

A decorative background graphic consisting of a network of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid blue, some are hollow blue, and others are grey. The lines are thin and grey, forming a complex web-like structure that is more dense on the left and right sides of the slide.

# Convolutional Neural Networks

Rodrigo Gonzalez, PhD

# Hello!

## I am Rodrigo Gonzalez, PhD

You can find me at

[rodrigo.gonzalez@ingenieria.uncuyo.edu.ar](mailto:rodrigo.gonzalez@ingenieria.uncuyo.edu.ar)



# Summary

1. A little of history
2. Main ideas
3. Basic CNN architecture
4. Padding and stride
5. CNN example
6. CNN in Keras
7. The book



1.

# A little of history

The goals of a CNN

## A little of history

1. Computer vision is the earliest and biggest success story of deep learning.
2. 1998, LeNet-5 (Yann LeCunn et al., 1989)
3. 2011, Dan Ciresan wins the ICDAR 2011 Chinese character recognition competition and the IJCNN 2011 German traffic signs recognition competition
4. 2012, AlexNet (Alex Krizhevsky et al., 2012). Hinton's group winning the high-profile ImageNet large-scale visual recognition challenge.
5. In 2013 and 2014, deep learning still faced intense skepticism from many senior
6. computer vision researchers.
7. It was only in 2016 that it finally became dominant.

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in blue.

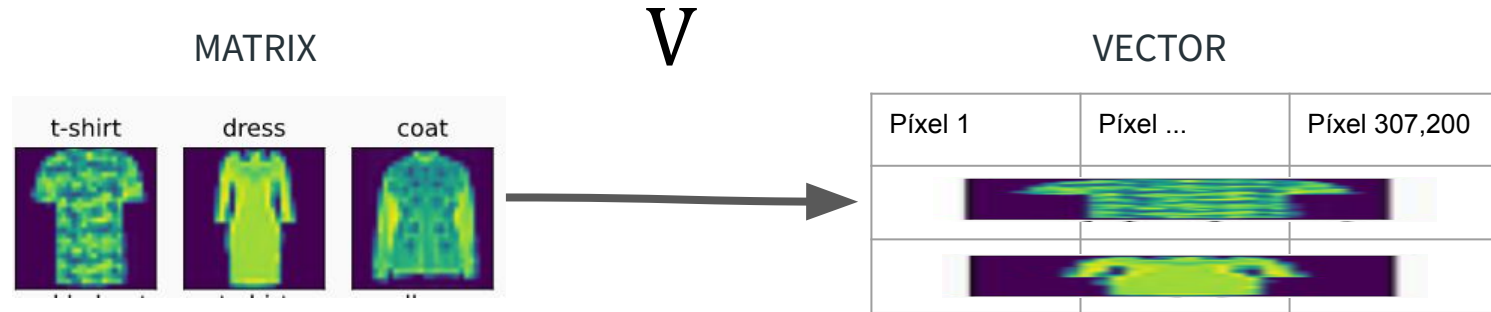
# 2.

## **Main ideas**

The goals of a CNN

# Common neural network approach

Dense layers learn **global** patterns in their input feature space



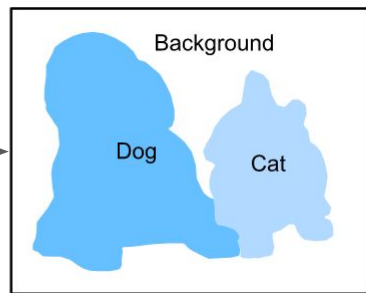
640 pixels x 480 pixels = 307,200 neurons

## CNN approach

Convolution layers learn **local** patterns—in the case of images, patterns found in small 2D windows of the inputs

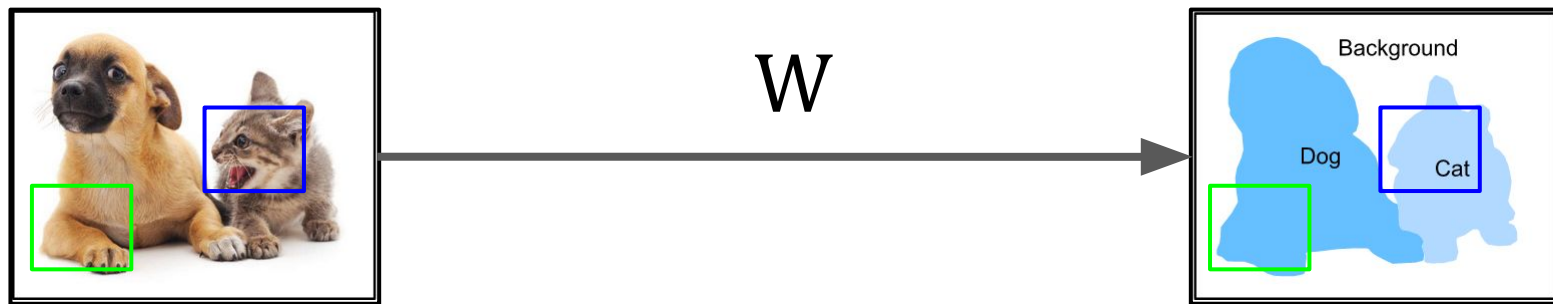
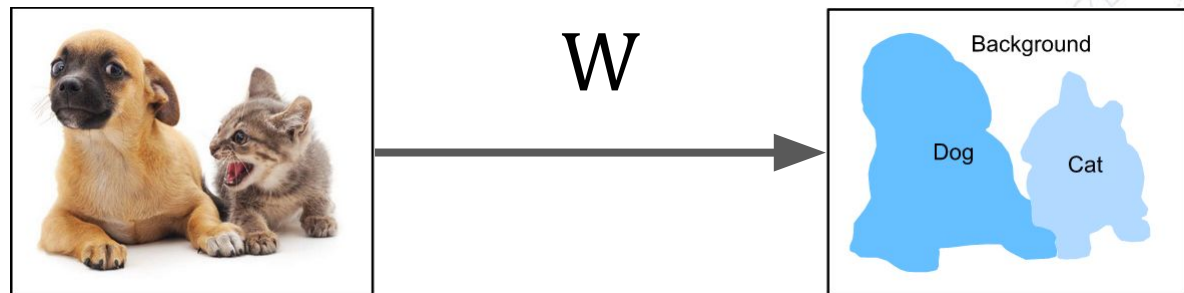


$W$





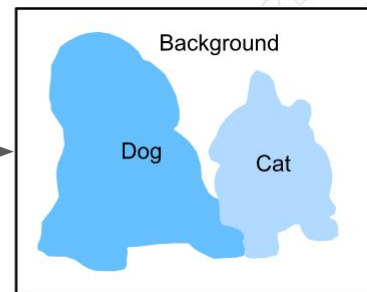
## Locality



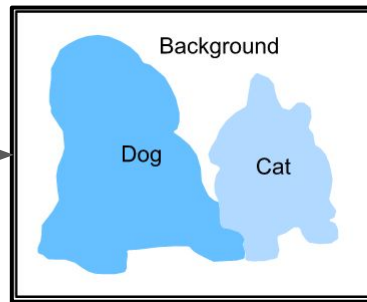
## Translation invariance



$W$



$W$

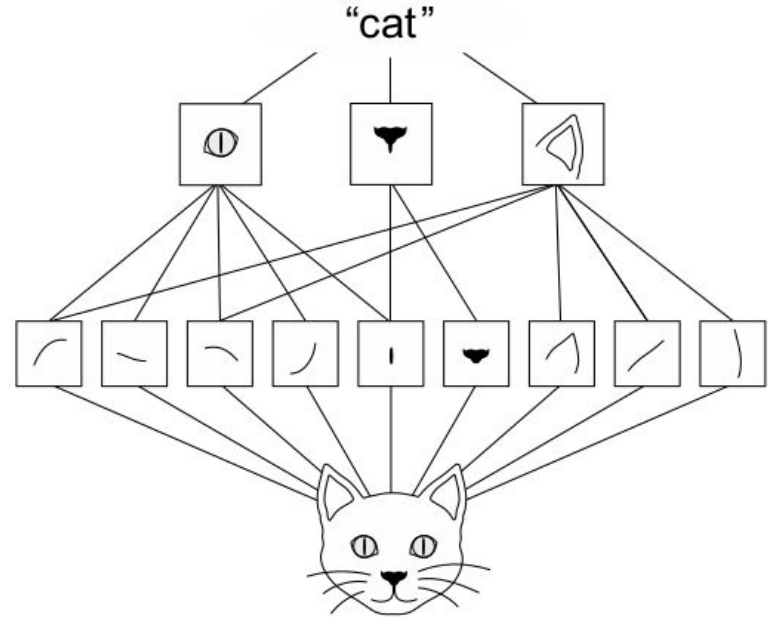


## Translation invariance



## Spatial hierarchies of patterns

- A first convolution layer will learn small local patterns such as edges
- A second convolution layer will learn larger patterns made of the features of the first layers, and so on



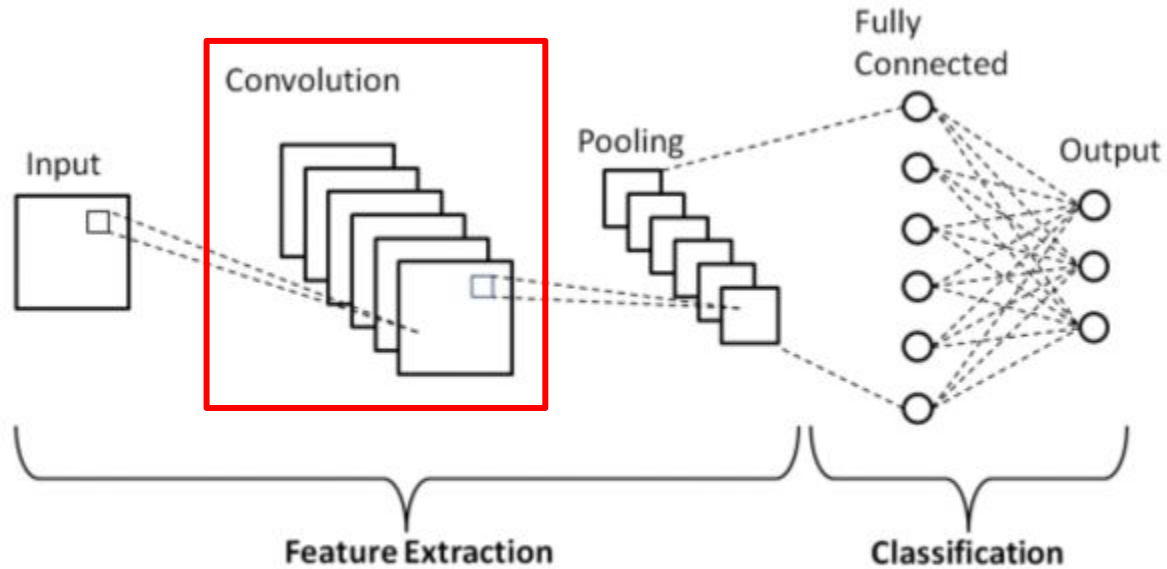


3.

# 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)

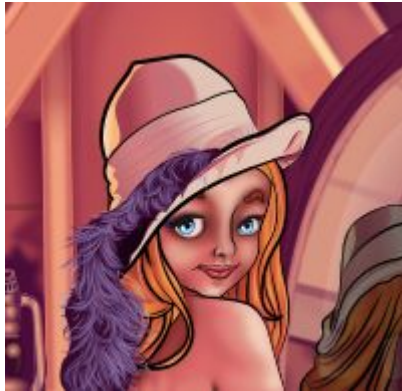
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

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



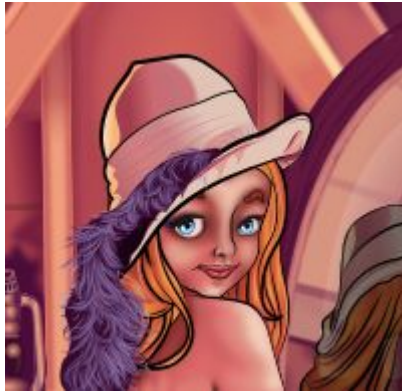
Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



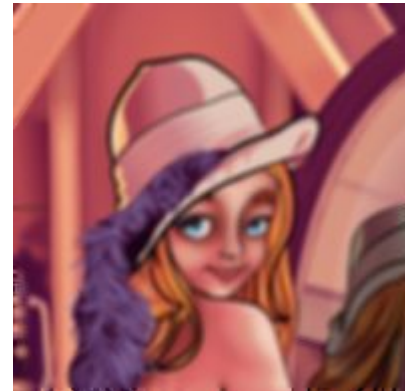
# Common 2D convolution

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



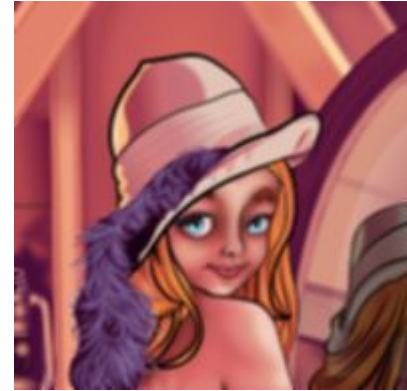
# Common 2D convolution

Smoothing



Kernel

1	1	1
1	2	1
1	1	1



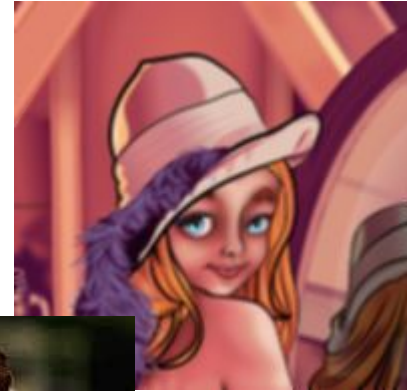
# Common 2D convolution

Smoothing

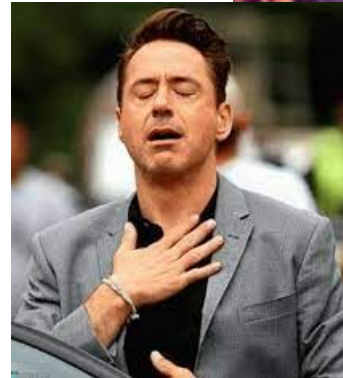


Kernel

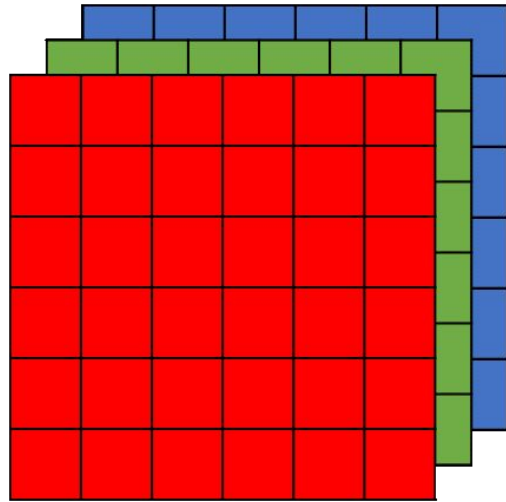
1	1	1
1	2	1
1	1	1



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

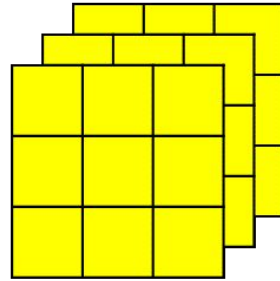


## Convolutions on RGB image



$$6 \times 6 \times 3$$

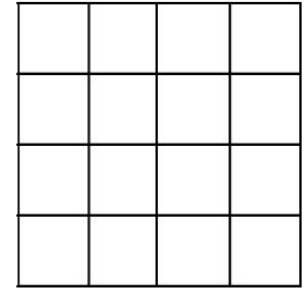
\*



$$3 \times 3 \times 3$$

$$k_h \times k_w$$

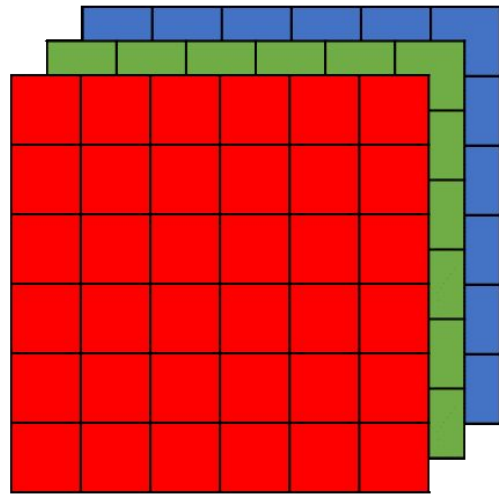
=



$$4 \times 4$$

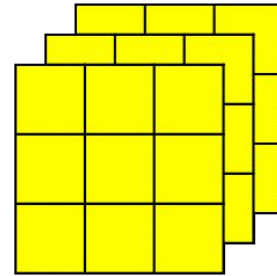
$$(n_h - k_h + 1) \times (n_w - k_w + 1).$$

## Convolutions on RGB image



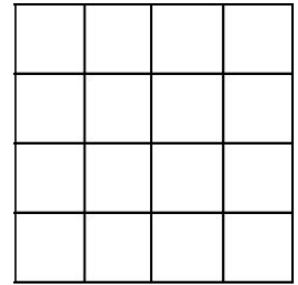
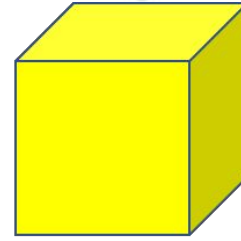
$6 \times 6 \times 3$

\*



$3 \times 3 \times 3$

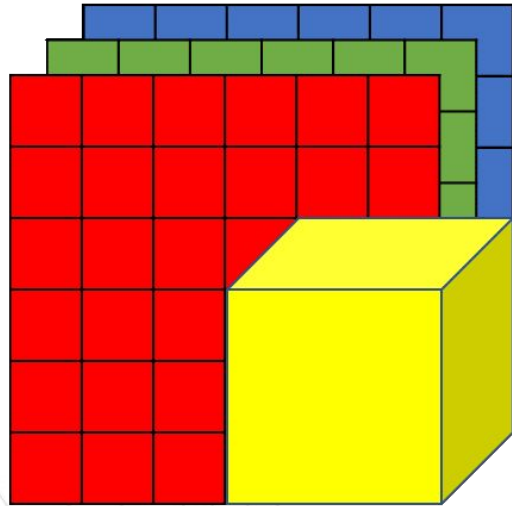
=



$4 \times 4$

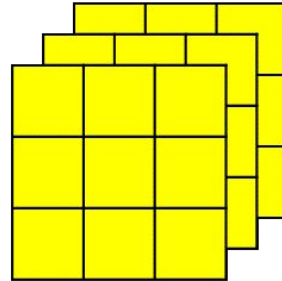


## Convolutions on RGB image



$6 \times 6 \times 3$

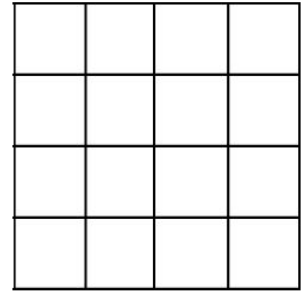
\*



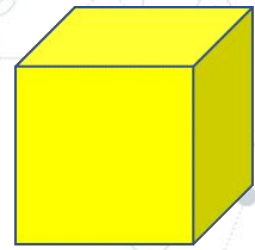
$3 \times 3 \times 3$

27 numbers

=



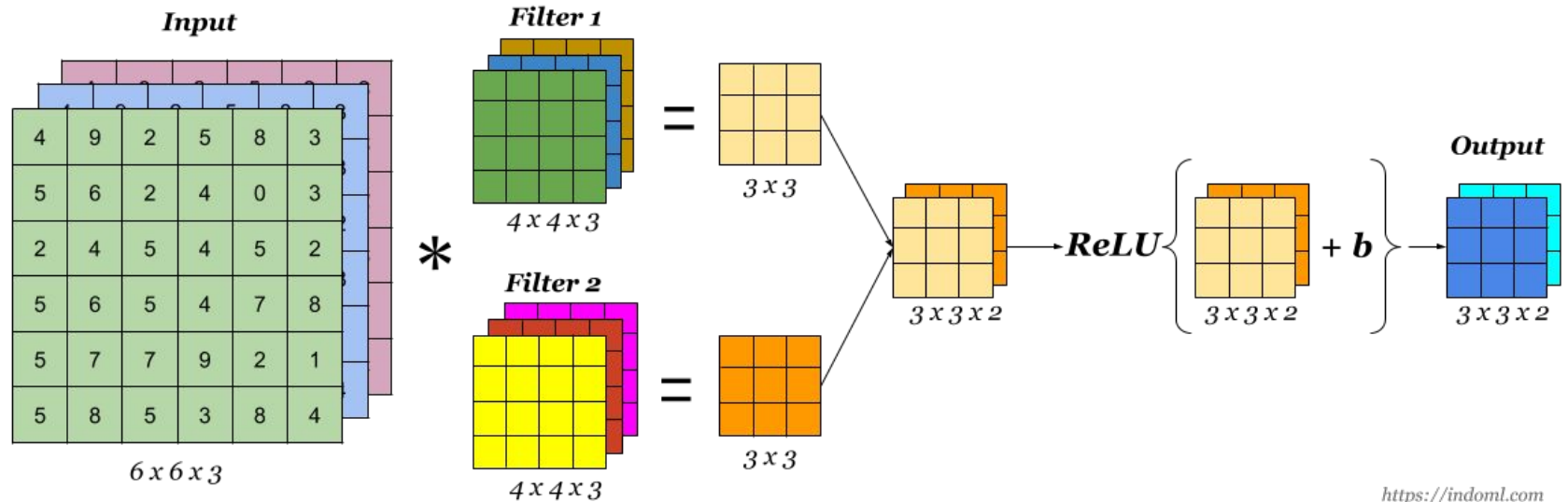
$4 \times 4$





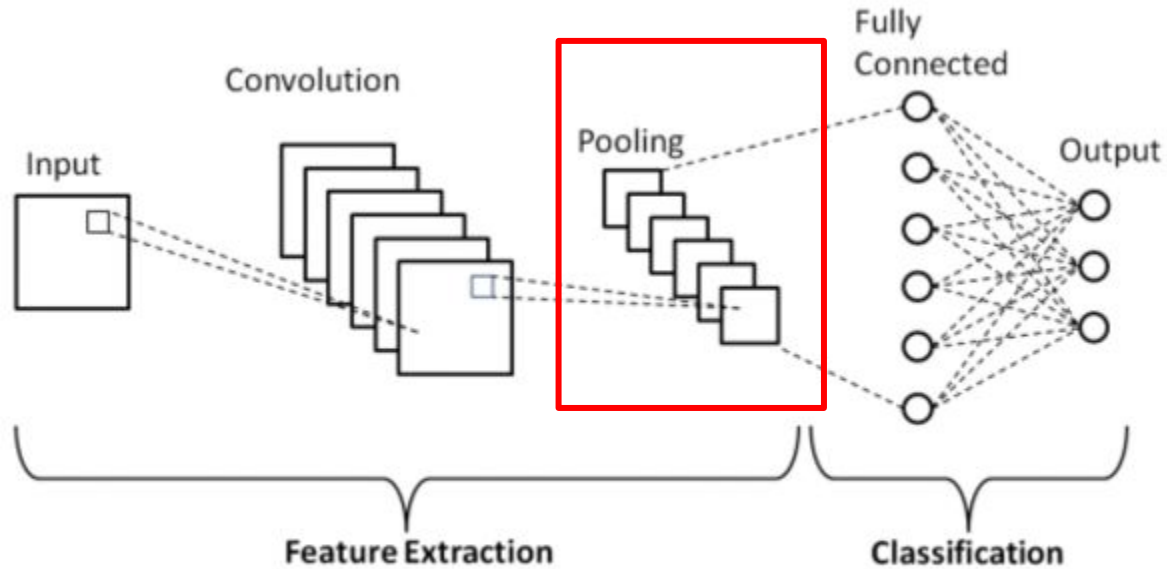
# Convolutions on RGB image

## 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

## Padding

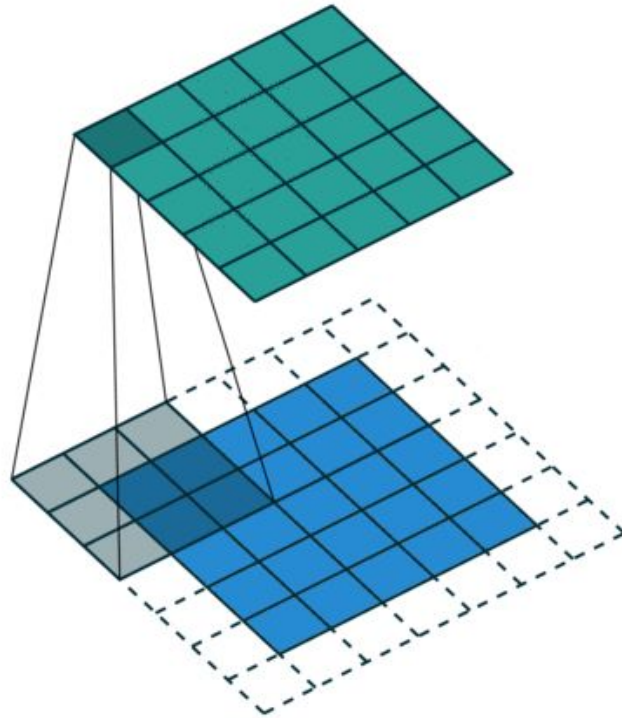
In general, a 2D convolution reduces the size of the image

$3_0$	$3_1$	$2_2$	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.

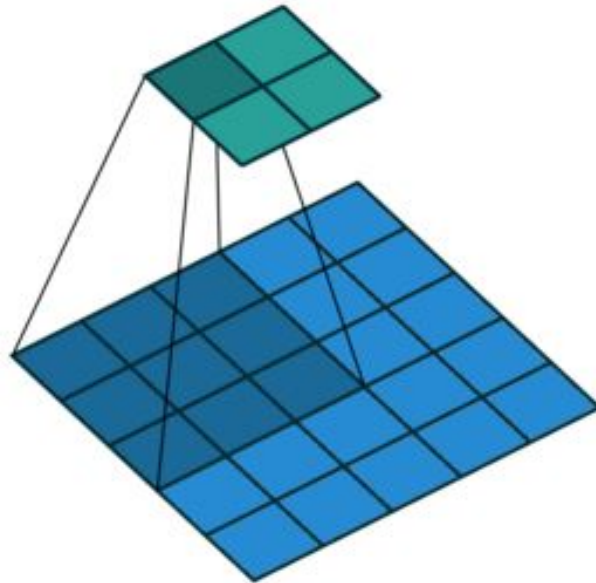
$3_0$	$3_1$	$2_2$	1	0
$0_2$	$0_2$	$1_0$	3	1
$3_0$	$1_1$	$2_2$	2	3
2	0	0	2	2
2	0	0	0	1


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.

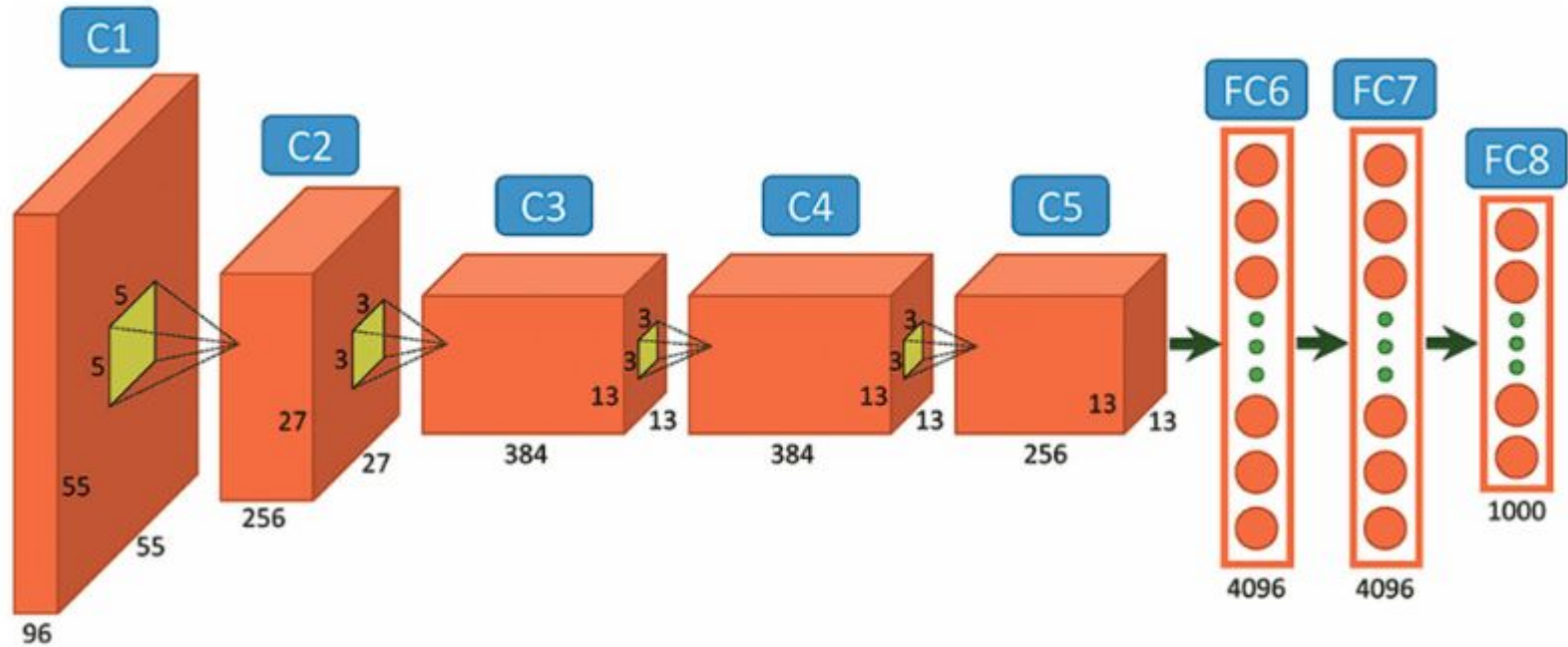
- ⦿ Computational efficiency
- ⦿ Downsampling





# 5. CNN example

# AlexNet



# AlexNet

## Input Layer

AlexNet takes images of the Input size of  $227 \times 227 \times 3$  RGB Pixels.

## Convolutional Layers

1. First Layer: The first layer uses 96 kernels of size  $11 \times 11$  with a stride of 4, activates them with the ReLU activation function, and then performs a Max Pooling operation.
2. Second Layer: The second layer takes the output of the first layer as the input, with 256 kernels of size  $5 \times 5 \times 48$ .
3. Third Layer: 384 kernels of size  $3 \times 3 \times 256$ . No pooling or normalization operations are performed on the third, fourth, and fifth layers.
4. Fourth Layer: 384 kernels of size  $3 \times 3 \times 192$ .
5. Fifth Layer: 256 kernels of size  $3 \times 3 \times 192$ .

## Fully Connected Layers

The fully connected layers have 4096 neurons each.

## Output Layer

The output layer is a SoftMax layer that outputs probabilities of the 1000 class labels.



# 6. **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)
```

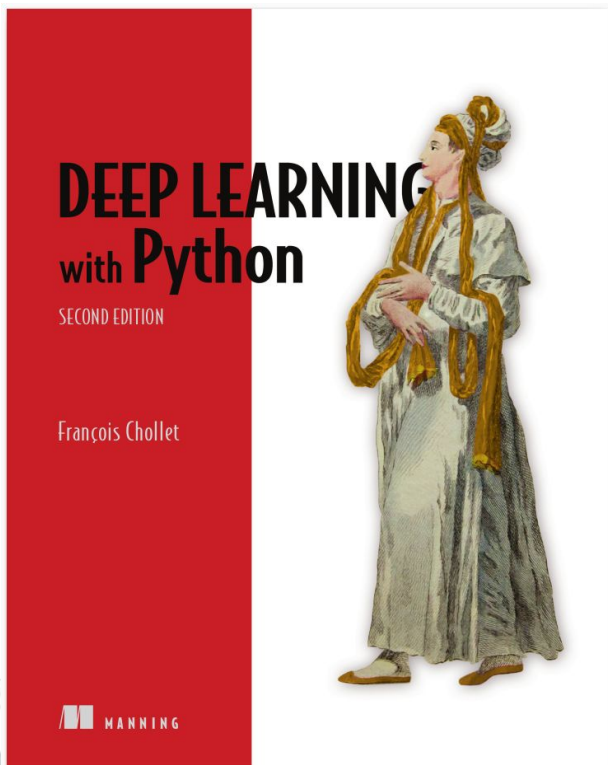
A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting a hierarchical or multi-layered structure. The lines are thin and gray, connecting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

# 7.

## **The book**



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



## Chapter 8