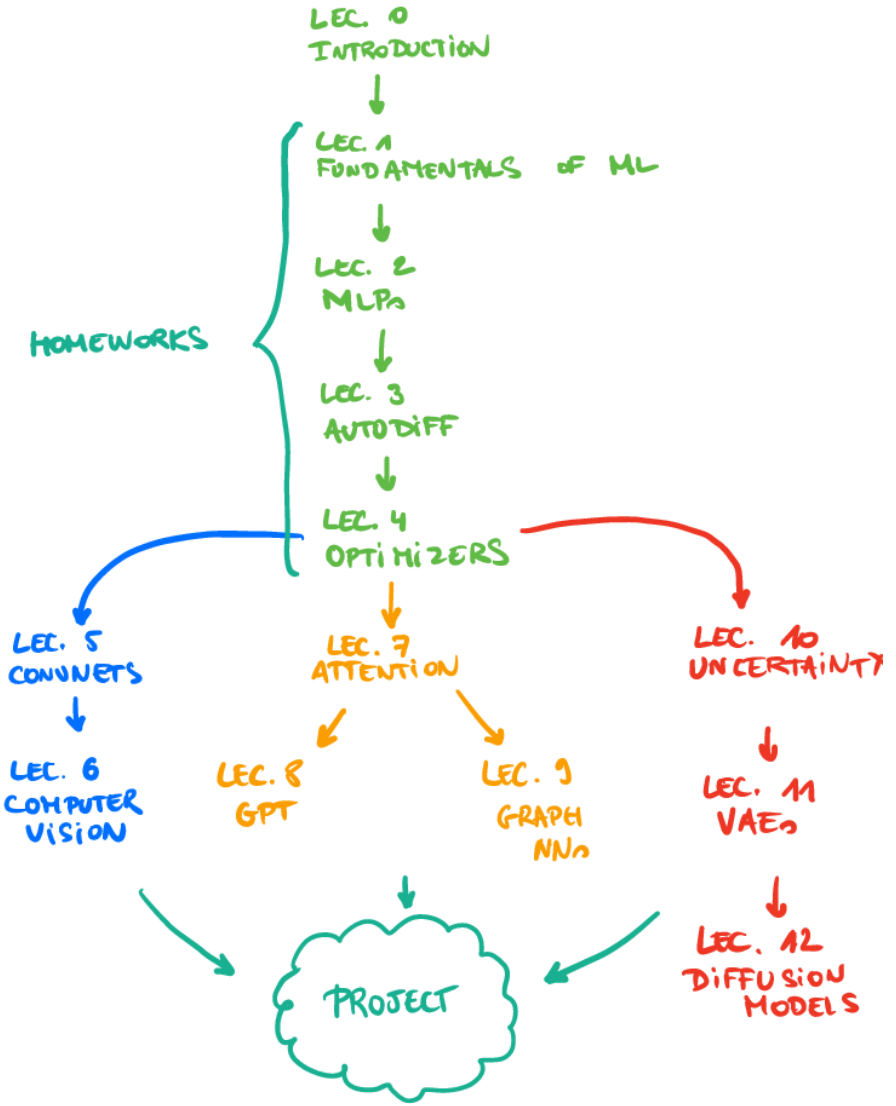
Deep Learning

Lecture 5: Convolutional networks

Prof. Gilles Louppe
g.louppe@uliege.be

LIÈGE université


LEC. 0
INTRODUCTION
LEC. 1
FUNDAMENTALS OF ML
LEC. 2
MLPs
LEC. 3
AUTODIFF
LEC. 4
OPTIMIZERS
HOMEWORKS
LEC. 5
CONVNETS
LEC. 6
COMPUTER VISION
LEC. 7
ATTENTION
LEC. 8
GPT
LEC. 9
GRAPH NNs
LEC. 10
UNCERTAINTY
LEC. 11
VAEs
LEC. 12
DIFFUSION MODELS
PROJECT

Today

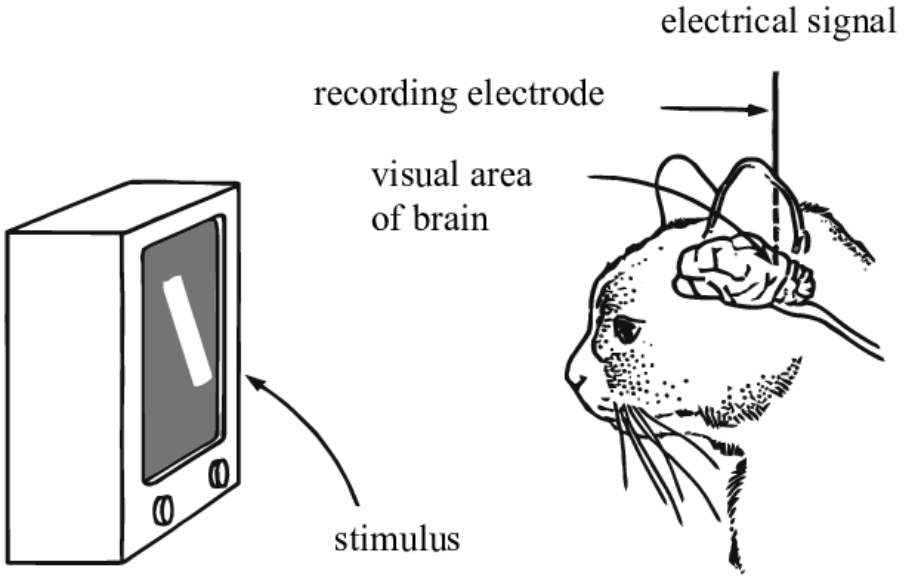
How to make neural networks see?

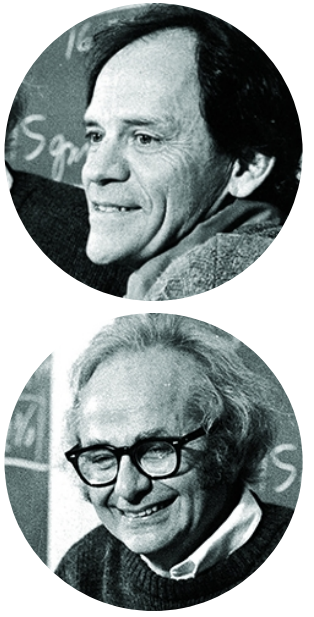
- Visual perception
- Convolutions
- Pooling
- Convolutional networks
- Under the hood

Visual perception

Visual perception

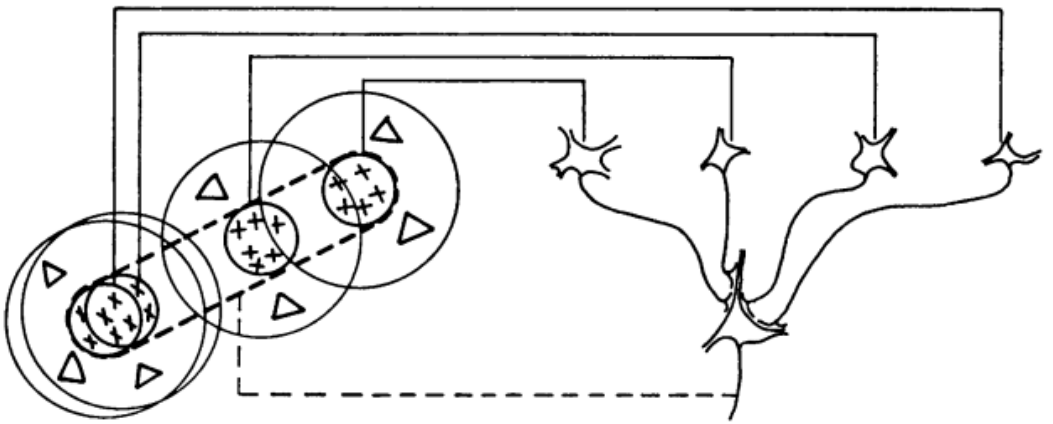
In 1959-1962, David Hubel and Torsten Wiesel identify the neural basis of information processing in the visual system. They are awarded the Nobel Prize of Medicine in 1981 for their discovery.





Hubel and Wiesel

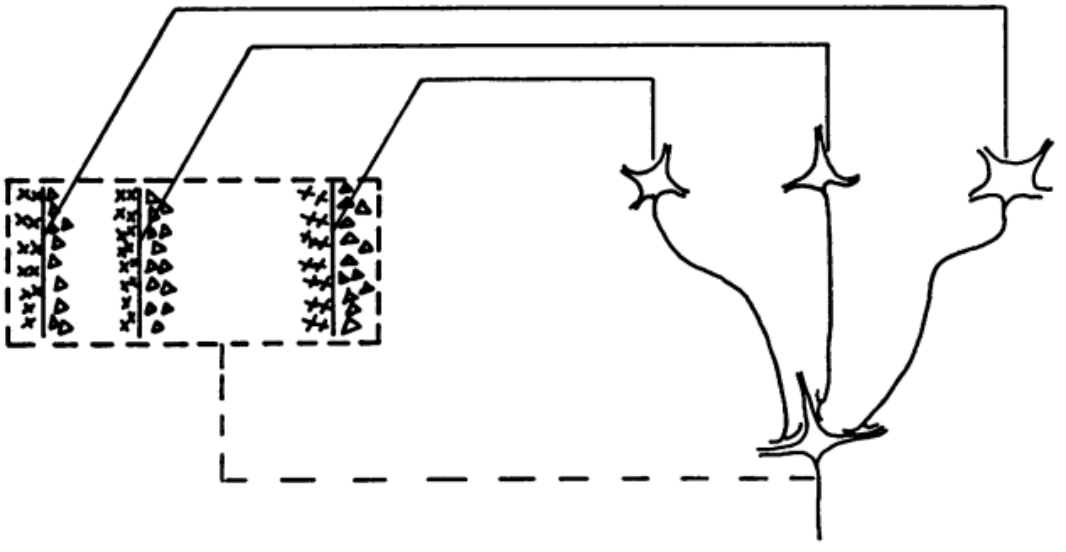
Hubel and Wiesel



Text-fig. 19. Possible scheme for explaining the organization of simple receptive fields. A large number of lateral geniculate cells, of which four are illustrated in the upper right in the figure, have receptive fields with 'on' centres arranged along a straight line on the retina. All of these project upon a single cortical cell, and the synapses are supposed to be excitatory. The receptive field of the cortical cell will then have an elongated 'on' centre indicated by the interrupted lines in the receptive-field diagram to the left of the figure.

—

Credits: Hubel and Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, 1962.



Text-fig. 20. Possible scheme for explaining the organization of complex receptive fields. A number of cells with simple fields, of which three are shown schematically, are imagined to project to a single cortical cell of higher order. Each projecting neurone has a receptive field arranged as shown to the left: an excitatory region to the left and an inhibitory region to the right of a vertical straight-line boundary. The boundaries of the fields are staggered within an area outlined by the interrupted lines. Any vertical-edge stimulus falling across this rectangle, regardless of its position, will excite some simple-field cells, leading to excitation of the higher-order cell.

—

Credits: Hubel and Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, 1962.

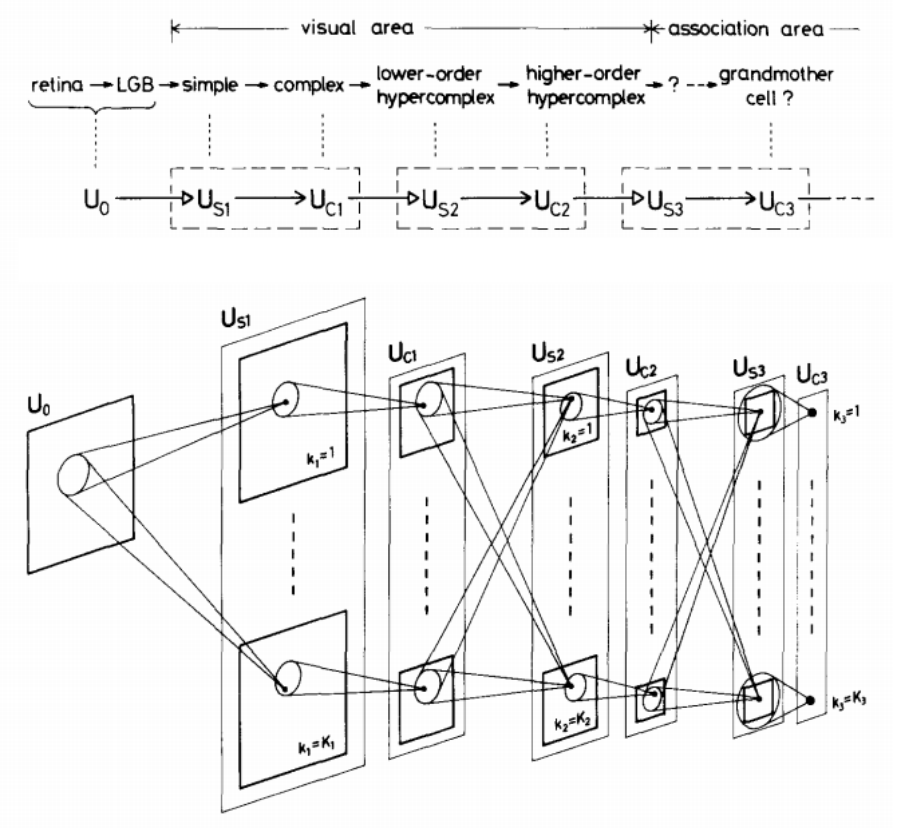
Inductive biases

Can we equip neural networks with inductive biases tailored for vision?

- Locality
- Invariance to translation
- Hierarchical compositionality

Neocognitron

In 1980, Fukushima proposes a direct neural network implementation of the hierarchy model of the visual nervous system of Hubel and Wiesel.



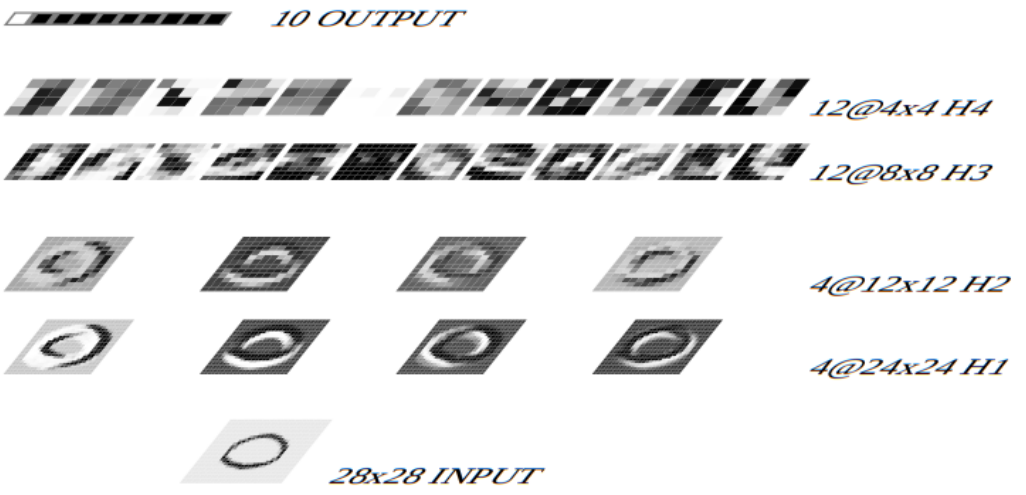
- Built upon convolutions and enables the composition of a feature hierarchy.
- Biologically-inspired training algorithm, which proves to be largely inefficient.

—

Credits: Kunihiko Fukushima, Neocognitron: A Self-organizing Neural Network Model, 1980.

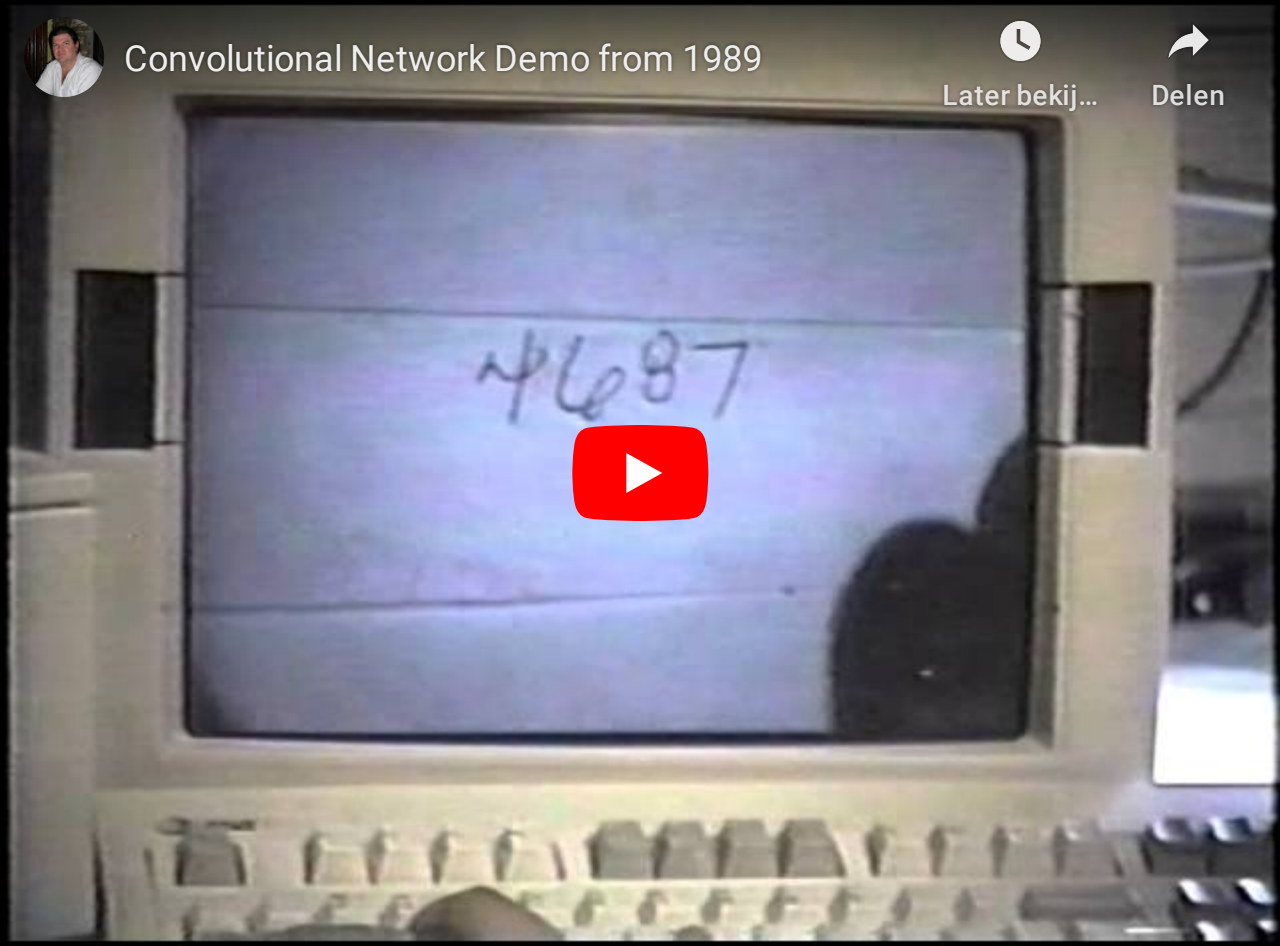
Convolutional networks

In 1990, LeCun trains a convolutional network by backpropagation. He advocates for end-to-end feature learning in image classification.



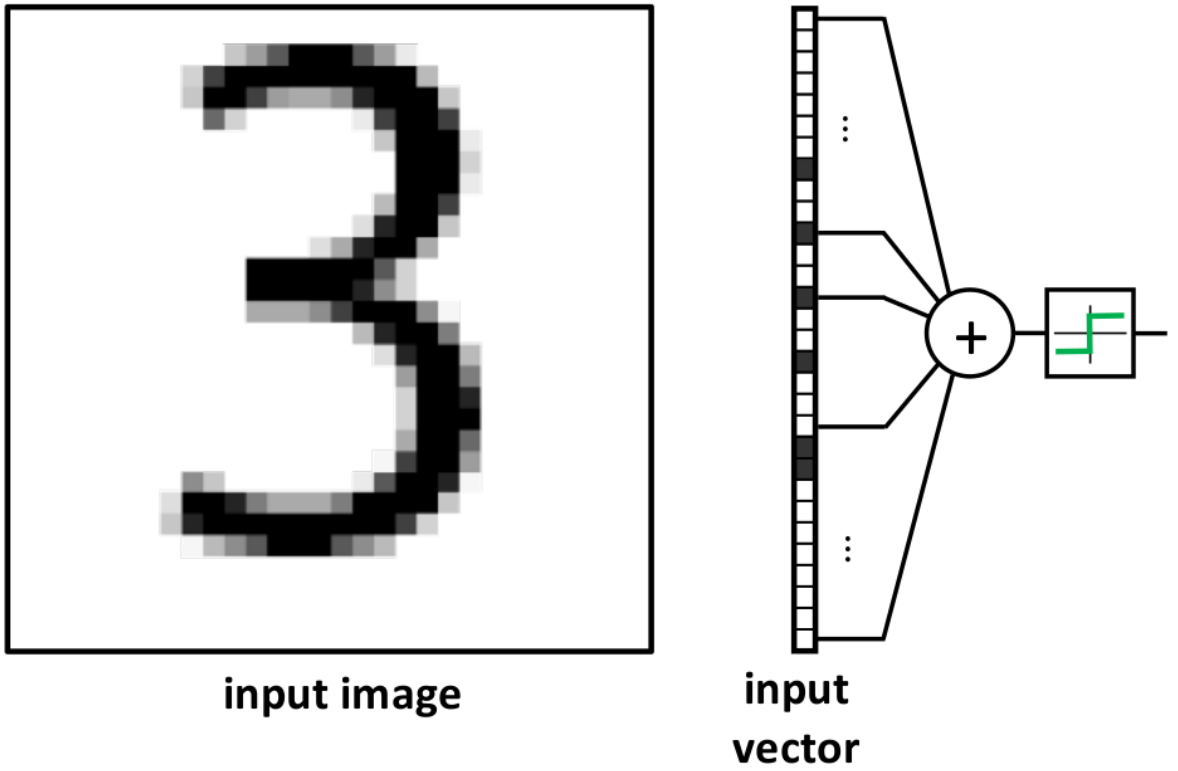
—

Credits: LeCun et al, Handwritten Digit Recognition with a Back-Propagation Network, 1990.



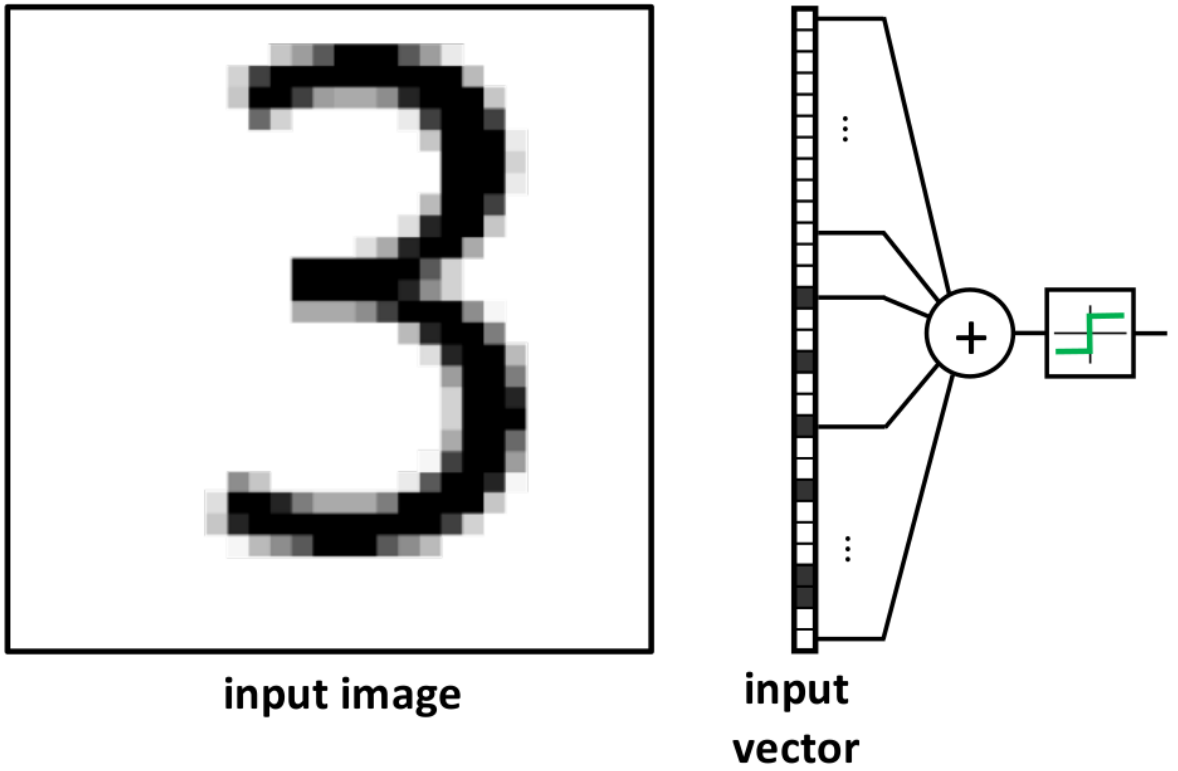
LeNet-1 (LeCun et al, 1993)

Convolutions



—

Credits: Francois Fleuret, EE559 Deep Learning, EPFL; Michael Bronstein, Geometric Deep Learning, 2020.



—

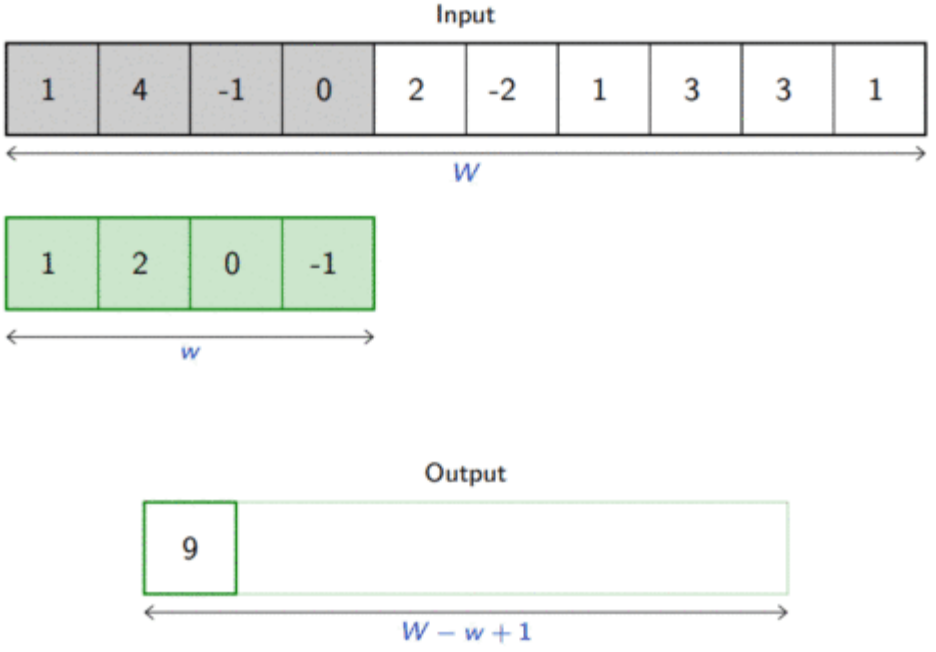
Credits: Francois Fleuret, EE559 Deep Learning, EPFL; Michael Bronstein, Geometric Deep Learning, 2020.

If they were handled as normal "unstructured" vectors, high-dimensional signals such as sound samples or images would require models of intractable size.

Large signals have some "invariance in translation". A representation meaningful at a certain location should be used everywhere.

Convolutions

A convolution layer applies the same linear transformation locally everywhere while preserving the signal structure.



1D convolution

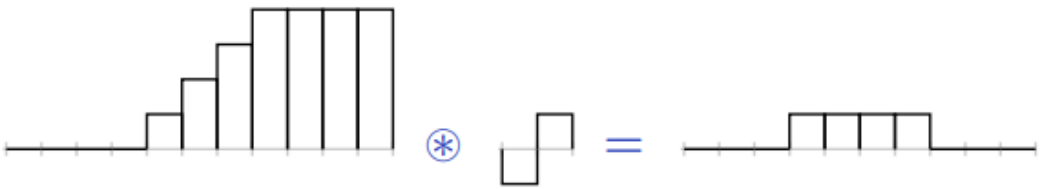
For one-dimensional tensors, given an input vector $\mathbf{x} \in \mathbb{R}^W$ and a convolutional kernel $\mathbf{u} \in \mathbb{R}^w$, the discrete convolution $\mathbf{x} \circledast \mathbf{u}$ is a vector of size $W - w + 1$ such that

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} u_m.$$

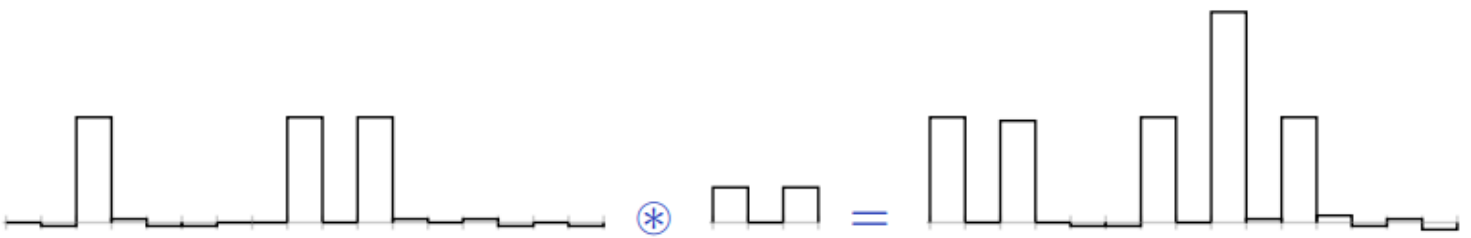
Technically, $\circledast$ denotes the cross-correlation operator. However, most machine learning libraries call it convolution.

Convolutions can implement differential operators:

$$(0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4) \circledast (-1, 1) = (0, 0, 0, 1, 1, 1, 1, 0, 0, 0)$$



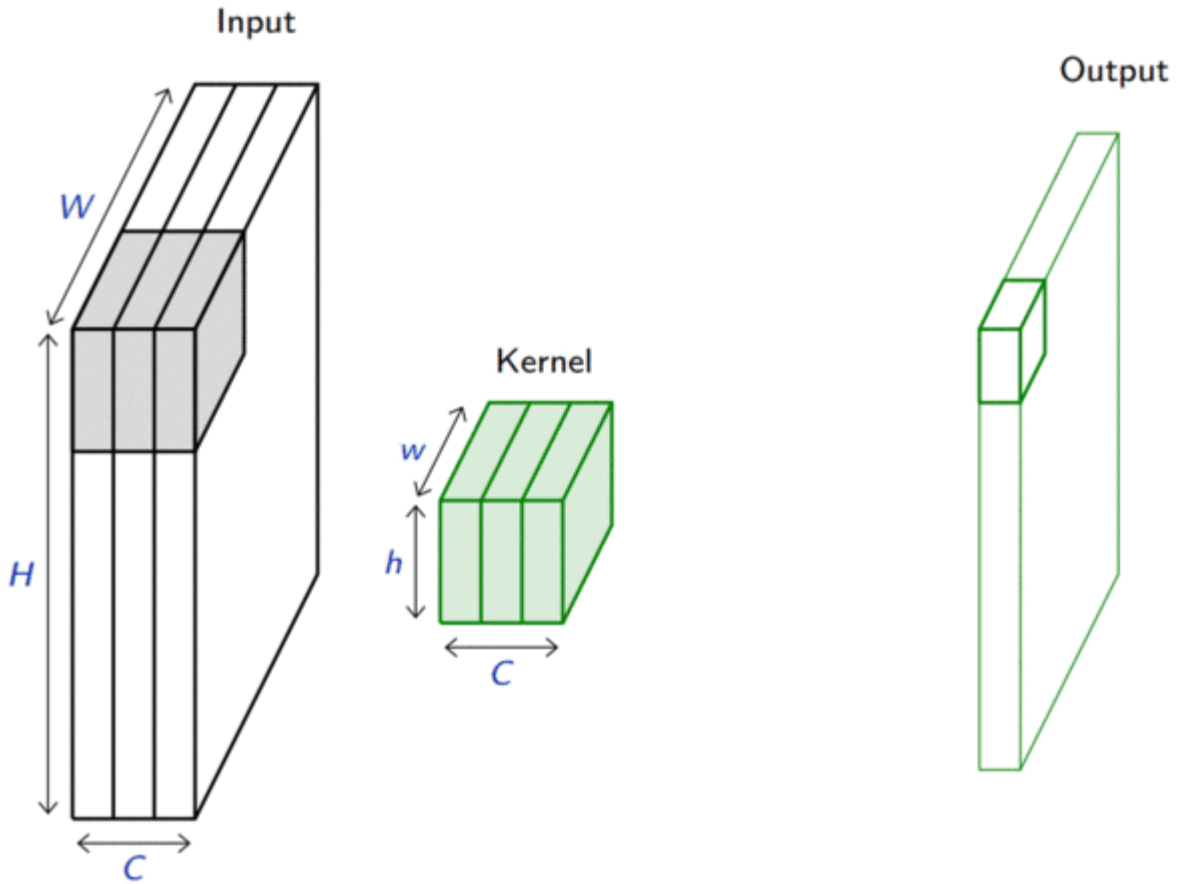
or crude template matchers:



n-D convolution

Convolutions generalize to multi-dimensional tensors:

- In its most usual form, a convolution takes as input a 3D tensor $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$, called the input feature map, where C is the number of input channels.
- A kernel $\mathbf{u} \in \mathbb{R}^{C \times h \times w}$ slides across the input feature map, along its height and width. The size $h \times w$ is the size of the receptive field.
- At each location, the element-wise product between the kernel and the input elements it overlaps is computed and the results are summed up.


Input
W
H
C
Kernel
w
h
C
Output

—

- The final output $\mathbf{o}$ is a 2D tensor of size $(H - h + 1) \times (W - w + 1)$ called the output feature map and such that:

$$\mathbf{o}_{j,i} = \sum_{c=0}^{C-1} (\mathbf{x}_c \circledast \mathbf{u}_c)[j, i] = \sum_{c=0}^{C-1} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{c,n+j,m+i} \mathbf{u}_{c,n,m}$$

  where $\mathbf{u}$ is a shared parameter to learn.

- D convolutions can be applied in the same way to produce a $D \times (H - h + 1) \times (W - w + 1)$ feature map, where D is the number of output channels.

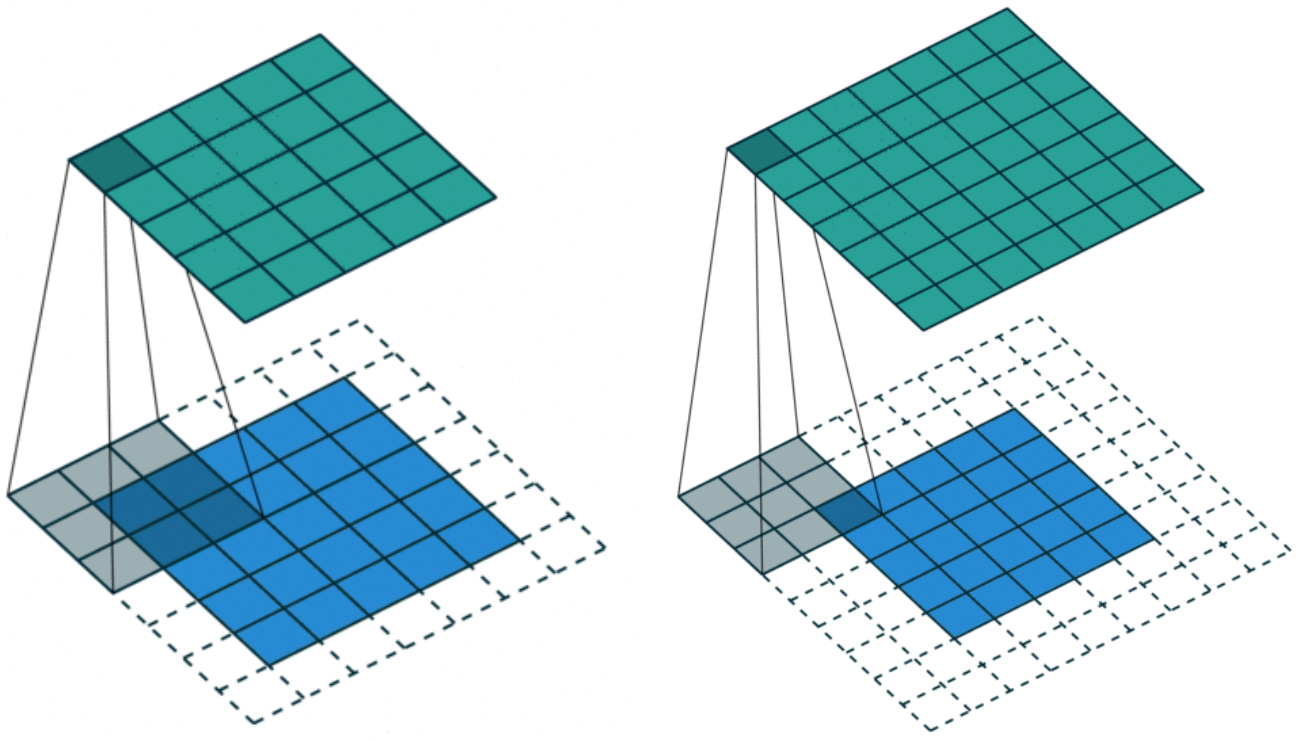
- Swiping across channels with a 3D convolution usually makes no sense, unless the channel index has some metric meaning.

Convolutions have three additional parameters:

- The padding specifies the size of a zeroed frame added arount the input.
- The stride specifies a step size when moving the kernel across the signal.
- The dilation modulates the expansion of the filter without adding weights.

Padding

Padding is useful to control the spatial dimension of the output feature map, for example to keep it constant across layers.

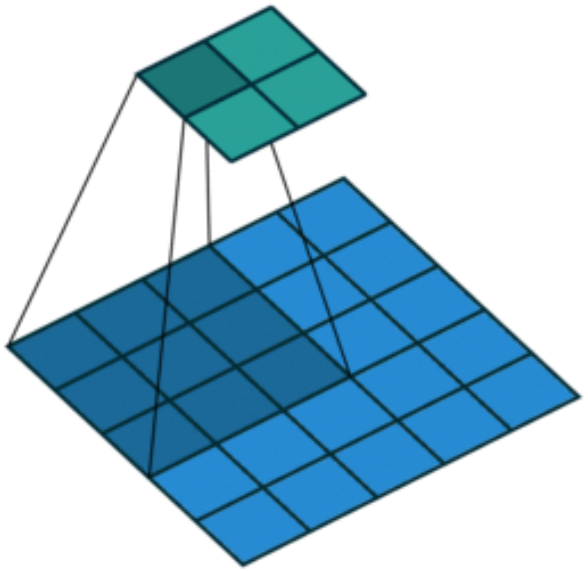


—

Credits: Dumoulin and Visin, A guide to convolution arithmetic for deep learning, 2016.

Strides

Stride is useful to reduce the spatial dimension of the feature map by a constant factor.



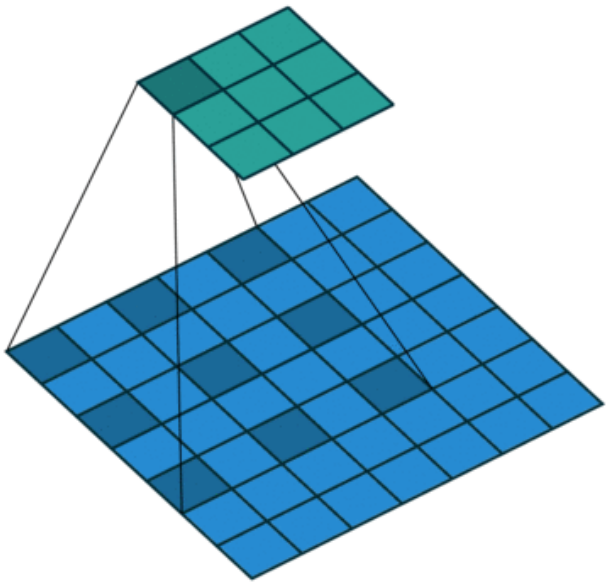
—

Credits: Dumoulin and Visin, A guide to convolution arithmetic for deep learning, 2016.

Dilation

The dilation modulates the expansion of the kernel support by adding rows and columns of zeros between coefficients.

Having a dilation coefficient greater than one increases the units receptive field size without increasing the number of parameters.



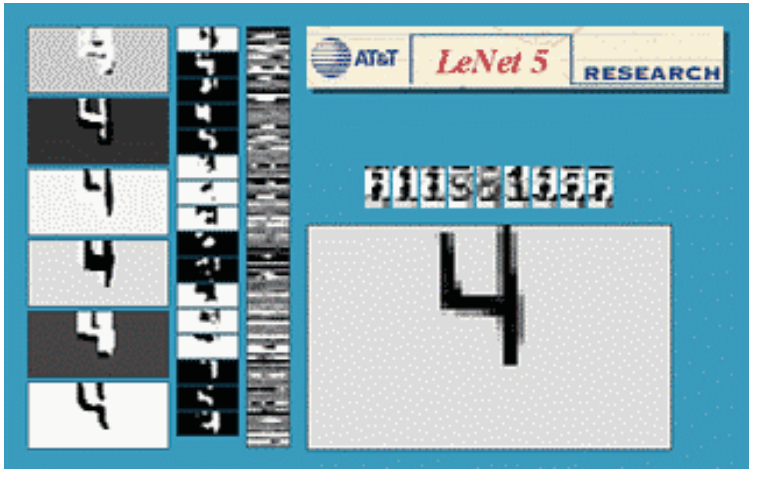
—

Credits: Dumoulin and Visin, A guide to convolution arithmetic for deep learning, 2016.

Equivariance

A function f is equivariant to g if $f(g(\mathbf{x})) = g(f(\mathbf{x}))$.

- Parameter sharing used in a convolutional layer causes the layer to be equivariant to translation.
- If g is any function that translates the input, the convolution function is equivariant to g.



If an object moves in the input image, its representation will move the same amount in the output.

—

Credits: LeCun et al, Gradient-based learning applied to document recognition, 1998.

- Equivariance is useful when we know some local function is useful everywhere (e.g., edge detectors).
- Convolution is not equivariant to other operations such as change in scale or rotation.

Convolutions as matrix multiplications

As a guiding example, let us consider the convolution of single-channel tensors $\mathbf{x} \in \mathbb{R}^{4\times4}$ and $\mathbf{u} \in \mathbb{R}^{3\times3}$:

$$\mathbf{x} \circledast \mathbf{u} = \begin{pmatrix} 4 & 5 & 8 & 7 \\ 1 & 8 & 8 & 8 \\ 3 & 6 & 6 & 4 \\ 6 & 5 & 7 & 8 \end{pmatrix} \circledast \begin{pmatrix} 1 & 4 & 1 \\ 1 & 4 & 3 \\ 3 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 122 & 148 \\ 126 & 134 \end{pmatrix}$$

The convolution operation can be equivalently re-expressed as a single matrix multiplication:

- the convolutional kernel $\mathbf{u}$ is rearranged as a sparse Toeplitz circulant matrix, called the convolution matrix:

$$\mathbf{U} = \begin{pmatrix} 1 & 4 & 1 & 0 & 1 & 4 & 3 & 0 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 1 & 4 & 3 & 0 & 3 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 1 & 4 & 3 & 0 & 3 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 1 & 4 & 3 & 0 & 3 & 3 & 1 \end{pmatrix}$$

- the input $\mathbf{x}$ is flattened row by row, from top to bottom:

$$v(\mathbf{x}) = \begin{pmatrix} 4 & 5 & 8 & 7 & 1 & 8 & 8 & 8 & 3 & 6 & 6 & 4 & 6 & 5 & 7 & 8 \end{pmatrix}^T$$

Then,

$$\mathbf{U}v(\mathbf{x}) = \begin{pmatrix} 122 & 148 & 126 & 134 \end{pmatrix}^T$$

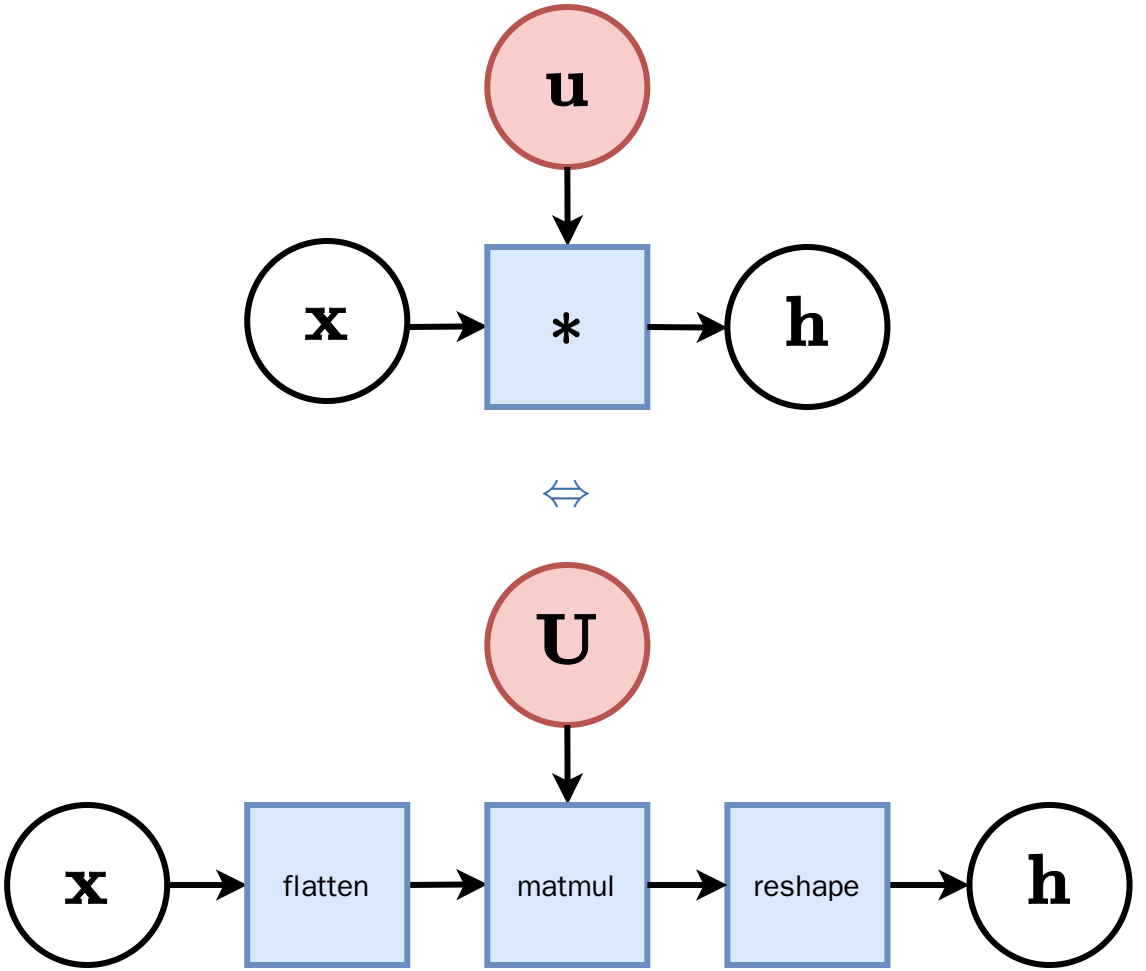
which we can reshape to a 2×2 matrix to obtain $\mathbf{x} \circledast \mathbf{u}$.

The same procedure generalizes to $\mathbf{x} \in \mathbb{R}^{H \times W}$ and convolutional kernel $\mathbf{u} \in \mathbb{R}^{h \times w}$, such that:

- the convolutional kernel is rearranged as a sparse Toeplitz circulant matrix $\mathbf{U}$ of shape $(H - h + 1)(W - w + 1) \times HW$ where
  - each row i identifies an element of the output feature map,
  - each column j identifies an element of the input feature map,
  - the value $\mathbf{U}_{i,j}$ corresponds to the kernel value the element j is multiplied with in output i;
- the input $\mathbf{x}$ is flattened into a column vector $v(\mathbf{x})$ of shape $HW \times 1$;
- the output feature map $\mathbf{x} \circledast \mathbf{u}$ is obtained by reshaping the $(H - h + 1)(W - w + 1) \times 1$ column vector $\mathbf{U}v(\mathbf{x})$ as a $(H - h + 1) \times (W - w + 1)$ matrix.

Therefore, a convolutional layer is a special case of a fully connected layer:

$$\mathbf{h} = \mathbf{x} \circledast \mathbf{u} \Leftrightarrow v(\mathbf{h}) = \mathbf{U}v(\mathbf{x}) \Leftrightarrow v(\mathbf{h}) = \mathbf{W}^T v(\mathbf{x})$$


u
x
*
h
⇔
U
x
flatten
matmul
reshape
h

Pooling

When the input volume is large, pooling layers can be used to reduce the input dimension while preserving its global structure, in a way similar to a down-scaling operation.

Pooling

Consider a pooling area of size $h \times w$ and a 3D input tensor $\mathbf{x} \in \mathbb{R}^{C \times (rh) \times (sw)}$.

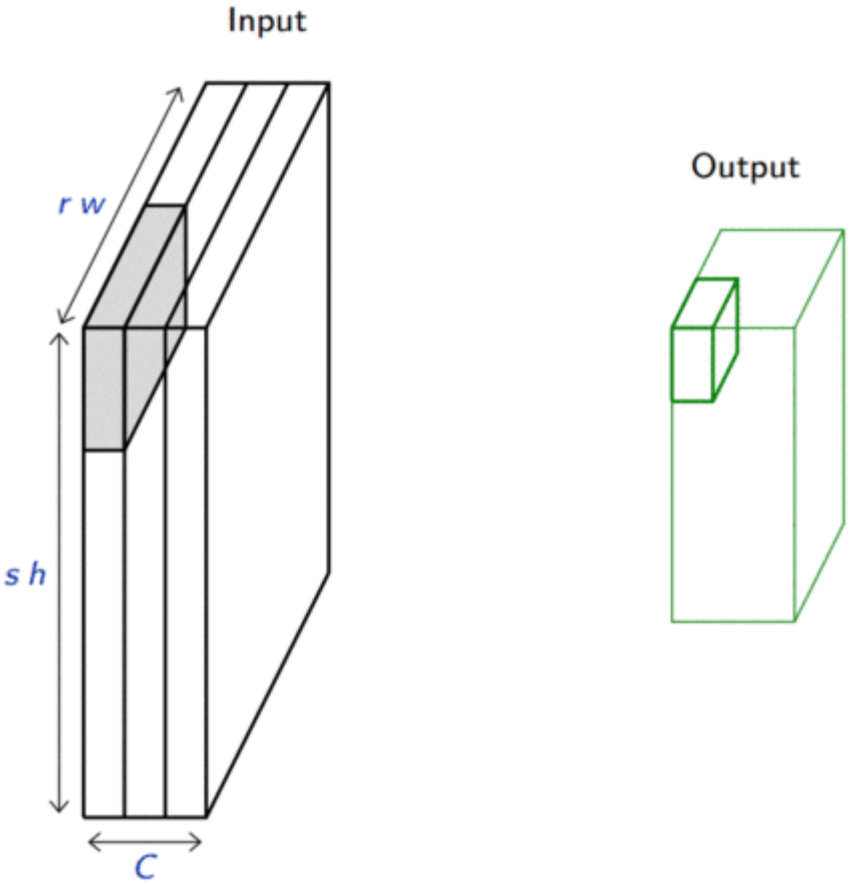
- Max-pooling produces a tensor $\mathbf{o} \in \mathbb{R}^{C \times r \times s}$ such that

$$\mathbf{o}_{c,j,i} = \max_{n<h, m<w} \mathbf{x}_{c,rj+n,si+m}.$$

- Average pooling produces a tensor $\mathbf{o} \in \mathbb{R}^{C \times r \times s}$ such that

$$\mathbf{o}_{c,j,i} = \frac{1}{hw} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{c,rj+n,si+m}.$$

Pooling is very similar in its formulation to convolution.

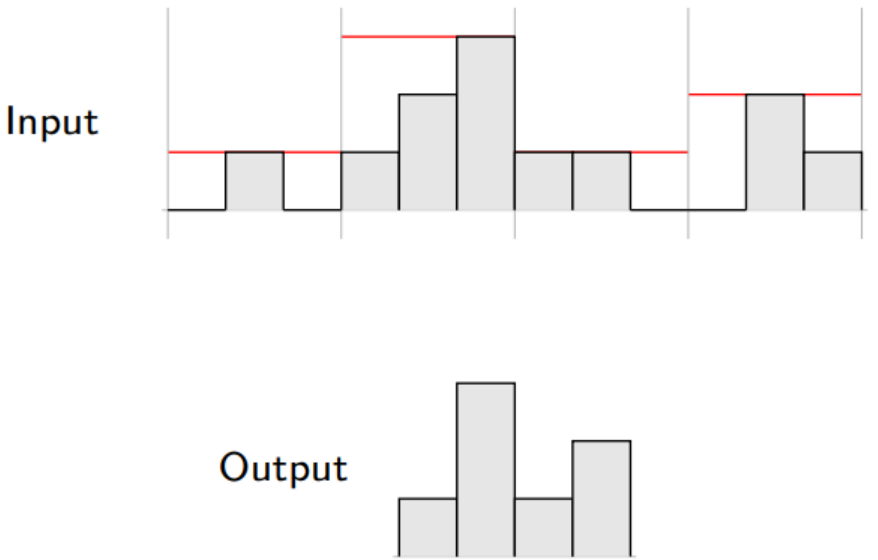

Input
r w
s h
C
Output

—

Invariance

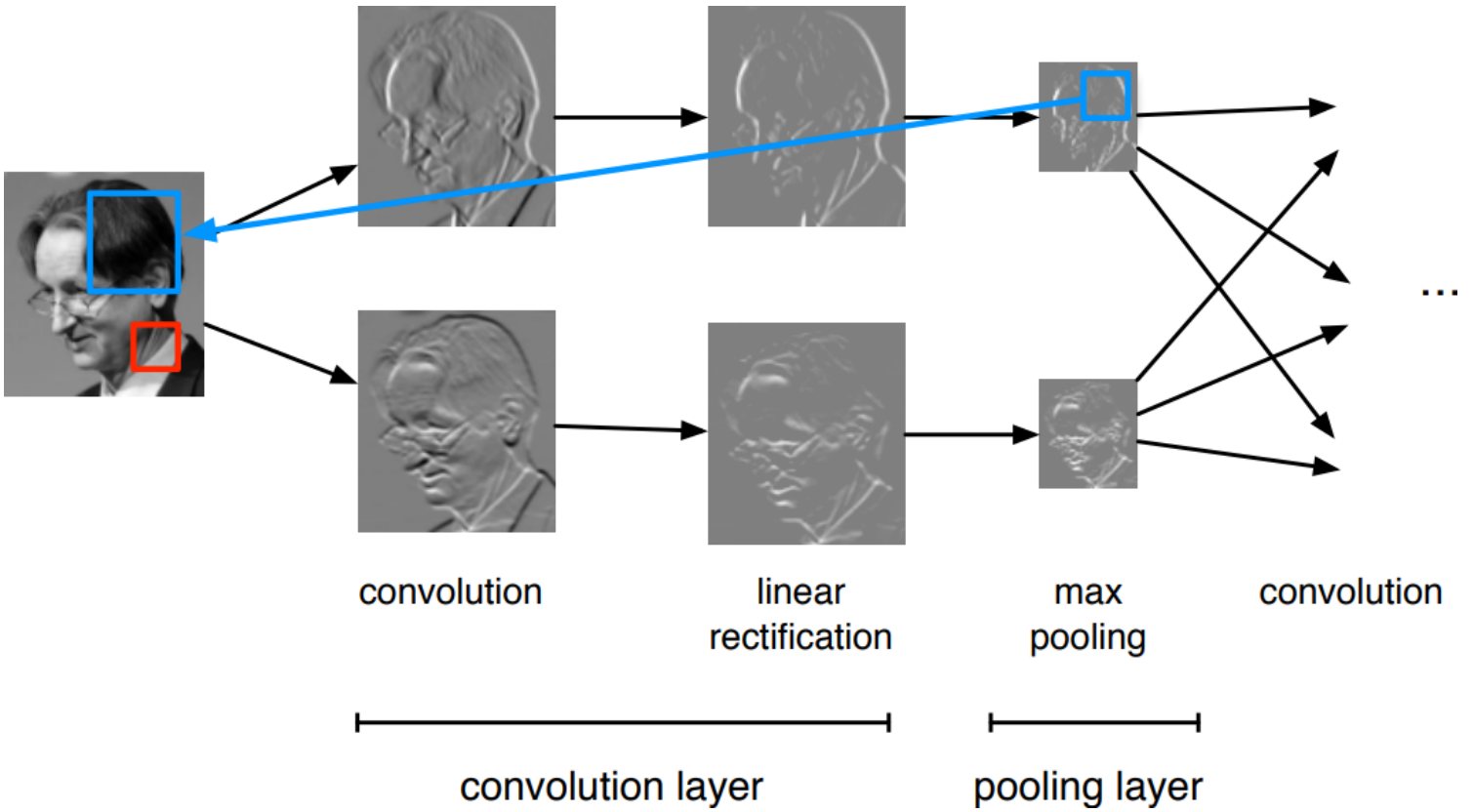
A function f is invariant to g if $f(g(\mathbf{x})) = f(\mathbf{x})$.

- Pooling layers provide invariance to any permutation inside one cell.
- It results in (pseudo-)invariance to local translations.
- This helpful if we care more about the presence of a pattern rather than its exact position.



Convolutional networks

A convolutional network is generically defined as a composition of convolutional layers (`CONV`), pooling layers (`POOL`), linear rectifiers (`ReLU`) and fully connected layers (`FC`).



The most common convolutional network architecture follows the pattern:

$$\texttt{INPUT} \rightarrow [[\texttt{CONV} \rightarrow \texttt{ReLU}]*N \rightarrow \texttt{POOL?}]*M \rightarrow [\texttt{FC} \rightarrow \texttt{ReLU}]*K \rightarrow \texttt{FC}$$

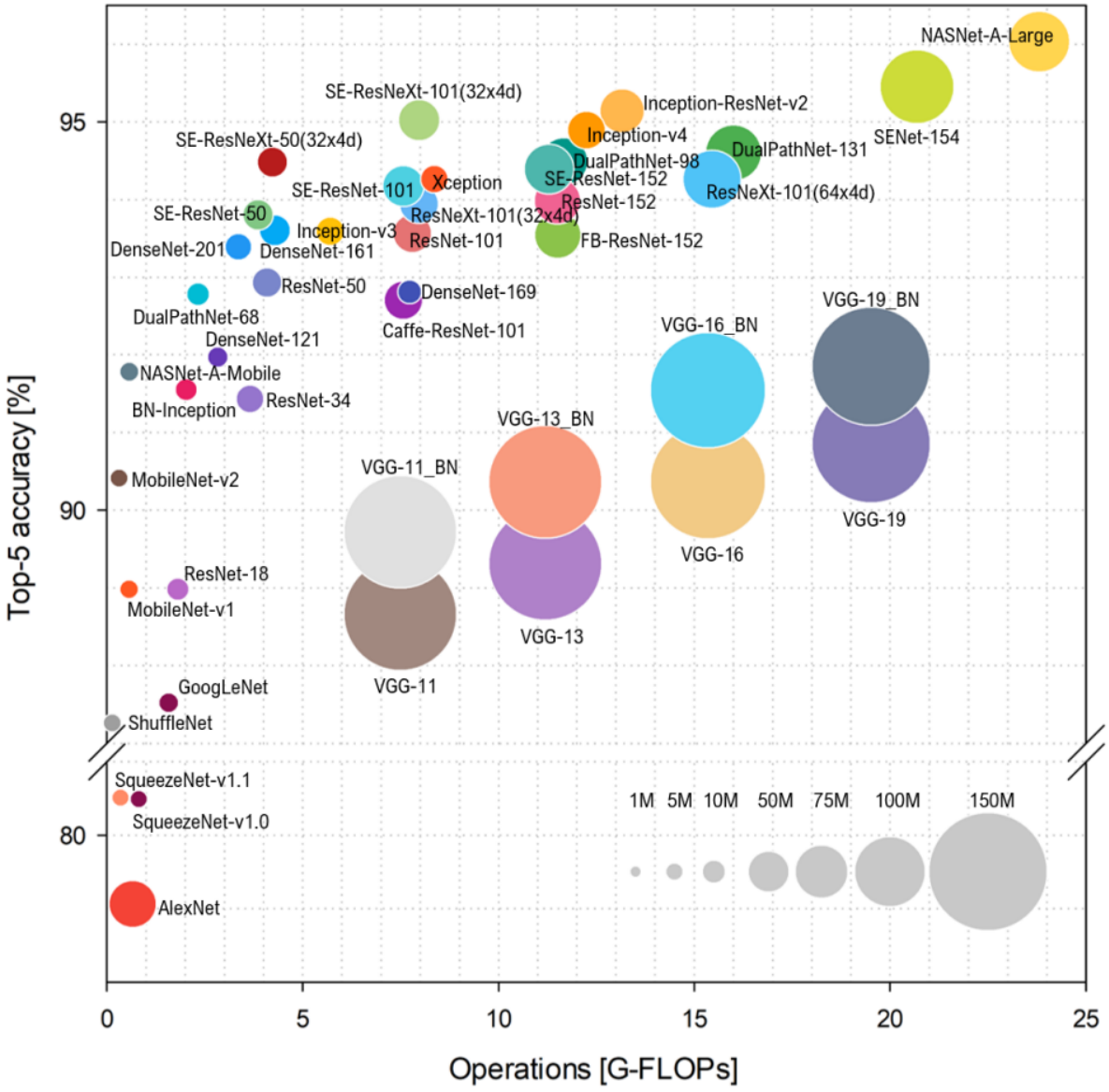
where:

- $*$ indicates repetition;
- $\texttt{POOL?}$ indicates an optional pooling layer;
- $N \geq 0$ (and usually $N \leq 3$), $M \geq 0$, $K \geq 0$ (and usually $K < 3$);
- the last fully connected layer holds the output (e.g., the class scores).

Some common architectures for convolutional networks following this pattern include:

- $\texttt{INPUT} \rightarrow \texttt{FC}$, which implements a linear classifier ($N = M = K = 0$).
- $\texttt{INPUT} \rightarrow \left[\texttt{FC} \rightarrow \texttt{ReLU}\right]*K \rightarrow \texttt{FC}$, which implements a K-layer MLP.
- $\texttt{INPUT} \rightarrow \texttt{CONV} \rightarrow \texttt{ReLU} \rightarrow \texttt{FC}$.
- $\texttt{INPUT} \rightarrow \left[\texttt{CONV} \rightarrow \texttt{ReLU} \rightarrow \texttt{POOL}\right]*2 \rightarrow \texttt{FC} \rightarrow \texttt{ReLU} \rightarrow \texttt{FC}$.
- $\texttt{INPUT} \rightarrow \left[\left[\texttt{CONV} \rightarrow \texttt{ReLU}\right]*2 \rightarrow \texttt{POOL}\right]*3 \rightarrow \left[\texttt{FC} \rightarrow \texttt{ReLU}\right]*2 \rightarrow \texttt{FC}$.

(demo)

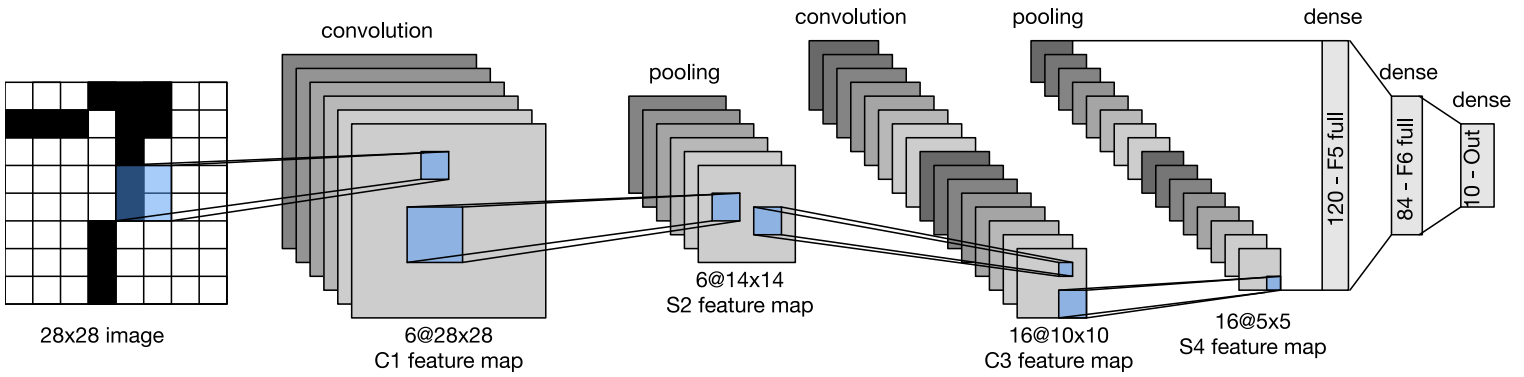


—

Credits: Bianco et al, 2018.

LeNet-5 (LeCun et al, 1998)

Composition of two `CONV` + `POOL` layers, followed by a block of fully-connected layers.



—

Credits: Dive Into Deep Learning, 2020.

AlexNet (Krizhevsky et al, 2012)

Composition of a 8-layer convolutional neural network with a 3-layer MLP.

The original implementation was made of two parts such that it could fit within two GPUs.

LeNet vs. AlexNet

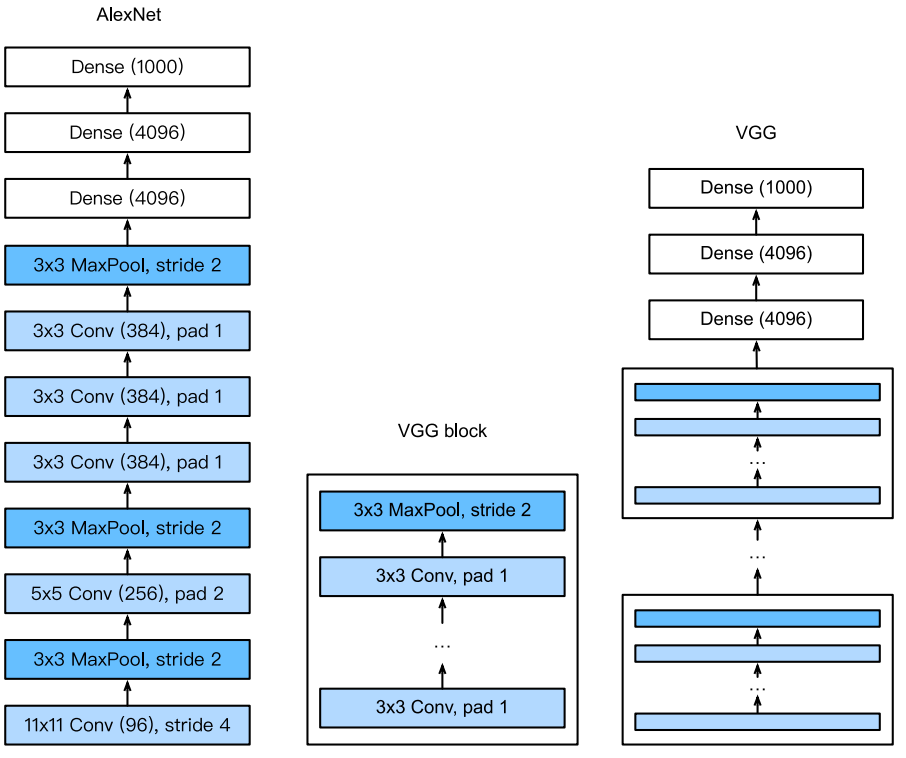
—

Credits: Dive Into Deep Learning, 2020.

VGG (Simonyan and Zisserman, 2014)

Composition of 5 VGG blocks consisting of `CONV` + `POOL` layers, followed by a block of fully connected layers.

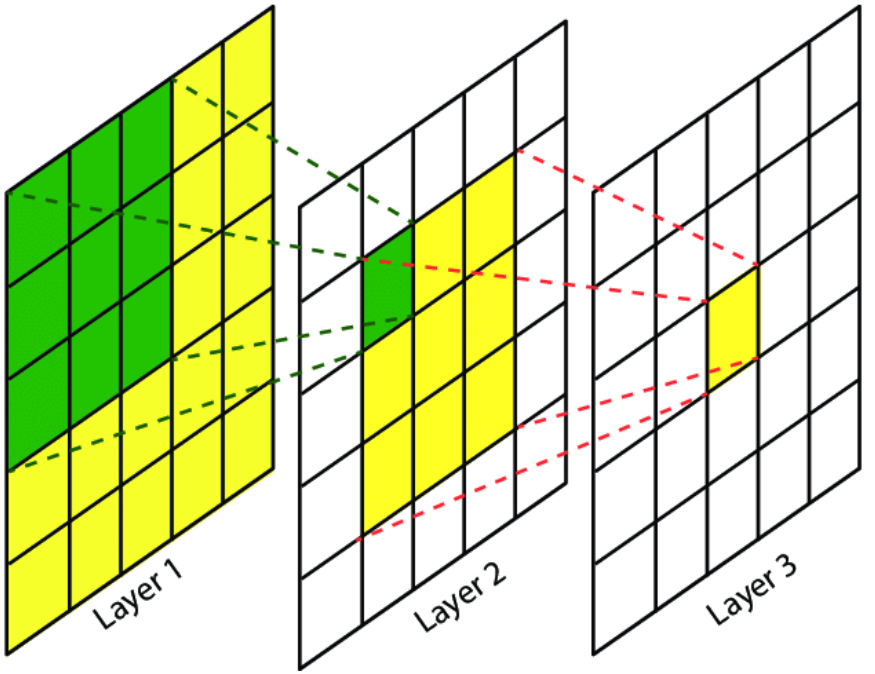
The network depth increased up to 19 layers, while the kernel sizes reduced to 3.



AlexNet vs. VGG

—

Credits: Dive Into Deep Learning, 2020.



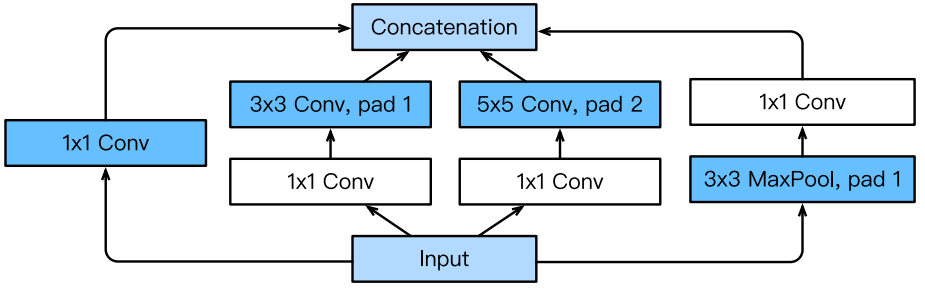
The effective receptive field is the part of the visual input that affects a given unit indirectly through previous convolutional layers. It grows linearly with depth.

E.g., a stack of two 3×3 kernels of stride 1 has the same effective receptive field as a single 5×5 kernel, but fewer parameters.

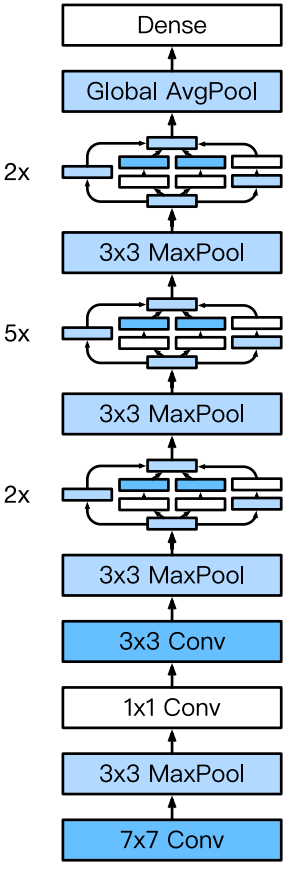
GoogLeNet (Szegedy et al, 2014)

Composition of two `CONV` + `POOL` layers, a stack of 9 inception blocks, and a global average pooling layer.

Each inception block is itself defined as a convolutional network with 4 parallel paths.



Inception block

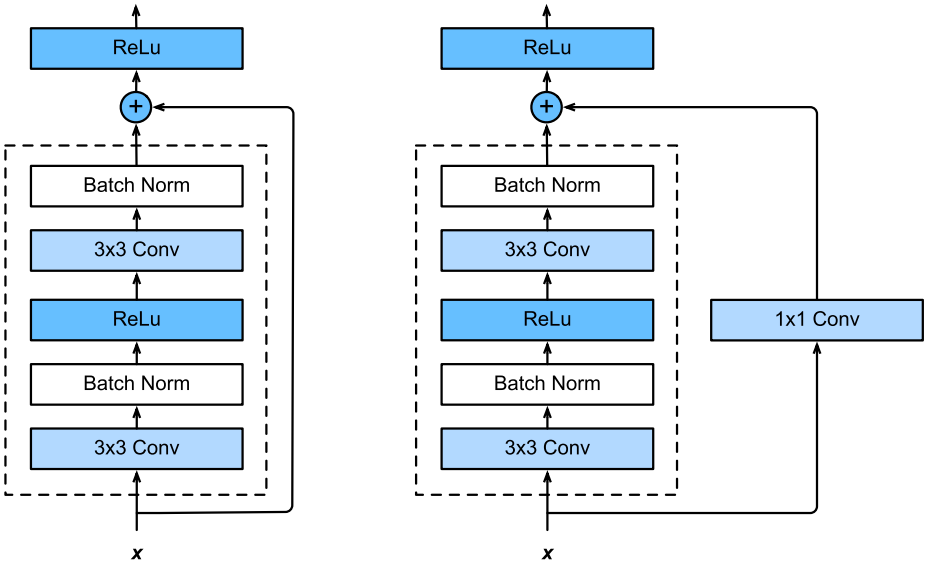


—

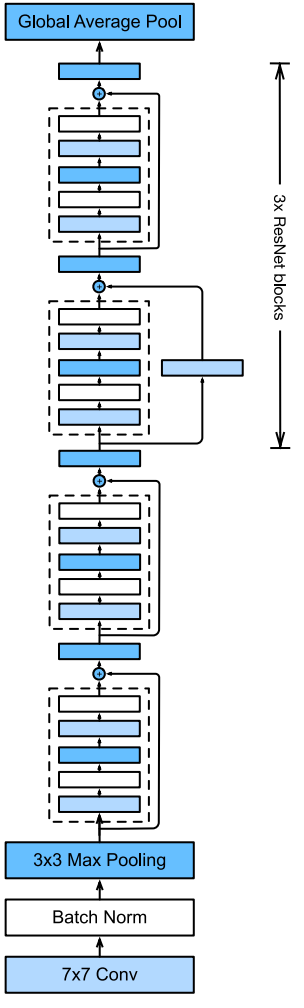
Credits: Dive Into Deep Learning, 2020.

ResNet (He et al, 2015)

Composition of first layers similar to GoogLeNet, a stack of 4 residual blocks, and a global average pooling layer. Extensions consider more residual blocks, up to a total of 152 layers (ResNet-152).



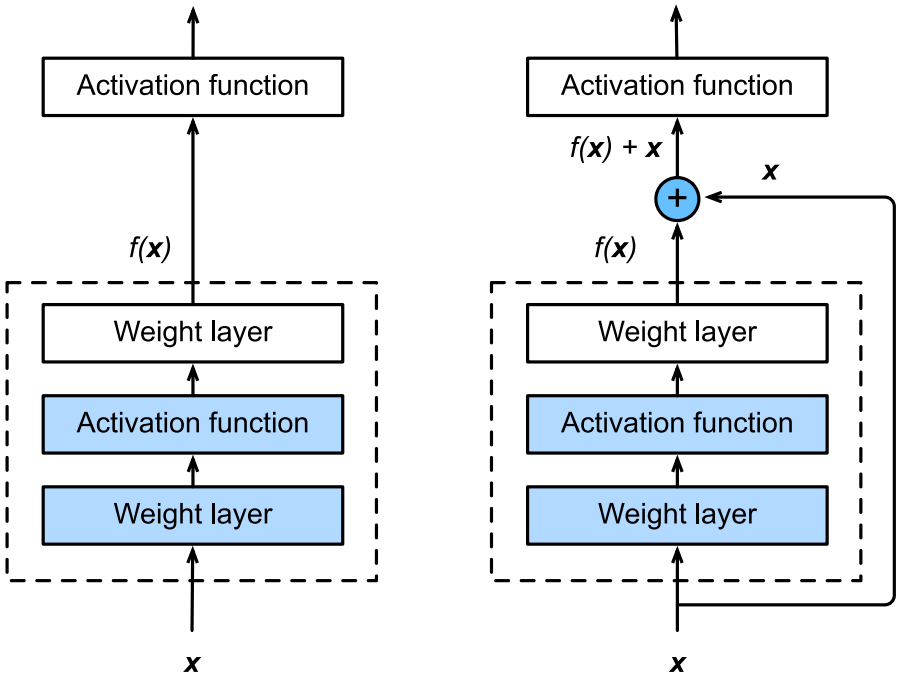
Regular ResNet block vs. ResNet block with 1×1 *convolution.*



—

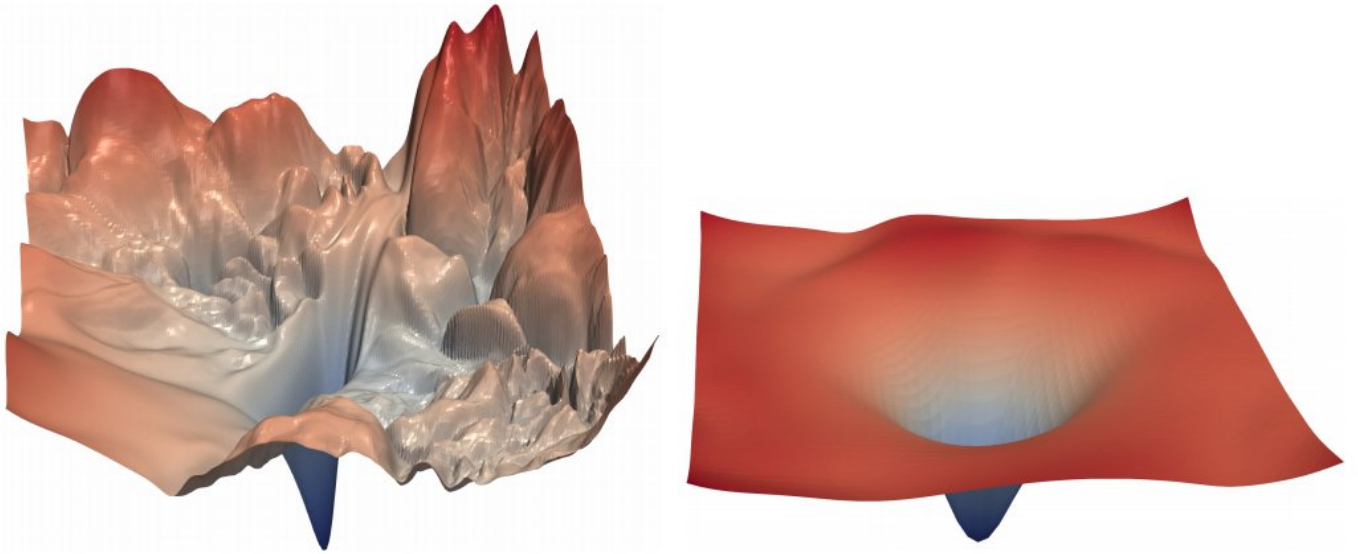
Credits: Dive Into Deep Learning, 2020.

Training networks of this depth is made possible because of the skip connections in the residual blocks. They allow the gradients to shortcut the layers and pass through without vanishing.



—

Credits: Dive Into Deep Learning, 2020.

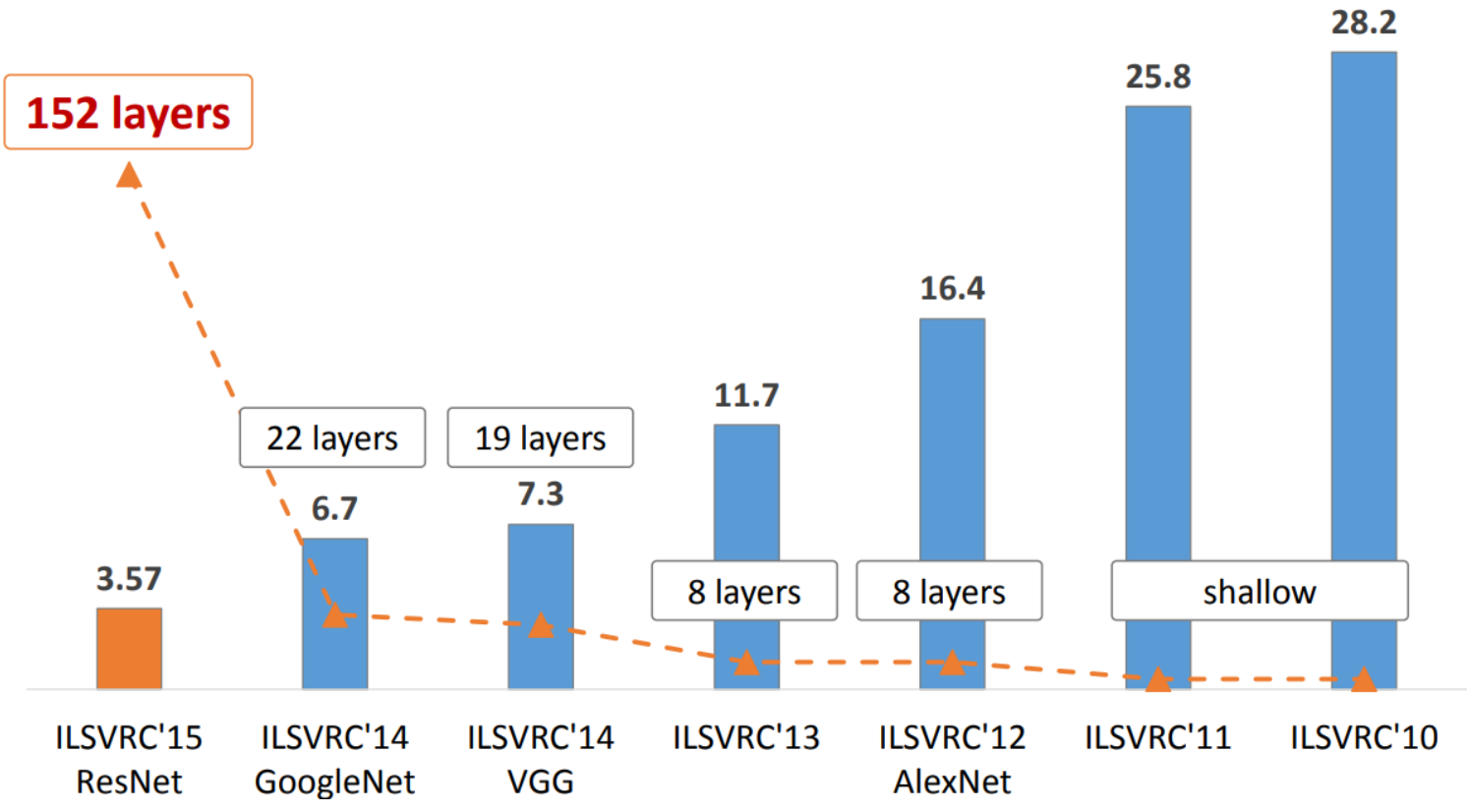


(a) without skip connections

(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

The benefits of depth



... and width and resolution

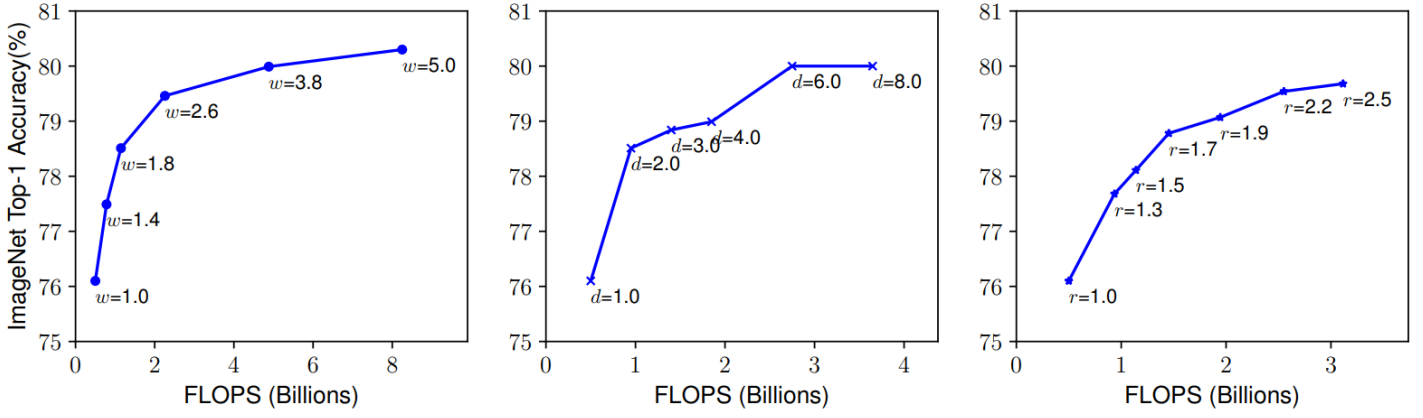


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

—

Credits: Tan and Le, 2019.

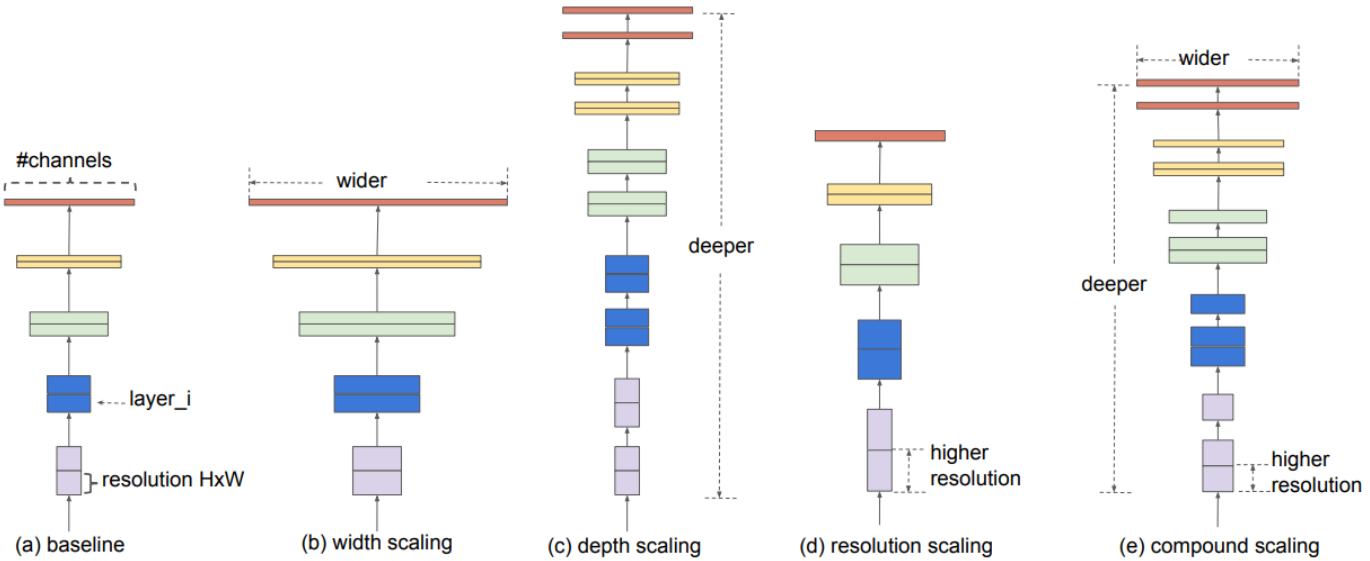


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

—

Credits: Tan and Le, 2019.

Under the hood

Understanding what is happening in deep neural networks after training is complex and the tools we have are limited.

In the case of convolutional neural networks, we can look at:

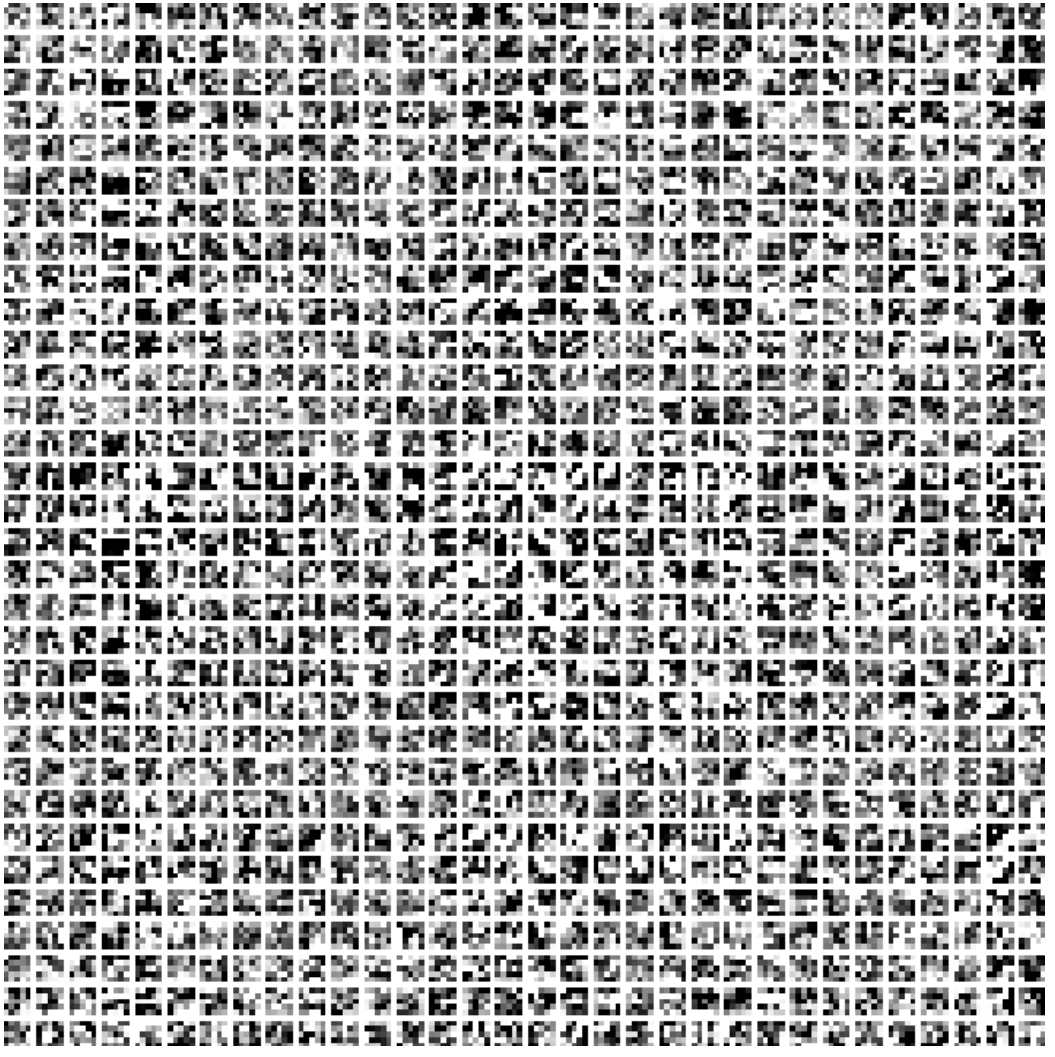
- the network's kernels as images
- internal activations on a single sample as images
- distributions of activations on a population of samples
- derivatives of the response with respect to the input
- maximum-response synthetic samples

Looking at filters

LeNet's first convolutional layer, all filters.

—

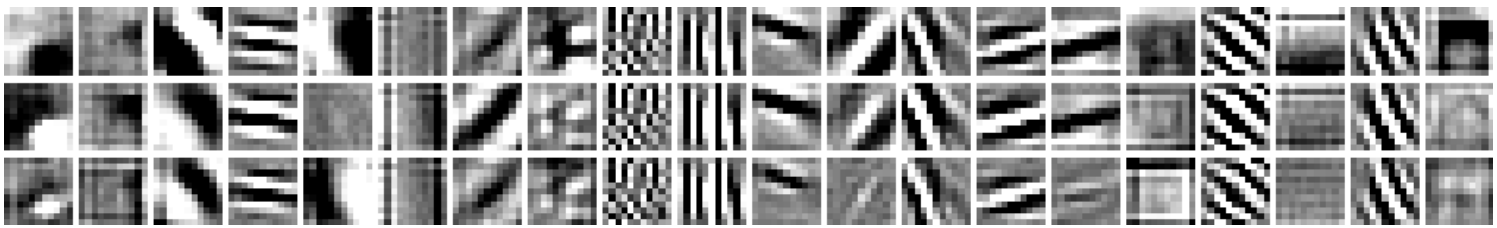
LeNet's second convolutional layer, first 32 filters.



—

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

AlexNet's first convolutional layer, first 20 filters.



—

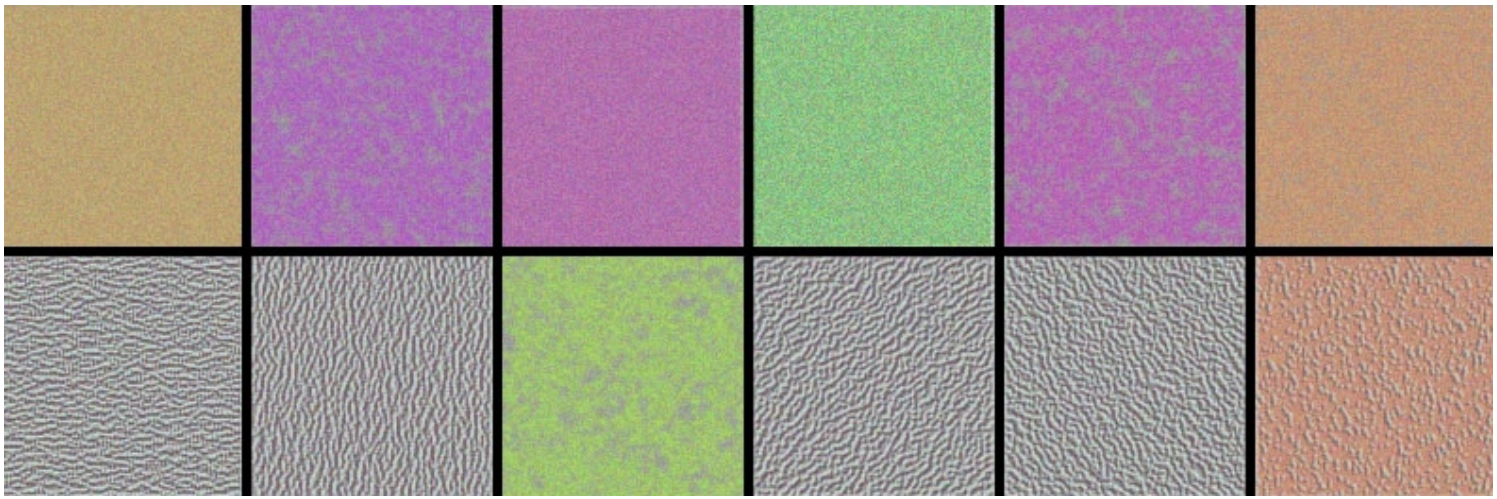
Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

Maximum response samples

Convolutional networks can be inspected by looking for synthetic input images $\mathbf{x}$ that maximize the activation $\mathbf{h}_{\ell,d}(\mathbf{x})$ of a chosen convolutional kernel $\mathbf{u}$ at layer ℓ and index d in the layer filter bank.

These samples can be found by gradient ascent on the input space:

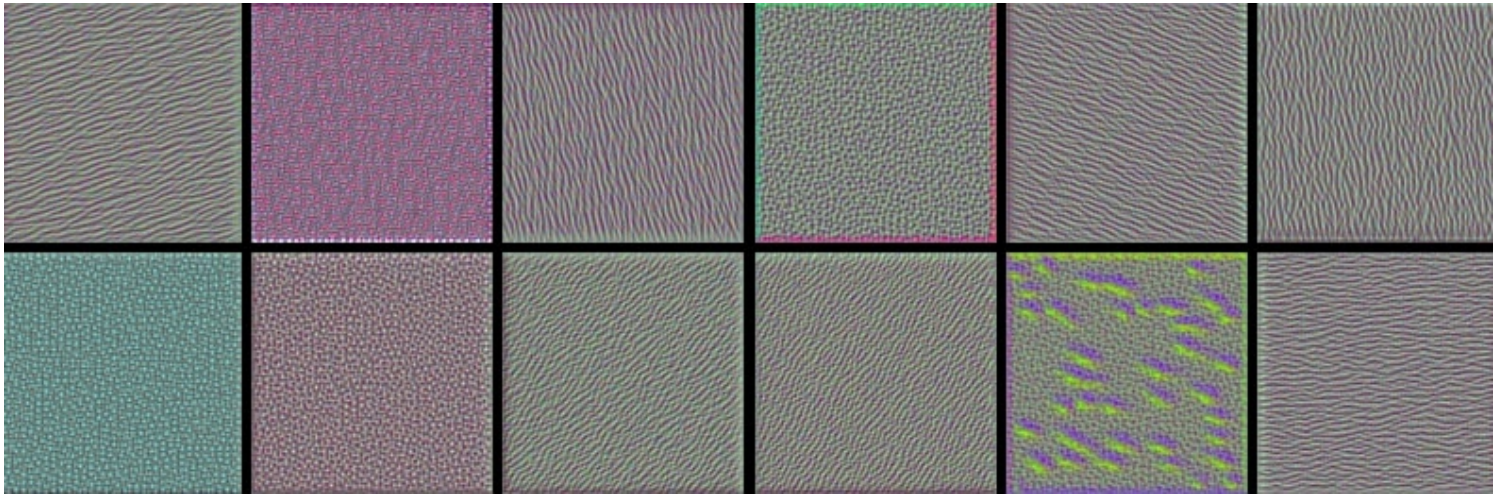
$$\begin{aligned}\mathcal{L}_{\ell,d}(\mathbf{x}) &= ||\mathbf{h}_{\ell,d}(\mathbf{x})||_2 \\ \mathbf{x}_0 &\sim U[0,1]^{C\times H\times W} \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \gamma \nabla_{\mathbf{x}} \mathcal{L}_{\ell,d}(\mathbf{x}_t)\end{aligned}$$



VGG-16, convolutional layer 1-1, a few of the 64 filters

—

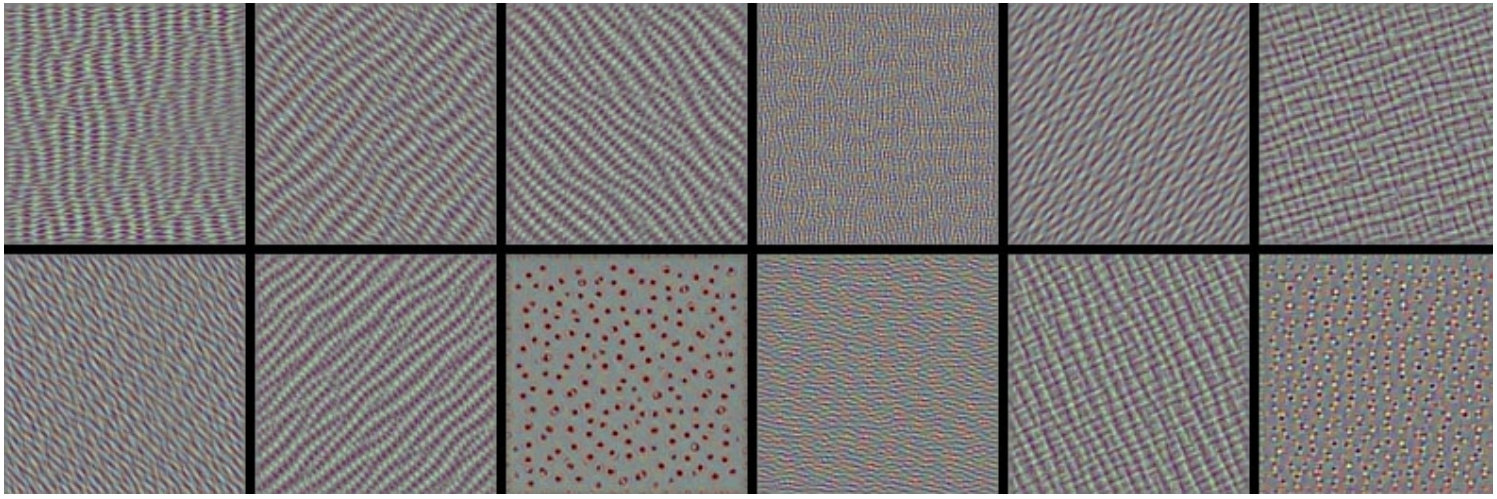
Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



VGG-16, convolutional layer 2-1, a few of the 128 filters

—

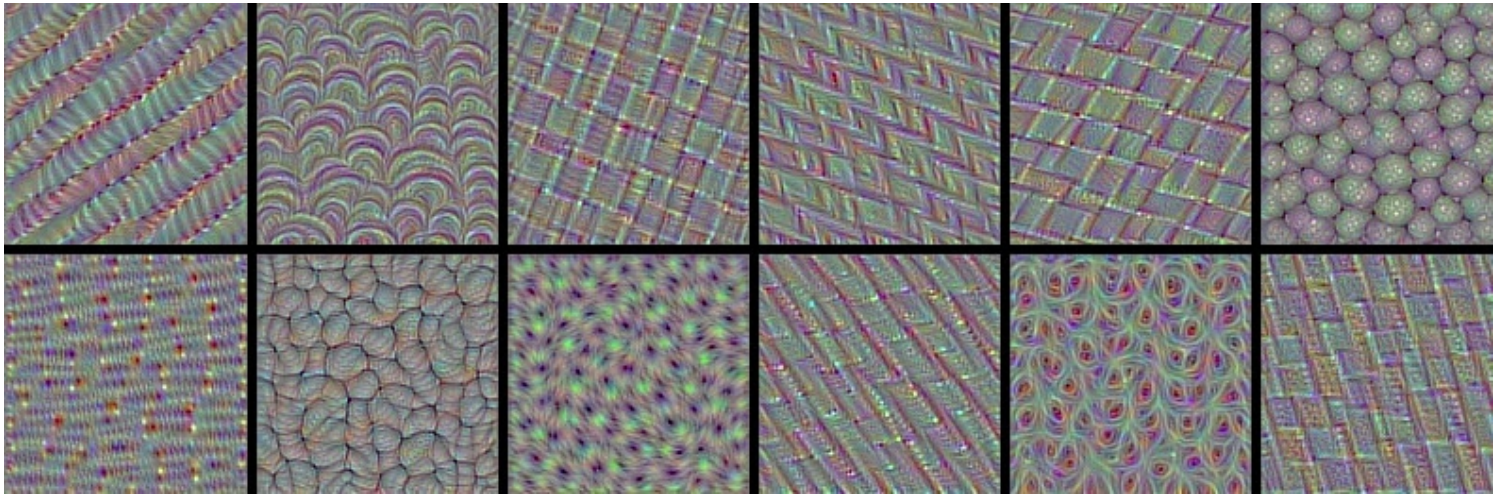
Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



VGG-16, convolutional layer 3-1, a few of the 256 filters

—

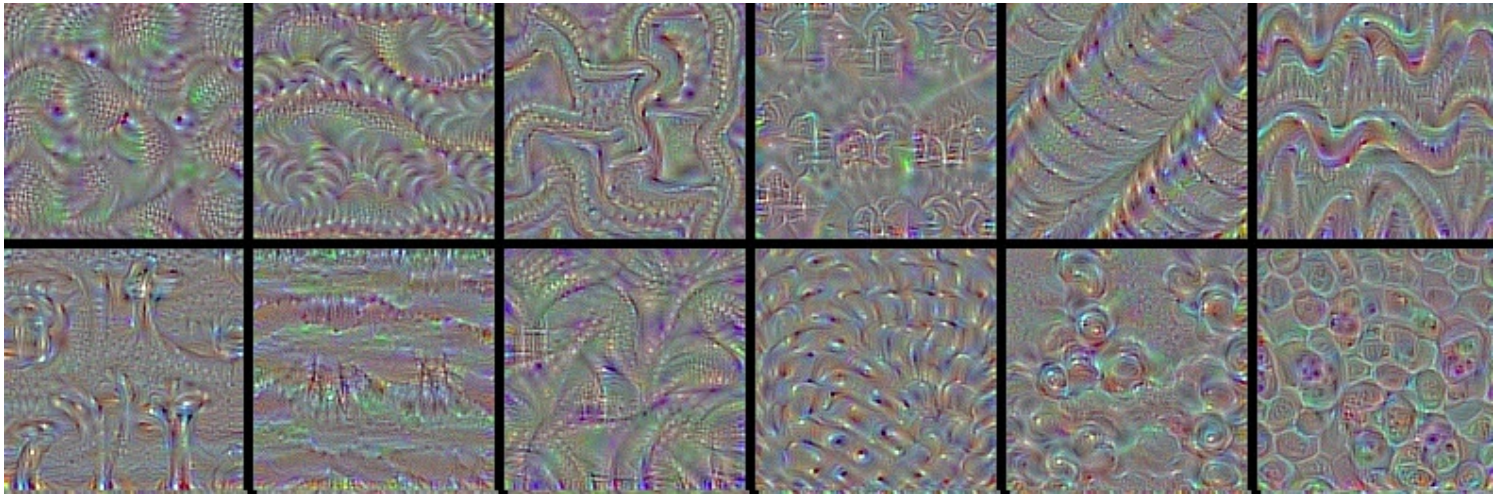
Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



VGG-16, convolutional layer 4-1, a few of the 512 filters

—

Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



VGG-16, convolutional layer 5-1, a few of the 512 filters

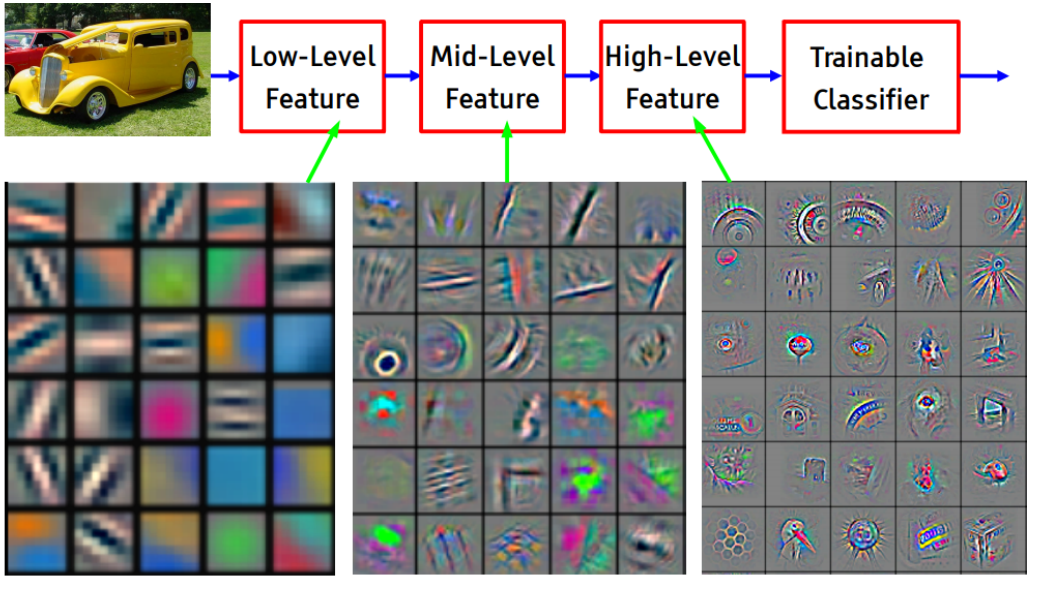
—

Credits: Francois Chollet, How convolutional neural networks see the world, 2016.

Some observations:

- The first layers appear to encode direction and color.
- The direction and color filters get combined into grid and spot textures.
- These textures gradually get combined into increasingly complex patterns.

The network appears to learn a **hierarchical composition of patterns**.

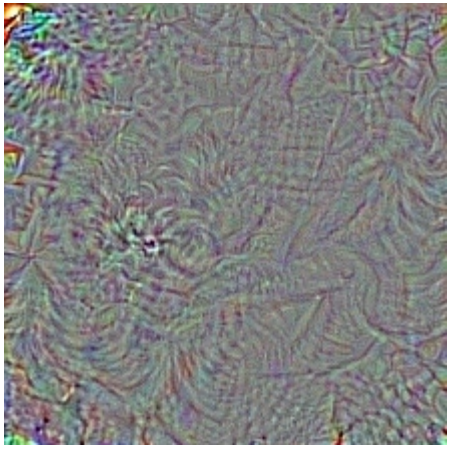


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

What if we build images that maximize the activation of a chosen class output?

What if we build images that maximize the activation of a chosen class output?

The left image is predicted with 99.9% confidence as a magpie!

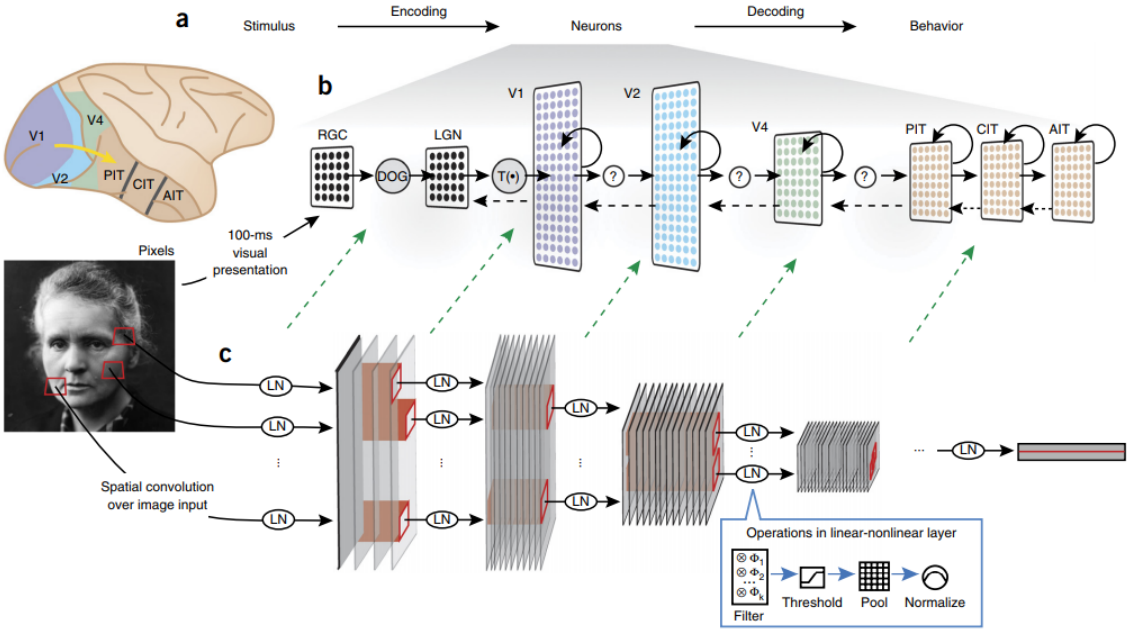


—

Credits: Francois Chollet, How convolutional neural networks see the world, 2016.

Deep Dream. Start from an image $\mathbf{x}_t$, offset by a random jitter, enhance some layer activation at multiple scales, zoom in, repeat on the produced image $\mathbf{x}_{t+1}$.

Biological plausibility



"Deep hierarchical neural networks are beginning to transform neuroscientists' ability to produce quantitatively accurate computational models of the sensory systems, especially in higher cortical areas where neural response properties had previously been enigmatic."

—

Credits: Yamins et al, Using goal-driven deep learning models to understand sensory cortex, 2016.

The end.

References

- Francois Fleuret, Deep Learning Course, 4.4. Convolutions, EPFL, 2018.
- Yannis Avrithis, Deep Learning for Vision, Lecture 1: Introduction, University of Rennes 1, 2018.
- Yannis Avrithis, Deep Learning for Vision, Lecture 7: Convolution and network architectures , University of Rennes 1, 2018.
- Olivier Grisel and Charles Ollion, Deep Learning, Lecture 4: Convolutional Neural Networks for Image Classification , Université Paris-Saclay, 2018.