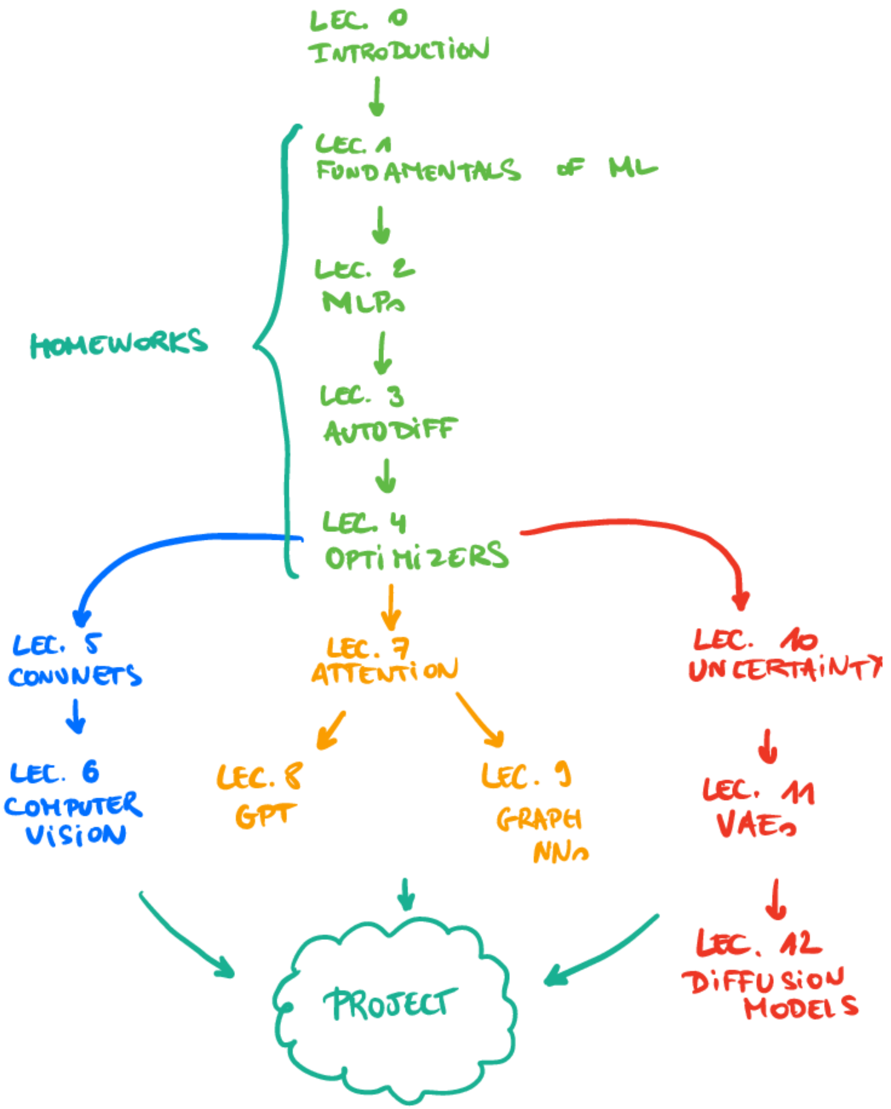
Deep Learning

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

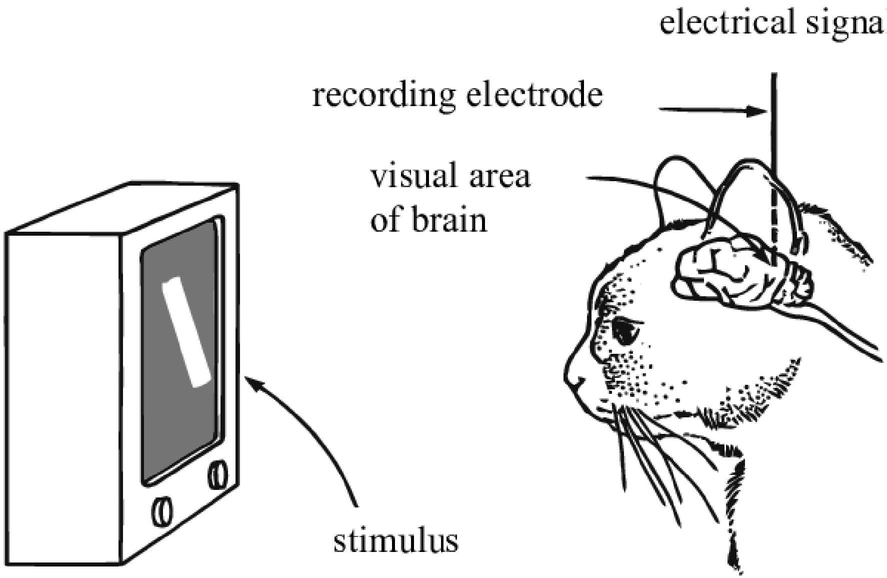
How to make neural networks see?

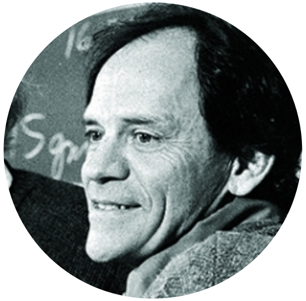
- Visual perception
- Convolutions
- Pooling
- Convolutional networks

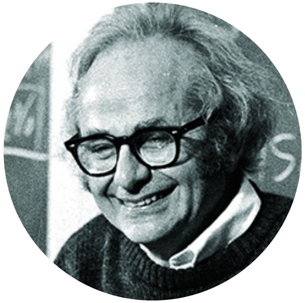
Visual perception

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.







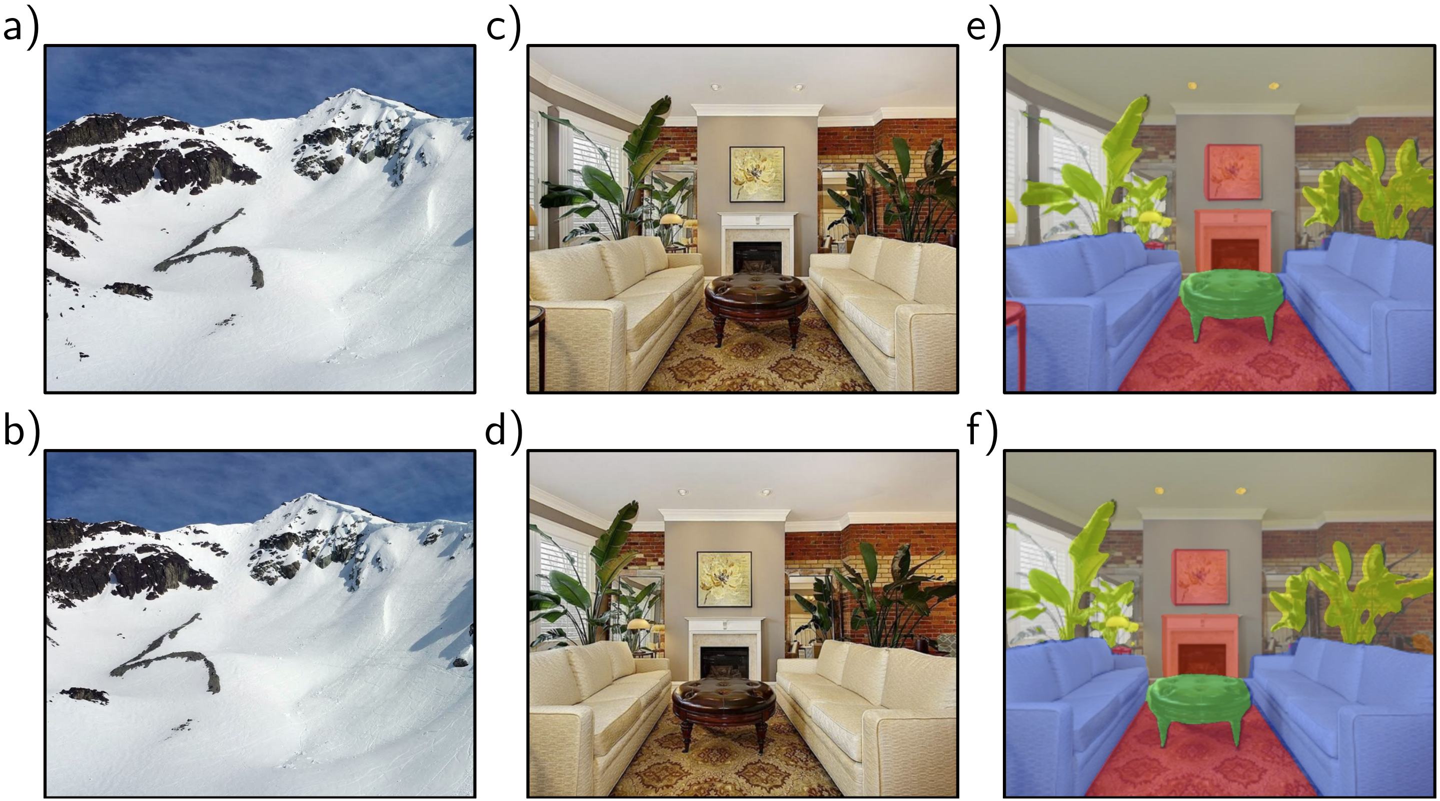
Hubel and Wiesel Cat Experiment

04 2 Simple Complex Cells
Later bekij...
Delen
0H09M41S
SP

Inductive biases

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

- Locality (as in simple cells)
- Invariance translation (as in complex cells)
- Hierarchical compositionality (as in hypercomplex cells)

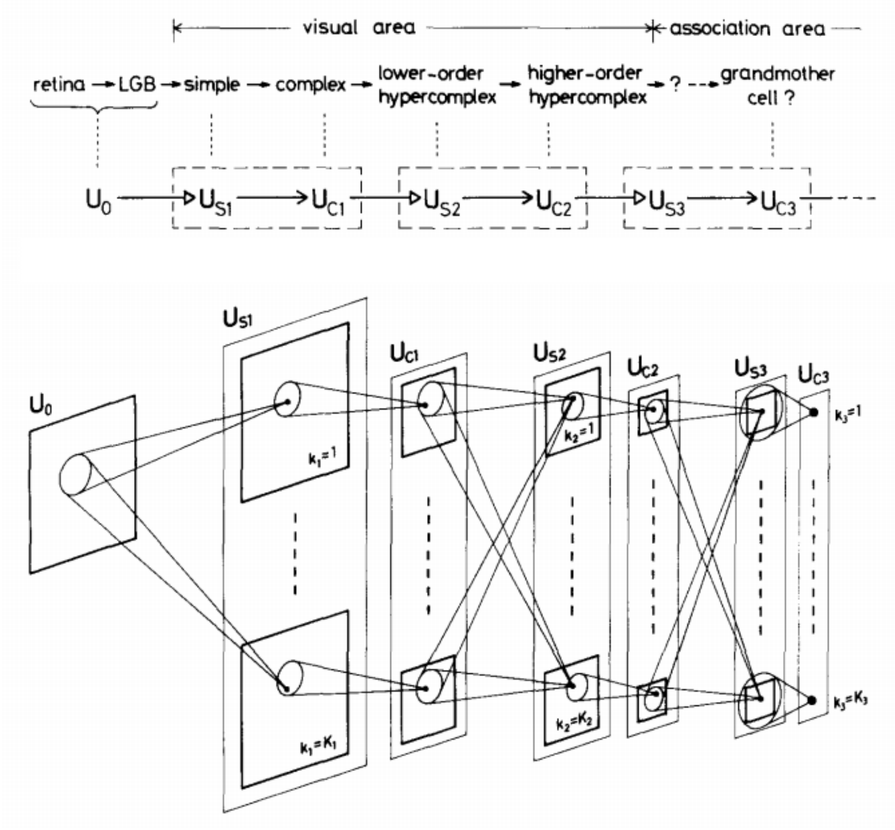


Invariance and equivariance to translation.

Credits: Simon J.D. Prince, Understanding Deep Learning, 2023.

Neocognitron

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



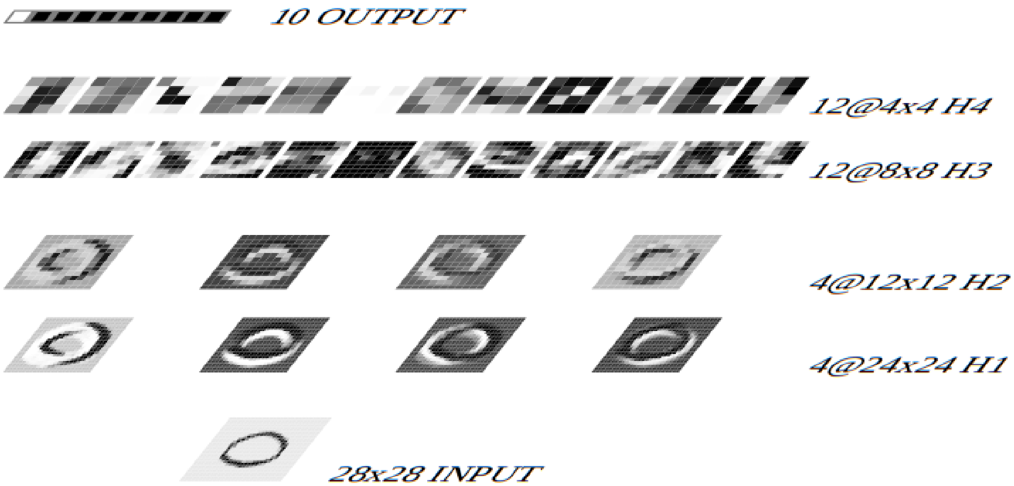
- Built upon convolutions and enables the composition of a feature hierarchy.
- Biologically-inspired training algorithm, which proves to be largely inefficient.

—

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

Convolutional networks

In the 1980-90s, LeCun trains a convolutional network by backpropagation. He advocates for end-to-end feature learning in image classification.



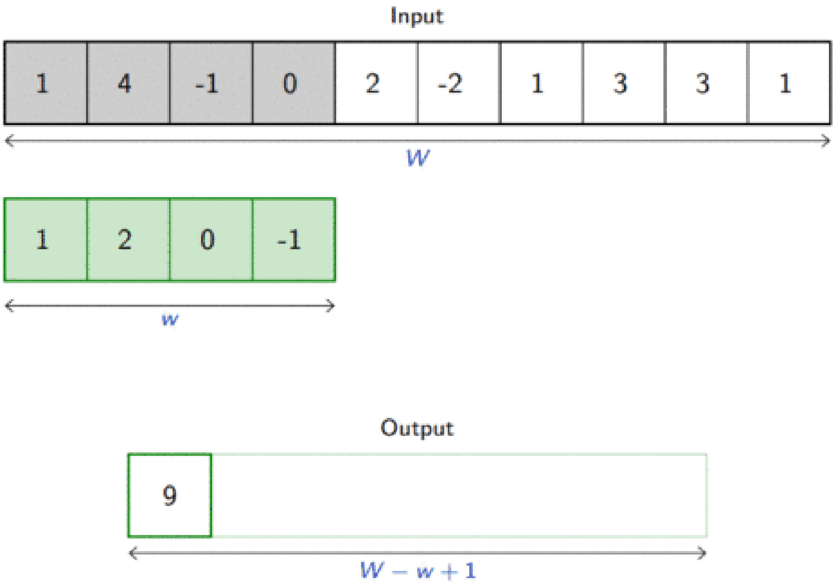
—

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

Convolutions

Convolutional layers

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



1	4	-1	0	2	-2	1	3	3	1

1	2	0	-1

9

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

1d convolution

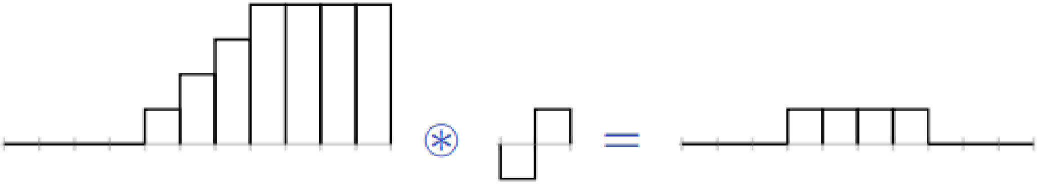
For the one-dimensional input $\mathbf{x} \in \mathbb{R}^W$ and the convolutional kernel $\mathbf{u} \in \mathbb{R}^w$, the discrete convolution $\mathbf{x} \circledast \mathbf{u}$ is a vector of size $W - w + 1$ such that

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

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

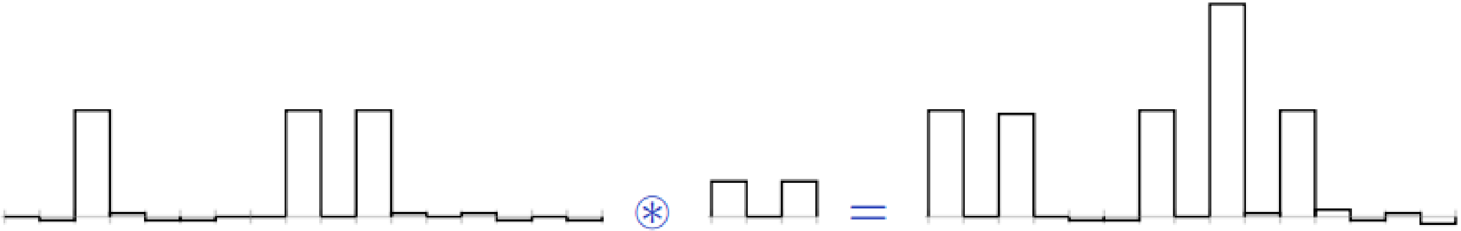
Convolutions can implement differential operators:

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



or crude template matchers:

$$(0,0,3,0,0,0,0,0,3,0,3,0,0,0) \circledast (1,0,1) = (3,0,3,0,0,0,3,0,6,0,3,0)$$



Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

2d convolution

For the 2d input tensor $\mathbf{x} \in \mathbb{R}^{H \times W}$ and the 2d convolutional kernel $\mathbf{u} \in \mathbb{R}^{h \times w}$, the discrete convolution $\mathbf{x} \circledast \mathbf{u}$ is a matrix of size $(H - h + 1) \times (W - w + 1)$ such that

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

Channels

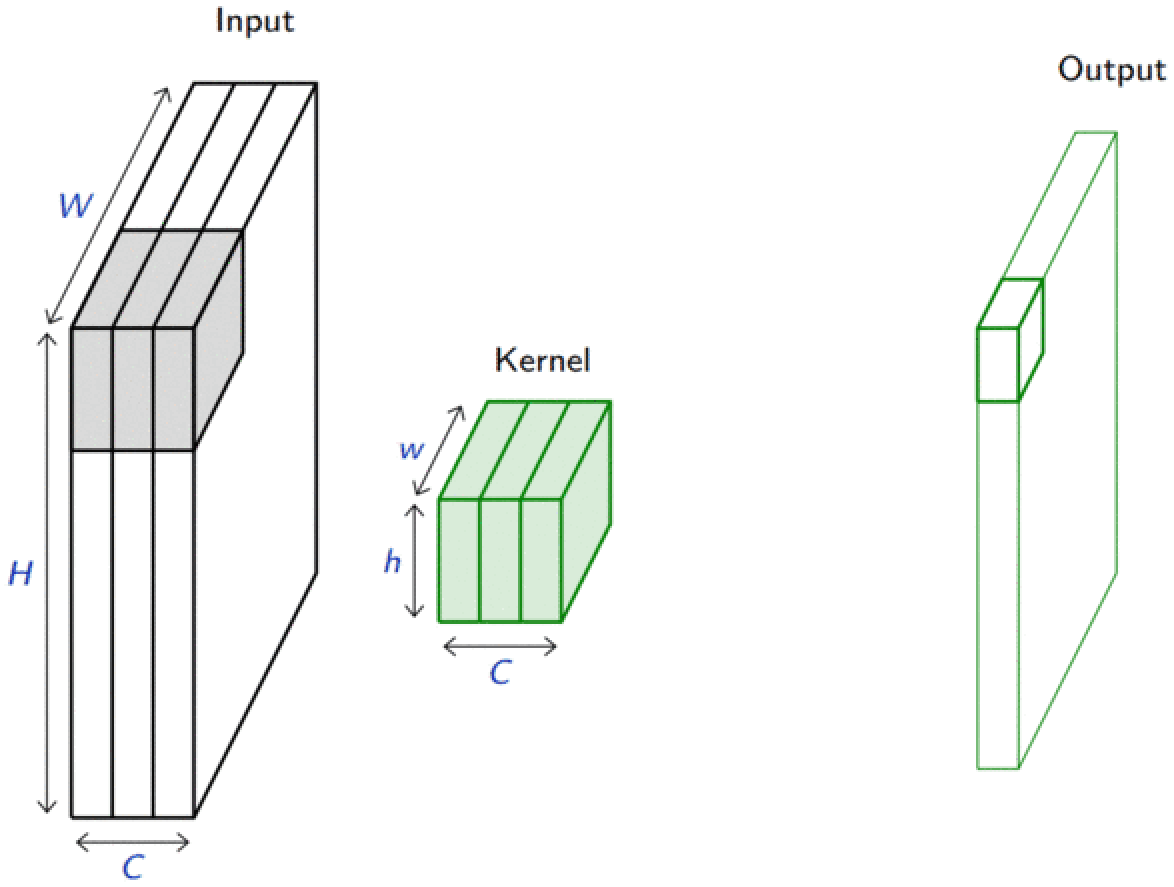
The 2d convolution can be extended to tensors with multiple channels.

For the 3d input tensor $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ and the 3d convolutional kernel $\mathbf{u} \in \mathbb{R}^{C \times h \times w}$, the discrete convolution $\mathbf{x} \circledast \mathbf{u}$ is a tensor of size $(H - h + 1) \times (W - w + 1)$ such that

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

Convolutional layers

A convolutional layer is defined by a set of K kernels $\mathbf{u}_k$ of size $C \times h \times w$. It applies the 2d convolution operation to the input tensor $\mathbf{x}$ of size $C \times H \times W$ to produce a set of K feature maps $\mathbf{o}_k$.

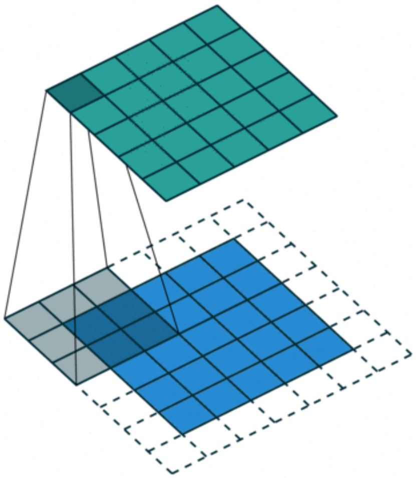


—

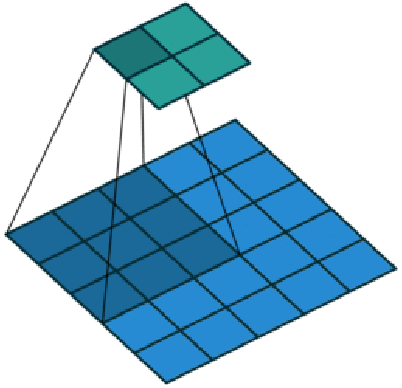
Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

Convolutions have three additional parameters:

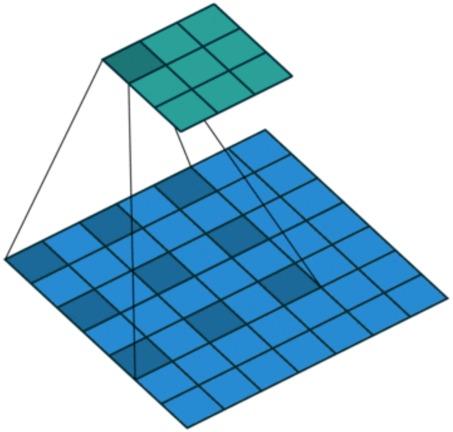
- The **padding** specifies the size of a zeroed frame added arount the input.
- The **stride** specifies a step size when moving the kernel across the signal.
- The **dilation** modulates the expansion of the filter without adding weights.



Padding=1



Stride=2



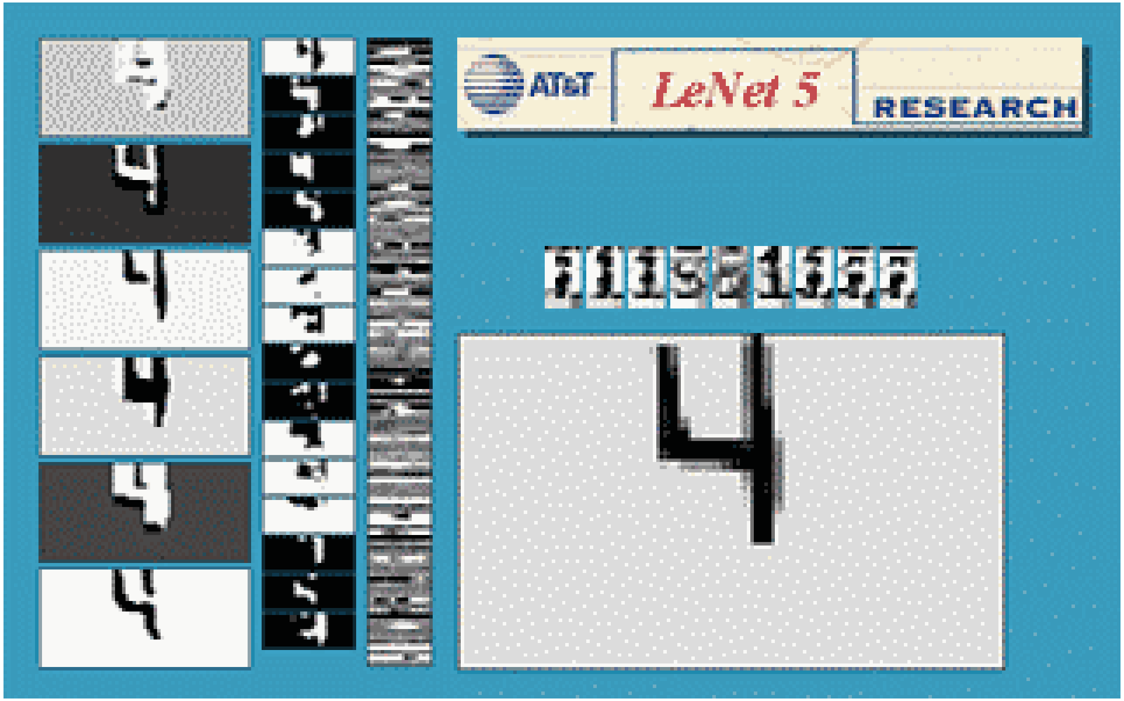
Dilation=2

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

Equivariance

Formally, 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 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.

Convolutions as matrix multiplications

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

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

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:

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

- 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,

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

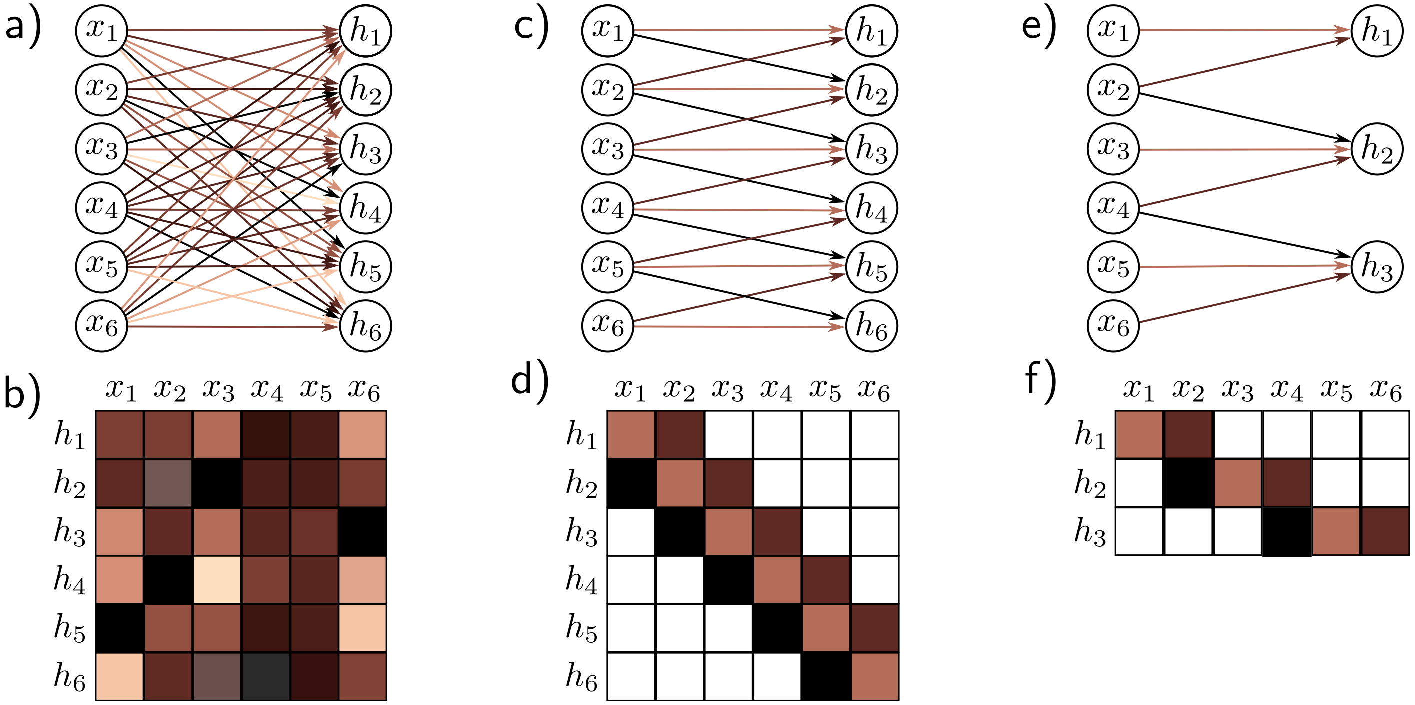
which we can reshape to a 2×2 matrix to obtain $\mathbf{x} \circledast \mathbf{u}$.

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:

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



Fully connected vs convolutional layers.

Credits: Simon J.D. Prince, Understanding Deep Learning, 2023.

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 $\mathbf{x} \in \mathbb{R}^{C \times (rh) \times (sw)}$.

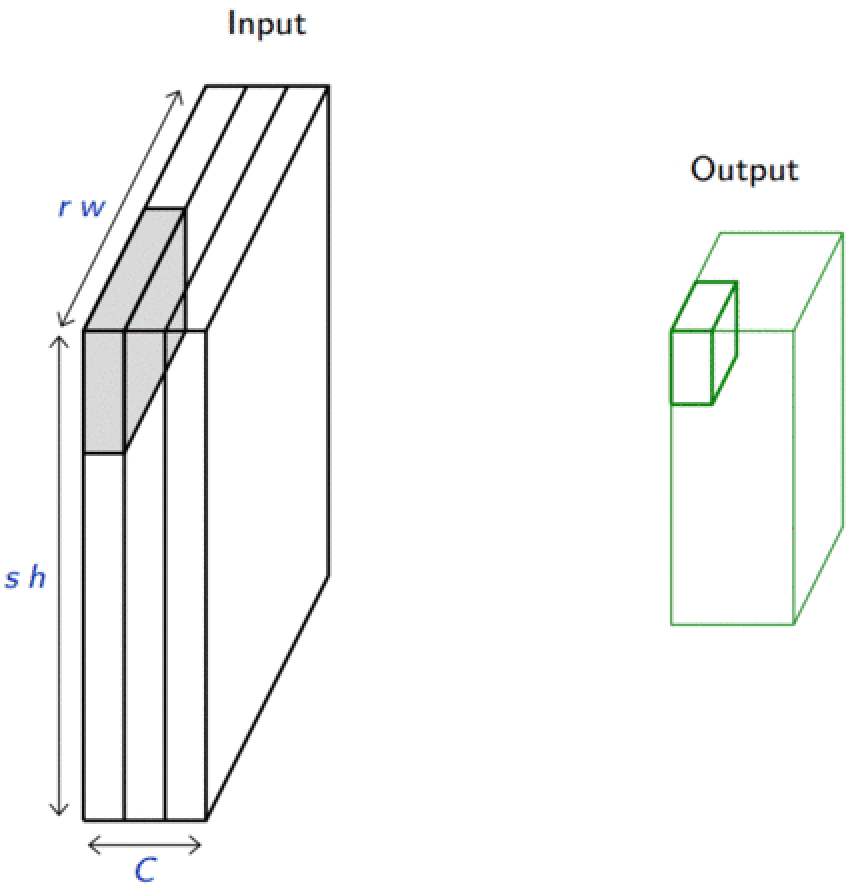
- Max-pooling produces a tensor $\mathbf{o} \in \mathbb{R}^{C \times r \times s}$ such that

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

- 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.



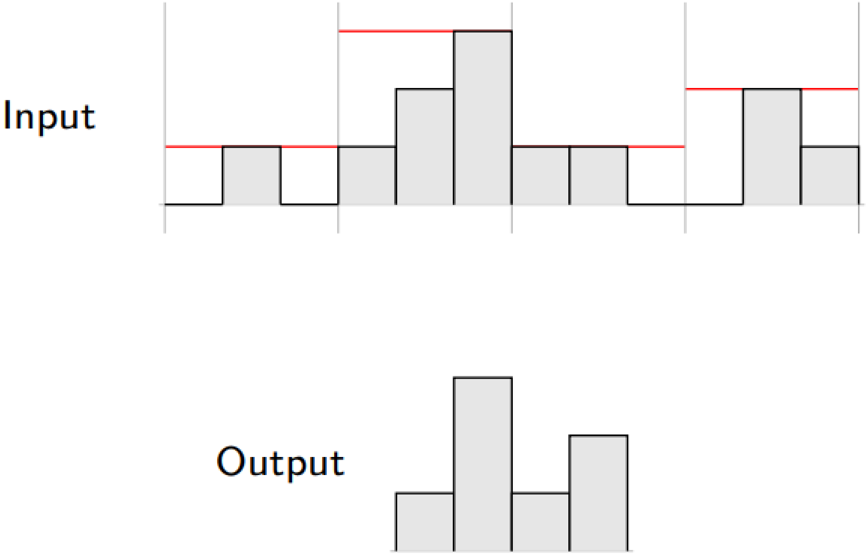
—

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

Invariance

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

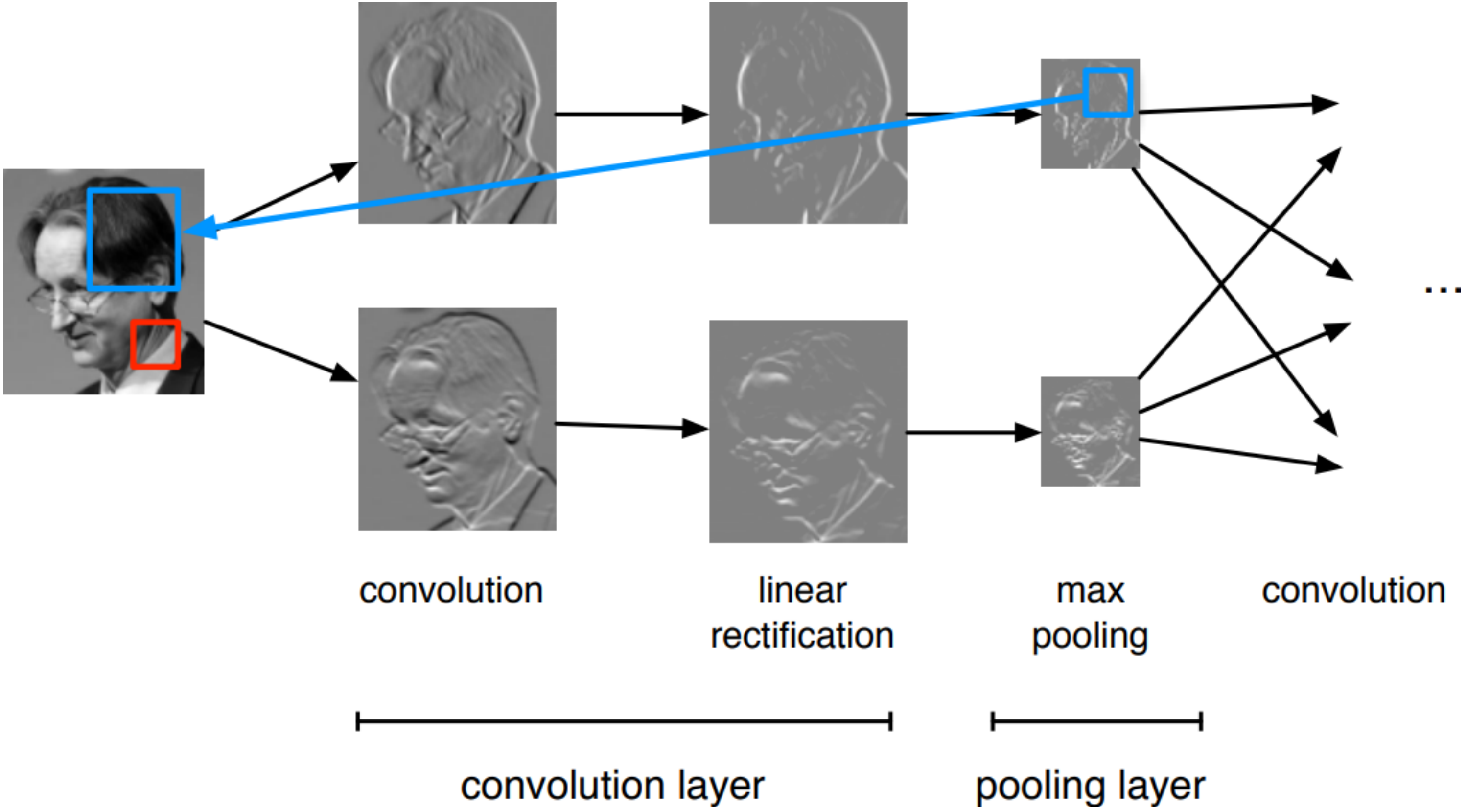
Pooling layers provide invariance to any permutation inside one cell, which results in (pseudo-)invariance to local translations.



—
Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

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:

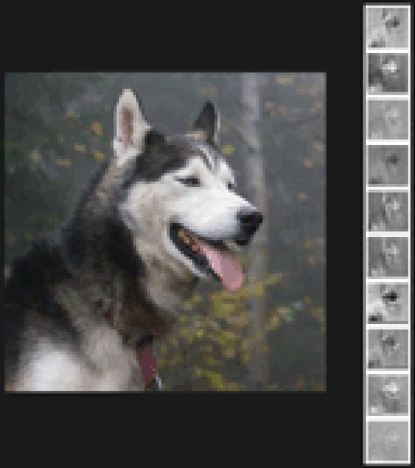
$$\texttt{INPUT} \rightarrow [[\texttt{CONV} \rightarrow \texttt{ReLU}]{*}N \rightarrow \texttt{POOL?}]{*}M \rightarrow [\texttt{FC} \rightarrow \texttt{ReLU}]{*}K \rightarrow \texttt{FC}$$

where:

- $*$ indicates repetition;
- `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).

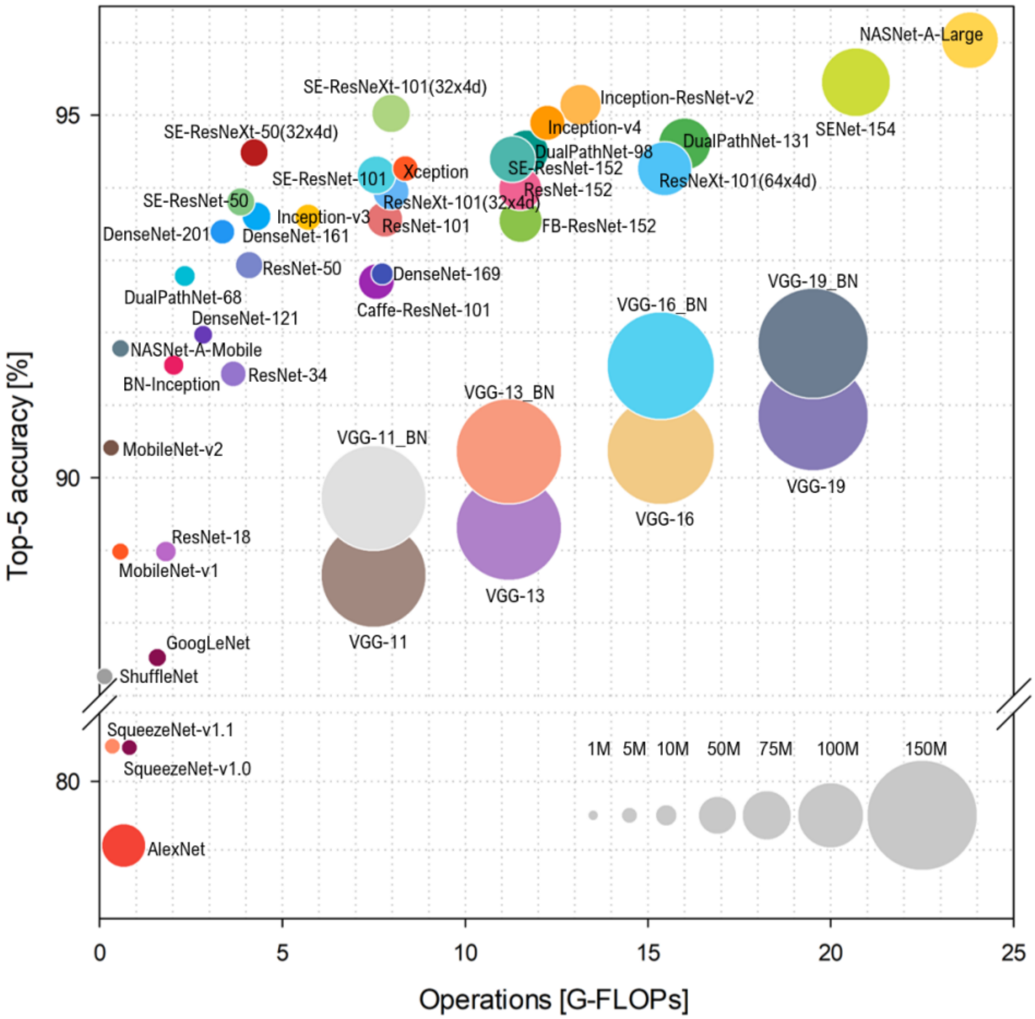
Some common architectures for convolutional networks following this pattern include:

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



(demo)

Architectures (some)

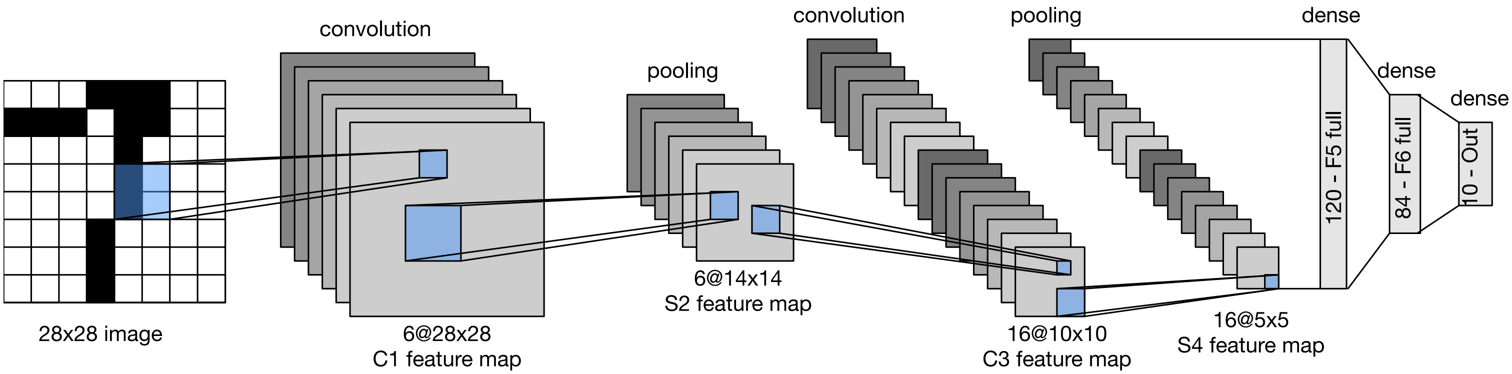


—

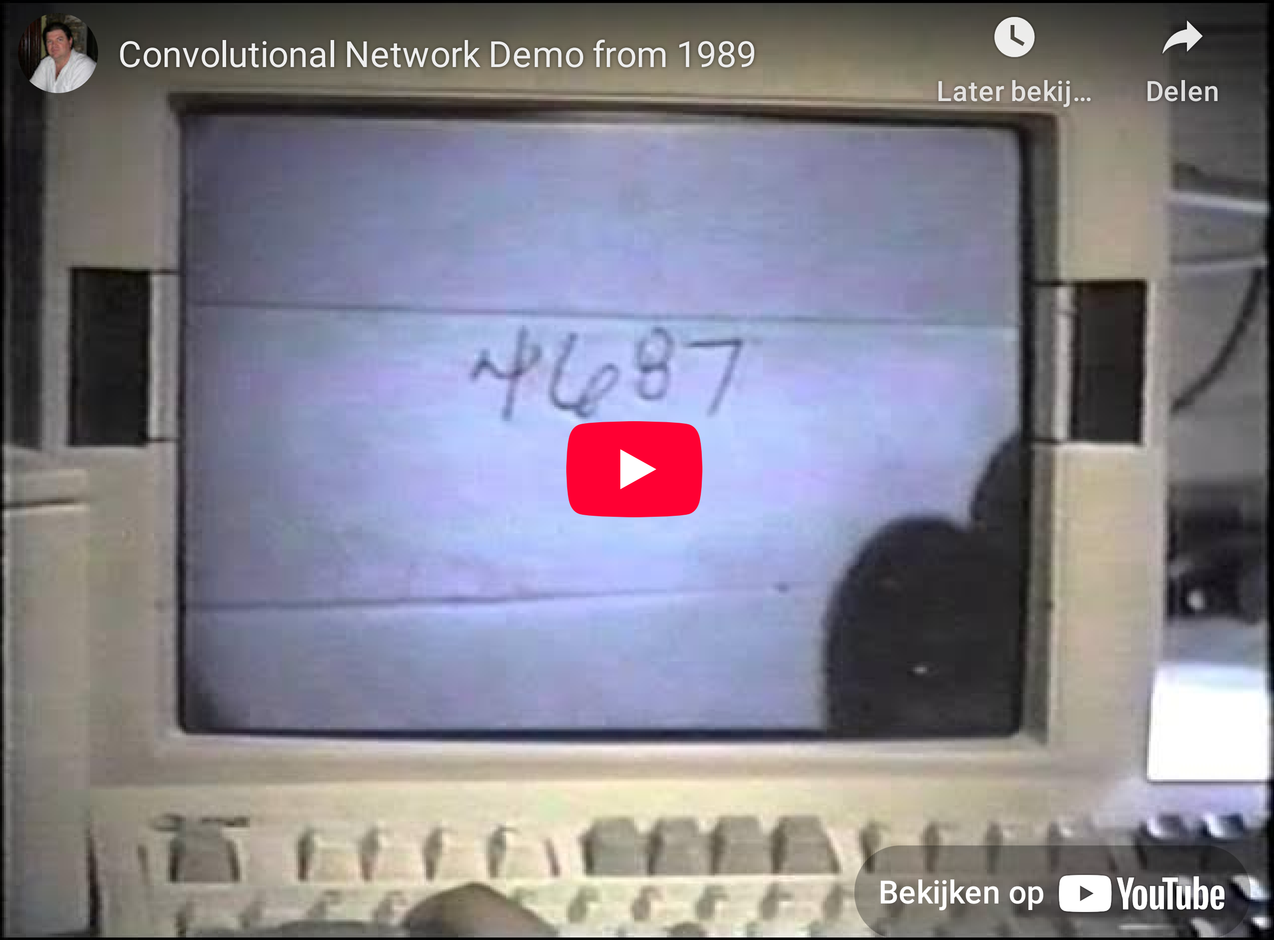
Credits: Bianco et al, 2018.

LeNet-5 (LeCun et al, 1998)

Composition of two $\texttt{CONV} + \texttt{POOL}$ layers, followed by a block of fully-connected layers.



—
Credits: Dive Into Deep Learning, 2020.

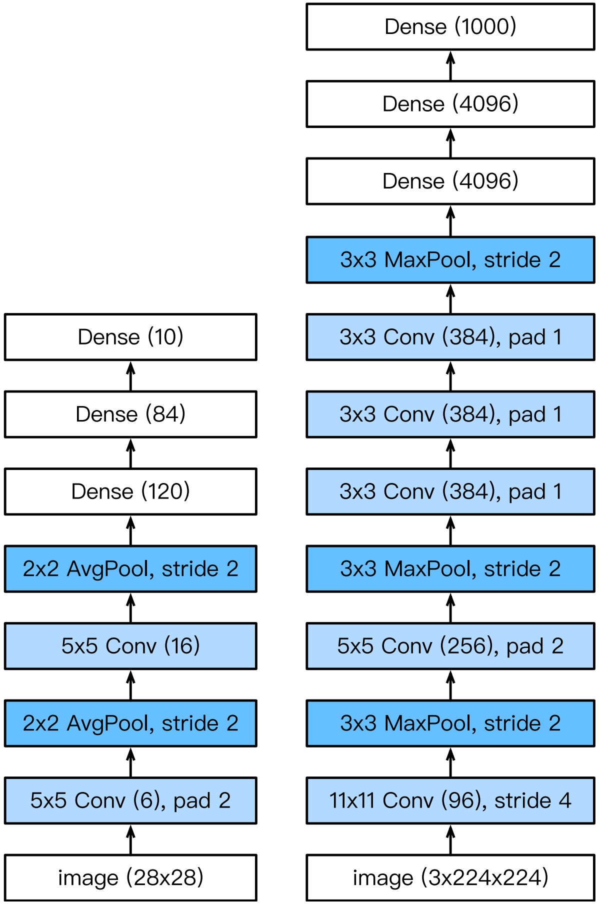


LeNet-1 (LeCun et al, 1993)

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.



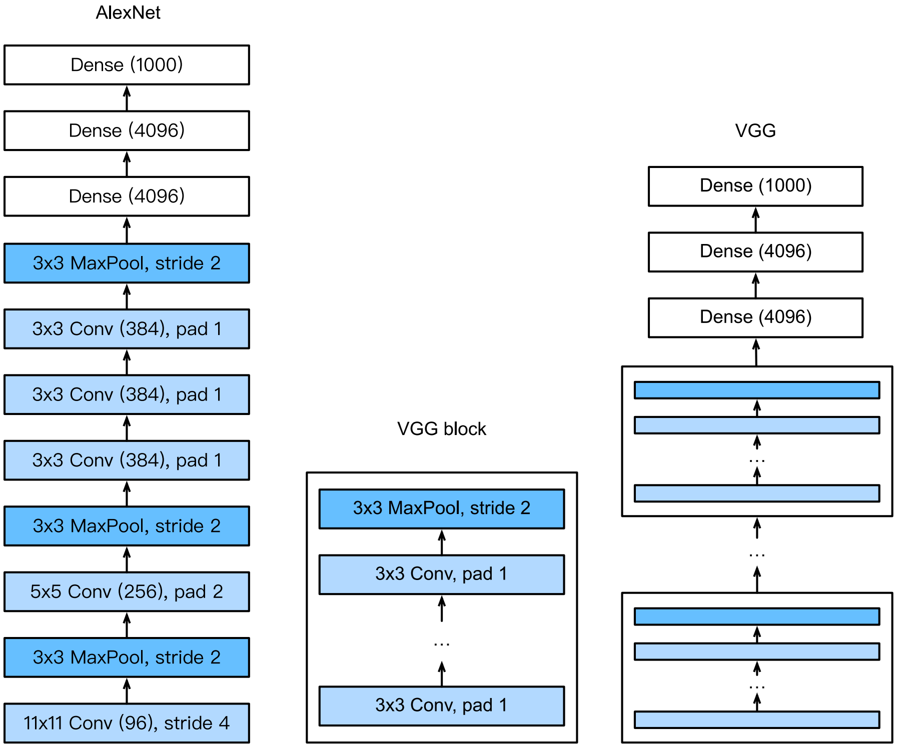
LeNet vs. AlexNet

—
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.

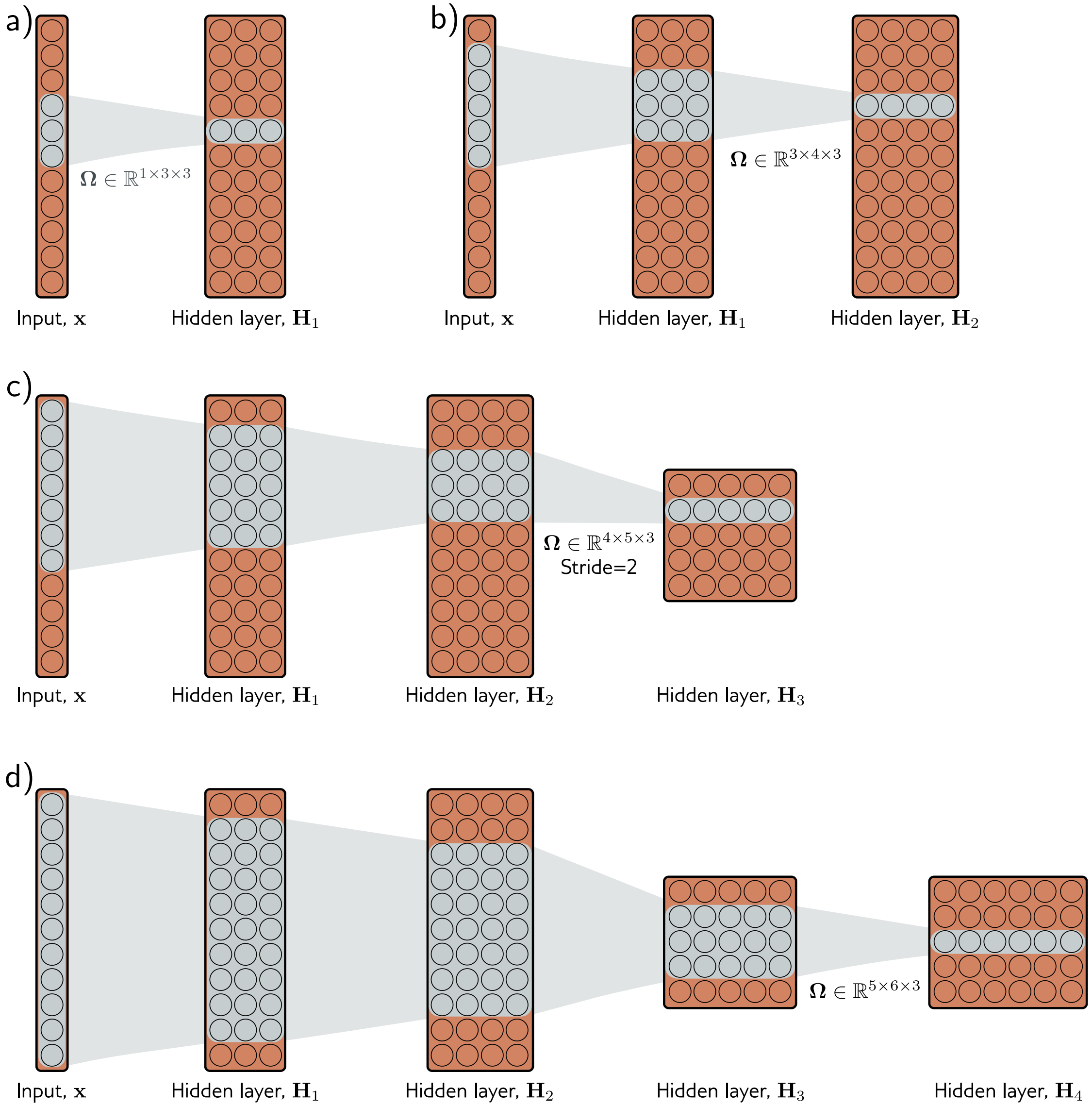


AlexNet vs. VGG

—
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 when chaining convolutional layers.
- It grows exponentially with depth when pooling layers (or strided convolutions) are interleaved with convolutional layers.

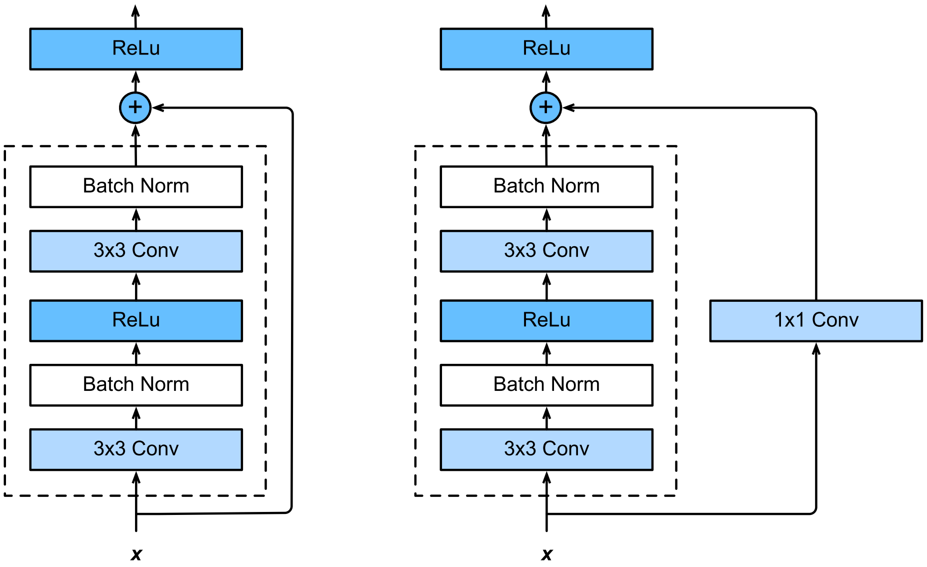

a)
$\Omega \in \mathbb{R}^{1\times3\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{H}_1$
b)
$\Omega \in \mathbb{R}^{3\times4\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{H}_1$
Hidden layer, $\mathbf{H}_2$
c)
$\Omega \in \mathbb{R}^{4\times5\times3}$
Stride=2
Input, $\mathbf{x}$
Hidden layer, $\mathbf{H}_1$
Hidden layer, $\mathbf{H}_2$
Hidden layer, $\mathbf{H}_3$
d)
$\Omega \in \mathbb{R}^{5\times6\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{H}_1$
Hidden layer, $\mathbf{H}_2$
Hidden layer, $\mathbf{H}_3$
Hidden layer, $\mathbf{H}_4$

—

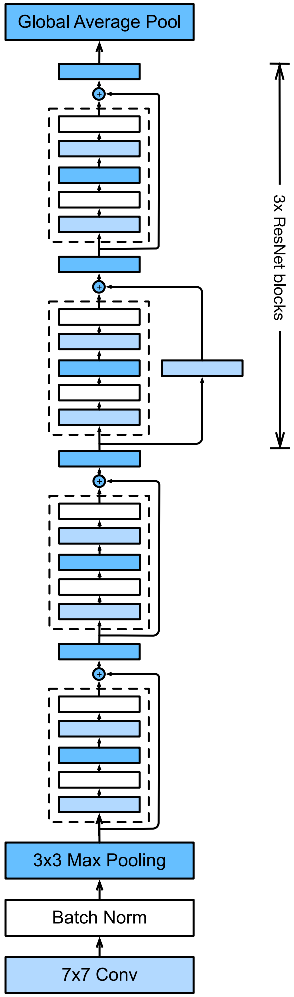
Credits: Simon J.D. Prince, Understanding Deep Learning, 2023.

ResNet (He et al, 2015)

Composition of convolutional and pooling layers organized in a stack of residual blocks. Extensions consider more residual blocks, up to a total of 152 layers (ResNet-152).

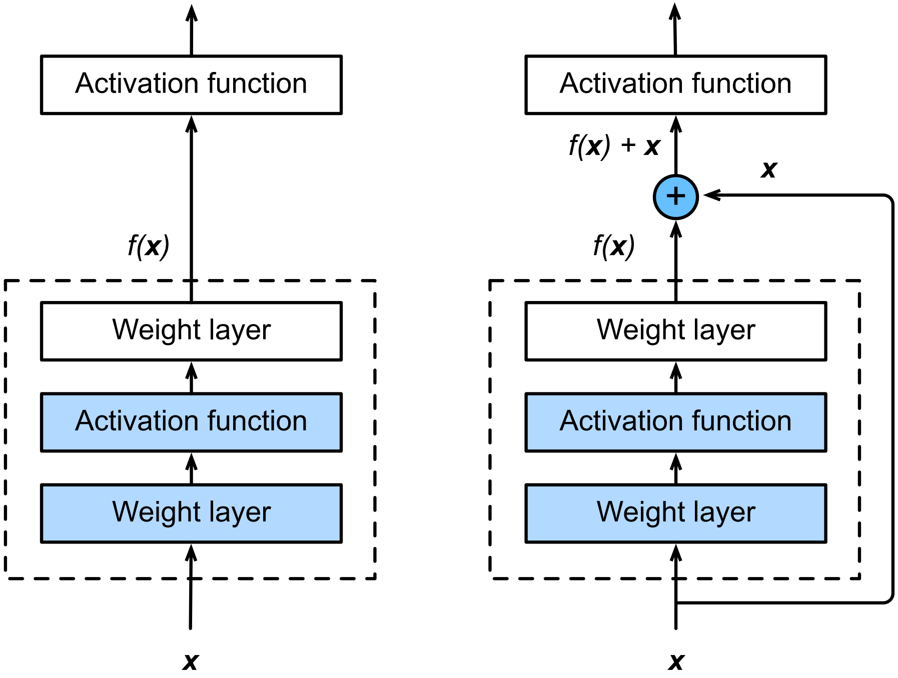


Regular ResNet block vs. ResNet block with 1×1 *convolution.*

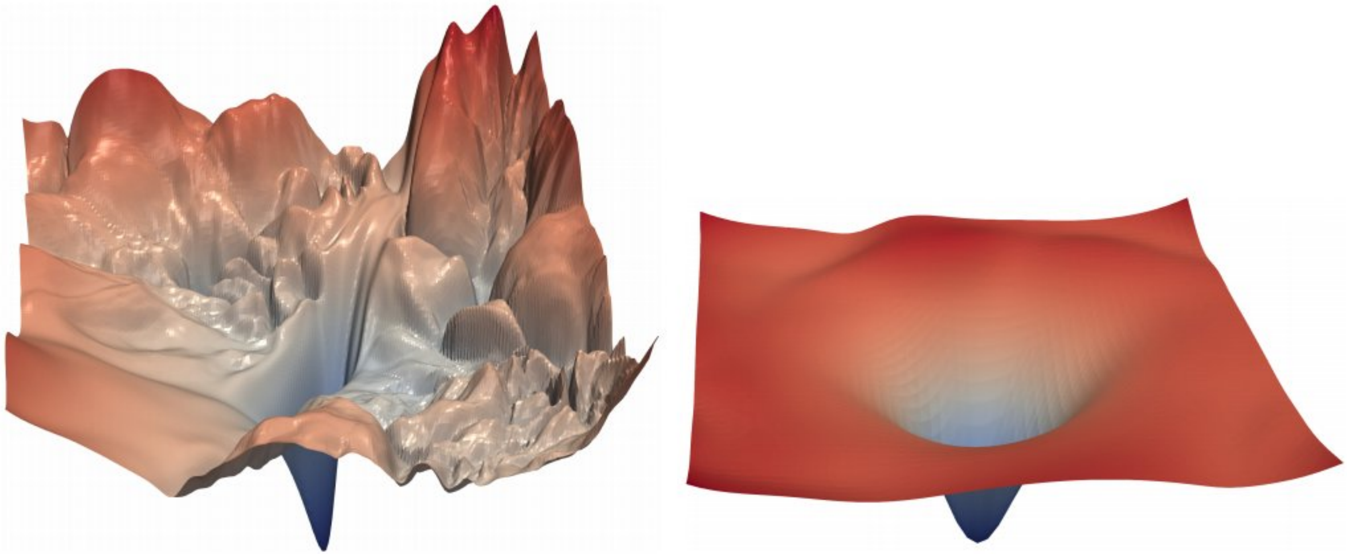


Credits: Dive Into Deep Learning, 2020.

Training networks of this depth is made possible because of the **skip connections** $\mathbf{h} = \mathbf{x} + f(\mathbf{x})$ in the residual blocks. They allow the gradients to shortcut the layers and pass through without vanishing.



—
Credits: Dive Into Deep Learning, 2020.

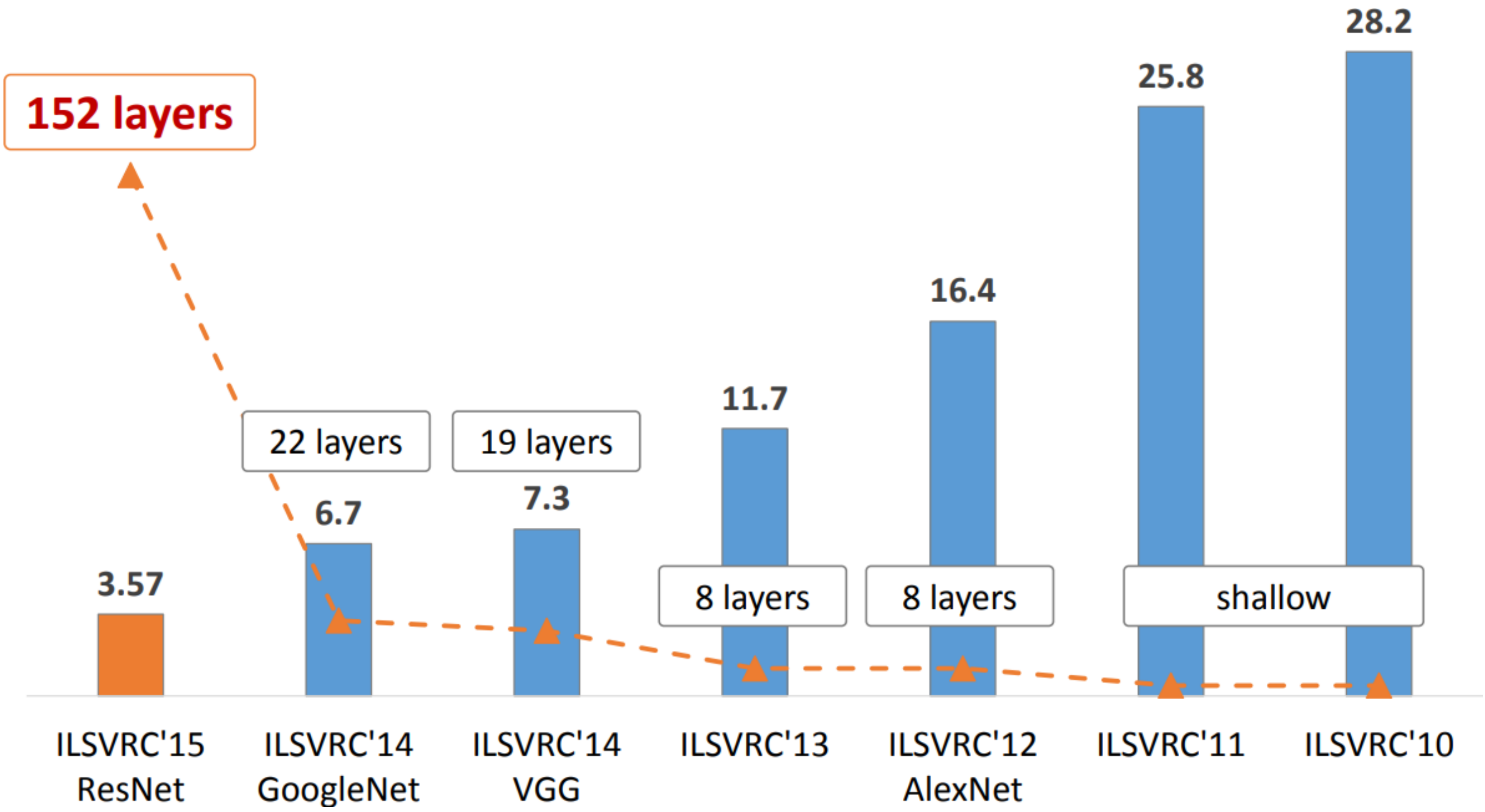


(a) without skip connections

(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 (Tan and Le, 2019)

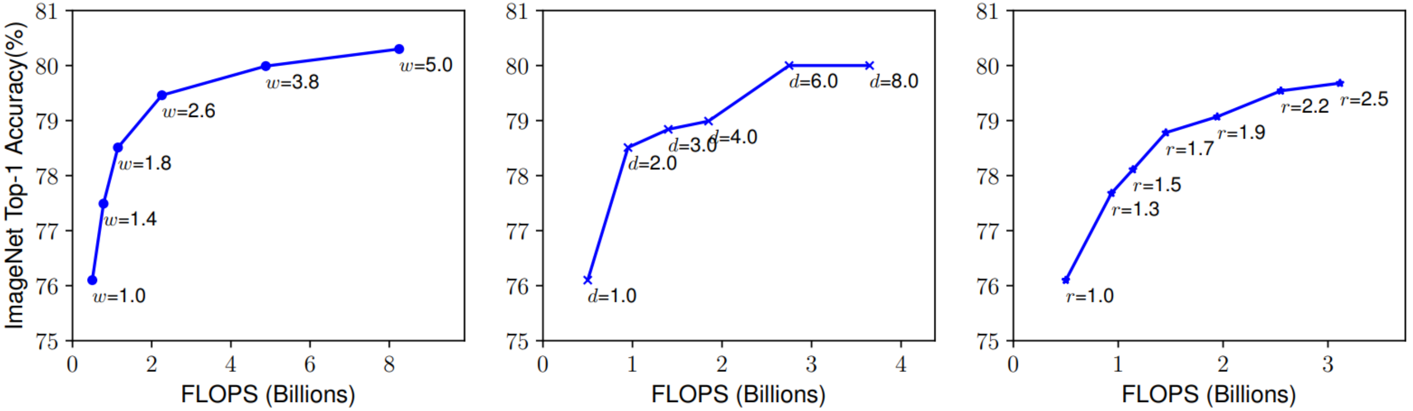


Figure 3. **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single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

—

Credits: Tan and Le, 2019.

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

—

Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

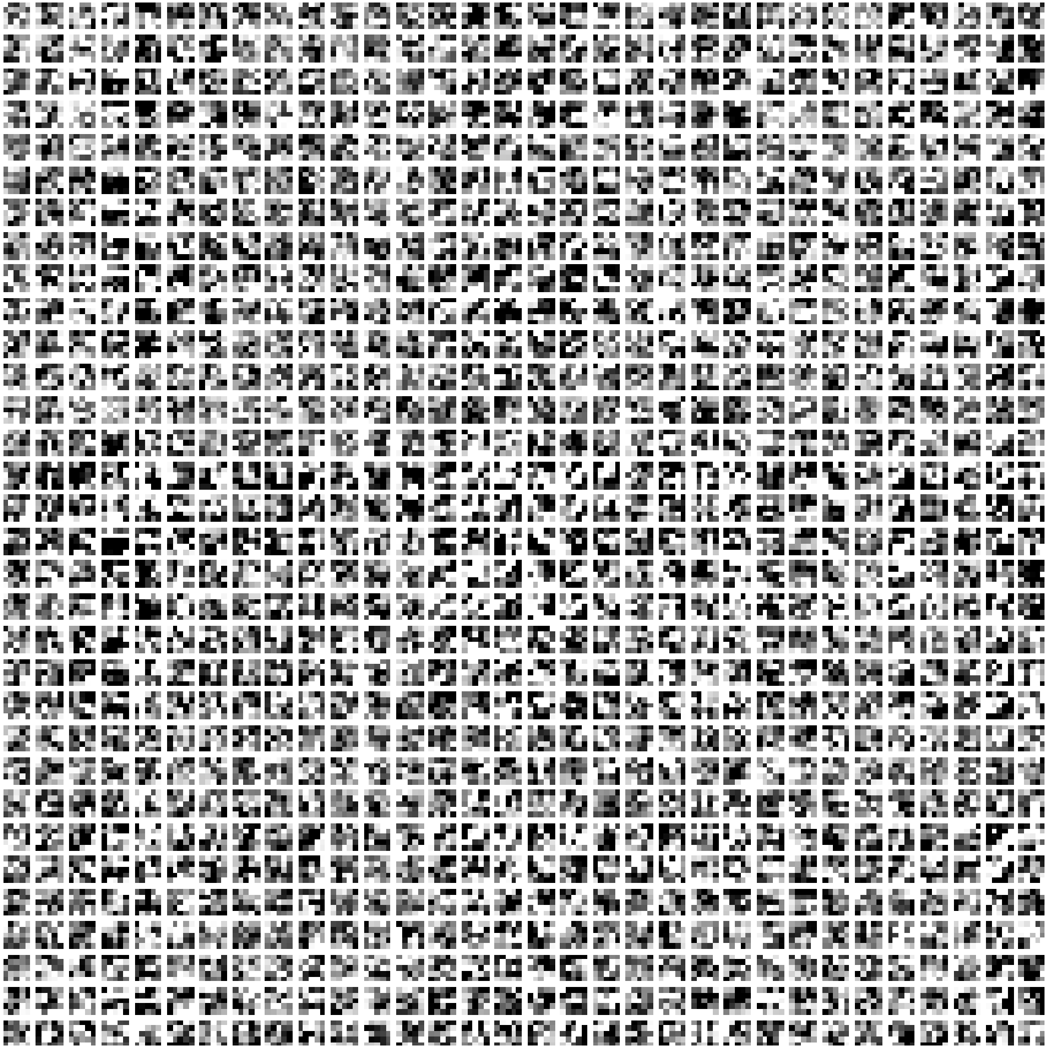
Looking at filters

LeNet's first convolutional layer, all filters.

—

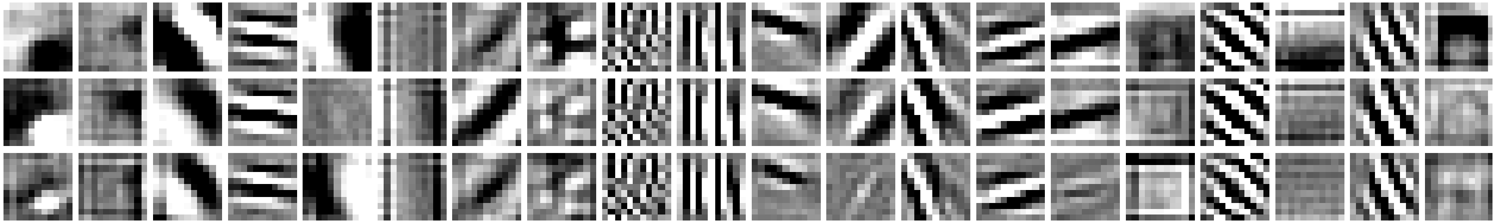
Credits: Francois Fleuret, EE559 Deep Learning, EPFL.

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.

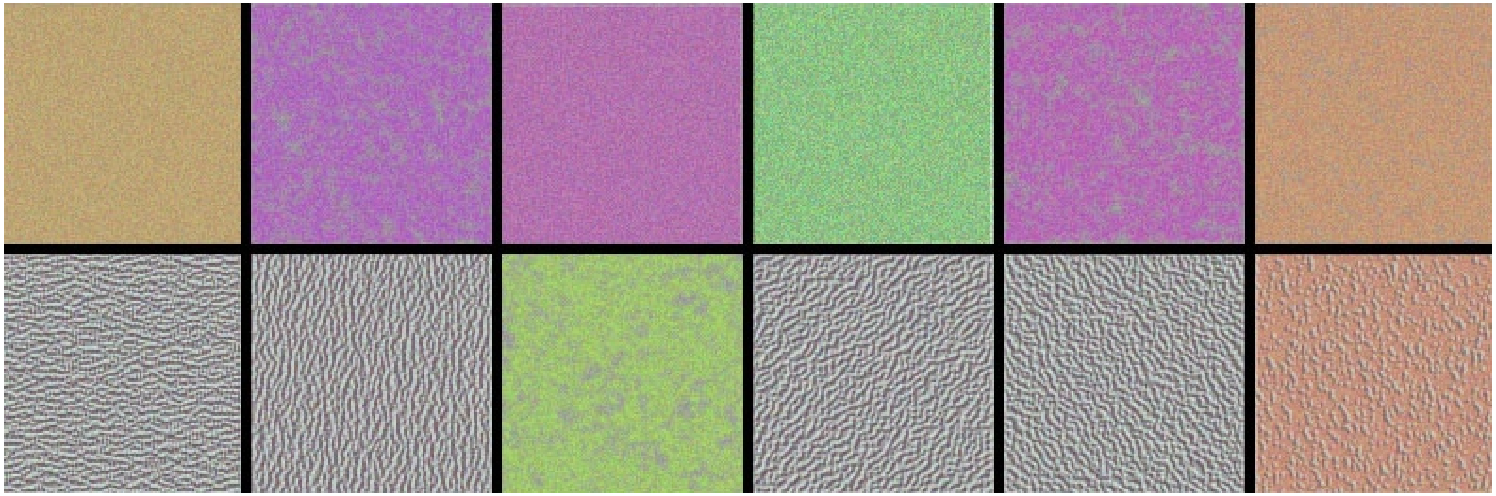
Maximum response samples

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:

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

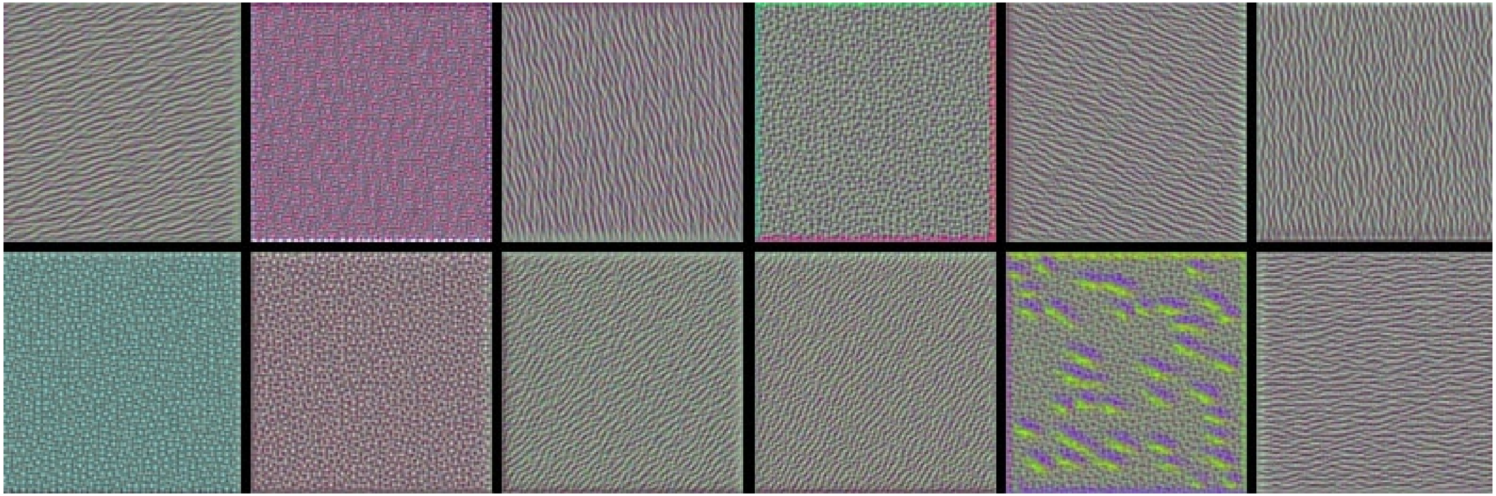
Here, $\mathcal{L}_{\ell,d}(\mathbf{x})$ represents the L2 norm of the activation $\mathbf{h}_{\ell,d}(\mathbf{x})$, $\mathbf{x}_0$ is the initial random input, $\mathbf{x}_{t+1}$ is the updated input at iteration $t+1$, and γ is the learning rate.



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

—

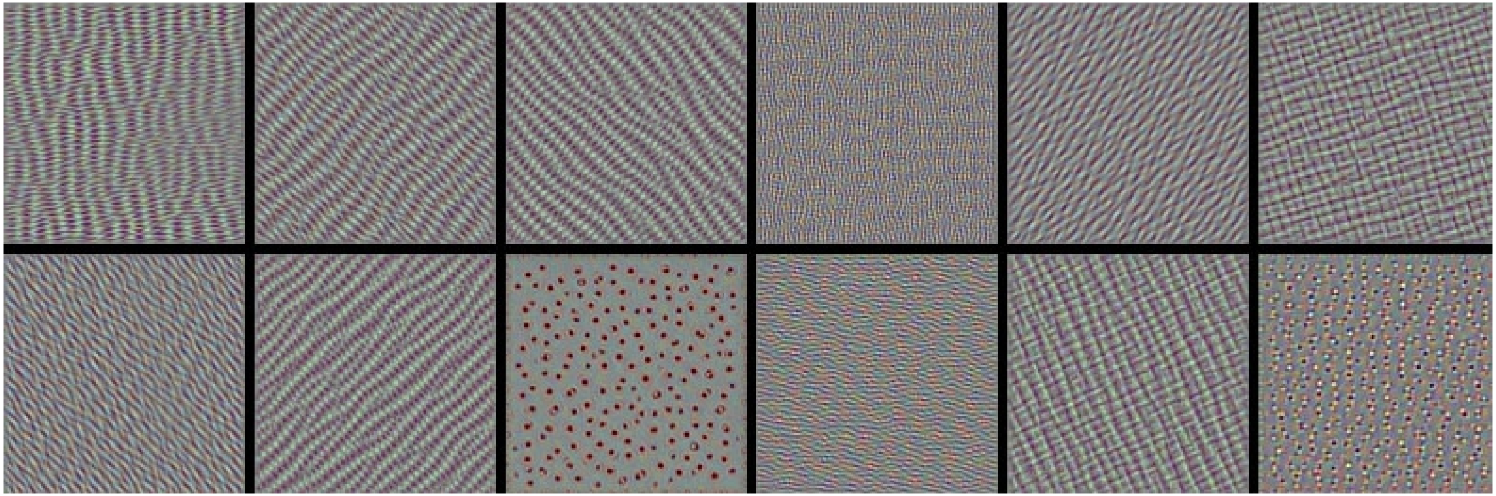
Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



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

—

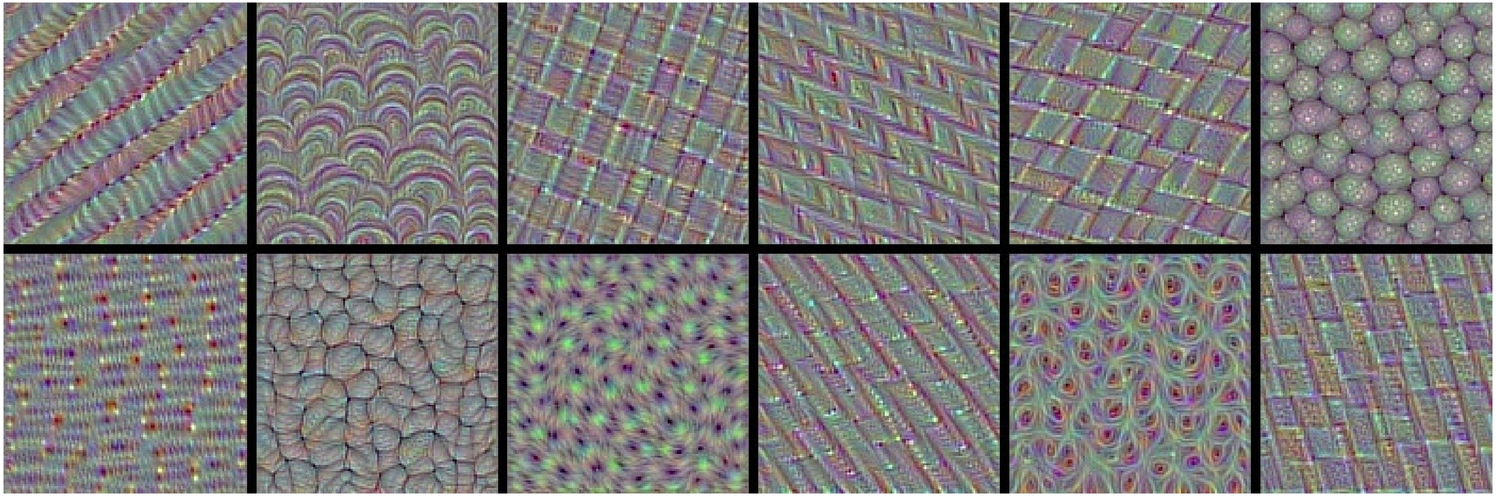
Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



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

—

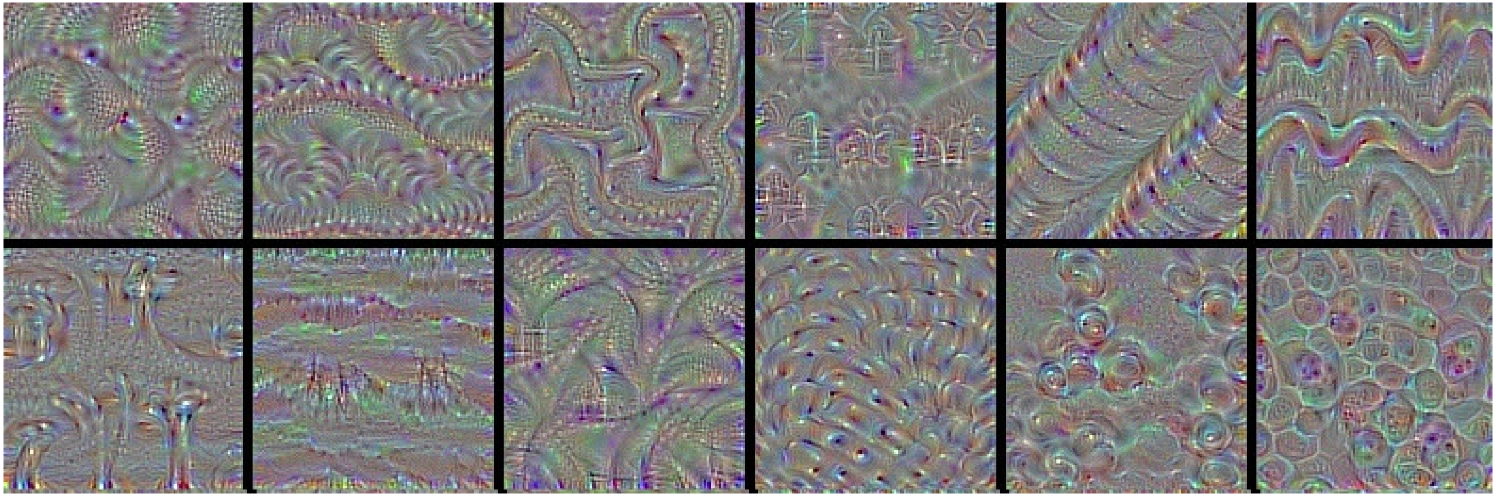
Credits: Francois Chollet, How convolutional neural networks see the world, 2016.



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

—

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



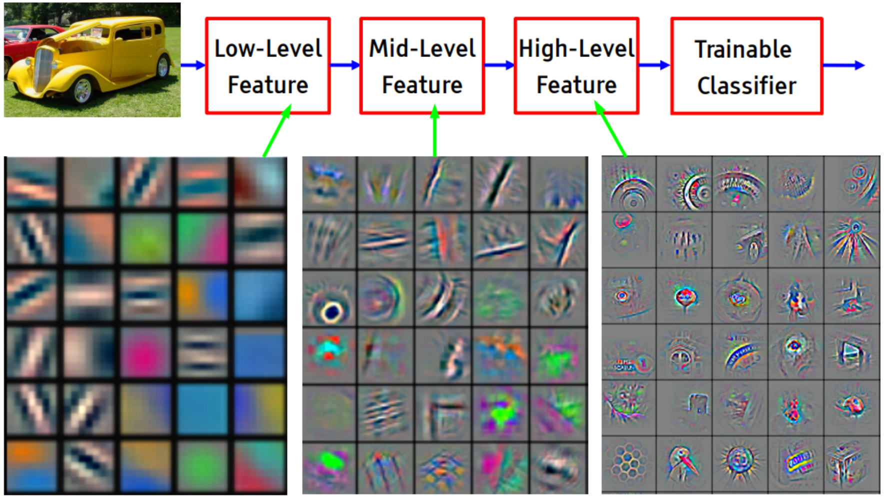
VGG-16, convolutional layer 5-1, a few of the 512 filters

—

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

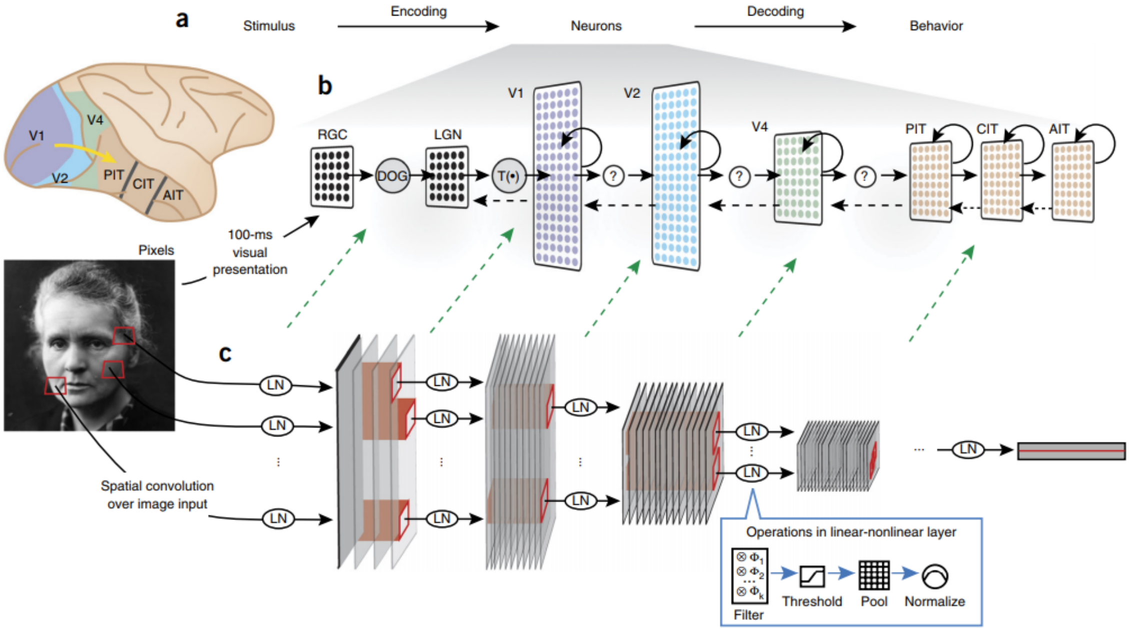
The network appears to learn a **hierarchical composition** of patterns:

- The first layers seem to encode basic features such as direction and color.
- These basic features are then combined to form more complex textures, such as grids and spots.
- Finally, these textures are further combined to create increasingly intricate patterns.



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

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.