Deep Learning

3/ Convolutional Neural Networks

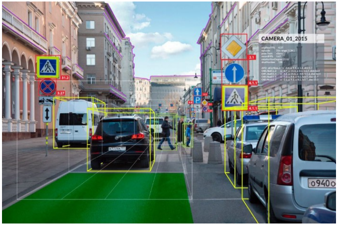
Francesca Galassi, MCF, Esir

francesca.galassi@irisa.fr
Lab Empenn Irisa-Inria

Computer Vision

Deep Learning in Computer Vision

➠ Self-driving Cars

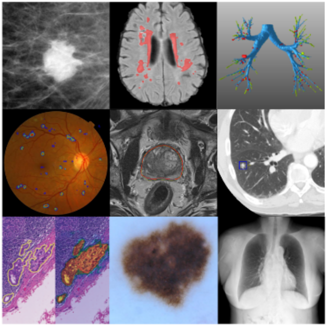


Courtesy of MIT

➠ Face Detection and Recognition

Courtesy of Streamverse

➠ Medicine, Biology, Healthcare



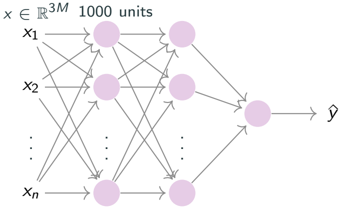
Litjens et al. (2019), *A survey on Deep Learning in Medical Image Analysis, Medical image analysis*

Challenges in Deep Learning for Computer Vision

➟ Image classification : fox ? (0 or 1)

ImageNet sample image

➟ Fully connected network :



➟ Low-res image $64 \times 64 \times 3 = 12228$ pixels

➟ High-res image $1000 \times 1000 \times 3 = 3$ million pixels

$W^{[1]} : (1000, 3M) \quad \longrightarrow \quad$ 3 billion parameters

➟ **Many parameters** to be learnt

➟ The **spatial structure** of the image **collapses**

Intuition on Deep Learning for Computer Vision

- Image processing for Object Detection :

Example :

detecting vertical edges :

detecting horizzontal edges :

- Deep learning learns features at different levels, i.e., a hierarchy of features.
  - *Low-level* : Initial layers extract basic features like edges.
  - *Mid-level* : Intermediate layers recognize more complex patterns such as facial features.
  - *High-level* : Final layers identify comprehensive patterns like entire faces.

Convolution Operation

Vertical Edge Detection

gray scale image

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

6×6

$*$

filter

1	0	-1
1	0	-1
1	0	-1

3×3

$=$

output

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

➟ Vertical edge detection using convolution operation

Vertical Edge Detection

- Vertical edge detection using convolution operation

gray scale image

3_1	0_0	1_{-1}	2	7	4
1_1	5_0	8_{-1}	9	3	1
2_1	7_0	2_{-1}	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6

$*$

filter

1	0	-1
1	0	-1
1	0	-1

3×3

$=$

output

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

Convolution operation : element-wise product followed by addition

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

- Shift the filter **one** step to the right

Vertical edge detection

➠ Vertical edge detection using convolution operation

gray scale image

3	0_1	1_0	2_{-1}	7	4
1	5_1	8_0	9_{-1}	3	1
2	7_1	2_0	5_{-1}	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6

$*$

filter

1	0	-1
1	0	-1
1	0	-1

3×3

$=$

output

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

➠ Shift the filter **one** step to the right

➠ End of line : shift the filter back to the left and one step down

Vertical Edge Detection

➟ Vertical edge detection using convolution operation

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6×6

$*$

filter

1	0	-1
1	0	-1
1	0	-1

3×3

$=$

output

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4×4

➟ Vertical edge *detector* : a 3-by-3 region with brighter pixels on the left and darker on the right

➟ A vertical edge is detected down the middle of the image

Vertical Edge Detection

➟ Various vertical edge filters :

1	0	-1
1	0	-1
1	0	-1

Sobel filter

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

Scharr filter

3	0	-3
10	0	-10
3	0	-3

➟ **What is the best set of parameters ?**

➟ **Learn the parameters using a Convolutional Neural Network**

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

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Learning to Detect Edges

➡ **Learning parameters using a Convolutional Neural Network**

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

6×6

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3×3

➡ Output matrix dimensions : $(n - f + 1) \times (n - f + 1) \quad \longrightarrow \quad 6 - 3 + 1 = 4$

$n \times n$ image size, $f \times f$ filter size

Remarks :

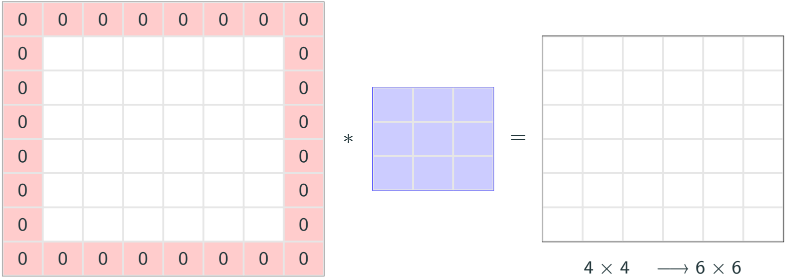
➡ **Output shrinking**

➡ **Less information from corners and edges**

Padding

Padding

- Padding the image with a zero border around the edges



$6 \times 6 \quad \longrightarrow 8 \times 8$

$4 \times 4 \quad \longrightarrow 6 \times 6$

- If the amount of padding is p, with an image of size $n \times n$ and a filter of size $f \times f$, then the output size is given by :

  $(n + 2p - f + 1) \times (n + 2p - f + 1) \quad \longrightarrow \quad p = 1, \ (6 + 2 \times 1 - 3 + 1) = 6$

- This preserves the input image size of 6×6.

Valid and Same convolutions

- *Valid* convolution : no padding

$$n \times n \quad * \quad f \times f \quad \longrightarrow \quad \boxed{(n - f + 1) \quad \times \quad (n - f + 1)}$$

$$6 \times 6 \quad * \quad 3 \times 3 \quad \longrightarrow \quad 4 \times 4$$

- *Same* convolution : padding so that output size is **the same** as the input size

$$(n + 2p - f + 1) \quad \times \quad (n + 2p - f + 1)$$

To make the output dimensions as the input :

$$\boxed{n + 2p - f + 1 = n \quad \Longrightarrow \quad p = \frac{f-1}{2}}$$

- e.g., $p = \frac{3-1}{2} = 1$
- by convention f is odd, i.e., central position, symmetric padding

Stride

Strided Convolutions

➟ Stride : the number of pixels the convolutional filter moves horizontally or vertically after each convolution operation

Input image

2_{3}	3_{4}	7_{4}	4	6	2	9
6_{1}	6_{0}	9_{2}	8	7	4	3
3_{-1}	4_{0}	8_{3}	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

7×7

$*$

filter

3	4	4
1	0	2
-1	0	3

3×3

$=$

Output image

91	100	2
0	-2	-4
-3	-2	-3

3×3

➟ Stride $= 2$

Summary

- Given :

  image $n \times n$, filter $f \times f$

  padding p, stride s

- The output image dimensions after convolution are :

$$\boxed{\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \quad \times \quad \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor}$$

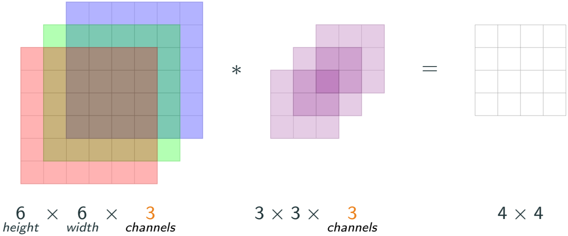
  Example : $\frac{7+0-3}{2} + 1 \quad \times \quad \frac{7+0-3}{2} + 1 \quad \longrightarrow \quad 3 \quad \times \quad 3$

- *Remark 1* : If the fraction is not an integer, round it down, i.e., $z = \text{floor}(z)$
- *Remark 2* : In mathematical textbooks, the convolution operation does double mirroring of the filter first.
- *Remark 3* : Cross-correlation in mathematical textbooks is the convolution operation in Deep Learning literature (simplified terminology).

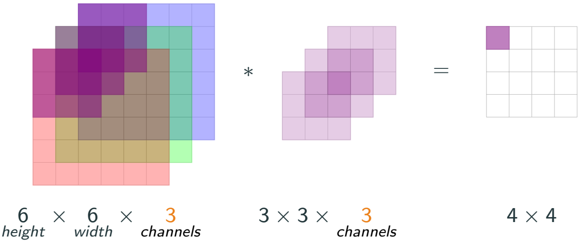
Convolution over Volume

Convolutions on RGB Images

- RGB images consist of 3 color channels : red, green, and blue.
- **The number of channels in the image must match the number of channels in the filter**.



Convolutions on RGB Images



(i.) Place the filter in the upper leftmost position of the input volume.

(ii.) Perform element-wise multiplication of the filter's values with the corresponding values in the input volume.

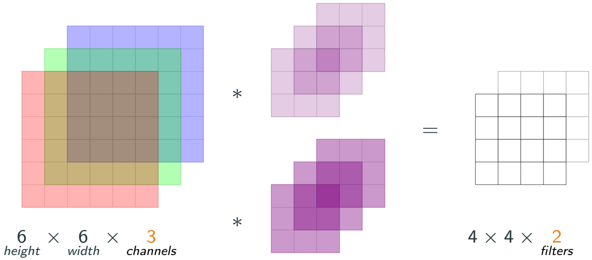
(iii.) Sum up the results of the multiplications to compute the output value.

(iv.) Slide the filter to the next position and repeat steps (ii.) and (iii.).

➟ **Remark** : Filters can have different values for different channels.

Multiple Filters

- **Multiple filters can be applied simultaneously**
- Their outputs are stacked together



- Dimensions :

$$\underset{6\times6\times3}{n \times n \times n_c} \quad * \quad \underset{3\times3\times3}{f \times f \times n_c} \quad \xrightarrow[\text{output}]{} \quad \underset{4\times4\times2}{(n-f+1)\times(n-f+1)\times n_c'}$$

n_c' is the number of filters, i.e., the *new* number of channels for a subsequent conv operation

- **Remark** : *depth* often indicates *the number of channels*

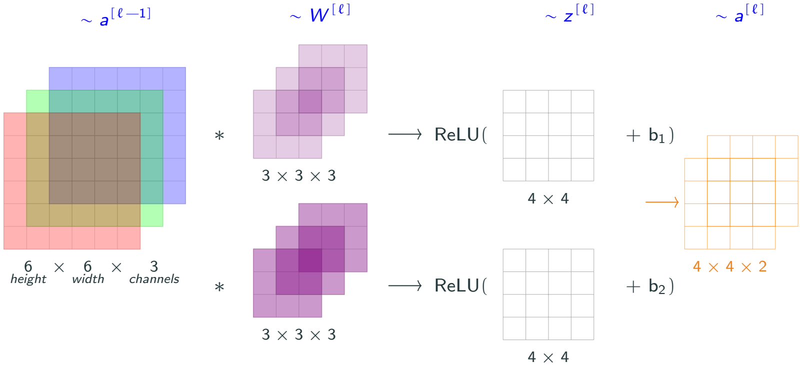
Convolutional Layer

Convolutional Layer

- Convolutional layer : add a bias and apply non-linearity, e.g., ReLU

- Analogy with standard Neural Network layer :

$$z^{[\ell]} = W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}$$
$$a^{[\ell]} = g\left(z^{[\ell]}\right)$$



Number of parameters in one layer

✍ Question. In one layer of a convolutional neural network, if you have 10 filters, and each filter has dimensions of $3 \times 3 \times 3$, how many parameters does this layer have ?

Answer.

1 10

$3 \times 3 \times 3$

27 weights + bias $\longrightarrow$ 28 parameters

10 filters $\longrightarrow$ 280 parameters

➡ Remark : no matter the height and width of the input image, the number of parameters does not change

Summary

➟ If layer ℓ is a convolution layer :

$f^{[\ell]} \times f^{[\ell]} \times n_c^{[\ell-1]} =$ filter size

$p^{[\ell]} =$ padding

$s^{[\ell]} =$ stride

$n_c^{[\ell]} =$ number of filters

Input : $n_H^{[\ell-1]} \times n_W^{[\ell-1]} \times n_c^{[\ell-1]}$

Output : $n_H^{[\ell]} \times n_W^{[\ell]} \times n_c^{[\ell]}$

$$n_{H,W}^{[\ell]} = \left\lfloor \frac{n_{H,W}^{[\ell-1]} + 2p^{[\ell]} - f^{[\ell]}}{s^{[\ell]}} + 1 \right\rfloor$$

➟ $n_c^{[\ell]} =$ number of filters $=$ number of channels output volume

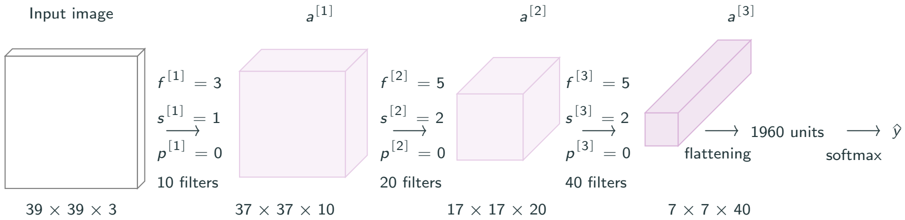
Weights : $f^{[\ell]} \times f^{[\ell]} \times n_c^{[\ell-1]} \times n_c^{[\ell]}$; Bias : $n_c^{[\ell]}$

ConvNet

Convolutional Neural Network

- ConvNet for image classification task, outputs at each layer
- Output dimensions at conv layer ℓ :

$$n_{H,W}^{[\ell]} = \left\lfloor \frac{n_{H,W}^{[\ell-1]} + 2p^{[\ell]} - f^{[\ell]}}{s^{[\ell]}} + 1 \right\rfloor$$



$n_h^{[0]} = n_w^{[0]} = 39$ | $n_H^{[1]} = n_W^{[1]} = 37$ | $n_H^{[2]} = n_W^{[2]} = 17$

$n_c^{[0]} = 3$ | $n_c^{[1]} = 10$ | $n_c^{[2]} = 20$

- Last step : unrolling into a vector (flattening) and feed it to a softmax unit
- As you go deeper, height and width decrease and number of channels increases

Pooling Layer

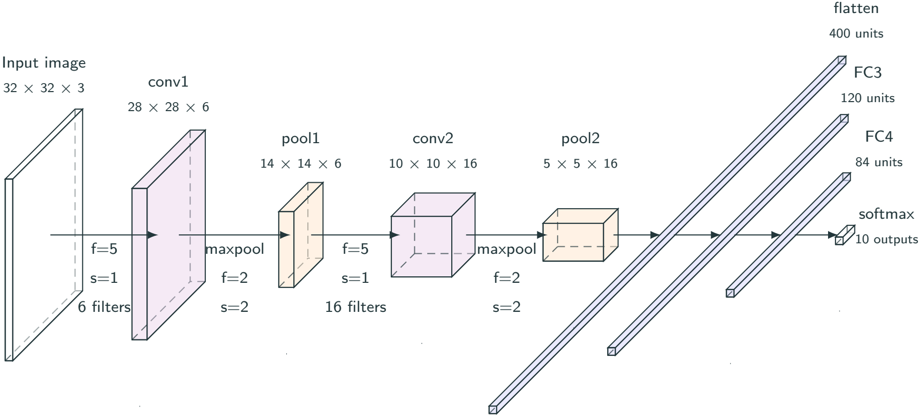
- The pooling layer reduces the spatial size of the input image while retaining the most important features.
- Pooling hyperparameters : filter size f ($f = 2$), stride s ($s = 2$), max or average :

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 9 & 2 \\ 6 & 3 \end{bmatrix}$$

- Pooling is applied on each channel (or feature map) independently.
- **Remark** : pooling layers do not have any learnable parameters.

ConvNet : Example

➟ ConvNet inspired by LeNet-5 :



➟ Fully-connected (FC) layer : single NN layer

e.g., FC3, $W^{[3]}$: $(120, 400)$, $b^{[3]}$: $(120, 1)$

ConvNet : example

	Activation shape	Activation size	# parameters
Input :	$(32, 32, 3)$	$3, 072$	0
conv1 (f=5, s=1)	$(28, 28, 6)$	4,704	456
pool1	$(14, 14, 6)$	1,176	0
conv2 (f=5, s=1)	$(10, 10, 16)$	1,600	2,416
pool2	$(5, 5, 16)$	400	0
FC3	$(120, 1)$	120	48,120
FC4	$(84, 1)$	84	10,164
softmax	$(10, 1)$	10	850

Why Convolutions ?

Main advantages over Fully Connected (FC) layers :

- Parameter sharing : Features learned in one part of the image are reused across different regions.
- Sparsity of connections : Each output value in a layer depends only on a subset of inputs, reducing computational complexity.