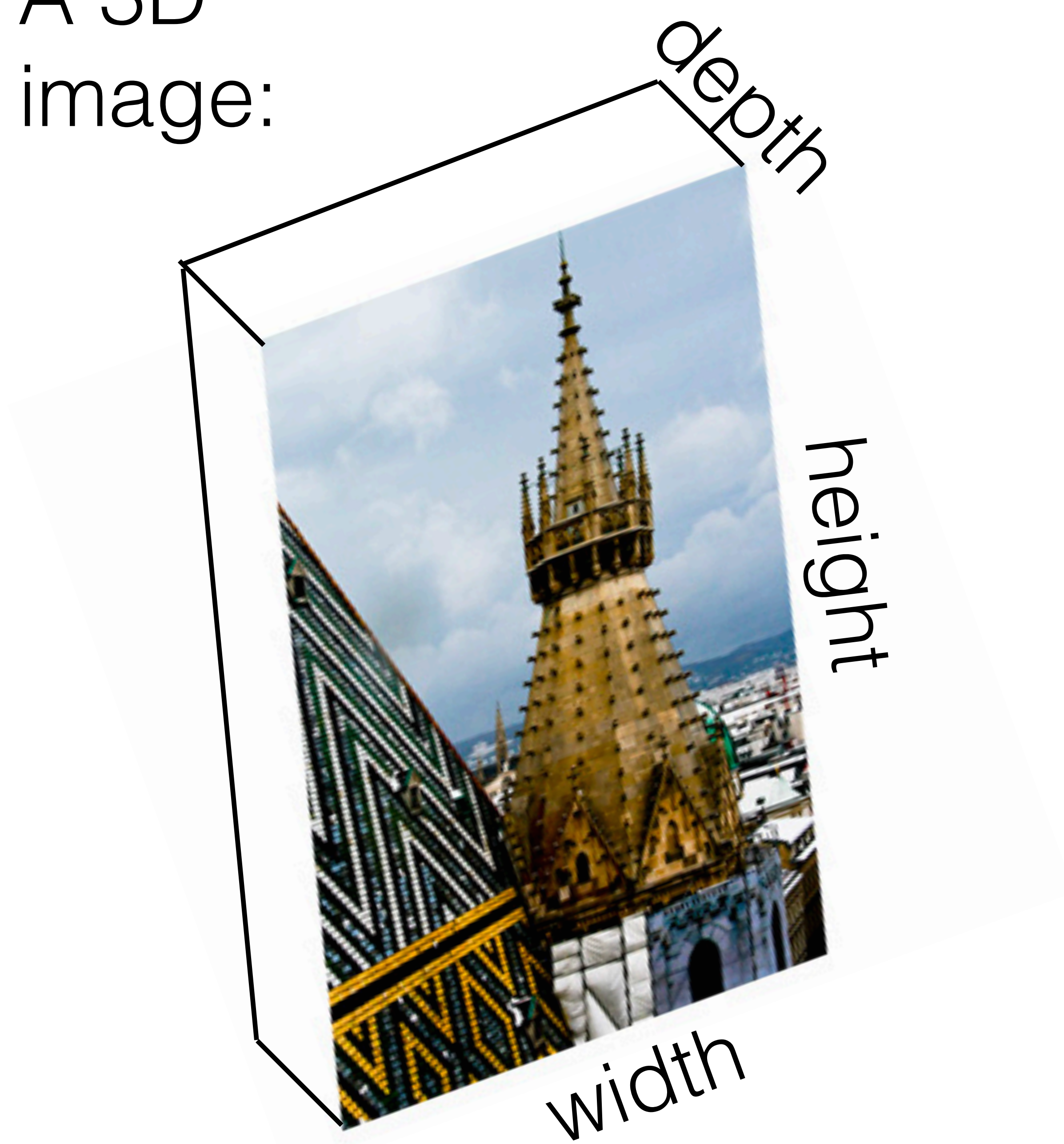
[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

[https://en.wikipedia.org/wiki/TensorFlow]

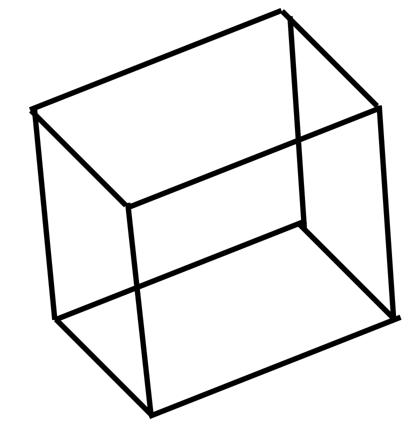
Convolutional Layer: 3D example

A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

A filter:



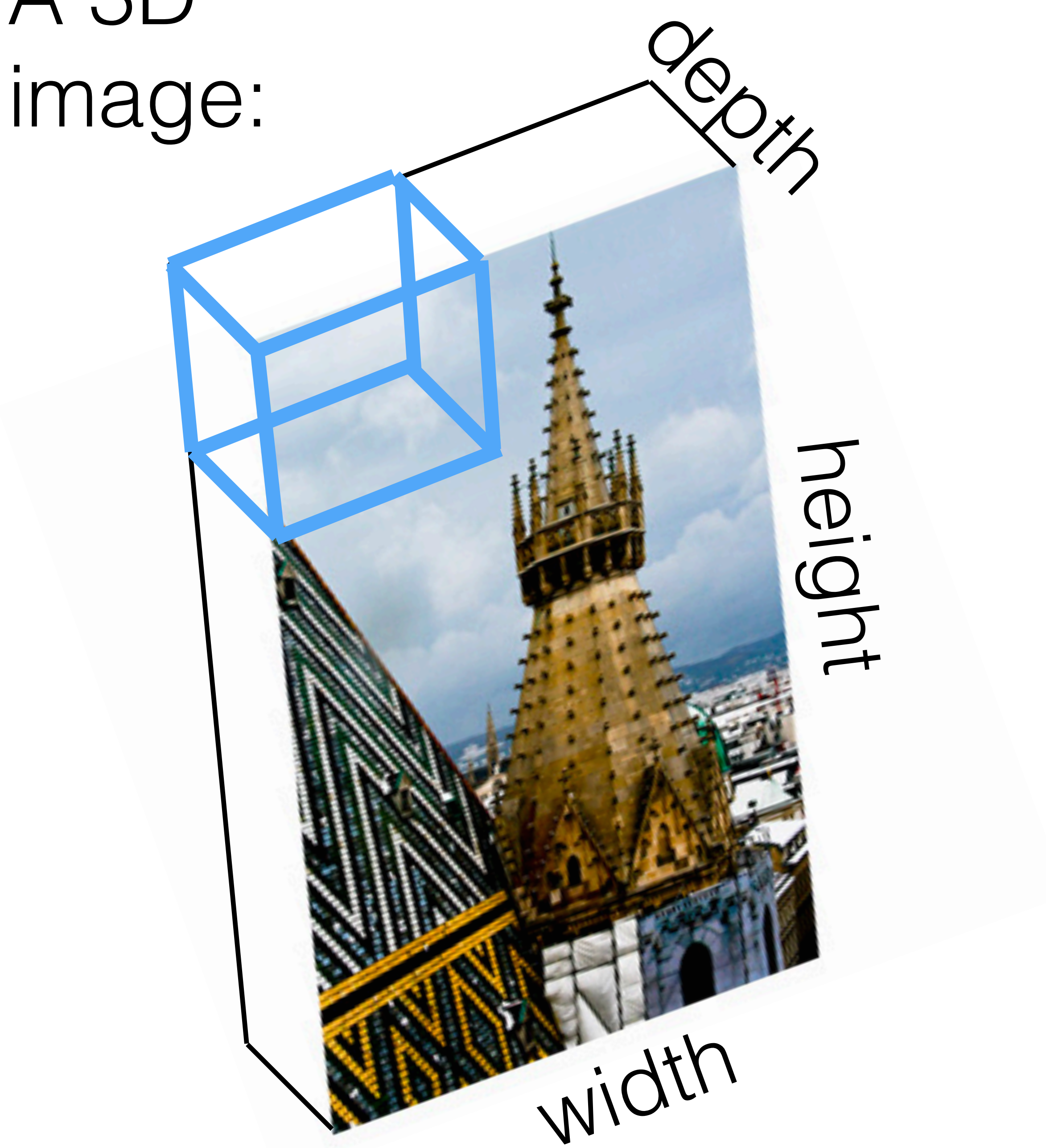
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

TensorFlow

[https://en.wikipedia.org/wiki/TensorFlow]

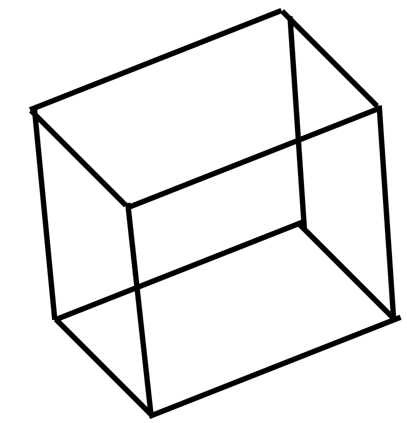
Convolutional Layer: 3D example

A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

A filter:



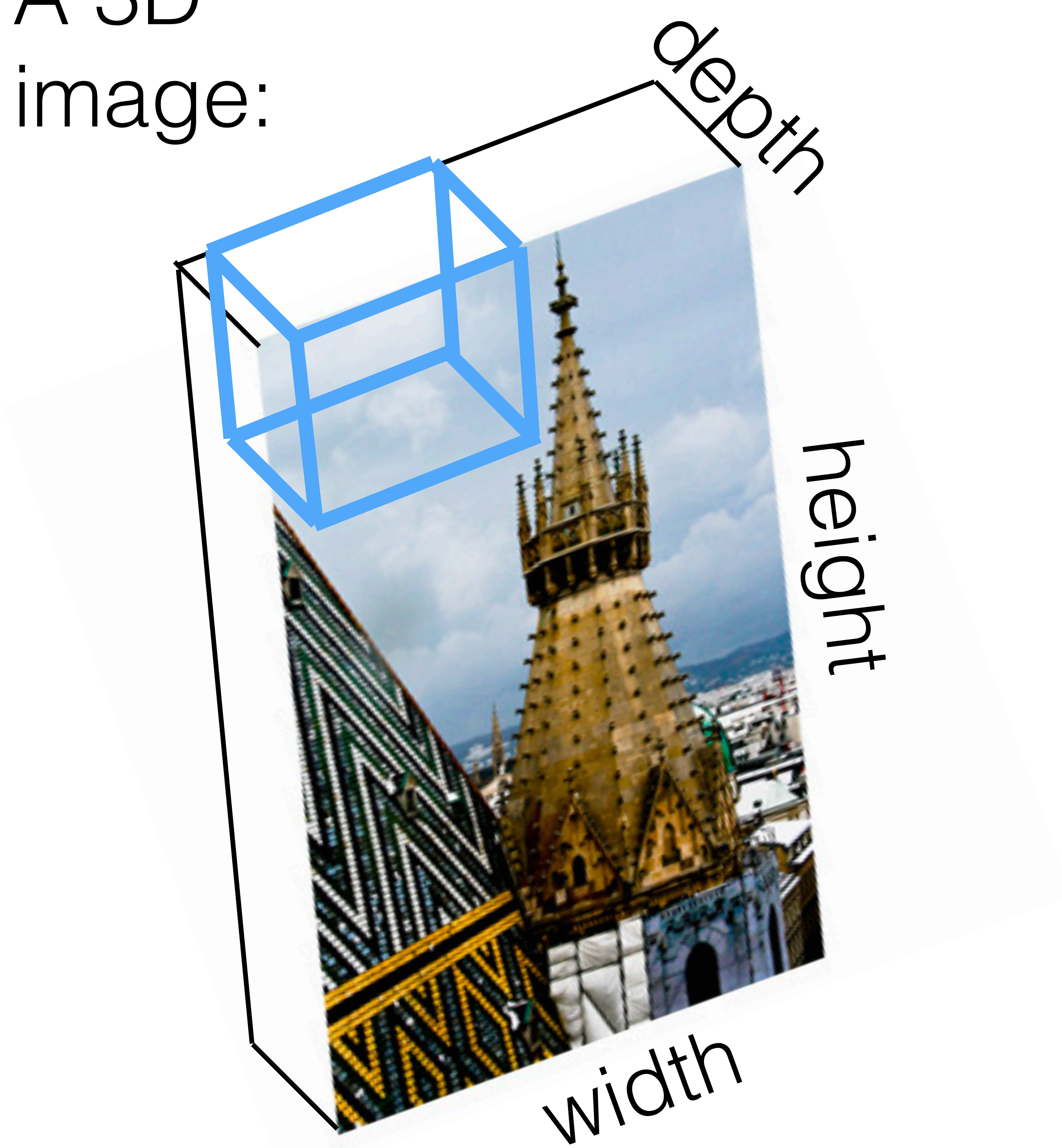
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

TensorFlow

[https://en.wikipedia.org/wiki/TensorFlow]

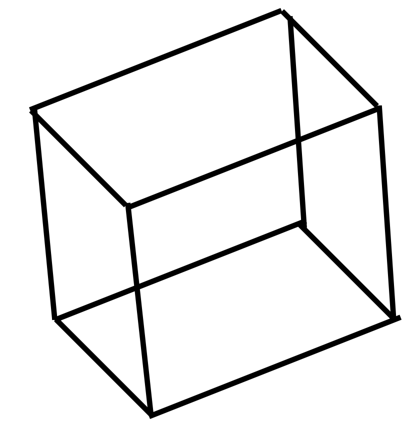
Convolutional Layer: 3D example

A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

A filter:

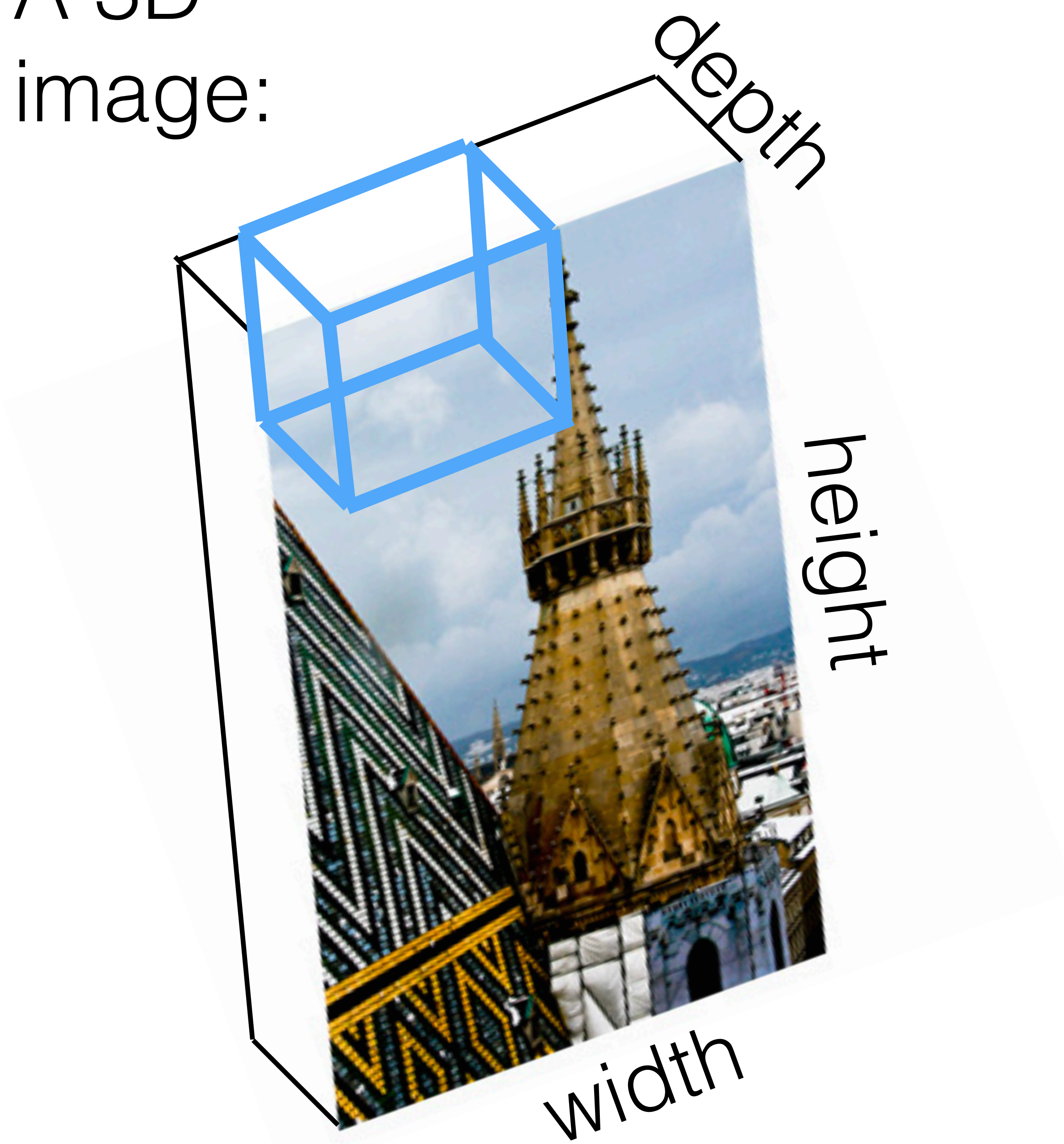


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

[https://en.wikipedia.org/wiki/TensorFlow]

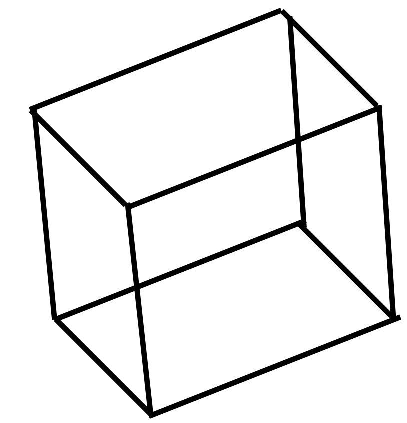
Convolutional Layer: 3D example

A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

A filter:



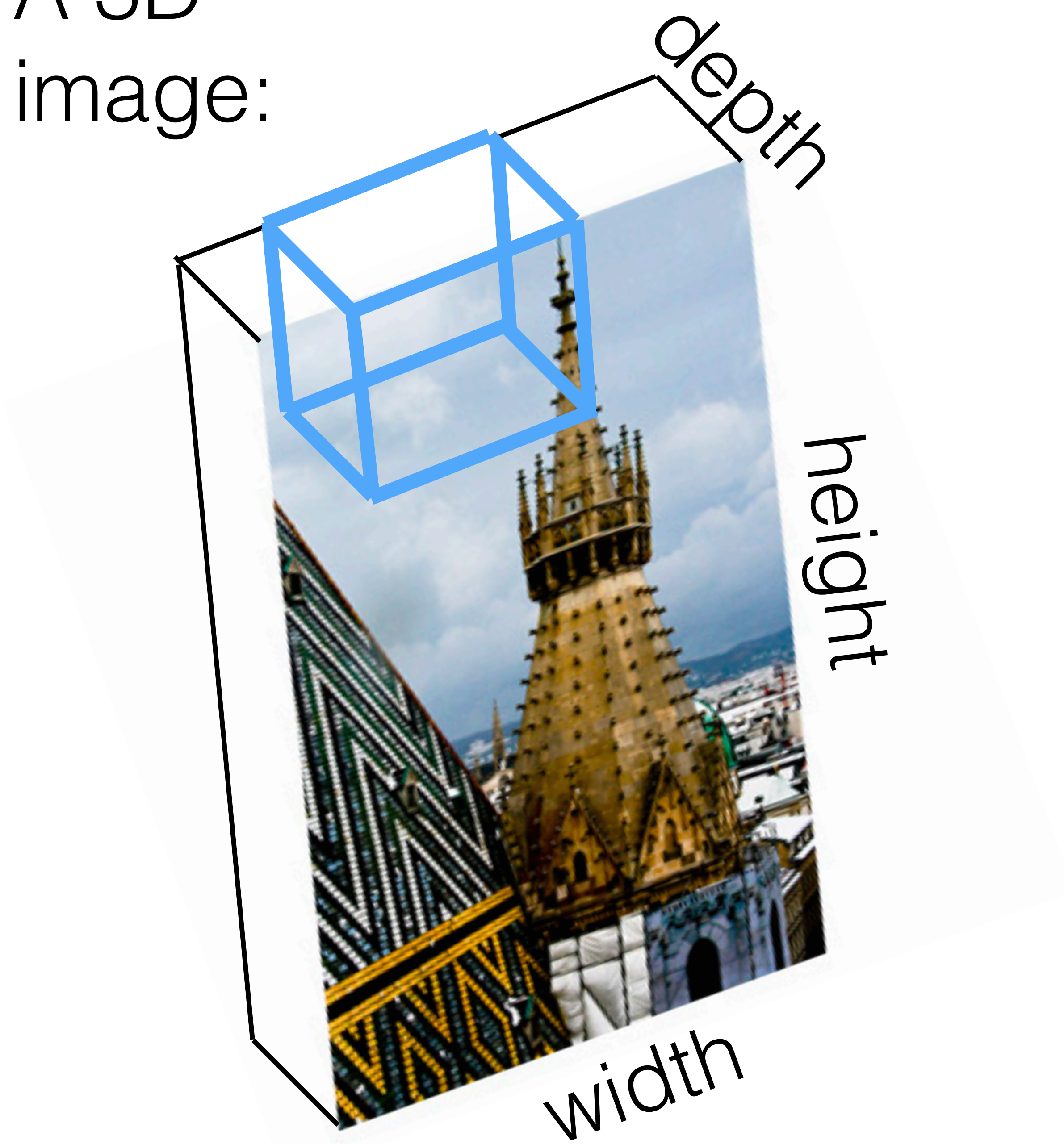
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

[https://en.wikipedia.org/wiki/TensorFlow]

TensorFlow

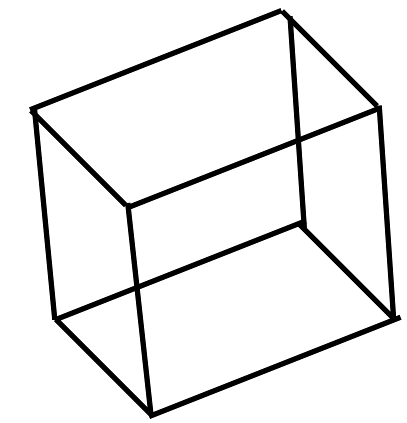
Convolutional Layer: 3D example

A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

A filter:

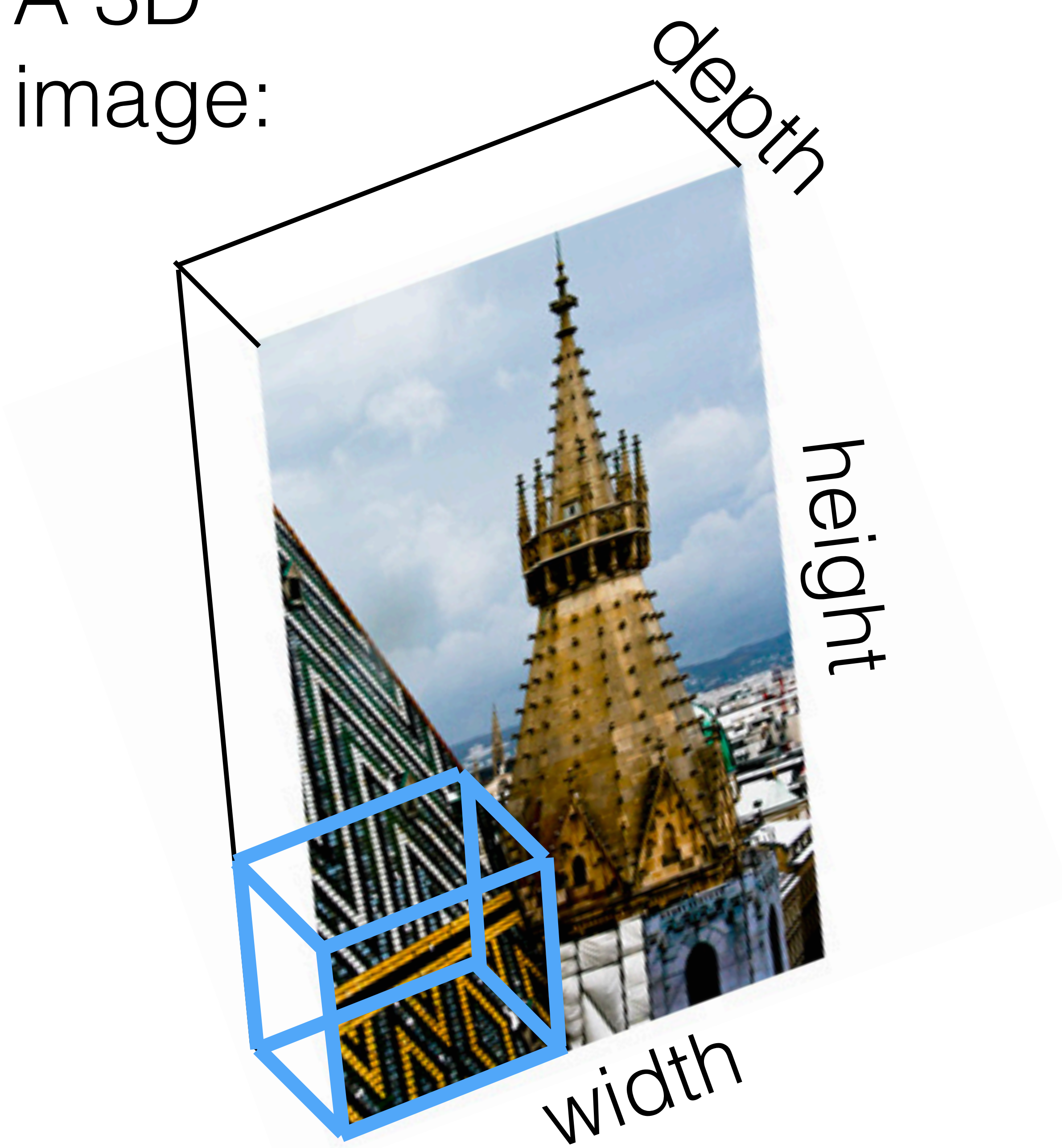


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

[https://en.wikipedia.org/wiki/TensorFlow]

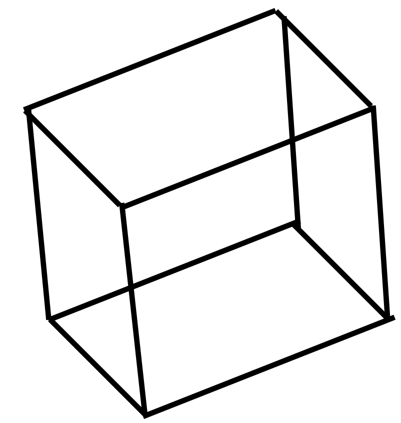
Convolutional Layer: 3D example

A 3D image:



[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

A filter:



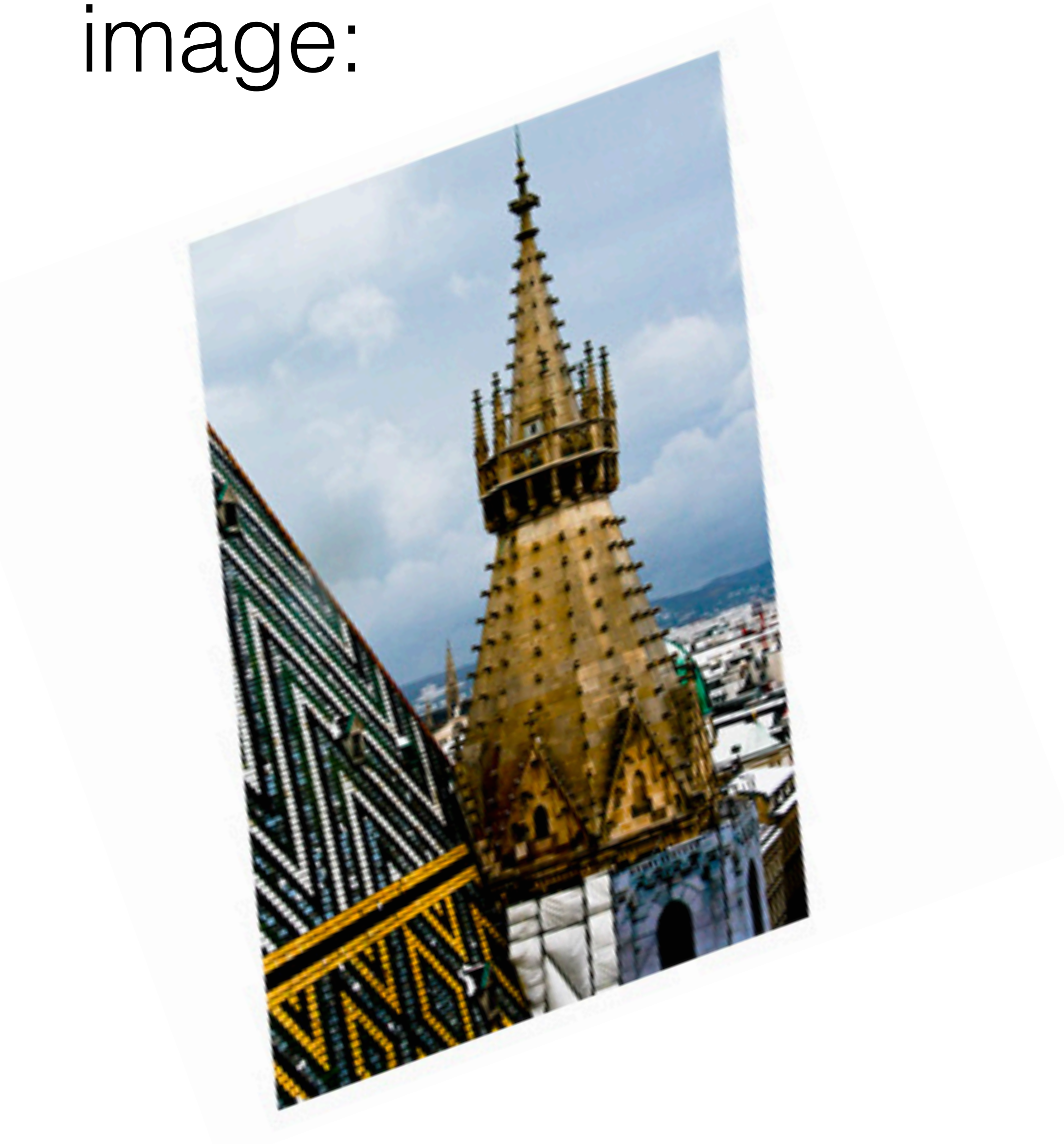
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

TensorFlow

[https://en.wikipedia.org/wiki/TensorFlow]

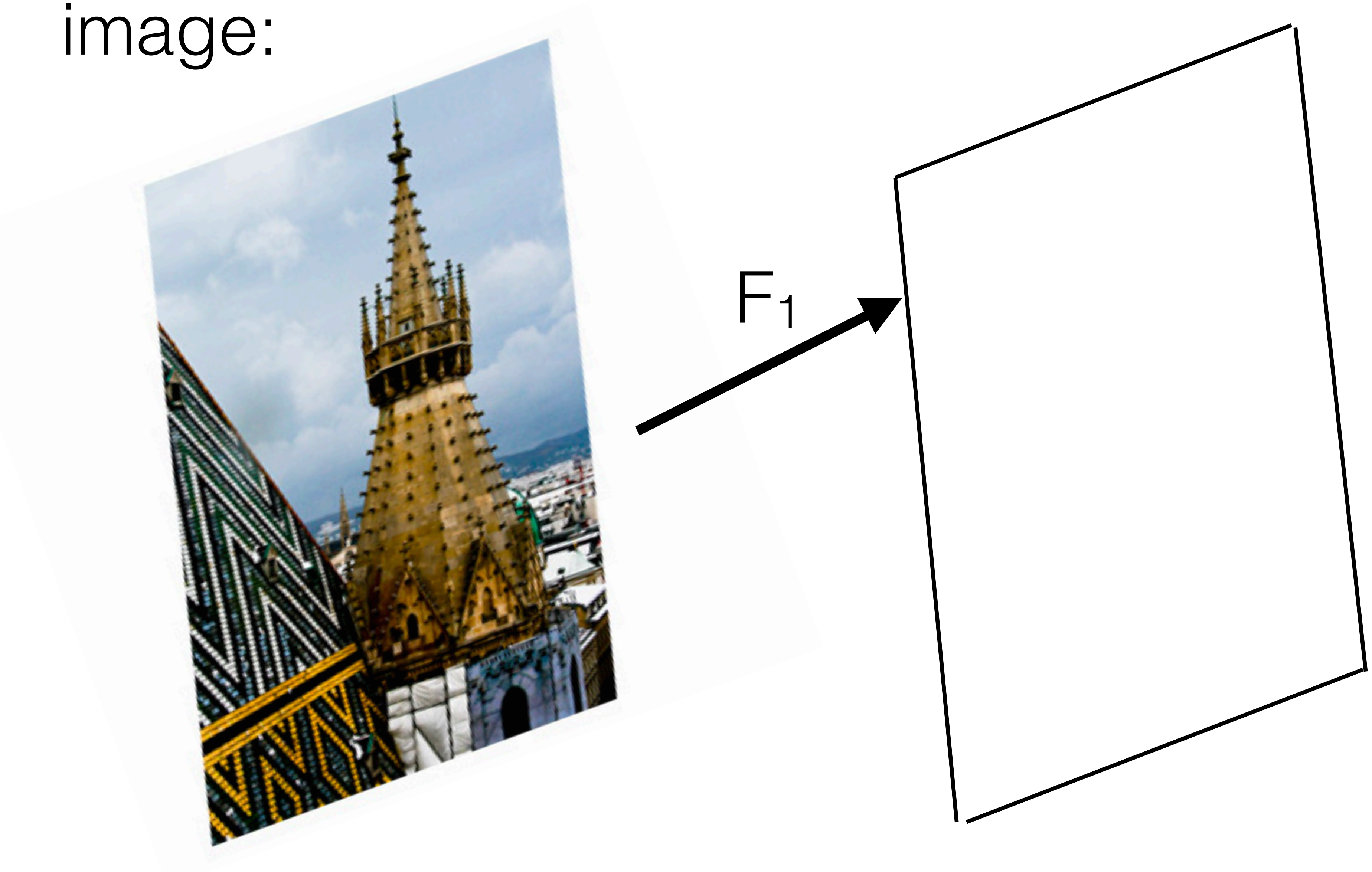
Convolutional Layer: multiple filters

An
image:



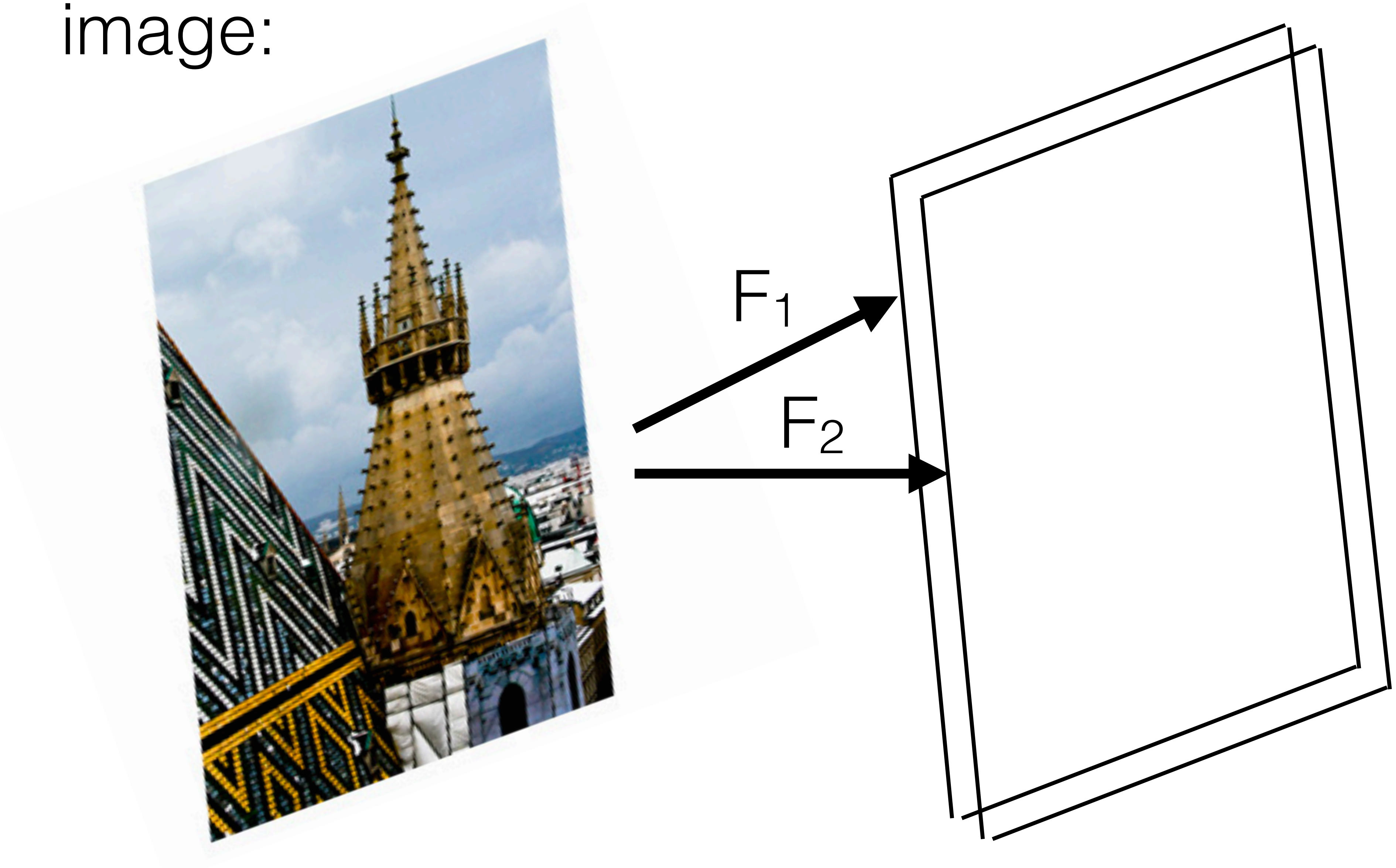
Convolutional Layer: multiple filters

An image:



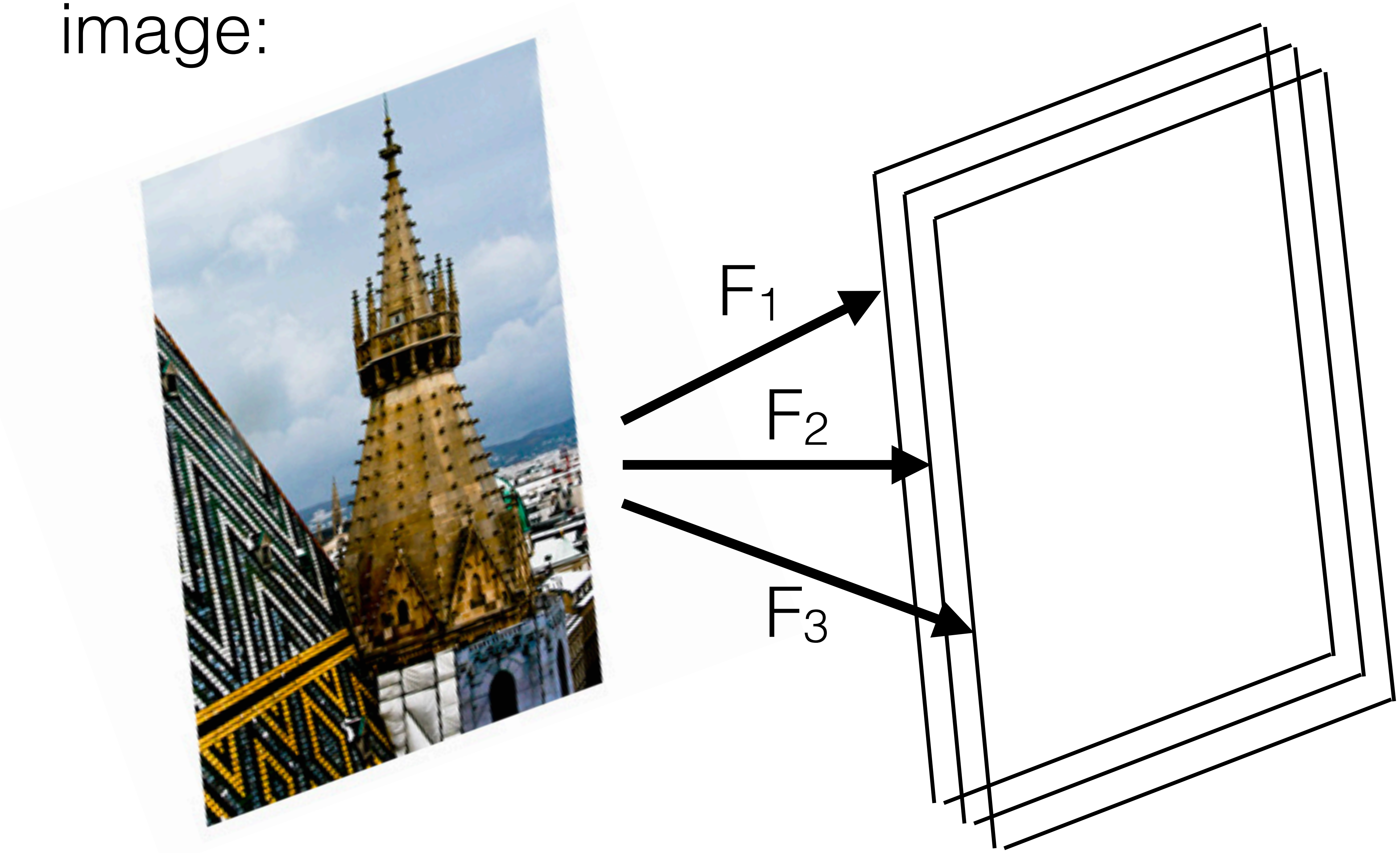
Convolutional Layer: multiple filters

An
image:



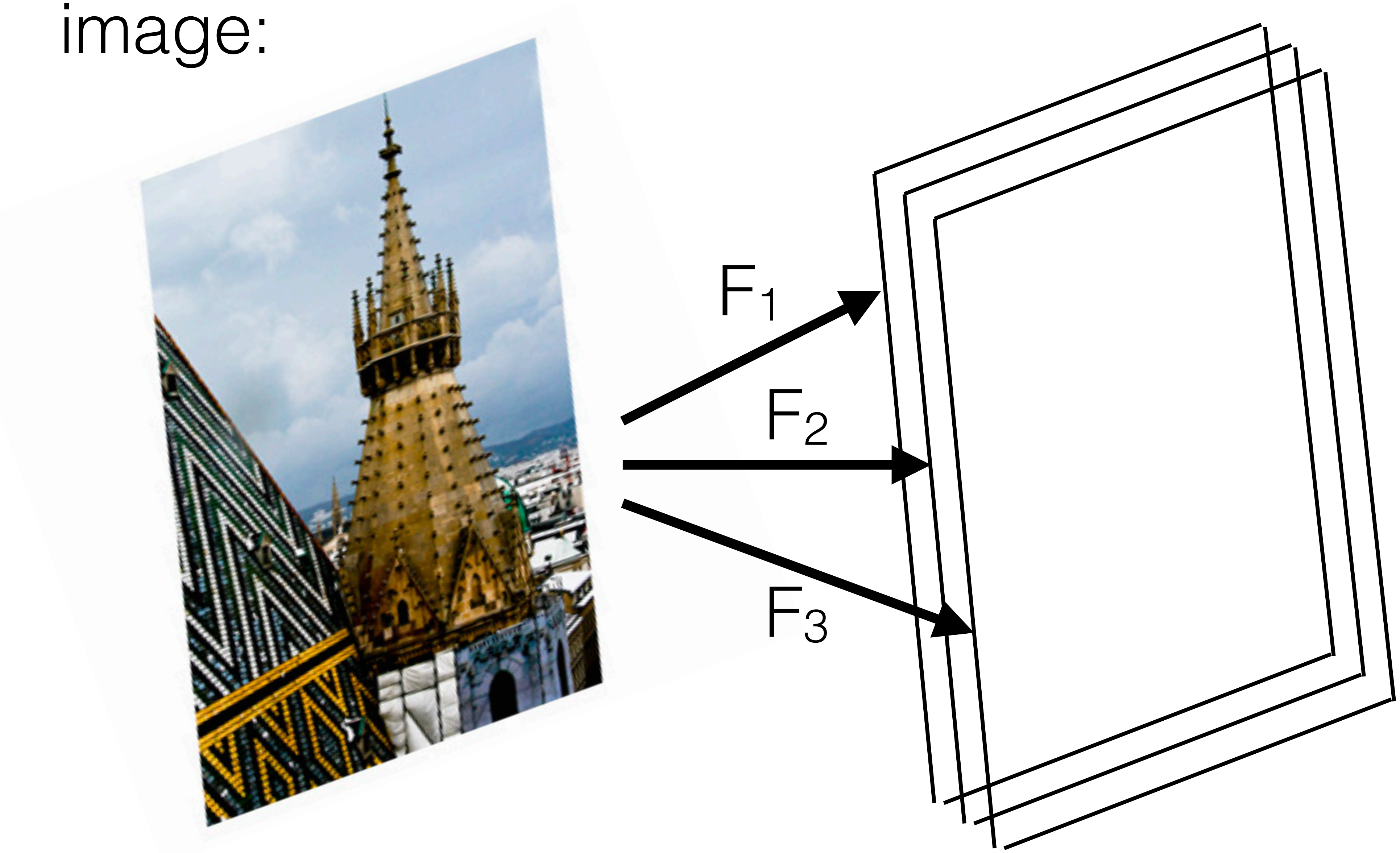
Convolutional Layer: multiple filters

An image:



Convolutional Layer: multiple filters

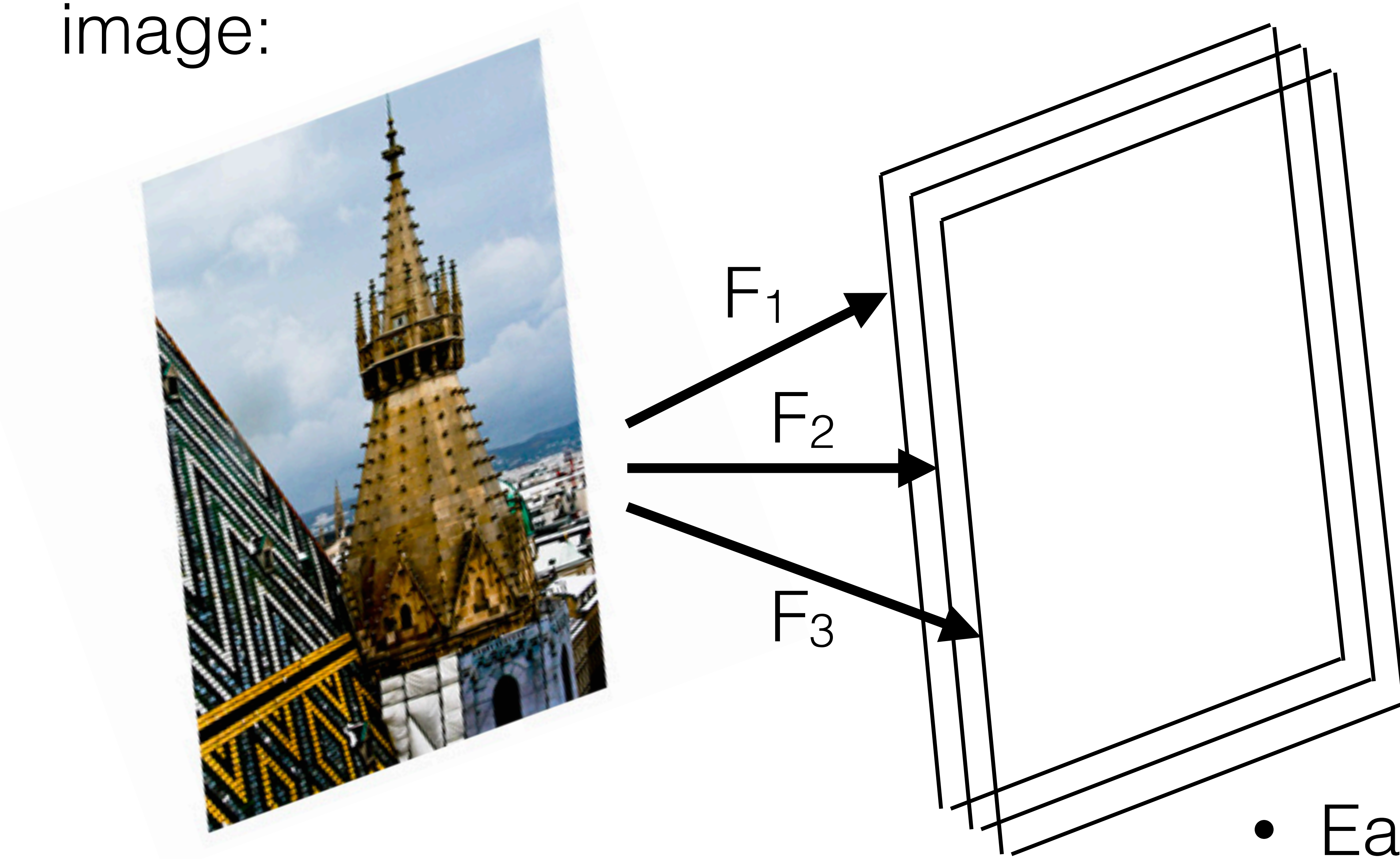
An image:



- Collection of filters in the layer: *filter bank*

Convolutional Layer: multiple filters

An image:



- Collection of filters in the layer: *filter bank*
- Each resulting image is a *channel*

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

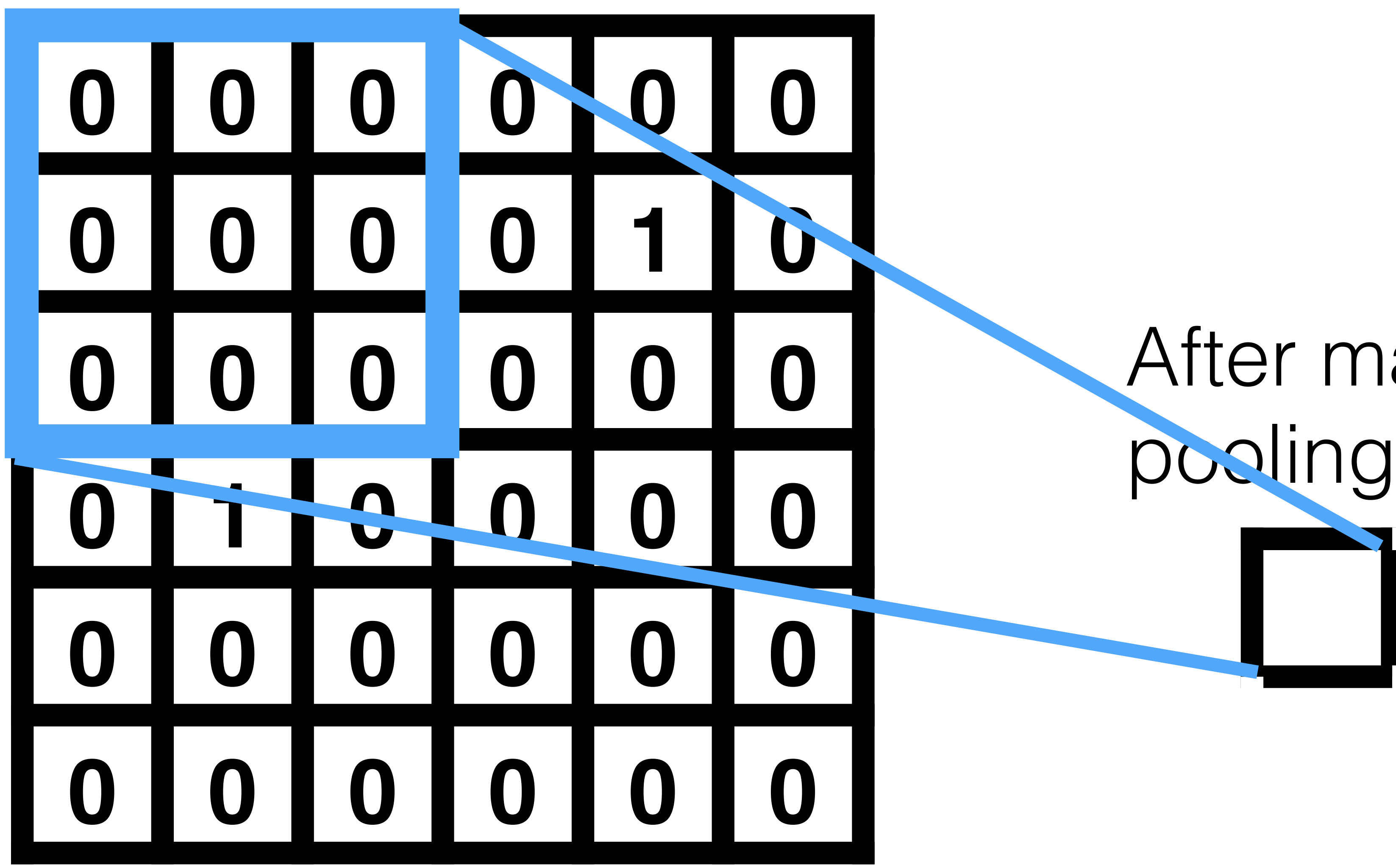
After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)



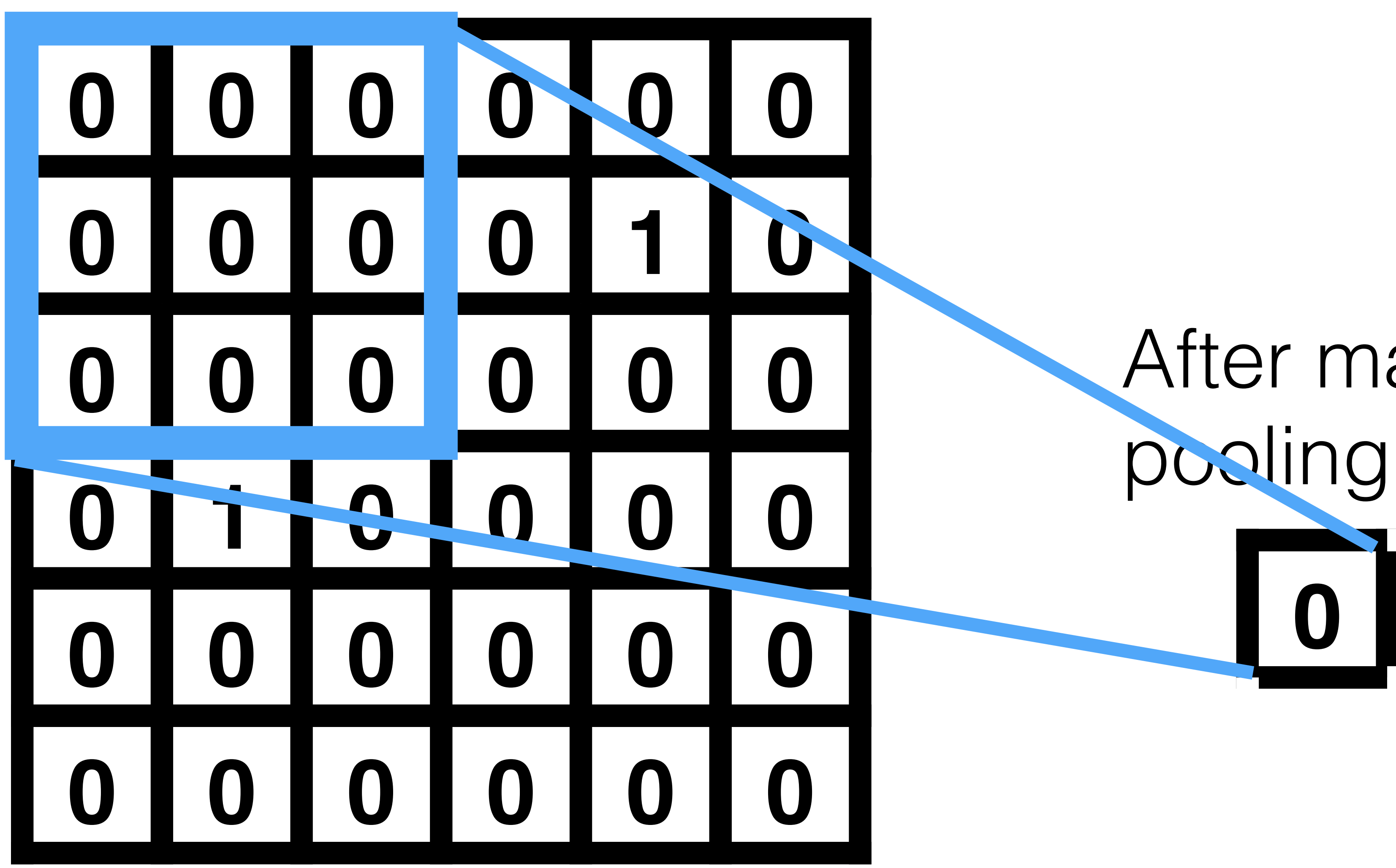
After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

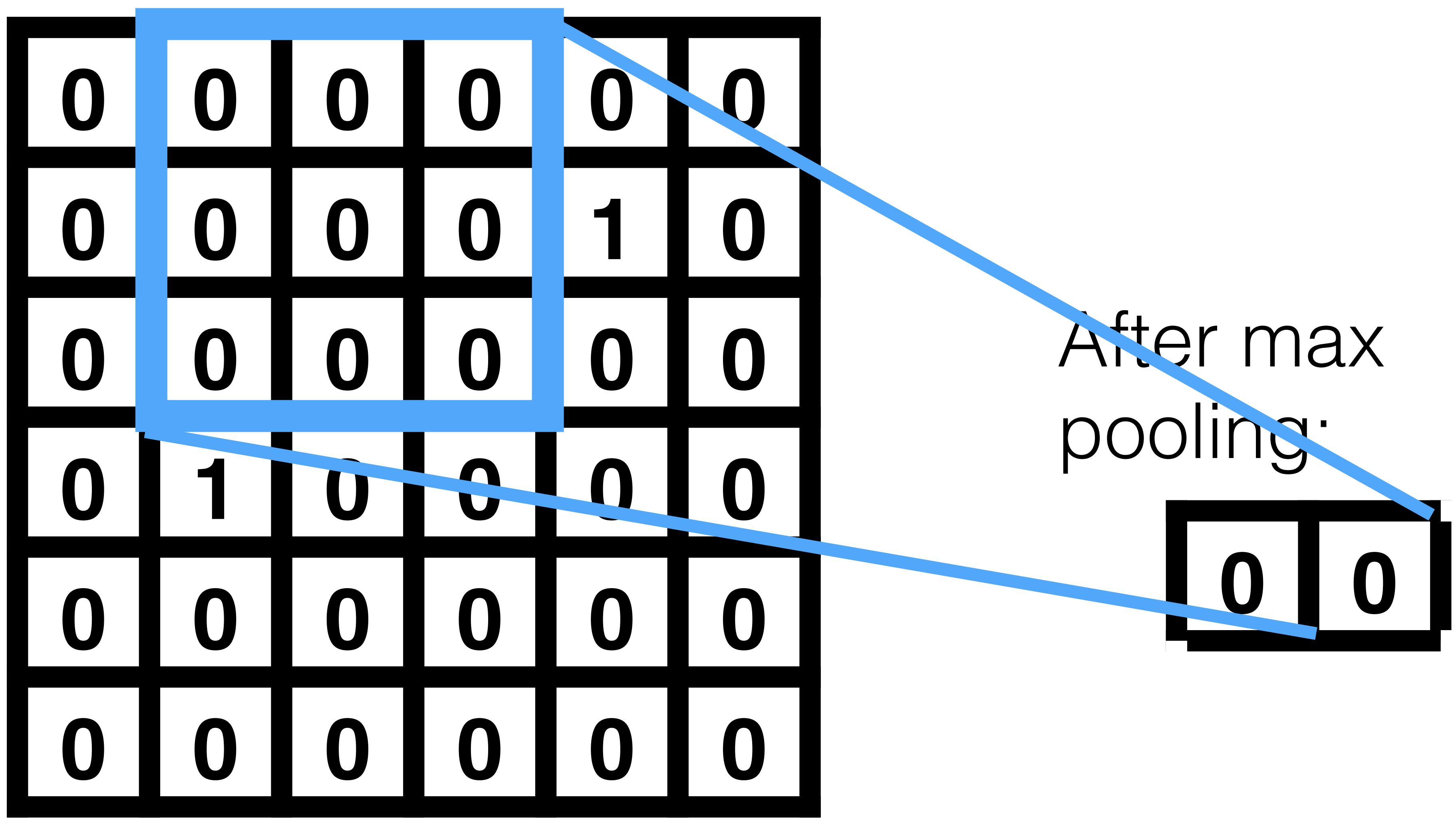


Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

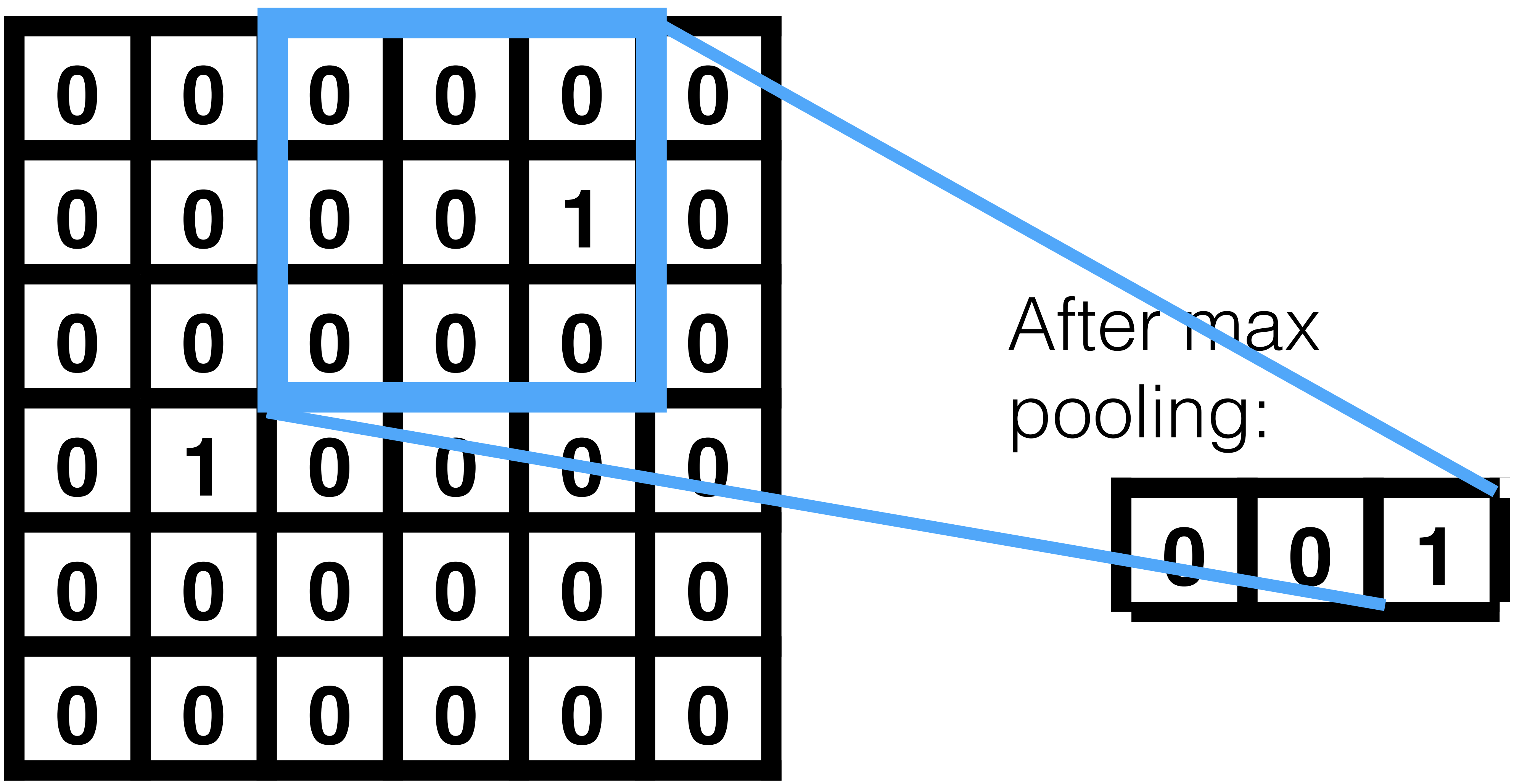


Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)



Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

0	0	1	1

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

0	0	1	1
1	1	1	1
1	1	0	0
1	1	0	0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

0	0	1	1
1	1	1	1
1	1	0	0
1	1	0	0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

0	**0**	**1**	**1**
1	**1**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**0**	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)

After max pooling:

0	**0**	**1**	**1**
1	**1**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**0**	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 1

After max pooling:

0	**0**	**1**	**1**
1	**1**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**0**	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 1

After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 1

After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

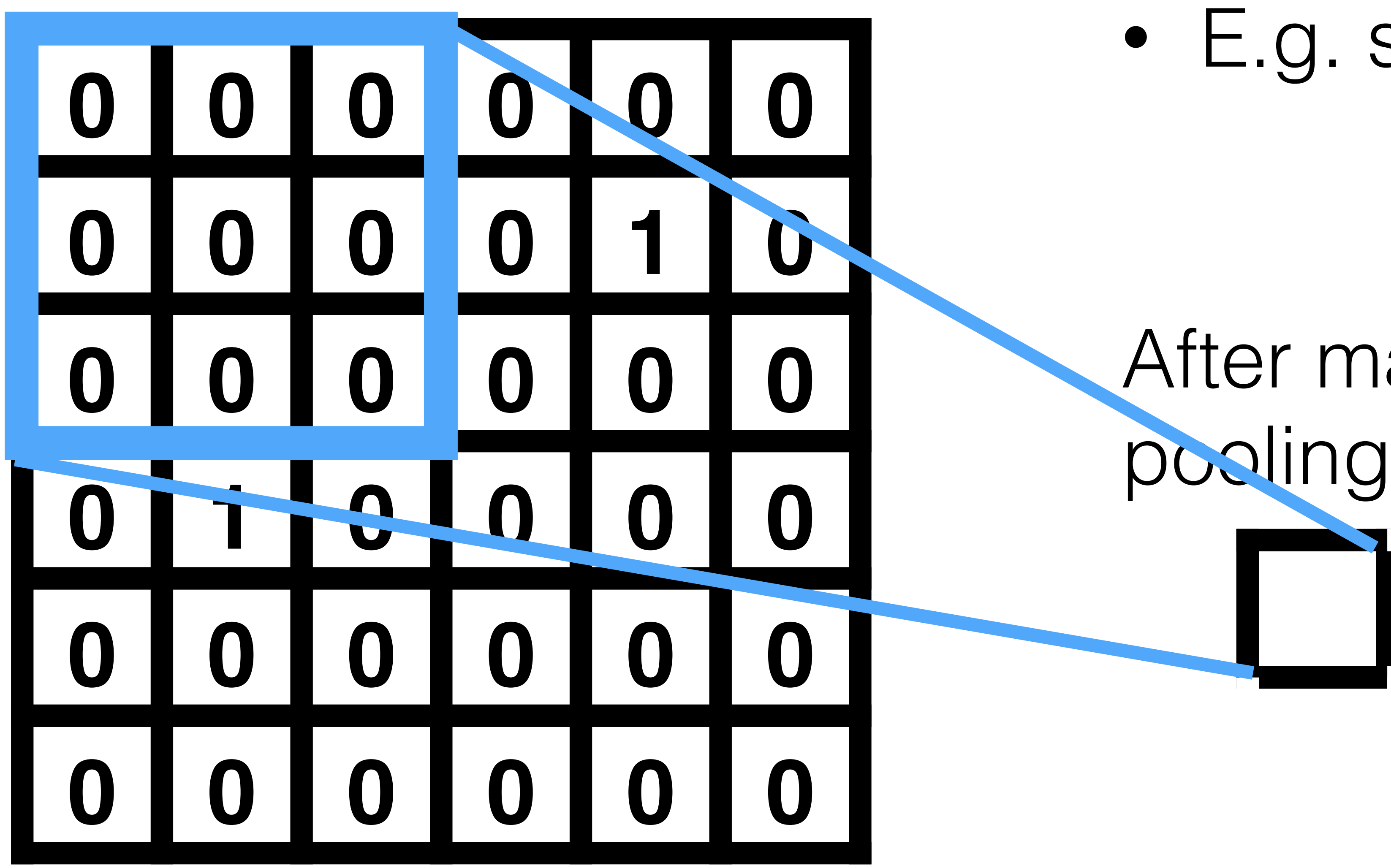
Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

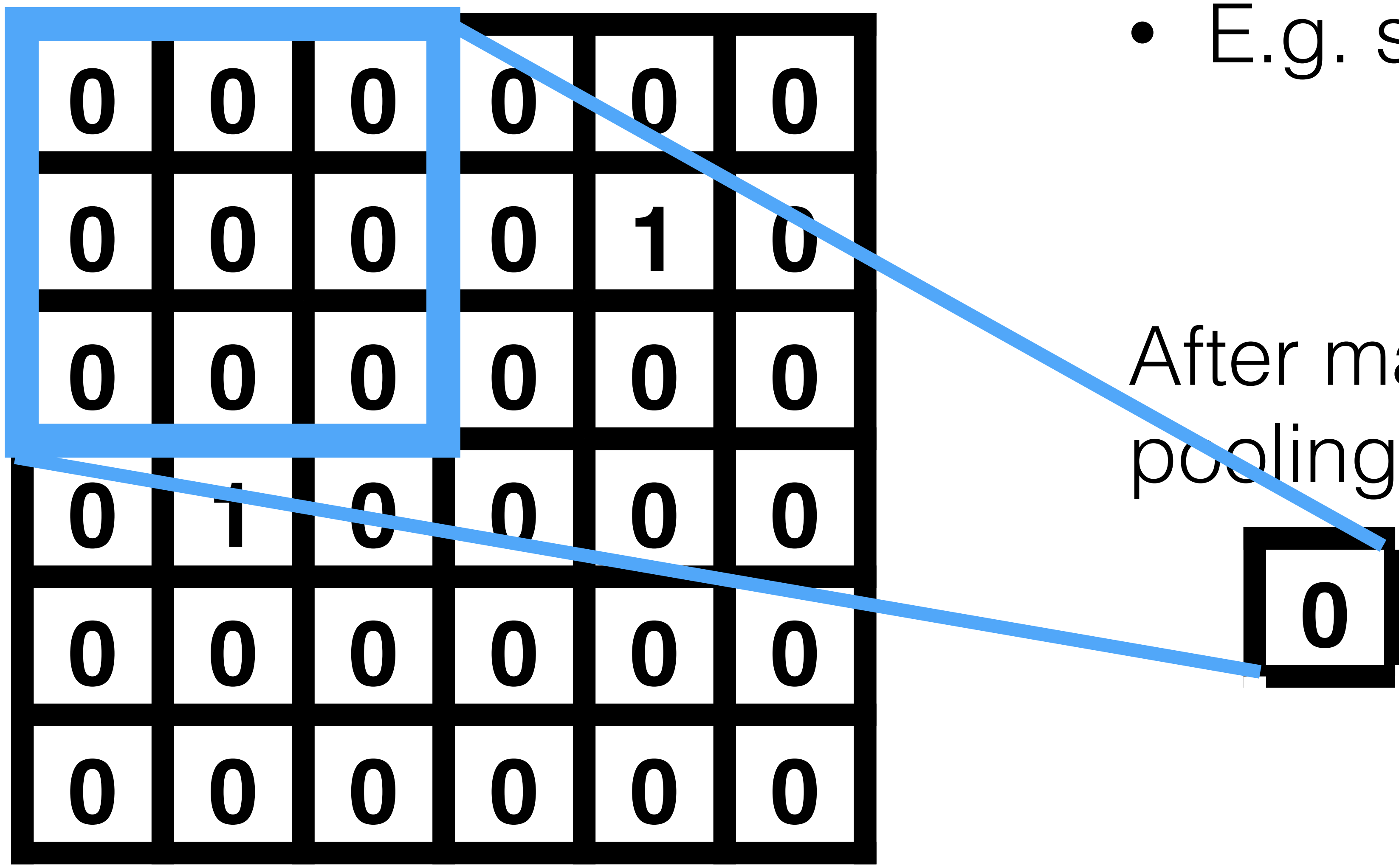
Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:



Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 ("size 3")
- E.g. stride 3

After max pooling:

0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

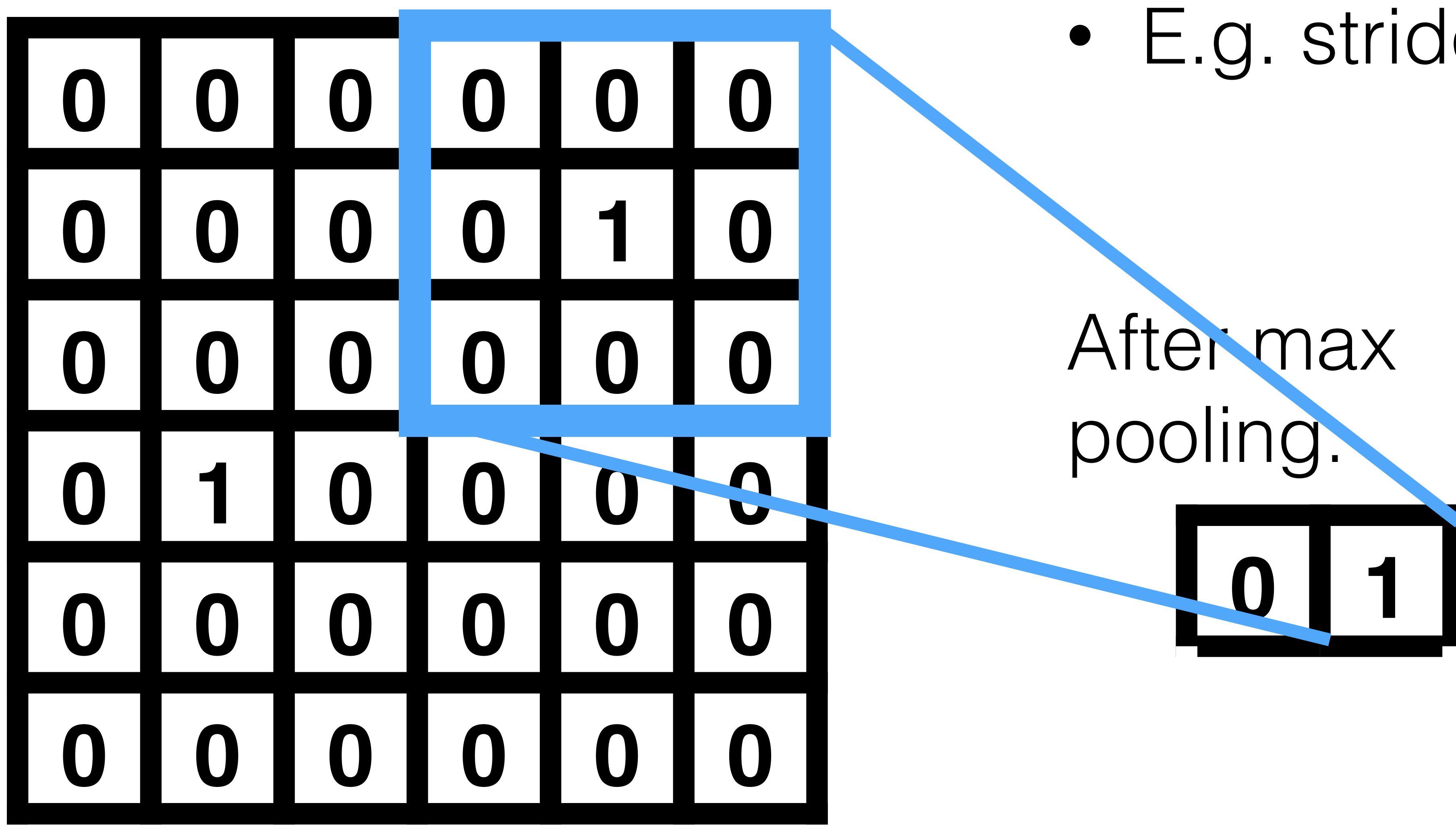
0	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3



Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	1

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	1

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	1

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**

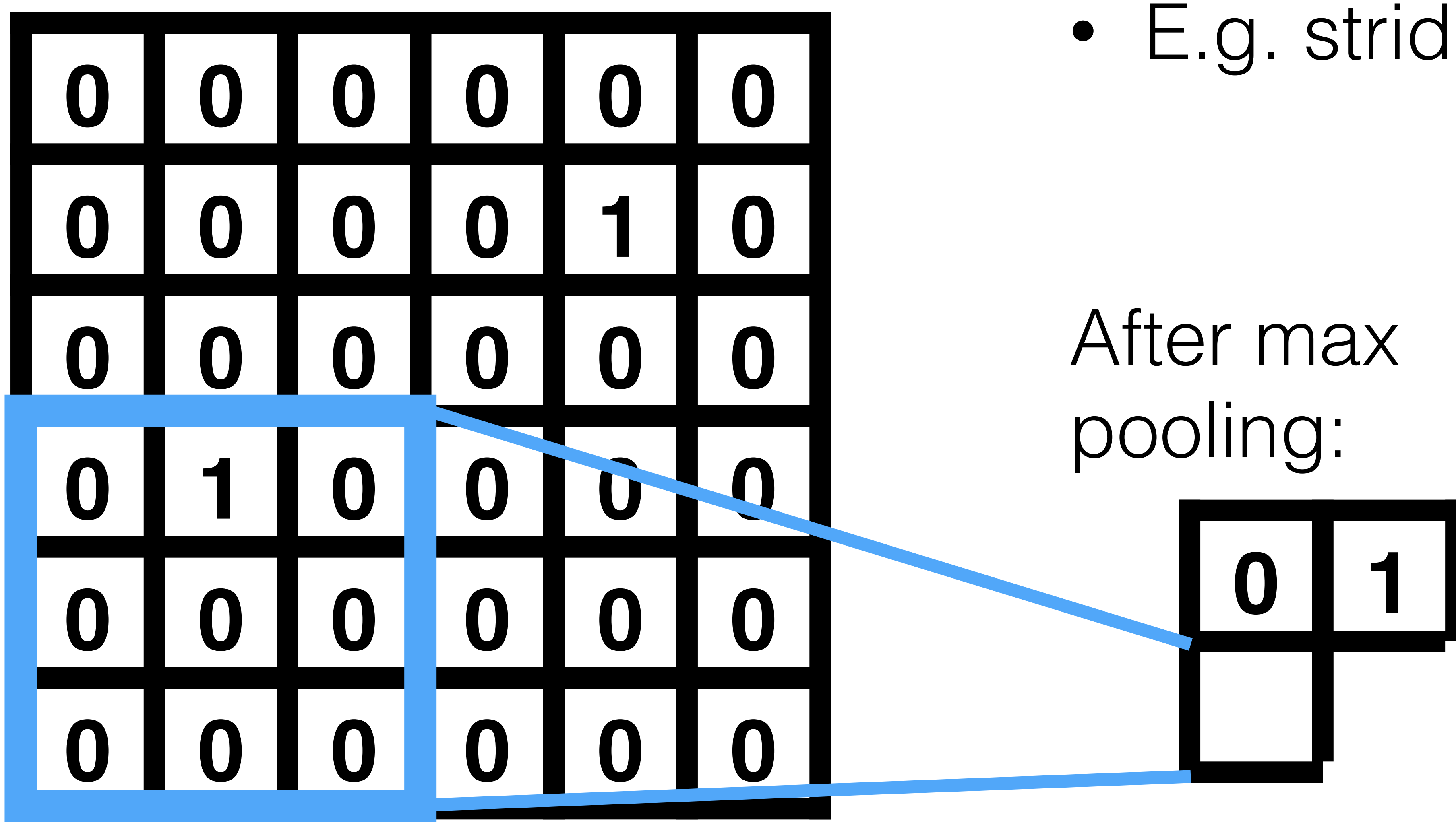
Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:



Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	1
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	1
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 ("size 3")
- E.g. stride 3

After max pooling:

0	1
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	1
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**
1	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**
1	**0**

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

- E.g. size 3x3 ("size 3")
- E.g. stride 3

After max pooling:

0	**1**
1	**0**

- Can use stride with filters too

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**

Max pooling: returns max of its arguments

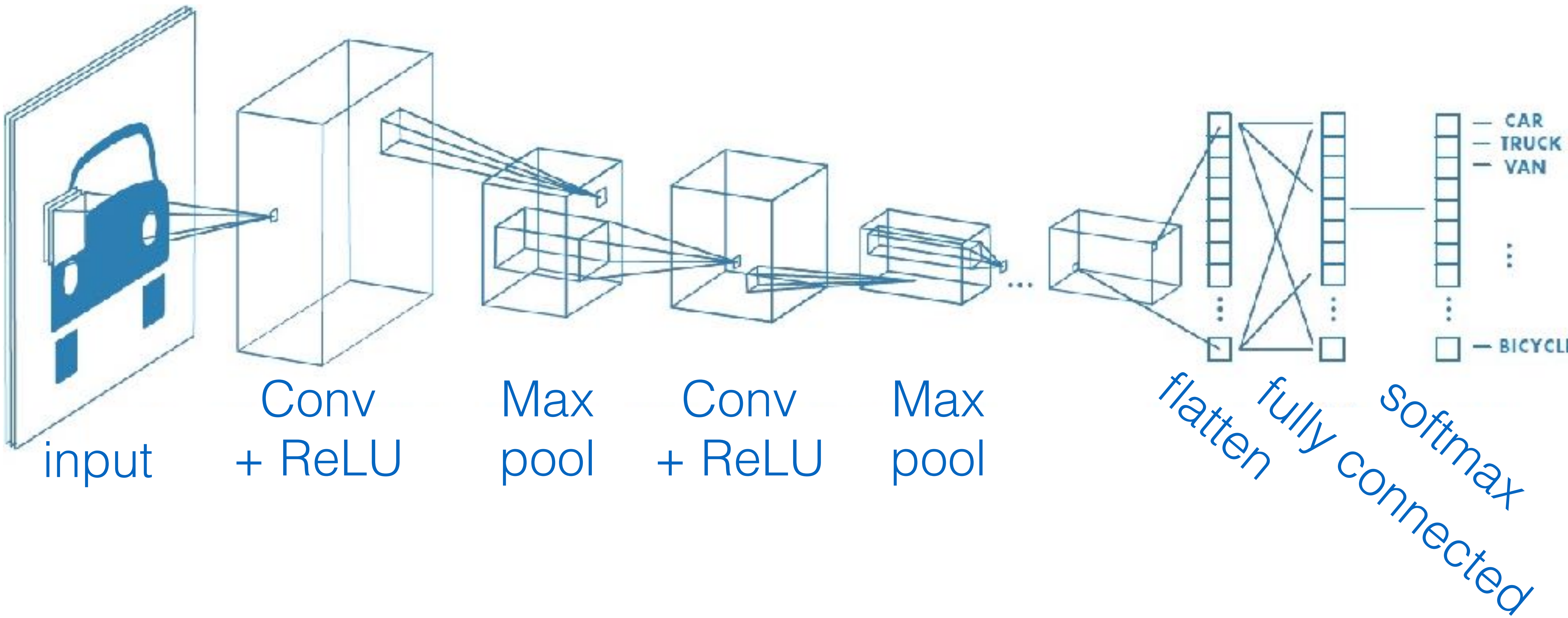
- E.g. size 3x3 (“size 3”)
- E.g. stride 3

After max pooling:

0	**1**
1	**0**

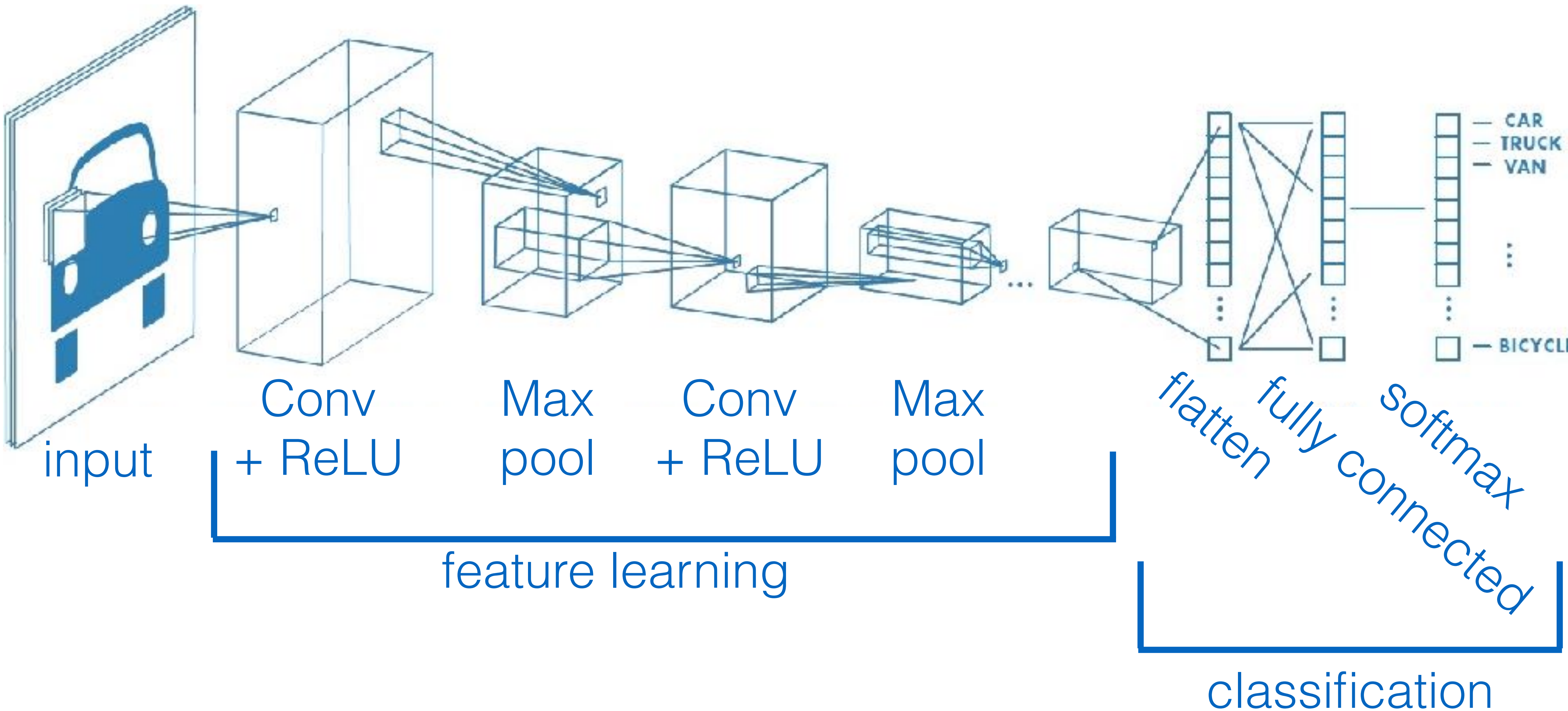
- Can use stride with filters too
- No weights in max pooling

CNNs: typical architecture



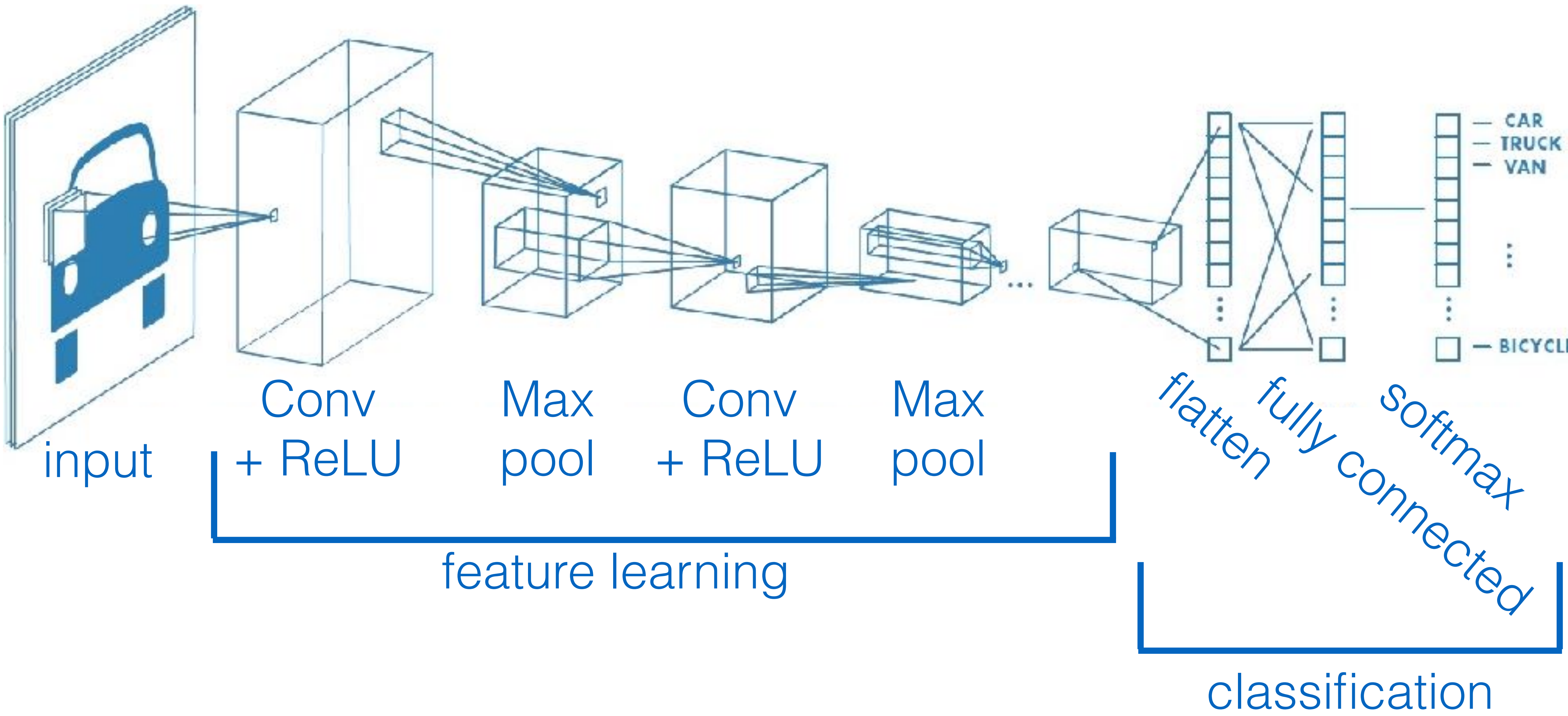
[https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html]

CNNs: typical architecture



[https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html]

CNNs: typical architecture



x $\longrightarrow$ $\mathrm{NN}(x; W, W_0)$

[https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html]

A familiar pattern

A familiar pattern

1. Choose how to predict label (given features & parameters)

A familiar pattern

1. Choose how to predict label (given features & parameters)

*i*th data point

$x^{(i)}$

A familiar pattern

1. Choose how to predict label (given features & parameters)

*i*th data point

Logistic regression: $x^{(i)}$

A familiar pattern

1. Choose how to predict label (given features & parameters)

*i*th data point

Logistic regression: $x^{(i)}$

prediction for *i*th point

$$\mathrm{LogiReg}(x^{(i)}; \theta, \theta_0)$$

A familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guessed label & actual label)

*i*th data point

prediction for *i*th point

Logistic regression: $x^{(i)}$ → $\text{LogiReg}(x^{(i)}; \theta, \theta_0)$

A familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guessed label & actual label)

	*i*th data point		prediction for *i*th point		training loss over points 1 to *n*
Logistic regression:	$x^{(i)}$	→	$\text{LogiReg}(x^{(i)}; \theta, \theta_0)$	→	$J_{\text{Logi}}(\theta, \theta_0)$

A familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guessed label & actual label)
3. Choose parameters by trying to minimize the training loss

*i*th data point

prediction for *i*th point

training loss over points 1 to *n*

Logistic regression: $x^{(i)} \Rightarrow \mathrm{LogiReg}(x^{(i)}; \theta, \theta_0) \Rightarrow J_{\mathrm{Logi}}(\theta, \theta_0)$

A familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guessed label & actual label)
3. Choose parameters by trying to minimize the training loss

	*i*th data point		prediction for *i*th point		training loss over points 1 to *n*
Logistic regression:	$x^{(i)}$	→	$\mathrm{LogiReg}(x^{(i)}; \theta, \theta_0)$	→	$J_{\mathrm{Logi}}(\theta, \theta_0)$
Linear regression:	$x^{(i)}$	→	$\mathrm{LinReg}(x^{(i)}; \theta, \theta_0)$	→	$J_{\mathrm{Lin}}(\theta, \theta_0)$

A familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guessed label & actual label)
3. Choose parameters by trying to minimize the training loss

	*i*th data point		prediction for *i*th point		training loss over points 1 to *n*
Logistic regression:	$x^{(i)}$	$\Rightarrow$	$\mathrm{LogiReg}(x^{(i)}; \theta, \theta_0)$	$\Rightarrow$	$J_{\mathrm{Logi}}(\theta, \theta_0)$
Linear regression:	$x^{(i)}$	$\Rightarrow$	$\mathrm{LinReg}(x^{(i)}; \theta, \theta_0)$	$\Rightarrow$	$J_{\mathrm{Lin}}(\theta, \theta_0)$
Neural networks (e.g. CNNs):	$x^{(i)}$	$\Rightarrow$	$\mathrm{NN}(x^{(i)}; W, W_0)$	$\Rightarrow$	$J_{\mathrm{NN}}(W, W_0)$

A familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guessed label & actual label)
3. Choose parameters by trying to minimize the training loss

	*i*th data point		prediction for *i*th point		training loss over points 1 to *n*
Logistic regression:	$x^{(i)}$	$\Rightarrow$	$\mathrm{LogiReg}(x^{(i)};\theta,\theta_0)$	$\Rightarrow$	$J_{\mathrm{Logi}}(\theta,\theta_0)$
Linear regression:	$x^{(i)}$	$\Rightarrow$	$\mathrm{LinReg}(x^{(i)};\theta,\theta_0)$	$\Rightarrow$	$J_{\mathrm{Lin}}(\theta,\theta_0)$
Neural networks (e.g. CNNs):	$x^{(i)}$	$\Rightarrow$	$\mathrm{NN}(x^{(i)};W,W_0)$	$\Rightarrow$	$J_{\mathrm{NN}}(W,W_0)$

* a key difference: NN can mean a lot of different things

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass: $Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

$$A^2 = (W^2)^\top A^1$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

$$A^2 = (W^2)^\top A^1$$

$$L(A^2, y) = (A^2 - y)^2$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \text{ReLU}(Z_i^1)$$

$$A^2 = (W^2)^\top A^1$$

$$L(A^2, y) = (A^2 - y)^2$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

$$L(A^2, y) = (A^2 - y)^2$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

Loss: 1x1

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

Loss: 1x1

- Part of the derivative for SGD:

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

Loss: 1x1

- Part of the derivative for SGD: $\dfrac{\partial \text{loss}}{\partial W^1} =$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

Loss: 1x1

- Part of the derivative for SGD: $\dfrac{\partial \text{loss}}{\partial W^1} = \qquad \cdot \dfrac{\partial \text{loss}}{\partial A^1}$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $\dfrac{\partial \mathrm{loss}}{\partial W^1} = \quad \cdot \dfrac{\partial A^1}{\partial Z^1} \cdot \dfrac{\partial \mathrm{loss}}{\partial A^1}$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $\dfrac{\partial \mathrm{loss}}{\partial W^1} = \dfrac{\partial Z^1}{\partial W^1} \cdot \dfrac{\partial A^1}{\partial Z^1} \cdot \dfrac{\partial \mathrm{loss}}{\partial A^1}$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

Loss: 1x1

- Part of the derivative for SGD: $$\underbrace{\frac{\partial \mathrm{loss}}{\partial W^1}}_{3\mathrm{x}1} = \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial \mathrm{loss}}{\partial A^1}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $$\underbrace{\frac{\partial \mathrm{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial \mathrm{loss}}{\partial A^1}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $$\underbrace{\frac{\partial \mathrm{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \frac{\partial \mathrm{loss}}{\partial A^1}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD:

$$\underbrace{\frac{\partial \mathrm{loss}}{\partial W^1}}_{3\mathrm{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\mathrm{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\mathrm{x}5} \cdot \underbrace{\frac{\partial \mathrm{loss}}{\partial A^1}}_{5\mathrm{x}1}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $$\underbrace{\frac{\partial \mathrm{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \mathrm{loss}}{\partial A^1}}_{5\text{x}1}$$

$$\frac{\partial Z^1}{\partial W^1}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $\underbrace{\frac{\partial \text{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \text{loss}}{\partial A^1}}_{5\text{x}1}$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{c} \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} \\ \left[\begin{array}{ccccc} \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \end{array} \right] \end{array} \begin{array}{c} W_1^1 \\ W_2^1 \\ W_3^1 \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $$\underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W^1}} = \underset{3\text{x}5}{\frac{\partial Z^1}{\partial W^1}} \cdot \underset{5\text{x}5}{\frac{\partial A^1}{\partial Z^1}} \cdot \underset{5\text{x}1}{\frac{\partial \text{loss}}{\partial A^1}}$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{c} \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} \\ \left[\begin{array}{ccccc} \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \end{array} \right] \end{array} \begin{array}{c} W_1^1 \\ W_2^1 \\ W_3^1 \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$

Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$

A^1: 5x1

$$A^2 = (W^2)^\top A^1$$

A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$

Loss: 1x1

- Part of the derivative for SGD:

$$\underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W^1}} = \underset{3\text{x}5}{\frac{\partial Z^1}{\partial W^1}} \cdot \underset{5\text{x}5}{\frac{\partial A^1}{\partial Z^1}} \cdot \underset{5\text{x}1}{\frac{\partial \text{loss}}{\partial A^1}}$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{c} \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} \\ \left[\begin{array}{ccccc} & & & & \\ & & & & \\ & \frac{\partial Z_2^1}{\partial W_3^1} & & & \end{array}\right] \end{array} \begin{array}{c} \\ W_1^1 \\ W_2^1 \\ W_3^1 \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]} \qquad Z^1\text{: 5x1}$$

$$A_i^1 = \mathrm{ReLU}(Z_i^1) \qquad A^1\text{: 5x1}$$

$$A^2 = (W^2)^\top A^1 \qquad A^2\text{: 1x1}$$

$$L(A^2, y) = (A^2 - y)^2 \qquad \text{Loss: 1x1}$$

- Part of the derivative for SGD:

$$\underbrace{\frac{\partial \text{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \text{loss}}{\partial A^1}}_{5\text{x}1}$$

$$\boxed{Z_2^1 = W_1^1 X_1 + W_2^1 X_2 + W_3^1 X_3}$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{cccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 & \\ \left[\begin{array}{c} \\ \\ \end{array} \right. & & & & & W_1^1 \\ & & & & & W_2^1 \\ & \frac{\partial Z_2^1}{\partial W_3^1} & & & \left. \right] & W_3^1 \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $$\underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W^1}} = \underset{3\text{x}5}{\frac{\partial Z^1}{\partial W^1}} \cdot \underset{5\text{x}5}{\frac{\partial A^1}{\partial Z^1}} \cdot \underset{5\text{x}1}{\frac{\partial \text{loss}}{\partial A^1}}$$

$$Z_2^1 = W_1^1 X_1 + W_2^1 X_2 + W_3^1 X_3$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{cc} \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} & \\ \left[\begin{array}{ccccc} & & & & \\ & & & & \\ & \boxed{\frac{\partial Z_2^1}{\partial W_3^1}} & & & \end{array}\right] & \begin{array}{c} W_1^1 \\ W_2^1 \\ W_3^1 \end{array} \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z^1_i = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A^1_i = \mathrm{ReLU}(Z^1_i)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD:

$$\underbrace{\frac{\partial \text{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \text{loss}}{\partial A^1}}_{5\text{x}1}$$

$$Z^1_2 = W^1_1 X_1 + W^1_2 X_2 + W^1_3 X_3$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{cc} \begin{array}{ccccc} Z^1_1 & Z^1_2 & Z^1_3 & Z^1_4 & Z^1_5 \end{array} & \\ \left[\begin{array}{ccccc} & & & & \\ & & & & \\ & \boxed{X_3} & & & \end{array}\right] & \begin{array}{c} W^1_1 \\ W^1_2 \\ W^1_3 \end{array} \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD: $$\underbrace{\frac{\partial \text{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \text{loss}}{\partial A^1}}_{5\text{x}1}$$

$$Z_2^1 = W_1^1 X_1 + W_2^1 X_2 + W_3^1 X_3$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{c|c} & \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} \\ \hline \begin{array}{c} W_1^1 \\ W_2^1 \\ W_3^1 \end{array} & \begin{bmatrix} & & & & \\ & & & & \\ & X_3 & & & \end{bmatrix} \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD:

$$\underset{3\text{x}1}{\frac{\partial \text{loss}}{\partial W^1}} = \underset{3\text{x}5}{\frac{\partial Z^1}{\partial W^1}} \cdot \underset{5\text{x}5}{\frac{\partial A^1}{\partial Z^1}} \cdot \underset{5\text{x}1}{\frac{\partial \text{loss}}{\partial A^1}}$$

$$Z_2^1 = W_1^1 X_1 + W_2^1 X_2 + W_3^1 X_3$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{cc} \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} & \\ \left[\begin{array}{ccccc} & X_1 & & & \\ & X_2 & & & \\ & X_3 & & & \end{array}\right] & \begin{array}{c} W_1^1 \\ W_2^1 \\ W_3^1 \end{array} \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD:

$$\underbrace{\frac{\partial \text{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \text{loss}}{\partial A^1}}_{5\text{x}1}$$

$$Z_2^1 = W_1^1 X_1 + W_2^1 X_2 + W_3^1 X_3$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{cc} \begin{array}{ccccc} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{array} & \\ \begin{bmatrix} X_0 & X_1 & X_2 & X_3 & X_4 \\ X_1 & X_2 & X_3 & X_4 & X_5 \\ X_2 & X_3 & X_4 & X_5 & X_6 \end{bmatrix} & \begin{array}{c} W_1^1 \\ W_2^1 \\ W_3^1 \end{array} \end{array}$$

CNNs: a taste of backpropagation

Regression. 1 filter: size 3 & padding; $x^{(j)}$ dimension: 5x1

- Forward pass:

$$Z_i^1 = (W^1)^\top X_{[i-1,i,i+1]}$$ Z^1: 5x1

$$A_i^1 = \mathrm{ReLU}(Z_i^1)$$ A^1: 5x1

$$A^2 = (W^2)^\top A^1$$ A^2: 1x1

$$L(A^2, y) = (A^2 - y)^2$$ Loss: 1x1

- Part of the derivative for SGD:

$$\underbrace{\frac{\partial \text{loss}}{\partial W^1}}_{3\text{x}1} = \underbrace{\frac{\partial Z^1}{\partial W^1}}_{3\text{x}5} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1}}_{5\text{x}5} \cdot \underbrace{\frac{\partial \text{loss}}{\partial A^1}}_{5\text{x}1}$$

$$Z_2^1 = W_1^1 X_1 + W_2^1 X_2 + W_3^1 X_3$$

$$\frac{\partial Z^1}{\partial W^1} = \begin{array}{cc} \begin{matrix} Z_1^1 & Z_2^1 & Z_3^1 & Z_4^1 & Z_5^1 \end{matrix} & \\ \begin{bmatrix} \boxed{X_0} & X_1 & X_2 & X_3 & X_4 \\ X_1 & X_2 & X_3 & X_4 & X_5 \\ X_2 & X_3 & X_4 & X_5 & \boxed{X_6} \end{bmatrix} & \begin{matrix} W_1^1 \\ W_2^1 \\ W_3^1 \end{matrix} \end{array}$$

MIT · 6.036 | Introduction to Machine Learning (2020)

MIT 6.036(2020)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1y44y187wN

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/mit-6.036

机器学习 循环神经 神经网络 感知器
特征构建 聚类 马尔可夫决策过程 网络
随机森林 决策树 逻辑回归 卷积神经网络 状态机

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	**图机器学习**	**深度强化学习**	**自动驾驶**
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]