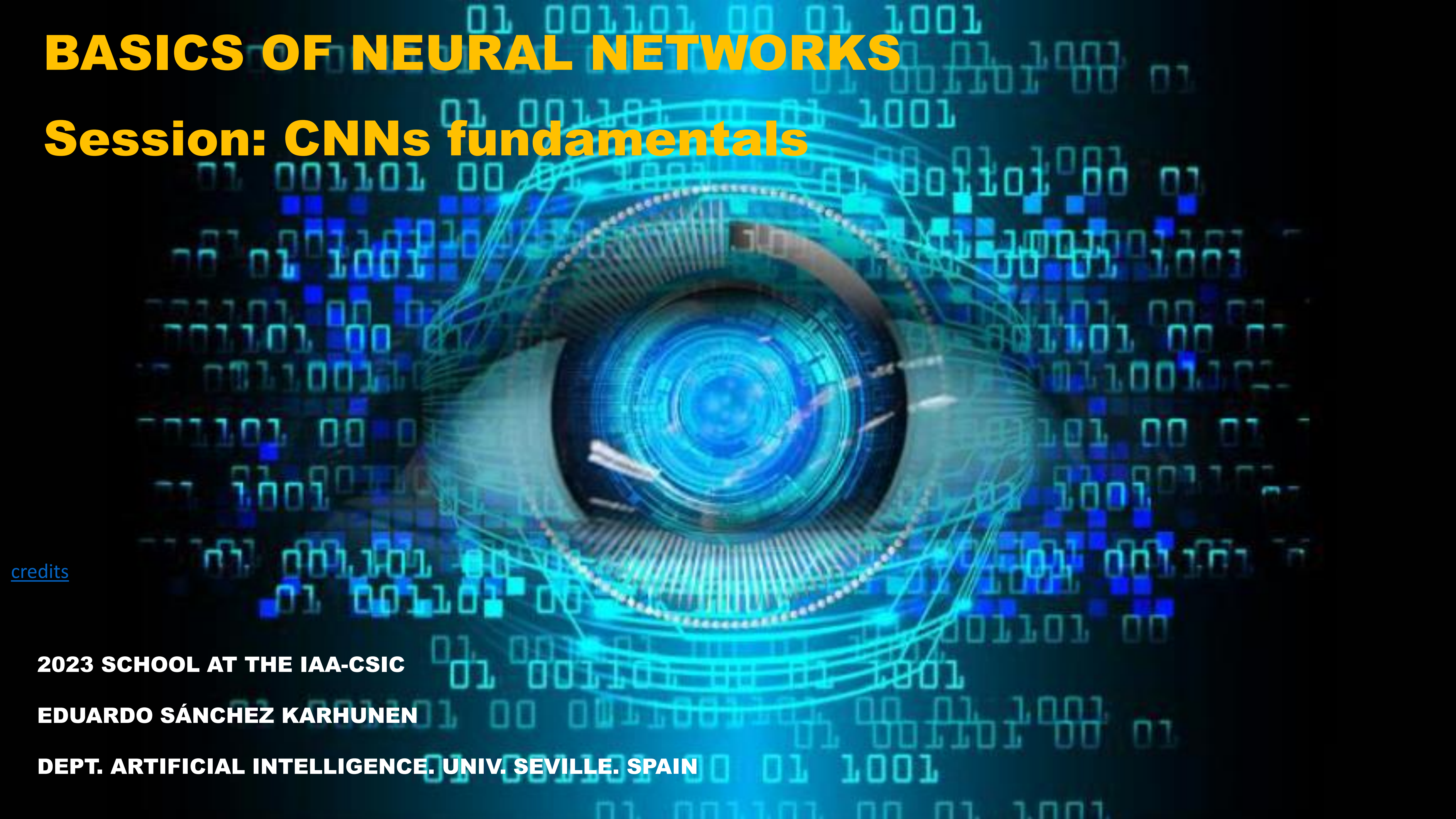


BASICS OF NEURAL NETWORKS

Session: CNNs fundamentals

credits

2023 SCHOOL AT THE IAA-CSIC

EDUARDO SÁNCHEZ KARHUNEN

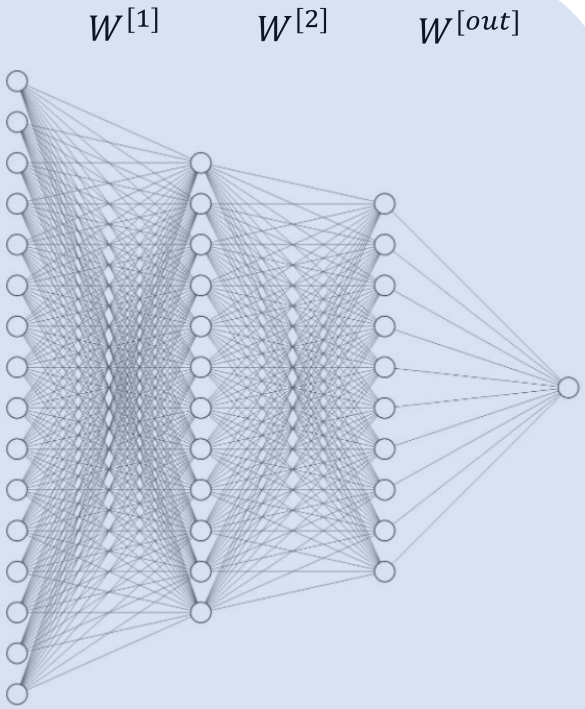
DEPT. ARTIFICIAL INTELLIGENCE. UNIV. SEVILLE. SPAIN

MLPs FOR IMAGES

Its power source becomes a problem:

- MLPs works properly only for very small images (e. g. MNIST, Fashion-MNIST).
  - **Drawbacks:**
    - Only work in 1D: image flattening is needed.
    - Dense layers: are fully connected.

- #parameters number explosion: unfeasible its use
  - Currently, easily image resolution > 12 Mpixels (e. g. $12 \cdot 1024 \ \times 1024$)
  - Input layer: $n_{input} \approx 12 \cdot 10^6$ neurons.
  - hidden_1: $n_1 = 10^6$ neurons.
  - $\#\mathbf{W}^{[1]} \approx 12 \cdot 10^6 \cdot 10^6 \approx 10^{13}$



$n_{input} \approx 12 \cdot 10^6$ $\quad n_1 = 10^6$ $\quad \ldots$

$\# \text{param} = n_{input} \cdot n1 + n_1 \cdot n_2 + \ldots \approx 10^{13}$

credits

REFOCUSING IS NEEDED FOR IMAGE PROBLEMS

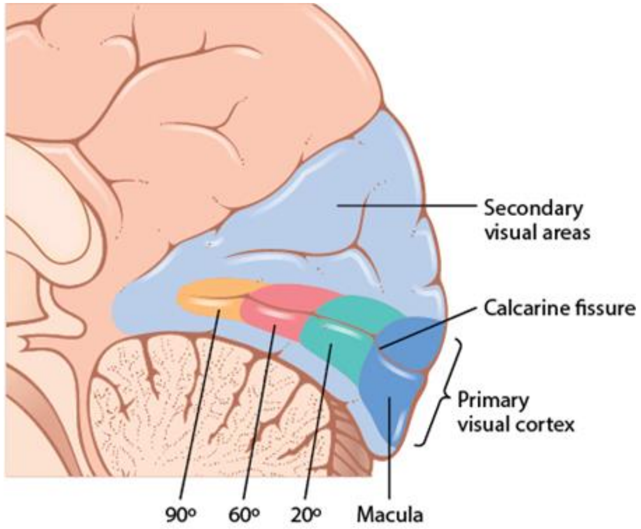
Inspiration: brain structures

a) Perception is processed in brain zones non-related to conscious activity.
   - We “sense” unconsciously.
   - We are not able to express it using rules. e. g. find a cat in an image.

b) There are specialized areas in the brain for each sense:
   - Vision: visual cortex. Hearing: auditive cortex.

c) Successive brain areas form a preprocessing pipeline:
   - Before reaching brain areas of conscious activity.

d) There are brain zones that combine information from different senses.



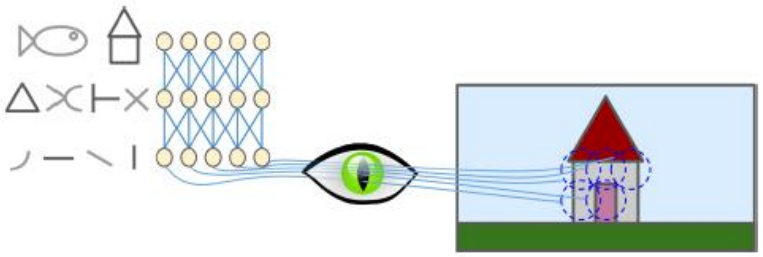
credits

Image processing in brain:

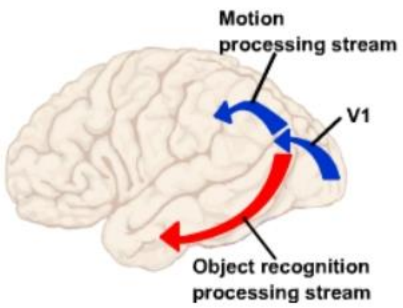
- Hubel & Torsten. Nobel Medicine Prize in 1981 for research on visual cortex.
  a) Idea of local receptive field:
  - Mapping retina -> cortex: small neuron groups react to stimulus in concrete regions of the visual field.
  - Overlapping groups cover the whole visual field.

  b) Groups of neurons reacts to simple patterns.
  - E. g. V, H orientations.

  c) Other groups of neurons react to more complex patterns:
  - Combination of simpler patterns.



credits



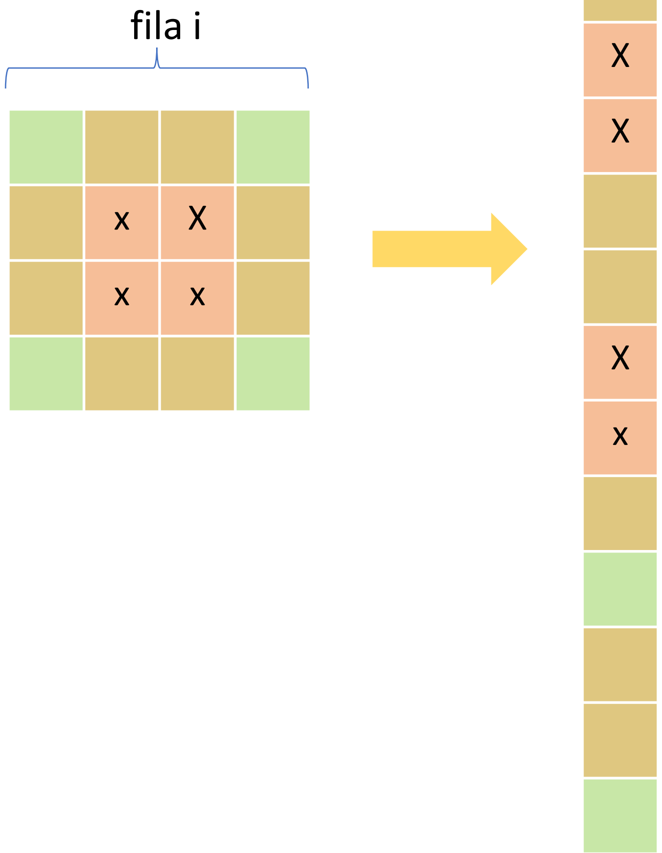
credits

APPLY CORTEX IDEAS TO NEURAL NETWORKS

- Constraints on architectures (LeCun, NY, 1998):
  - Idea:
    - At our disposal, a priori knowledge of the problem.
    - Constraints on model architecture, based on this knowledge.
  - Impact:
    - Improvement of model generalization capacity.
    - A significant reduction of #parameters.

Directly applicable in images:

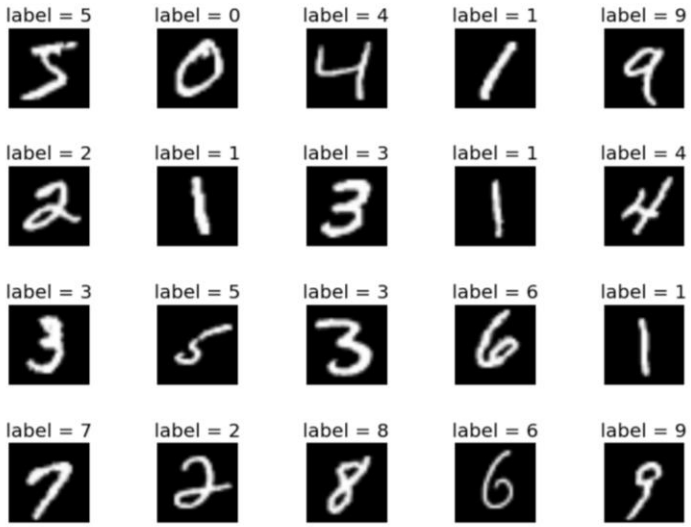
- Flattening -> loss of neighbourhood info.
- It makes sense to define a grid.
- Good idea to extract local features and combine them.



NOVEL ARCHITECTURE = CNNs

LeNet-5 (LeCun, 1989-1998):

- Applied the novel backpropagation technique to images problems.
- Selected problem: handwritten digits recognition.
- Released a reference dataset: MNIST.
- Foundation of current CNNs.



credits

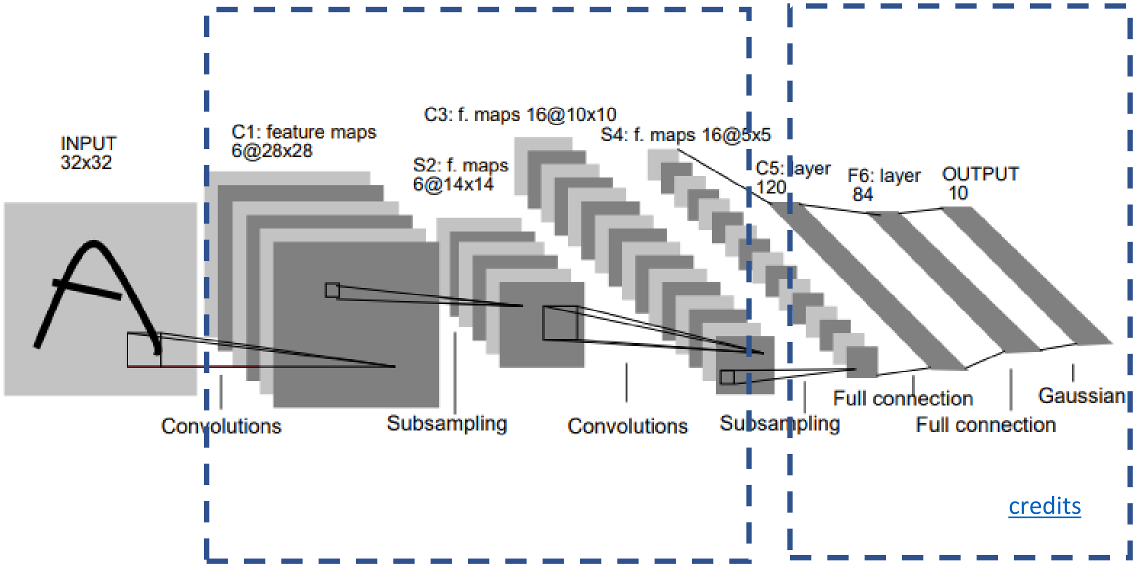
A model ahead of its time:

- It was not popular at its time: due to the lack of computation HW.
- Explosion of interest in CNNs: (when GPUs appeared)
  - In AlexNet (2012): a slight variation of LeNet trained on GPU
  - Won the ImageNet: image recognition challenge.
  - Since then, CNNs became the de facto tool for image classification.

CNN BASIC BLOCKS

Constraints on the architecture:

- CNN has two blocks:
  - A final “classical” MLP predicting the class.
  - A new “preprocessing block” before the MLP



credits

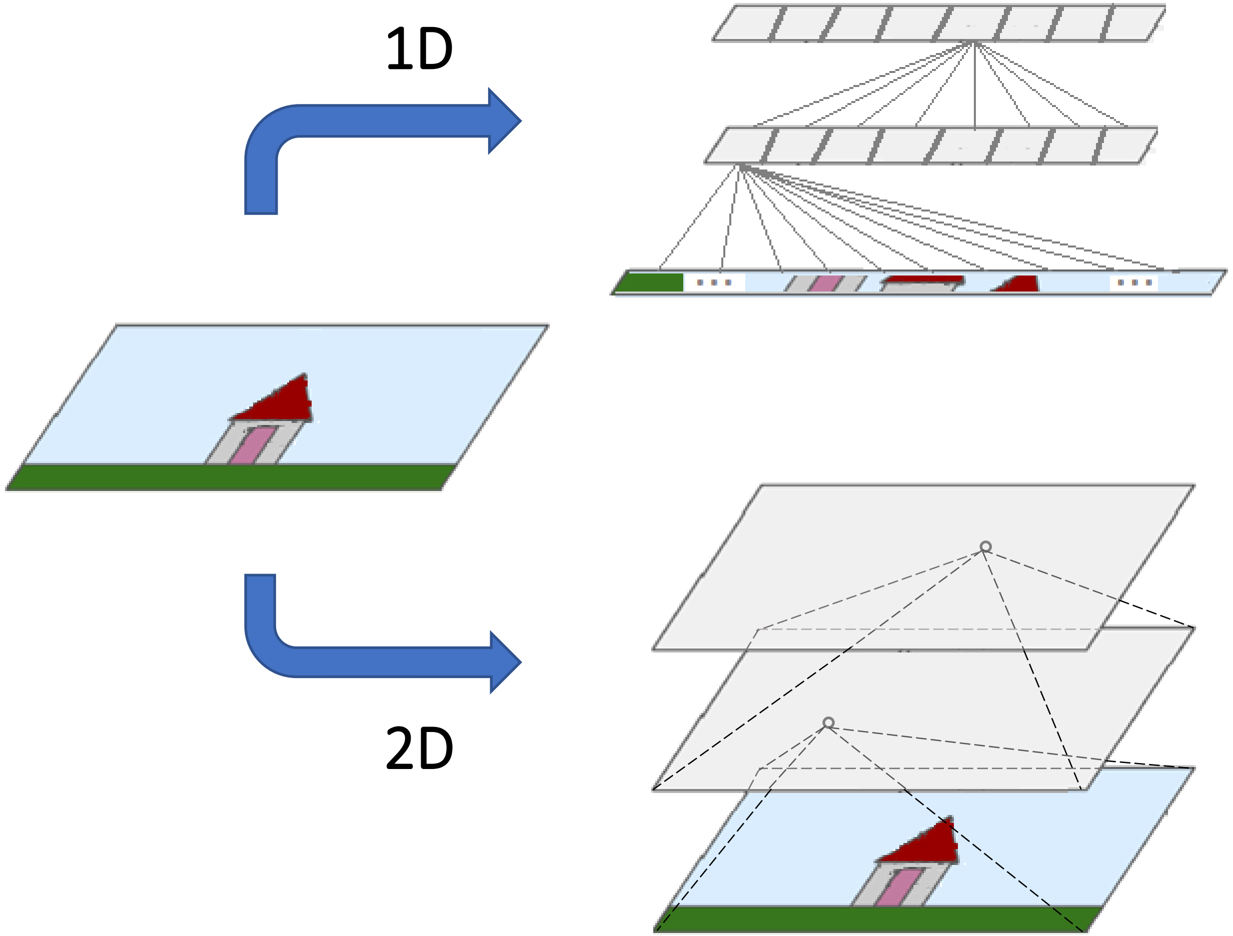
Preprocessing

MLP

Why this preprocessing block:

- MLP still needs a flattened 1D-array.
- Idea: inject in this 1D-array as much info as possible.
- Creating new layers that perform operations in 2D:
  - Convolutional layer: extracts image features.
  - Pooling layer: subsamples image.

PROBLEMS: FROM 1D TO 2D



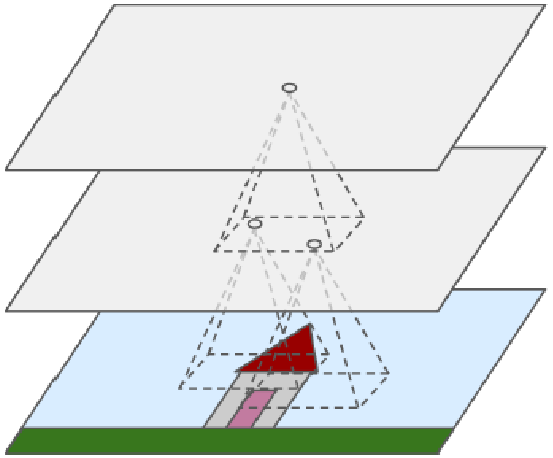
Problems:

- In dense layers:
  - Interconnection: each neuron one layer
  - With all neurons of the previous layer.
- Impact:
  - Huge number of weights (parameters)
- Extension to 2D dense layers:
  - An even bigger number of weights.
  - An alternative approach is needed.

CONVOLUTIONAL LAYER: INTUITION

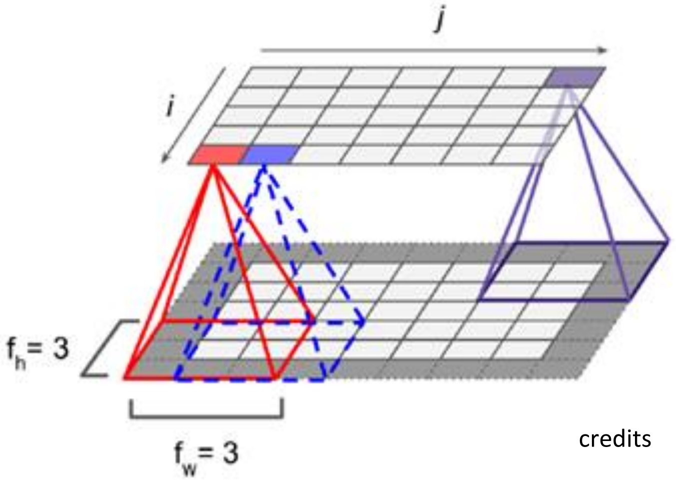
Emulating local receptive fields:

- Goal: avoid losing pixel neighbourhood info.
- Perform operations in 2D: no flattening.
- Each neuron of the layer:
  - No more connected to all neurons of the previous layer.
  - Is only connected to a rectangle (f_W, f_H): pixel local info es captured.



Intuitive interpretation of layer operation:

- All pixels (i, j) of the convolutional layer are traversed.
- An operation is applied on a rectangle of pixels of the previous layer.



credits

CONVOLUTIONAL LAYER: OPERATION

Renaming a dense 2D-layer operation: convolution

- Input image= matrix of values in range 0-255 (grayscale)
- Kernel (or filter): 3x3, 5x5, 7x7. *Dim(Kernel) << Dim(Imagen)*
  - Classical dense layer (but 2D) and with all weights zero except for the concrete rectangle of the kernel
- Convolution: $H = I \otimes K,$ element-wise product of both matrices and a final sum of all its elements.
  - Output layer: Feature Map.

2	9	11	22	42
12	43	2	0	65
9	87	65	2	90
21	21	44	7	2
45	2	21	2	0

Image: convolutional layer input
5x5

$\otimes$

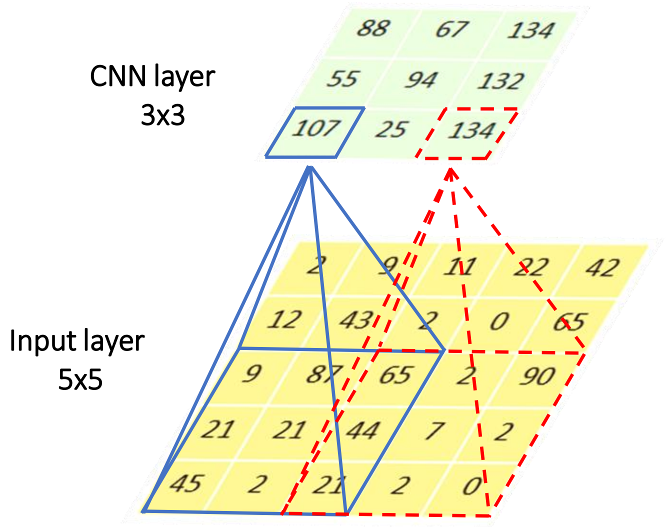
0	0	1
1	0	0
0	0	1

Kernel
3x3

=

88	67	134
55	94	132
107	25	134

Feature map
convolutional layer output
3x3



CONVOLUTION BY HAND

2_0	9_0	11_1	22	42
12_1	43_0	2_0	0	65
9_0	87_0	65_1	2	90
21	21	44	7	2
45	2	21	2	0

=

convolution

88		

2	9	11	22	42
12_0	43_0	2_1	0	65
9_1	87_0	65_0	2	90
21_0	21_0	44_1	7	2
45	2	21	2	0

=

convolution

88		
55		

Kernel

0	0	1
1	0	0
0	0	1

2	9	11	22	42
12	43	2	0	65
9_0	87_0	65_1	2	90
21_1	21_0	44_0	7	2
45_0	2_0	21_1	2	0

=

convolution

88		
55		
107		

...

2	9_0	11_0	22_1	42
12	43_1	2_0	0_0	65
9	87_0	65_0	2_1	90
21	21	44	7	2
45	2	21	2	0

=

convolution

88	67	
55		
107		

...

KERNEL UTILITY: FEATURE EXTRACTION

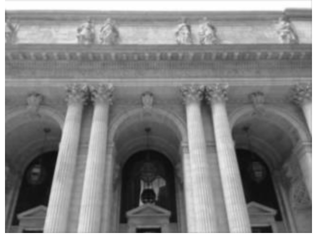
⊗

Identity kernel

0	0	0
0	1	0
0	0	0

=

Feature map 1



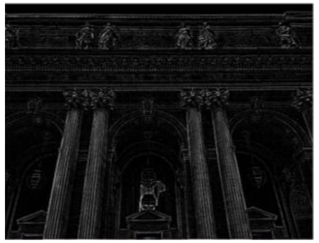
⊗

Outline kernel

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

=

Feature map 2



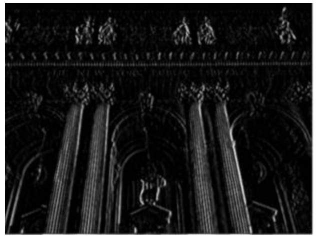
⊗

Sobel left

1	0	-1
2	0	-2
1	0	-1

=

Feature map 3



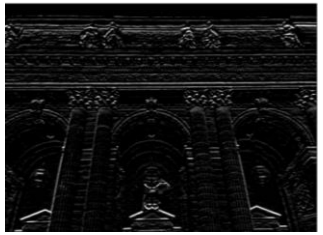
⊗

Sobel bottom

-1	-2	-1
0	0	0
1	2	1

=

Feature map 4



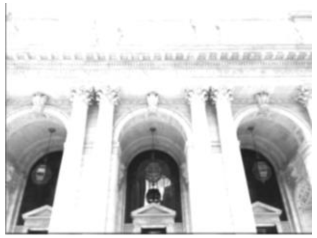
⊗

Lighten kernel

0	0	0
0	2	0
0	0	0

=

Feature map 5



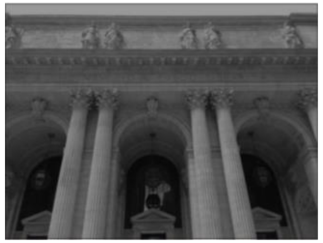
⊗

Darken kernel

0	0	0
0	0,5	0
0	0	0

=

Feature map 6

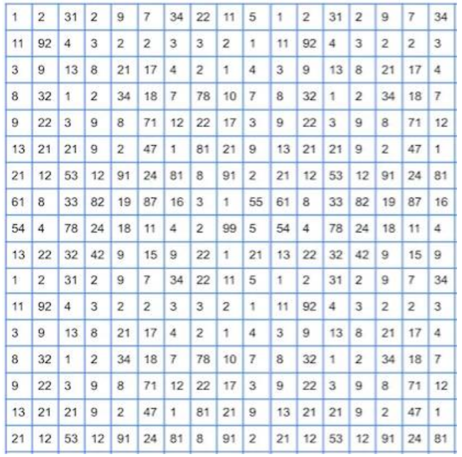


credits

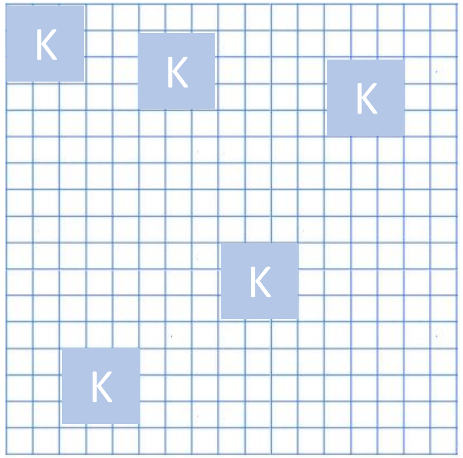
https://setosa.io/ev/image-kernels/

CONV LAYER = PARTICULAR DENSE LAYER

- **CNN training:** kernel weights
- **Sparse weights:**
  - Dense layers: each neuron has weights to all neurons of the prev. layer.
  - Conv layers: each neuron has only weights to a rectangle of the previous layer.
- **Parameter (weights) sharing:**
  - Dense layers: $W^{[i]}$, in general, weight values are different.
  - Convolutional: kernel weights are shared by all neurons in the layer.
- **Equivariance to translation:**
  - Kernel can detect the feature in any area of the image (under translation).
  - Non-equivariance under other transformations (e. g. rotation).



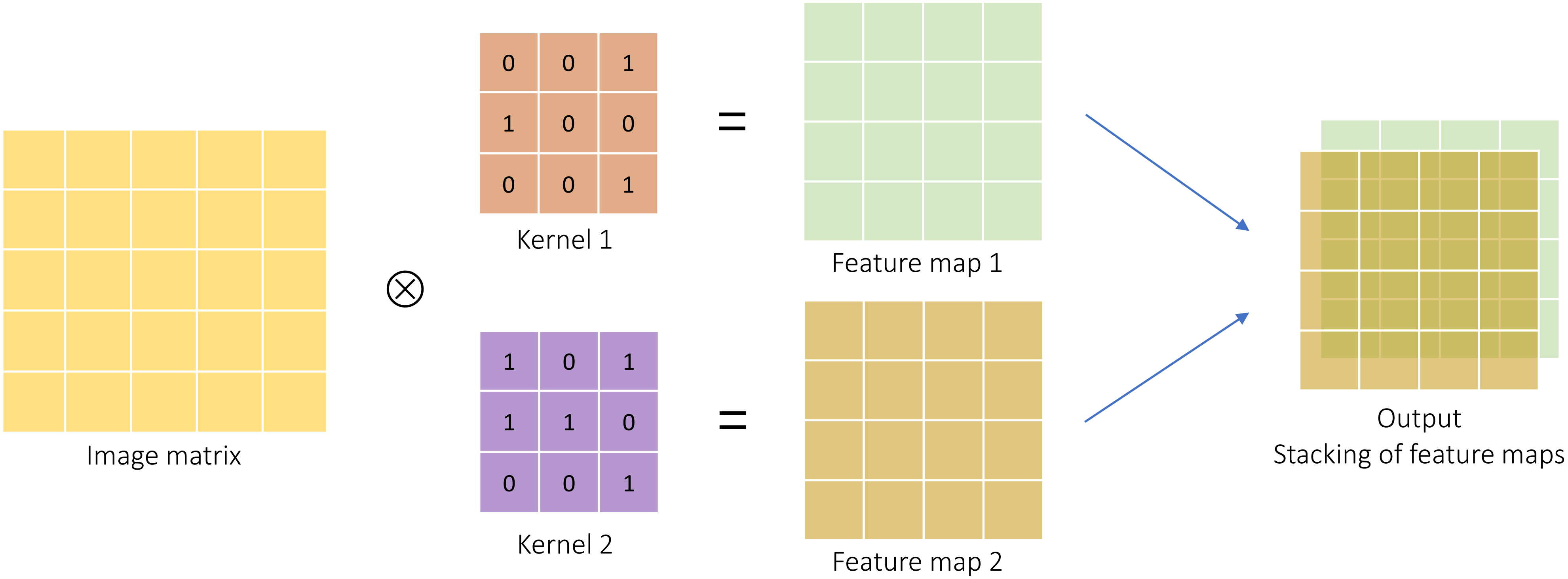
1	2	31	2	9	7	34	22	11	5	1	2	31	2	9	7	34
11	92	4	3	2	2	3	3	2	1	11	92	4	3	2	2	3
3	9	13	8	21	17	4	2	1	4	3	9	13	8	21	17	4
8	32	1	2	34	18	7	78	10	7	8	32	1	2	34	18	7
9	22	3	9	8	71	12	22	17	3	9	22	3	9	8	71	12
13	21	21	9	2	47	1	81	21	9	13	21	21	9	2	47	1
21	12	53	12	91	24	81	8	91	2	21	12	53	12	91	24	81
61	8	33	82	19	87	16	3	1	55	61	8	33	82	19	87	16
54	4	78	24	18	11	4	2	99	5	54	4	78	24	18	11	4
13	22	32	42	9	15	9	22	1	21	13	22	32	42	9	15	9
1	2	31	2	9	7	34	22	11	5	1	2	31	2	9	7	34
11	92	4	3	2	2	3	3	2	1	11	92	4	3	2	2	3
3	9	13	8	21	17	4	2	1	4	3	9	13	8	21	17	4
8	32	1	2	34	18	7	78	10	7	8	32	1	2	34	18	7
9	22	3	9	8	71	12	22	17	3	9	22	3	9	8	71	12
13	21	21	9	2	47	1	81	21	9	13	21	21	9	2	47	1
21	12	53	12	91	24	81	8	91	2	21	12	53	12	91	24	81



STACKING OF FEATURE MAPS

More than one kernel can be applied:

- Each kernel generates a feature map.
- Apply different kernels to an image, as much as needed.
- Each kernel has its own weights.



0	0	1
1	0	0
0	0	1

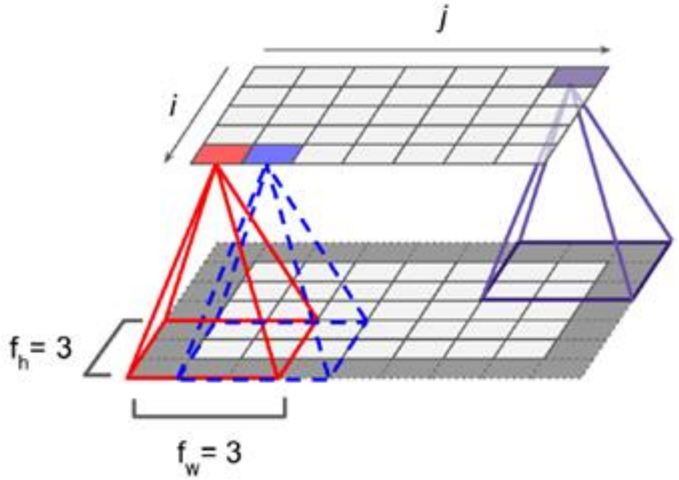
Kernel 1

1	0	1
1	1	0
0	0	1

Kernel 2

CONV LAYER PARAMETERS: PADDING

- Conv layer alters original image shape:
  - No-padding: loss of $(\boldsymbol{f_H} - \mathbf{1})$ rows and $(f_W - \mathbf{1})$ columns.
  - Zero-padding:
    - Original image margins are padded with zeros.
    - Goal: maintain original image size after conv layer application.



2_0	9_0	11_1	22	42
12_1	43_0	2_0	0	65
9_0	87_0	65_1	2	90
21	21	44	7	2
45	2	21	2	0

=

88		

3x3

5x5 + padding = No
(in TF, *valid*)

0_0	0_0	0_1	0	0	0	0
0_1	2_0	9_0	11	22	42	0
0_0	12_1	43_0	2	0	65	0
0	9	87	65	2	90	0
0	21	21	44	7	2	0
0	45	2	21	2	0	0
0	0	0	0	0	0	0

=

12				
	88			

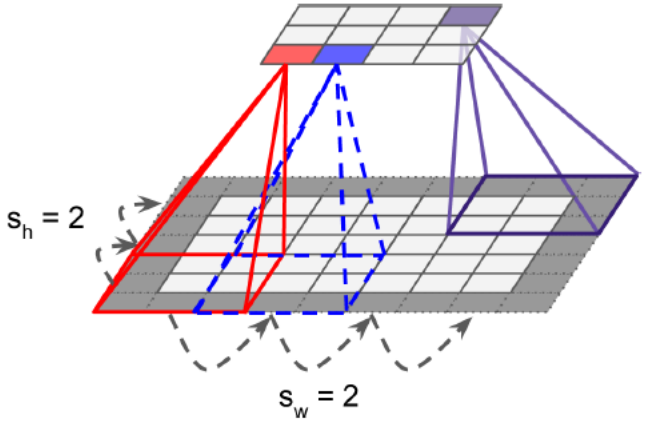
5x5

5x5 + padding = yes
(in TF, *same*)

CONV LAYER PARAMETERS: STRIDE

Kernel sliding control:

- Stride: kernel displacement while sliding along the image.
  - Typically, stride = 1.
  - If stride > 1, output size < original image size.
  - If necessary, S_V: vertical stride ≠ S_H: horizontal stride.



0_0	0_0	0_1	0	0	0	0
0_1	2_0	9_0	11	22	42	0
0_0	12_1	43_0	2	0	65	0
0	9	87	65	2	90	0
0	21	21	44	7	2	0
0	45	2	21	2	0	0
0	0	0	0	0	0	0

=

Stride = 2

12		

0	0	0	0	0	0	0
0	2	9	11	22	42	0
0_0	12_0	43_1	2	0	65	0
0_1	9_0	87_0	65	2	90	0
0_0	21_1	21_0	44	7	2	0
0	45	2	21	2	0	0
0	0	0	0	0	0	0

=

Stride = 2

12		
64		

FROM MATRICES TO TENSORS

Images representation:

- Grayscale image: 1 x 2D-array.
- Colour image: 3 x 2D-arrays (or channels).
  - 1 channel for each colour: Red (R), Green (G), Blue (B).
- Depending on the problem extra channel are possible:
  - E. g. satellite images: multiple additional infrared channels.

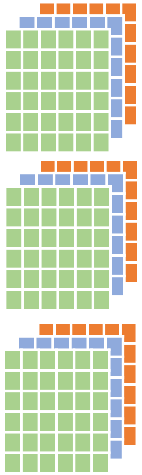
Grayscale (2D-Tensor)

2	7	1	6	0	0
9	9	2	6	1	8
1	4	9	8	7	3
0	4	2	5	0	6
6	3	0	3	3	1
7	5	1	3	7	4

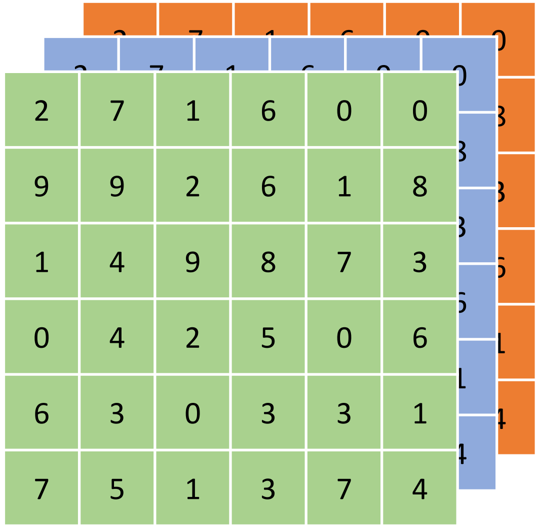
In NN frameworks:

- 3D-tensors: [*height, width, channels*]
- Mini-batch: [*batchsize, height, width, channels*]

Mini-batch



Colour-3 channels (3D-Tensor)

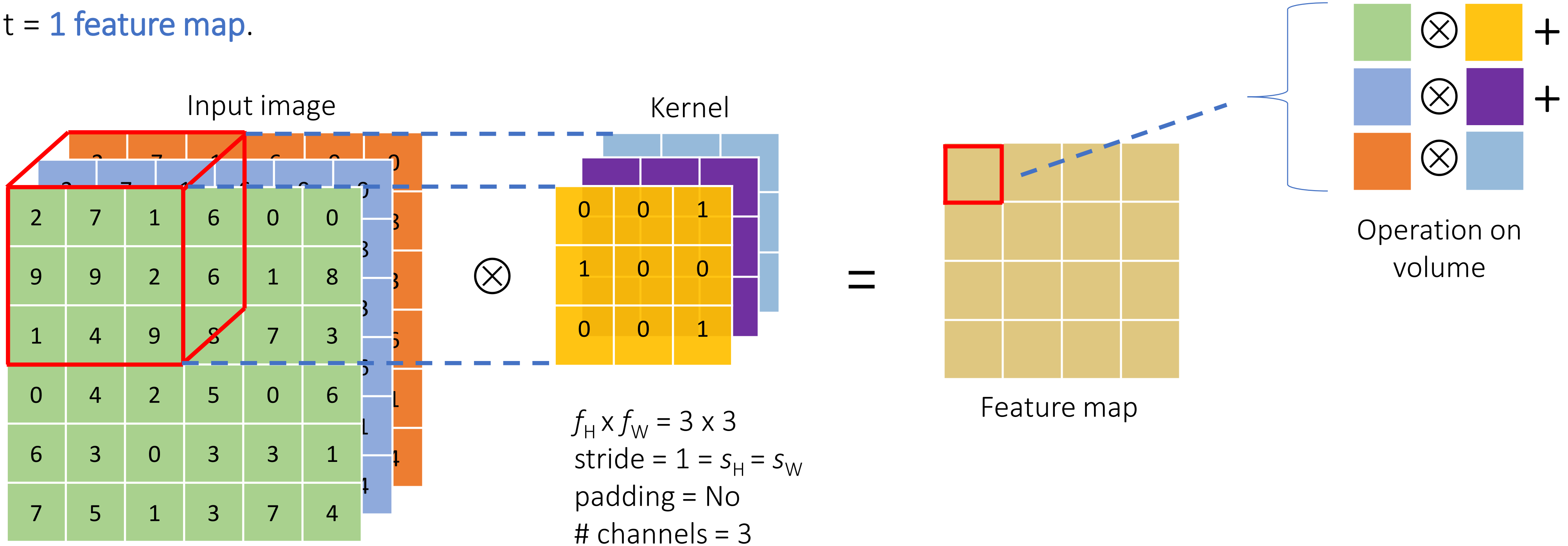


2	7	1	6	0	0
9	9	2	6	1	8
1	4	9	8	7	3
0	4	2	5	0	6
6	3	0	3	3	1
7	5	1	3	7	4

IMAGES WITH MULTIPLE CHANNELS

Convolutions on volume:

- Now kernel has multiple channels
- #channels kernel = #channels image
- Σ(each kernel "channel" acts on its associated image channel).
- Output = 1 feature map.

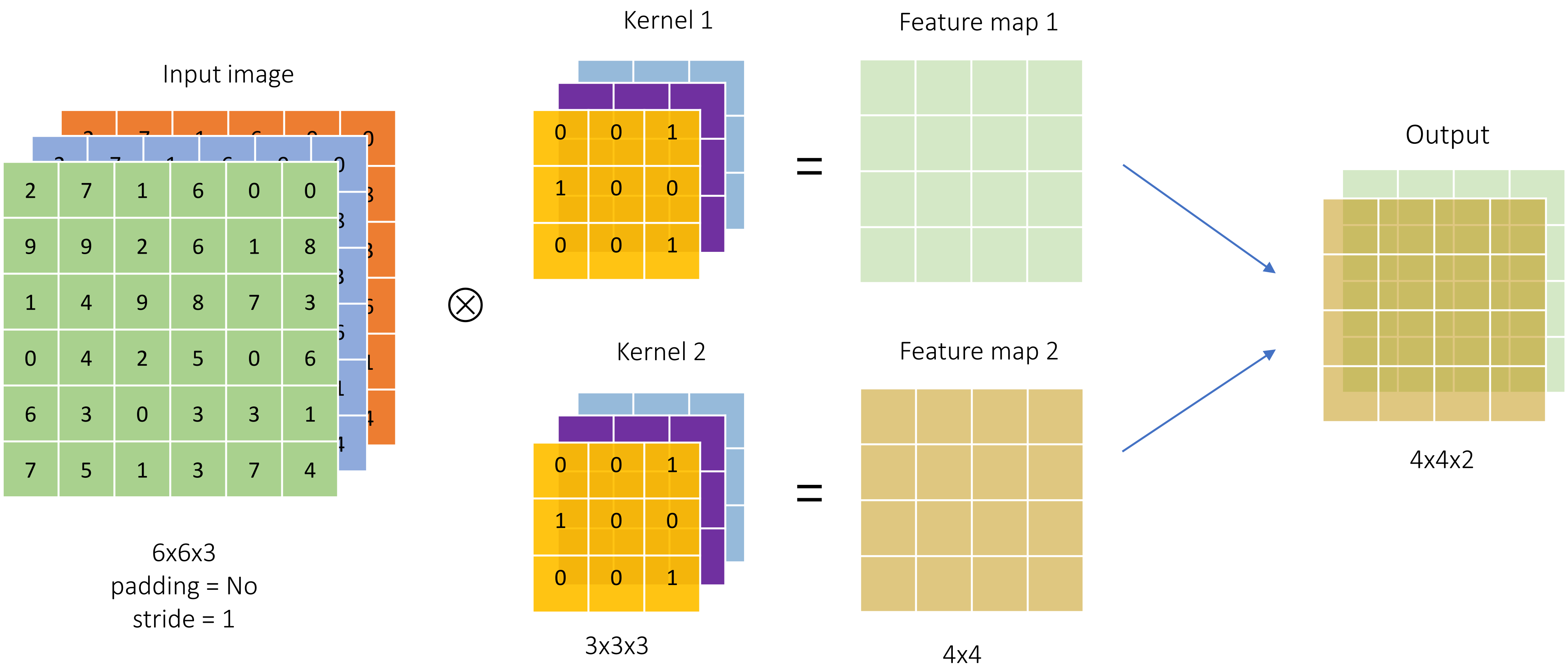


2	7	1	6	0	0
9	9	2	6	1	8
1	4	9	8	7	3
0	4	2	5	0	6
6	3	0	3	3	1
7	5	1	3	7	4

0	0	1
1	0	0
0	0	1

f_H x f_W = 3 x 3
stride = 1 = s_H = s_W
padding = No
# channels = 3

MULTIPLE CHANNELS & MULTIPLE KERNELS



Input image

2	7	1	6	0	0
9	9	2	6	1	8
1	4	9	8	7	3
0	4	2	5	0	6
6	3	0	3	3	1
7	5	1	3	7	4

6x6x3
padding = No
stride = 1

⊗

Kernel 1

0	0	1
1	0	0
0	0	1

Kernel 2

0	0	1
1	0	0
0	0	1

3x3x3

=

Feature map 1

=

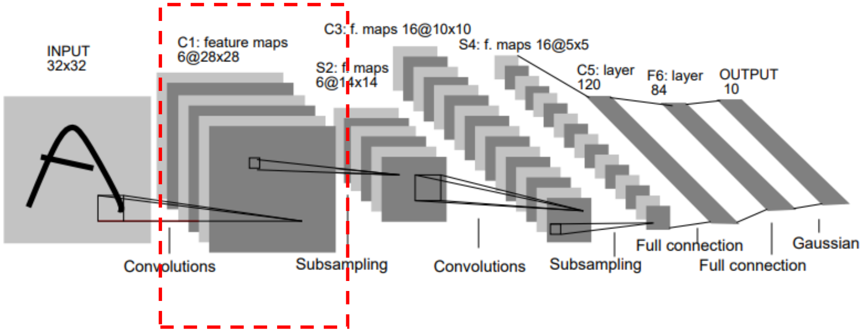
Feature map 2

4x4

Output

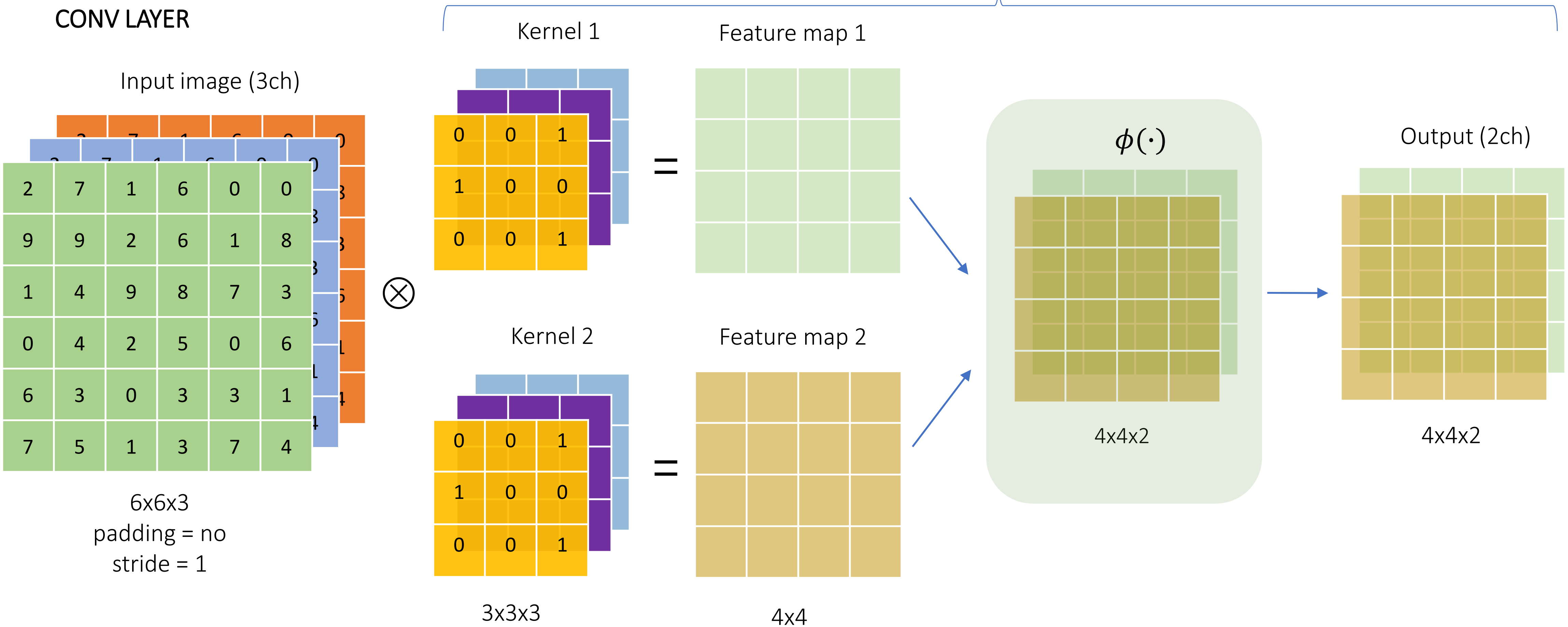
4x4x2

WHOLE CONVOLUTIONAL LAYER



CONV LAYER

Training: learn kernel weights



2	7	1	6	0	0
9	9	2	6	1	8
1	4	9	8	7	3
0	4	2	5	0	6
6	3	0	3	3	1
7	5	1	3	7	4

Kernel 1

0	0	1
1	0	0
0	0	1

Kernel 2

0	0	1
1	0	0
0	0	1

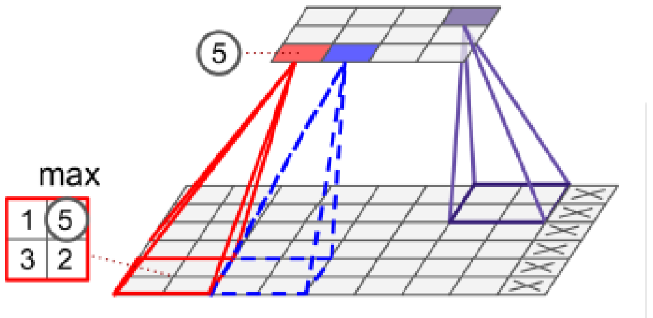
POOLING: LAYER

Subsampling:

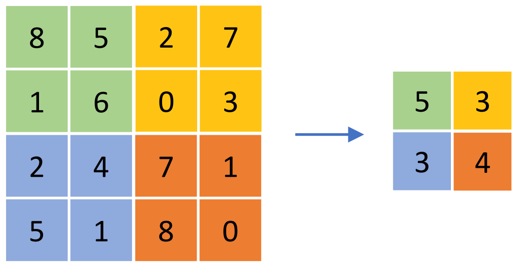
- Goal: reduce size of each feature map.
- Parallelism with convolution:
  - Pooling kernel = identity kernel. Typ. 2x2.
  - Not sum, but another aggregation operation: AVG, MAX
- Applied to each input channel separately
- # input channels = # output channels.

Layer parameters:

- No-Padding, to reduce output size.
- Stride > 1, to reduce output size. Typ. stride = 2.



Avg Pooling

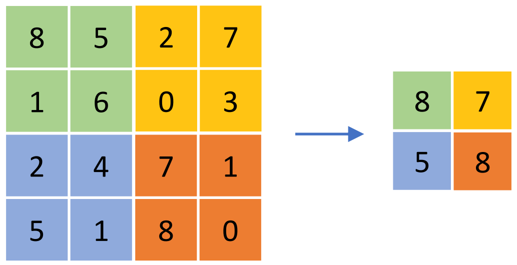


8	5	2	7
1	6	0	3
2	4	7	1
5	1	8	0

→

5	3
3	4

Max Pooling



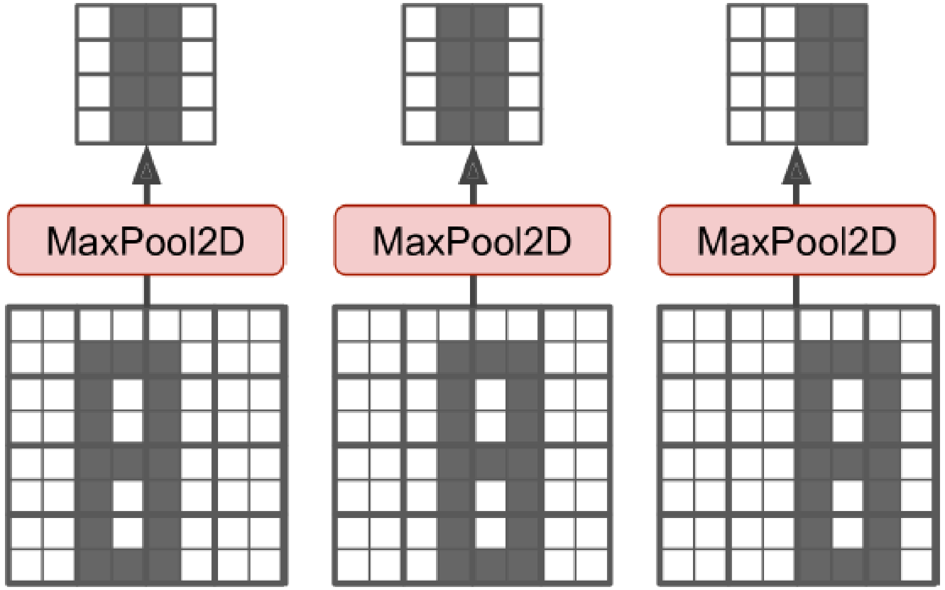
8	5	2	7
1	6	0	3
2	4	7	1
5	1	8	0

→

8	7
5	8

POOLING LAYER: INVARIANCE

- Common pooling layers:
  - Older: AvgPool2D.
    - Averaging: less info is lost.
  - Bets results: MaxPool2D.
    - Preserves stronger features.
    - Sends clearer signals to the next layer.
  - Lately: GlobalAvgPool2D.
    - Highly destructive.
    - Return a unique value. Not a feature map.
    - Useful as an output layer.

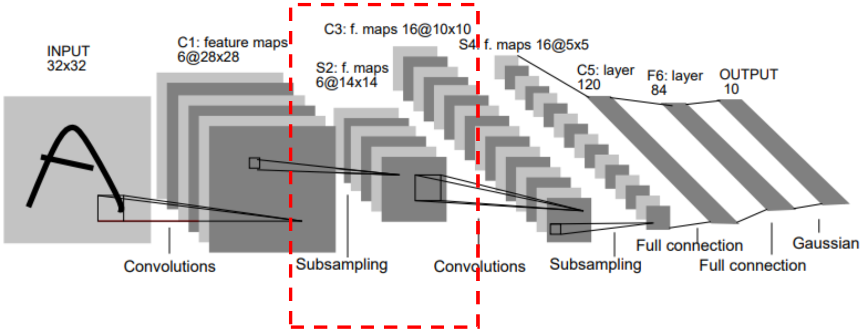


credits

Capture small invariances:

- Translations, rotations and scaling:
  1. Useful in classification problems
  2. Non-useful in segmentation problems

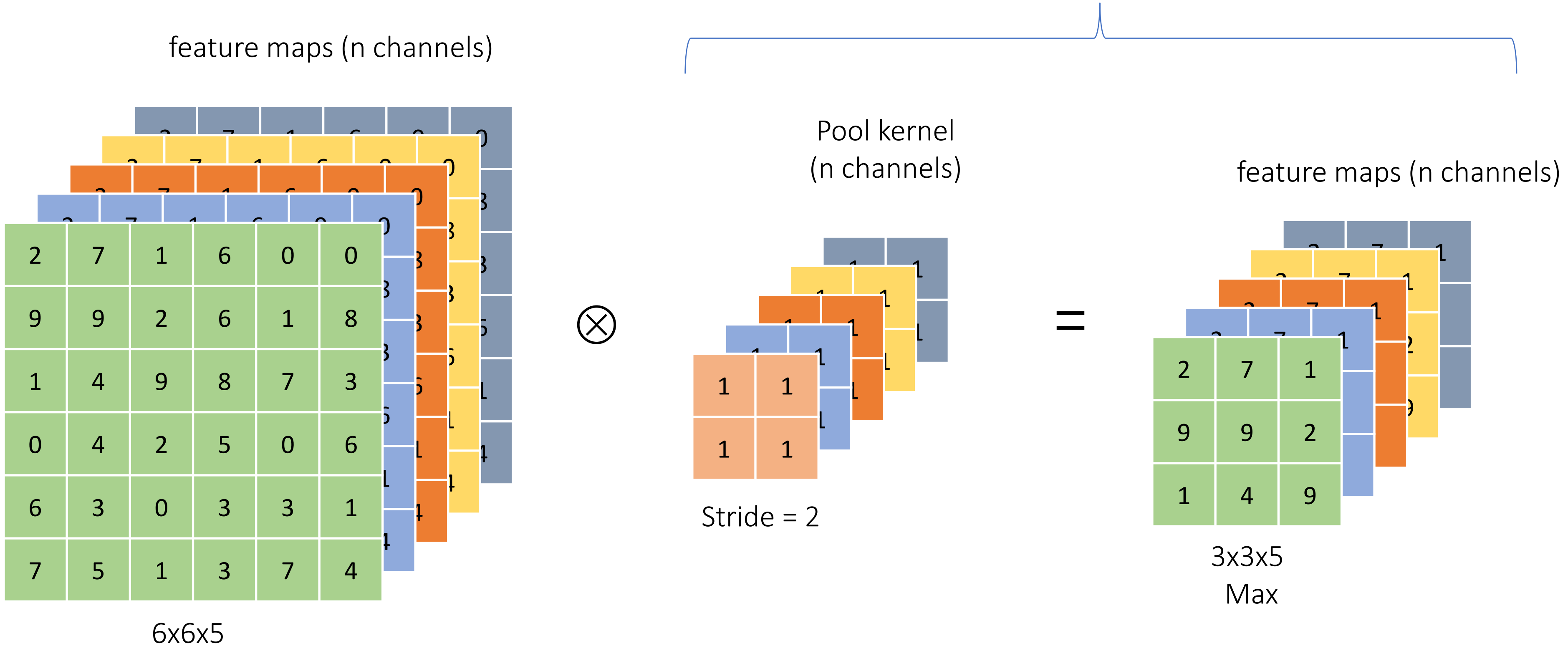
WHOLE POOLING LAYER



POOLING LAYER

Training: nothing to learn. All weights = 1

POOL LAYER: SUBSAMPLES 2:1



feature maps (n channels)

2	7	1	6	0	0
9	9	2	6	1	8
1	4	9	8	7	3
0	4	2	5	0	6
6	3	0	3	3	1
7	5	1	3	7	4

6x6x5

⊗

Pool kernel
(n channels)

1	1
1	1

Stride = 2

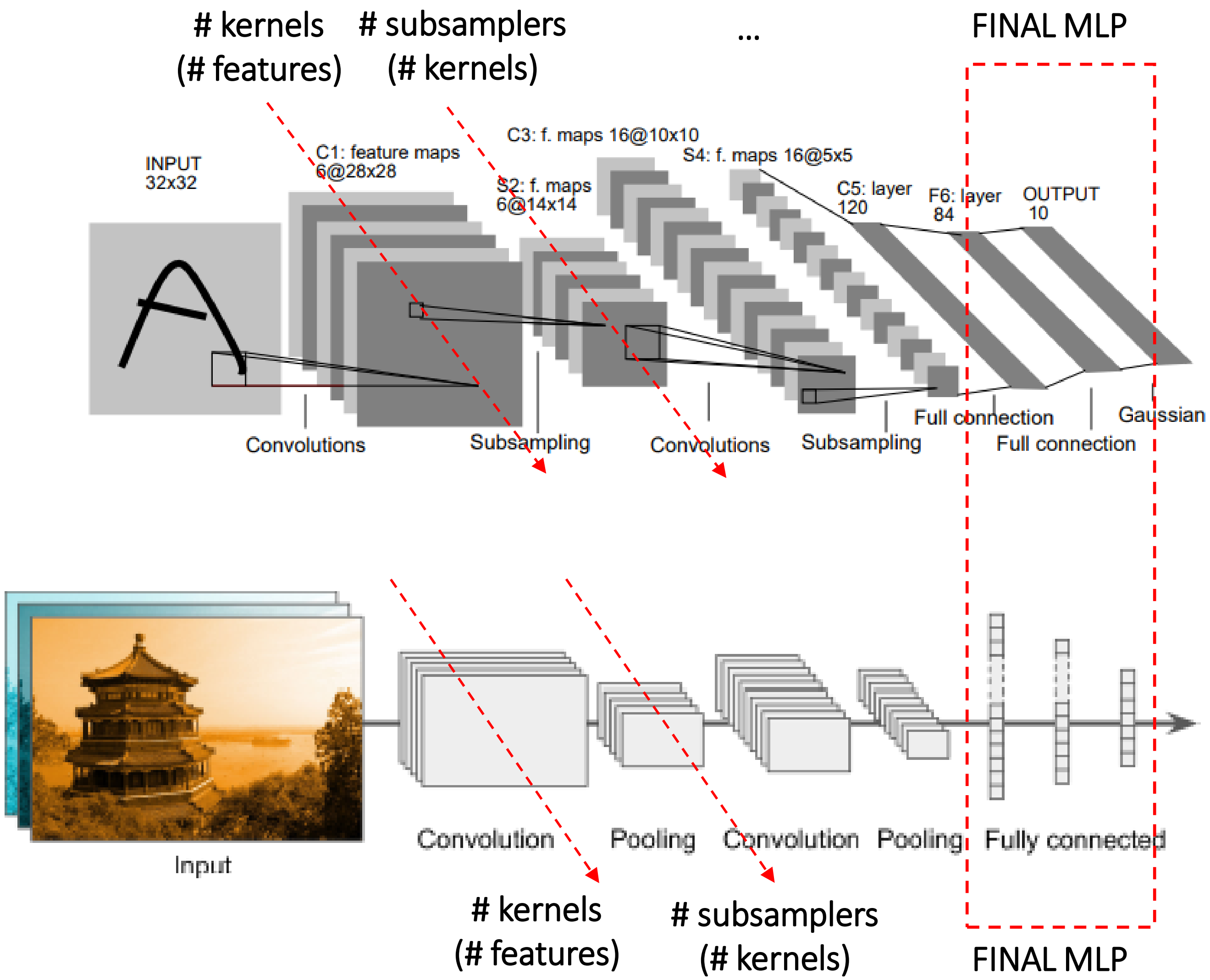
=

feature maps (n channels)

2	7	1
9	9	2
1	4	9

3x3x5
Max

TYPICAL CNN ARQUITECTURE



As image progresses:

# kernels increases

# several conv + pool blocks

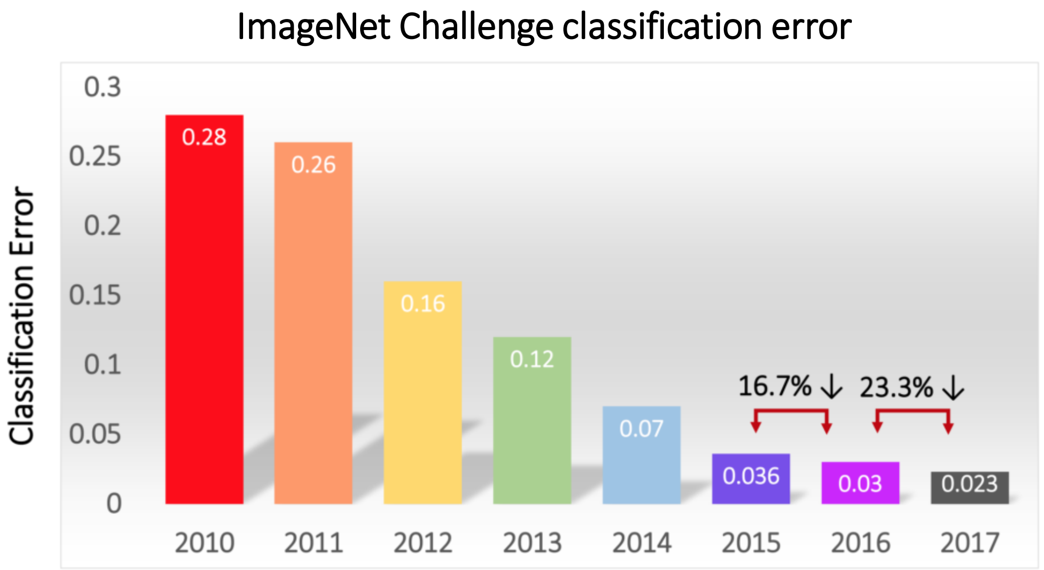
REFERENCE CHALLENGE

ImageNET challenge (2010):

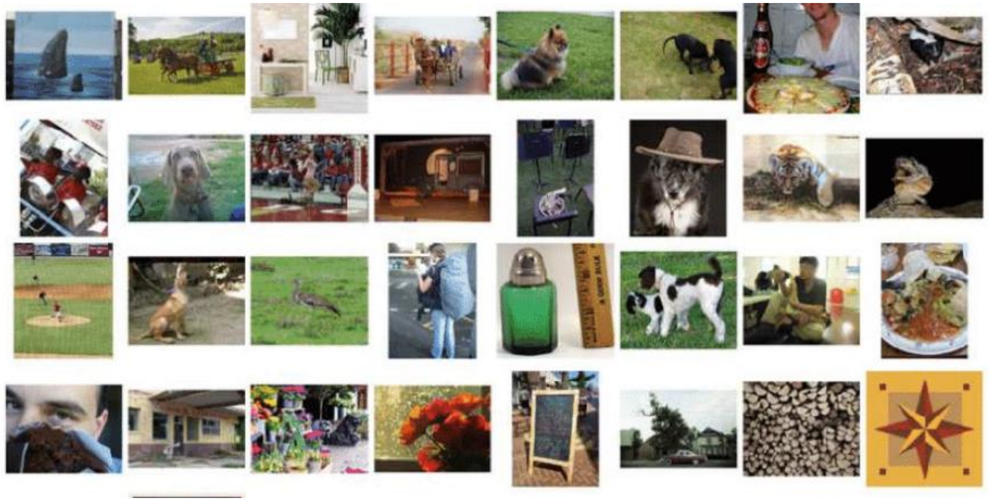
- ILSVRC (ImageNet Large Scale Visual Recognition Challenge)
- 1.2M images (up to 256 pixels).
- Classification problem.
- #clases = 10.000

Winner model: top-5 error rate

- Classifier outputs probability of each class.
- Classes with top-5 probabilities are selected.
- % times correct class is not among them.
- Also exists top-1 rate.



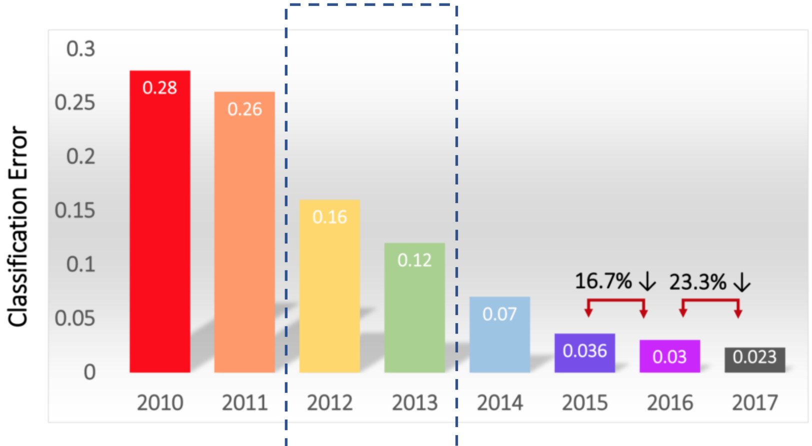
credits



FIRST HIT OF CNNs

AlexNet (2012, error 16%):

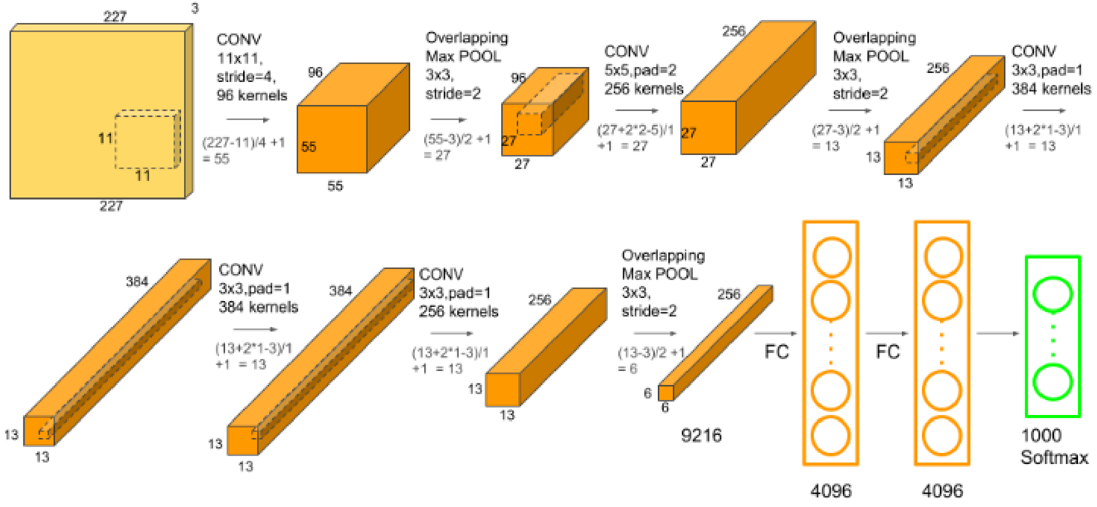
- Based on LeNet-5.
- First conv stacking without pooling.
- # layers = 5 conv + 3 dense
- # params = 60M
- Dropout 50%.
- Data augm.: brightness, shifting and flipping.



credits

ZF Net (2013, error 12%)

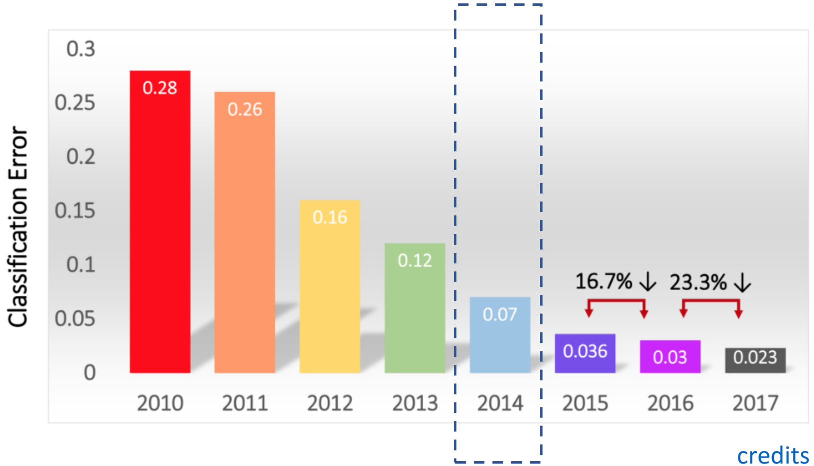
- Hyperparameters tuning:
  - #feature maps
  - kernel size, stride.



credits

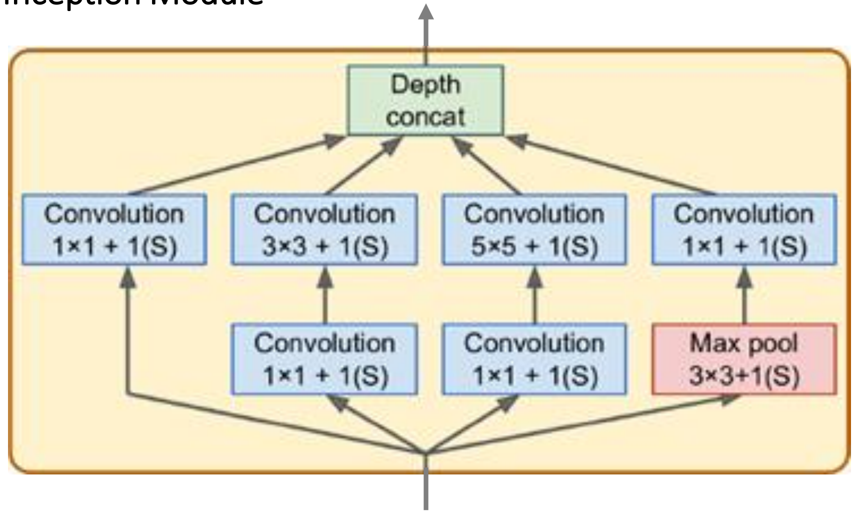
2nd SIGNIFICANT ADVANCE

- GoogLeNet (2014, error 7%):
  - Innovation: Inception module.
  - # layers = 22 (much deeper)
  - # params = 6M



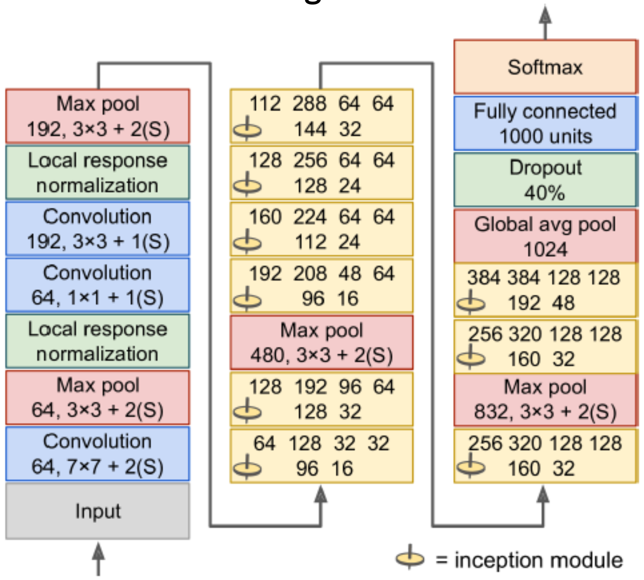
credits

Inception Module



3x3 + 1(S) => Kernel 3x3 + Stride=1 + “Same”

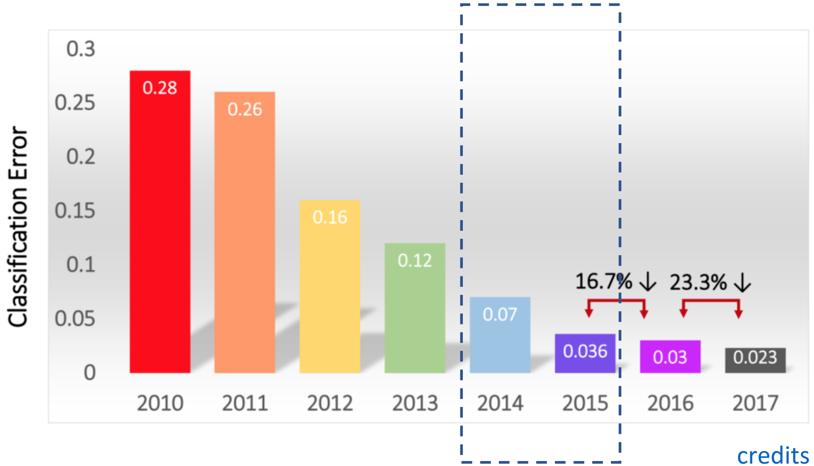
GoogleNet



2014 SILVER MEDAL: VGGNet

VGGNet (2014, error 7.3%):

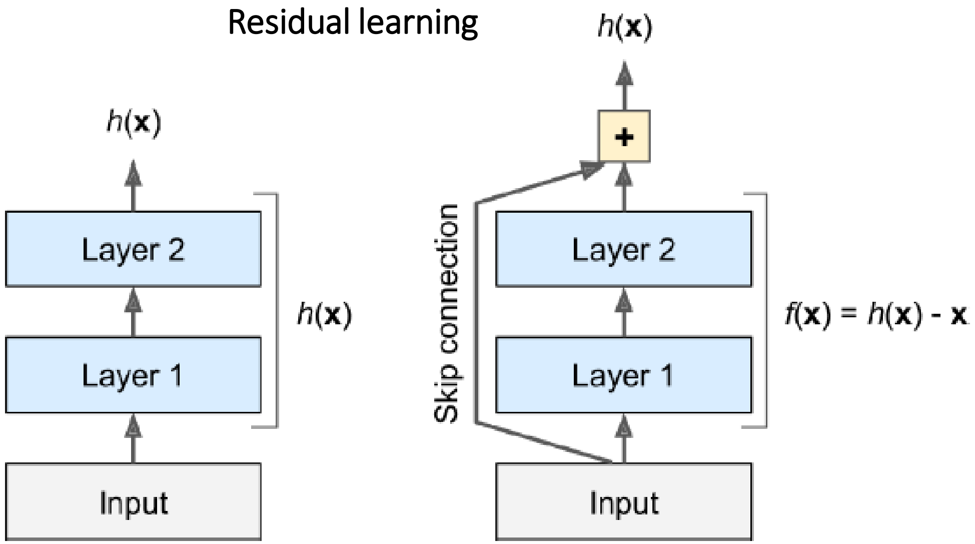
- A very simple classical.
- Stacking de 2 conv + pooling.
- # layers = 16, 19 conv + 2 dense.
- VGG16, VGG19, …
- # params = 140M.



credits

ResNet (2015, error 3.6%):

- Innovation: residual nets.
- # layers = 152.
- # params = 11M .



credits