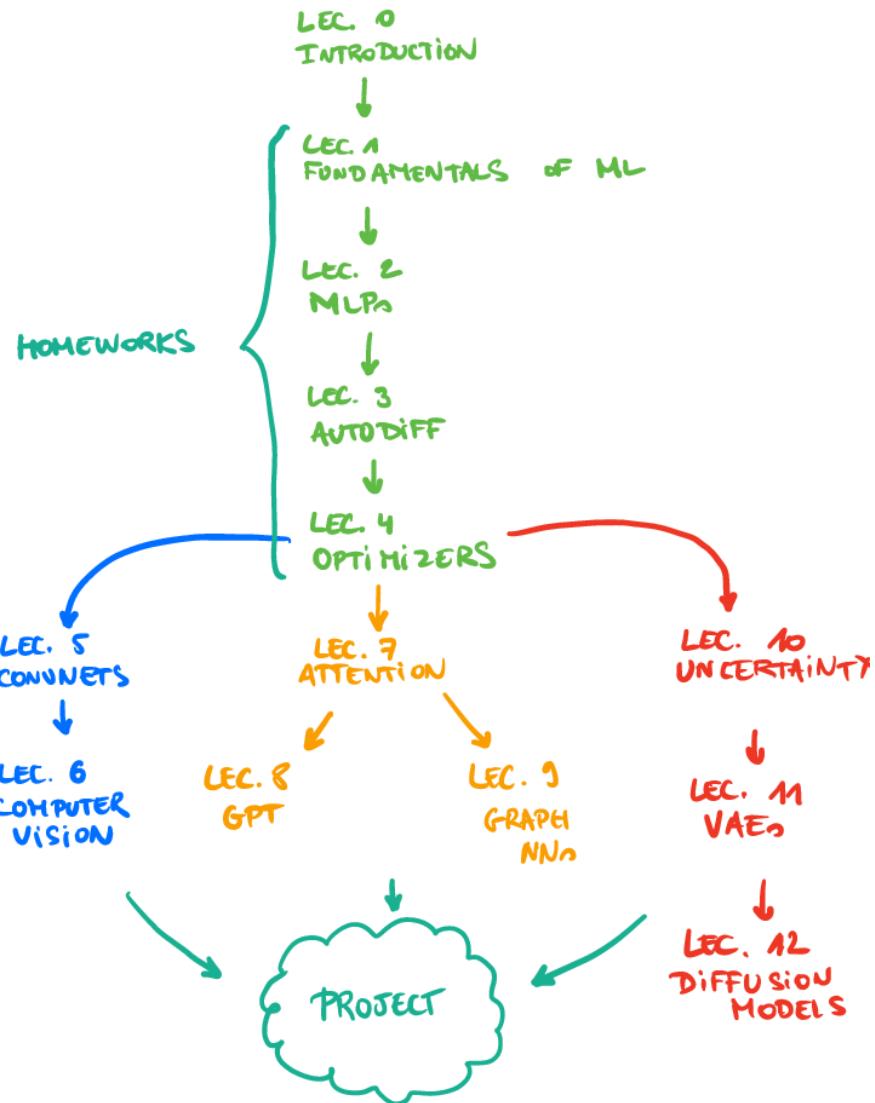


# Deep Learning

Lecture 5: Convolutional networks

Prof. Gilles Louppe  
[g.louppe@uliege.be](mailto:g.louppe@uliege.be)



# Today

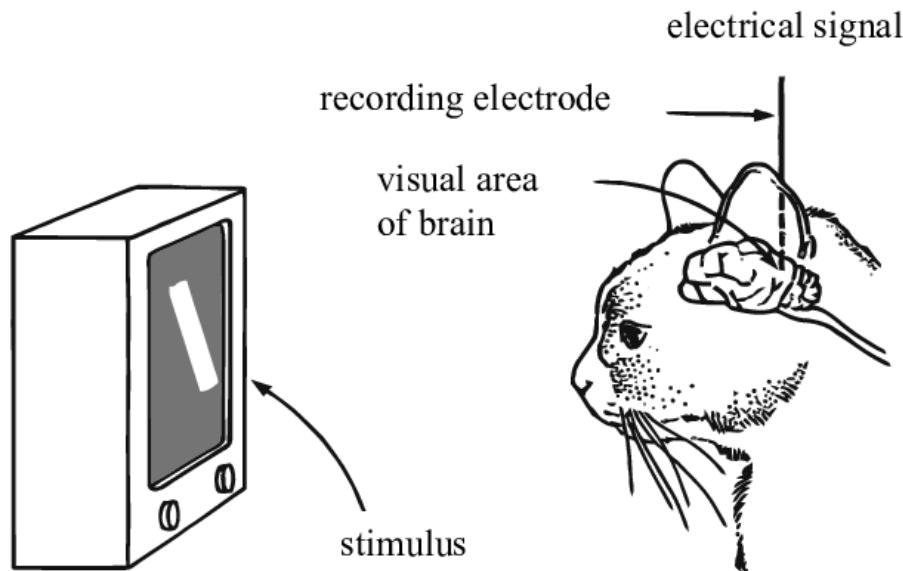
How to make neural networks see?

- 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 Cat Experiment



Later bekij...  
...

Delen





04 2 Simple Complex Cells



Later bekij...  
...

Delen

0H09M41S



SP



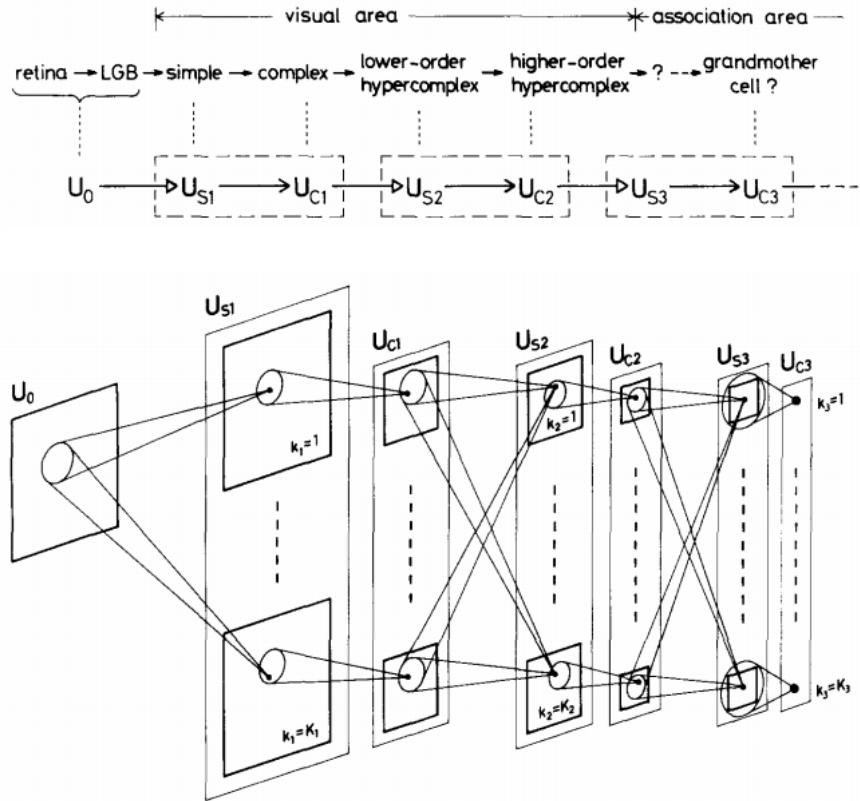
# Inductive biases

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

- Locality
- Invariance to translation
- Hierarchical compositionality

# Neocognitron

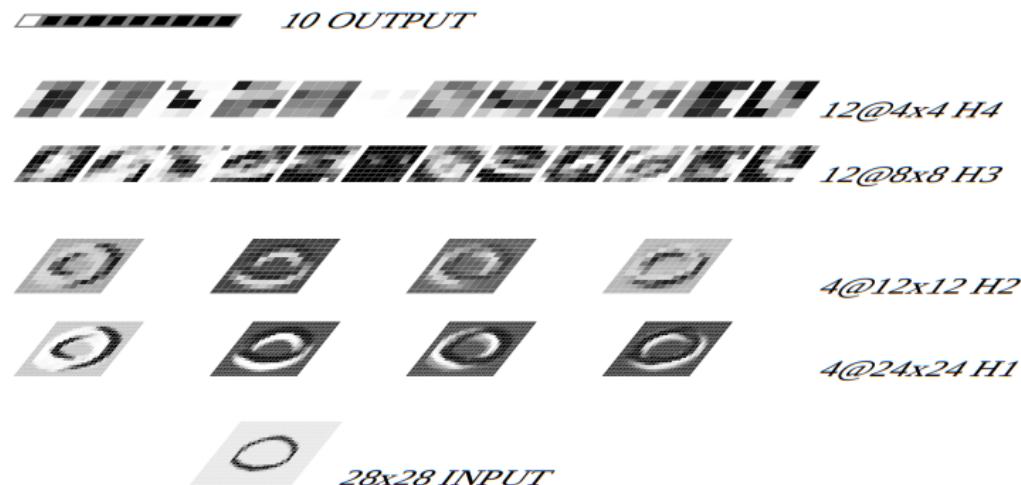
In 1980, Fukushima proposes a direct neural network implementation of the hierarchy model of the visual nervous system of Hubel and Wiesel.



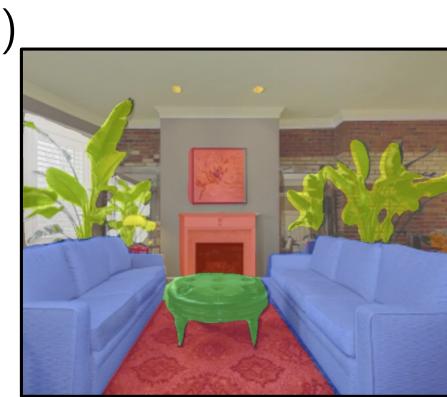
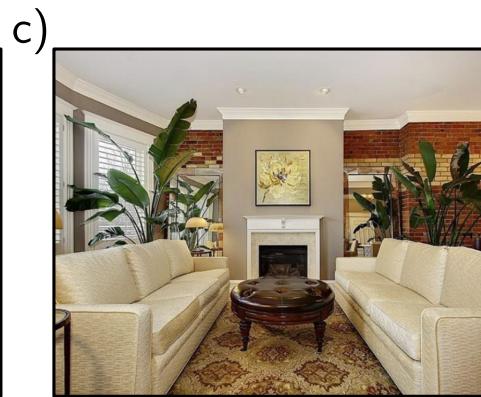
- Built upon **convolutions** and enables the composition of a **feature hierarchy**.
- Biologically-inspired training algorithm, which proves to be largely **inefficient**.

# Convolutional networks

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



# Convolutions



Invariance and equivariance for translation.

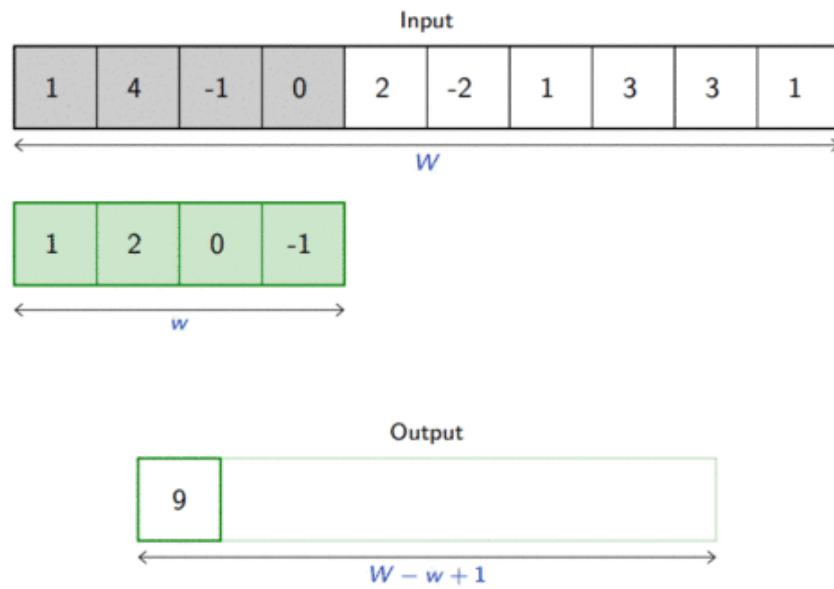
Formally, a function  $f$  is

- **invariant** to  $g$  if  $f(g(\mathbf{x})) = f(\mathbf{x})$
- **equivariant** to  $g$  if  $f(g(\mathbf{x})) = g(f(\mathbf{x})).$

How can we enforce these properties in a neural network?

# Convolutional layers

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



## 1d convolution

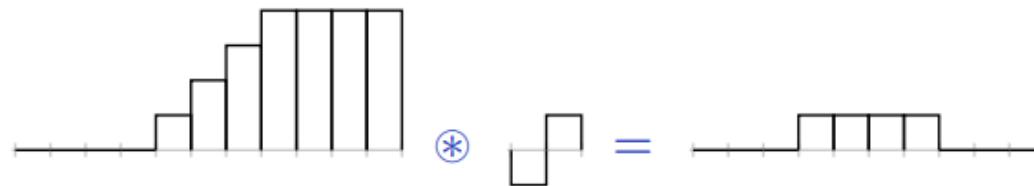
For the one-dimensional input  $\mathbf{x} \in \mathbb{R}^W$  and the convolutional kernel  $\mathbf{u} \in \mathbb{R}^w$ , the discrete convolution  $\mathbf{x} \circledast \mathbf{u}$  is a vector of size  $W - w + 1$  such that

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

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

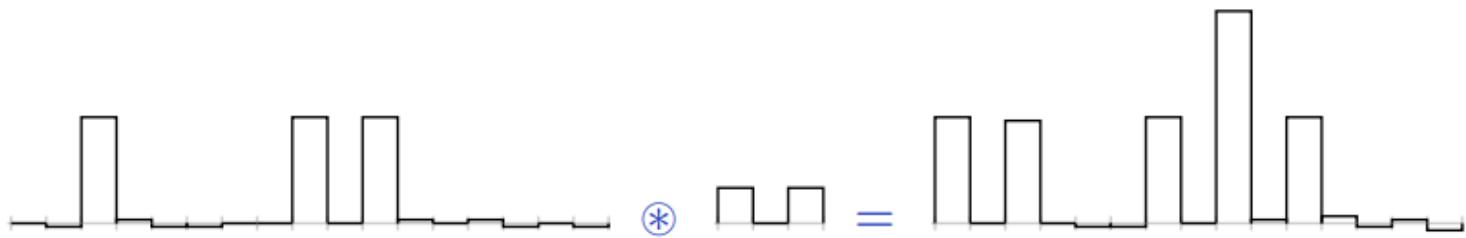
Convolutions can implement differential operators:

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



or crude template matchers:

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



## 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

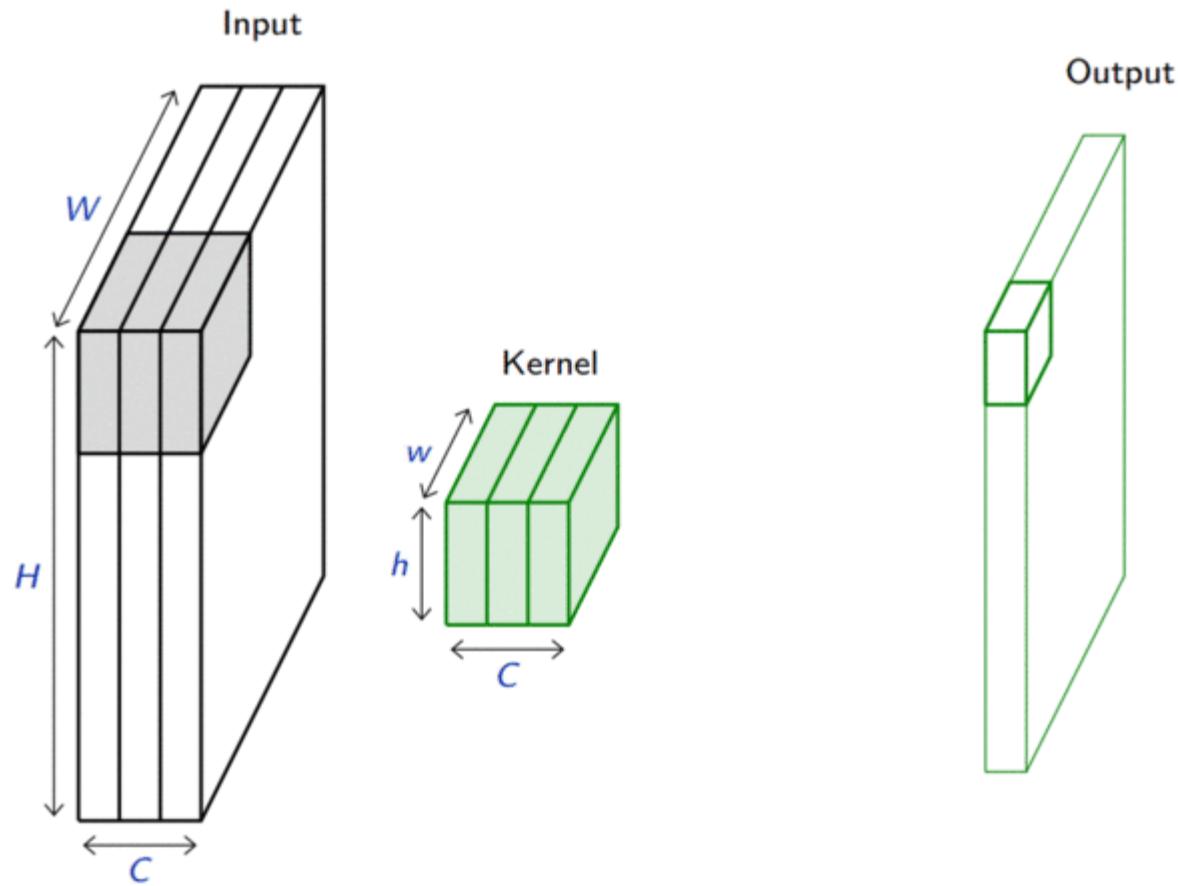
The 2d convolution can be extended to tensors with multiple channels.

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  $\mathbf{u}$  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  $\mathbf{o}$ .

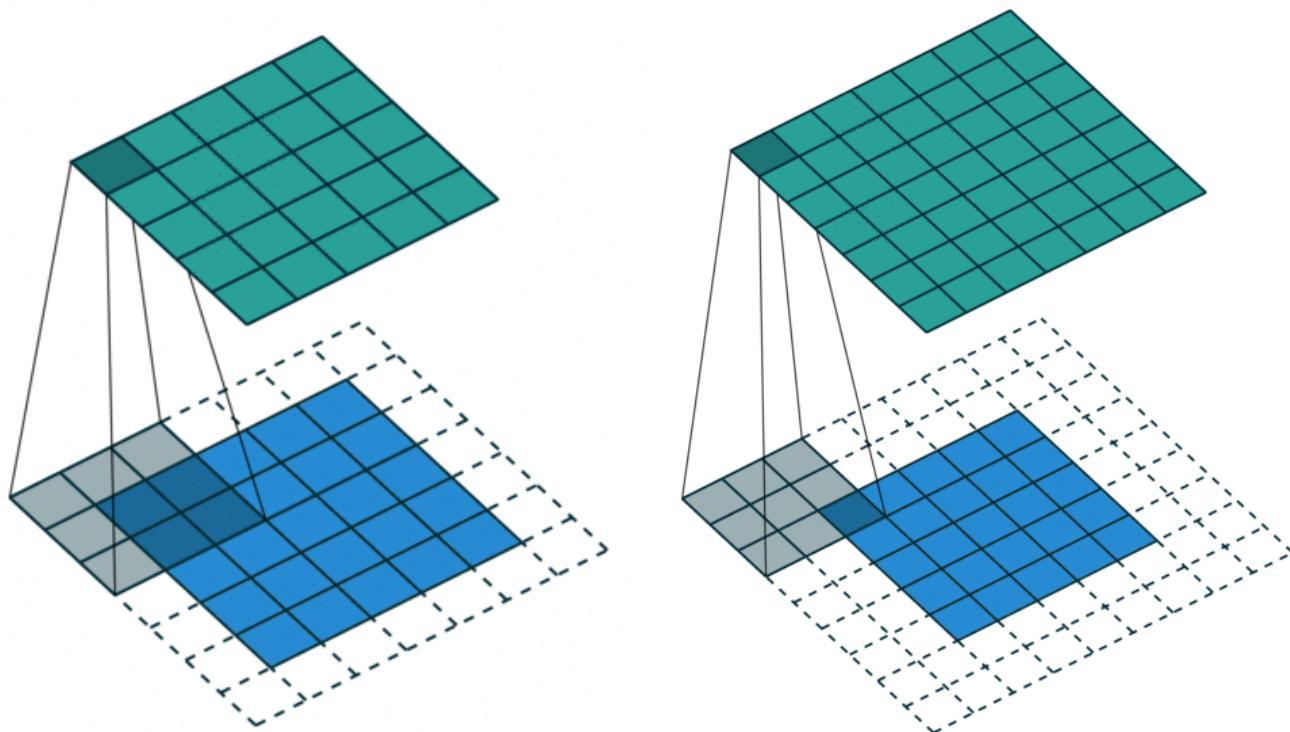


Convolutions have three additional parameters:

- The padding specifies the size of a zeroed frame added around 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.

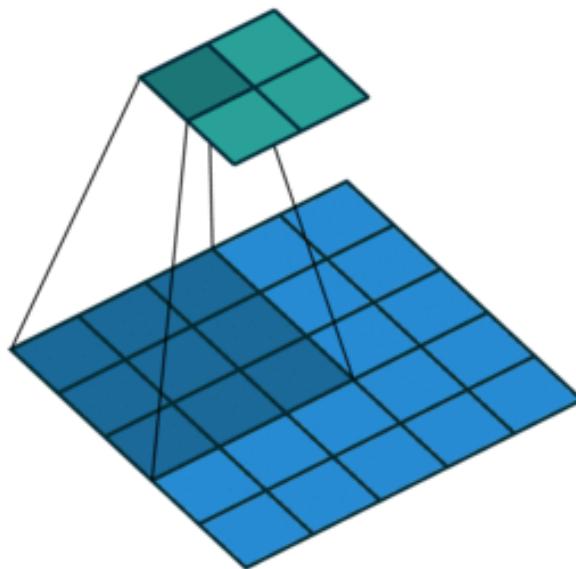
# Padding

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



## Strides

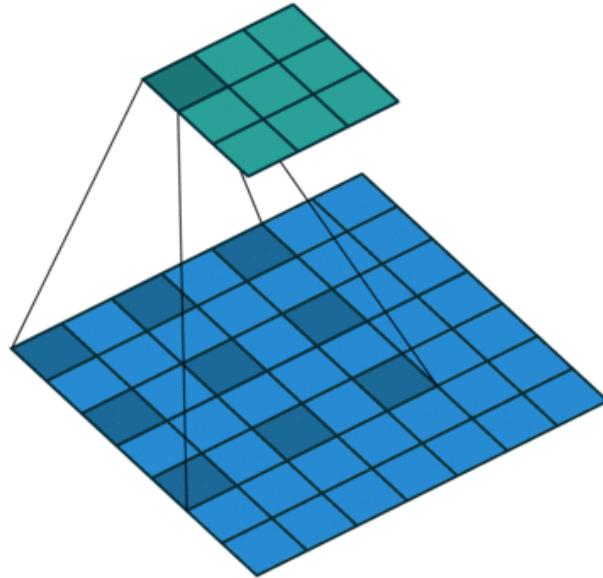
Stride is useful to reduce the spatial dimension of the feature map by a constant factor.



## Dilation

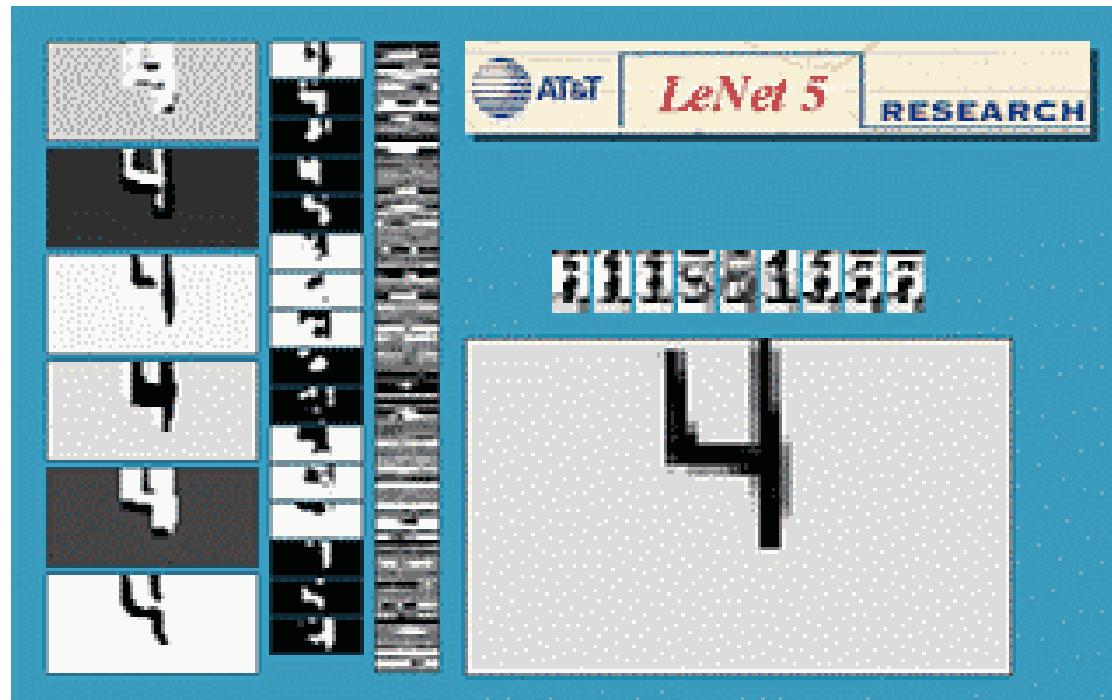
The dilation modulates the expansion of the kernel support by adding rows and columns of zeros between coefficients.

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



# Equivariance

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.*

# 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:

$$v(\mathbf{x}) = (4 \quad 5 \quad 8 \quad 7 \quad 1 \quad 8 \quad 8 \quad 8 \quad 3 \quad 6 \quad 6 \quad 4 \quad 6 \quad 5 \quad 7 \quad 8)^T$$

Then,

$$\mathbf{U}v(\mathbf{x}) = (122 \quad 148 \quad 126 \quad 134)^T$$

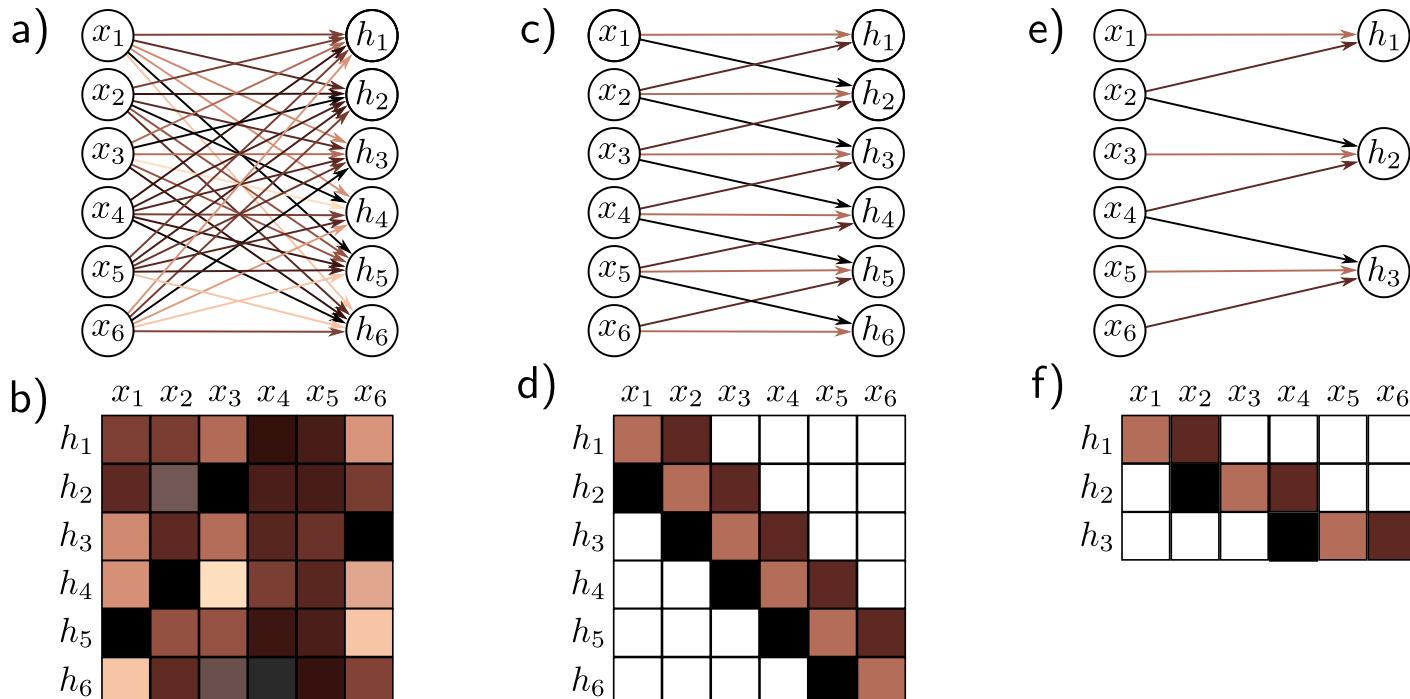
which we can reshape to a  $2 \times 2$  matrix to obtain  $\mathbf{x} \circledast \mathbf{u}$ .

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.

# 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 downscaling operation.

# Pooling

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

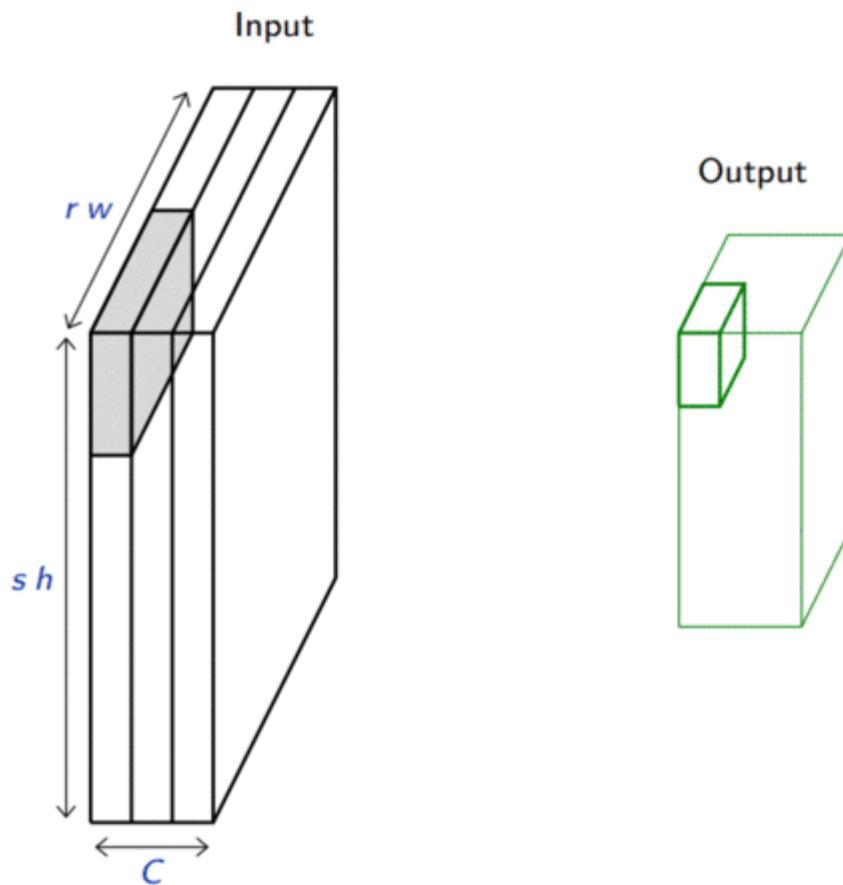
- Max-pooling produces a tensor  $\mathbf{o} \in \mathbb{R}^{C \times r \times s}$  such that

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

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

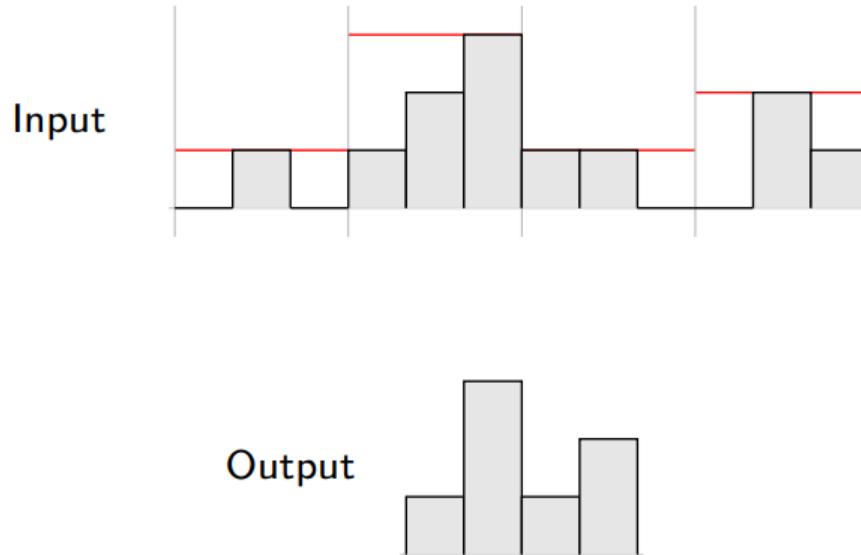
$$\mathbf{o}_{c,j,i} = \frac{1}{hw} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{c,rj+n,si+m}.$$

Pooling is very similar in its formulation to convolution.



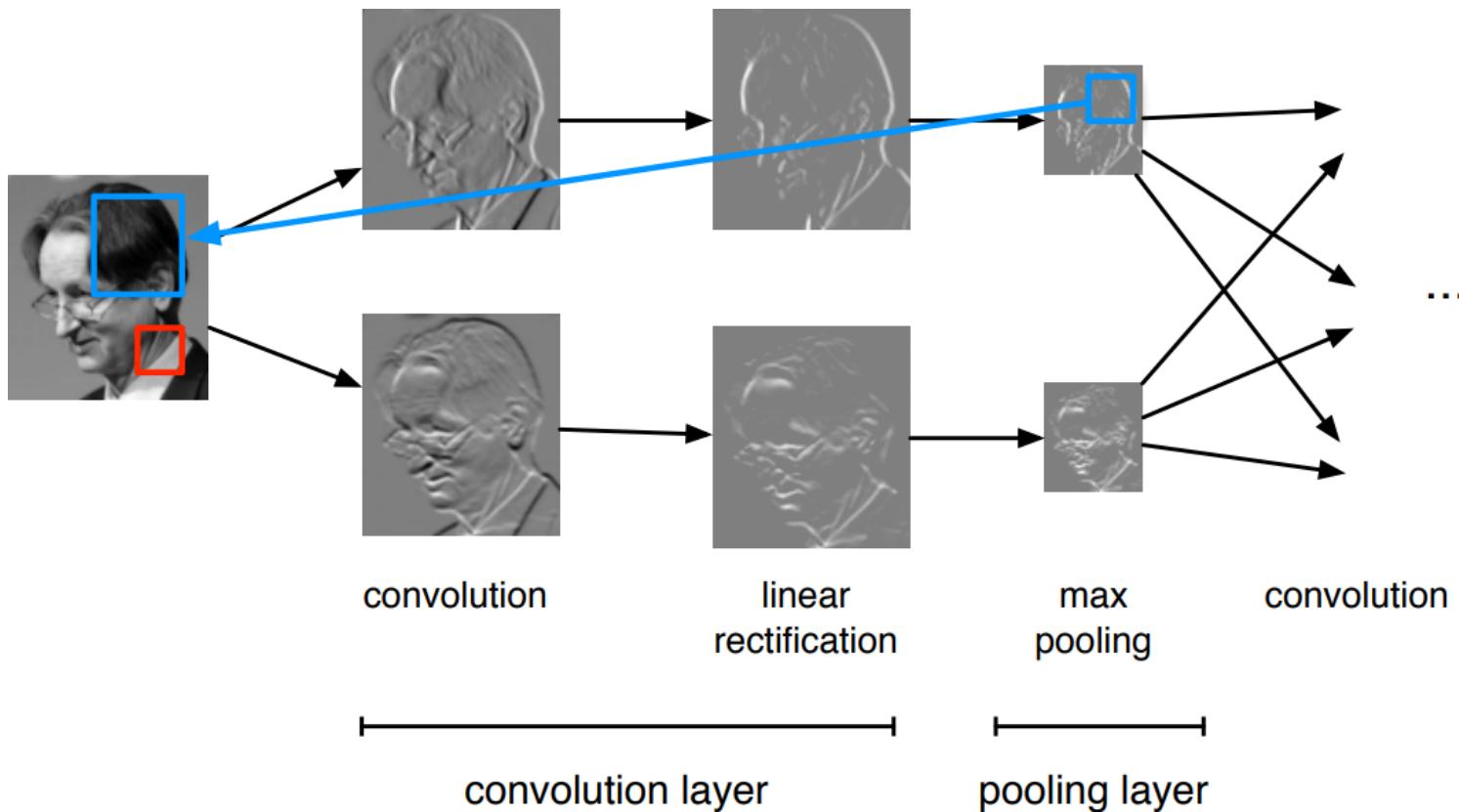
## Invariance

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



# 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:

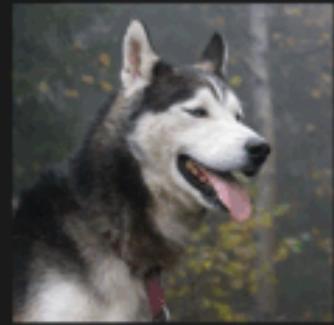
**INPUT** →  $[[\text{CONV} \rightarrow \text{ReLU}] * N \rightarrow \text{POOL?}] * M \rightarrow [\text{FC} \rightarrow \text{ReLU}] * K \rightarrow \text{FC}$

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).

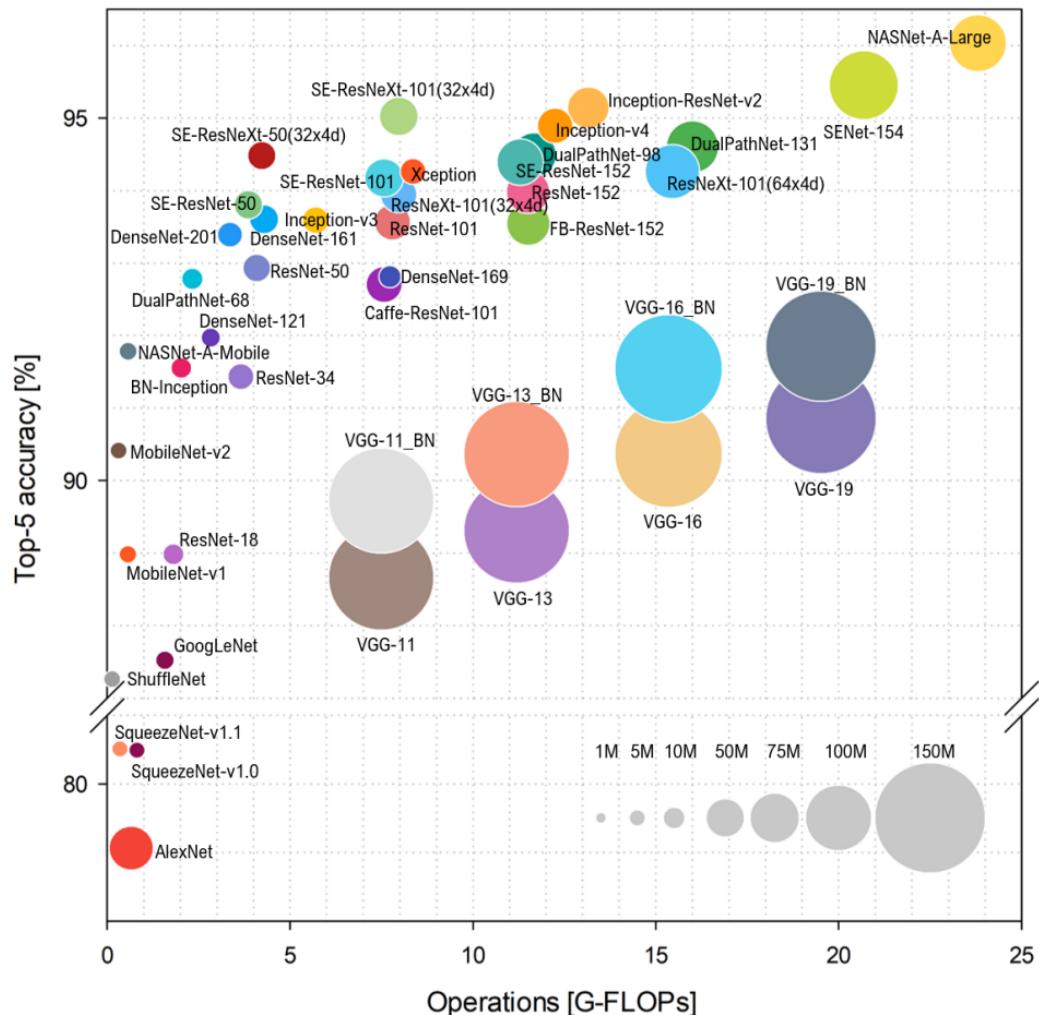
Some common architectures for convolutional networks following this pattern include:

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



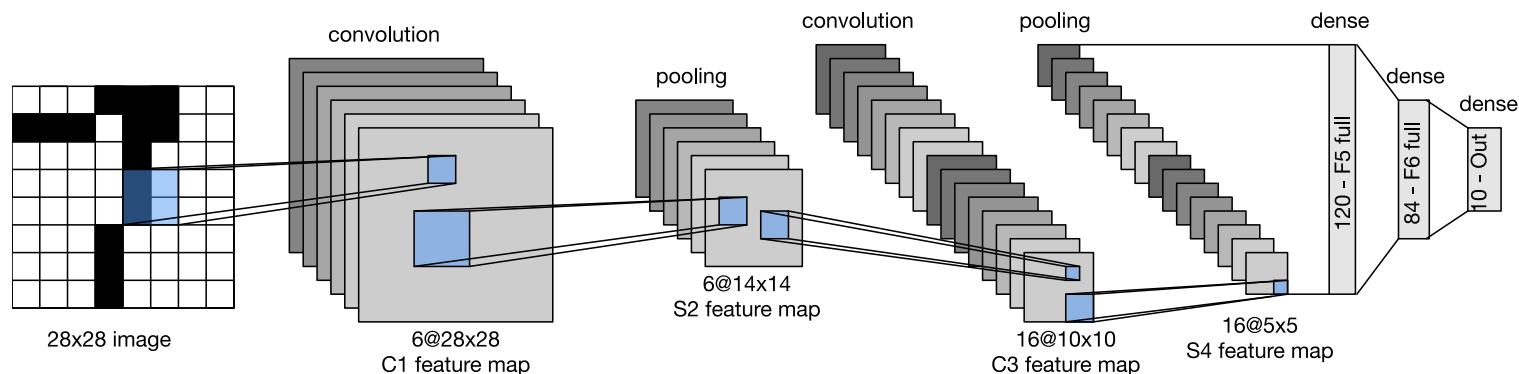
(demo)

# Common architectures



## LeNet-5 (LeCun et al, 1998)

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





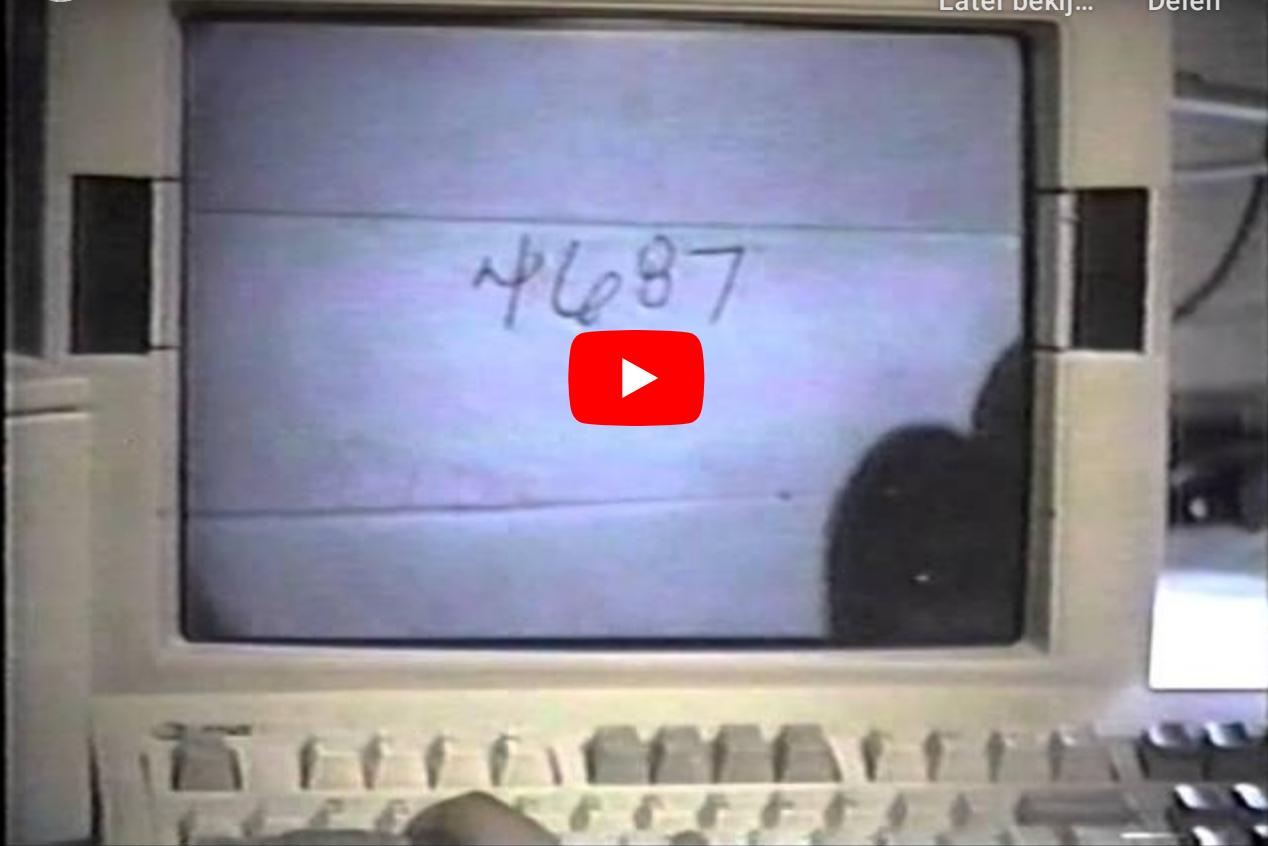
Convolutional Network Demo from 1989



Later bekij...  
...



Delen

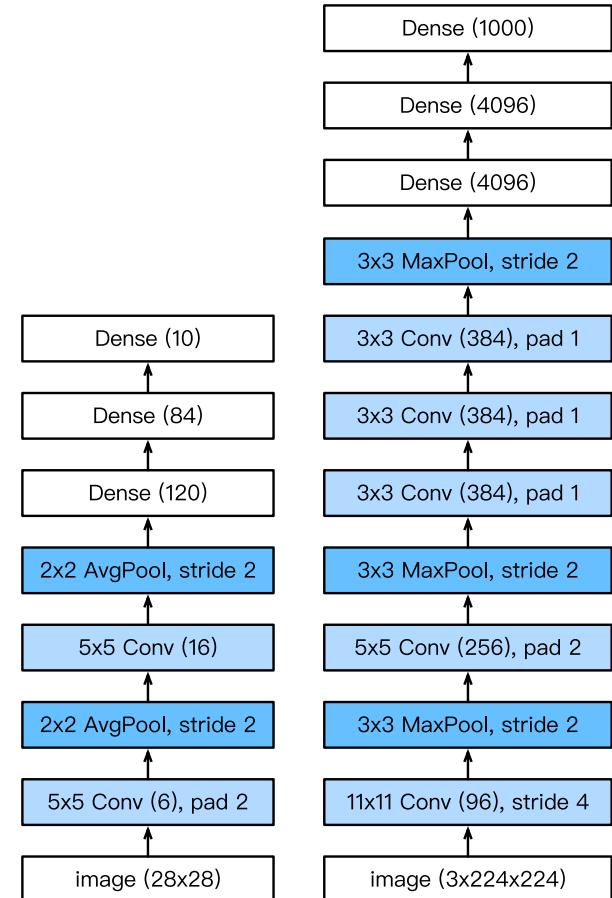


LeNet-1 (LeCun et al, 1993)

## 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.

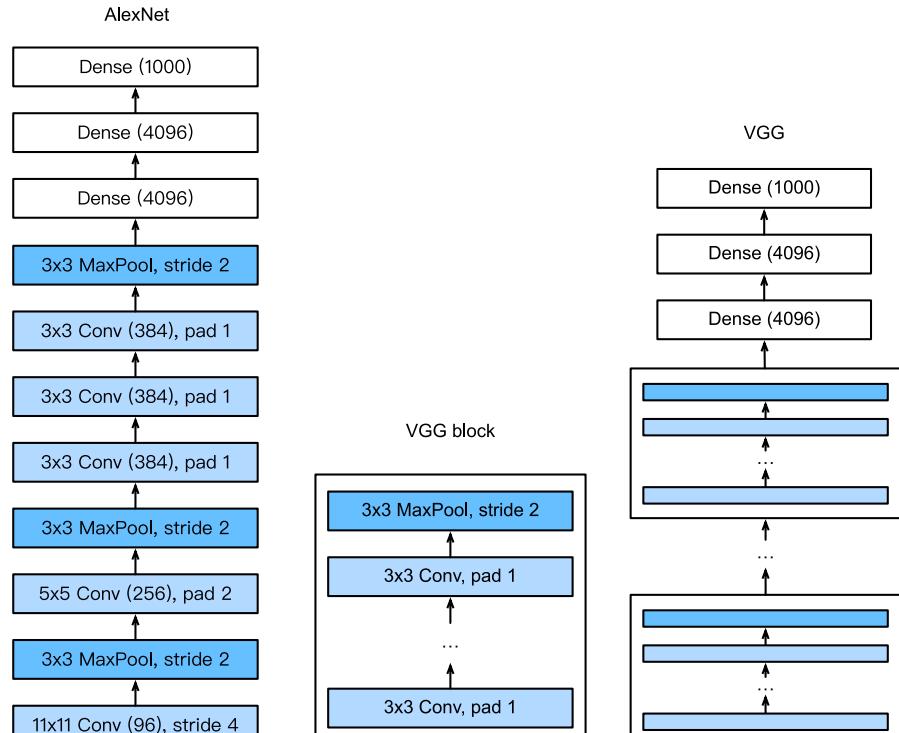


LeNet vs. AlexNet

## 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.

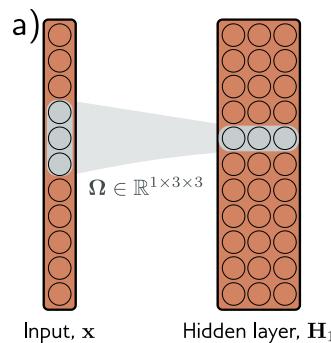


AlexNet vs. VGG

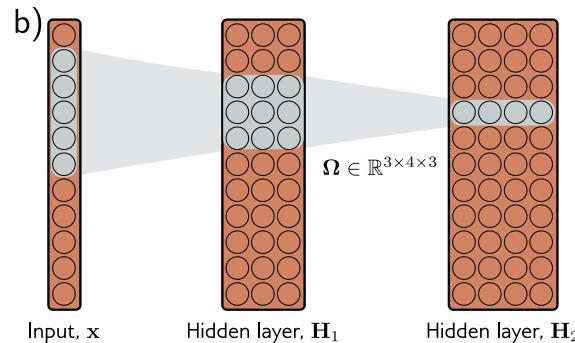
The **effective receptive field** is the part of the visual input that affects a given unit indirectly through previous convolutional layers.

- 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.

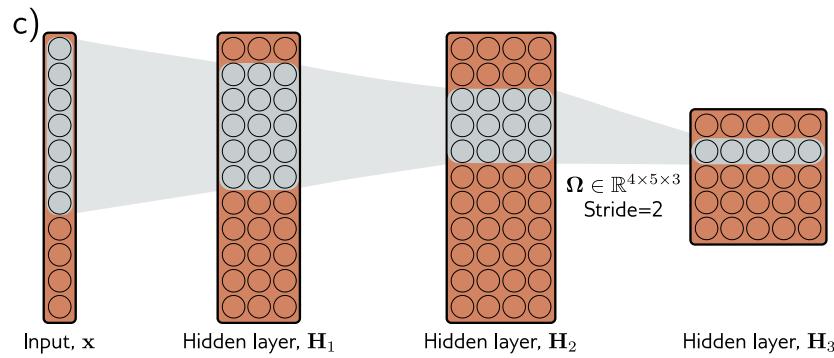
a)



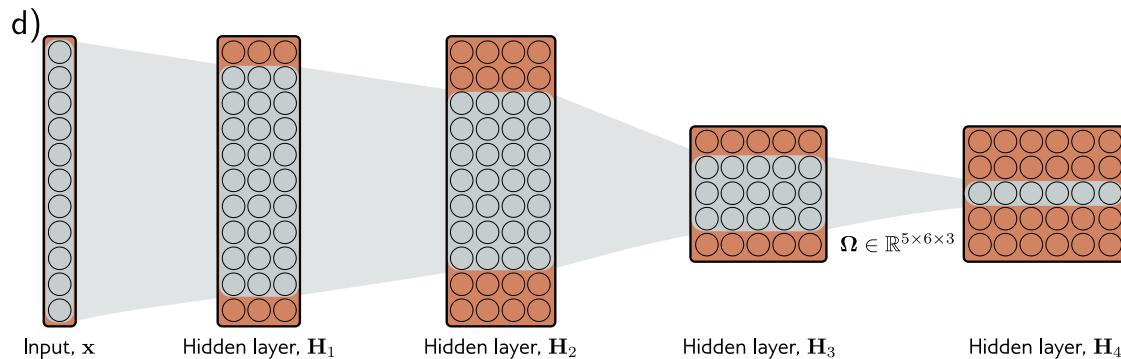
b)



c)

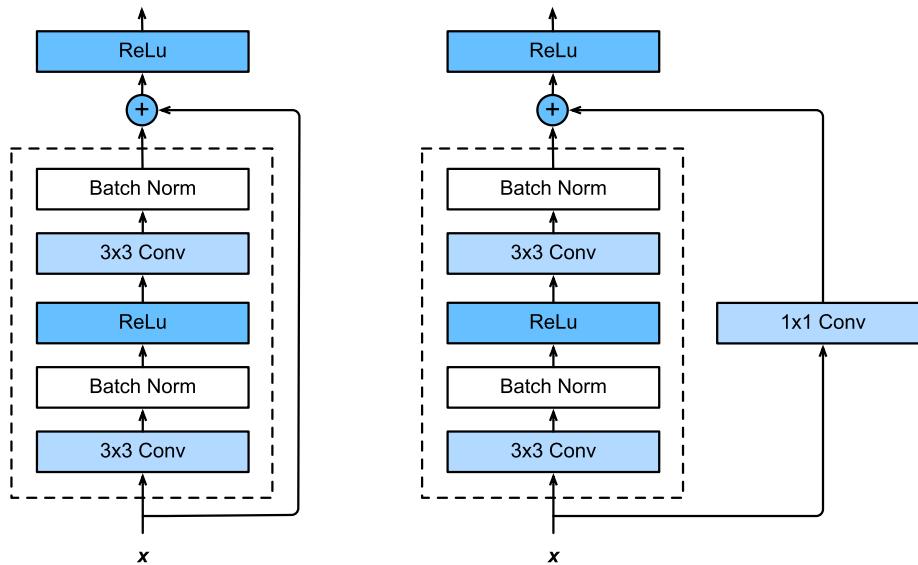


d)

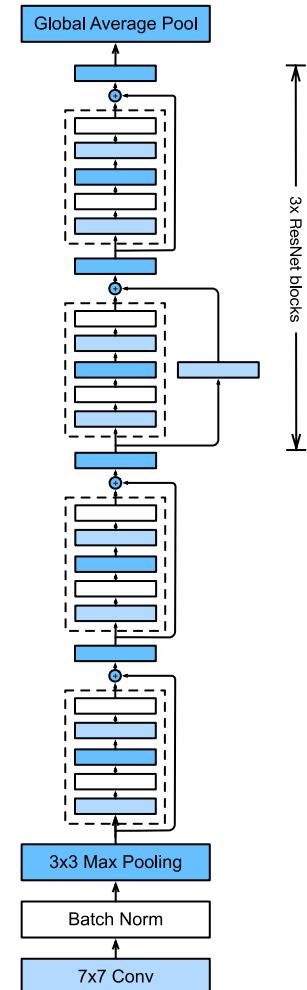


# ResNet (He et al, 2015)

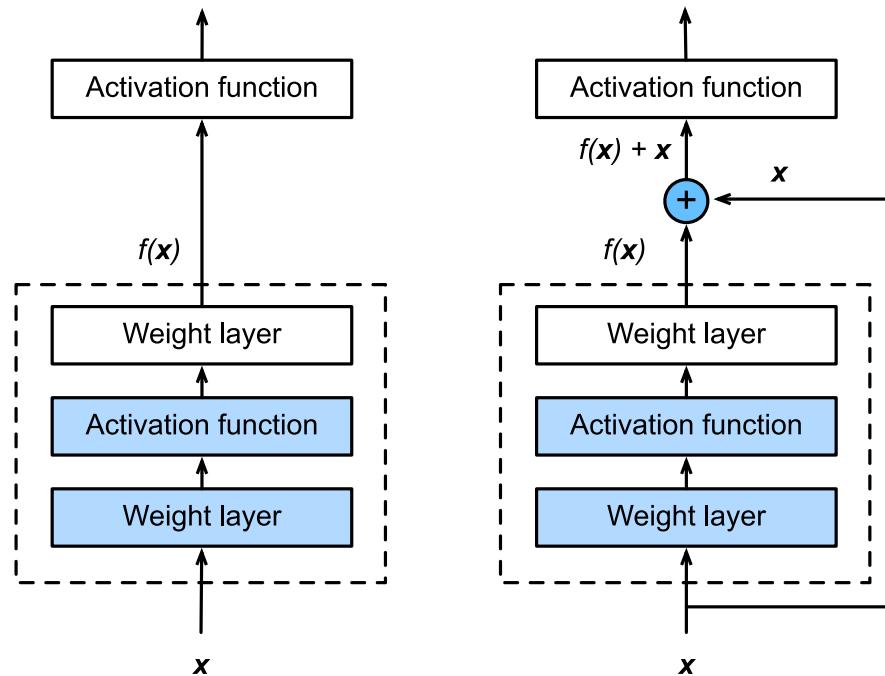
Composition of convolutional and pooling layers organized in a stack of residual blocks. Extensions consider more residual blocks, up to a total of 152 layers (ResNet-152).



Regular ResNet block vs. ResNet block with  $1 \times 1$  convolution.



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.



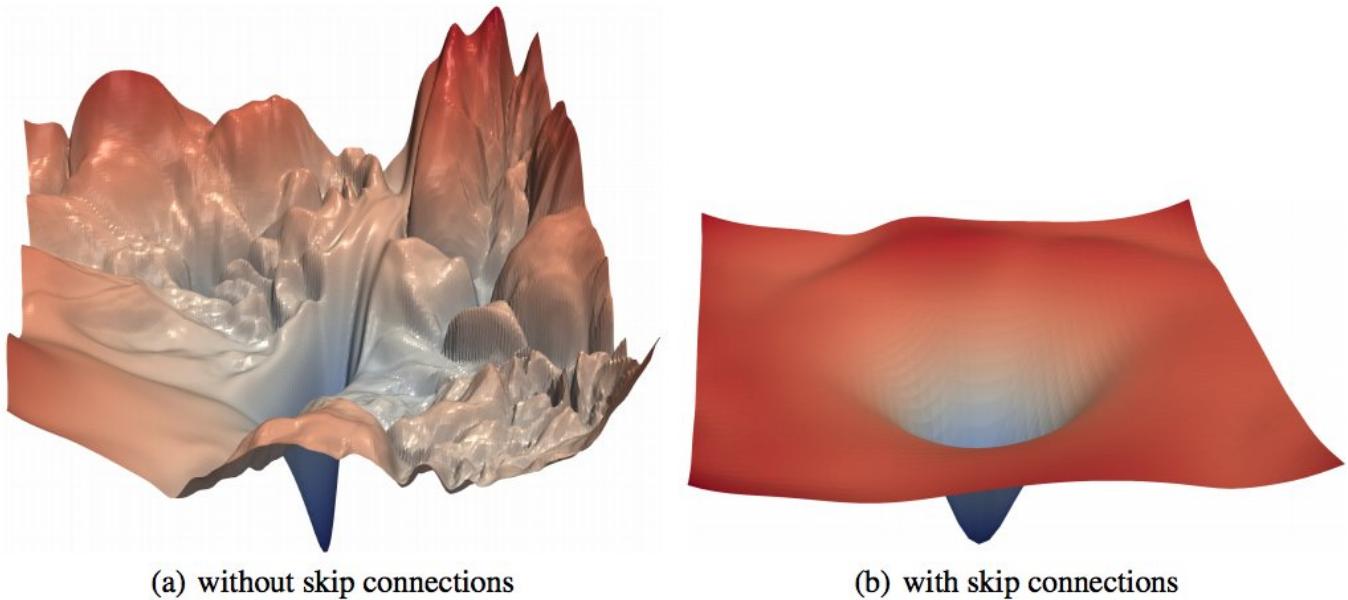
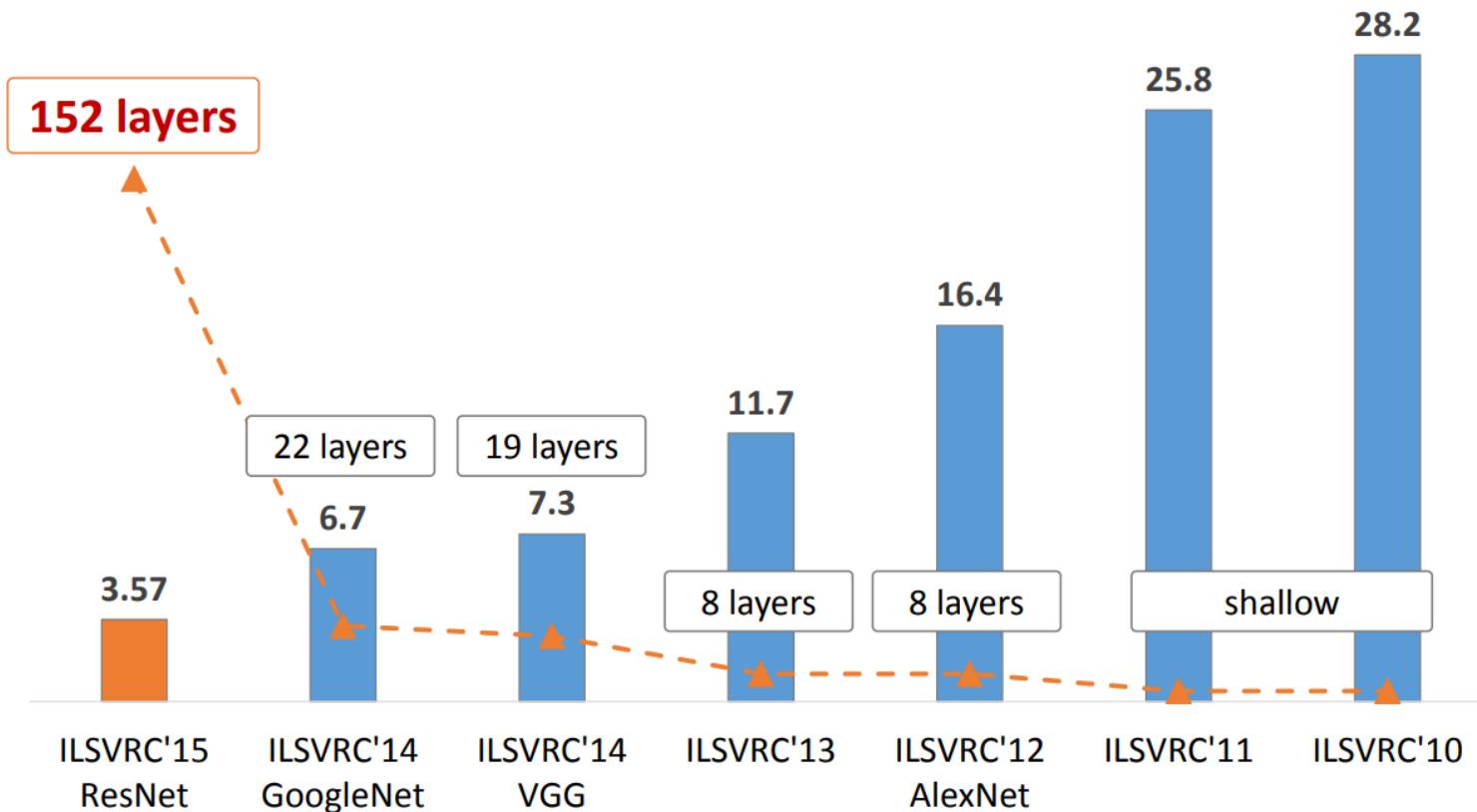
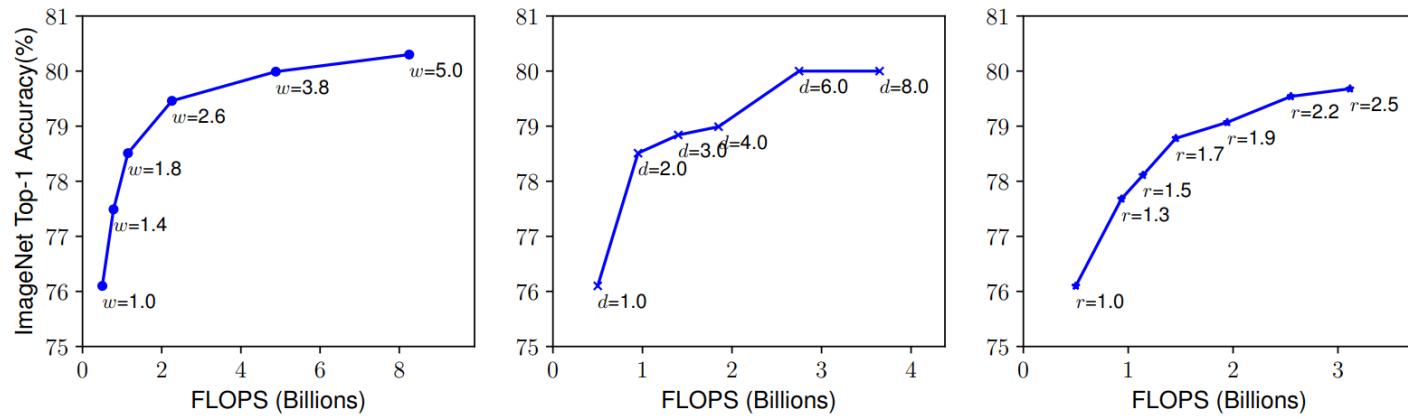


Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

## 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 saturates after reaching 80%, demonstrating the limitation of single dimension scaling. Baseline network is described in Table 1.

# **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:

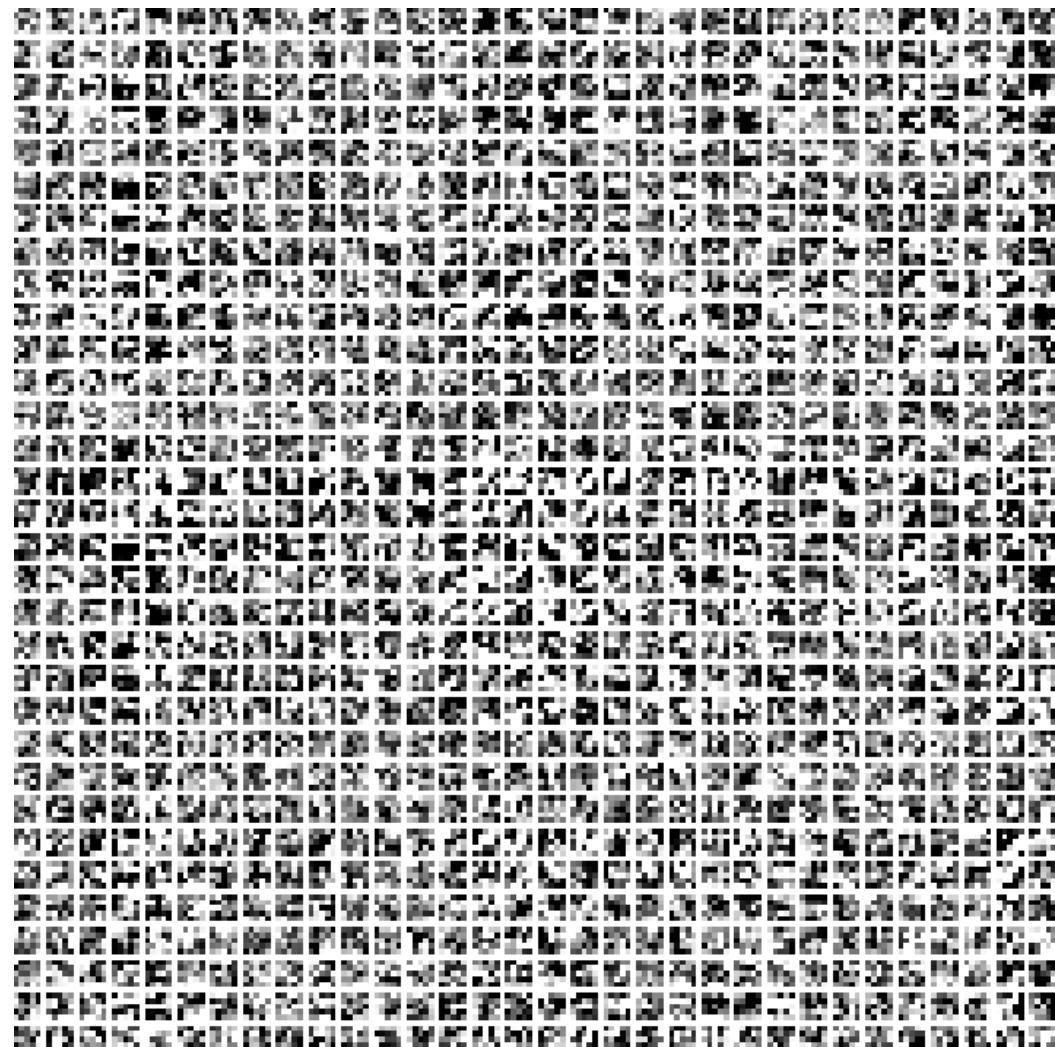
- the network's kernels as images
- internal activations on a single sample as images
- distributions of activations on a population of samples
- derivatives of the response with respect to the input
- maximum-response synthetic samples

# Looking at filters

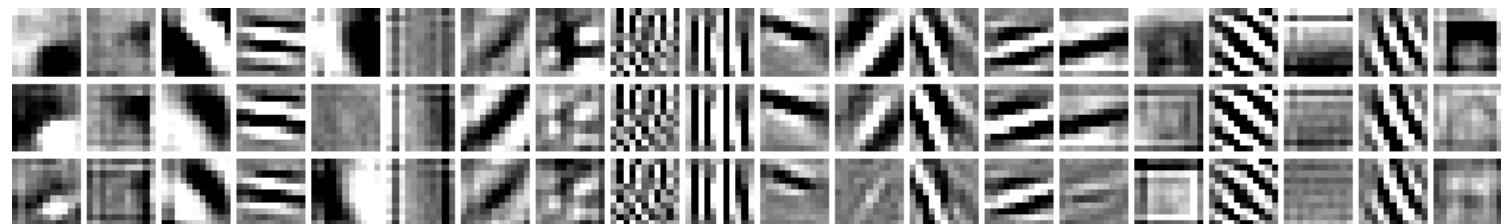
LeNet's first convolutional layer, all filters.



## LeNet's second convolutional layer, first 32 filters.



AlexNet's first convolutional layer, first 20 filters.

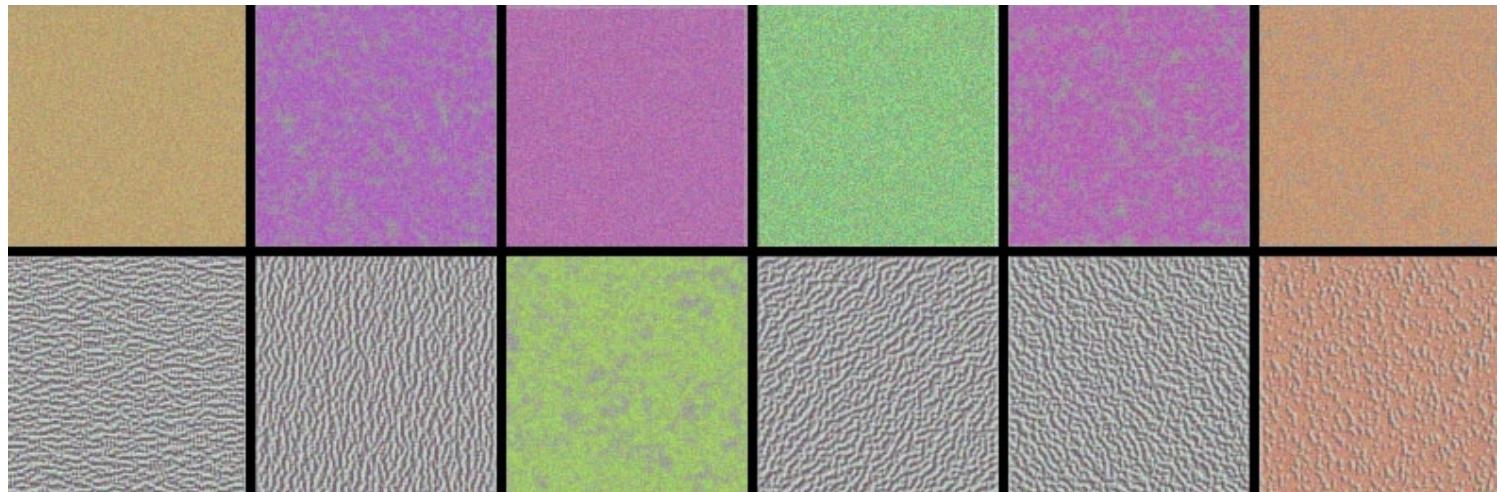


# Maximum response samples

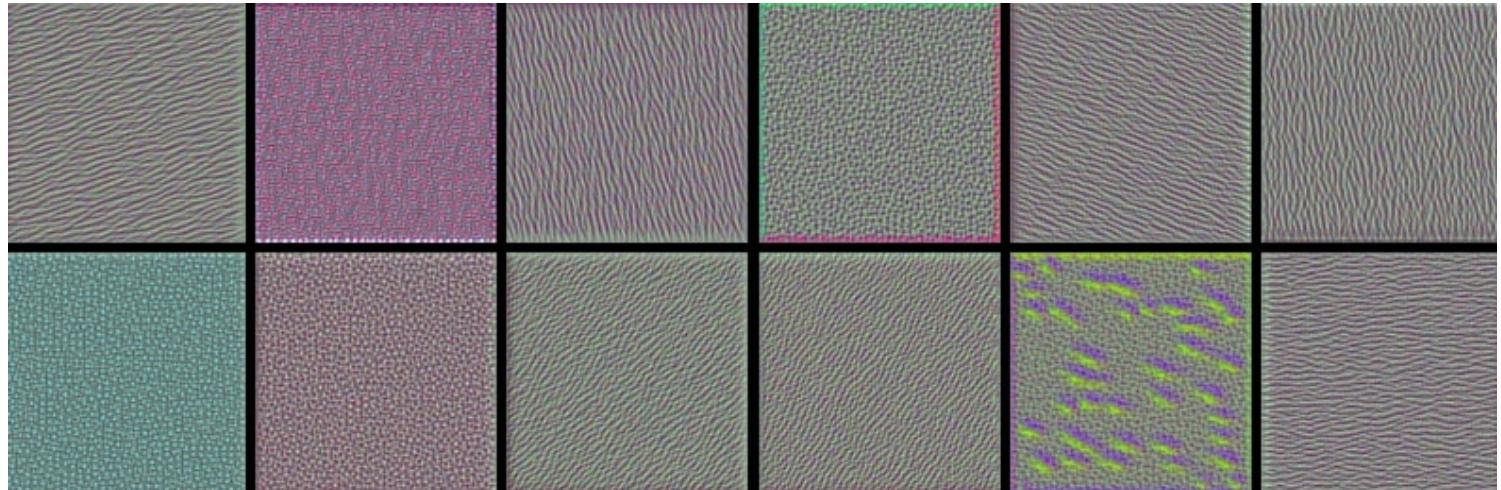
Convolutional networks can be inspected by looking for synthetic input images  $\mathbf{x}$  that maximize the activation  $\mathbf{h}_{\ell,d}(\mathbf{x})$  of a chosen convolutional kernel  $\mathbf{u}$  at layer  $\ell$  and index  $d$  in the layer filter bank.

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

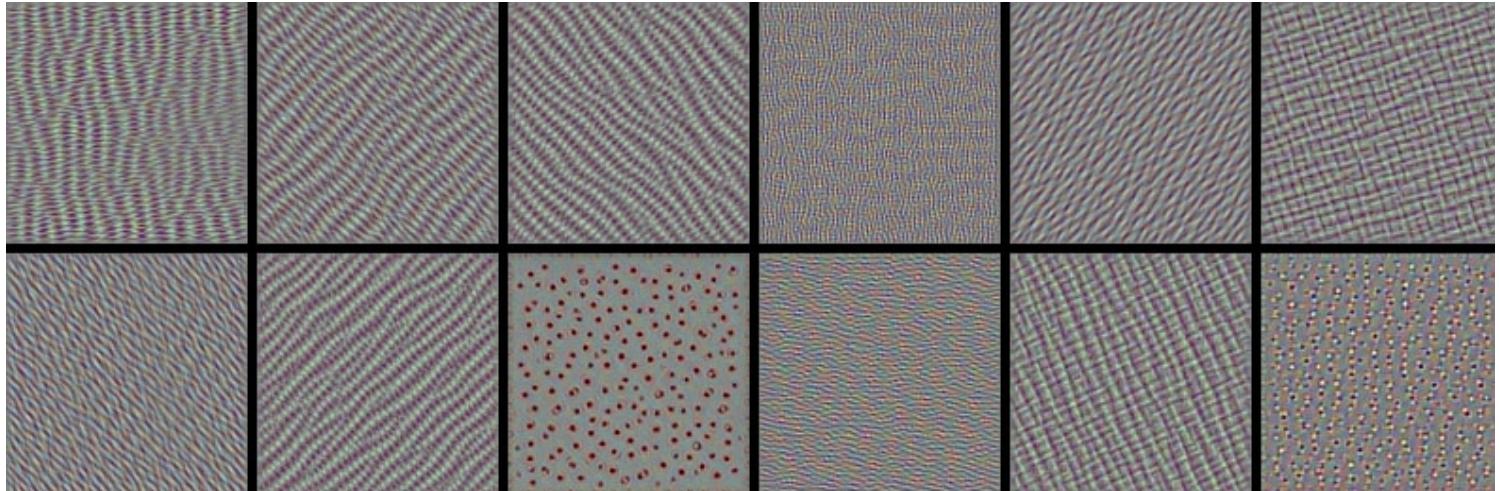
$$\begin{aligned}\mathcal{L}_{\ell,d}(\mathbf{x}) &= \|\mathbf{h}_{\ell,d}(\mathbf{x})\|_2 \\ \mathbf{x}_0 &\sim U[0, 1]^{C \times H \times W} \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \gamma \nabla_{\mathbf{x}} \mathcal{L}_{\ell,d}(\mathbf{x}_t)\end{aligned}$$



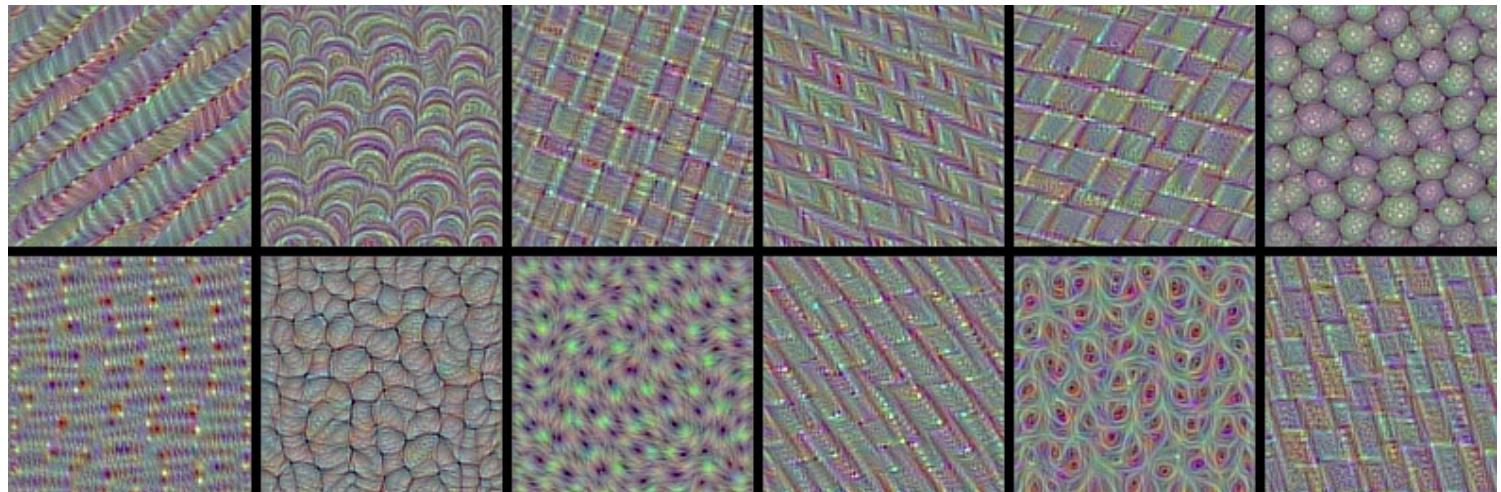
VGG-16, convolutional layer 1-1, a few of the 64 filters



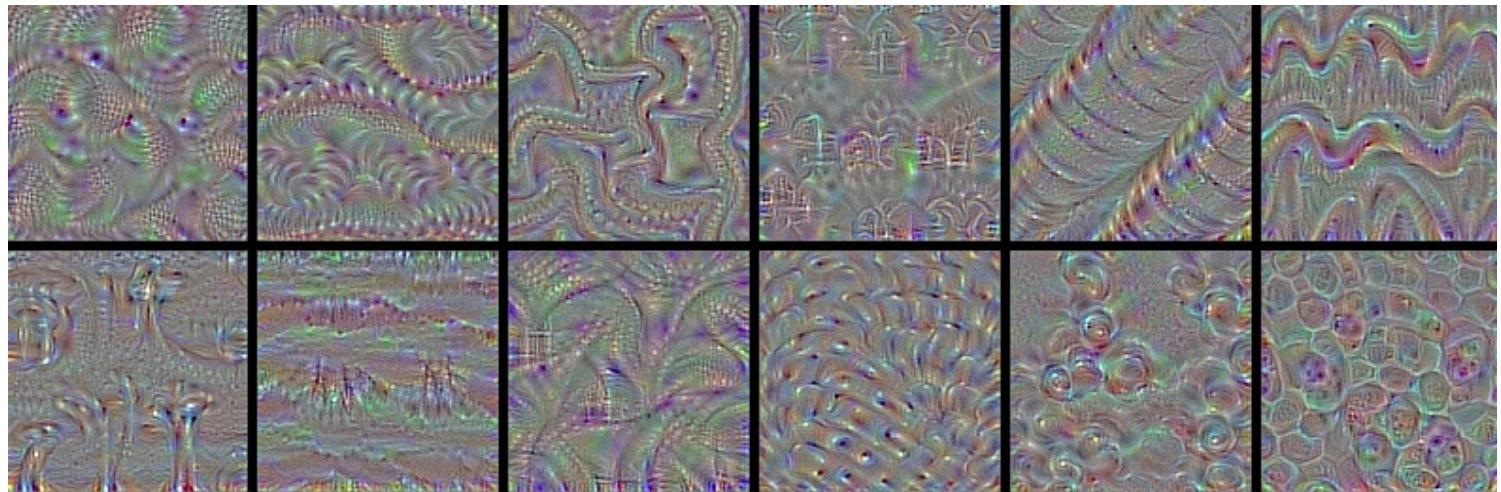
VGG-16, convolutional layer 2-1, a few of the 128 filters



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



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

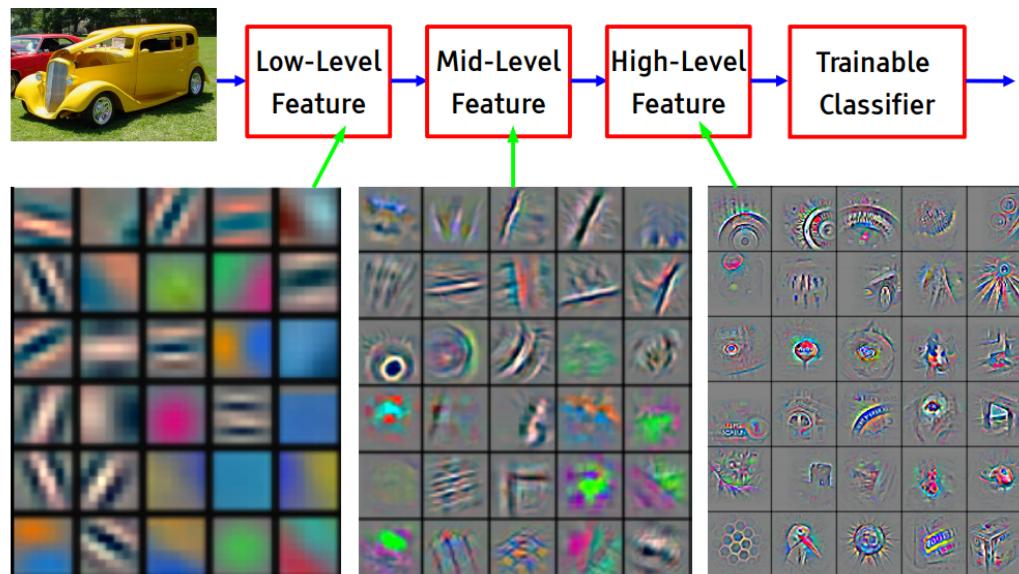


VGG-16, convolutional layer 5-1, a few of the 512 filters

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**.

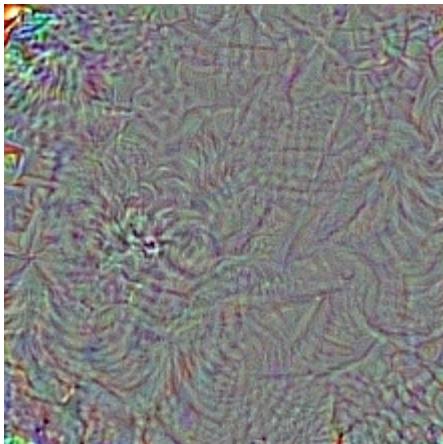


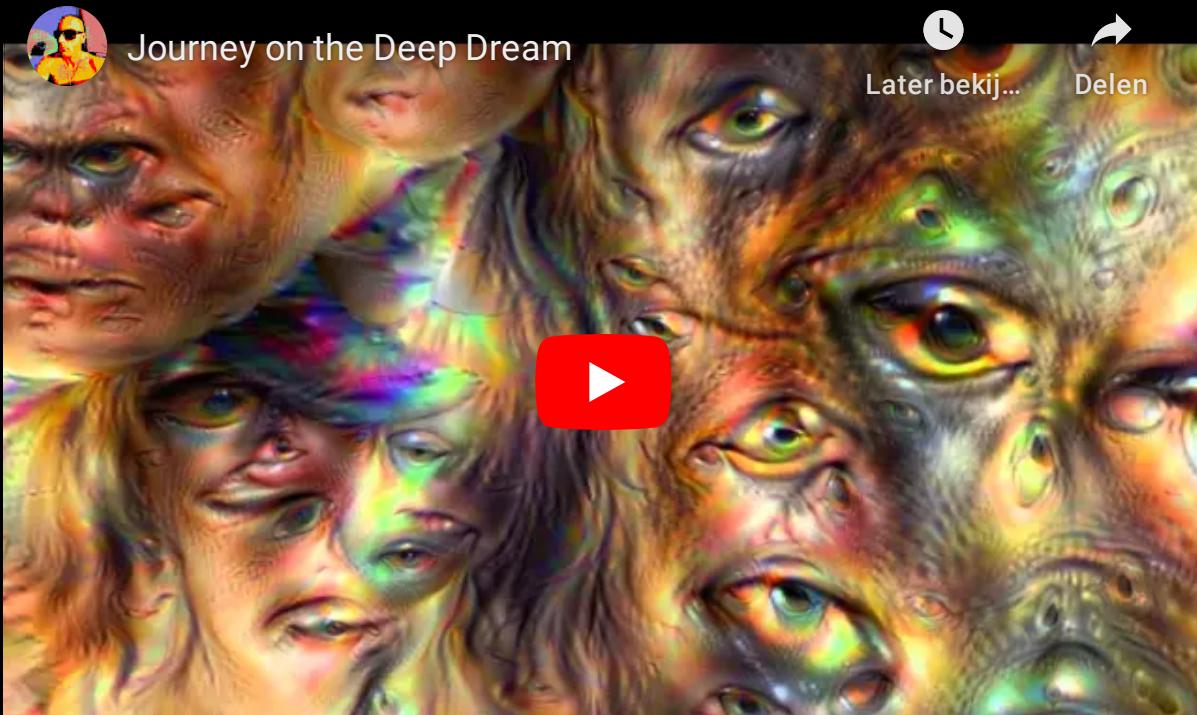
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

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?

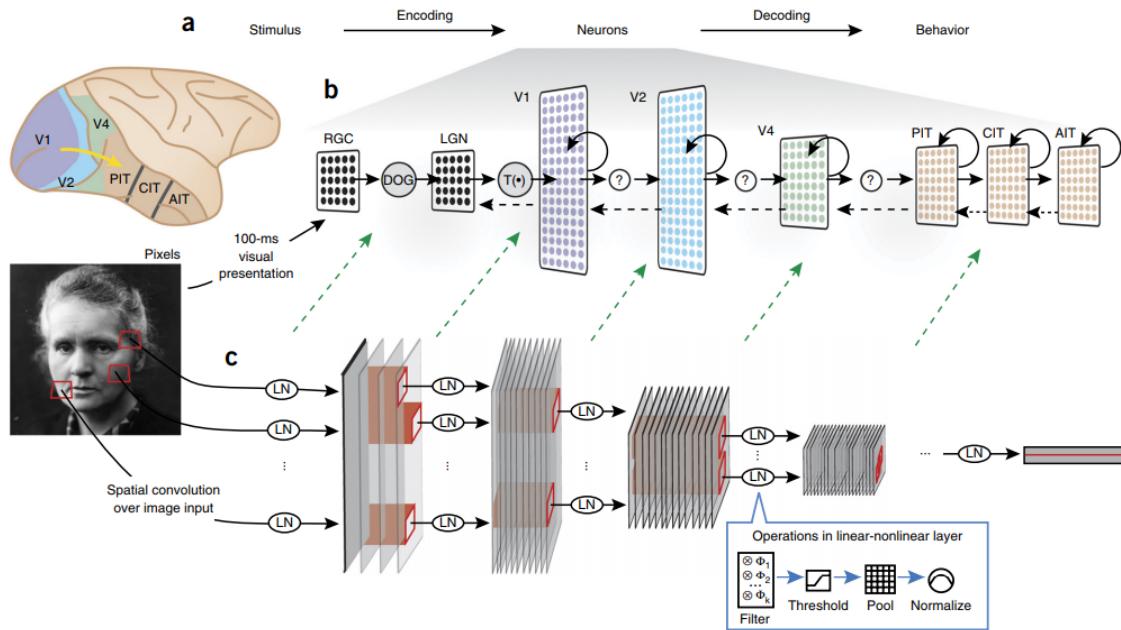
The left image is predicted **with 99.9% confidence** as a magpie!





**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."*

