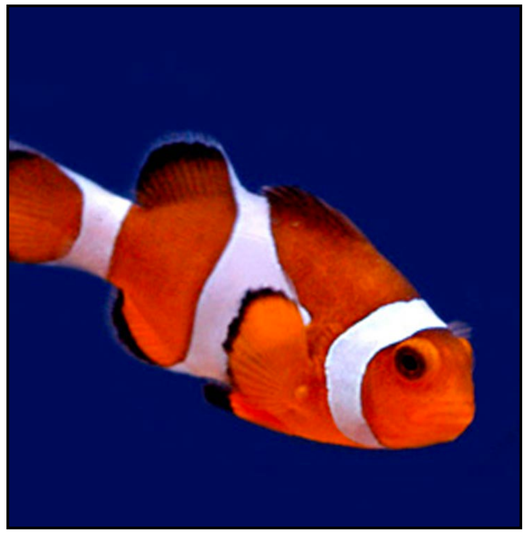
Lecture 9: Convolutional networks

Image classification



Classifier

“Fish”

image **x**

label y

Source: Isola, Torralba, Freeman

Image classification

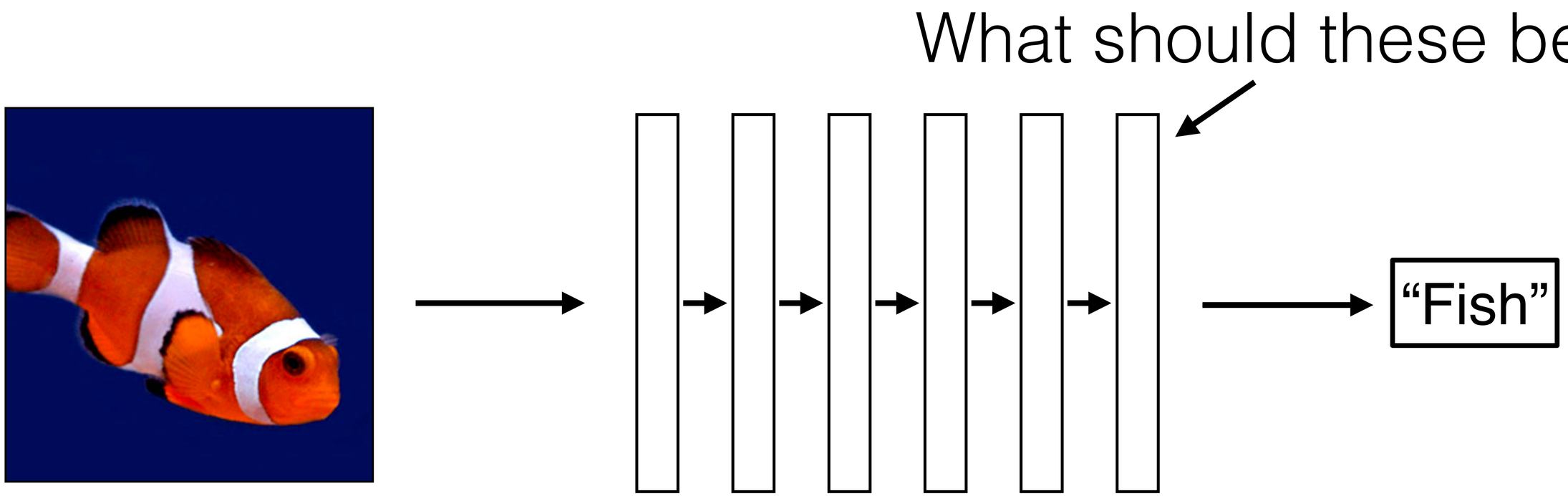


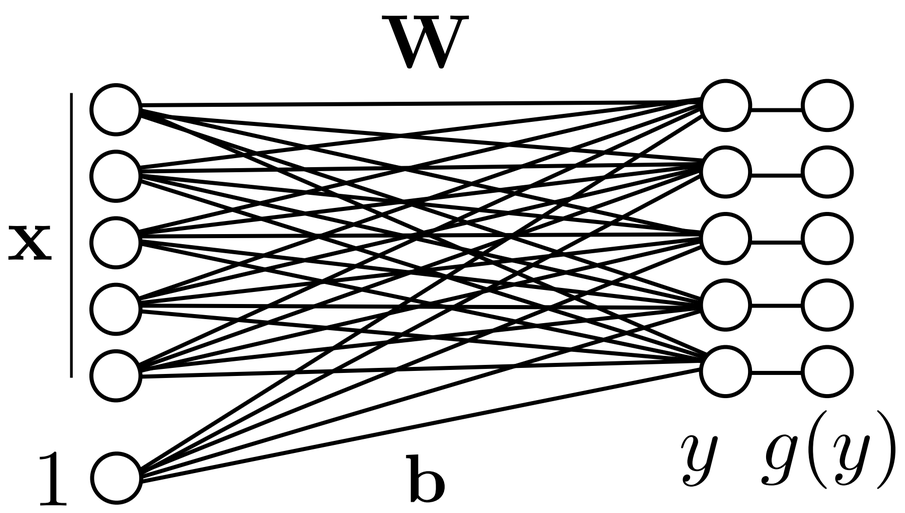
image **x**

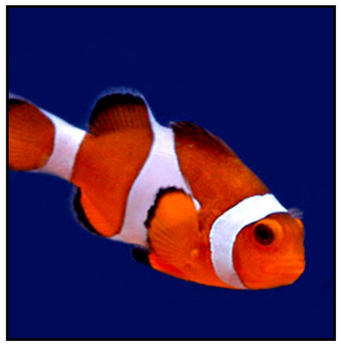
label y

Source: Isola, Torralba, Freeman

Fully-connected network

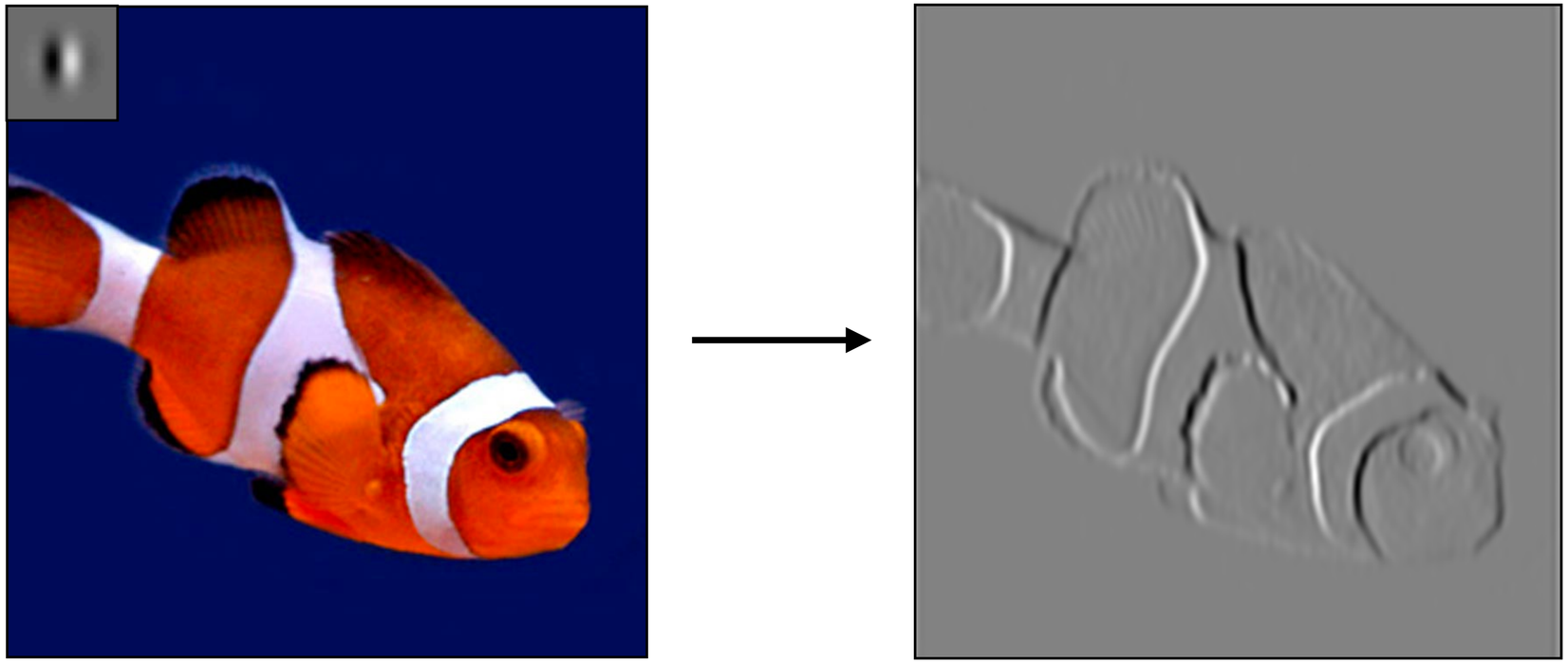
Fully-connected (fc) layer



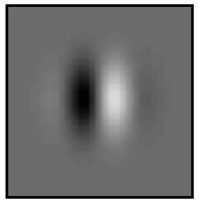
$\mathbf{x} =$ 

- $\mathbf{x}$ is really big! E.g. 256 x 256
- If a feature is useful in one location, it should be useful in others, too

Can we use convolution in a neural network?

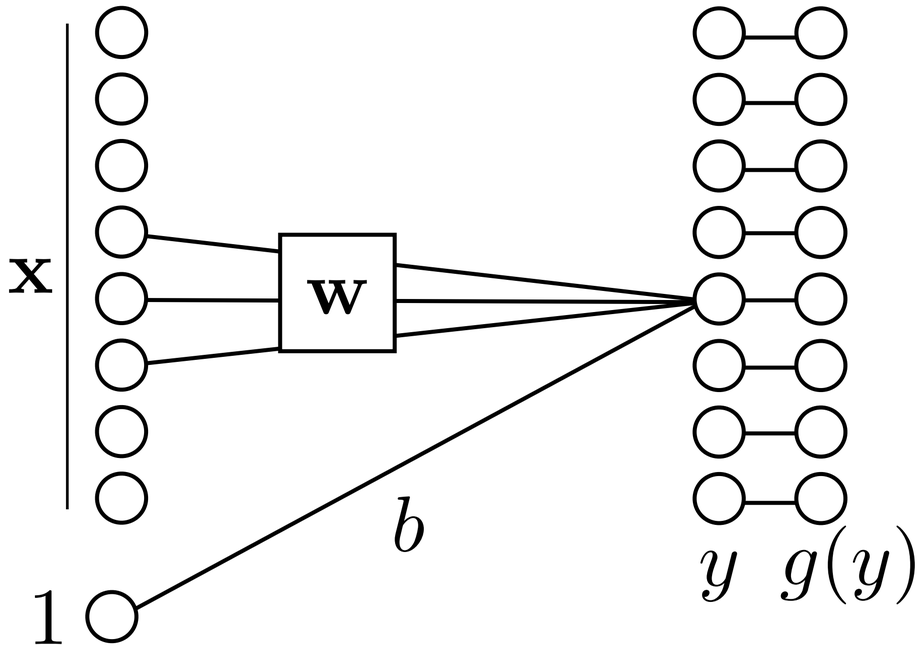


filter



Source: Isola, Torralba, Freeman

Locally connected network

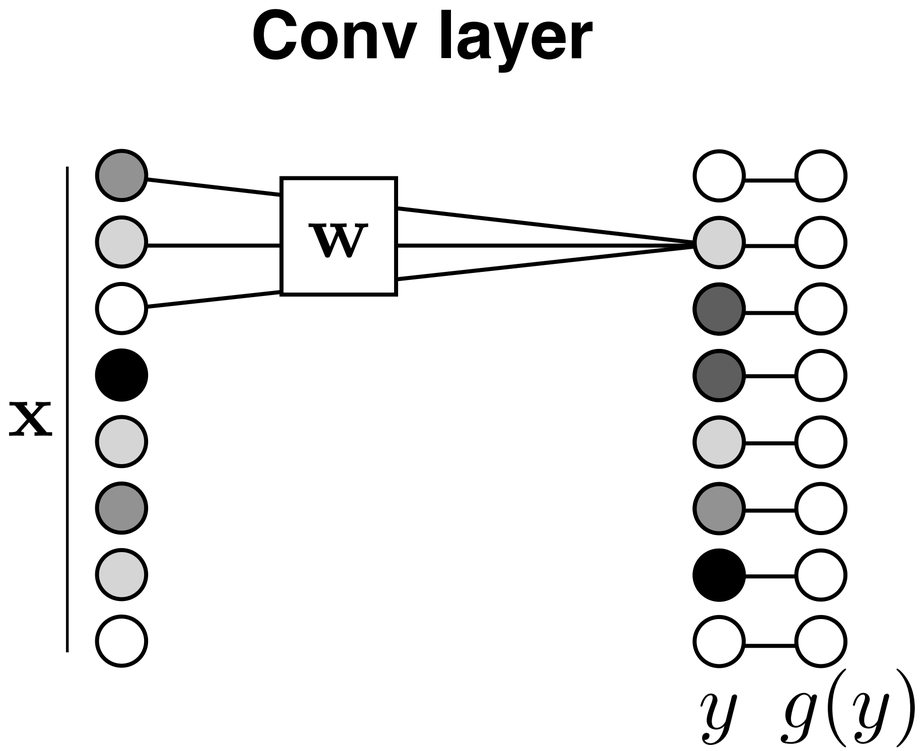


Often, we assume output is a **local** function of input.

If we use the same weights (**weight sharing**) to compute each local function, we get a convolutional neural network.

Source: Isola, Torralba, Freeman

Convolutional neural network



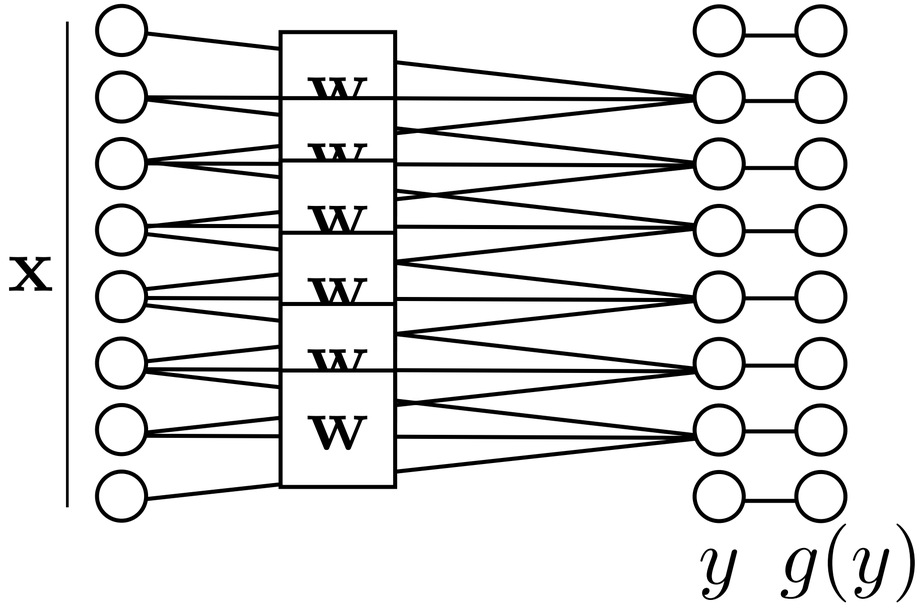
Each output unit is computed from an image patch.

If we use the same weights (**weight sharing**) to compute each local function, we get a convolutional neural network.

Source: Isola, Torralba, Freeman

Weight sharing

Conv layer



Each output unit is computed from an image patch.

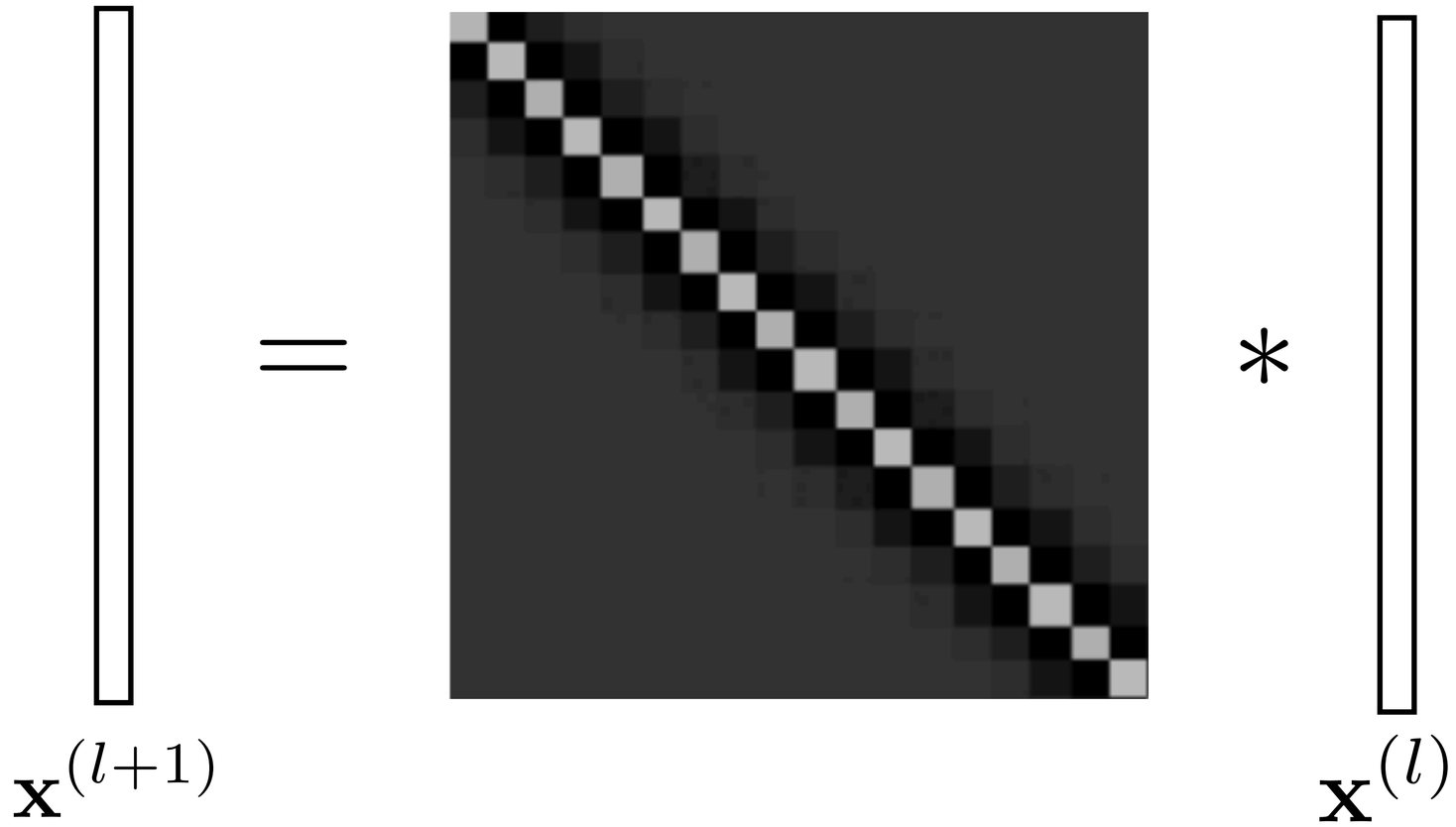
If we use the same weights (**weight sharing**) to compute each local function, we get a convolutional neural network.

Source: Isola, Torralba, Freeman

Recall: convolution is a linear function

Toeplitz matrix

$$\begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix}$$

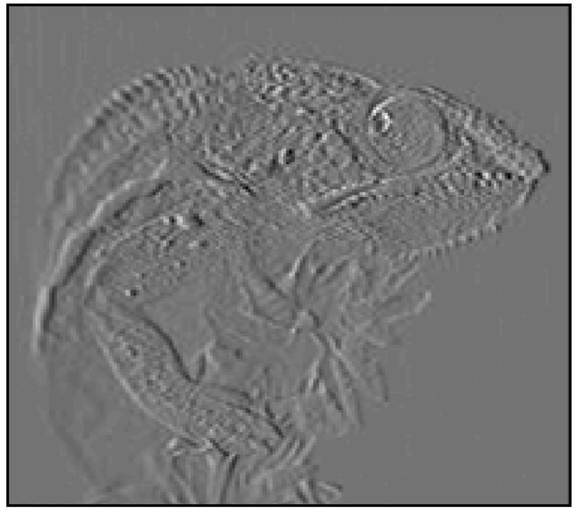


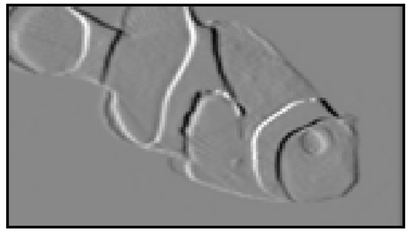
e.g., pixel image

- Constrained linear layer
- Fewer parameters: easier to learn, less overfitting

Source: Isola, Torralba, Freeman





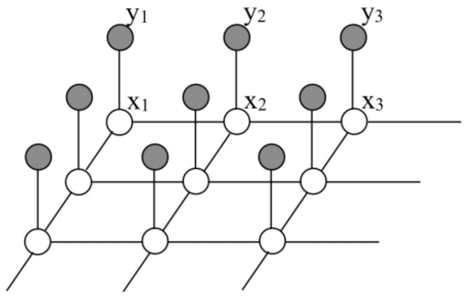
Convolution layers can be applied to arbitrarily-sized inputs

Source: Isola, Torralba, Freeman

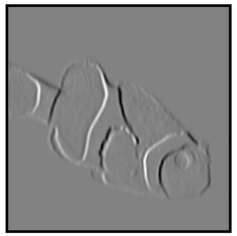
Different interpretations of convolutional layers

1. Equivariant with translation: $f(\texttt{translate}(x)) = \texttt{translate}(f(x))$

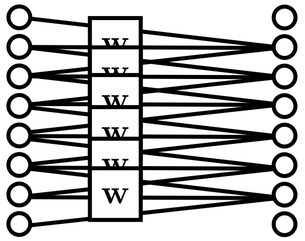
2. Patch processing (Markov assumption)



3. Image filter



4. Parameter sharing



5. A way to process variable-sized tensors

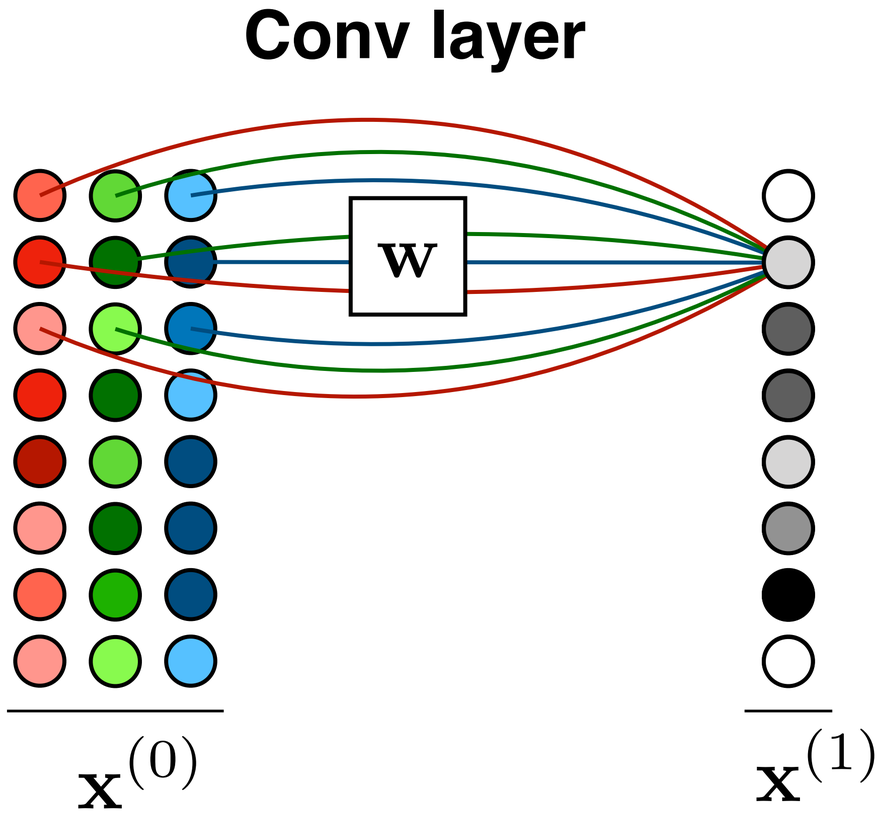
Source: Isola, Torralba, Freeman

What if we have color?

(And multiple input channels in general)

Source: Isola, Torralba, Freeman

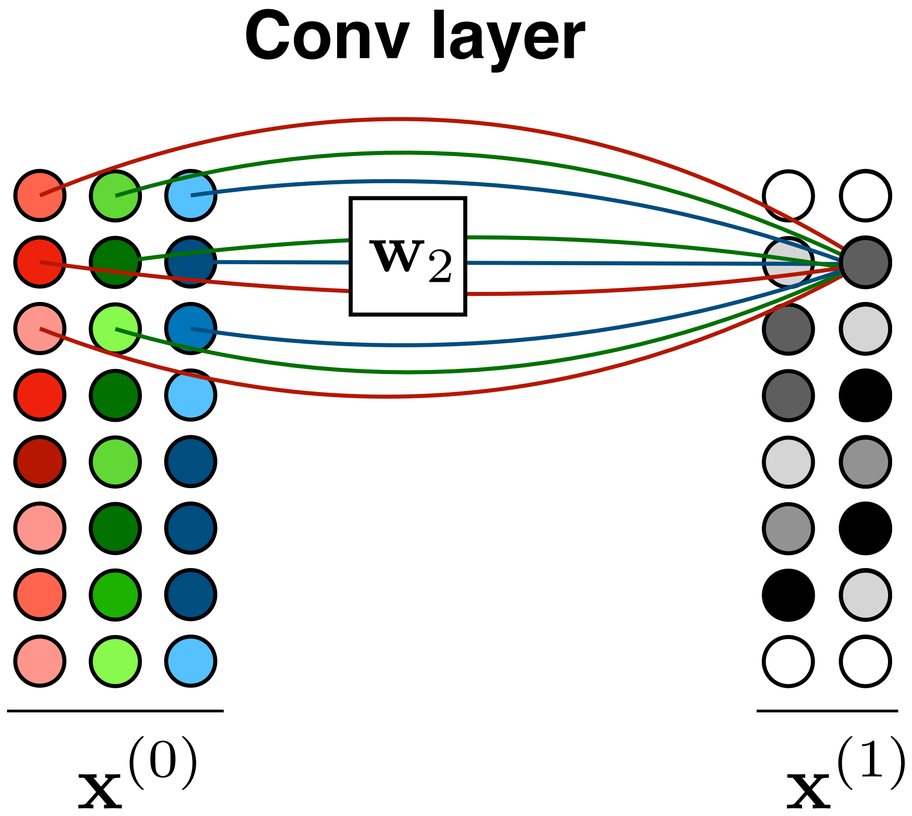
Multiple channels



$$\mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{N \times 1}$$

Source: Isola, Torralba, Freeman

Multiple channels

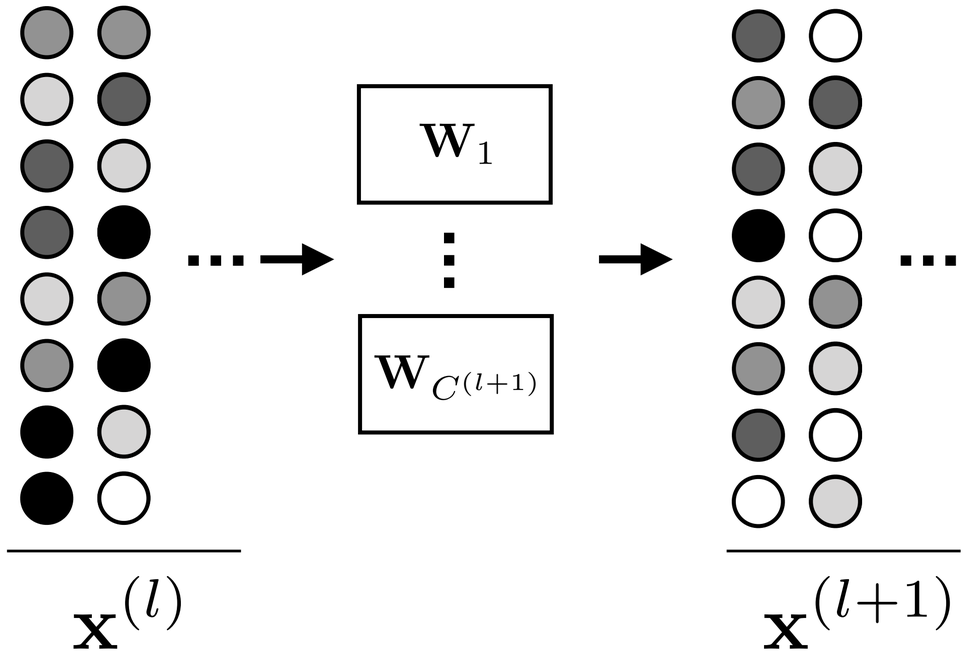


$$\mathbb{R}^{N \times C^{(0)}} \rightarrow \mathbb{R}^{N \times C^{(1)}}$$

Source: Isola, Torralba, Freeman

Multiple channels

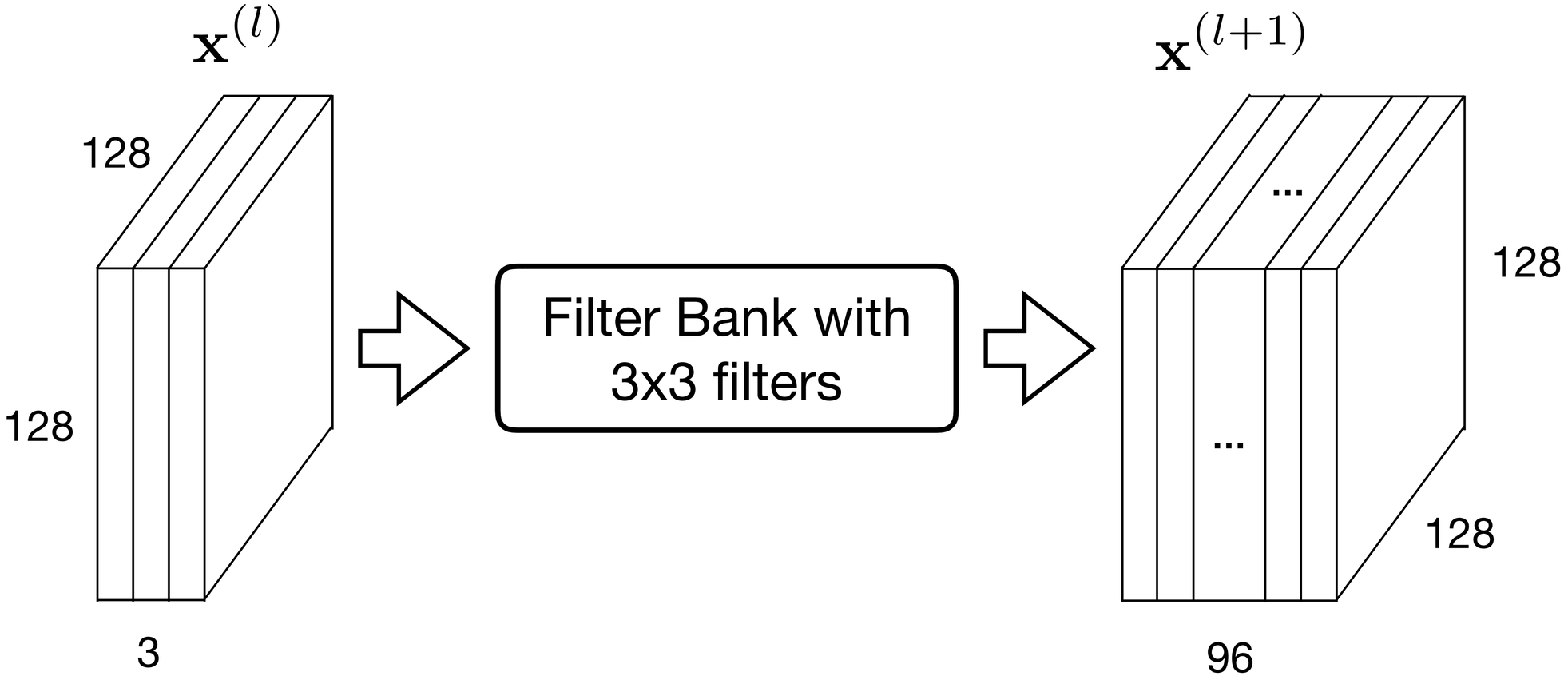
Conv layer



$$\mathbb{R}^{N \times C^{(l)}} \rightarrow \mathbb{R}^{N \times C^{(l+1)}}$$

Source: Isola, Torralba, Freeman

Multiple channels: Example

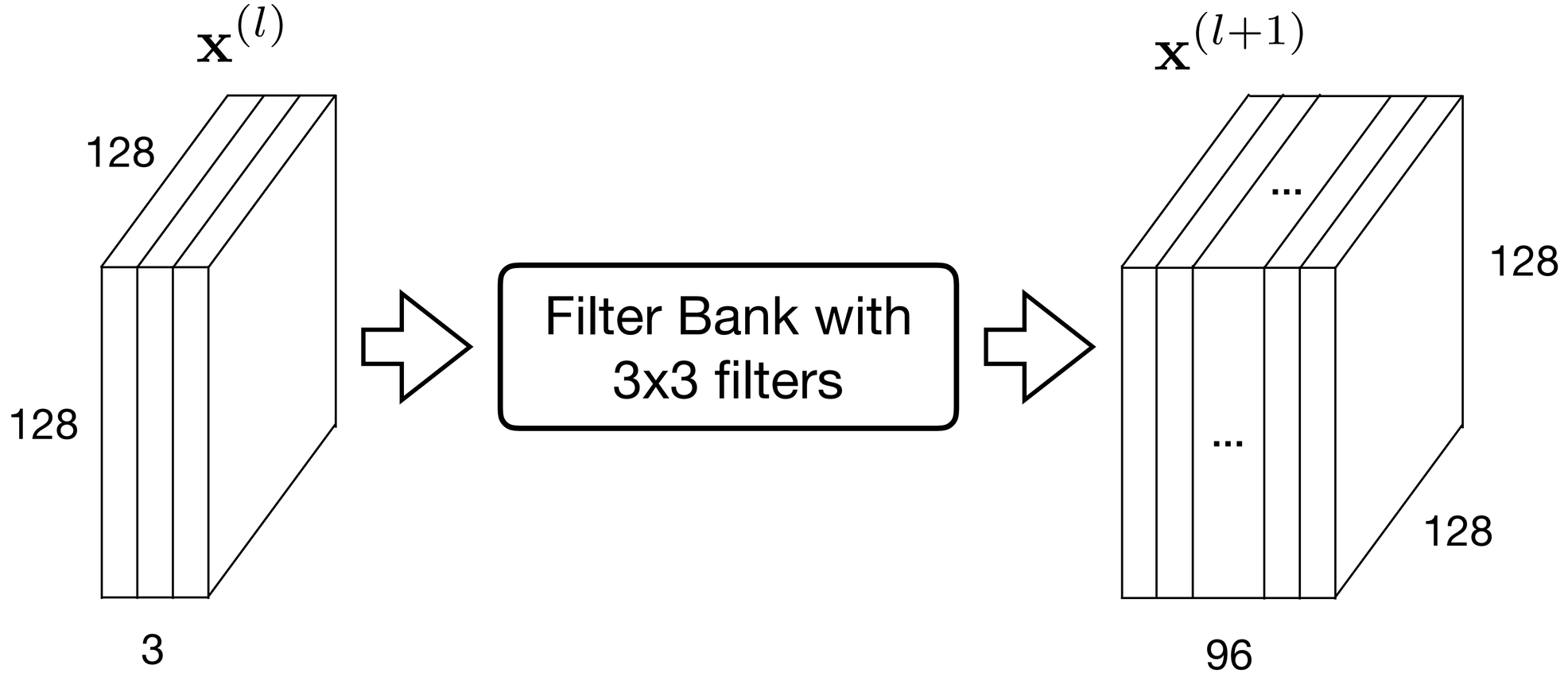


How many parameters does *each filter* have?

(a) 9 (b) 27 (c) 96 (d) 2592

Source: Isola, Torralba, Freeman

Multiple channels: Example



How many parameters *total* does this layer have?

(a) 9 (b) 27 (c) 96 (d) 2592

Filter sizes

When mapping from

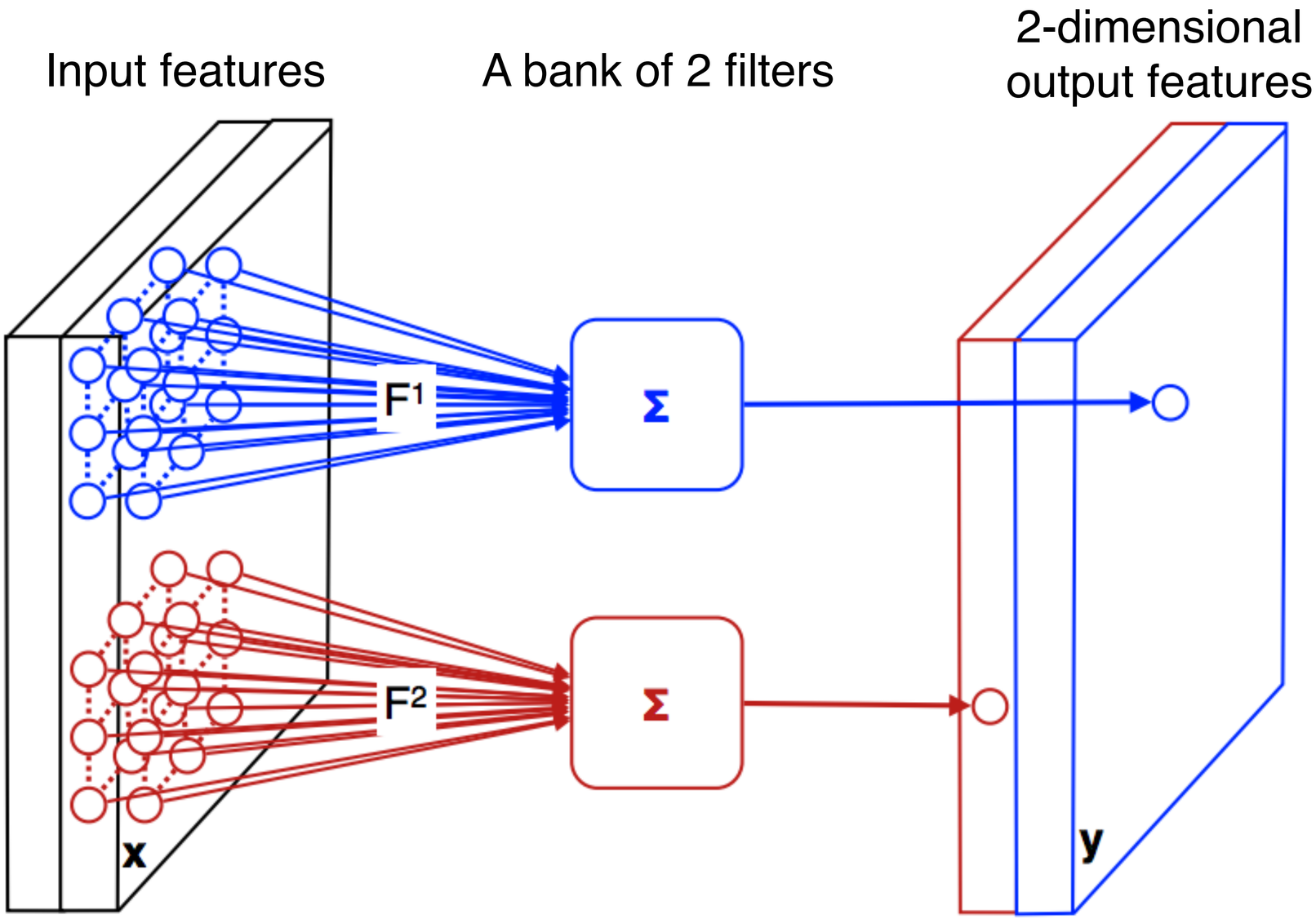
$$\mathbf{x}^{(l)} \in \mathbb{R}^{H \times W \times C^{(l)}} \rightarrow \mathbf{x}^{(l+1)} \in \mathbb{R}^{H \times W \times C^{(l+1)}}$$

using an filter of spatial extent $M \times N$

Number of parameters per filter: $M \times N \times C^{(l)}$

Number of filters: $C^{(l+1)}$

Source: Isola, Torralba, Freeman



$$\mathbb{R}^{H \times W \times C^{(l)}} \rightarrow \mathbb{R}^{H \times W \times C^{(l+1)}}$$

[Figure from Andrea Vedaldi]
Source: Isola, Torralba, Freeman

Image classification

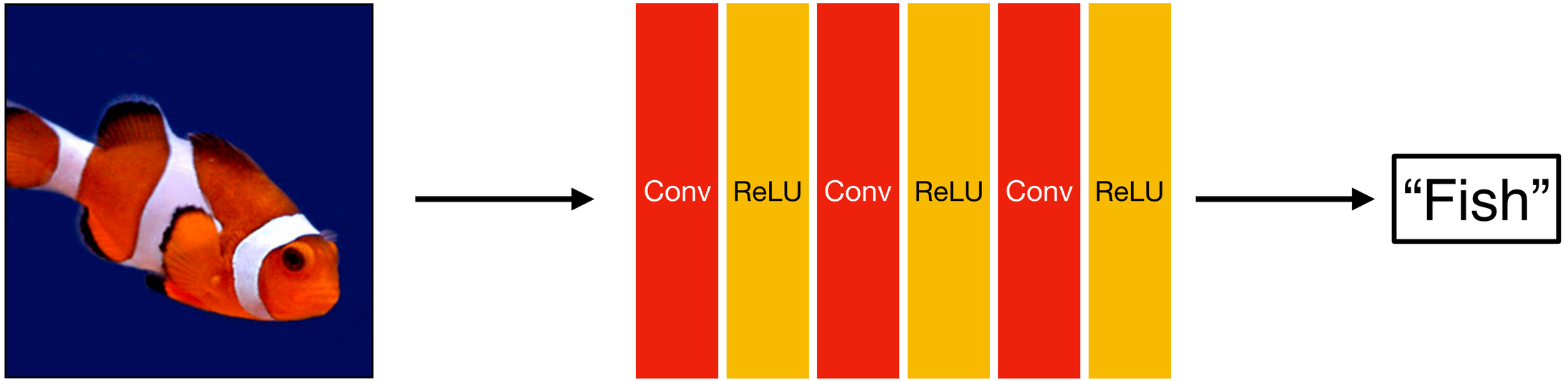
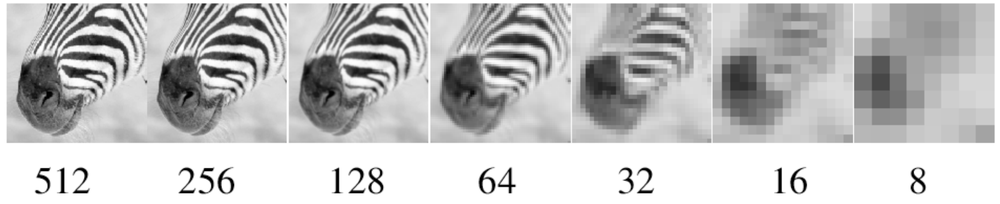


image **x**

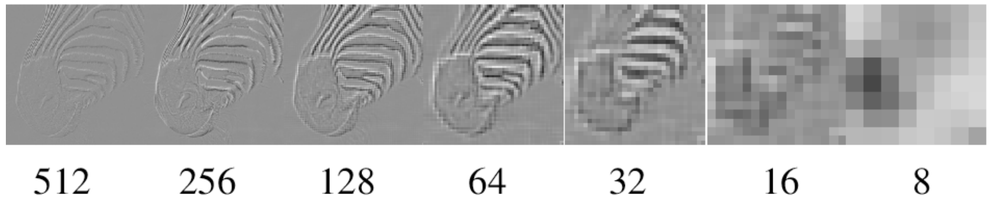
label y

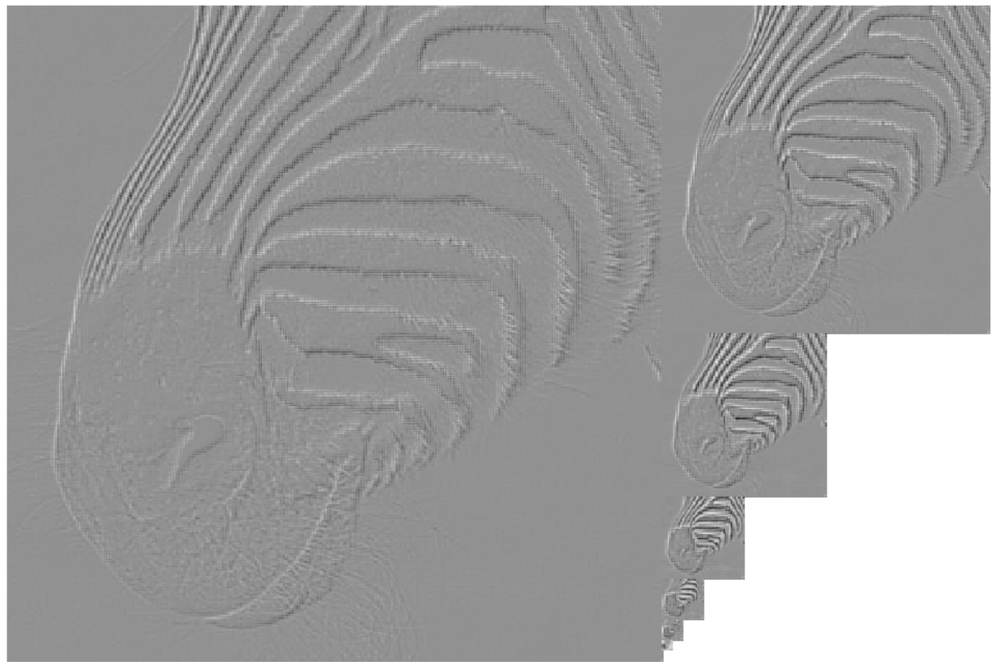
Source: Isola, Torralba, Freeman

Recall: pyramid representations



Gaussian Pyramid



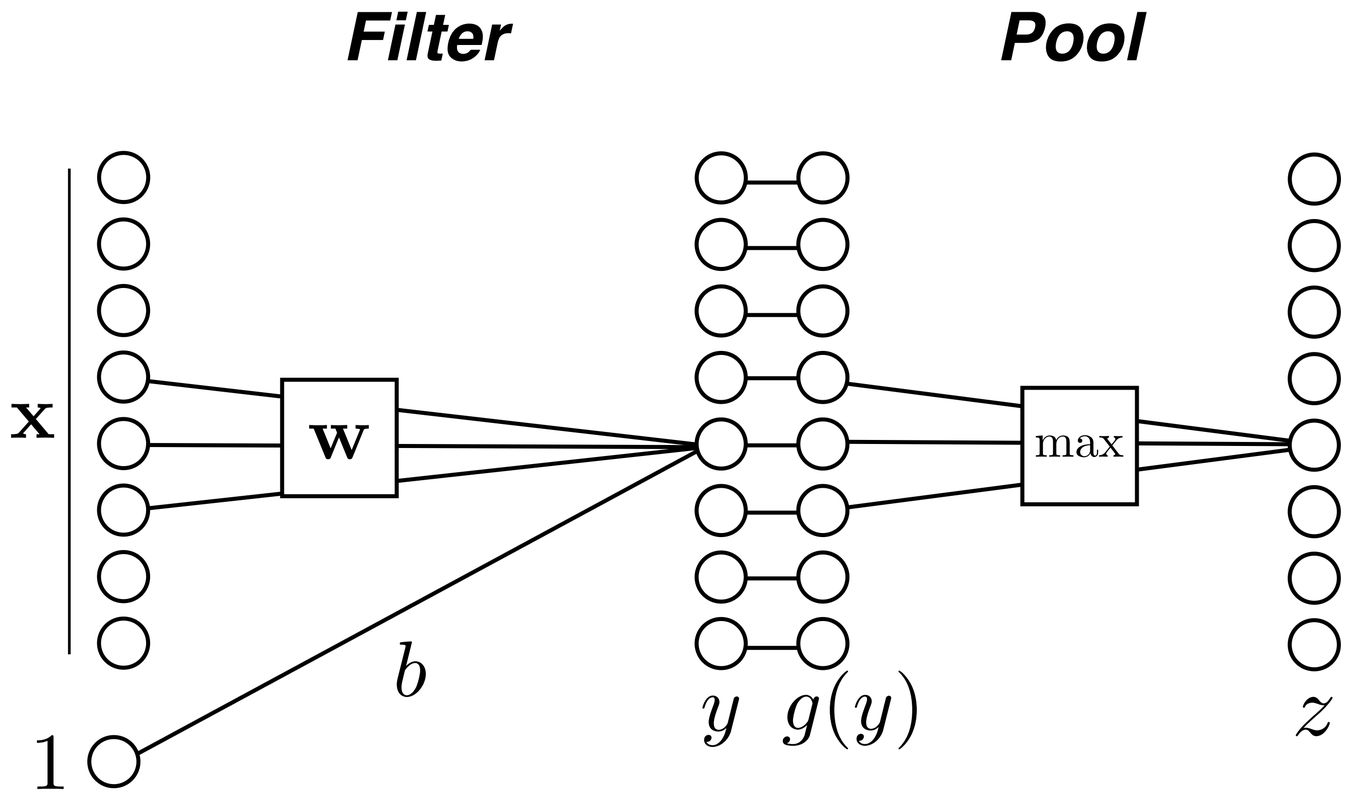


Laplacian Pyramid

How can we use multi-scale modeling in CNNs?

Source: Isola, Torralba, Freeman

Pooling

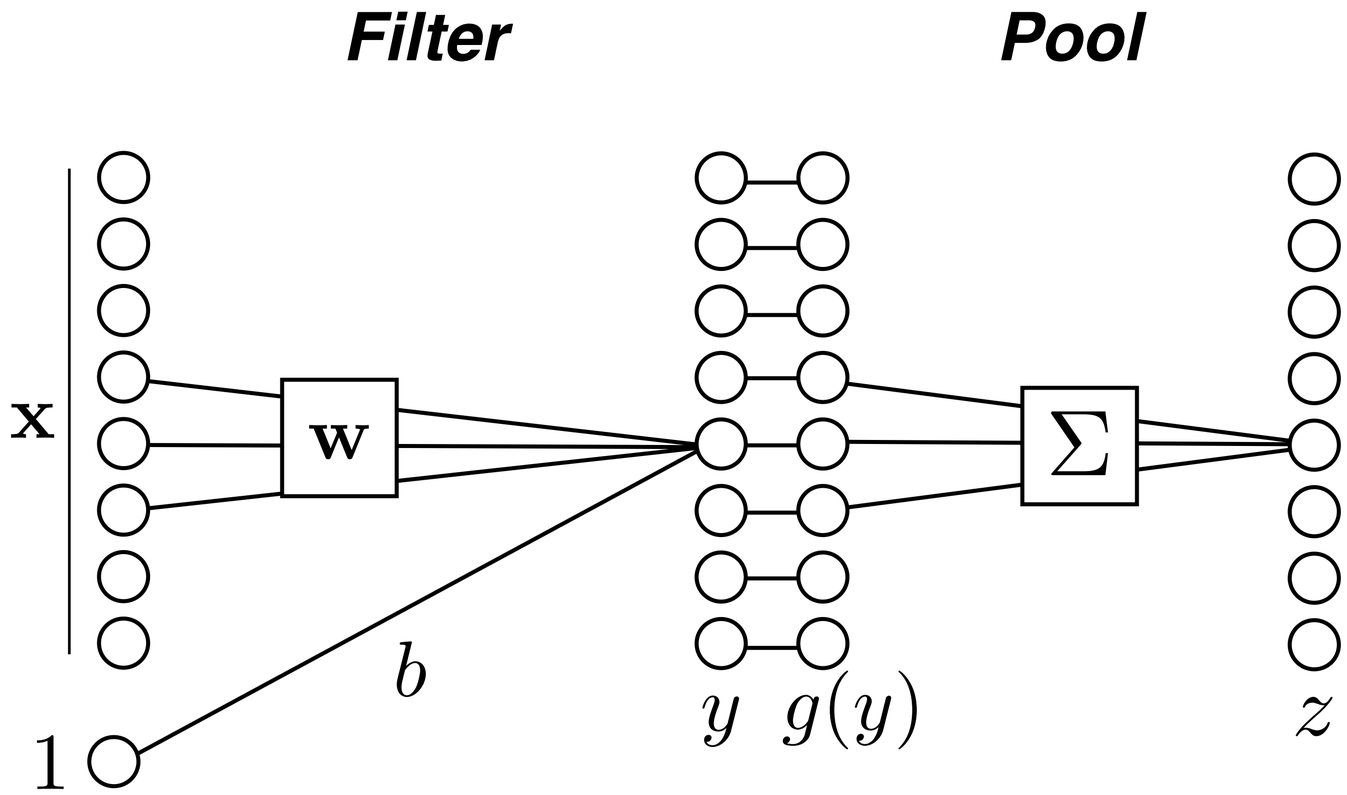


Max pooling

$$z_k = \max_{j \in \mathcal{N}(j)} g(y_j)$$

Source: Isola, Torralba, Freeman

Pooling



Max pooling

$$z_k = \max_{j \in \mathcal{N}(j)} g(y_j)$$

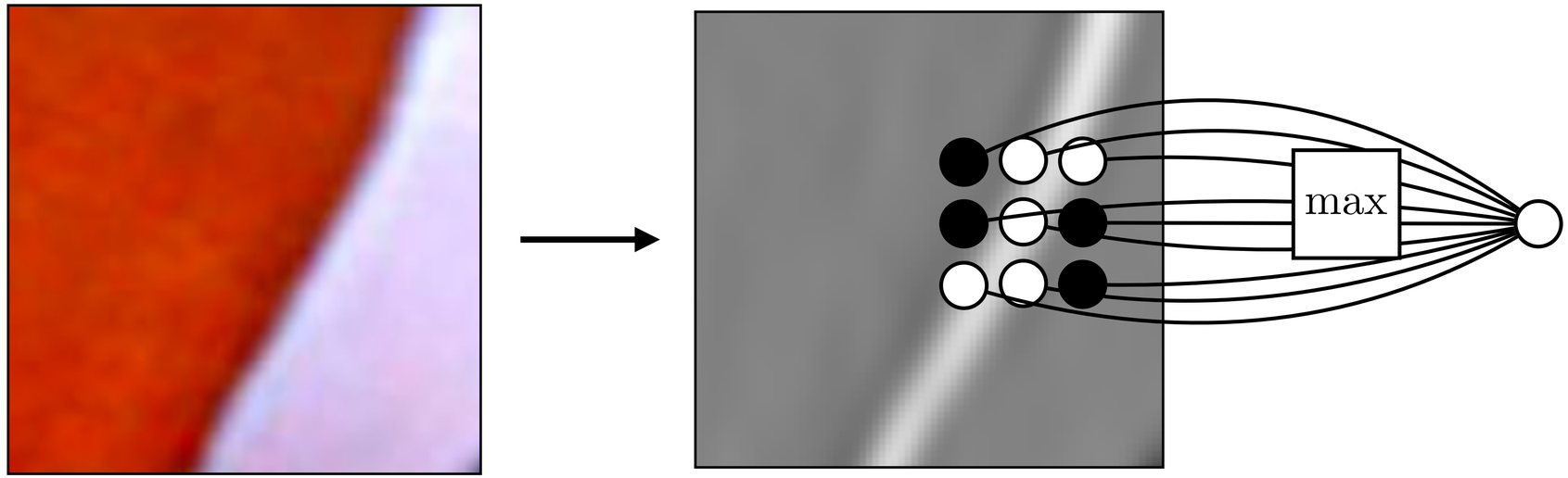
Mean pooling

$$z_k = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}(j)} g(y_j)$$

Source: Isola, Torralba, Freeman

Pooling — Why?

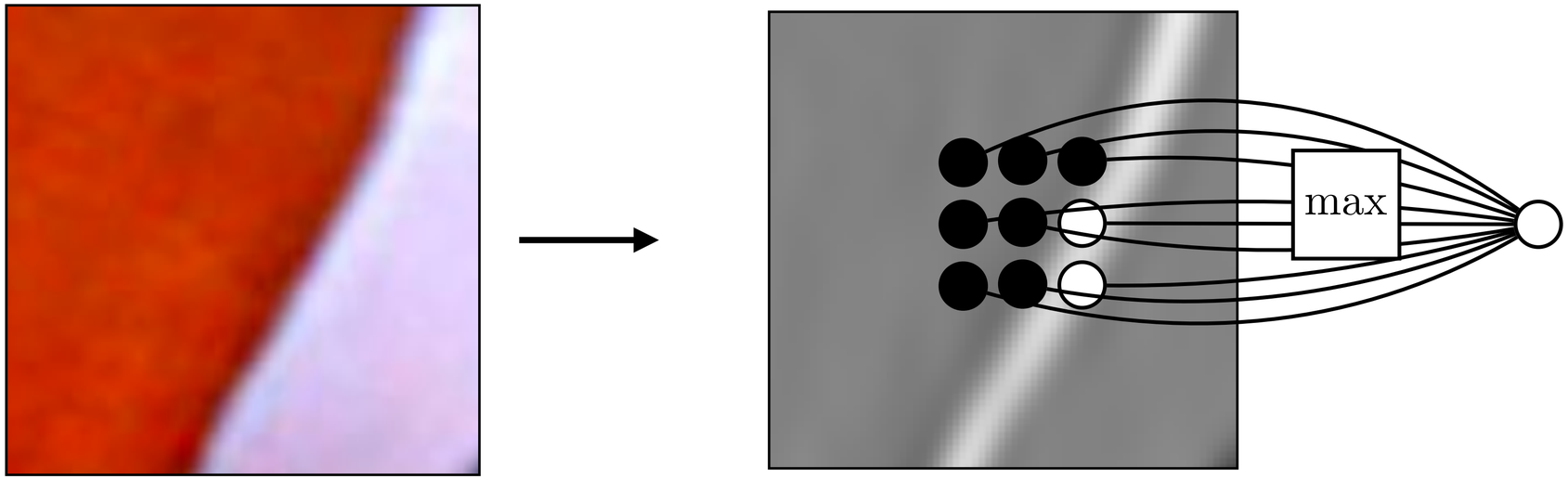
Pooling across spatial locations achieves stability w.r.t. small translations:



Source: Isola, Torralba, Freeman

Pooling — Why?

Pooling across spatial locations achieves stability w.r.t. small translations:

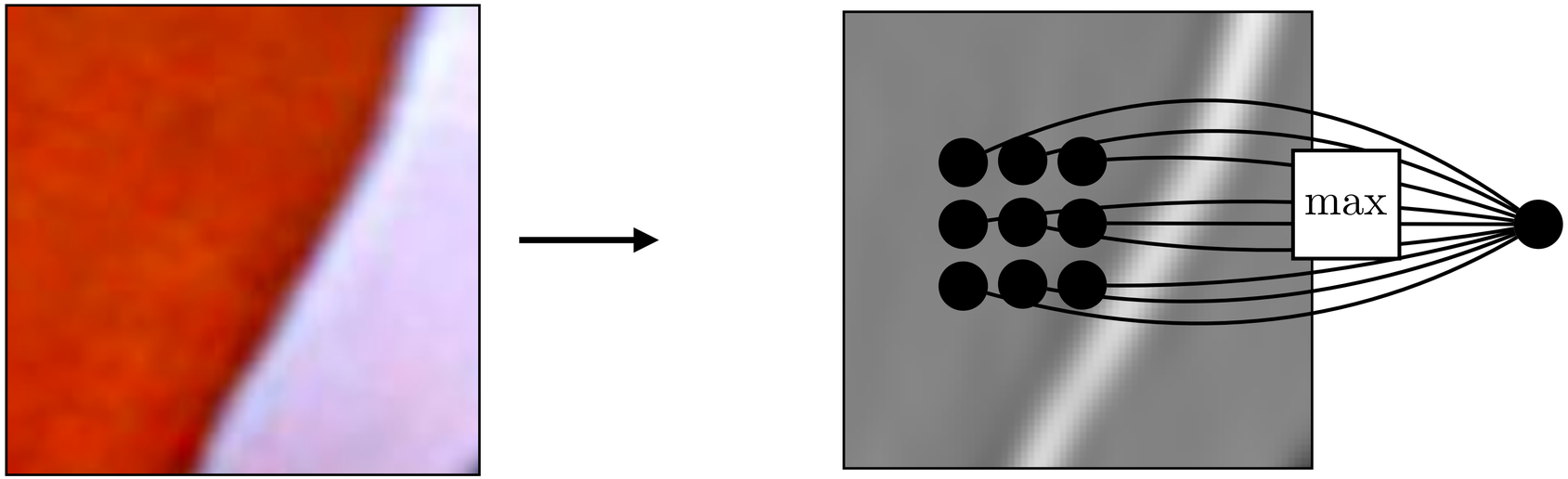


large response regardless of exact position of edge

Source: Isola, Torralba, Freeman

Pooling — Why?

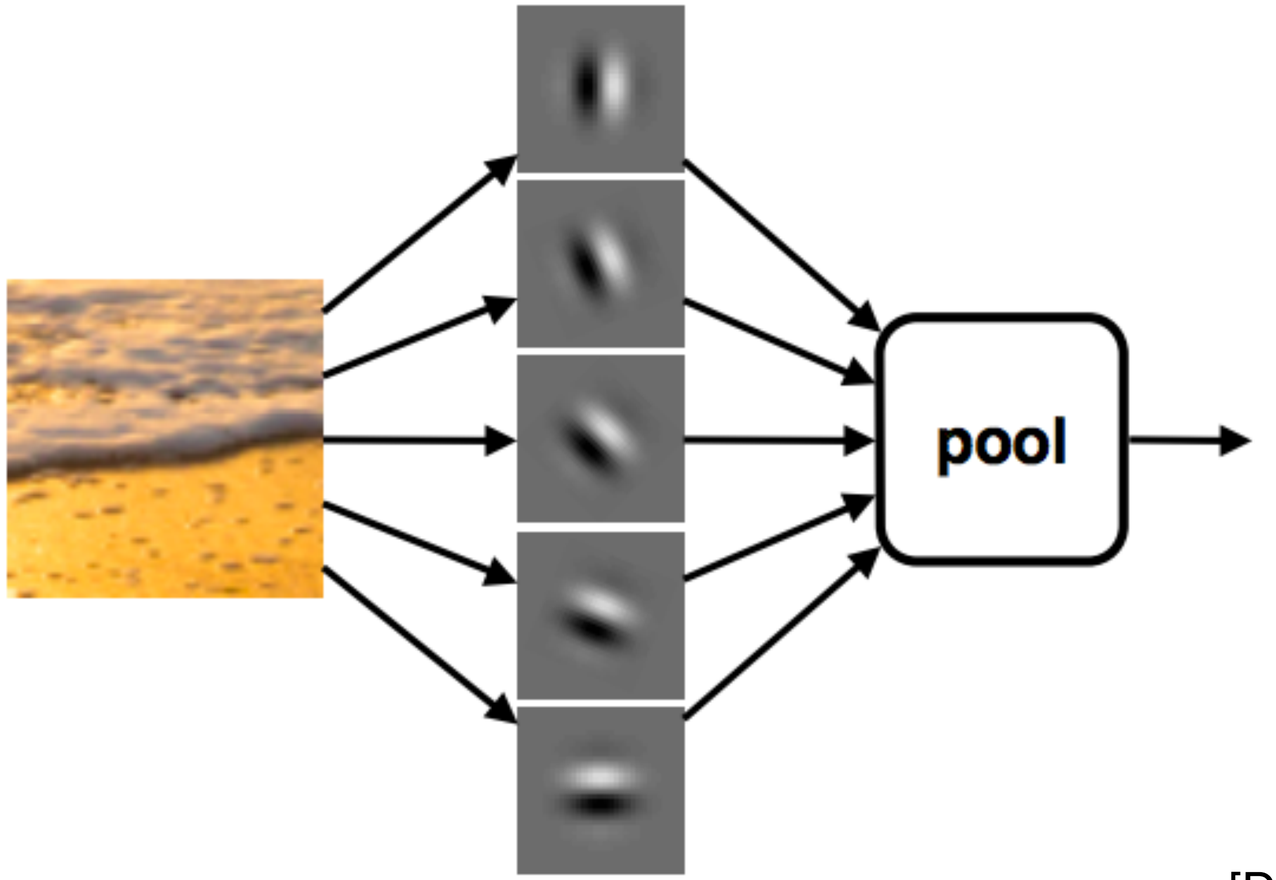
Pooling across spatial locations achieves stability w.r.t. small translations:



Source: Isola, Torralba, Freeman

Pooling — Why?

Pooling across feature channels (filter outputs) can achieve other kinds of invariances:

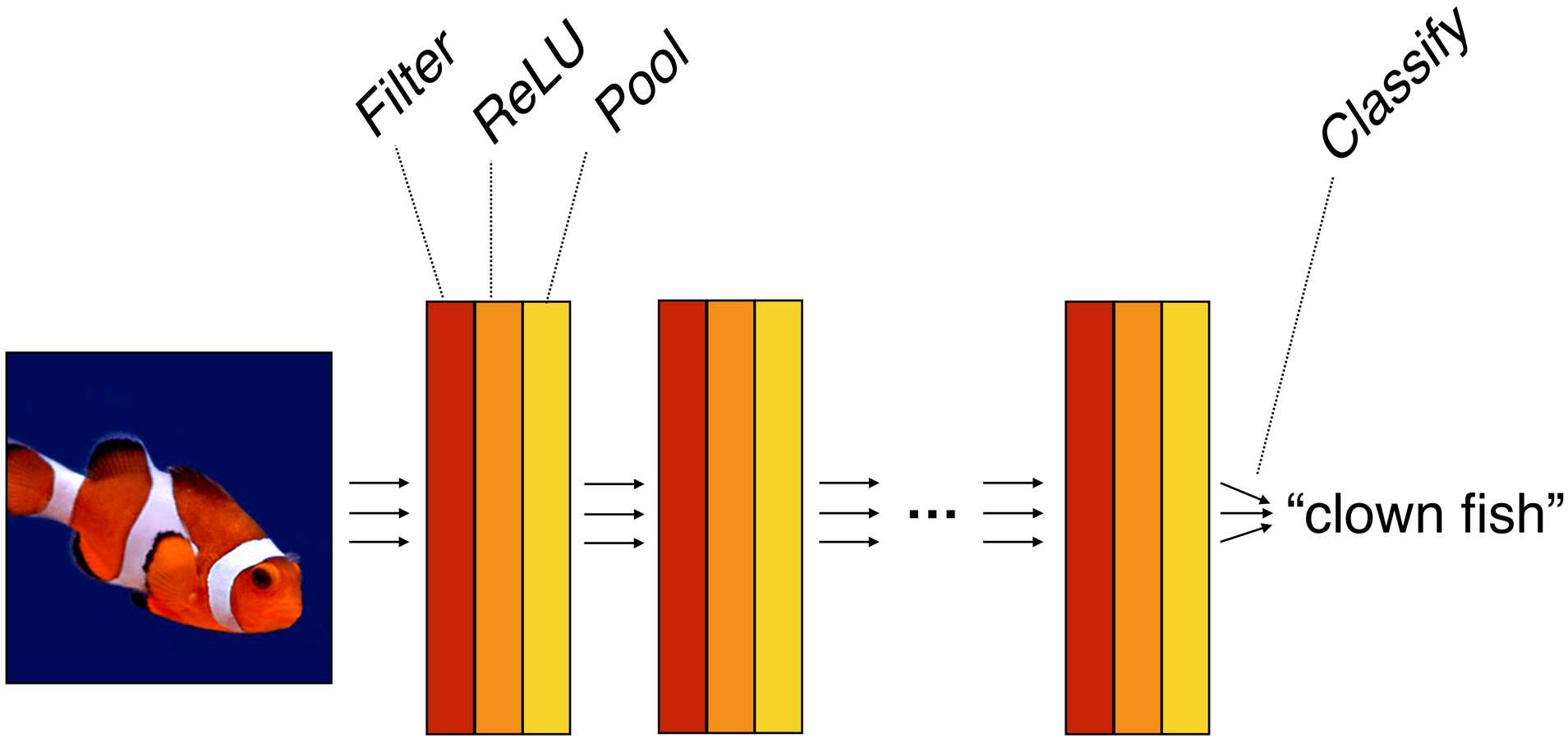


large response for any edge, regardless of its orientation

[Derived from slide by Andrea Vedaldi]

Source: Isola, Torralba, Freeman

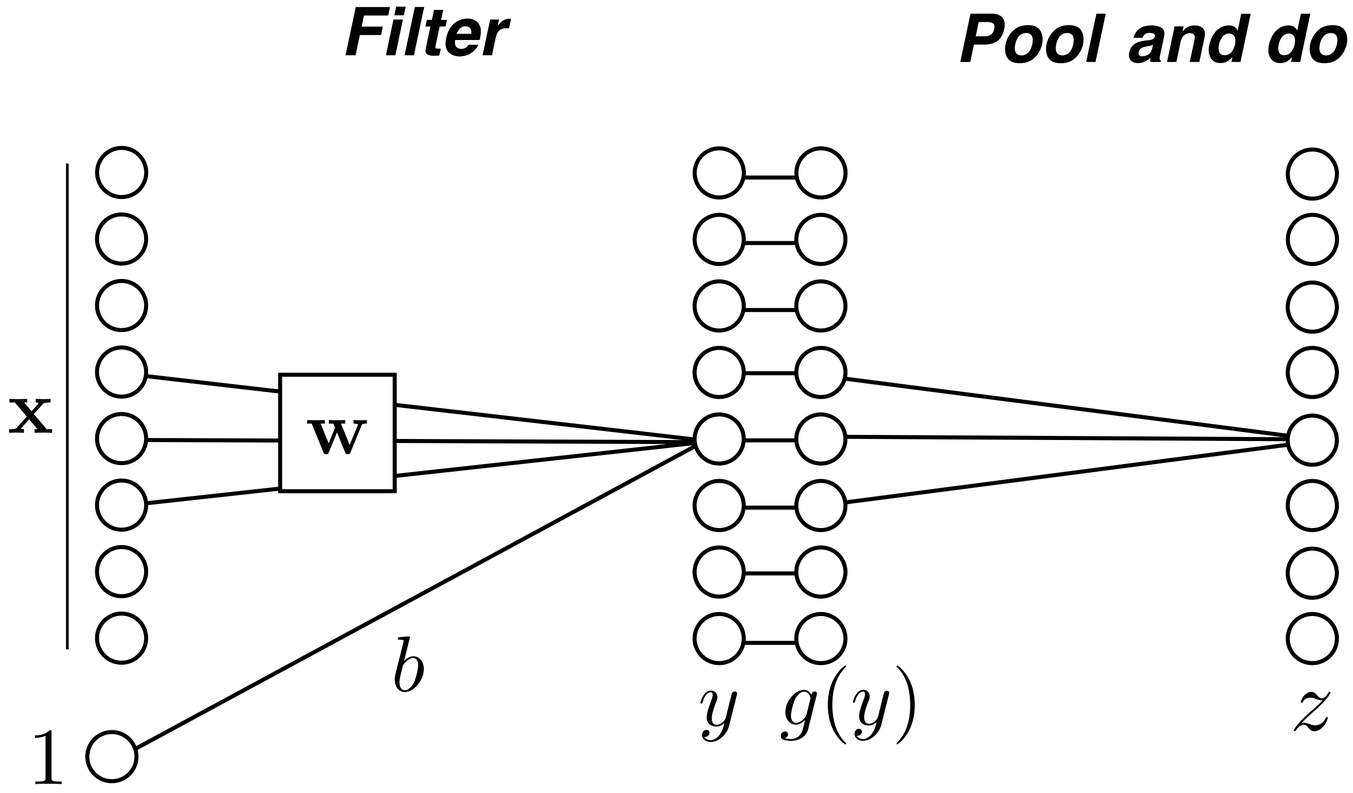
Computation in a neural net



$$f(\mathbf{x}) = f_L(\dots f_2(f_1(\mathbf{x})))$$

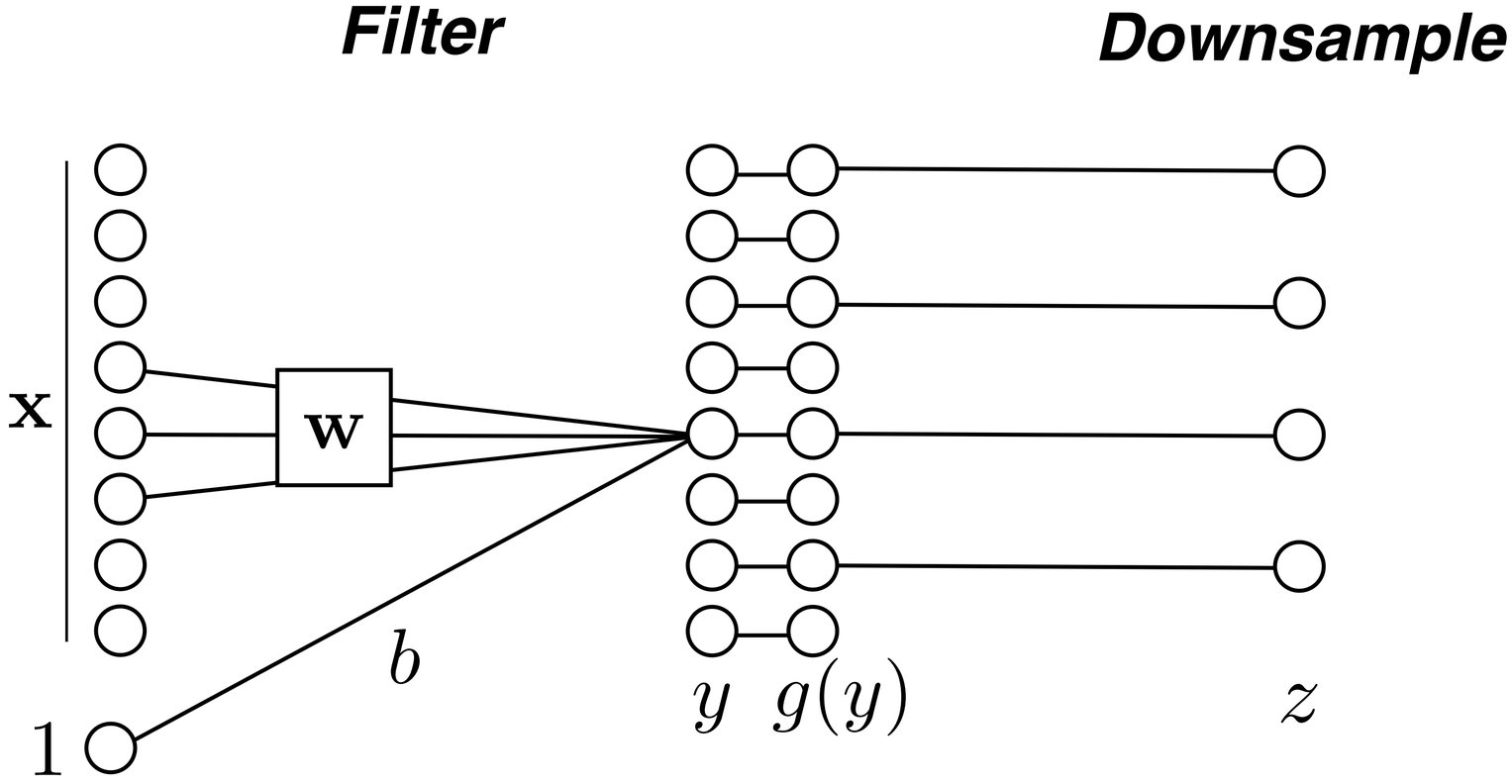
Source: Isola, Torralba, Freeman

Downsampling



Source: Isola, Torralba, Freeman

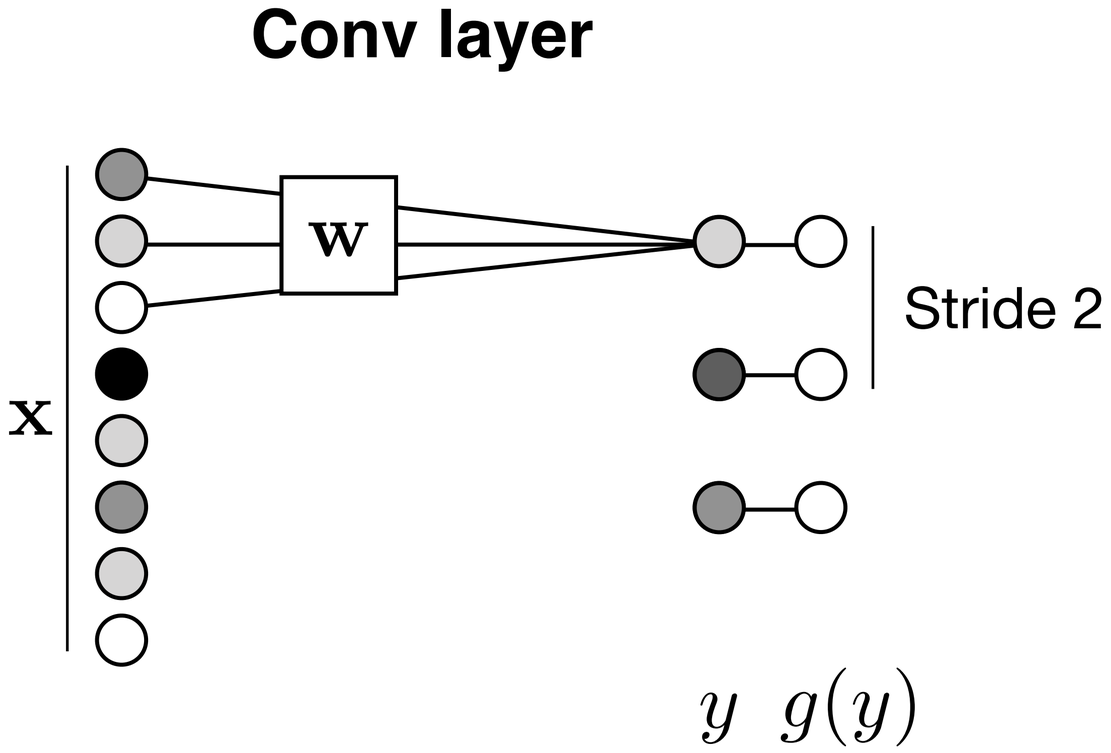
Downsampling



$$\mathbb{R}^{H^{(l)} \times W^{(l)} \times C^{(l)}} \rightarrow \mathbb{R}^{H^{(l+1)} \times W^{(l+1)} \times C^{(l+1)}}$$

Source: Isola, Torralba, Freeman

Strided operations

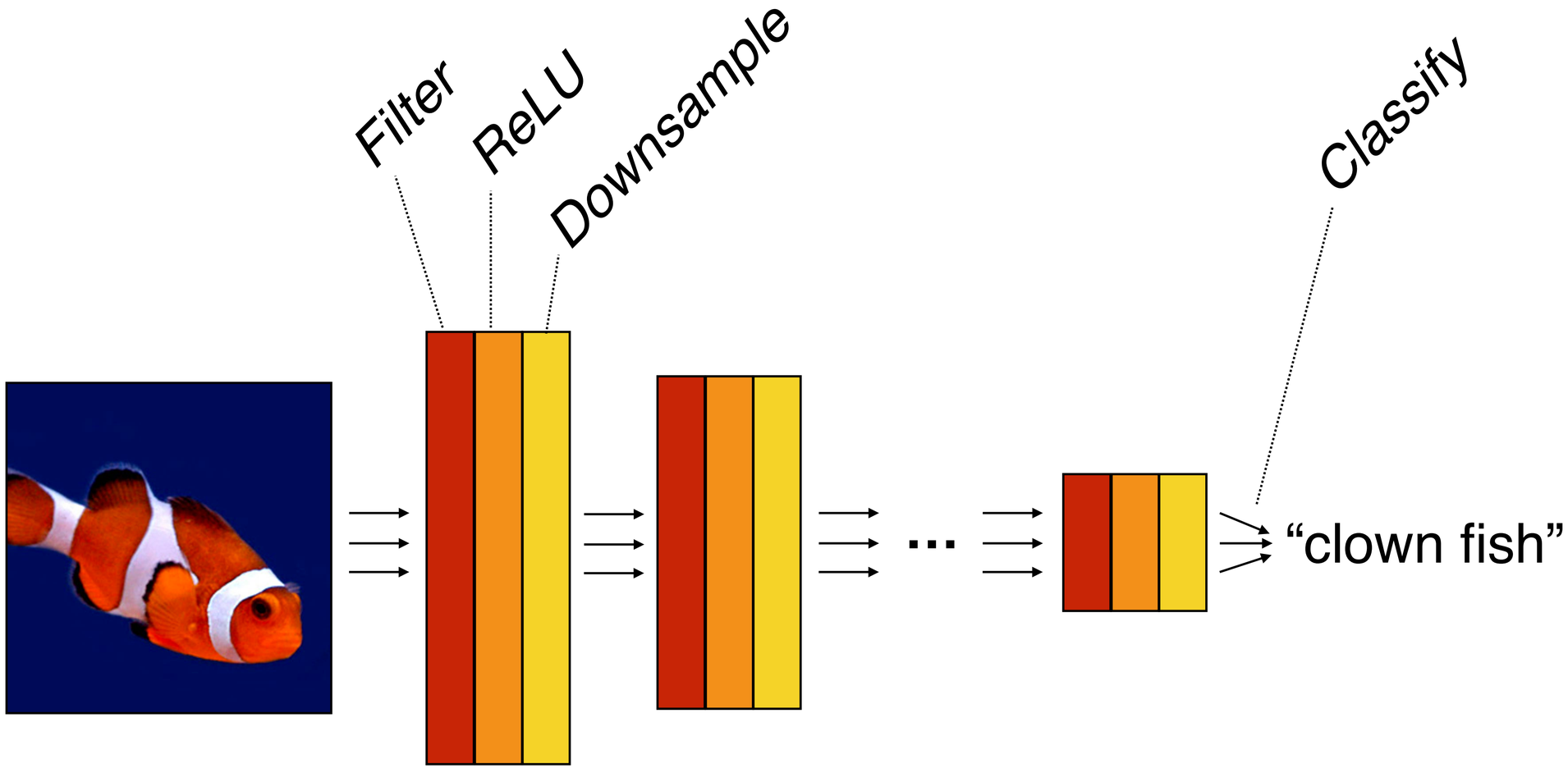


Strided operations combine a given operation (convolution or pooling) and downsampling into a single operation.

Strided convolution is an alternative to pooling layers: just do a strided convolution!

Source: Isola, Torralba, Freeman

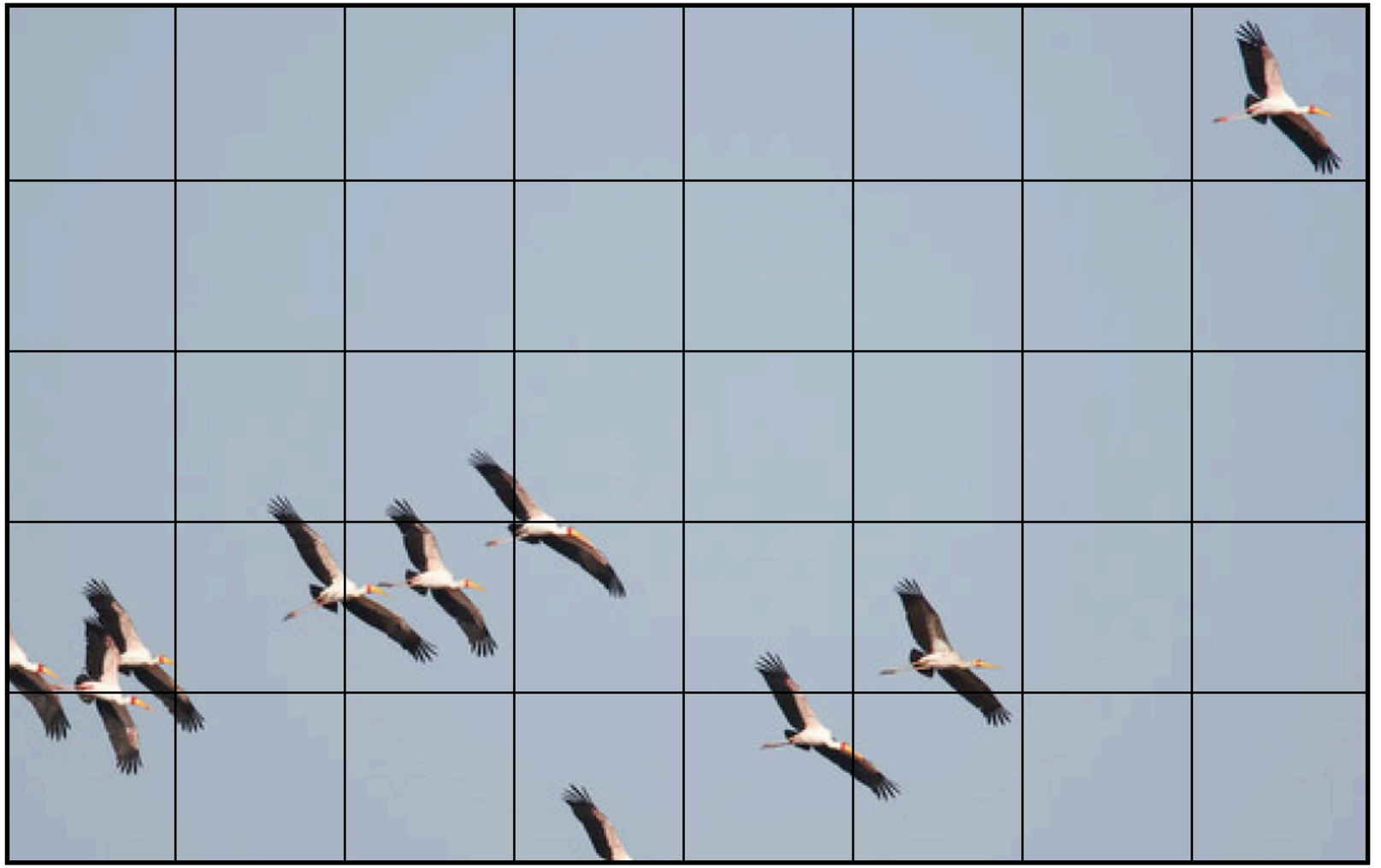
Computation in a neural net



$$f(\mathbf{x}) = f_L(\dots f_2(f_1(\mathbf{x})))$$

Source: Isola, Torralba, Freeman

Receptive fields



Source: Isola, Torralba, Freeman

Receptive fields

Pool and downsample by 2

3x1 Filter

Pool and downsample by 2

RF = RF*2

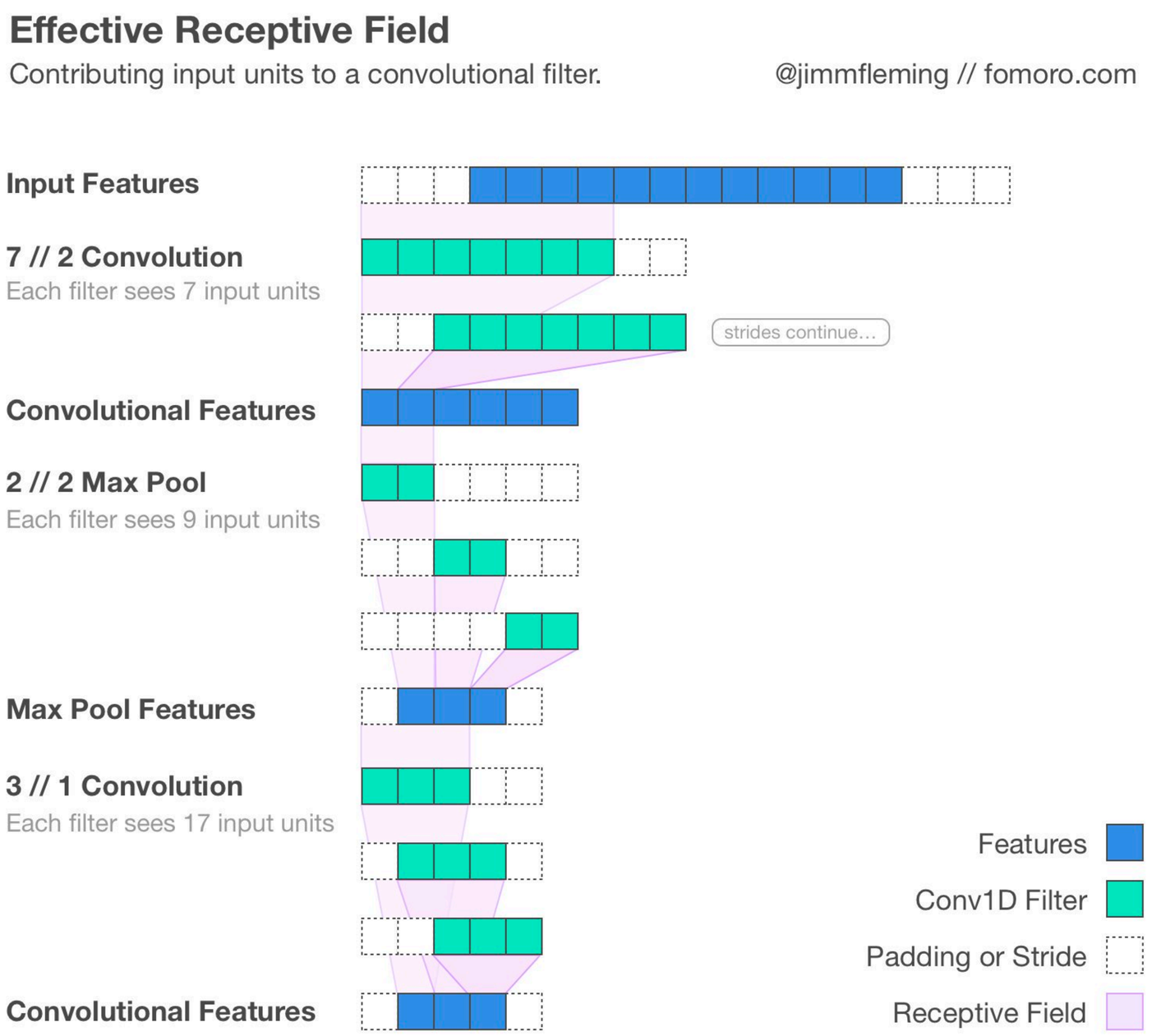
RF = RF + floor(3/2)*2

RF = RF*2

kernel size

scale factor

Source: Isola, Torralba, Freeman



[http://fomoro.com/tools/receptive-fields/index.html]

Source: Isola, Torralba, Freeman

Gradients for backprop

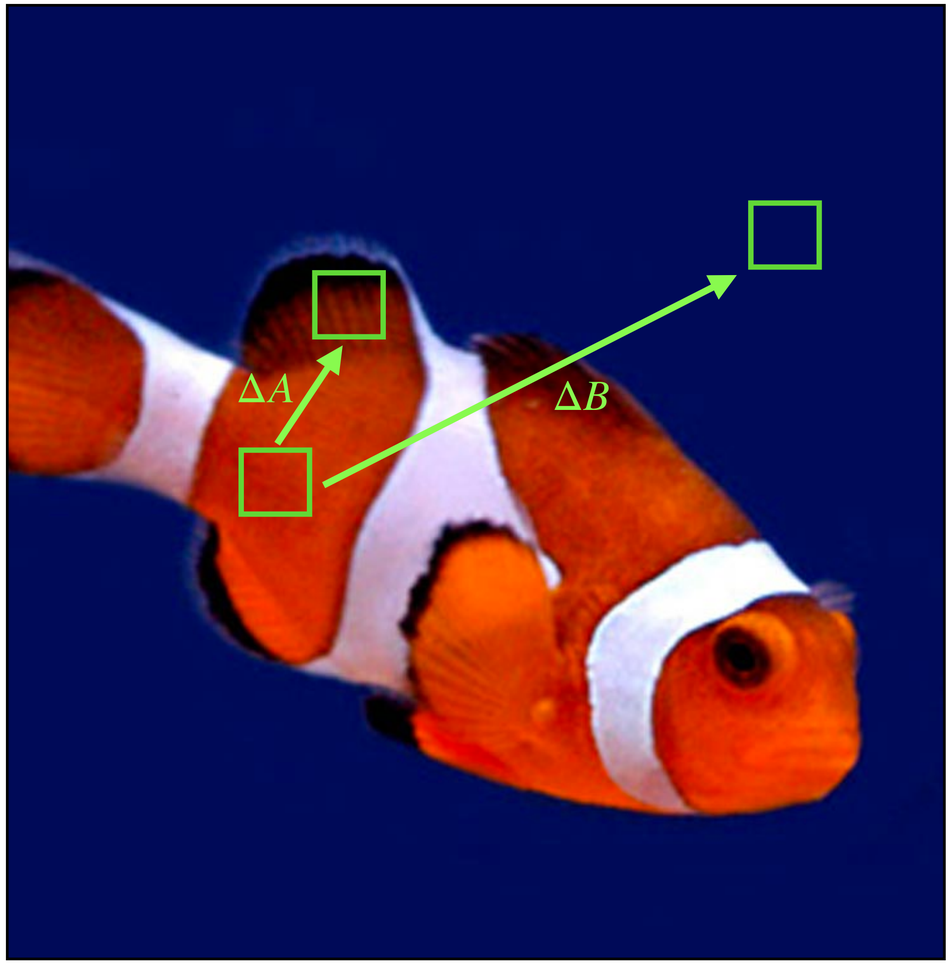
Convolutional layers:

Option 1: Just make the giant convolution matrix and use the fully backprop equations for fully connected layers!
Early deep learning frameworks did this.

Option 2: Use a more optimized implementation. Result looks similar to convolution.

Max pooling: similar to ReLU gradient.

Local vs. global processing



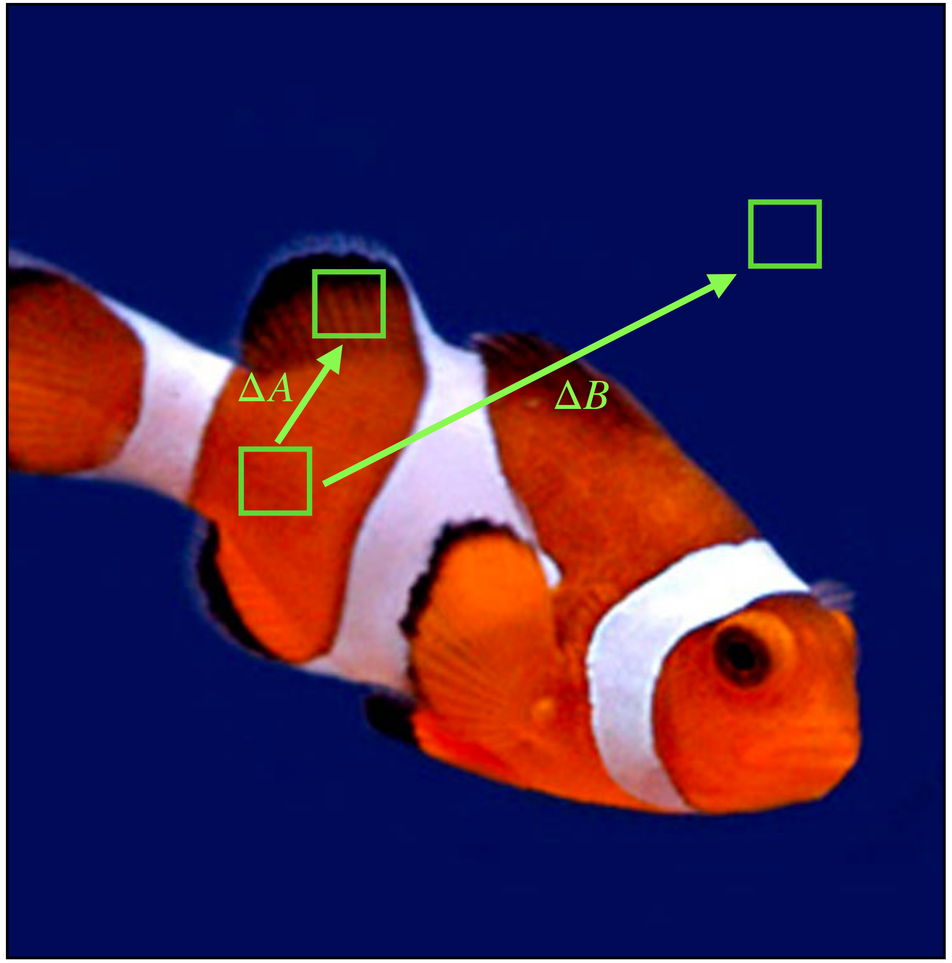
Across all images, which is higher:

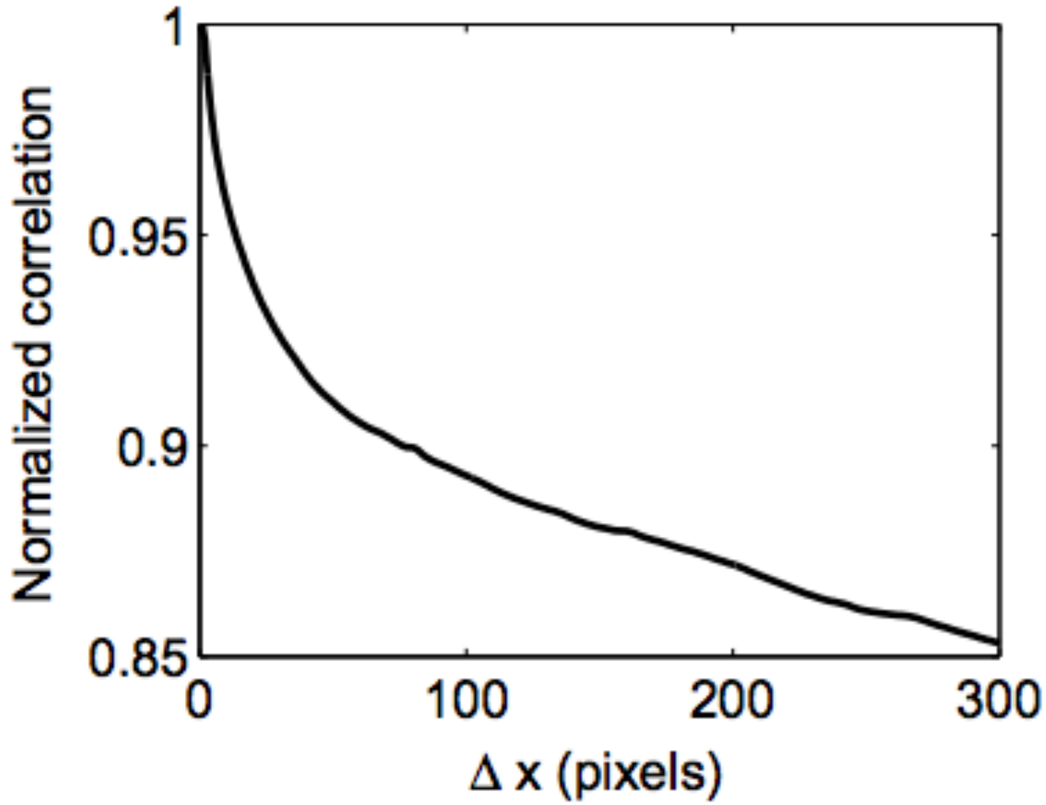
(1) correlation between points with distance ΔA

(2) correlation between points with distance ΔB

(3) can't say

Local vs. global processing





[Simoncelli; Statistical Modelling of Photographic Images, 2005]

CNNs — Why?

Statistical dependences between pixels decay as a power law of distance between the pixels.

It is therefore often sufficient to model local dependences only. —> **Convolution**

More generally, we should allocate parameters that model dependencies in proportion to the strength of those dependences. —> **Multiscale, hierarchical representations**

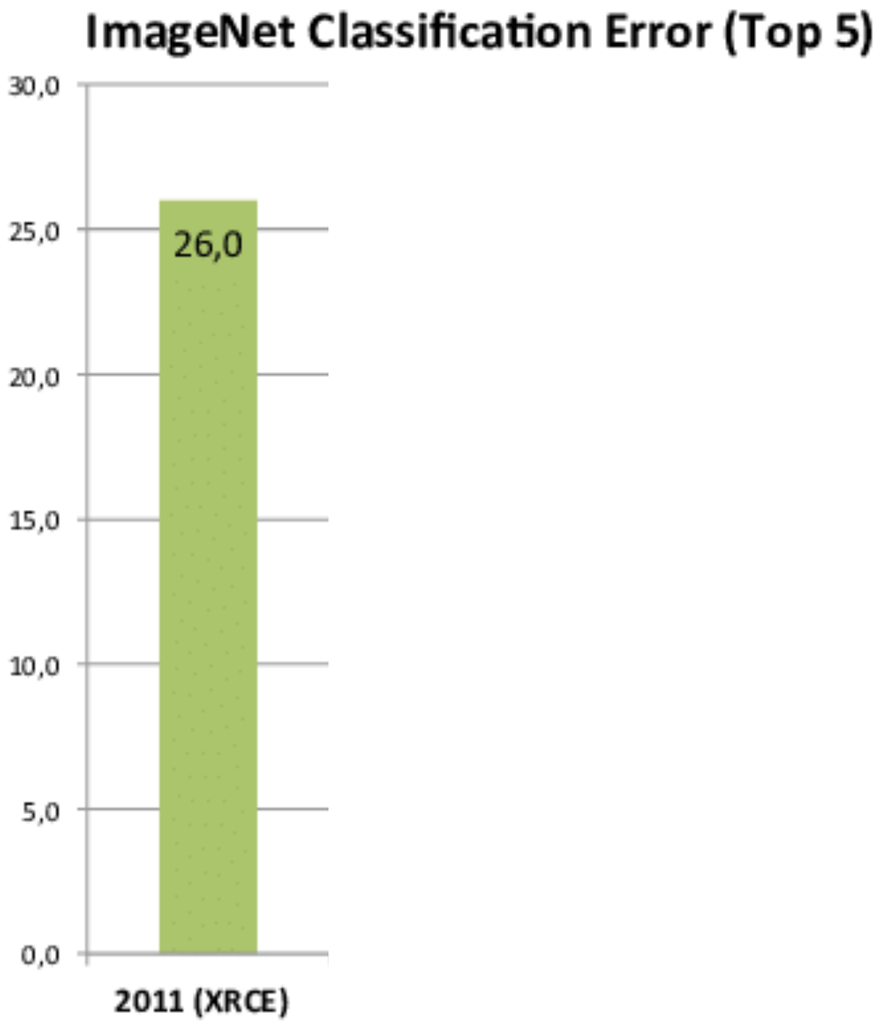
[For more discussion, see “Why does Deep and Cheap Learning Work So Well?”, Lin et al. 2017]

Source: Isola, Torralba, Freeman

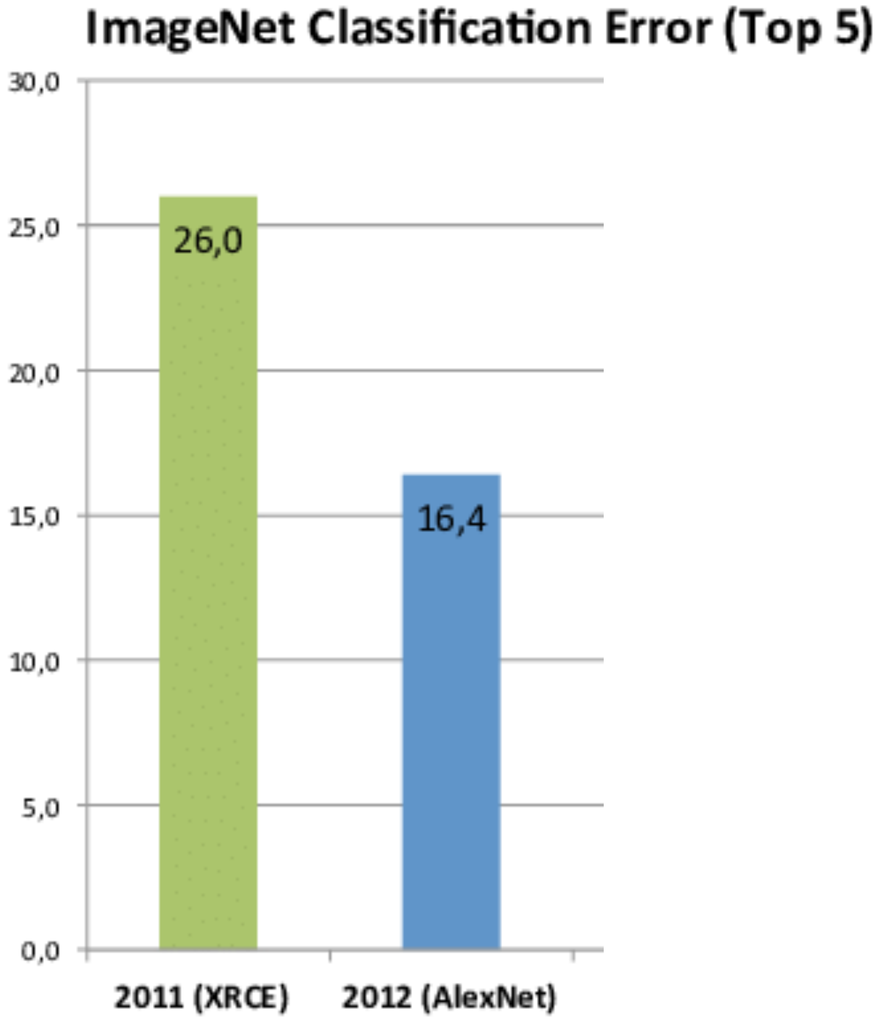
Some networks

… and what makes them work

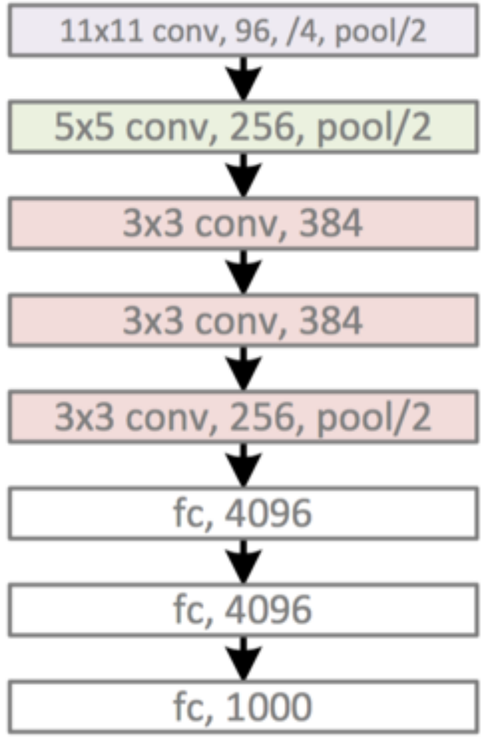
Source: Isola, Torralba, Freeman



Source: Isola, Torralba, Freeman



2012: AlexNet
5 conv. layers

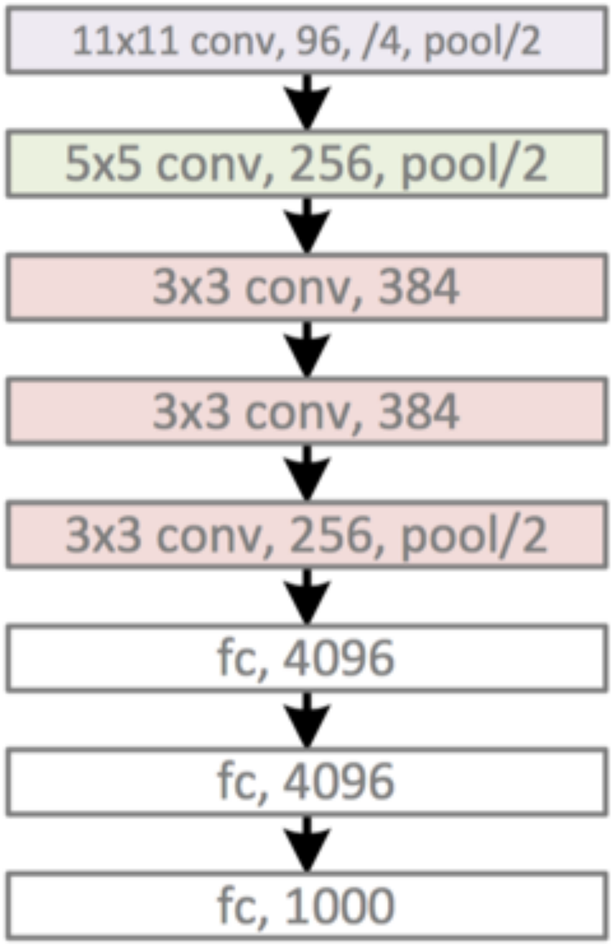


Error: 16.4%

[Krizhevsky et al: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012]

Source: Isola, Torralba, Freeman

Alexnet — [Krizhevsky et al. NIPS 2012]



[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

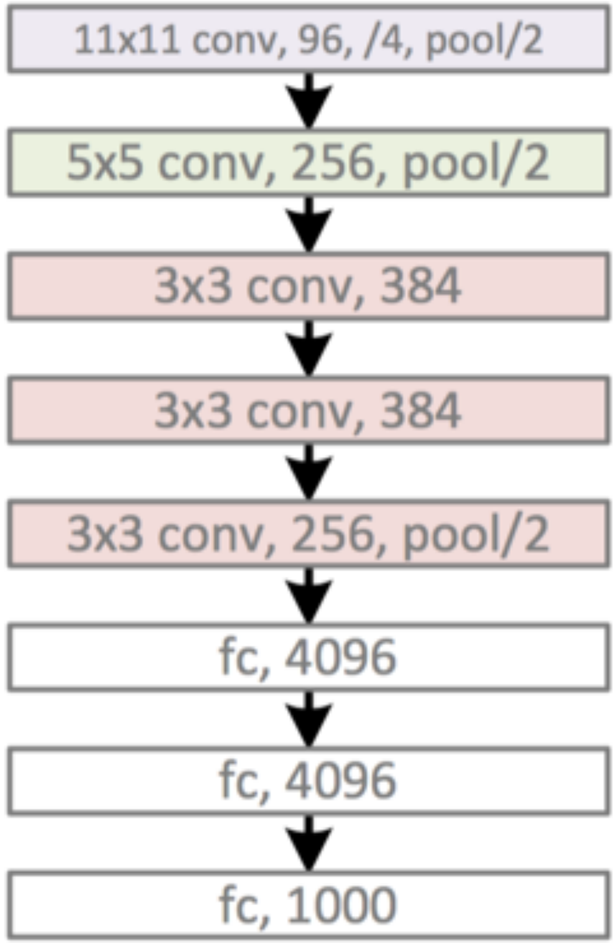
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

Source: Isola, Torralba, Freeman

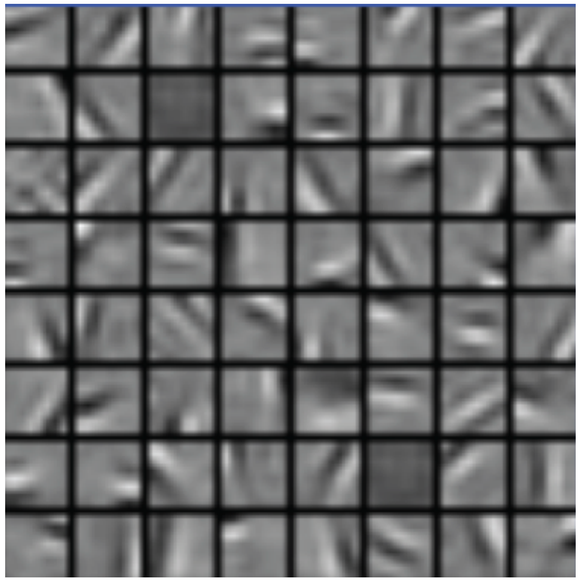


What filters are learned?

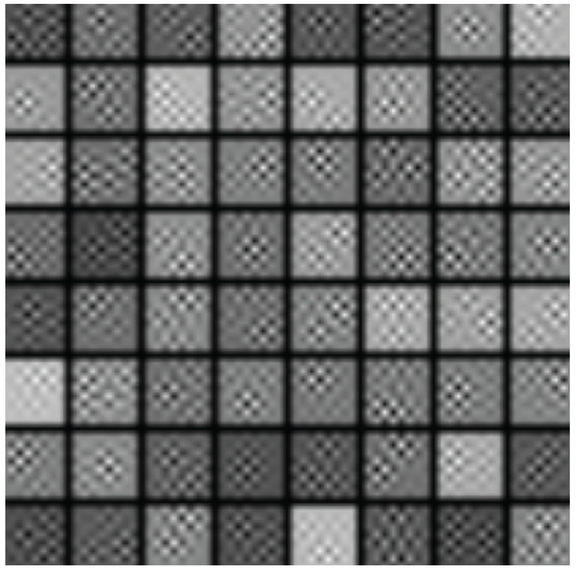
Source: Isola, Torralba, Freeman

What filters are learned?

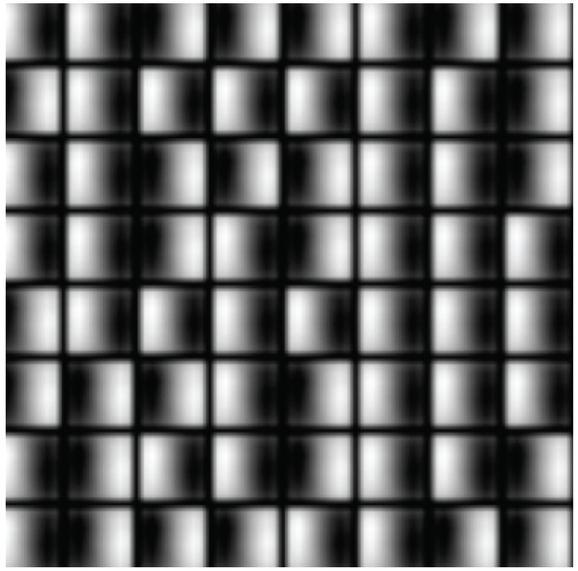
A



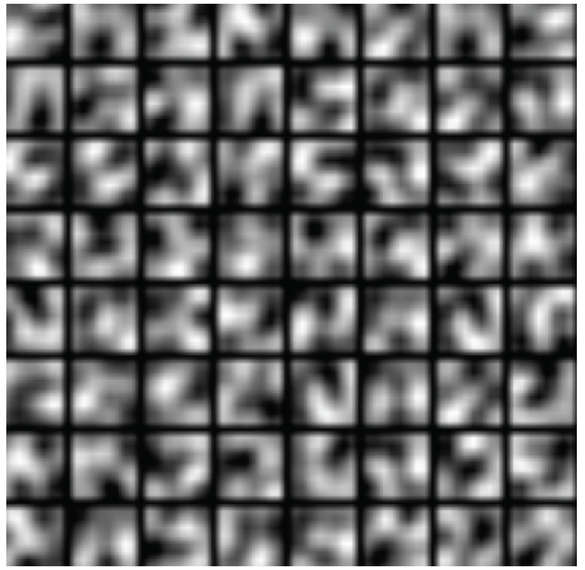
B



C

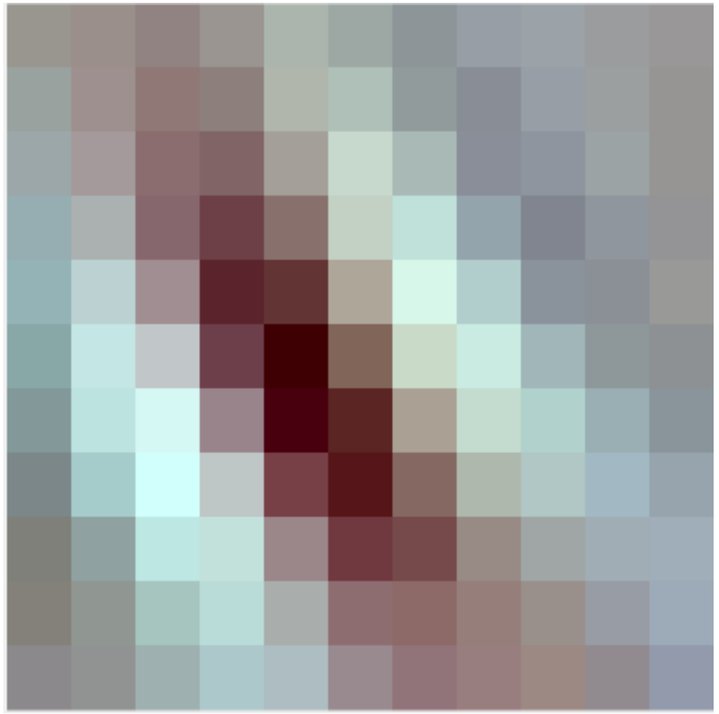


D

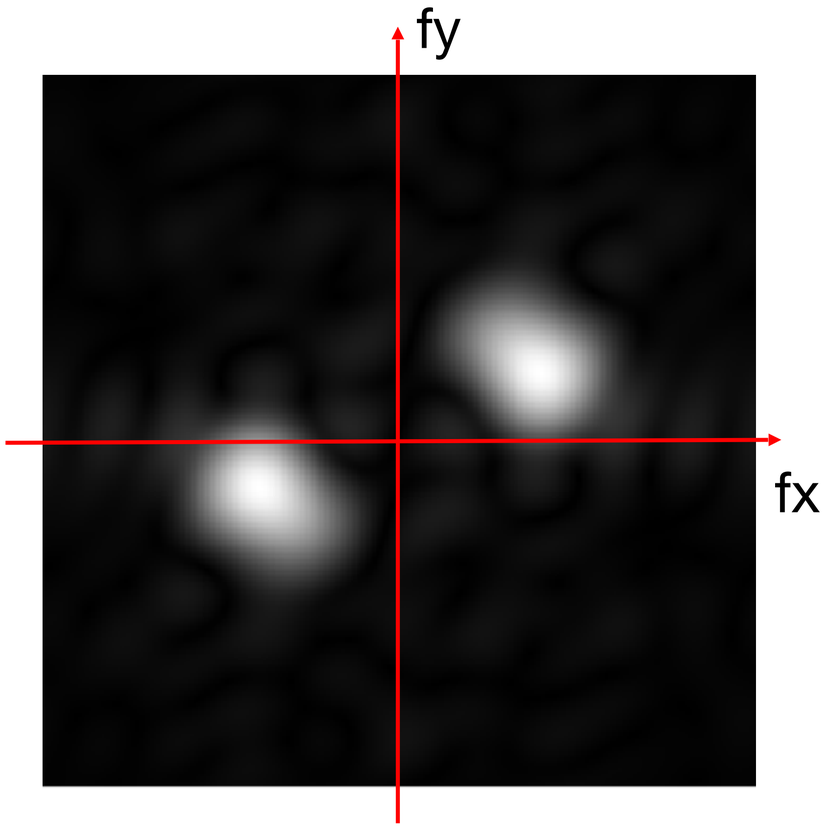


Source: Isola, Torralba, Freeman

Get to know your units

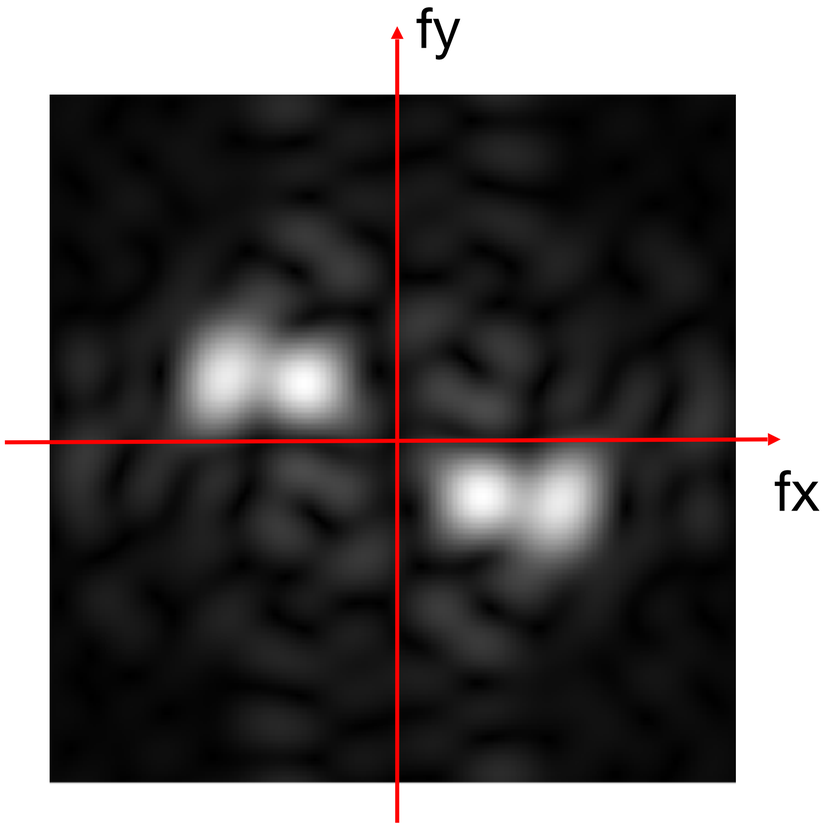


11x11 convolution kernel
(3 color channels)



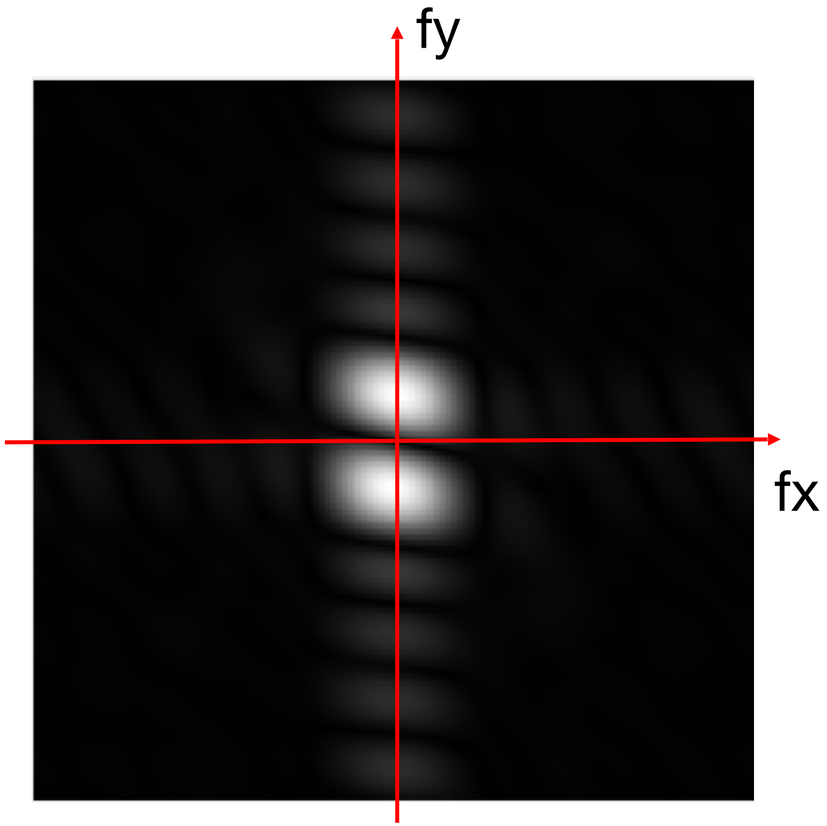
Source: Isola, Torralba, Freeman

Get to know your units



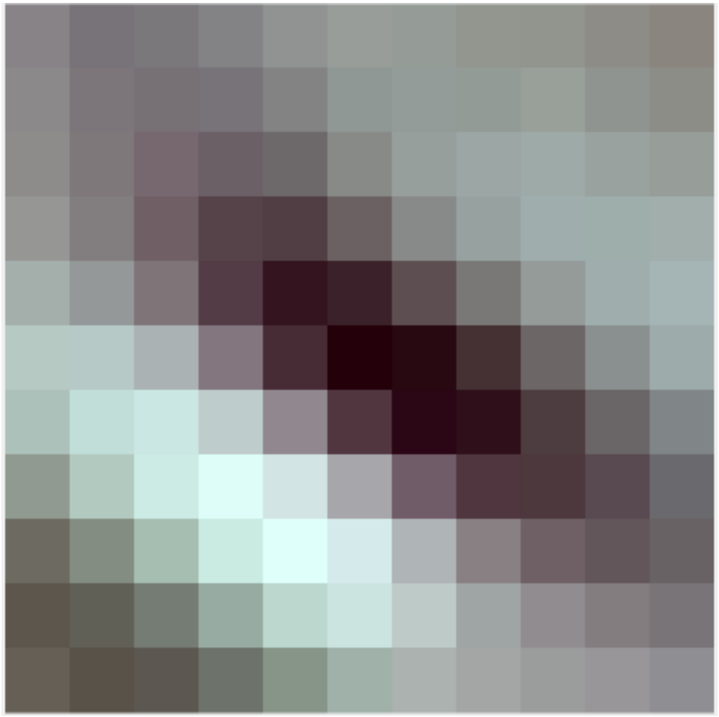
Source: Isola, Torralba, Freeman

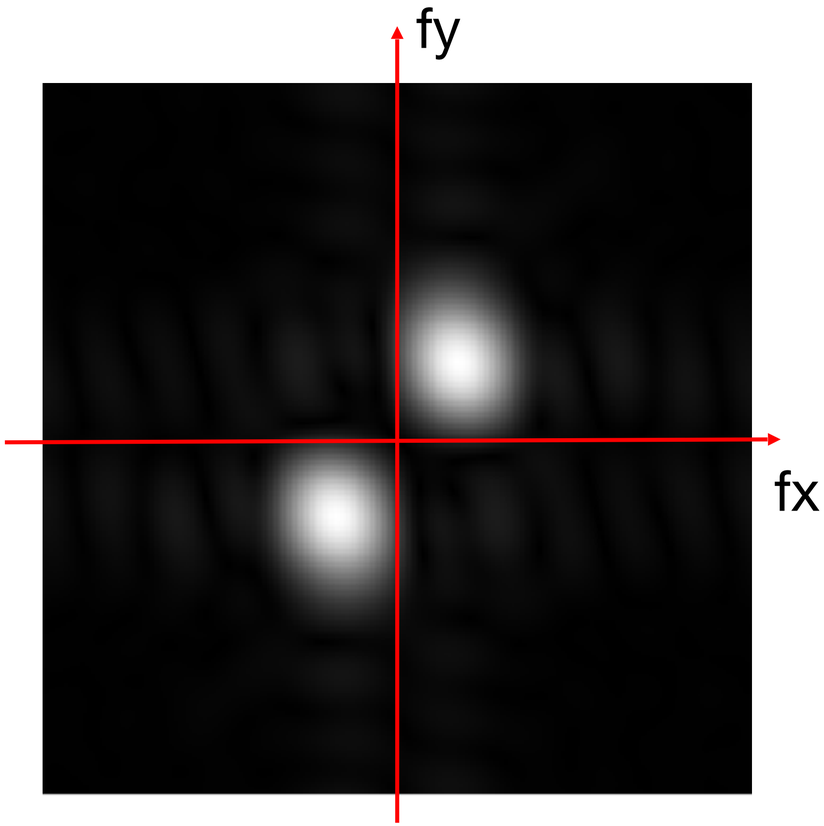
Get to know your units



Source: Isola, Torralba, Freeman

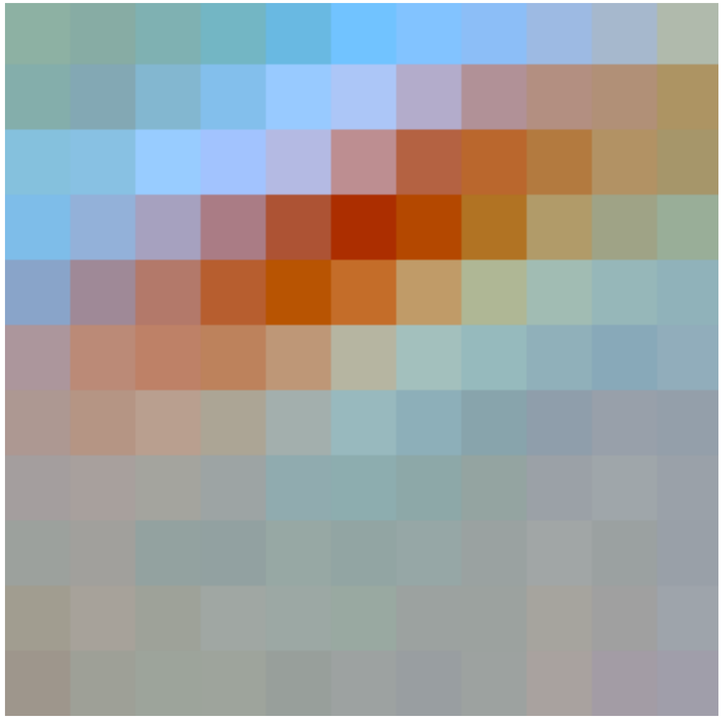
Get to know your units

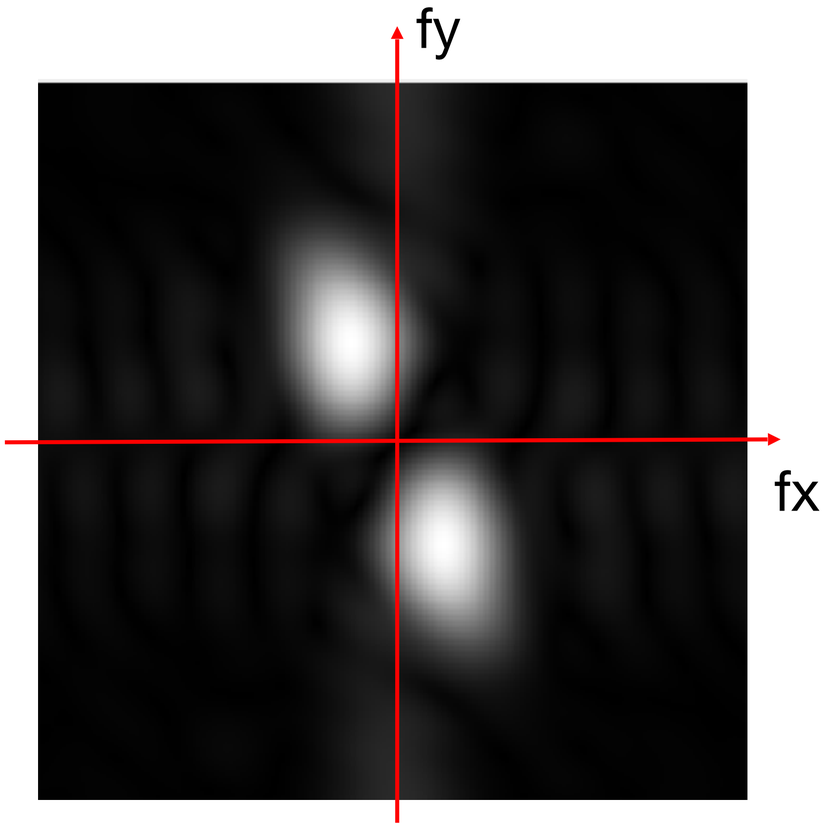




Source: Isola, Torralba, Freeman

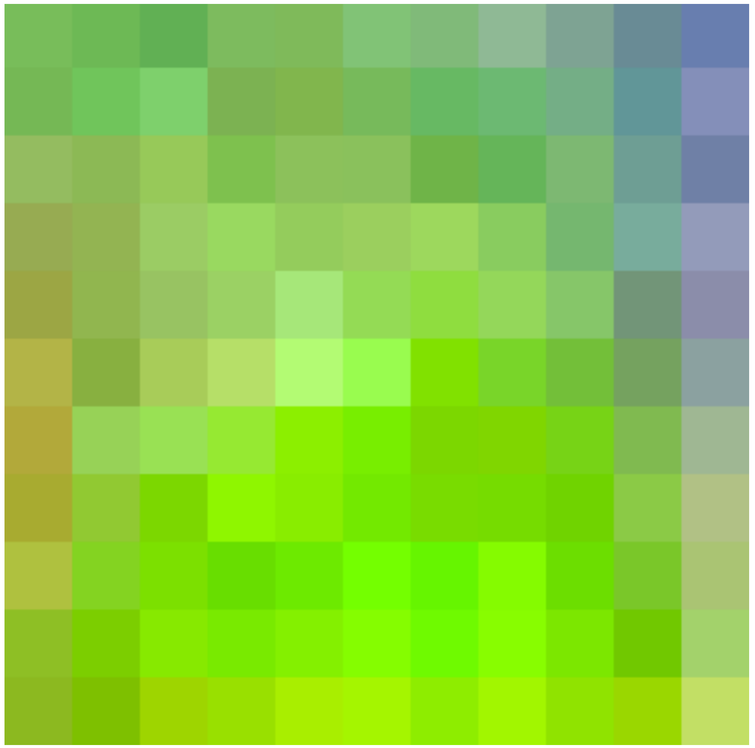
Get to know your units

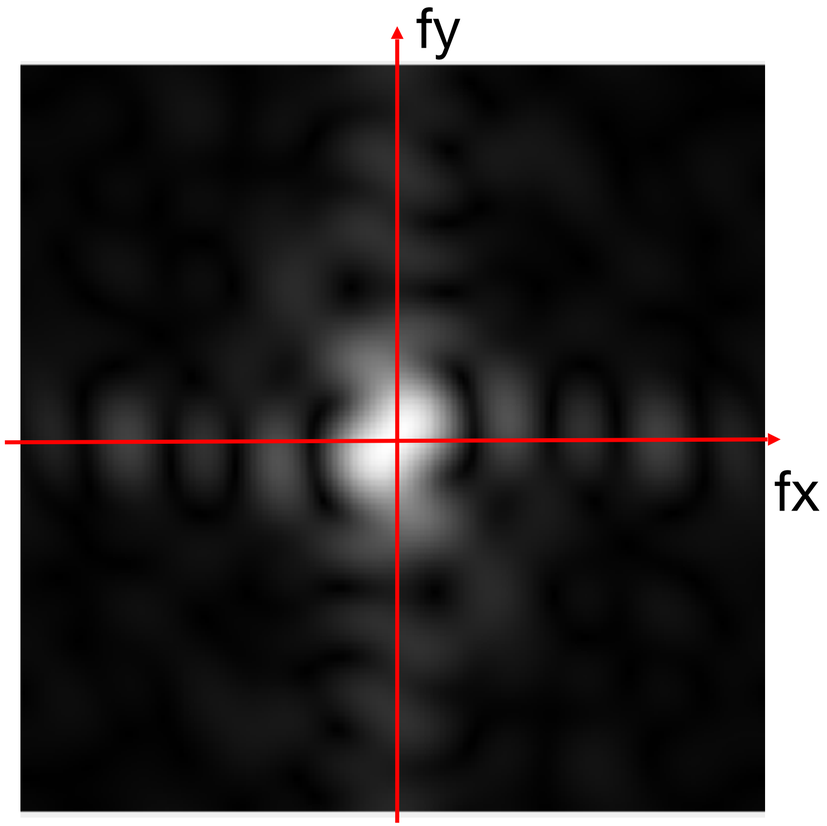




Source: Isola, Torralba, Freeman

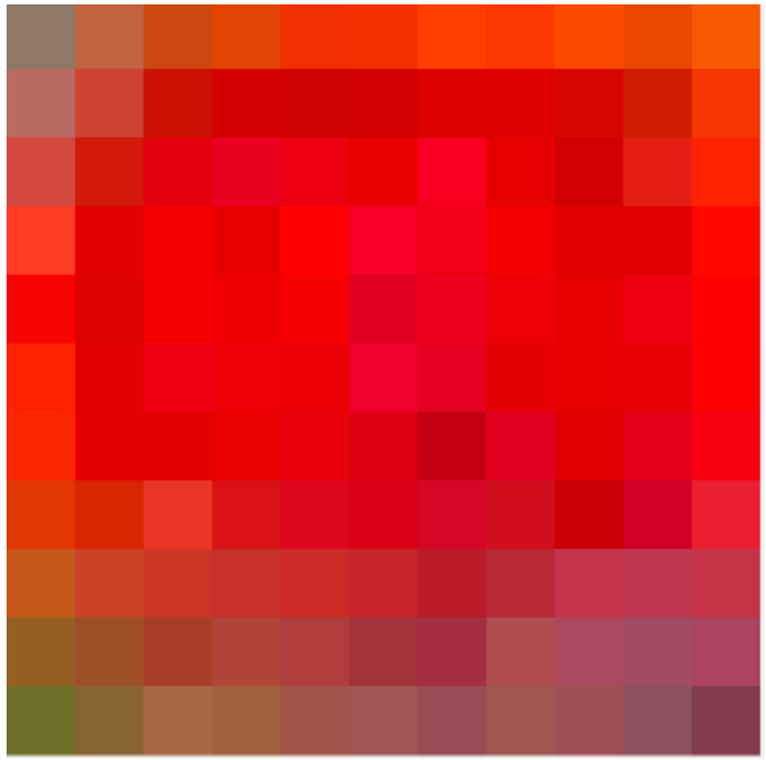
Get to know your units

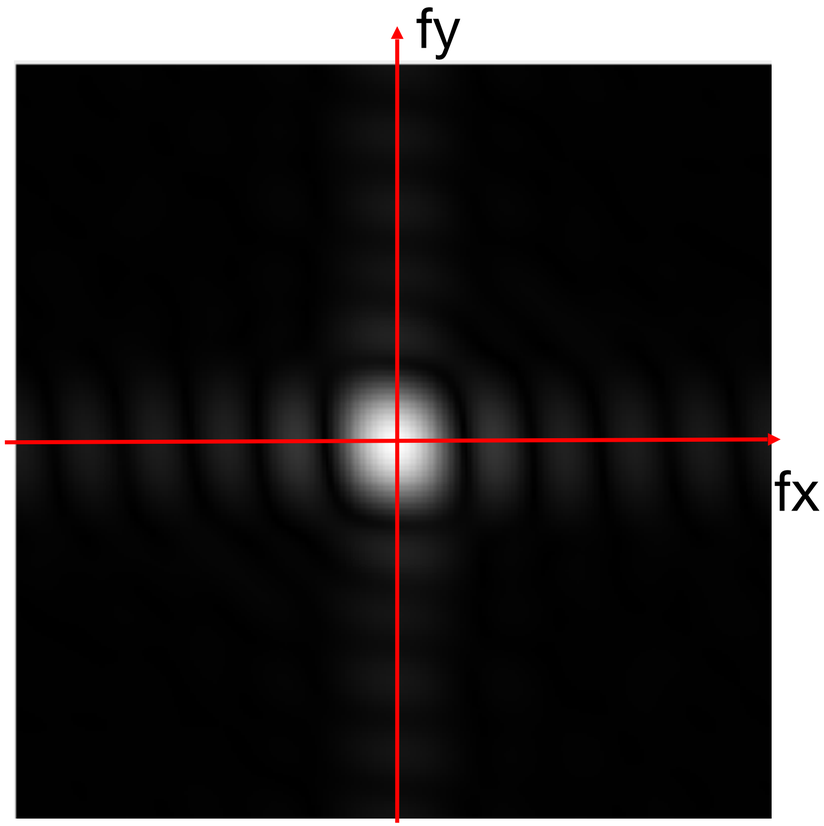




Source: Isola, Torralba, Freeman

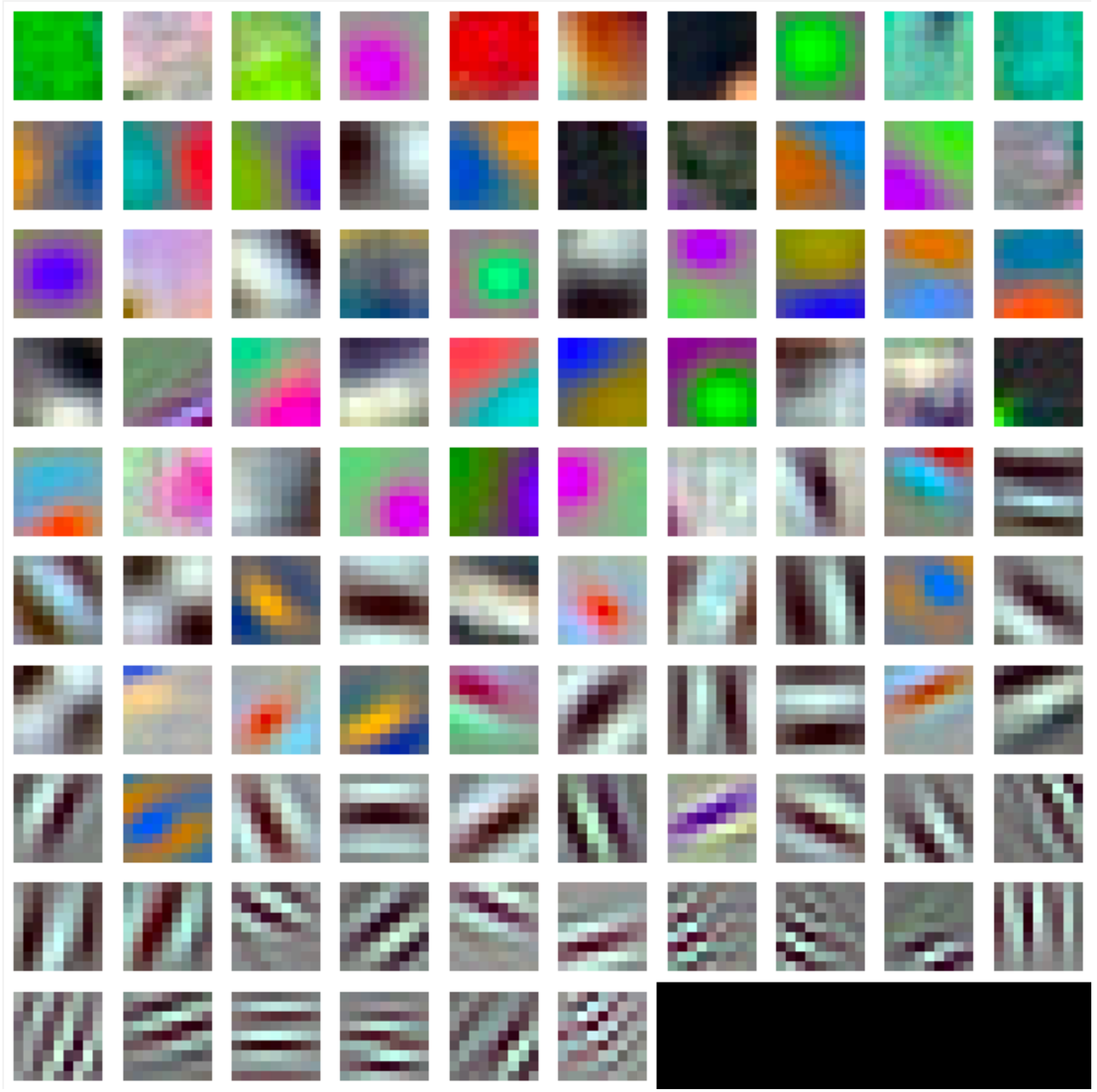
Get to know your units



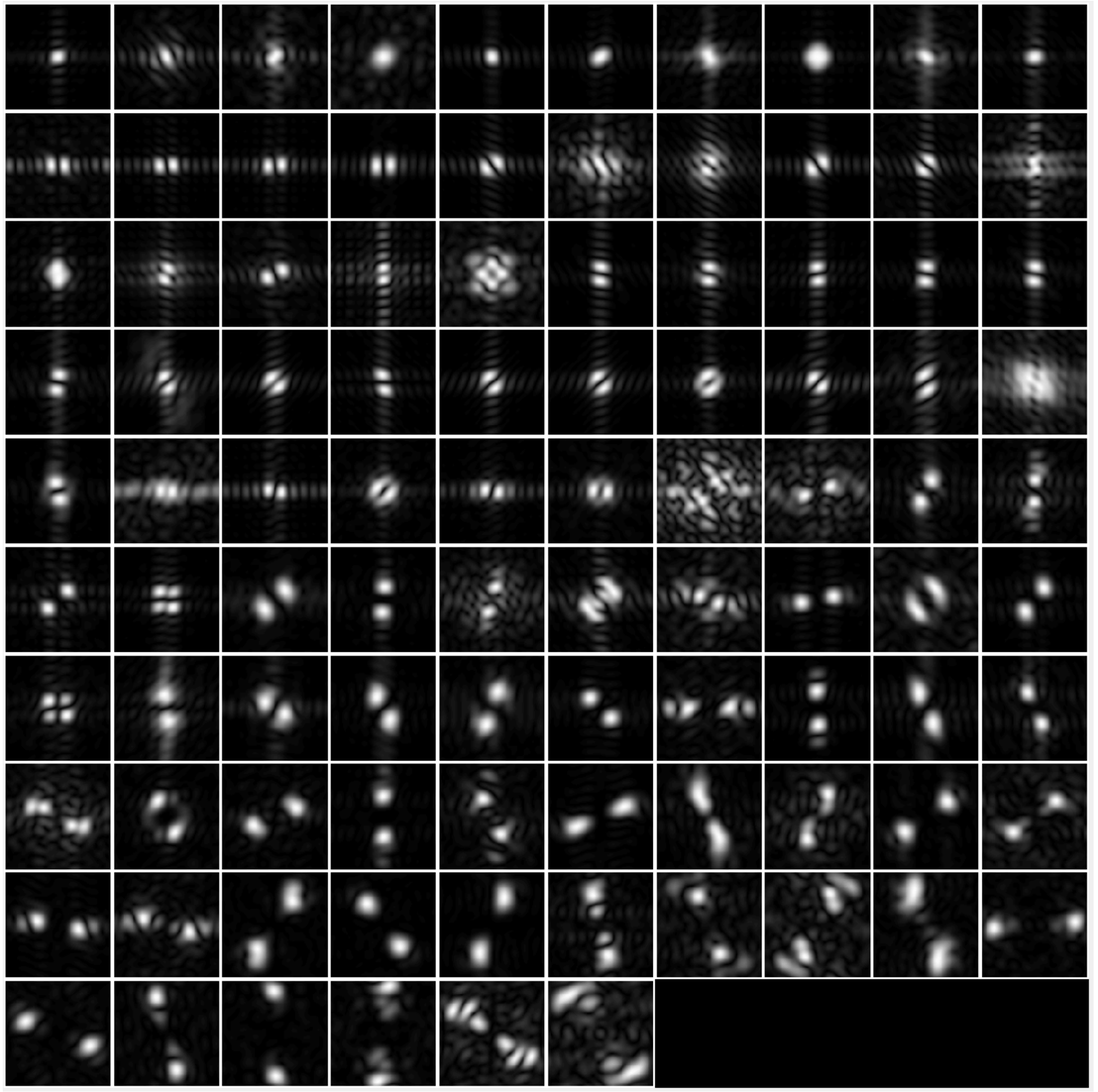


Source: Isola, Torralba, Freeman

Get to know your units

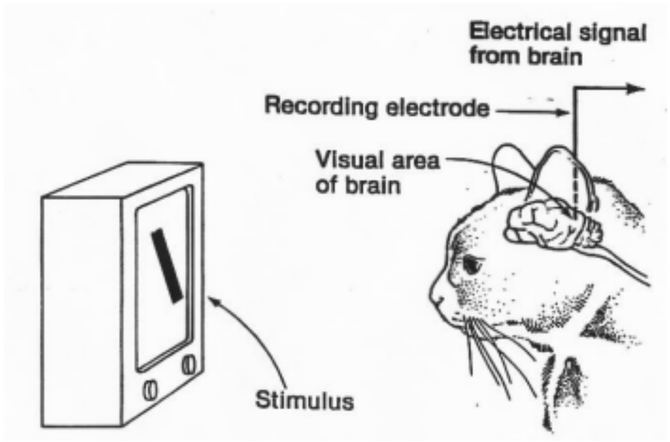


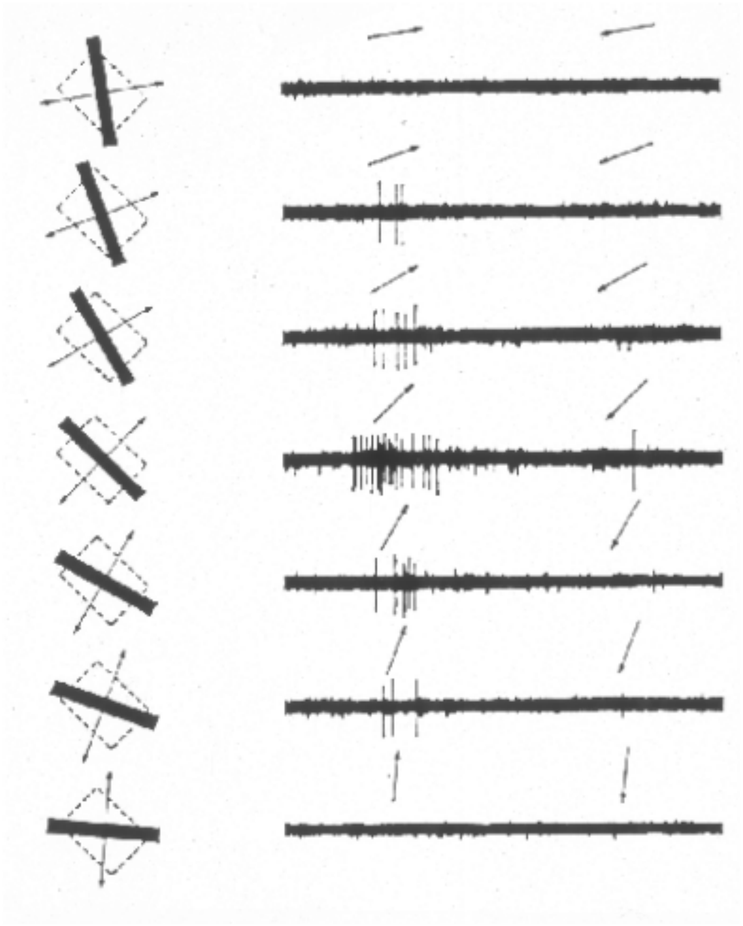
96 Units in conv1

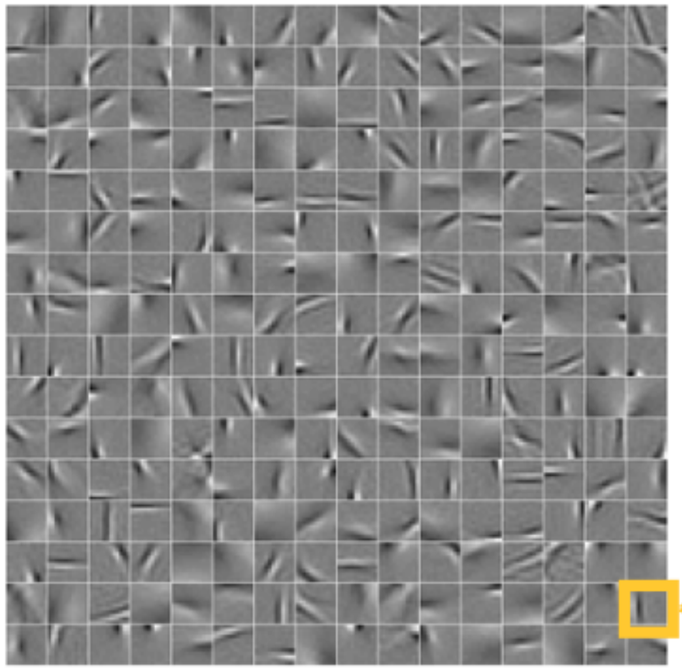


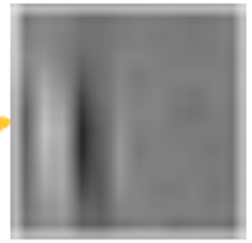
Source: Isola, Torralba, Freeman

[Hubel and Wiesel 59]







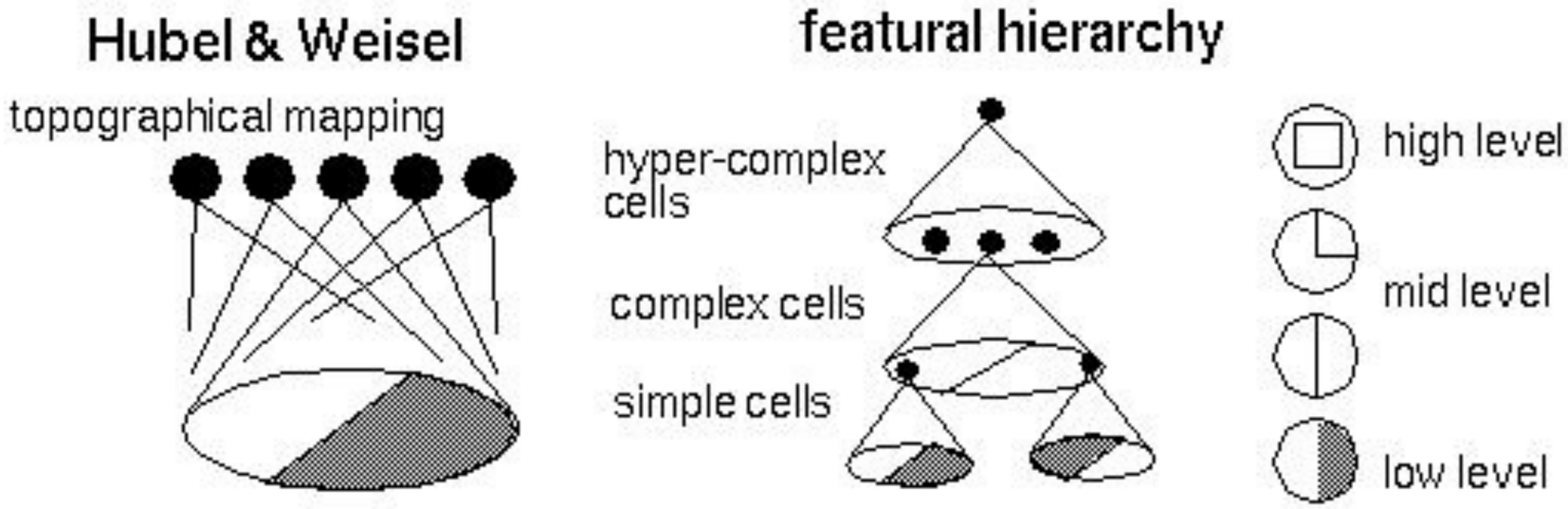


oriented filter

Source: Andrea Vedaldi

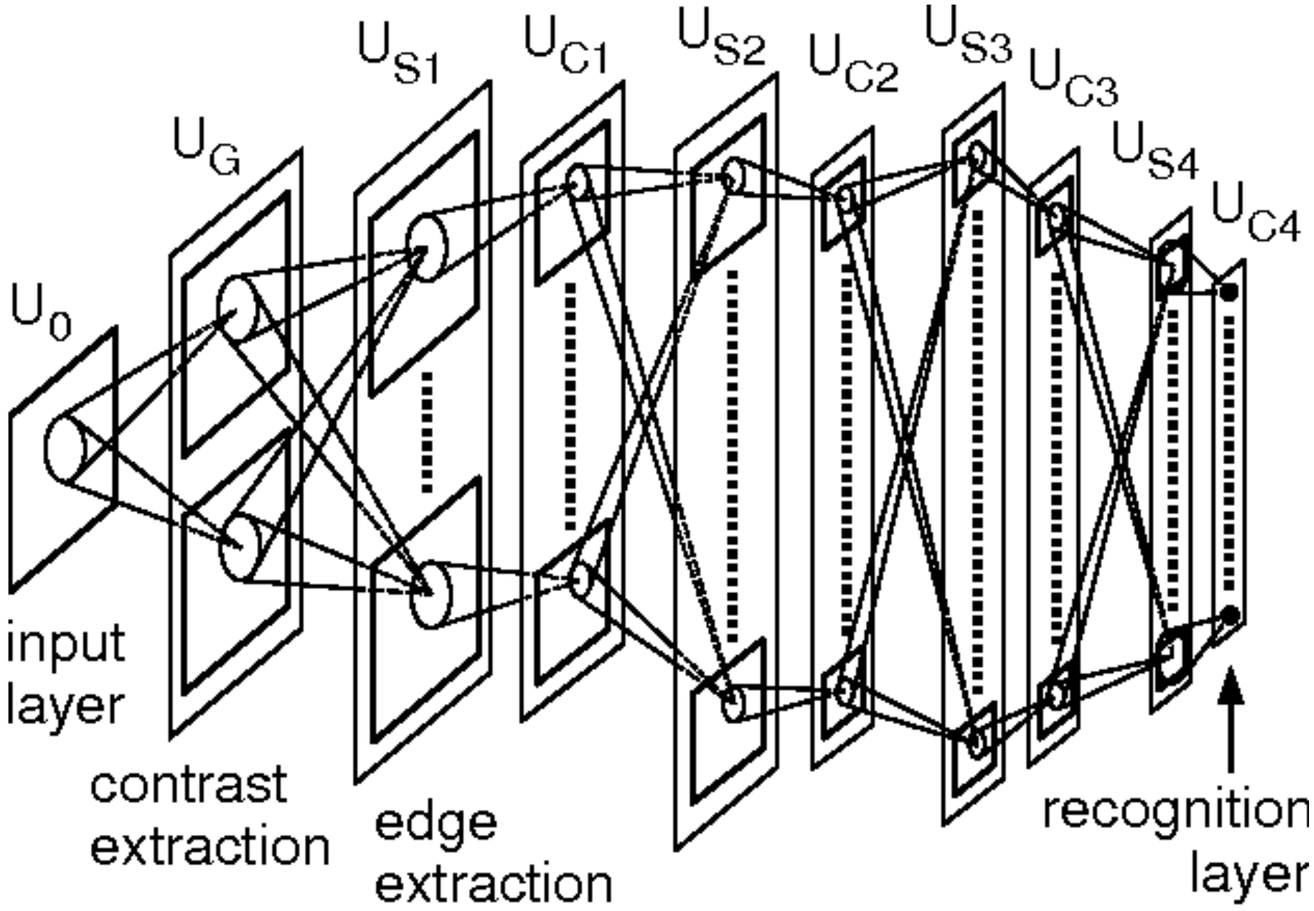
Biological visual system

- D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of ***simple***, ***complex***, and ***hyper-complex*** cells



Source: S. Lazebnik

History: Neocognitron

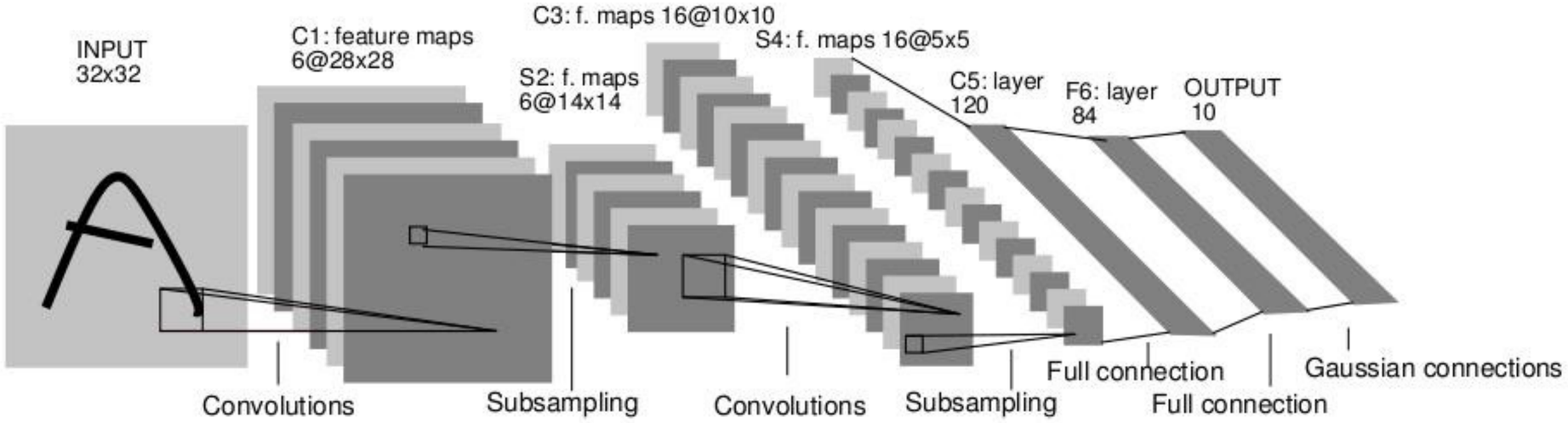


K. Fukushima, 1980s

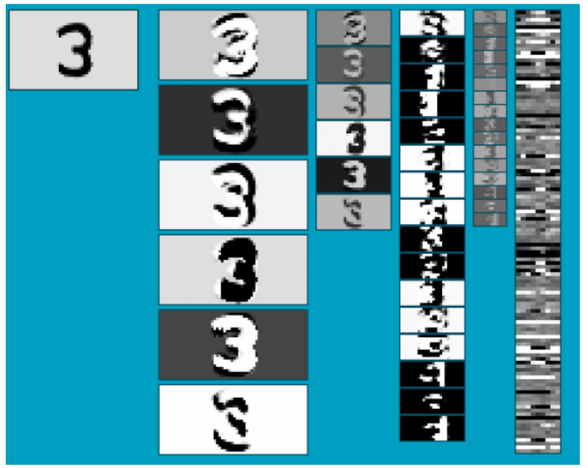
https://en.wikipedia.org/wiki/Neocognitron

Source: S. Lazebnik

History: LeNet-5



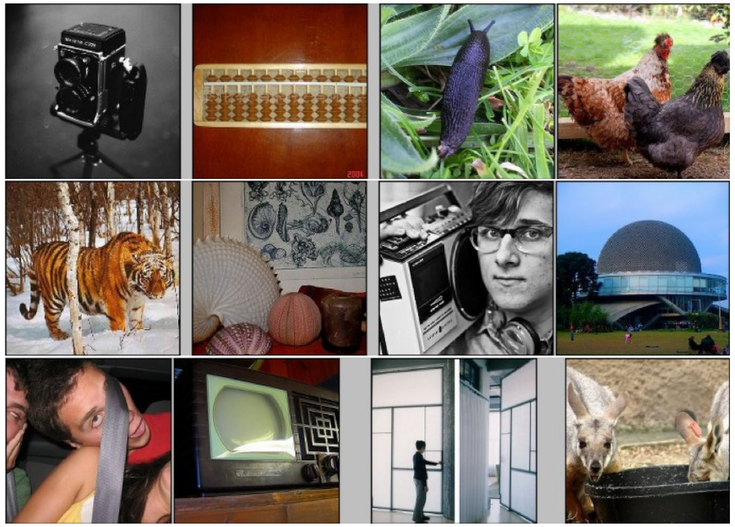
- Neocognitron + backpropagation
- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Source: S. Lazebnik

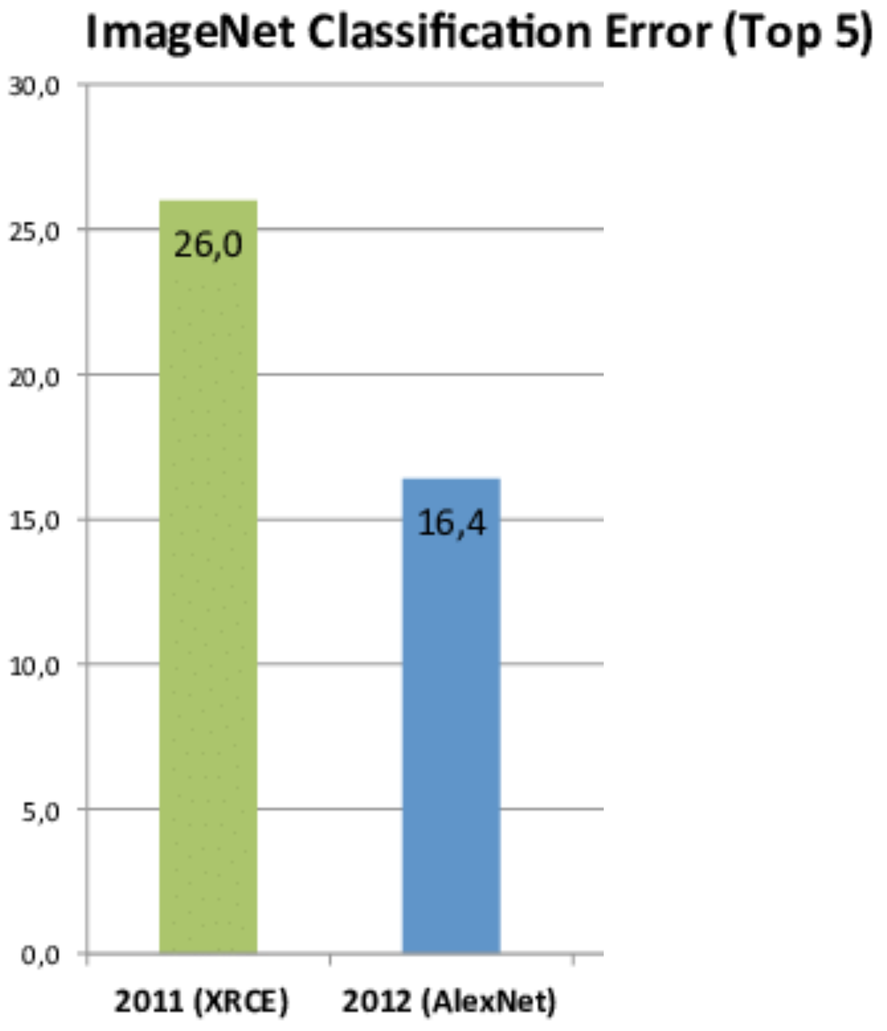
ImageNet Challenge



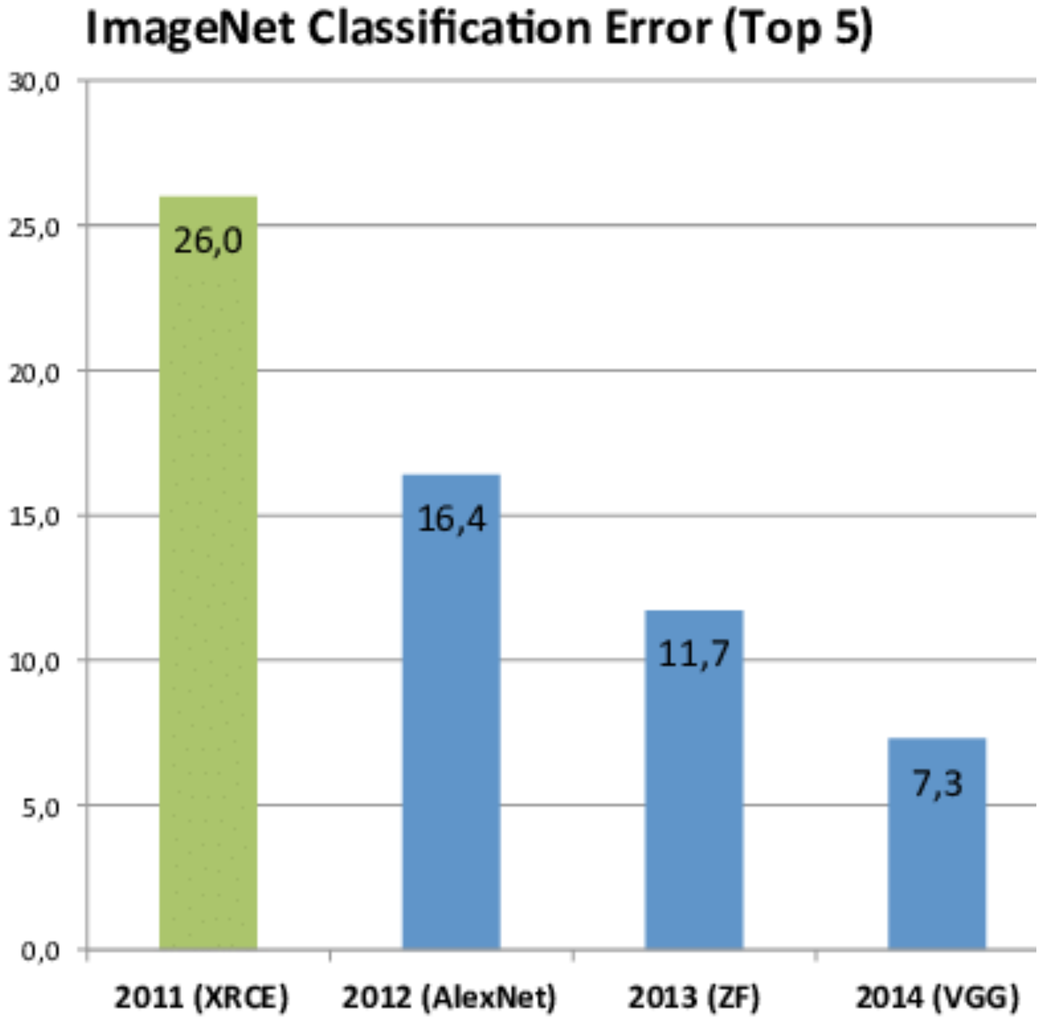
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/

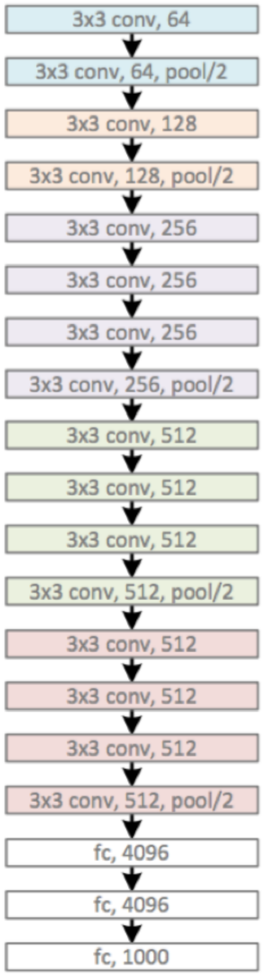
Source: Isola, Torralba, Freeman



Source: Isola, Torralba, Freeman



2014: VGG
16 conv. layers



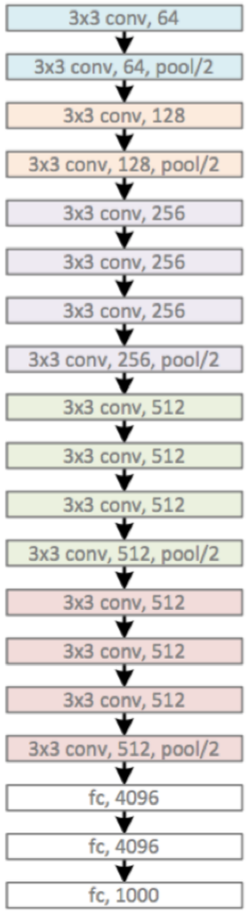
Error: 7.3%

[Simonyan & Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015]

Source: Isola, Torralba, Freeman

VGG-Net [Simonyan & Zisserman, 2015]

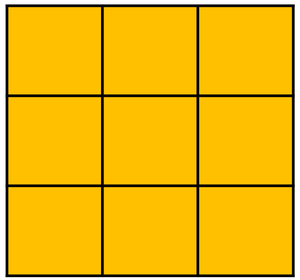
2014: VGG
16 conv. layers



Error: 7.3%

Main developments

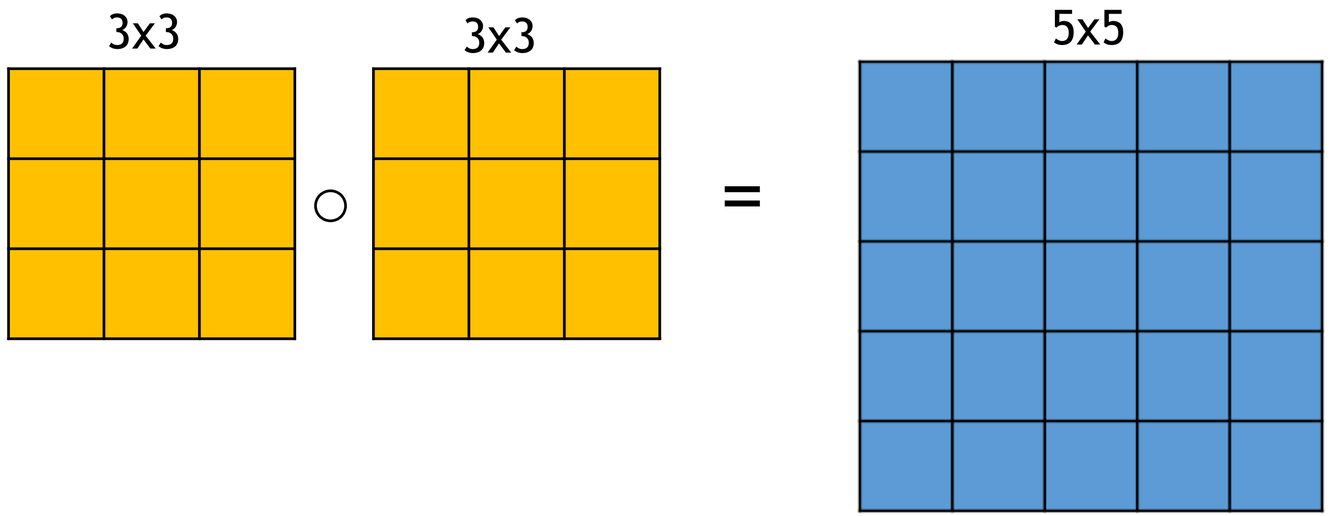
- Small convolutional kernels: only 3x3



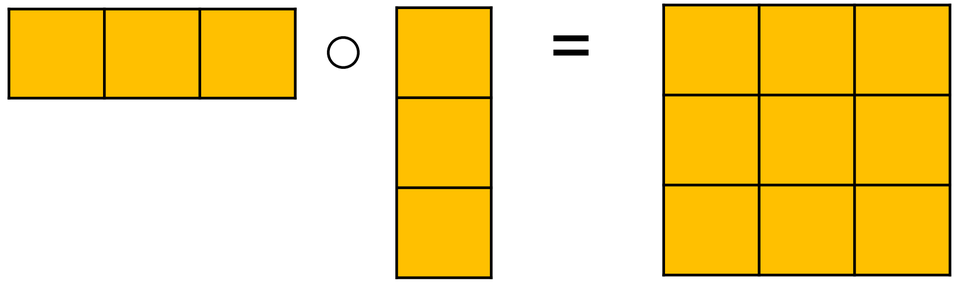
- Increased depth (5 -> 16/19 layers)

Source: Isola, Torralba, Freeman

Chaining convolutions



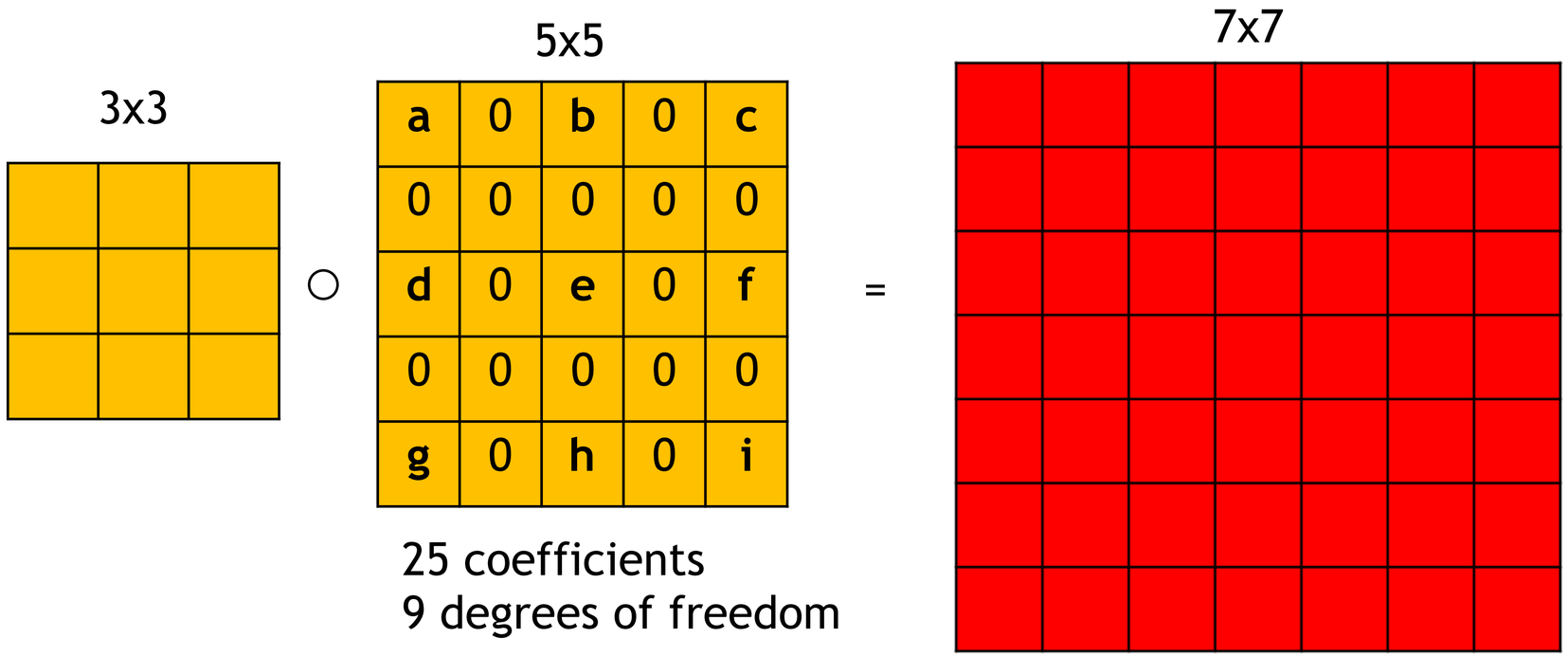
25 coefficients, but only
18 degrees of freedom



9 coefficients, but only
6 degrees of freedom.
Only separable filters… would this be enough?

Source: Isola, Torralba, Freeman

Dilated convolutions



What is lost?

[https://arxiv.org/pdf/1511.07122.pdf]

Source: Isola, Torralba, Freeman

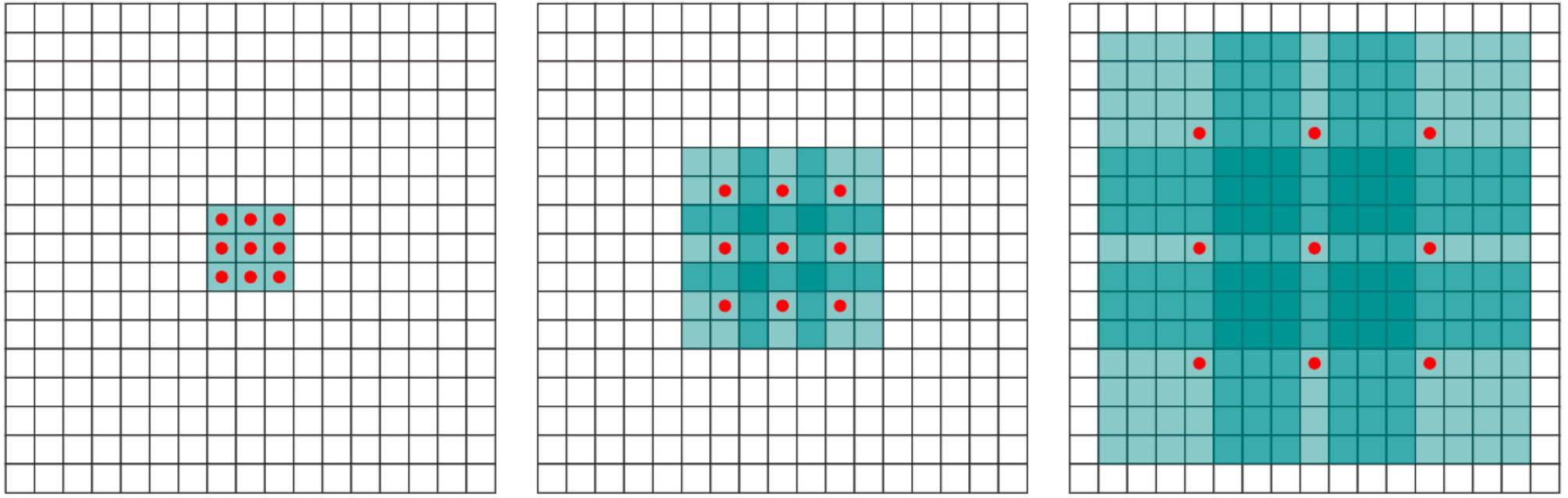
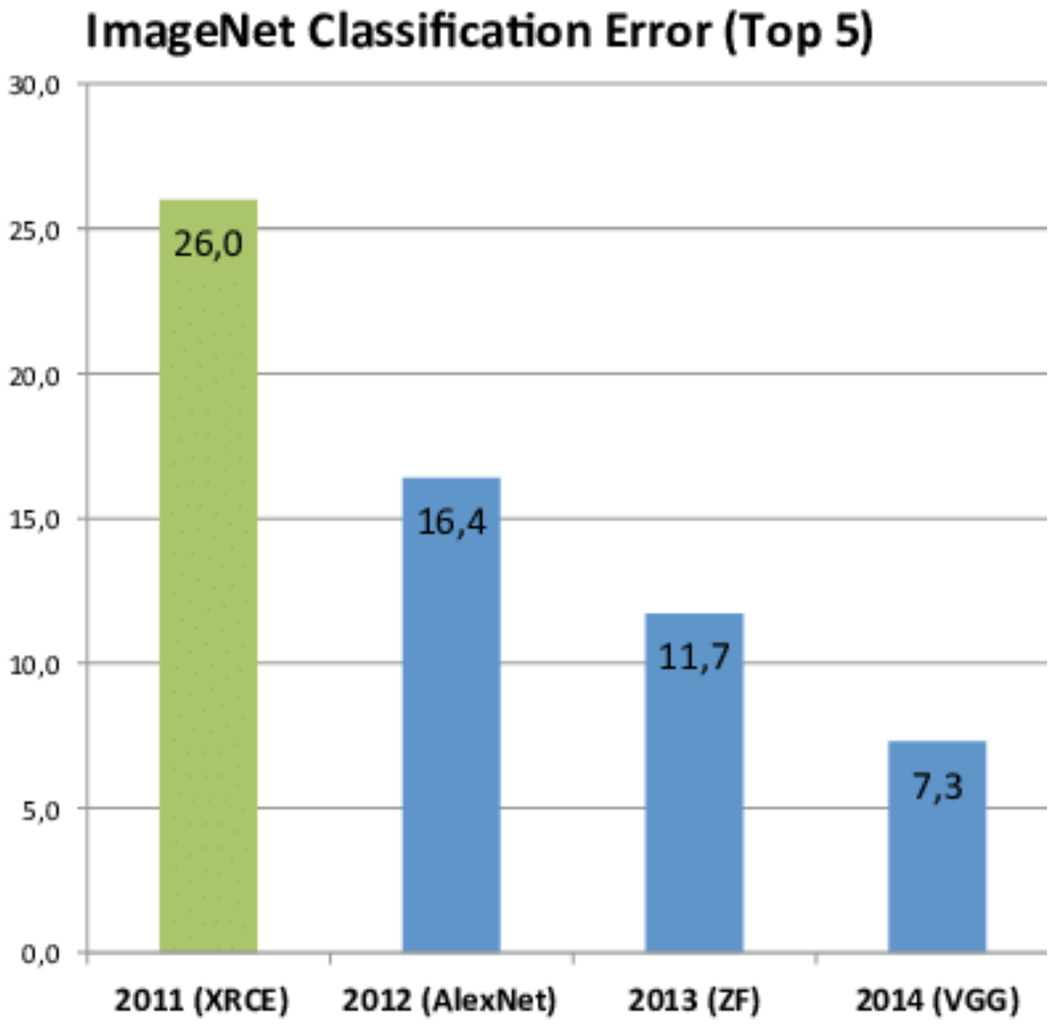


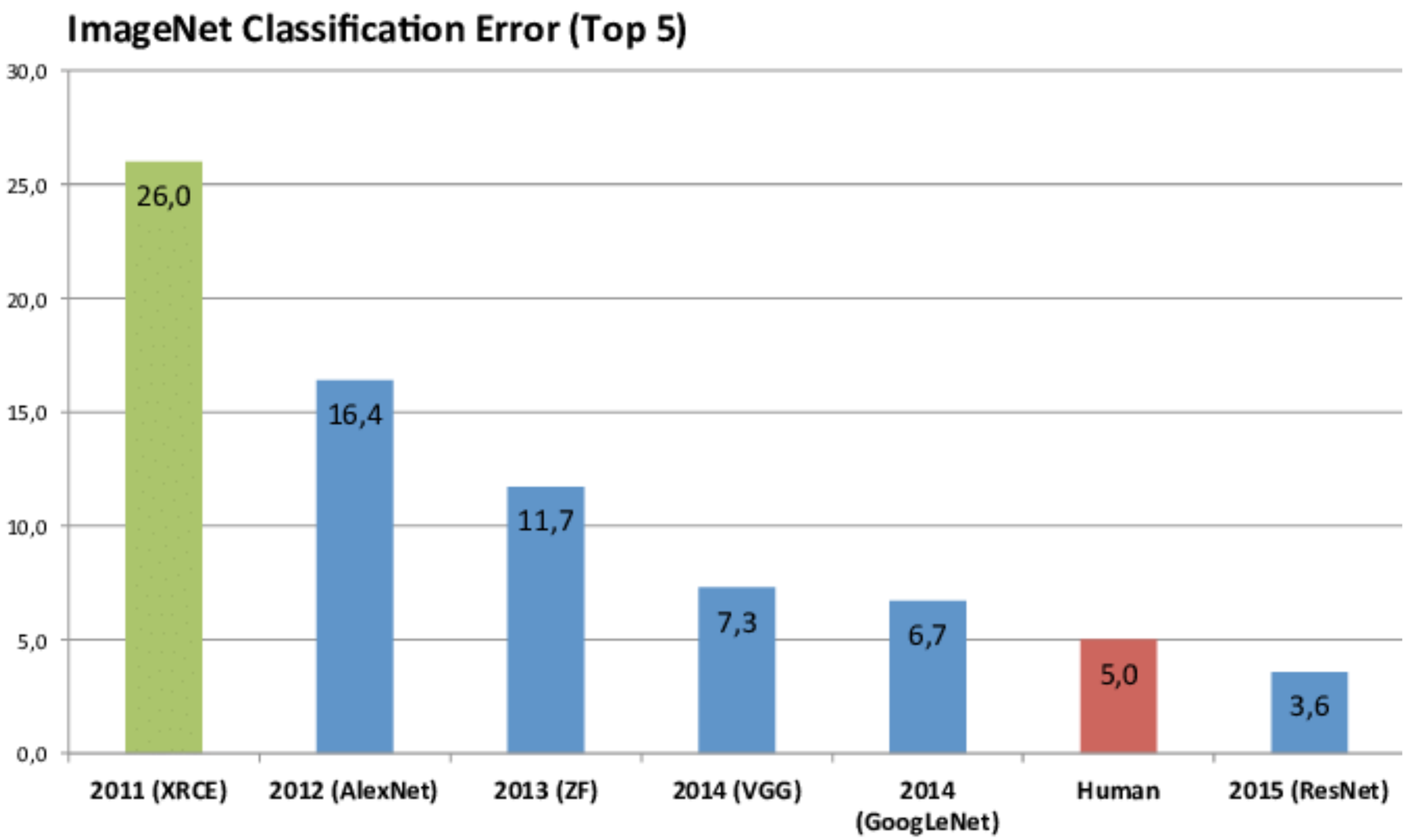
Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

[https://arxiv.org/pdf/1511.07122.pdf]

Source: Isola, Torralba, Freeman



Source: Isola, Torralba, Freeman



2016: ResNet
>100 conv. layers

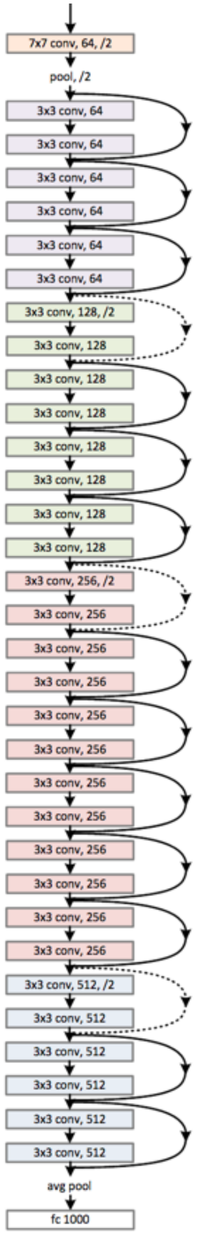
Error: 3.6%

[He et al: Deep Residual Learning for Image Recognition, CVPR 2016]

Source: Isola, Torralba, Freeman

2016: ResNet
>100 conv. layers

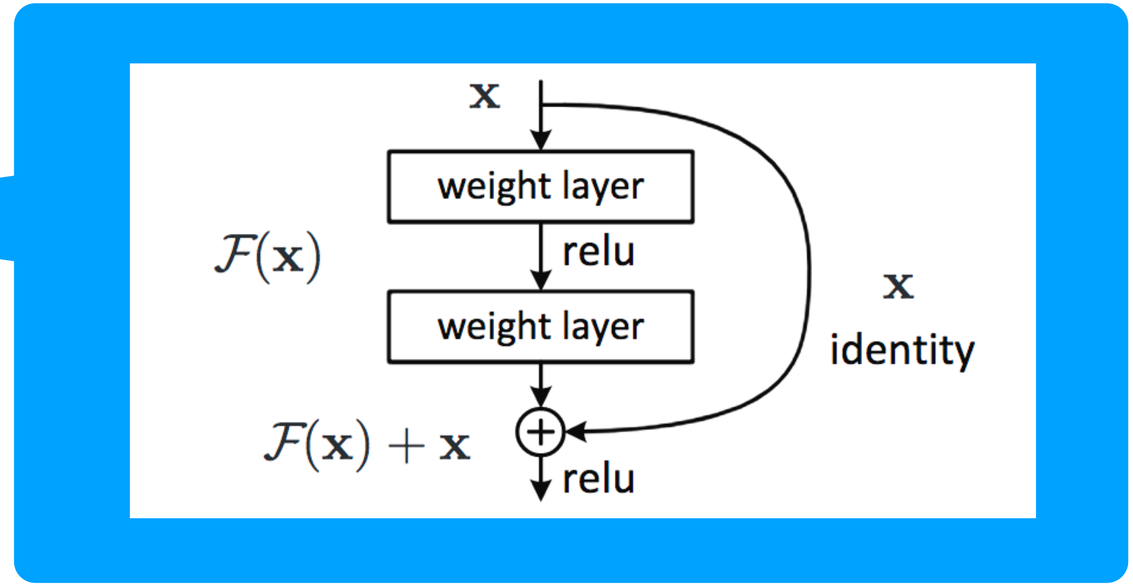
ResNet [He et al, 2016]



Error: 3.6%

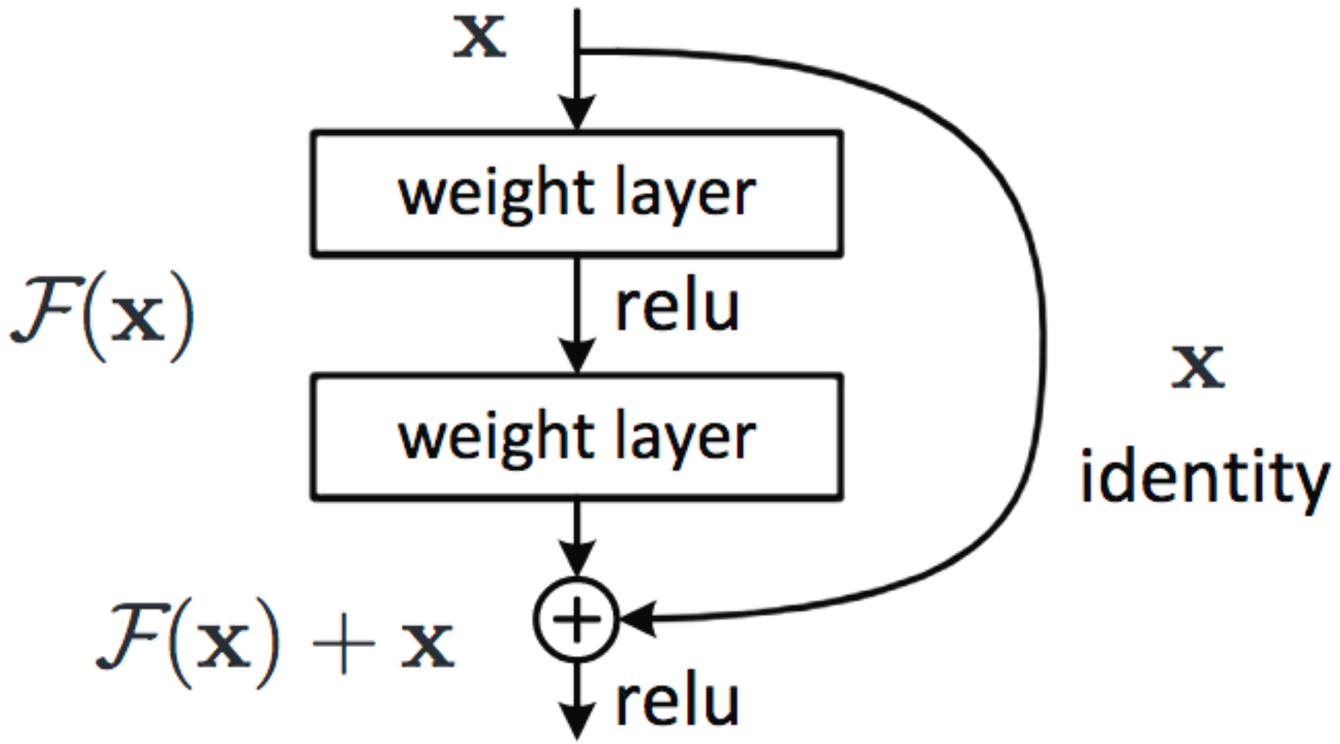
Main developments

- Increased depth possible through residual blocks



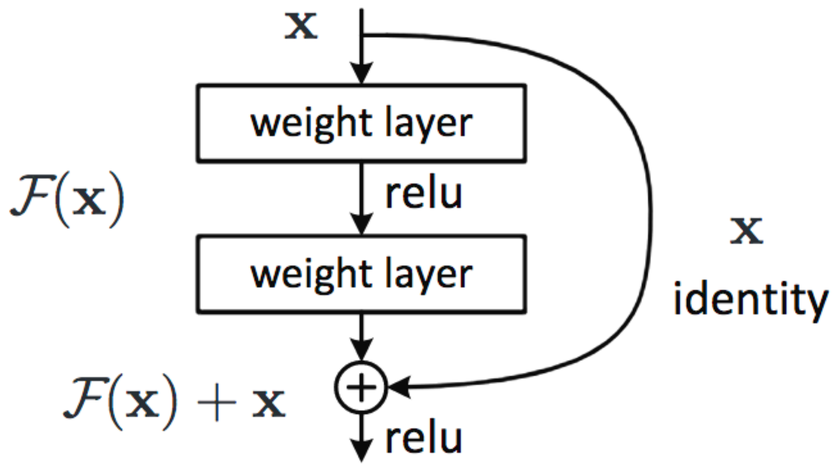
Source: Isola, Torralba, Freeman

Residual Blocks



Source: Isola, Torralba, Freeman

Residual Blocks

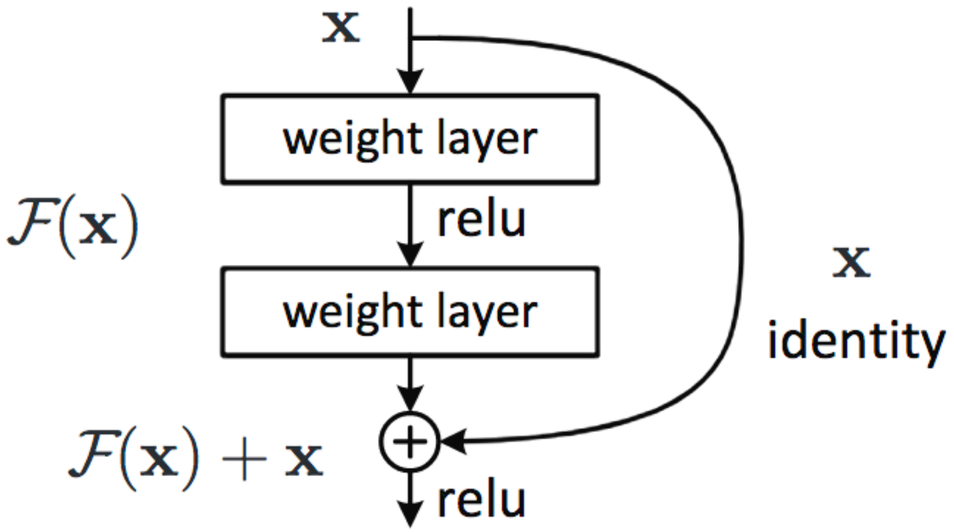


Why do they work?

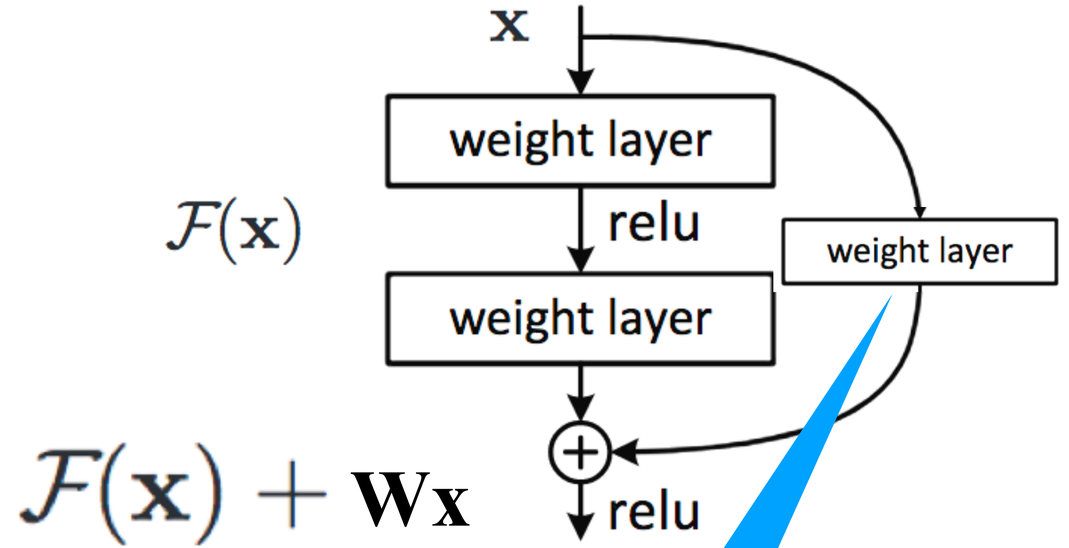
- Gradients can propagate faster (via the identity mapping)
- Within each block, only small residuals have to be learned

Source: Isola, Torralba, Freeman

If output has same size as input:



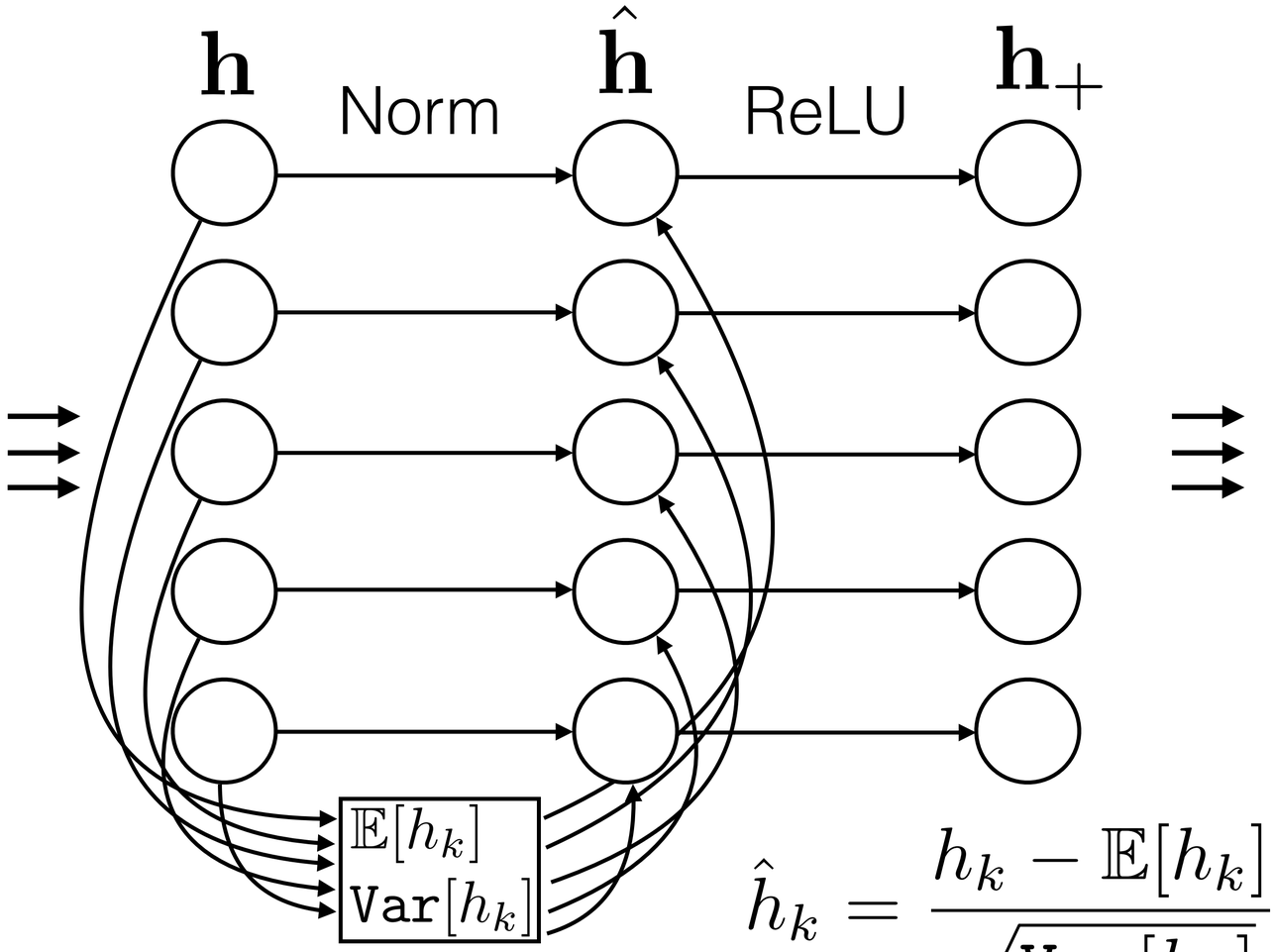
If output has a different size:



Projects into the right dimensionality:
dim(F(x)) = dim(Wx)

Source: Isola, Torralba, Freeman

Normalization layers



$$\hat{h}_k = \frac{h_k - \mathbb{E}[h_k]}{\sqrt{\mathtt{Var}[h_k]}}$$

Source: Isola, Torralba, Freeman

Normalization layers

Keep track of mean and variance of a unit (or a population of units) over time.

Standardize unit activations by subtracting mean and dividing by variance.

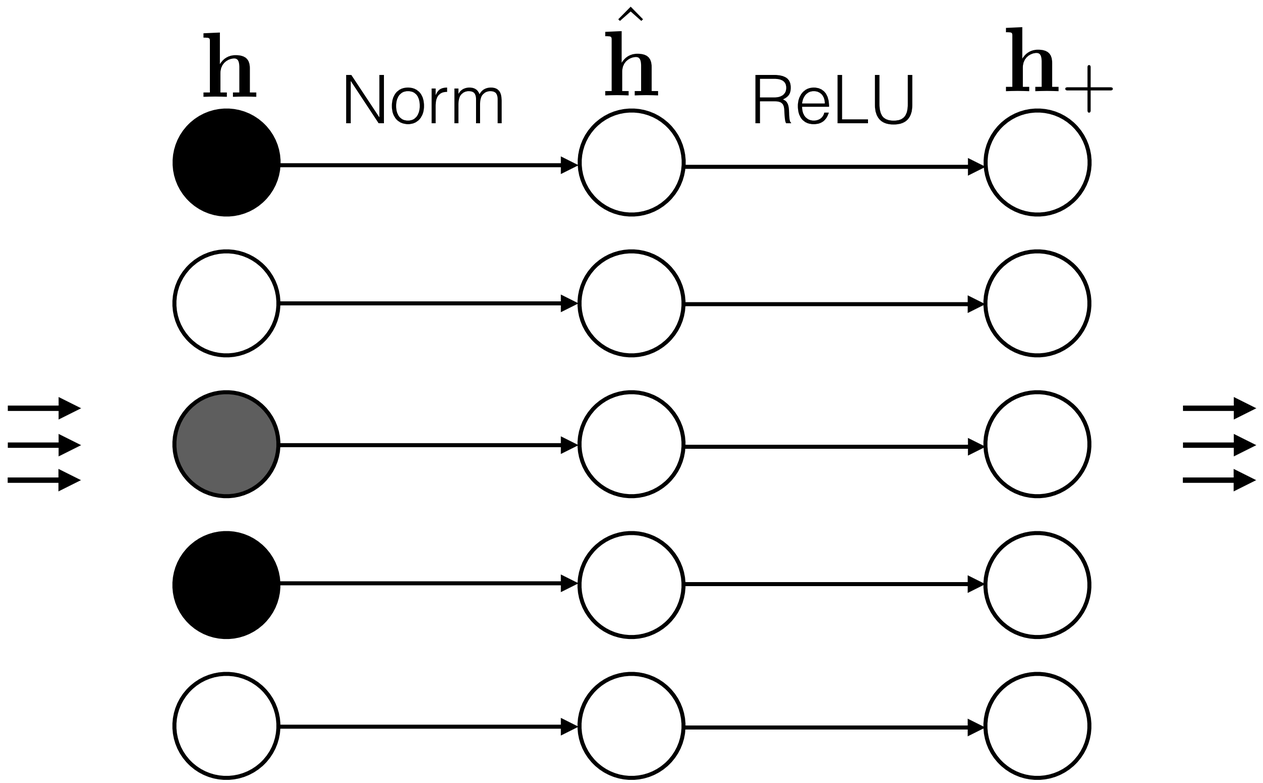
Squashes units into a **standard range**, avoiding overflow.

Also achieves **invariance** to mean and variance of the training signal.

Both these properties reduce the effective capacity of the model, i.e. regularize the model.

Source: Isola, Torralba, Freeman

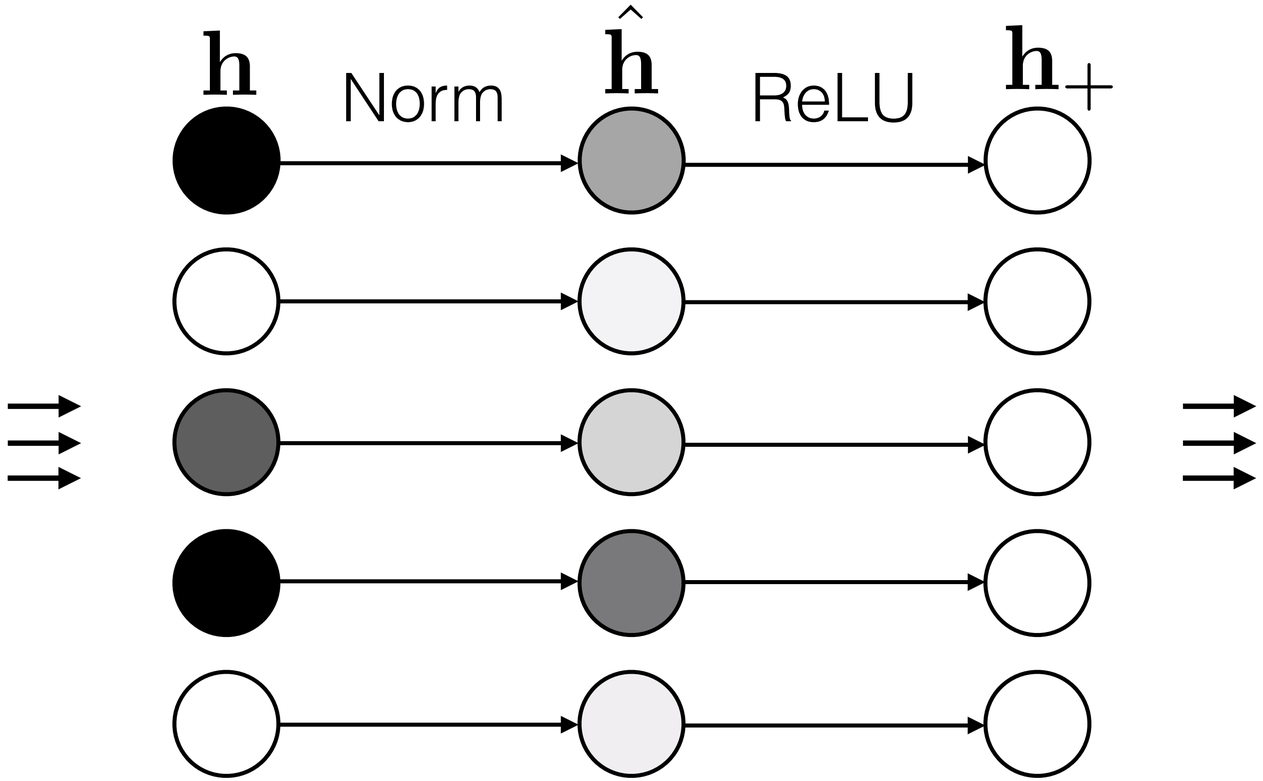
Normalization layers



$$\hat{h}_k = \frac{h_k - \mathbb{E}[h_k]}{\sqrt{\texttt{Var}[h_k]}}$$

Source: Isola, Torralba, Freeman

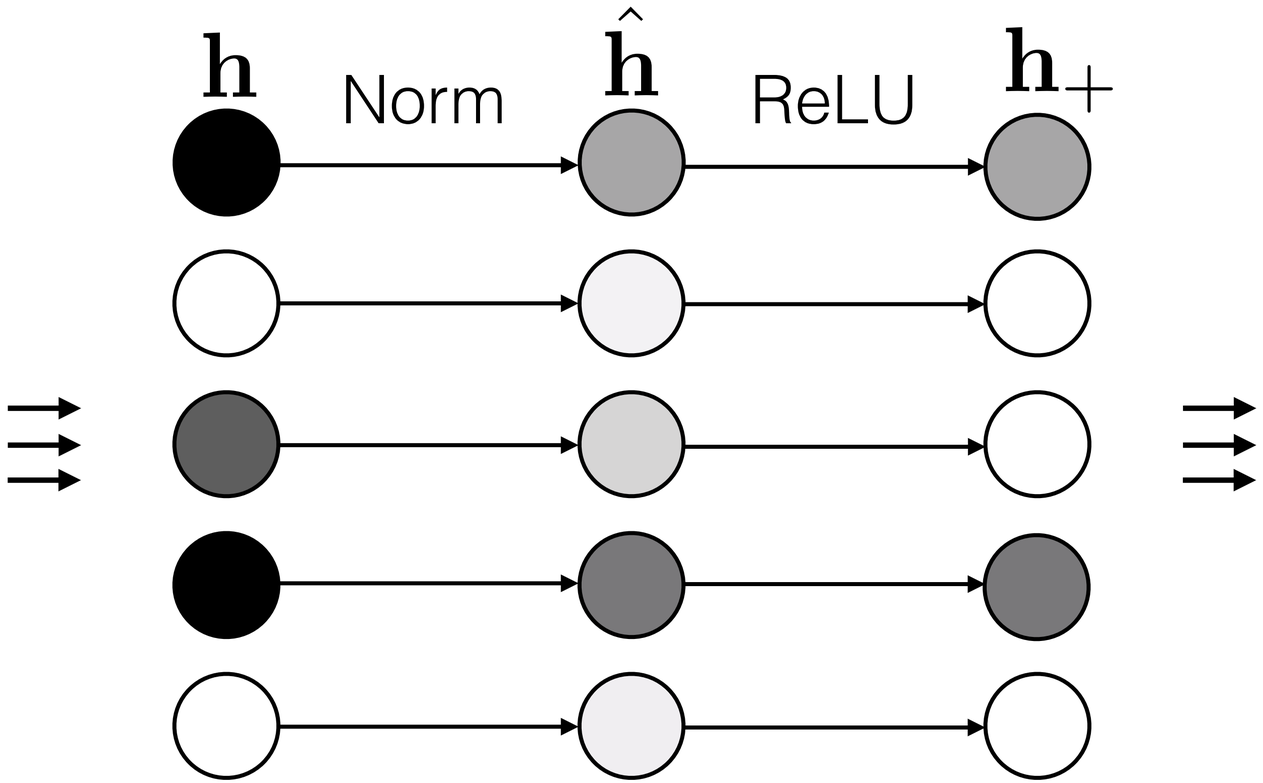
Normalization layers



$$\hat{h}_k = \frac{h_k - \mathbb{E}[h_k]}{\sqrt{\texttt{Var}[h_k]}}$$

Source: Isola, Torralba, Freeman

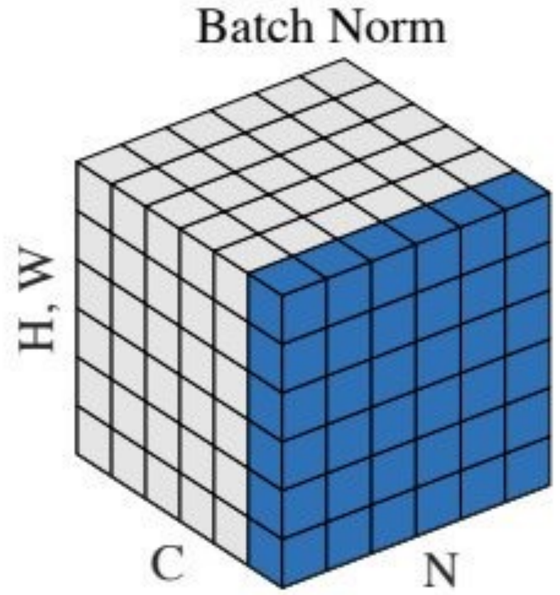
Normalization layers



$$\hat{h}_k = \frac{h_k - \mathbb{E}[h_k]}{\sqrt{\texttt{Var}[h_k]}}$$

Source: Isola, Torralba, Freeman

Batch normalization



Filter value in a CNN layer

Average and standard deviation each filter response, estimated over whole batch.

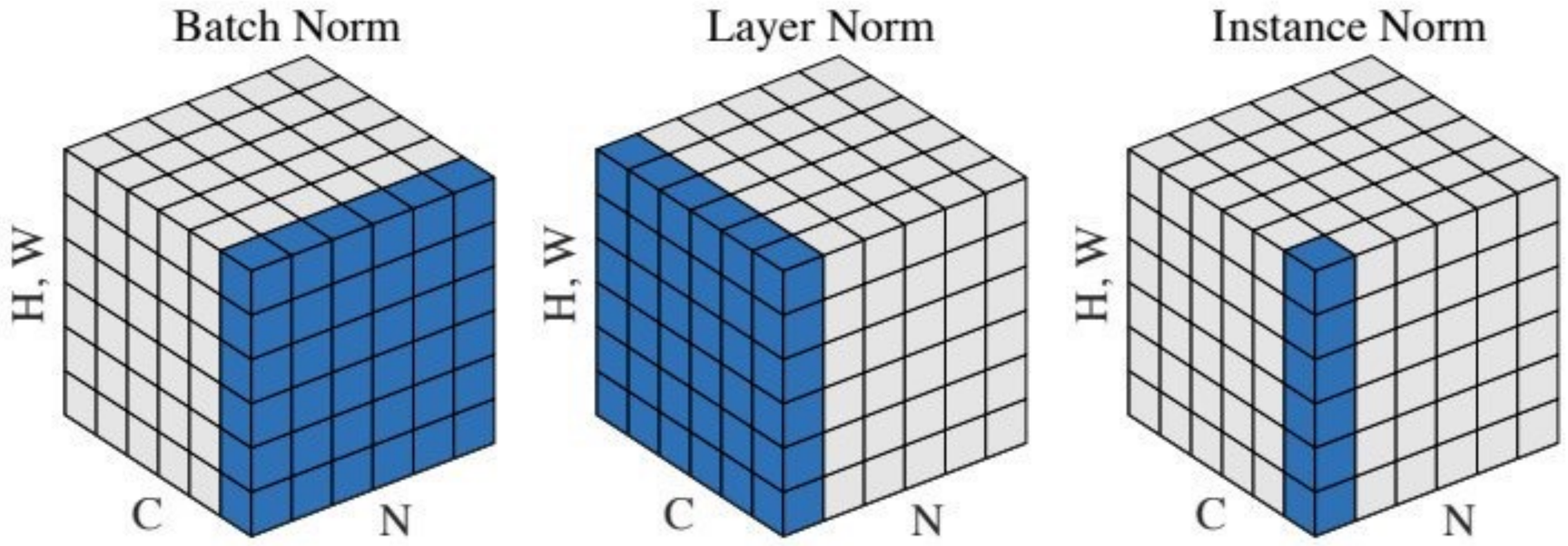
$$\hat{h}_k = \frac{h_k - \mathbb{E}[h_k]}{\sqrt{\texttt{Var}[h_k]}}$$

Normalize w.r.t. a single hidden unit's pattern of activation over training examples (a batch of examples).

[Figure from Wu & He, arXiv 2018] [Ioffe & Szegedy, 2015]

Source: Isola, Torralba, Freeman

Normalization layers

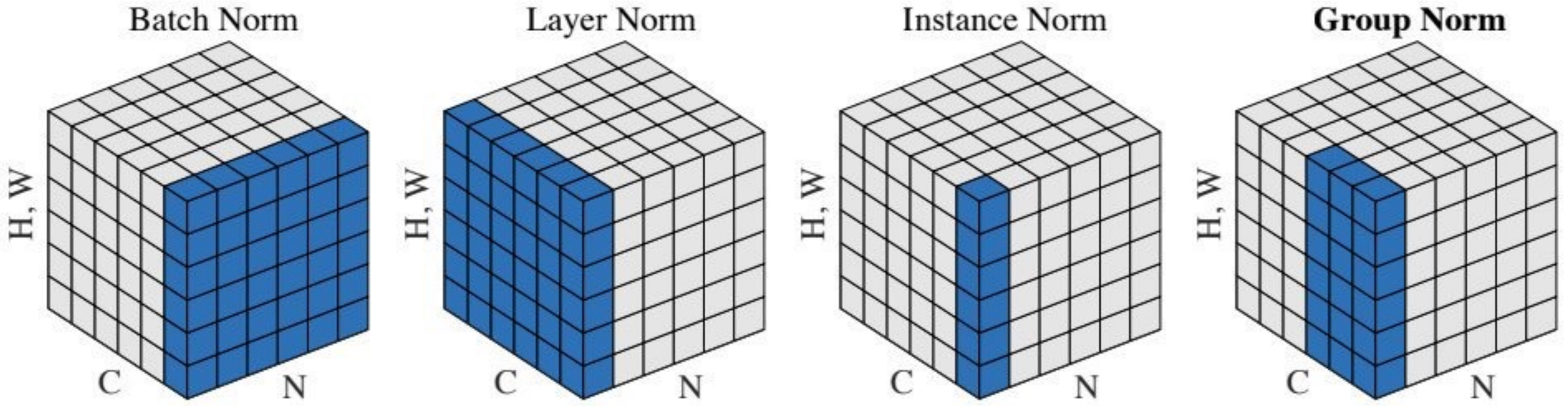


Normalize w.r.t. the mean and variance of the activations of all the hidden units (neurons) on this layer (c) that process this particular location (h,w) in the image.

[Figure from Wu & He, ECCV 2018]

Source: Isola, Torralba, Freeman

Normalization layers



All used in CNNs, especially when batch size is small.

[Figure from Wu & He, ECCV 2018]

Source: Isola, Torralba, Freeman