AGENDA

GENERAL IDEA The origin of CNNs

How does a CNN work?

Example VGGnet



GENERAL IDEA

THE ORIGIN OF CNNS



THE ORIGIN OF CNNS

- Used for image recognition since 1980s
- INSPIRED BY THE BRAIN'S VISUAL CORTEX
 - NEURONS IN VISUAL CORTEX HAVE A SMALL LOCAL RECEPTIVE FIELD
 - RECEPTIVE FIELDS OF DIFFERENT NEURONS OVERLAP
 - TOGETHER THEY TILE THE WHOLE VISUAL FIELD
 - SOME NEURONS ONLY REACT TO SPECIFIC SHAPES
 - SOME NEURONS REACT TO MORE COMPLEX SHAPES FROM LOWER
 LEVELS
- POWERFUL ARCHITECTURE OF LOWER AND HIGHER-LEVEL NEURONS TO DETECT COMPLEX PATTERNS



Source: Géron (2019)

FIELDS OF APPLICATIONS

MAGE DETECTION •





VOICE RECOGNITION



NATURAL LANGUAGE PROCESSING (NLP) ightarrow





CONVOLUTION

HOW A CNN WORKS



HOW DOES A CNN WORK?





WHAT DOES A COMPUTER SEE?



Input image 3600 x 2400 Resized image 30 x 30

									0.1																					
								0.1																						
										0.2	0.2				0				0											
														0.1		0			0.1	0	0	0								
				0.1	0.1	0.1	0.1				0.2										0.2									
				0.3	0.4	0.4	0.2	0.2	0.3	0.3	0.3	0.2	0.2	0.2	0.1			0.1			0.5	0.4								
					0.2	0.3	0.4	0.5	0.5	0.6	0.5	0.4	0.4	0.4	0.2	0.1	0.1			0.2	0.2									
							0.3	0.6	0.6	0.6	0.7	0.6	0.6	0.5	0.1				0											
								0.5	0.5	0.6	0.7	0.6	0.6	0.4	0.1			0												
								0.1	0.3	0.6	0.6	0.6	0.6	0.4	0.1				0											
									0.2	0.5	0.6	0.6	0.6	0.4						0	0	0	0	0						
0	0	0	0	0	0	0	0	0	0.1	0.4	0.6	0.6	0.6	0.4			0		0.1	0.2	0.1			0.1	0	0	0	0	0	
0	0	0	0	0	0	0	0	0.1	0.2		0.6	0.6	0.4	0.3	0.2	0.1	0	0	0	0.2				0.1	0.2	0.2	0.1	0	0	
0	0	0	0	0.1	0.2	0.2	0.2	0.4	0.2	0.3	0.6	0.6	0.4	0.4	0.4	0.2		0	0	0			0.1	0.4	0.6	0.5	0.4	0.2	0.1	
0	0	0.1	0.3	0.5	0.6	0.4		0.4	0.2	0.4	0.6	0.6	0.5	0.6	0.2			0					0.4	0.6	0.6	0.6	0.5	0.2	0.2	
	0.1	0.3	0.6	0.6	0.6	0.5	0.3	0.4	0.4	0.5	0.6	0.6	0.6	0.6				0	0	0.1	0.1	0.1	0.5	0.7	0.6	0.6	0.5	0.3	0.2	
	0.2	0.0	0.7	0.7	0.7	0.0	0.2		0.5	0.6	0.6	0.0	0.0	0.6		0.1	0.1	0		0	0	0.2	0.4	0.6	0.0	0.7	0.5	0.3	0.2	
	0.4	0.7	0.7	0.7	0.7	0.0	0.4		0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.2	0					0.4	0.5	0.5	0.0	0.7	0.5	0.2	0.2	
11 12	0.6	0.7	0.7	0.7	0.7	0.0	0.5	0.5	0.4	0.7	0.7	0.6	0.5	0.5	0.4	0.1	0		0.1	0.1	0.1	0.4	0.6	0.4	0.5	0.7	0.4	0.2	0.3	
	0.0	0.7	0.7	0.7	0.0	0.0	0.5	0.4	0.4	0.3	0.7	0.5	0.0	0.0	0.3	0.1	0.1			0.1	0.4	0.4	0.5	0.5	0.4	0.3		0.4	0.4	
2.5	0.7	0.0	0.6	0.7	0.0	0.6	0.5	0.5	0.5	0.4	0.3	0.5	0.6	0.6	0.2	01	03	0.2	0	0.2	0.4	0.5	0.4	0.4	0.1	0.2	0.2	0.5	0.5	
0.6	0.7	0.7	0.6	0.6	0.6	0.5	0.6	0.7	0.5	0.3		0.4	0.6	0.6	0.4	0.3	0.4	0.3	0.2		0.1	0.1	0.3	0.3	0	ů 0	0.1	0.4	0.4	
0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.7	0.7	0.6	0.2		03	0.5	0.5	0.5	0.5	0.6	0.5	0.2				01	01	0	0	0	01	0.2	
0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.3		0.4	0.5	0.5	0.5	0.5	0.6	0.5	0.3		0.3	0.2			0	ő	ů 0	0	0.1	
0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.5	0.4	0.5	0.4	0.3	0.2	0.5	0.6	0.5				0.3	0.2		0.1	0	0	0	0	
0.6	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.6	0.6	0.4	0.2	0.1	0.3	0.4	0.4			0.2	0.2	0.2			0	0	0		
).5	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.6	0.6	0.4			0.1	0.1	0.1	0.1	0.2		0.1	0.1			0.1			0.1	
).5	0.5	0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.6	0.6	0.3	0.1	0	0	0	0	0	0.2		0.2	0.2		0	0	0	0.1	0.1	
1		1									-																			

Image as matrix of numbers [0, 1]

0	0	0	0	0	0	0	0	0	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.1	0.2	0.2	0.2	0.2	0.2	0.1	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.1	0.2	0.3	0.3	0.3	0.2	0.2	0.1	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.1	0.1	0.1	0.1	0.2	0.3	0.3	0.2	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.1	0	0	0	0	0	0
0	0	0	0	0.3	0.4	0.4	0.2	0.2	0.3	0.3	0.3	0.2	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.3	0.5	0.4	0.1	0	0	0	0	0	0
0	0	0	0	0.1	0.2	0.3	0.4	0.5	0.5	0.6	0.5	0.4	0.4	0.4	0.2	0.1	0.1	0	0.1	0.2	0.2	0.1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.3	0.6	0.6	0.6	0.7	0.6	0.6	0.5	0.1	0.2	0.2	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.1	0.5	0.5	0.6	0.7	0.6	0.6	0.4	0.1	0.1	0.1	0	0.1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.1	0.3	0.6	0.6	0.6	0.6	0.4	0.1	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.2	0.5	0.6	0.6	0.6	0.4	0.2	0.1	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.1	0.4	0.6	0.6	0.6	0.4	0.2	0.1	0	0	0.1	0.2	0.1	0.1	0.1	0.1	0	0	0	0	0
0	0	0	0	0	0	0	0	0.1	0.2	0.3	0.6	0.6	0.4	0.3	0.2	0.1	0	0	0	0.2	0.1	0.1	0.1	0.1	0.2	0.2	0.1	0	0
0	0	0	0	0.1	0.2	0.2	0.2	0.4	0.2	0.3	0.6	0.6	0.4	0.4	0.4	0.2	0.1	0	0	0	0.1	0.1	0.1	0.4	0.6	0.5	0.4	0.2	0.1
0	0	0.1	0.3	0.5	0.6	0.4	0.3	0.4	0.2	0.4	0.6	0.6	0.5	0.6	0.2	0.1	0.1	0	0	0.1	0.1	0.1	0.4	0.6	0.6	0.6	0.5	0.2	0.2
0	0.1	0.3	0.6	0.6	0.6	0.5	0.3	0.4	0.4	0.5	0.6	0.6	0.6	0.6	0.3	0.1	0.1	0	0	0.1	0.1	0.1	0.5	0.7	0.6	0.6	0.5	0.3	0.2
0	0.2	0.6	0.7	0.7	0.7	0.6	0.2	0.3	0.5	0.6	0.6	0.6	0.6	0.6	0.3	0.1	0.1	0	0.1	0	0	0.2	0.4	0.6	0.6	0.7	0.5	0.3	0.2
0.1	0.4	0.7	0.7	0.7	0.7	0.6	0.4	0.3	0.6	0.6	0.6	0.6	0.6	0.5	0.3	0.2	0	0.1	0.1	0.1	0.1	. 0.4	0.5	0.5	0.6	0.7	0.5	0.2	0.2
0.1	0.6	0.7	0.7	0.7	0.7	0.6	0.5	0.3	0.4	0.7	0.7	0.6	0.5	0.5	0.4	0.1	0	0.1	0.1	0.1	0.1	0.4	0.6	0.4	0.5	0.7	0.4	0.2	0.3
0.2	0.6	0.7	0.7	0.7	0.6	0.6	0.5	0.4	0.3	0.5	0.7	0.5	0.6	0.6	0.3	0.1	0	0.1	0	0	0.3	0.4	0.5	0.5	0.4	0.5	0.3	0.2	0.3
0.3	0.6	0.6	0.7	0.7	0.6	0.6	0.5	0.5	0.4	0.3	0.5	0.5	0.6	0.6	0.2	0	0.1	0.1	0	0.1	0.4	0.5	0.4	0.4	0.2	0.2	0.3	0.4	0.4
0.5	0.7	0.7	0.6	0.7	0.7	0.6	0.6	0.6	0.5	0.4	0.3	0.6	0.6	0.6	0.2	0.1	0.3	0.2	0	0.2	0.4	0.4	0.4	0.4	0.1	0	0.2	0.5	0.5
0.6	0.7	0.7	0.6	0.6	0.6	0.5	0.6	0.7	0.5	0.3	0.3	0.4	0.6	0.6	0.4	0.3	0.4	0.3	0.2	0.2	0.1	0.1	0.3	0.3	0	0	0.1	0.4	0.4
0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.7	0.7	0.6	0.2	0.3	0.3	0.5	0.5	0.5	0.5	0.6	0.5	0.2	0.2	0.1	0.1	0.1	0.1	0	0	0	0.1	0.2
0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.3	0.3	0.4	0.5	0.5	0.5	0.5	0.6	0.5	0.3	0.3	0.3	0.2	0.1	0.1	0	0	0	0	0.1
0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.5	0.4	0.5	0.4	0.3	0.2	0.5	0.6	0.5	0.3	0.3	0.3	0.3	0.2	0.1	0.1	0	0	0	0
0.6	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.6	0.6	0.4	0.2	0.1	0.3	0.4	0.4	0.3	0.3	0.2	0.2	0.2	0.1	0.1	0	0	0	0
0.5	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.6	0.6	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.5	0.5	0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.7	0.6	0.6	0.3	0.1	0	0	0	0	0	0.2	0.2	0.2	0.2	0.1	0	0	0	0.1	0.1

Matrix of numbers [0, 1] (900 values)

AUSTRIA

Source:

https://www.youtube.com/watch?v=iaSUYvmCekl

Images detection & CNNs

TASK IN COMPUTER VISION



Input image

0 0.1 0.1 0.1 0.1 0.2 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0101050706060503 0103060606050304040506060606030101 020607 06070503 0.5 0.5 0.6 0.7 0.5 0.2 10104060405070402 30606070 0.5 0.5 0.5 0.6 0.6 0.6 0.6 0.7 0.7 0.7 0.7 0.6 0.6 0.3 0.1 0 0 0 0 0 0.2 0.2 0.2 0.2 0.1 0 0 0 0.1

Matrix of numbers

[0.8] Harrison Ford Sean Connery [0.1] Classificatio Roger Moore [0.05] Tom Cruise [0.05]

n

Prediction



WHY NOT SIMPLY USE A DEEP NETWORK WITH FULLY CONNECTED LAYERS?

Indie resized to 100 x 100



The picture has 10.000 pixels

With a 1.000 neuron input layer ...

and a **fully connected** 1st hidden layer, ...

this first operation amounts to a total of **10 million** connections (weights, parameters).

And that is just the 1st layer!

For **large images**, a deep neural network breaks down.

AND we do not capture the spatial information of the pixels



EURO



THE CONVOLUTION LAYER – USING SPATIAL STRUCTURE

000000000000000000 00000000000000

> Input image 1 neuron for 1 pixel

00000000000 0000000000

Hidden layer 1 neuron for all pixels in **receptive field** (filter) Idea: connect smaller sections of the input image to respective neuron in the hidden layer.

Receptive field (FILTER) marks a specific area in the input image.

Use a **sliding window** to define the connections.

The GOAL: How to **weight** the **FILTER** to detect particular features in the image?



CONVOLUTION

> Input image 1 neuron for 1 pixel

> Hidden layer 1 neuron for all pixels **in receptive field** (filter)

- Apply a set of weights a filter to extract local features
- Use multiple filters to extract different features
- Spatially share parameters of each filter



Convolution

ELEMENT-WISE MULTIPLY AND ADD THE OUTPUTS



Part of input

image



Filter

19

(1 * 1) + (3 * 2) + (5 * 2) + (2 * 1) = 19

EURO AUSTRIA

Filters

APPLICATION OF A FILTER

Convolutional layer: Connection between neurons and only those pixels within their **receptive field**.

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





4	3	2
2	4	3
2	3	4

Feature map (1 * 1) + (1 * 0) + (1 * 1) + (0 * 0) + (1 * 1) +(1 * 0) + (0 * 1) + (0 * 0) + (1 * 1) = 4

(1 * 1) + (1 * 0) + (0 * 1) + (1 * 0) + (1 * 1) +(1 * 0) + (0 * 1) + (1 * 0) + (1 * 1) = 3



APPLYING DIFFERENT FILTERS

Horizontal edge detection





Vertical edge detection





Mixed edge detection







Filters

PADDING

Adding additional space to preserve the same height and width of previous layer.

 \otimes

Zero padding

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

Image

	0	1	
)	1	0	
	0	1	
	Filter		

2	2	3	1	1
1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
0	2	2	1	1

Feature map Also, different kinds of paddings possible (i.e., One padding). Highlight pixels at the edges of image.



USING A LARGER STRIDE

The shift from one receptive field to the next one is called **stride**. **Stride = 2**

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

Image





Feature map Connect larger input to smaller layer.

Reduction of the model's computational complexity.



STACKING MULTIPLE FEATURE MAPS

Applying different filters ...



... results in different feature maps.

- Convolutional layers with multiple feature maps
- Representation in 3D



• • •





Filters

HOW DOES A CNN LEARN THE FILTERS?





How are all these filters determined or learned?

- NUMBER AND SIZE OF FILTERS IN EACH CONVOLUTION LAYER ARE SET BY DESIGN
- FILTERS INITIALIZED AT RANDOM AND THEN LEARNED (FWD PASS, BACKWARD PROP)
- CONVOLUTIONAL LAYER LEARNS MOST USEFUL FILTERS AUTOMATICALLY
 DURING TRAINING FOR ITS TASK
- LAYERS AFTER THIS WILL LEARN TO **COMBINE** THEM TO MORE **COMPLEX PATTERNS**



CAN YOU SPOT THE DIFFERENCE?

Image subsampled by **POOLING**



Full sized image





Pooling

POOLING LAYERS

Subsample (i.e., **shrink**) the input image to reduce the computational load (memory usage, number of parameters)

	Filte	er = 3	3x3	
1	1	2	0	4
0	1	7	1	0
0	8	1	1	1
9	0	1	3	5
0	4	1	0	0

Strida - C

Image

Max pooling

8	7
9	5

Feature map Set filter size, stride and padding as for convolution layer.

No weights attached.

The layer aggregates input with an aggregation function (i.e., max or mean).



ACTIVATION BY NON-LINEARITY

APPLY AFTER EVERY CONVOLUTION OPERATION

• RECTIFIED LINEAR UNIT (RELU)

• f(x) = MAX(0, x)

 PIXEL-BY-PIXEL OPERATION THAT REPLACES ALL NEGATIVE VALUES BY **ZERO**



Vertical edges activated



CNN ARCHITECTURE

EXAMPLE VGGNET



FROM BASIC TO DETAILED FEATURES





Mid level features

Low level

features (1st Conv Layers)



High level features (Last Conv Layers)

Remember the visual cortex?





VGGNET

- Invented by Simonyan and Zisserman from Visual Geometry Group (VGG) at University of Oxford in 2014^[1]
- LARGE SCALE VISUAL RECOGNITION
- Fixed filter size of 3×3 and the stride of 1
- DIFFERENT VERSIONS (VGG16, VGG19, ETC.)
- WHY? REDUCE THE # OF PARAMETERS IN THE CONV LAYERS AND IMPROVE ON TRAINING TIME.

^[1] K. S. a. A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in International Conference on Learning Representations (ICLR), San Diego, 2015.



VGGNET 16

Each CNN layer learns filters of increasing complexity.



Higher representations (recognize full objects, in different shapes and positions)

EURC

AUSTRIA

A CLOSER LOOK ON FINAL PREDICTION

