How does a convolutional neural network work?

Valerio Velardo

CNNs

- Mainly used for processing images
- Perform better than multilayer perceptron
- Less parameters than dense layers

Intuition

- Image data is structured
  - Edges, shapes
  - Translation invariance
  - Scale invariance
- CNN emulates human vision system
- Components of a CNN learn to extract different features

CNN components

- Convolution
- Pooling

Convolution

- *Kernel* = grid of weights
- Kernel is “applied” to the image
- Traditionally used in image processing

1	2	-1
0	1	2
-2	1	0

Convolution

Convolution

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Convolution

Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Convolution

Image

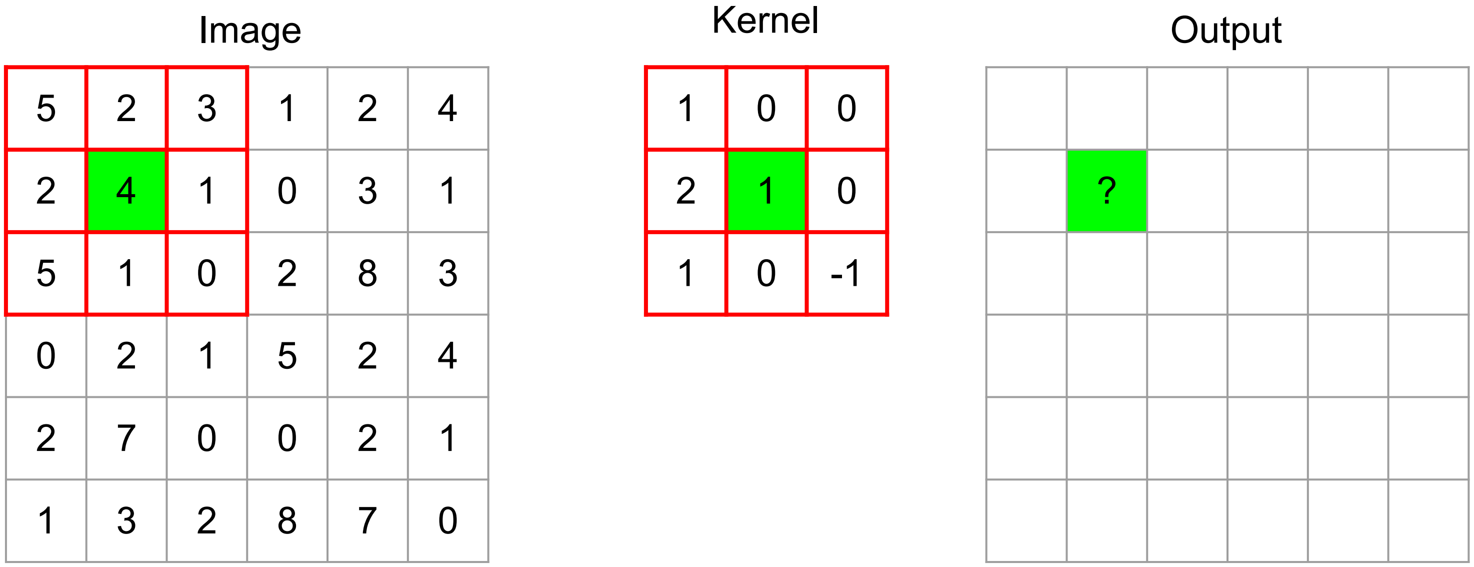
5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

Convolution



Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

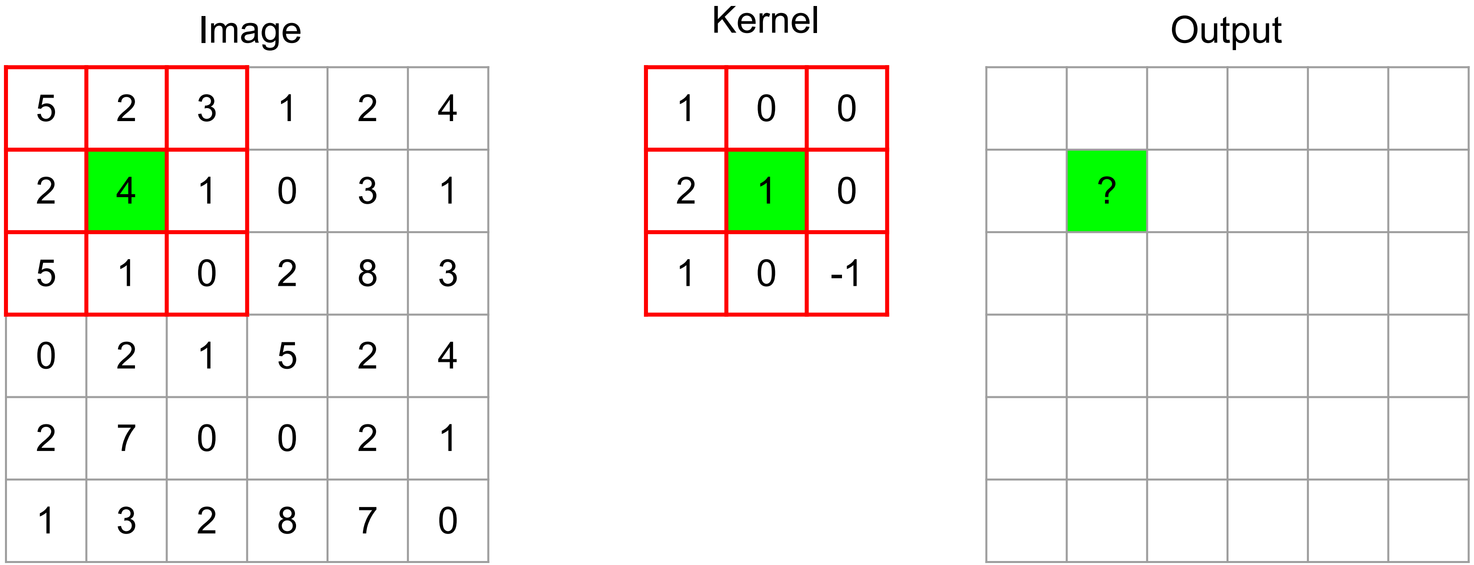
Kernel

1	0	0
2	1	0
1	0	-1

Output

	?				

Convolution



Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

	?				

$$\sum_{i=1}^{P} image_i \cdot K_i$$

Convolution

Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

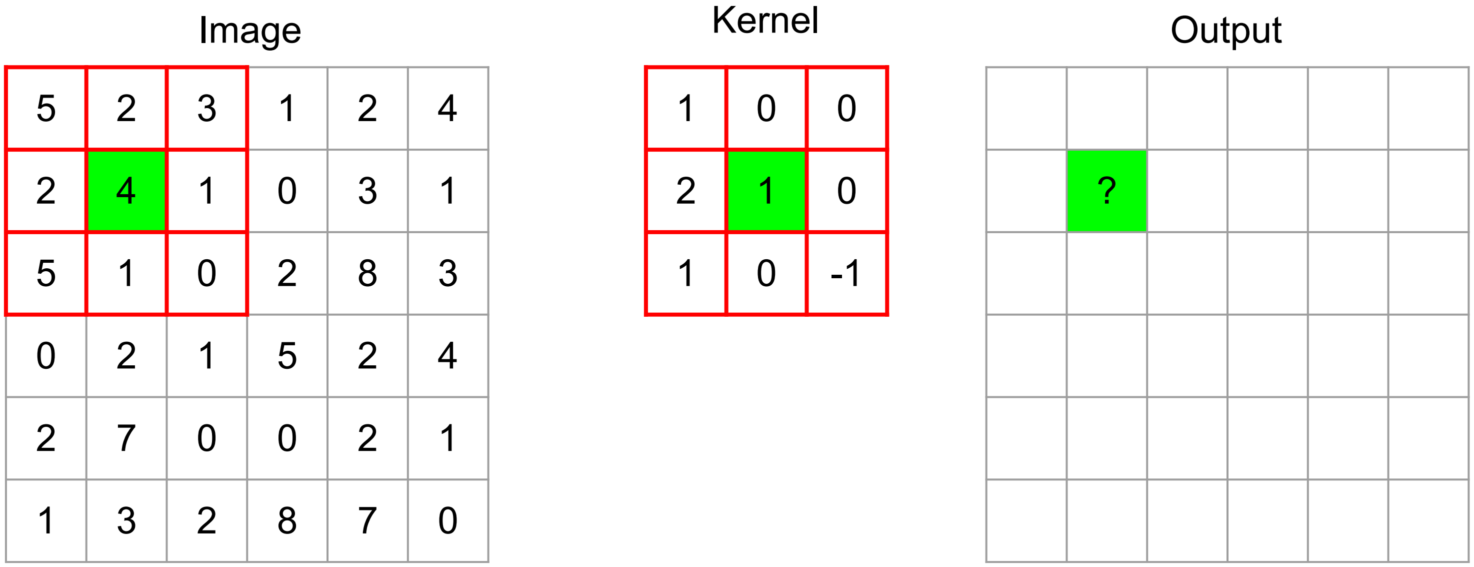
1	0	0
2	1	0
1	0	-1

Output

	?				

$$\sum_{i=1}^{P} image_i \cdot K_i = \boxed{5\cdot 1+2\cdot 0+\ldots+0\cdot -1}$$

Convolution



Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

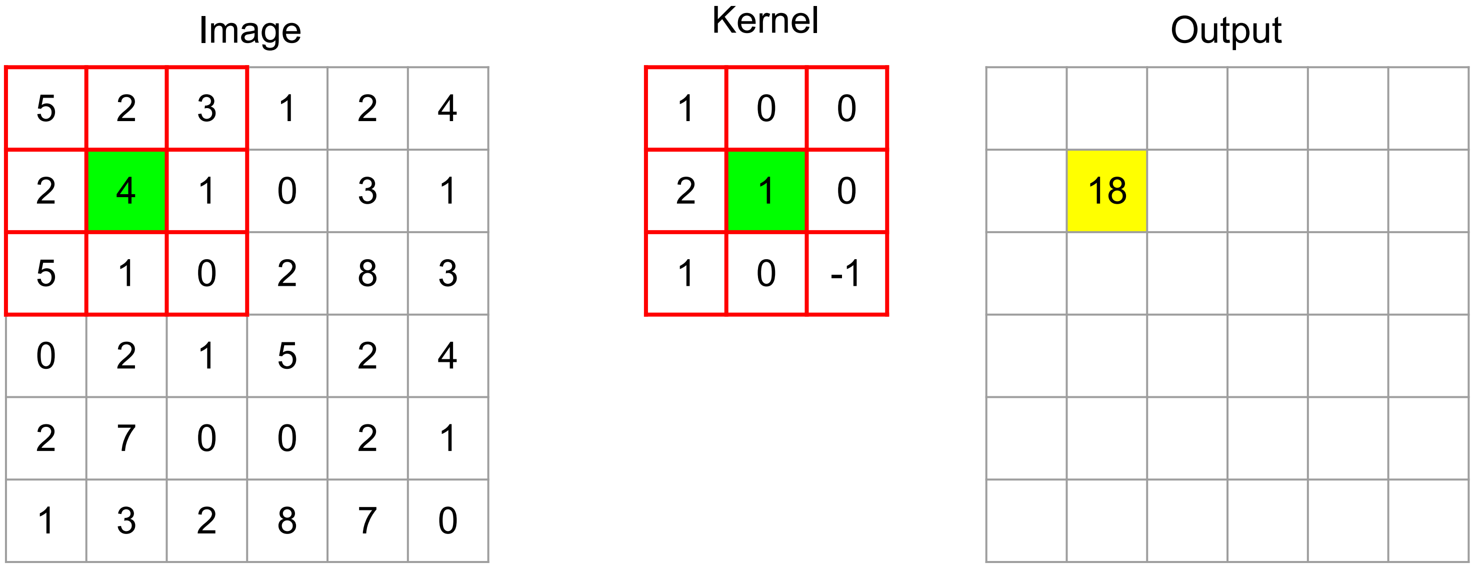
1	0	0
2	1	0
1	0	-1

Output

	?				

$$\sum_{i=1}^{P} image_i \cdot K_i = 5 \cdot 1 + 2 \cdot 0 + \ldots + 0 \cdot -1 = \boxed{18}$$

Convolution



Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

	18				

$$\sum_{i=1}^{P} image_i \cdot K_i = 5\cdot 1+2\cdot 0+\ldots+0\cdot -1 = 18$$

Convolution

Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

	18	10			

Convolution

Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

	18	10	-3		

Convolution

Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

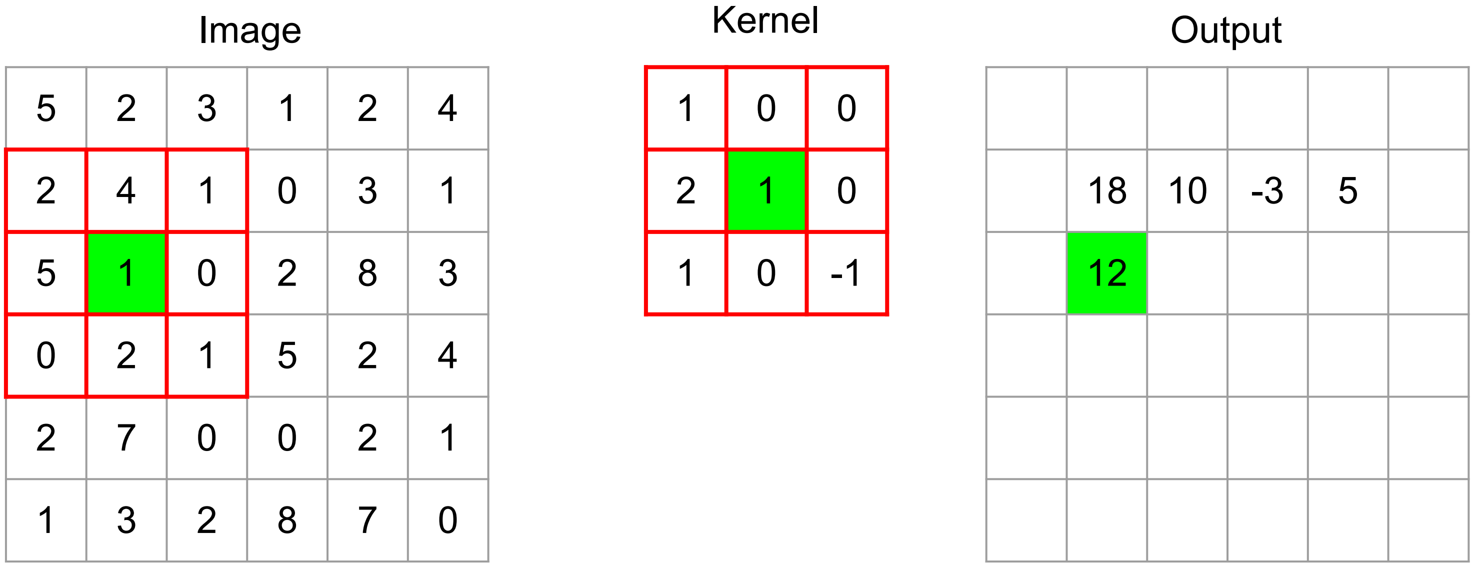
Kernel

1	0	0
2	1	0
1	0	-1

Output

	18	10	-3	5	

Convolution



Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

	18	10	-3	5	
	12				

Convolution

Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

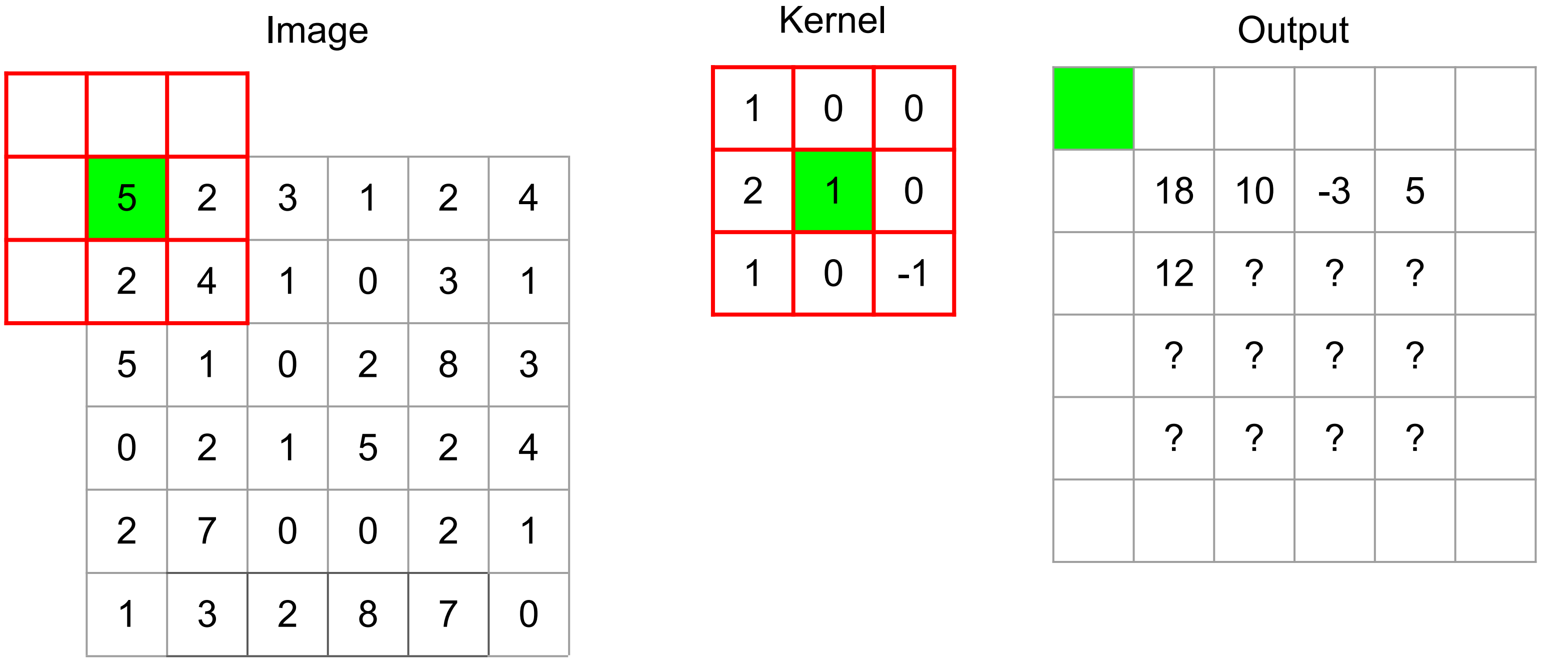
Kernel

1	0	0
2	1	0
1	0	-1

Output

	18	10	-3	5	
	12	?	?	?	
	?	?	?	?	
	?	?	?	?	

Convolution



Image

5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

🟩					
	18	10	-3	5	
	12	?	?	?	
	?	?	?	?	
	?	?	?	?	

Convolution: Zero padding

Image

0	0	0	0	0	0	0	0
0	5	2	3	1	2	4	0
0	2	4	1	0	3	1	0
0	5	1	0	2	8	3	0
0	0	2	1	5	2	4	0
0	2	7	0	0	2	1	0
0	1	3	2	8	7	0	0
0	0	0	0	0	0	0	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

-1					
	18	10	-3	5	
	12	?	?	?	
	?	?	?	?	
	?	?	?	?	

Convolution

Image

0	0	0	0	0	0	0	0
0	5	2	3	1	2	4	0
0	2	4	1	0	3	1	0
0	5	1	0	2	8	3	0
0	0	2	1	5	2	4	0
0	2	7	0	0	2	1	0
0	1	3	2	8	7	0	0
0	0	0	0	0	0	0	0

Kernel

1	0	0
2	1	0
1	0	-1

Output

-1	?	?	?	?	?
?	18	10	-3	5	?
?	12	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?

Kernels

- Feature detectors
- Kernels are learned

Oblique line detector

1	0	0
0	1	0
0	0	1

Vertical line detector

0	1	0
0	1	0
0	1	0

Architectural decisions for convolution

- Grid size
- Stride
- Depth
- Number of kernels

Grid size

- # of pixels for height/width
- Odd numbers

Grid size

- # of pixels for height/width
- Odd numbers

3 by 3

1	2	9
1	6	5
2	2	3

Grid size

- # of pixels for height/width
- Odd numbers

5 by 5

1	2	9	8	7
1	6	5	0	0
2	2	3	1	0
1	1	-3	0	-1
1	-2	2	2	3

Grid size

- # of pixels for height/width
- Odd numbers

5 by 5

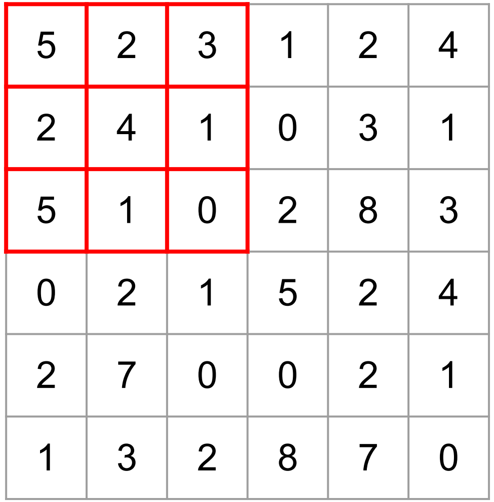
1	2	9	8	7
1	6	5	0	0
2	2	3	1	0
1	1	-3	0	-1
1	-2	2	2	3

Stride

- Step size used for sliding kernel on image
- Indicated in pixels

Stride

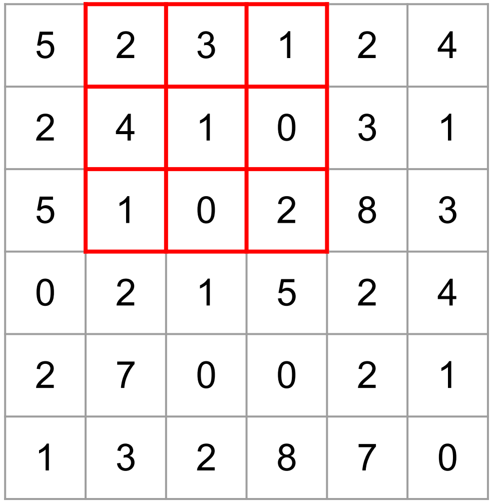
- Step size used for sliding kernel on image
- Indicated in pixels



5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Stride

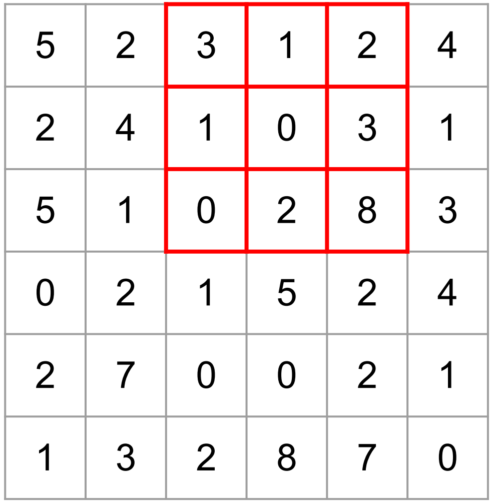
- Step size used for sliding kernel on image
- Indicated in pixels



5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Stride

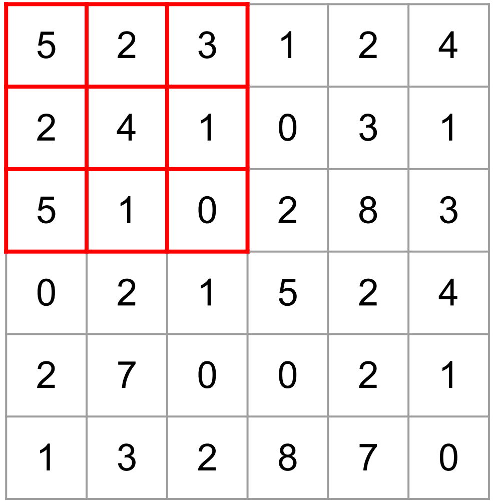
- Step size used for sliding kernel on image
- Indicated in pixels



5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Stride

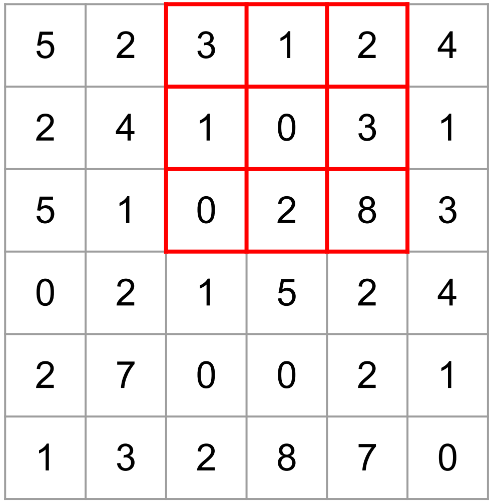
- Step size used for sliding kernel on image
- Indicated in pixels



5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

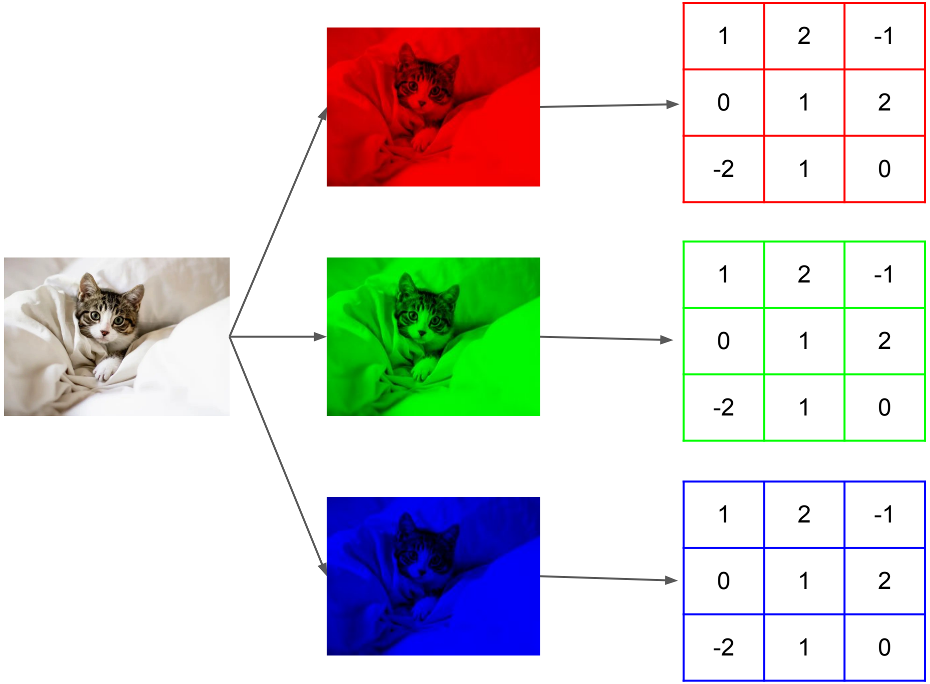
Stride

- Step size used for sliding kernel on image
- Indicated in pixels



5	2	3	1	2	4
2	4	1	0	3	1
5	1	0	2	8	3
0	2	1	5	2	4
2	7	0	0	2	1
1	3	2	8	7	0

Depth

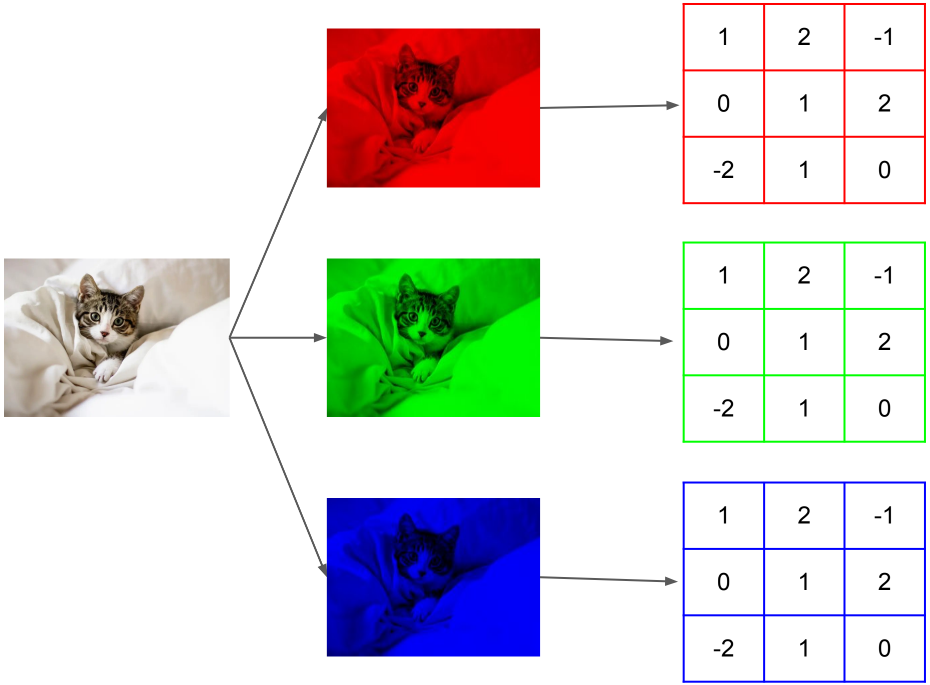


1	2	-1
0	1	2
-2	1	0

1	2	-1
0	1	2
-2	1	0

1	2	-1
0	1	2
-2	1	0

Depth



1	2	-1
0	1	2
-2	1	0

1	2	-1
0	1	2
-2	1	0

1	2	-1
0	1	2
-2	1	0

Kernel = 3 x 3 x 3

# weights = 27

# of kernels

- A conv layer has multiple kernels
- Each kernel outputs a single 2D array
- Output from a layer has as many 2d arrays as # kernels

Pooling

- Downsample the image
- Overlaying grid on image
- Max/average pooling
- No parameters

Pooling settings

- Grid size
- Stride
- Type (e.g., max, average)

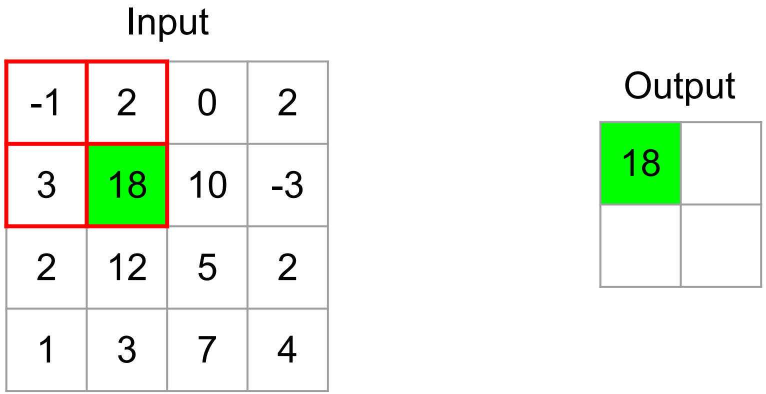
Max pooling (2x2, stride 2)

Input

-1	2	0	2
3	18	10	-3
2	12	5	2
1	3	7	4

Output

Max pooling (2x2, stride 2)



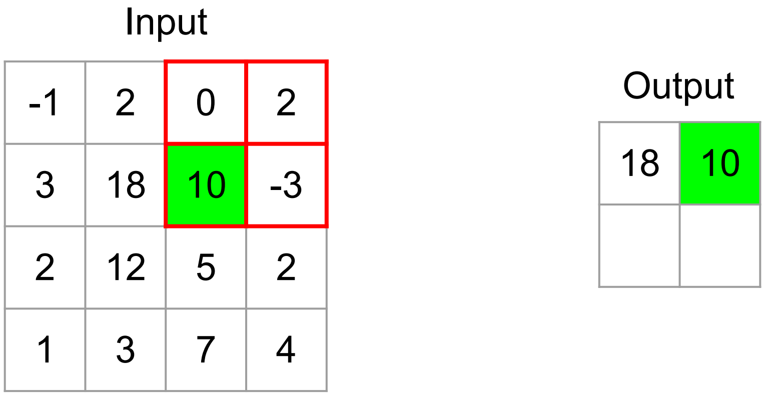
Input

-1	2	0	2
3	18	10	-3
2	12	5	2
1	3	7	4

Output

18	

Max pooling (2x2, stride 2)



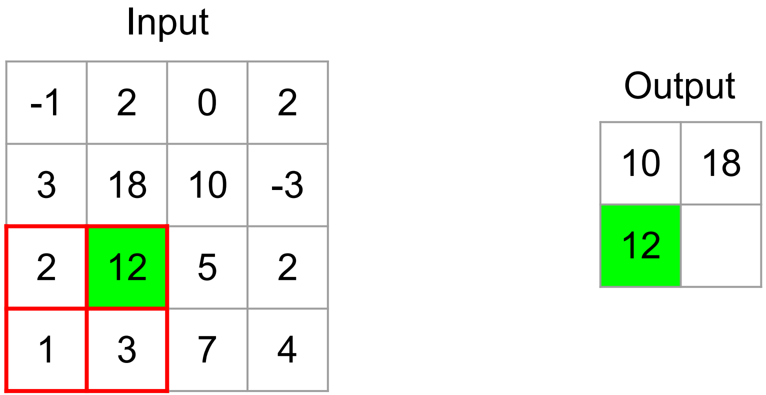
Input

-1	2	0	2
3	18	10	-3
2	12	5	2
1	3	7	4

Output

18	10

Max pooling (2x2, stride 2)



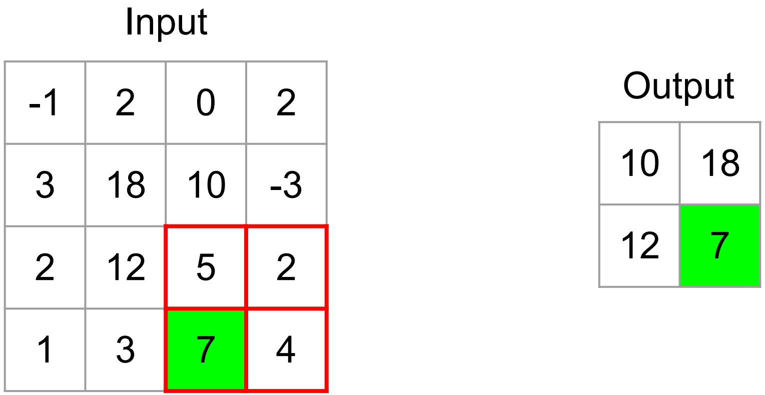
Input

-1	2	0	2
3	18	10	-3
2	12	5	2
1	3	7	4

Output

10	18
12	

Max pooling (2x2, stride 2)



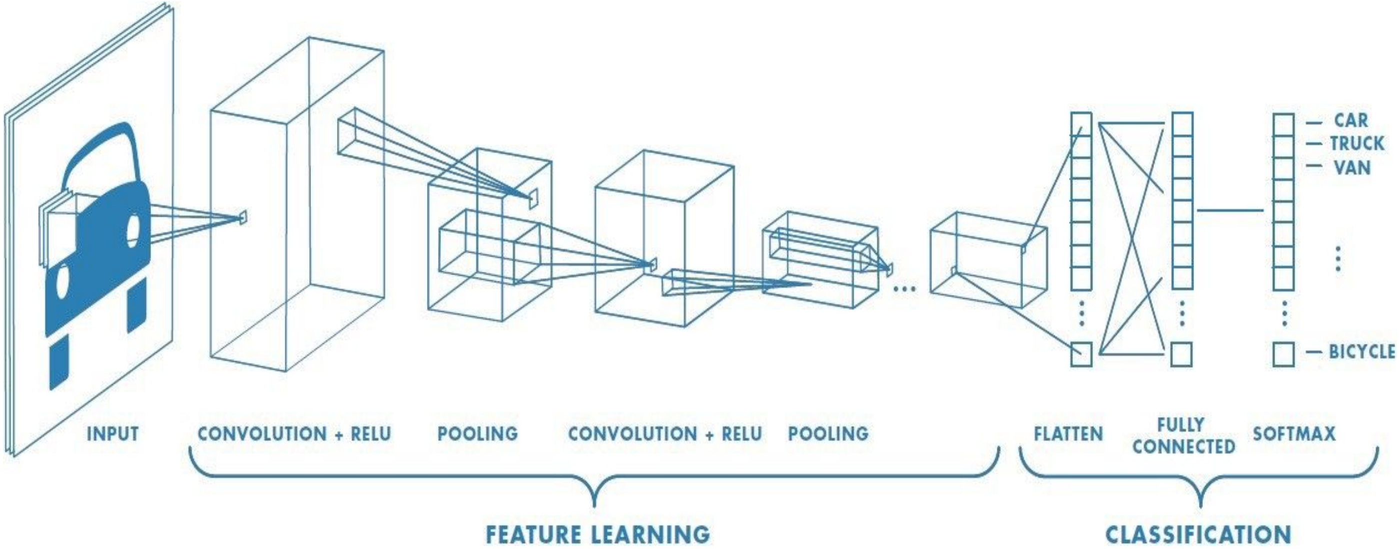
Input

-1	2	0	2
3	18	10	-3
2	12	5	2
1	3	7	4

Output

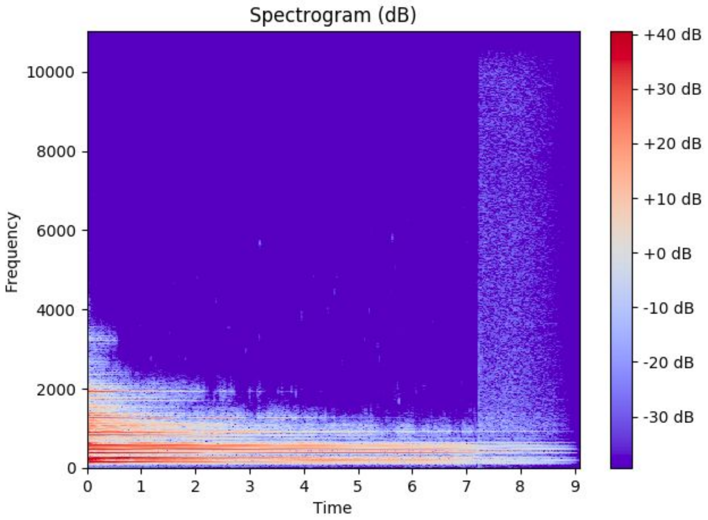
10	18
12	7

CNN architecture



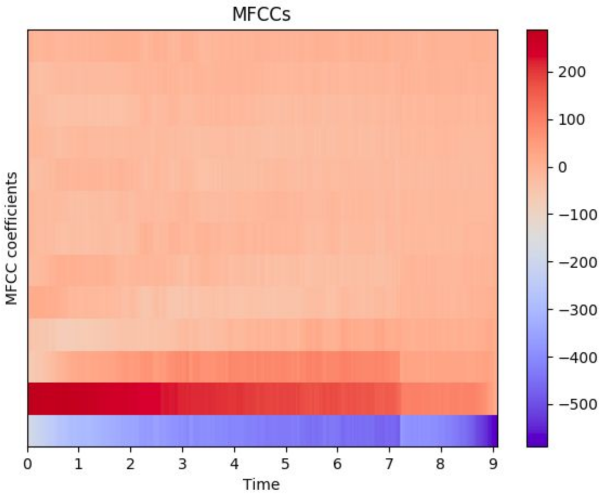
How does convolution/pooling apply to audio?

- Spectrogram/MFCC = image
- Time, frequency = x, y
- Amplitude = pixel value



Preparing MFCCs for a CNN

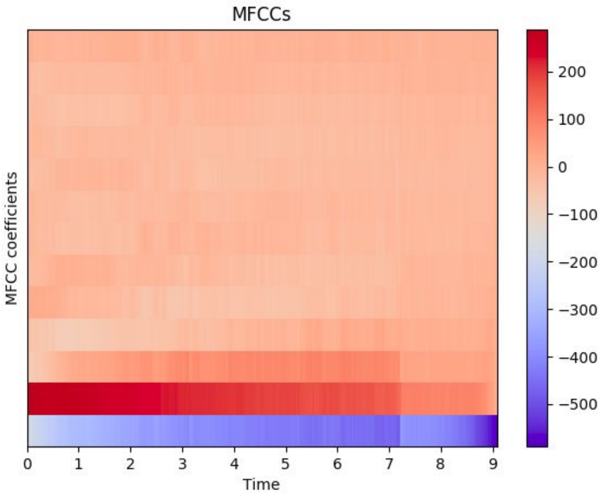
- 13 MFCCs
- Hop length = 512 samples
- # samples in audio file = 51200



Preparing MFCCs for a CNN

- 13 MFCCs
- Hop length = 512 samples
- # samples in audio file = 51200

Data shape = 100 x 13 x 1



What’s up next?

- Implement CNN for music genre classification