Introduction to Convolutional Neural Networks (CNN)

High-Dimensional-Deep-Learning – juliette.chevallier@insa-toulouse.fr

INSA Toulouse, Applied Mathematics, 5th year

Some Classic Problems in Image Processing

Some Classic Problems in Image Processing

Classification

or: Annotation, *Labelling*

Main class: **Person**

Secondary classes:
Building, trees, grass, sky

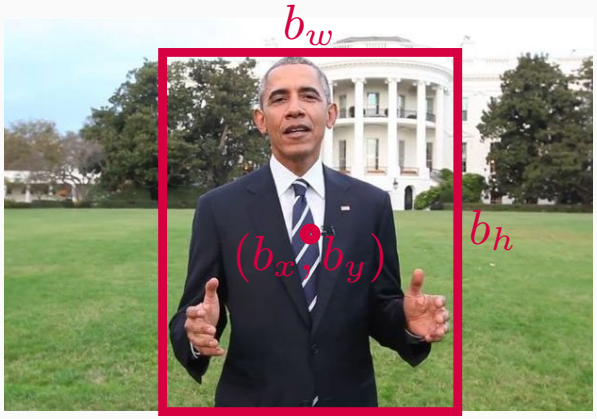
Some Classic Problems in Image Processing

Classification

or: Annotation, *Labelling*

Main class: **Person**

Secondary classes:
Building, trees, grass, sky



Localization

Goal: Delineate the position of the object, using a *bounding box*.

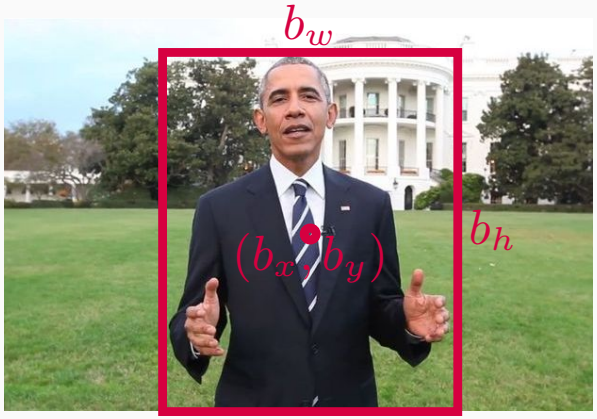
Some Classic Problems in Image Processing

Classification

or: Annotation, *Labelling*

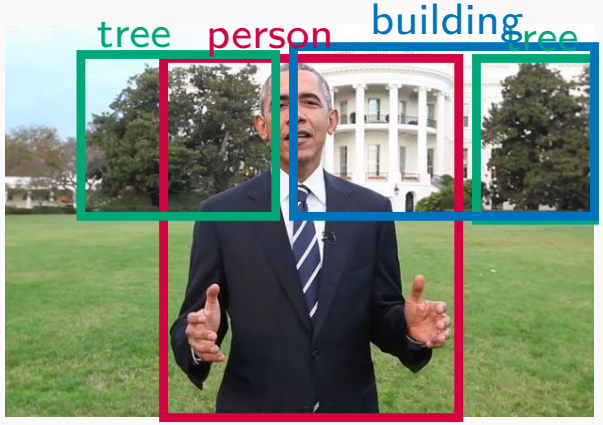
Main class: **Person**

Secondary classes:
Building, trees, grass, sky



Localization

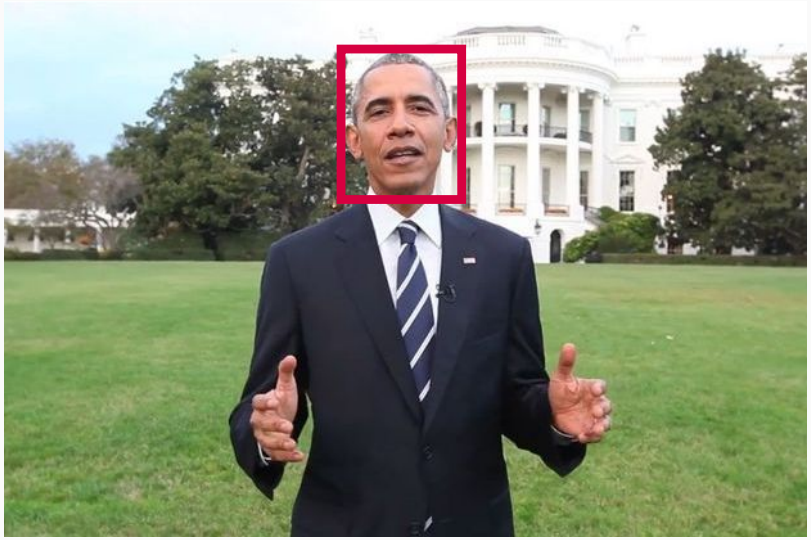
Goal: Delineate the position of the object, using a *bounding box*.



Object detection

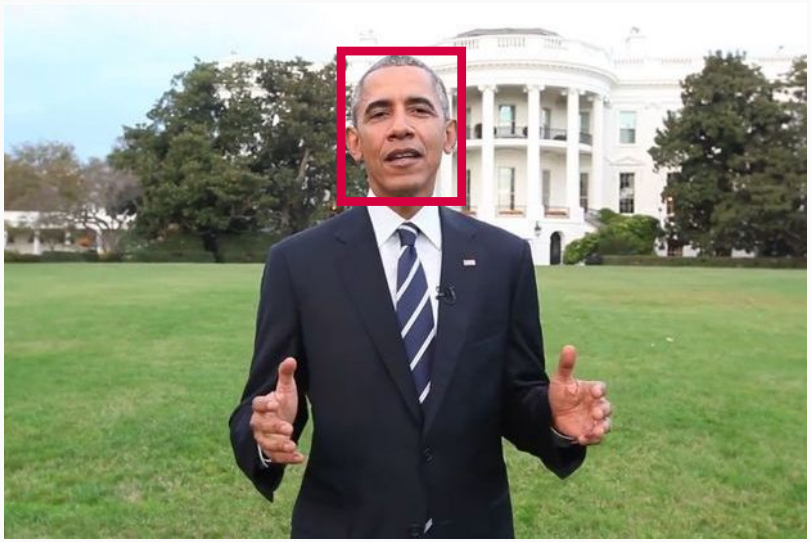
Goal: Delineate the position of multiple objects, possibly with multiple instances, always using bounding boxes.

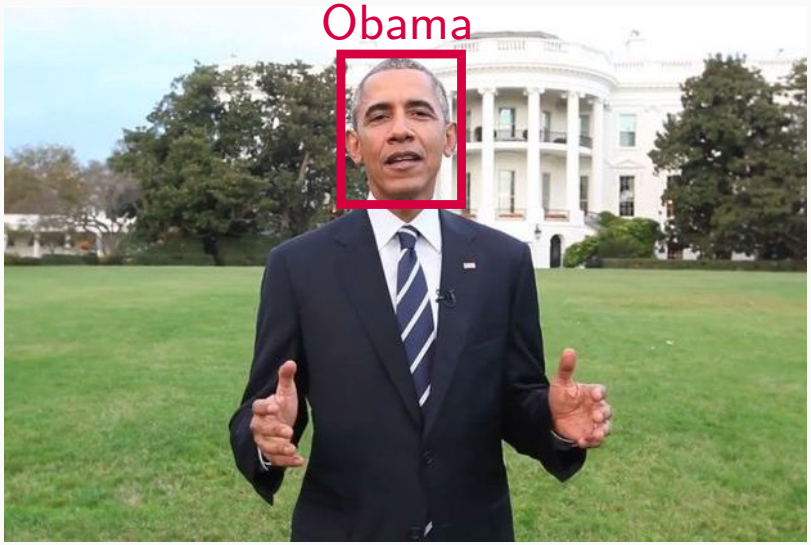
A Famous Sub-Problem of Object Detection: Face Detection



A Famous Sub-Problem of Object Detection: Face Detection

. . . and the recognition that goes with it!





Goal: Given a detected face and a face database, recognize the person.

Remark: *This problem* ***cannot*** *be formulated as a classification problem: Too many classes, and not enough examples.*

This problem is a typical example of ***few-shot learning****.*

Segmentation

Object Segmentation

Goal: Binary classification of image pixels as part of the object or background.

Segmentation

Object Segmentation

Goal: Binary classification of image pixels as part of the object or background.

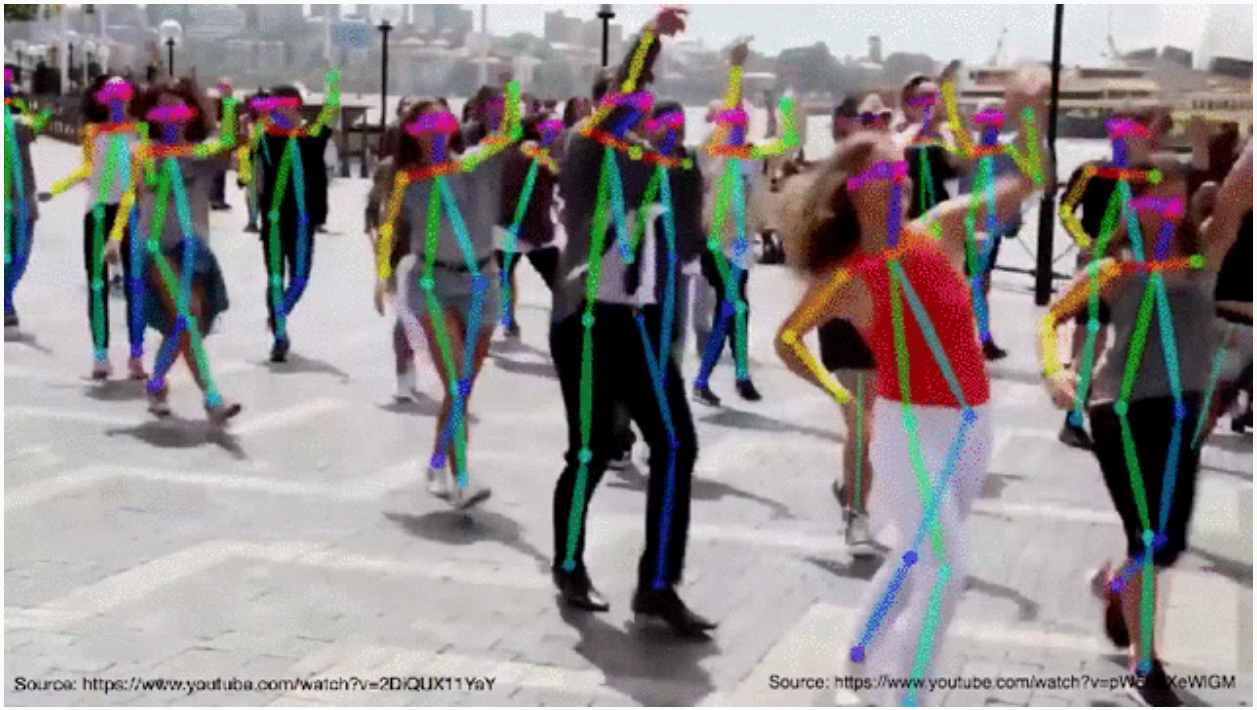
Image Segmentation

ou *Image parsing*

Goal: Classification in n-area of the pixels of the image.

But Also

Pose estimation: Locating the "joints" or articulations of persons in images.



And many others, but rather using *generative models* like GAN:
Style transfer, B&W image colorization, Image reconstruction, Super-resolution image, Image synthesis, *etc.*

Dense Network Limit – MLP

Example *: Image classification on* `ImageNet`*.*

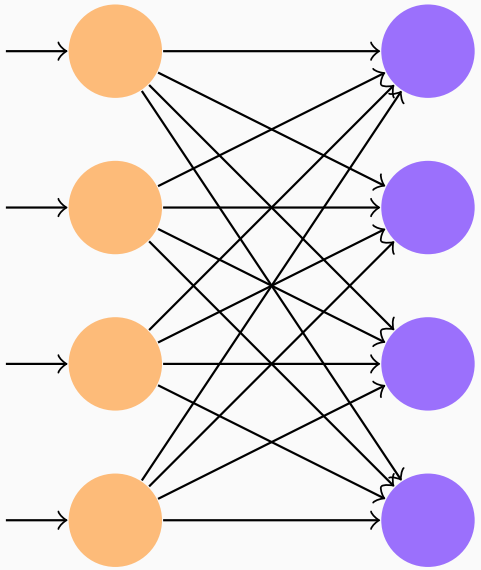
Images of size $224 \times 224 \times 3$, 1000 classes.

```
model = Sequential()
model.add(Input(shape=(224*224*3,)))
model.add(Dense(1000, activation='softmax'))
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1000)	150,529,000

Total params: 150,529,000 (574.22 MB)
Trainable params: 150,529,000 (574.22 MB)
Non-trainable params: 0 (0.00 B)



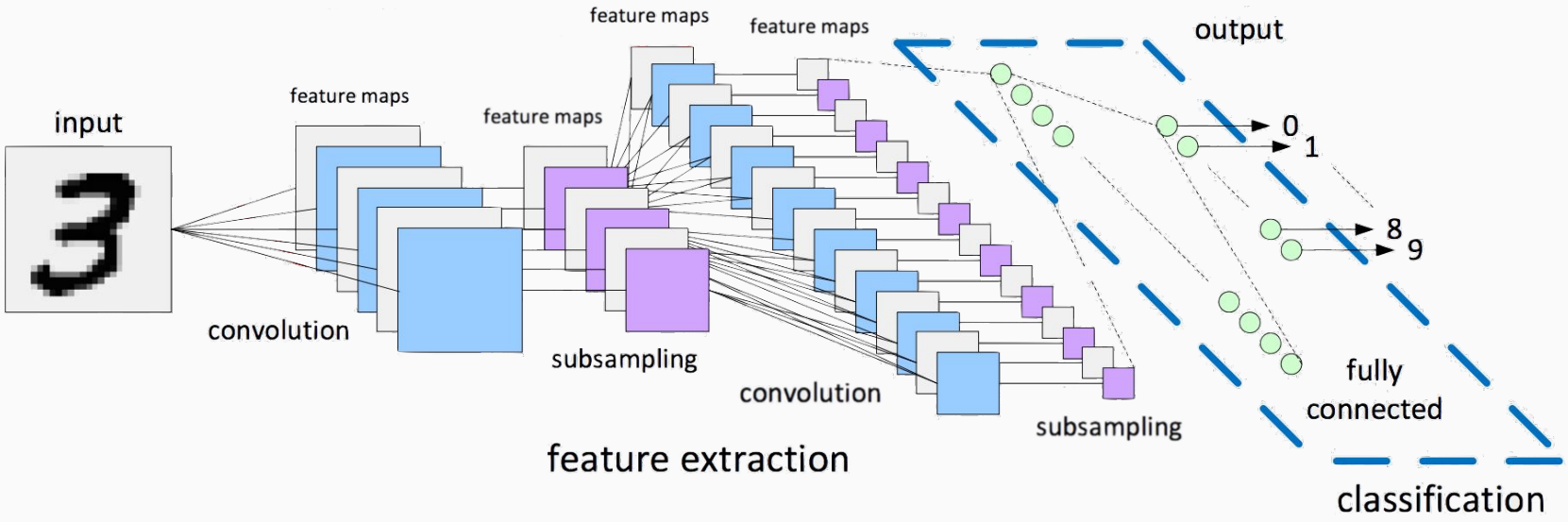
$224 \times 224 \times 3$ pixels

$224 \times 224 \times 3$ neurones

1000 neurones

- A single-layer perceptron requires $224 \times 224 \times 3 \times 1000 + 1000$ parameters, i.e more than 150 millions of parameters !
- Spatial information is lost with *flattening*.

Typical Architecture of a Convolutional Neural Network



3 type of Layers:

- *Firsts layers*: Convolution layers combined with pooling layers,
- *Last layers*: Fully-connected or dense layers.

Convolution Layers

Convolution

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

Image $\mathbf{4 \times 4}$

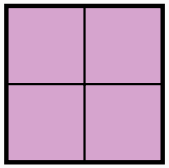
$\circledast$

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

Filter 3×3

$f = 3$

$=$



Response $\mathbf{2 \times 2}$

Convolution

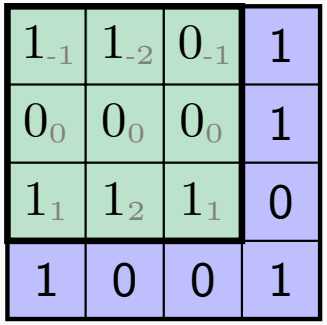
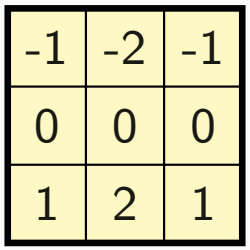


Image $\mathbf{4 \times 4}$

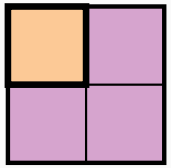
$\circledast$



Filter 3×3

$f = 3$

$=$



Response $\mathbf{2 \times 2}$

Convolution

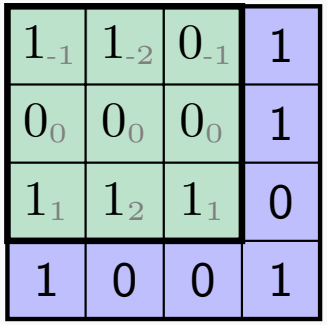
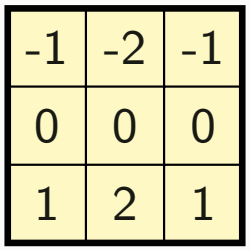


Image $\mathbf{4 \times 4}$

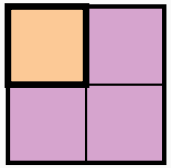
$\circledast$



Filter 3×3

$f = 3$

$=$



Response $\mathbf{2 \times 2}$

$$-1 \times 1 - 2 \times 1 - 1 \times 0 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times 1 + 2 \times 1 + 1 \times 1 = \mathbf{1}$$

Convolution

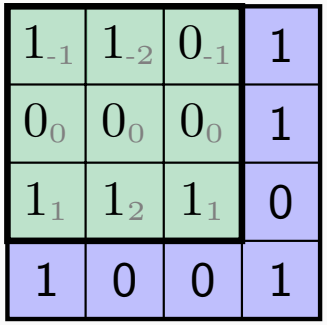
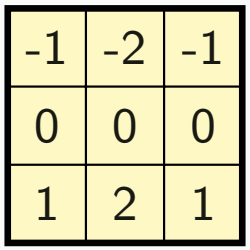


Image $\mathbf{4 \times 4}$

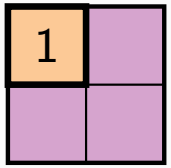
$\circledast$



Filter 3×3

$f = 3$

$=$



Response $\mathbf{2 \times 2}$

$$-1 \times 1 - 2 \times 1 - 1 \times 0 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times 1 + 2 \times 1 + 1 \times 1 = \mathbf{1}$$

Convolution

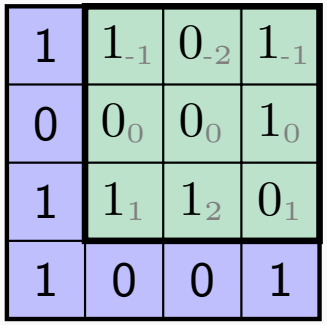
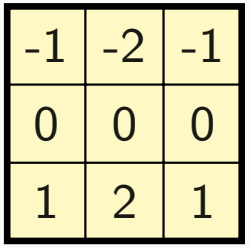


Image $\mathbf{4 \times 4}$

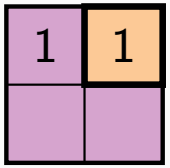
$\circledast$



Filter 3×3

$f = 3$

$=$



Response $\mathbf{2 \times 2}$

$$-1 \times 1 - 2 \times 0 - 1 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times 1 + 2 \times 1 + 1 \times 0 = \mathbf{1}$$

Convolution

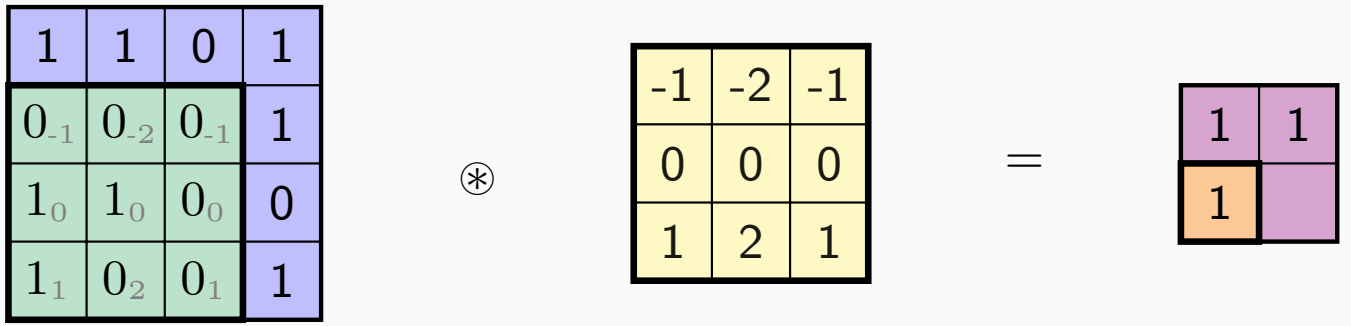


Image $\mathbf{4 \times 4}$

Filter 3×3

$f = 3$

Response $\mathbf{2 \times 2}$

$$-1 \times 0 - 2 \times 0 - 1 \times 1 + 0 \times 1 + 0 \times 1 + 0 \times 0 + 1 \times 1 + 2 \times 0 + 1 \times 0 = \mathbf{1}$$

Convolution

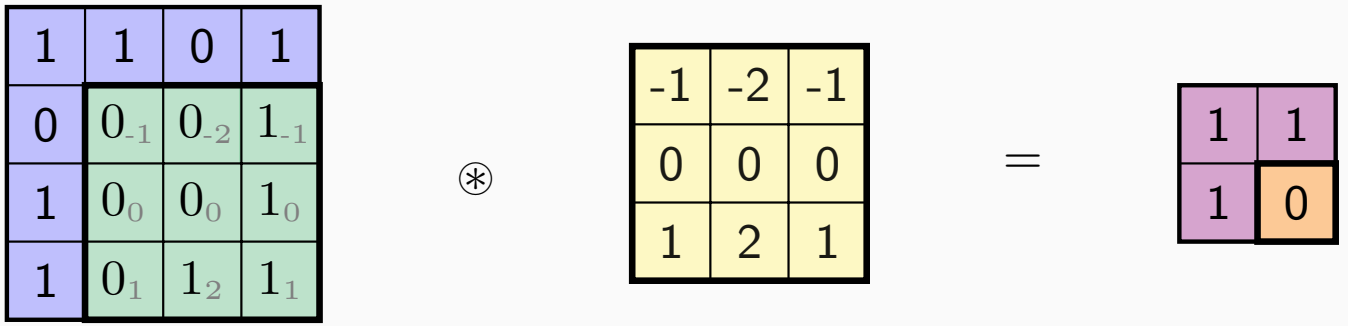


Image $\mathbf{4 \times 4}$

Filter 3×3

$f = 3$

Response $\mathbf{2 \times 2}$

$$-1 \times 0 - 2 \times 0 - 1 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 1 + 1 \times 0 + 2 \times 1 + 1 \times 1 = \mathbf{0}$$

Convolution

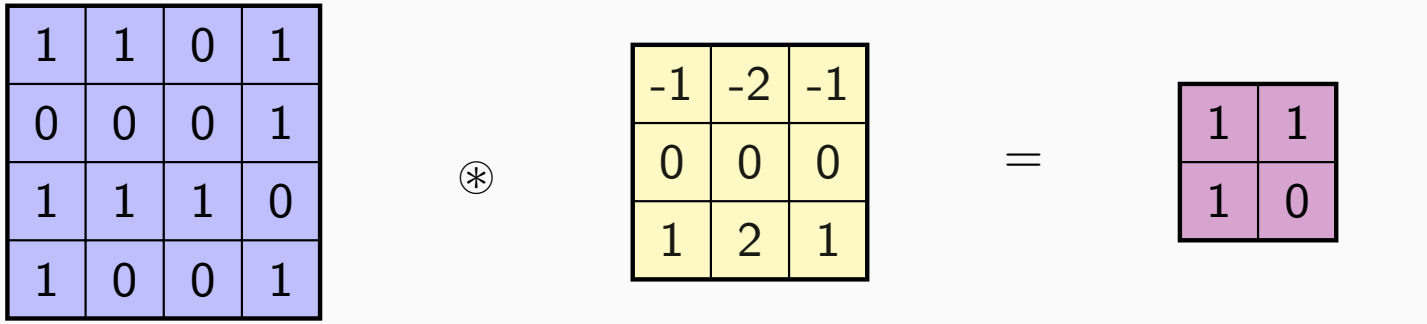


Image 4×4

Filter 3×3

$f = 3$

Response $\mathbf{2 \times 2}$

2D Discrete Convolution

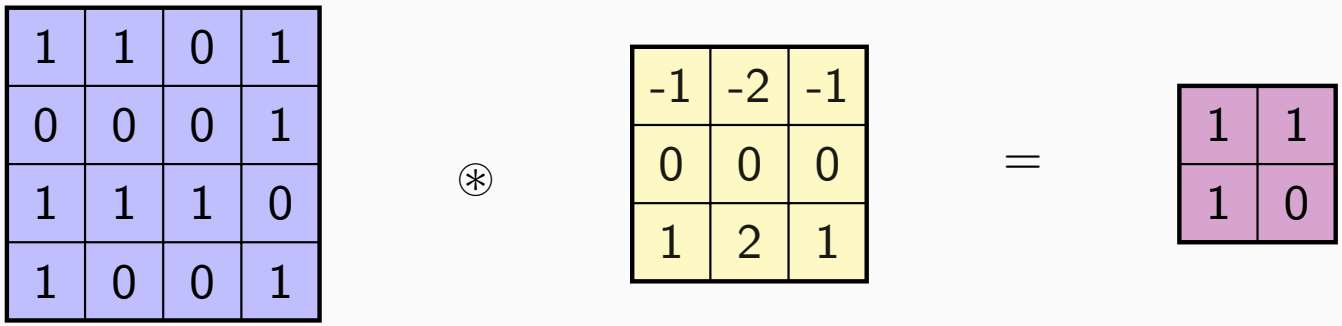


Image $\mathbf{4 \times 4}$

Filter 3×3

$f = 3$

Response $\mathbf{2 \times 2}$

2D discrete convolution:

$$(F \circledast I)(x, y) = \sum_{m,n} F(n, m) \cdot I(x-m, y-n)$$

2D Discrete Convolution

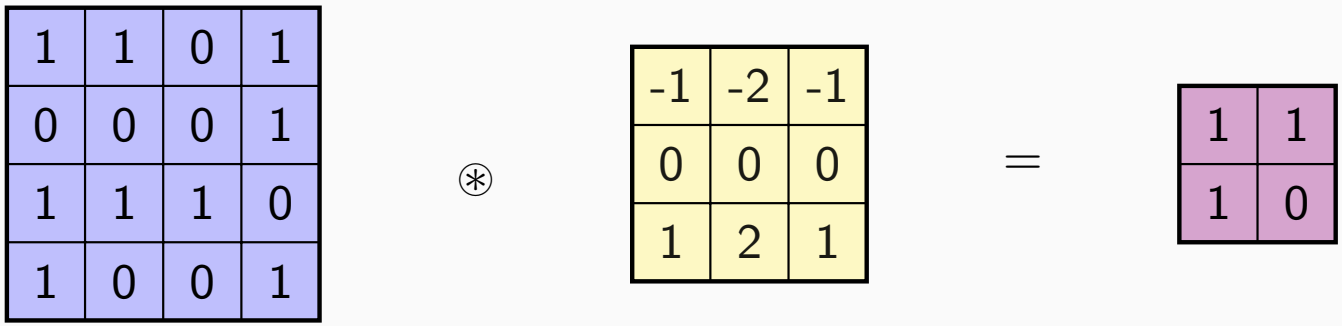


Image $\mathbf{4 \times 4}$

Filter 3×3
$f = 3$

Response $\mathbf{2 \times 2}$

Given :

- A grayscale image I (a single *color channel*) of dimension $w \times h$,
- A filter F of size f,

$\rightsquigarrow$ Dimension of the response $F \circledast I$:

$$\boxed{(w - f + 1) \times (h - f + 1)}$$

2D discrete convolution:

$$(F \circledast I)(x, y) = \sum_{m,n} F(n, m) \cdot I(x - m, y - n)$$

2D Discrete Convolution

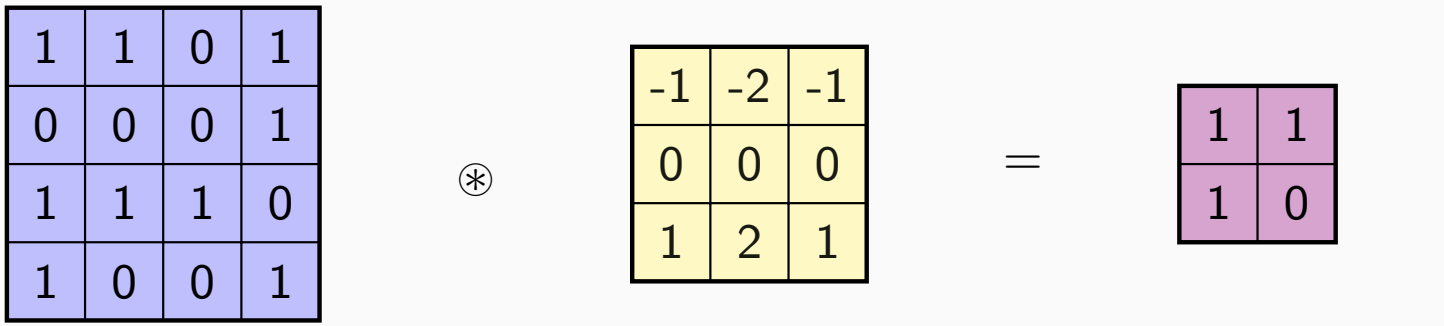


Image $\mathbf{4 \times 4}$

Filter 3×3
$f = 3$

Response $\mathbf{2 \times 2}$

Given :

- A grayscale image I (a single *color channel*) of dimension $w \times h$,
- A filter F of size f,

$\rightsquigarrow$ Dimension of the response $F \circledast I$:

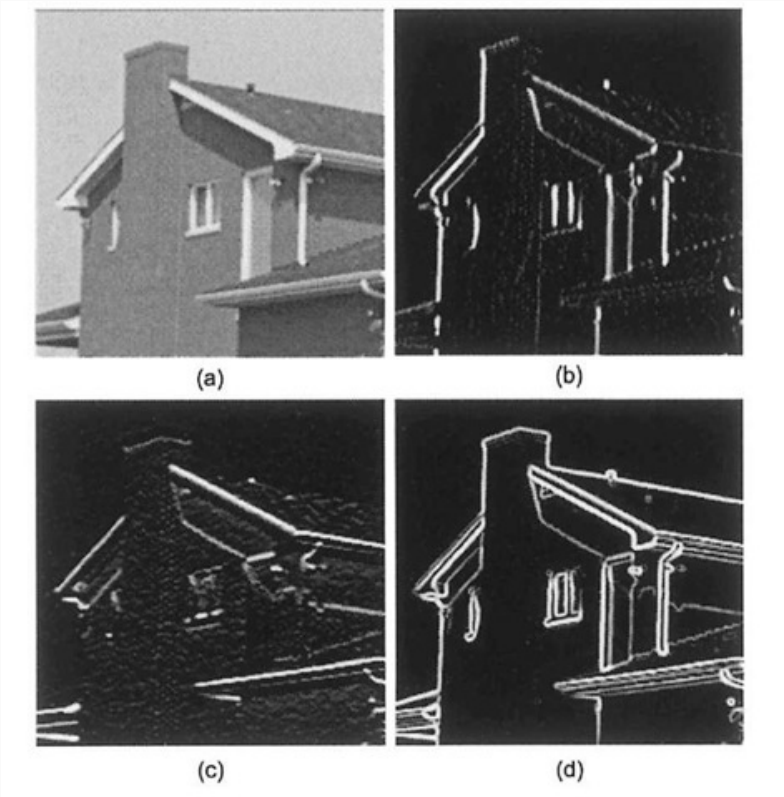
$$(w - f + 1) \times (h - f + 1)$$

2D discrete convolution:

$$(F \circledast I)(x, y) = \sum_{m,n} F(n, m) \cdot I(x - m, y - n)$$

- takes into account the spatial organization,
- translation invariant,
- parameter sharing.

Convolution in Computer Vision



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

Gx

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

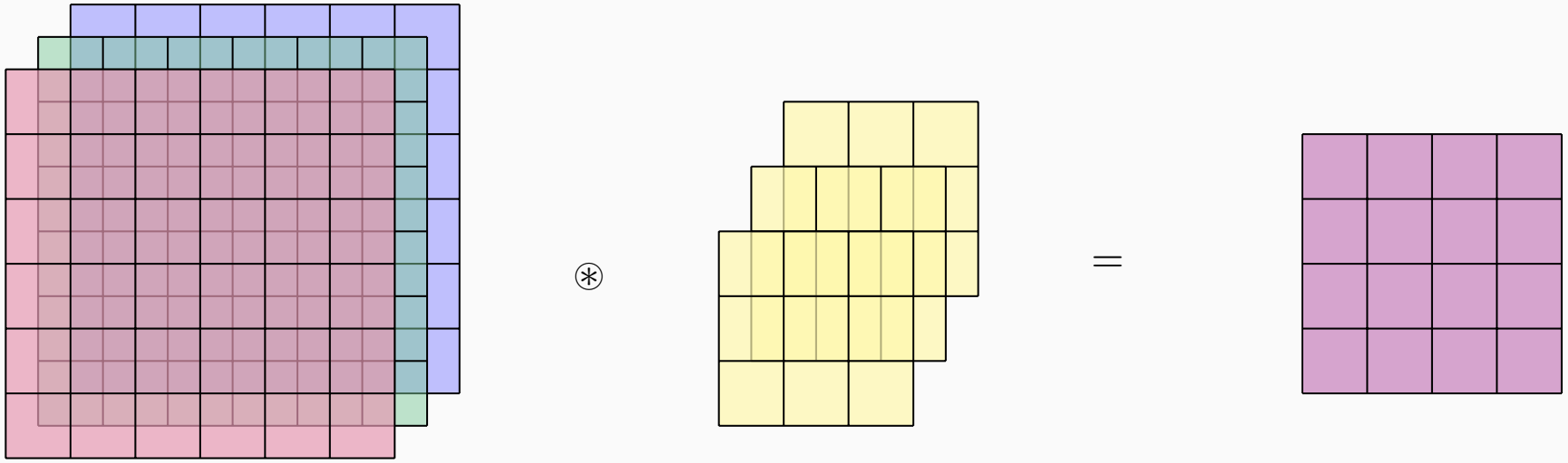
Gy

- Convolution filters (or kernels) have long been used to detect patterns in images, such as contours (here: *Sobel filters*);

- A white pixel indicates a high filter response. Here, this means a pixel located on the contour of an object, with a strong local gradient.

2D Volume Convolution: Multiple Color Channels

Consider a colored image, in 3 dimensions: height, width, channels (RGB).



Image

$6 \times 6 \times \mathbf{3}$

Filters

$3 \times 3 \times \mathbf{3}$

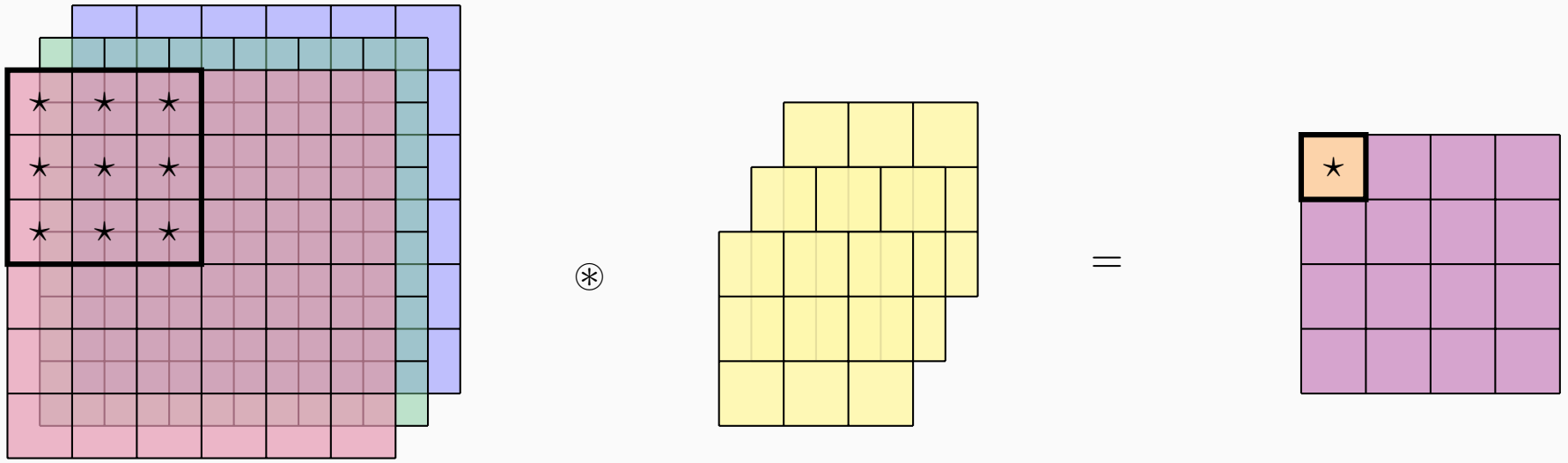
Response

$4 \times 4 \times 1$

The number of channels in the input image and the depth of the convolution filters are necessarily identical.

2D Volume Convolution: Multiple Color Channels

Consider a colored image, in 3 dimensions: height, width, channels (RGB).



Image

$6 \times 6 \times \mathbf{3}$

Filters

$3 \times 3 \times \mathbf{3}$

Response

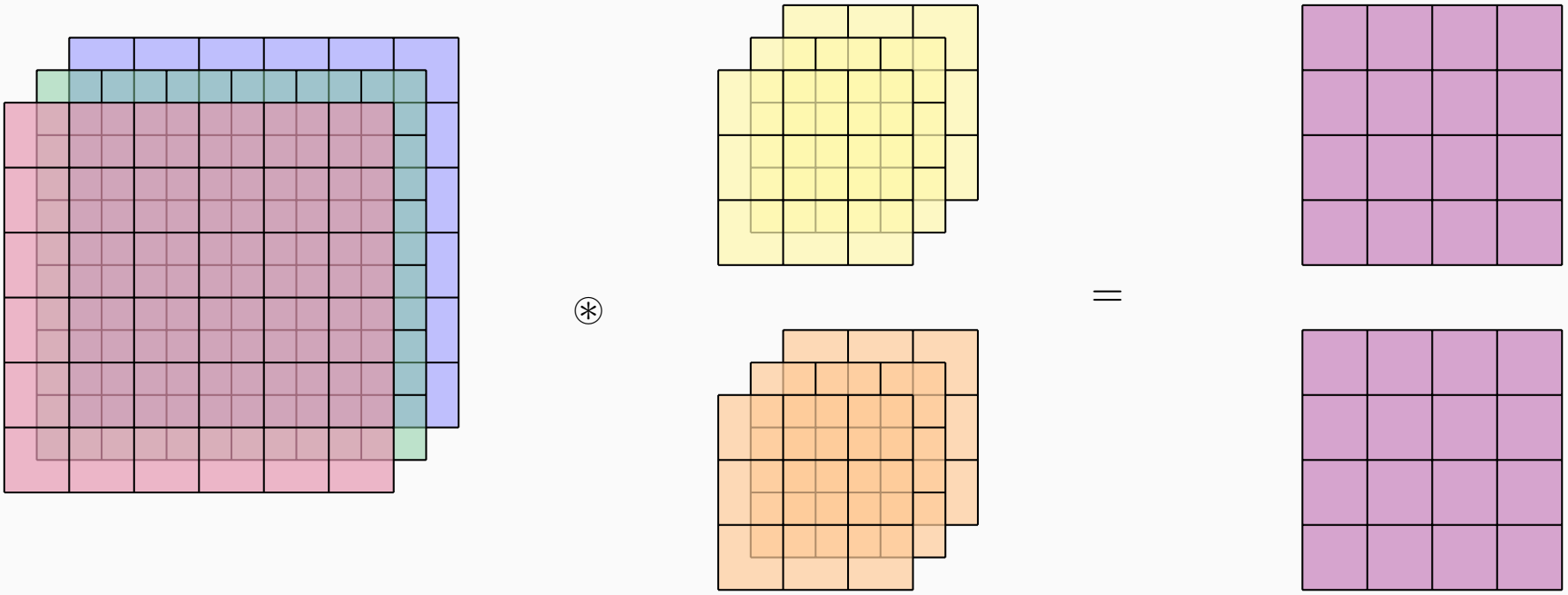
$4 \times 4 \times 1$

$$(F \circledast I)(x, y) = \sum_{\boldsymbol{c}} \sum_{m} \sum_{n} F^{c}(n, m) \cdot I^{c}(x - m, y - n)$$

The number of channels in the input image and the depth of the convolution filters are necessarily identical.

2D Volume Convolution Layer

Consider a colored image, in 3 dimensions: height, width, channels (RGB).



Image

$6 \times 6 \times \mathbf{3}$

Filtres

$3 \times 3 \times \mathbf{3} \times 2$

Réponse

$4 \times 4 \times 2$

Number of Parameters in a Convolution Layer

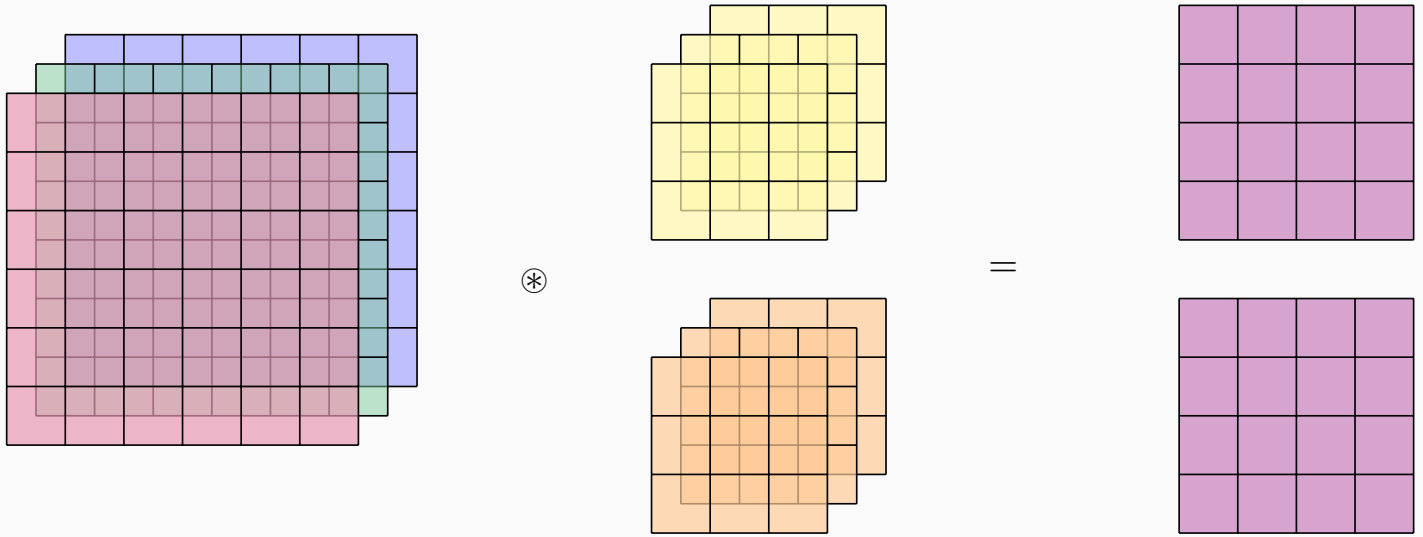


Image
$6 \times 6 \times \mathbf{3}$

Filtres
$3 \times 3 \times \mathbf{3} \times 2$

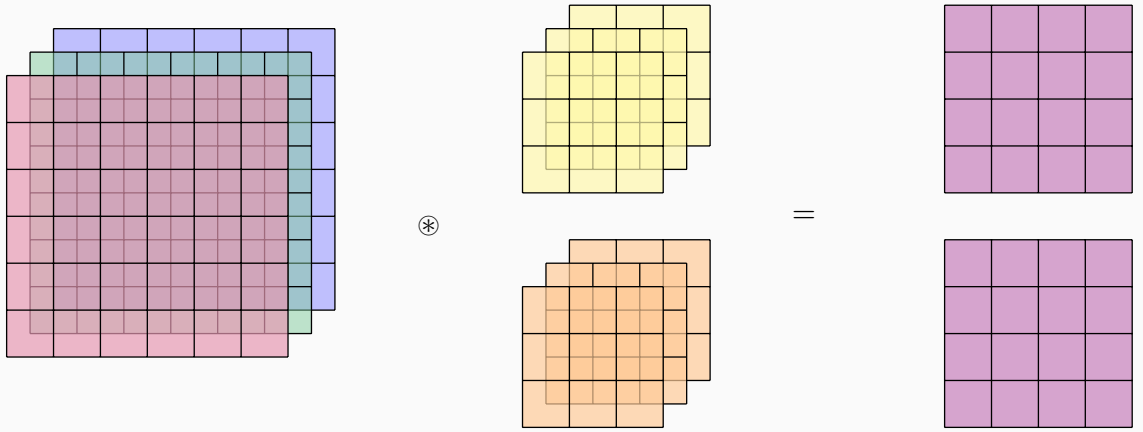
Réponse
$4 \times 4 \times 2$

Two types of parameters in a convolution layer :

- The **coefficients of the convolution** filters: $f \times f \times \#\text{channels} \times \#\text{filters}$,
- The **biases** added to the *response* of the convolution filters, before application of the activation function. There is exactly one bias per filter, i.e. $\#\text{filters}$.

$\rightsquigarrow$ In the above example, there are $3 \times 3 \times 3 \times 2 + 2 = 56$ parameters.

Number of Operations in a Convolution Layer



It is interesting to count the number of operations of a neural network to characterize its complexity, and the required resources required for its execution. We focus mainly on **addition** and **multiplication**.

Here, each element of the answer requires :

- $(f \times f \times \#\text{channels})$ multiplications,
- $(f \times f \times \#\text{channels} - 1)$ additions between multiplied values,
- 1 additional addition for the bias.

$\rightsquigarrow$ In the above example, there are $(4 \times 4 \times 2) \times (3 \times 3 \times 3 \times 2) = 1728$ operations.

Remark : CPU *vs.* GPU $\rightsquigarrow$ *TPU !*

CPU:

- Few cores ...
- ... but extremely complexe and fast.
- Shared memory system.

$\rightsquigarrow$ *Sequential tasks*

GPU:

- A lot of cores ...
- ... but simple and slow.
- Not shared memory system

$\rightsquigarrow$ *Parallel tasks*

Remark : CPU *vs.* GPU $\rightsquigarrow$ *TPU !*

CPU:

- Few cores ...
- ... but extremely complexe and fast.
- Shared memory system.

$\rightsquigarrow$ *Sequential tasks*

GPU:

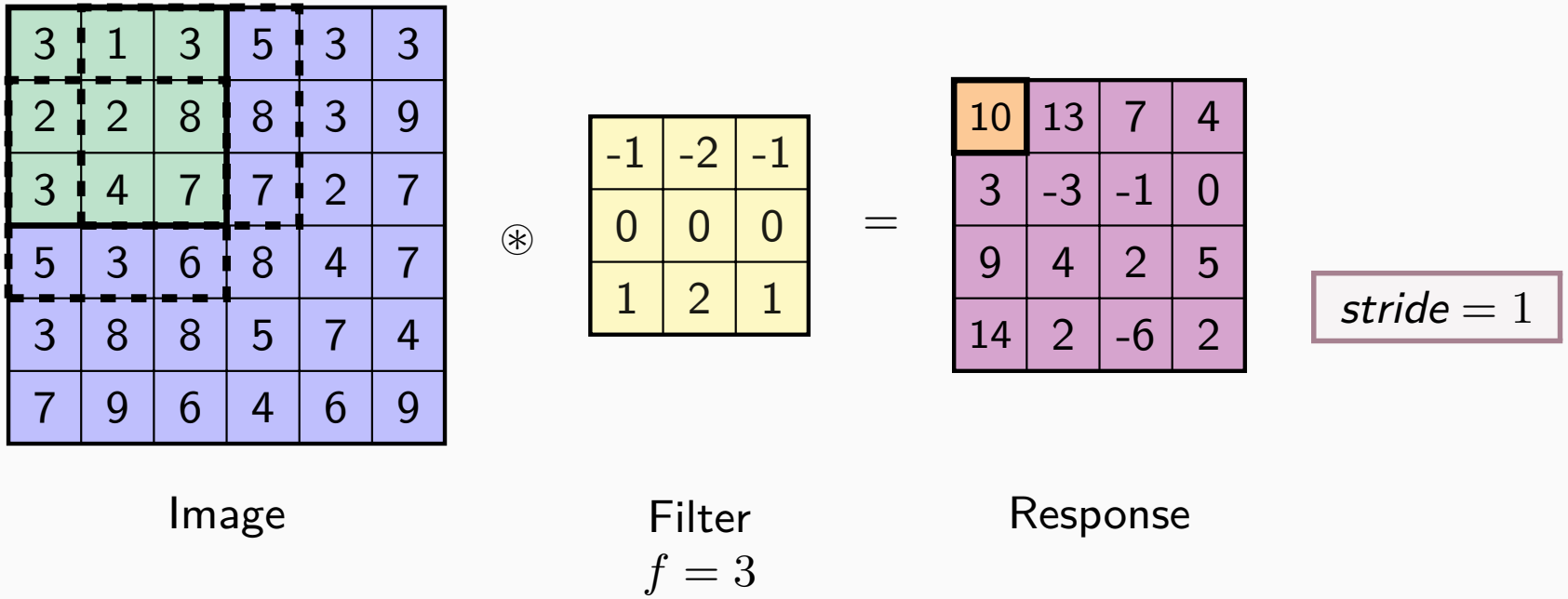
- A lot of cores ...
- ... but simple and slow.
- Not shared memory system

$\rightsquigarrow$ *Parallel tasks*

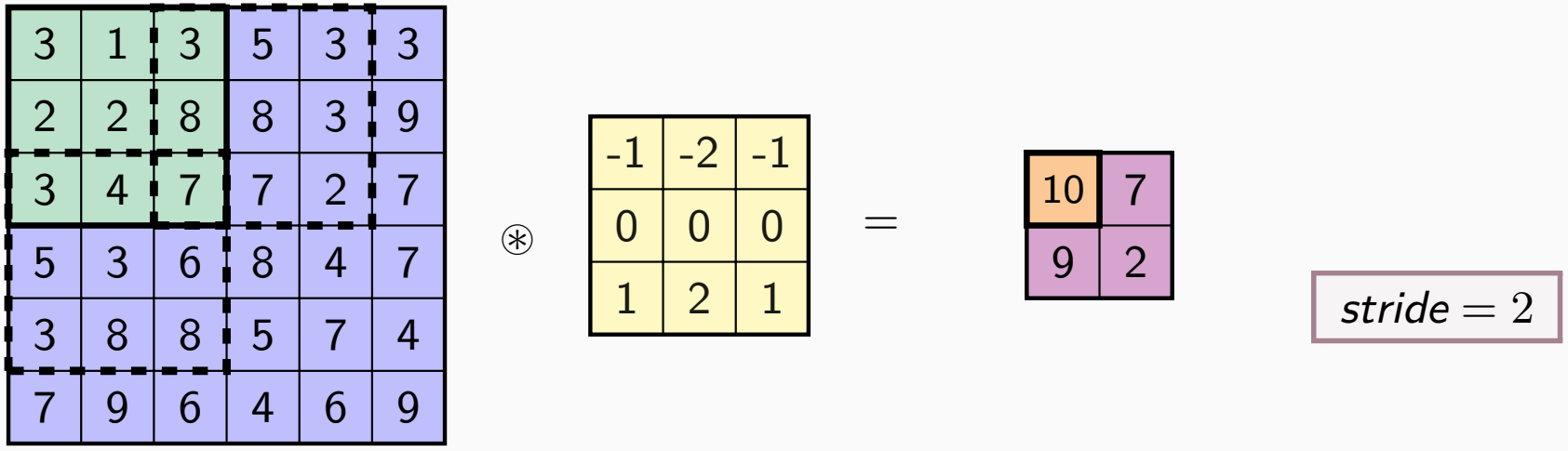
- GPU is not adapted to all ML algorithms
- Data loading time to GPU can be really high
  - GPU is useful only if computation time exceed loading time
  - i.e. useful for complex model

Convolution Layers

Stride

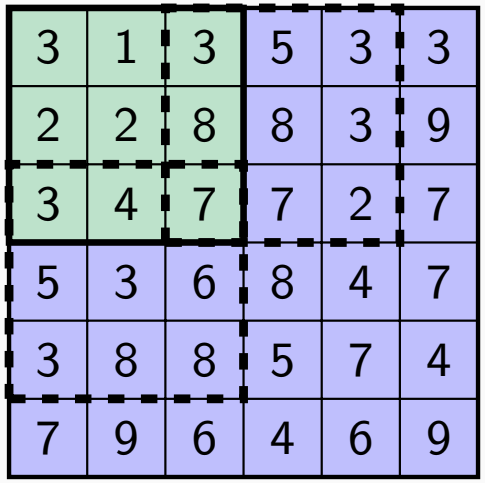

3 1 3 5 3 3
2 2 8 8 3 9
3 4 7 7 2 7
5 3 6 8 4 7
3 8 8 5 7 4
7 9 6 4 6 9
⊛
-1 -2 -1
0 0 0
1 2 1
=
10 13 7 4
3 -3 -1 0
9 4 2 5
14 2 -6 2
stride = 1
Image
Filter
$f = 3$
Response

Stride



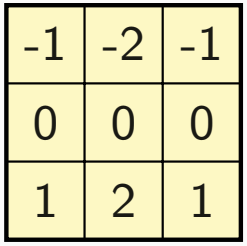
- Reduces the dimension of tensors,
- Limiting the loss of information: *Same coefficient influences several elements of the response* to the convolution filter.

Stride



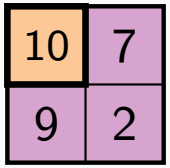
Image

$\circledast$



Filter
$f = 3$

$=$



Response

$stride = 2$

- Reduces the dimension of tensors,
- Limiting the loss of information: *Same coefficient influences several elements of the response* to the convolution filter.

Given:

- A grayscale image I (a single *color channel*) of size $w \times h$,
- A filter F of size f, and stride s,

$\rightsquigarrow$ Dimension of the response $F \circledast I$:

$$\left\lfloor \frac{w-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h-f}{s} + 1 \right\rfloor$$

Padding

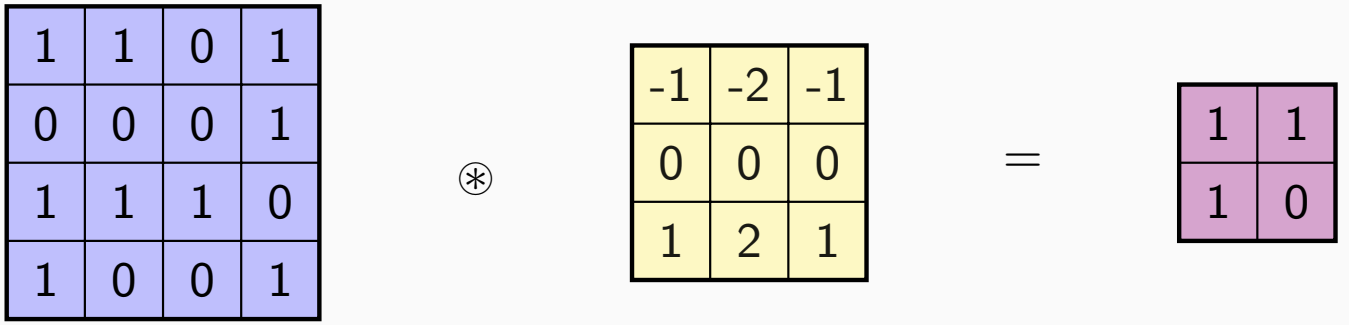


Image Filter Response

Padding

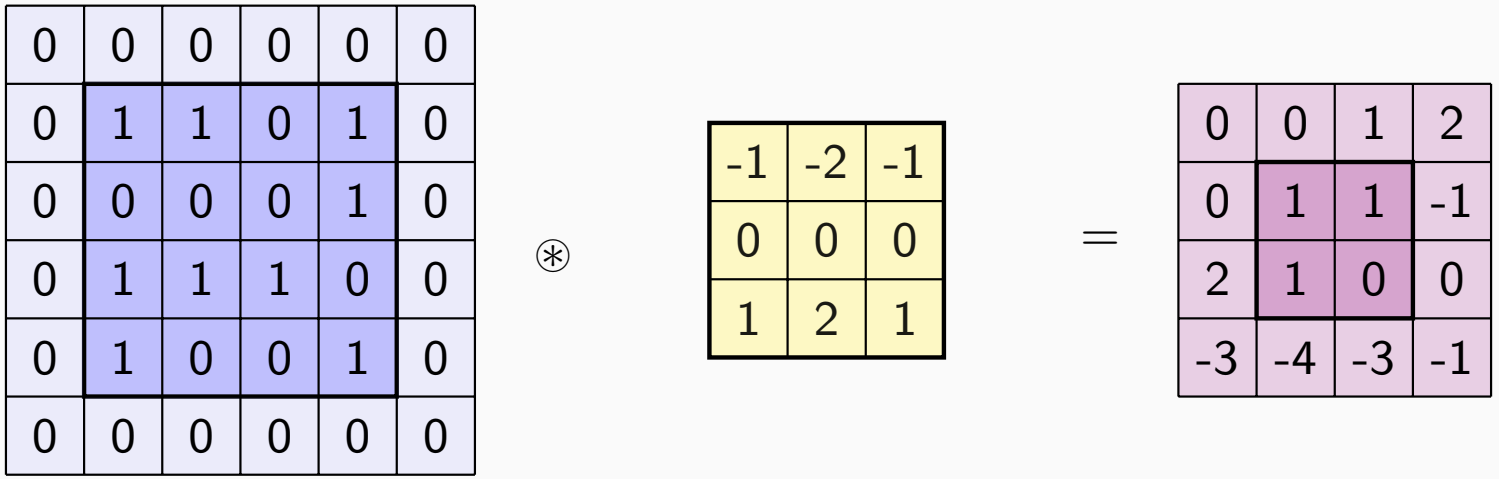


Image
$p = 1$

Filter

Response

- Zero-padding: Add zeros to image edges,
- For example, allow to obtain a response of the same dimension as the input image (same convolution)
  $\rightsquigarrow$ Facilitates the chaining of convolution layers in a neural network.

Padding

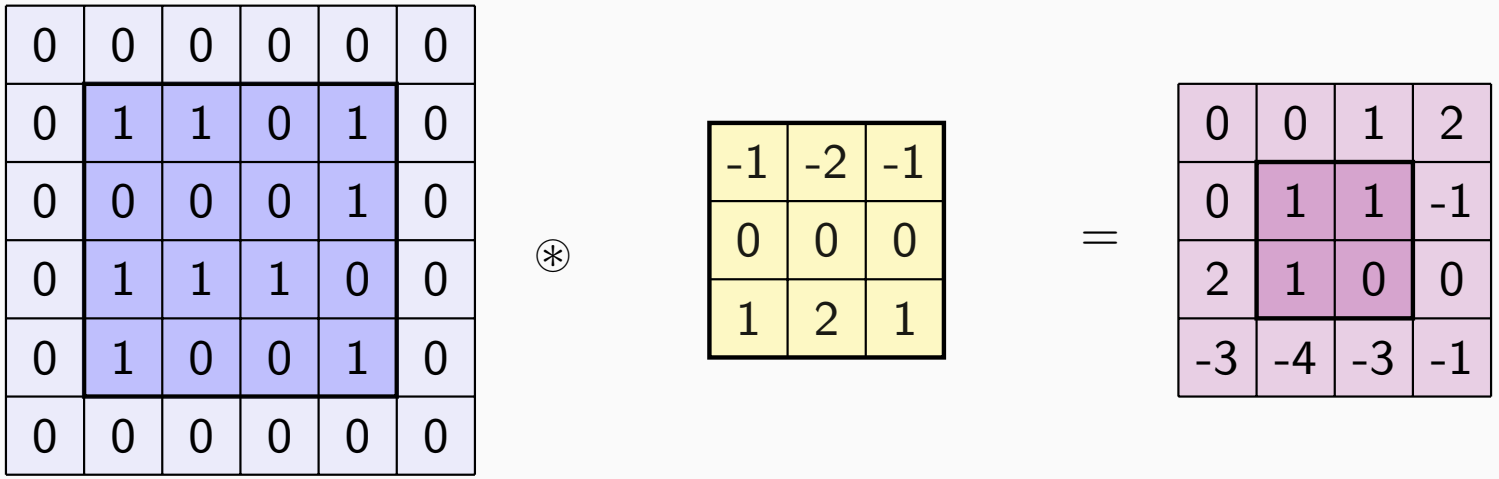


Image
$p = 1$

Filter

Response

- Zero-padding: Add zeros to image edges,
- For example, allow to obtain a response of the same dimension as the input image (same convolution)
  $\rightsquigarrow$ Facilitates the chaining of convolution layers in a neural network.

Given

- A grayscale image I (a single *color channel*) of size $w \times h$,
- A filter F of size f, and padding p,

$\rightsquigarrow$ Dimension of the response $F \circledast I$:

$$(w + 2p - f + 1) \times (h + 2p - f + 1)$$

Impact on Tensor Dimension

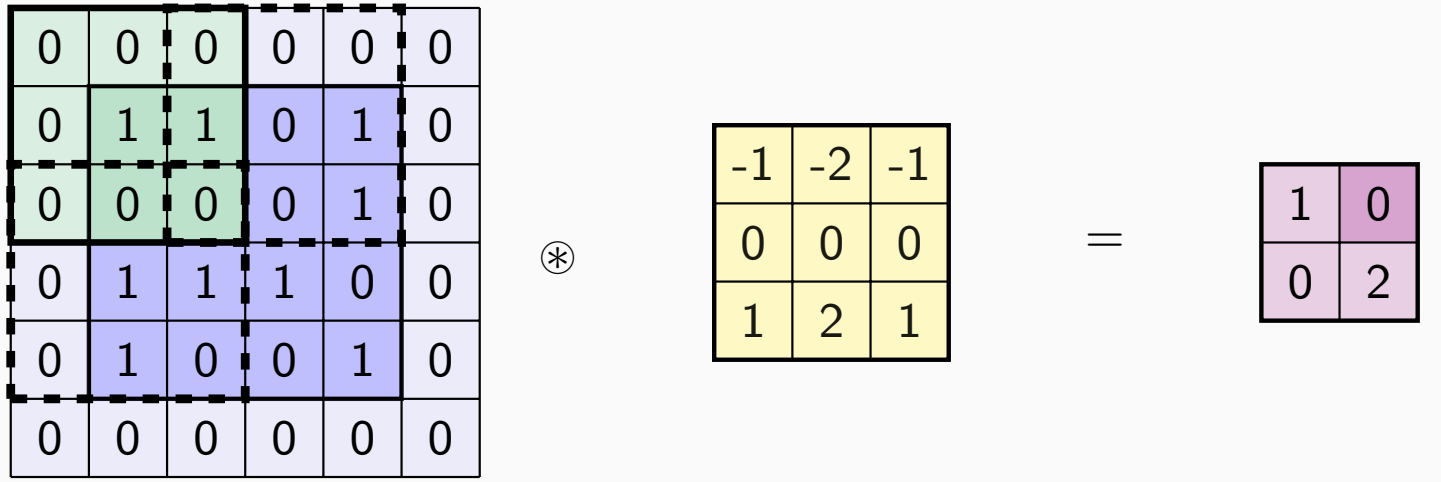


Image
$p = 1$

Filter
stride $= 2$

Response

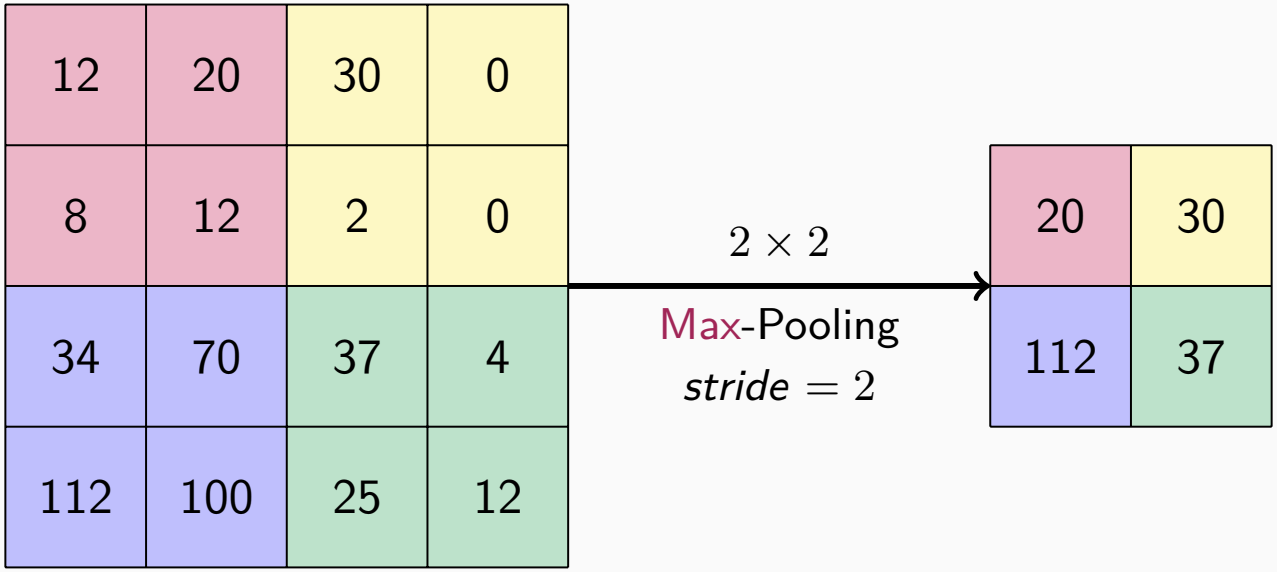
Given:

- A grayscale image I (a single *color channel*) of size $w \times h$,
- A filter F of size f, padding p and stride s

$\rightsquigarrow$ Dimension of the response $F \circledast I$ of image I to filter F:

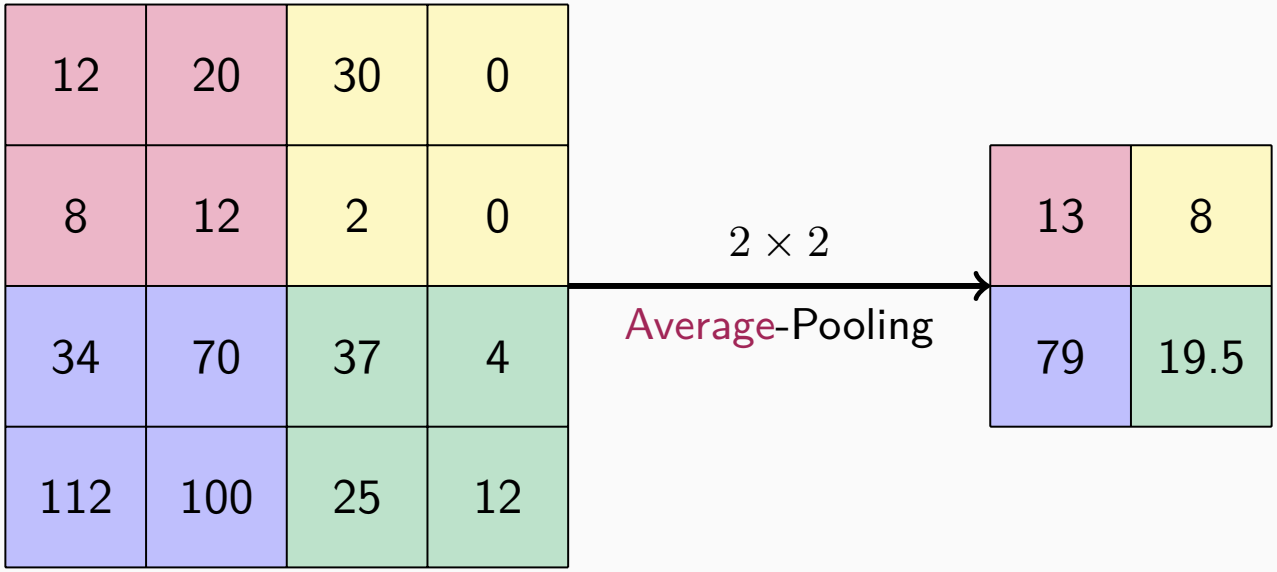
$$\left\lfloor \frac{w + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h + 2p - f}{s} + 1 \right\rfloor$$

Pooling Layer



- Reduces tensor dimensions, to counterbalance the multiplication of convolution filter responses,
- **Idea**: Preserve the high responses to convolution filters,
- Translation invariant,
- No parameters to learn!

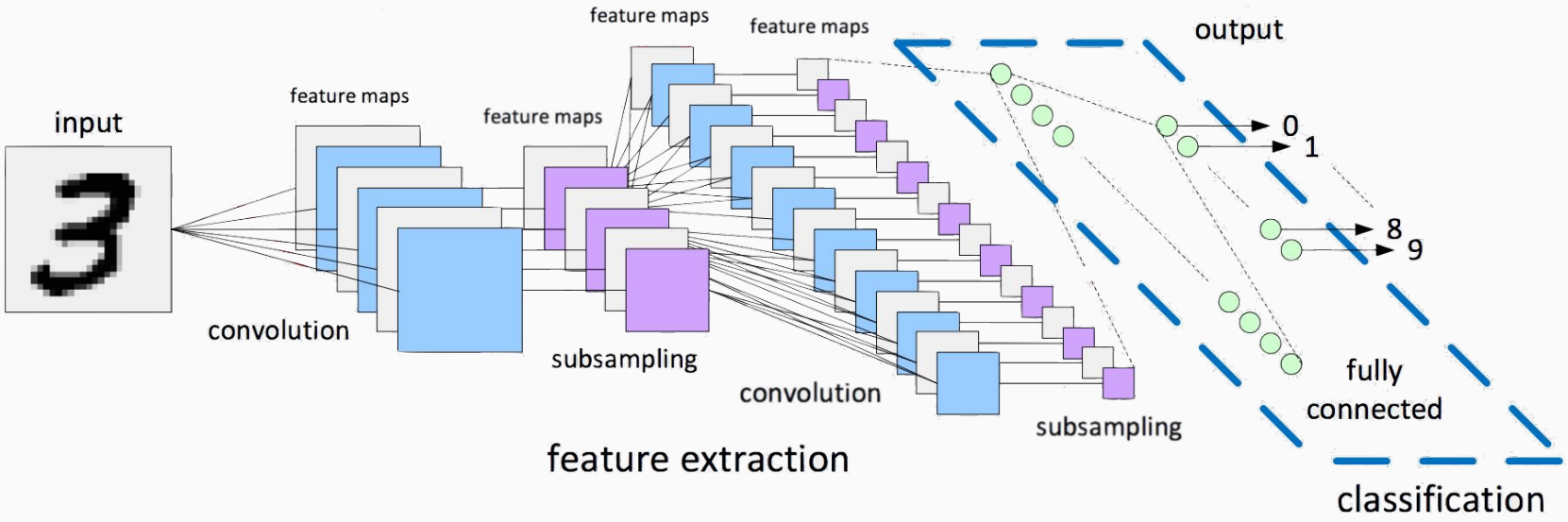
Pooling Layer



- Reduces tensor dimensions, to counterbalance the multiplication of convolution filter responses,
- **Idea**: Preserve the high responses to convolution filters,
- Translation invariant,
- No parameters to learn!

- **Average-Pooling** : Alternatively, we can average the values rather than keeping the maximum,
- **But**: Standard architectures favor **Max-Pooling**.

Typical Architecture of a Convolutional Neural Network (back again)



3 type of Layers:

- *Firsts layers*: Convolution layers combined with pooling layers,
- *Last layers*: Fully-connected or dense layers.

Convolution Layers

Advantages of Convolutional Neural Networks

Parameter sharing :

- Convolution layer : Equivalent to a dense layer in which some synaptic weights are **shared**, and the majority is zero.

- These weights, i.e. the convolution coefficients, may be adjusted *multiple times* for the *same example* in the training database.

$\rightsquigarrow$ A much smaller number of parameters for a convolutional network than for a fully connected network.

A Look Back at Image Classification on `ImageNet`

Images of size $224 \times 224 \times 3$, 1000 classes.

```
model = Sequential()
model.add(Input(shape=(224,224,3)))
model.add(Conv2D(1000, kernel_size=(3,3),  activation='relu'))
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 222, 222, 1000)	28,000

Total params: 28,000 (109.38 KB)
Trainable params: 28,000 (109.38 KB)
Non-trainable params: 0 (0.00 B)

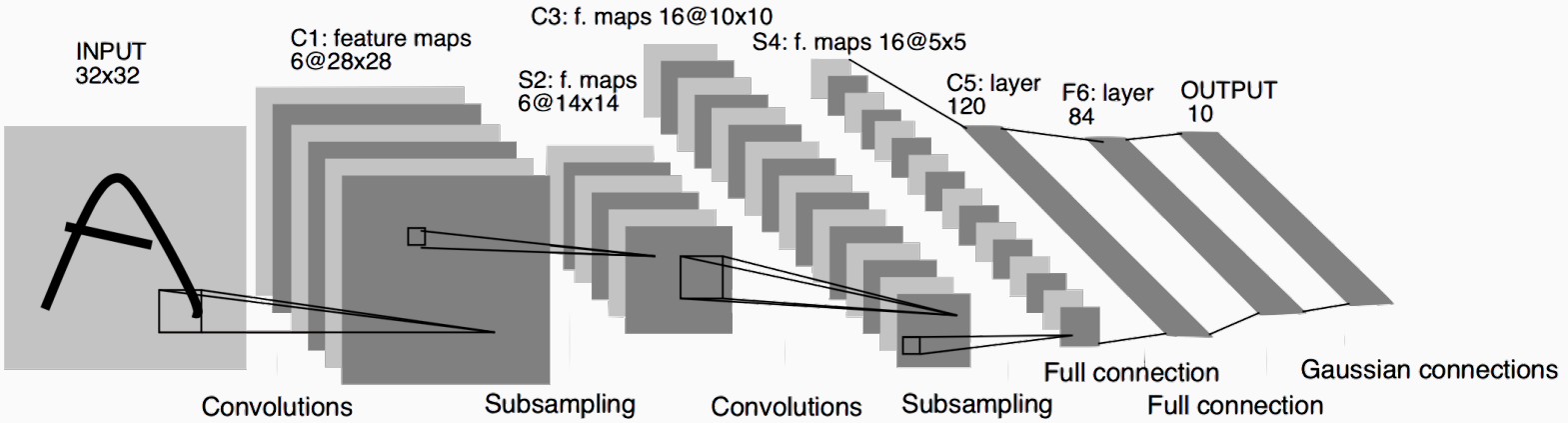
- With a perceptron *mono*-layer, more than 150 **millions** of parameters,
- With a simple convolution layer, size 3 filter, a single filter:
  "only" $28\,000$ $(=(3^3+1)\times 1000)$ parameters.

CNNs and Image Classification

3.1 Main Architectures

3.2 ImageNet Challenge

A Pioneer : LeNet (1998)



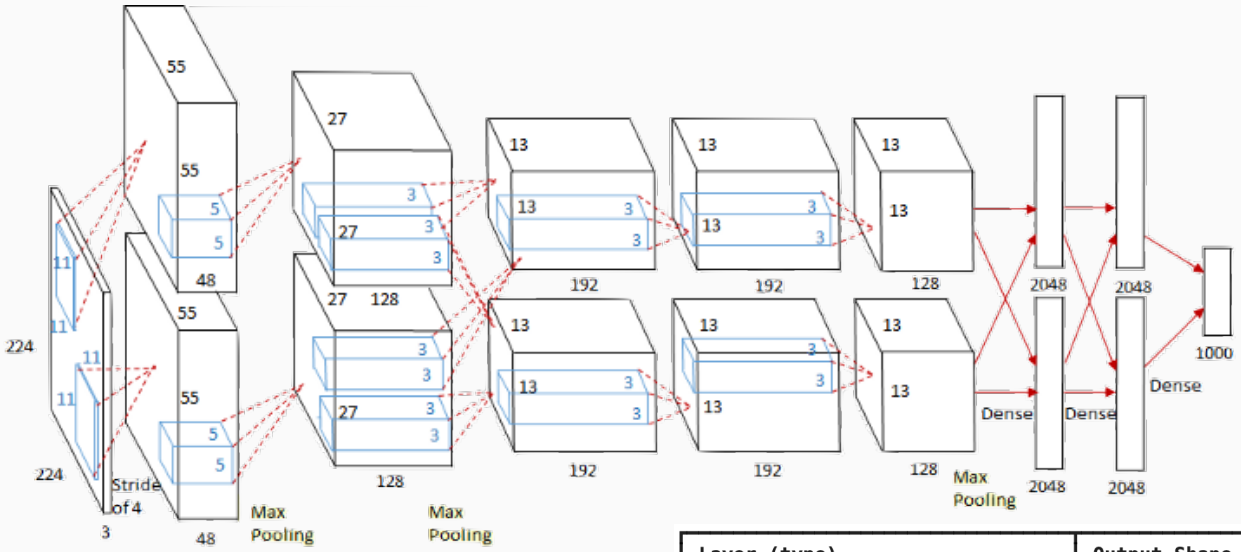
- $\simeq$ 60k parameters,
- 2 to 3 days of training for 20 *epochs* on MNIST (in 1998!),
- Sigmoid or tanh activation functions.

[LeCun et al.] Gradient-based learning applied to document recognition.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d (AveragePooling2D)	(None, 14, 14, 6)	0
activation (Activation)	(None, 14, 14, 6)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	2,416
average_pooling2d_1 (AveragePooling2D)	(None, 5, 5, 16)	0
activation_1 (Activation)	(None, 5, 5, 16)	0
conv2d_2 (Conv2D)	(None, 1, 1, 120)	48,120
flatten (Flatten)	(None, 120)	0
dense (Dense)	(None, 84)	10,164
dense_1 (Dense)	(None, 10)	850

Total params: 61,706 (241.04 KB)
Trainable params: 61,706 (241.04 KB)
Non-trainable params: 0 (0.00 B)

The Rocker : AlexNet (2012)



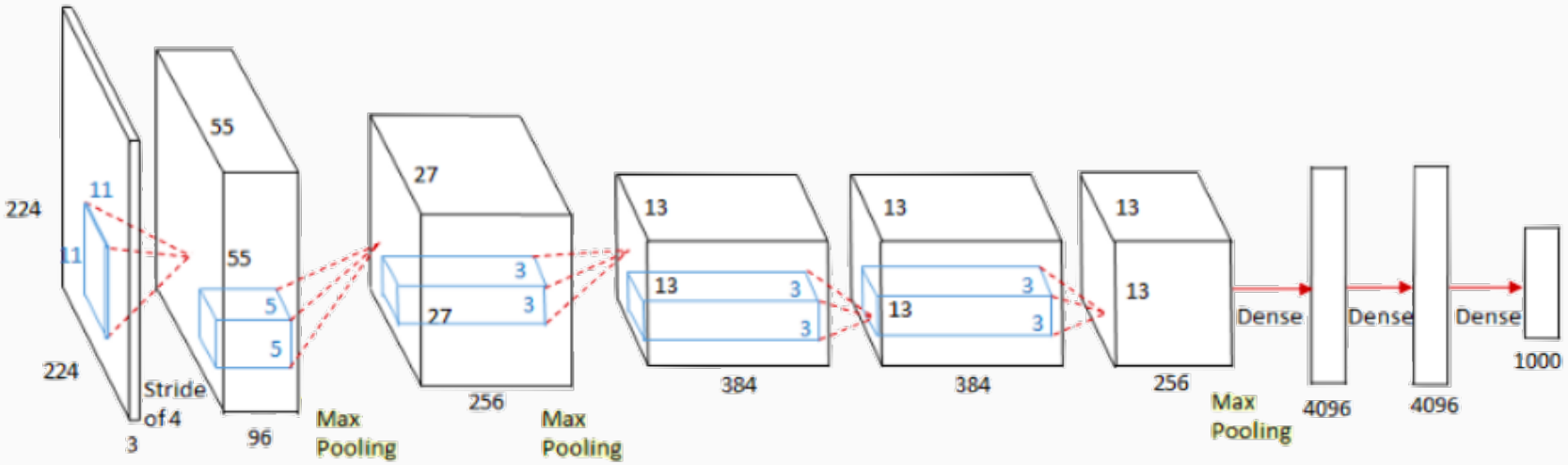
- $\simeq$ 60M parameters, 8 layers,
- Architecture designed to be deployed on 2 GPUs,
- Introduces the use of the `reLU` function as a standard for deep network training.

[Krizhevsky et al.] `ImageNet` Classification with Deep Convolutional Neural Netv

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 54, 96)	34,944
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 17, 17, 256)	2,973,952
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 256)	0
conv2d_2 (Conv2D)	(None, 6, 6, 384)	885,120
conv2d_3 (Conv2D)	(None, 4, 4, 384)	1,327,488
conv2d_4 (Conv2D)	(None, 2, 2, 256)	884,992
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 4096)	1,052,672
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16,781,312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 1000)	4,097,000
dropout_2 (Dropout)	(None, 1000)	0
dense_3 (Dense)	(None, 17)	17,017

Total params: 28,054,497 (107.02 MB)

The Rocker : AlexNet (2012)

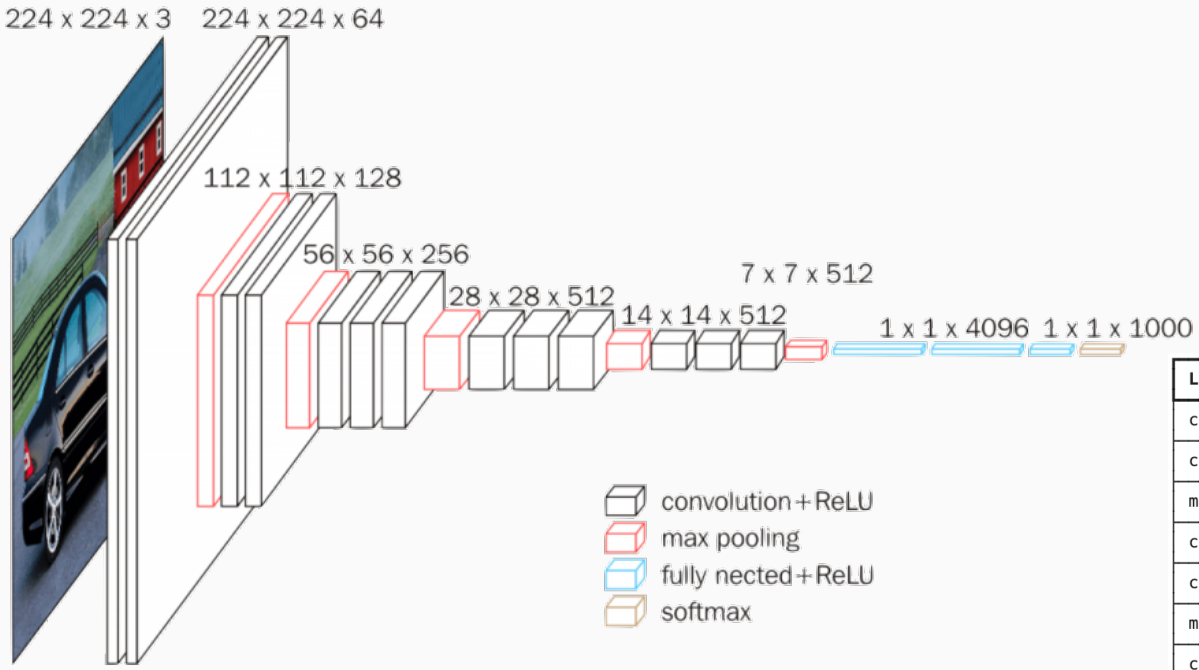


Observations :

- Gradual reduction of *filter* size: $11 \to 5 \to 3$;
- Gradual reduction of *image* size: $224 \to 55 \to 27 \to 13$;
- Gradual increase in the number of filters: $96 \to 256 \to 384$;
- *Stride* then *Max Pooling*

[Krizhevsky et al.] `ImageNet` Classification with Deep Convolutional Neural Networks.

A “simplified” Version: VGG-16 (2014)



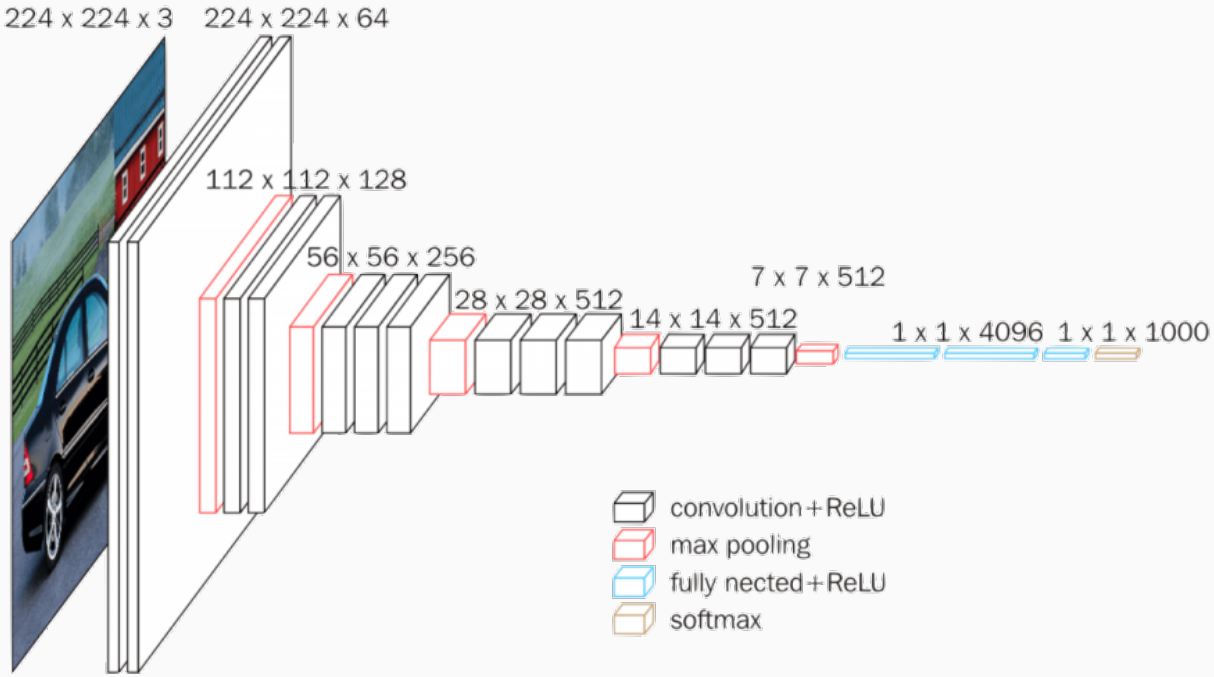
$\simeq$ 138M parameters, 16 layers in its standard version.

[Simonyan et Zisserman] Very Deep Convolutional Networks for Large-Scale Image Recognition

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73,856
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147,584
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295,168
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1,180,160
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2,359,808
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2,359,808
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_10 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4096)	102,764,544
dense_1 (Dense)	(None, 4096)	16,781,312
dense_2 (Dense)	(None, 17)	69,649

Total params: 134,330,193 (512.43 MB)

A “simplified” Version: VGG-16 (2014)



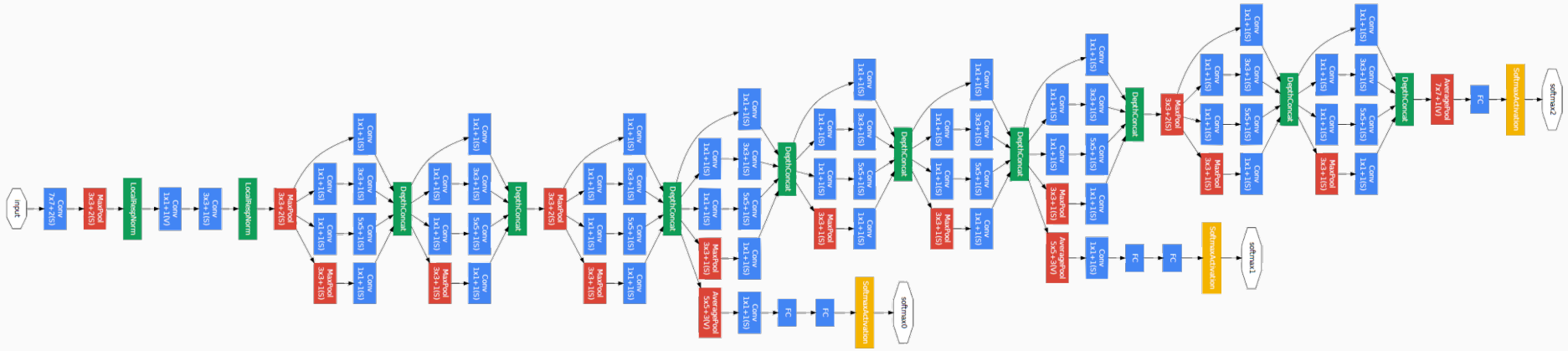
Objective : Study the impact of *depth* on network performance.

⇝ To this end, the authors made the network architecture very regular:

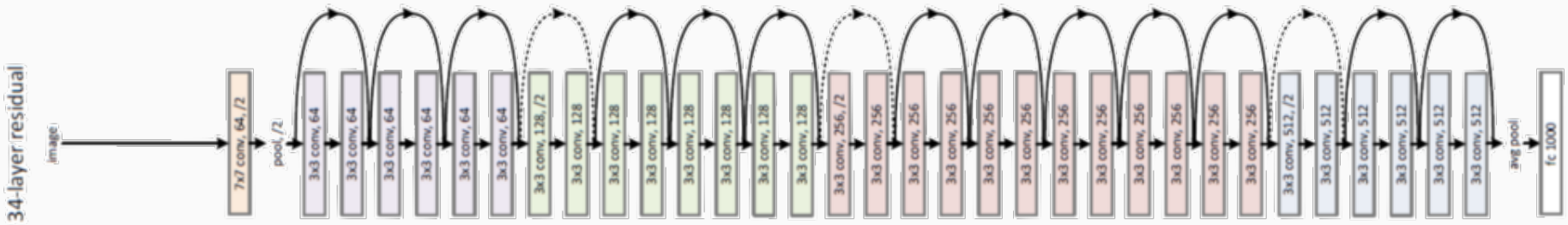
- Systematic use of convolutions 3×3,
- The main features of `AlexNet` are retained, but regularized:
  - *Gradual reduction in image size:* $224 \to 112 \to 56 \ldots$
  - *Gradual increase in the number of filters:* $64 \to 128 \to 256$

[Simonyan et Zisserman] Very Deep Convolutional Networks for Large-Scale Image Recognition

More Advanced Architectures: Inception & ResNet



[Szegedy et al.] Going deeper with convolutions.



[He et al.] Deep Residual Learning for Image Recognition

CNNs and Image Classification

3.1 Main Architectures

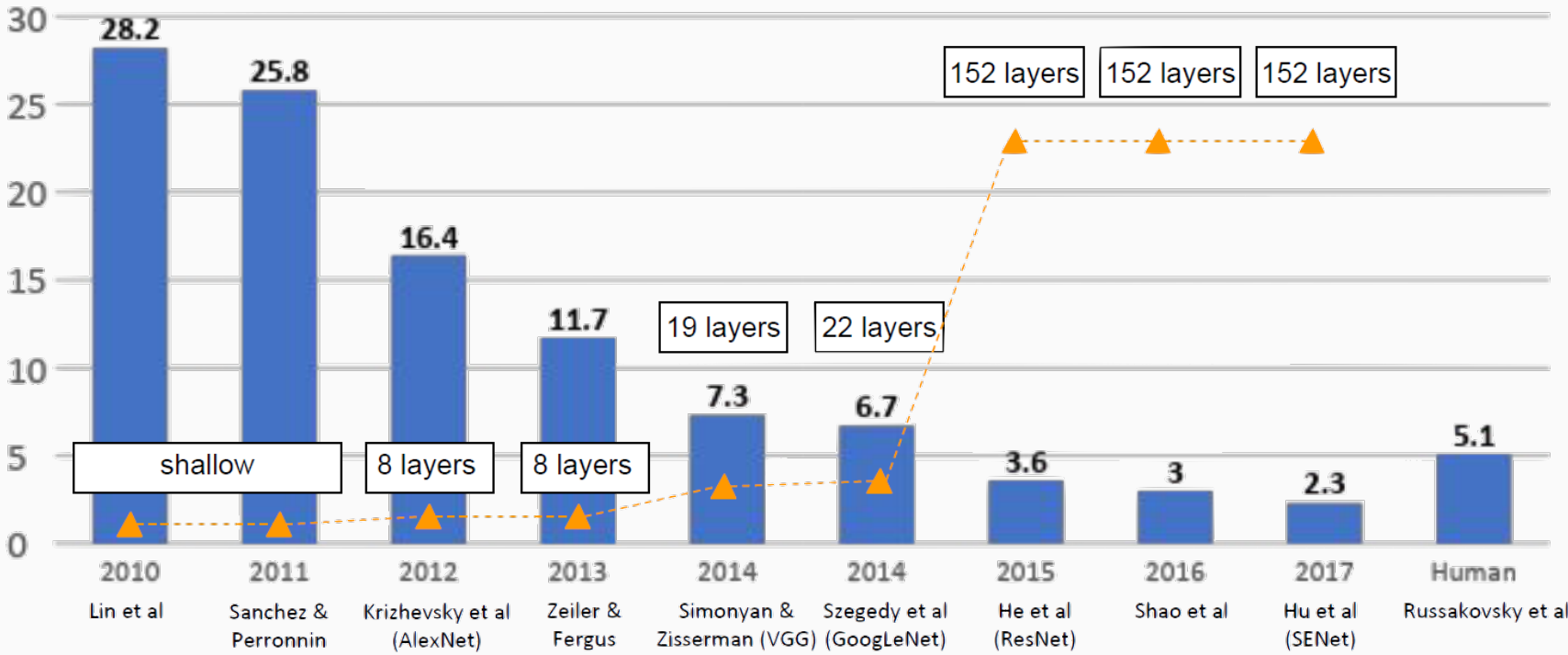
3.2 `ImageNet` Challenge

2012-1017 : Large Scale Visual Recognition Challenge

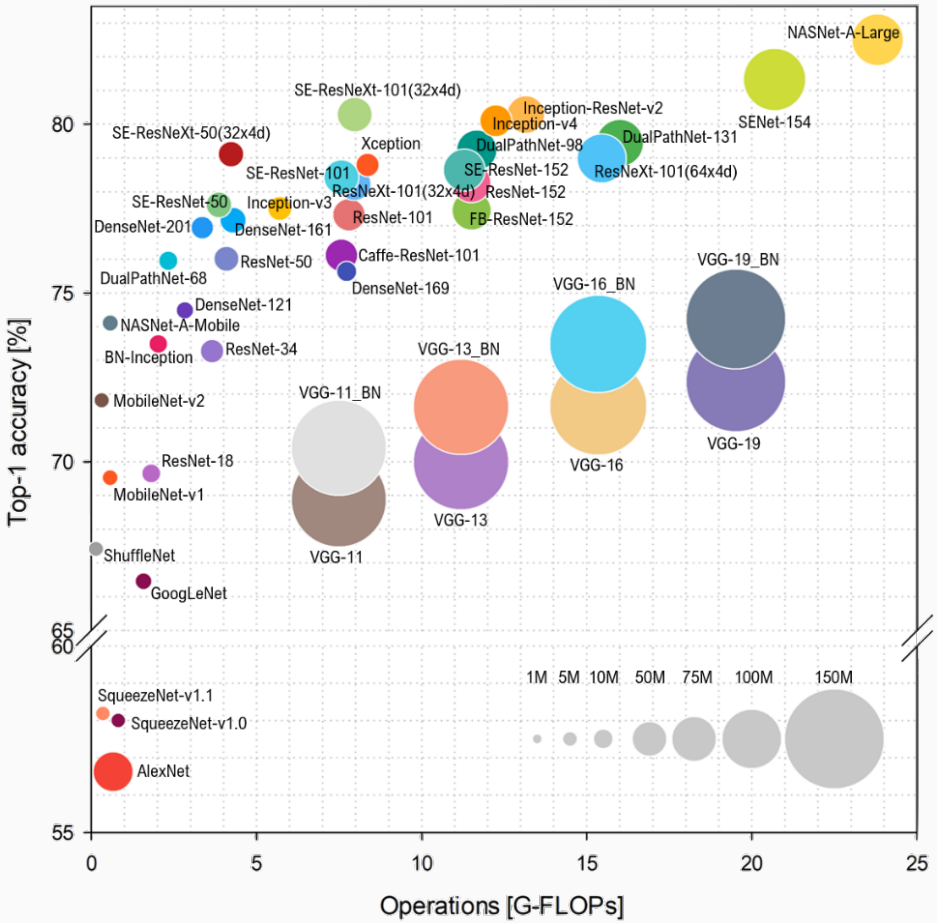
IMAGENET

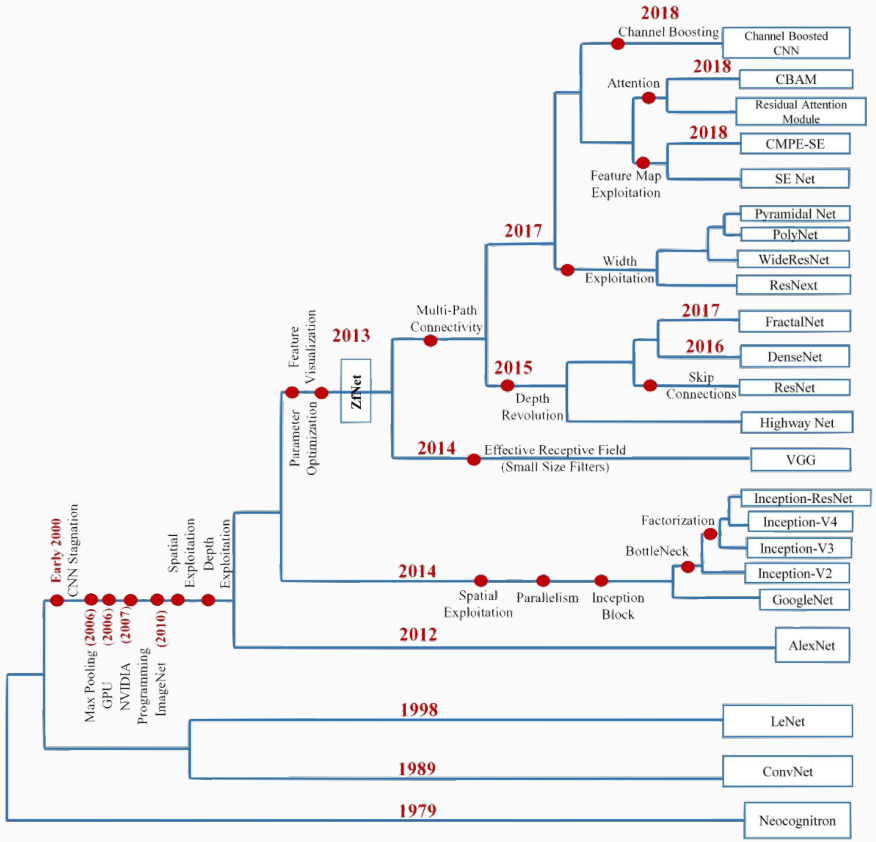
- 1000 object classes,
- 1 281 167 training images,
- 50 000 validation images,
- 100,000 test images.

ImageNet Large Scale Visual Recognition Challenge – ILSVRC



From 2015





[Bianco et al.] Benchmark Analysis of Representative Deep Neural Network Architectures (2018)

Transfer Learning

Representation Learning

Aim: *Image classification*

X

h

$\longrightarrow$

Y

"person"

Representation Learning

Aim: *Image classification*

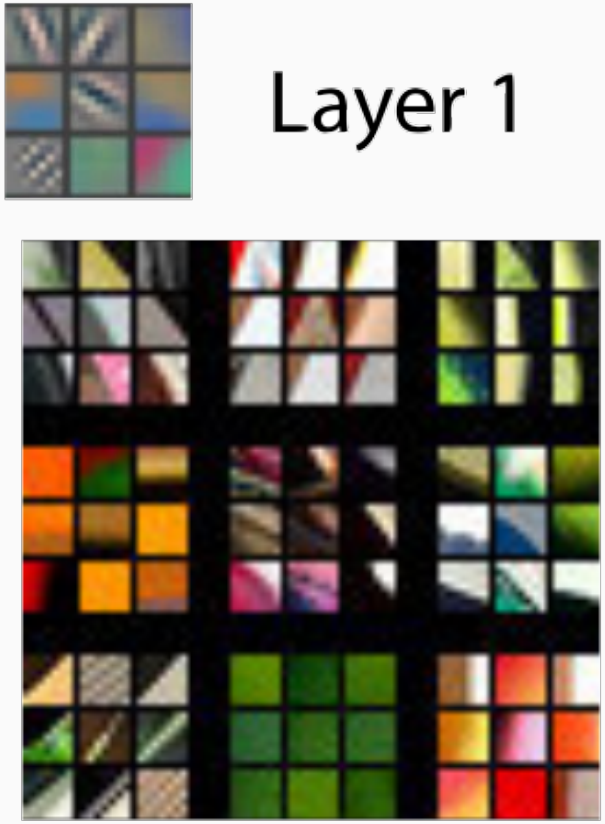
X $\xrightarrow{h}$ Y

"person"

Feature learning:

X $\xrightarrow{f}$ $f(X)$ $\xrightarrow{h'}$ Y

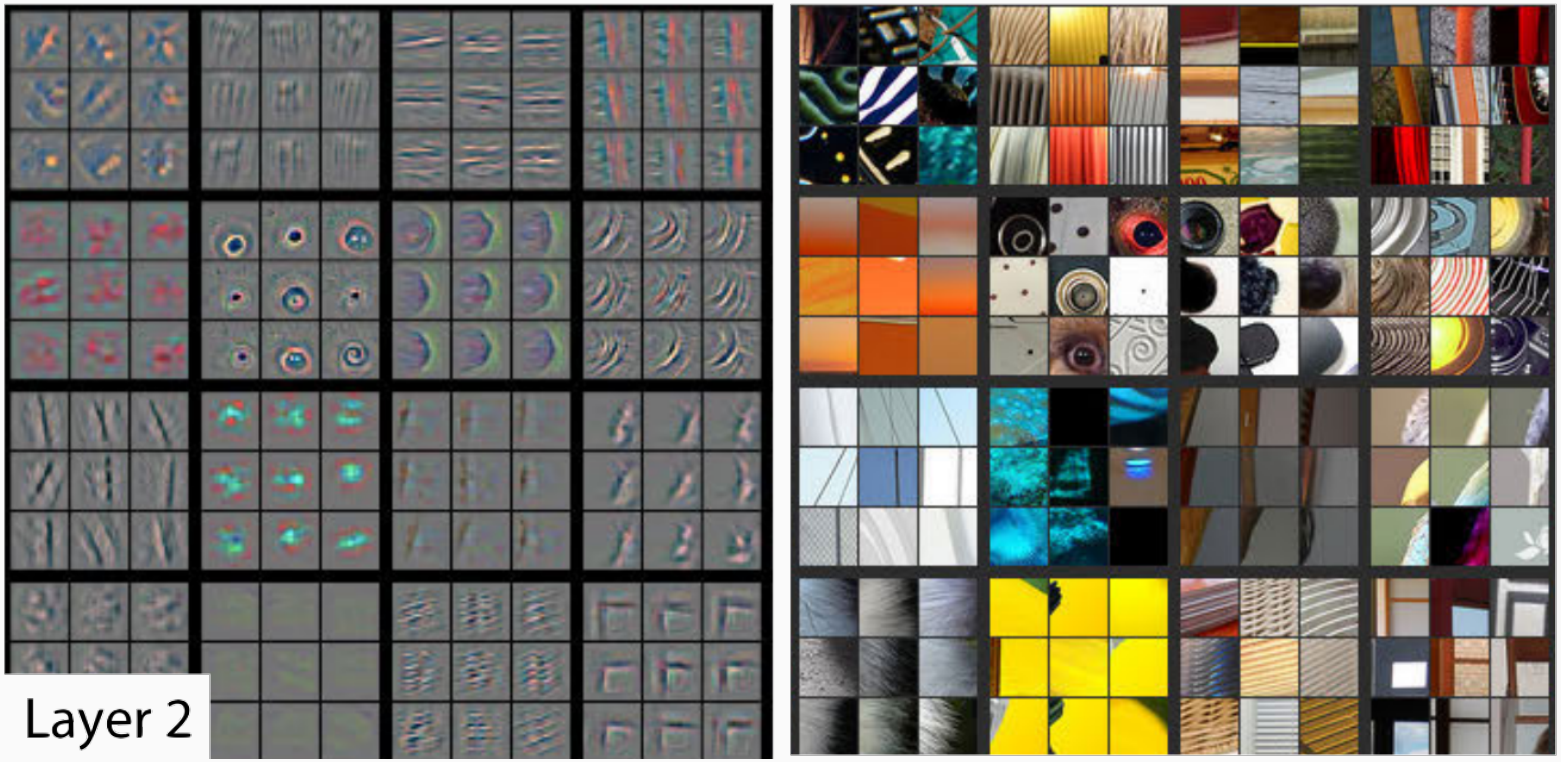
What do Convolutional Networks "Learn"?



- Example of filters learned on the **first layer** of a network (similar to `AlexNet`),
- For each filter, listing of the 9 image patches with the highest filter activation.

[Zeiler et al.] Visualizing and understanding convolutional networks.

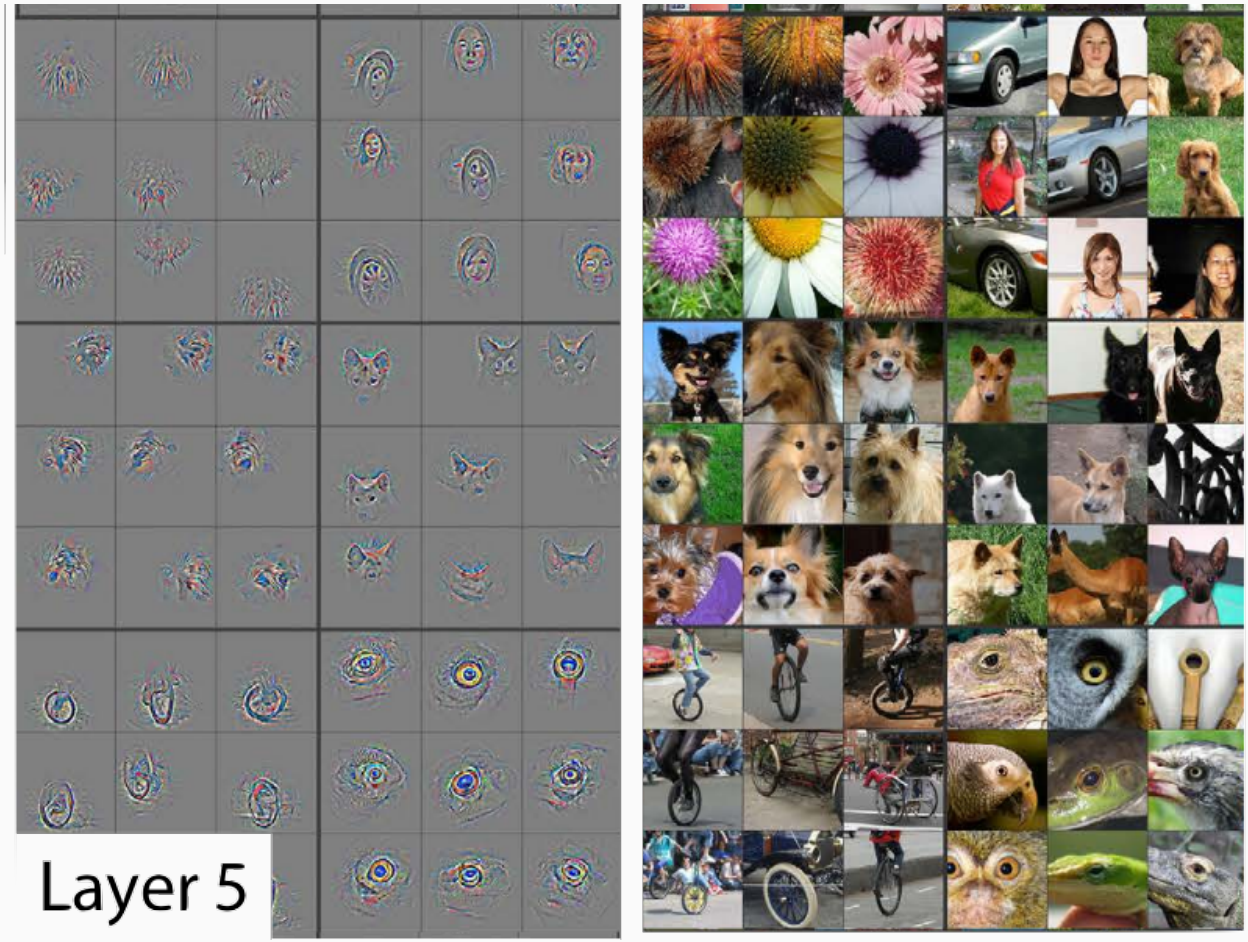
What do Convolutional Networks "Learn"?



- Same visualization as above, but for **second-layer** filters,
- Note that the filters on the left-hand side are not present in the network as they are, but that *this visual representation is reconstructed.*

[Zeiler et al.] Visualizing and understanding convolutional networks.

What do Convolutional Networks "Learn"?



The patterns detected are of a higher semantic level as you progress through the network layers.

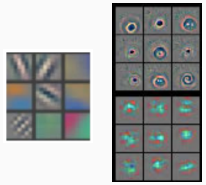
[Zeiler et al.] Visualizing and understanding convolutional networks.

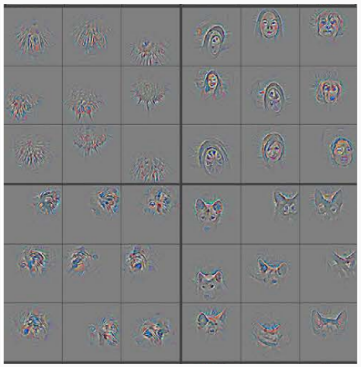
Representation Learning: An Interpretation of CNNs

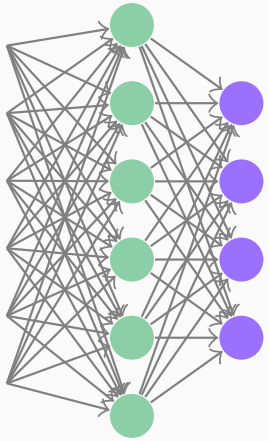
X $f(X)$ h' Y

features extraction

Predictor







"person"

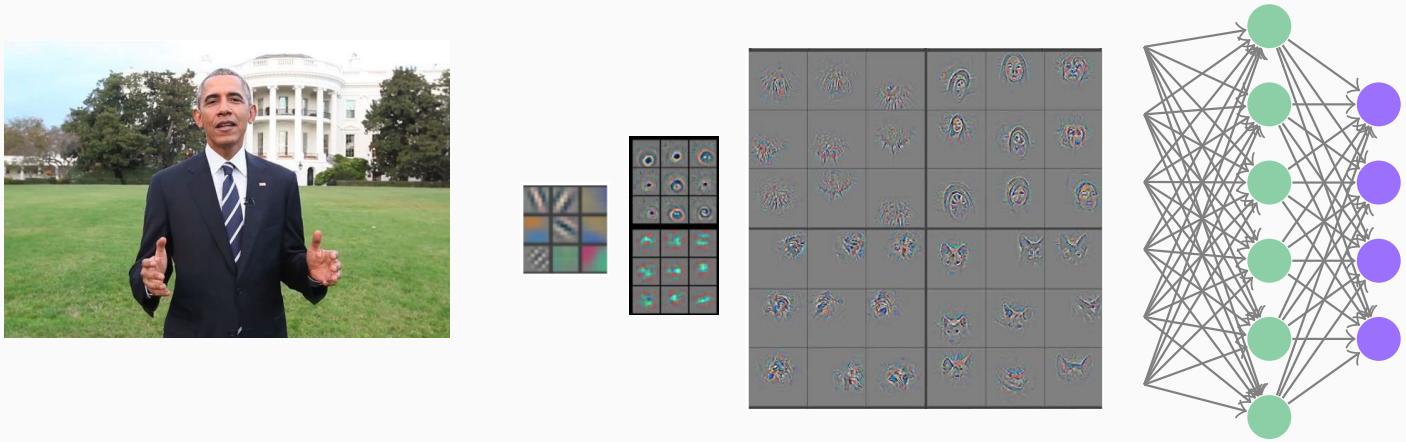
A CNN can be interpreted in terms of **representational learning**:

- The convolutional part is a feature extractor $f(X)$ ...
- ... and dense top layers are a predictor h'.

In this way, deep learning can learn features in addition to the predictor!

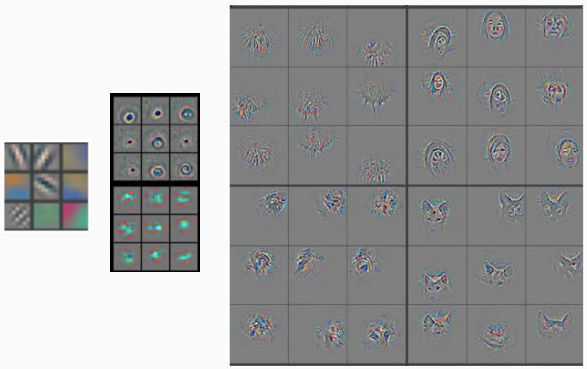
Transfer Learning

Feature Extraction



- Assume we have a CNN trained on a *large database*, such as `ImageNet` ($\approx$ 14 million images),

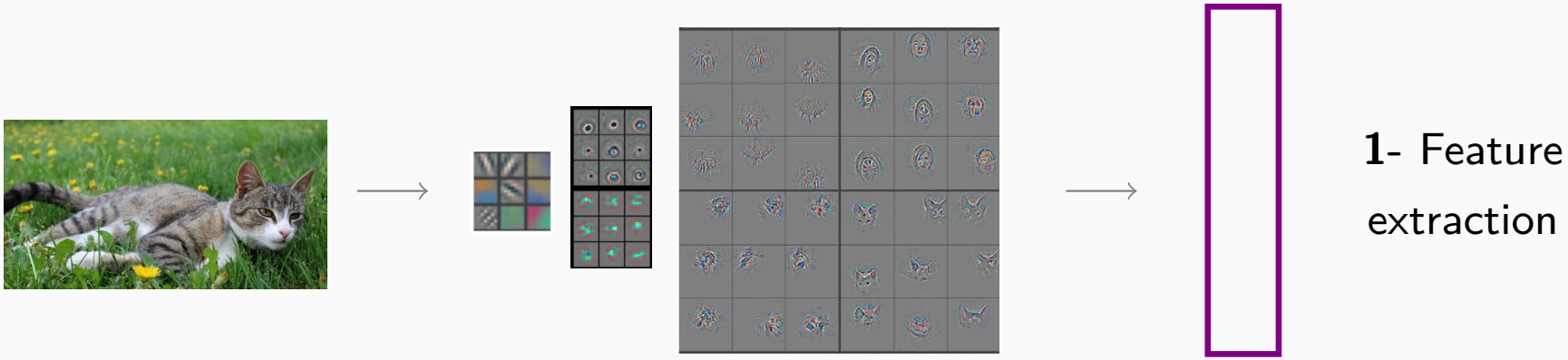
Feature Extraction



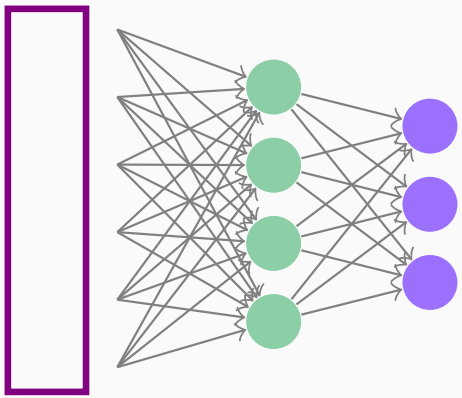
- Assume we have a CNN trained on a *large database*, such as `ImageNet` ($\approx$ 14 million images),
- The convolutional base, which acts as a feature extractor, can be extracted and reused for another task.

  $\rightsquigarrow$ ***Transfer learning***

“Static” Transfer Learning



1- Feature extraction

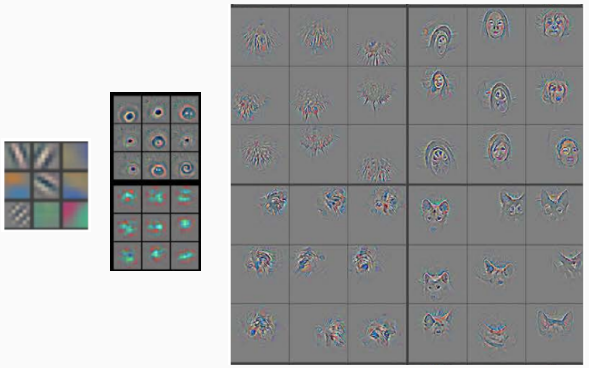


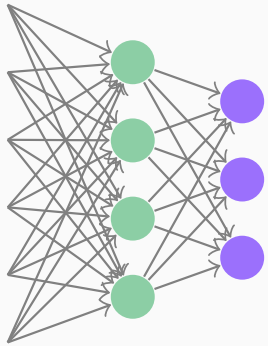
“cat”

2- Training a classifier of these features

We can use a pre-trained network to extract features from a new database, then train a simple classifier on these features.

Transfer Learning





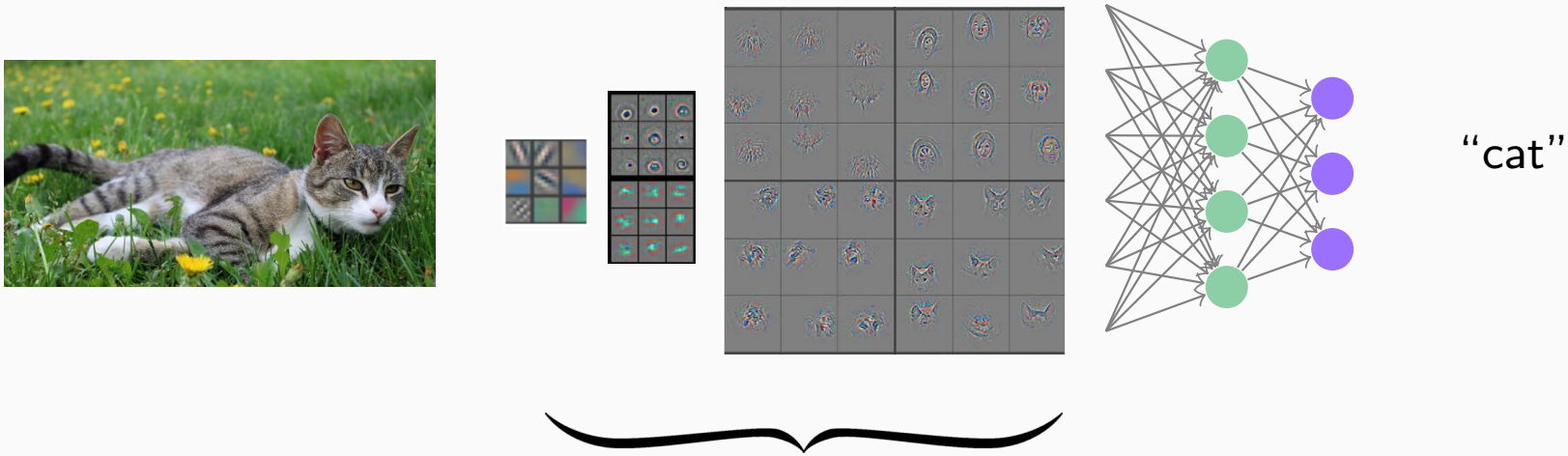
"cat"

1- Freeze feature extractor parameters

2- Training the last layers of the classifier

If feature extraction is included in the classifier, but its parameters are locked, the learning transfer supports data augmentation.

Fine-Tuning



Unfreeze extractor parameters

and fully re-train of the classifier

Fine tuning: Once the last layers of the classifier have been trained, the parameters of the convolutional base can be unlocked and the whole network re-trained, to "specify" it for the new task.

Attention! *The learning rate must be very small to avoid destroying the general filters obtained during pre-training.*