

CONVOLUTIONAL NEURAL NETWORKS

Vincent Barra
LIMOS, UMR 6158 CNRS, Université Clermont Auvergne

INTRODUCTION
○○○○○

LAYERS
○○○○○○○○

ARCHITECTURES
○○○○

UNDERSTANDING THE BLACK BOX
○○

INTRODUCTION

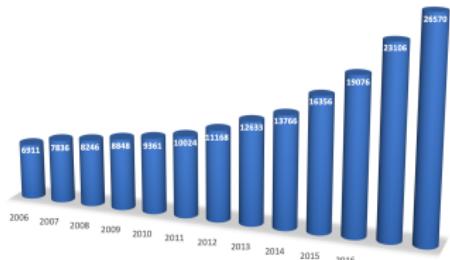
LAYERS

ARCHITECTURES

UNDERSTANDING THE BLACK BOX

CONVOLUTIONAL NEURAL NETWORKS (CNNs)

- ▶ dedicated to computer vision problems and more generally to any problem with a spatial (or sequential) structure.
- ▶ change the classical paradigm of image analysis



Number of publications per year (IEEE+Springer+Elsevier)

SOME APPLICATIONS

Deep is everywhere ;-)

- ▶ Medicine
 - ▶ Security
 - ▶ Internet
 - ▶ Art
 - ▶ NLP
 - ▶ Games
 - ▶ Images and videos analysis
 - ▶ Vocal synthesis
 - ▶ Pattern matching
 - ▶ Autonomous driving
 - ▶ Robotics
 - ▶ Domotics
 - ▶ Many More



MLP AND IMAGES

Why ?

- ▶ A question of size: a 512×512 RGB image = 786 432 values.
Let's build a 1 hidden layer MLP, producing an image from an image.
Then the number of parameters to train is approximatively $6.19.10^{11}$,
more than 1Tb in memory → untractable.
- ▶ A question of information: pixel values may be related (correlated) to
the values surrounding the pixel position → a 1D representation
cannot easily handle this.
- ▶ A question of invariance: a representation meaningful at a certain
location should be used everywhere

MLP AND IMAGES

- ▶ A 2-layer MLP can easily classify MNIST data, BUT images are vectorized : $28 \times 28 \rightarrow 784 \times 1$
- ▶ What if pixels values are shuffled ?



Original images

Shuffled images

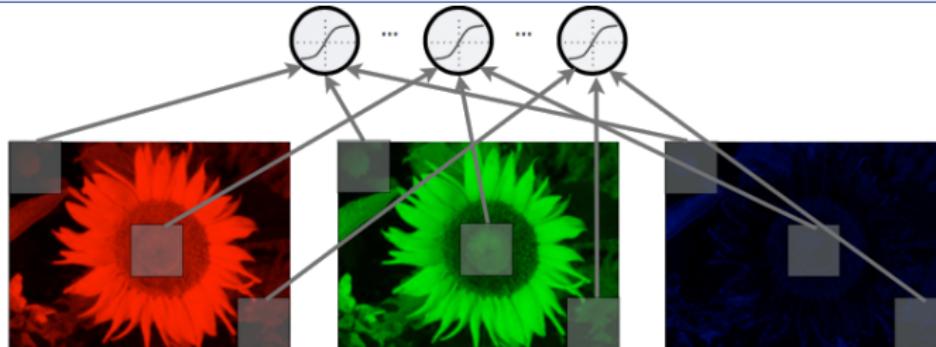
Another set of 2,6 and 9

How can you recognize the digits ???

CONVOLUTIONAL NEURAL NETWORKS (CNN)

Key ideas of CNN

- ▶ *Local connectivity*: each neuron is connected to a patch in the image, not to the whole set of pixels.
- ▶ *Convolution layers* apply the same linear transformation locally everywhere while preserving the signal structure.
- ▶ *Parameter sharing* across patches, allows to be equivariant to translation.
- ▶ *Pooling layers* allows to be pseudo invariant to local translations and noise.



CONVOLUTION

Definition

In 1D, convolution (cross-correlation) between f and g :

$$f \circledast g(x) = \sum_{y+z=x} f(y).g(z) = \sum_y f(y)g(x+y)$$

In 2D: $g(x, y) = w \circledast f(x, y) = \sum_m \sum_n w(m, n).f(x + m, y + n)$

w : convolution Kernel (or filter) applied to image f .

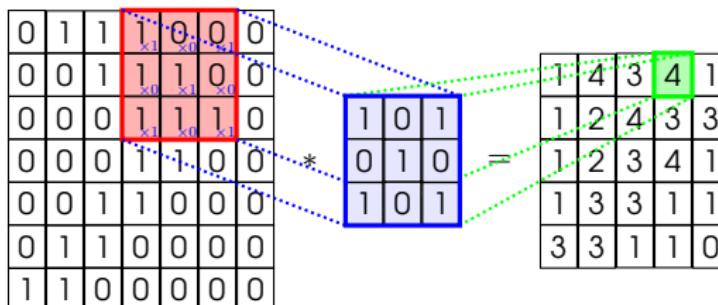
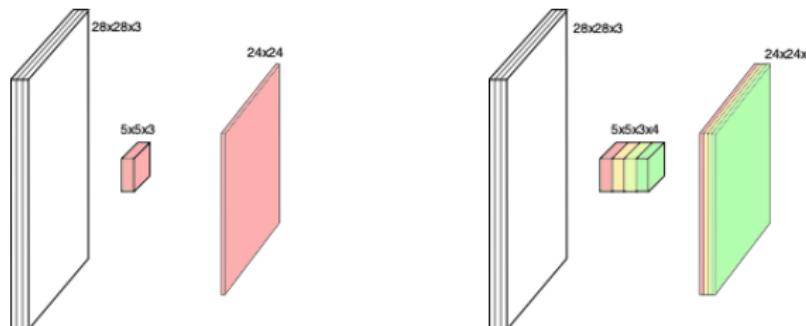


FIGURE: Exemple de convolution discrète

CONVOLUTION

- ▶ For multichannel images (eg color images), convolutions are usually computed for each channel and summed.
 - ▶ Multiple convolutions
- ⇒ Convolutions applied to *tensors*.



$$w \circledast f(x, y) = \sum_{i=0}^2 w_c \circledast f_c(x, y) \quad \text{4 kernels } 5 \times 5 \text{ on color images}$$

CONVOLUTION LAYER

Let

- ▶ x be the input tensor of size $C \times H \times W$.
- ▶ k be a kernel of size $C \times h \times w$ ($h \times w = \text{receptive field}$)
- ▶ y be the output tensor (the *feature map*), resulting from the convolutions.

A convolution layer implements K convolutions, using K kernels k .

y is of size $K \times (H - h + 1) \times (W - w + 1)$ and

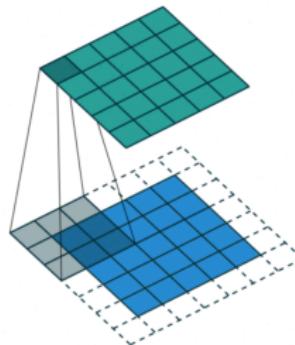
$$(\forall k) \quad y(i, j) = \sum_{c=0}^{C-1} (x_c \circledast k_c)(i, j) + b_{ij}$$

The k 's and b are shared parameters to learn.

CONVOLUTION LAYER

Additional Parameters

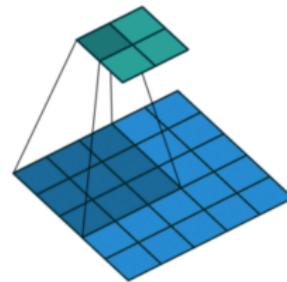
- ▶ *padding*: size of a zeroed frame added around the input
→ controls the spatial dimension of the feature map
- ▶
- ▶



CONVOLUTION LAYER

Additional Parameters

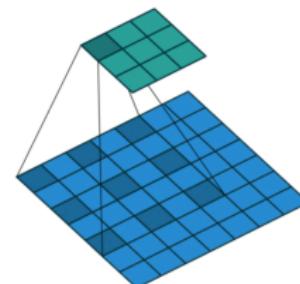
- ▶ *padding*: size of a zeroed frame added around the input
→ controls the spatial dimension of the feature map
- ▶ *stride*: step size when moving the kernel across the signal
→ reduces the spatial dimension of the feature map
- ▶



CONVOLUTION LAYER

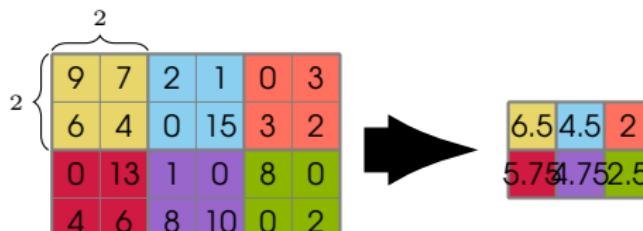
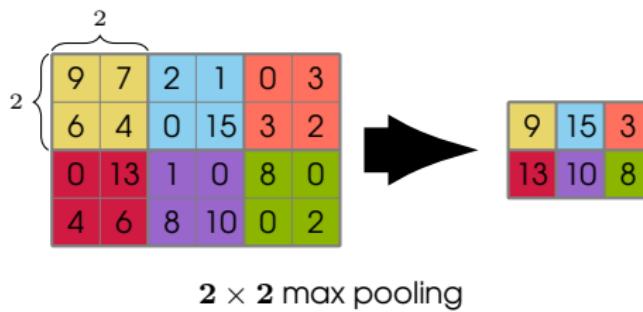
Additional Parameters

- ▶ *padding*: size of a zeroed frame added around the input
 - controls the spatial dimension of the feature map
- ▶ *stride*: step size when moving the kernel across the signal
 - reduces the spatial dimension of the feature map
- ▶ *dilation*: modulates the expansion of the kernels without adding weights.
 - increases the units receptive field size without increasing the number of parameters



POOLING LAYER

Pooling \approx downsampling: considers a receptive field of size $h \times w$ and replaces the set of values by the max, the mean (main pooling operations)



POOLING LAYER

- ▶ No parameter to learn !
- ▶ Pooling layers provide invariance to any permutation inside one cell.
- ▶ pseudo-invariance to local translations.
- ▶ Interesting if we care more about the presence of a pattern rather than its exact position.

OTHER TYPICAL LAYERS

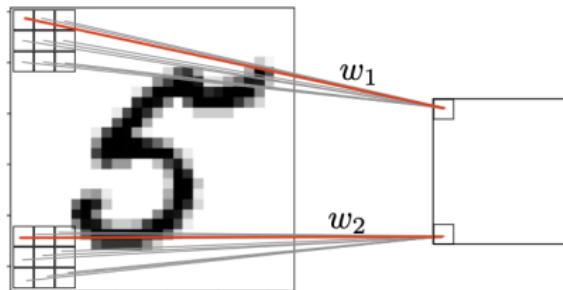
- ▶ *Activation*: add an activation function after the output of a layer (can be integrated in the layer itself).
- ▶ *Dropout*: randomly sets input units to 0 with a given frequency at each step during training. Helps prevent overfitting.
- ▶ *Batch normalization*: applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1. Works differently during training and during inference. Fasten the training process.
- ▶ *Fully connected* = MLP

Additional ressource

See Slides “Normalization” and “Dropout”.

BACKPROPAGATION IN CNN

Same concept as for MLP: multivariable chain rule, with weight sharing constraint.

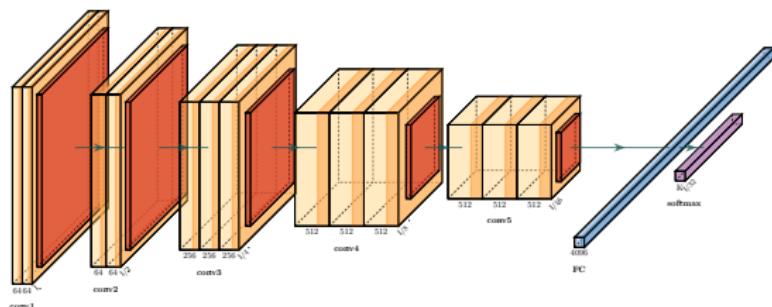


$$w_1 = w_2 = w_1 - \frac{\eta}{2} \left(\frac{\partial \mathcal{L}}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial w_2} \right)$$

Additional resource

See slides “Optimization for deep Learning” and “weight initialization”

- ▶ Classical architectures: succession of (Conv -Activation -Pooling) blocks + fully connected layer(s) + softmax (for classification)



- ▶ Since 2014, several other architectures proposed.

See...

Lectures “CNN architectures” & “Transfer Learning”.

STATE OF THE ART

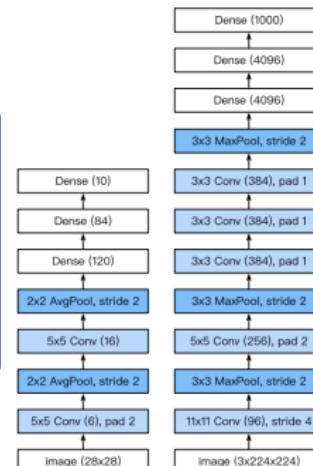
Method	# Params	Extra Data	ImageNet		ImageNet-ReaL [6] Precision@1
			Top-1	Top-5	
ResNet-50 [24]	26M	—	76.0	93.0	82.94
ResNet-152 [24]	60M	—	77.8	93.8	84.79
DenseNet-264 [28]	34M	—	77.9	93.9	—
Inception-v3 [62]	24M	—	78.8	94.4	83.58
Xception [11]	23M	—	79.0	94.5	—
Inception-v4 [61]	48M	—	80.0	95.0	—
Inception-resnet-v2 [61]	56M	—	80.1	95.1	—
ResNeXt-101 [78]	84M	—	80.9	95.6	85.18
PolyNet [87]	92M	—	81.3	95.8	—
SENet [27]	146M	—	82.7	96.2	—
NASNet-A [90]	89M	—	82.7	96.2	82.56
AmoebaNet-A [52]	87M	—	82.8	96.1	—
PNASNet [39]	86M	—	82.9	96.2	—
AmoebaNet-C + AutoAugment [12]	155M	—	83.5	96.5	—
GPipe [29]	557M	—	84.3	97.0	—
EfficientNet-B7 [63]	66M	—	85.0	97.2	—
EfficientNet-B7 + FixRes [70]	66M	—	85.3	97.4	—
EfficientNet-L2 [63]	480M	—	85.5	97.5	—
ResNet-50 Billion-scale SSL [79]	26M	3.5B labeled Instagram	81.2	96.0	—
ResNeXt-101 Billion-scale SSL [79]	193M	3.5B labeled Instagram	84.8	—	—
ResNeXt-101 WSL [42]	829M	3.5B labeled Instagram	85.4	97.6	88.19
FixRes ResNeXt-101 WSL [69]	829M	3.5B labeled Instagram	86.4	98.0	89.73
Big Transfer (BiT-L) [33]	928M	300M labeled JFT	87.5	98.5	90.54
Noisy Student (EfficientNet-L2) [77]	480M	300M unlabeled JFT	88.4	98.7	90.55
Noisy Student + FixRes [70]	480M	300M unlabeled JFT	88.5	98.7	—
Vision Transformer (ViT-H) [14]	632M	300M labeled JFT	88.55	—	90.72
EfficientNet-L2-NoisyStudent + SAM [16]	480M	300M unlabeled JFT	88.6	98.6	—
Meta Pseudo Labels (EfficientNet-B6-Wide)	390M	300M unlabeled JFT	90.0	98.7	91.12
Meta Pseudo Labels (EfficientNet-L2)	480M	300M unlabeled JFT	90.2	98.8	91.02

Source: Meta Pseudo Labels, Hieu Pham et al. (01/2021)

EXAMPLES

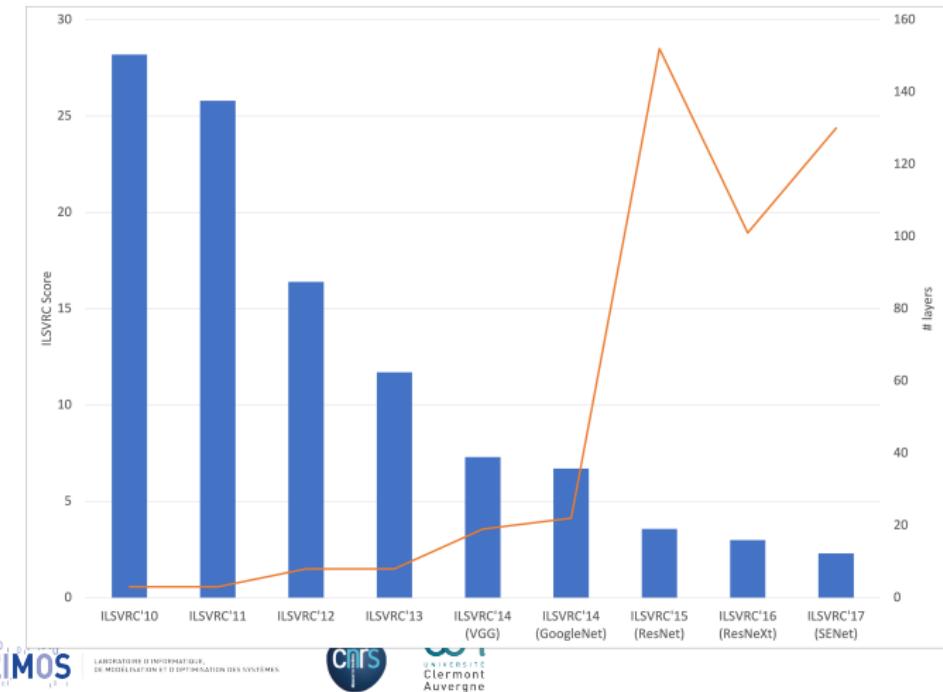
Two classical (and old) architectures

- ▶ LeNet-5 (LeCun et al, 1998): 61 706 trainable parameters
- ▶ AlexNet (Krizhevsky et al, 2012): 61 100 840 trainable parameters



THE DEEPER, THE BETTER ?

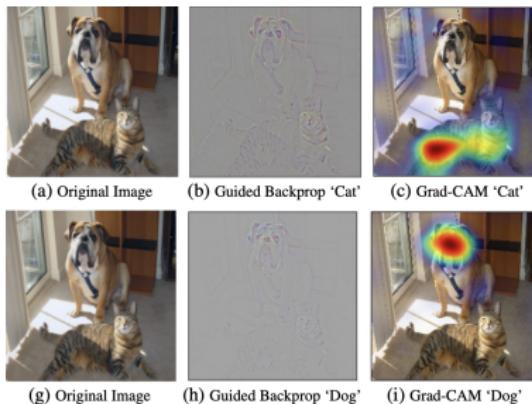
Image Large Scale Visual Recognition Challenge (Classification task)



INSIDE THE CNN

What we can do

- ▶ filters → images
- ▶ distributions of activations on a batch of samples
- ▶ gradient of the response with respect to the input
- ▶ create a synthetic image that maximize a given filter



INSIDE THE CNN

It seems that

- ▶ the first layers encode edges, directions and colorimetric properties
- ▶ directions and colors are combined to "textures"
- ▶ these patterns combine to more complex patterns → semantic

