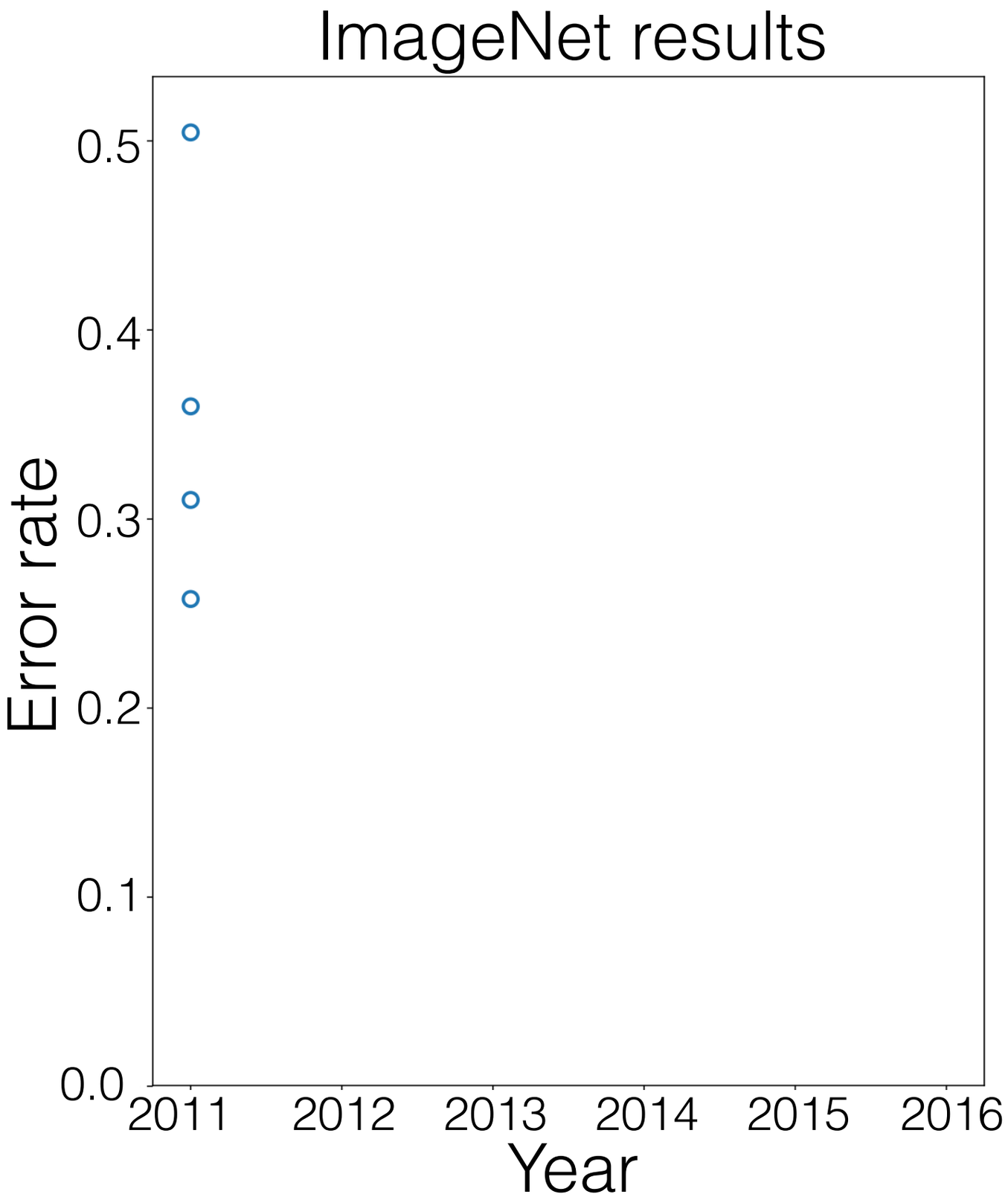
MIT EECS
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.036: Convolutional Neural Networks (CNNs, ConvNets)

Prof. Tamara Broderick
EECS, MIT

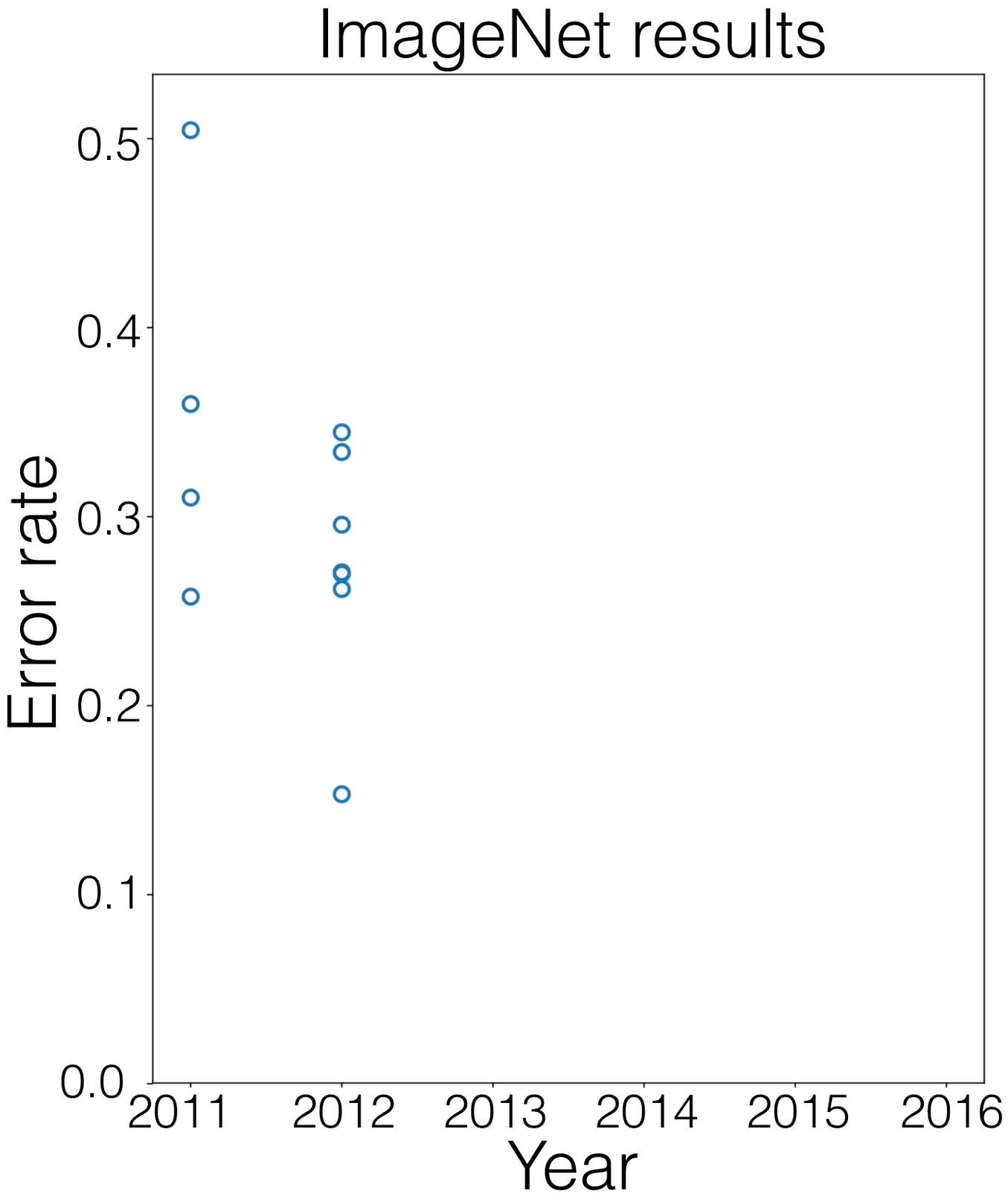
Impact of CNNs



- Since 2010: large-scale image classification challenge

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

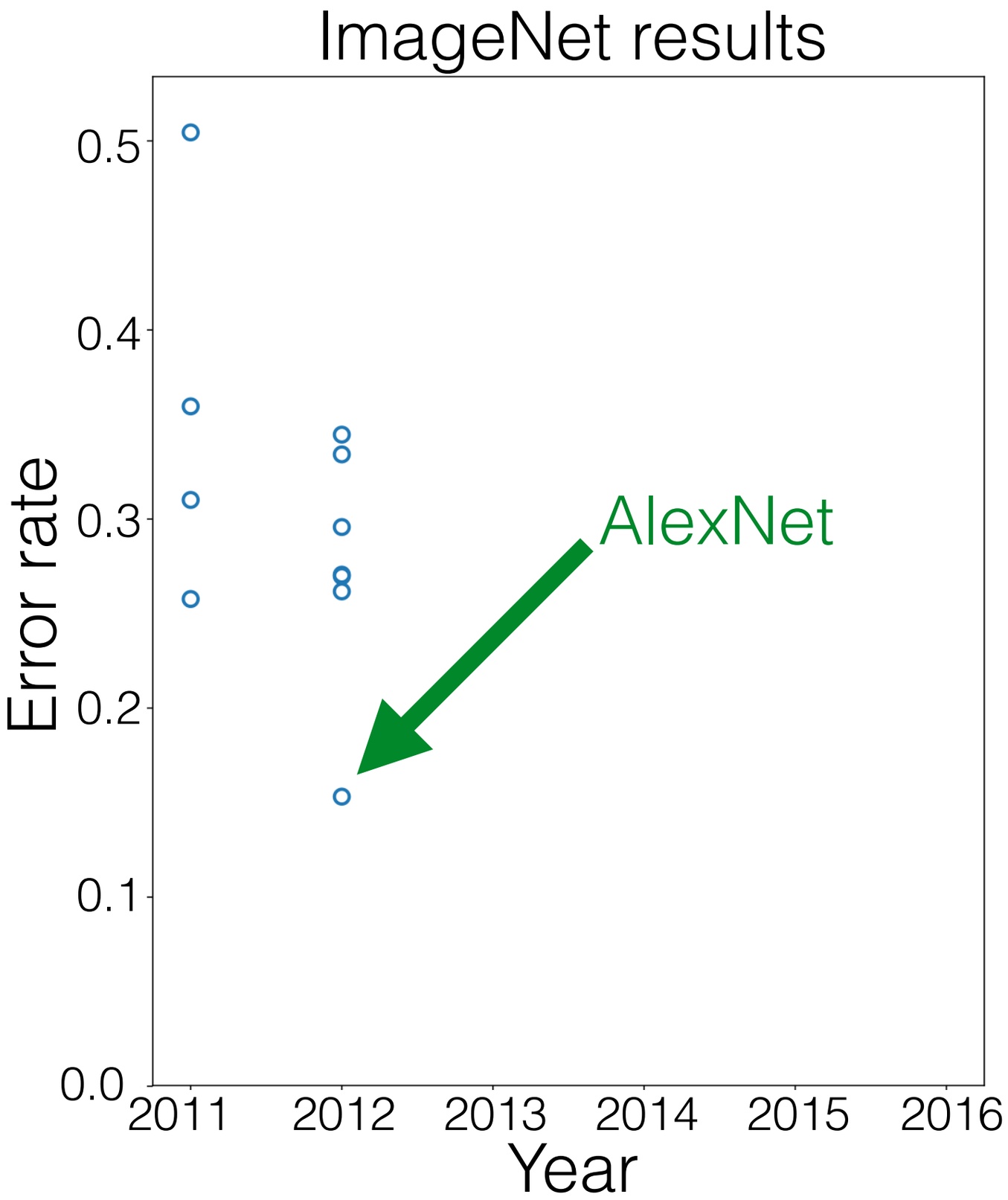
Impact of CNNs



- Since 2010: large-scale image classification challenge

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

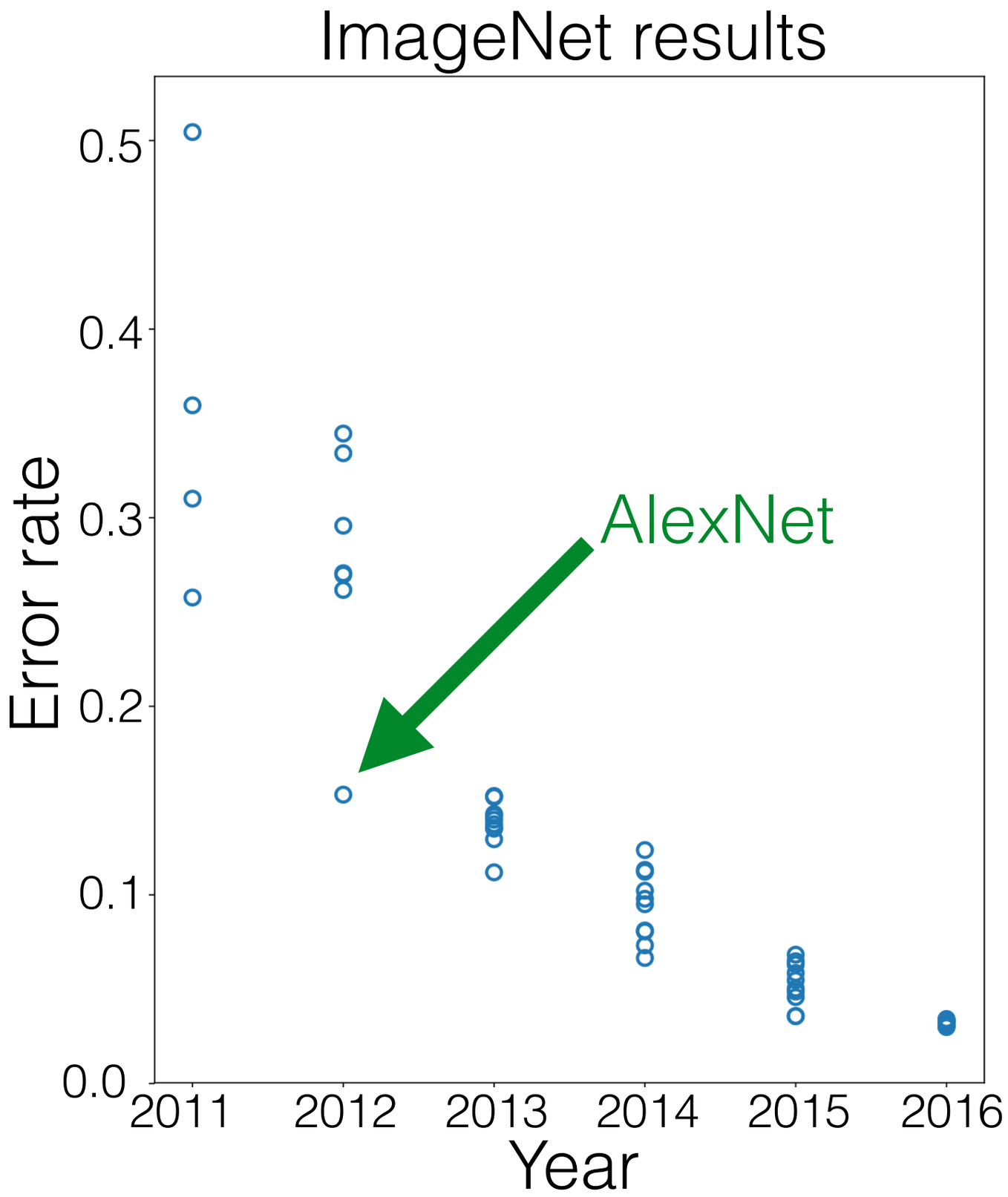
Impact of CNNs



- Since 2010: large-scale image classification challenge

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

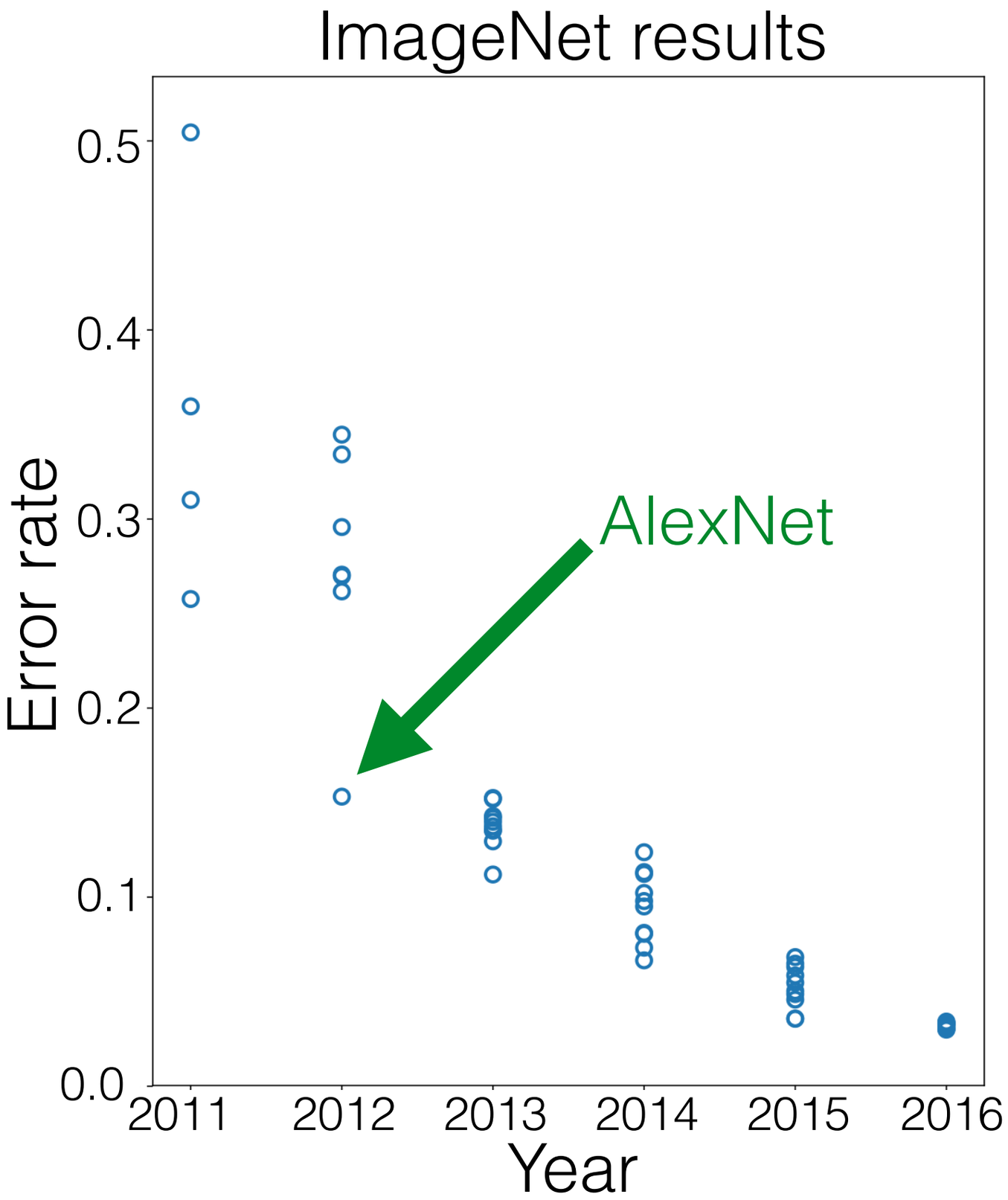
Impact of CNNs



- Since 2010: large-scale image classification challenge

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

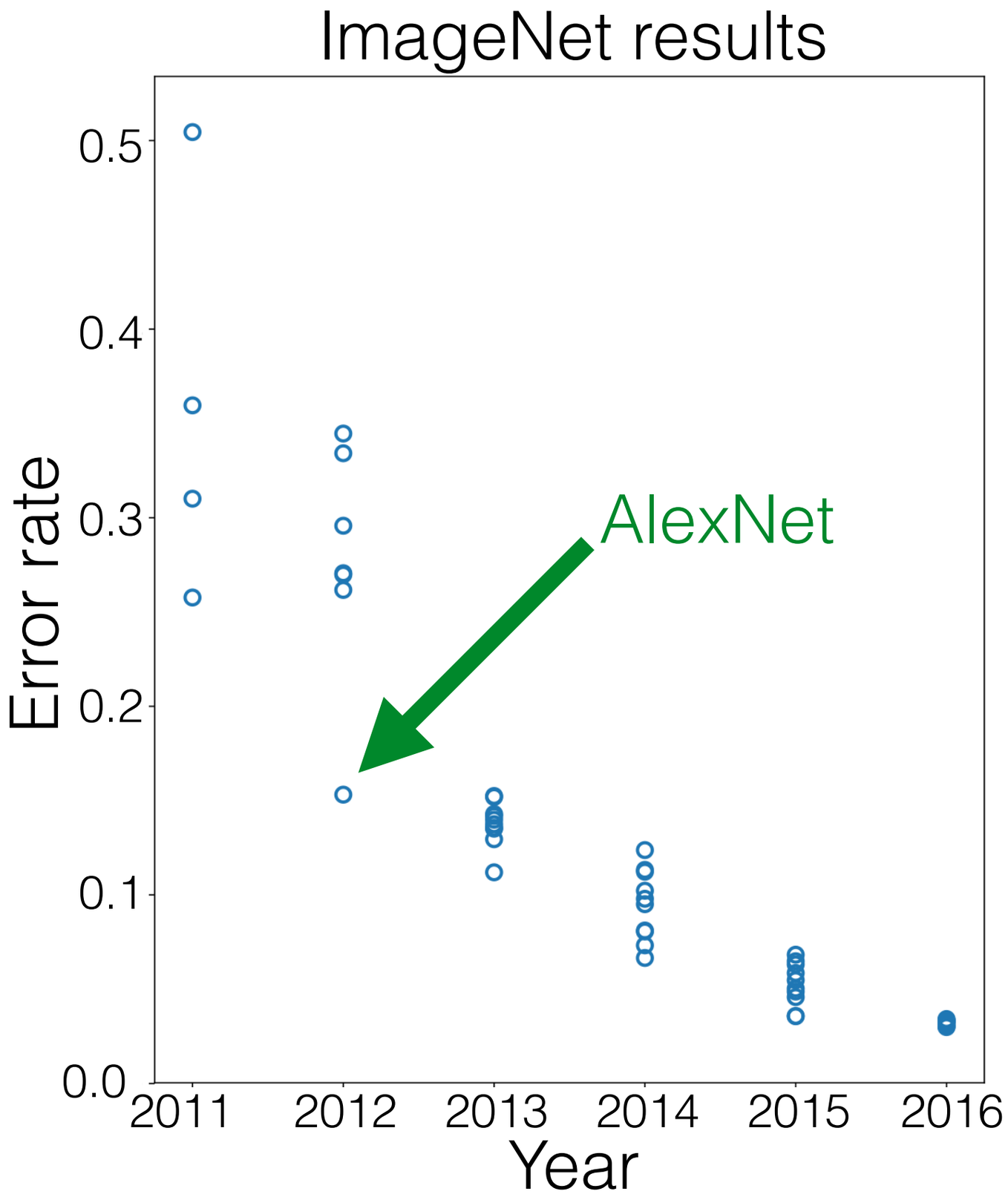
Impact of CNNs



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

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

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]

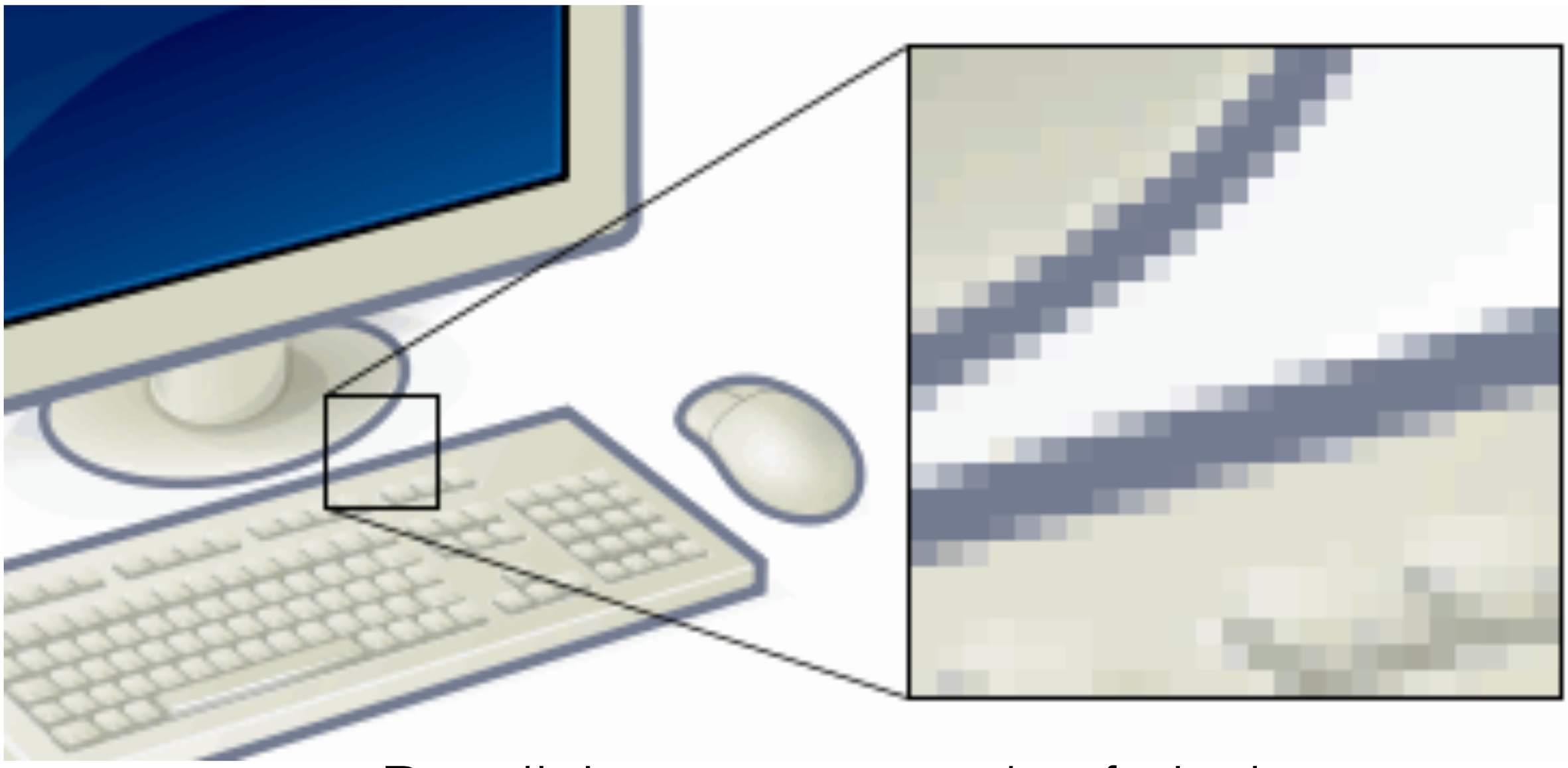
Images

- 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



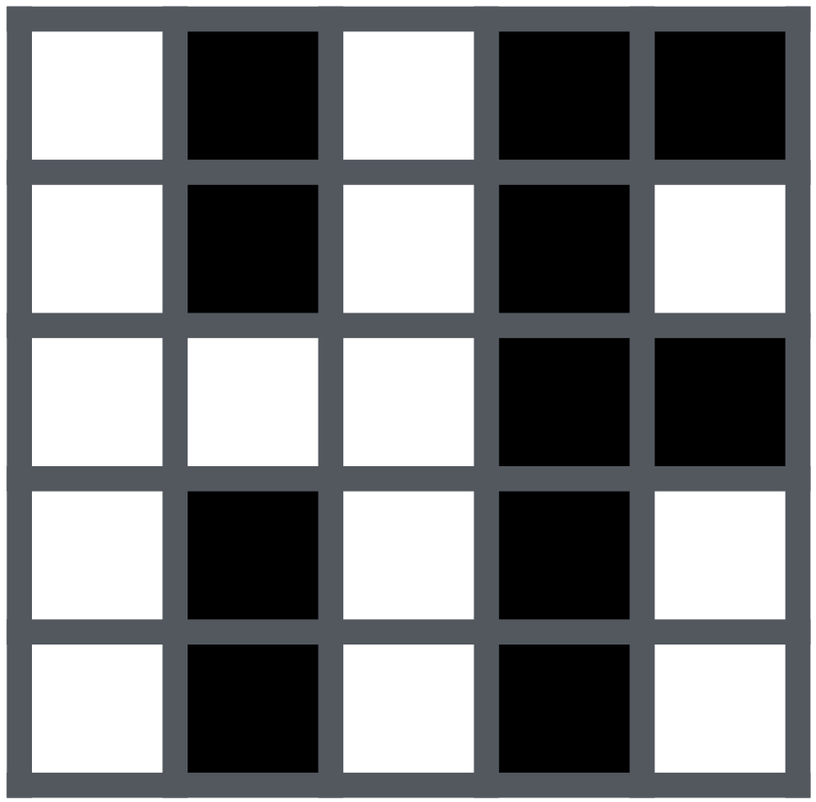
- Recall: images are made of pixels

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

Images

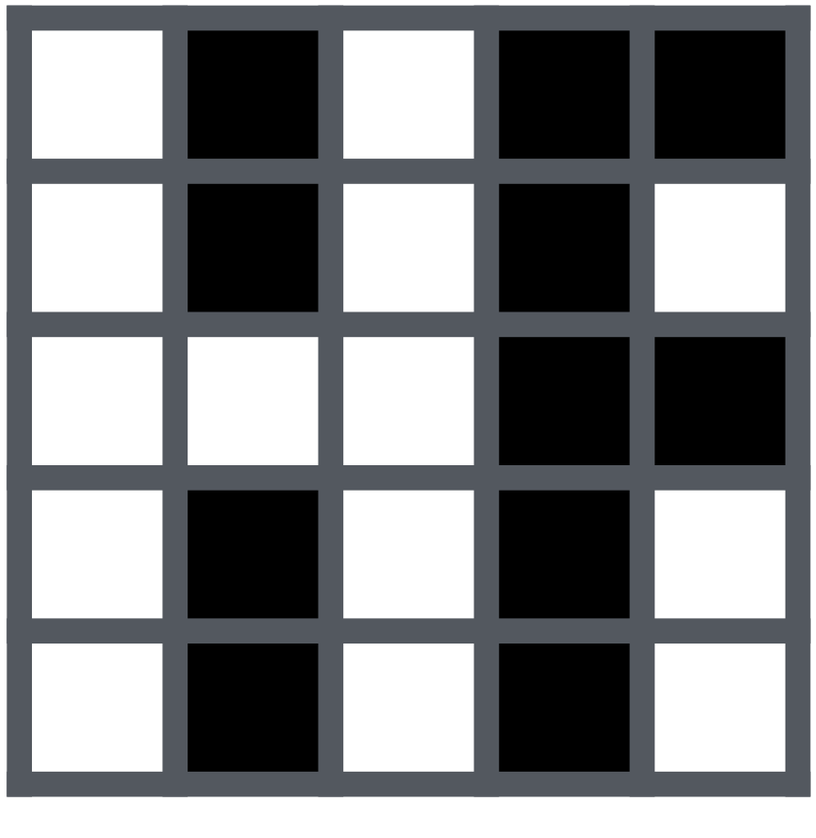
- We’ll focus on grayscale images

Images



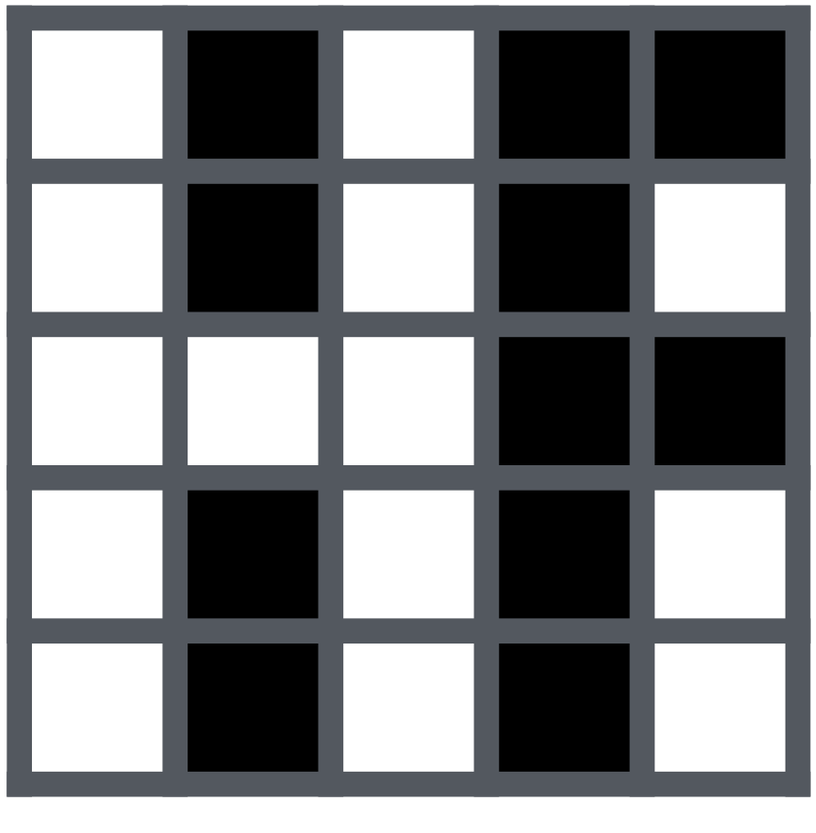
- We'll focus on grayscale images

Images



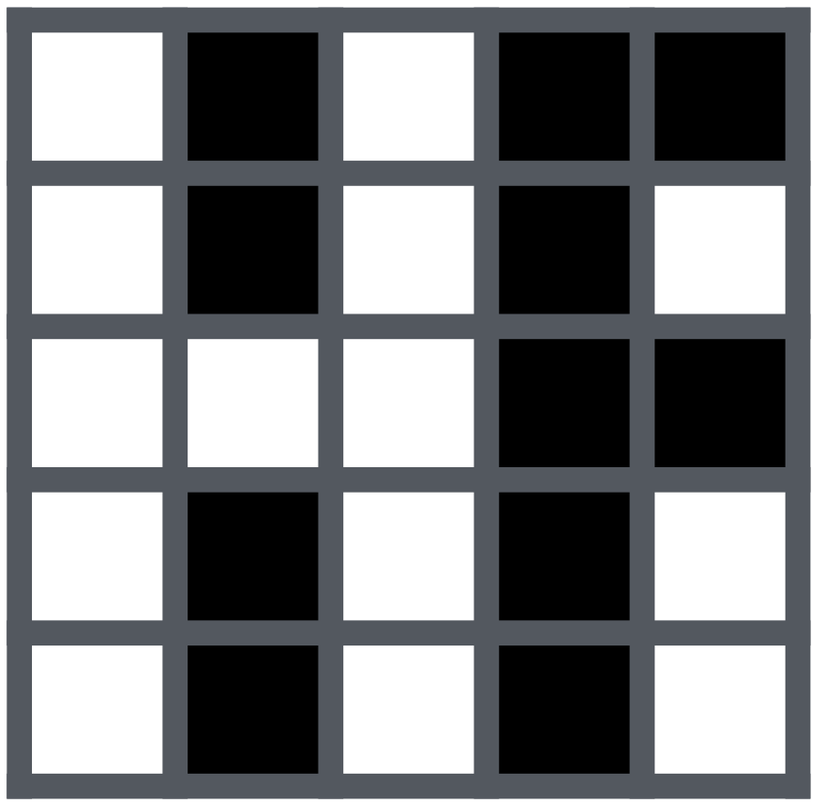
- We'll focus on grayscale images
  - Each pixel takes a value between 0 and *P*

Images



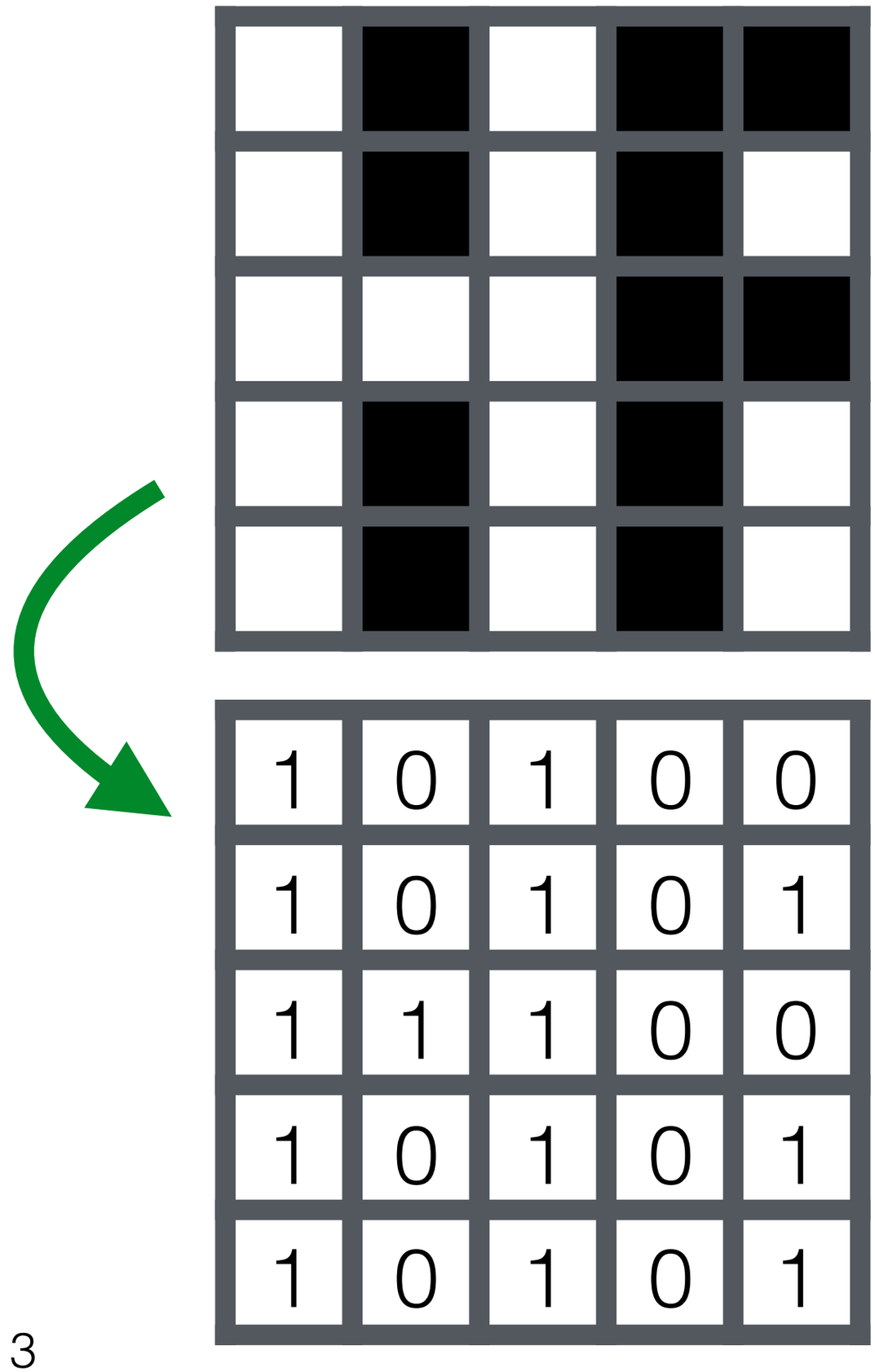
- We’ll focus on grayscale images
  - Each pixel takes a value between 0 and P
  - Here, 0: black, 1: white

Images



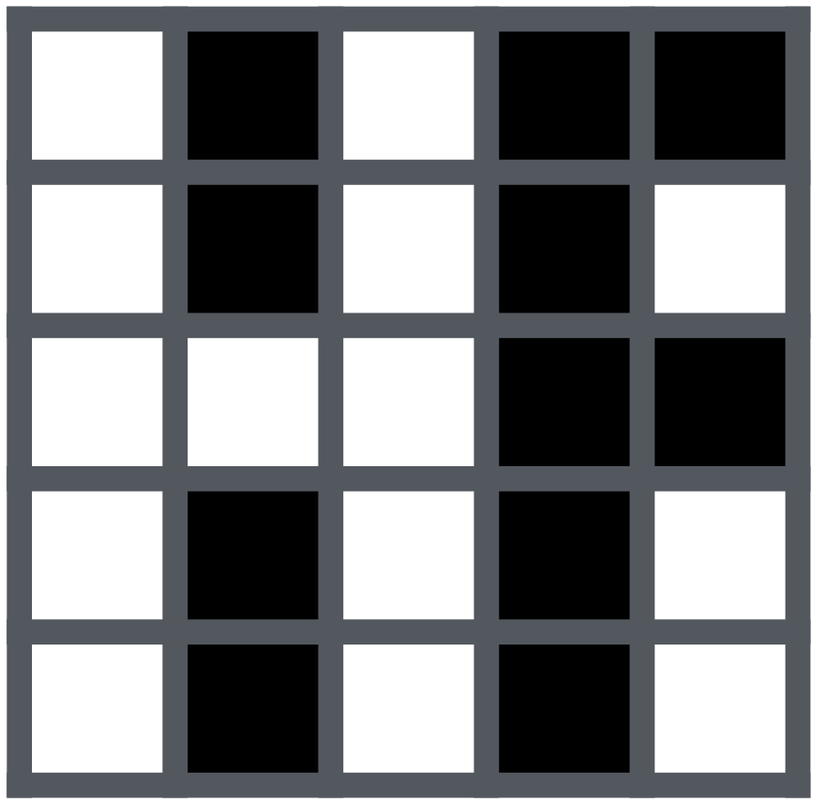
- We'll focus on grayscale images
  - Each pixel takes a value between 0 and *P*
  - Here, 0: black, 1: white
  - Larger *P* in Lab Week 08

Images



- 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

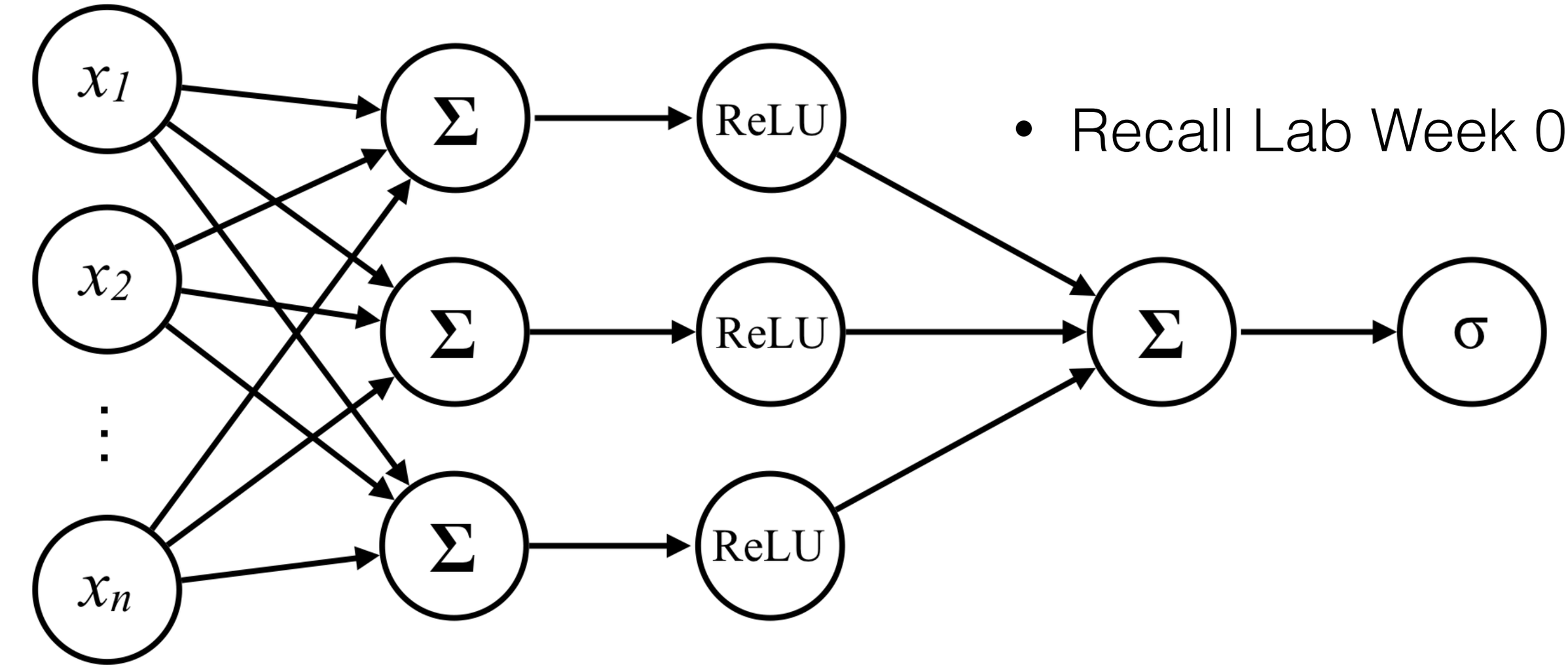
- We'll focus on grayscale images
  - Each pixel takes a value between 0 and *P*
  - Here, 0: black, 1: white
  - Larger *P* in Lab Week 08

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

Previous neural nets in this class

- Recall Lab Week 07

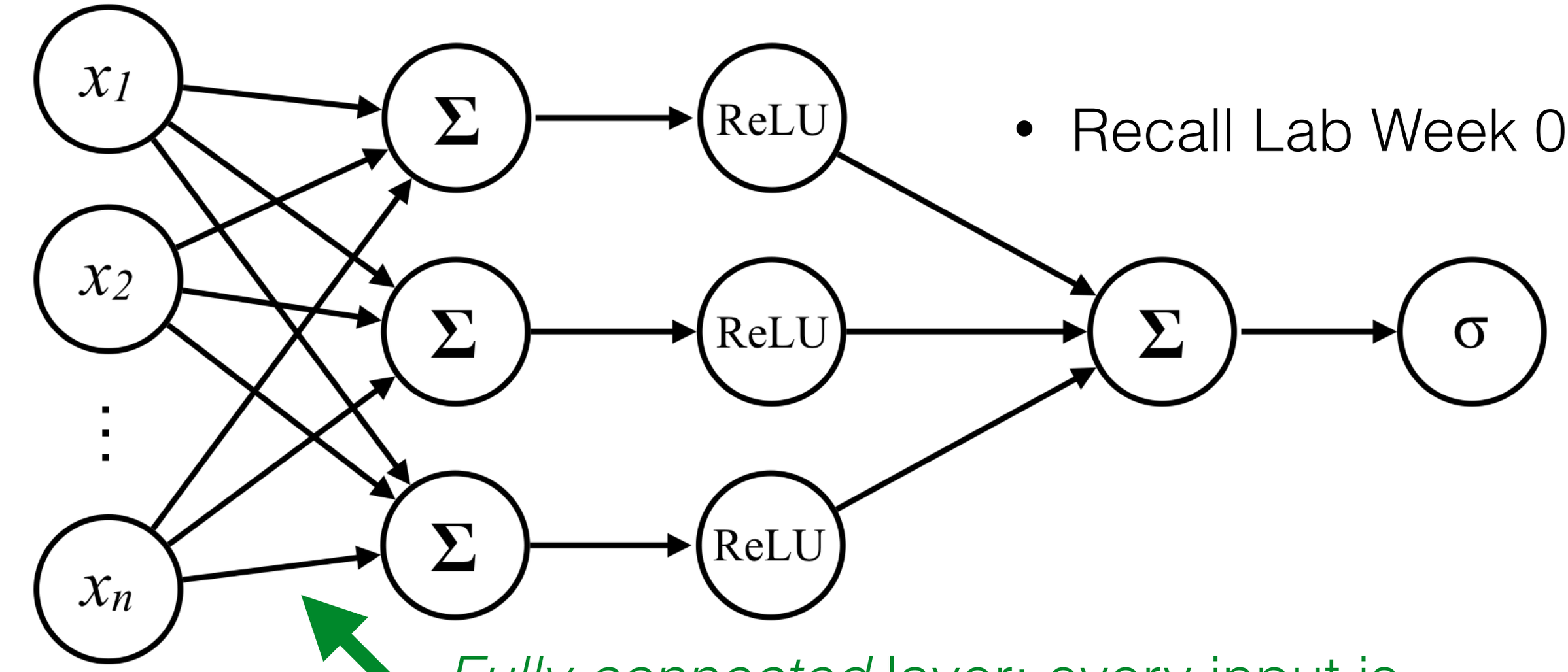
Previous neural nets in this class



- Recall Lab Week 07

input x
(n x 1)

Previous neural nets in this class

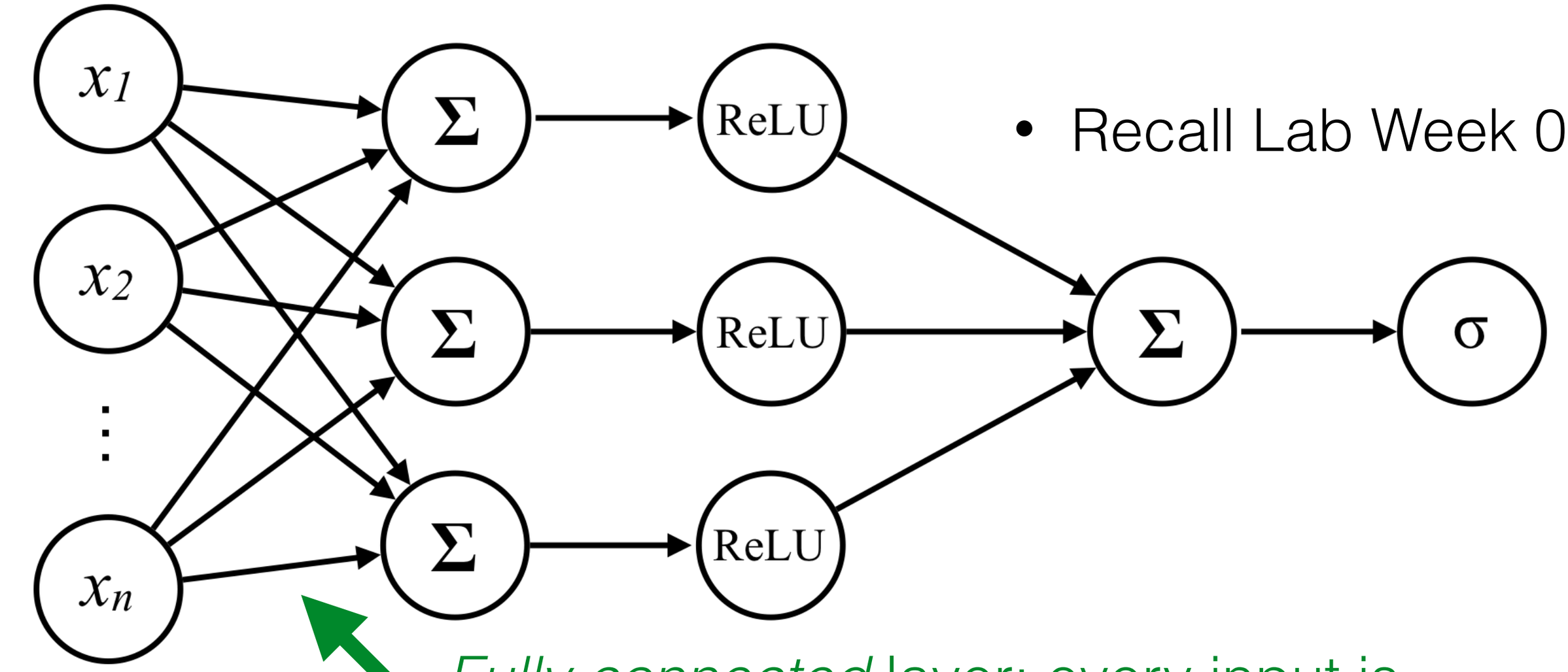


- Recall Lab Week 07

input x
(n x 1)

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

Previous neural nets in this class



- Recall Lab Week 07

input x
(n x 1)

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

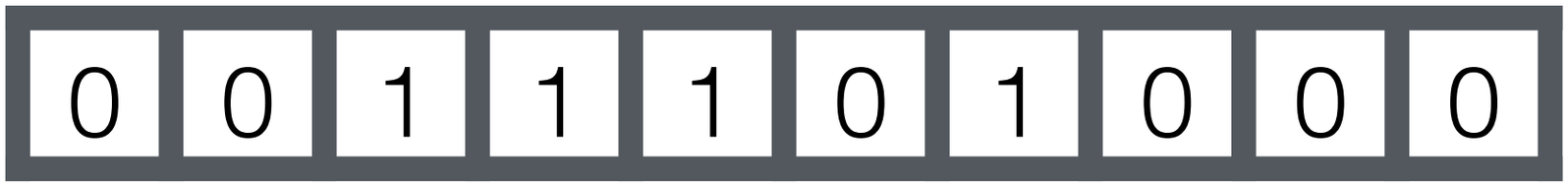
But we know more about images:

- Spatial locality
- Translation invariance

Convolutional Layer: 1D example

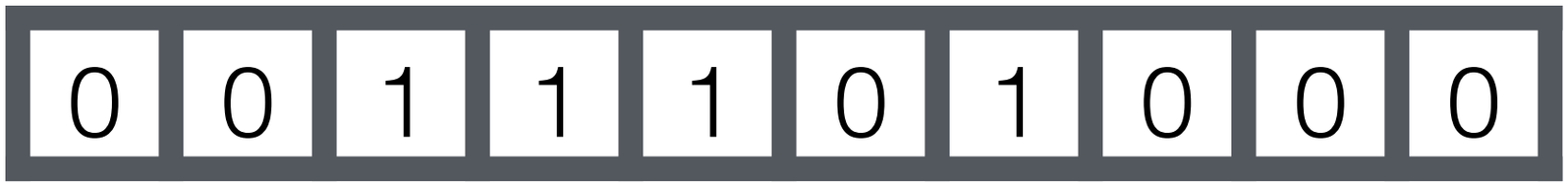
Convolutional Layer: 1D example

A 1D image:

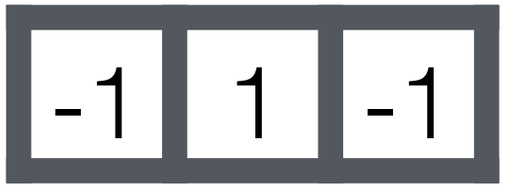


Convolutional Layer: 1D example

A 1D image:



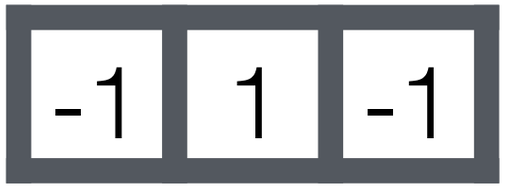
A filter:



Convolutional Layer: 1D example

A 1D image:

A filter:



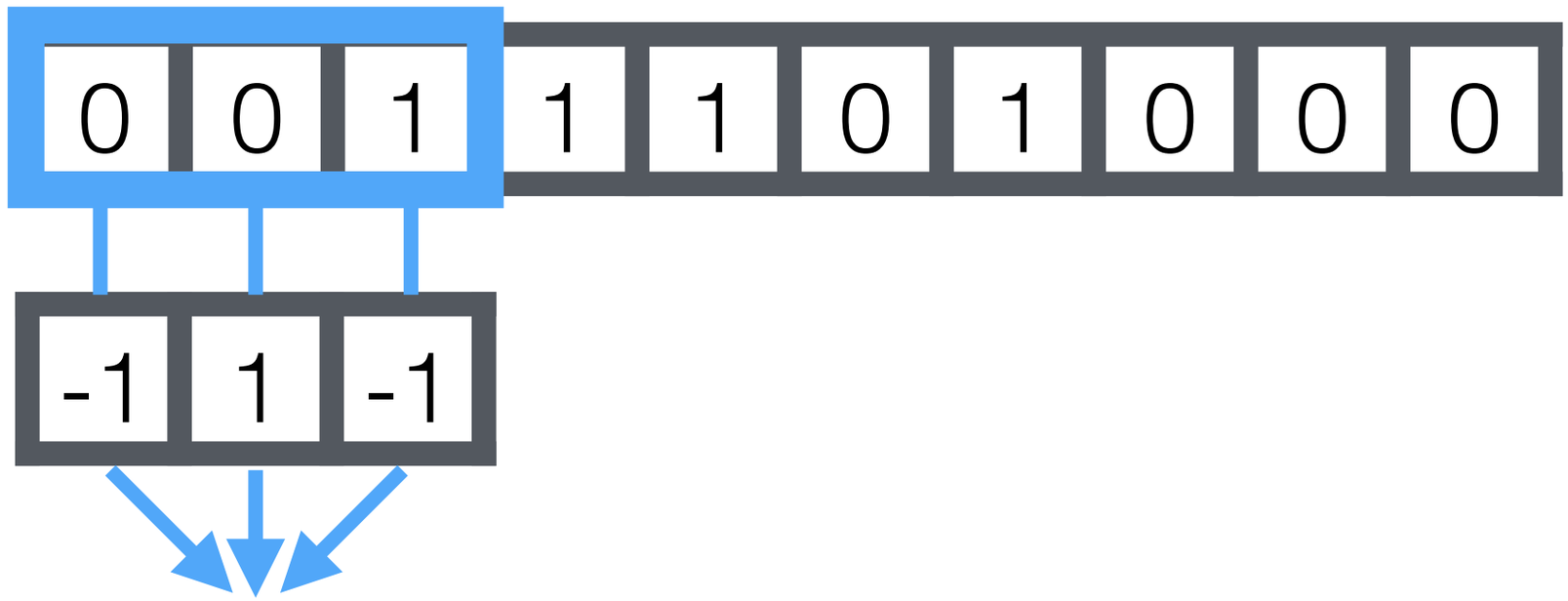
After convolution*:

Convolutional Layer: 1D example

A 1D image:

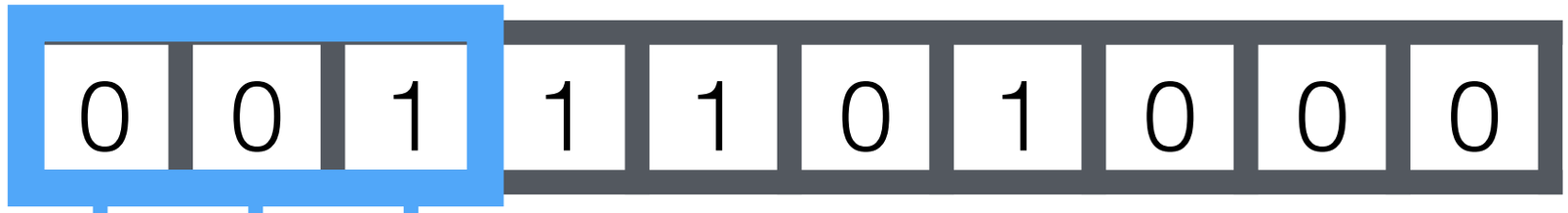
A filter:

After convolution*:

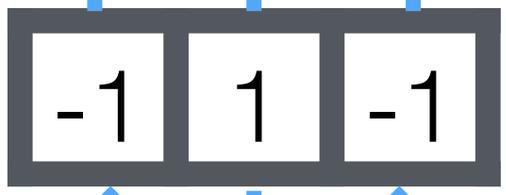


Convolutional Layer: 1D example

A 1D image:



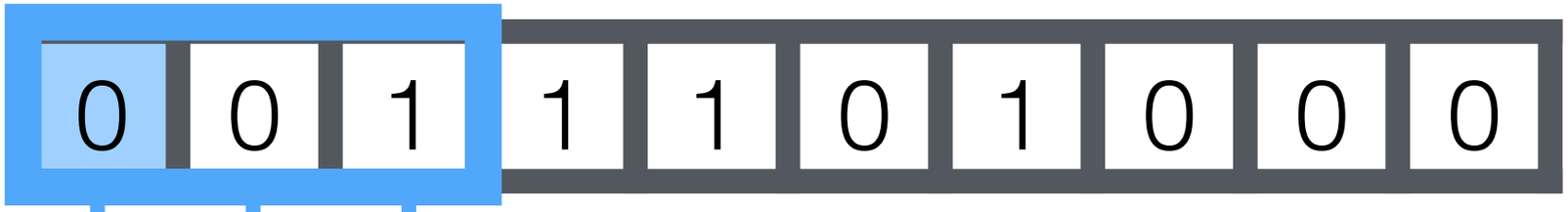
A filter:



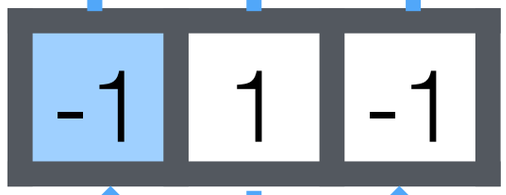
After convolution*:

Convolutional Layer: 1D example

A 1D image:



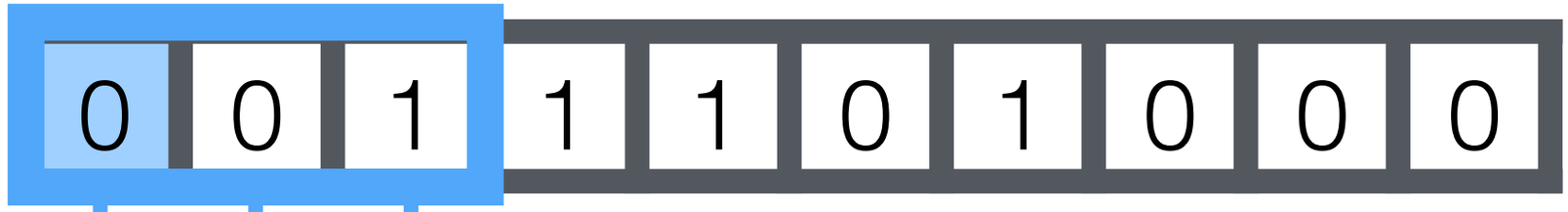
A filter:



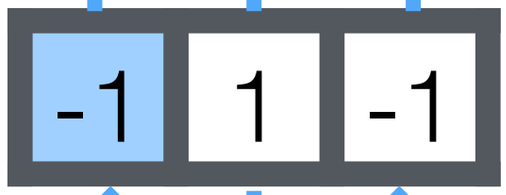
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

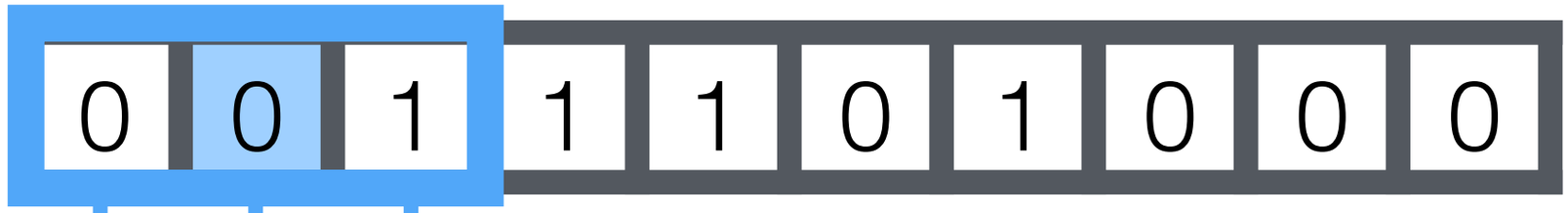


0 * -1

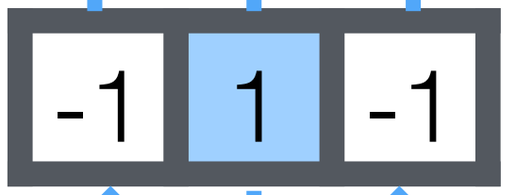
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

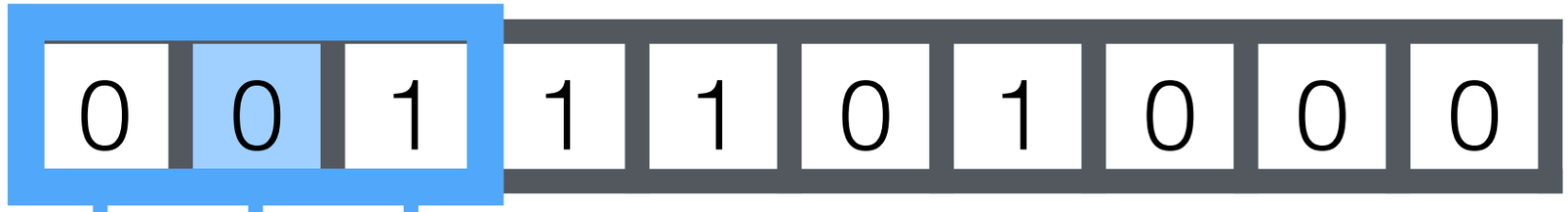


0 * -1

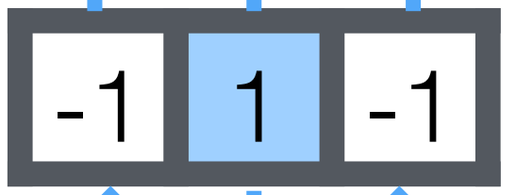
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

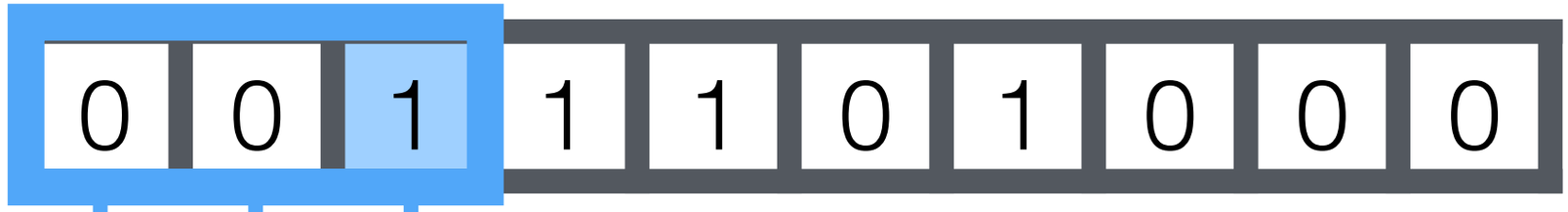


0 * -1 + 0 * 1

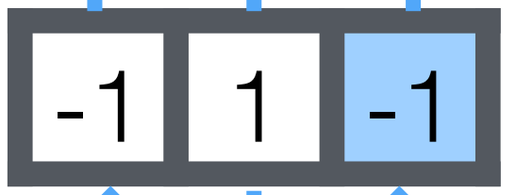
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

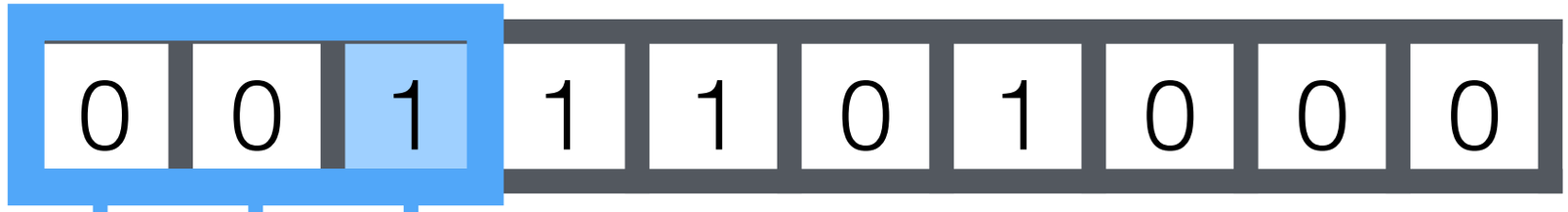


0 * -1 + 0 * 1

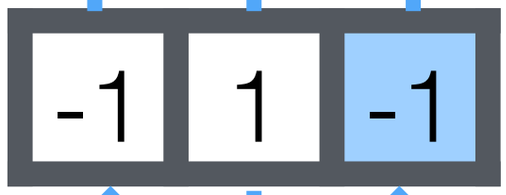
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

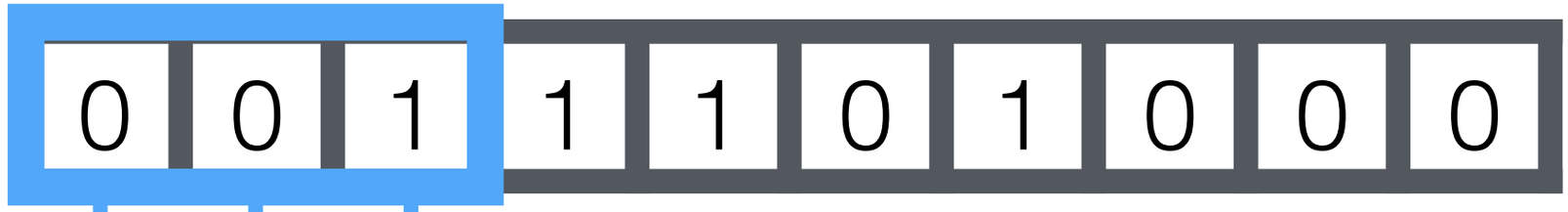


0 * -1 + 0 * 1 + 1 * -1

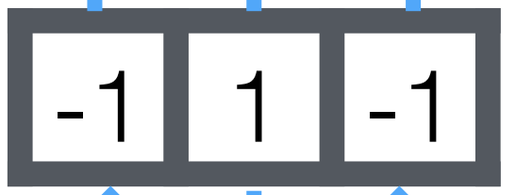
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

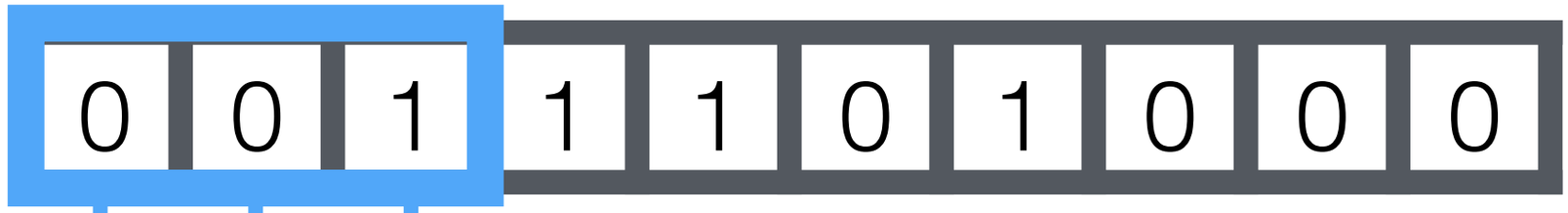


0 * -1 + 0 * 1 + 1 * -1

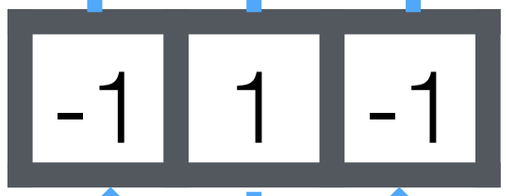
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

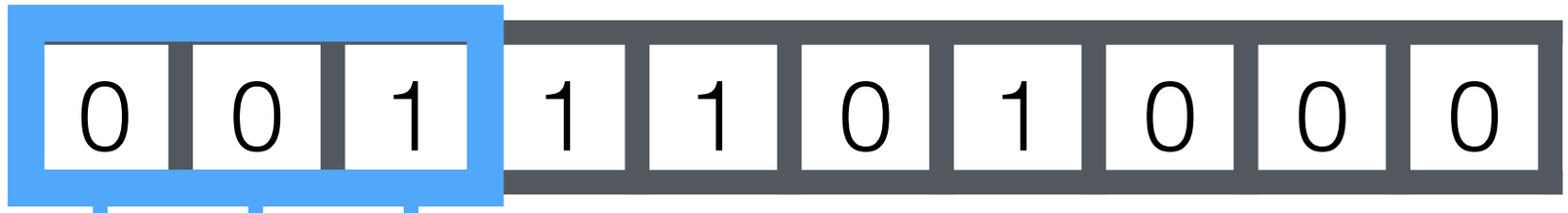


0 * -1 + 0 * 1 + 1 * -1 = -1

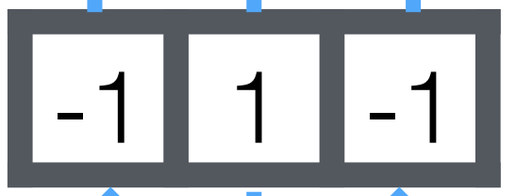
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:

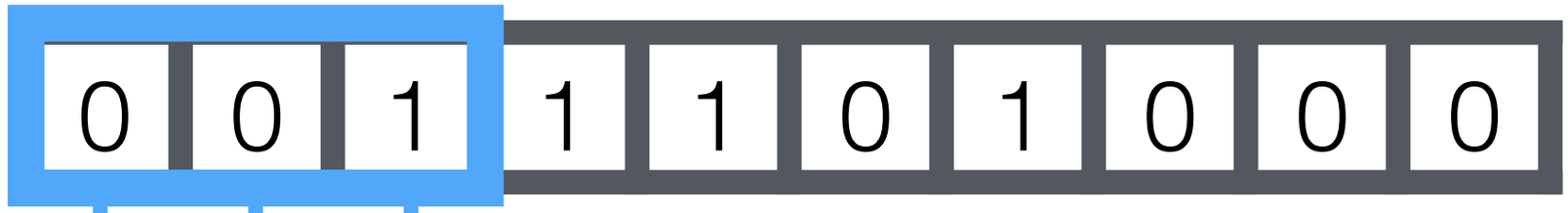


0 * -1 + 0 * 1 + 1 * -1 = -1

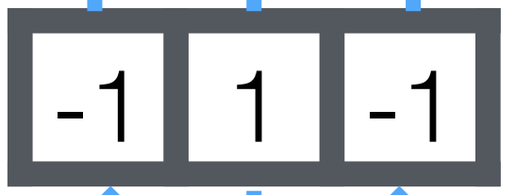
After convolution*:

Convolutional Layer: 1D example

A 1D image:



A filter:



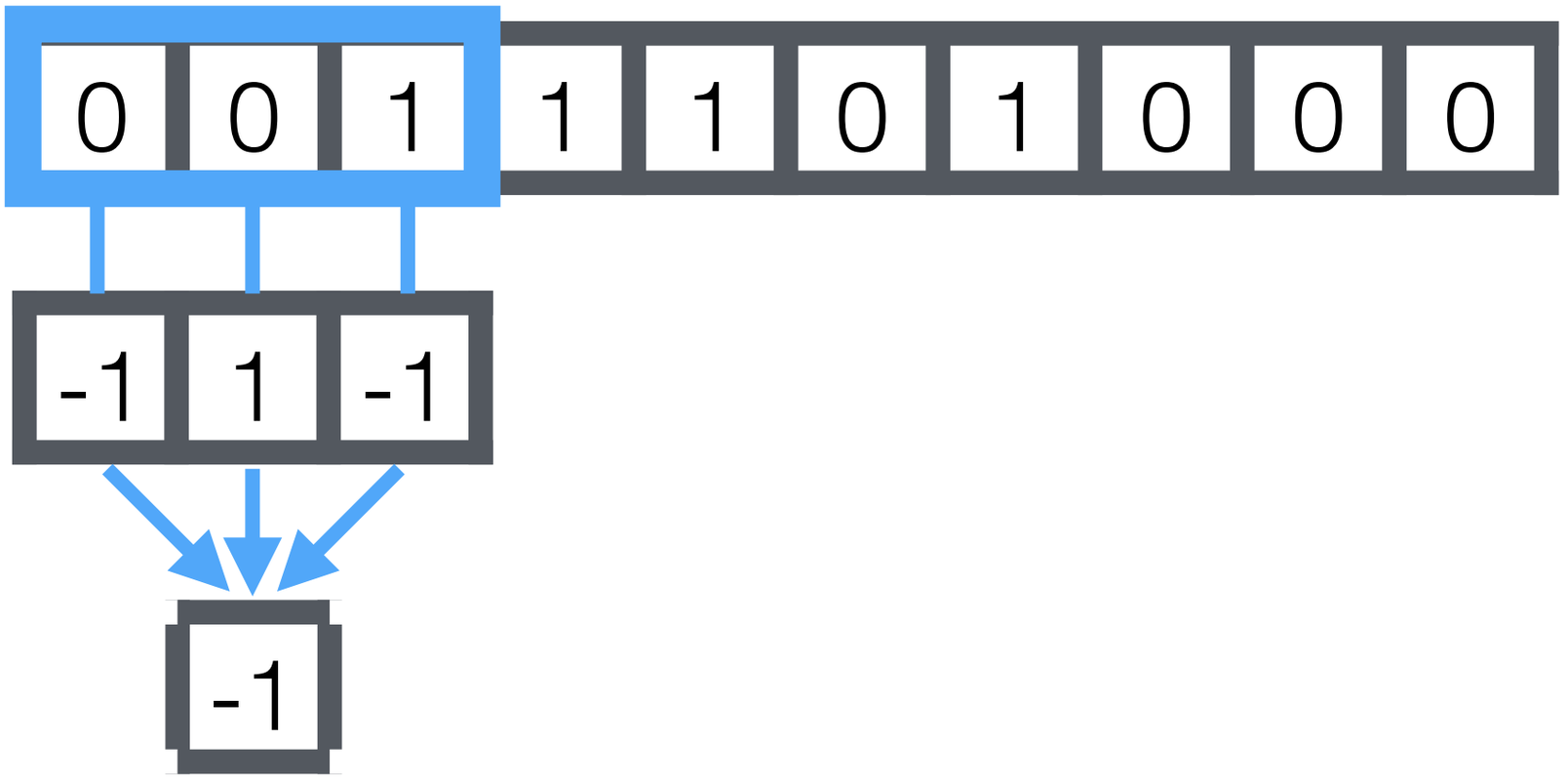
After convolution*:

Convolutional Layer: 1D example

A 1D image:

A filter:

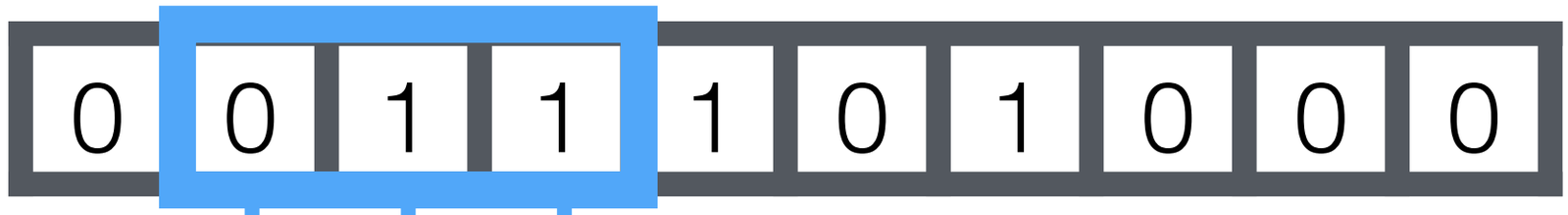
After convolution*:



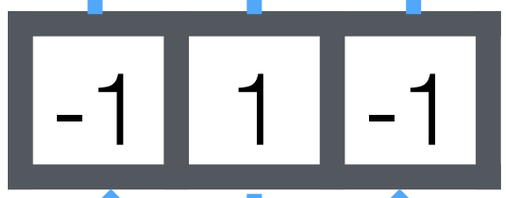
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



After convolution*:

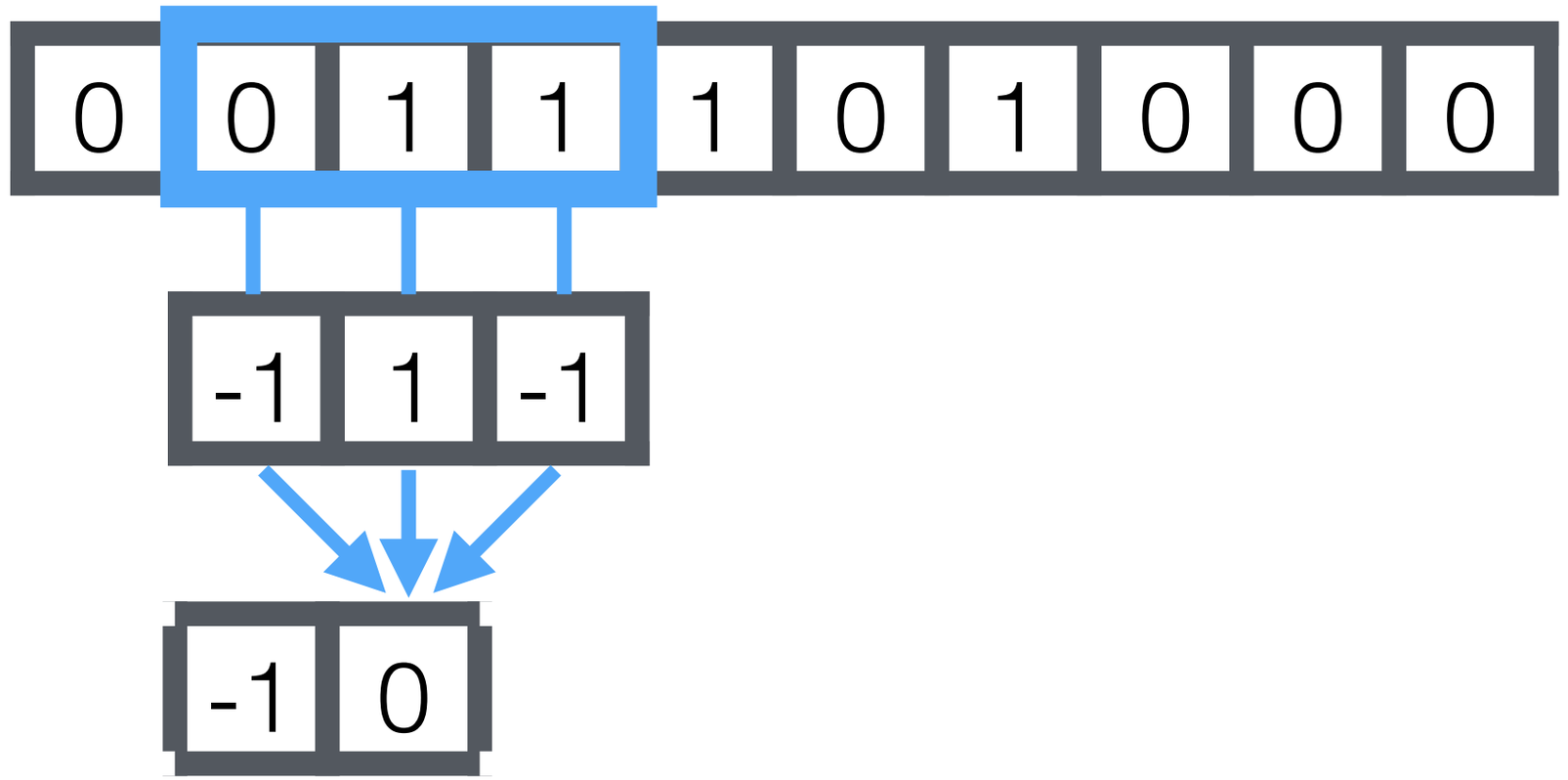
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

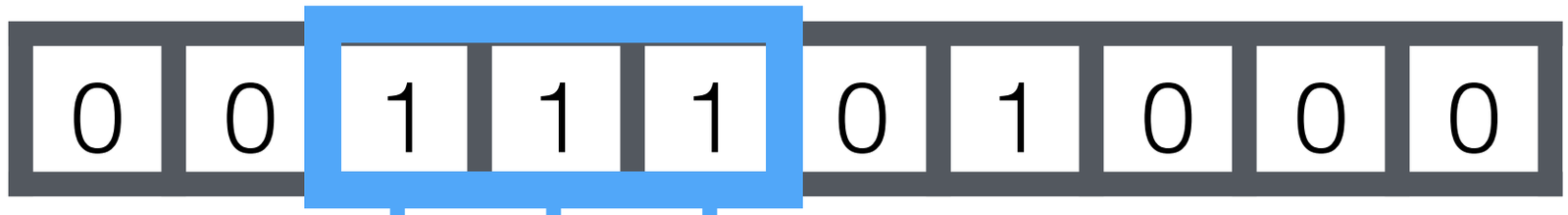
After convolution*:



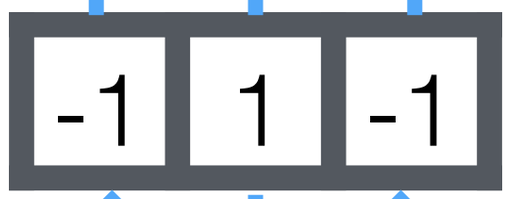
*correlation

Convolutional Layer: 1D example

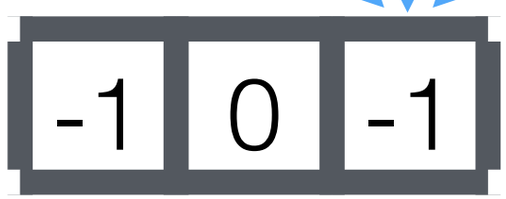
A 1D image:



A filter:



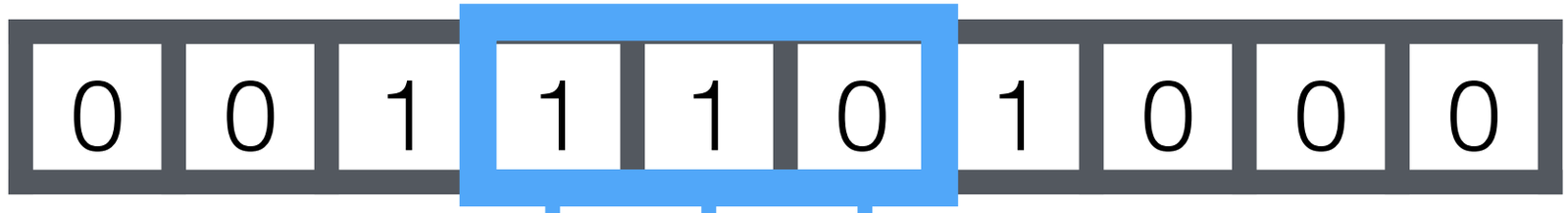
After convolution*:



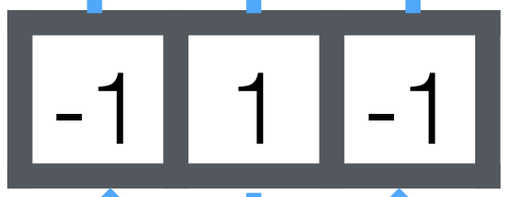
*correlation

Convolutional Layer: 1D example

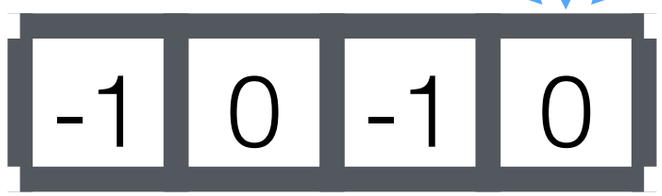
A 1D image:



A filter:



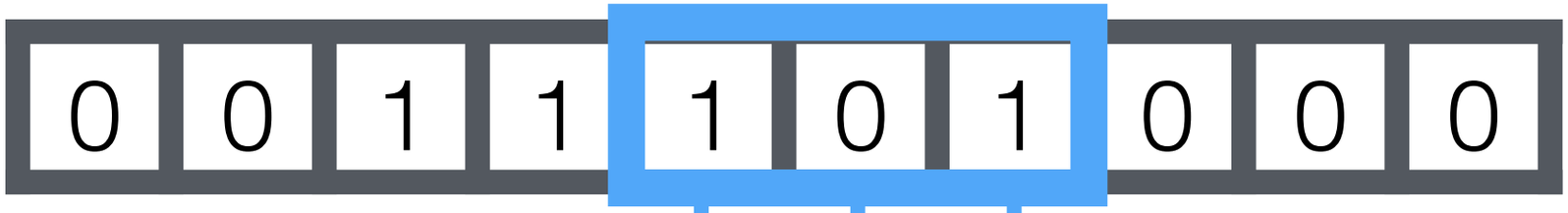
After convolution*:



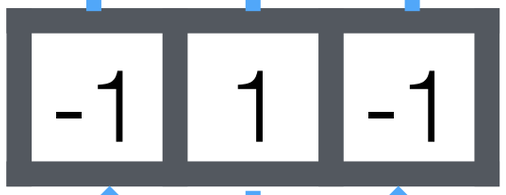
*correlation

Convolutional Layer: 1D example

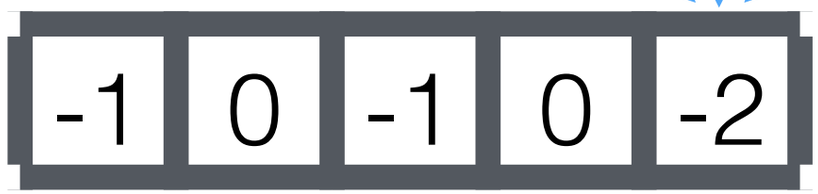
A 1D image:



A filter:



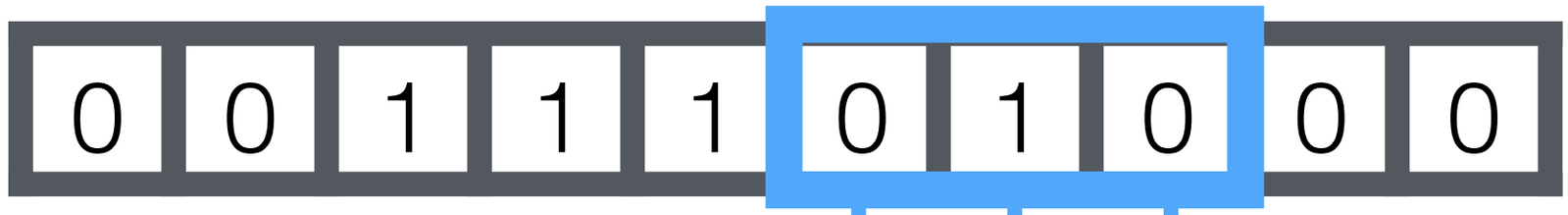
After convolution*:



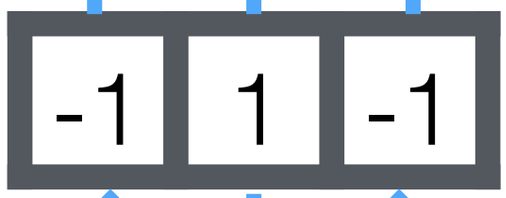
*correlation

Convolutional Layer: 1D example

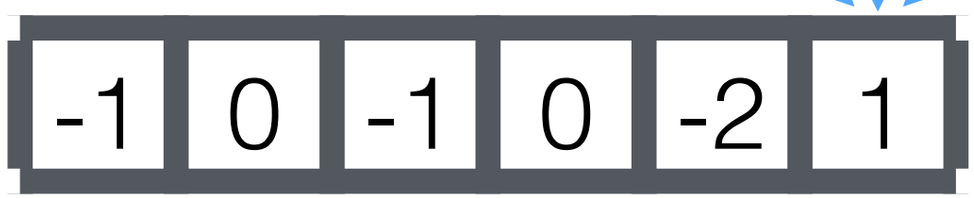
A 1D image:



A filter:



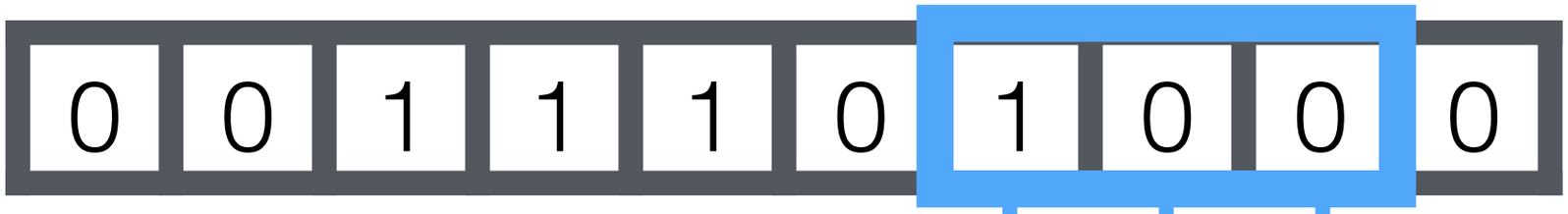
After convolution*:



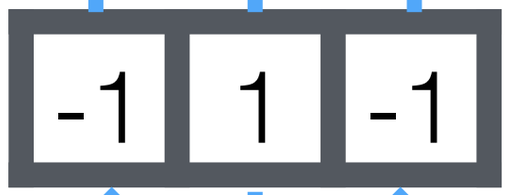
*correlation

Convolutional Layer: 1D example

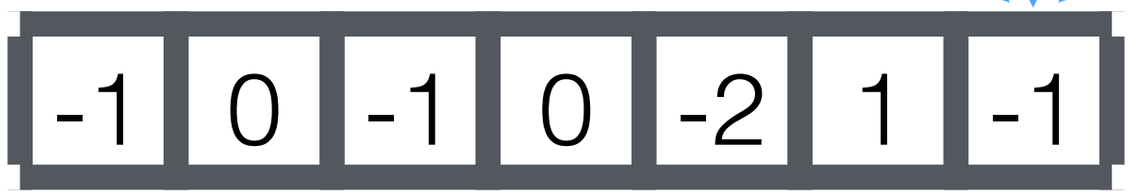
A 1D image:



A filter:



After convolution*:



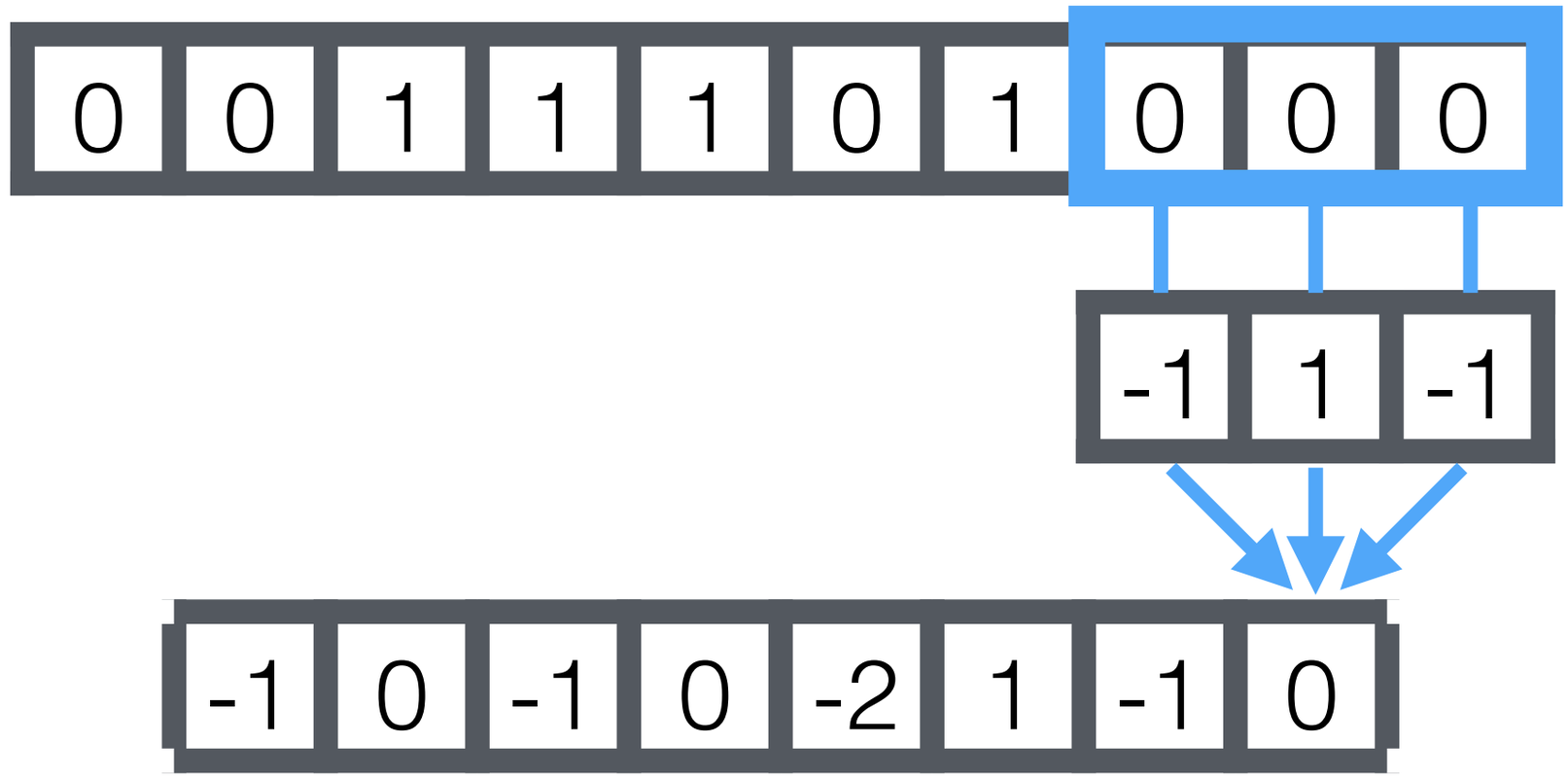
*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:

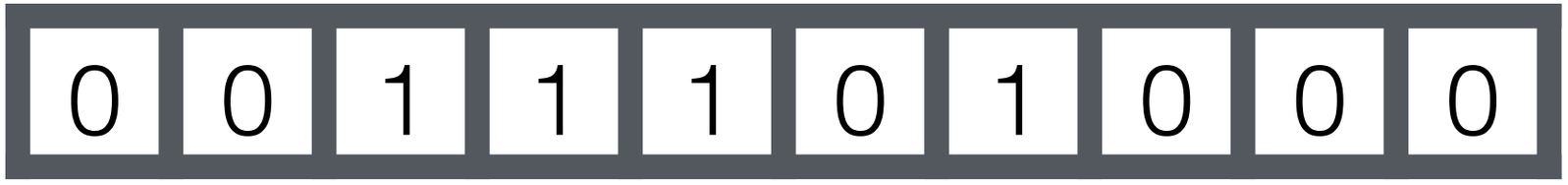
After convolution*:



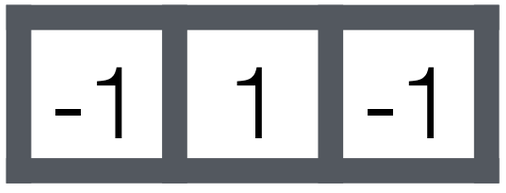
*correlation

Convolutional Layer: 1D example

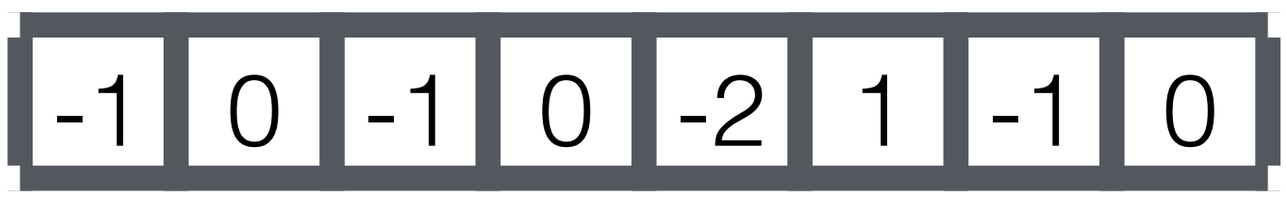
A 1D image:



A filter:



After convolution*:

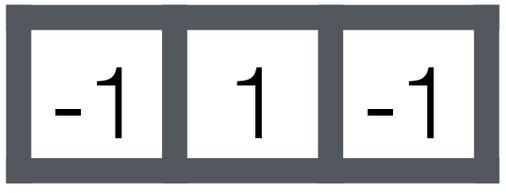


*correlation

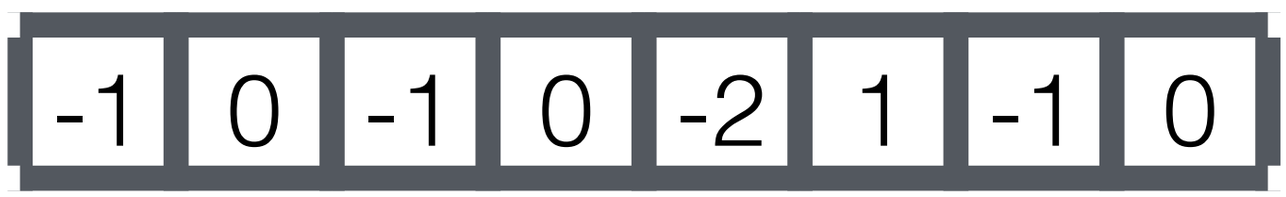
Convolutional Layer: 1D example

A 1D image:

A filter:



After convolution*:

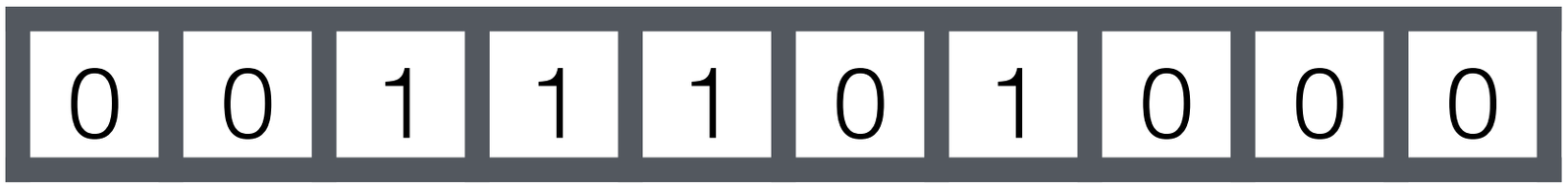


After ReLU:

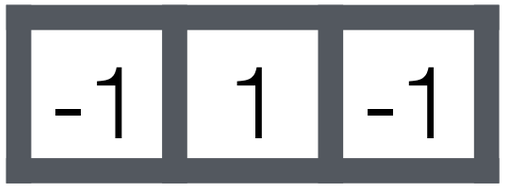
*correlation

Convolutional Layer: 1D example

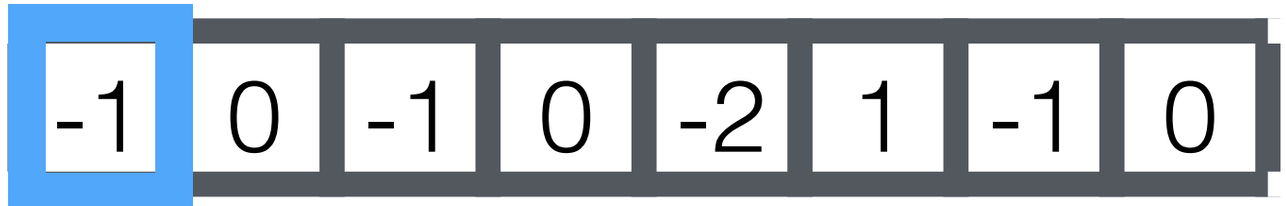
A 1D image:



A filter:



After convolution*:

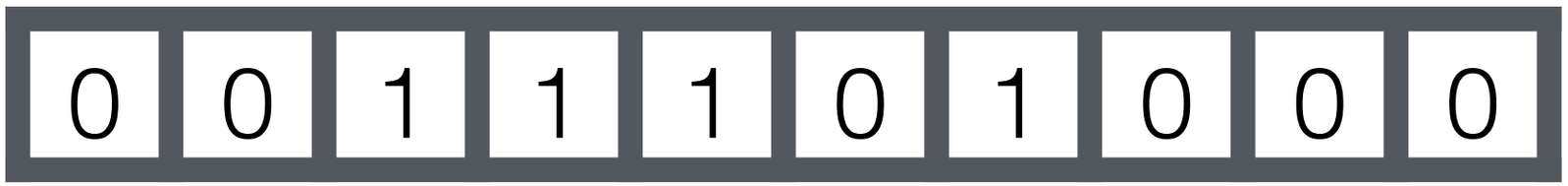


After ReLU:

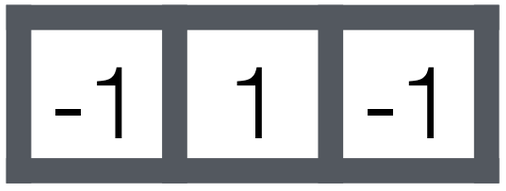
*correlation

Convolutional Layer: 1D example

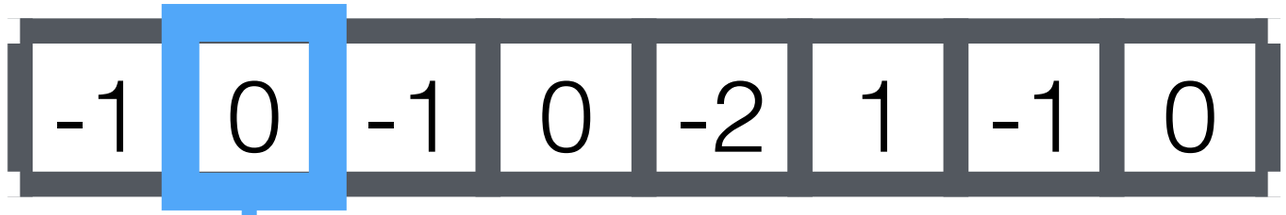
A 1D image:



A filter:



After convolution*:

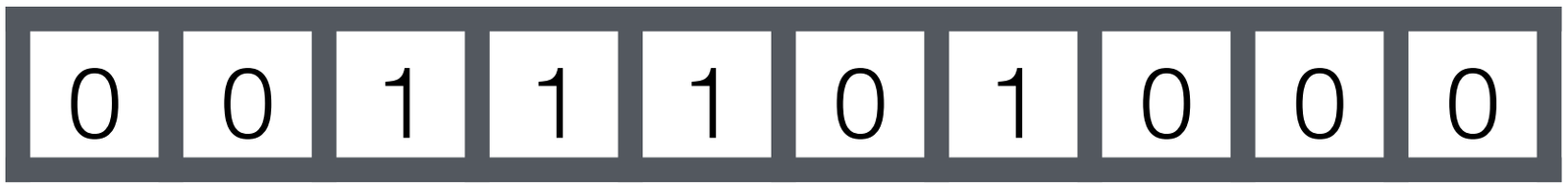


After ReLU:

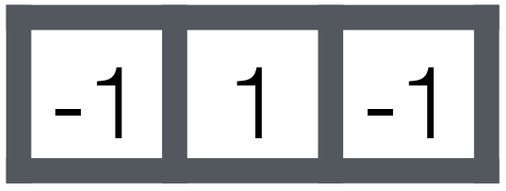
*correlation

Convolutional Layer: 1D example

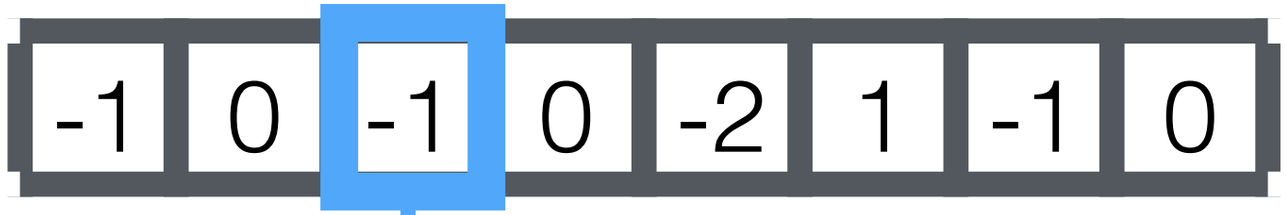
A 1D image:



A filter:



After convolution*:

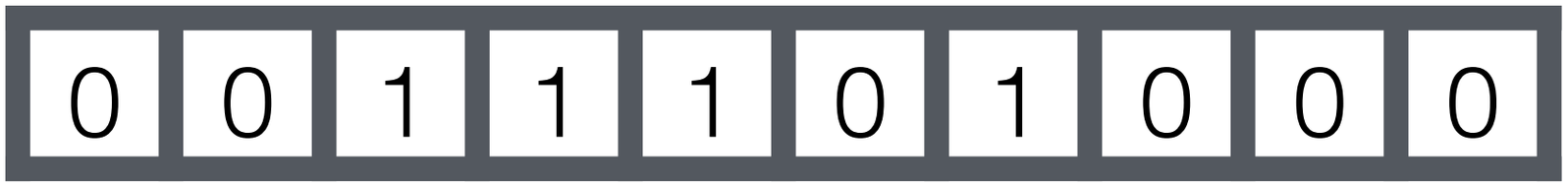


After ReLU:

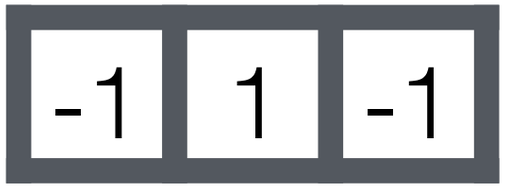
*correlation

Convolutional Layer: 1D example

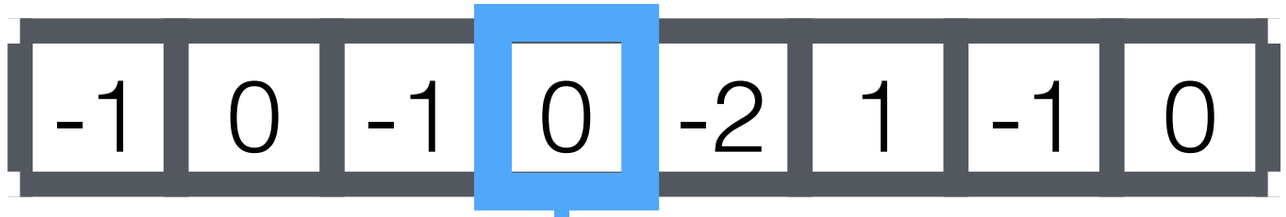
A 1D image:



A filter:



After convolution*:

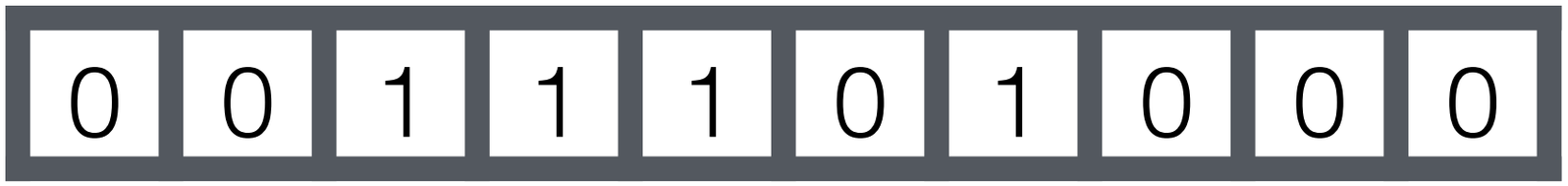


After ReLU:

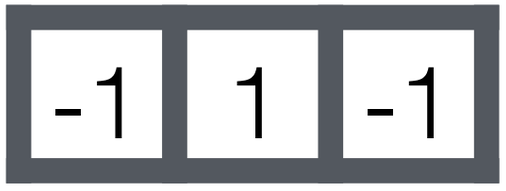
*correlation

Convolutional Layer: 1D example

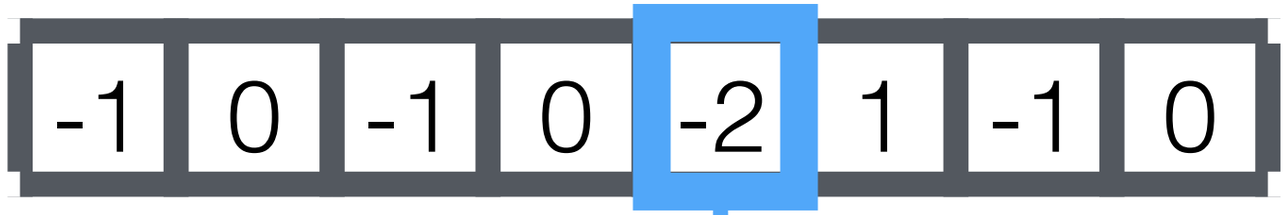
A 1D image:



A filter:



After convolution*:

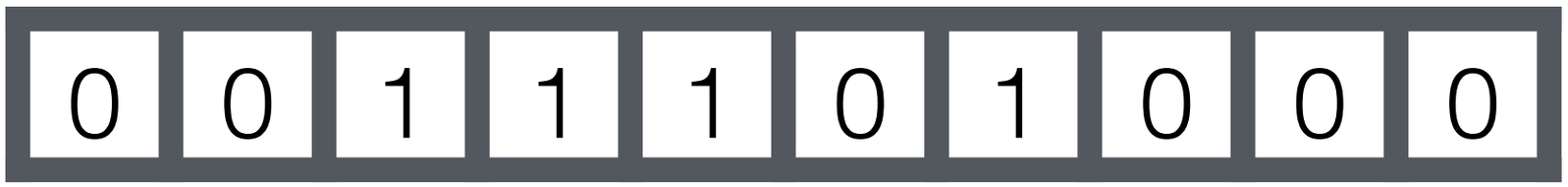


After ReLU:

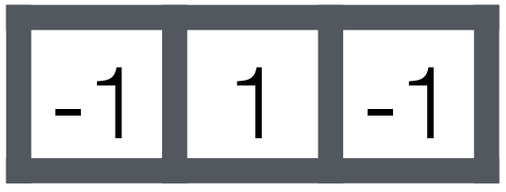
*correlation

Convolutional Layer: 1D example

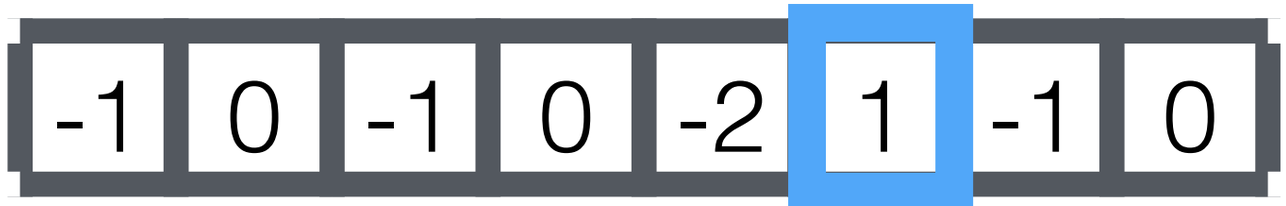
A 1D image:



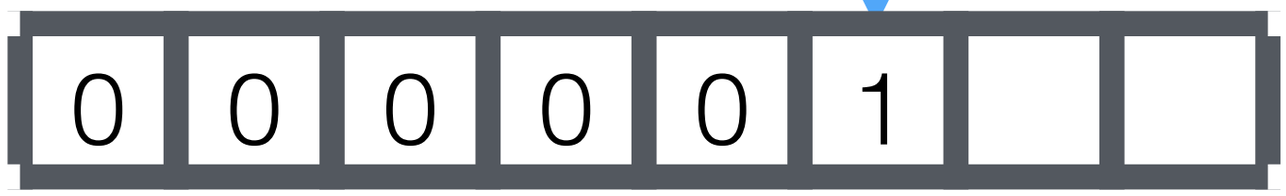
A filter:



After convolution*:



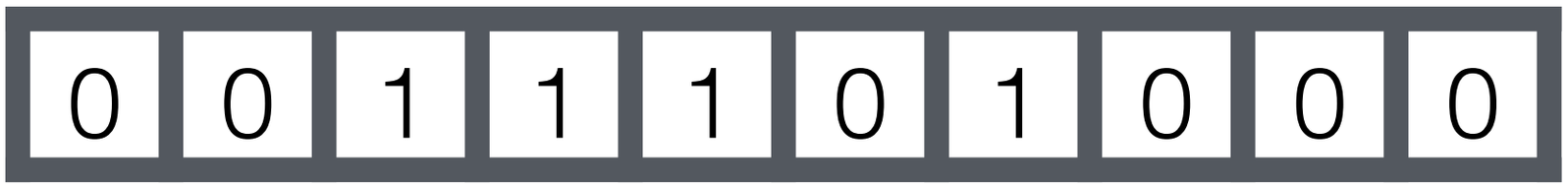
After ReLU:



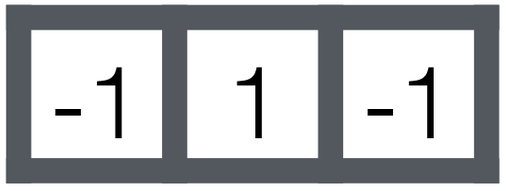
*correlation

Convolutional Layer: 1D example

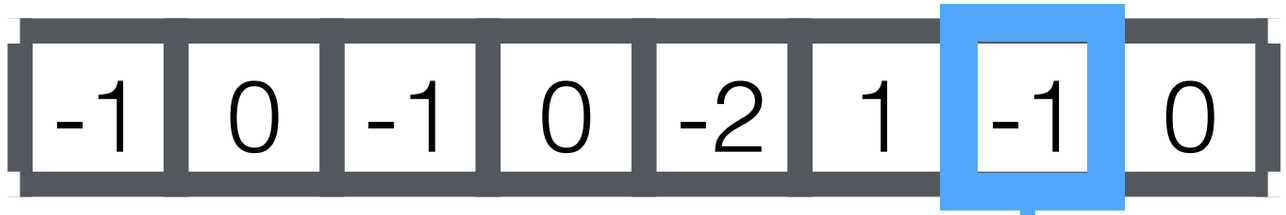
A 1D image:



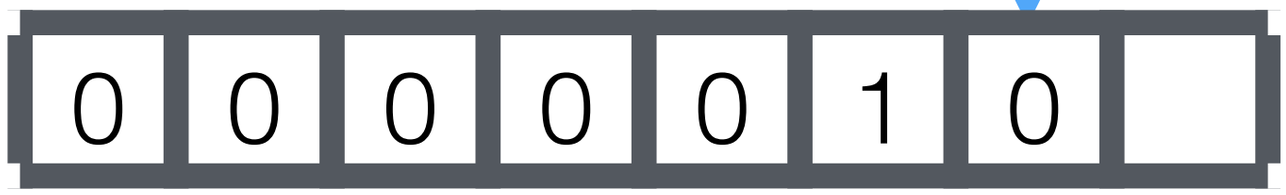
A filter:



After convolution*:



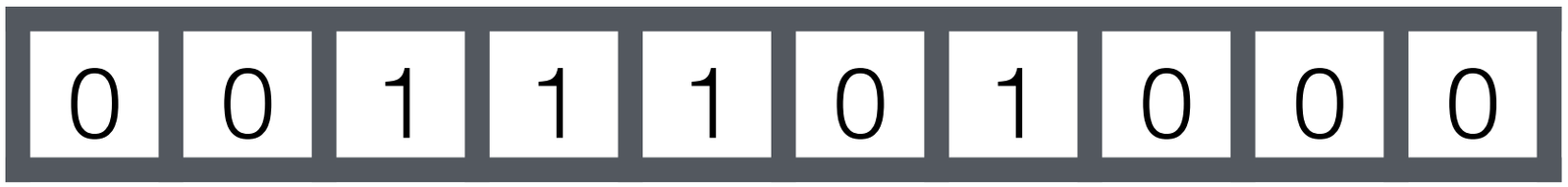
After ReLU:



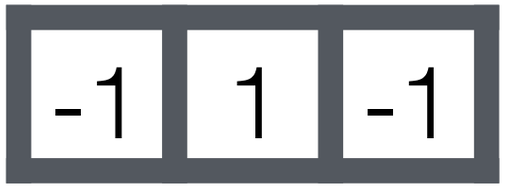
*correlation

Convolutional Layer: 1D example

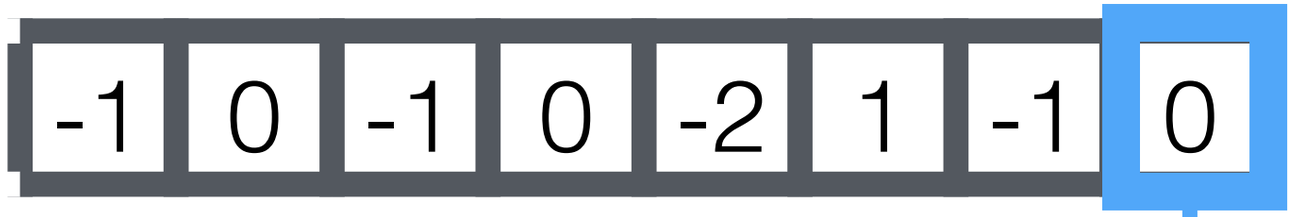
A 1D image:



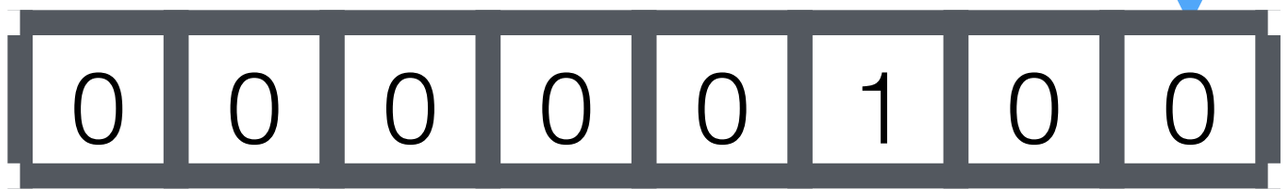
A filter:



After convolution*:



After ReLU:

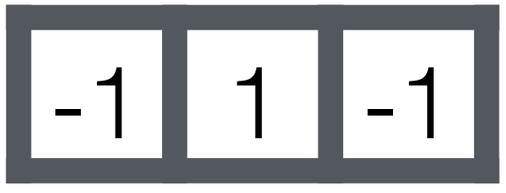


*correlation

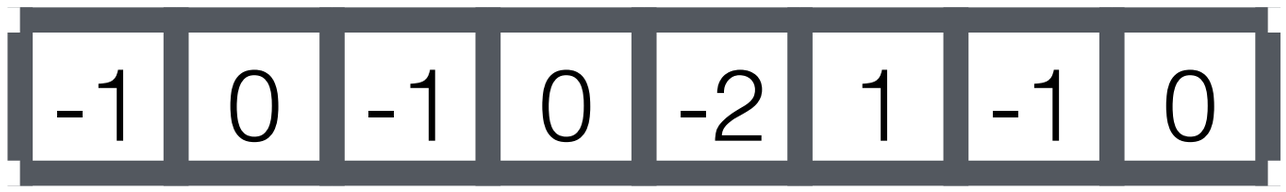
Convolutional Layer: 1D example

A 1D image:

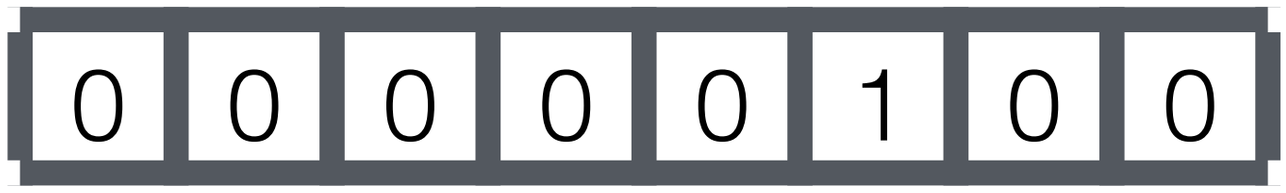
A filter:



After convolution*:



After ReLU:

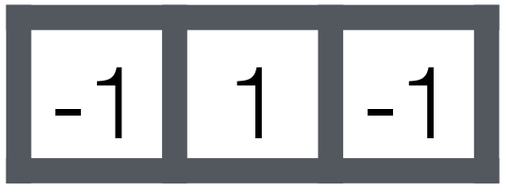


*correlation

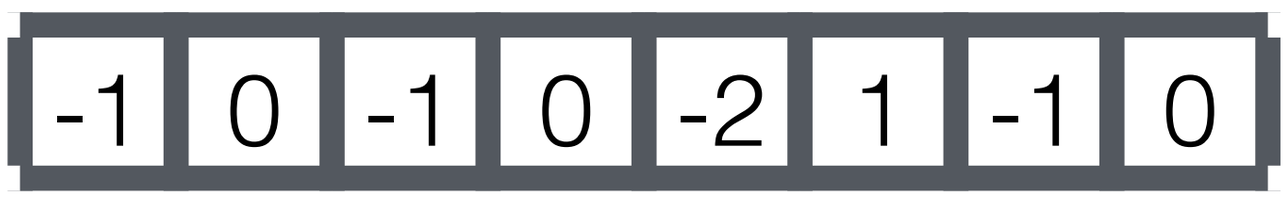
Convolutional Layer: 1D example

A 1D image:

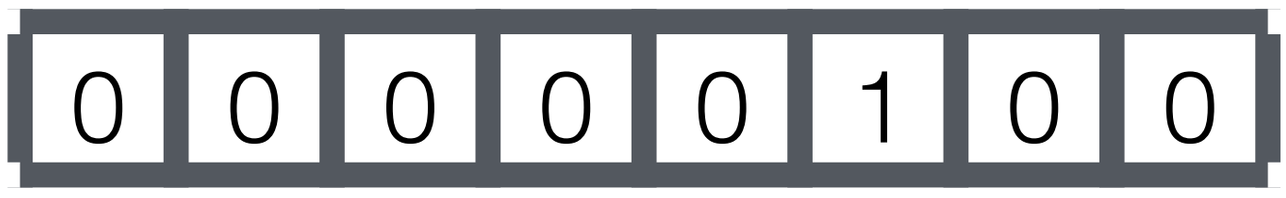
A filter:



After convolution*:



After ReLU:



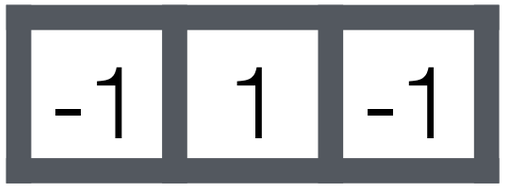
What does the filter do?

*correlation

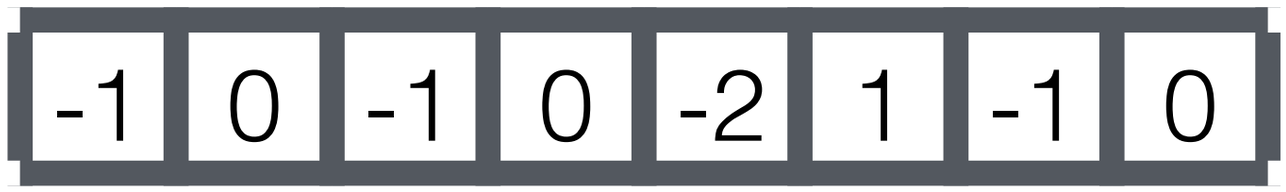
Convolutional Layer: 1D example

A 1D image:

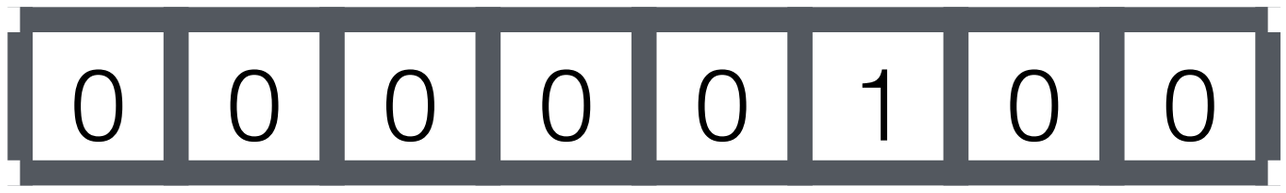
A filter:



After convolution*:



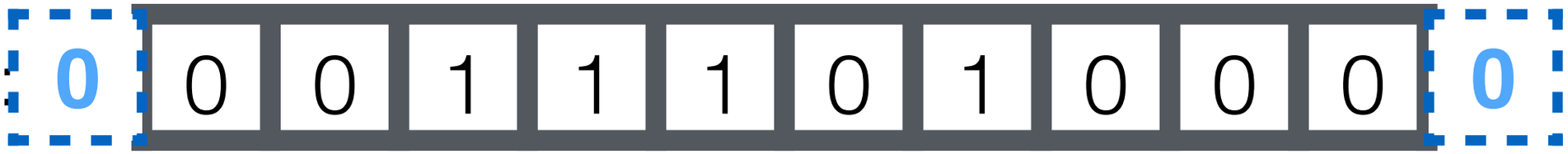
After ReLU:



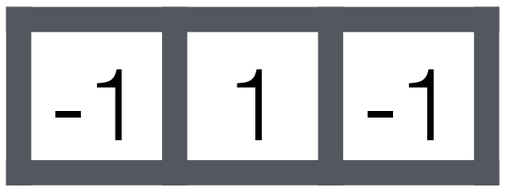
*correlation

Convolutional Layer: 1D example

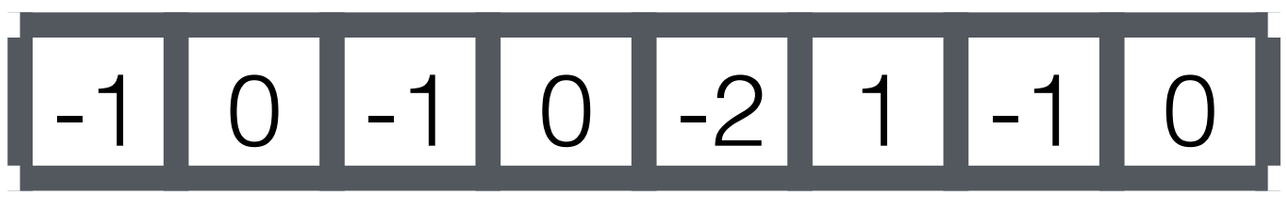
A 1D image:



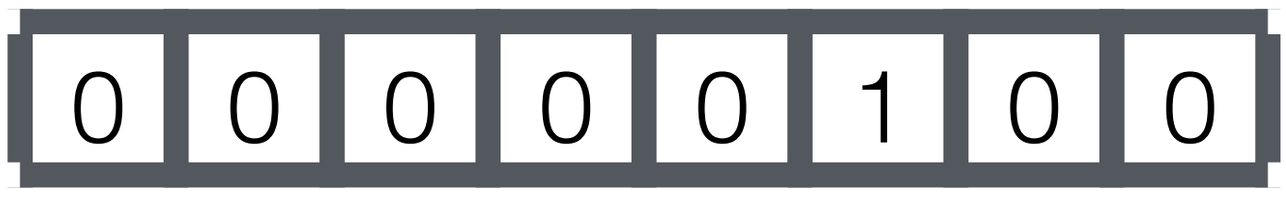
A filter:



After convolution*:

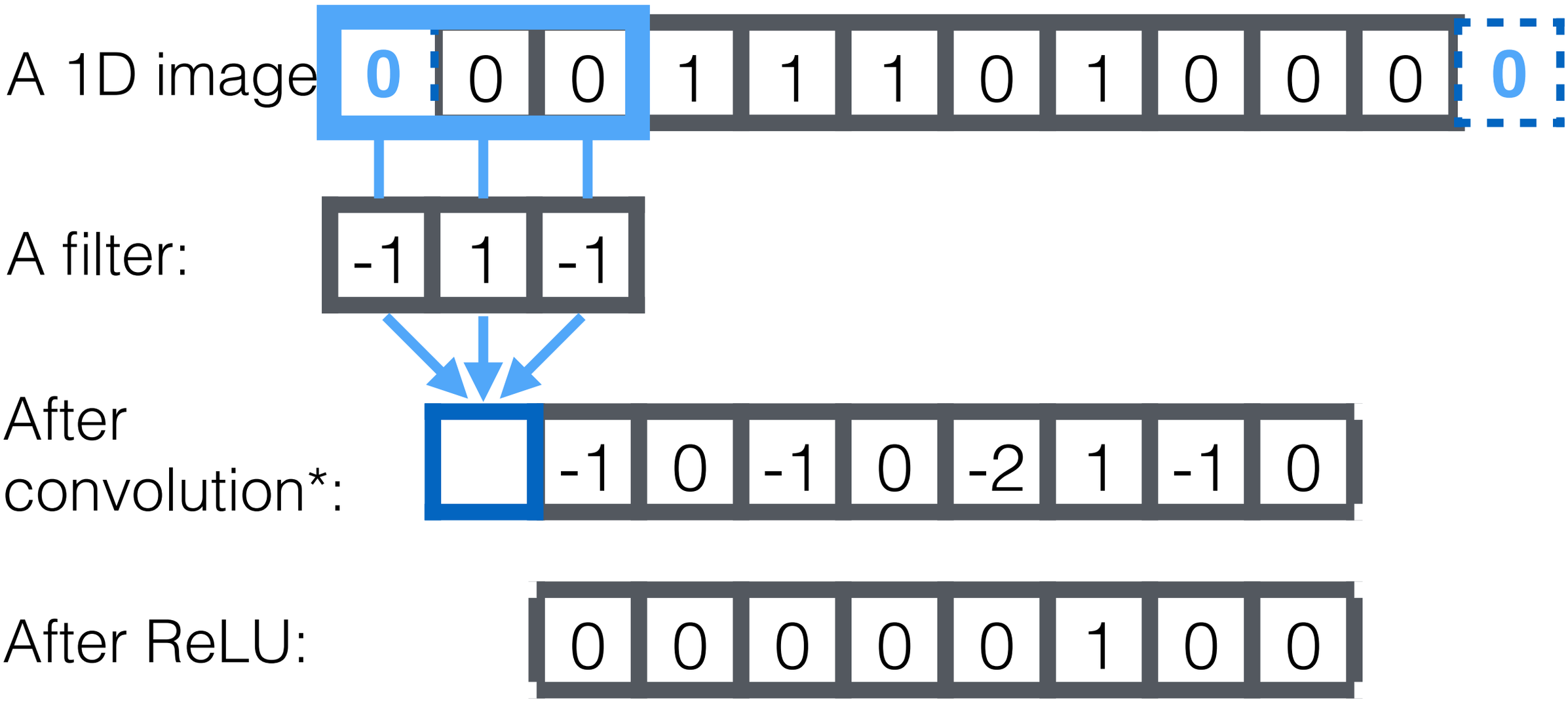


After ReLU:



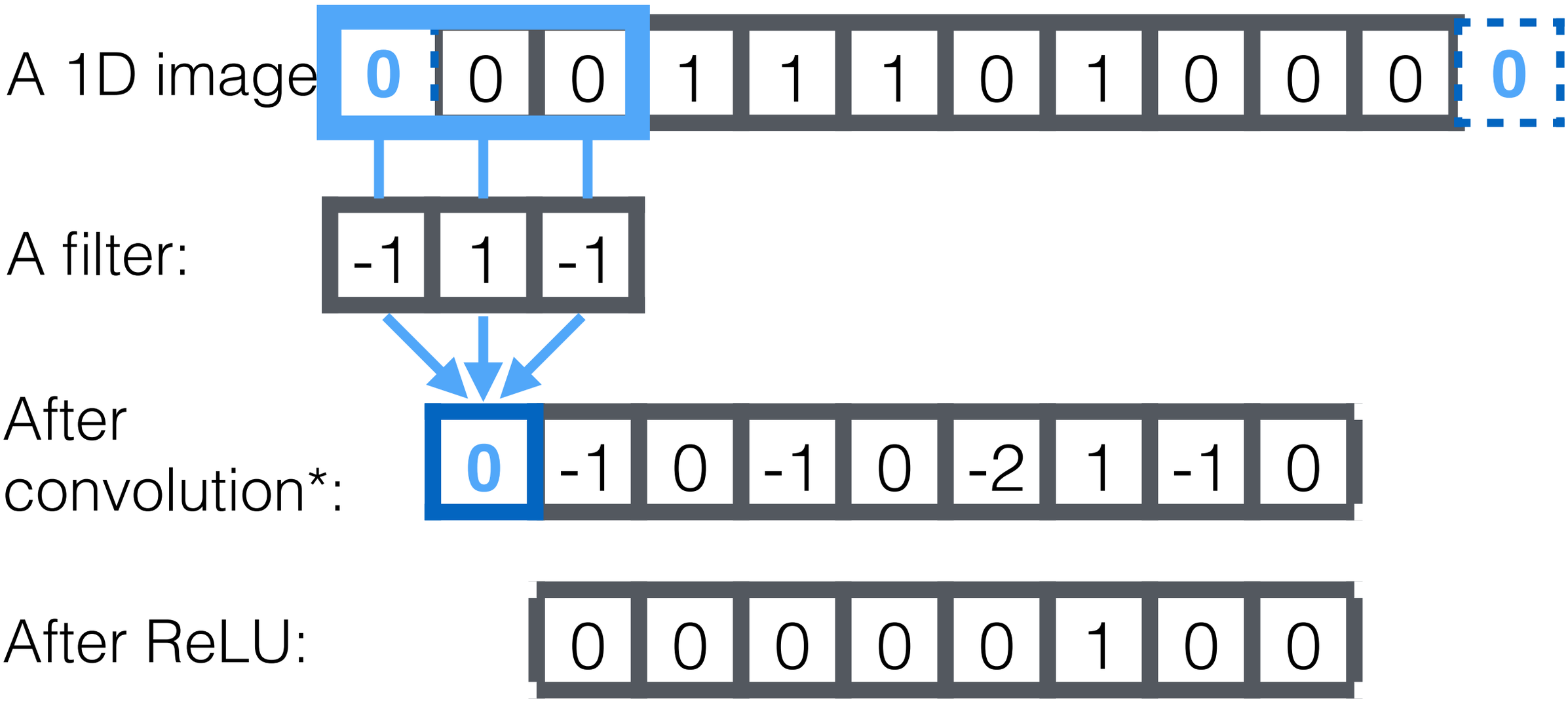
*correlation

Convolutional Layer: 1D example



*correlation

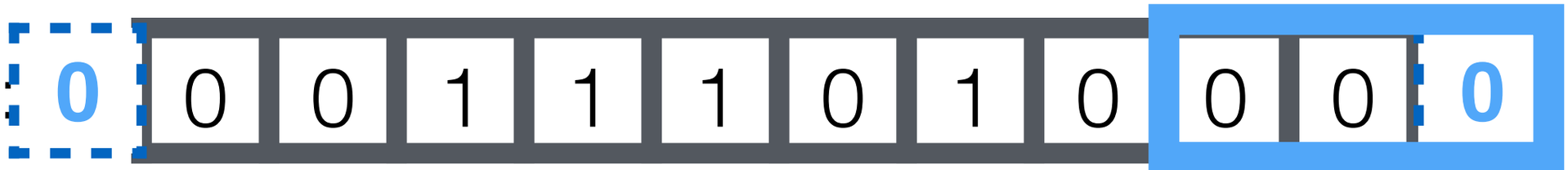
Convolutional Layer: 1D example



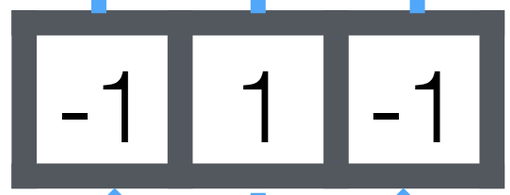
*correlation

Convolutional Layer: 1D example

A 1D image:

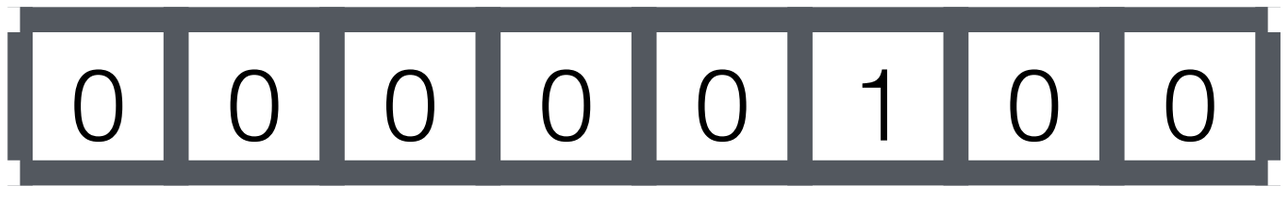


A filter:



After convolution*:

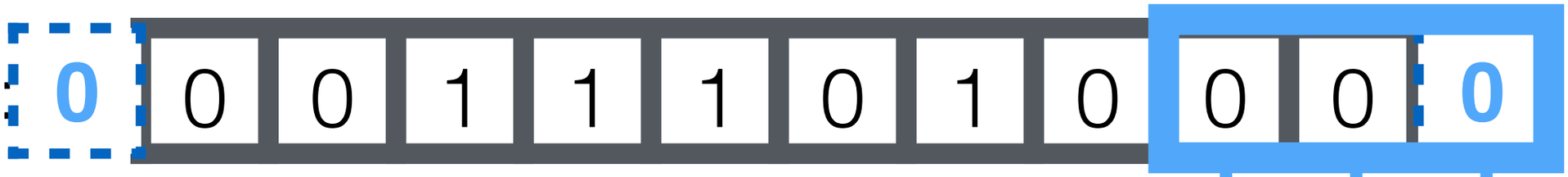
After ReLU:



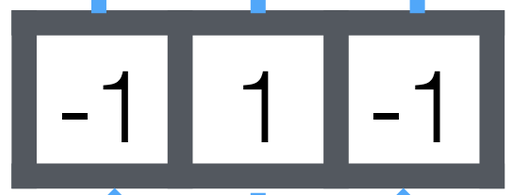
*correlation

Convolutional Layer: 1D example

A 1D image:

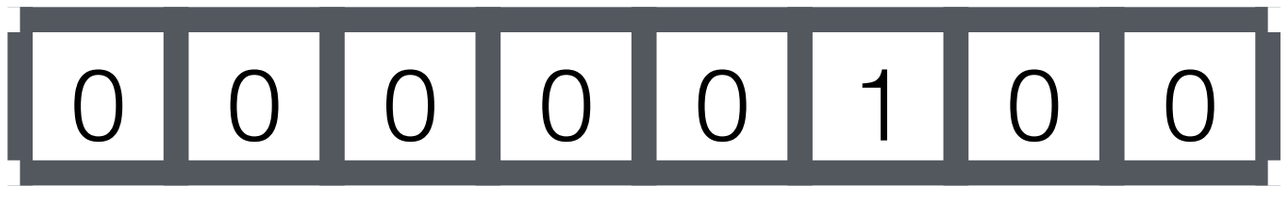


A filter:



After convolution*:

After ReLU:

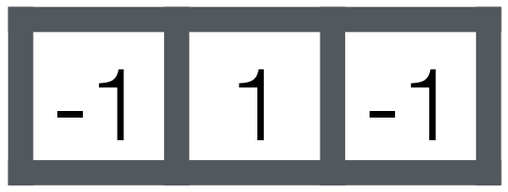


*correlation

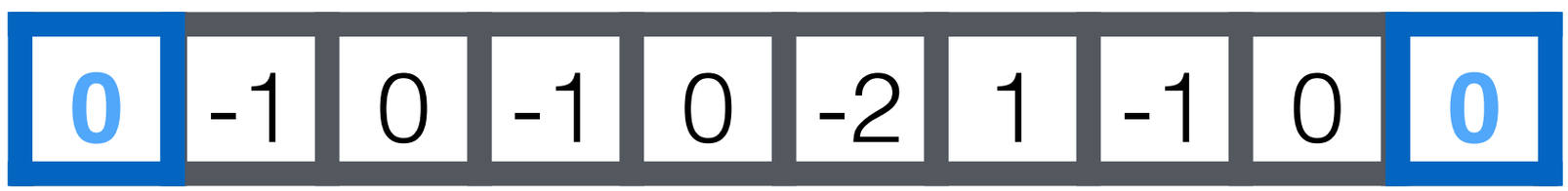
Convolutional Layer: 1D example

A 1D image:

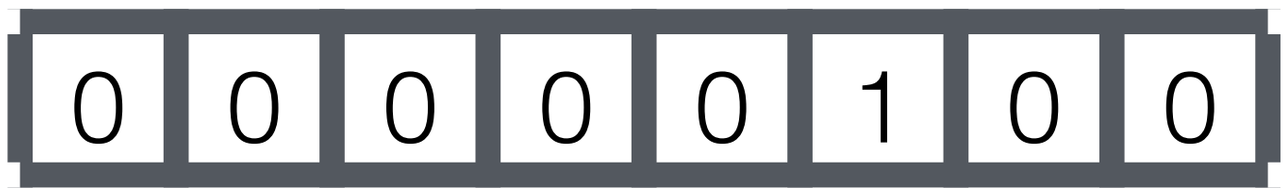
A filter:



After convolution*:



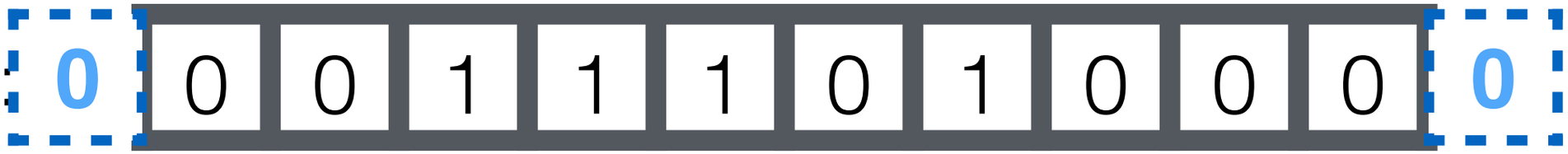
After ReLU:



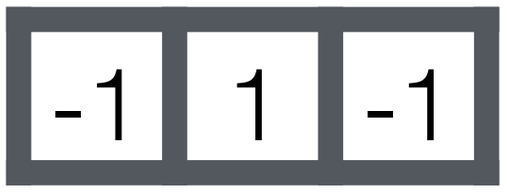
*correlation

Convolutional Layer: 1D example

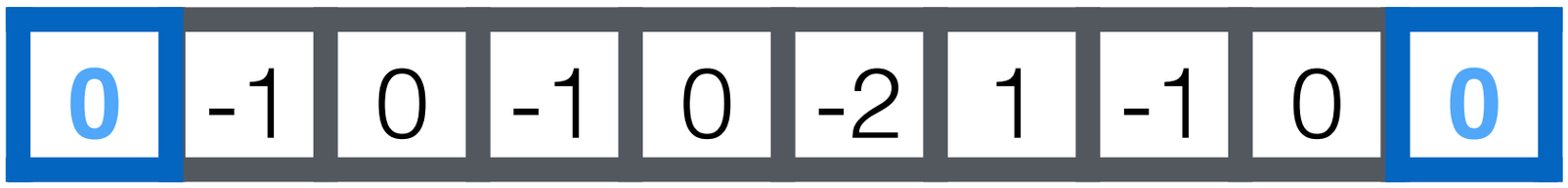
A 1D image:



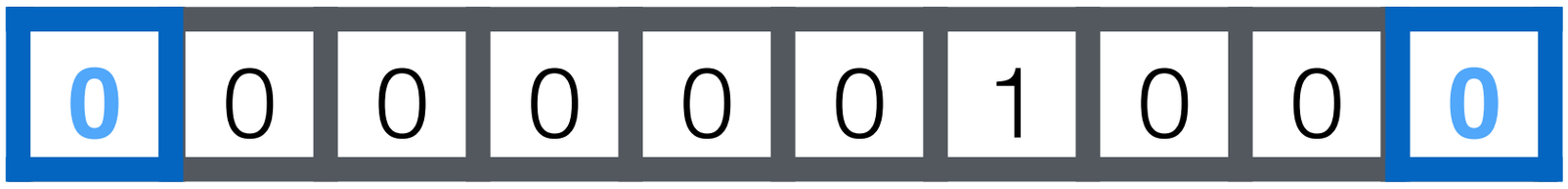
A filter:



After convolution*:



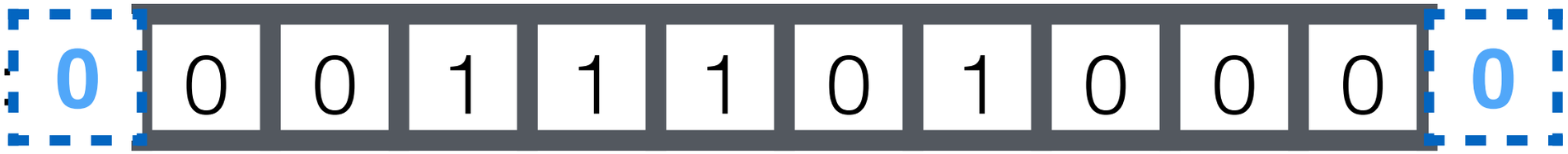
After ReLU:



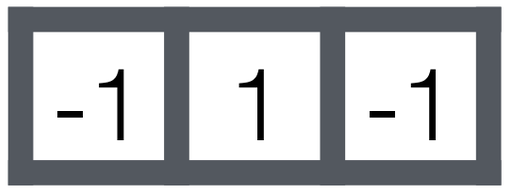
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



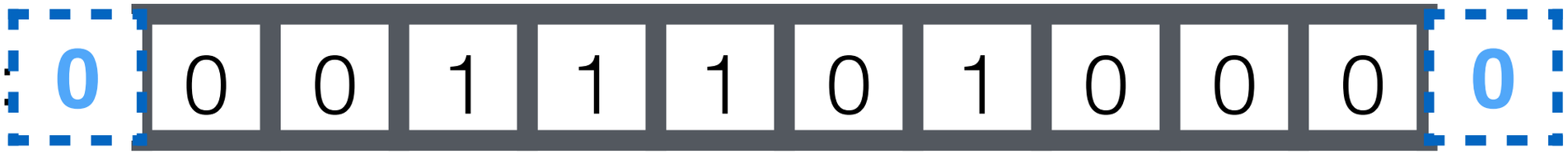
After convolution*:

After ReLU:

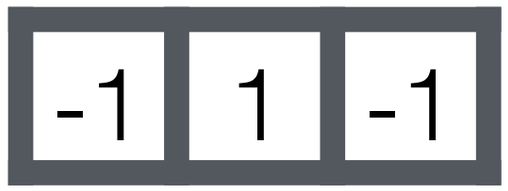
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias +1

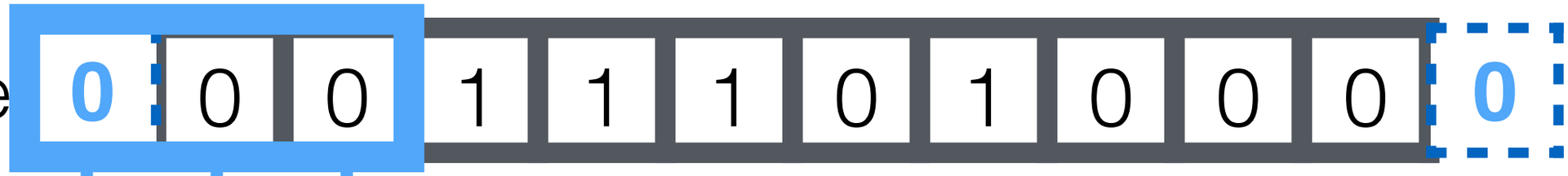
After convolution*:

After ReLU:

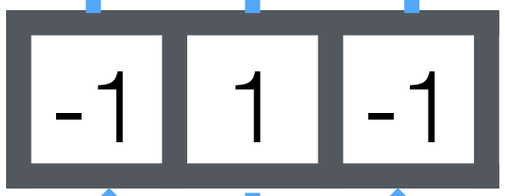
*correlation

Convolutional Layer: 1D example

A 1D image



A filter:



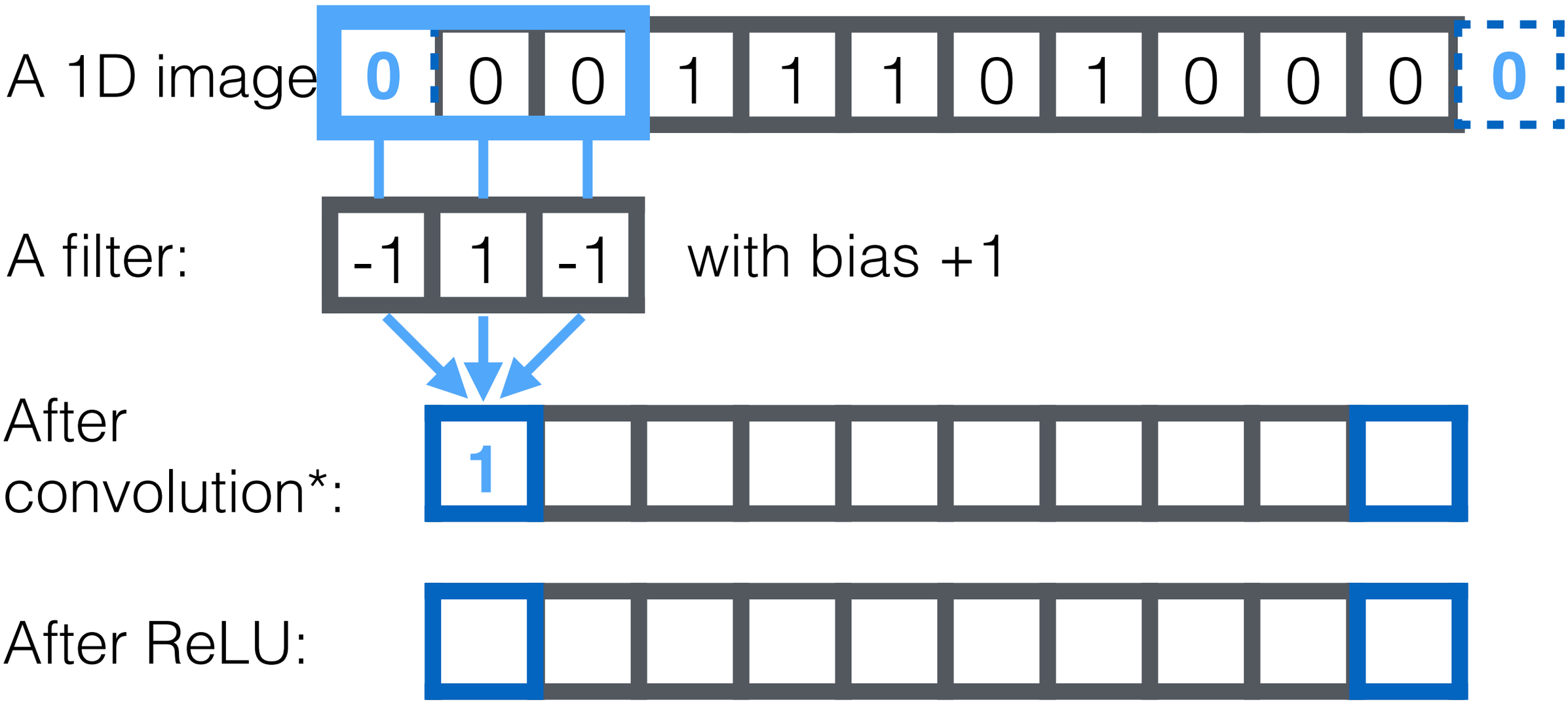
with bias +1

After convolution*:

After ReLU:

*correlation

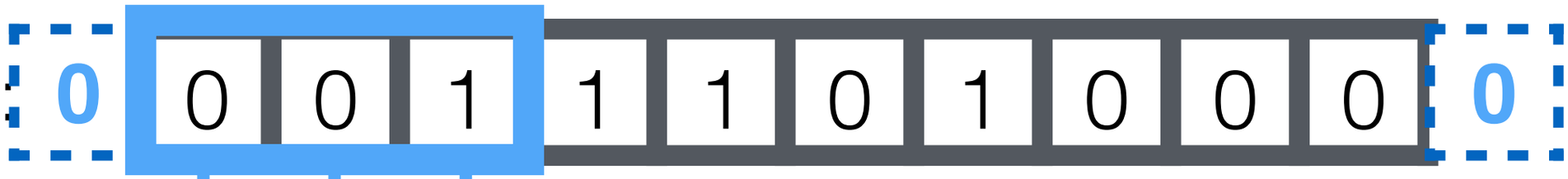
Convolutional Layer: 1D example



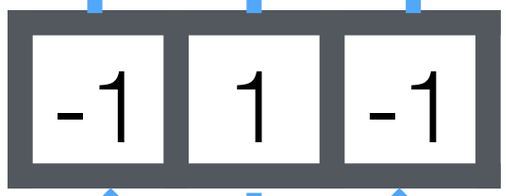
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias +1

After convolution*:

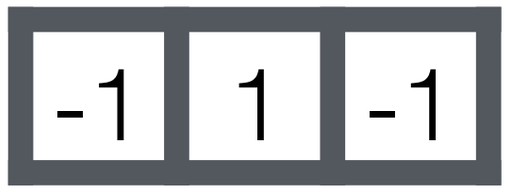
After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:



with bias +1

After convolution*:

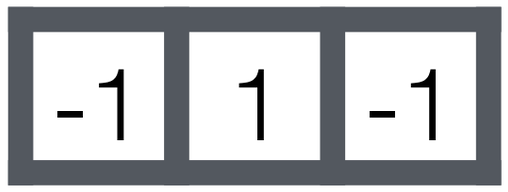
After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image:

A filter:



with bias +1

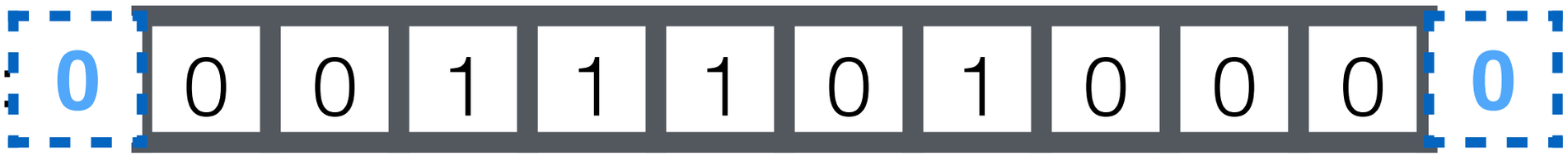
After convolution*:

After ReLU:

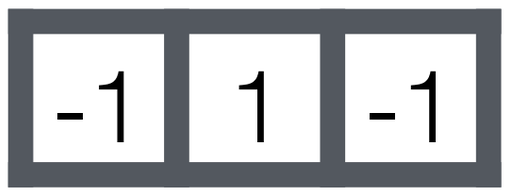
*correlation

Convolutional Layer: 1D example

A 1D image:

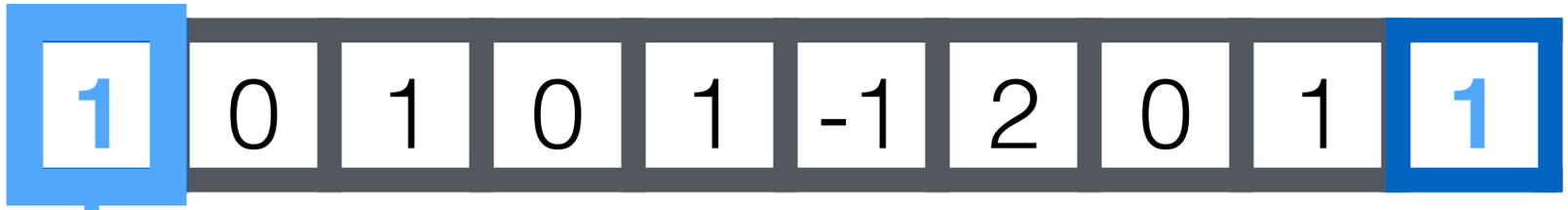


A filter:



with bias +1

After convolution*:

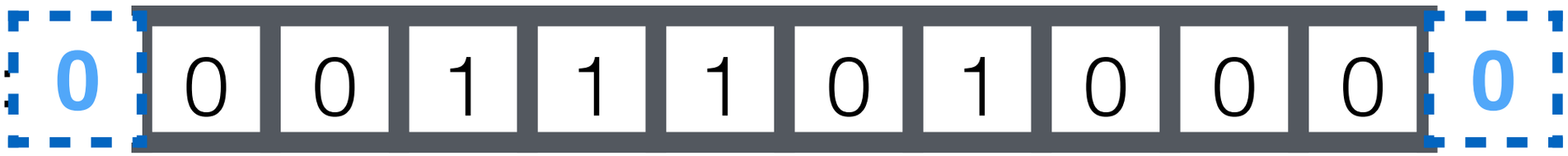


After ReLU:

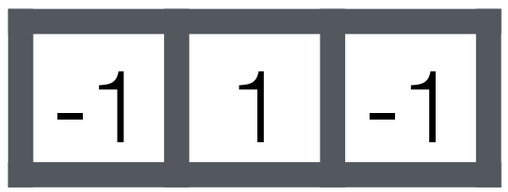
*correlation

Convolutional Layer: 1D example

A 1D image:

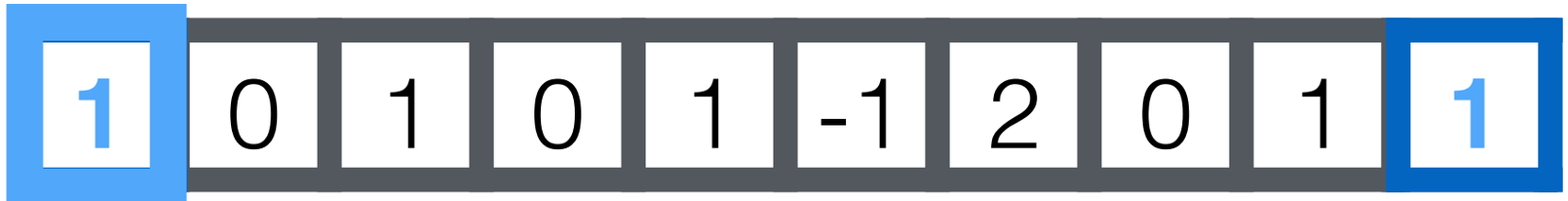


A filter:



with bias +1

After convolution*:

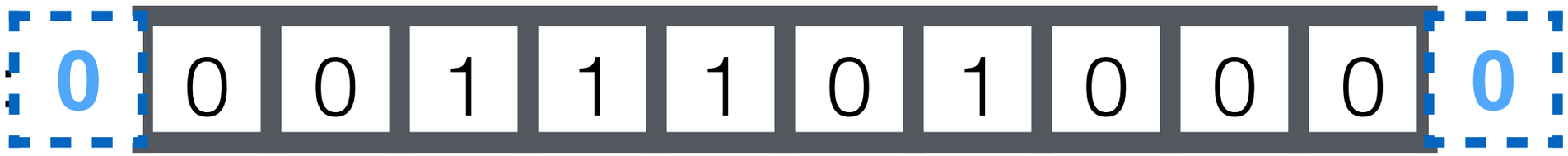


After ReLU:

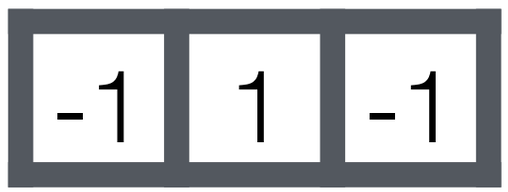
*correlation

Convolutional Layer: 1D example

A 1D image:

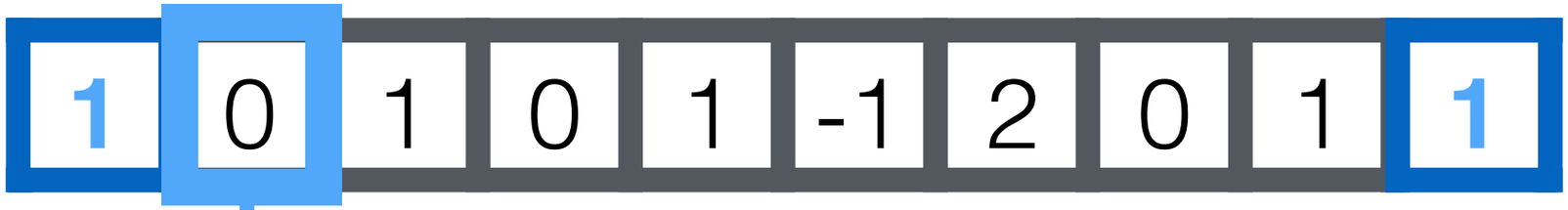


A filter:



with bias +1

After convolution*:



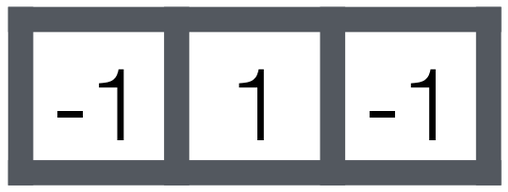
After ReLU:

*correlation

Convolutional Layer: 1D example

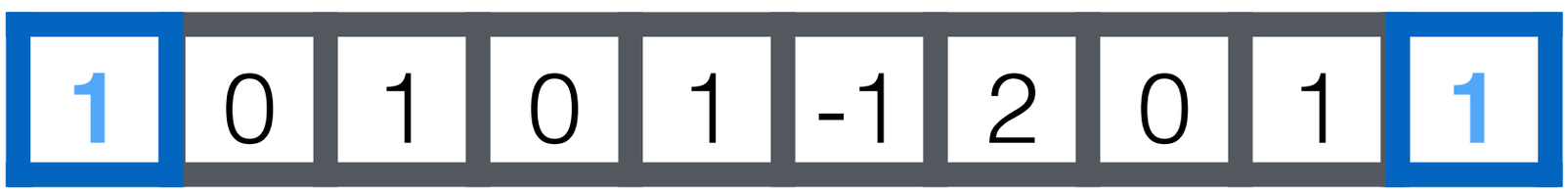
A 1D image:

A filter:



with bias +1

After convolution*:

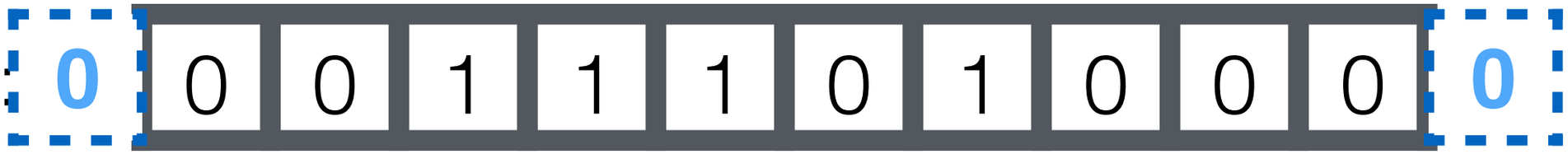


After ReLU:

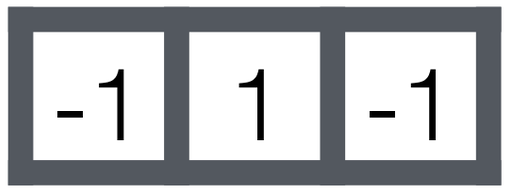
*correlation

Convolutional Layer: 1D example

A 1D image:

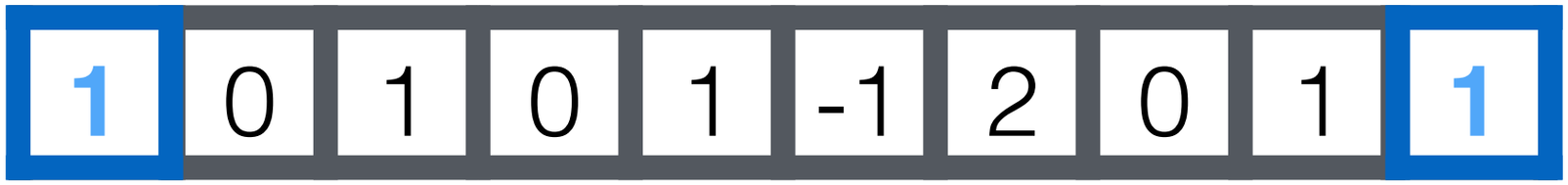


A filter:

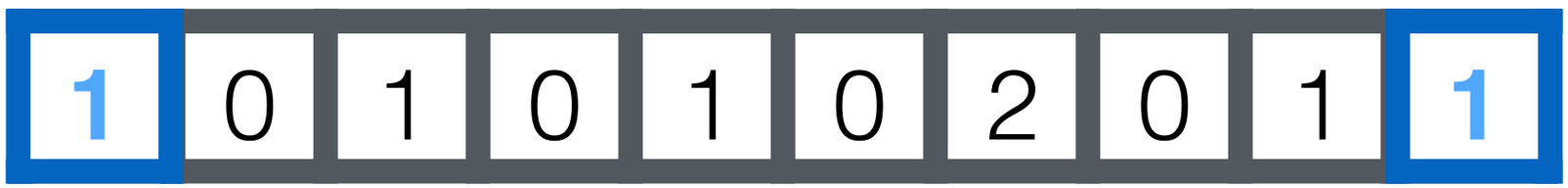


with bias +1

After convolution*:



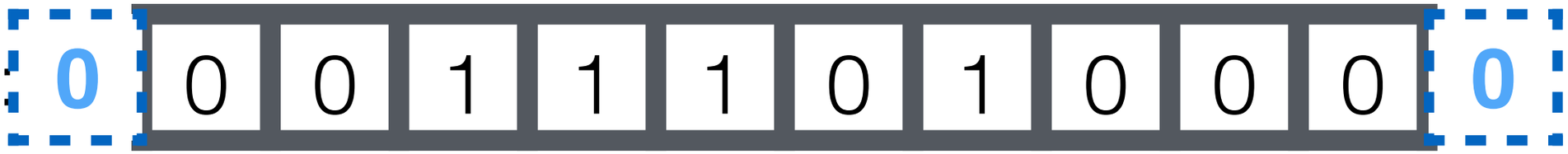
After ReLU:



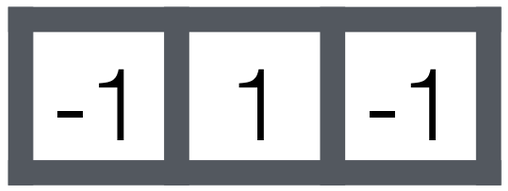
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias +1

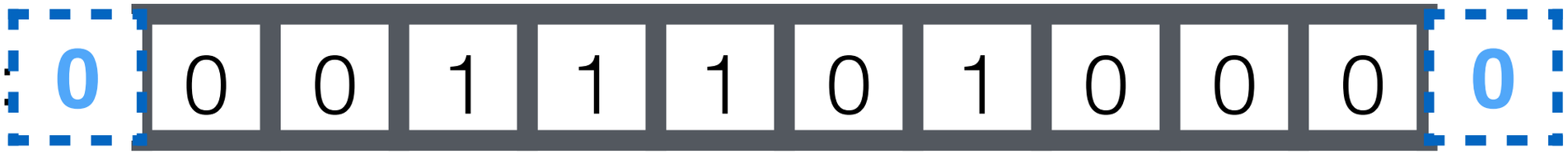
After convolution*:

After ReLU:

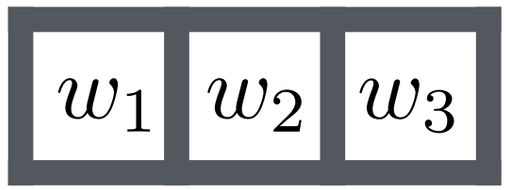
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias *b*

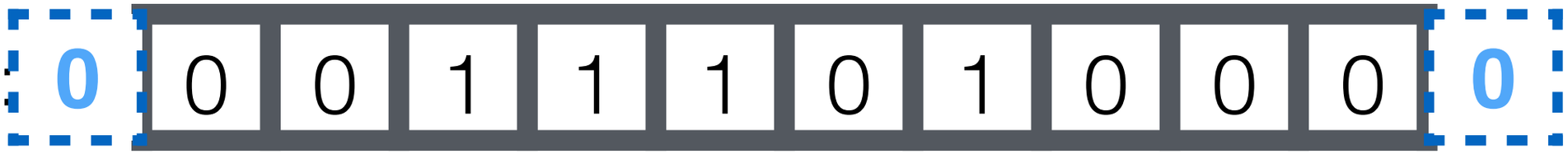
After convolution*:

After ReLU:

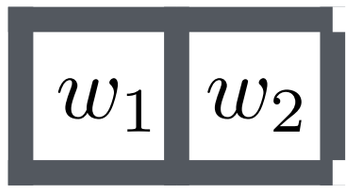
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias b

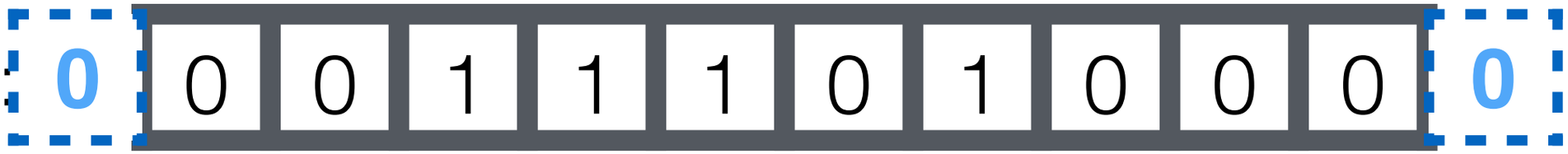
After convolution*:

After ReLU:

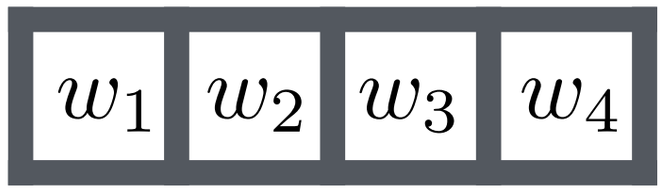
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias *b*

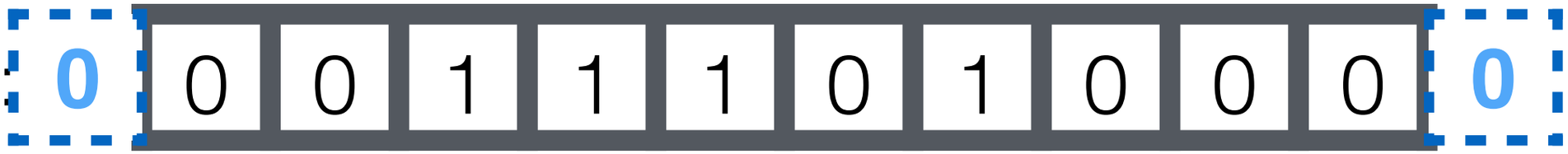
After convolution*:

After ReLU:

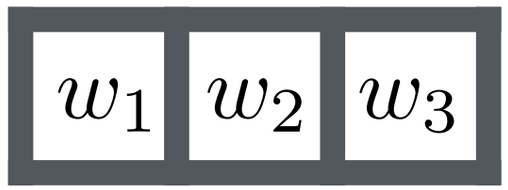
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



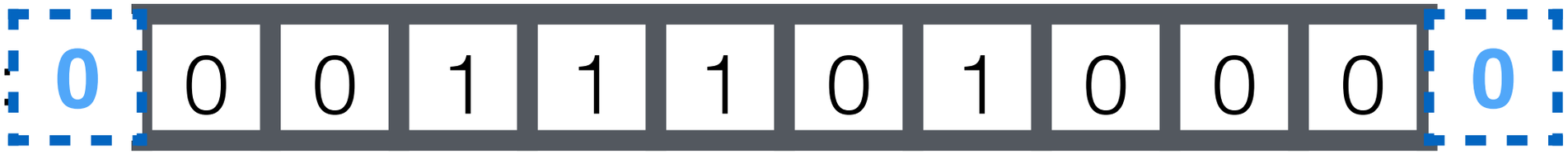
with bias *b*

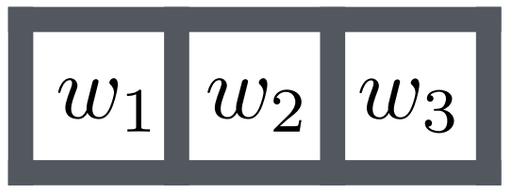
After convolution*:

After ReLU:

*correlation

Convolutional Layer: 1D example

A 1D image: 

A filter: 

with bias *b*

After convolution*: 

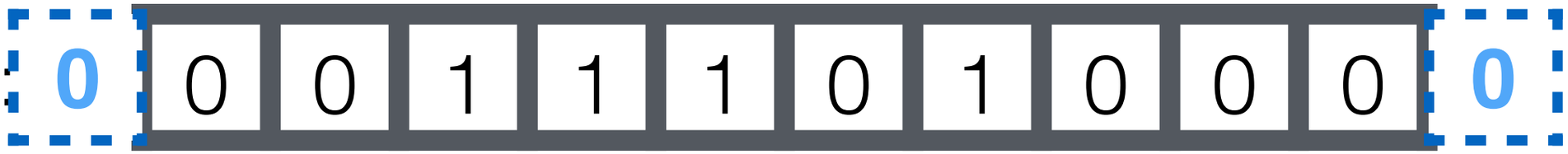
After ReLU: 

- How many weights (including bias)?

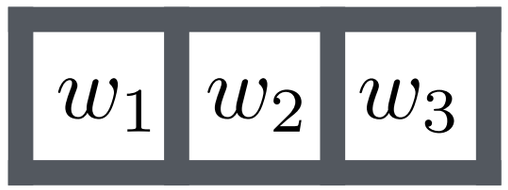
*correlation

Convolutional Layer: 1D example

A 1D image:



A filter:



with bias *b*

After convolution*:

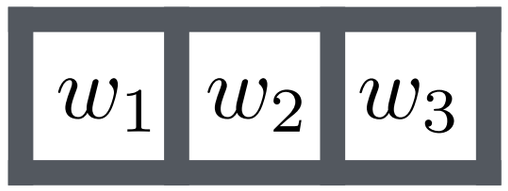
After ReLU:

- How many weights (including bias)? 4

*correlation

Convolutional Layer: 1D example

A 1D image: 

A filter: 

with bias *b*

After convolution*: 

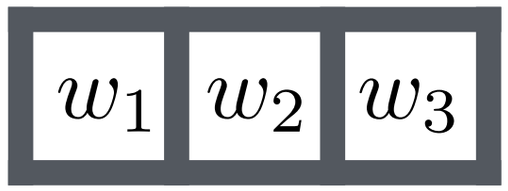
After ReLU: 

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

*correlation

Convolutional Layer: 1D example

A 1D image: 

A filter: 

with bias *b*

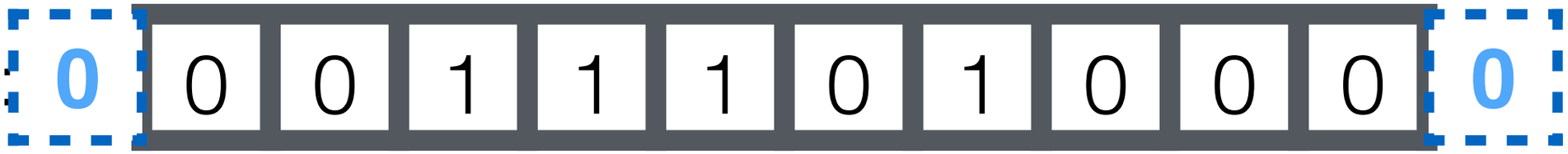
After convolution*: 

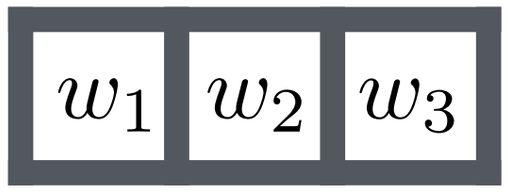
After ReLU: 

- 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: 

A filter: 

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

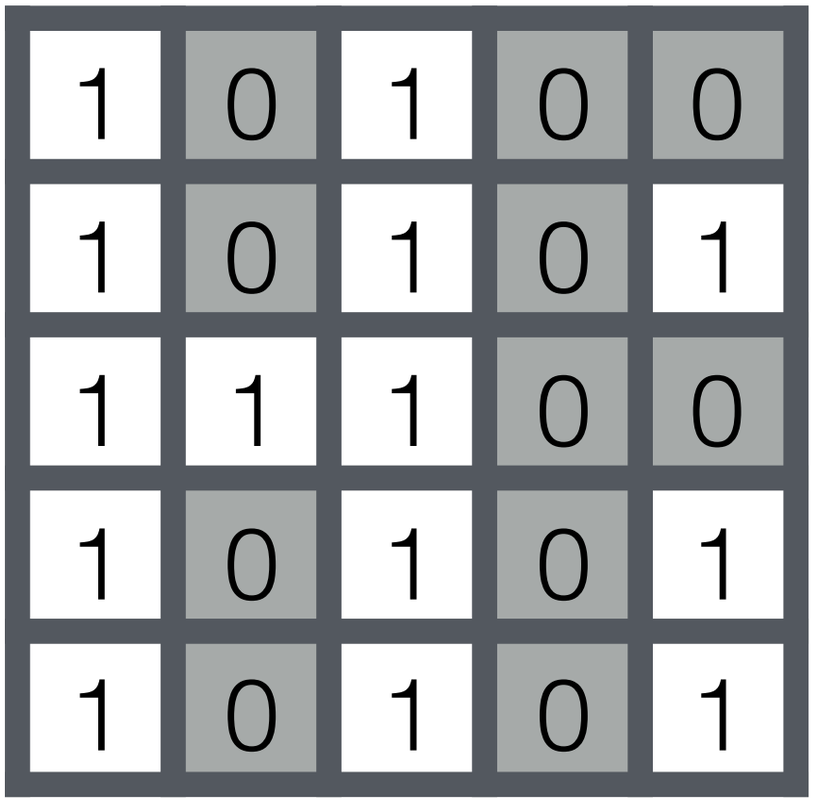
Convolutional Layer: 2D example

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:



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

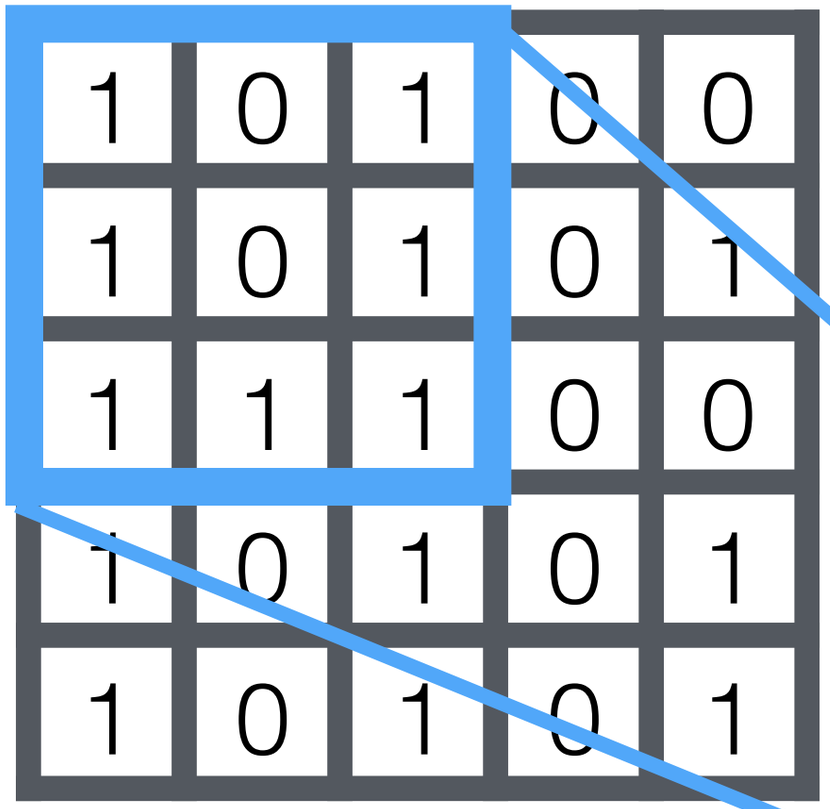
A filter:

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

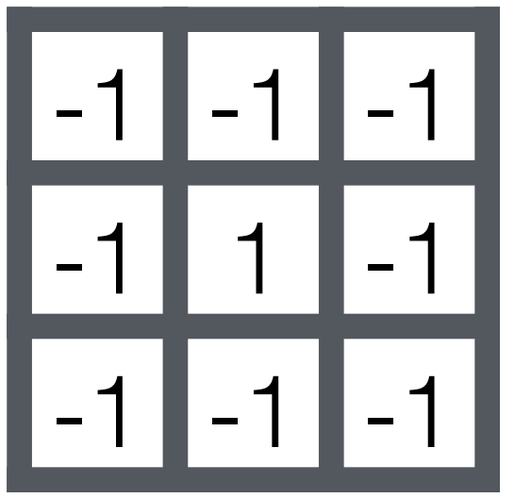
After convolution:

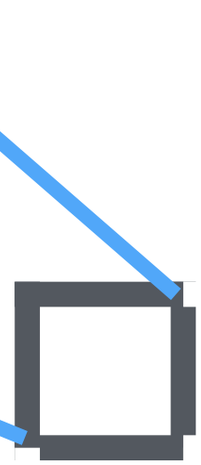
Convolutional Layer: 2D example

A 2D image:



A filter:

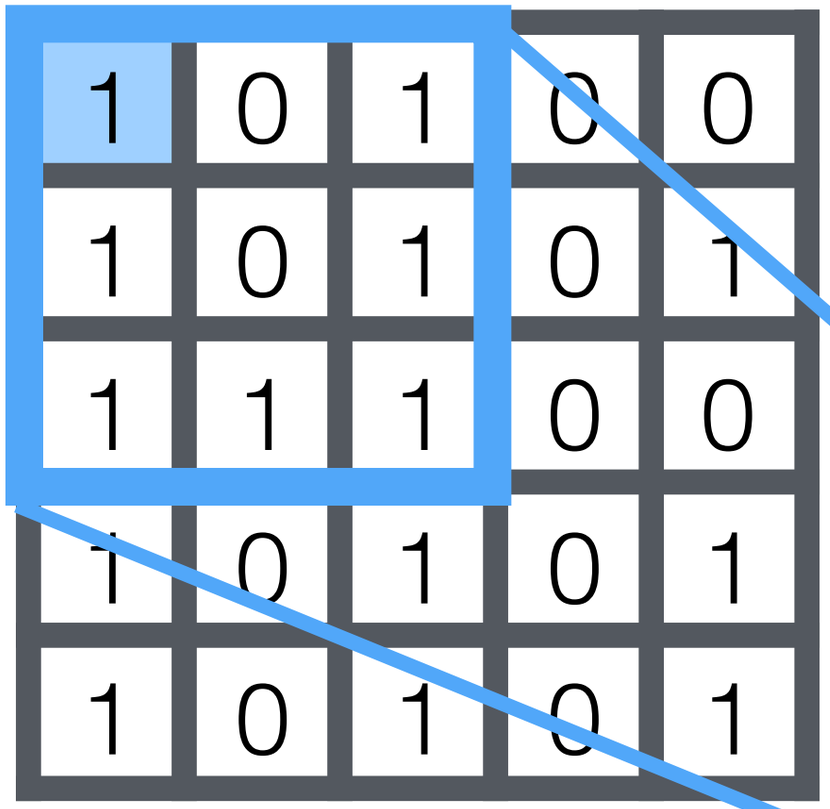




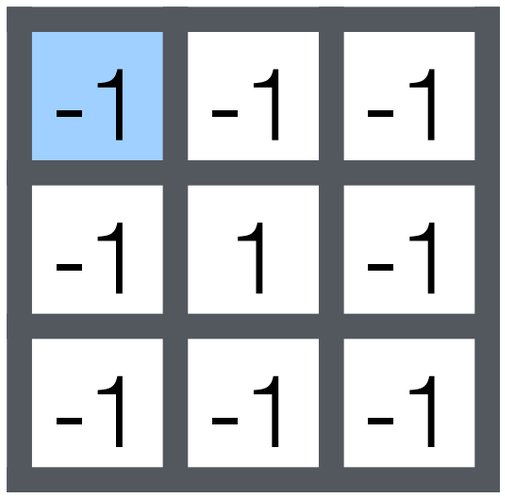
After convolution:

Convolutional Layer: 2D example

A 2D image:



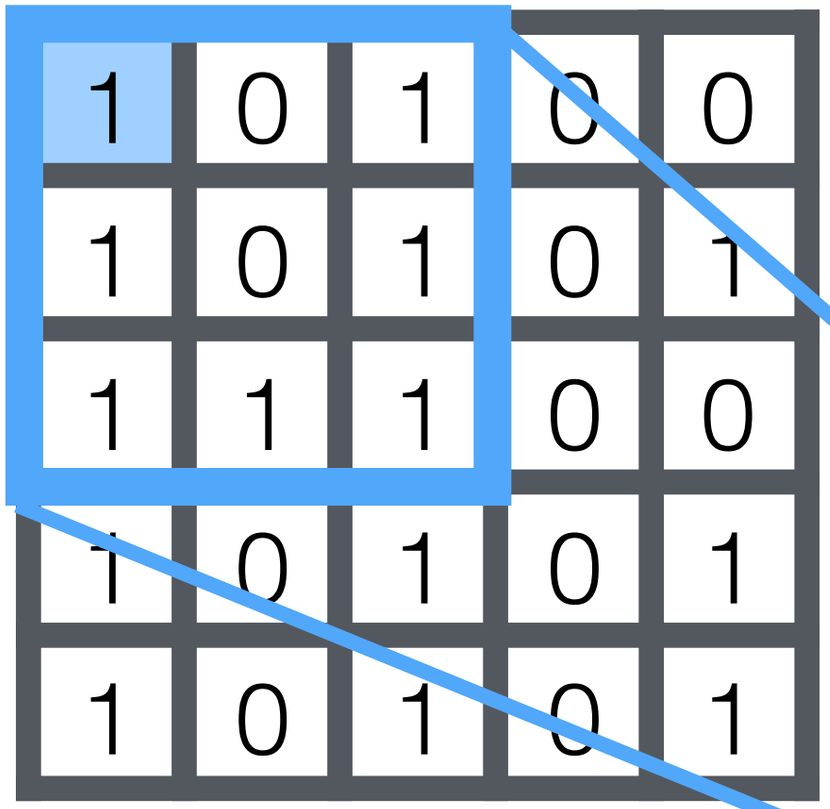
A filter:



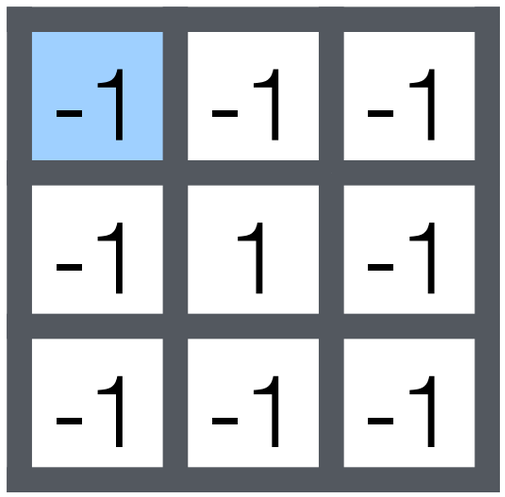
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

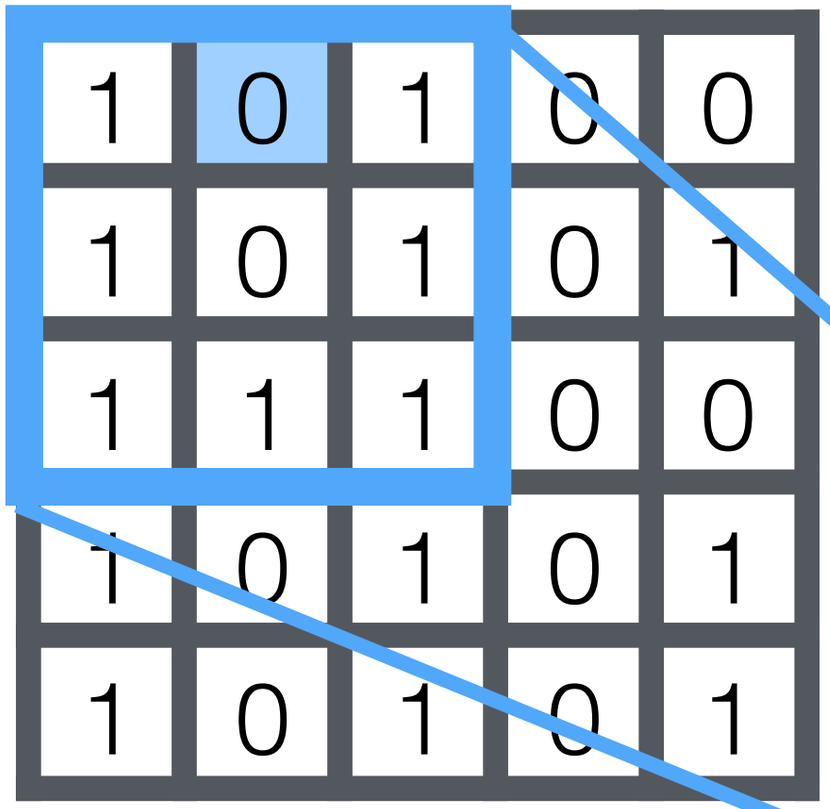


-1

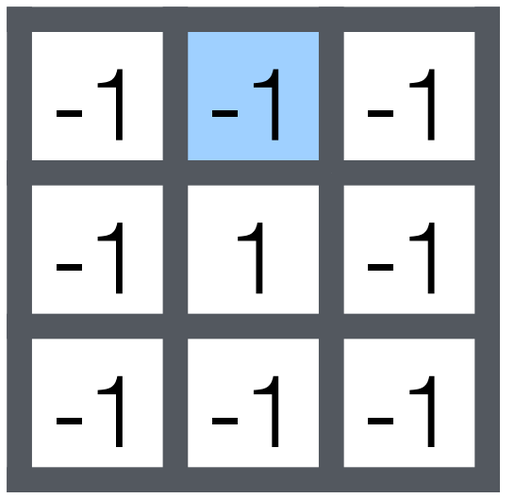
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

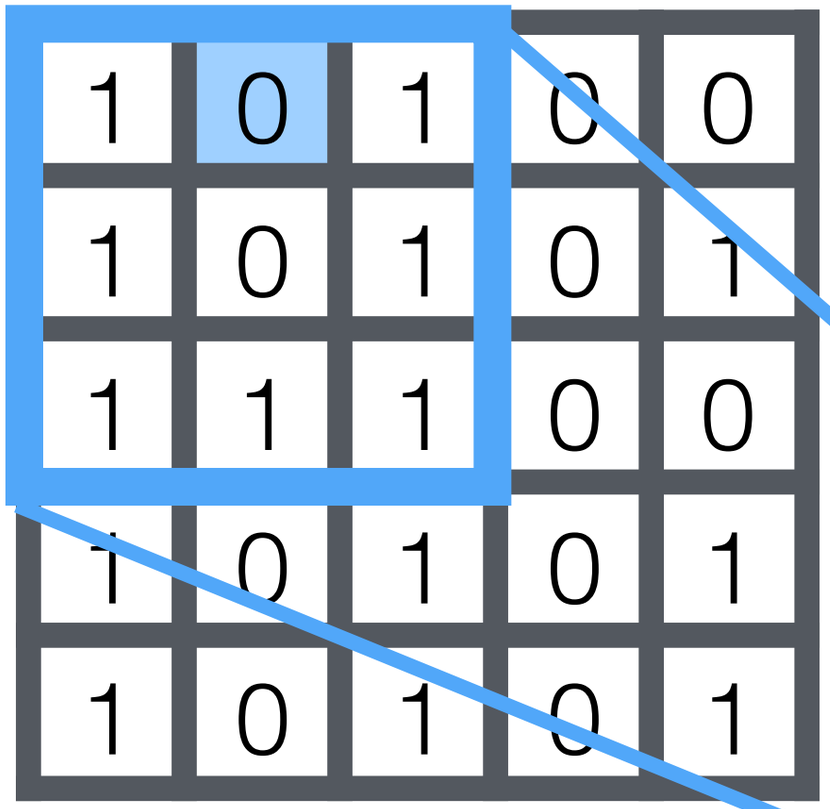


-1

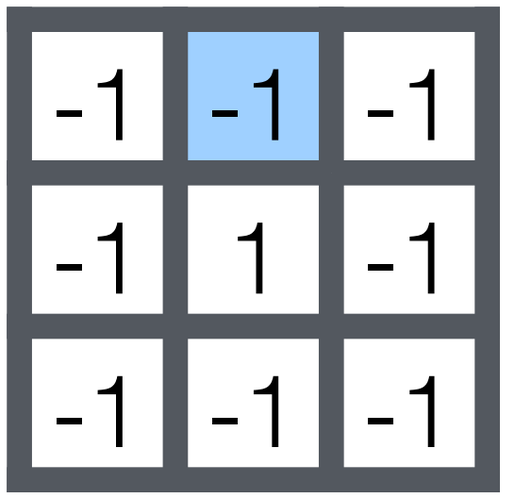
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

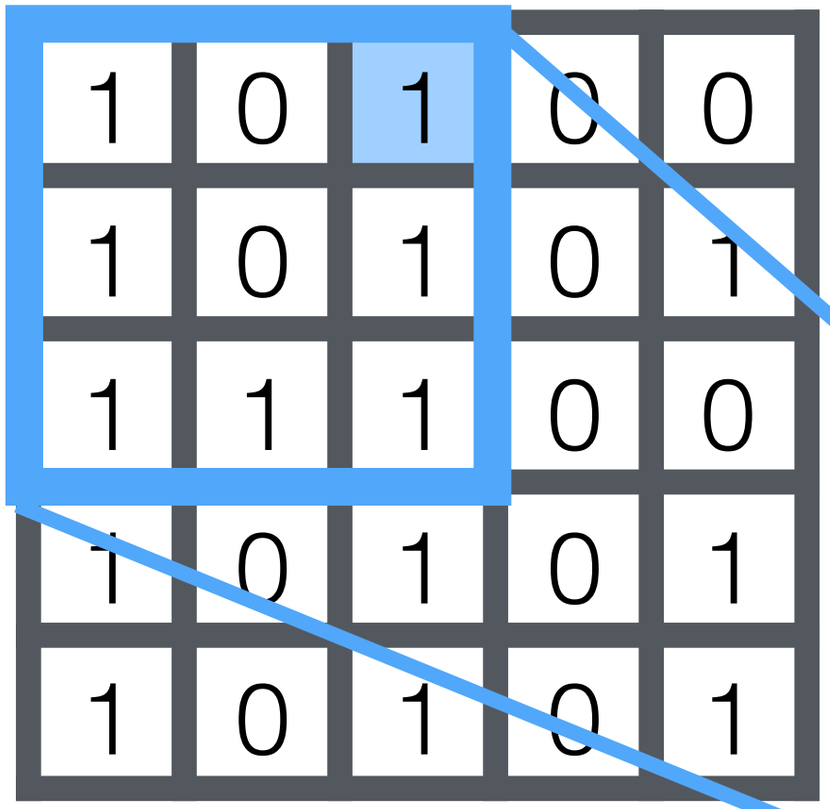


-1 + 0

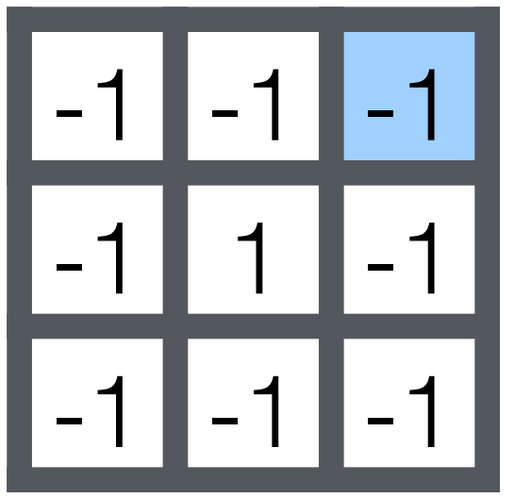
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

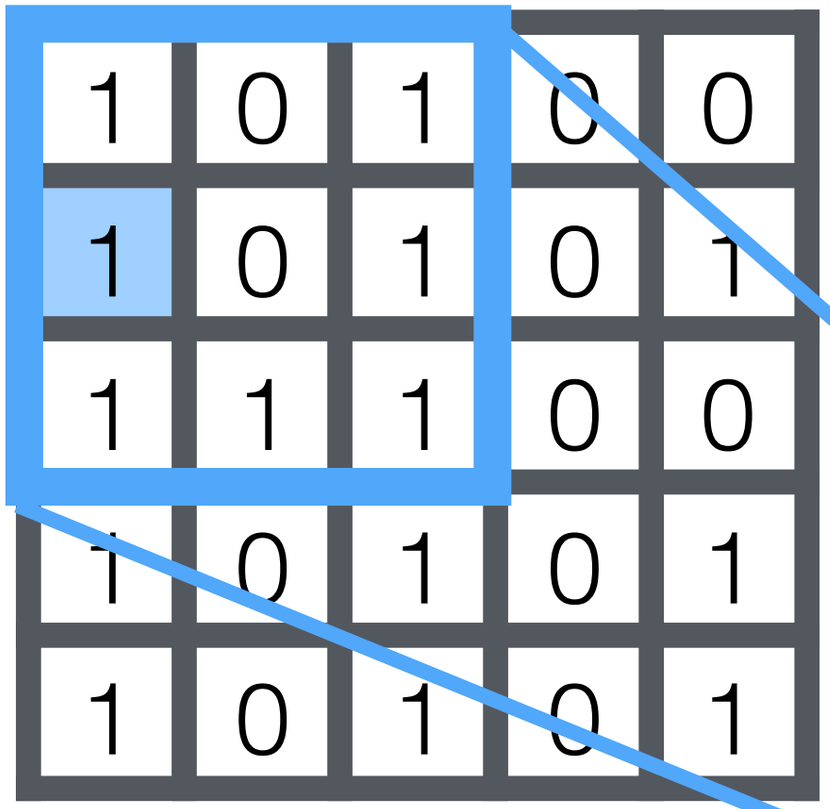


-1 + 0 + -1

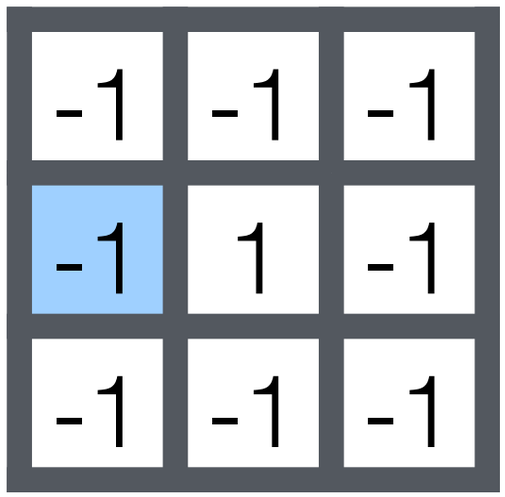
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

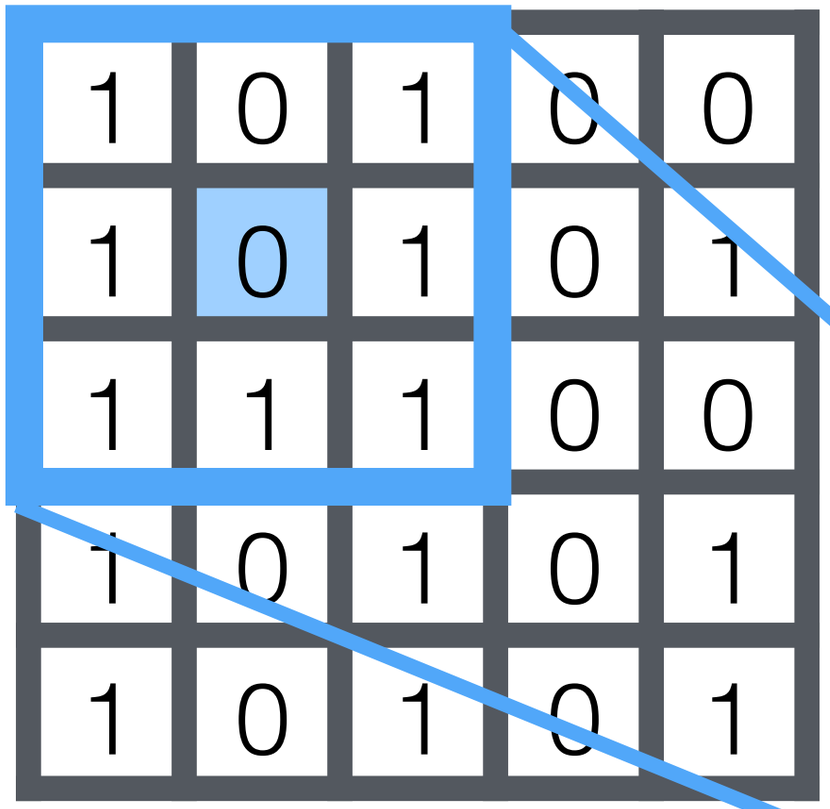


-1 + 0 + -1
+ -1

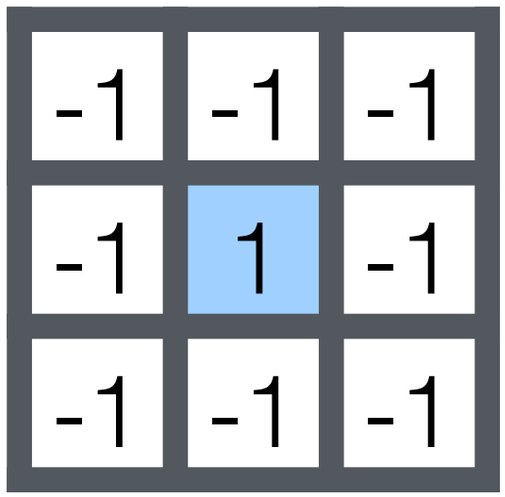
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

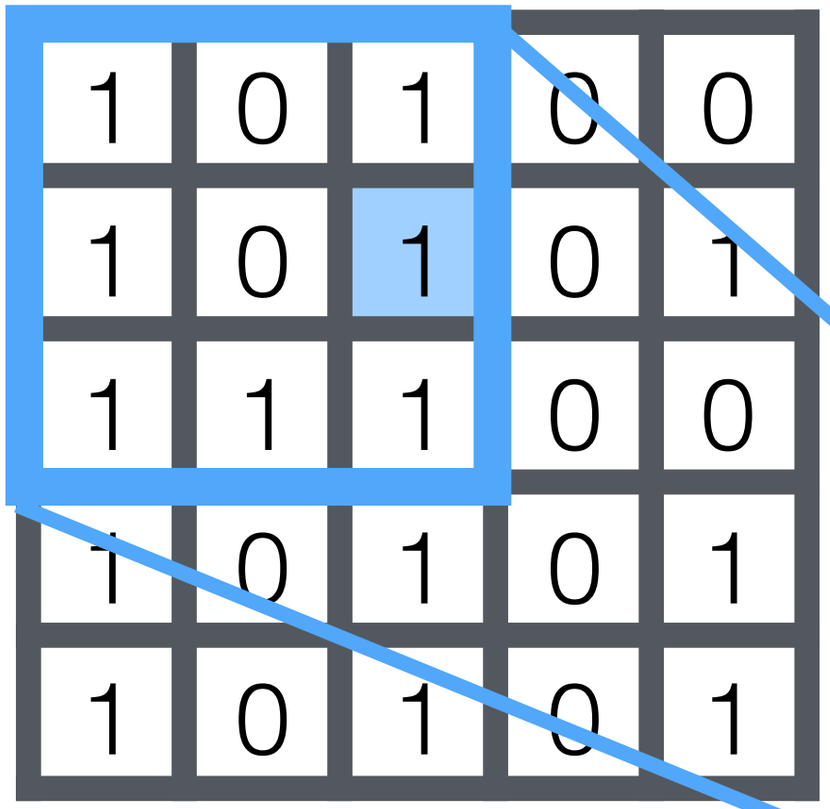


-1 + 0 + -1
+ -1 + 0

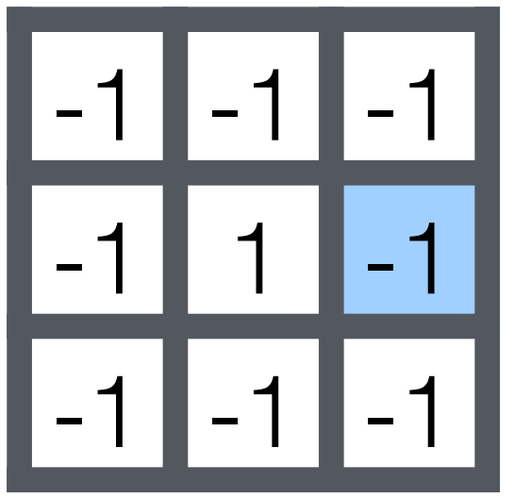
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

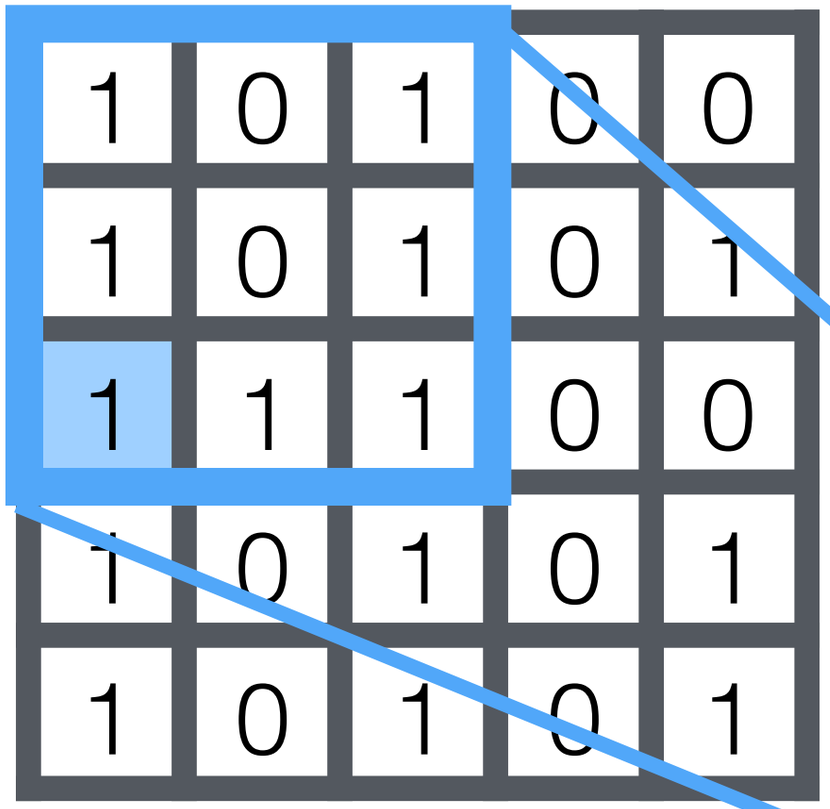


-1 + 0 + -1
+ -1 + 0 + -1

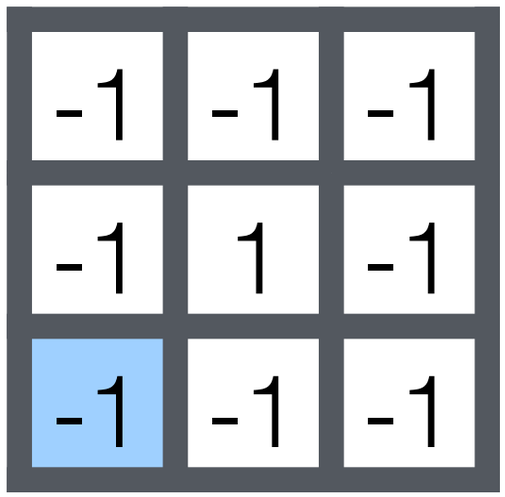
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

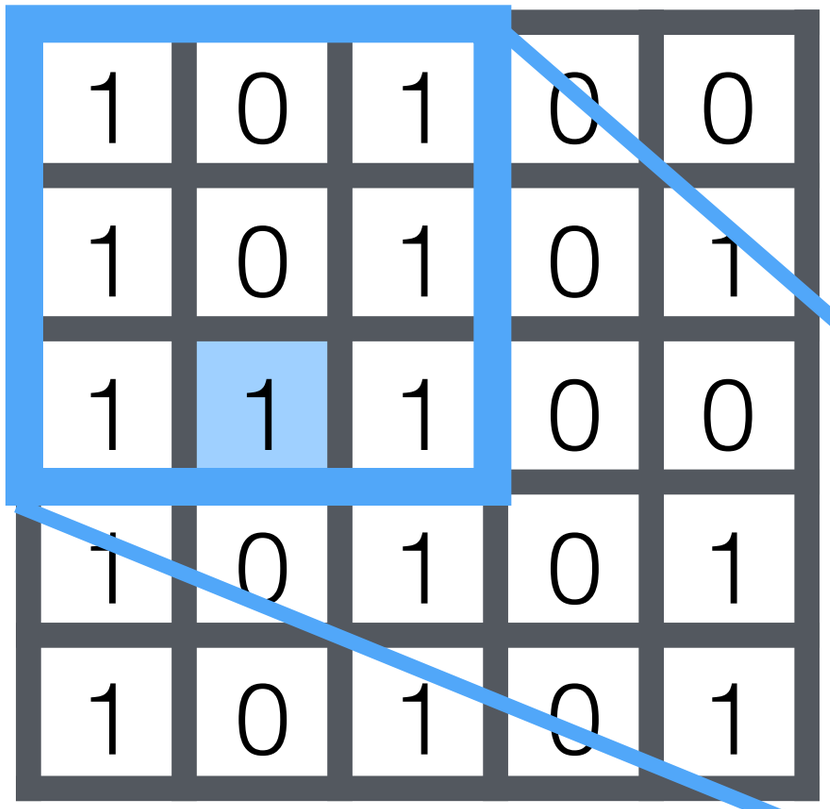


-1 + 0 + -1
+ -1 + 0 + -1
+ -1

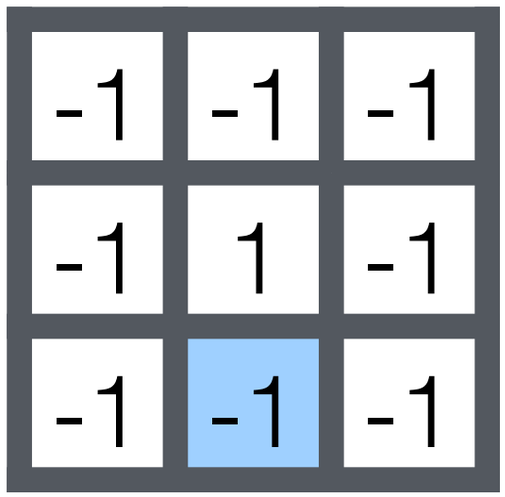
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

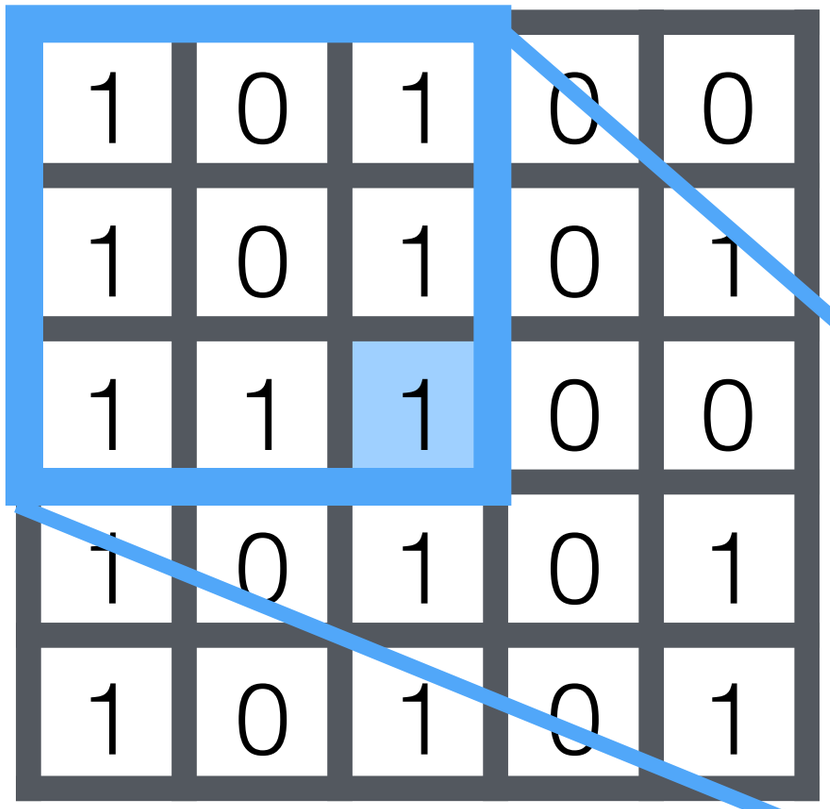


-1 + 0 + -1
+ -1 + 0 + -1
+ -1 + - 1

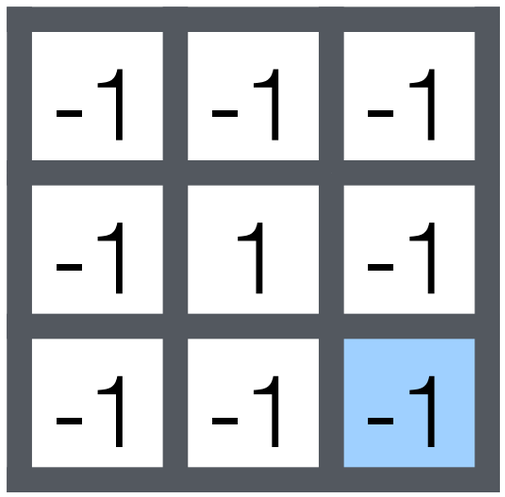
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

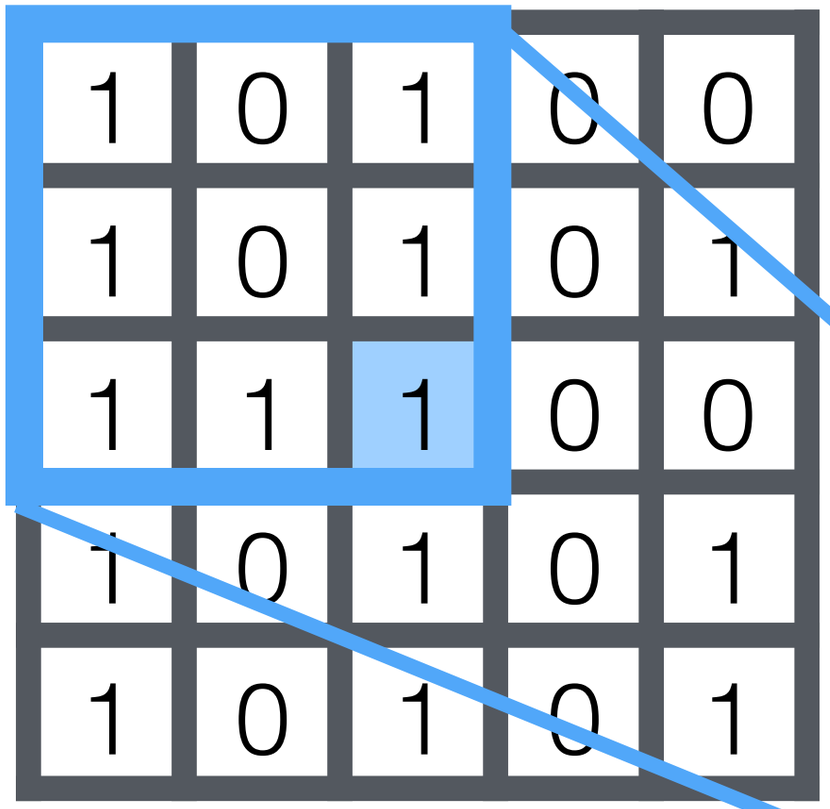


-1 + 0 + -1
+ -1 + 0 + -1
+ -1 + - 1 + -1

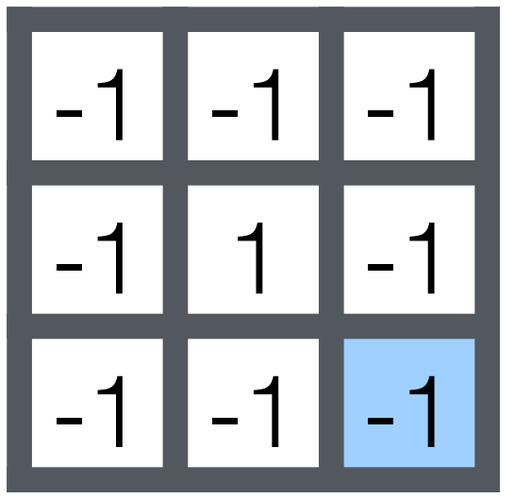
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

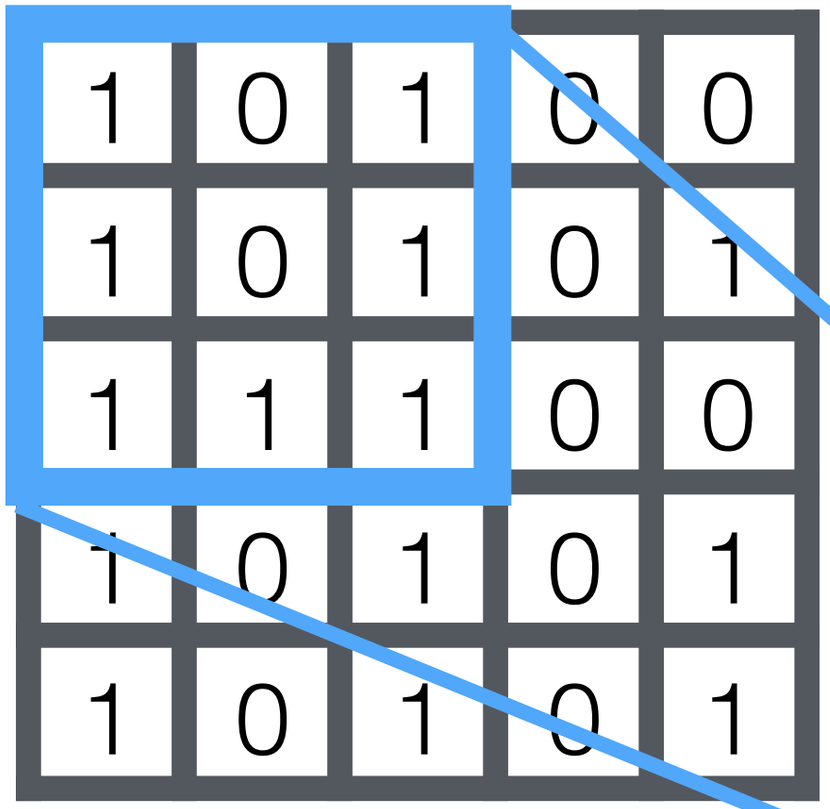


-1 + 0 + -1
+ -1 + 0 + -1
+ -1 + - 1 + -1
= -7

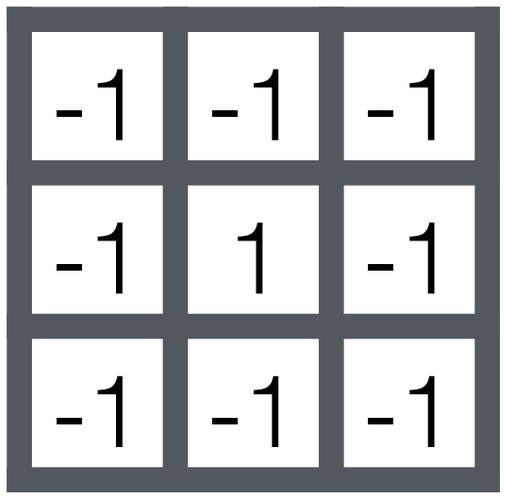
After convolution:

Convolutional Layer: 2D example

A 2D image:



A filter:

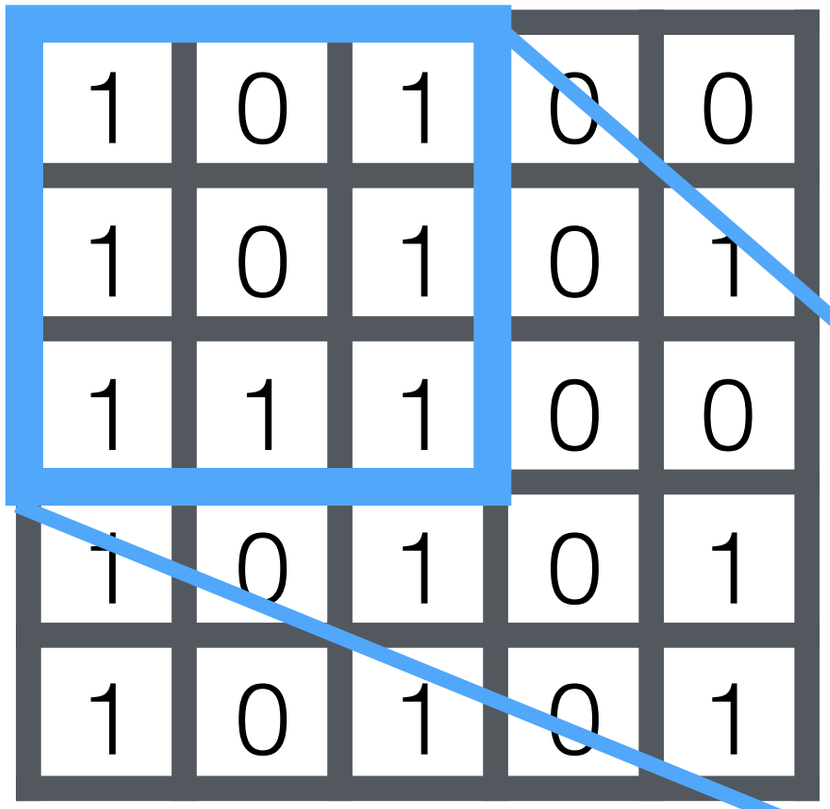


-1 + 0 + -1
+ -1 + 0 + -1
+ -1 + - 1 + -1
= -7

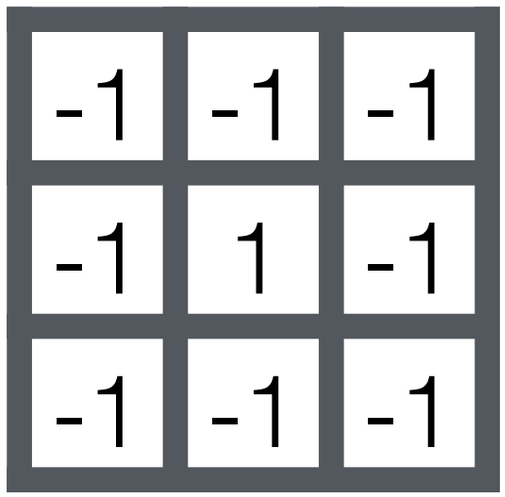
After convolution:

Convolutional Layer: 2D example

A 2D image:



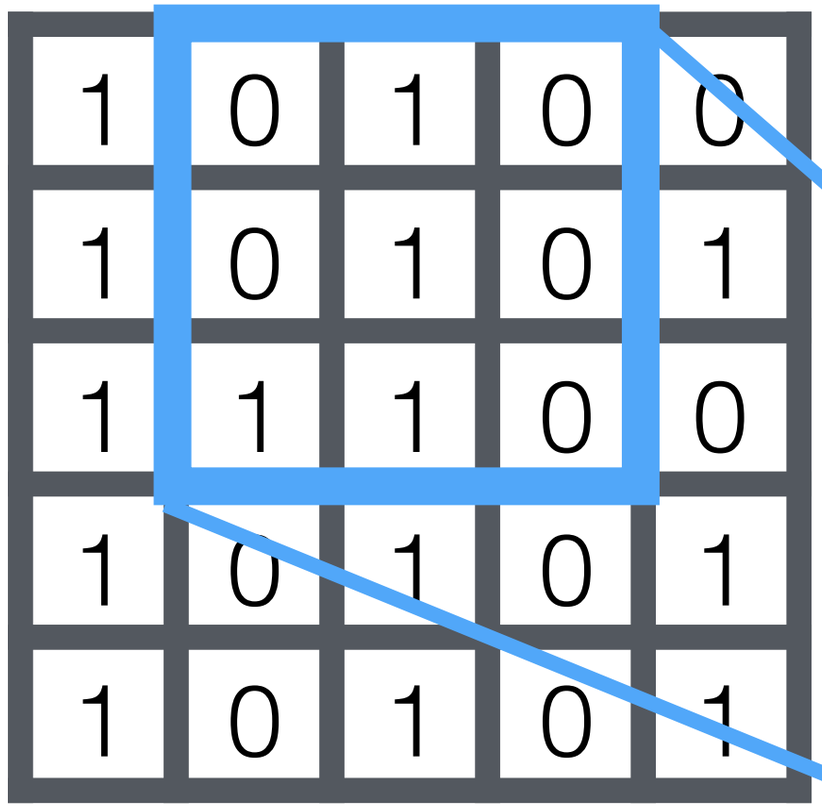
A filter:



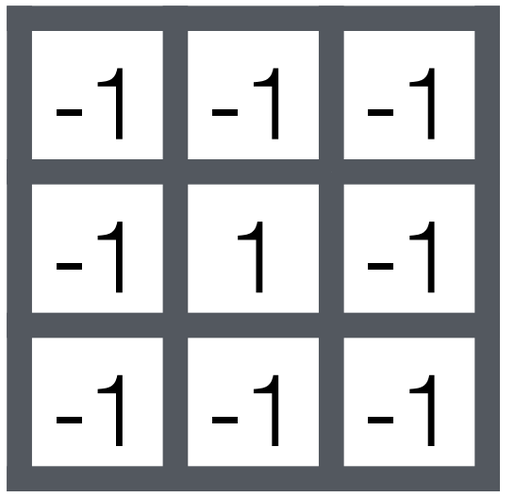
After convolution:

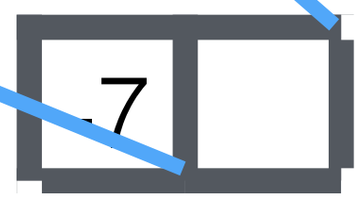
Convolutional Layer: 2D example

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

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	

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

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		

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	-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

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		

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:

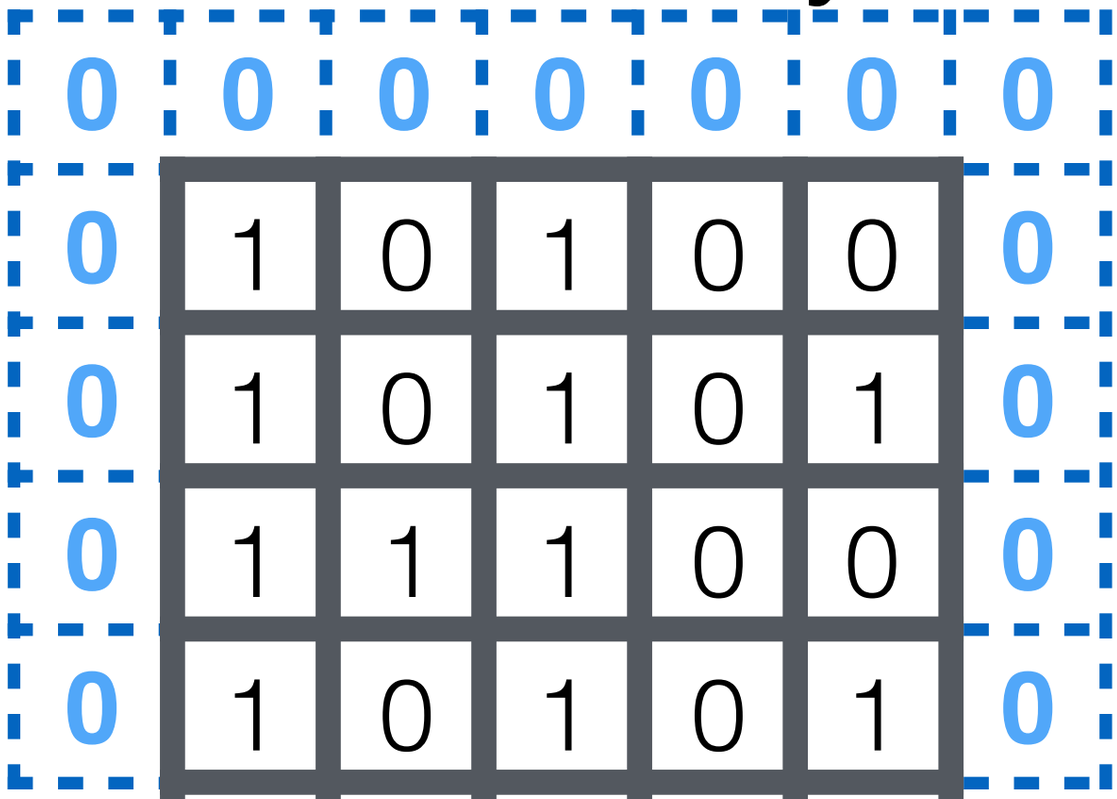
-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

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

Convolutional Layer: 2D example

A 2D image:



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

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:

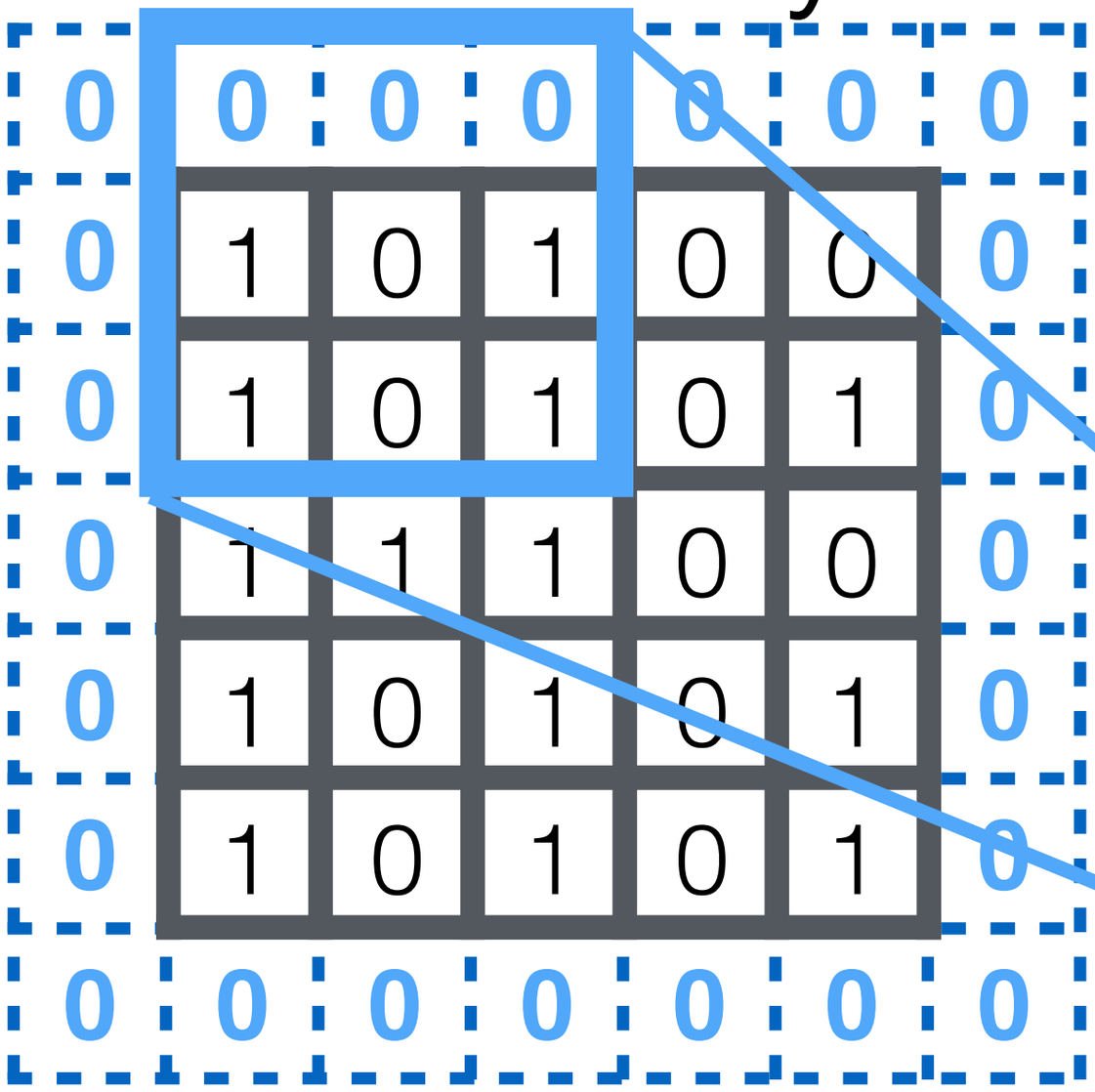
-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution:

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

Convolutional Layer: 2D example

A 2D image:



A filter:

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

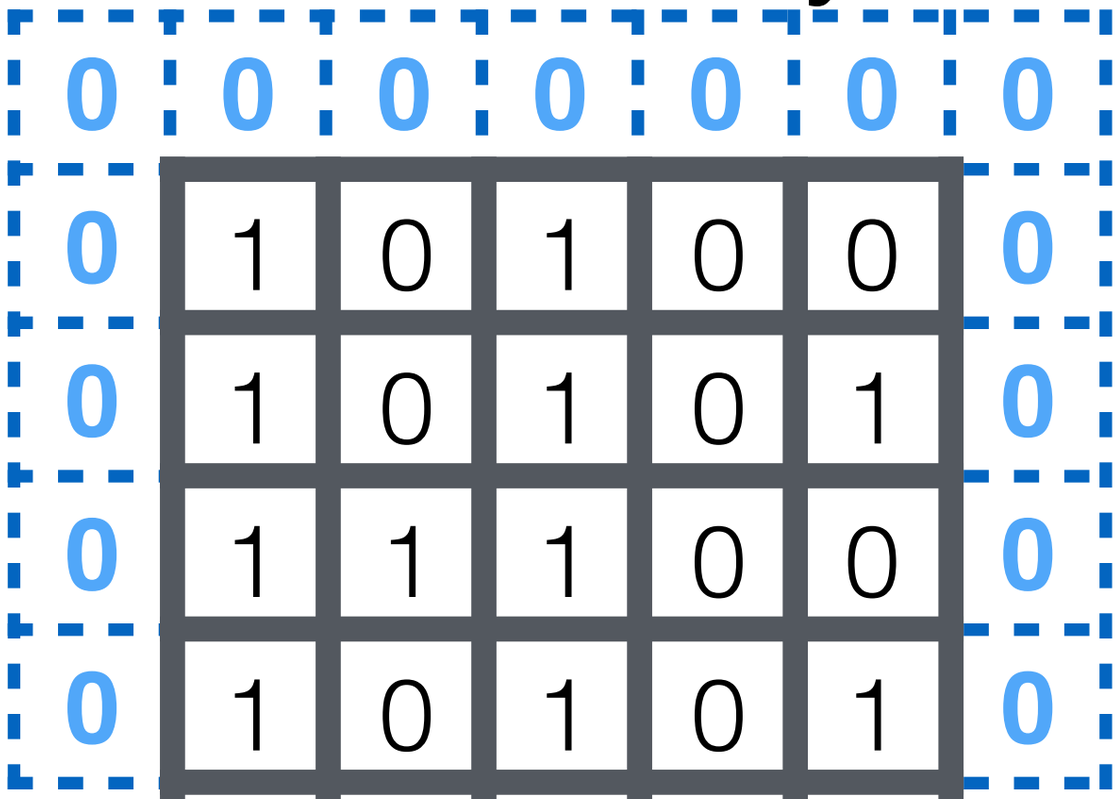
After convolution:

0	-4

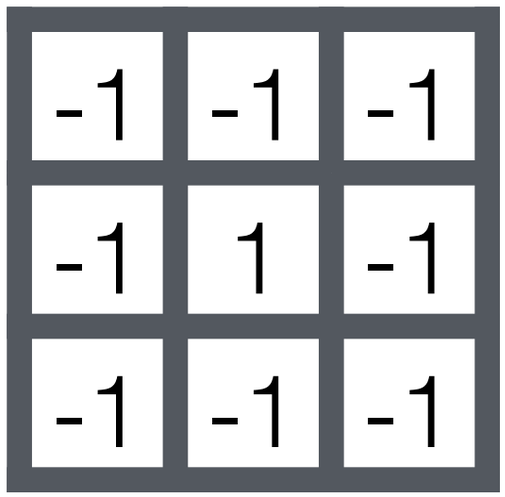
-7	-2	-4
-5	-2	-5
-7	-2	-5

Convolutional Layer: 2D example

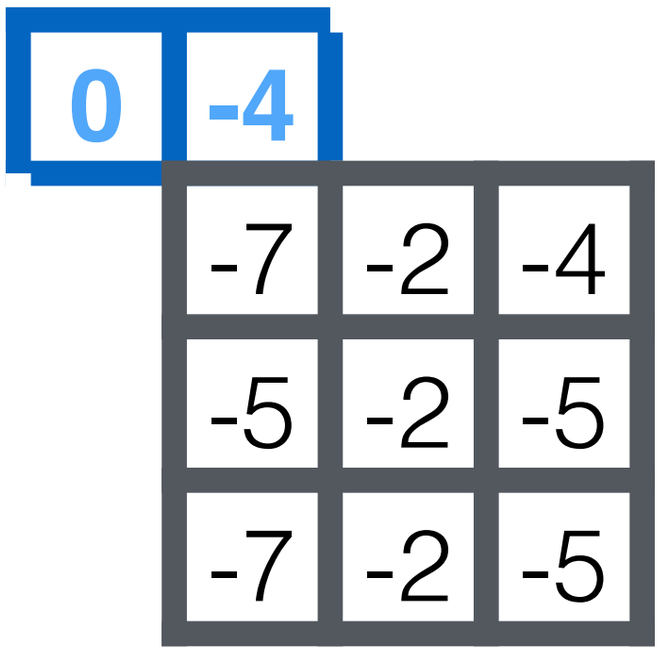
A 2D image:



A filter:



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

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:

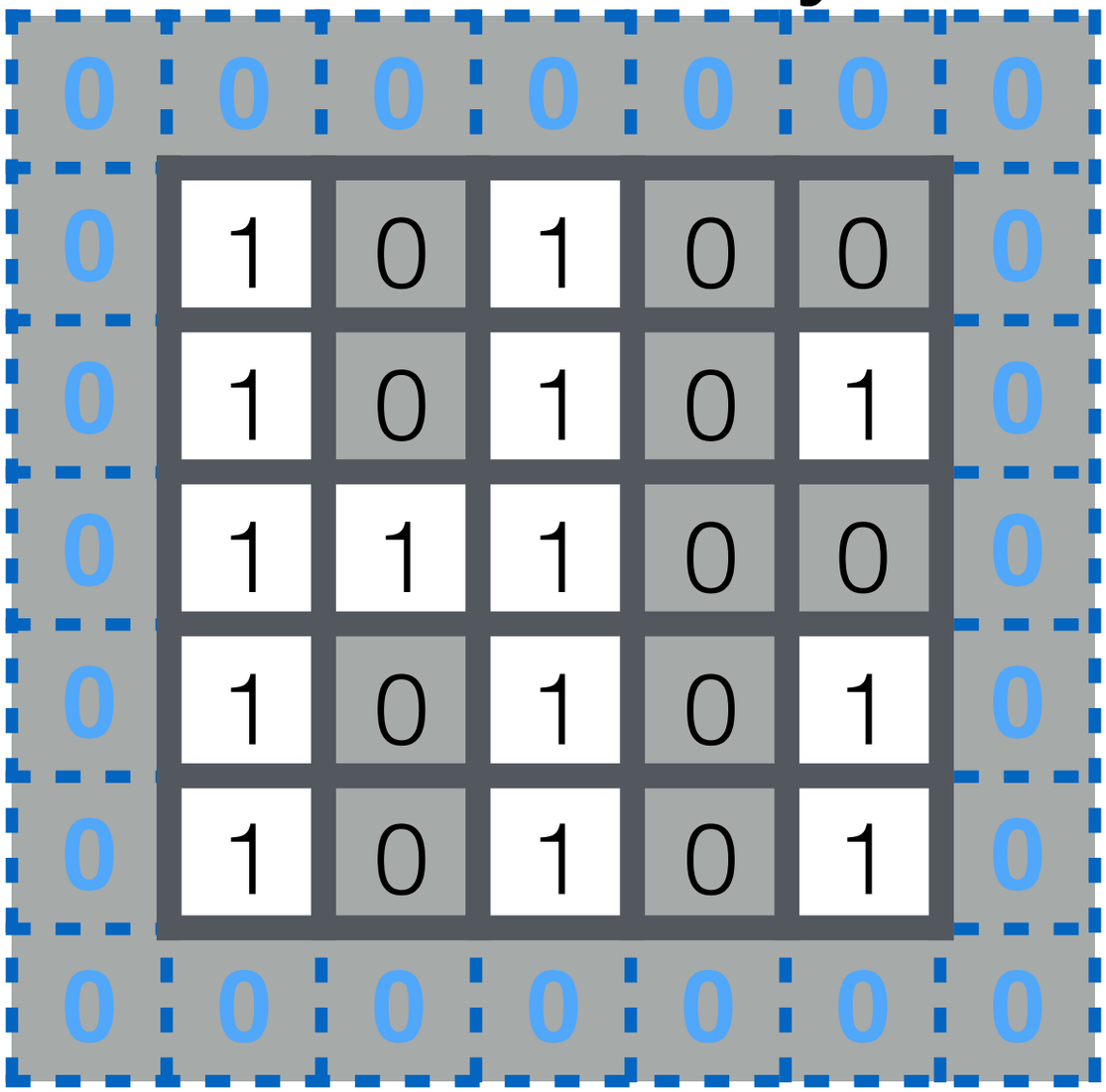
-1	-1	-1
-1	1	-1
-1	-1	-1

After convolution & ReLU:

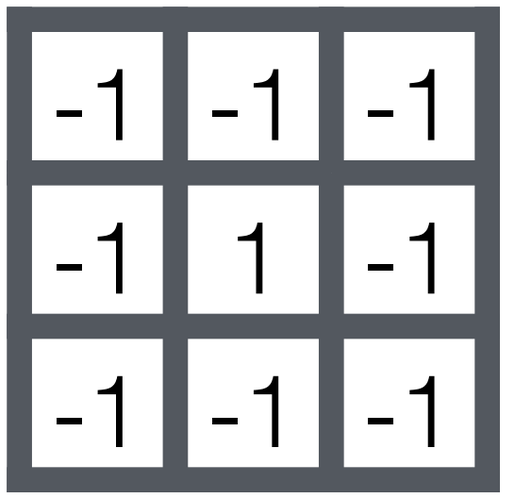
0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Convolutional Layer: 2D example

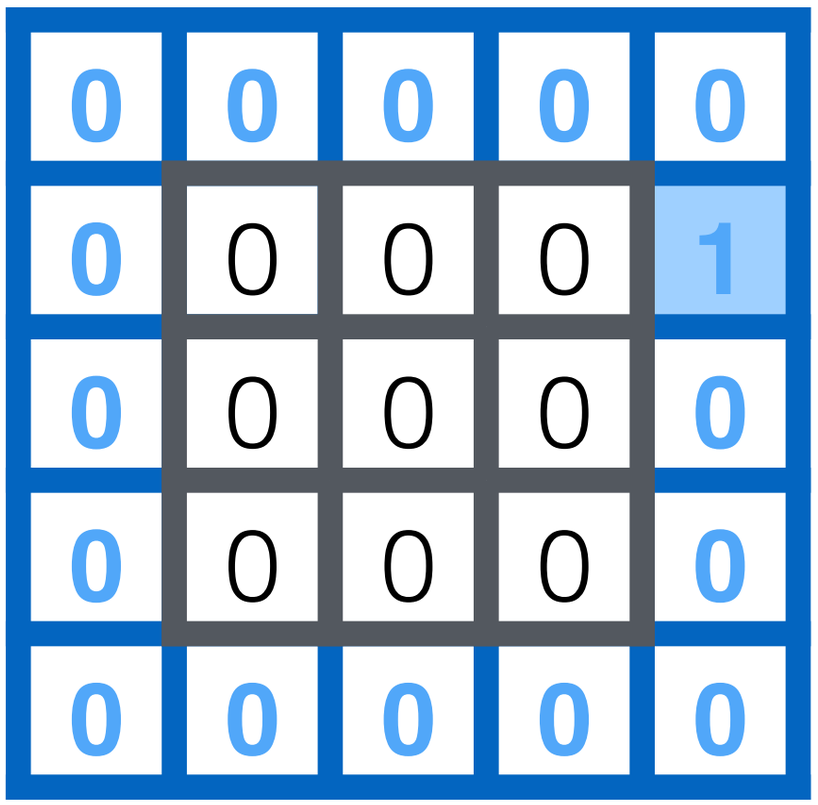
A 2D image:



A filter:

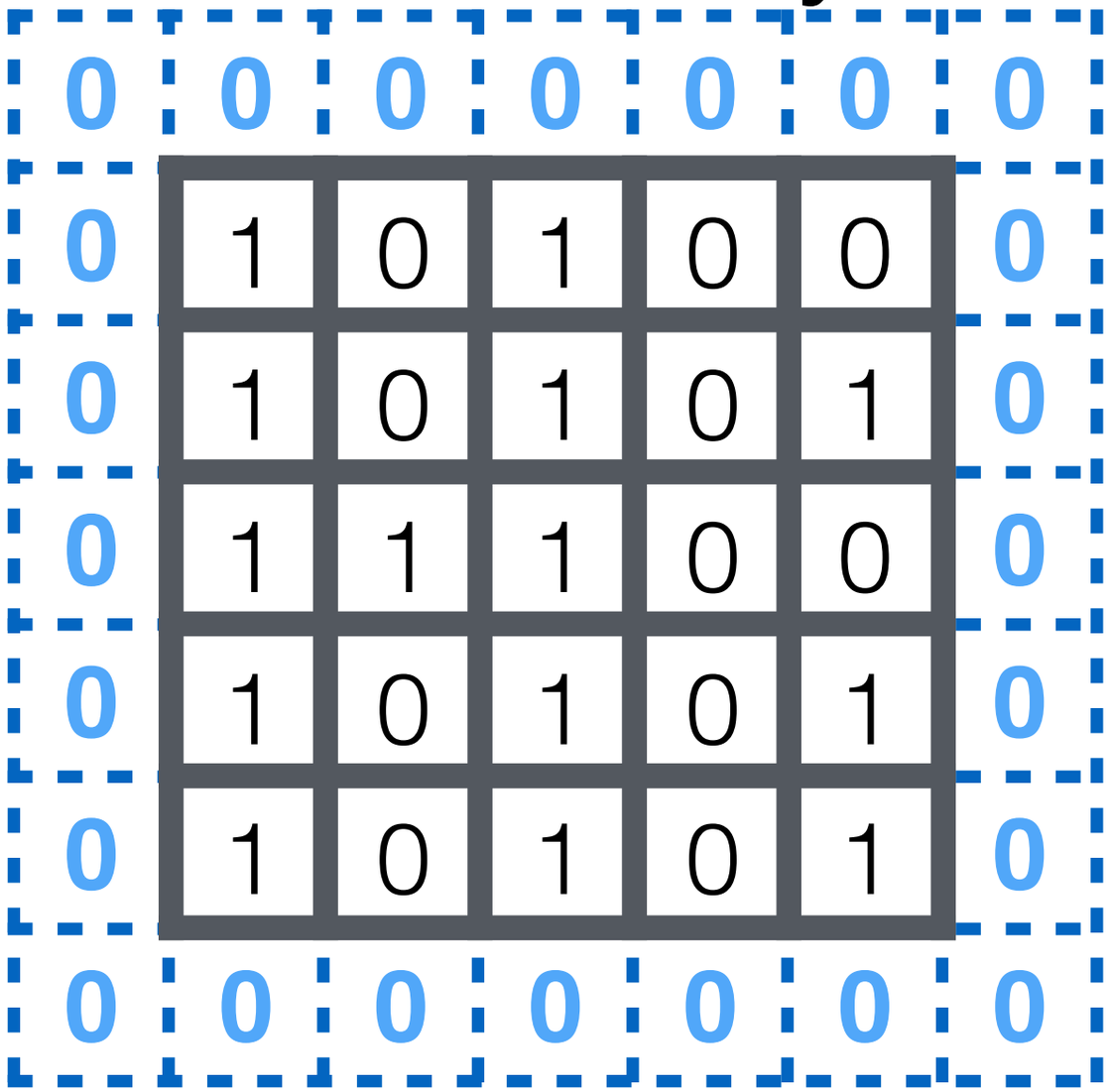


After convolution & ReLU:

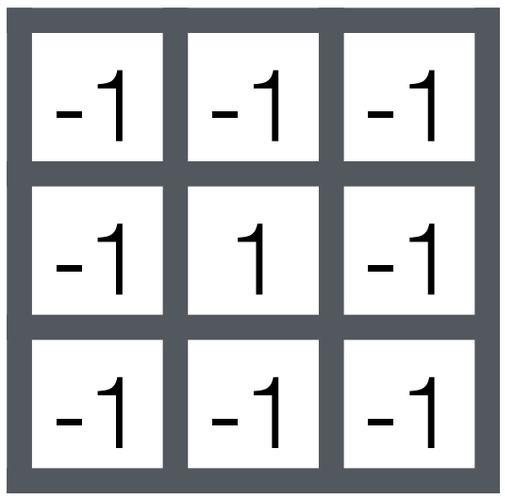


Convolutional Layer: 2D example

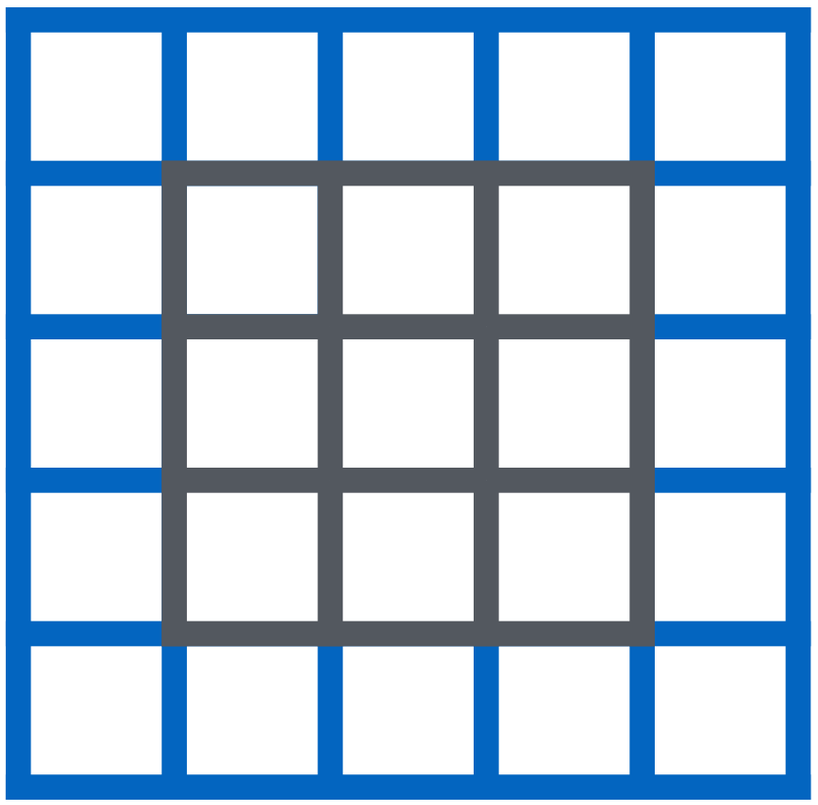
A 2D image:



A filter:

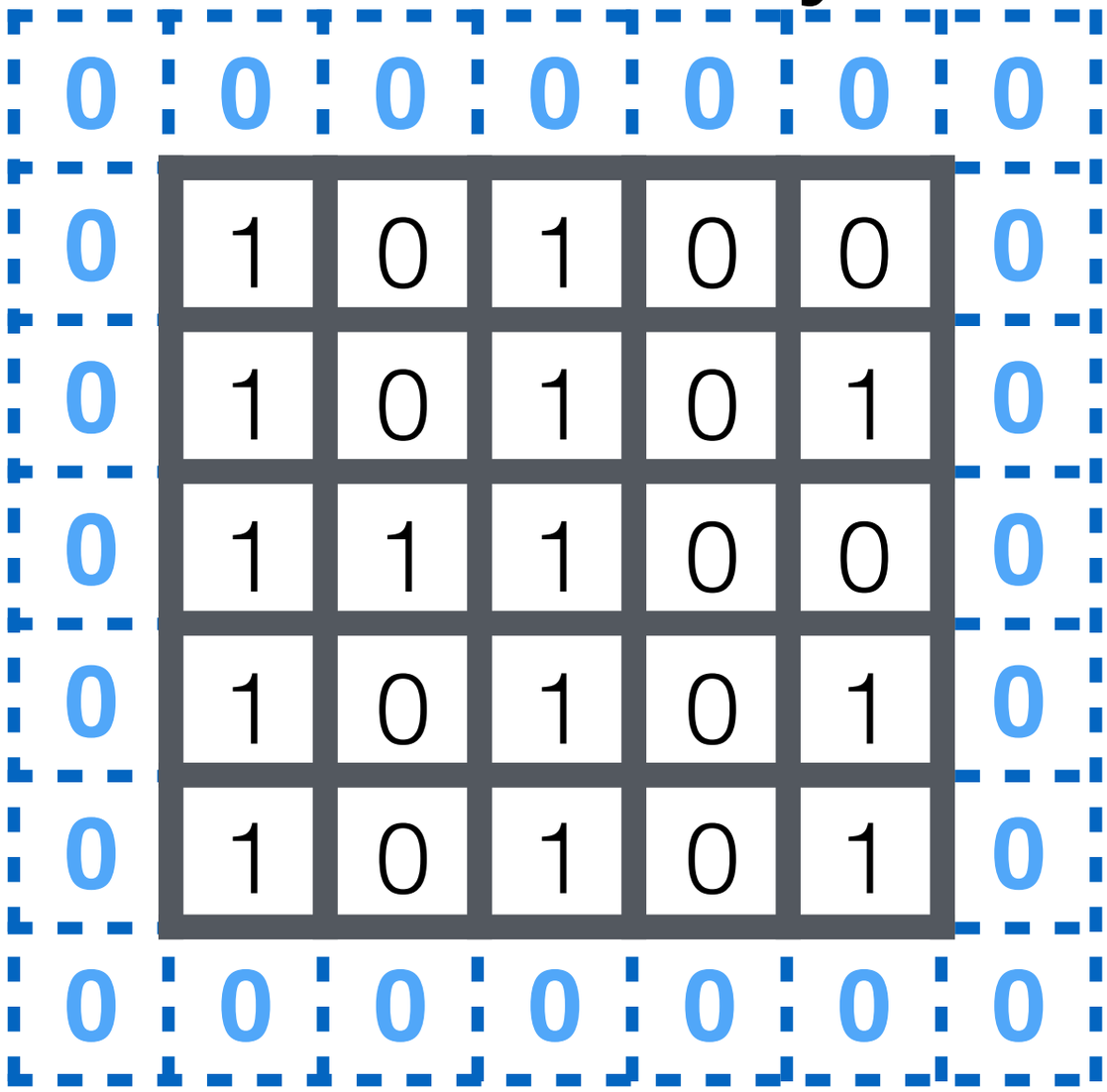


After convolution:

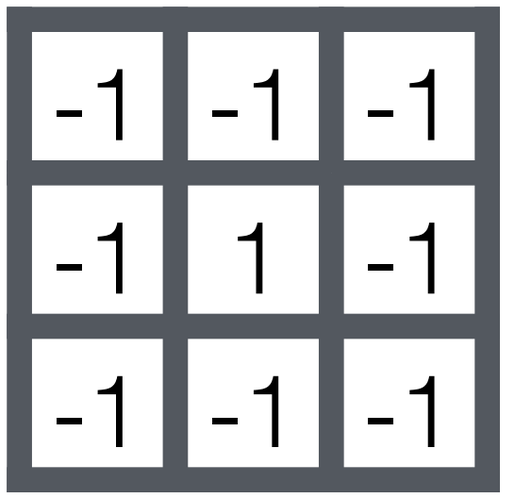


Convolutional Layer: 2D example

A 2D image:

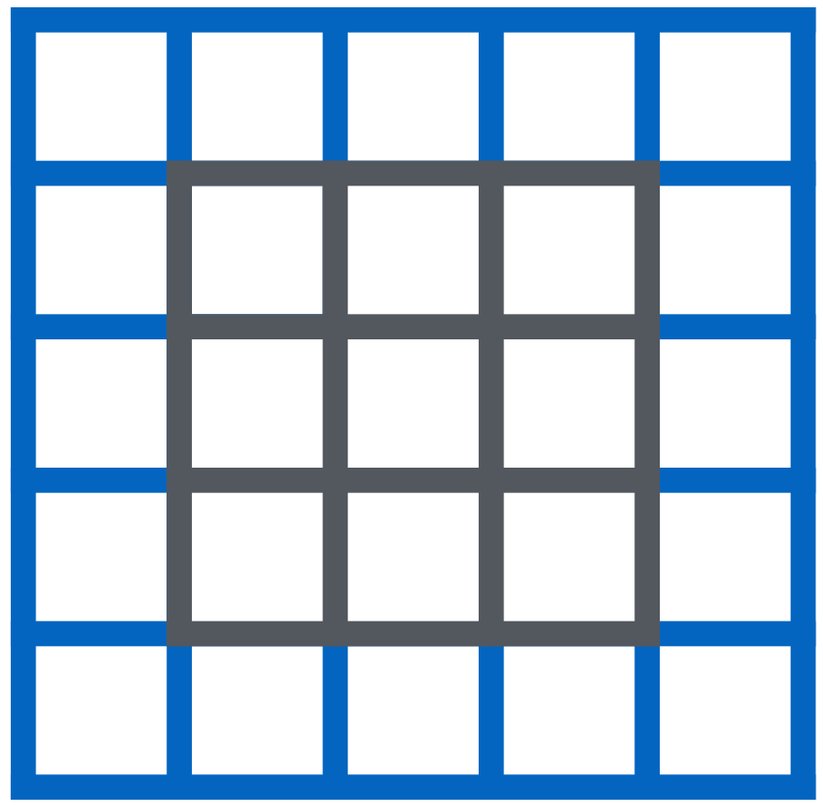


A filter:



with bias 2

After convolution:



Convolutional Layer: 2D example

A 2D image:

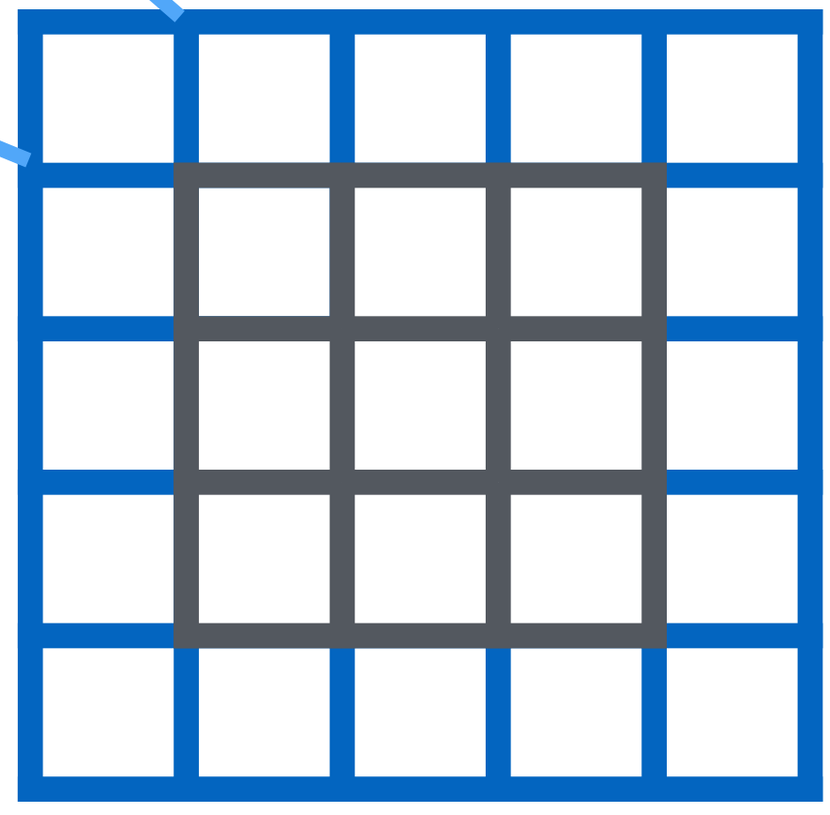
0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
0	1	0	1	0	1	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

A filter:

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

with bias 2

After convolution:



Convolutional Layer: 2D example

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

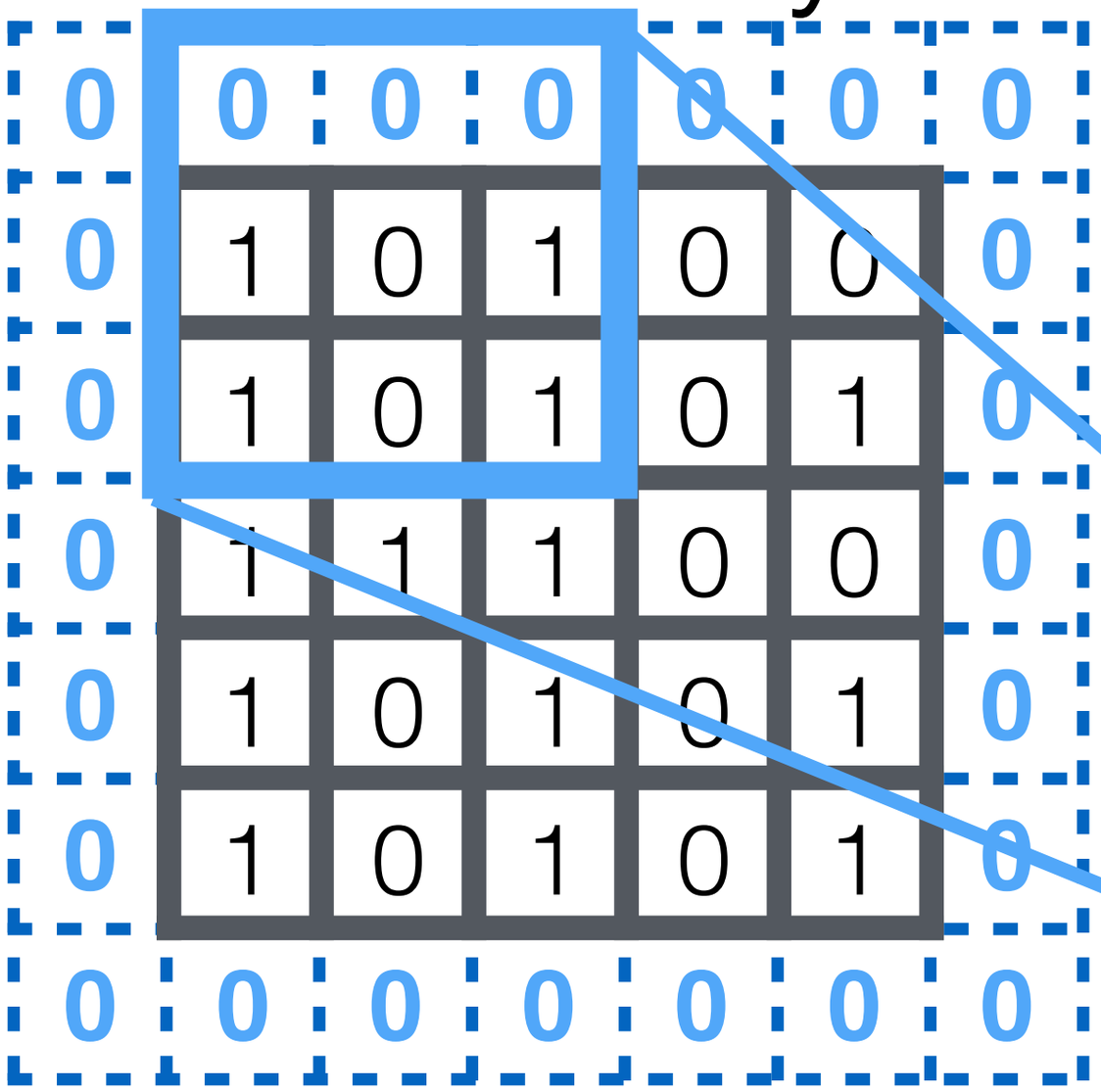
with bias 2

After convolution:

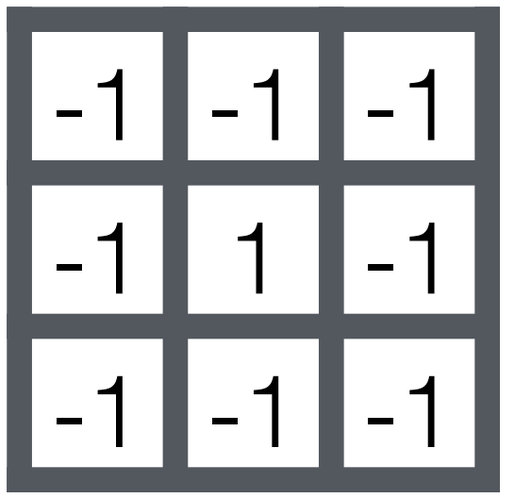
2				

Convolutional Layer: 2D example

A 2D image:

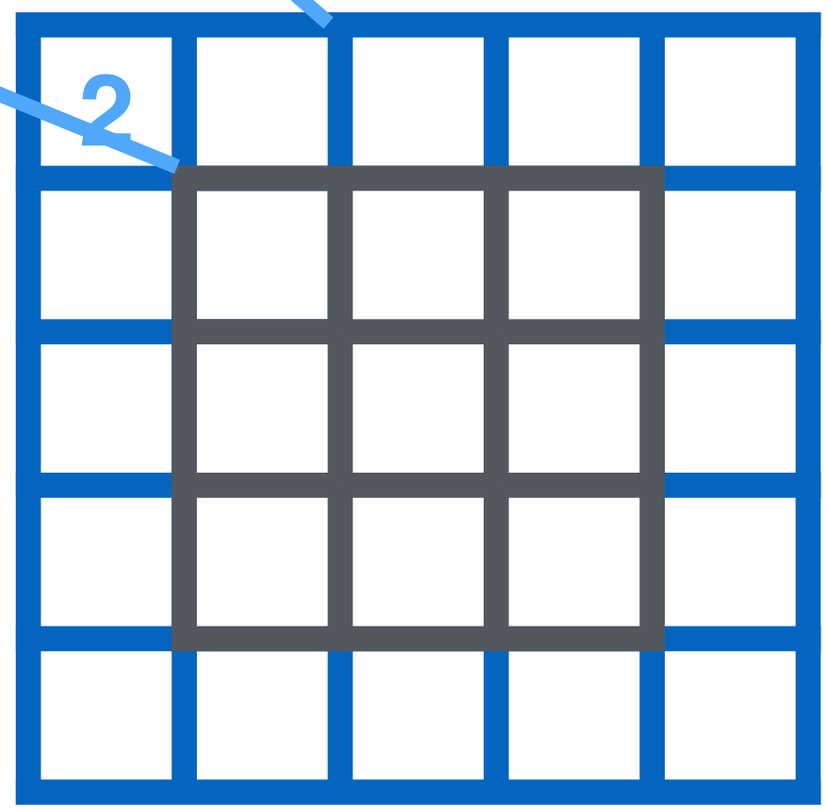


A filter:



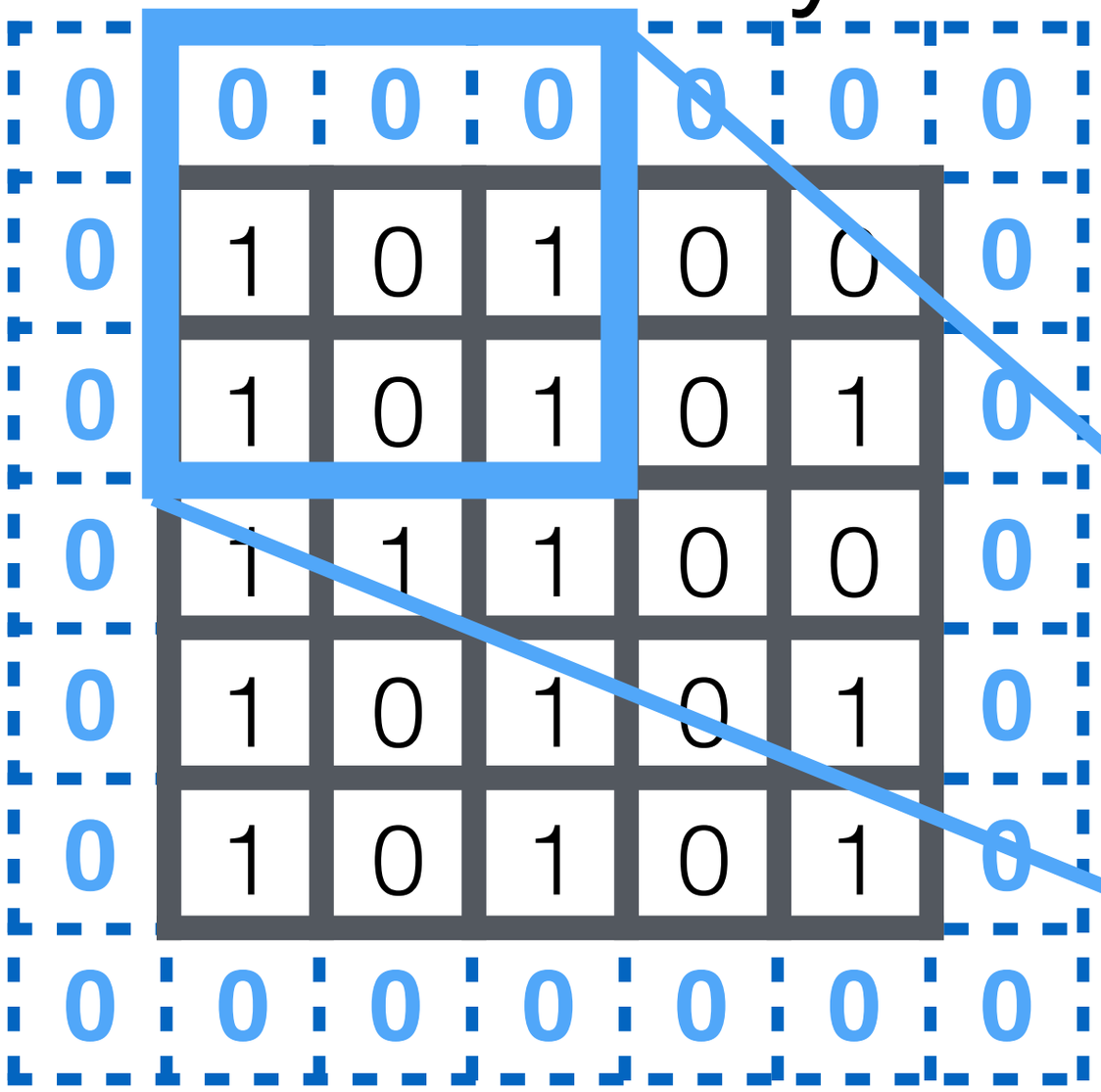
with bias 2

After convolution:

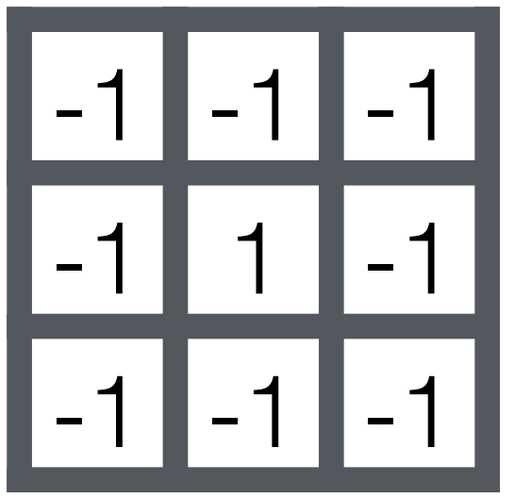


Convolutional Layer: 2D example

A 2D image:

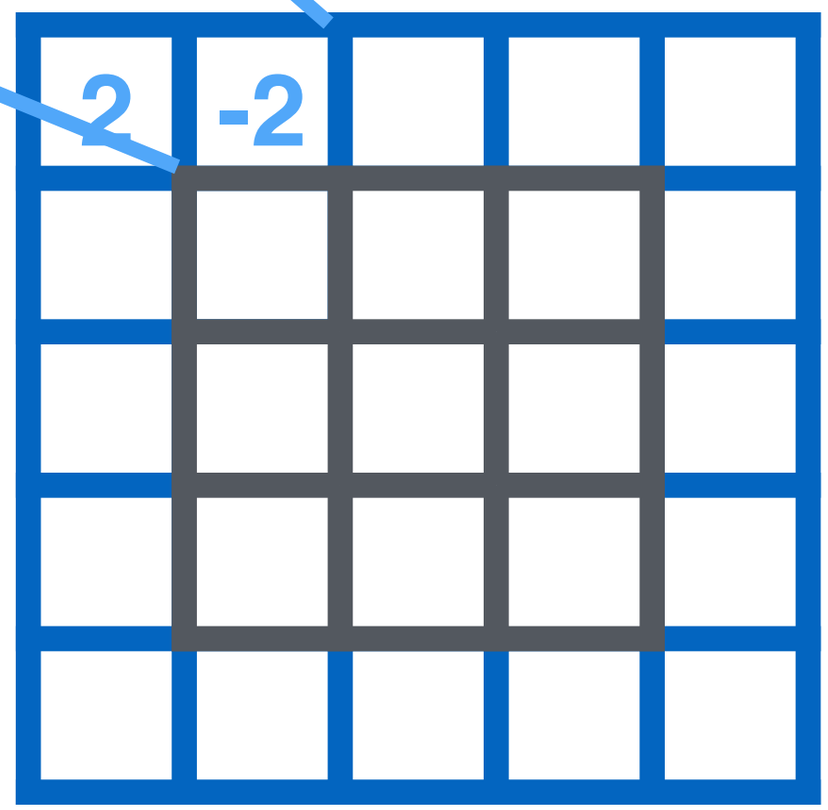


A filter:



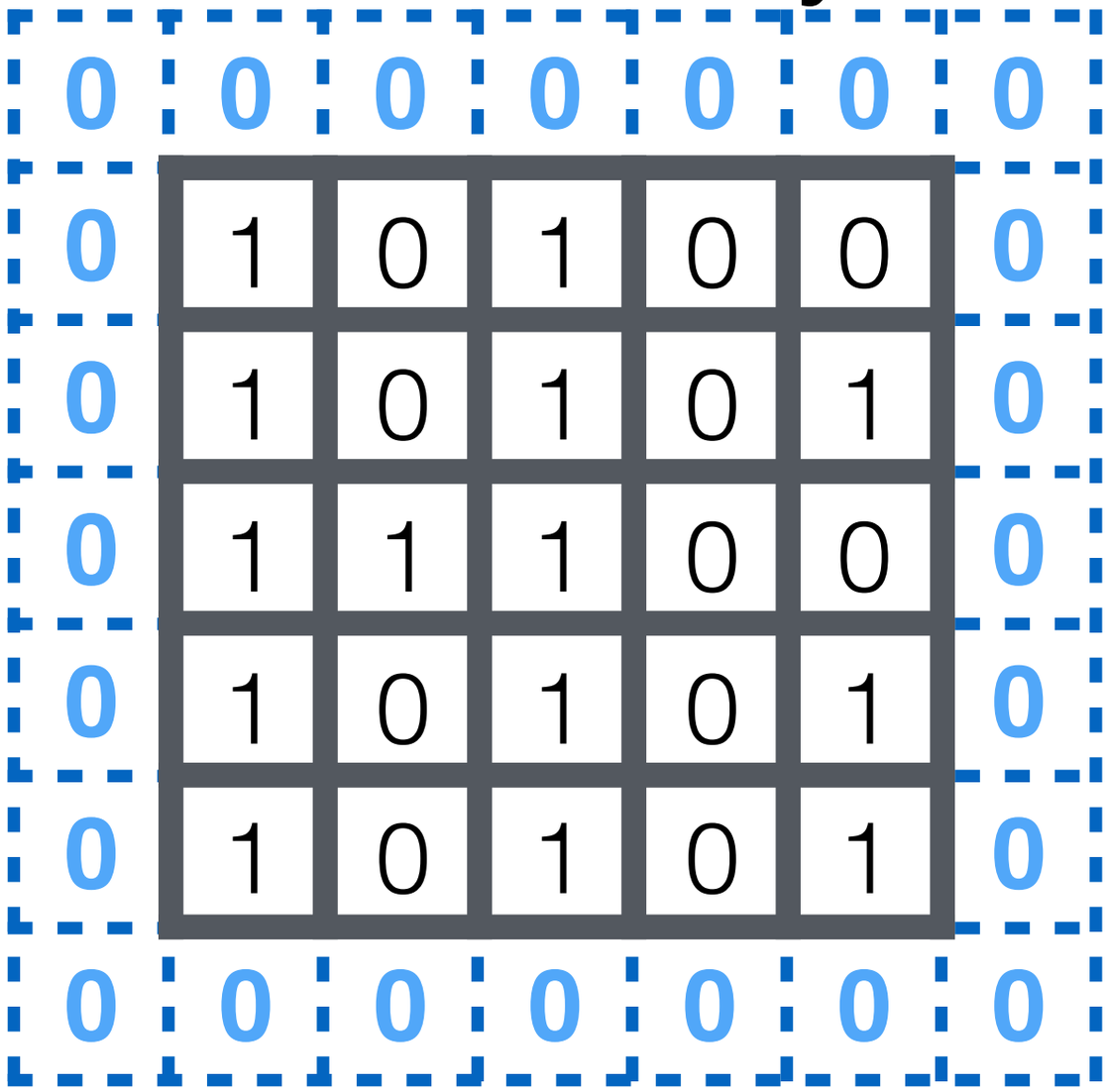
with bias 2

After convolution:

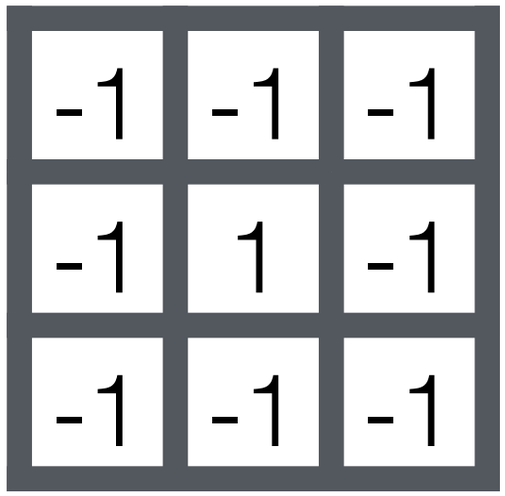


Convolutional Layer: 2D example

A 2D image:

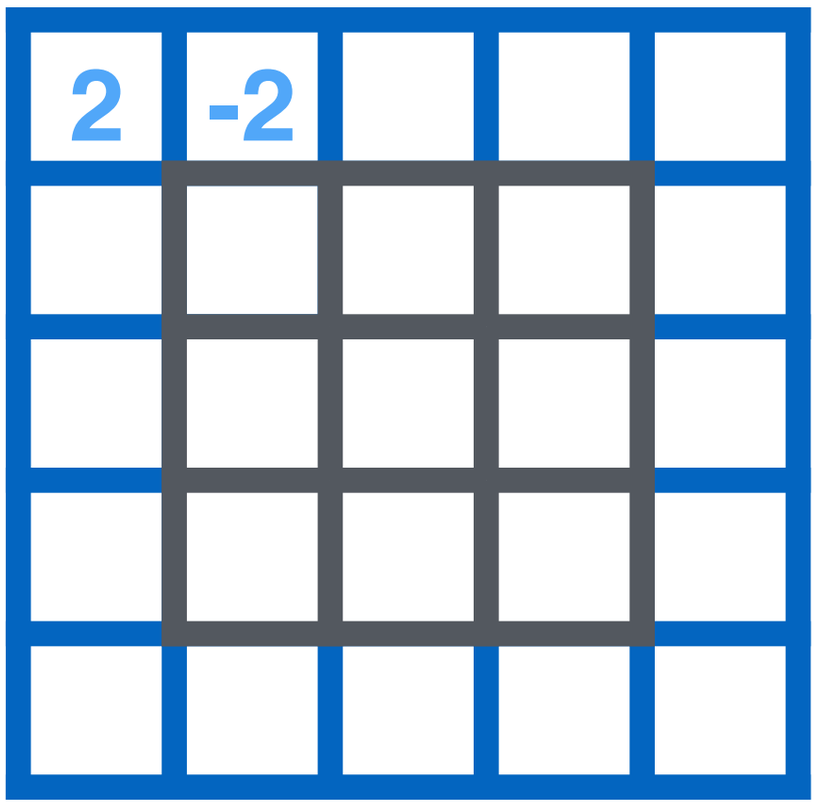


A filter:



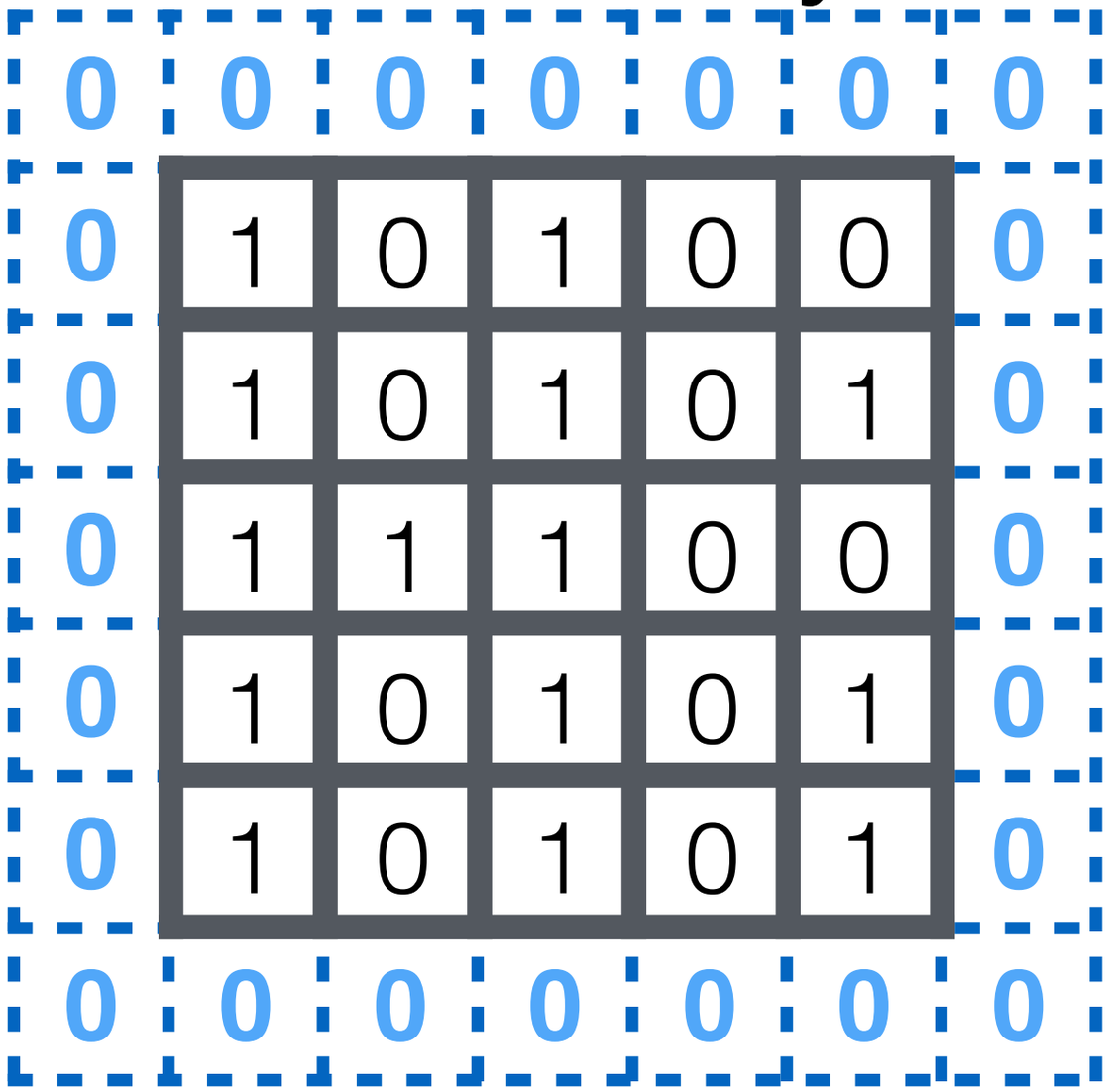
with bias 2

After convolution:

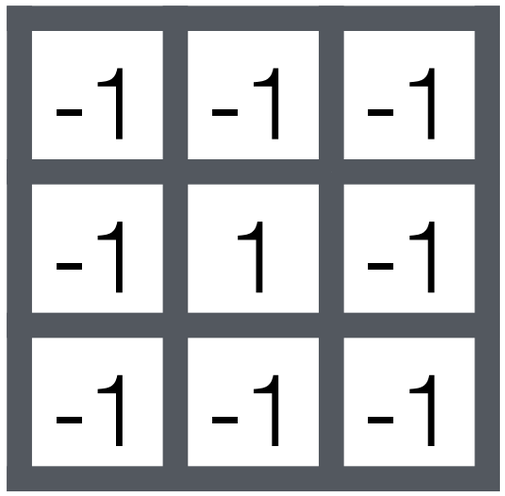


Convolutional Layer: 2D example

A 2D image:

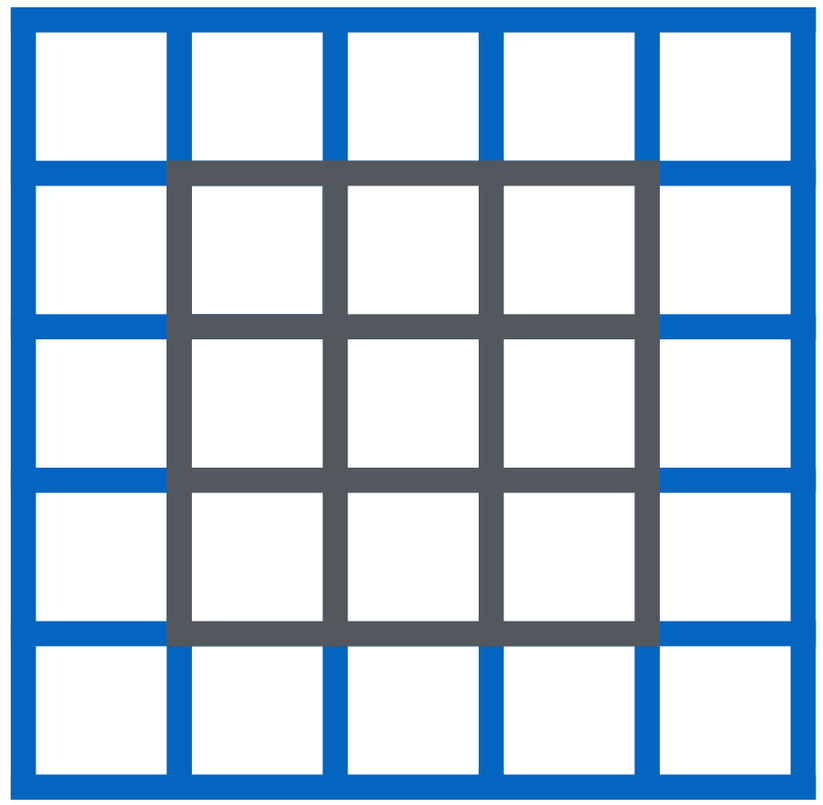


A filter:



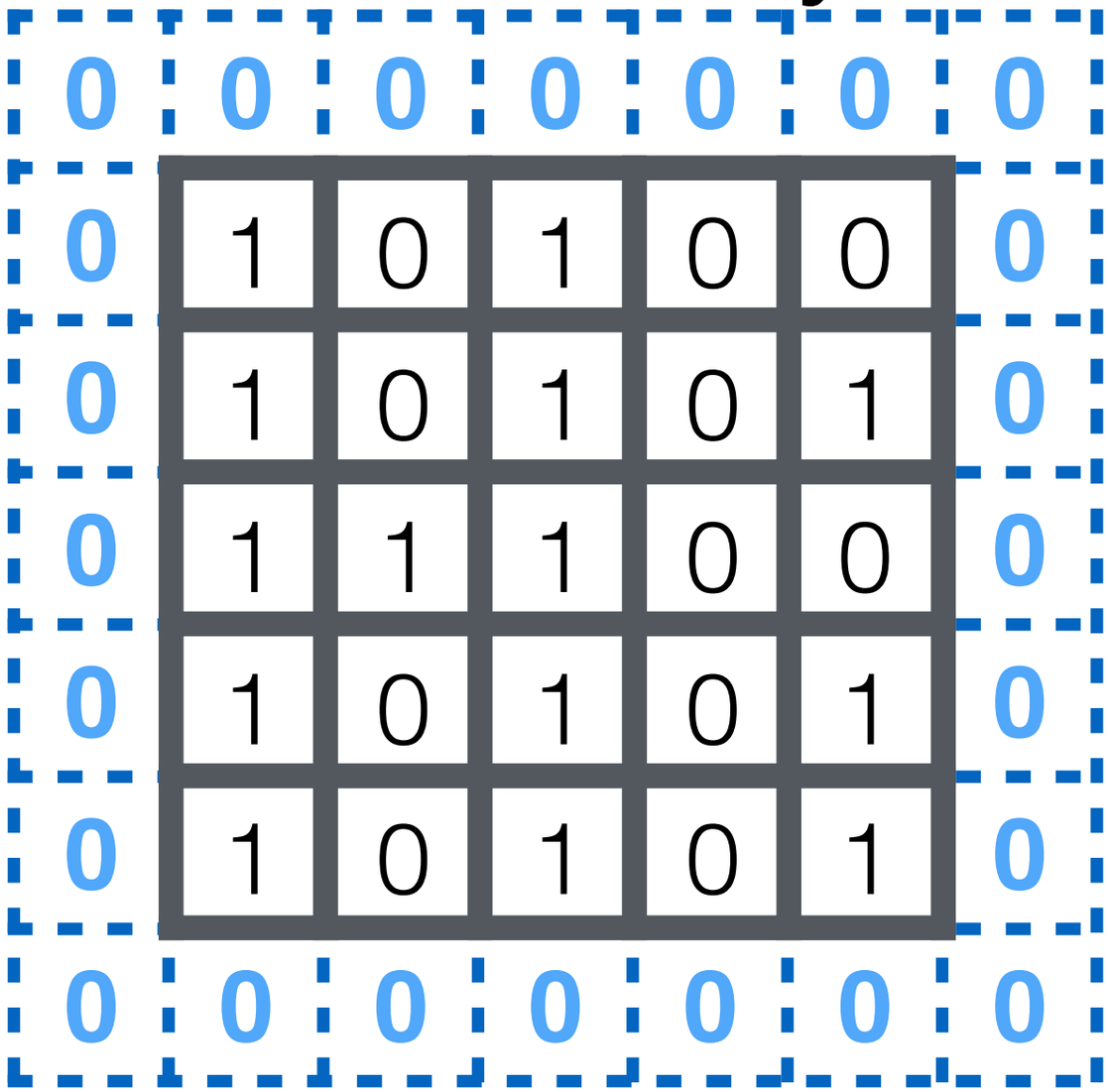
with bias 2

After convolution:

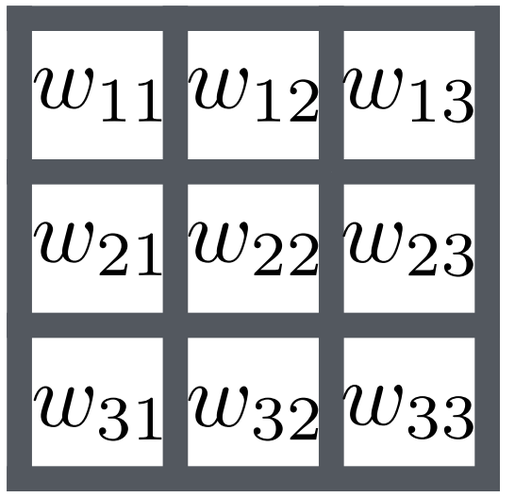


Convolutional Layer: 2D example

A 2D image:

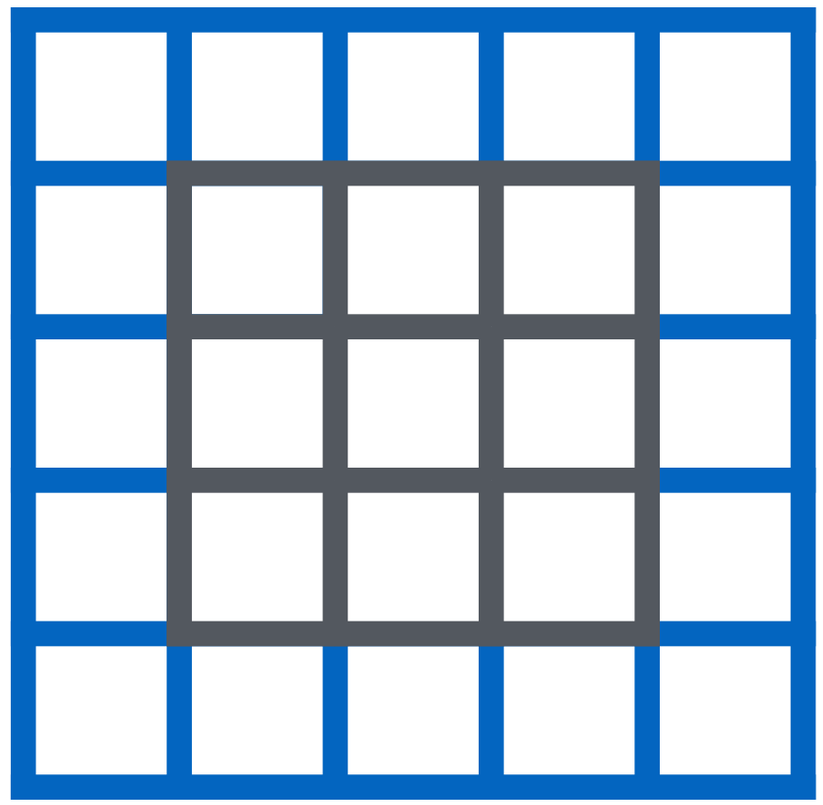


A filter:



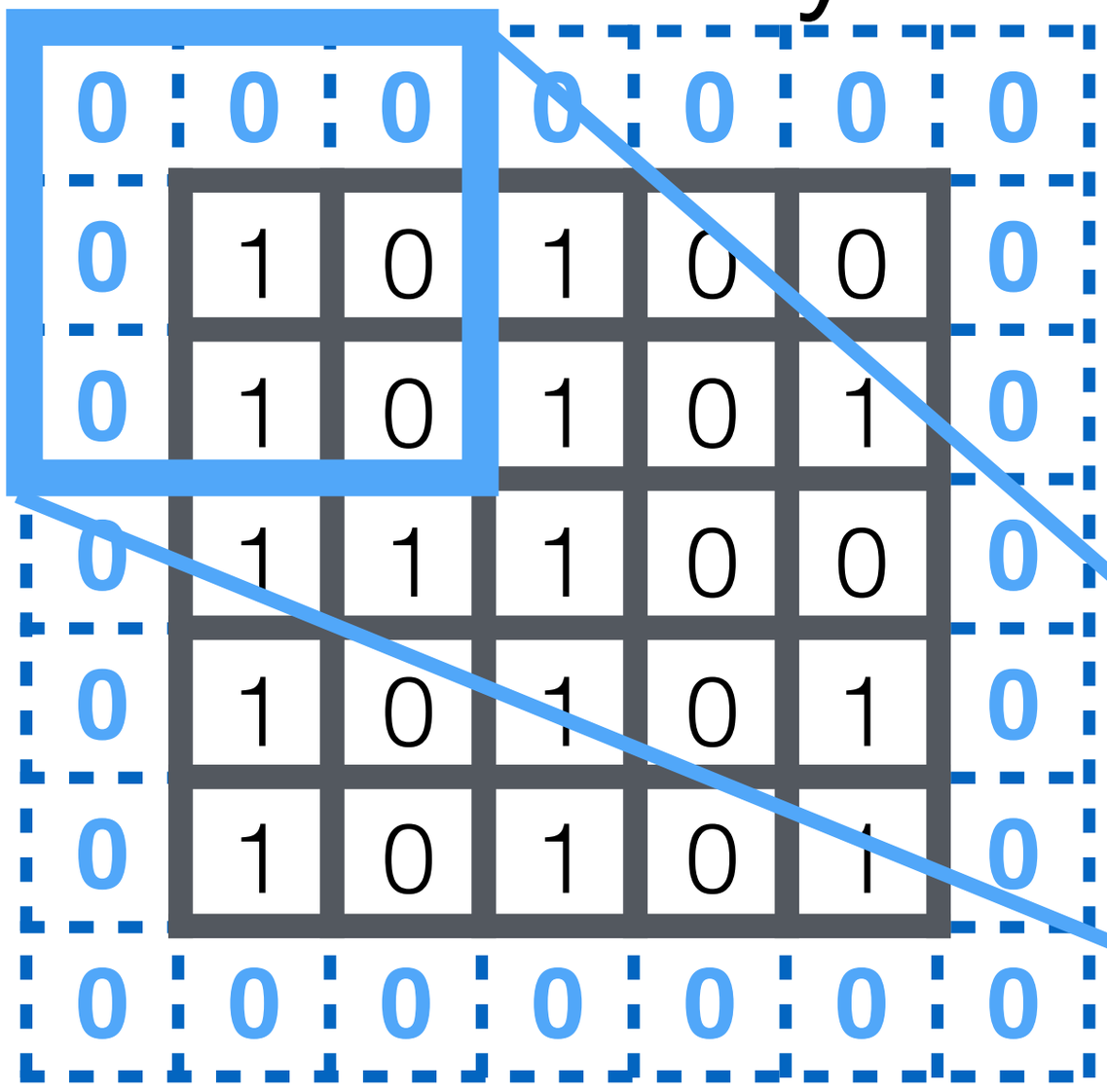
with bias b

After convolution:



Convolutional Layer: 2D example

A 2D image:

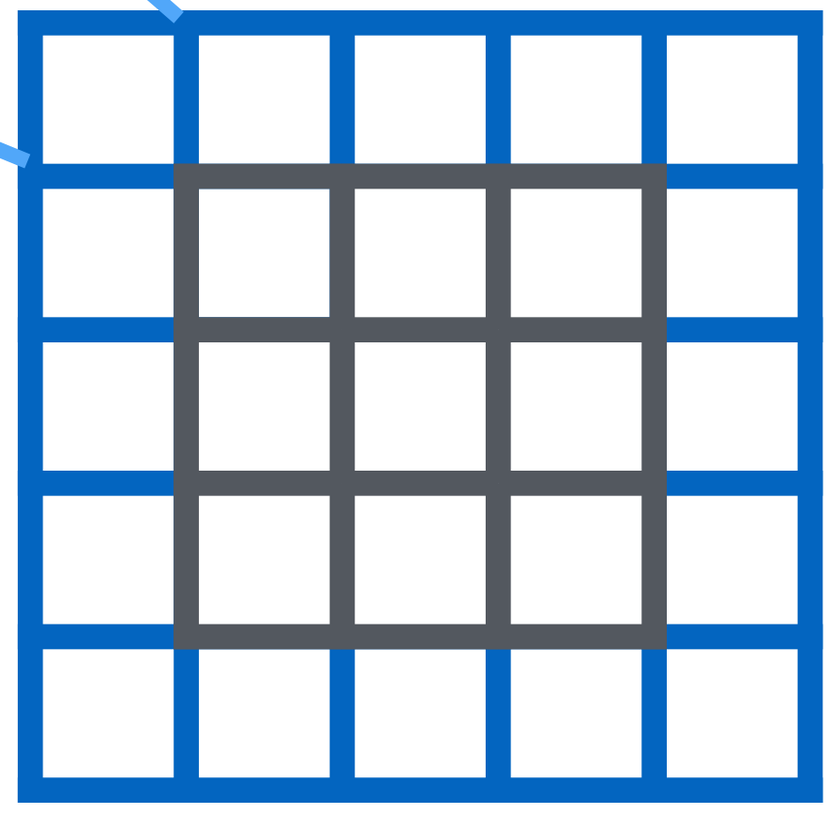


A filter:

$$\begin{matrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{matrix}$$

with bias b

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:

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

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_{11}	w_{12}
w_{21}	w_{22}

with bias b

Convolutional Layer: 3D example

A 3D image:

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

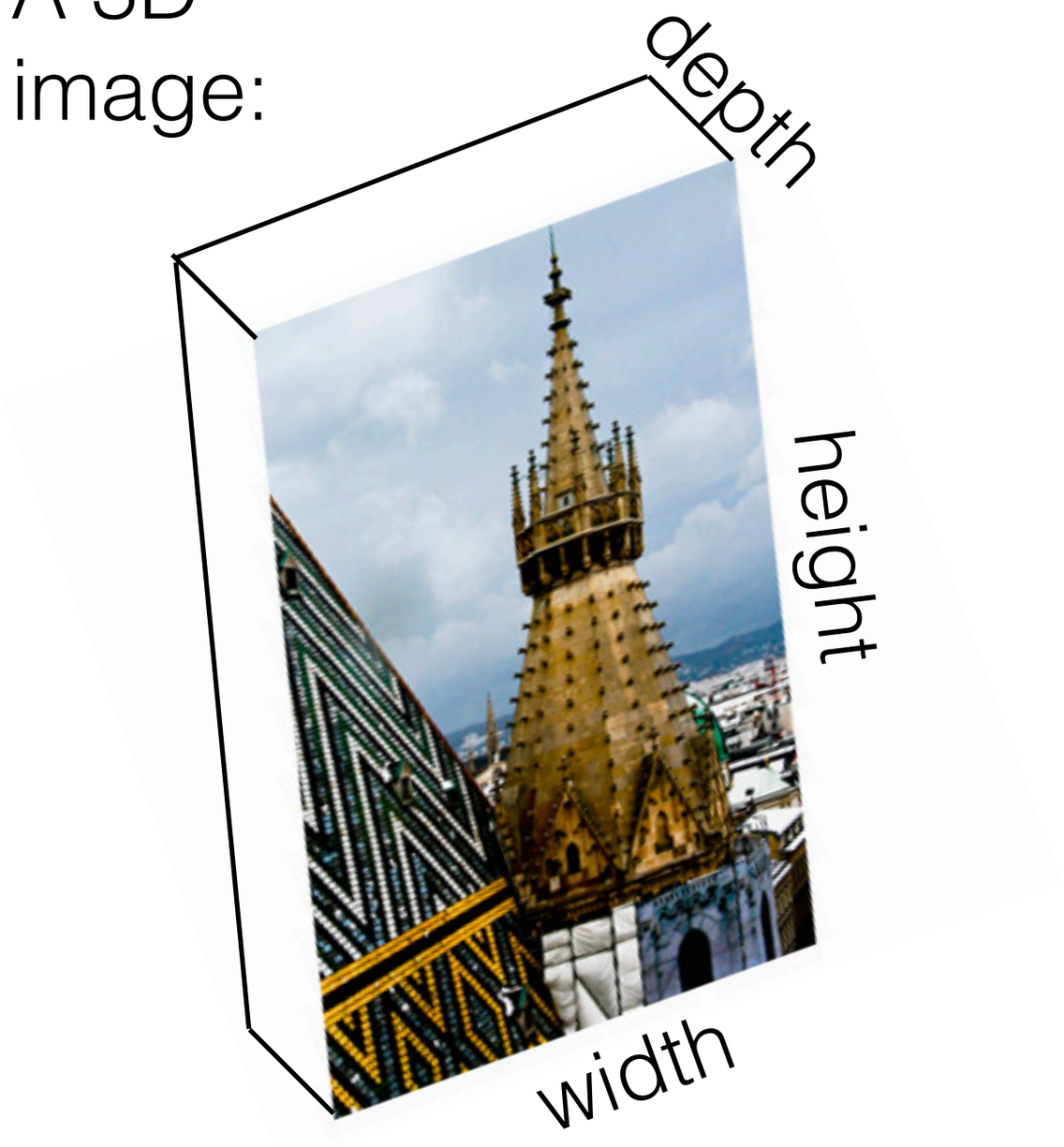
Convolutional Layer: 3D example

A 3D image:

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

Convolutional Layer: 3D example

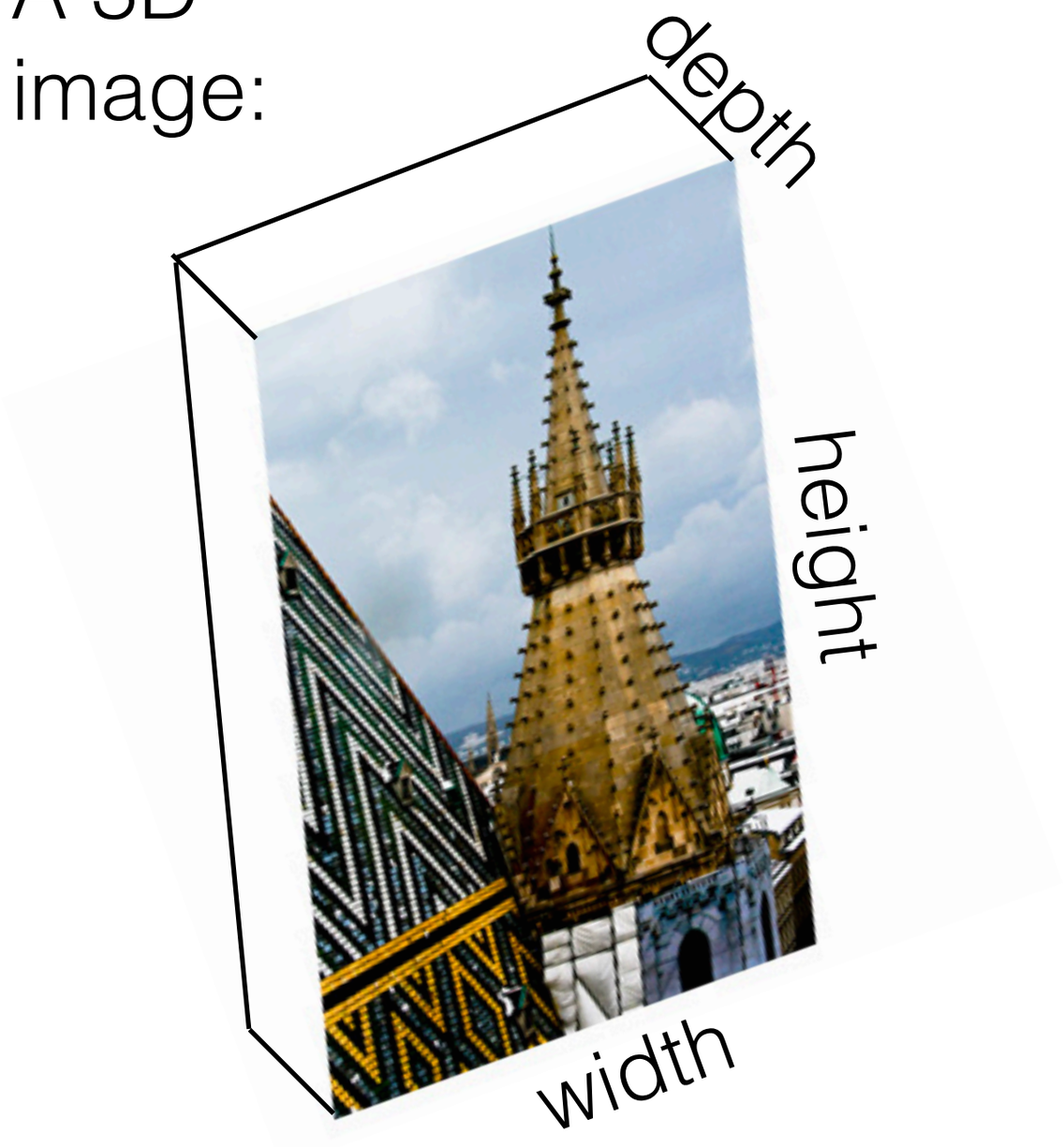
A 3D image:



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

Convolutional Layer: 3D example

A 3D image:

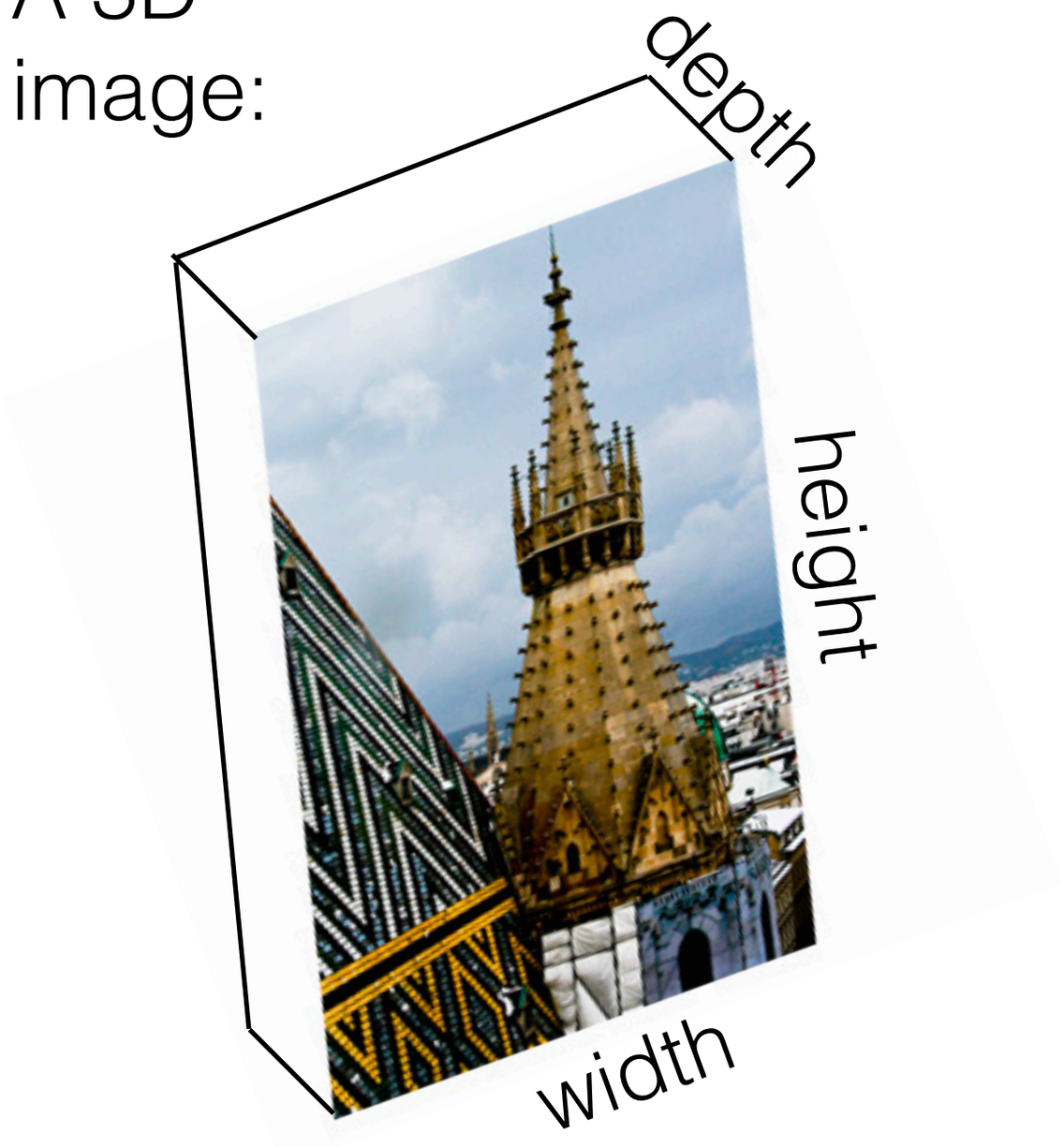


[https://helpx.adobe.com/photoshop/key-concepts/skew.html]

- 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]

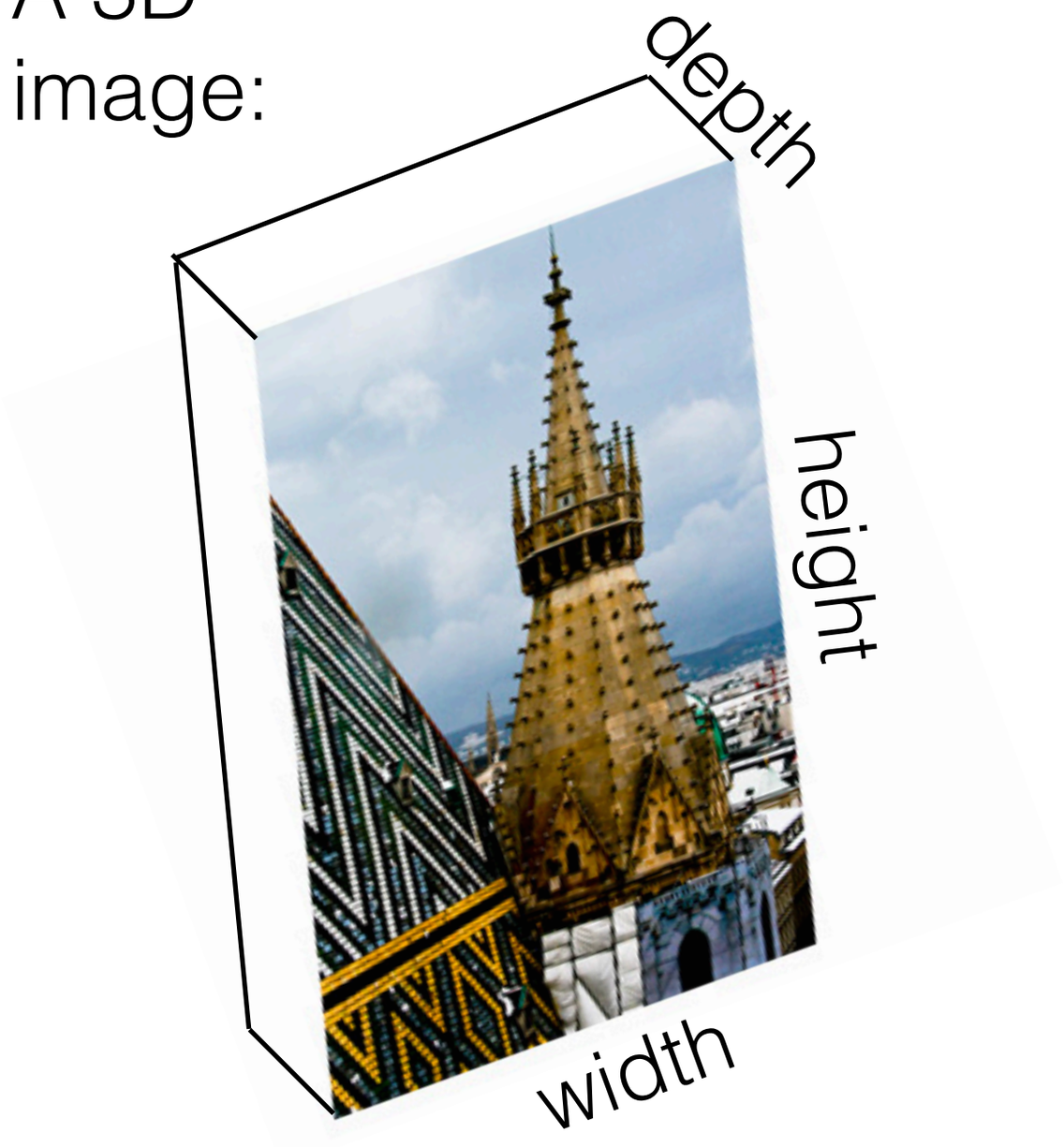
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

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

TensorFlow

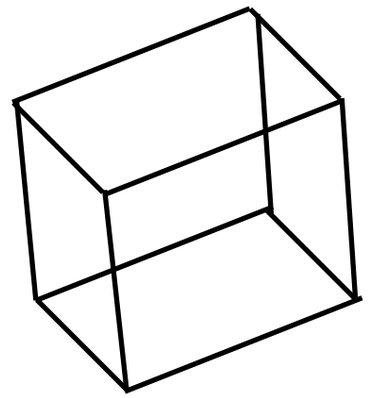
Convolutional Layer: 3D example

A 3D image:



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

A filter:

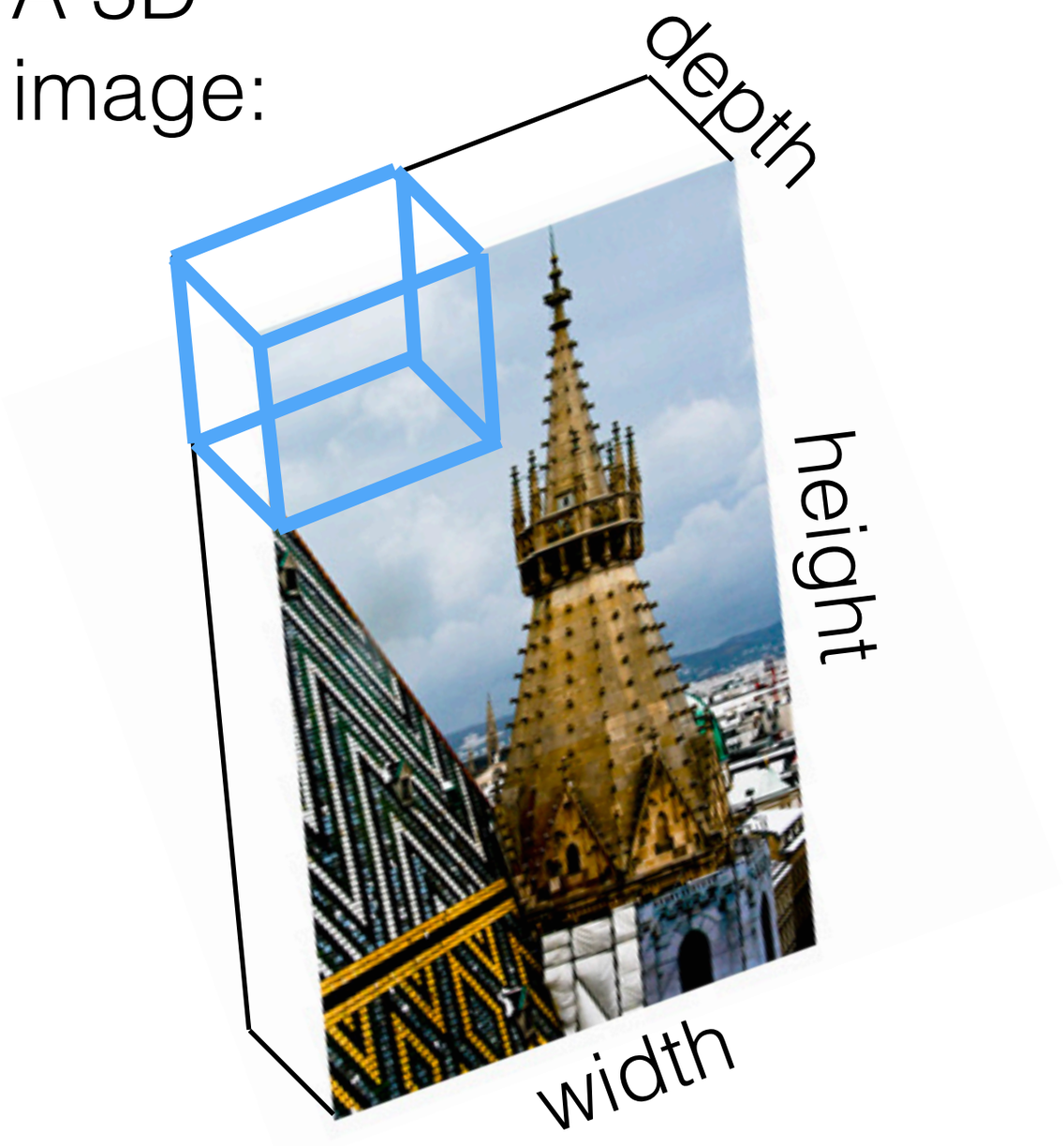


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

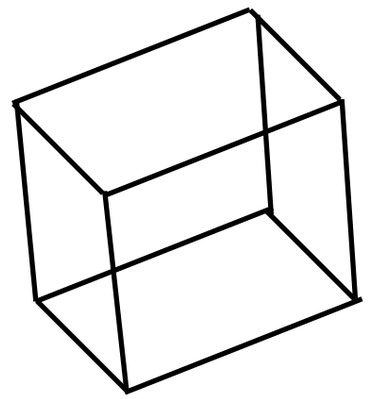
[https://en.wikipedia.org/wiki/TensorFlow]

Convolutional Layer: 3D example

A 3D image:



A filter:



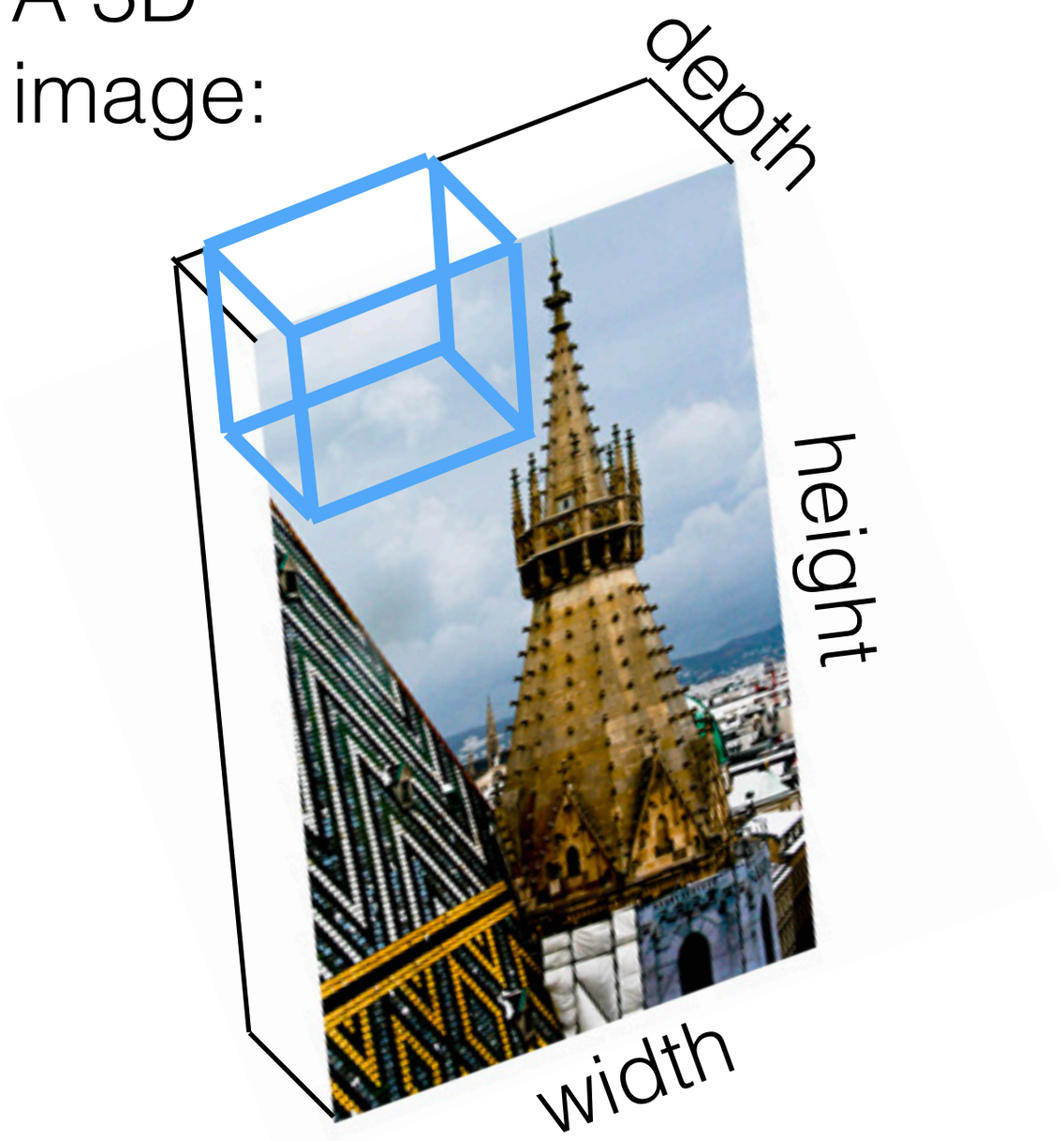
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

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

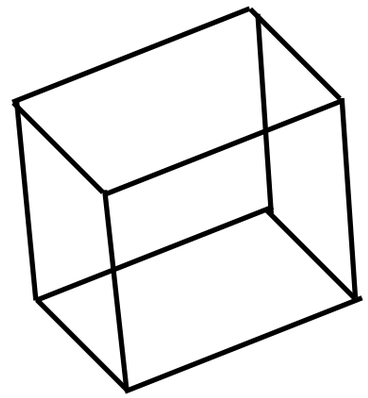
[https://en.wikipedia.org/wiki/TensorFlow]

Convolutional Layer: 3D example

A 3D image:



A filter:



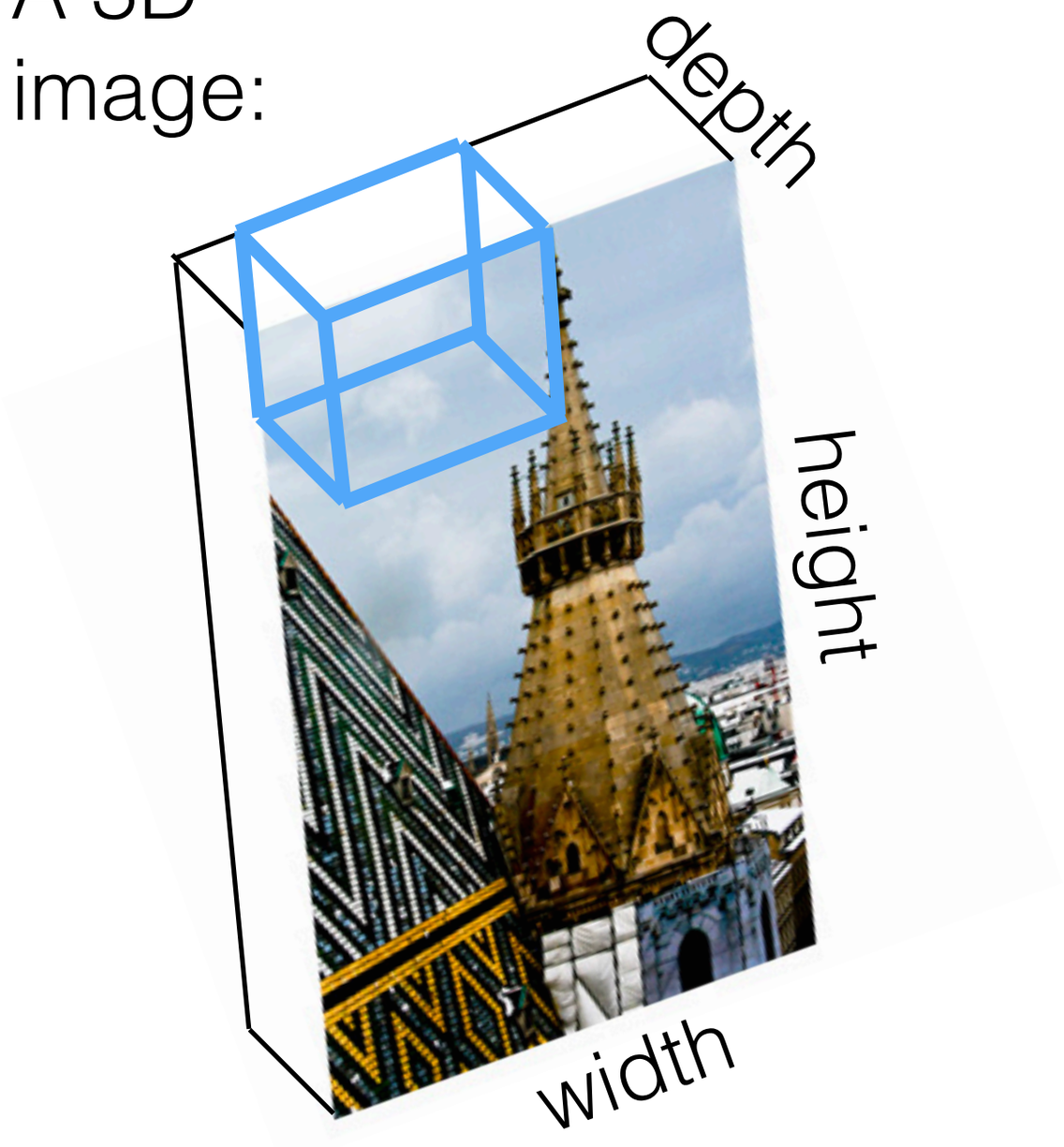
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

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

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

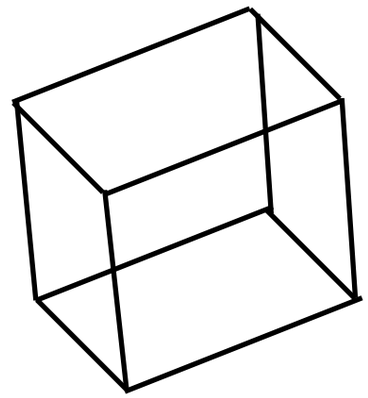
Convolutional Layer: 3D example

A 3D image:



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

A filter:

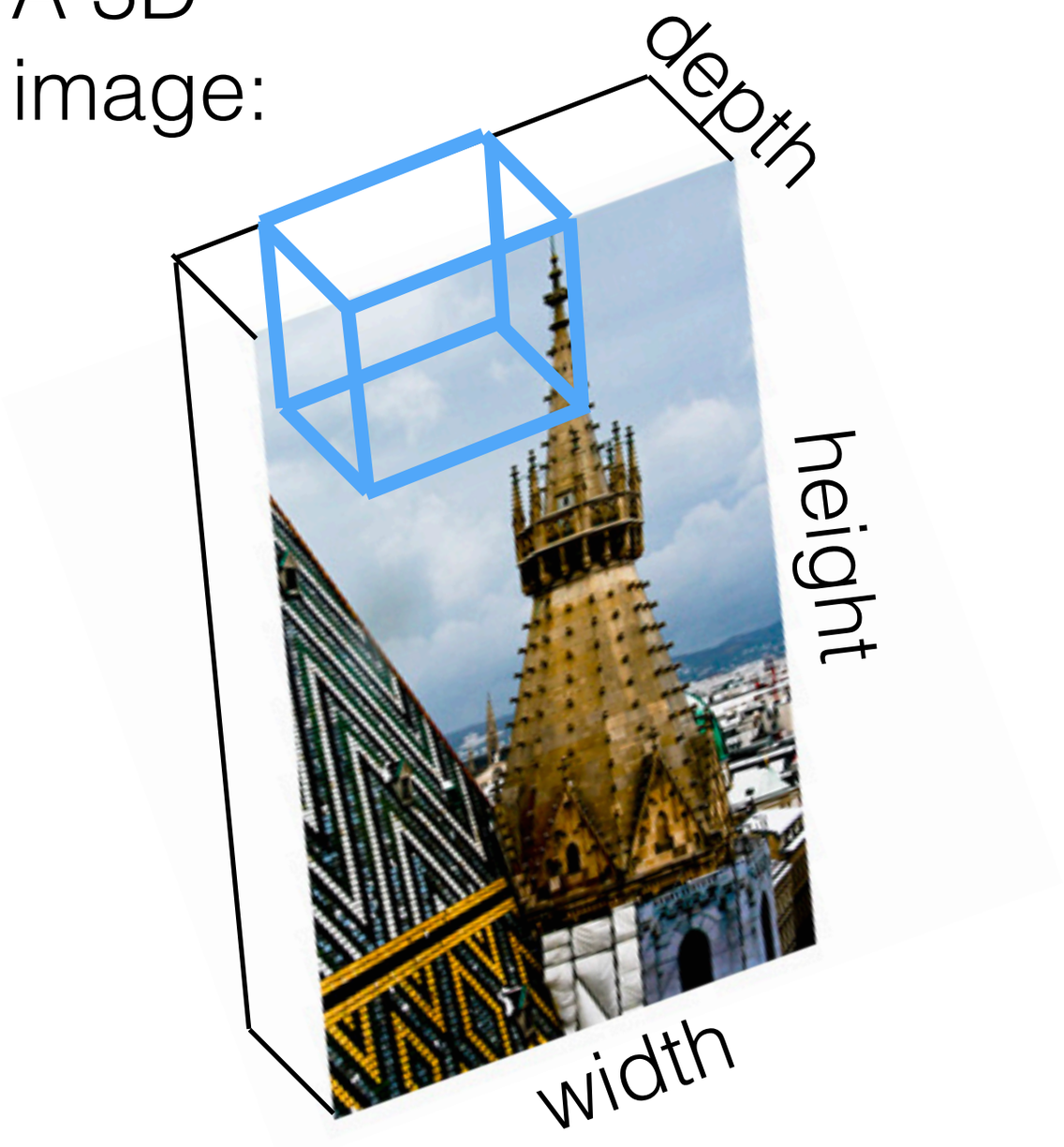


- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

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

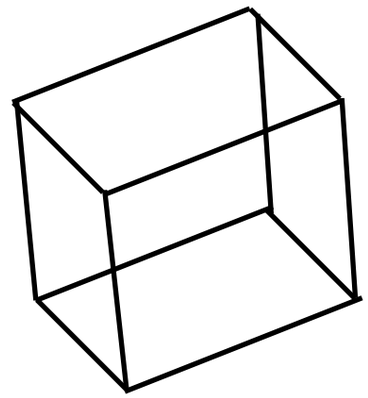
Convolutional Layer: 3D example

A 3D image:



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

A filter:



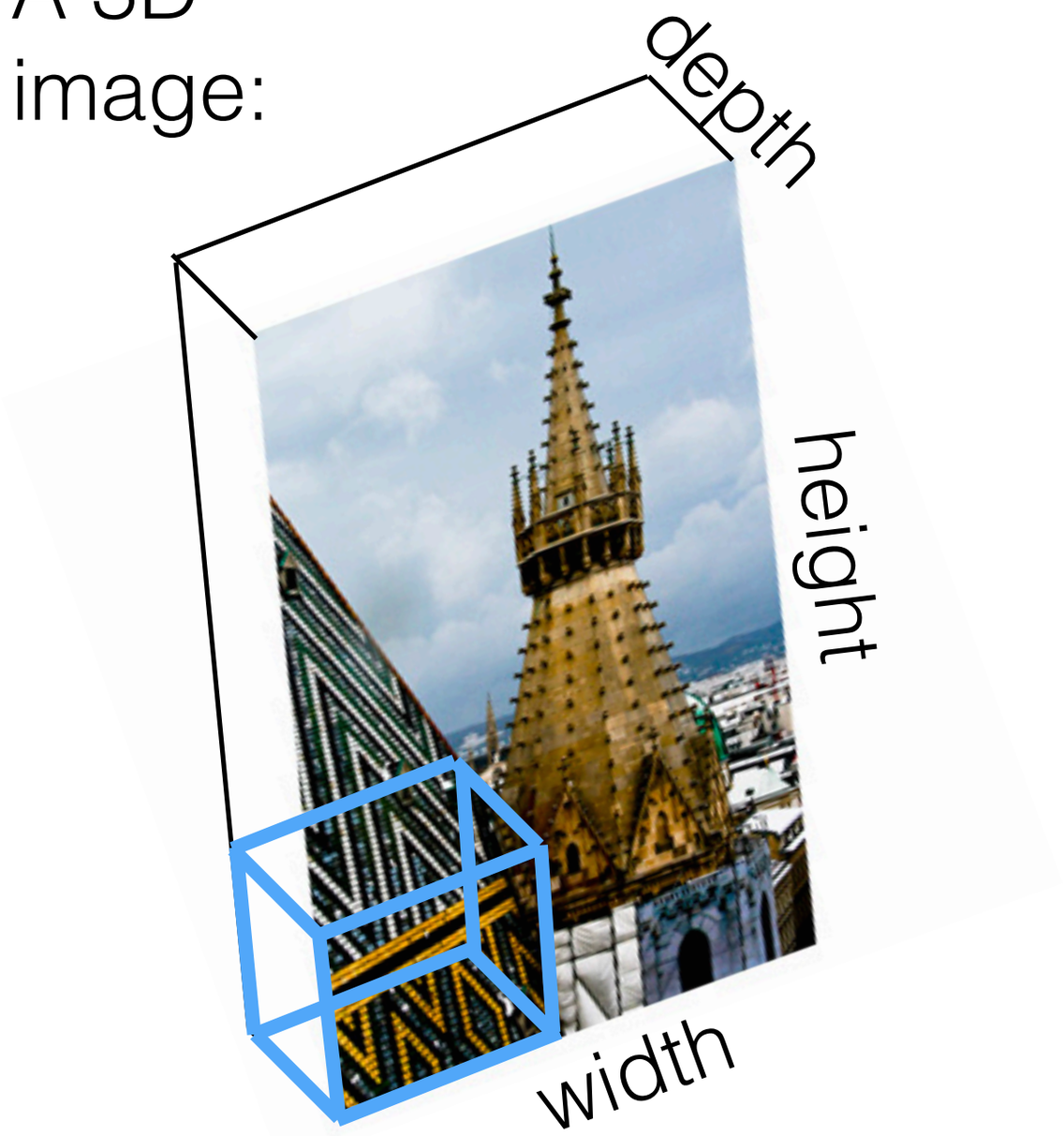
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

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

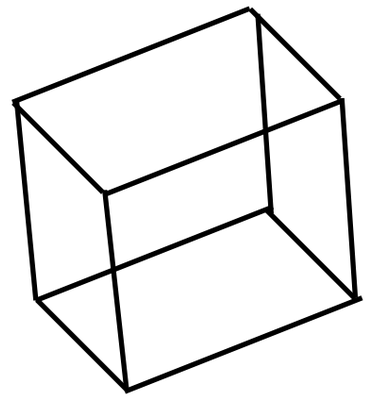
TensorFlow

Convolutional Layer: 3D example

A 3D image:



A filter:



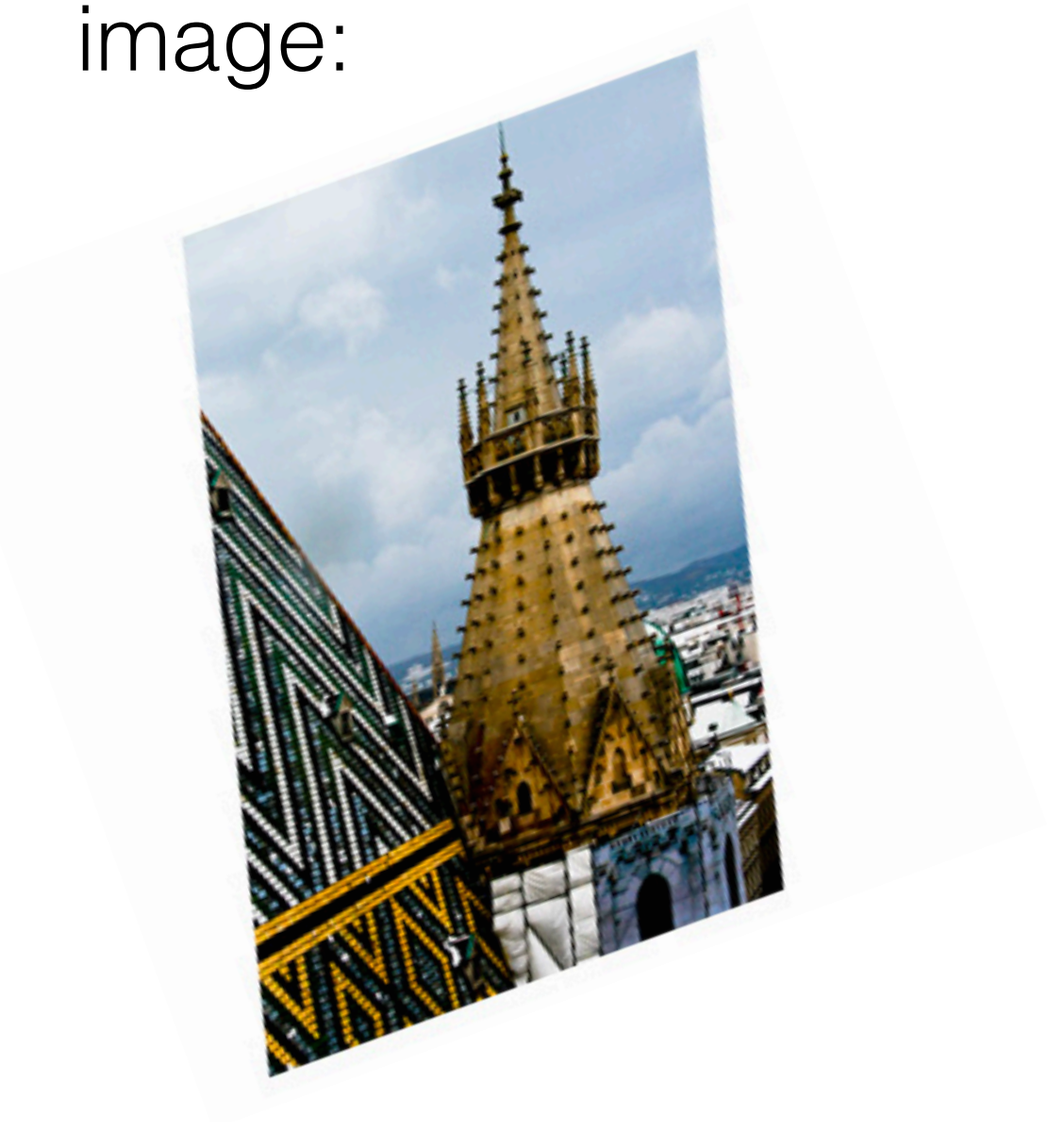
- Tensor: generalization of a matrix
  - E.g. 1D: vector, 2D: matrix

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

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

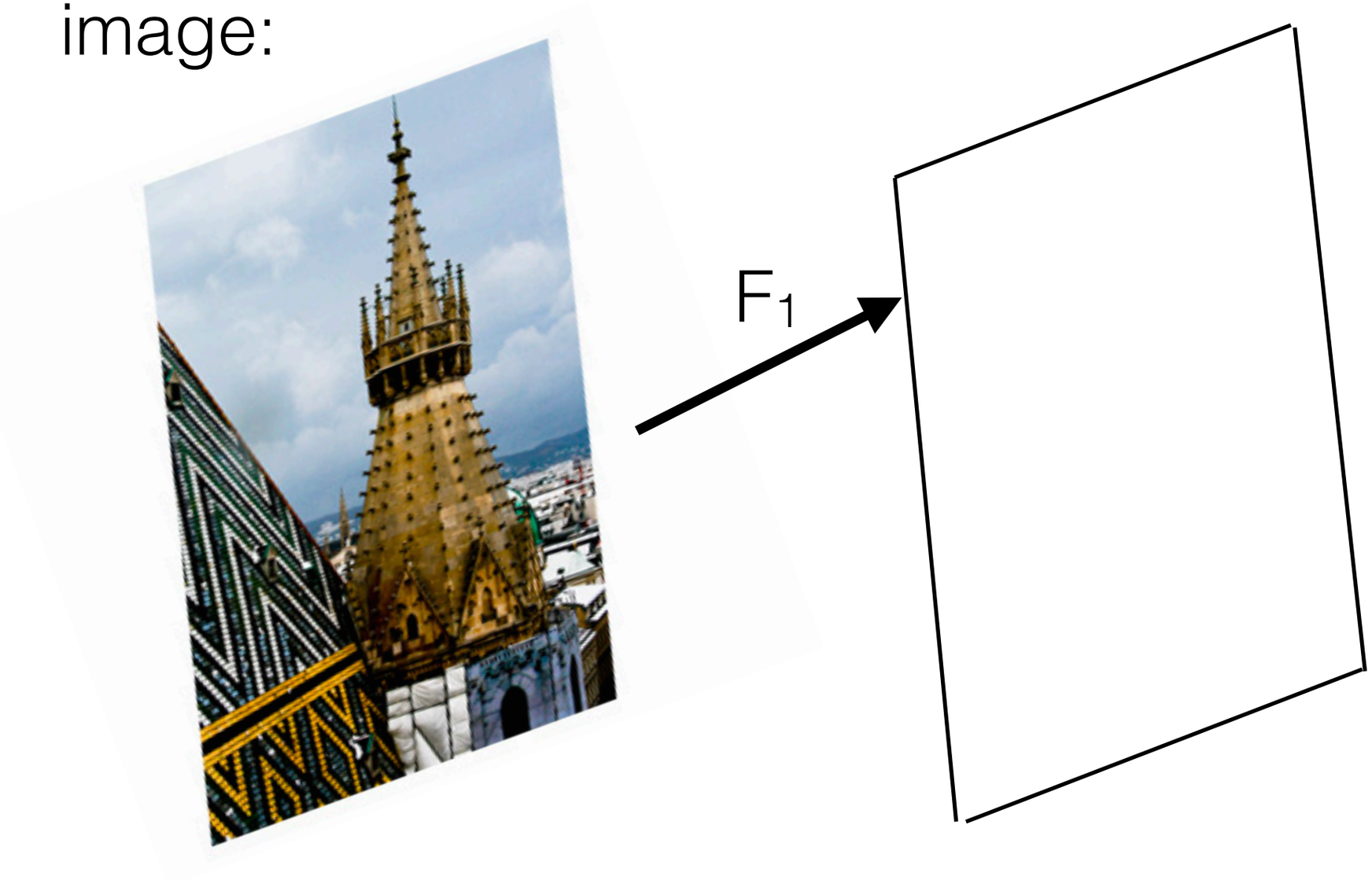
Convolutional Layer: multiple filters

An image:



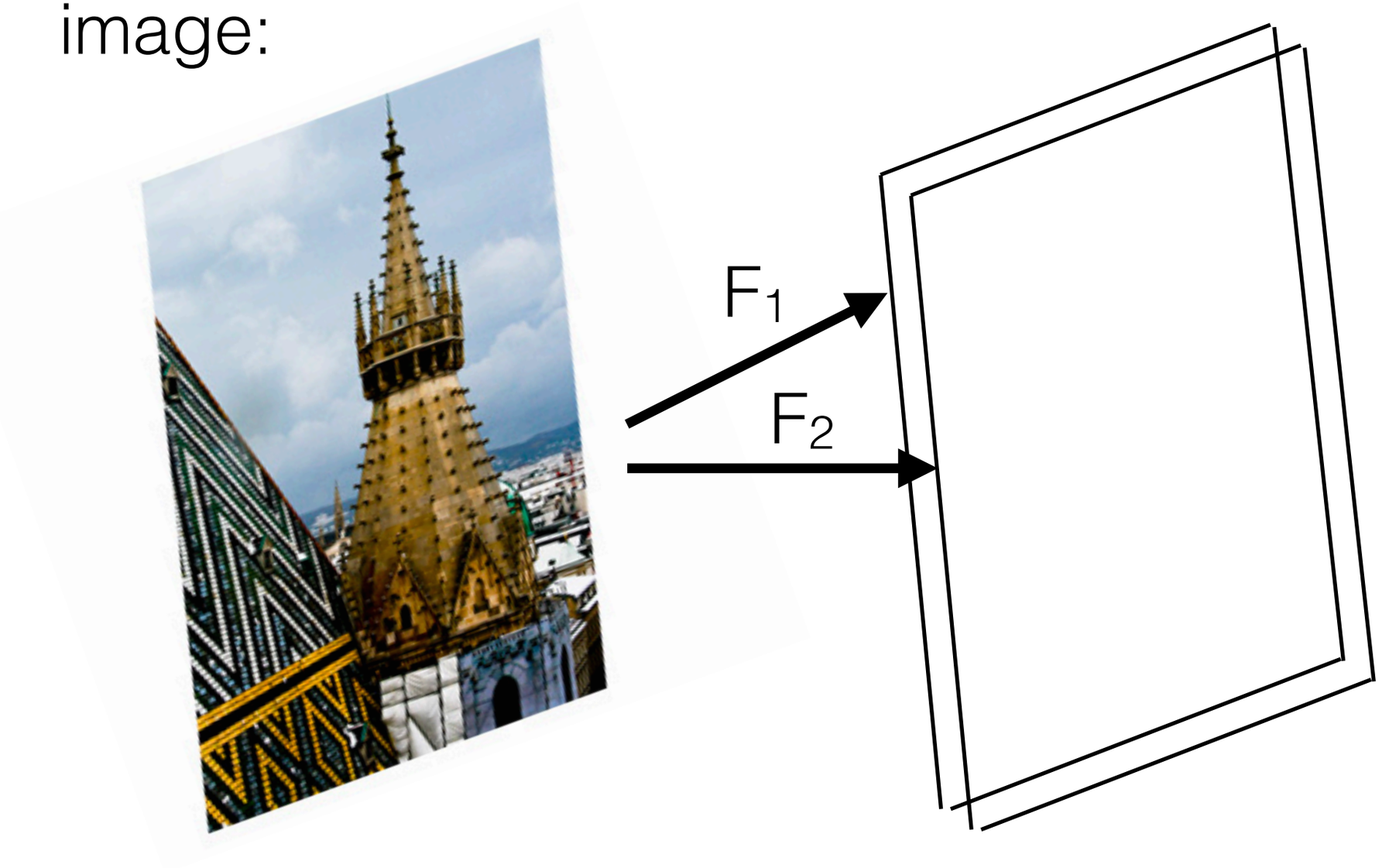
Convolutional Layer: multiple filters

An image:



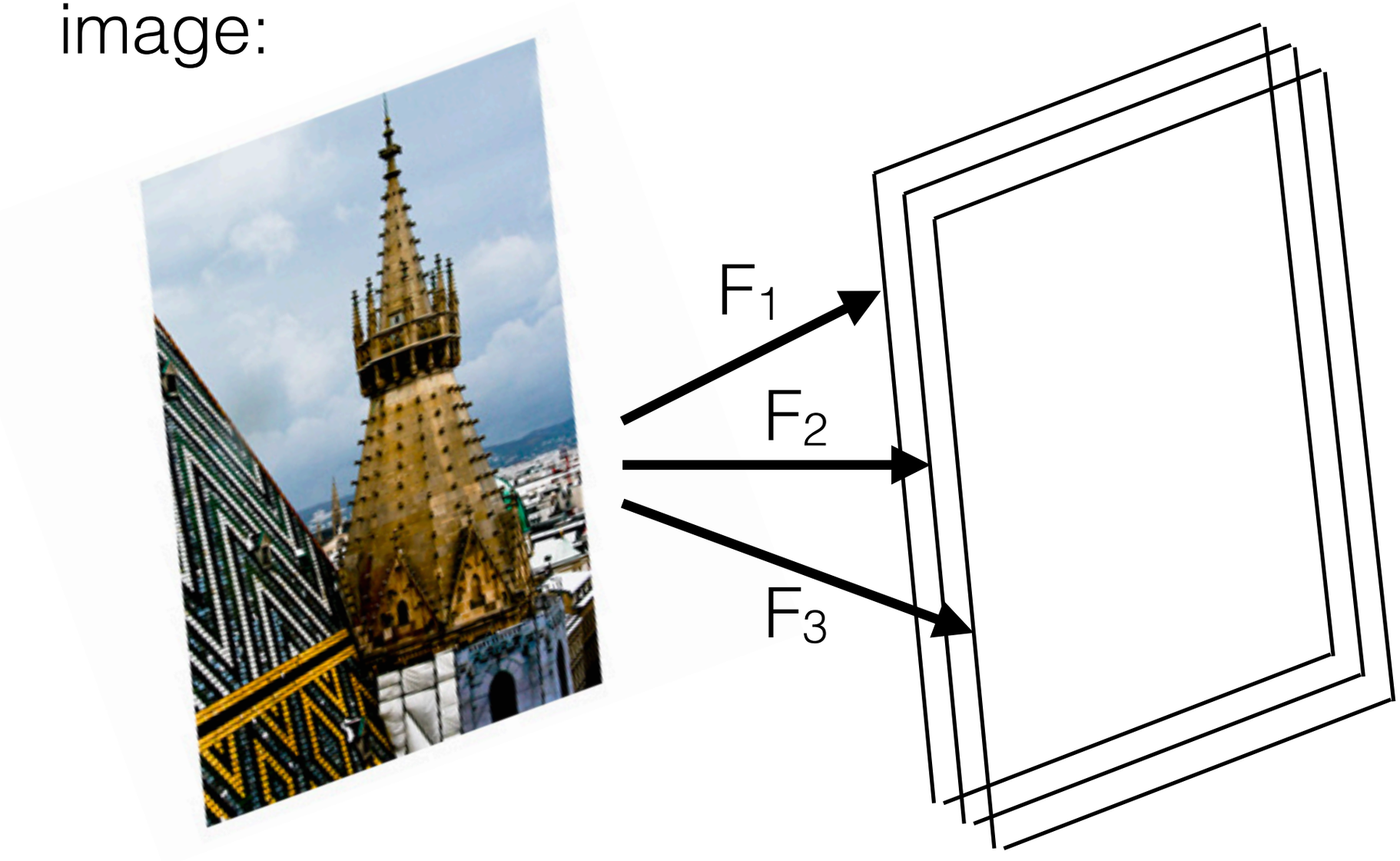
Convolutional Layer: multiple filters

An image:



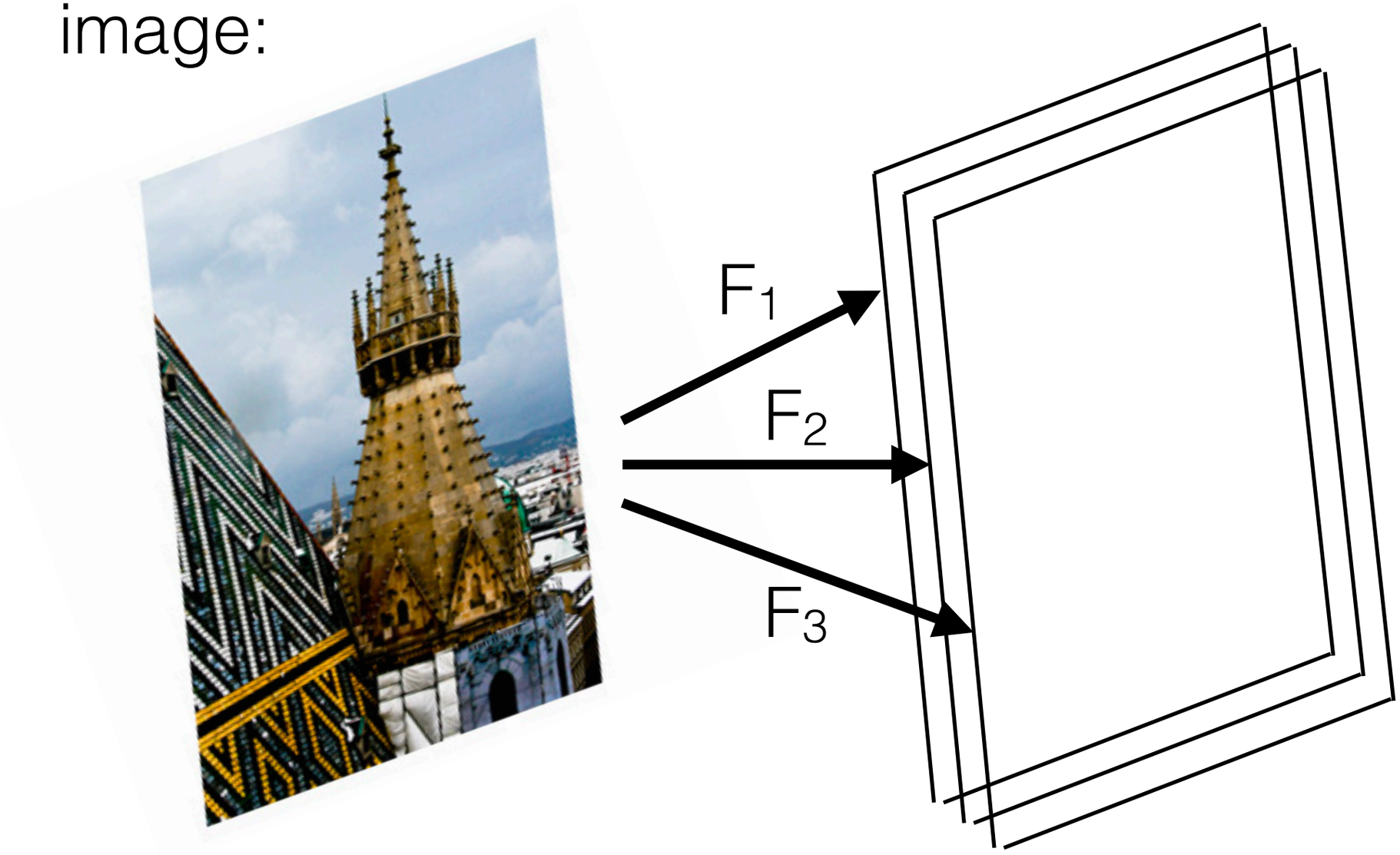
Convolutional Layer: multiple filters

An image:



Convolutional Layer: multiple filters

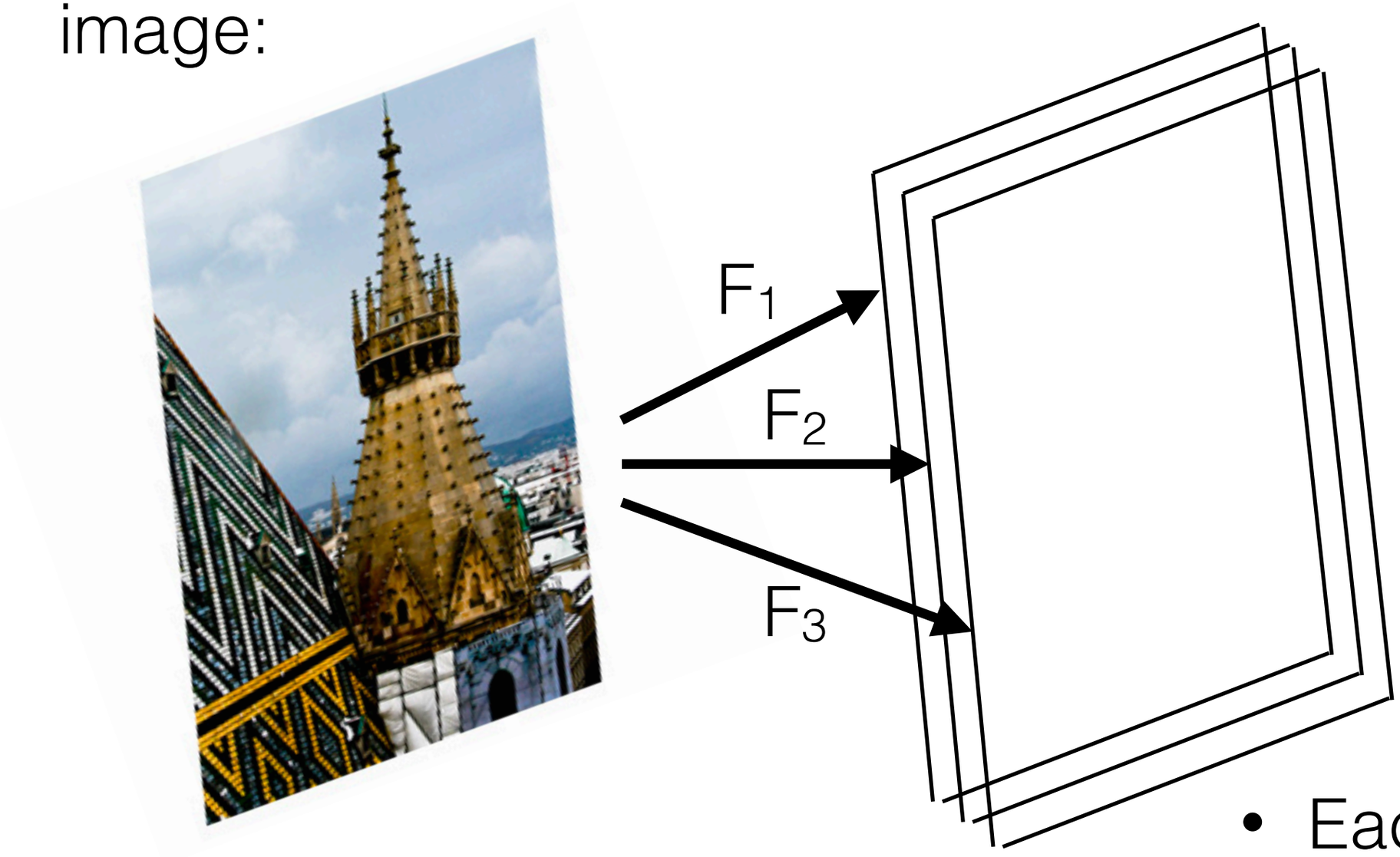
An image:



- 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:

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

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

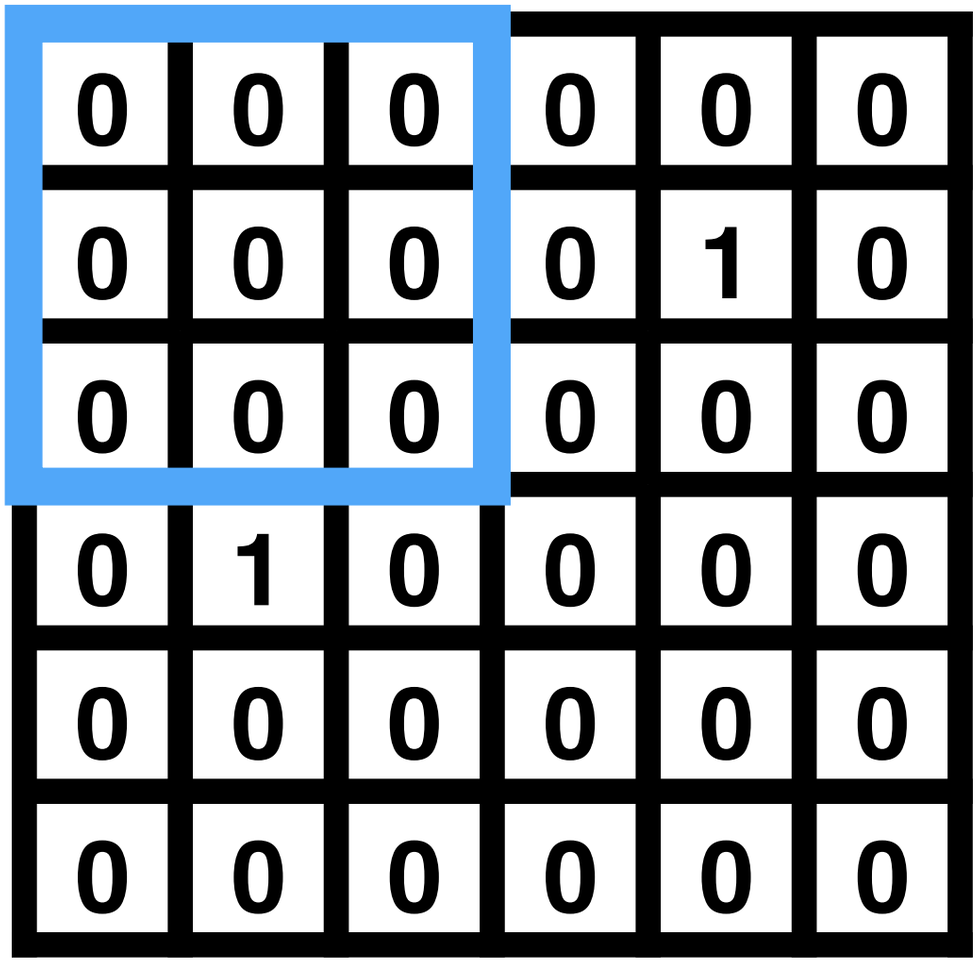
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Max pooling: returns max of its arguments

- size 3x3 (“size 3”)

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

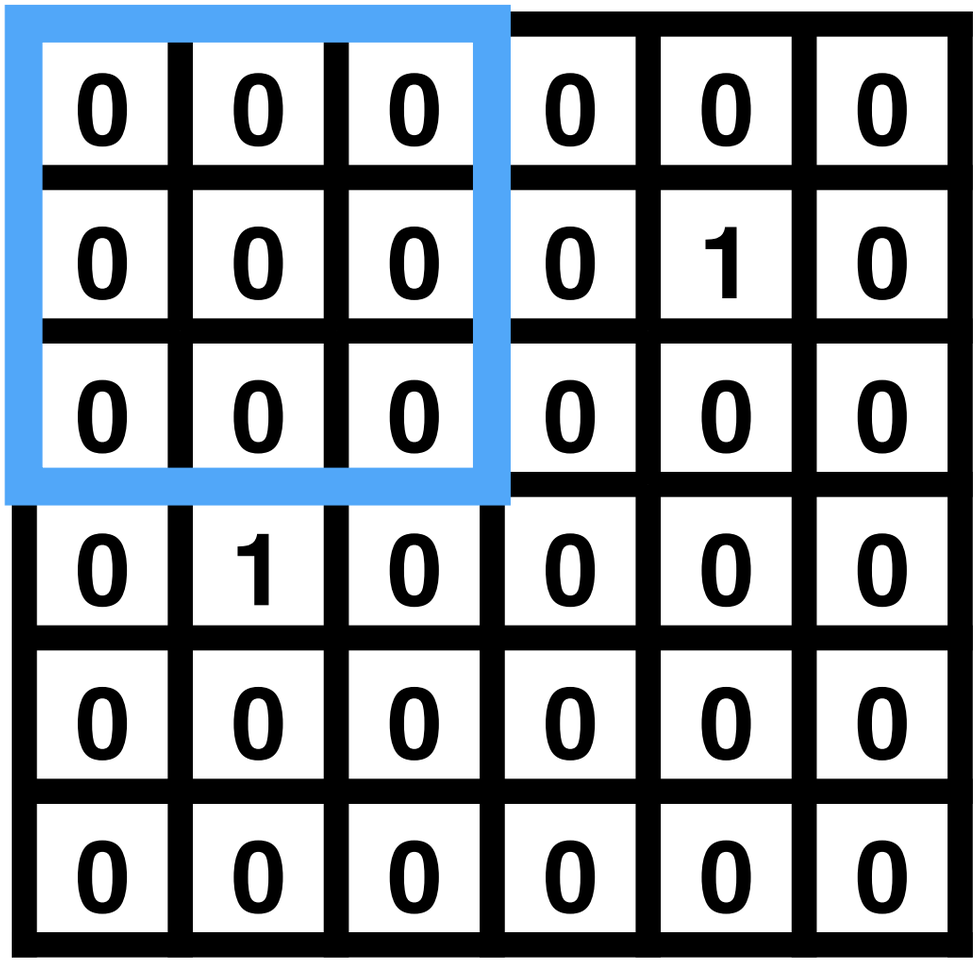


Max pooling: returns max of its arguments

- size 3x3 (“size 3”)

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



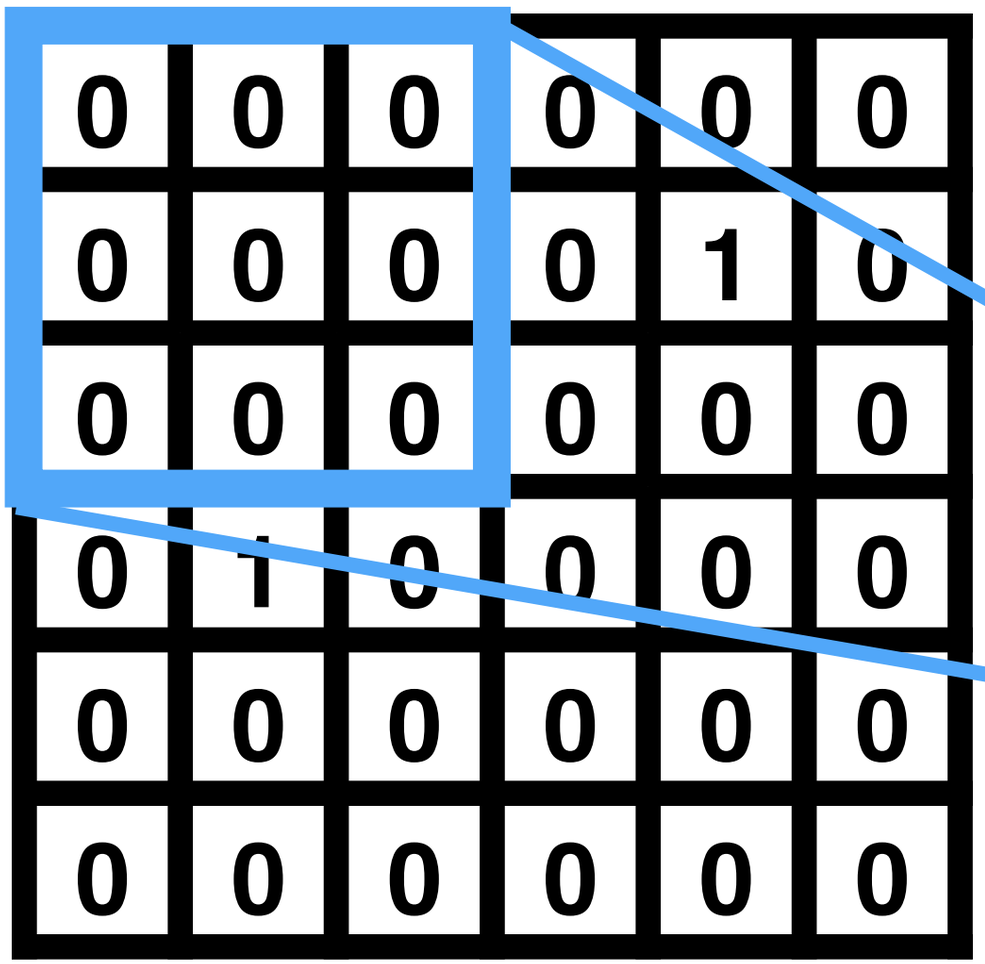
Max pooling: returns max of its arguments

- size 3x3 (“size 3”)

After max pooling:

Max pooling layer: 2D example

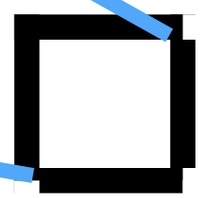
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

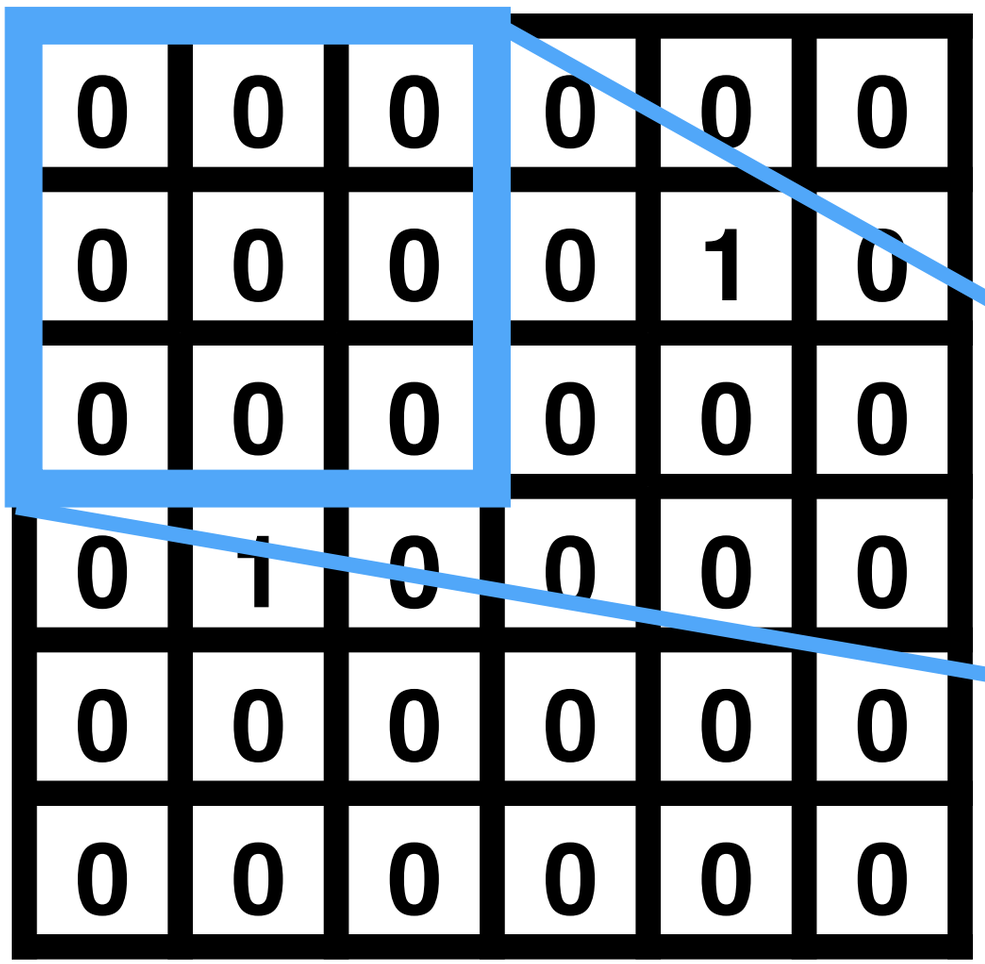
- size 3x3 (“size 3”)

After max pooling:



Max pooling layer: 2D example

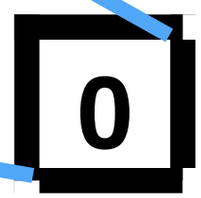
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- size 3x3 (“size 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

- size 3x3 (“size 3”)

After max pooling:

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

- size 3x3 (“size 3”)

After max pooling:

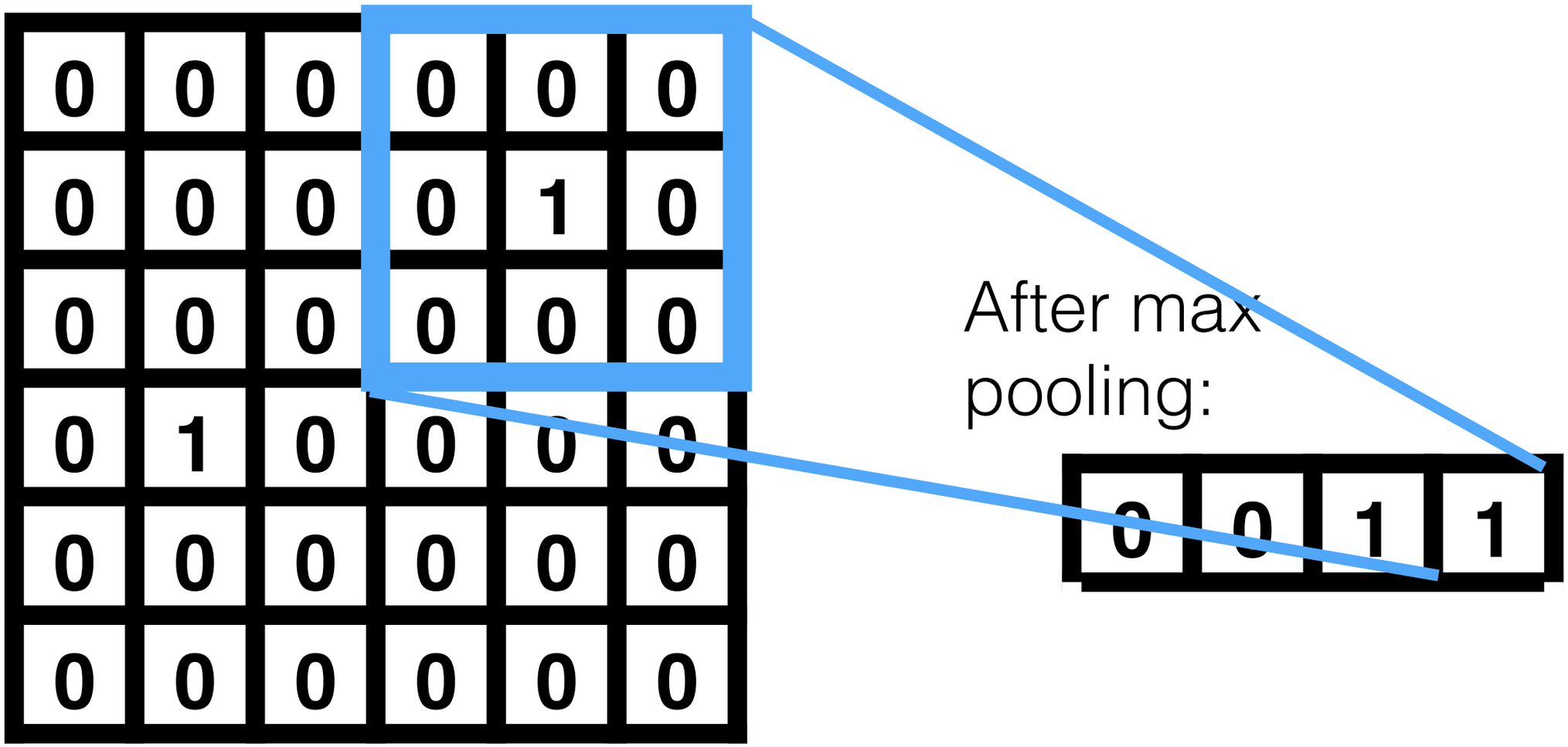
0	0	1

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:

Max pooling: returns max of its arguments

- 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

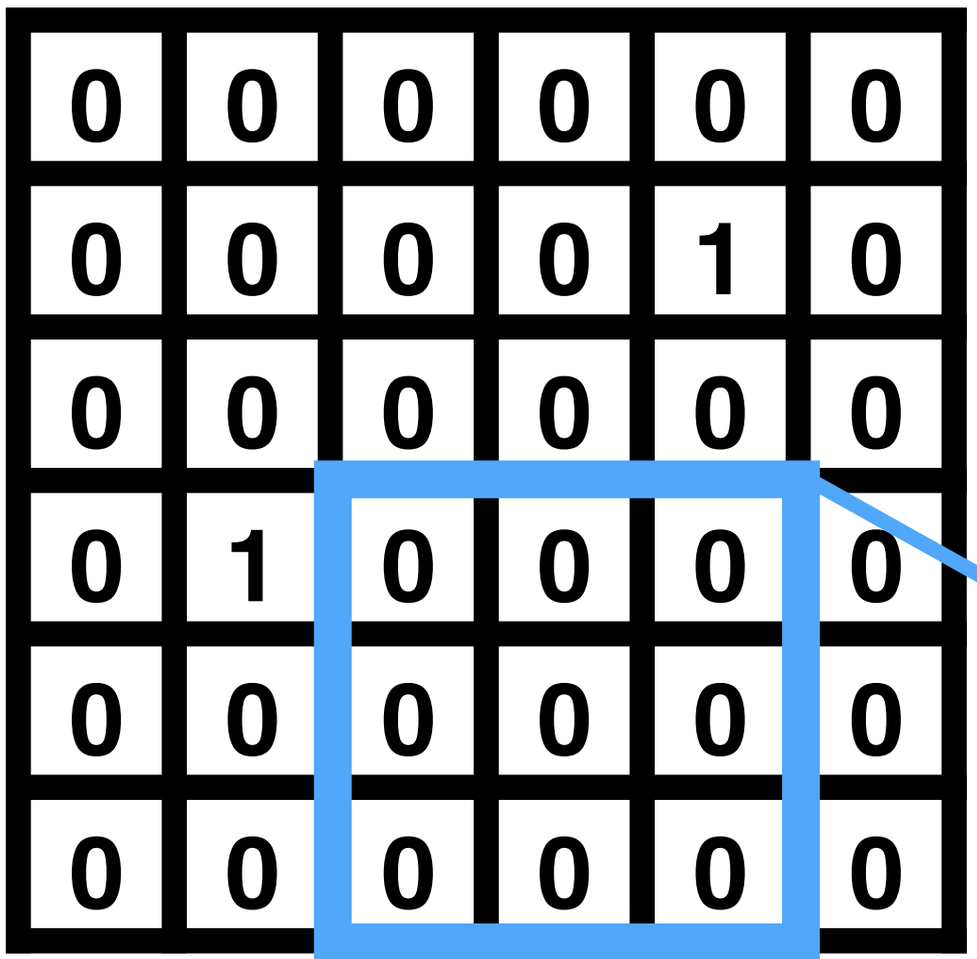
- size 3x3 (“size 3”)

After max pooling:

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

Max pooling layer: 2D example

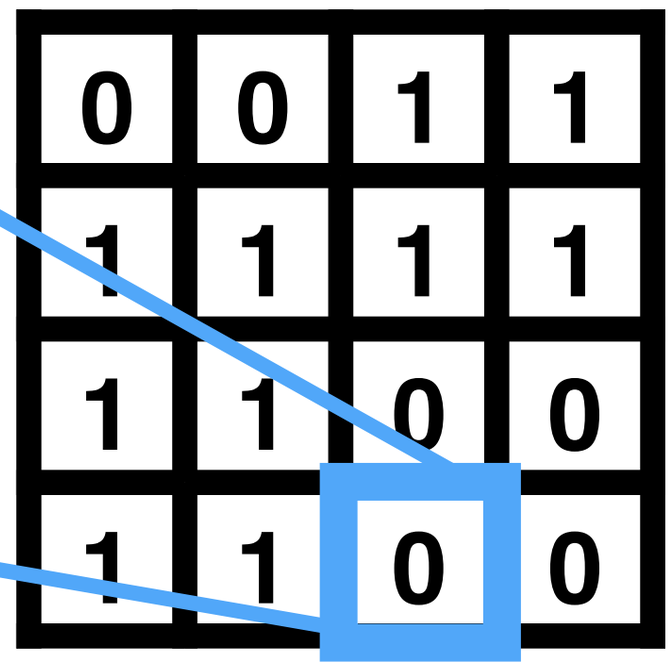
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- size 3x3 (“size 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

- 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

- 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

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

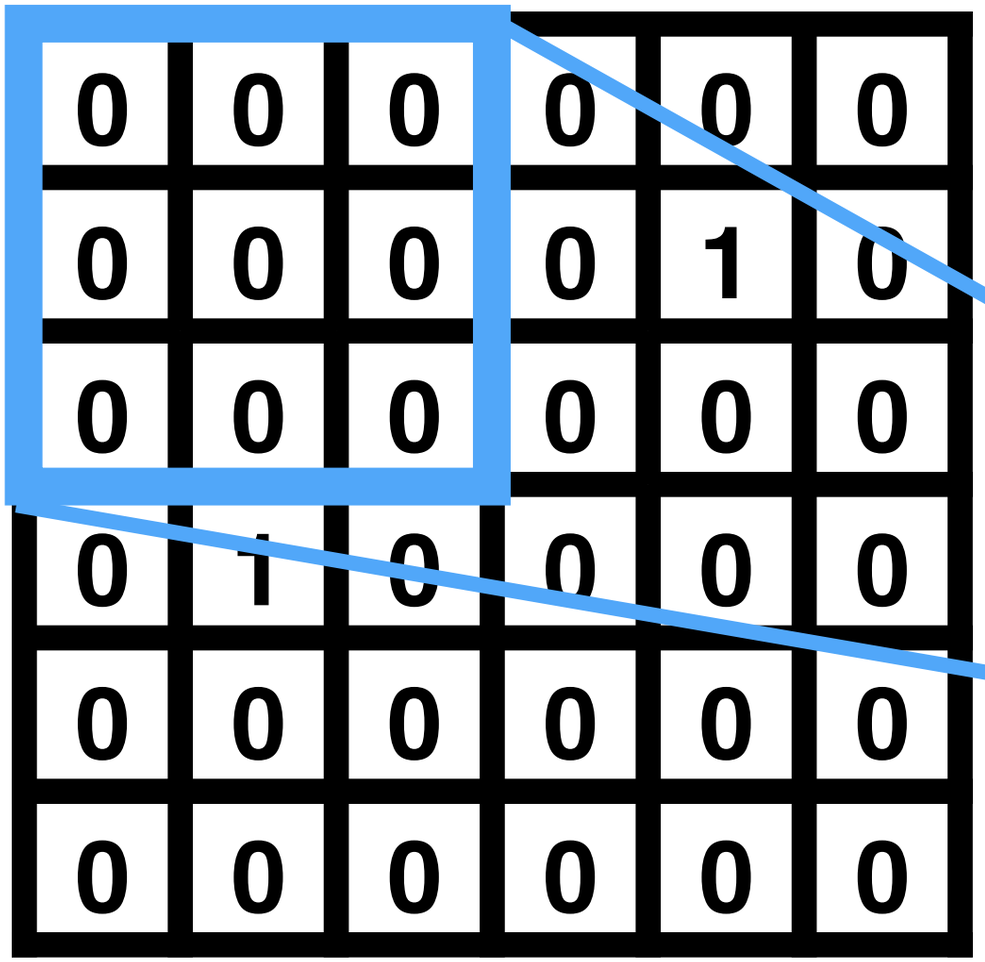
Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- stride 3

After max pooling:

Max pooling layer: 2D example

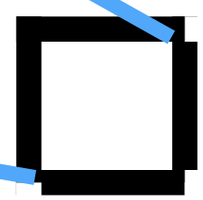
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

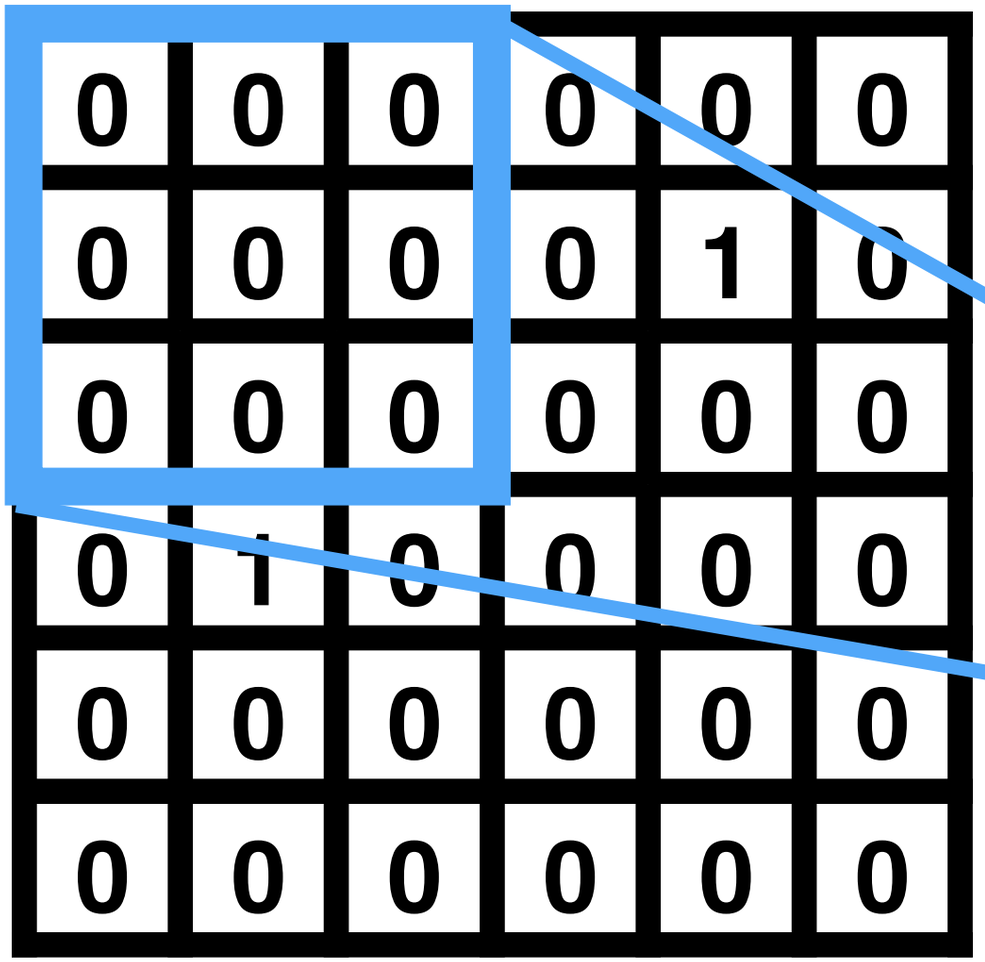
- size 3x3 (“size 3”)
- stride 3

After max pooling:



Max pooling layer: 2D example

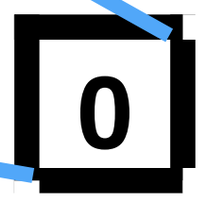
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

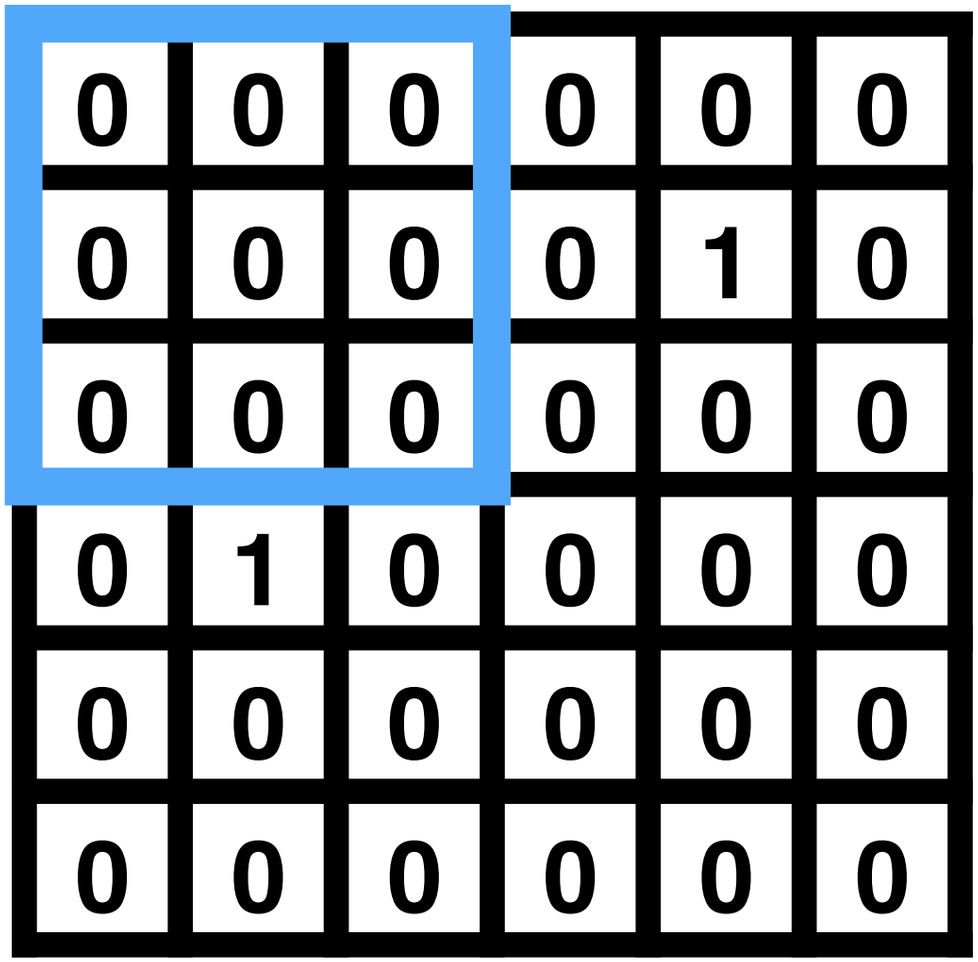
- size 3x3 (“size 3”)
- stride 3

After max pooling:



Max pooling layer: 2D example

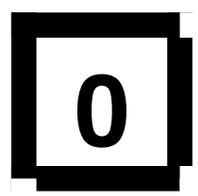
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

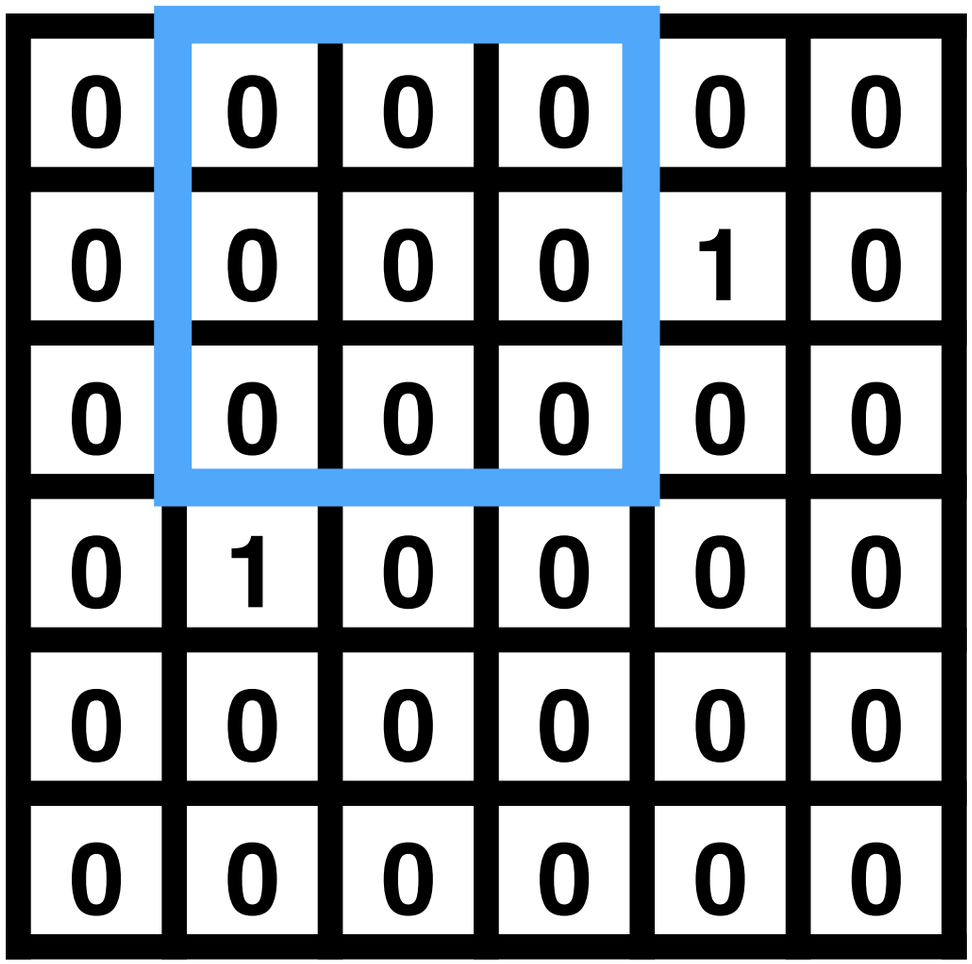
- size 3x3 (“size 3”)
- stride 3

After max pooling:



Max pooling layer: 2D example

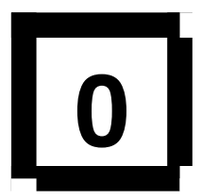
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

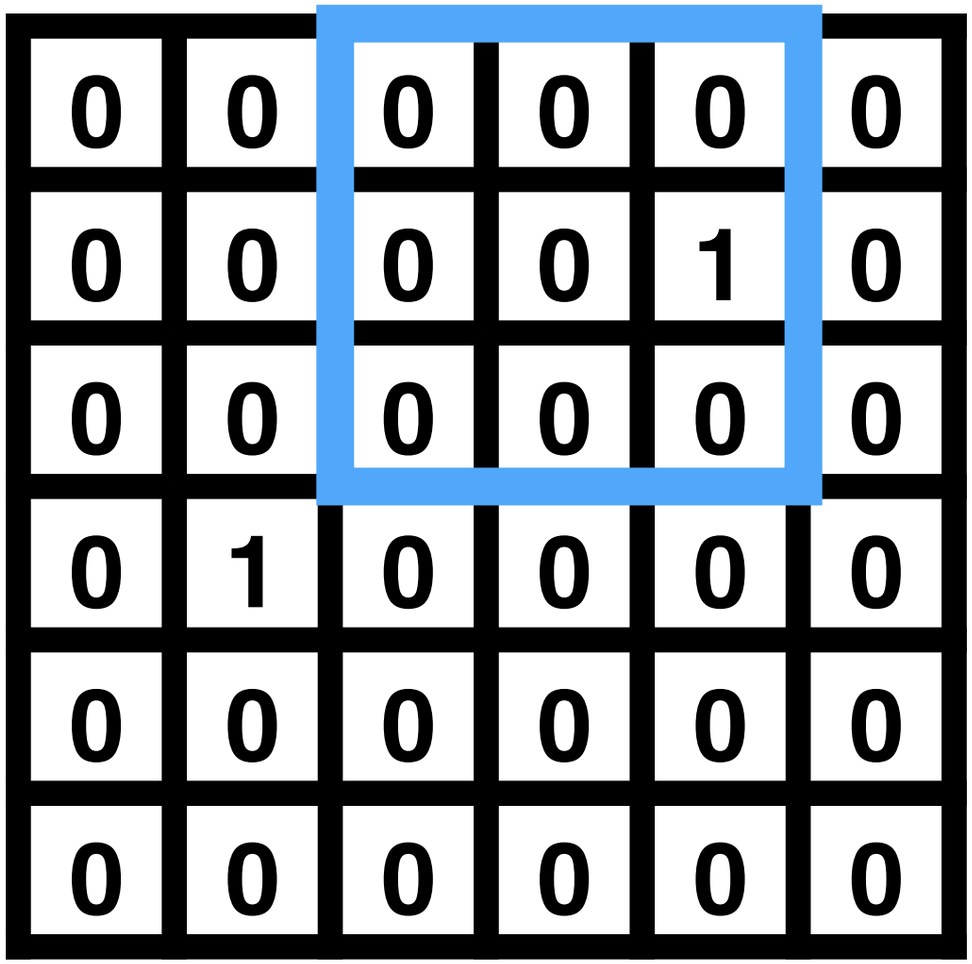
- size 3x3 (“size 3”)
- stride 3

After max pooling:



Max pooling layer: 2D example

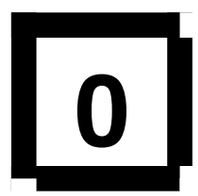
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- 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

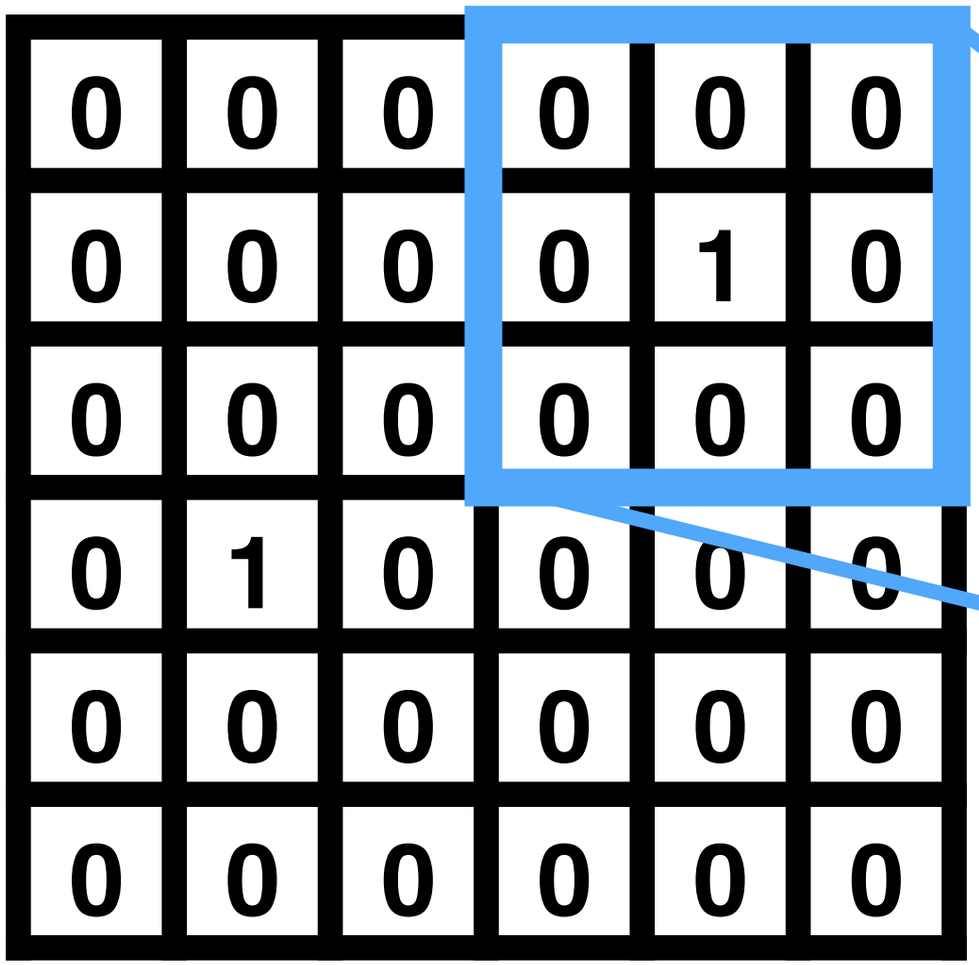
- size 3x3 (“size 3”)
- stride 3

After max pooling:

0

Max pooling layer: 2D example

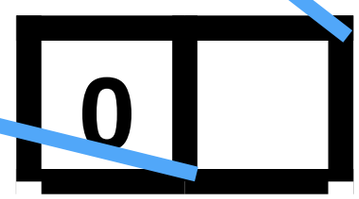
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

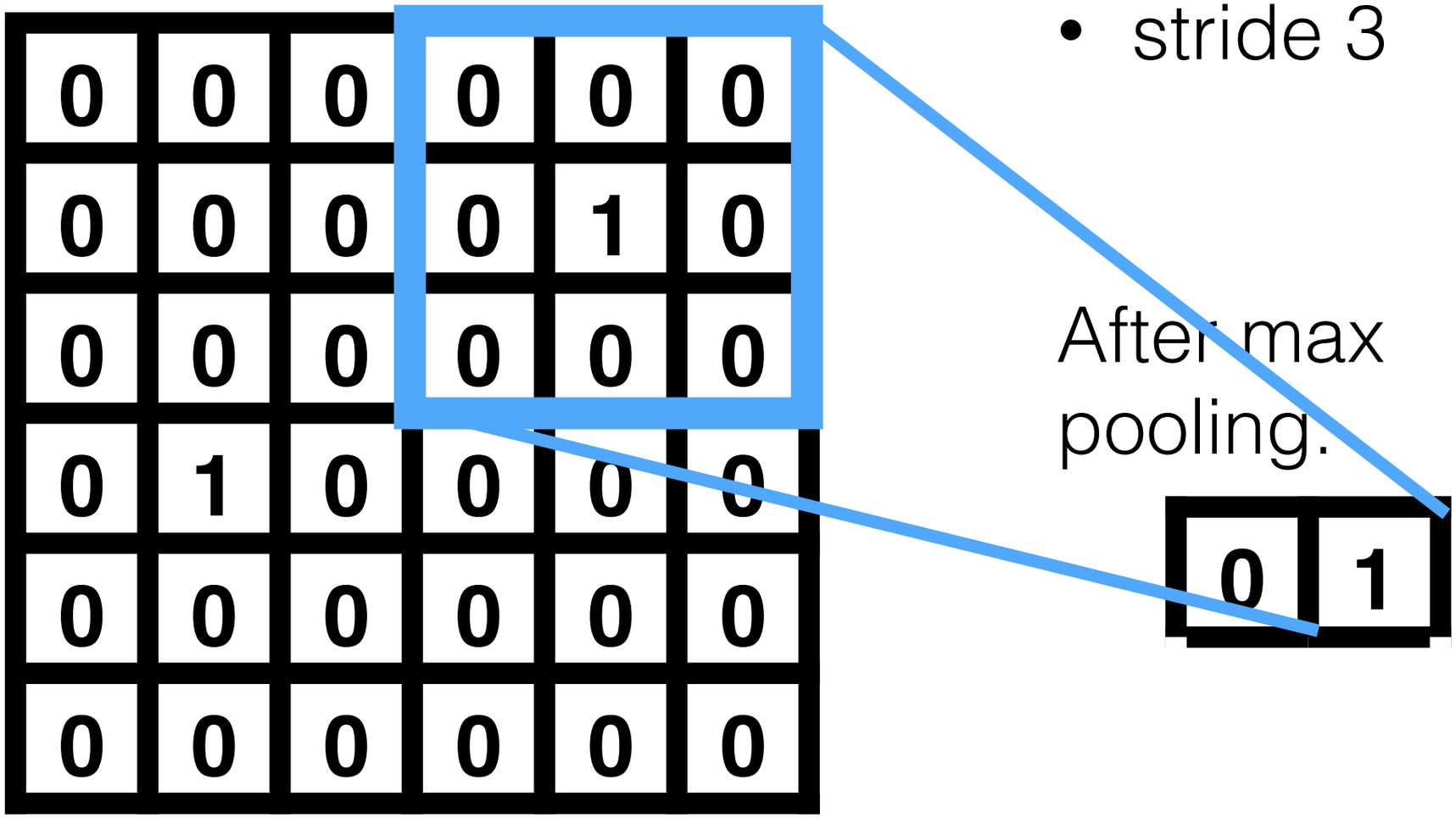
- size 3x3 (“size 3”)
- stride 3

After max pooling:



Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

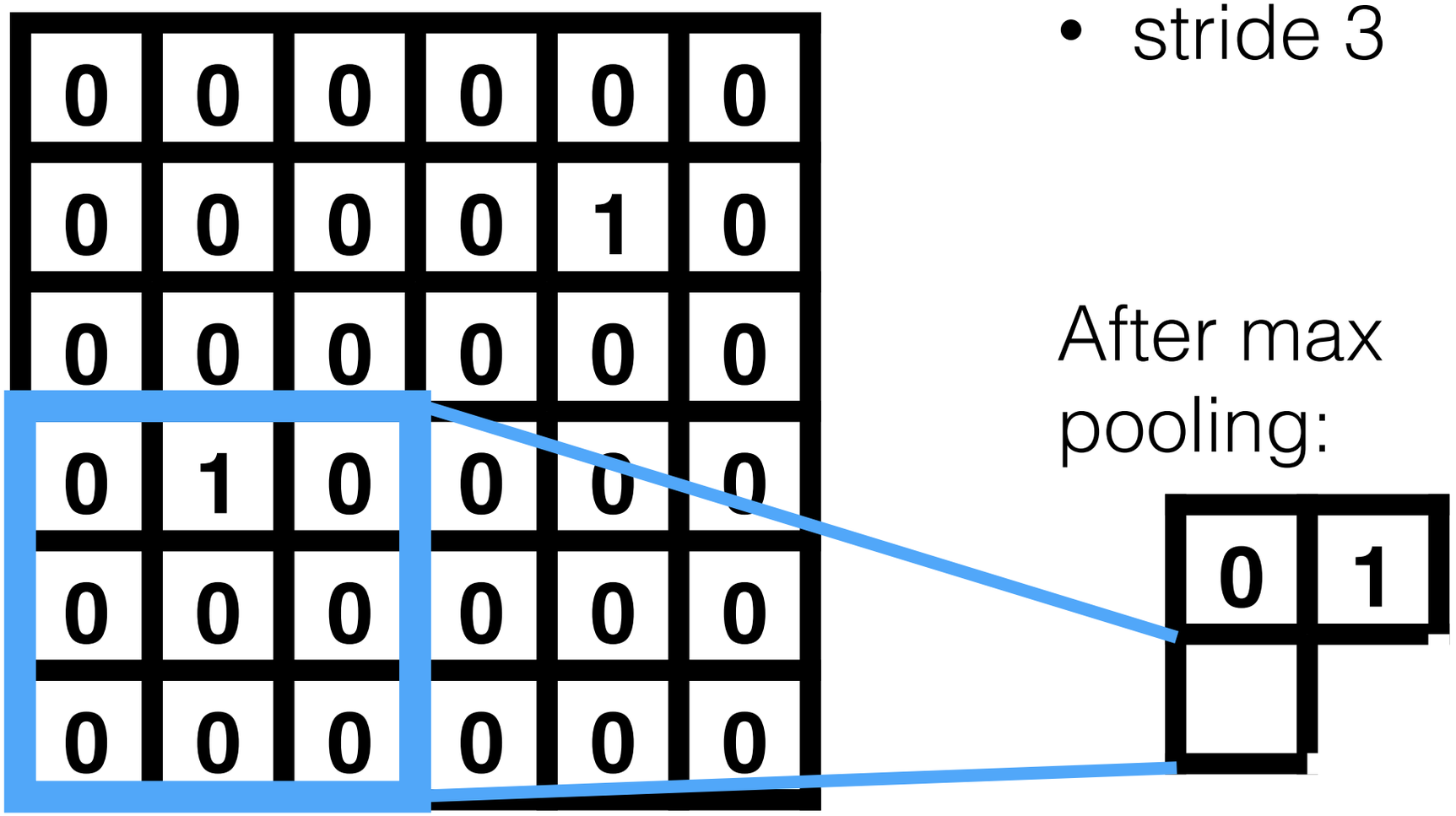
- size 3x3 (“size 3”)
- stride 3

After max pooling:

0	1

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



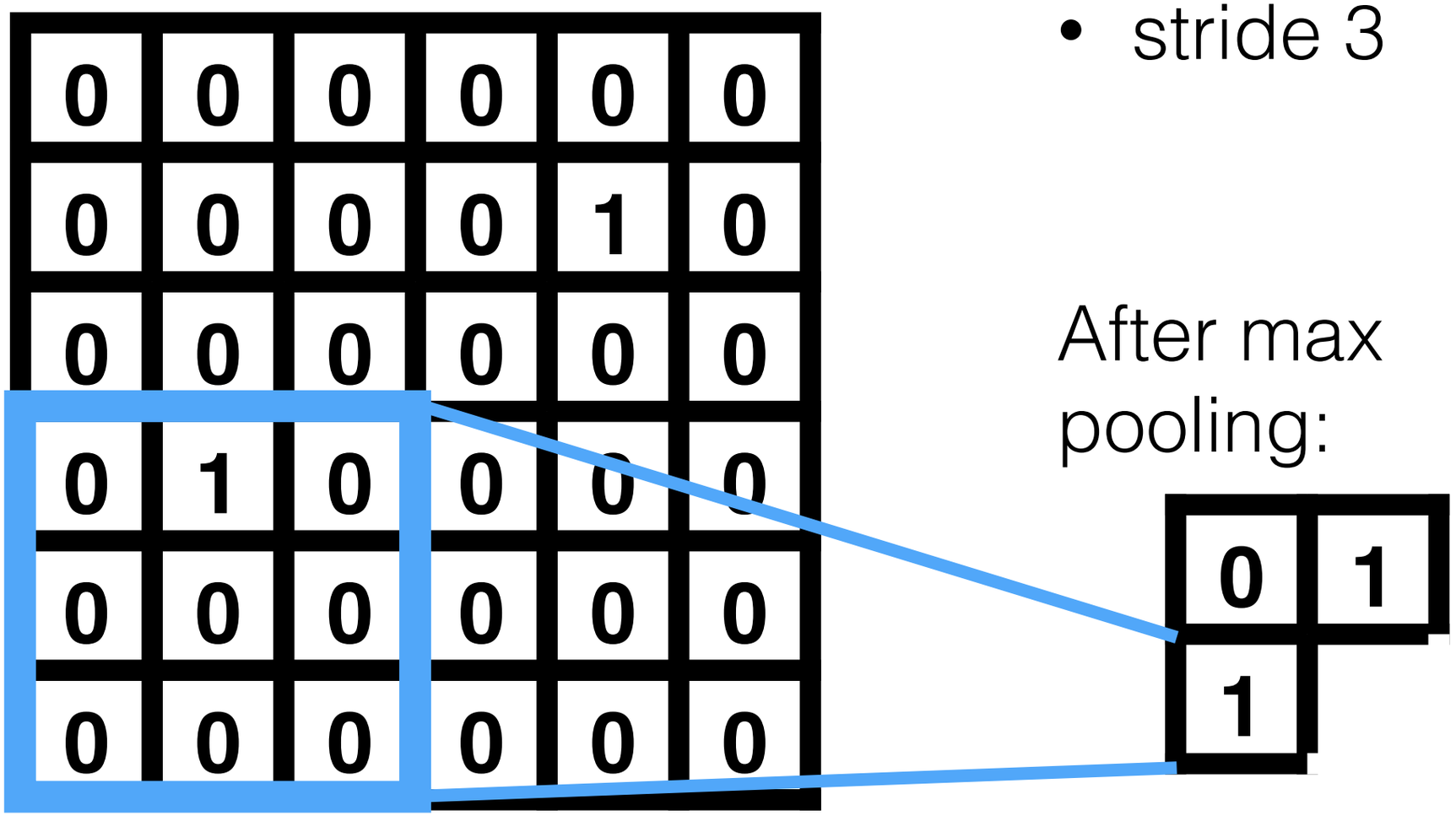
Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- stride 3

After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



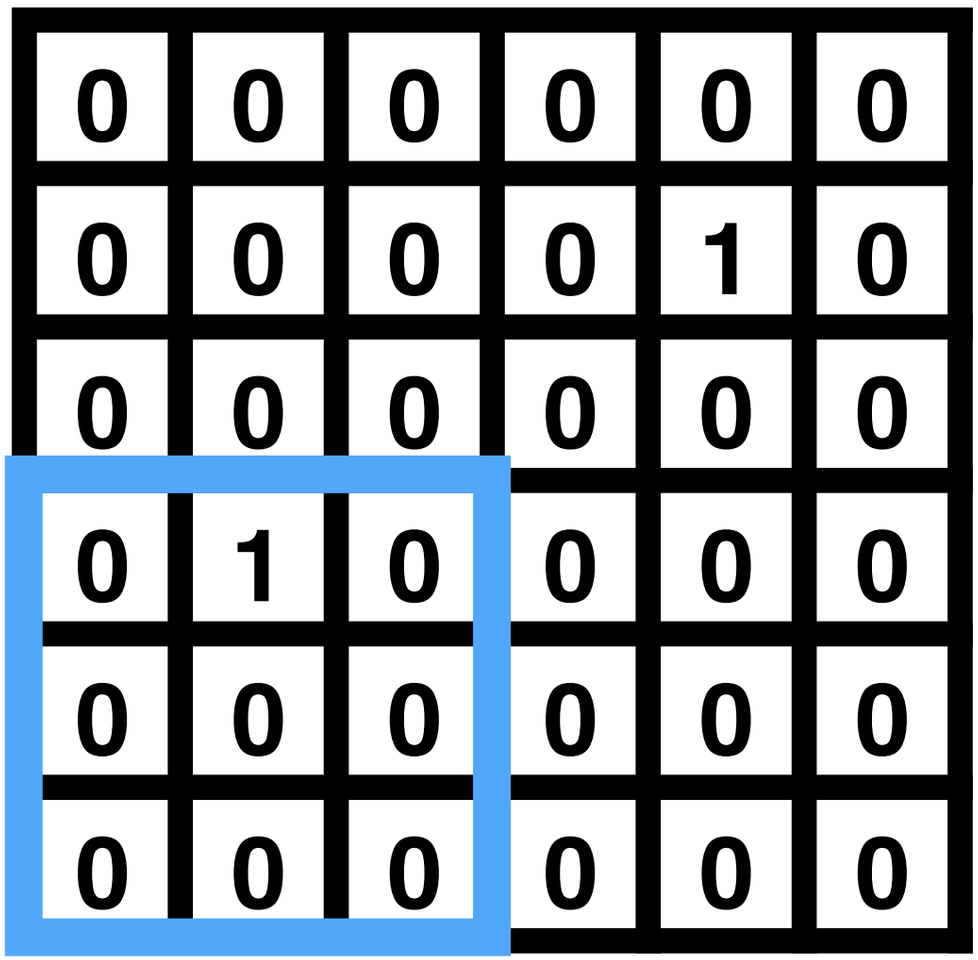
Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- stride 3

After max pooling:

Max pooling layer: 2D example

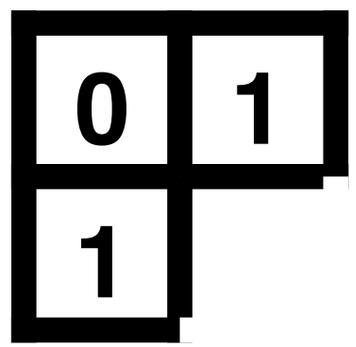
Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

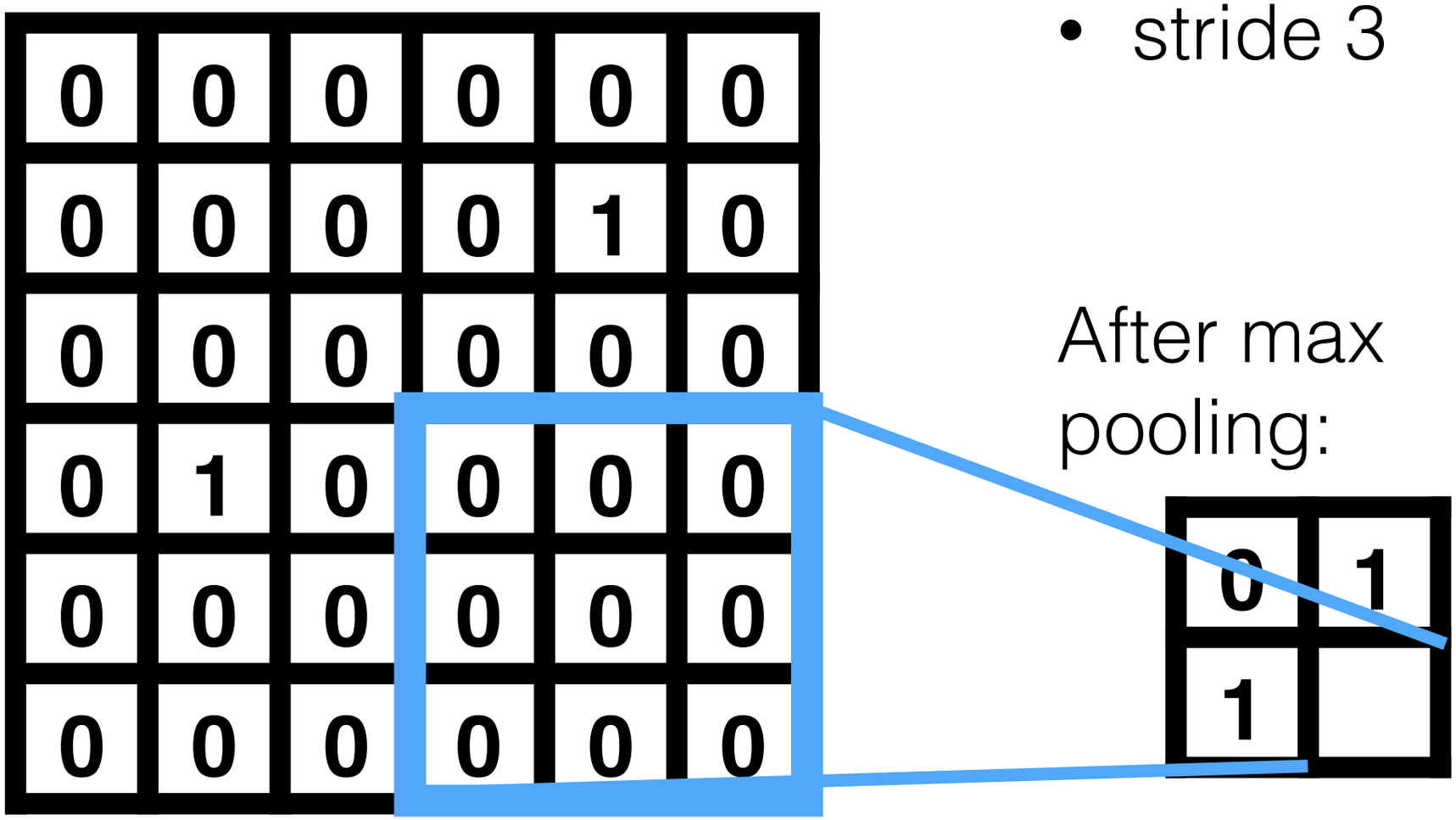
- size 3x3 (“size 3”)
- stride 3

After max pooling:

0	1
1	

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



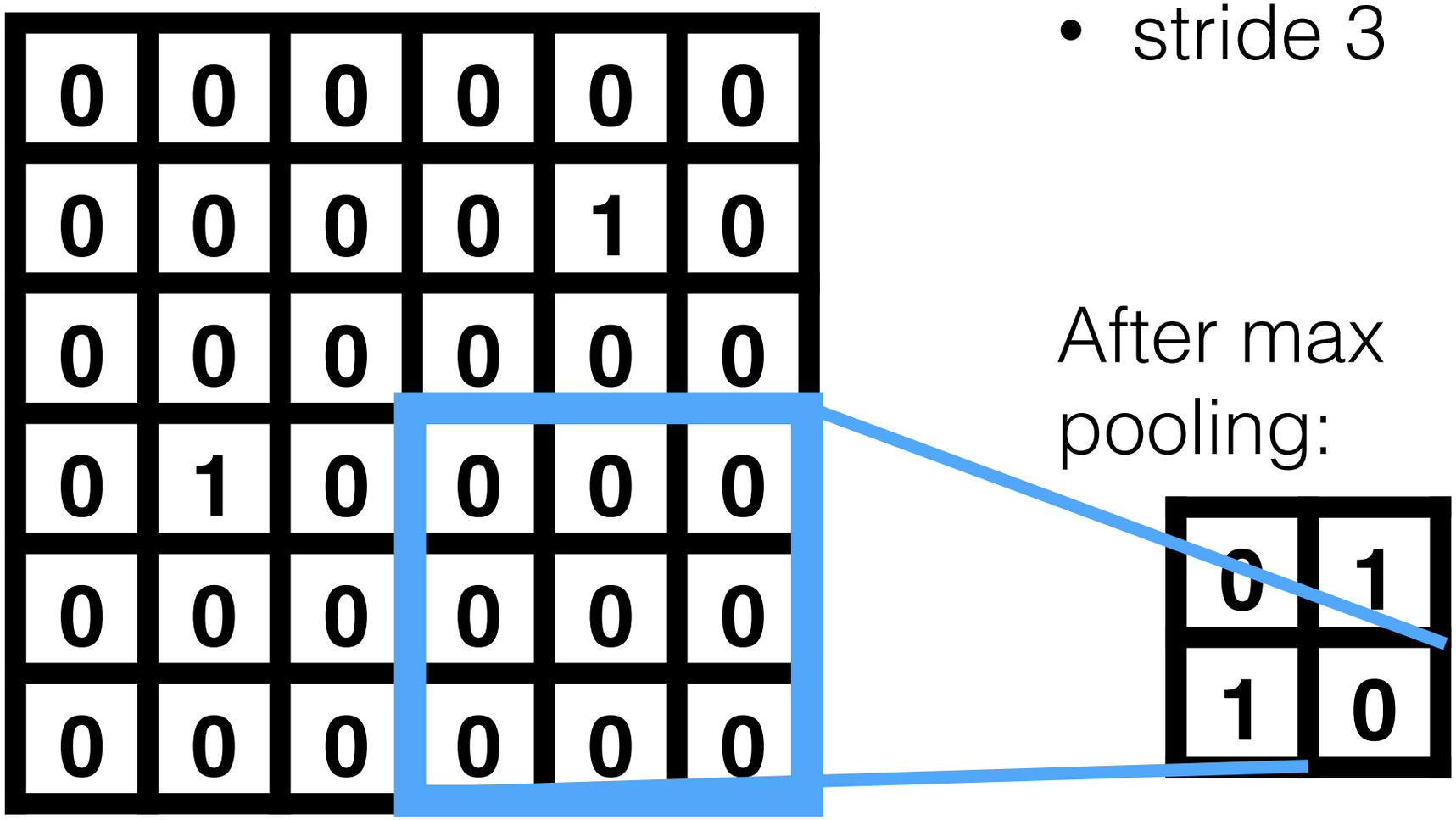
Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- stride 3

After max pooling:

Max pooling layer: 2D example

Output from the convolutional layer & ReLU:



Max pooling: returns max of its arguments

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

- size 3x3 (“size 3”)
- 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

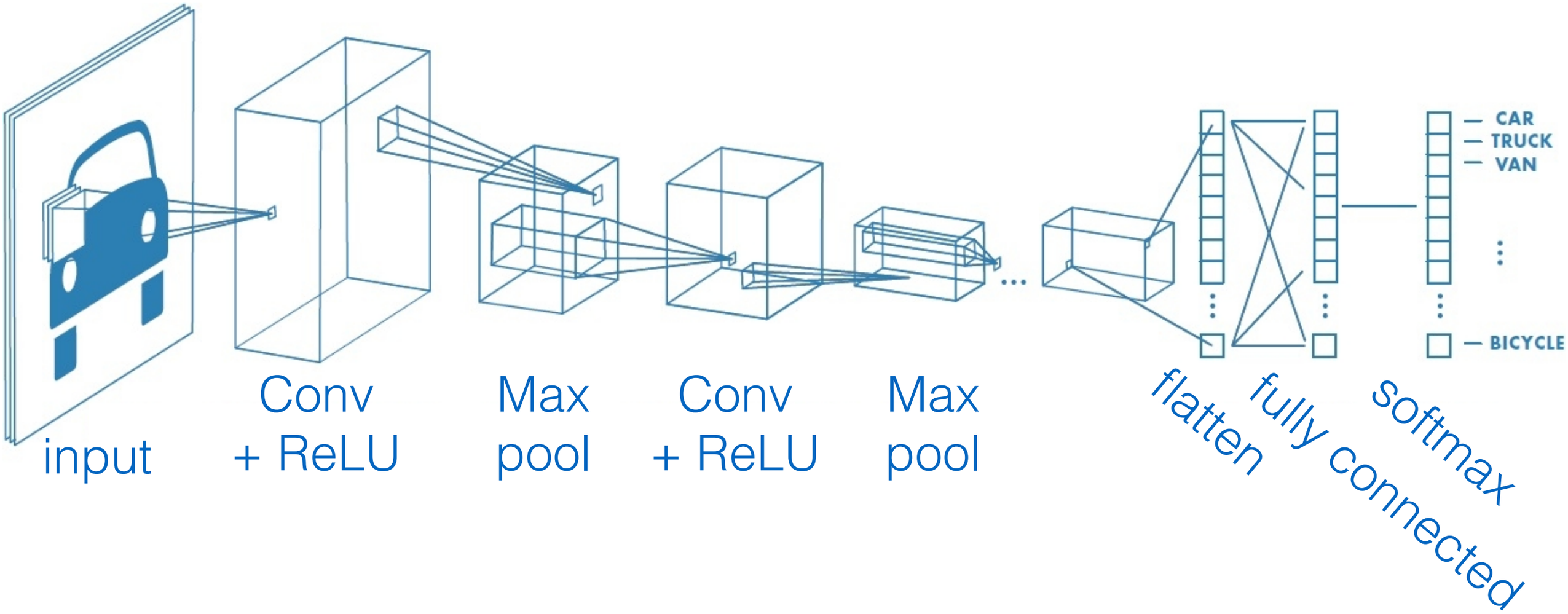
- size 3x3 (“size 3”)
- stride 3

After max pooling:

0	1
1	0

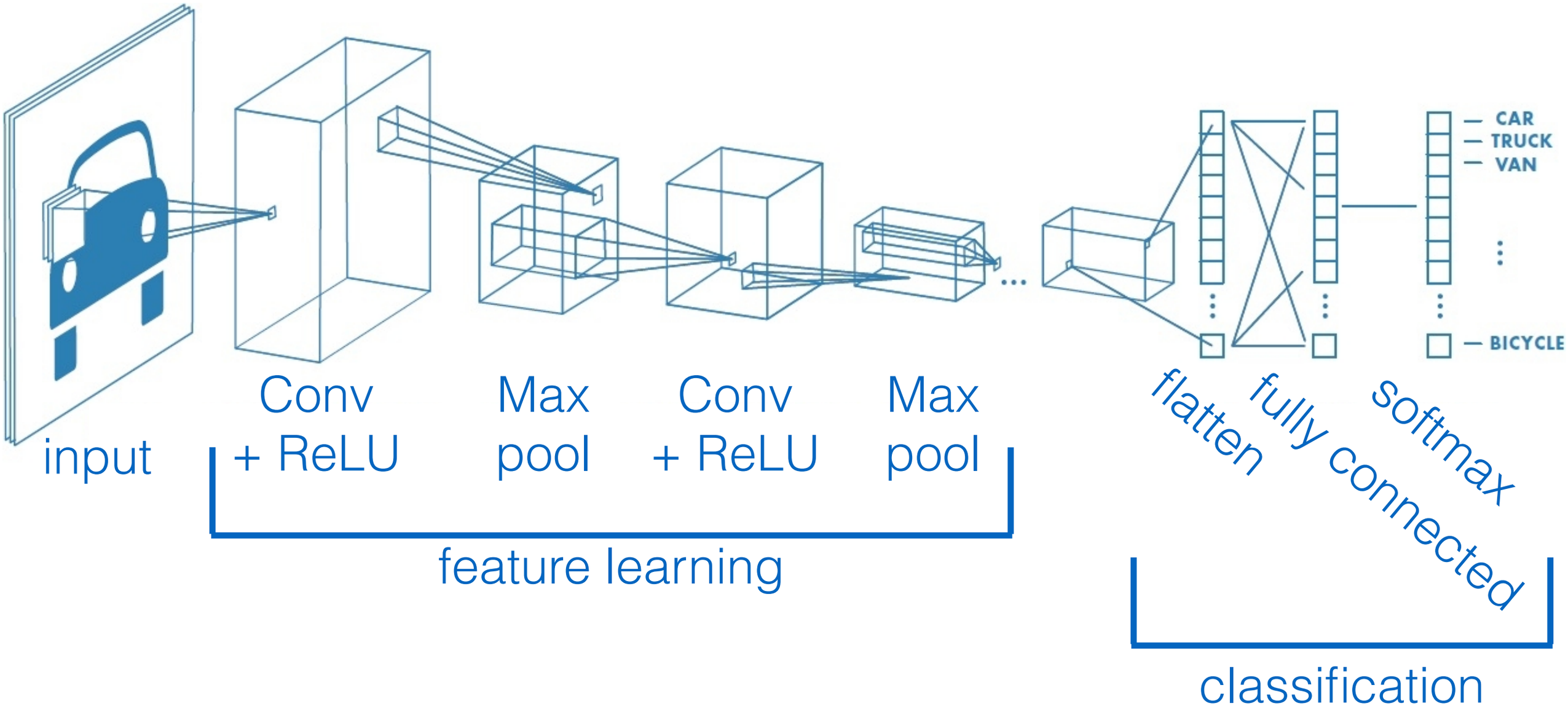
- Can use stride with filters too
- No weights in max pooling

CNNs: typical architecture



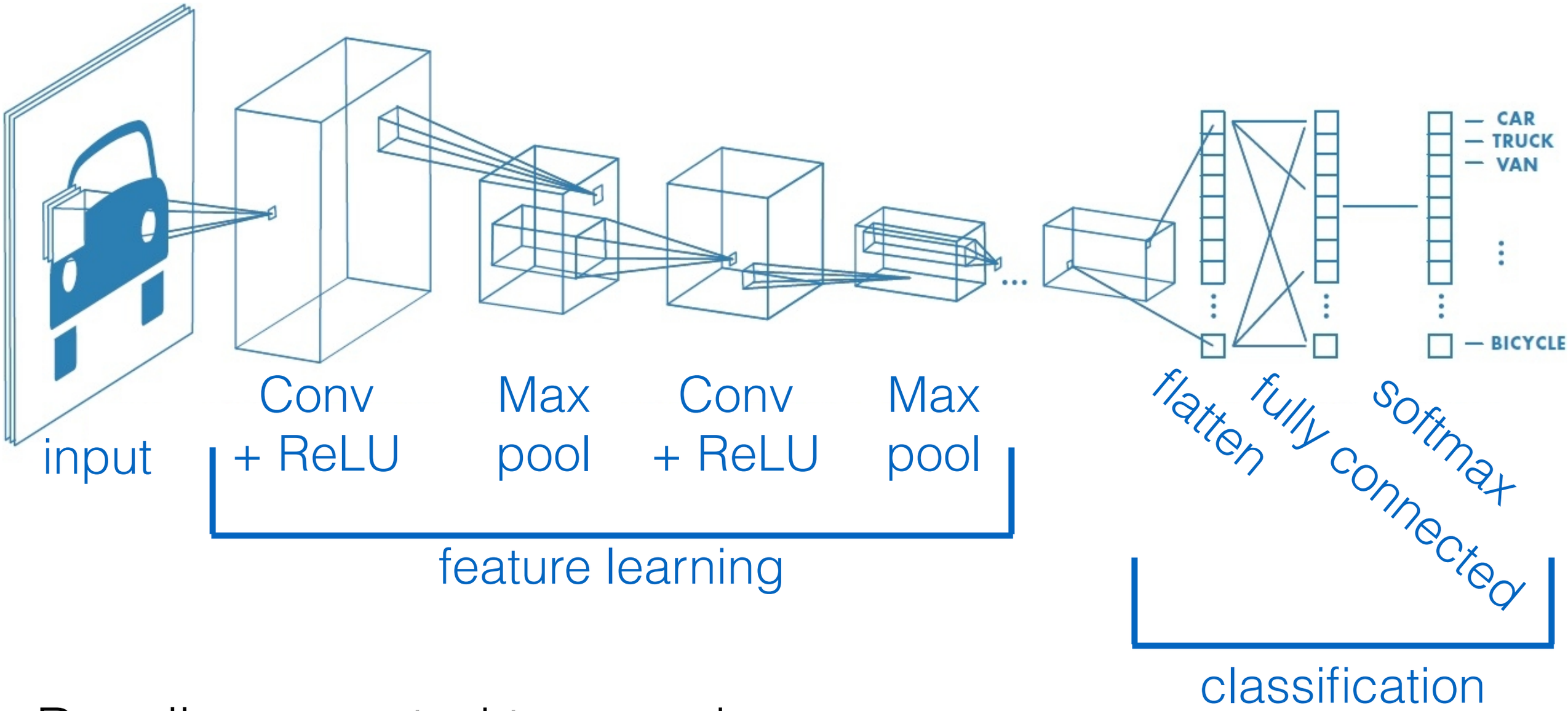
[https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html]

CNNs: typical architecture



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

CNNs: typical architecture

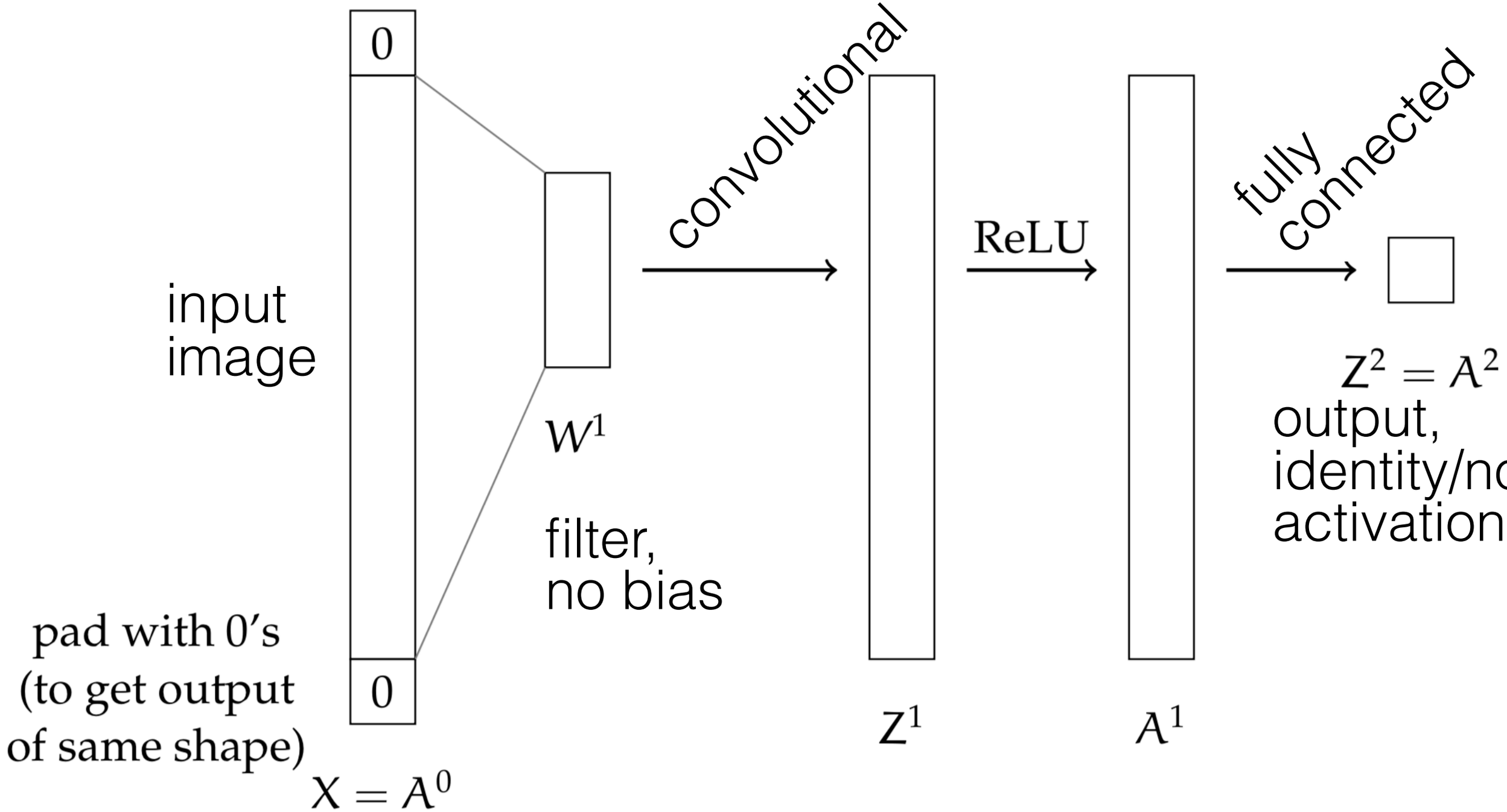


Recall: we wanted to encode

- Spatial locality
- Translation invariance

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

CNNs: a taste of backpropagation



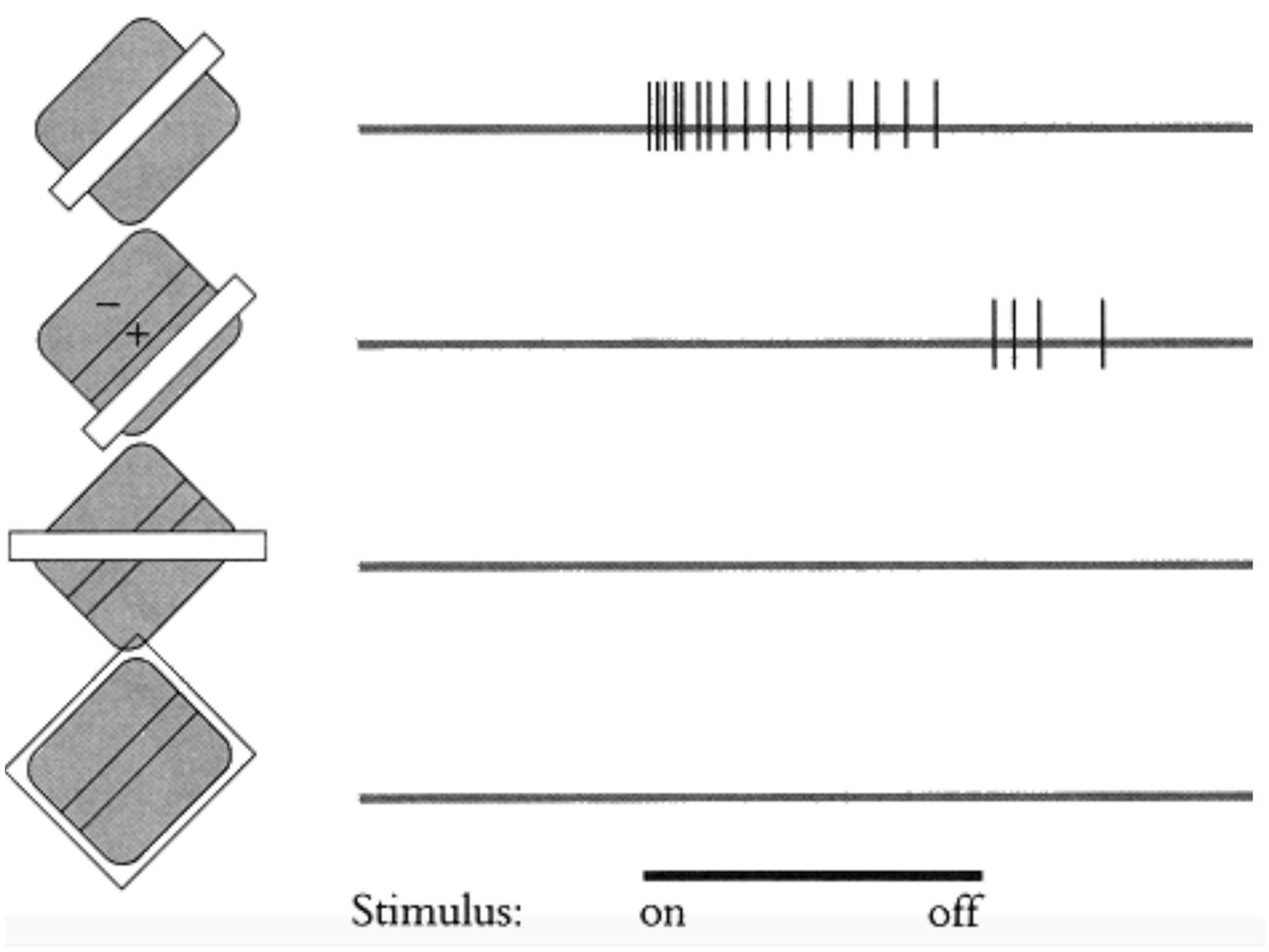
Cat neurons [Hubel, Weisel 1959, 1962]

(Be careful with biology analogies)

Cat neurons [Hubel, Weisel 1959, 1962]

(Be careful with biology analogies)

receptive field



- *simple cells*
- *complex cells*

[http://fourier.eng.hmc.edu/e180/lectures/v1/node7.html]