

A decorative background featuring a network diagram with nodes and connecting lines. The nodes are represented by circles of varying sizes and colors (blue, grey, and white), and the lines are thin and grey. The network is spread across the top and bottom corners of the slide.

Convolutional Neural Networks

Rodrigo Gonzalez, PhD

Hello!

I am Rodrigo Gonzalez, PhD

You can find me at rodraz@qmail.com



Summary

1. Main ideas
2. Basic CNN architecture
3. Padding and stride
4. CNN in Keras

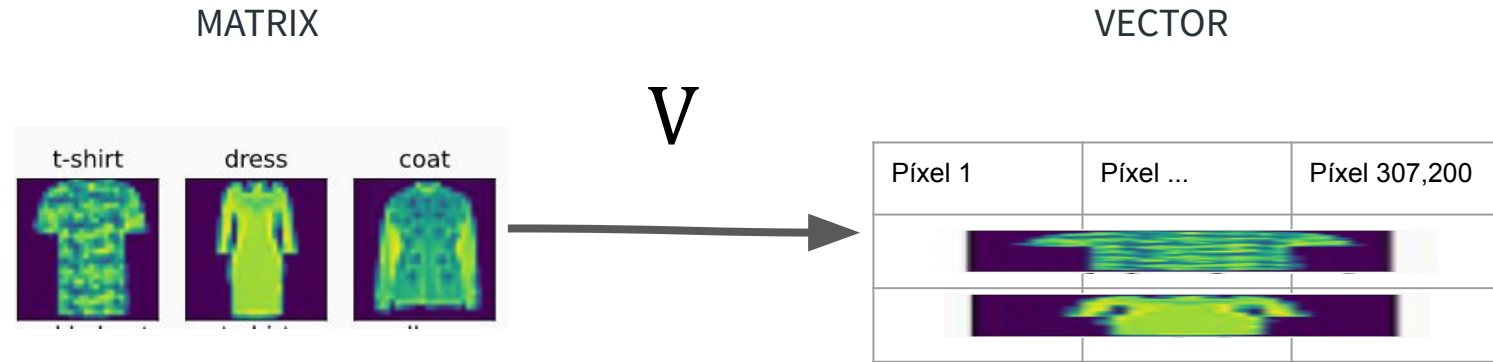


1.

Main ideas

The goals of a CNN

Common neural network approach



640 pixels x 480 pixels = 307,200 input neurons

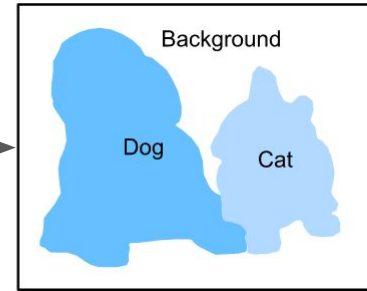
CNN approach

MATRIX

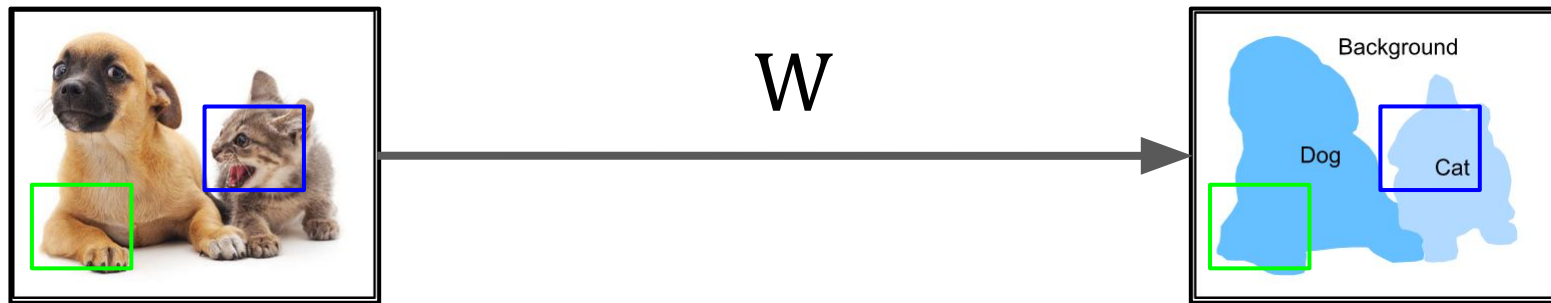
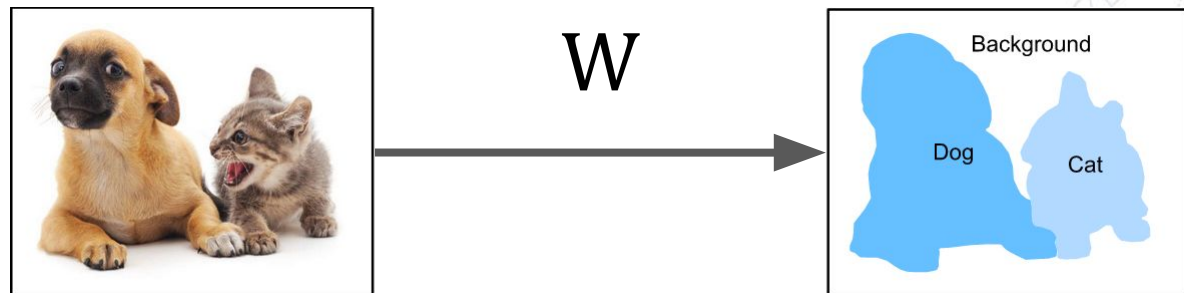


W

MATRIX



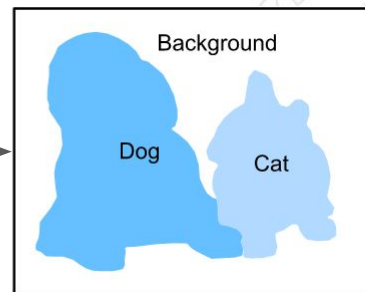
Locality



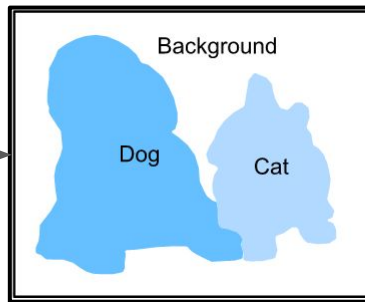
Translation invariance



W



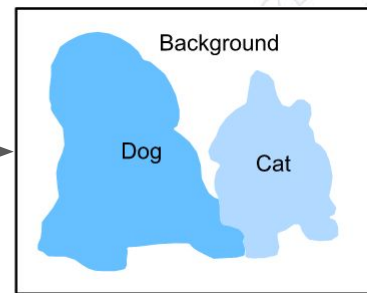
W



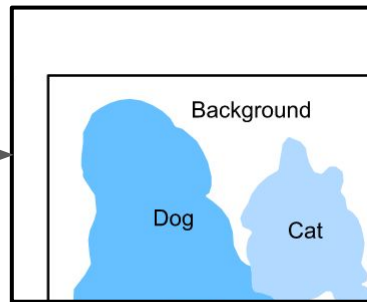
Translation invariance



W



W



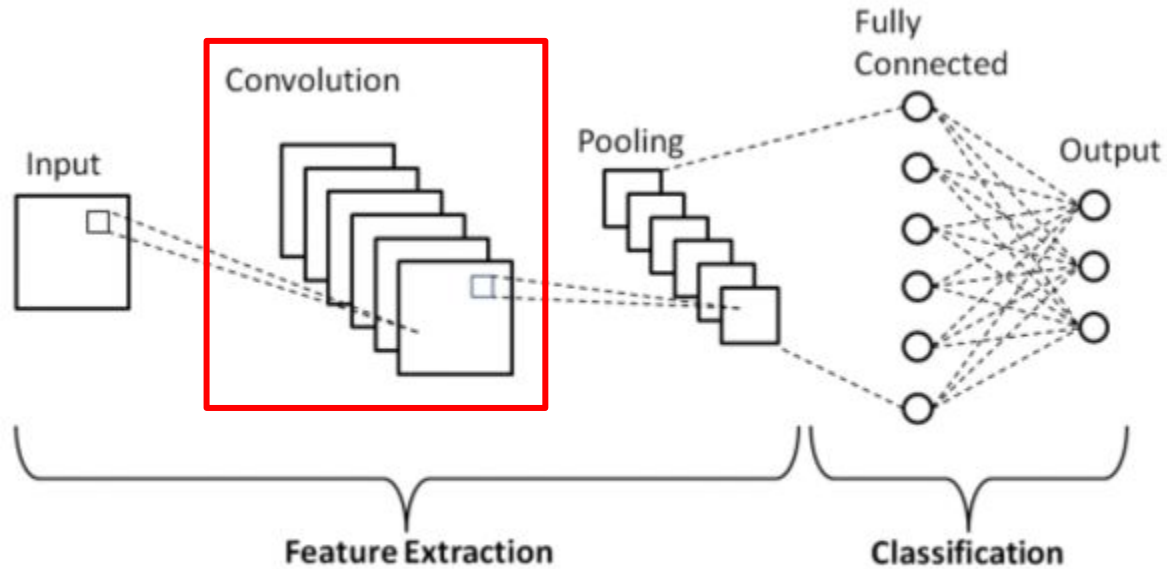


2.

Basic CNN architecture

Kernel and pooling

Basic CNN architecture



Convolution

2D convolution is a **dot product** between an image (nxn matrix) and a **kernel** (3x3)

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Convolution

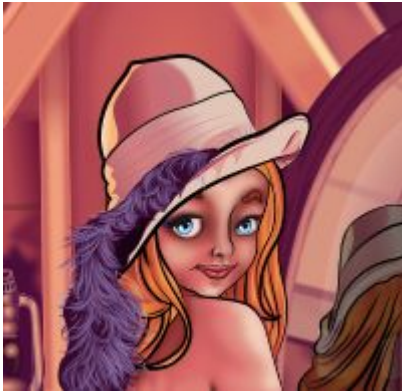
2D convolution is a **dot product** between an image (nxn matrix) and a **kernel** (3x3)

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

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Common 2D convolution

Edge detector



Kernel

-1	-1	-1
-1	8	-1
-1	-1	-1



<https://planetcalc.com/9313/>

Common 2D convolution

Sharpening



Kernel

-1	-1	-1
-1	9	-1
-1	-1	-1



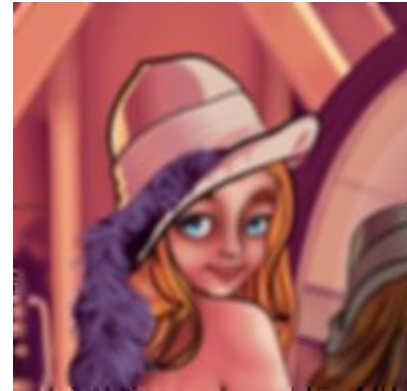
Common 2D convolution

Gaussian



Kernel

1	2	3	2	1
2	4	5	4	2
3	5	6	5	3
2	4	5	4	2
1	2	3	2	1

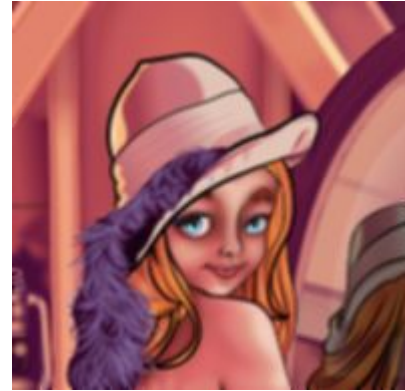


Common 2D convolution

Smoothing



Kernel		
1	1	1
1	2	1
1	1	1



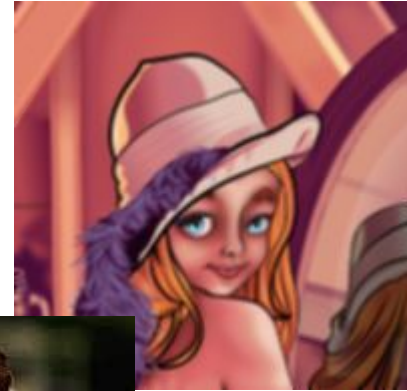
Common 2D convolution

Smoothing

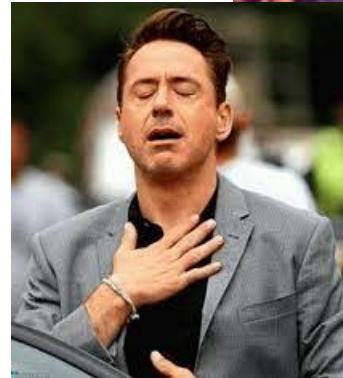


Kernel

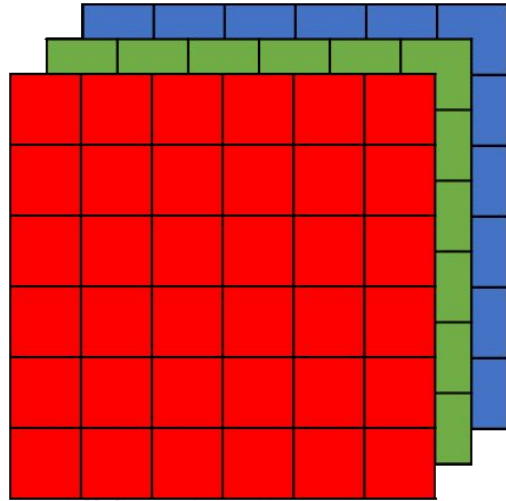
1	1	1
1	2	1
1	1	1



Don't worry! The CNN will learn the kernels!

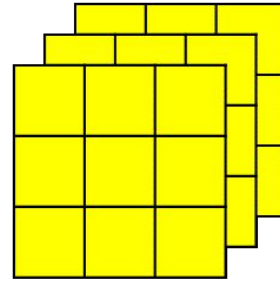


Convolutions on RGB image



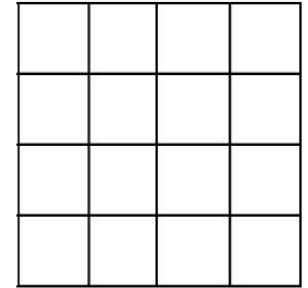
$6 \times 6 \times 3$

*



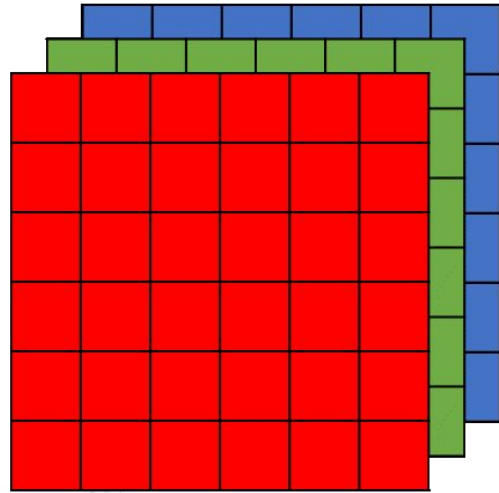
$3 \times 3 \times 3$

=



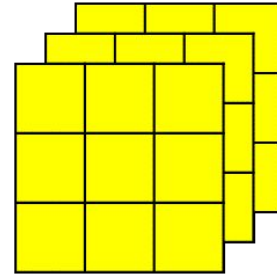
4×4

Convolutions on RGB image



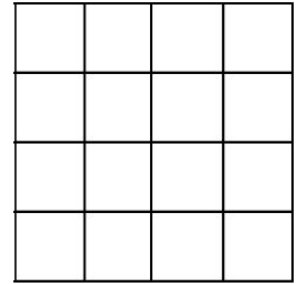
$6 \times 6 \times 3$

*

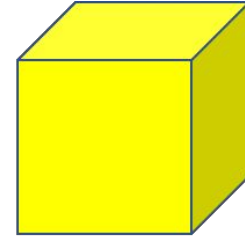


$3 \times 3 \times 3$

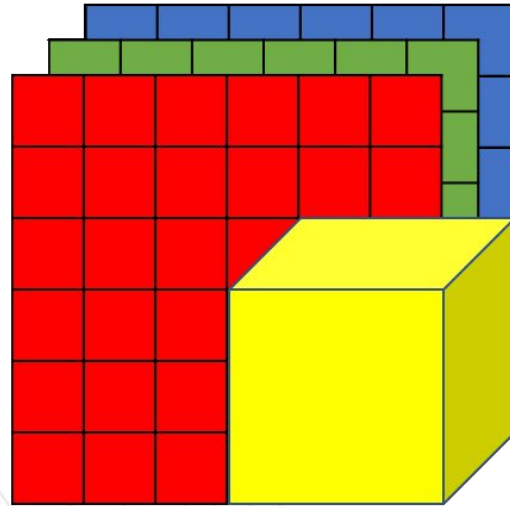
=



4×4

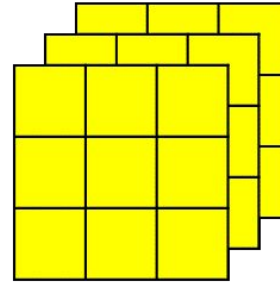


Convolutions on RGB image



$6 \times 6 \times 3$

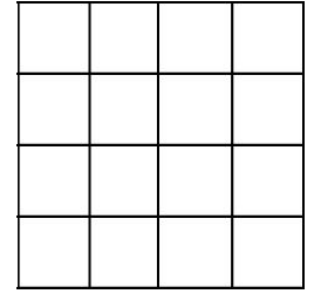
*



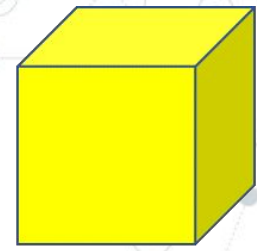
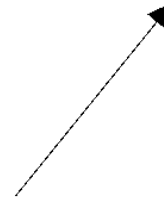
$3 \times 3 \times 3$

27 numbers

=

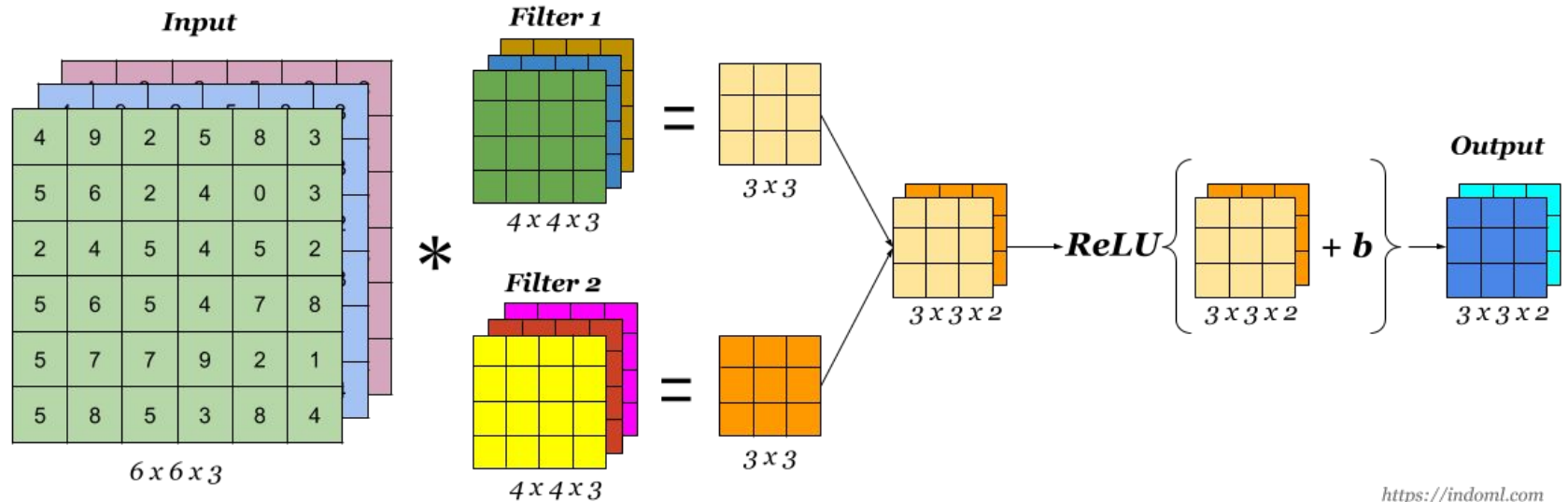


4×4



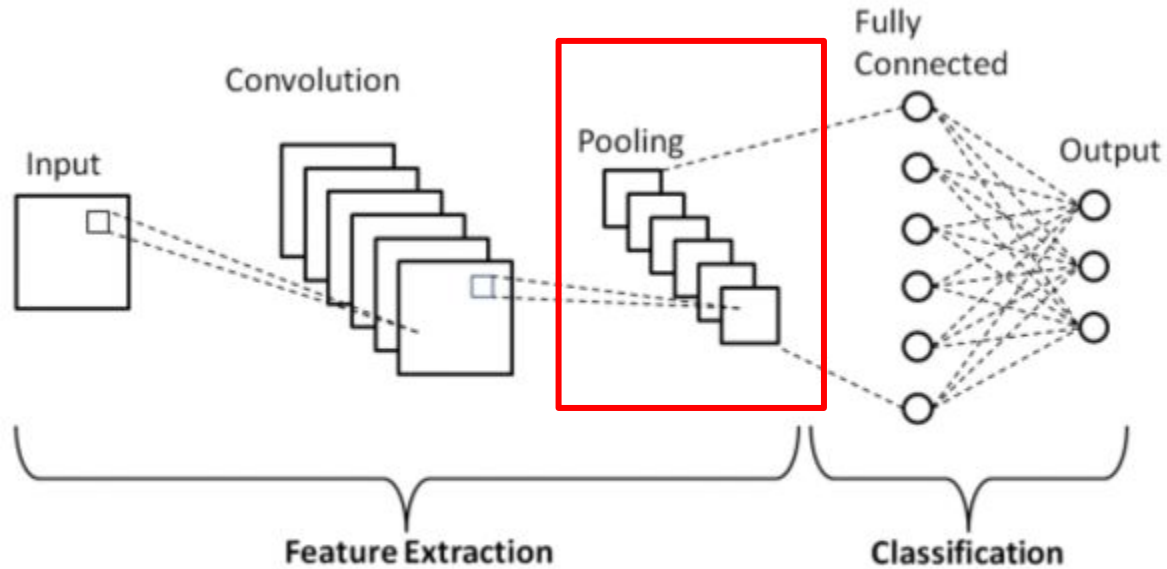
Convolutions on RGB image

A Convolution Layer



<https://indoml.com>

Basic CNN architecture



Pooling



Pooling



Pooling



- ◎ Pooling is image compression
- ◎ Reduce computation



Max pooling and Average pooling

No need to learn a Max Pooling

3	13	17	11
5	3	1	23
7	1	2	3
11	17	1	4

Max Pooling

13	23
17	4

Average Pooling

6	13
9	2.5



3.

Padding and stride

Padding

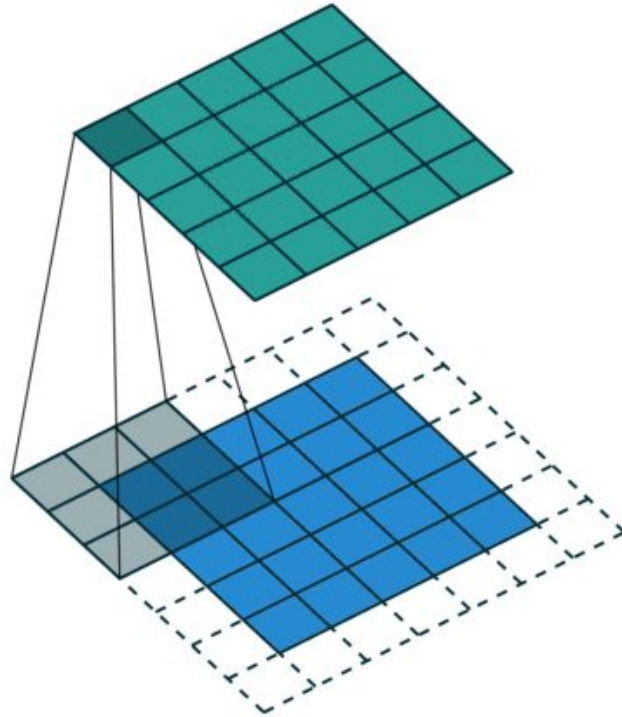
In general, a 2D convolution reduces the size of the image

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Padding

In general, a 2D convolution reduces the size of the image



Stride

Stride is the number of rows and columns traversed per slide. In general, a 2D convolution the kernel moves 1 row at a time in both directions.

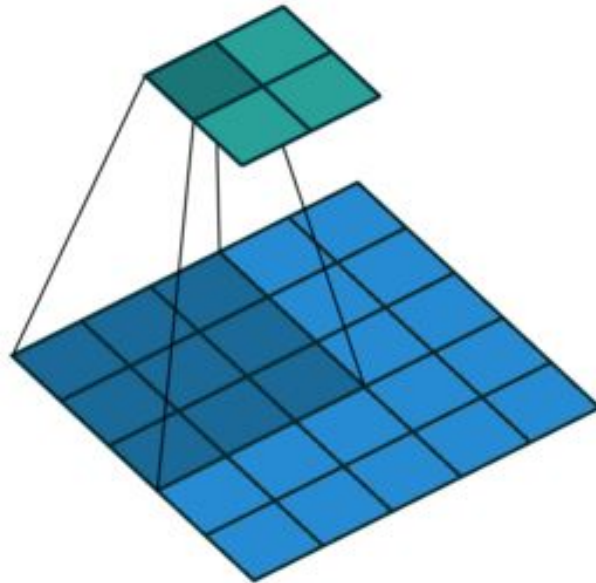
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Stride

Stride of 2 vertically and 2 horizontally.

- ⦿ Computational efficiency
- ⦿ Downsampling





4. **CNN in Keras**

Keras

Listing 8.1 Instantiating a small convnet

```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(inputs)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

LeNet-5

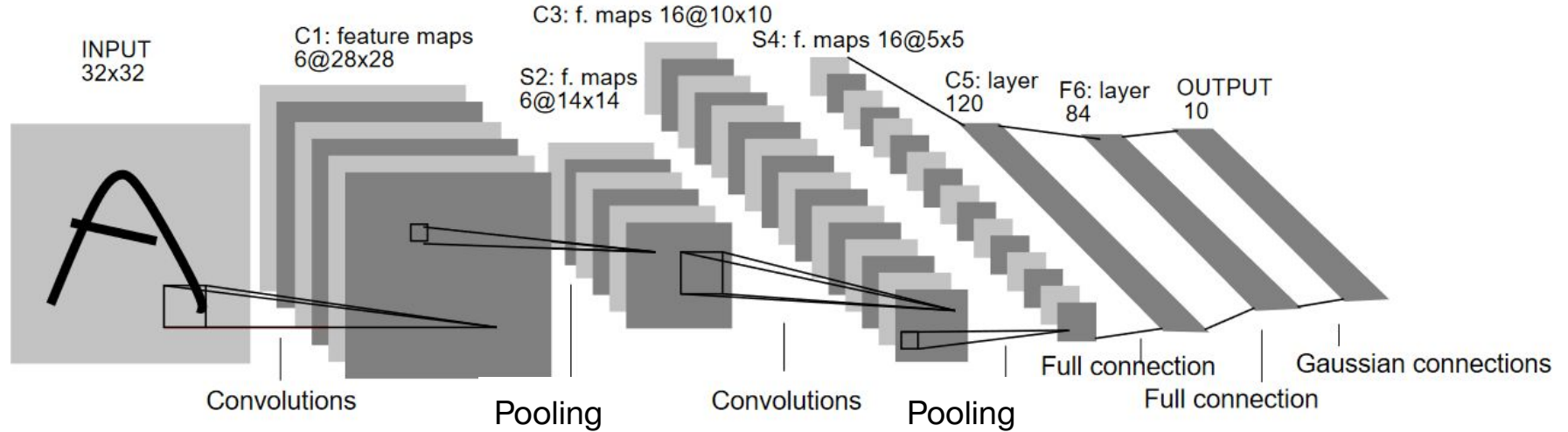


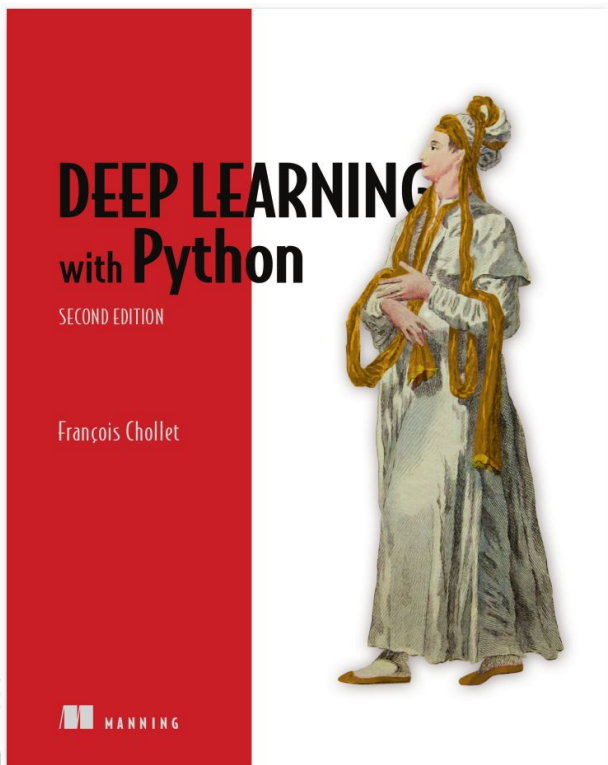
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner:
Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE*, 86(11):2278-2324, November **1998**,
[\cite{lecun-98}](#).

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting a hierarchical or multi-layered structure. The lines are thin and gray, connecting the nodes in a non-linear fashion.

5. Book

Deep Learning with Python, 2nd Ed. by Francois Chollet



Chapter 8



Thanks!

Any questions?

You can find me at:
rodraz@gmail.com