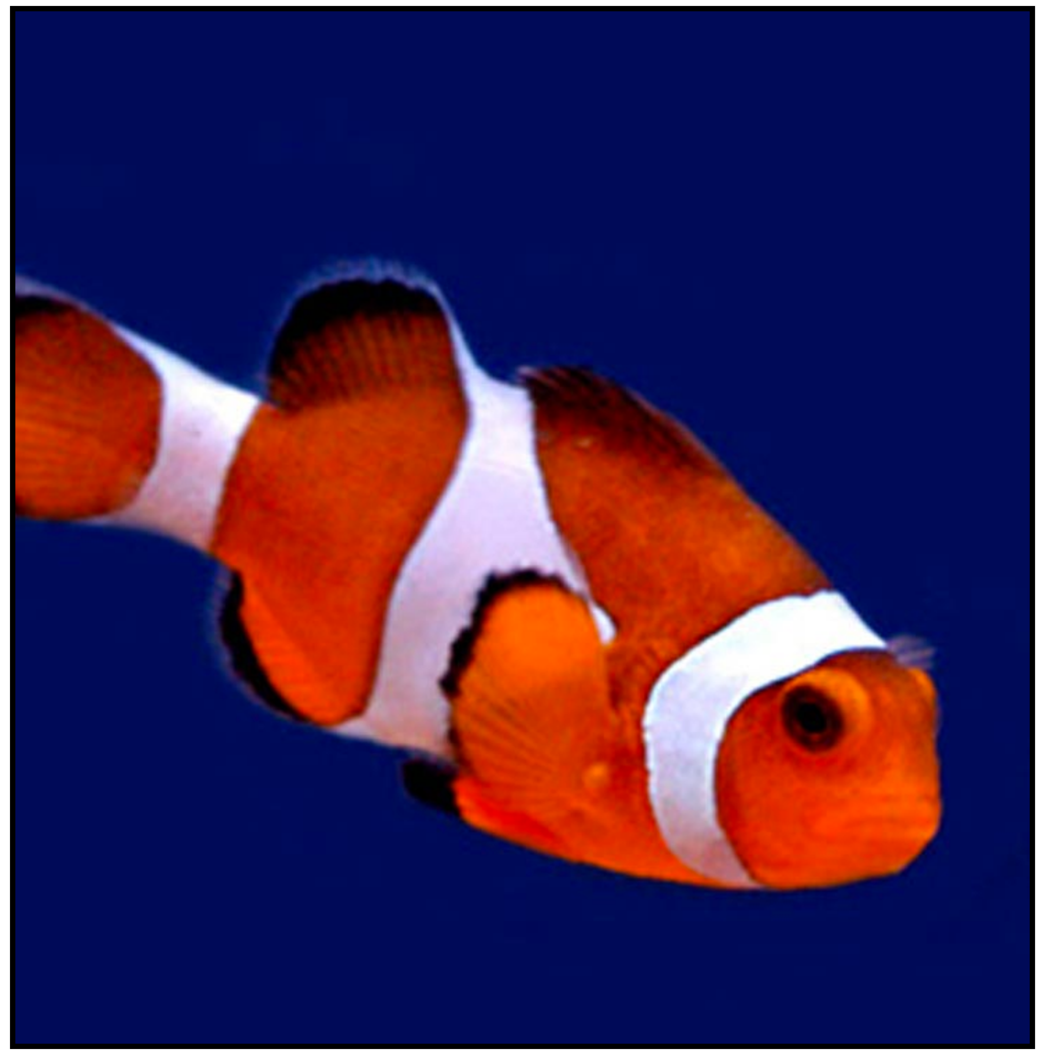
Lecture 9: Convolutional networks

Announcements

- PS3 due tonight, PS4 out tonight
- Next week: guest lectures on image generation by Daniel Geng, Sarah Jabbour, and Yiming Dou

Image classification



Classifier

"Fish"

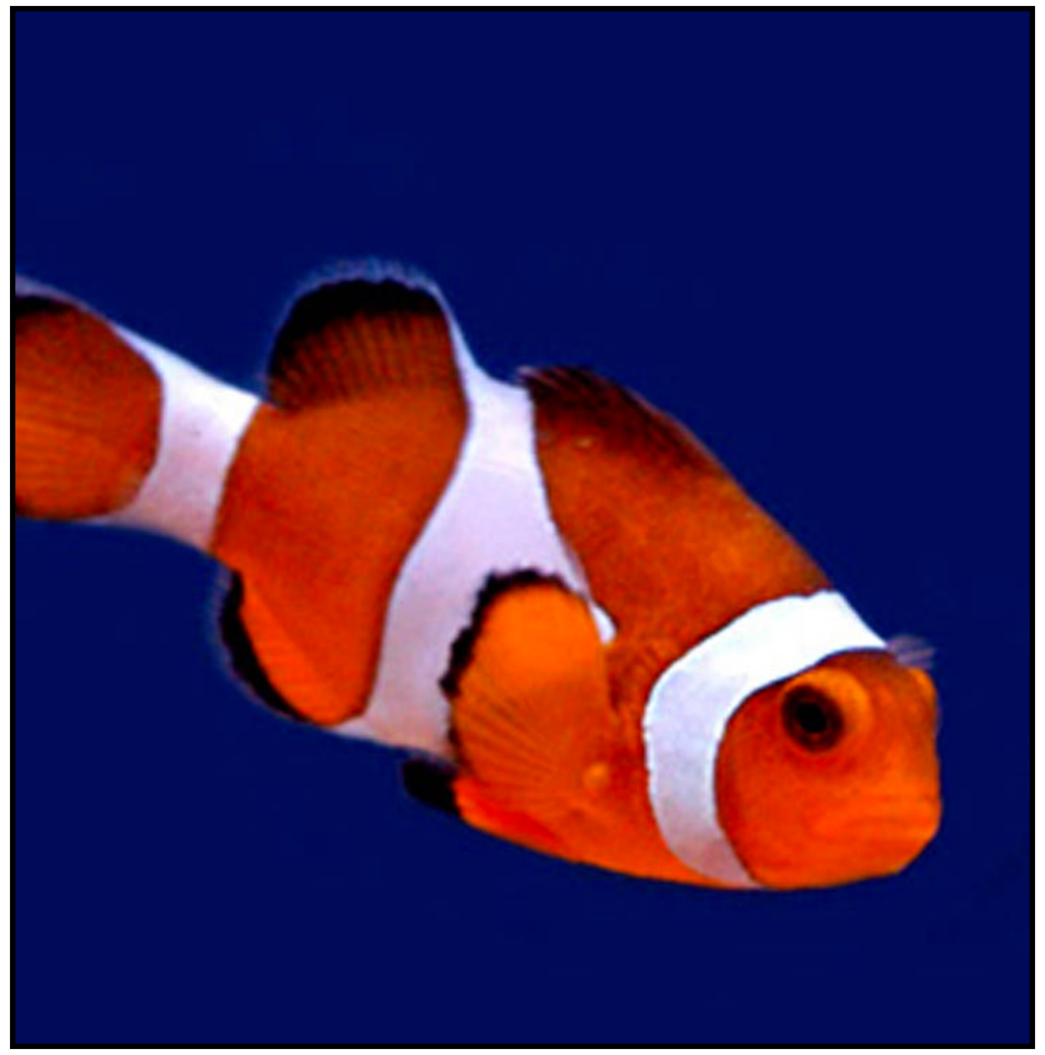
image **x**

label y

Source: Isola, Torralba, Freeman

Image classification

What should these be?



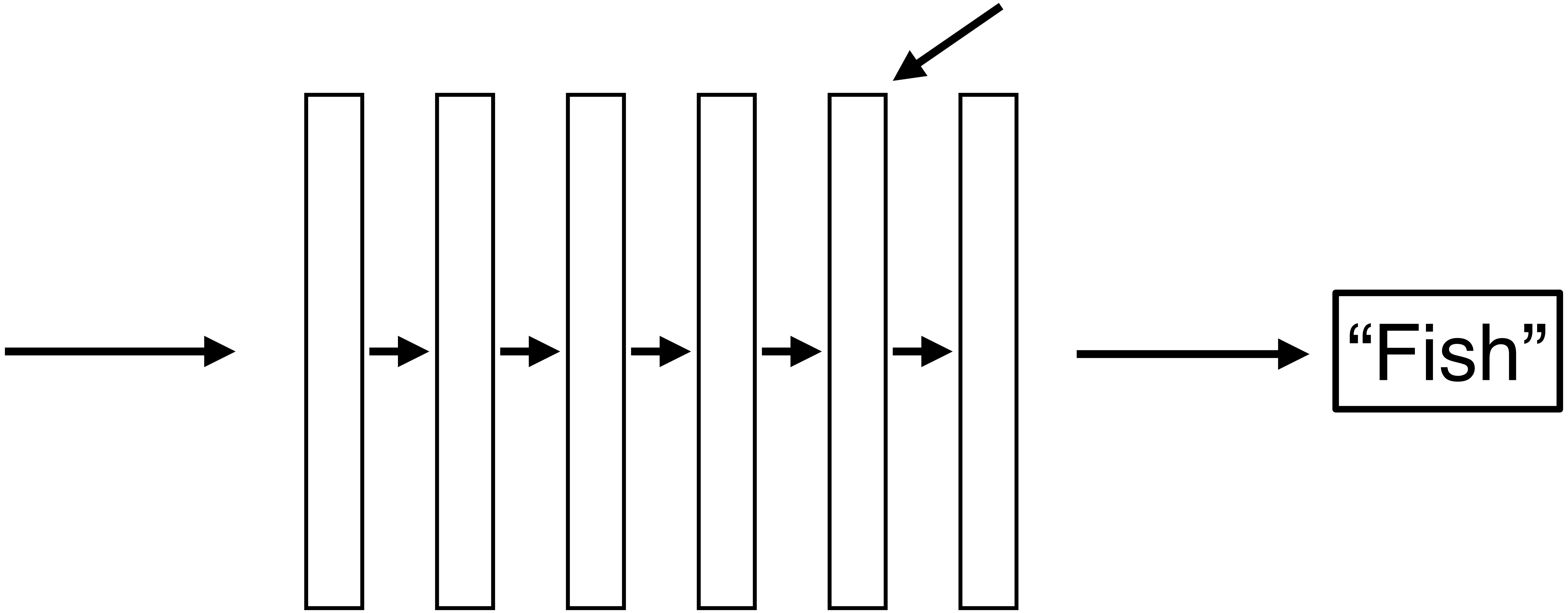


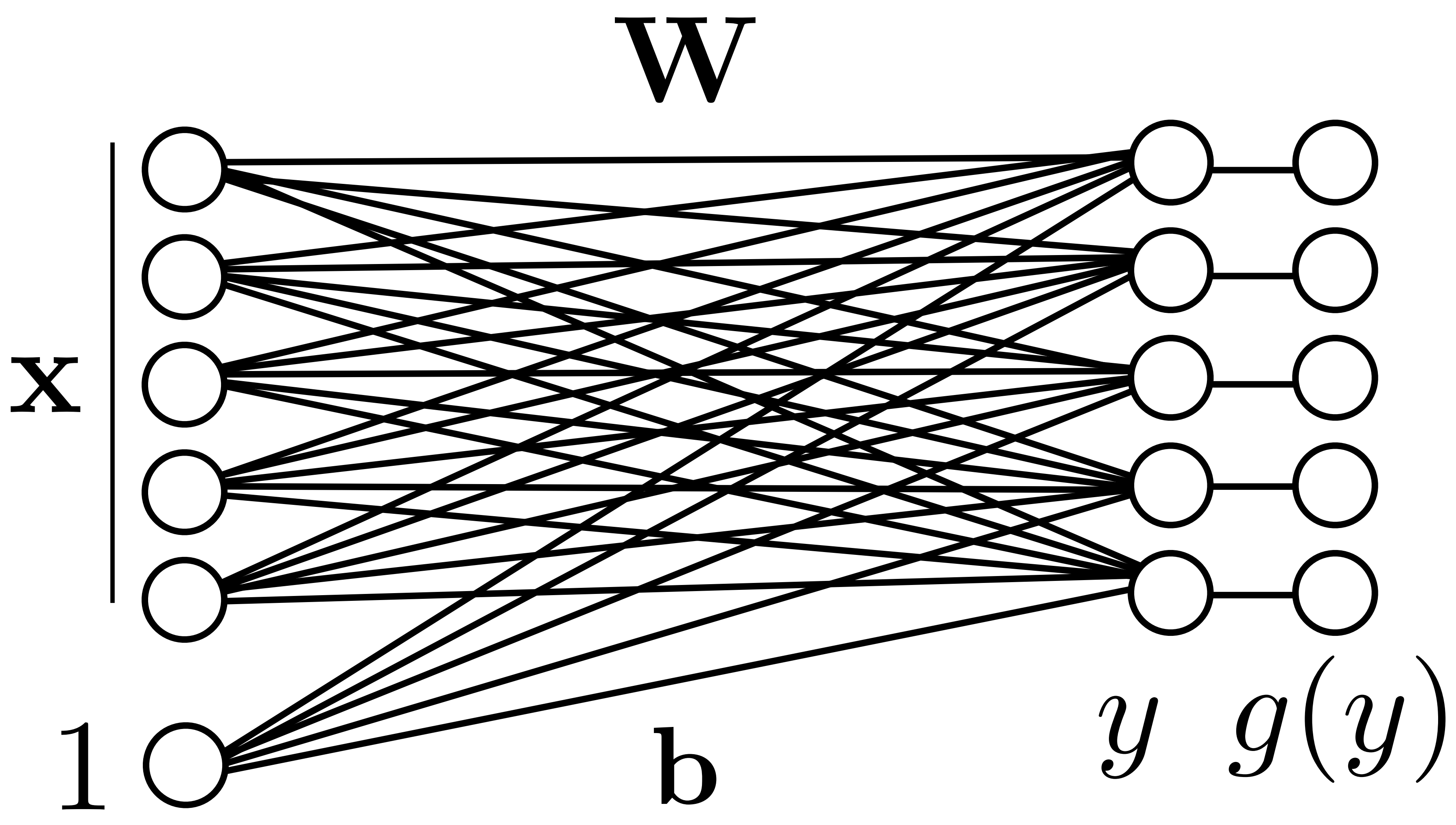
image **x**

label y

Source: Isola, Torralba, Freeman

Idea #1: Fully-connected network

Fully-connected (a.k.a. linear) layer

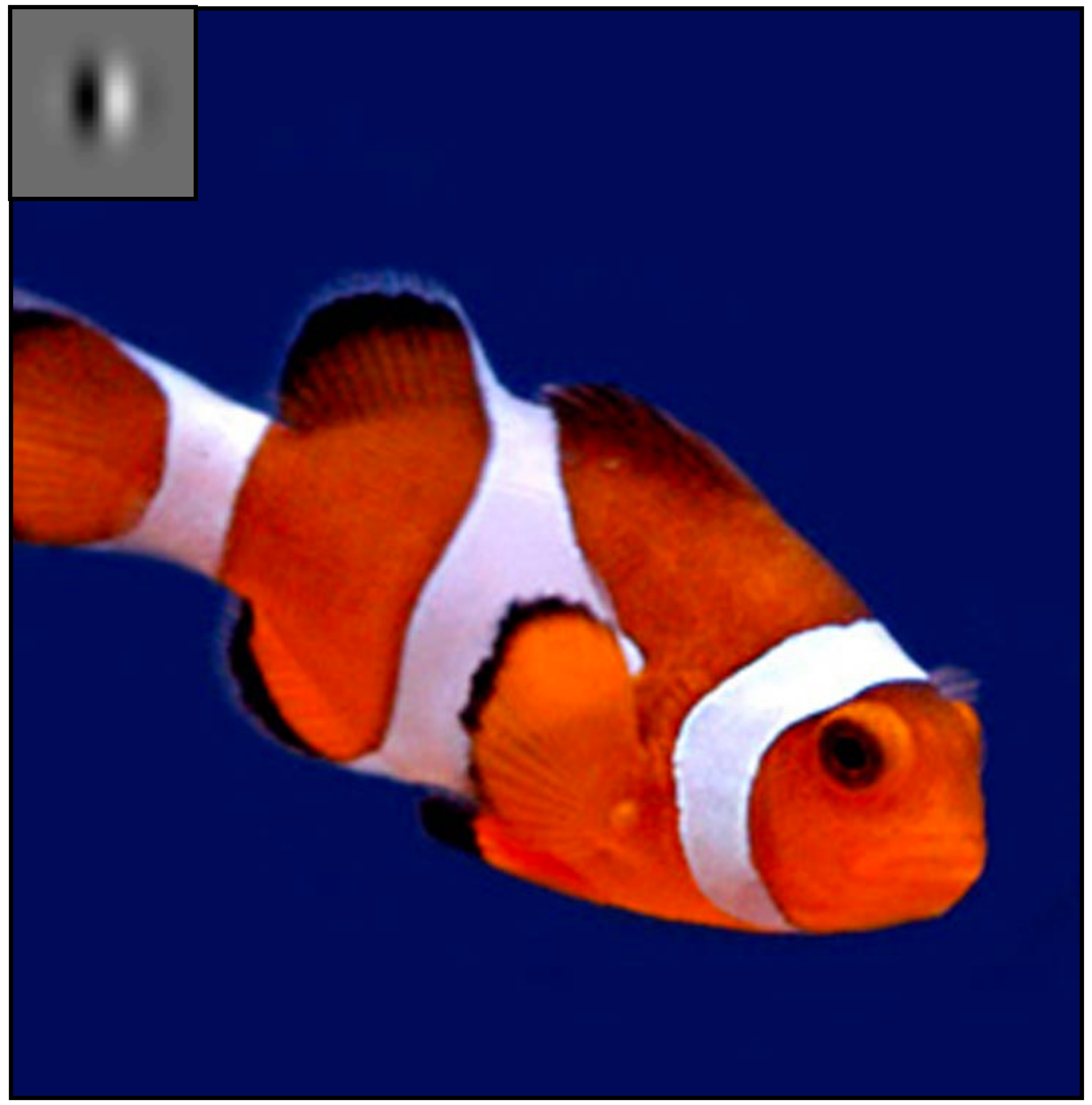


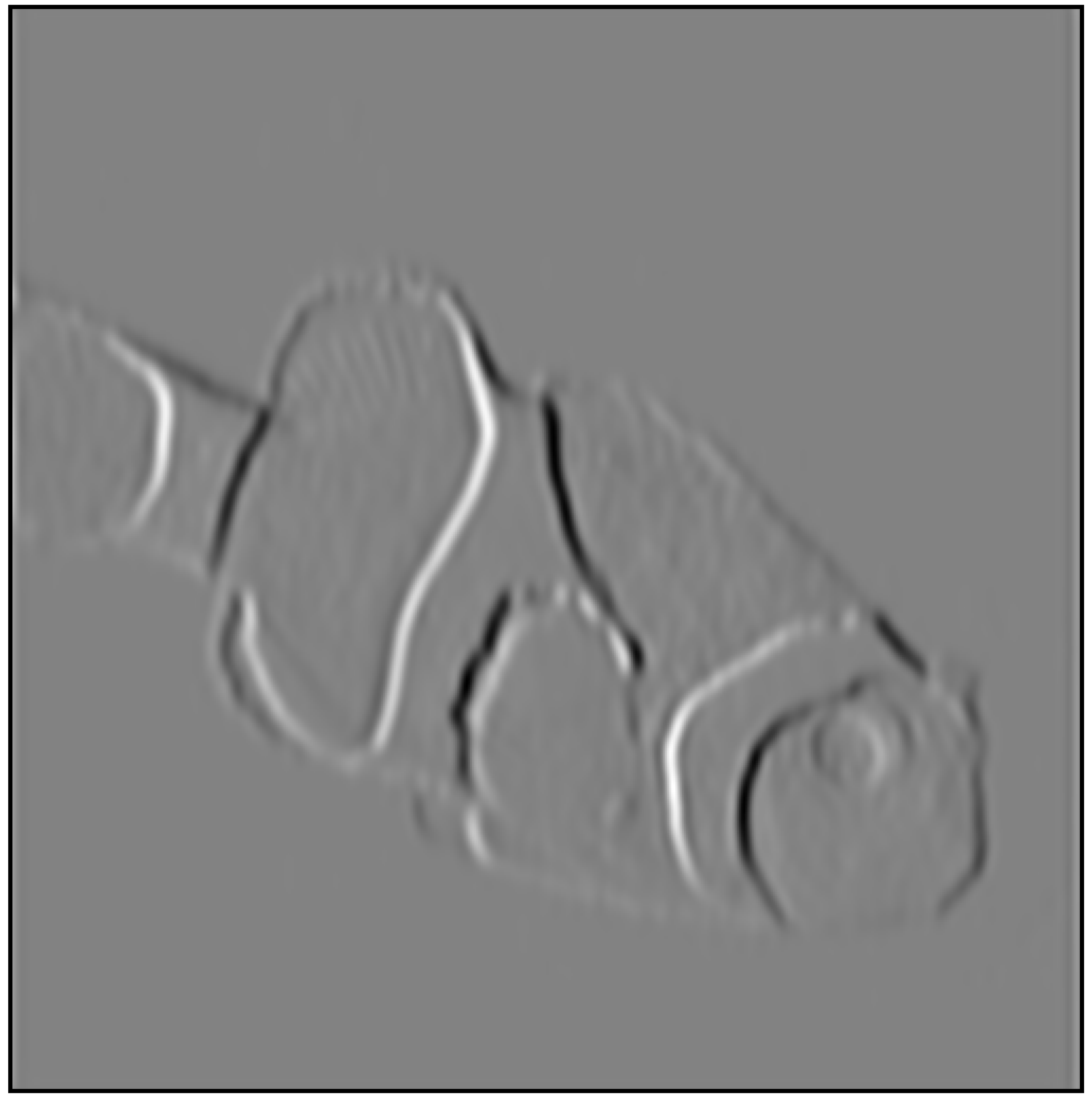
$\mathbf{x} =$

But $\mathbf{x}$ is really big!

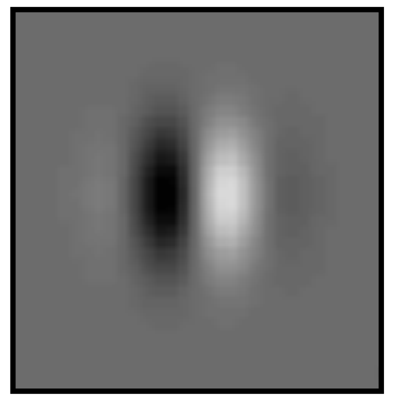
Say, 256 x 256 x 3 = 197k

Can we use convolution in a neural network?



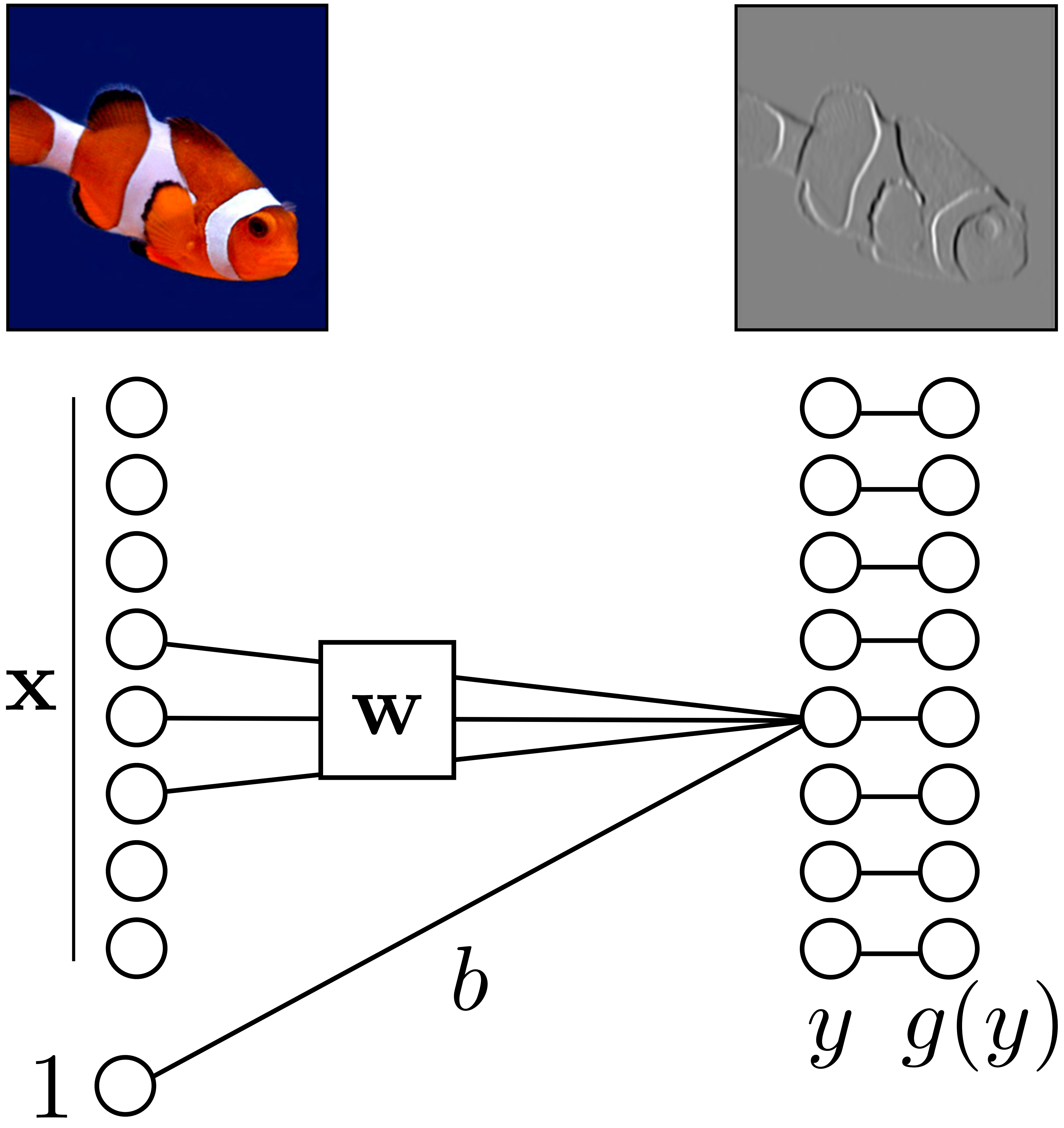


filter



Source: Isola, Torralba, Freeman

Recall: Sparsely connected network

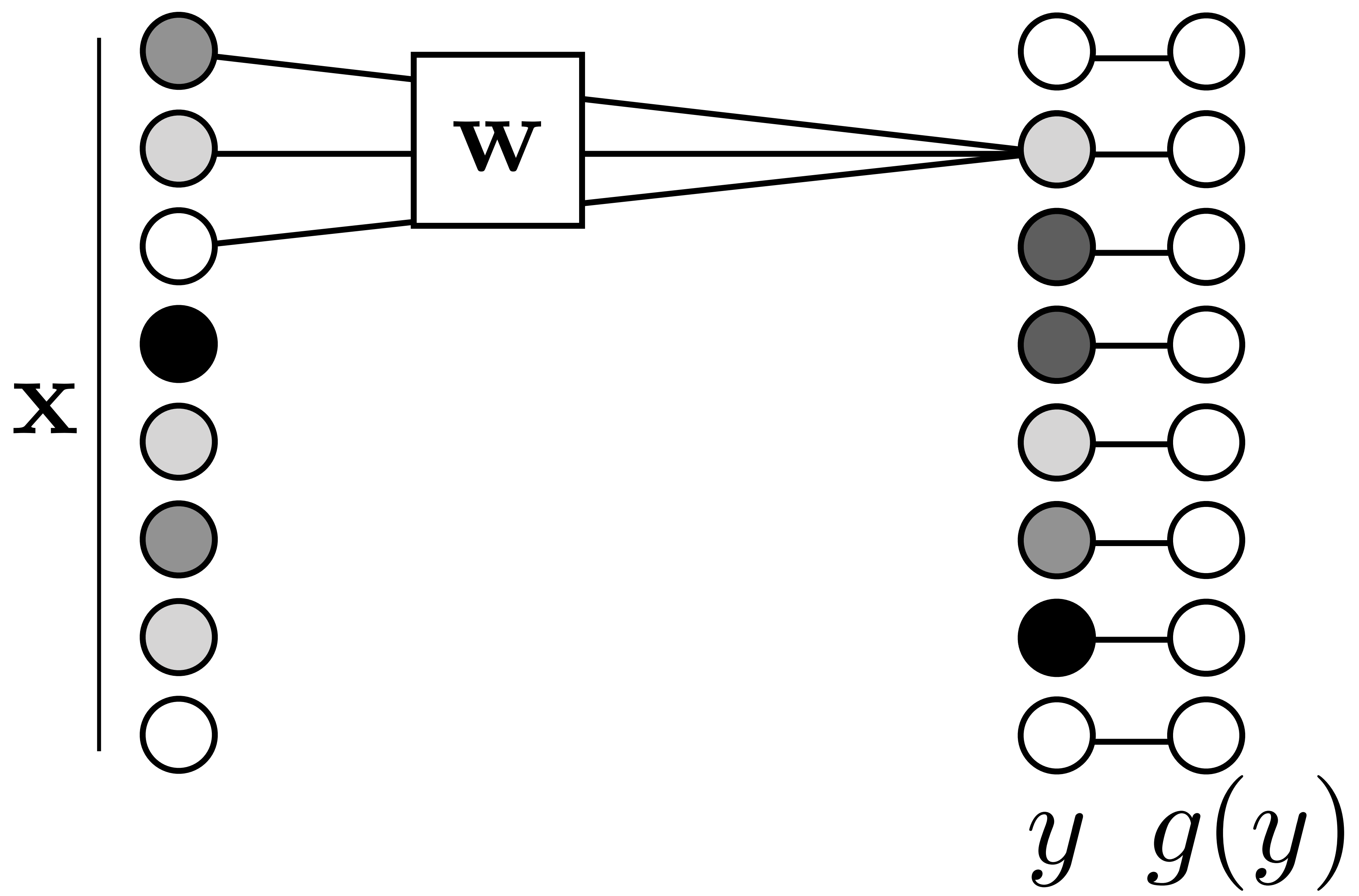


Each unit is connected to a subset of the units in the previous layer.

Source: Isola, Torralba, Freeman

Convolutional neural network

Conv layer

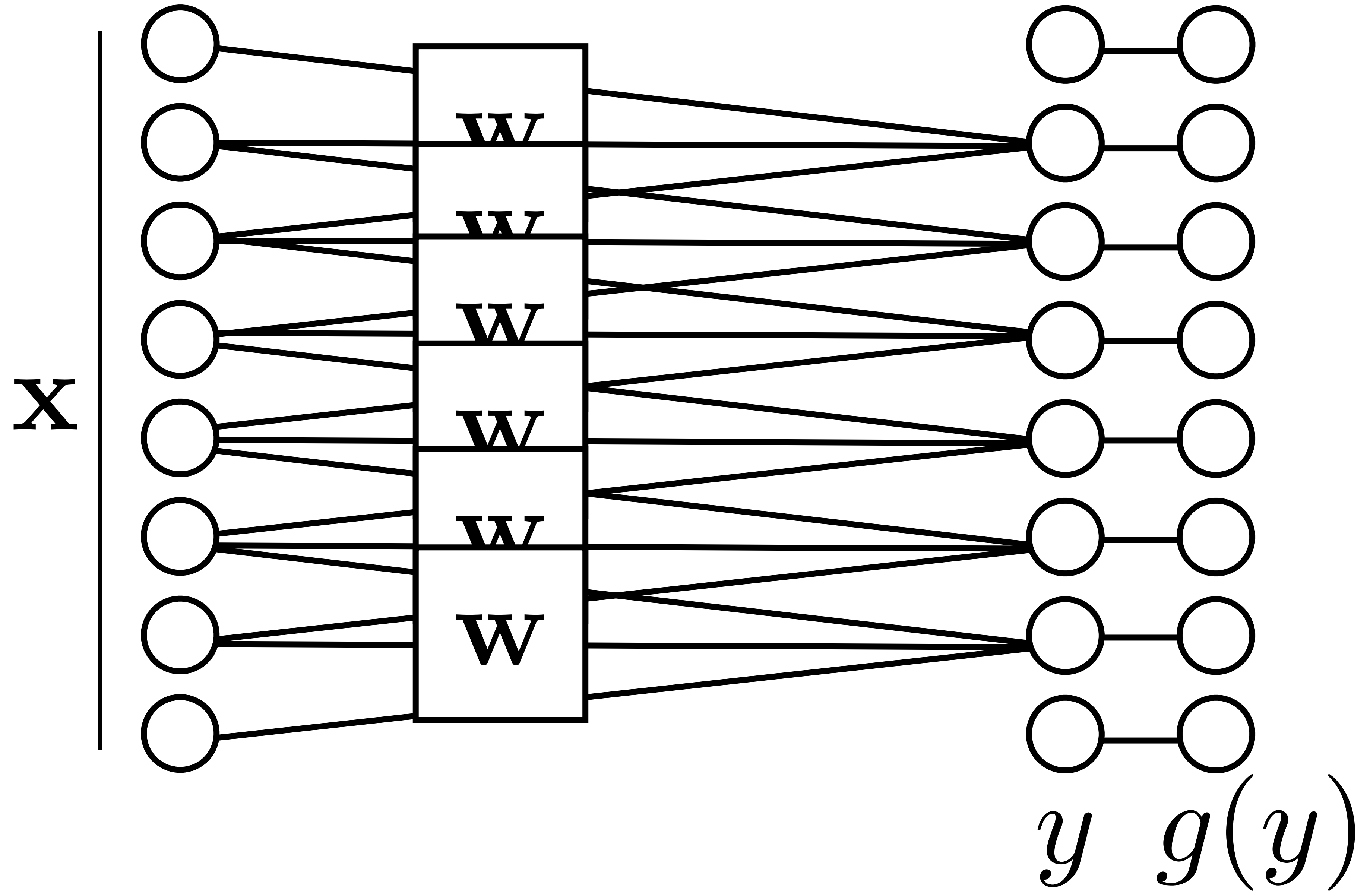


Each output unit is computed from an image patch.

Source: Isola, Torralba, Freeman

Weight sharing

Conv layer

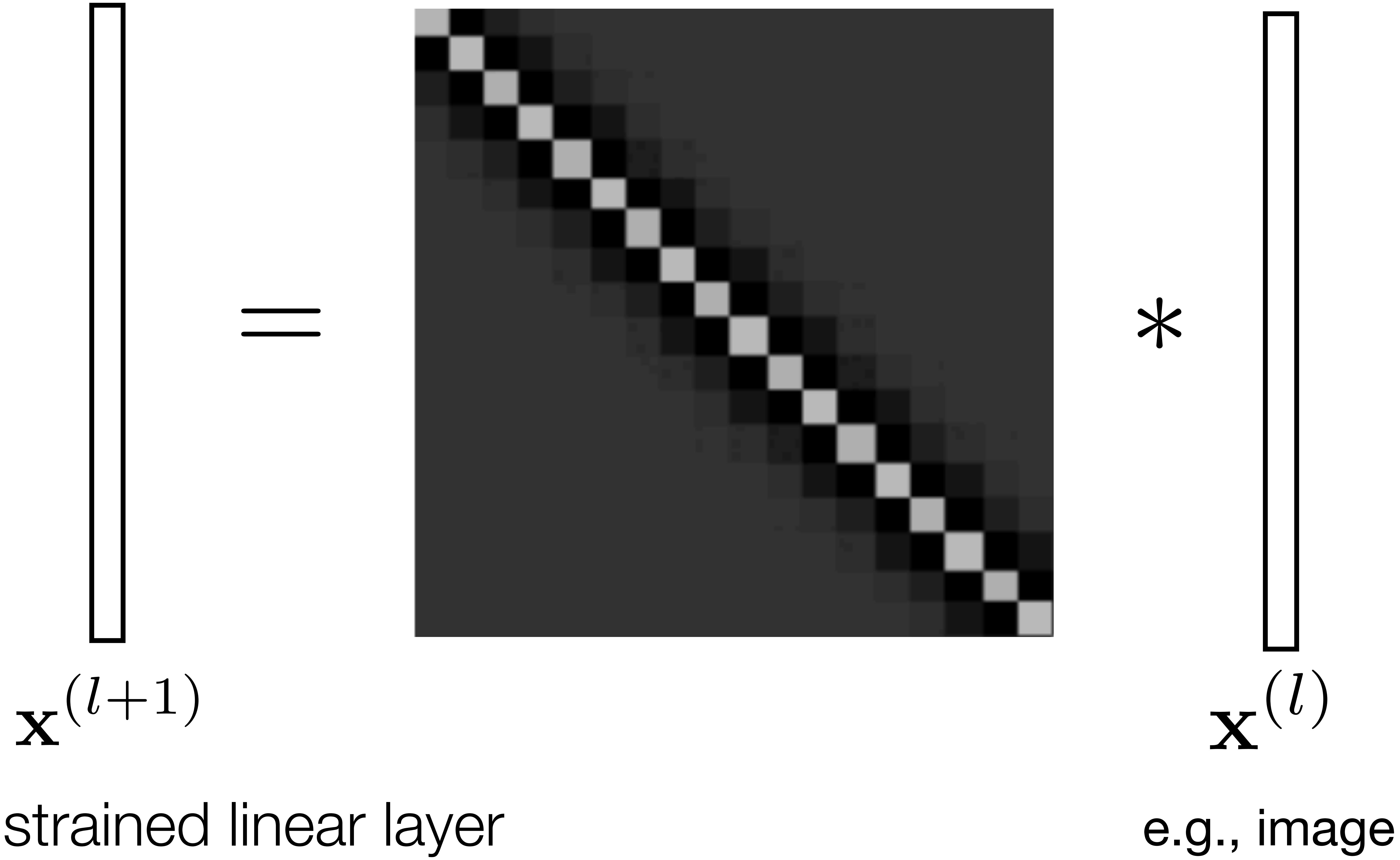


We "share" weights for each patch.

If a feature is useful in one position, it should be useful in others, too.

Source: Isola, Torralba, Freeman

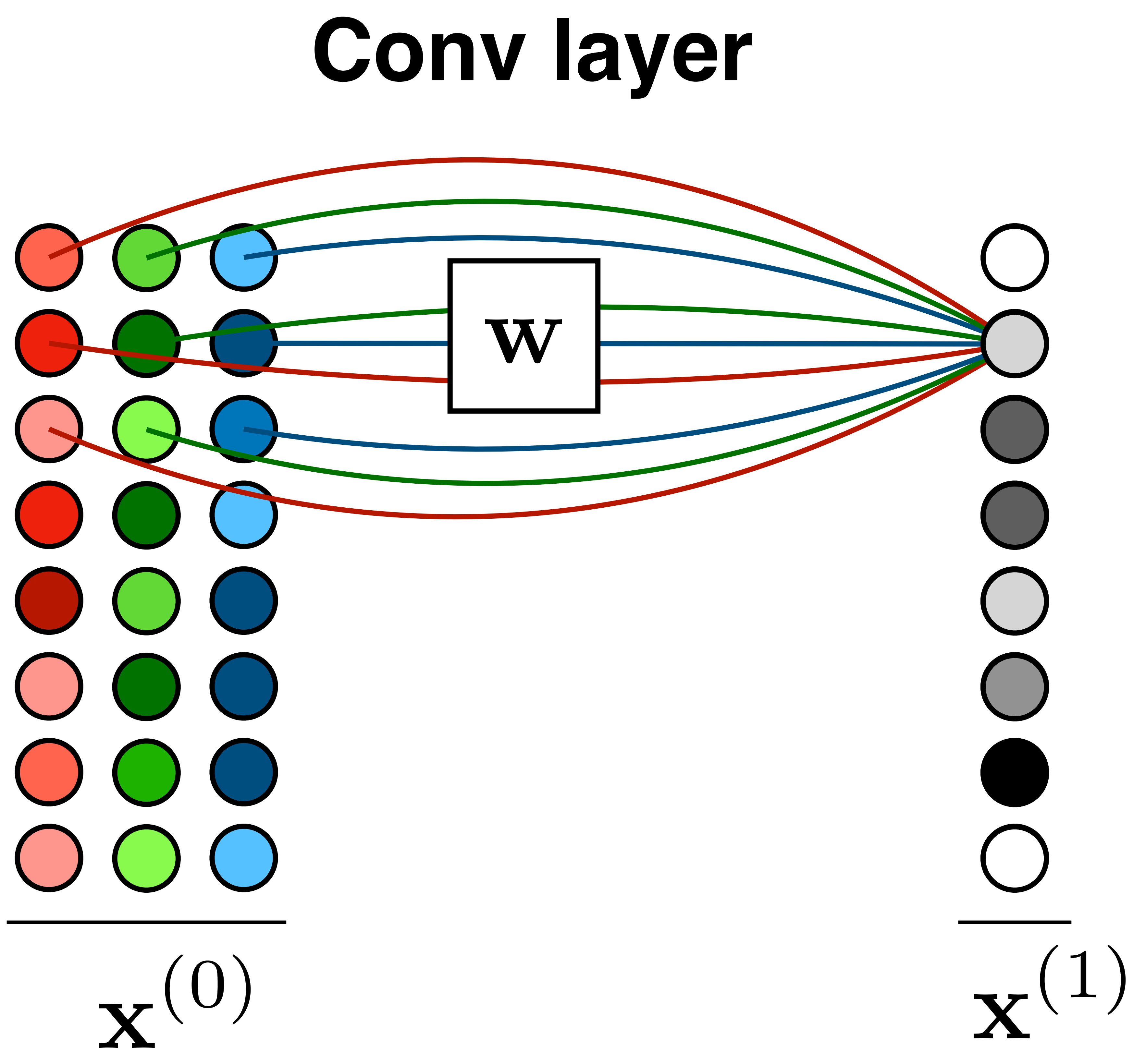
Convolution is a linear function



- Constrained linear layer
- Fewer parameters: easier to learn, less overfitting
- Usually use zero padding

Source: Isola, Torralba, Freeman

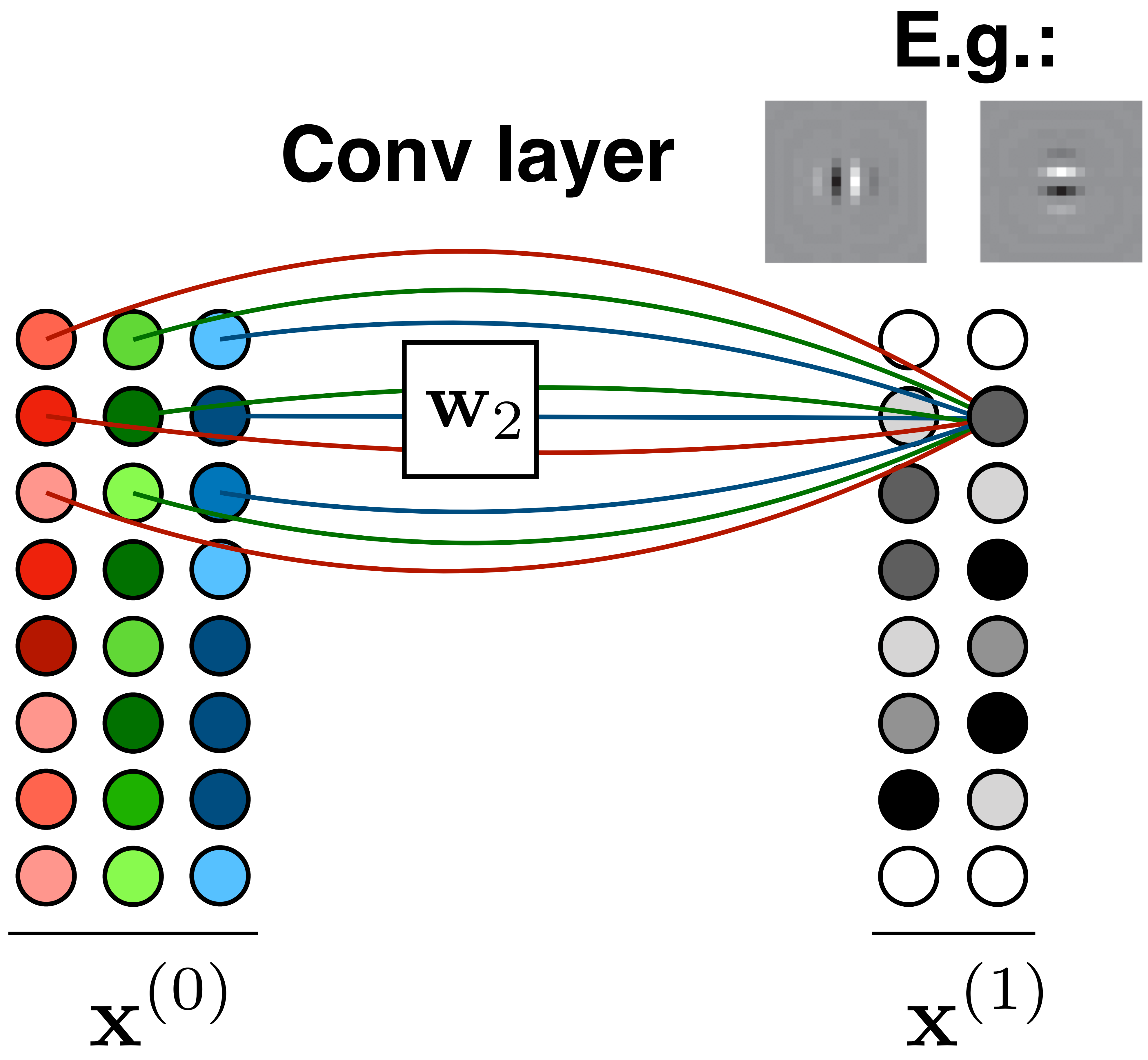
Multiple channels



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

Source: Isola, Torralba, Freeman

Multiple channels

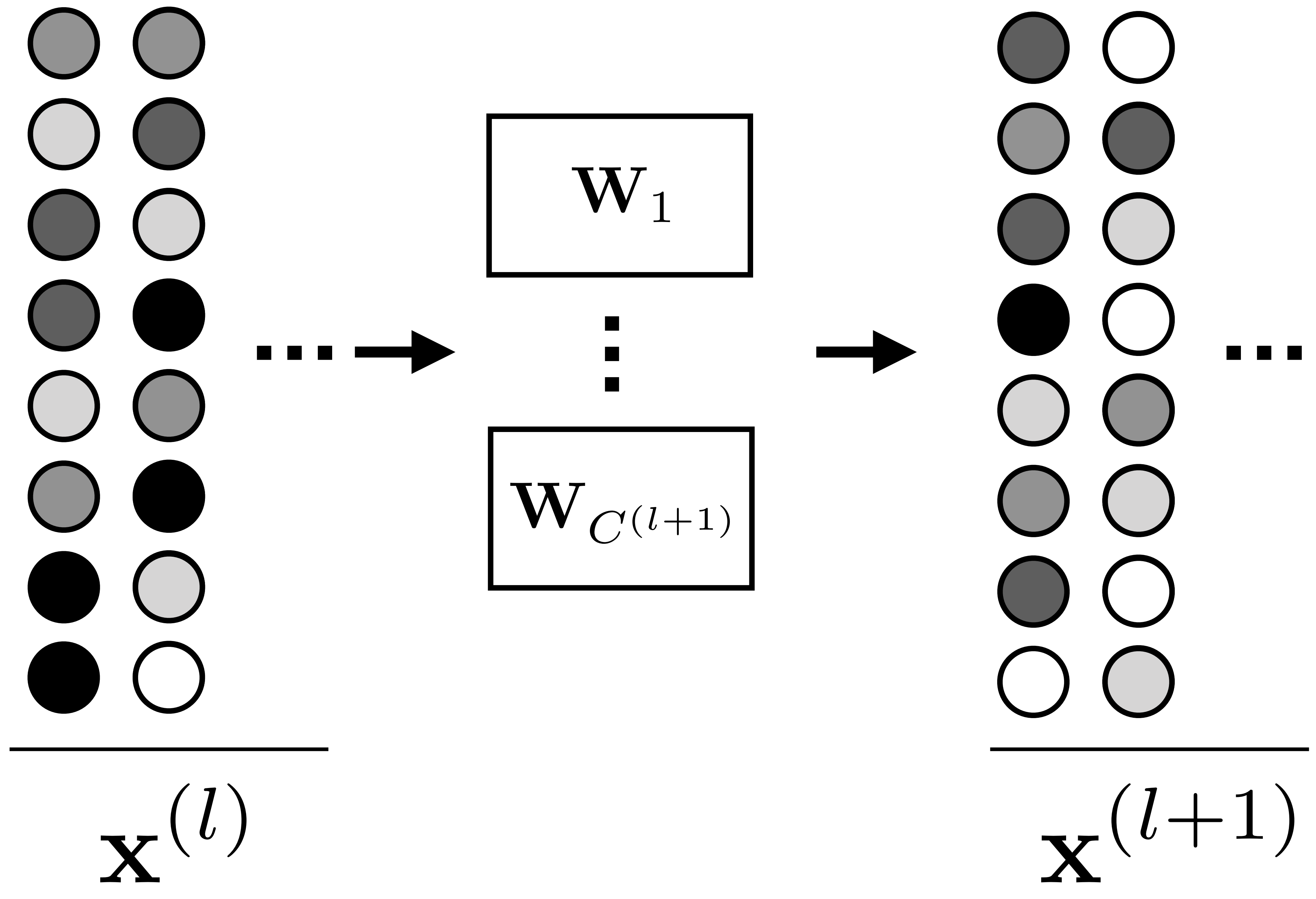


$$\mathbb{R}^{N \times C^{(0)}} \to \mathbb{R}^{N \times C^{(1)}}$$

Source: Isola, Torralba, Freeman

Multiple channels

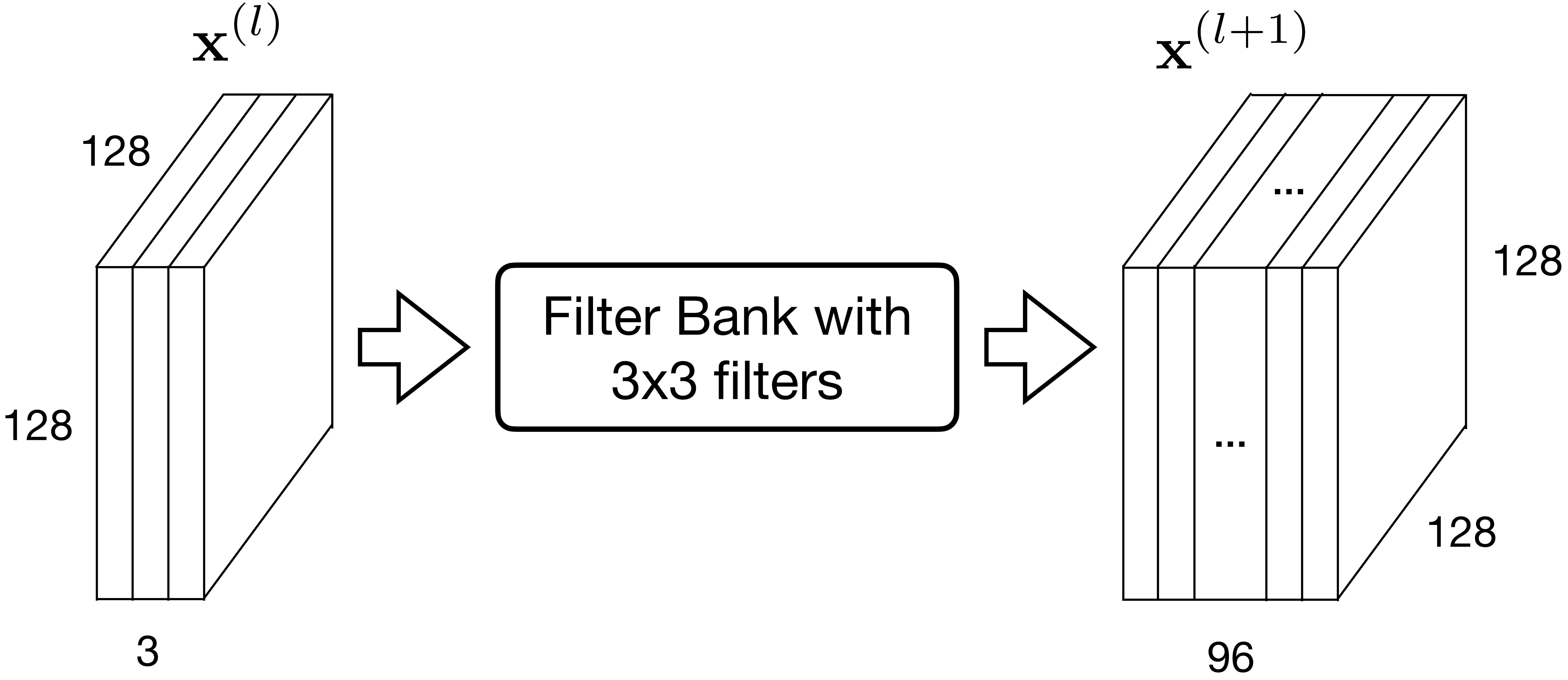
Conv layer



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

Source: Isola, Torralba, Freeman

Multiple channels: Example

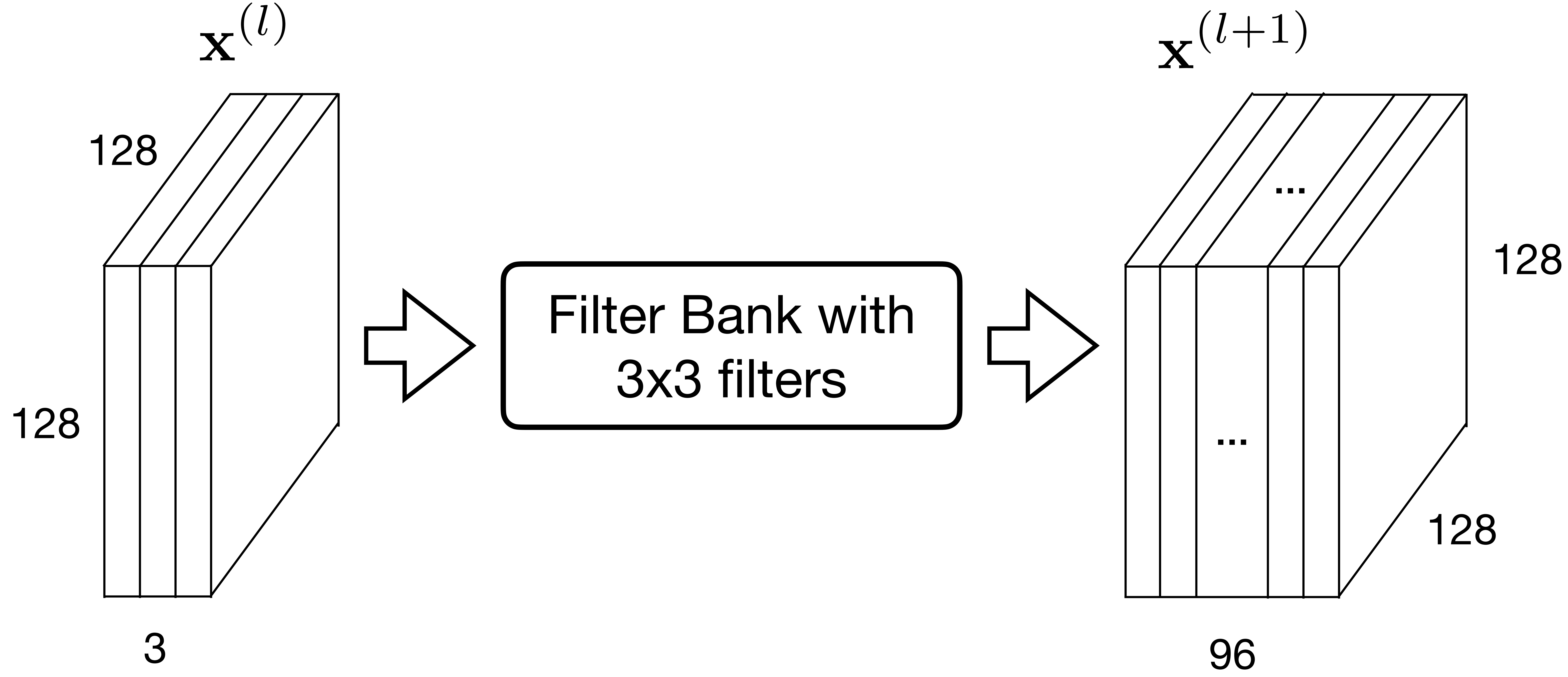


How many parameters does *each filter* have?

(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

Image classification

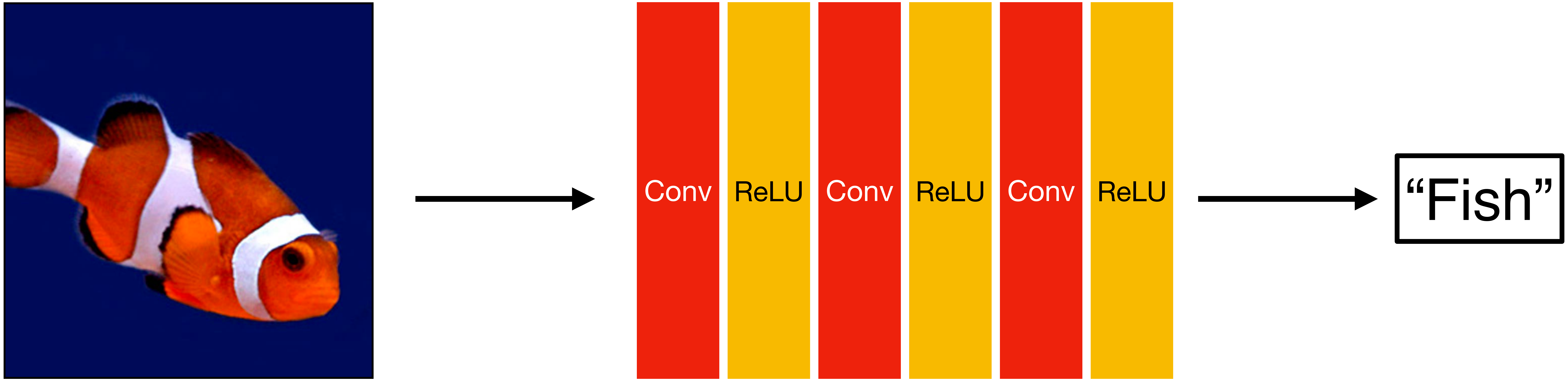


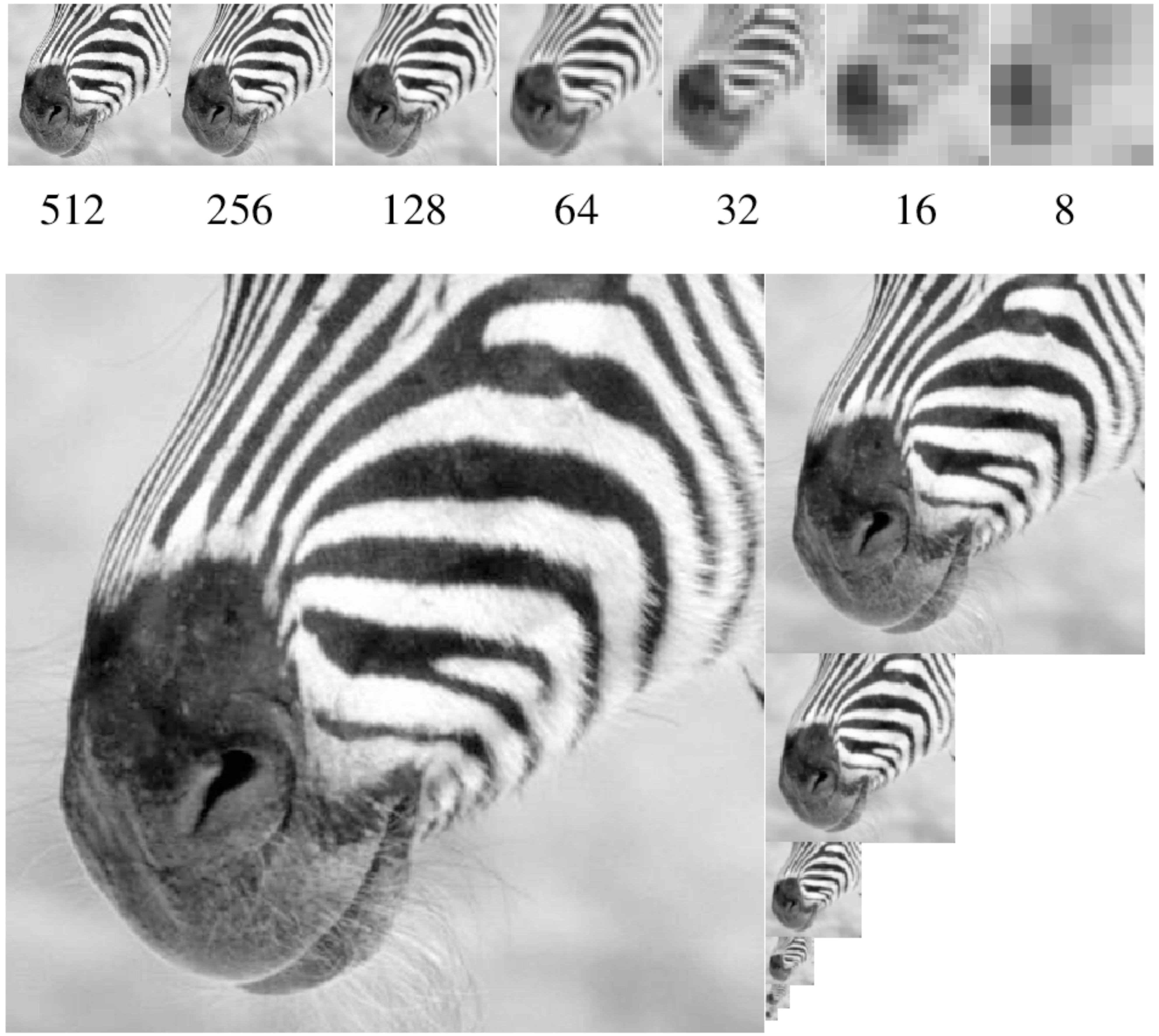
image **x** label y

Problems with this idea:

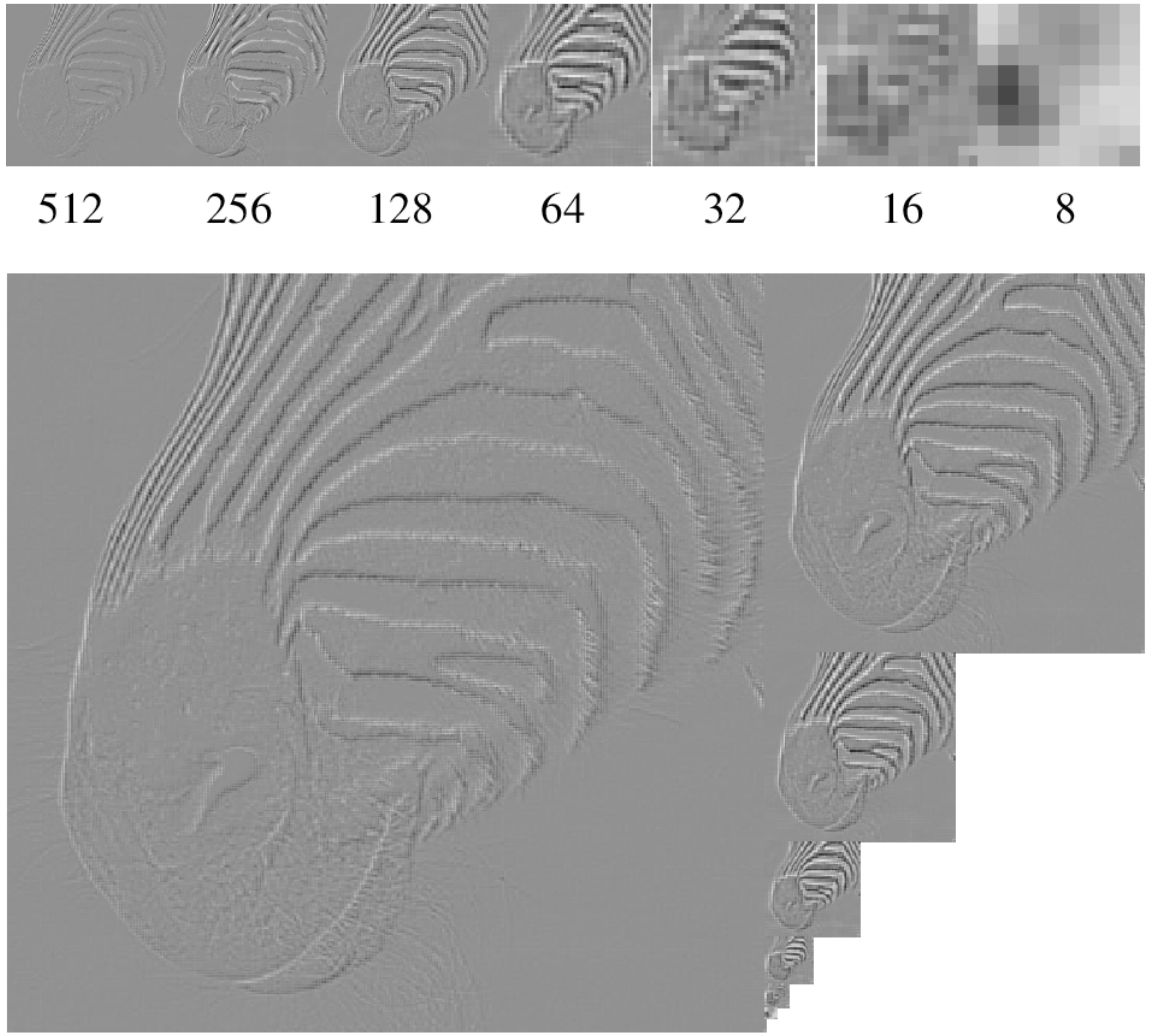
1. No “global” processing.
2. How do you get the final label?

Source: Isola, Torralba, Freeman

Recall: pyramid representations



Gaussian Pyramid

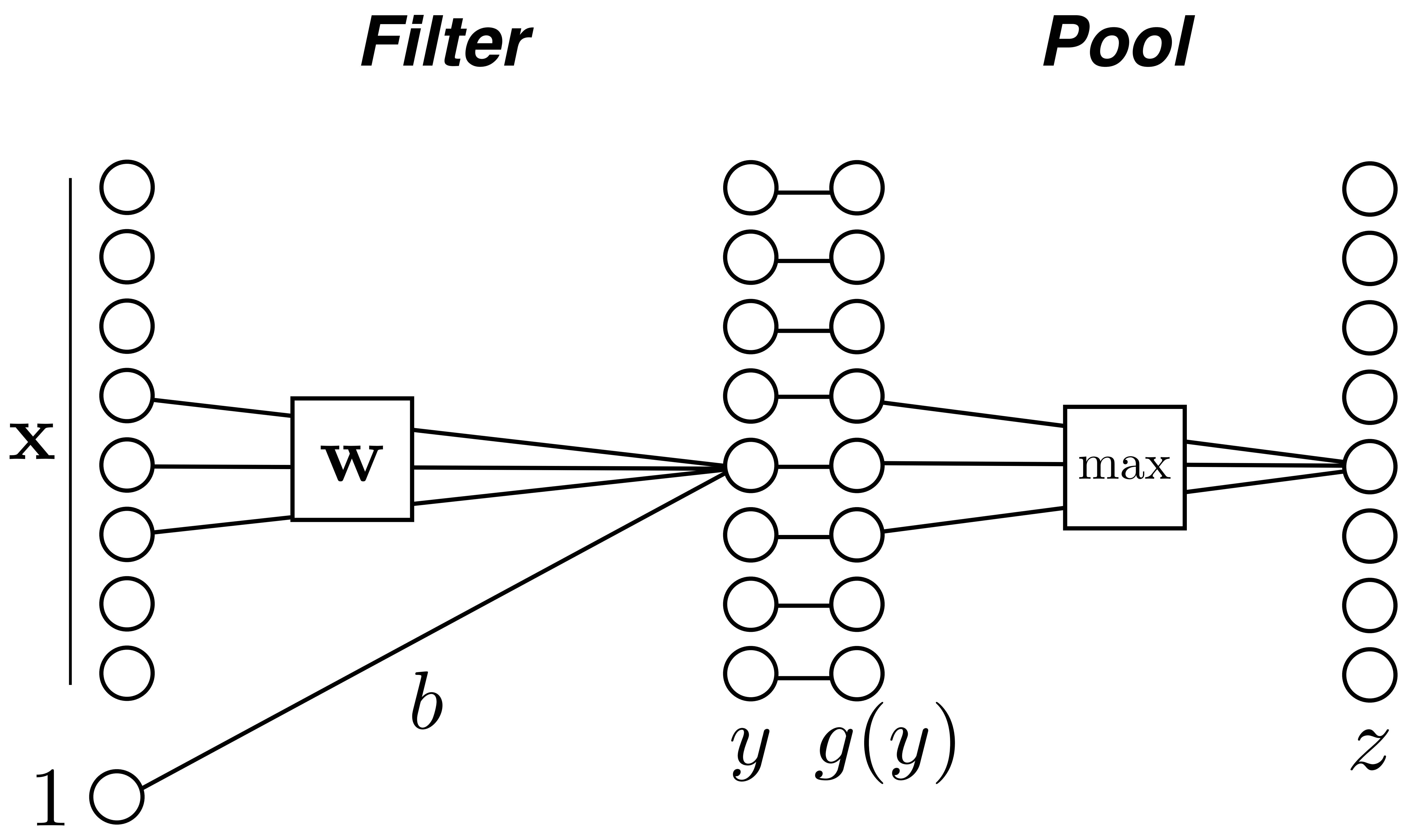


Laplacian Pyramid

Can we use a similar idea in CNNs?

Source: Isola, Torralba, Freeman

Pooling

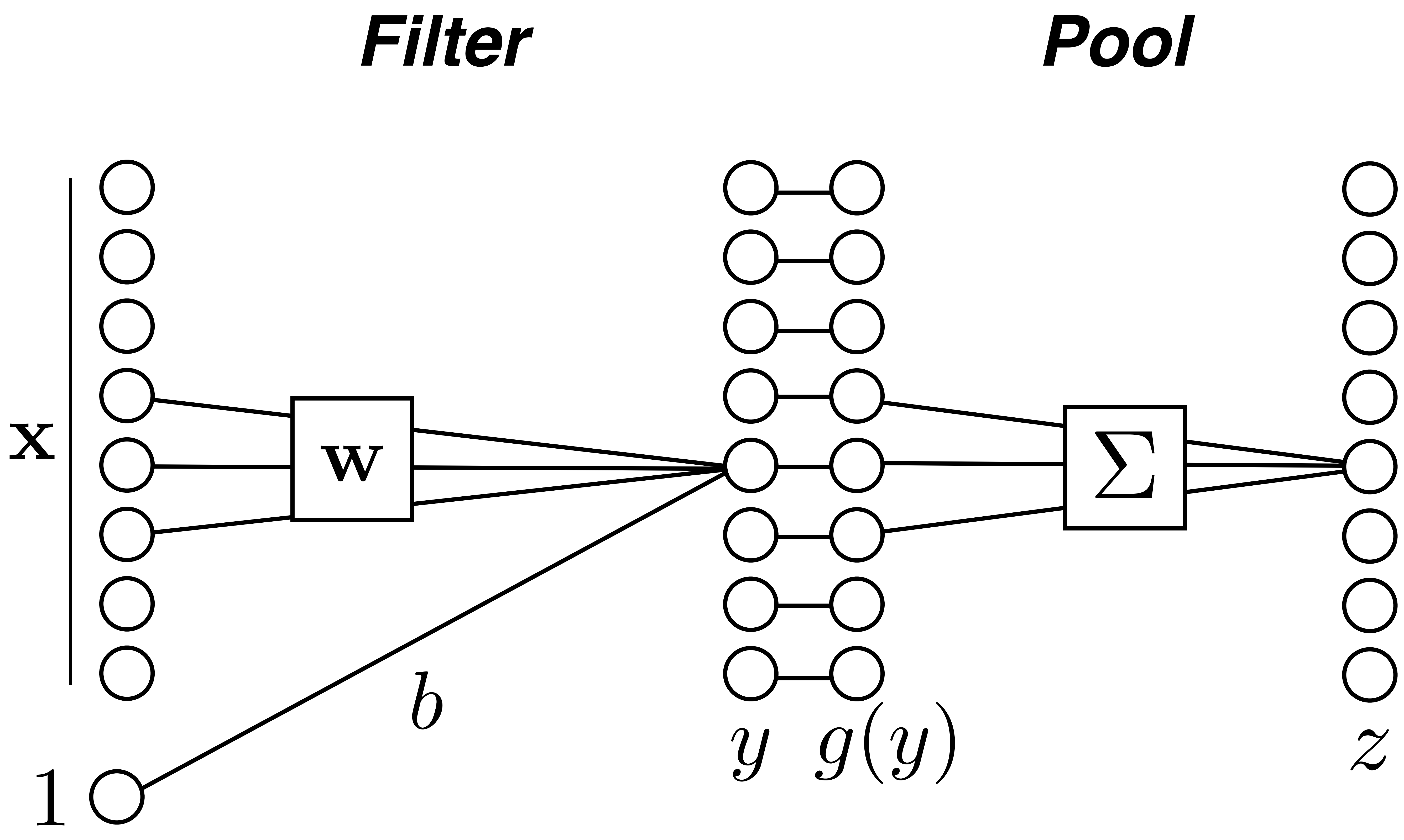


Max pooling

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

Source: Isola, Torralba, Freeman

Pooling



Max pooling

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

Mean pooling

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

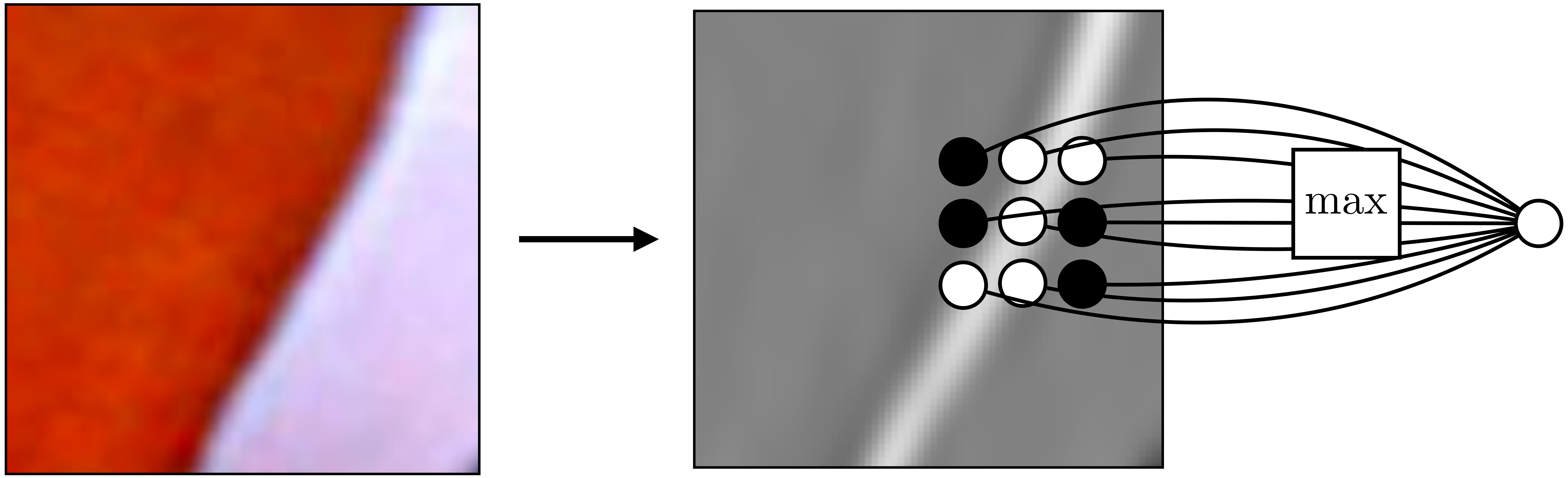
Blurring [Zhang 2019]

$$z = \mathrm{conv}(y, \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix})$$

Source: Isola, Torralba, Freeman

Pooling — Why?

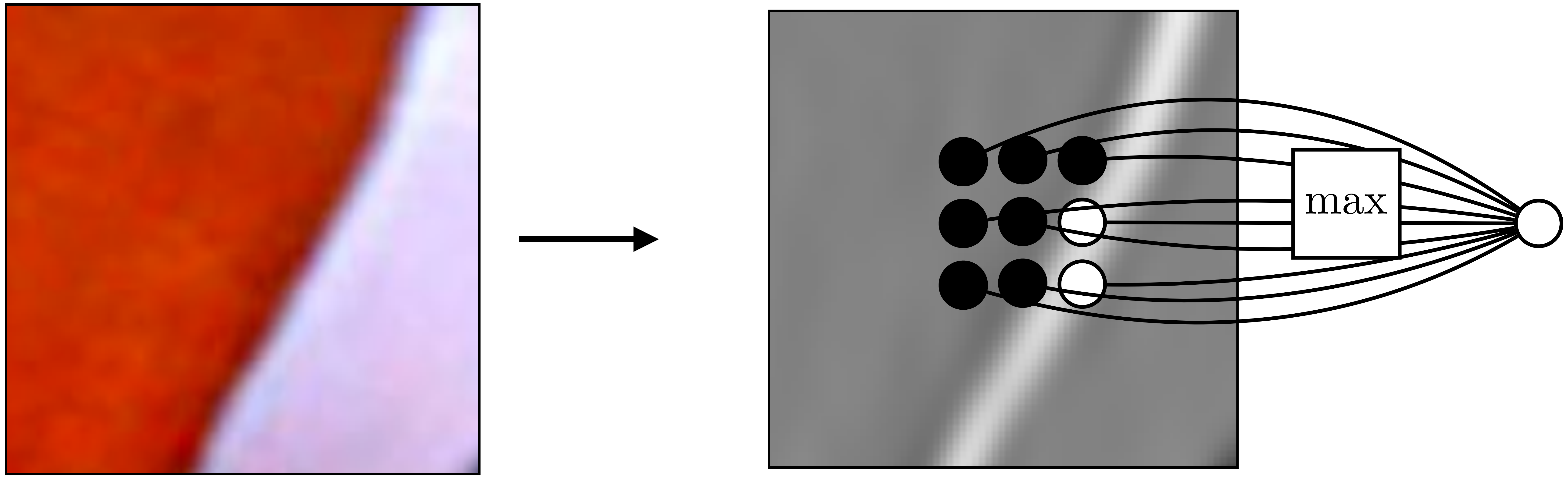
Pooling across spatial locations achieves stability w.r.t. small translations:



Source: Isola, Torralba, Freeman

Pooling — Why?

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

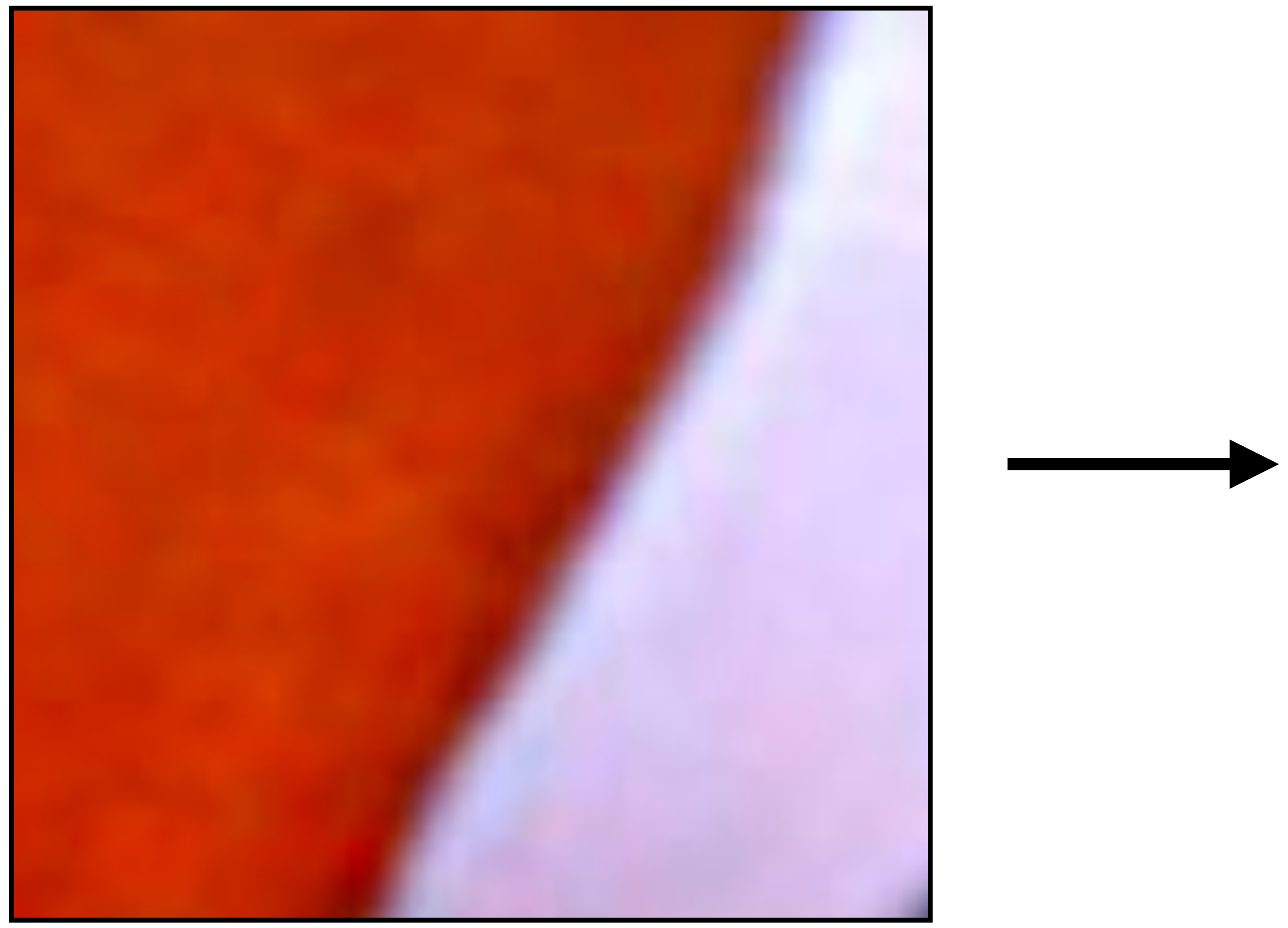


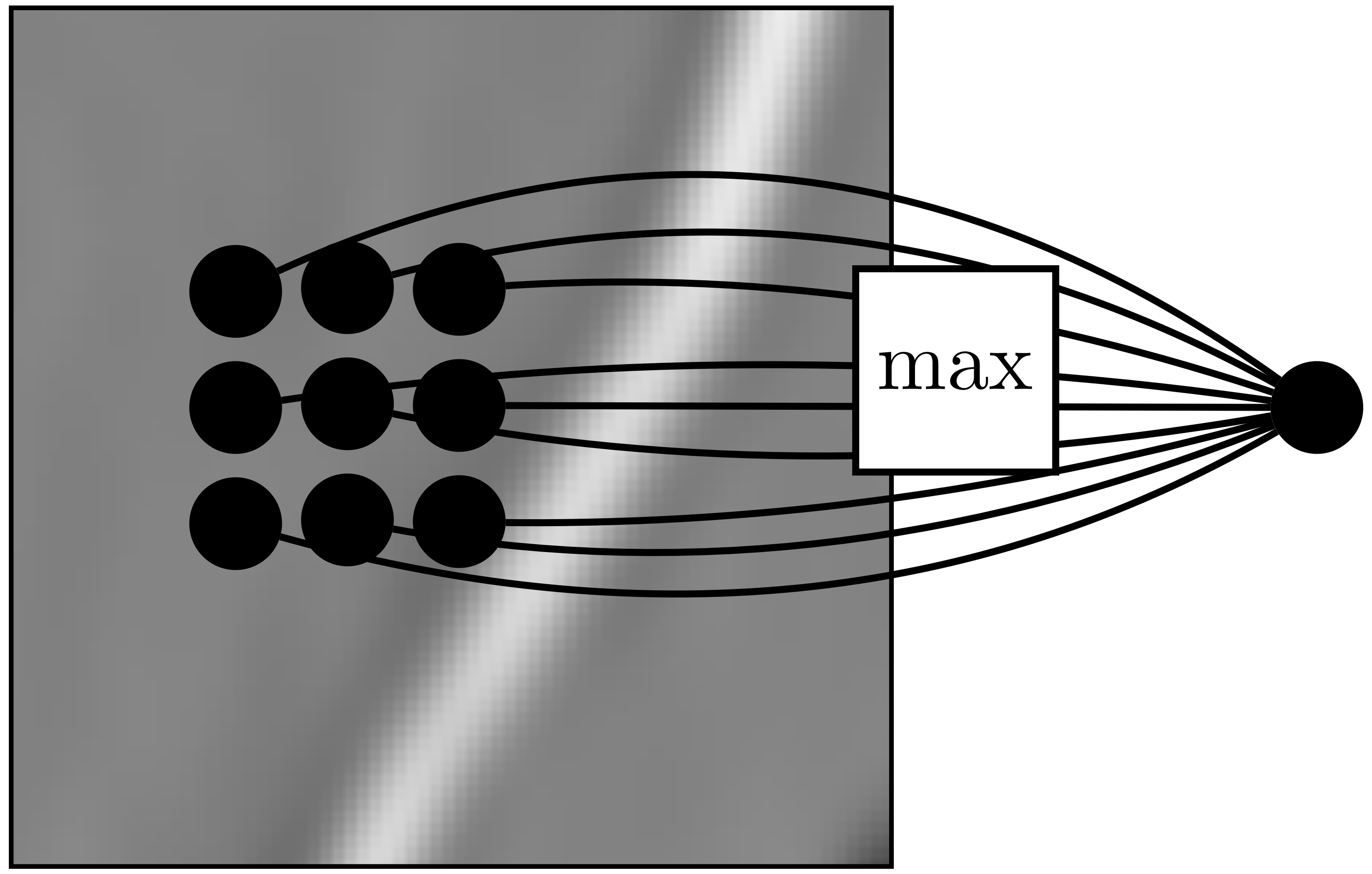
same large response regardless of exact position of edge

Source: Isola, Torralba, Freeman

Pooling — Why?

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

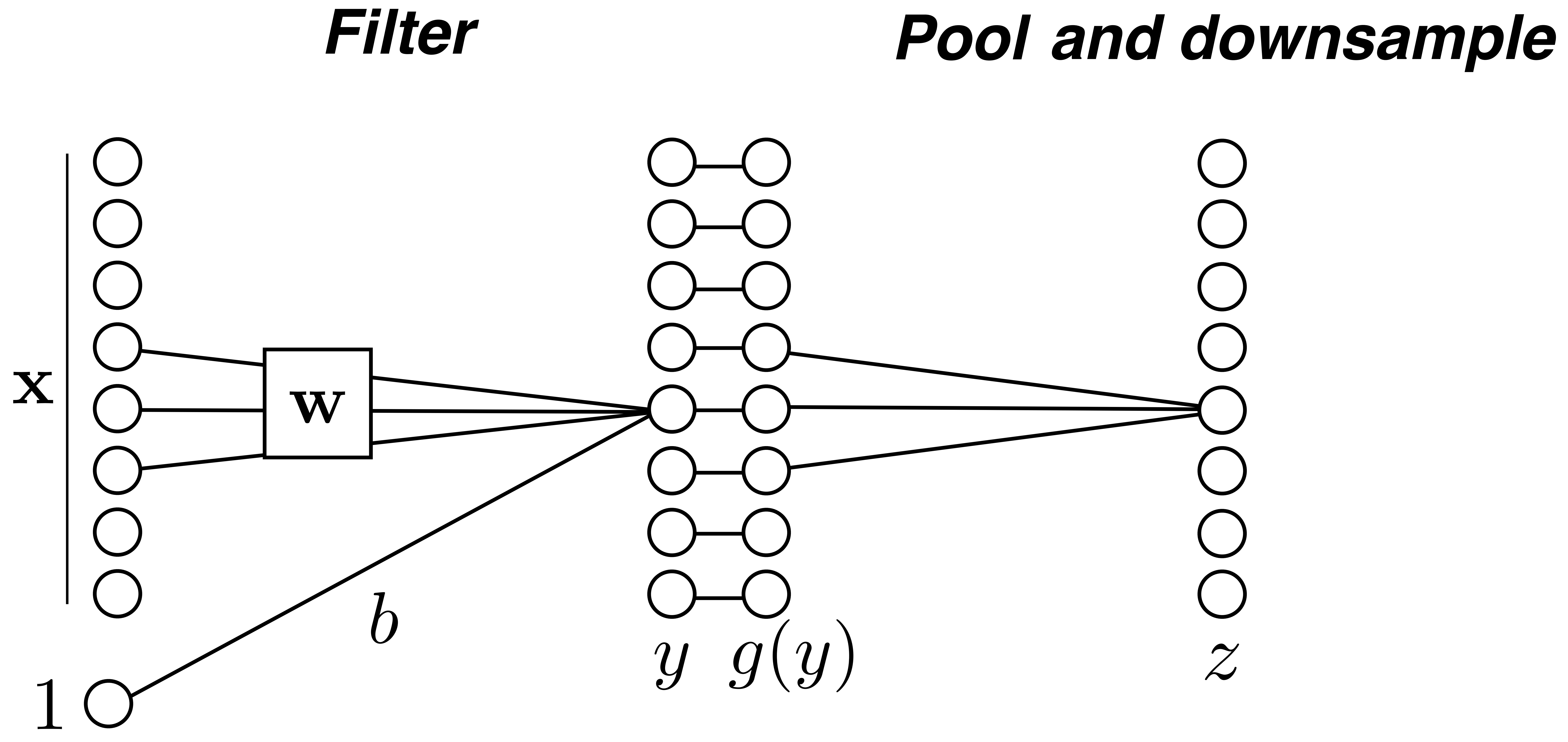




however, if the image is translated a lot…

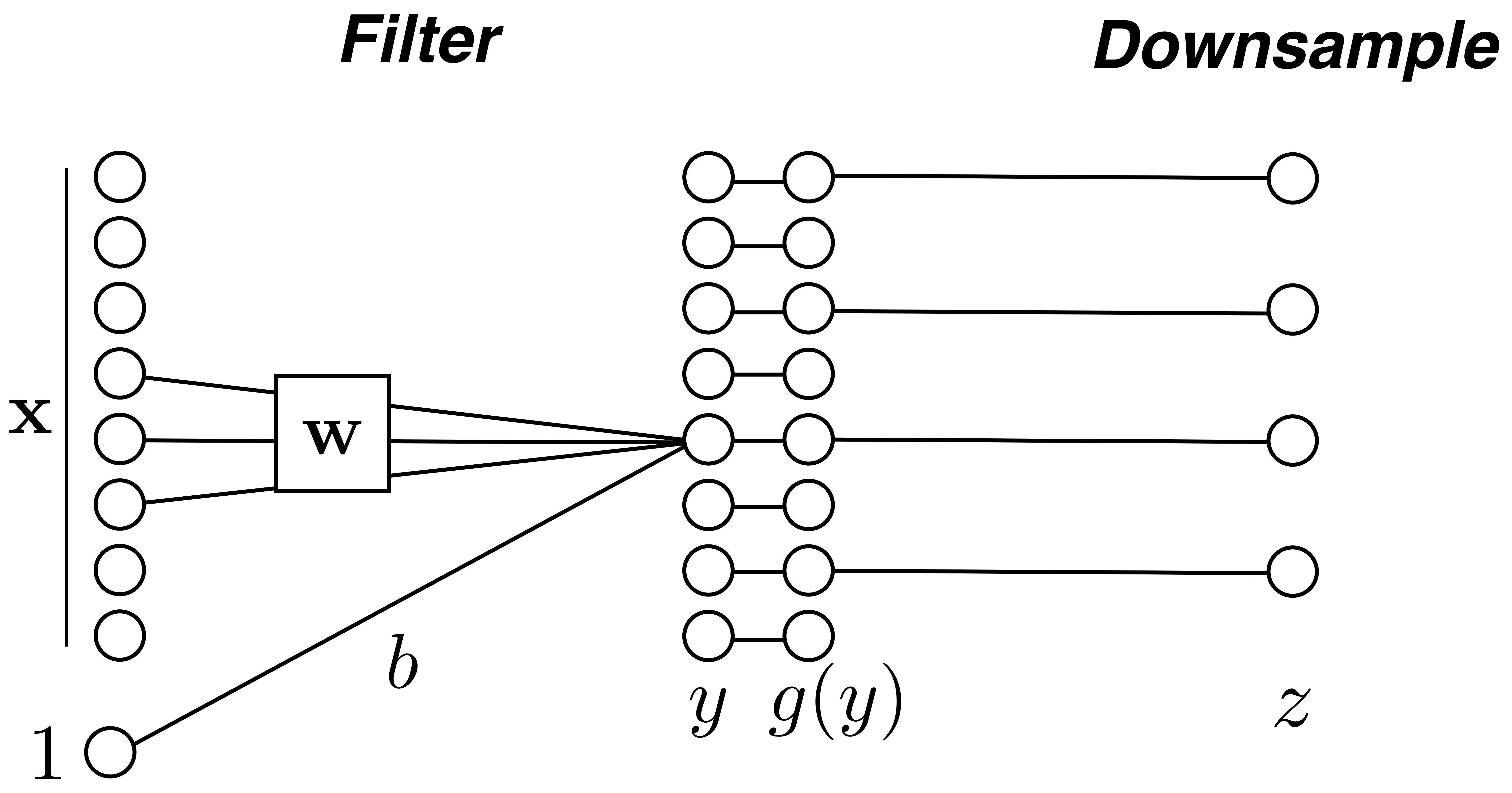
Source: Isola, Torralba, Freeman

Downsampling



Source: Isola, Torralba, Freeman

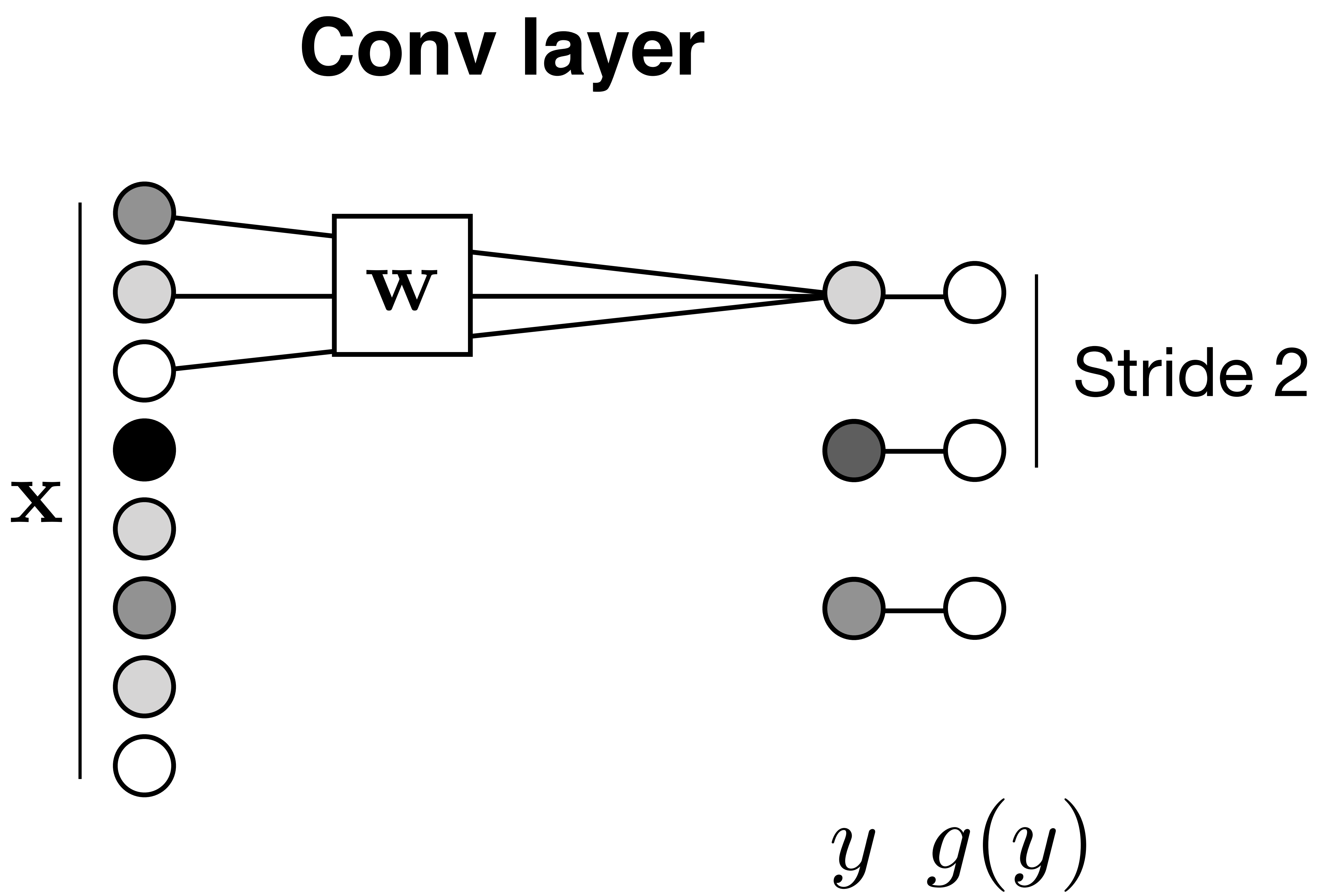
Downsampling



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

Source: Isola, Torralba, Freeman

Strided operations

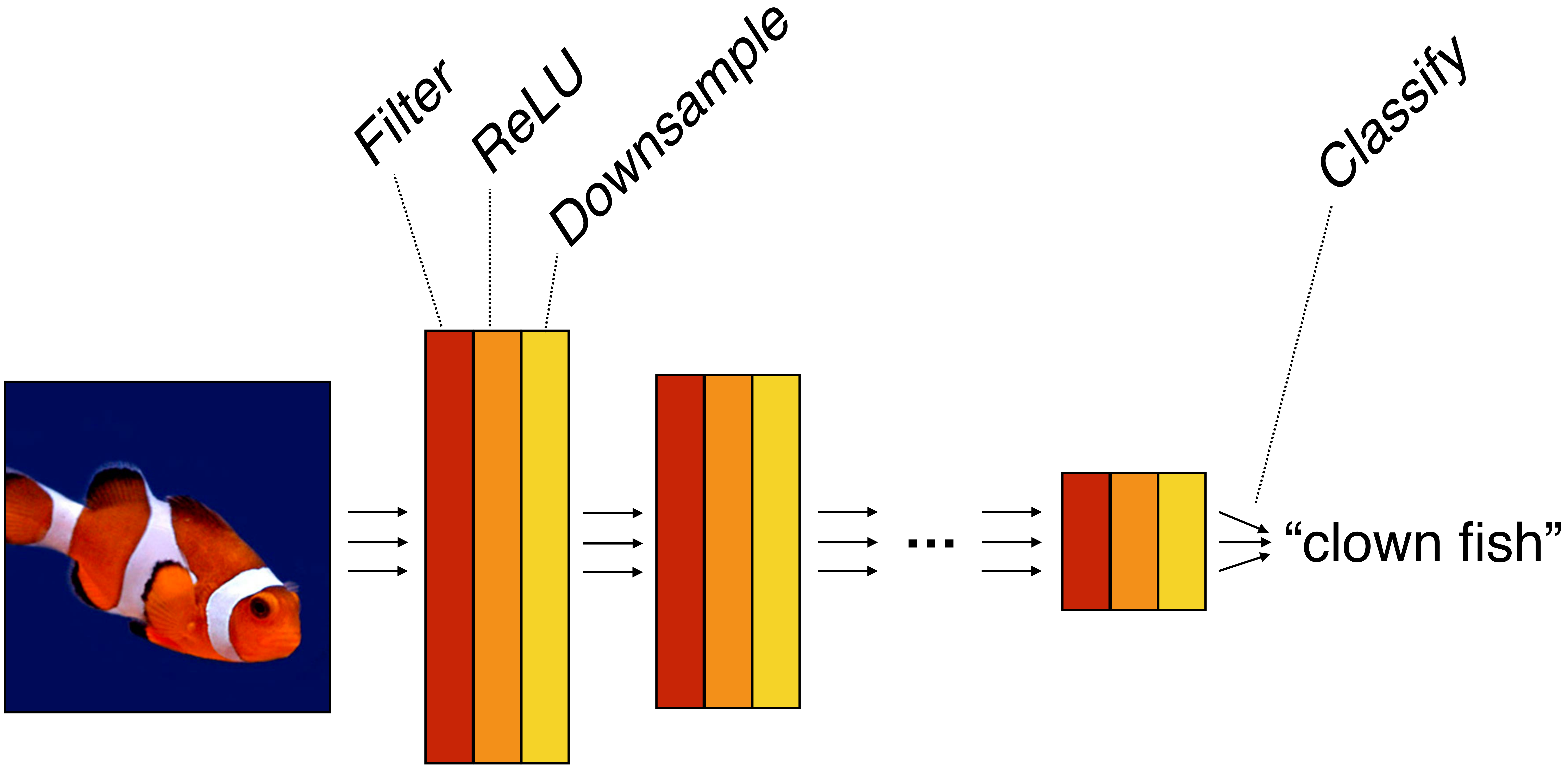


Strided operations combine a given operation (convolution or pooling) and downsampling into a single operation.

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

Source: Isola, Torralba, Freeman

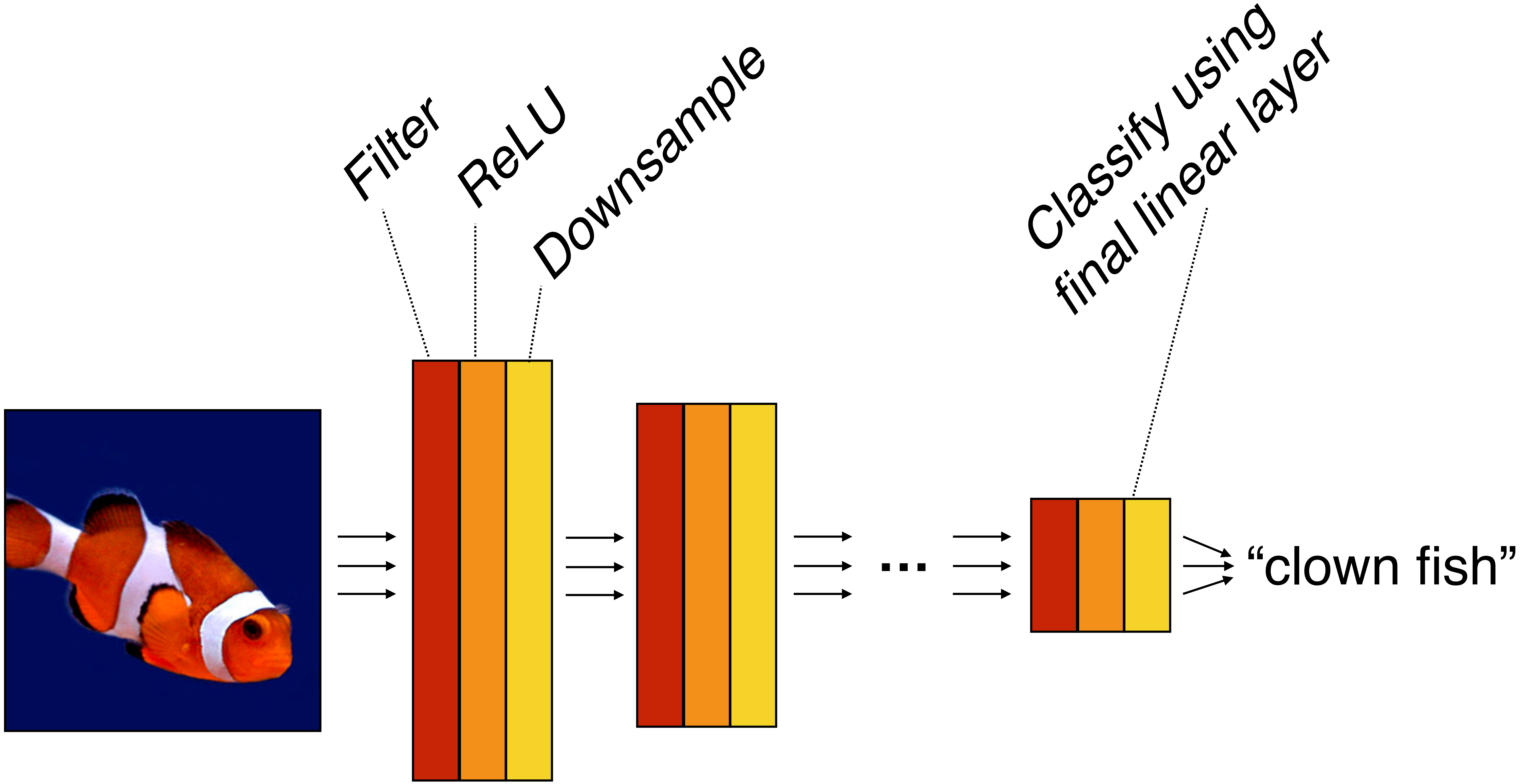
Computation in a neural net



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

Source: Isola, Torralba, Freeman

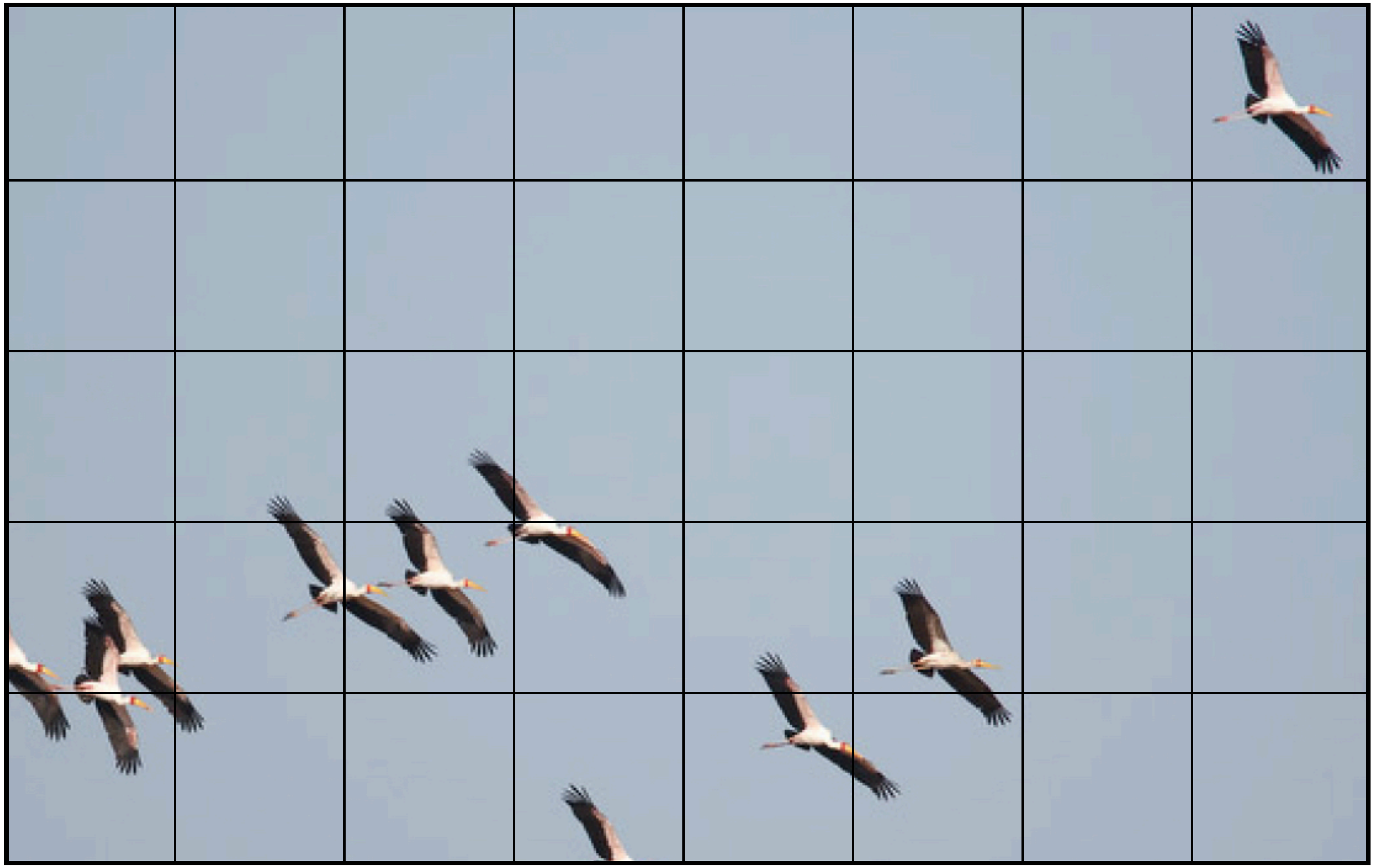
Computation in a neural net



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

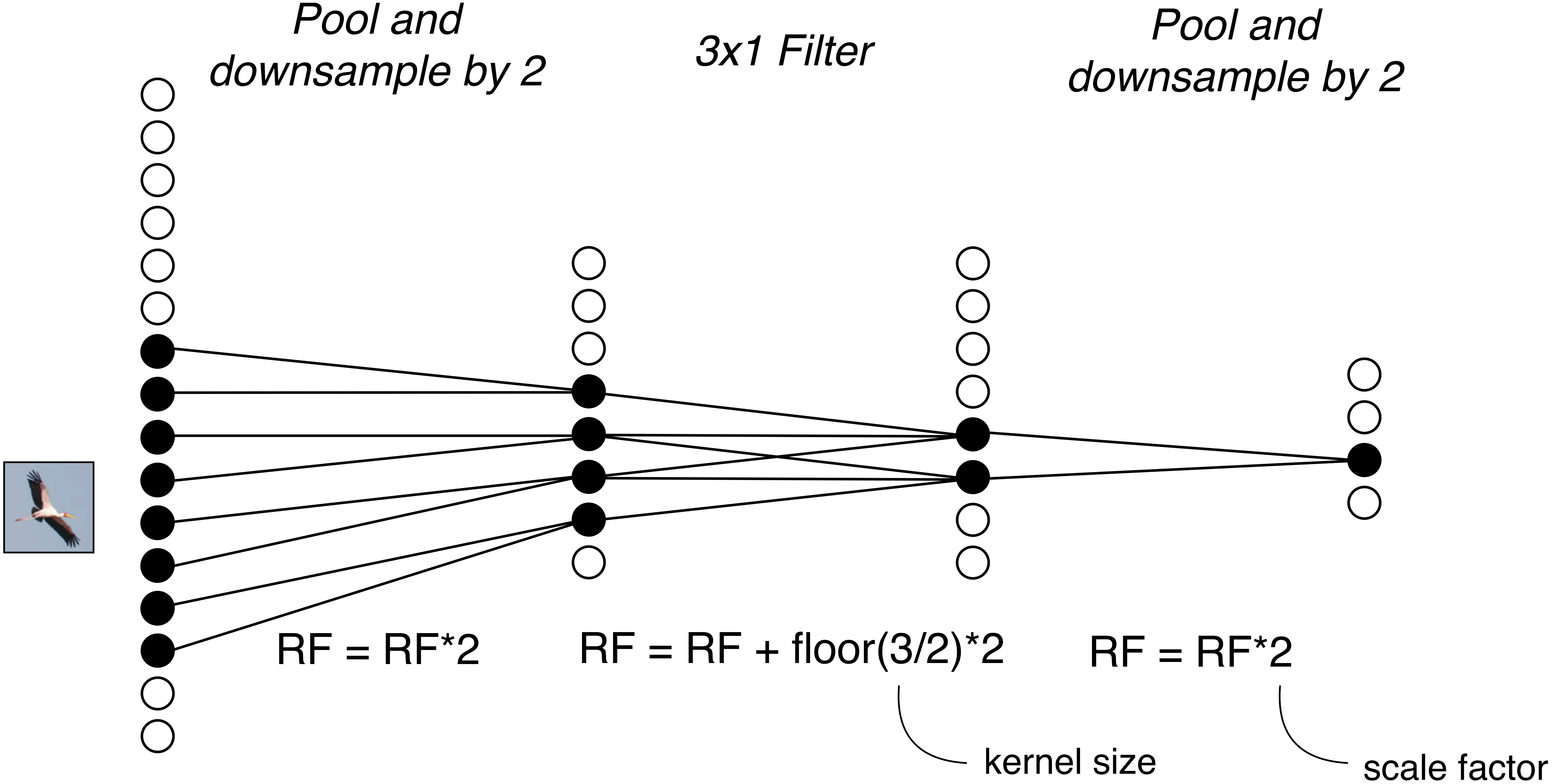
Source: Isola, Torralba, Freeman

Receptive fields



Source: Isola, Torralba, Freeman

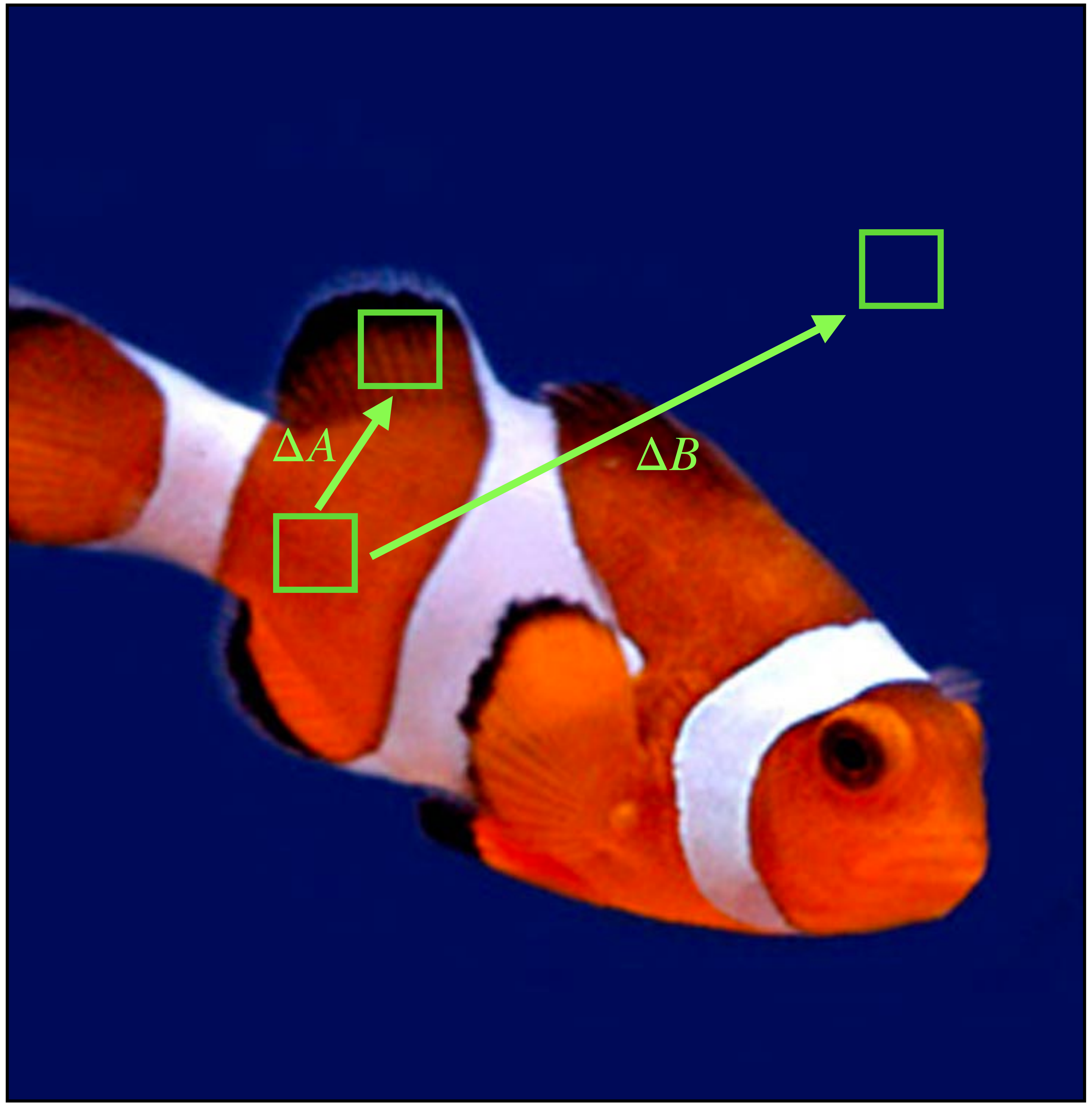
Receptive fields

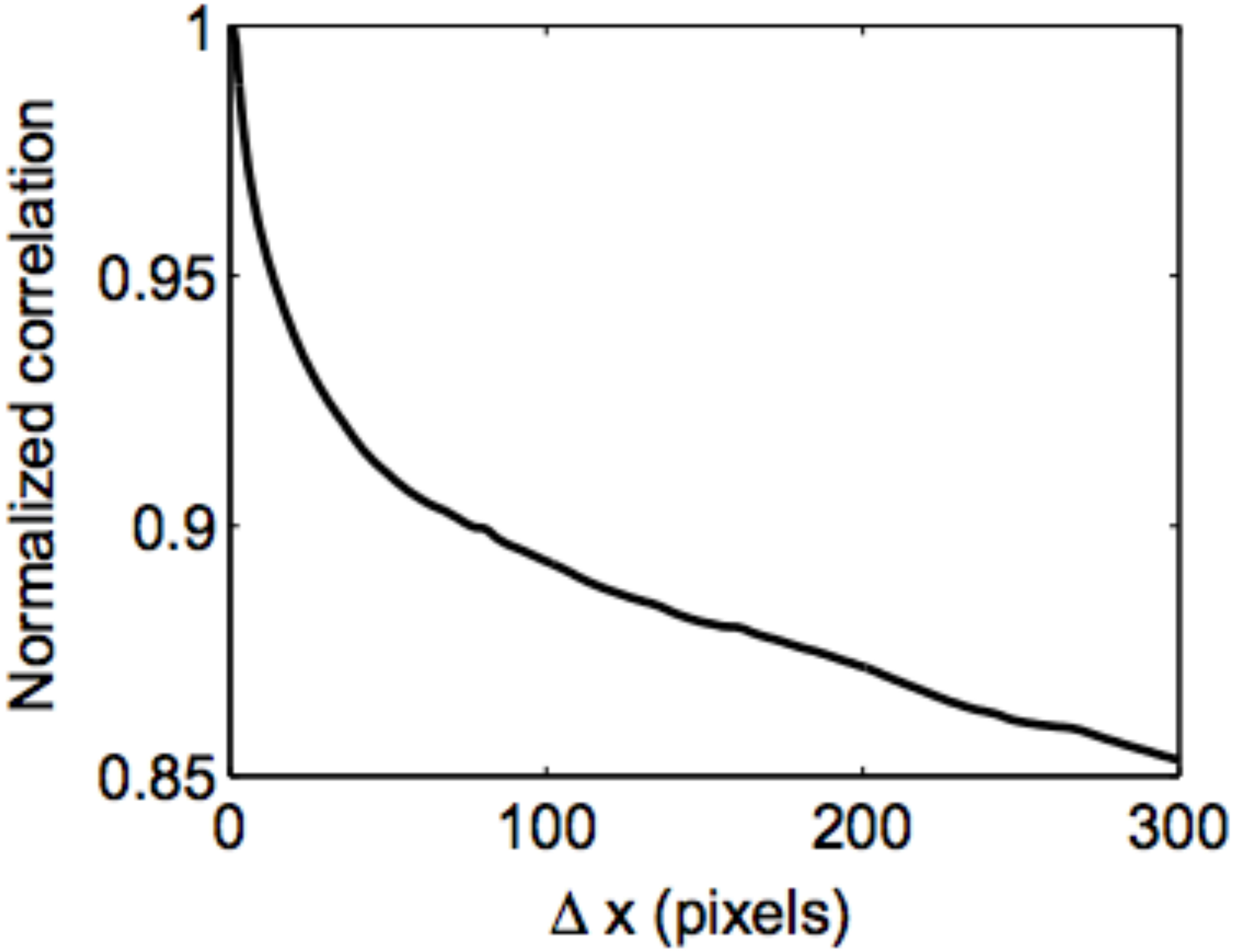


[See also: http://fomoro.com/tools/receptive-fields/index.html]

Source: Isola, Torralba, Freeman

Why have coefficients only for nearby pixels?



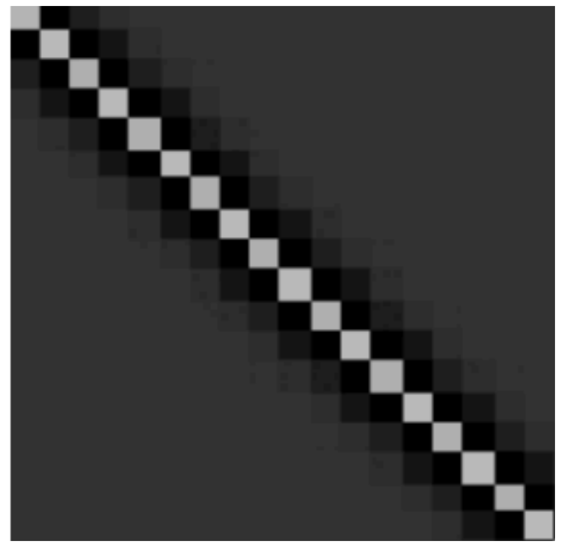


[Simoncelli: Statistical Modelling of Photographic Images, 2005]

Implementation details

Convolutional layers

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



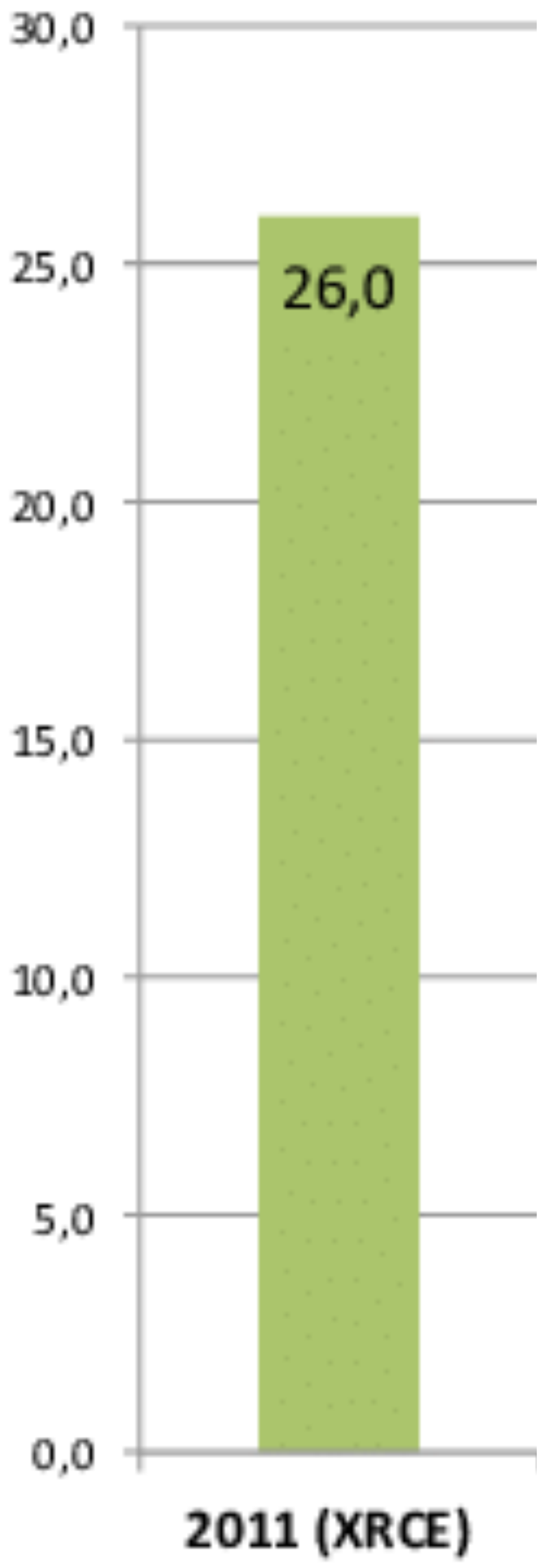
Option 2: Use a more optimized implementation. There are optimized GPU implementations. For very large filters, these use Fourier Transform.

Backwards pass: These take the form of convolutions.

Max pooling: backwards pass similar to ReLU.

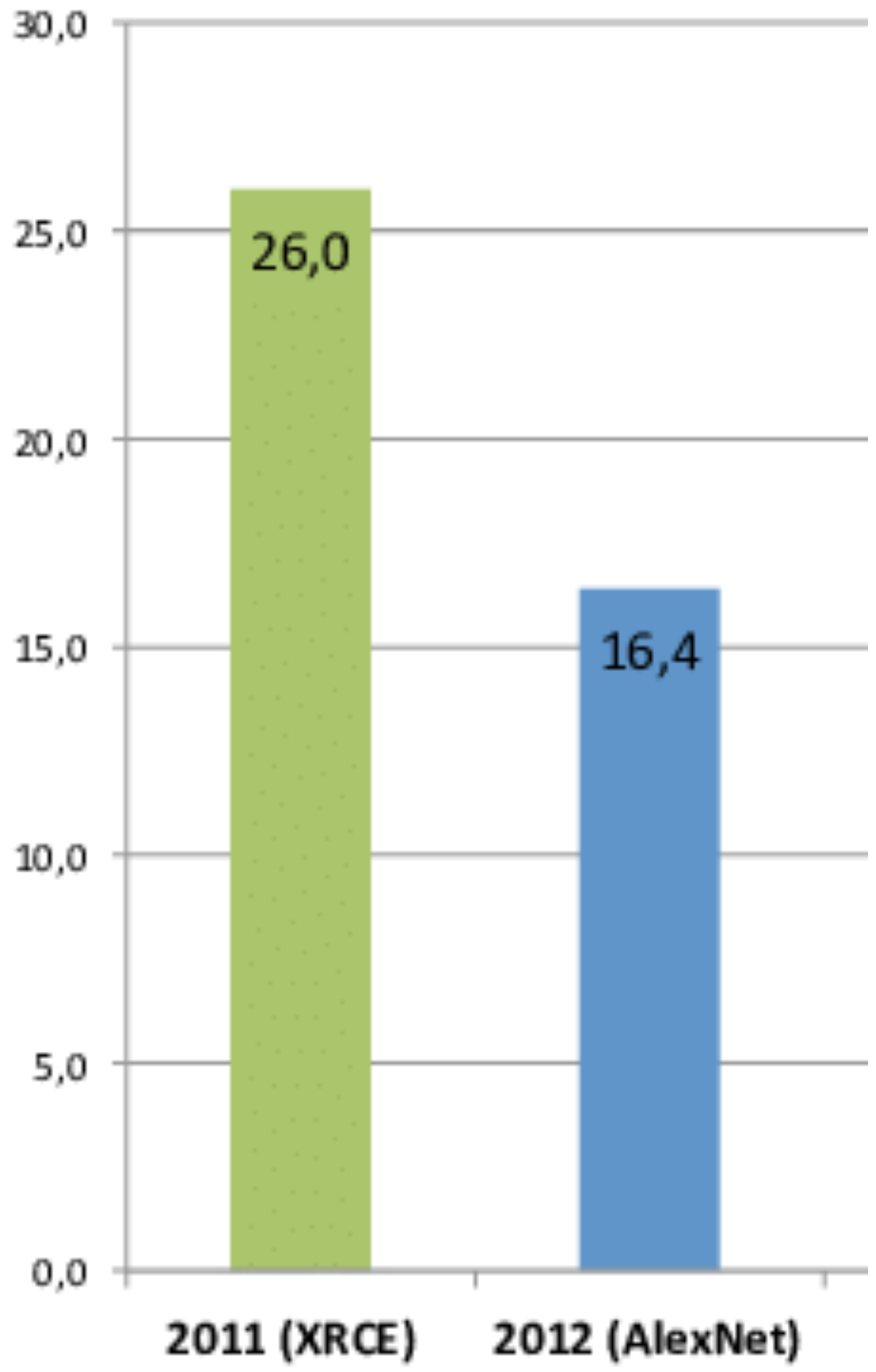
Network designs

ImageNet classification (top 5)

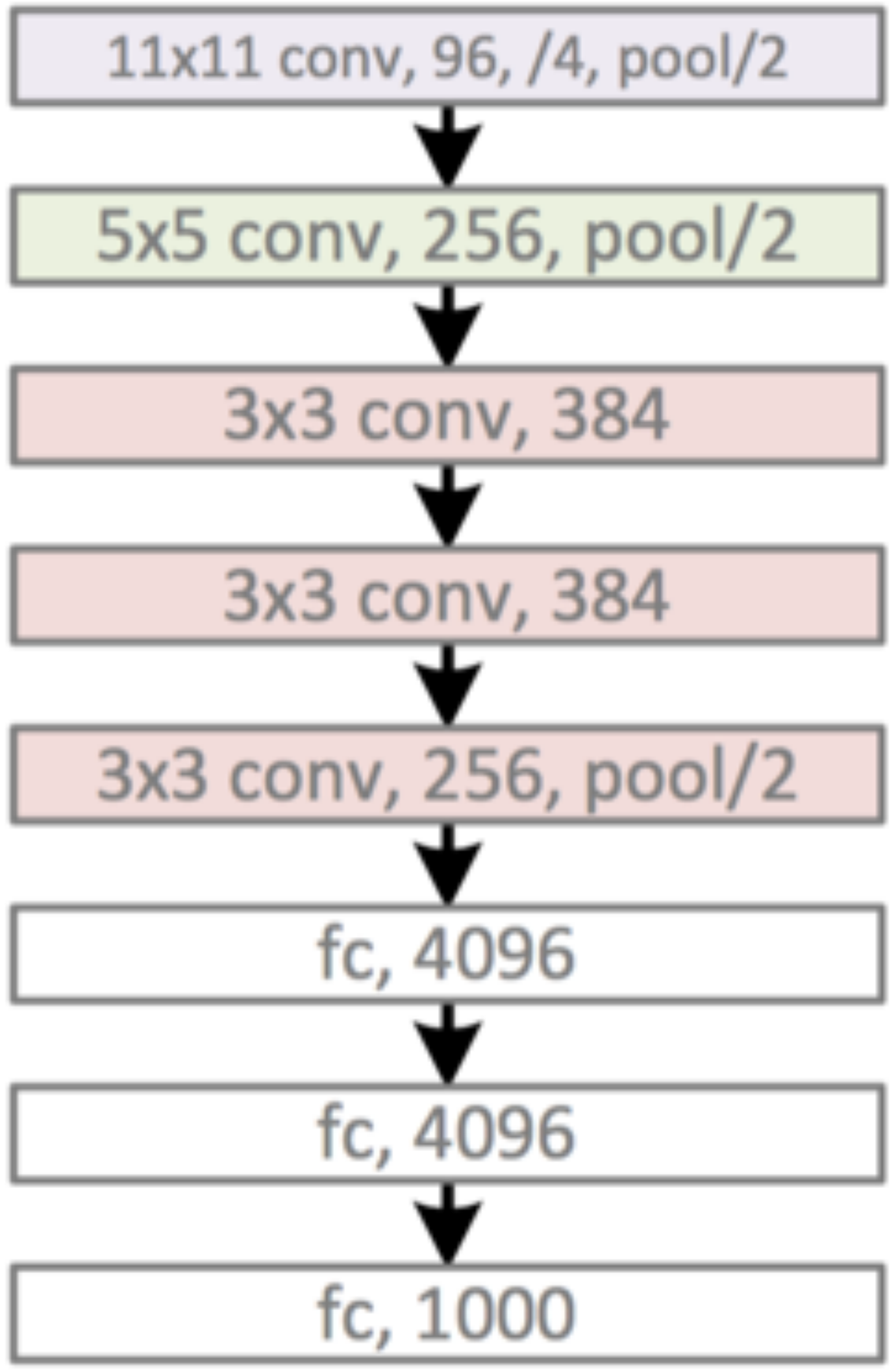


Source: Isola, Torralba, Freeman

ImageNet classification (top 5)



2012: AlexNet
5 conv. layers

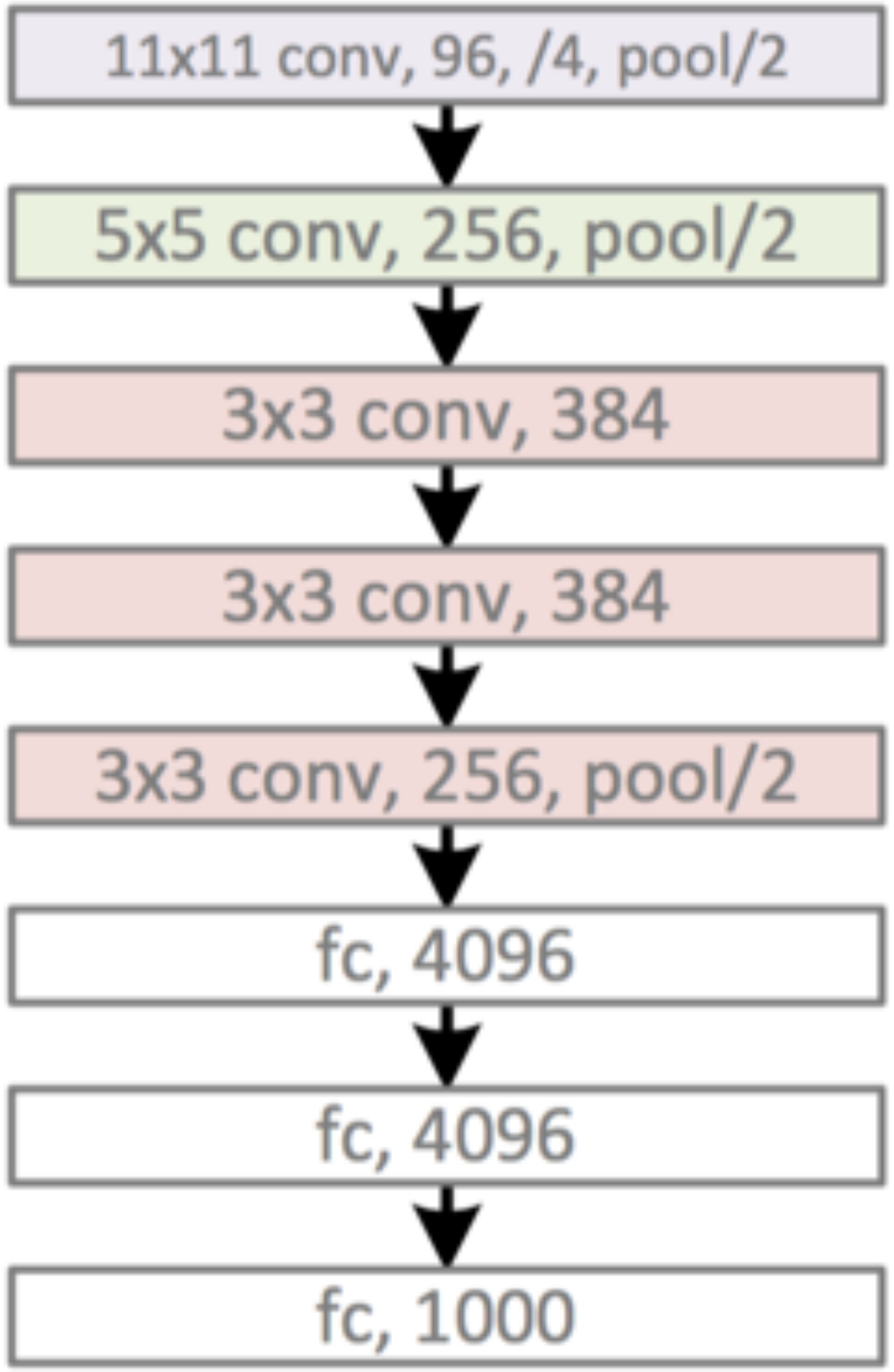


Error: 16.4%

[Krizhevsky et al: ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS 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

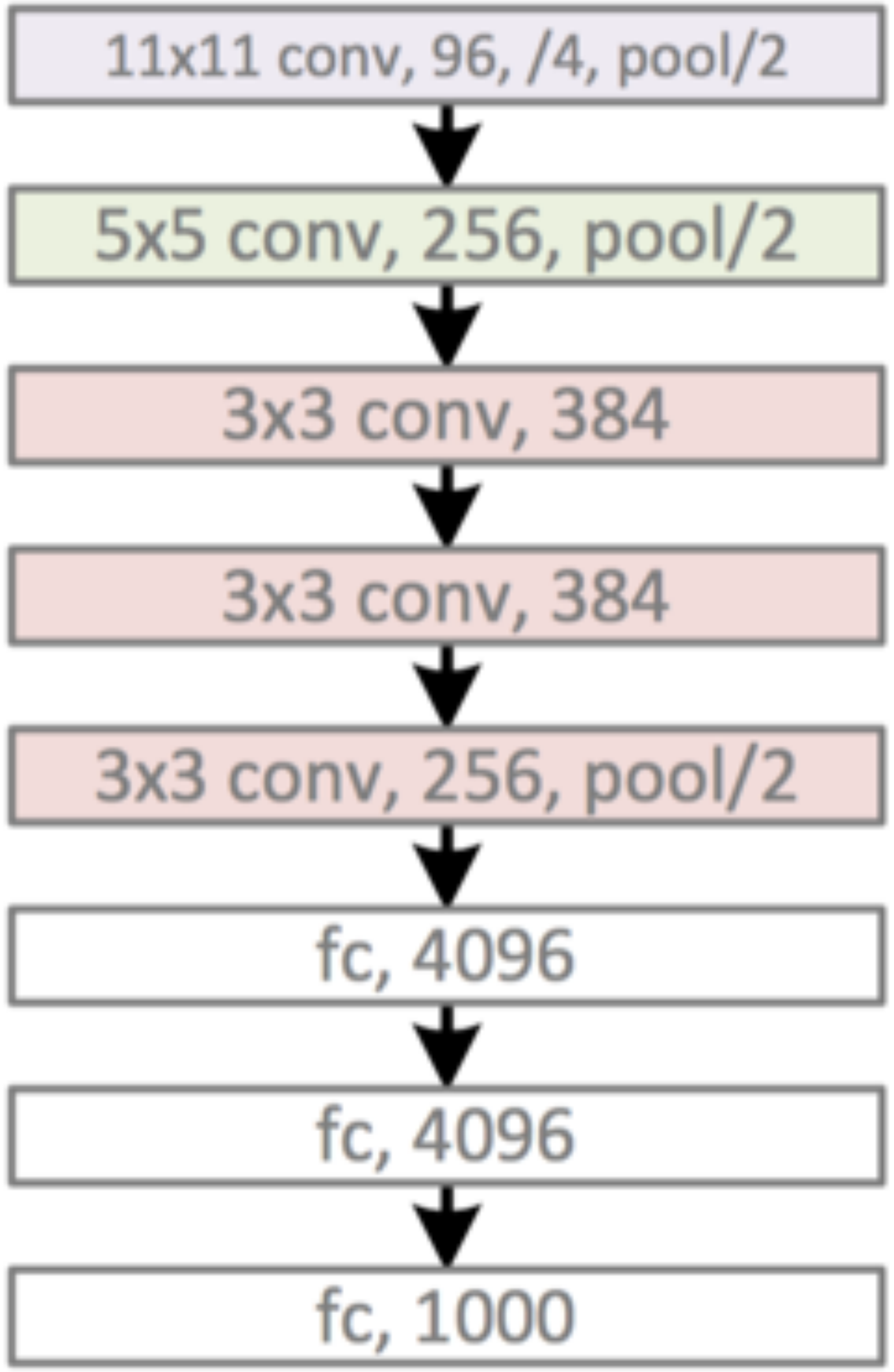
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

Source: Isola, Torralba, Freeman

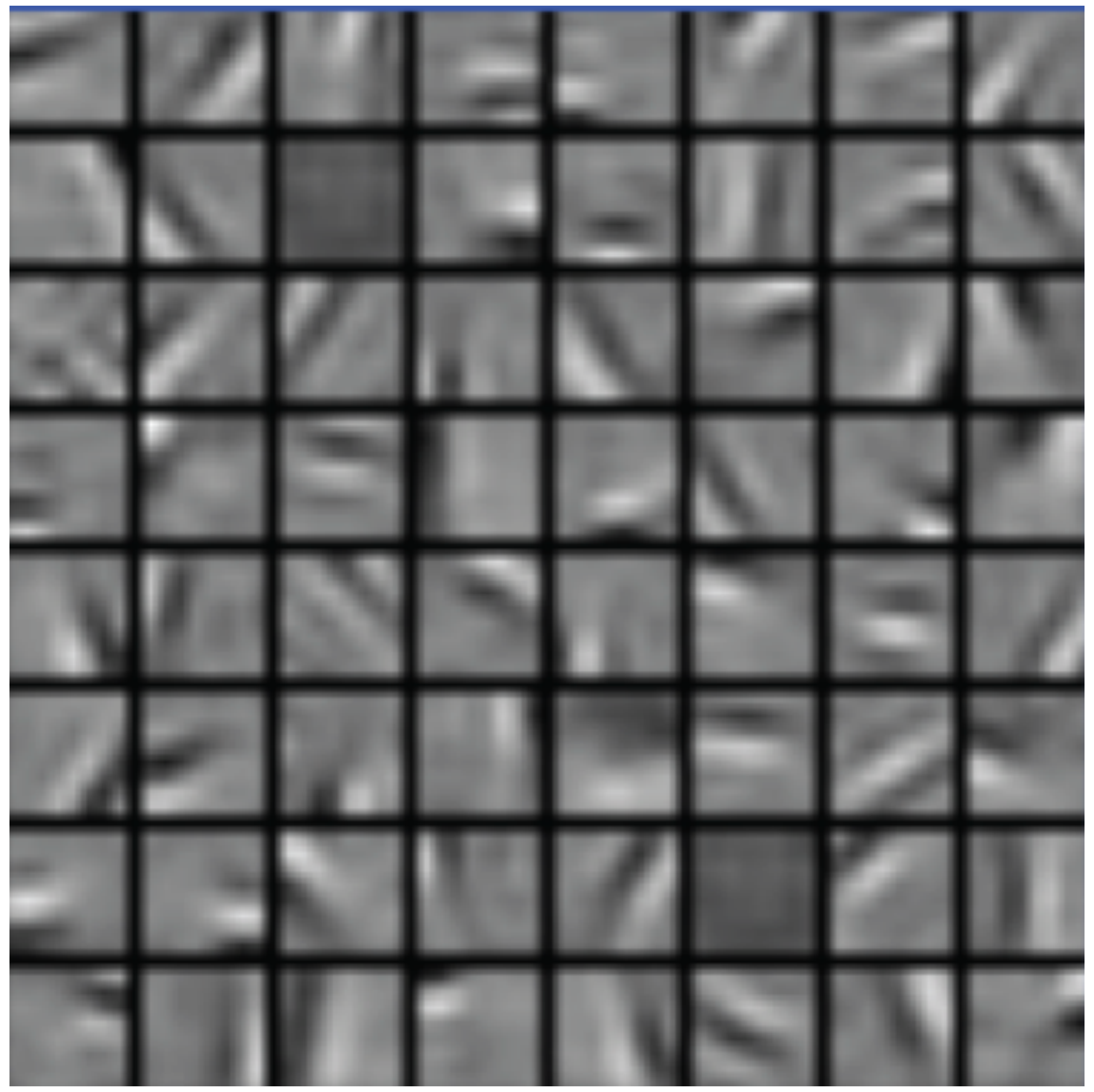


What filters are learned?

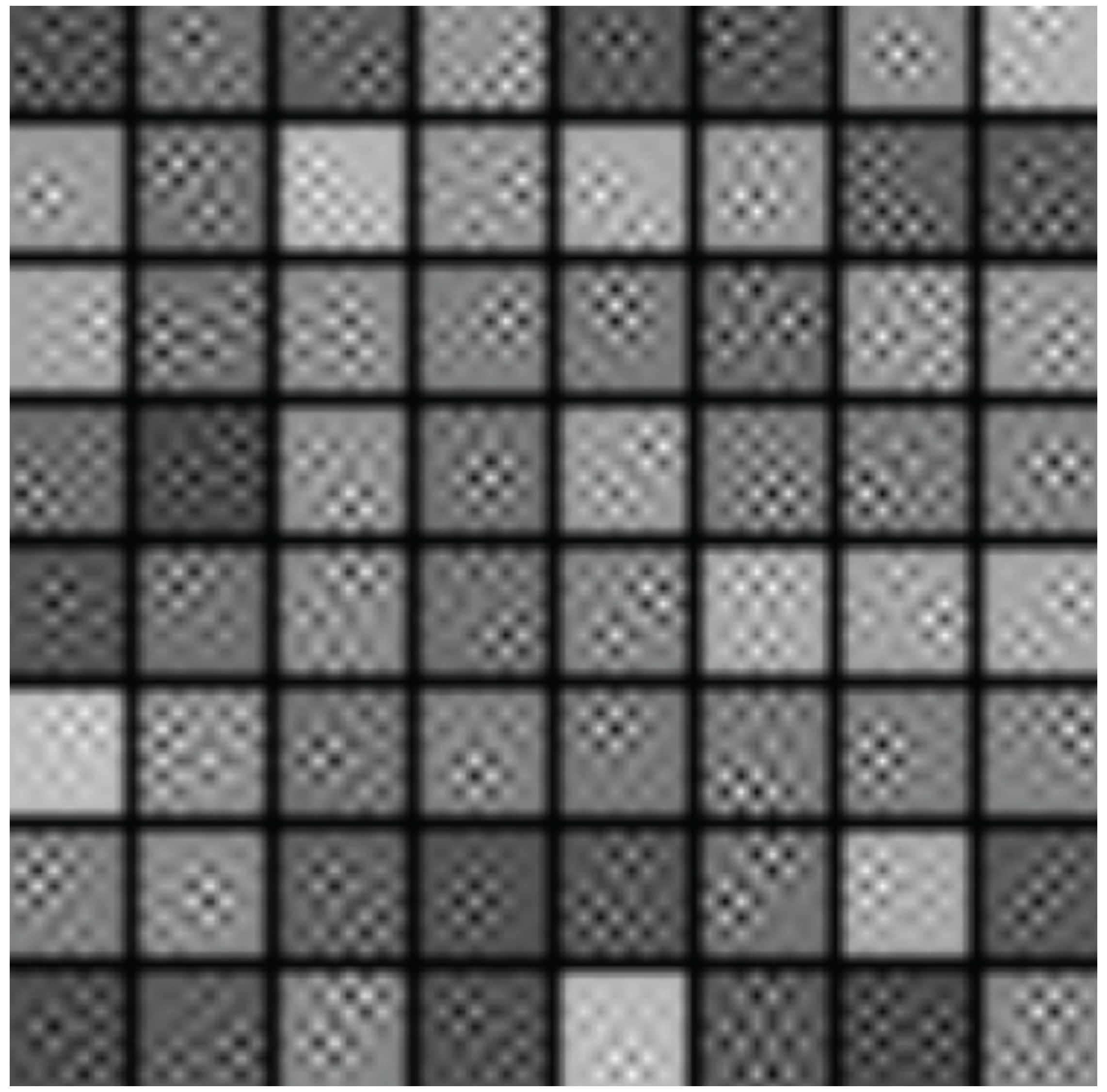
Source: Isola, Torralba, Freeman

What filters are learned?

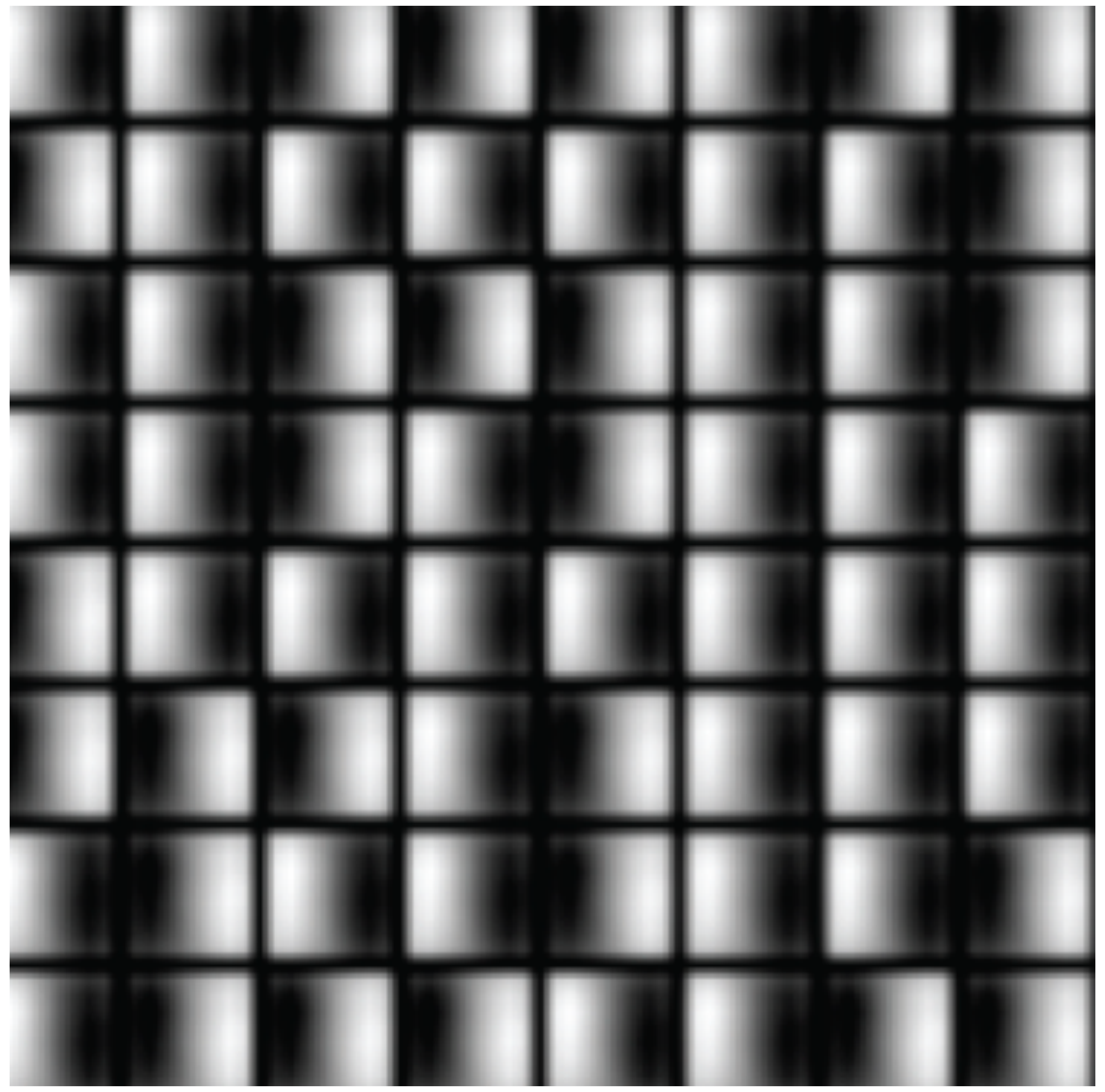
A



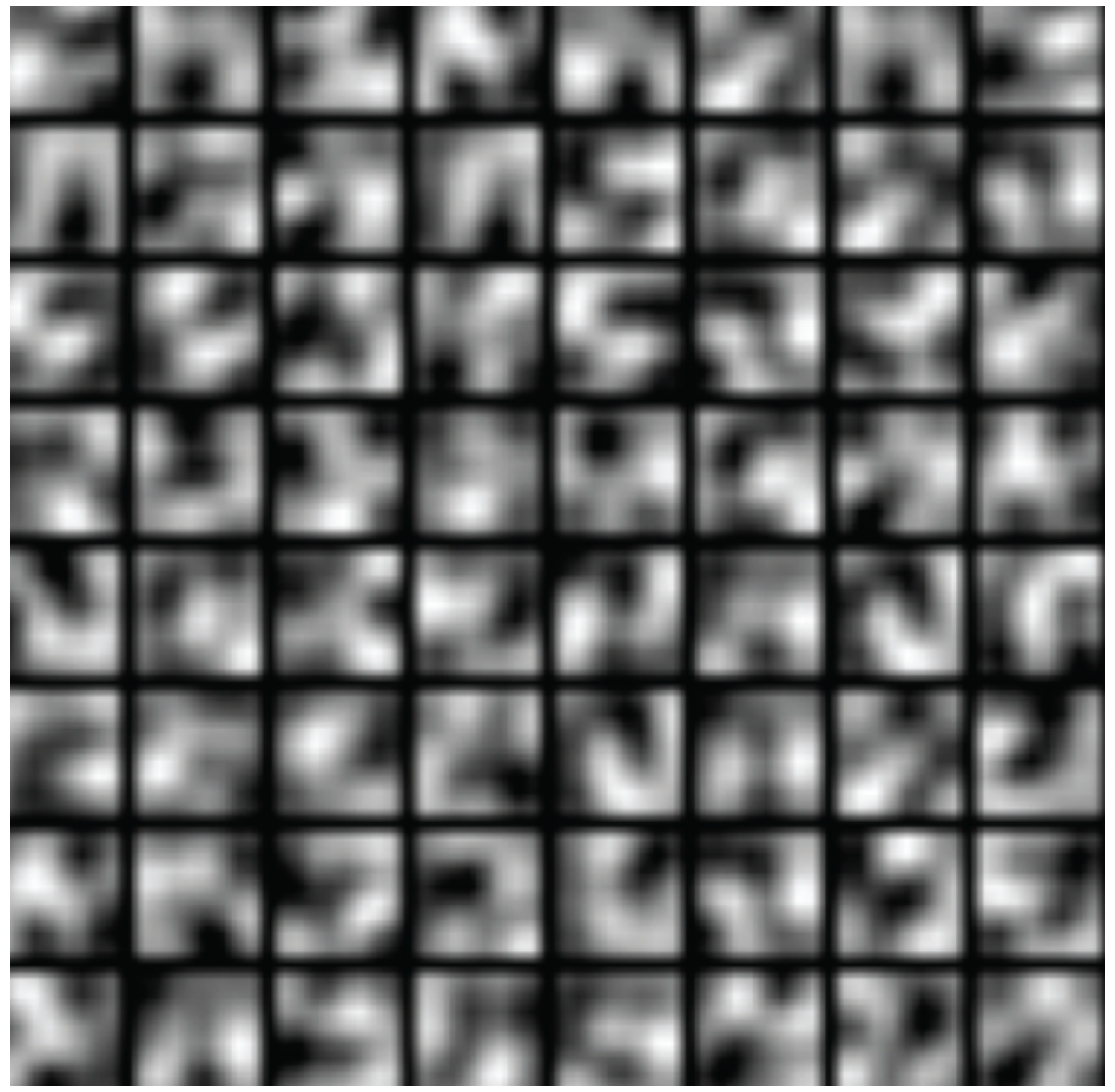
B



C

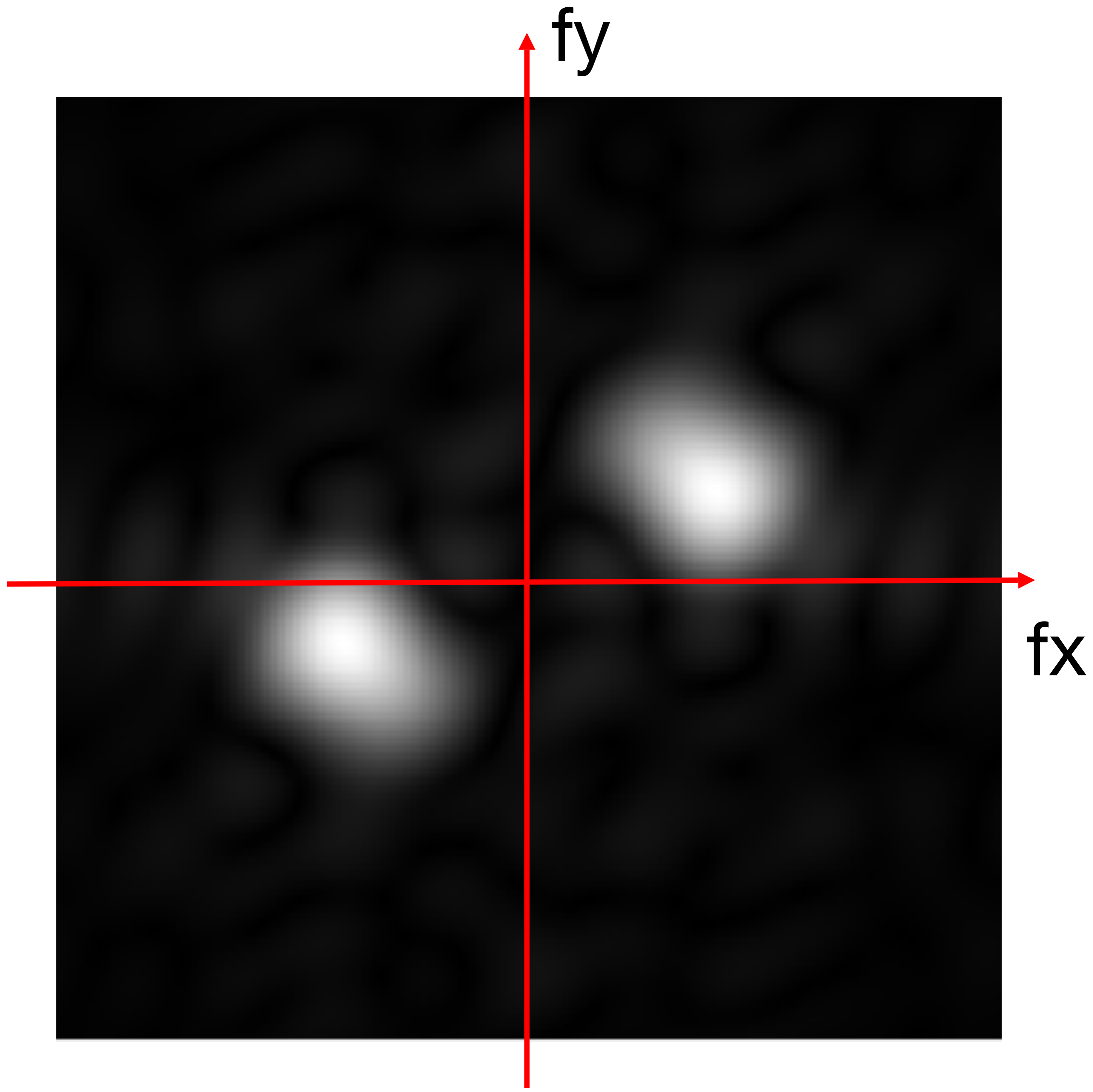


D



Source: Marc'Aurelio Ranzato & Isola, Torralba, Freeman

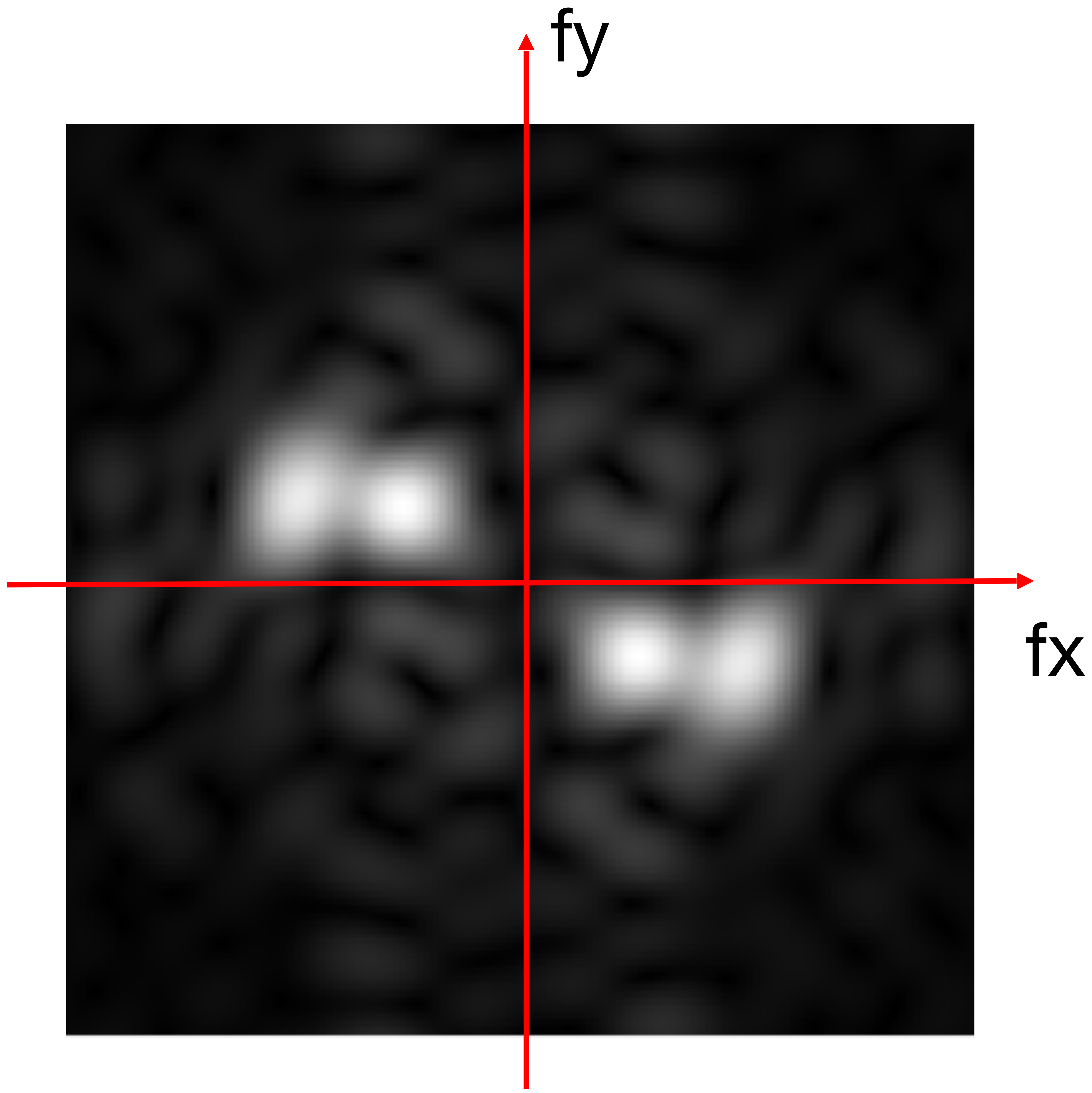
Filters in first layer



11x11 convolution kernel
(3 color channels)

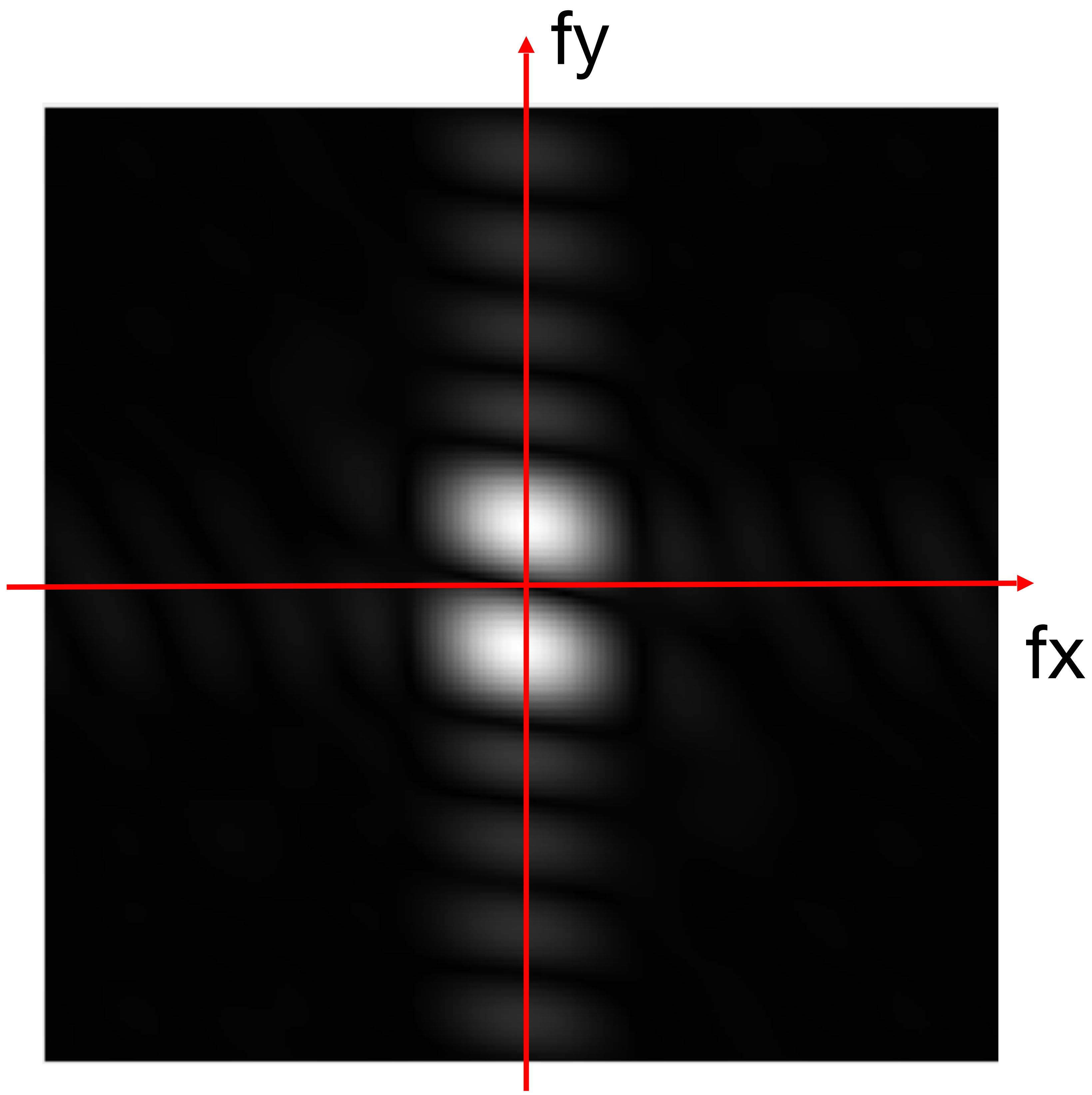
Source: Isola, Torralba, Freeman

Filters in first layer



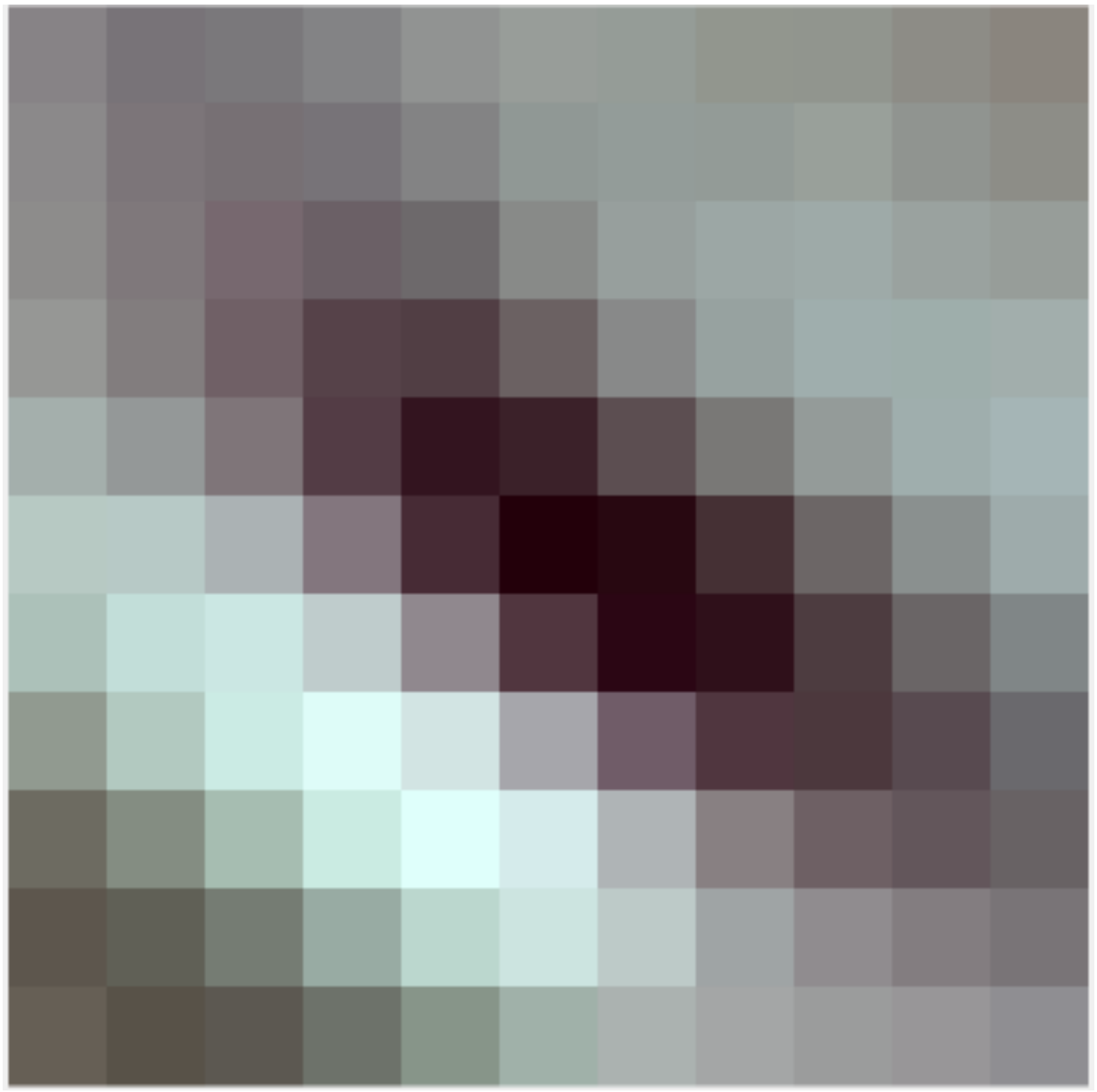
Source: Isola, Torralba, Freeman

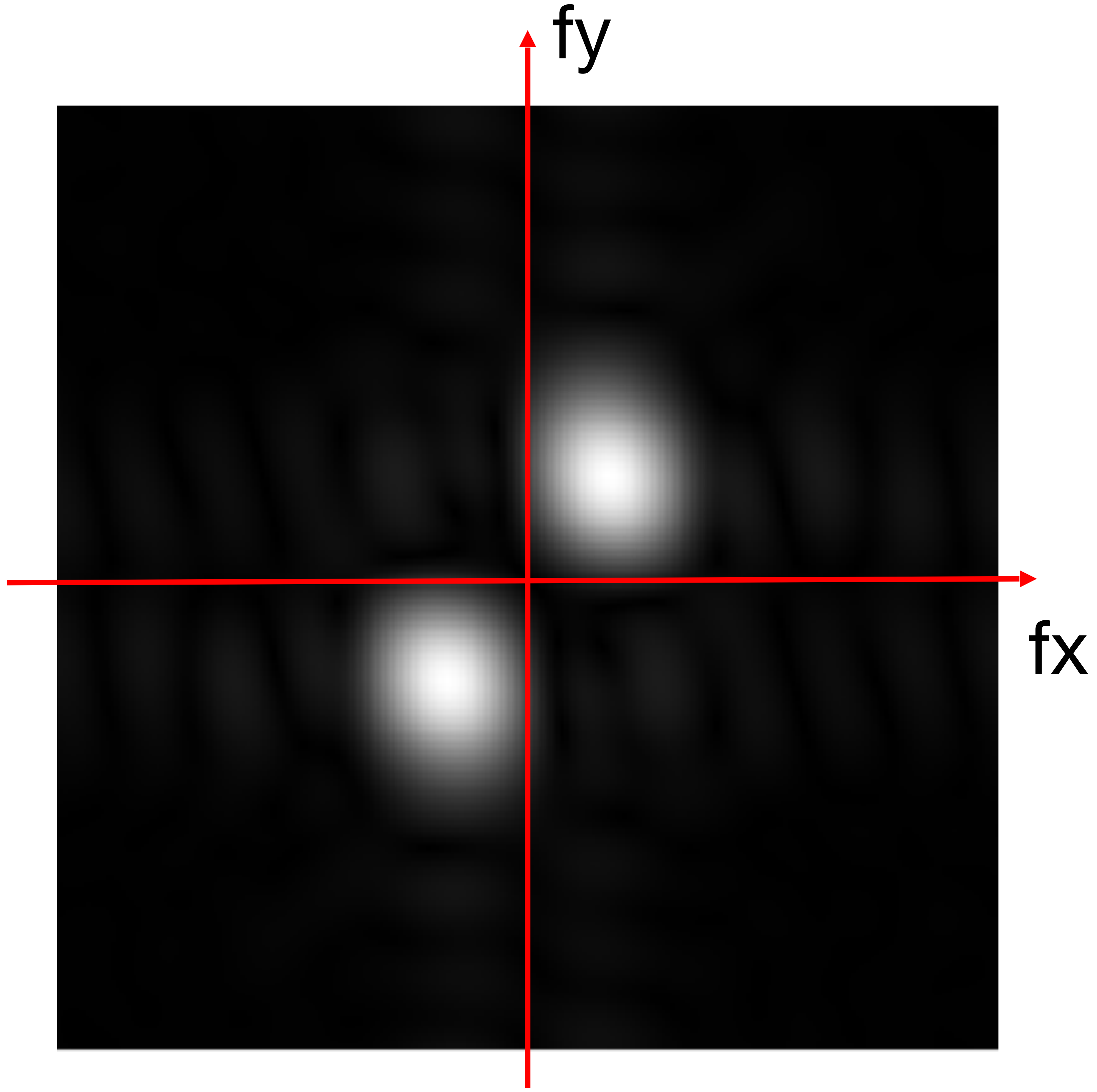
Filters in first layer



Source: Isola, Torralba, Freeman

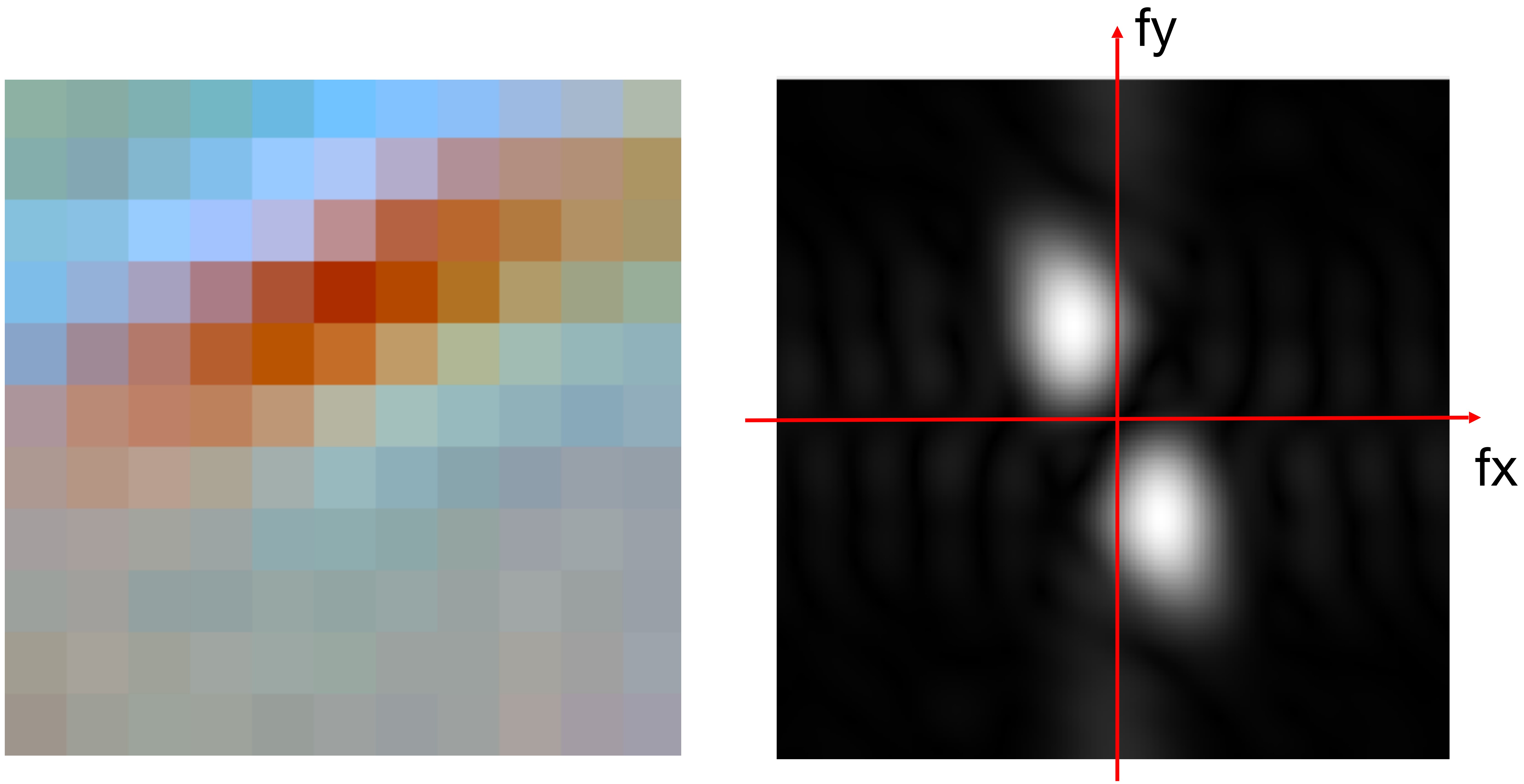
Filters in first layer





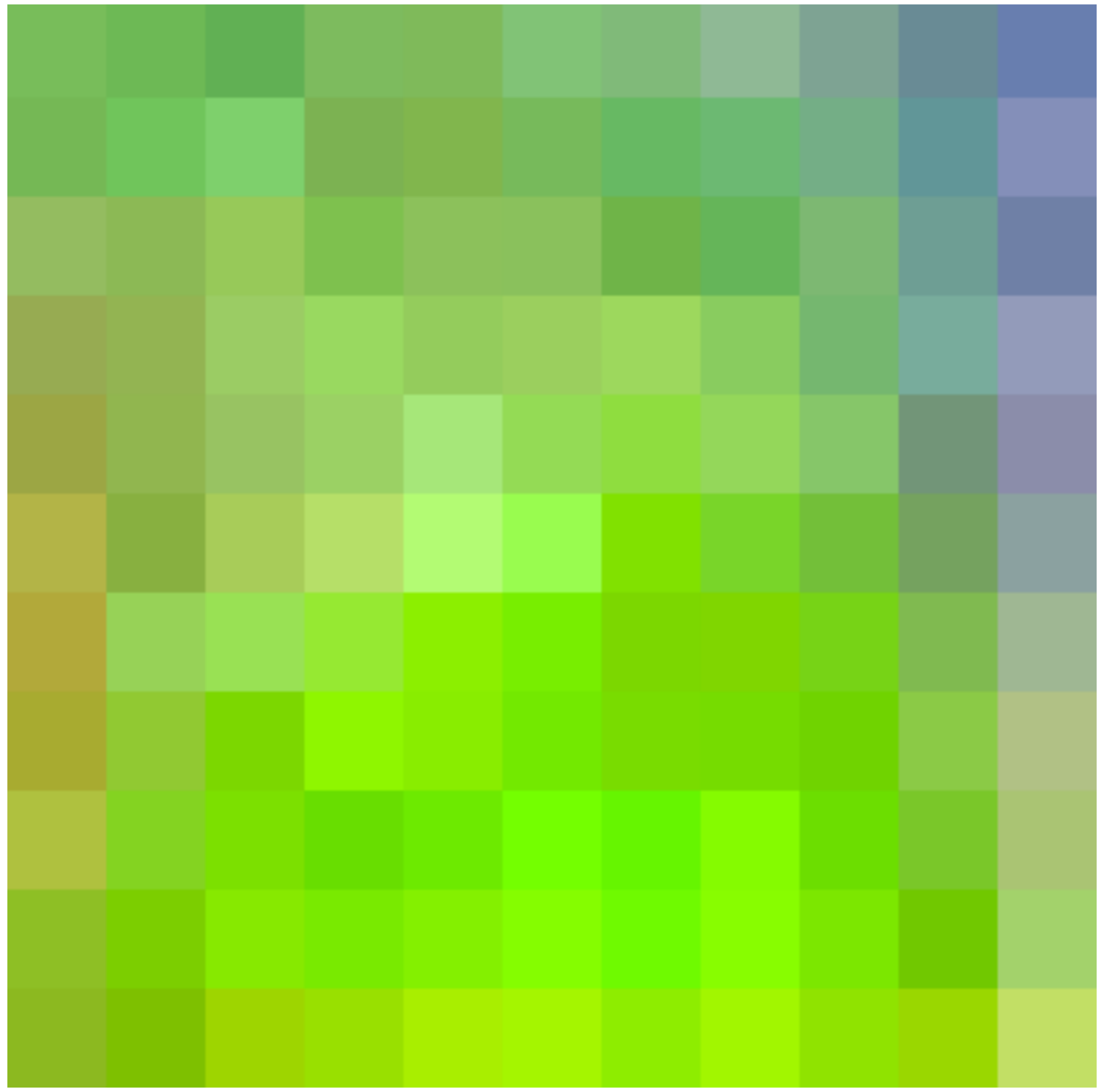
Source: Isola, Torralba, Freeman

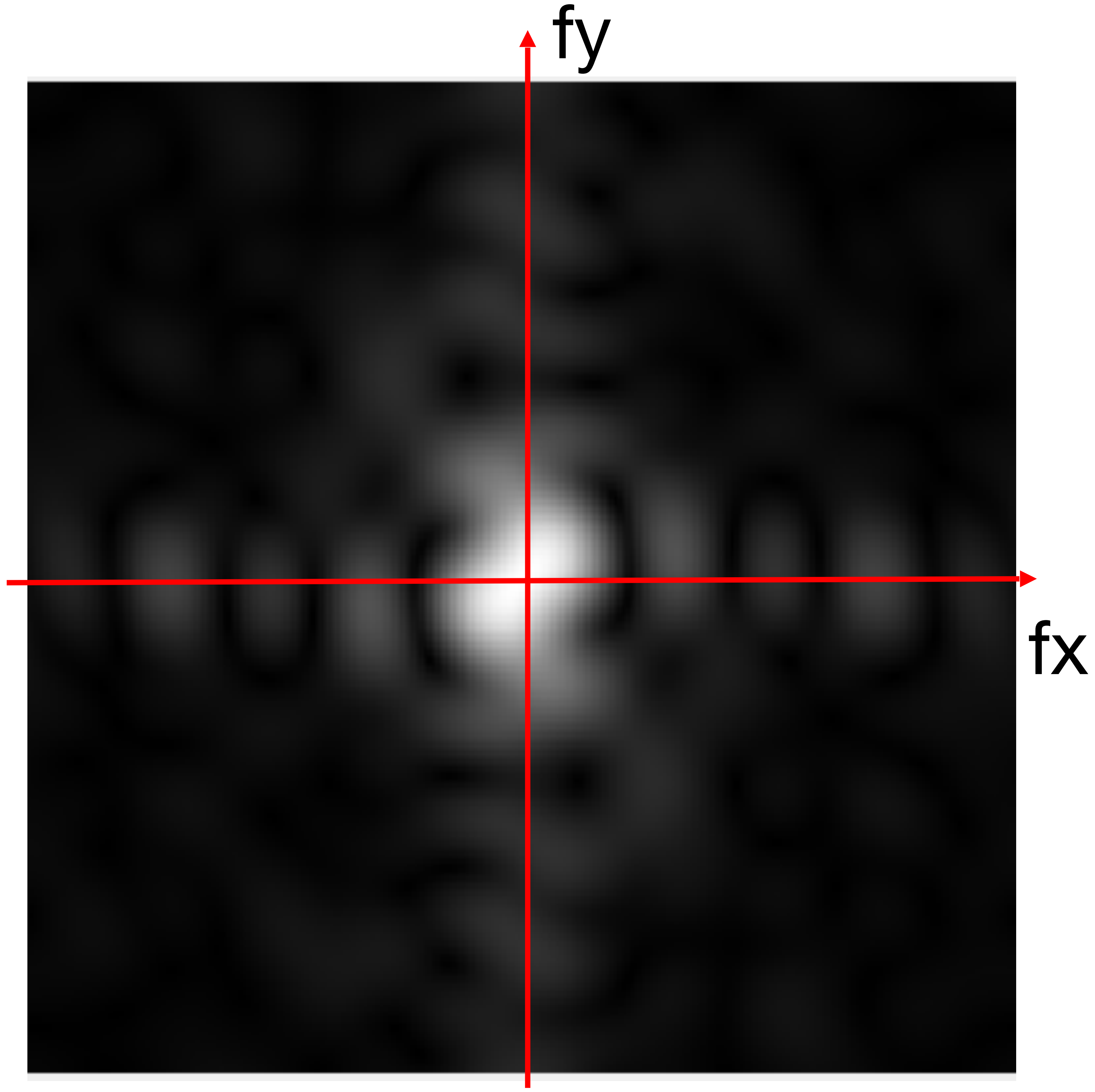
Filters in first layer



Source: Isola, Torralba, Freeman

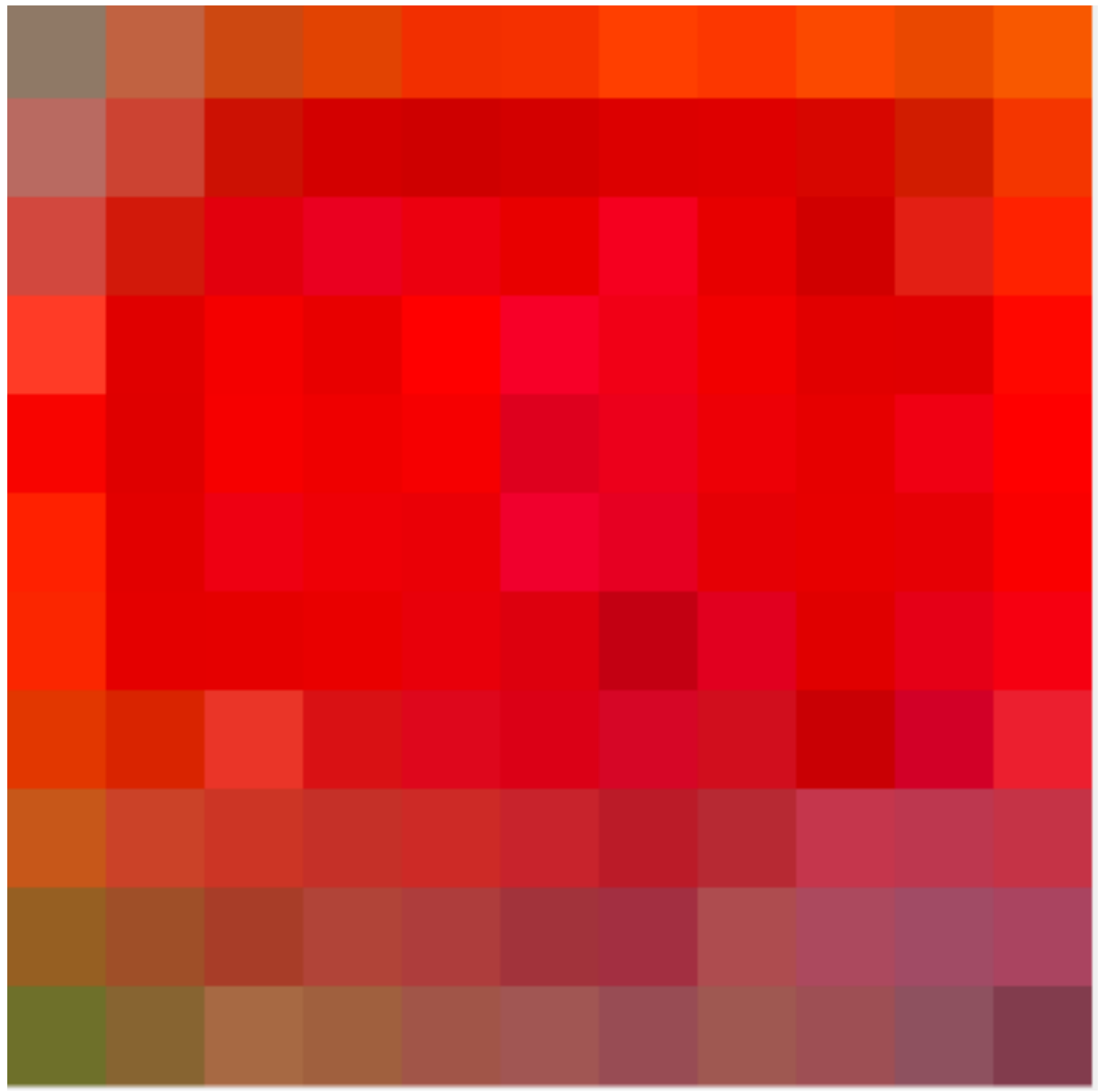
Filters in first layer

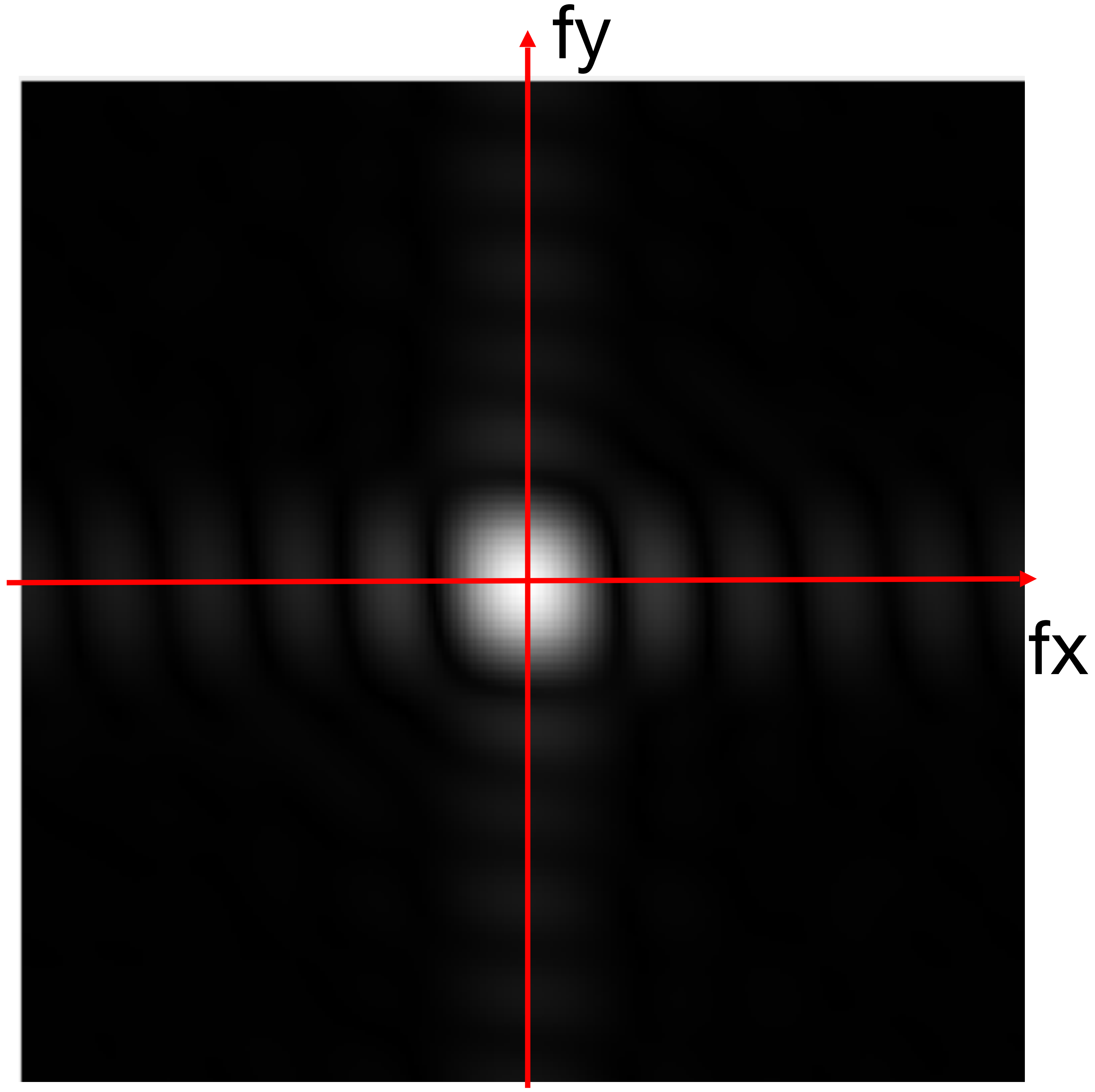




Source: Isola, Torralba, Freeman

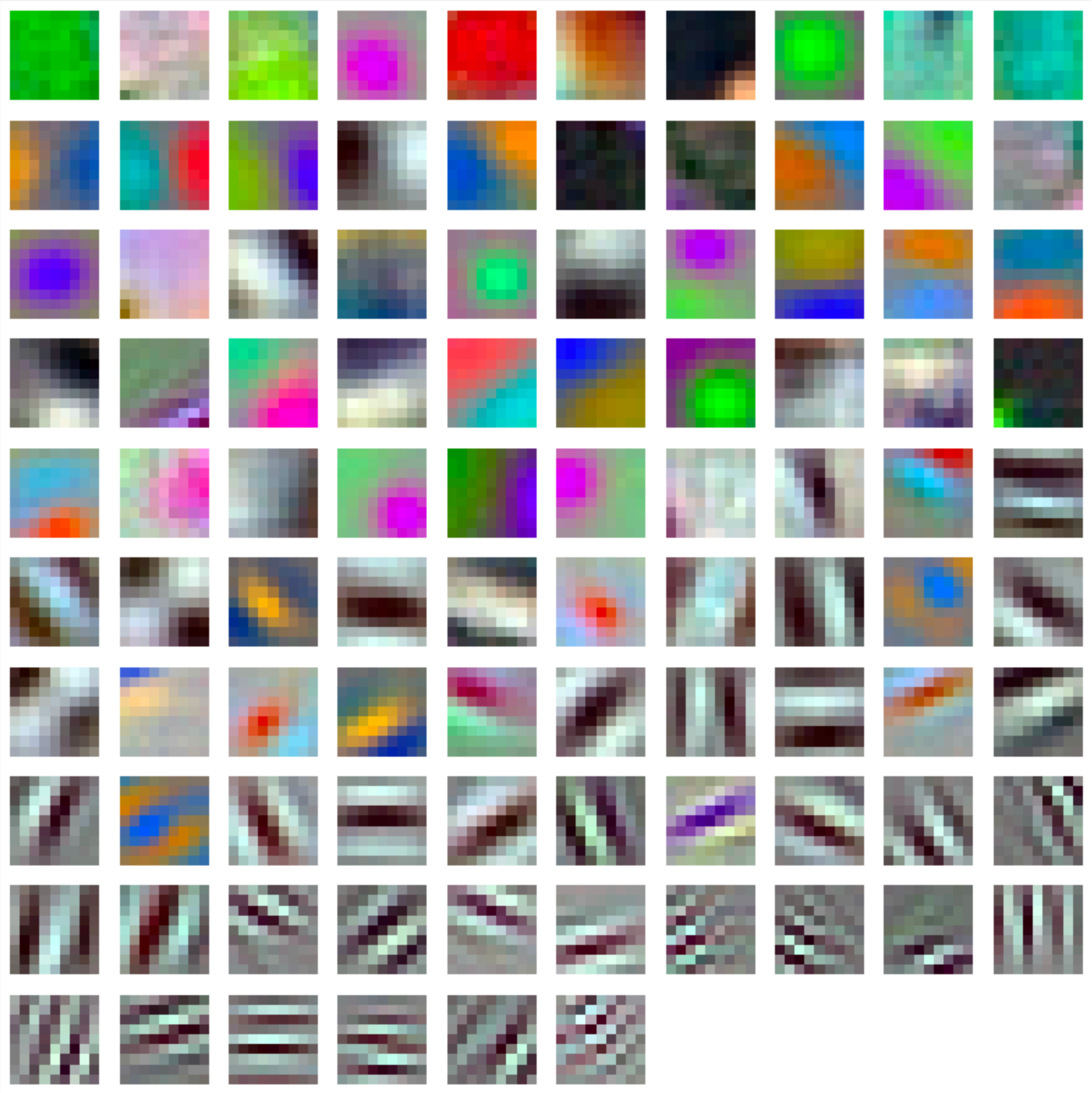
Filters in first layer



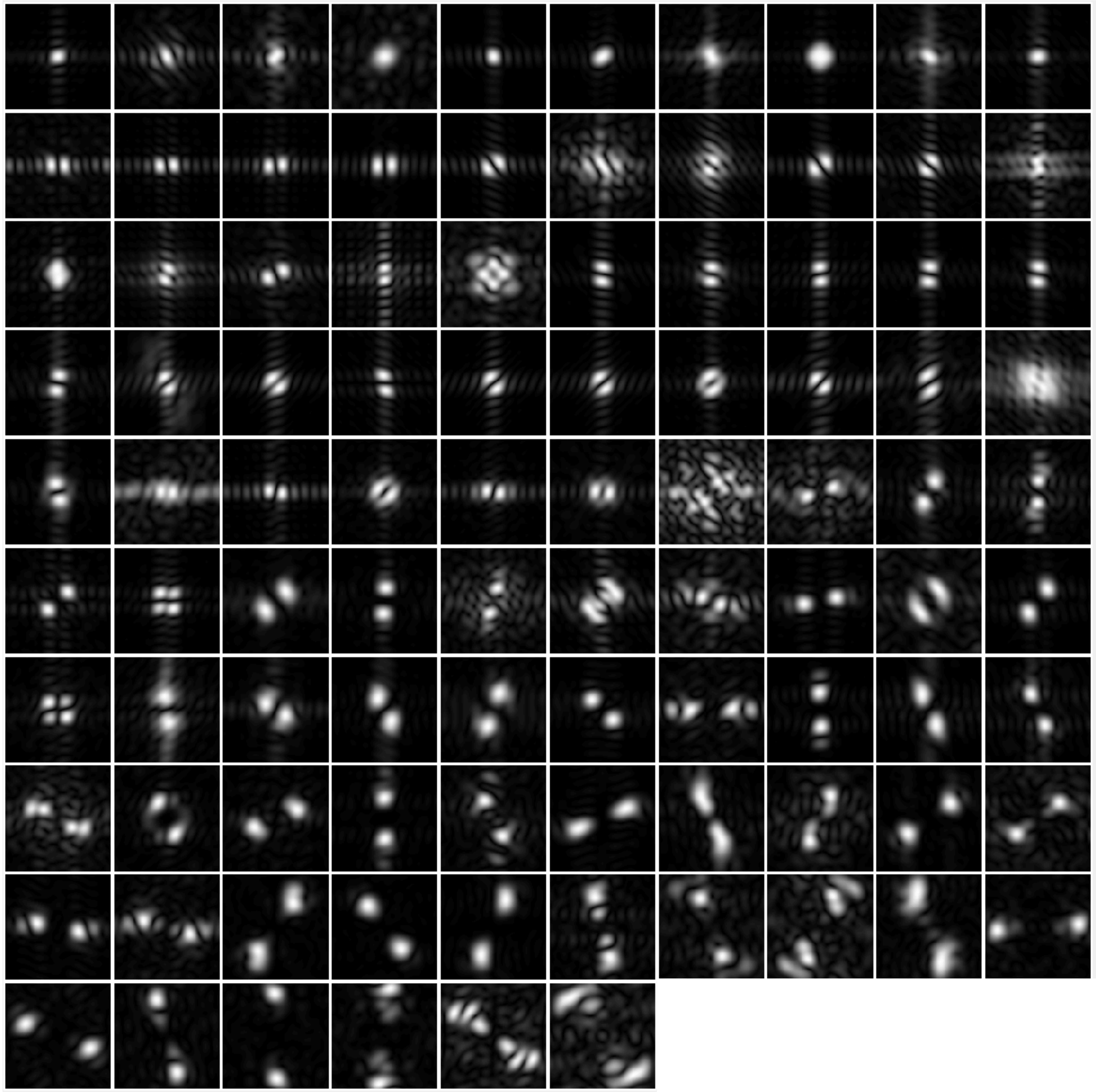


Source: Isola, Torralba, Freeman

Filters in first layer

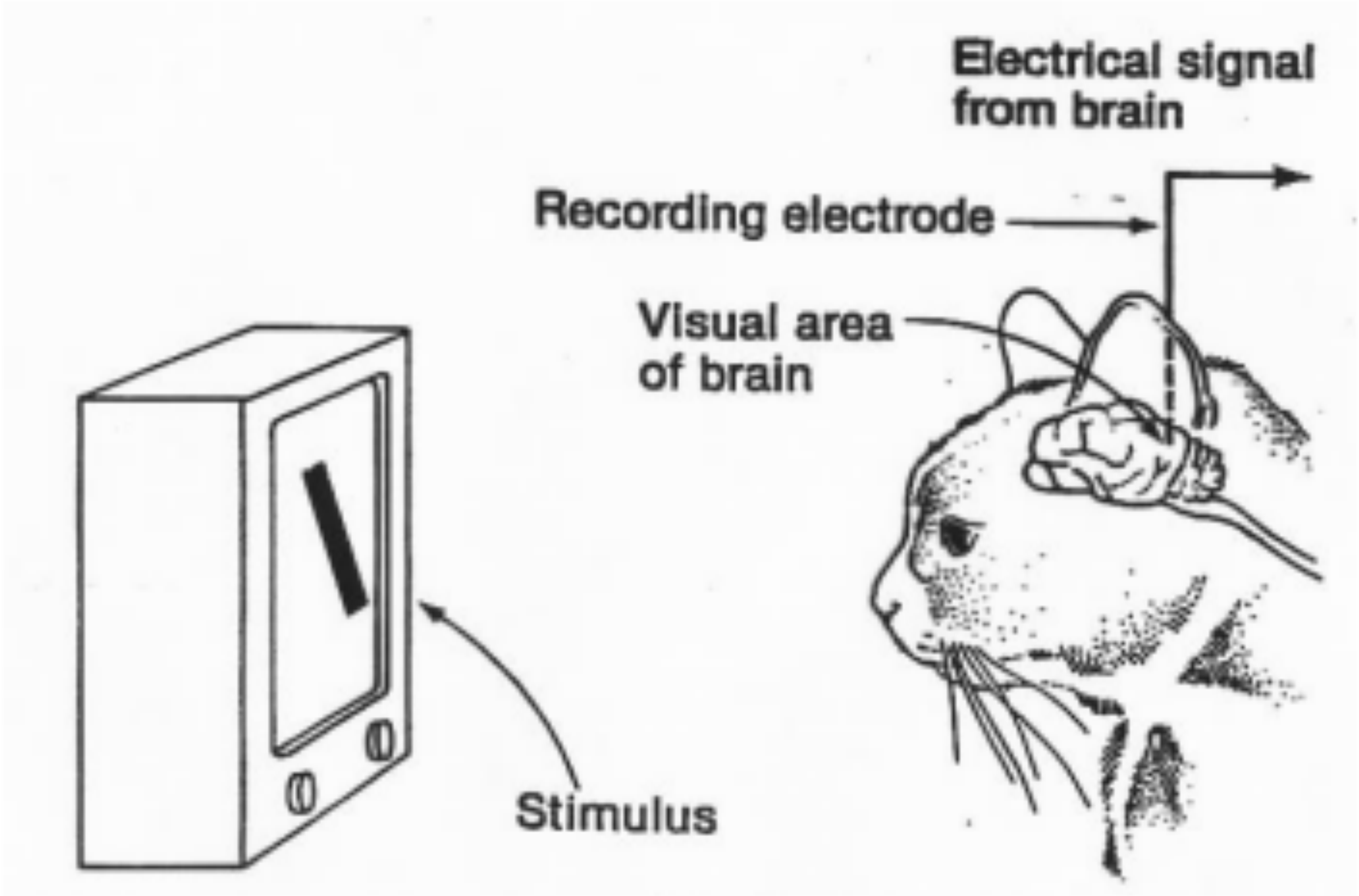


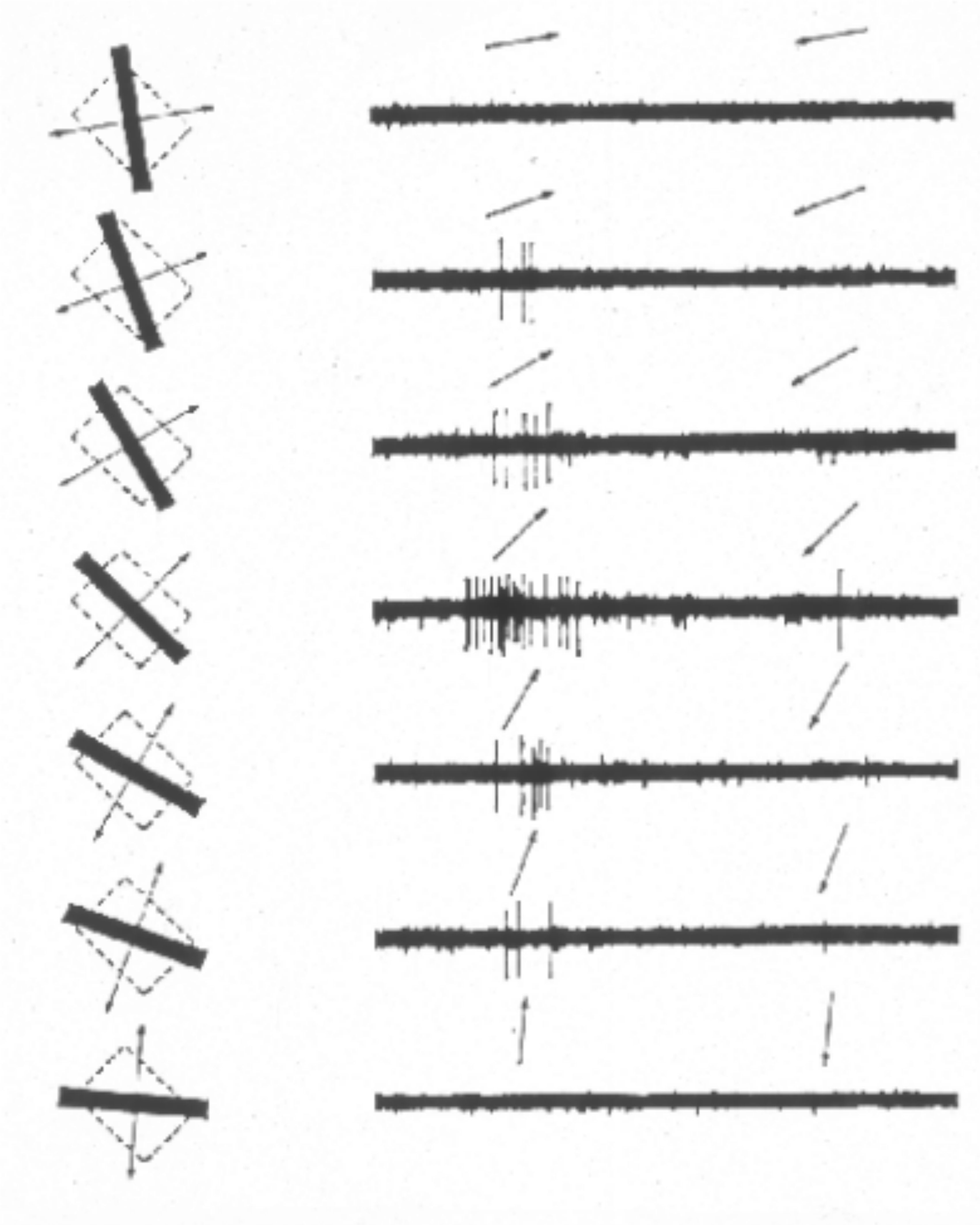
96 Units in conv1



Source: Isola, Torralba, Freeman

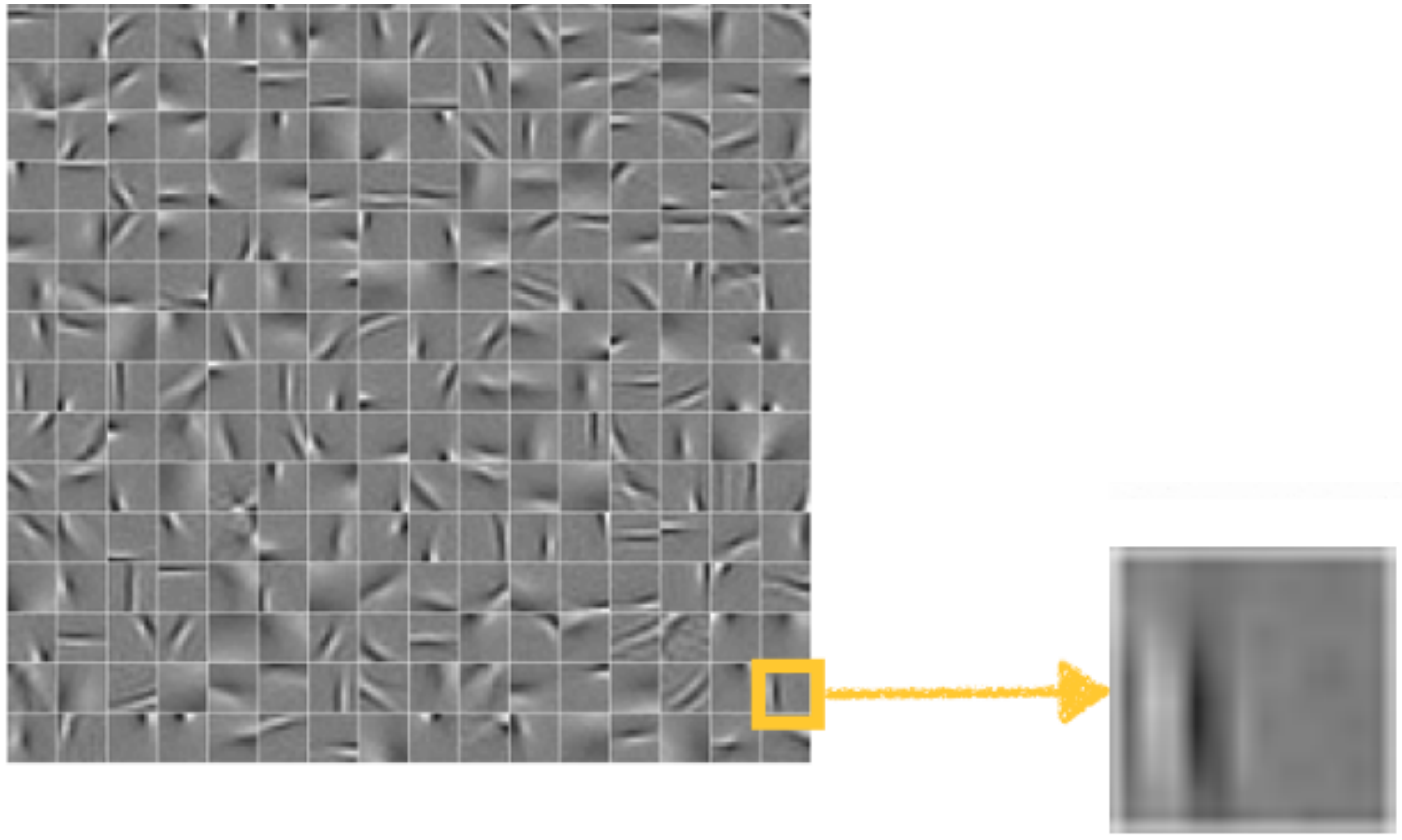
[Hubel and Wiesel 59]



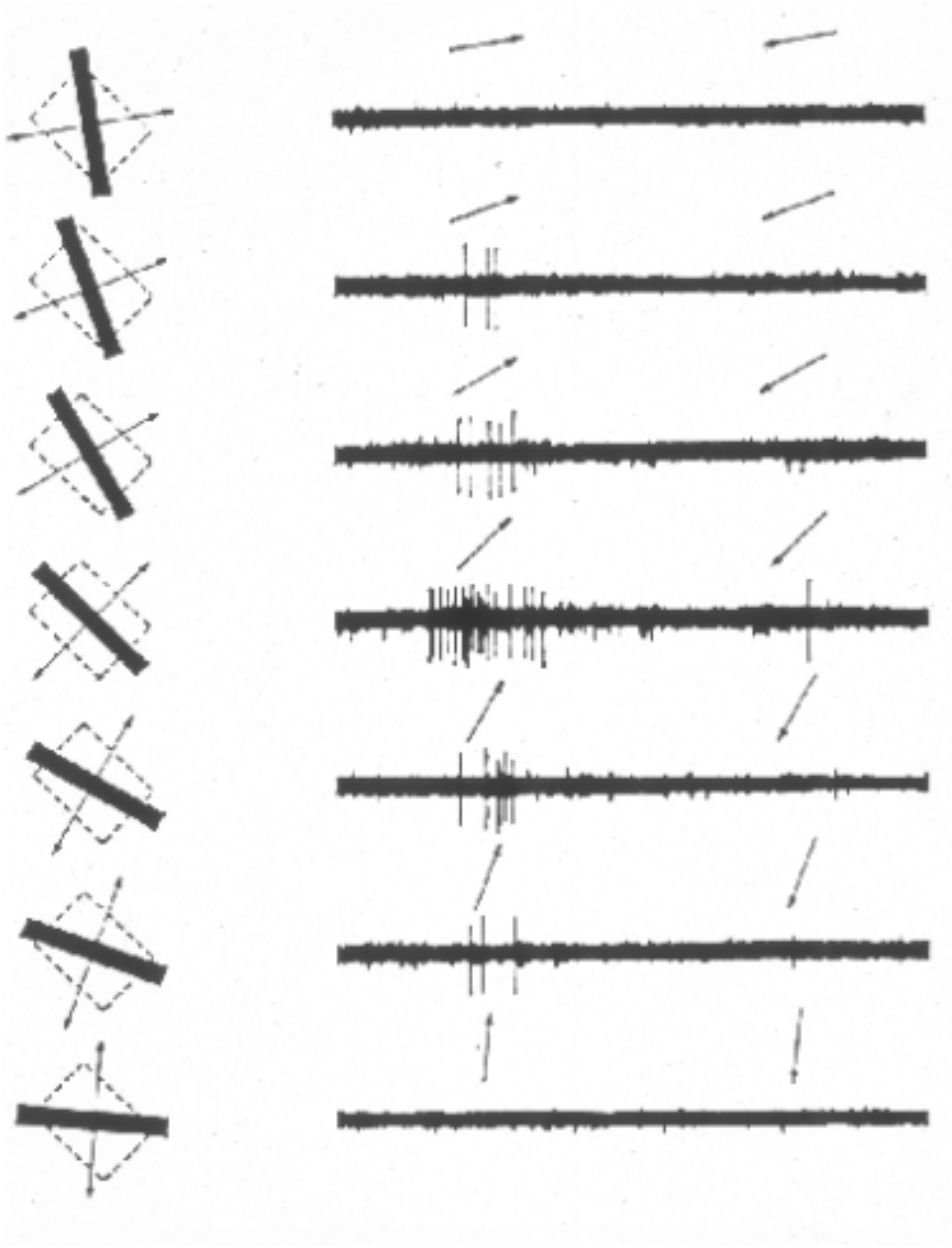


Source: Andrea Vedaldi

A good fit to the experimental results

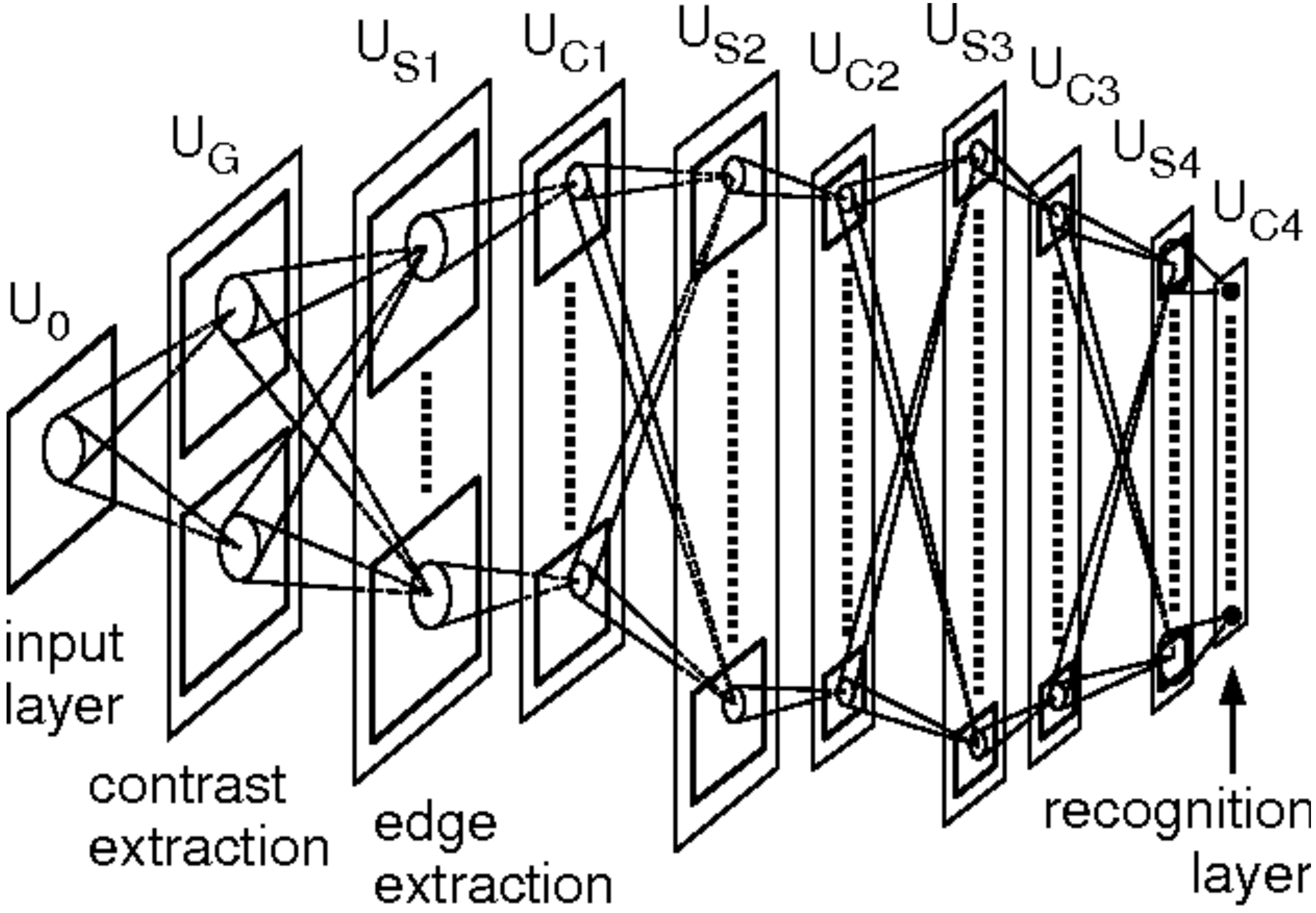


Oriented filters



Source: Andrea Vedaldi

History: Neocognitron

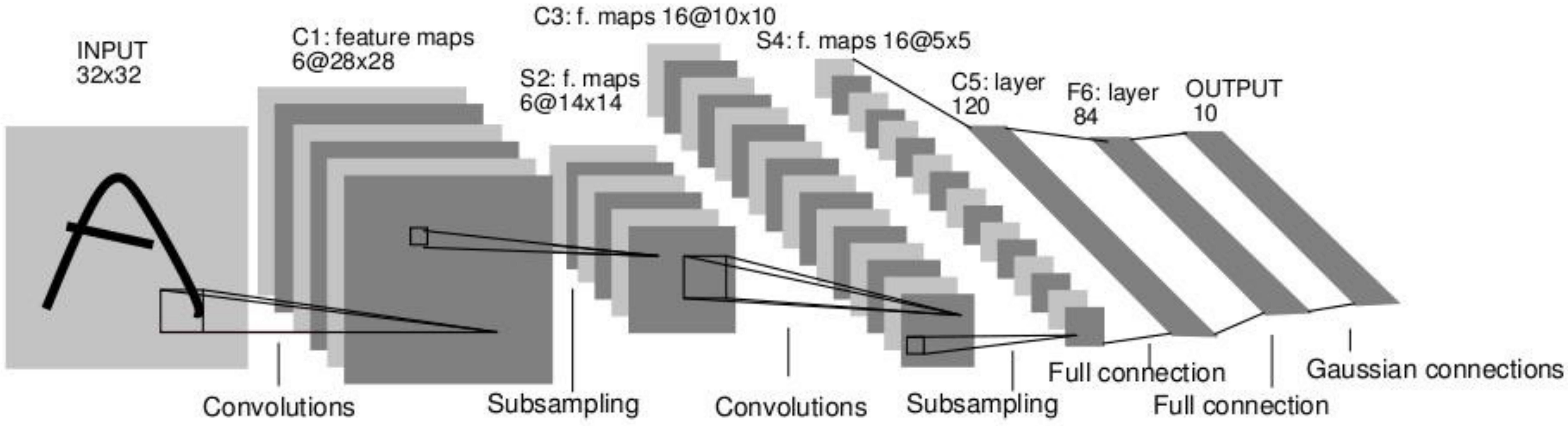


K. Fukushima, 1980s

Figure source: Wikipedia

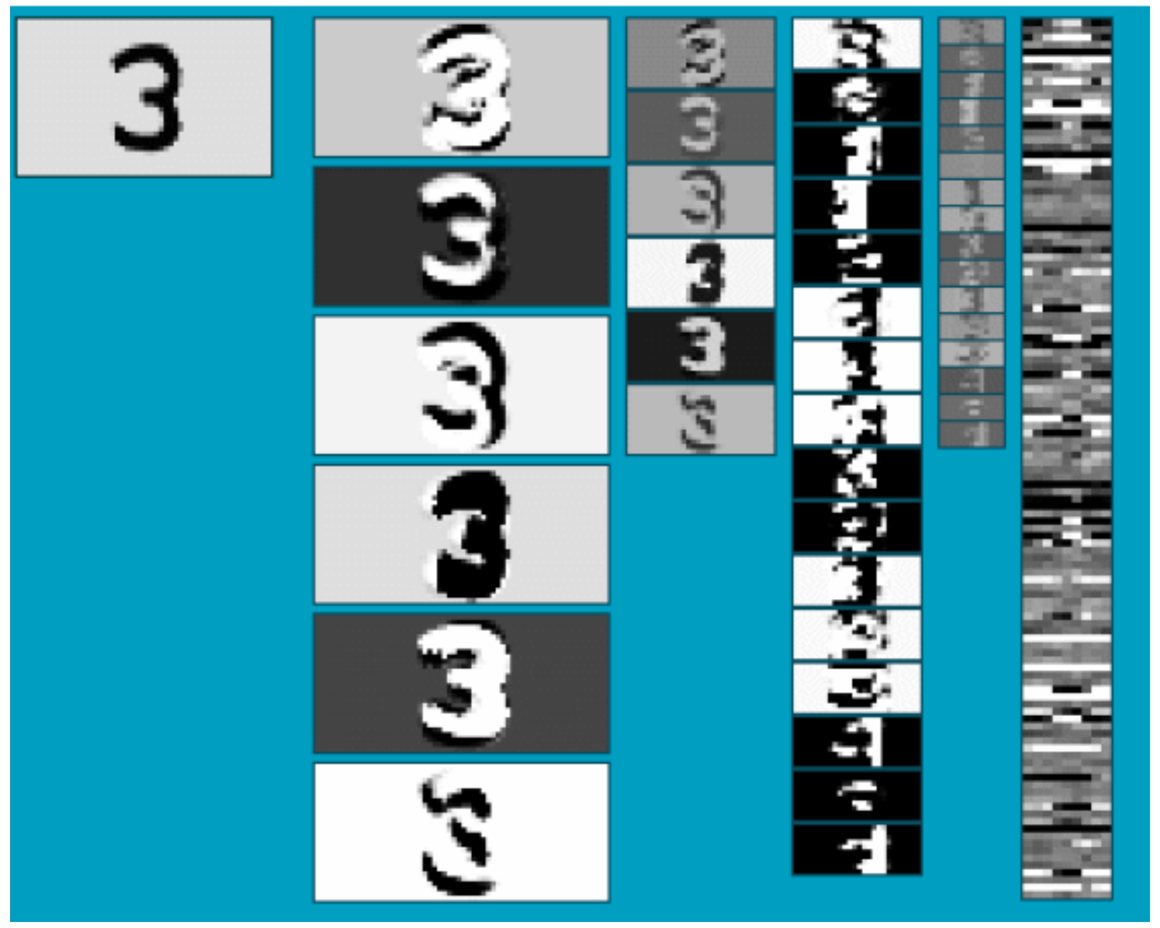
Source: S. Lazebnik

History: LeNet



[LeCun, Bottou, Bengio, Haffner. "Gradient-based learning applied to document recognition", 1998]

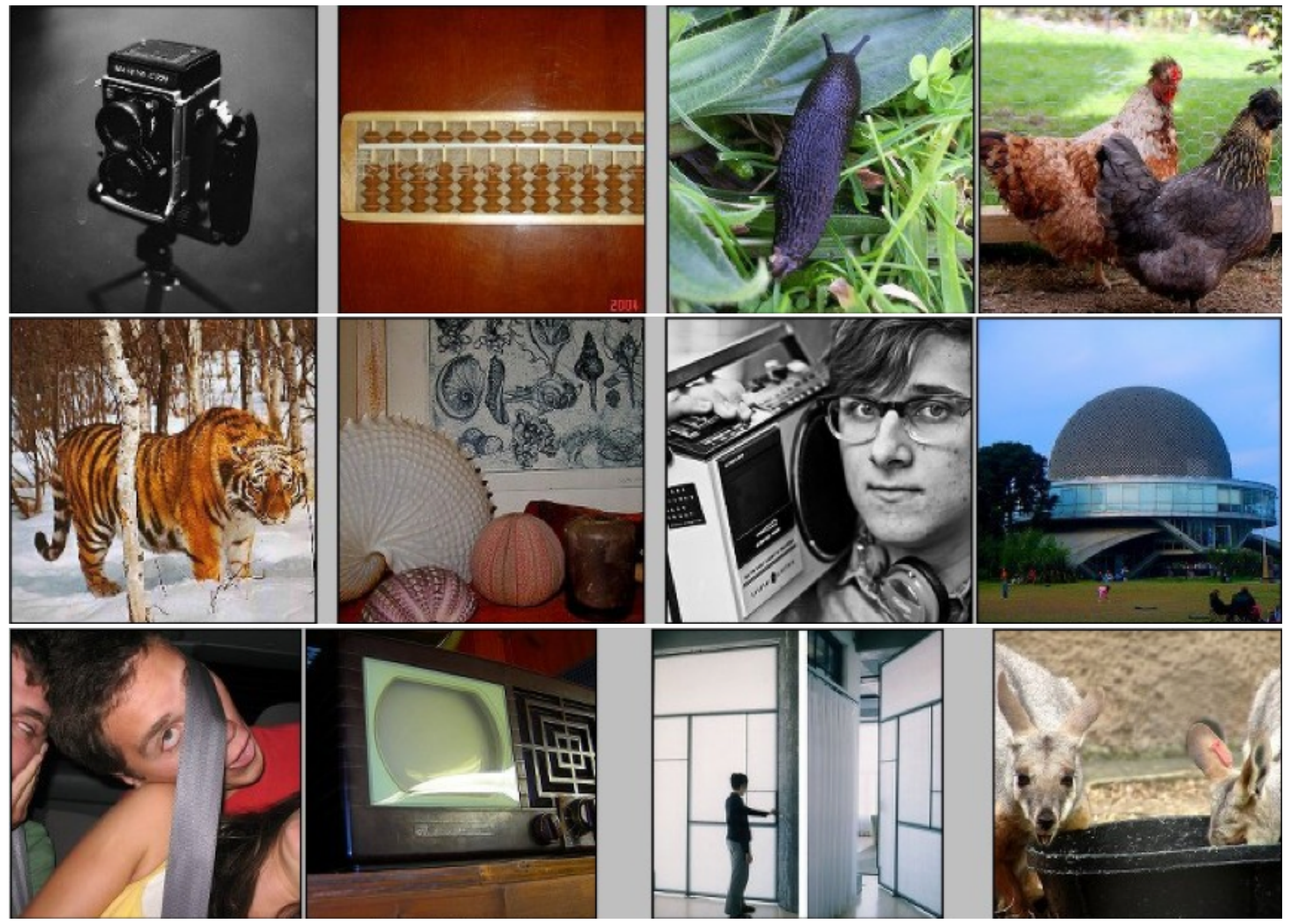
- Neocognitron + backpropagation
- Average pooling
- Sigmoid or tanh nonlinearity
- Fully connected layers at the end
- Trained on MNIST digit dataset with 60K training examples



Source: S. Lazebnik

ImageNet Challenge

IMAGENET



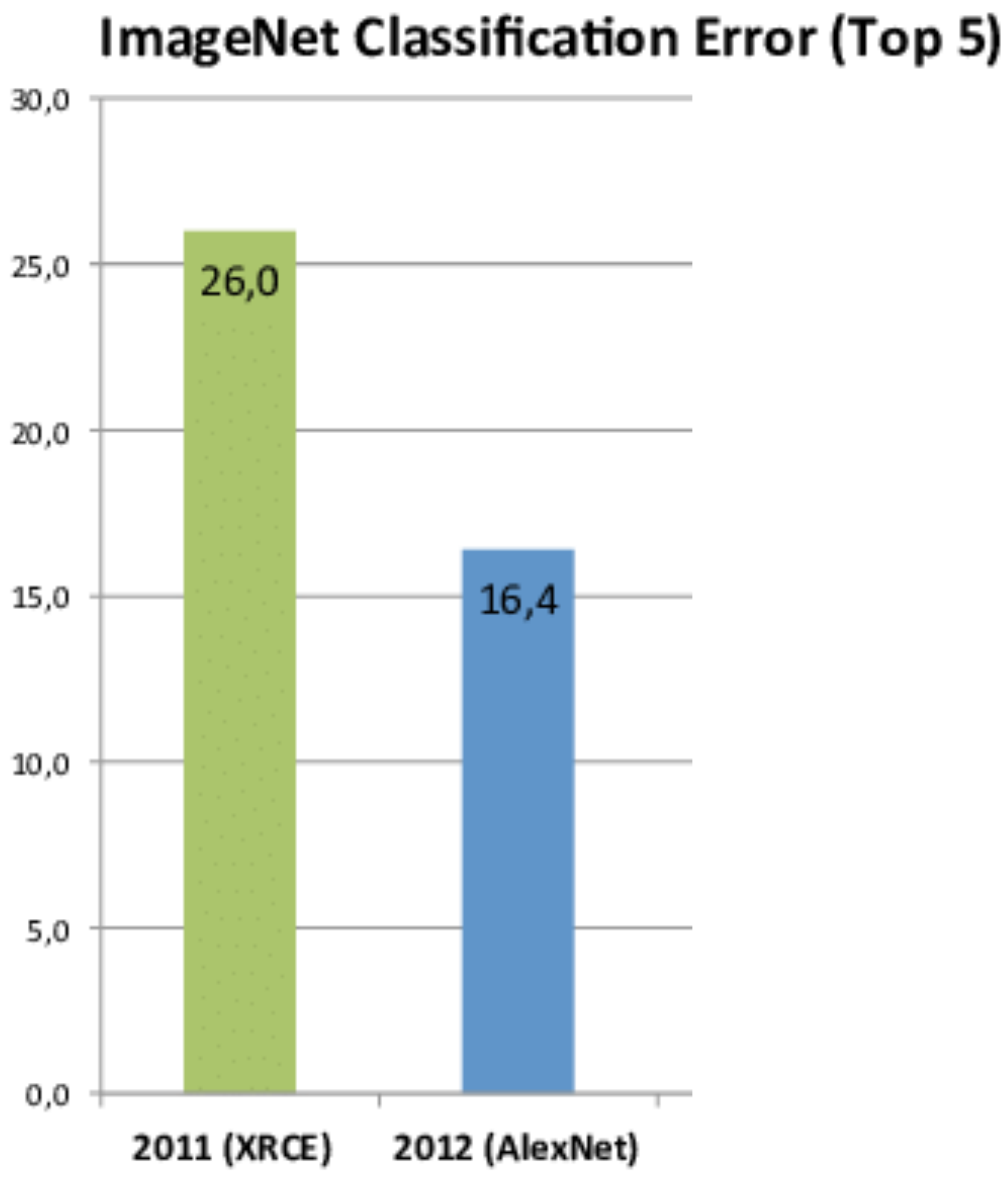
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

[Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berge, Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge”, 2015]

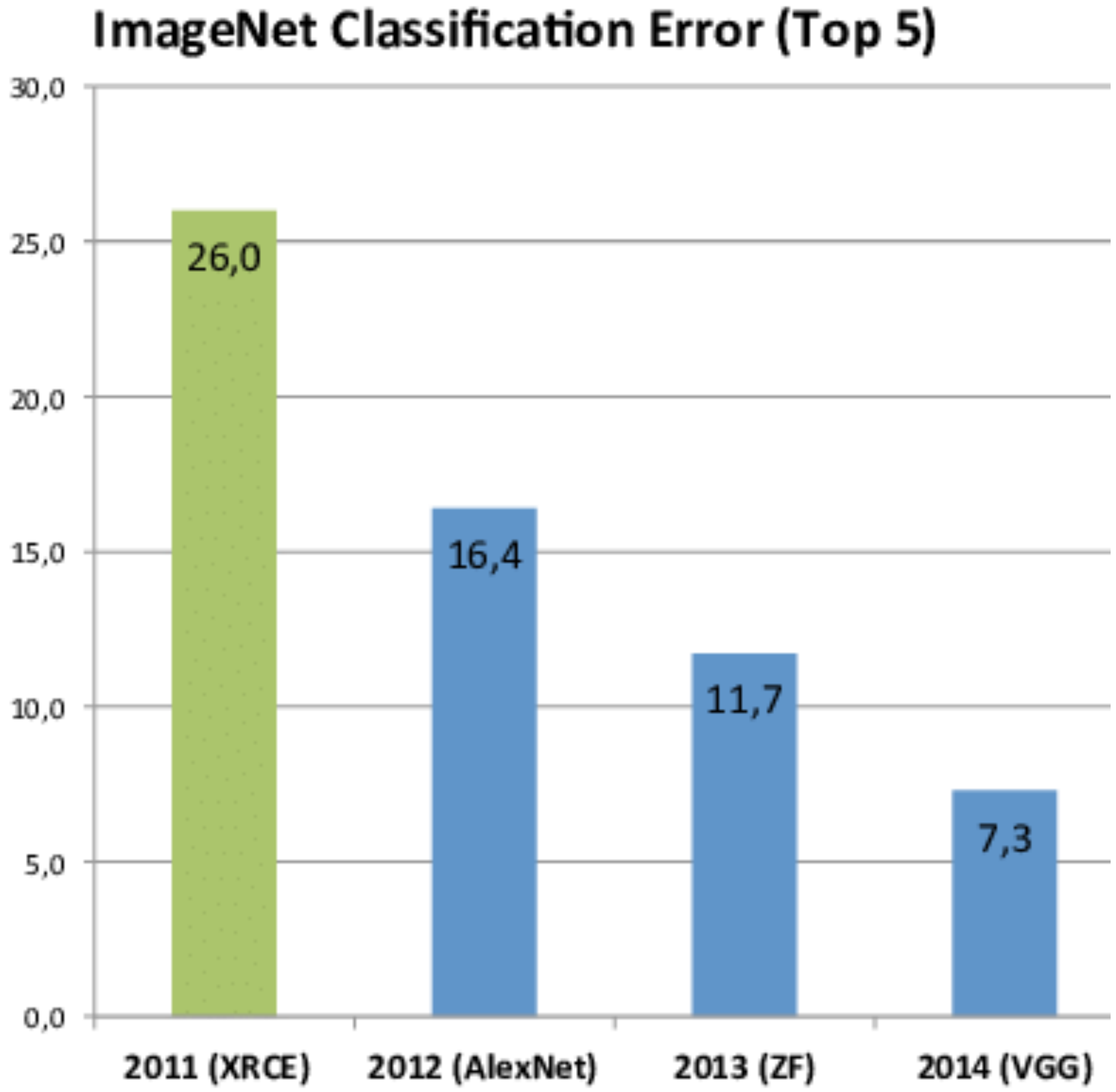
More recent networks

And advances that make them work:

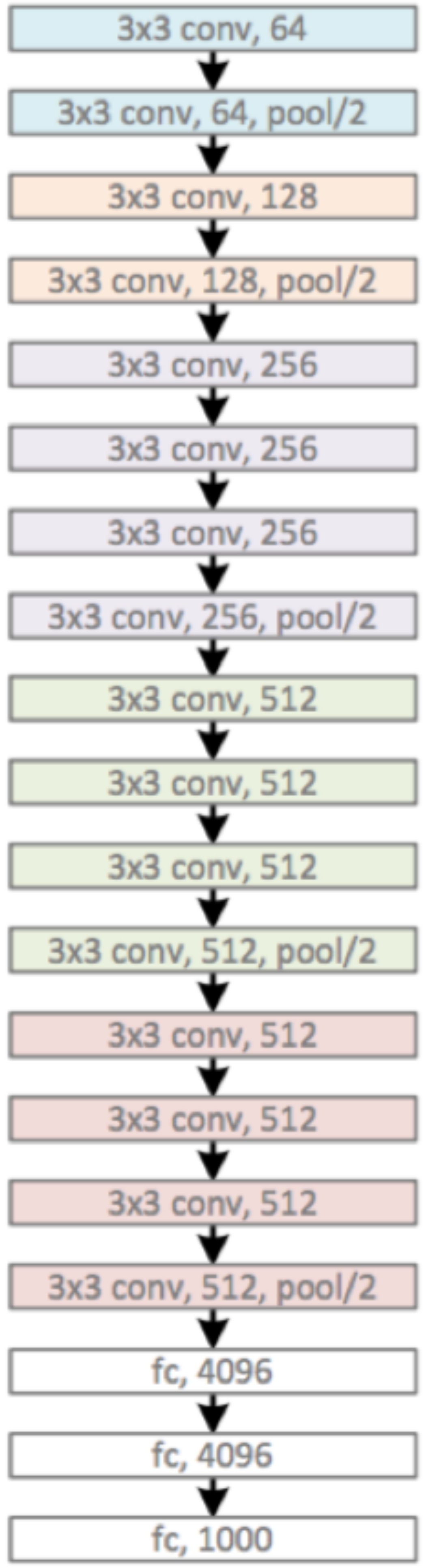
- Chaining small filters
- Residual layers
- Normalization



Source: Isola, Torralba, Freeman



2014: VGG
16 conv. layers



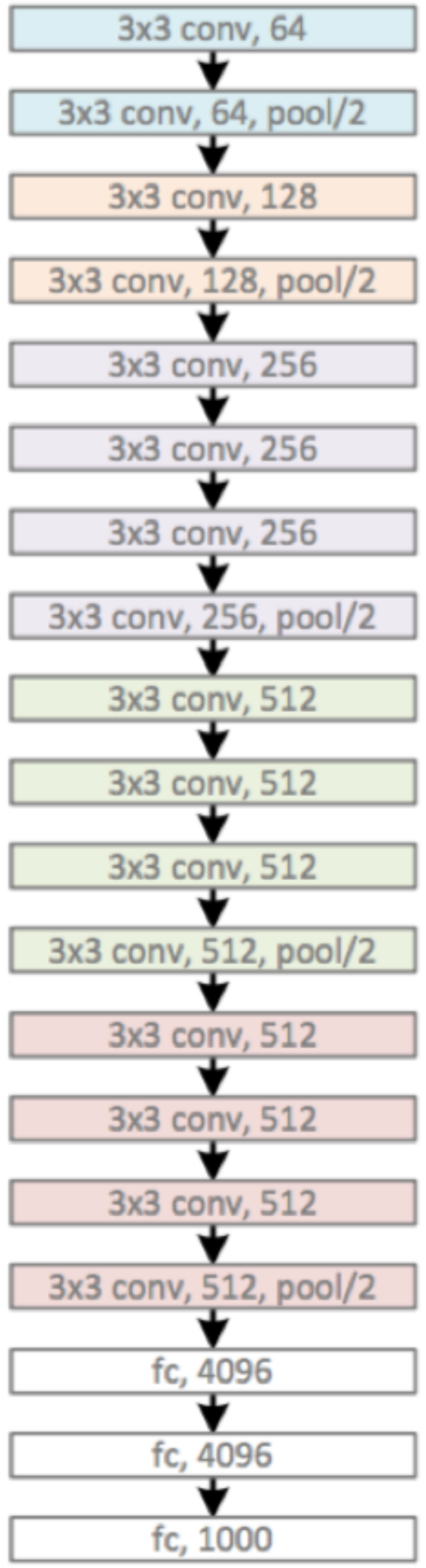
Error: 7.3%

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

Source: Isola, Torralba, Freeman

VGG-Net [Simonyan & Zisserman, 2015]

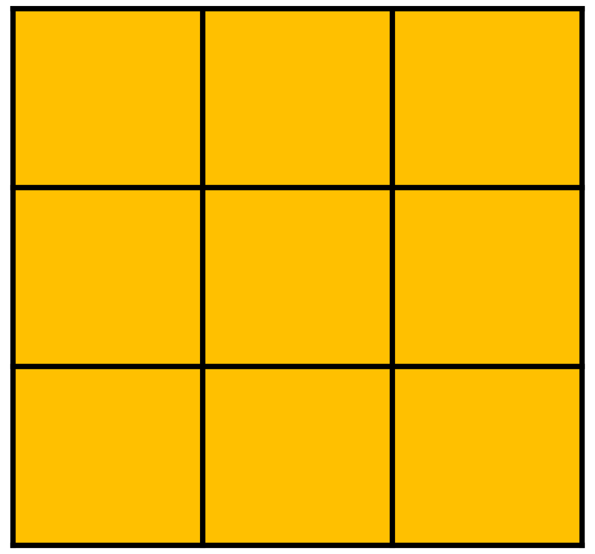
2014: VGG
16 conv. layers



Error: 7.3%

Main developments

- Small convolutional kernels: only 3x3

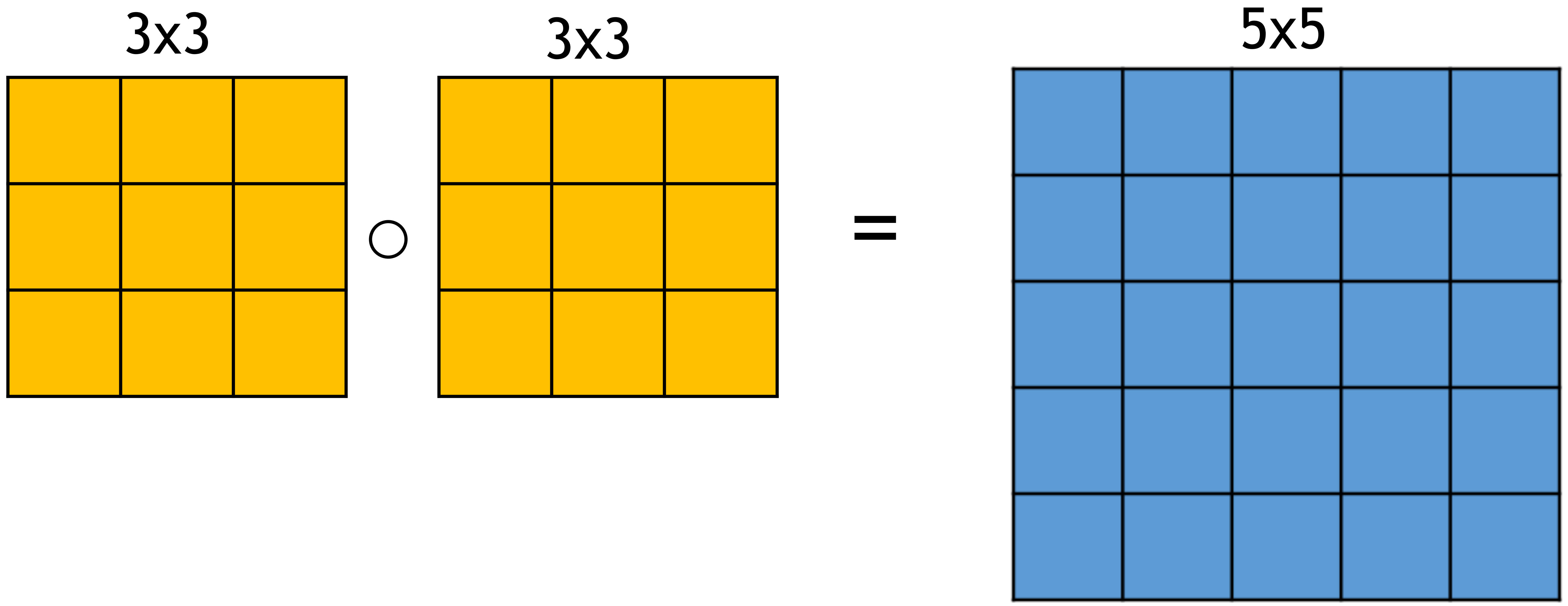


- Increased depth (5 -> 16/19 layers)

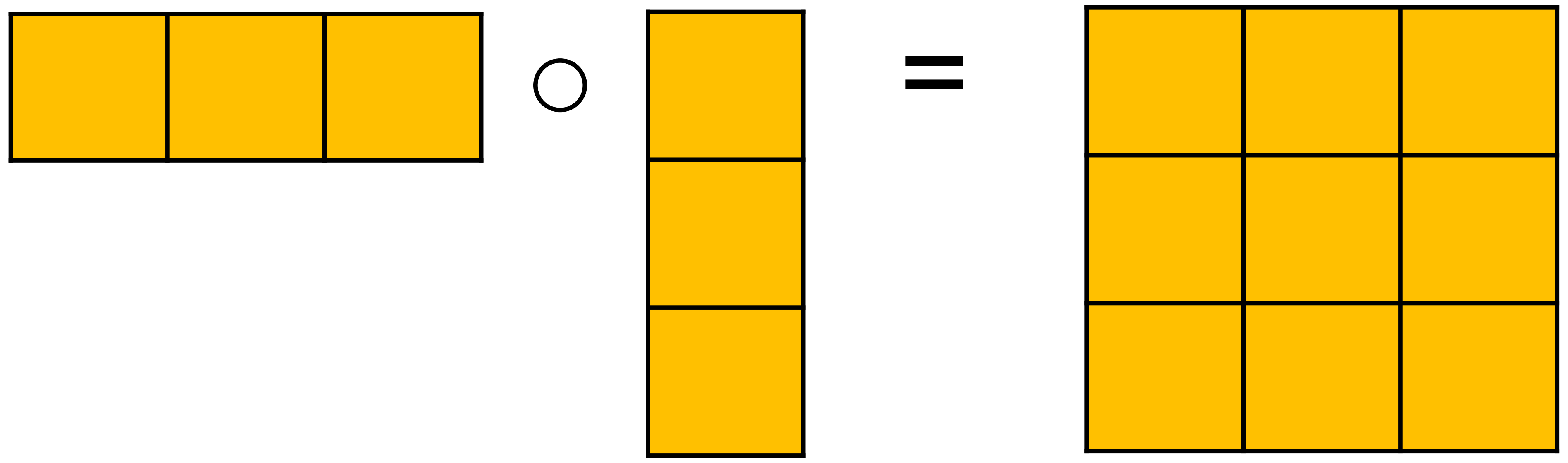
Source: Isola, Torralba, Freeman

Other tricks for designing convolutional nets

Chaining convolutions



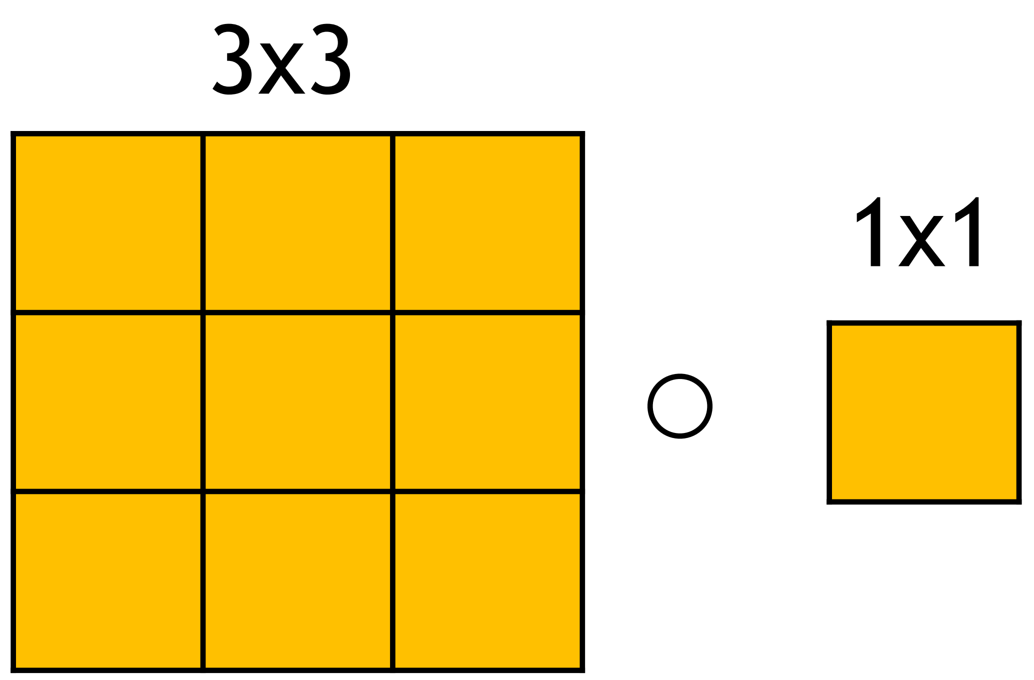
Nonlinearity in between.
25 coefficients, but only
18 degrees of freedom



9 coefficients, but only
6 degrees of freedom.
Less common.

Source: Isola, Torralba, Freeman

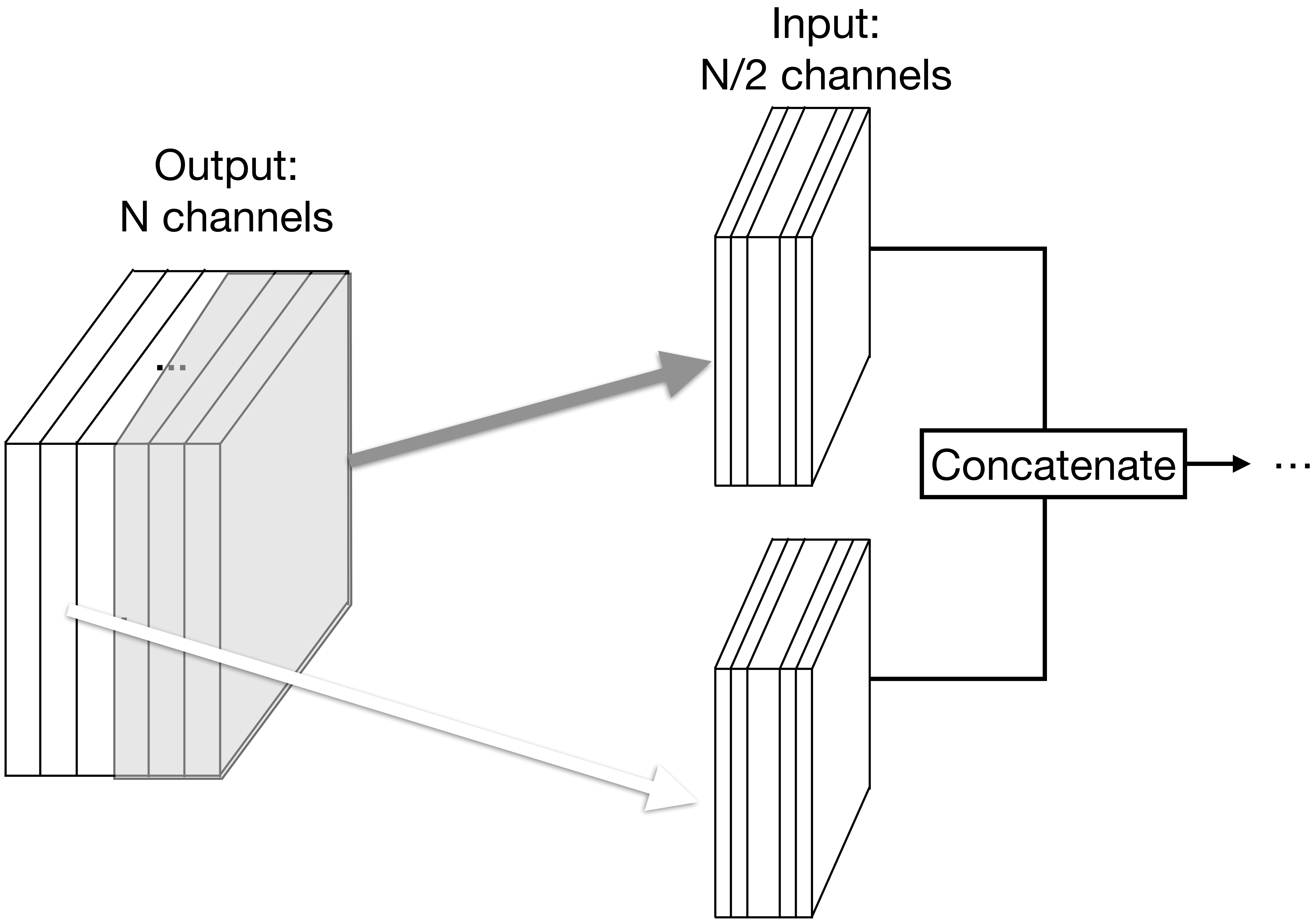
1x1 convolutions



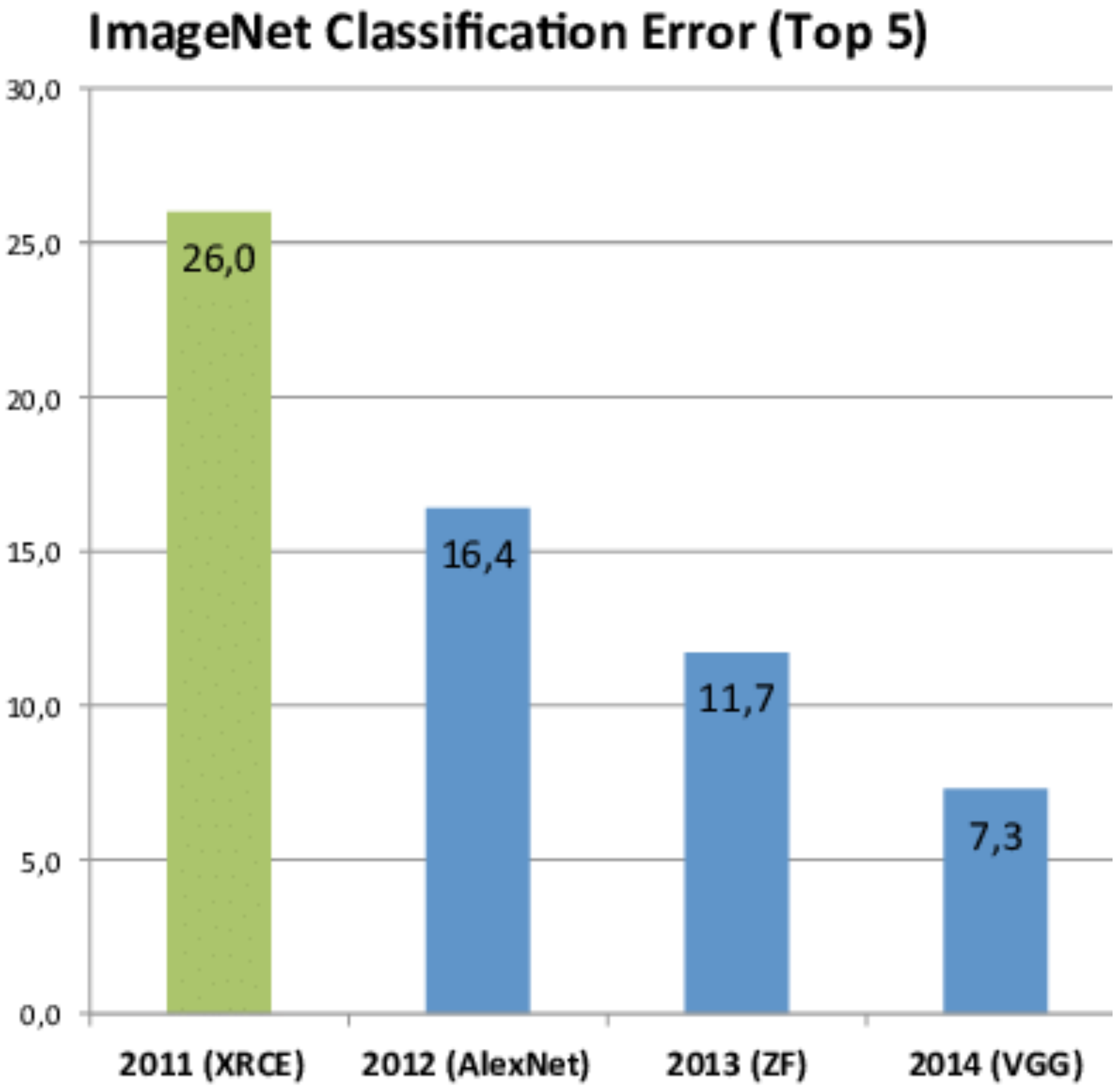
Why do this?

(nonlinearity in between)

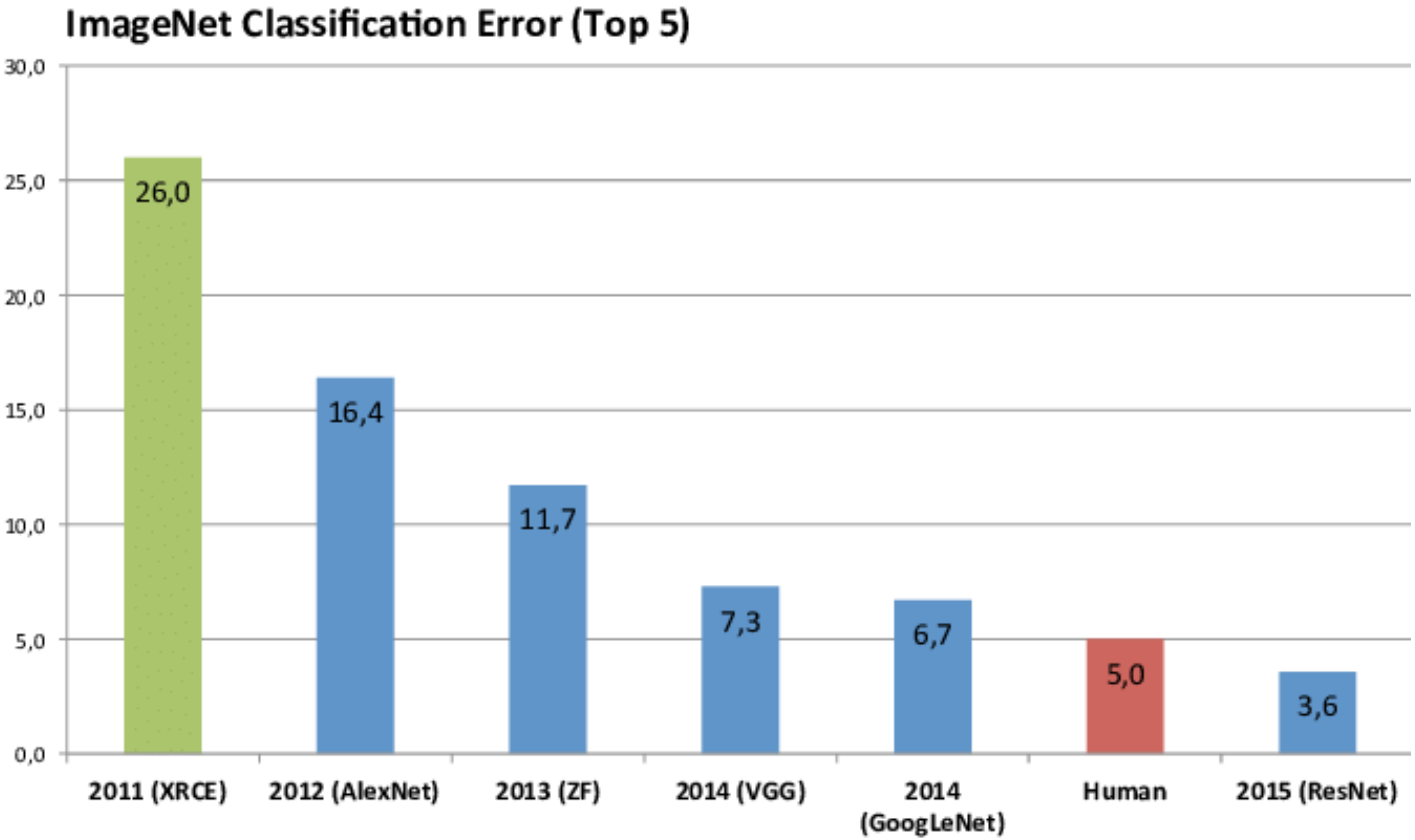
Grouped Convolutions



Split channels into N groups, and process separately with N convolution layers.



Source: Isola, Torralba, Freeman



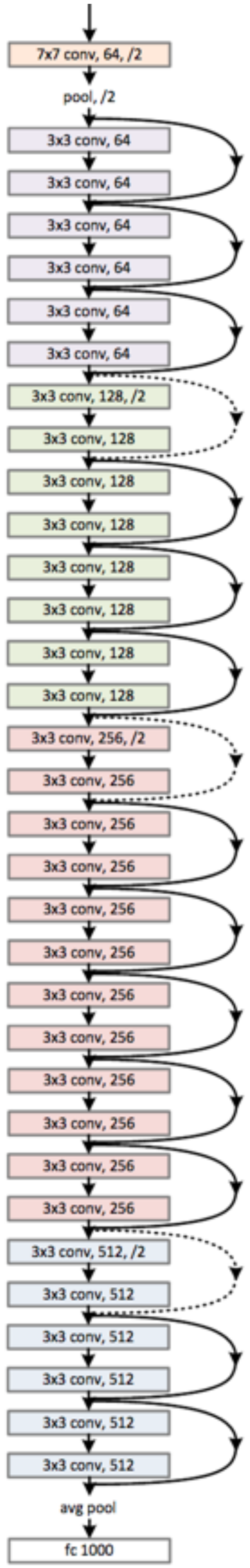
2016: ResNet
>100 conv. layers

Error: 3.6%

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

Source: Isola, Torralba, Freeman

2016: ResNet
>100 conv. layers

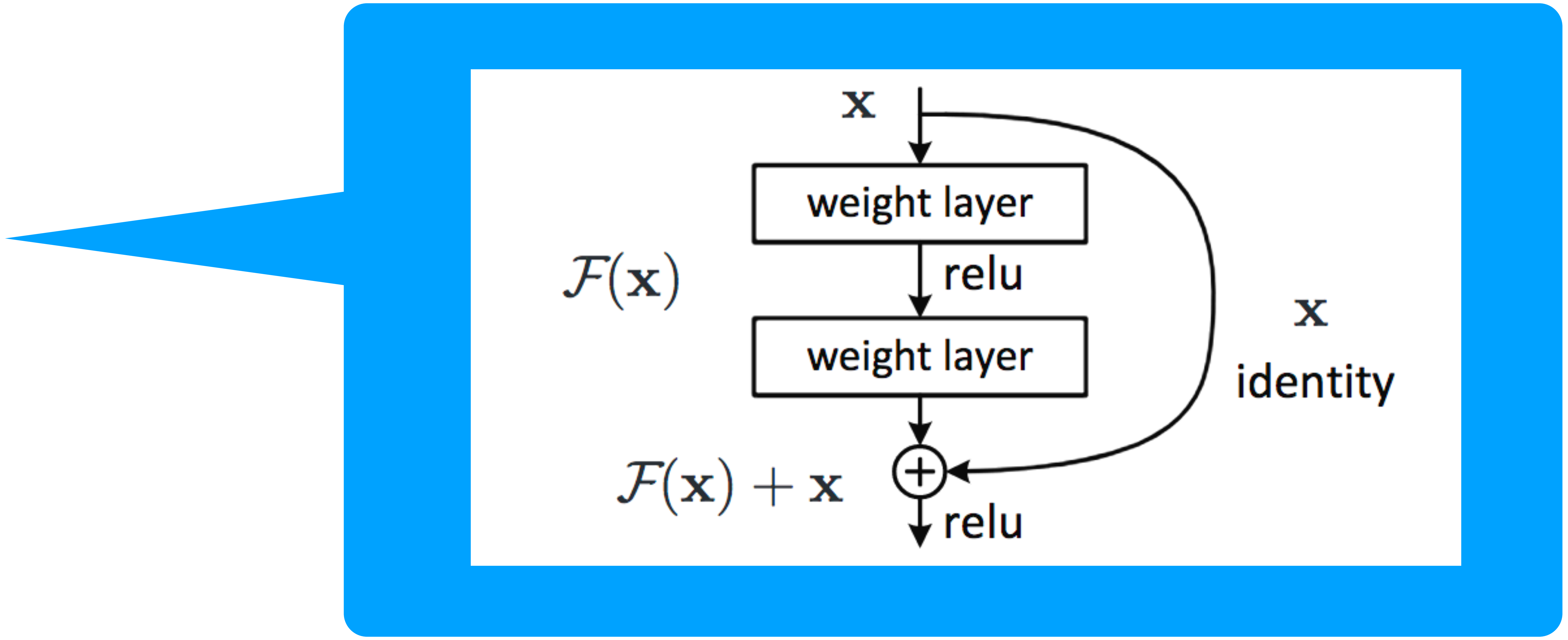


Error: 3.6%

ResNet [He et al, 2016]

Main developments

- Increased depth possible through residual blocks



Source: Isola, Torralba, Freeman

Residual Blocks

Problem: Hard to train very deep nets (50+ layers). This is an optimization issue, not overfitting: shallow models often get higher *training* accuracy than deep ones!

Idea: Make it easy to represent for the network to implement the identity.

Normal convolution + relu:

$$x_{i+1} = \mathrm{relu}(x_i \circ f)$$

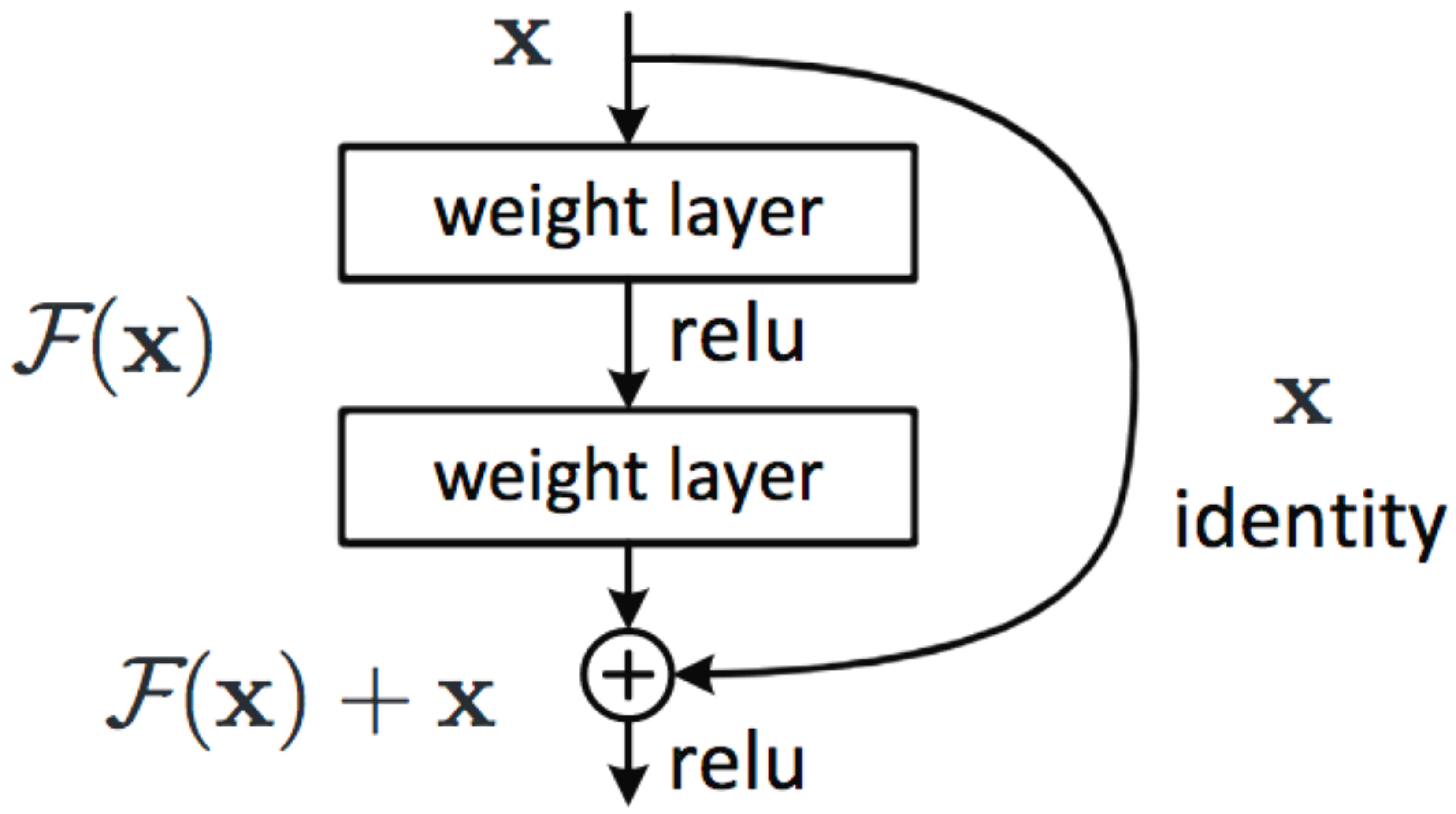
Residual connection

$$x_{i+1} = \mathrm{relu}((x_i \circ f) + x_i)$$

In general, do multiple convolutions (with nonlinearities) before summing:

$$x_{i+1} = \mathrm{relu}(\mathcal{F}(x_i) + x_i)$$

Residual Blocks



Why do they work?

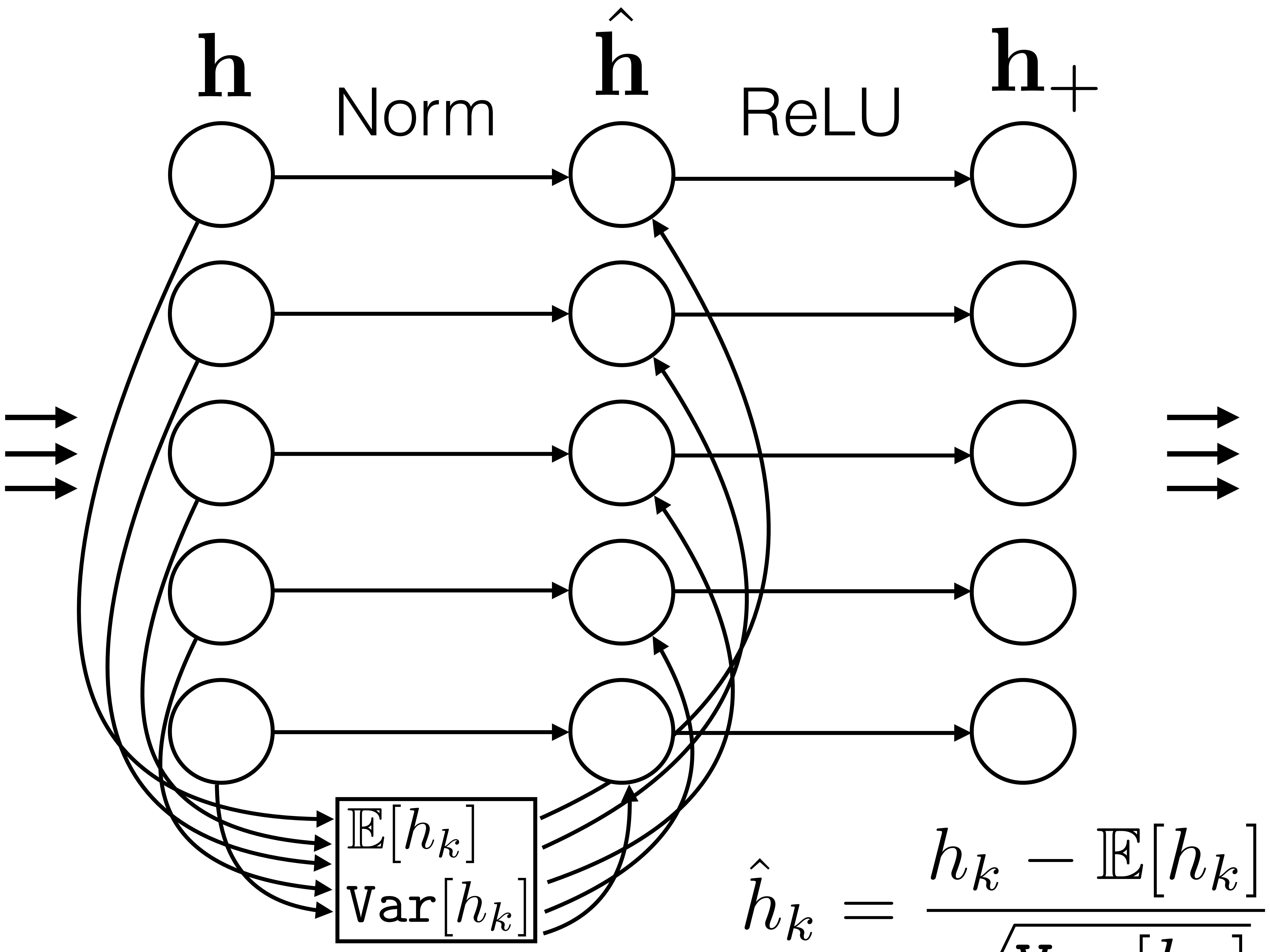
- Gradients can propagate faster (via the identity mapping)
- Within each block, only small residuals have to be learned

Source: Isola, Torralba, Freeman

Normalization layers

- Standardize activations by subtracting mean and dividing by standard deviation (averaged over all spatial locations).
- This provides a constant “interface” for later layers of the networks. Ensures that the previous layer will have unit variance and zero mean.
- Obtains invariance to mean and variance.
- Can allow you to train with larger learning rate and significantly speed up training!

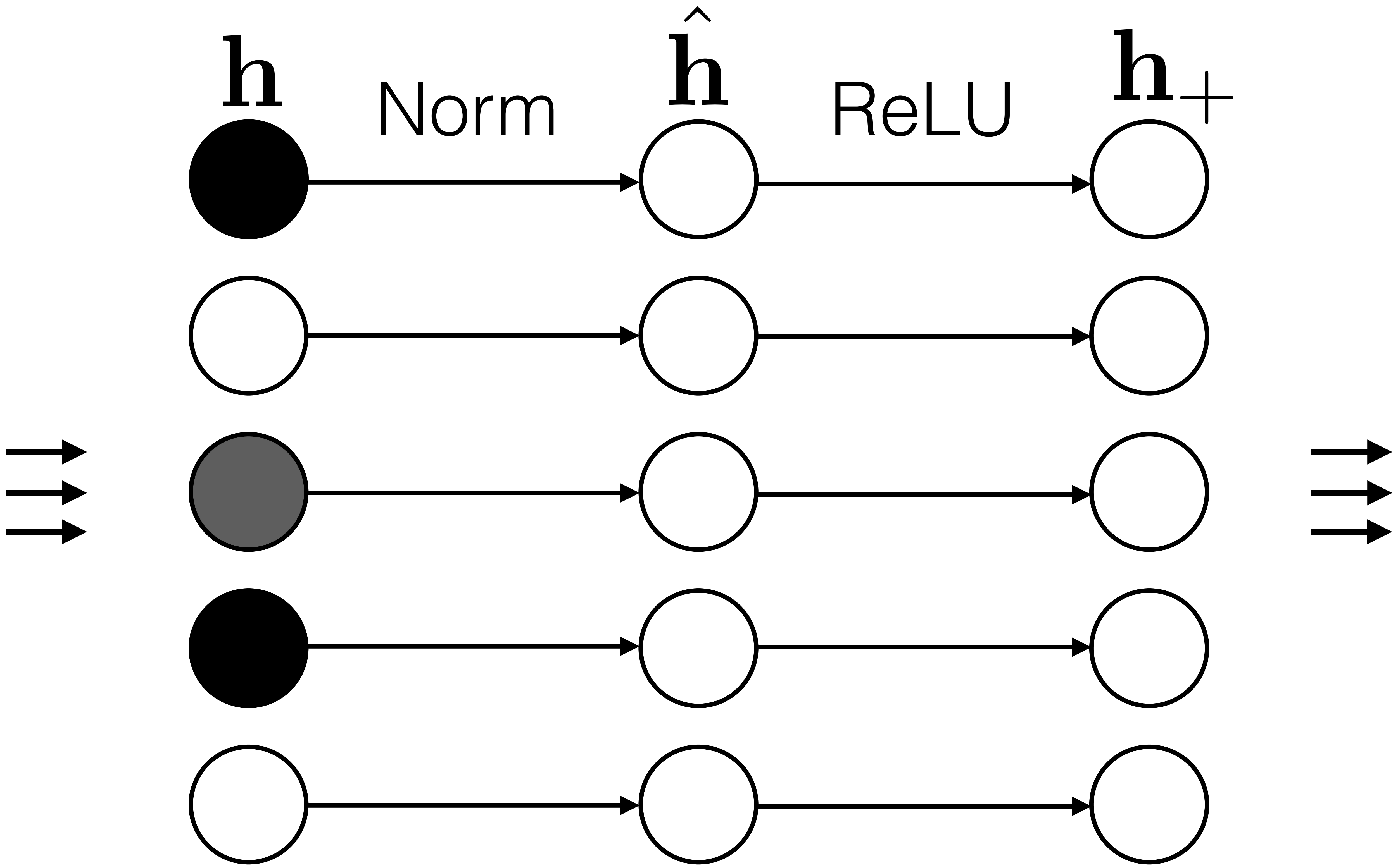
Normalization layers



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

Source: Isola, Torralba, Freeman

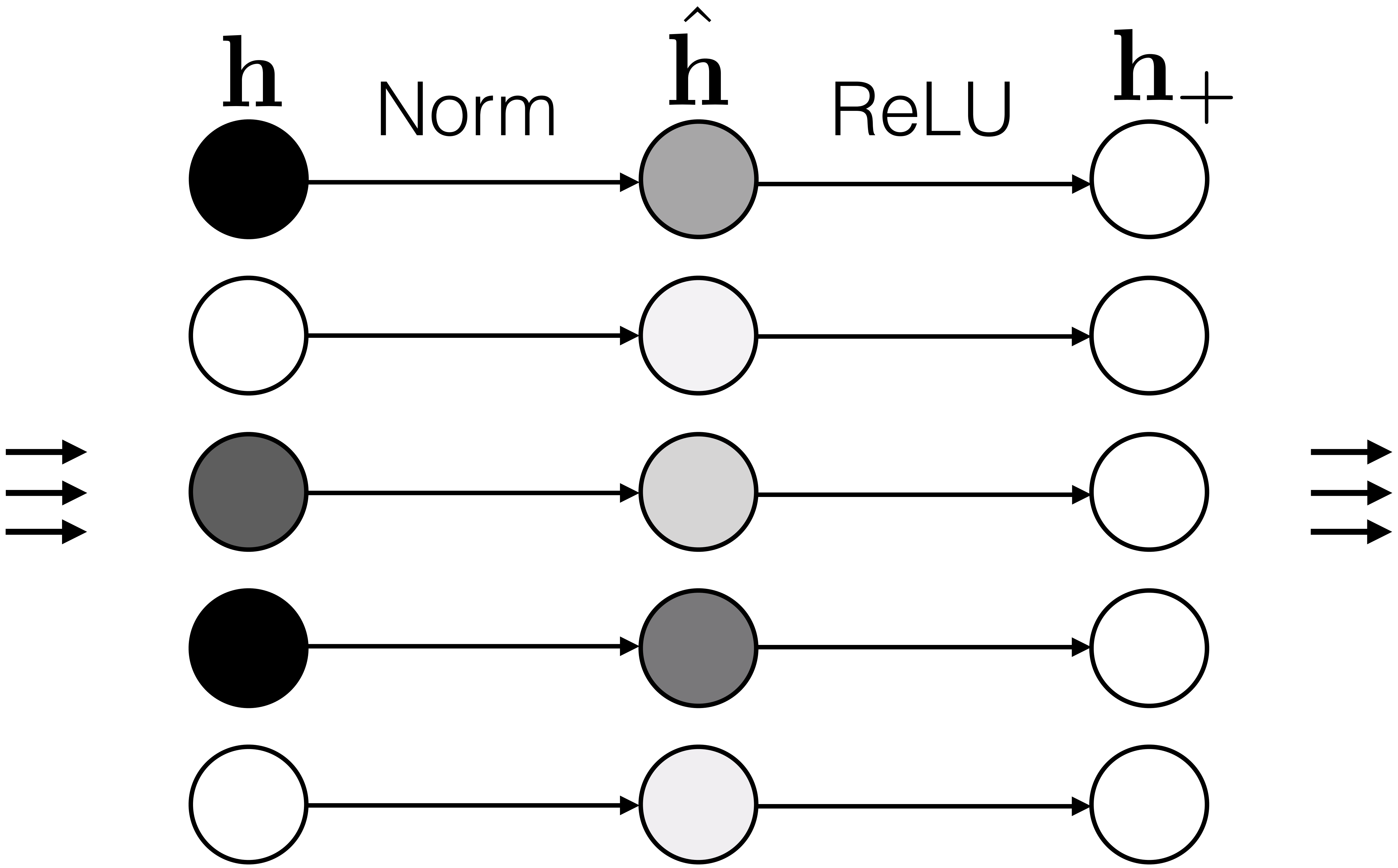
Normalization layers



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

Source: Isola, Torralba, Freeman

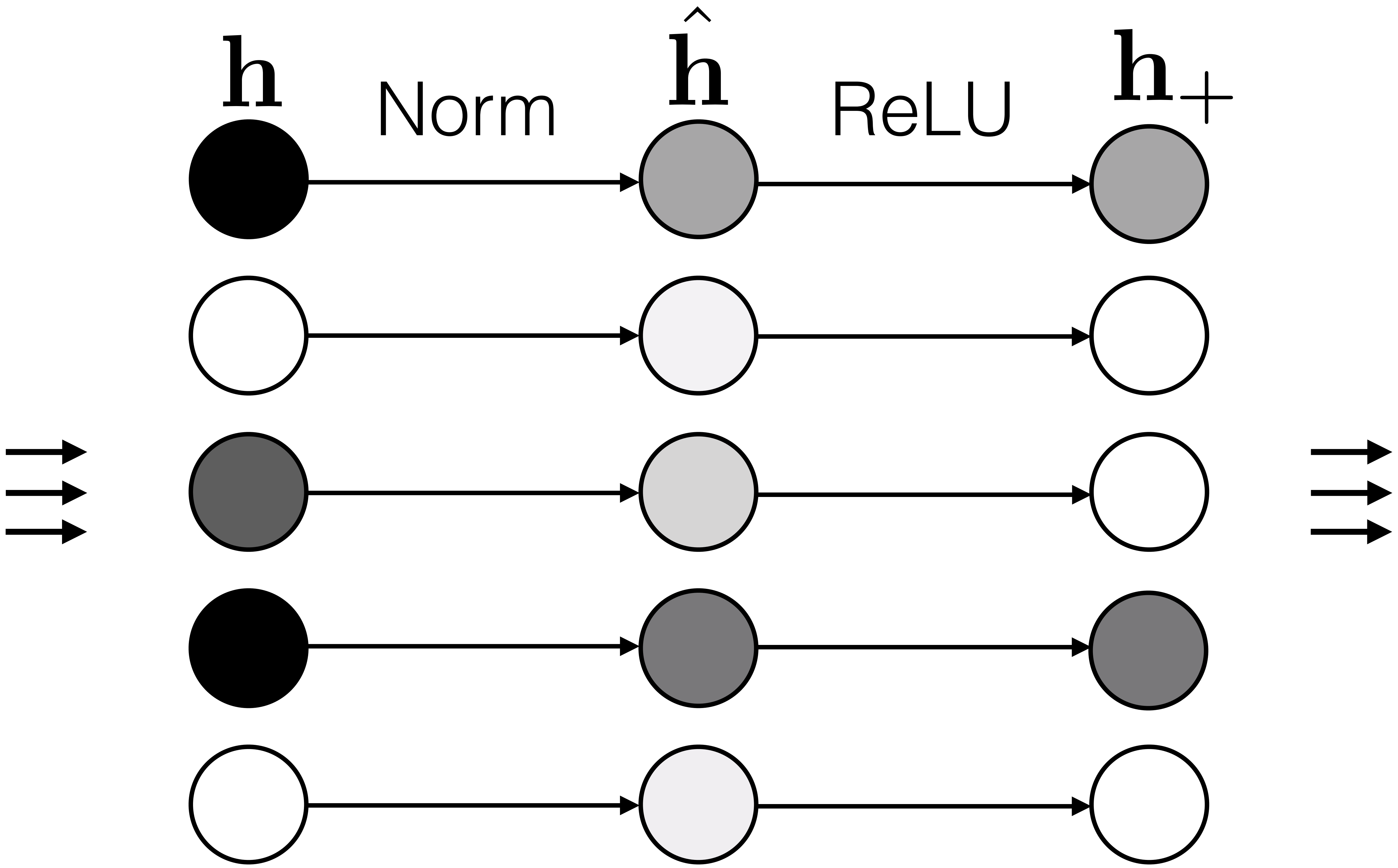
Normalization layers



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

Source: Isola, Torralba, Freeman

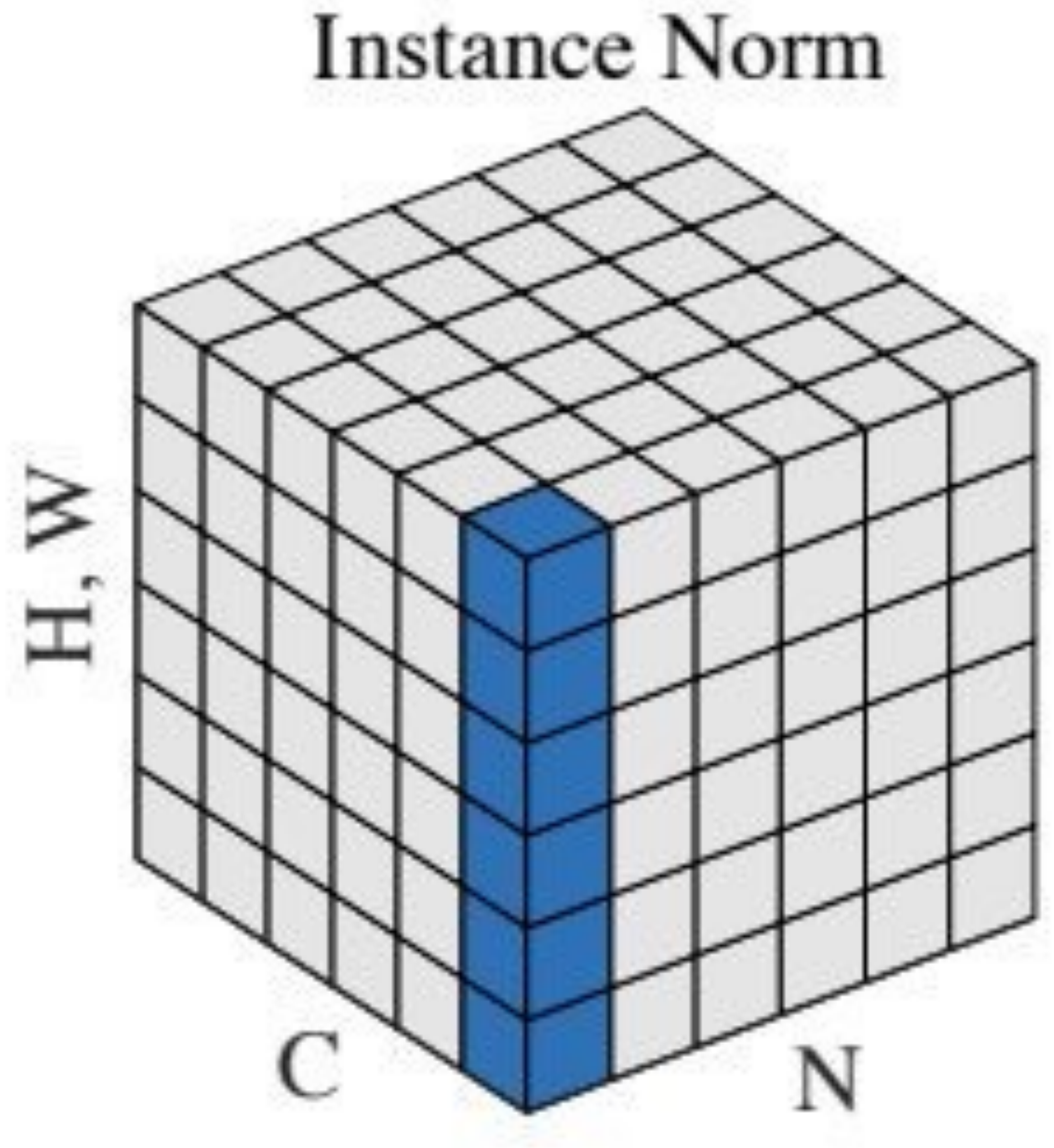
Normalization layers



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

Source: Isola, Torralba, Freeman

Instance normalization



Filter value in a CNN layer

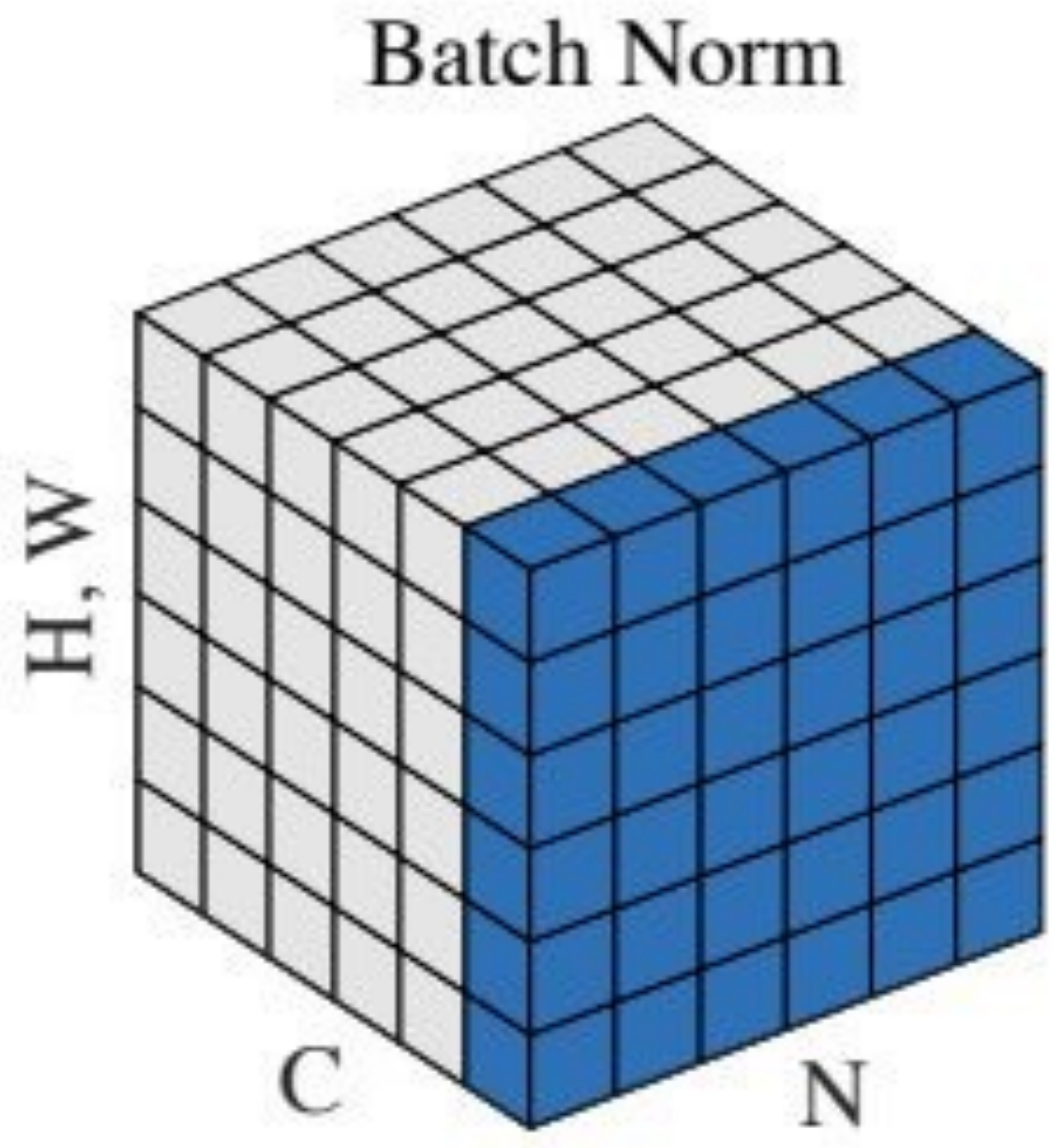
Average and standard deviation each filter response, estimated over **instance**.

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

Normalize a single hidden unit's activations to be mean 0, standard deviation 1.

[Figure from Wu & He, 2018] [Ulyanov et al., 2015]

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 a single hidden unit's activations to be mean 0, standard deviation 1.

At test time, remember the mean and standard deviation seen during training.

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

A few practical issues

Dealing with rectangular images

Rectangular images

Resize, then take square crop from center

Training with data augmentation

Original image

Flipping

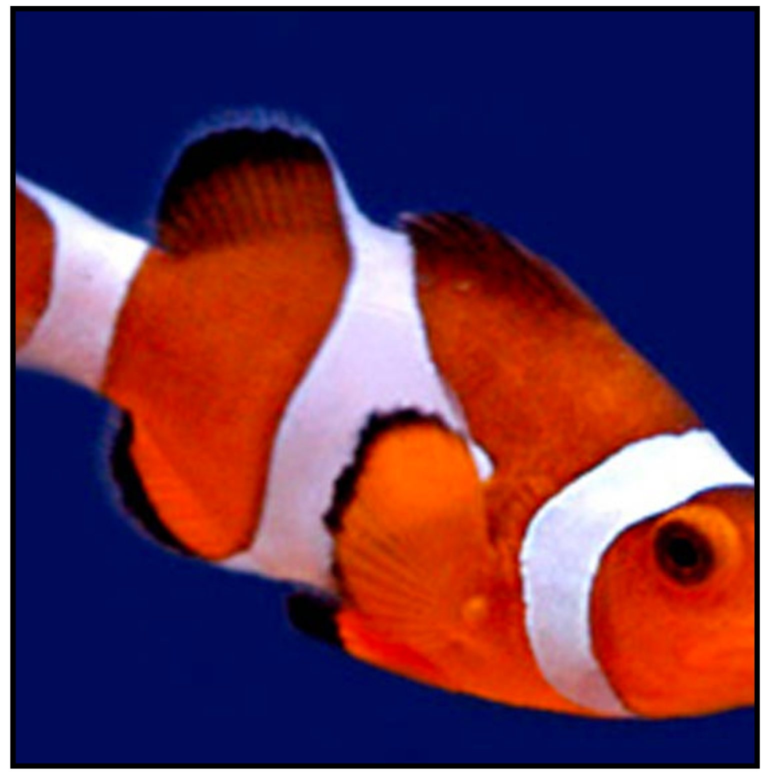
- Less susceptible to overfitting.
- Improves performance by simulating examples.

Training with data augmentation

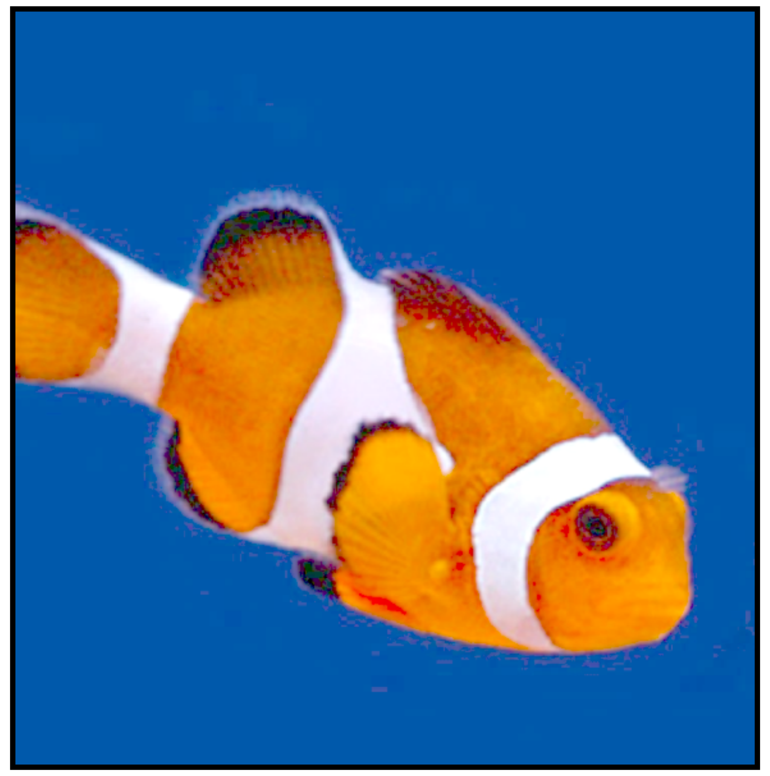
Original image

Flipping

Cropping



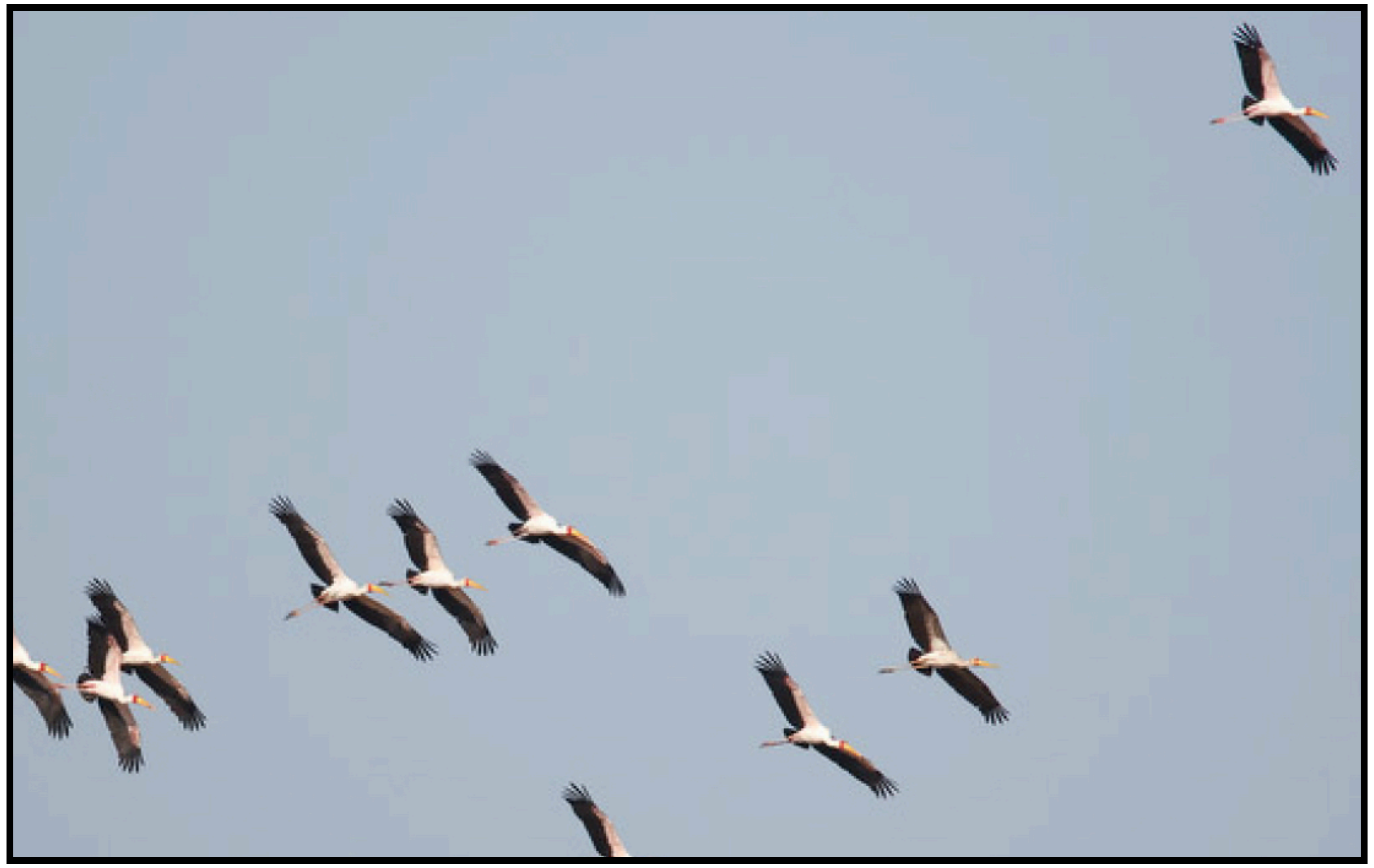
Scaling



Color jittering

Beyond image labeling

Object recognition: what objects are in the image?

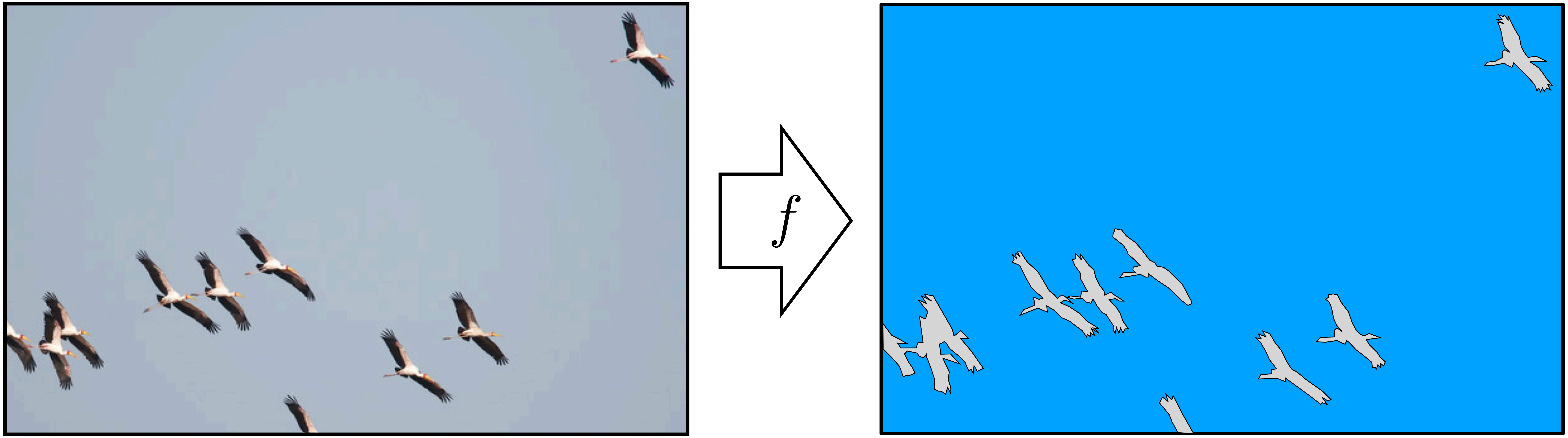


$$\xrightarrow{f}$$

"Birds"

Source: Torralba, Freeman, Isola

Semantic segmentation

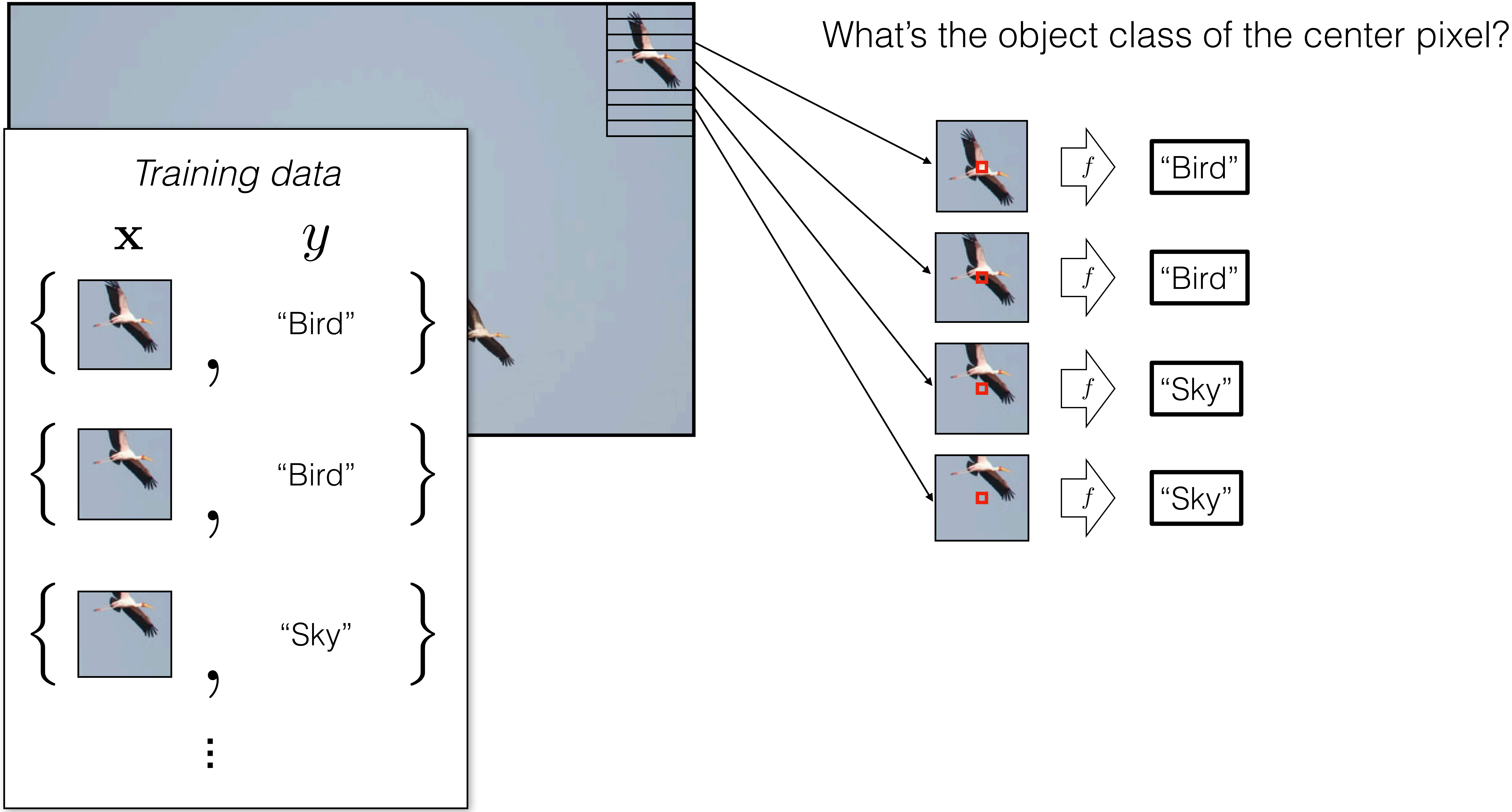


(Colors represent categories)

General technique: predict something at every pixel!

Source: Torralba, Freeman, Isola

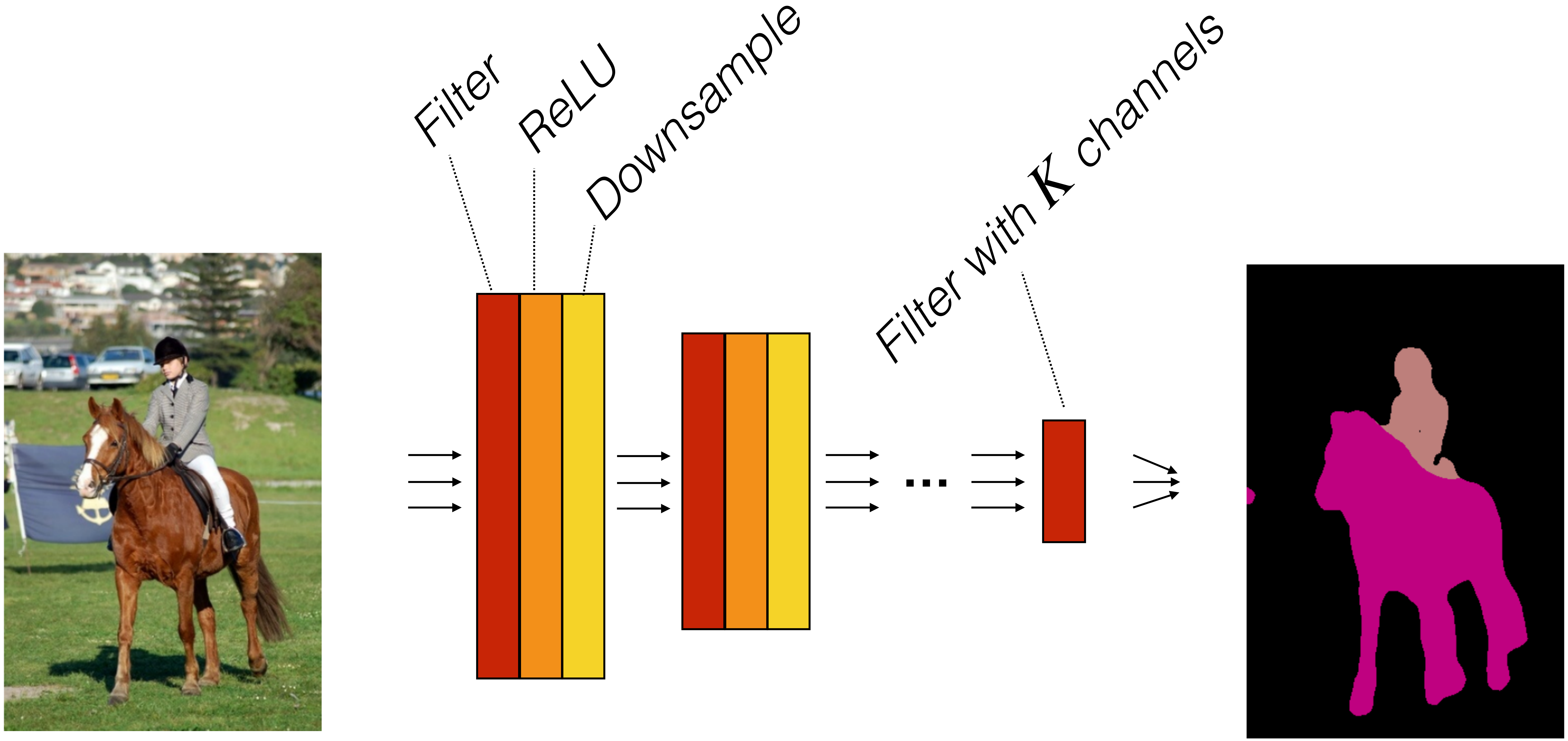
Idea #1: Independently classify windows



Source: Torralba, Freeman, Isola

Idea #2: Fully convolutional networks

Fully convolutional network



Classification problem with K classes

Idea #3: Dilated convolutions

Dilated convolutions

3x3

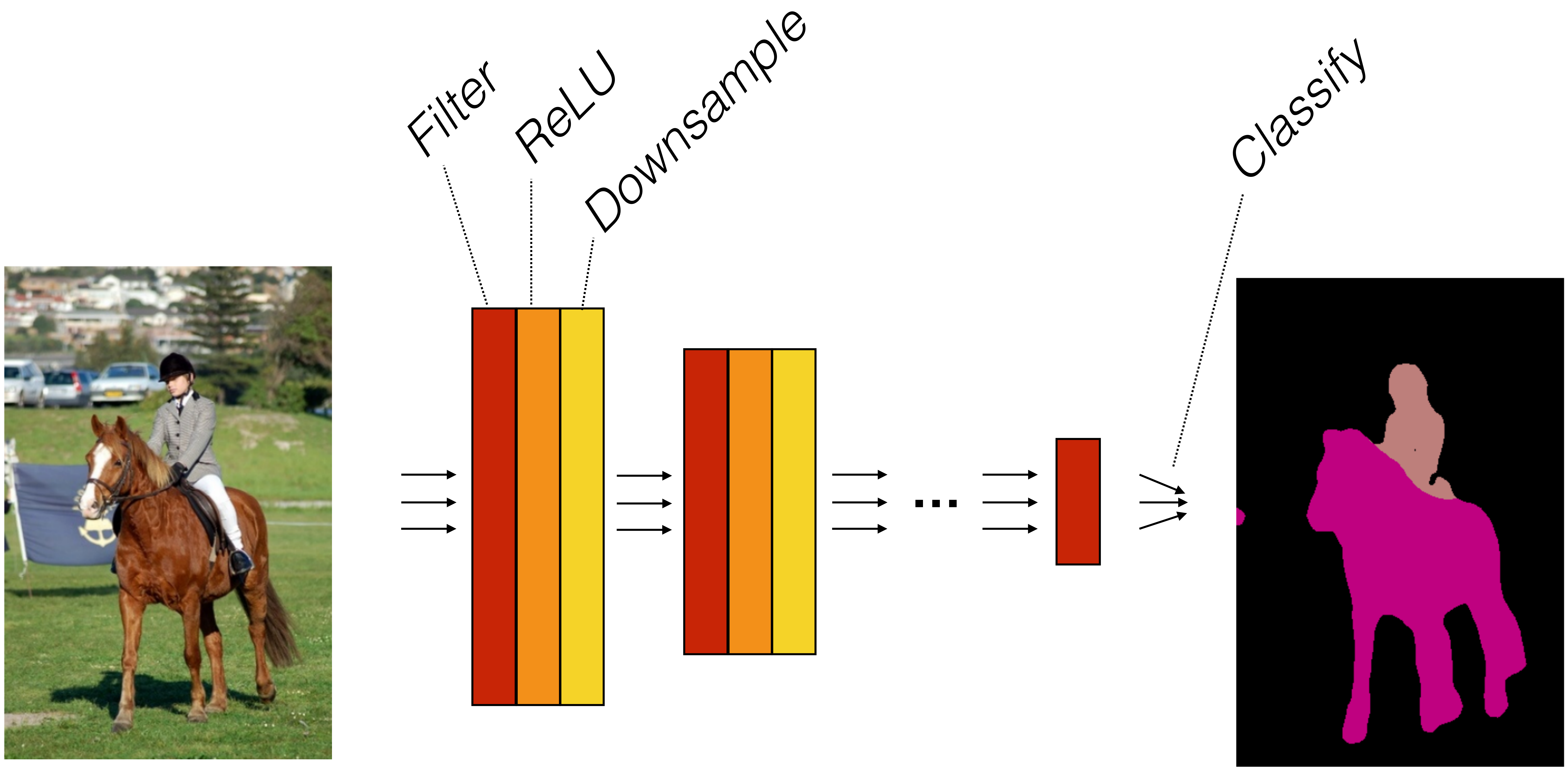
a	**b**	**c**
d	**e**	**f**
g	**h**	**i**

5x5

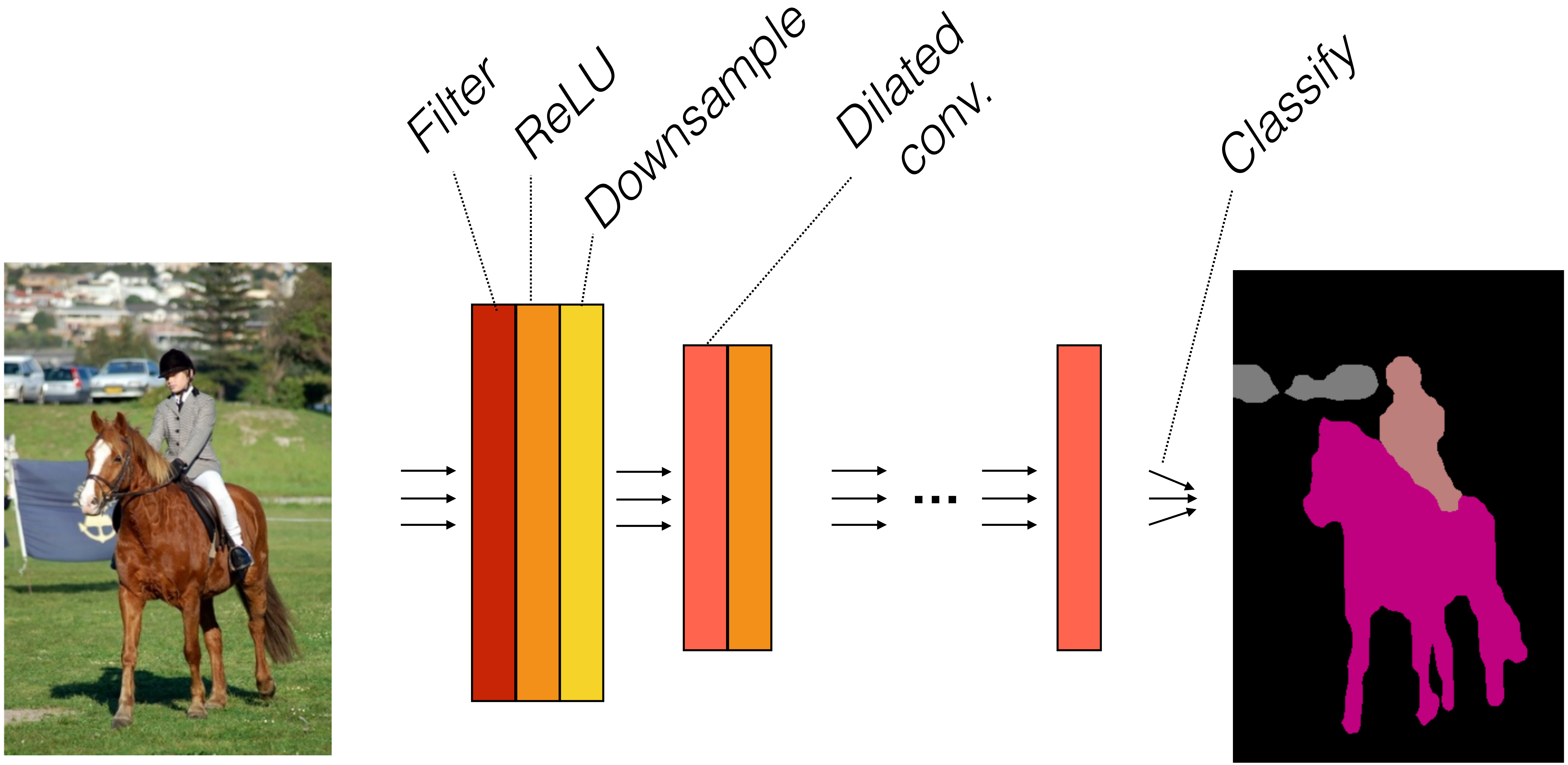
a	0	**b**	0	**c**
0	0	0	0	0
d	0	**e**	0	**f**
0	0	0	0	0
g	0	**h**	0	**i**

- Alternative to pooling that preserves input size
- 9 degrees of freedom
- 5x5 receptive field

CNN without dilated convolutions



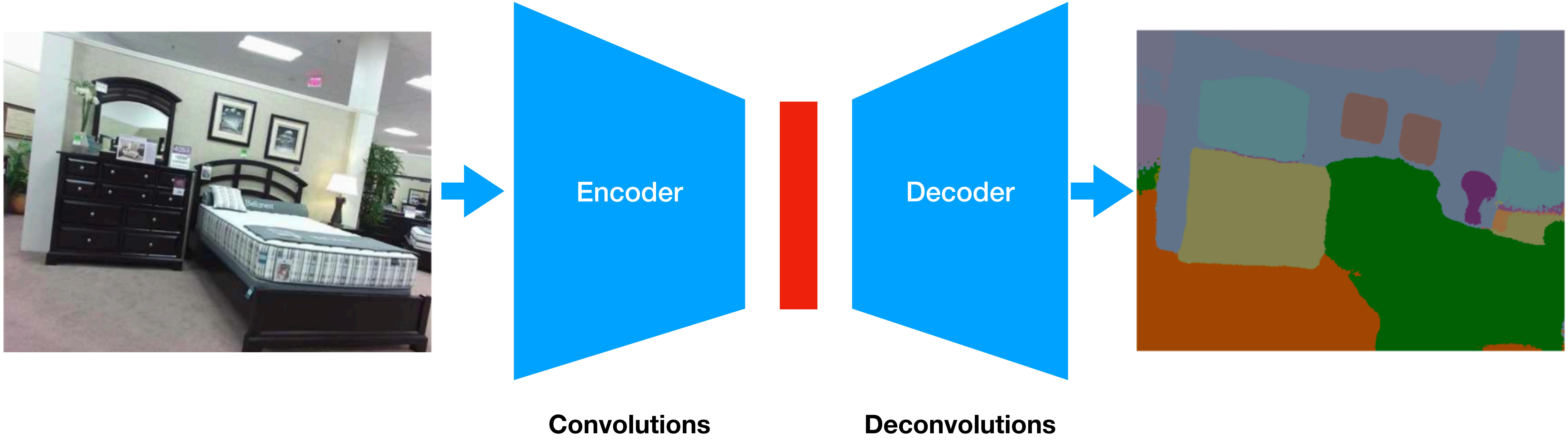
CNN with dilated convolutions



Even with dilated convolutions, still not full resolution

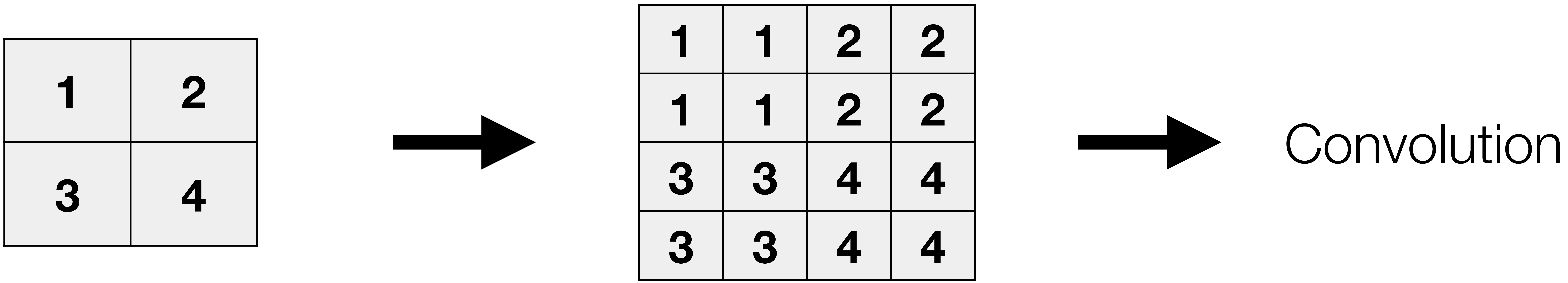
Idea #4: Encoder-decoder models

Encoder-decoder architectures



Source: Torralba, Freeman, Isola

Upsampling



- Often using nearest-neighbor upsampling
- Can also use interpolation.
- Produces fewer “checkerboard” artifacts

Transposed convolution

$$\underset{I}{\begin{bmatrix}1&1\\2&3\end{bmatrix}} \quad \underset{F}{\begin{bmatrix}1&2\\3&4\end{bmatrix}} \;\rightarrow\; + \begin{matrix}\begin{bmatrix}1&2&0\\3&4&0\\0&0&0\end{bmatrix} & \begin{bmatrix}0&1&2\\0&3&4\\0&0&0\end{bmatrix}\\[6pt] \begin{bmatrix}0&0&0\\2&4&0\\6&8&0\end{bmatrix} & \begin{bmatrix}0&0&0\\0&3&6\\0&9&12\end{bmatrix}\end{matrix} \;\rightarrow\; \begin{bmatrix}1&3&2\\5&14&10\\6&17&12\end{bmatrix}$$

- Weight the filter by the image coefficient and sum.
- Also sometimes called “upconvolution” or “deconvolution”.

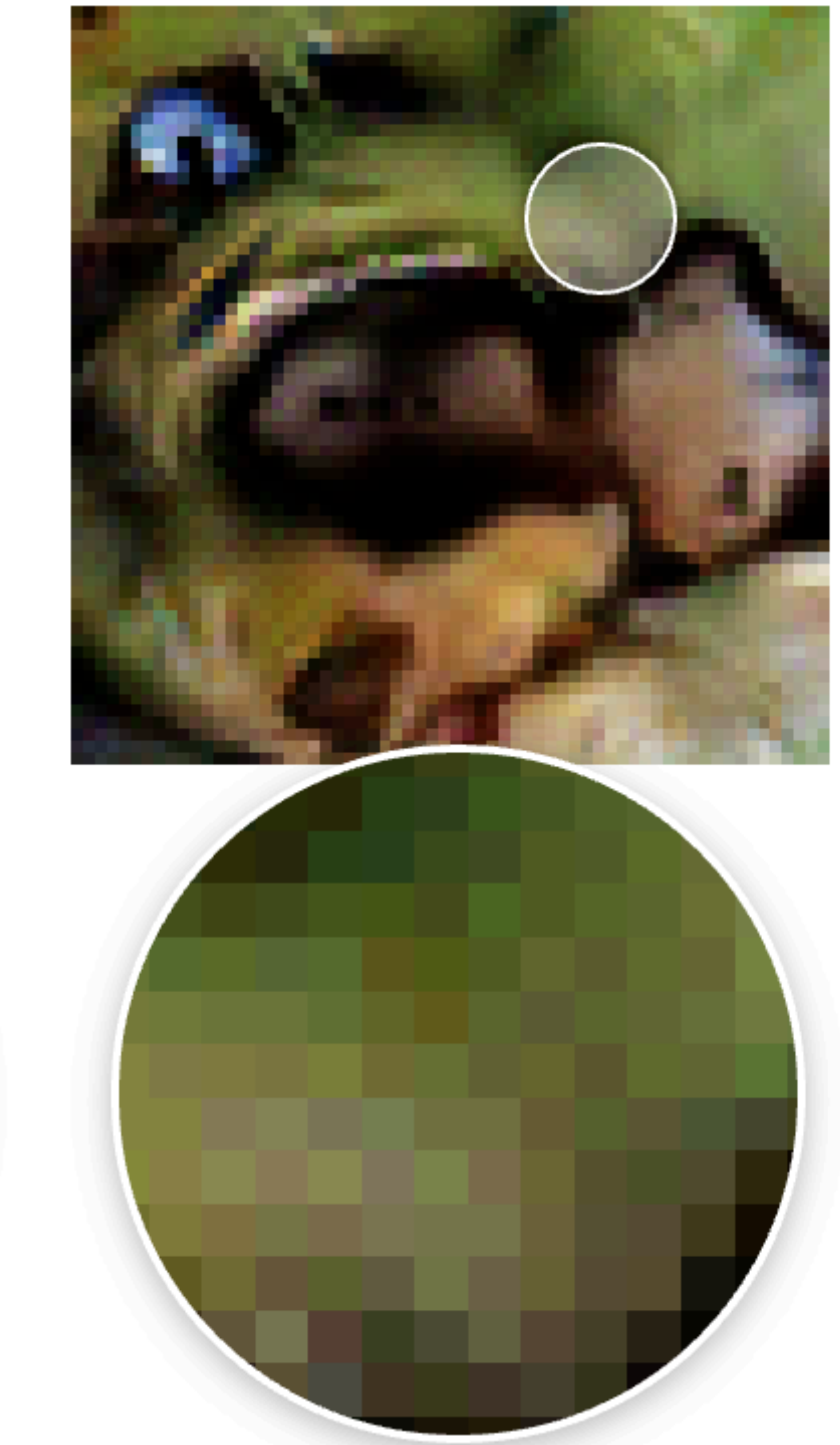
Transposed convolution

1	1
2	3

I

1	2
3	4

F

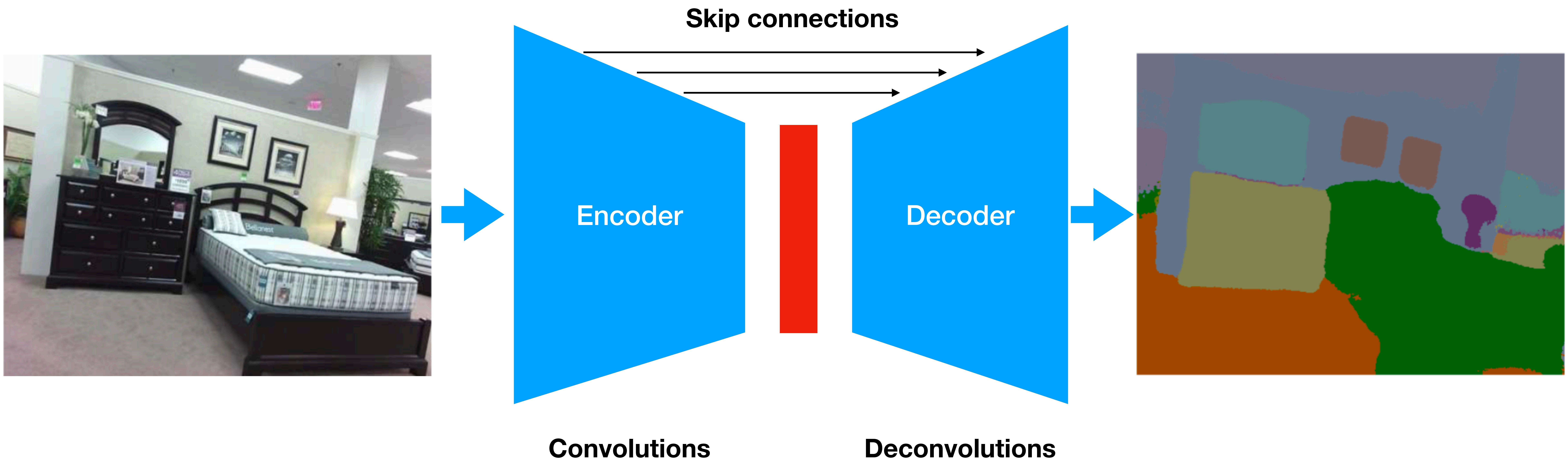


Donahue, et al., 2016 [3]

[Odena et al. Distill article]

Can lead to “checkerboard” artifacts.

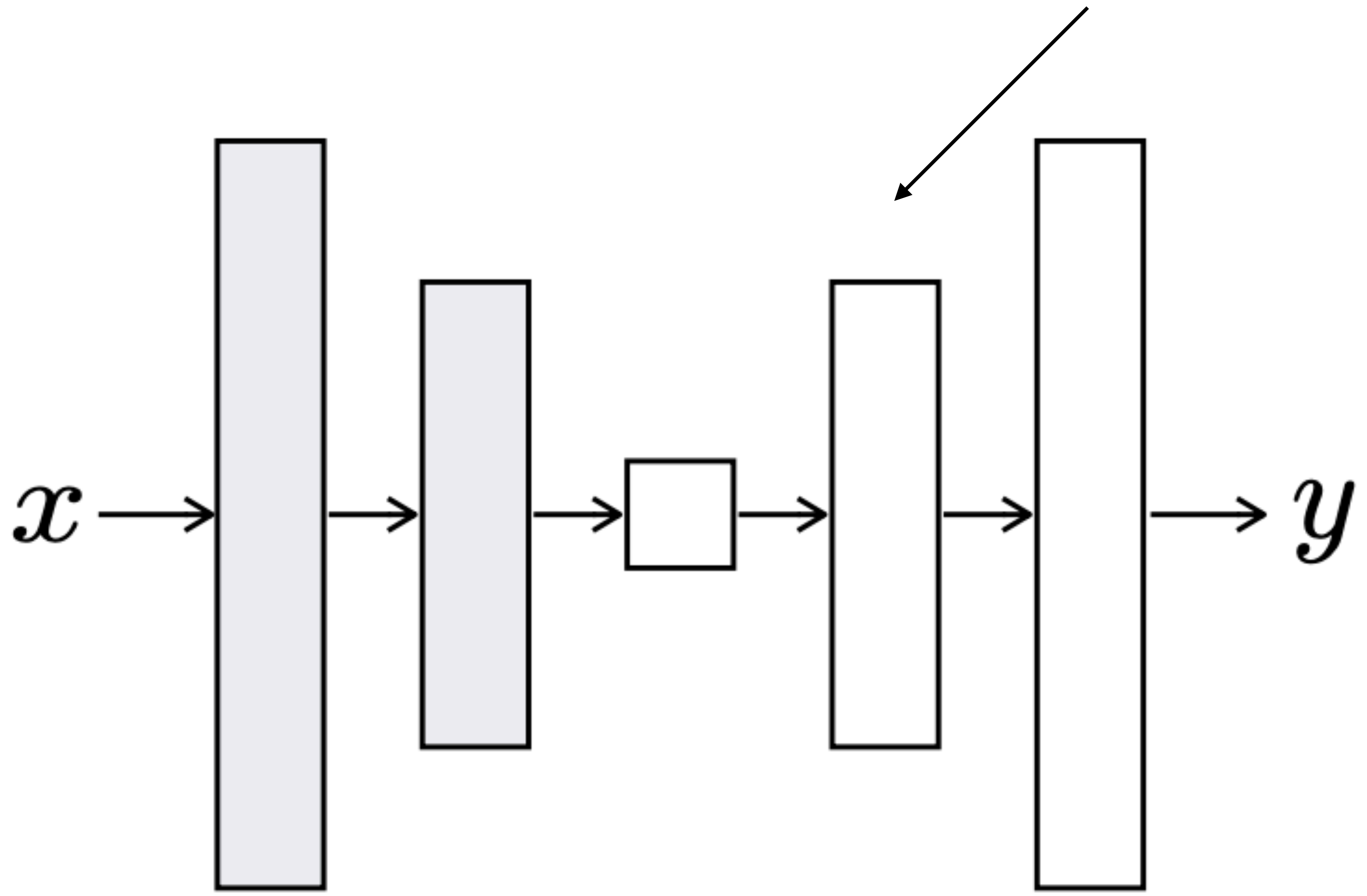
Encoder-decoder architectures



Source: Torralba, Freeman, Isola

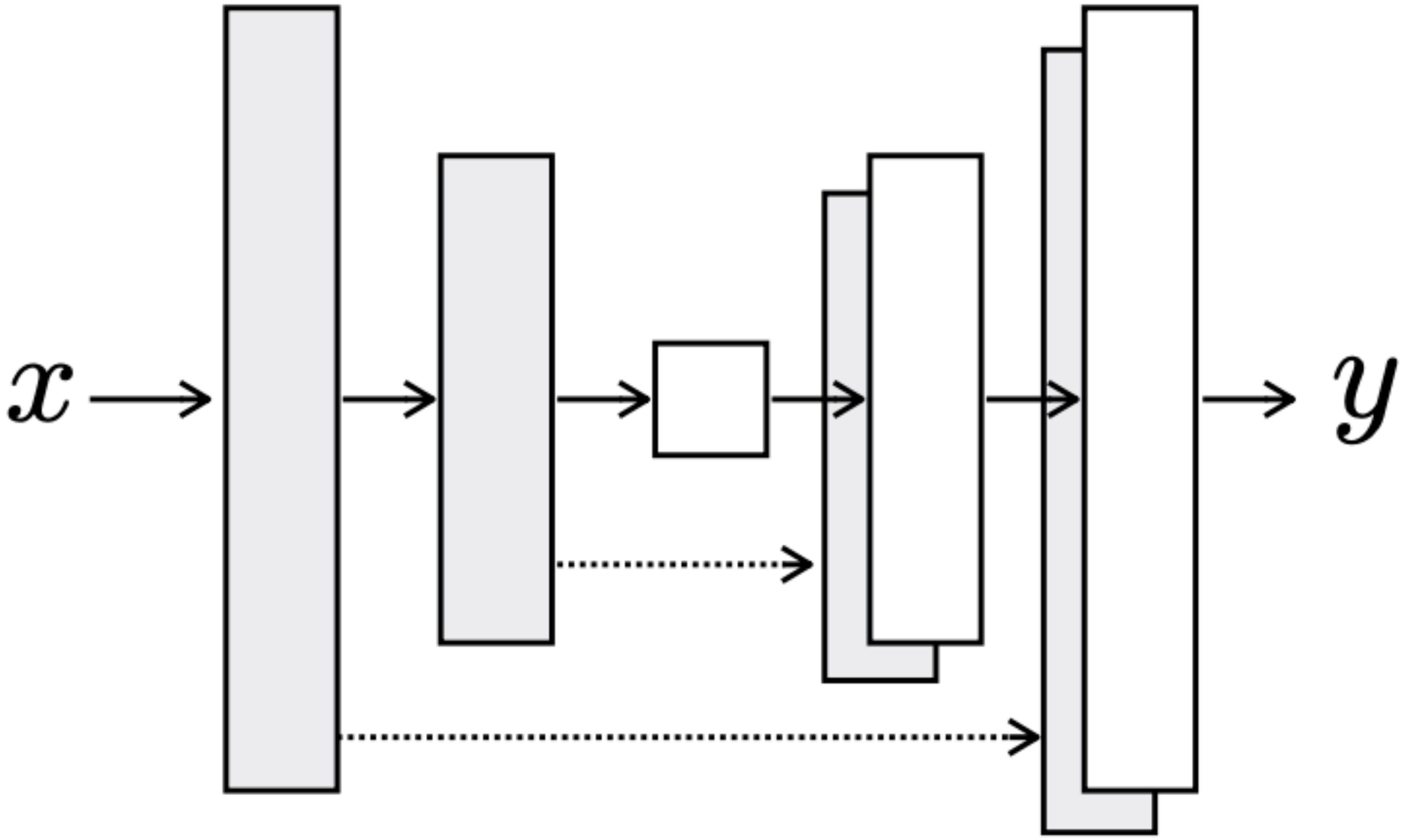
Encoder-decoder architectures

Transposed convolution



“Vanilla” encoder-decoder architecture

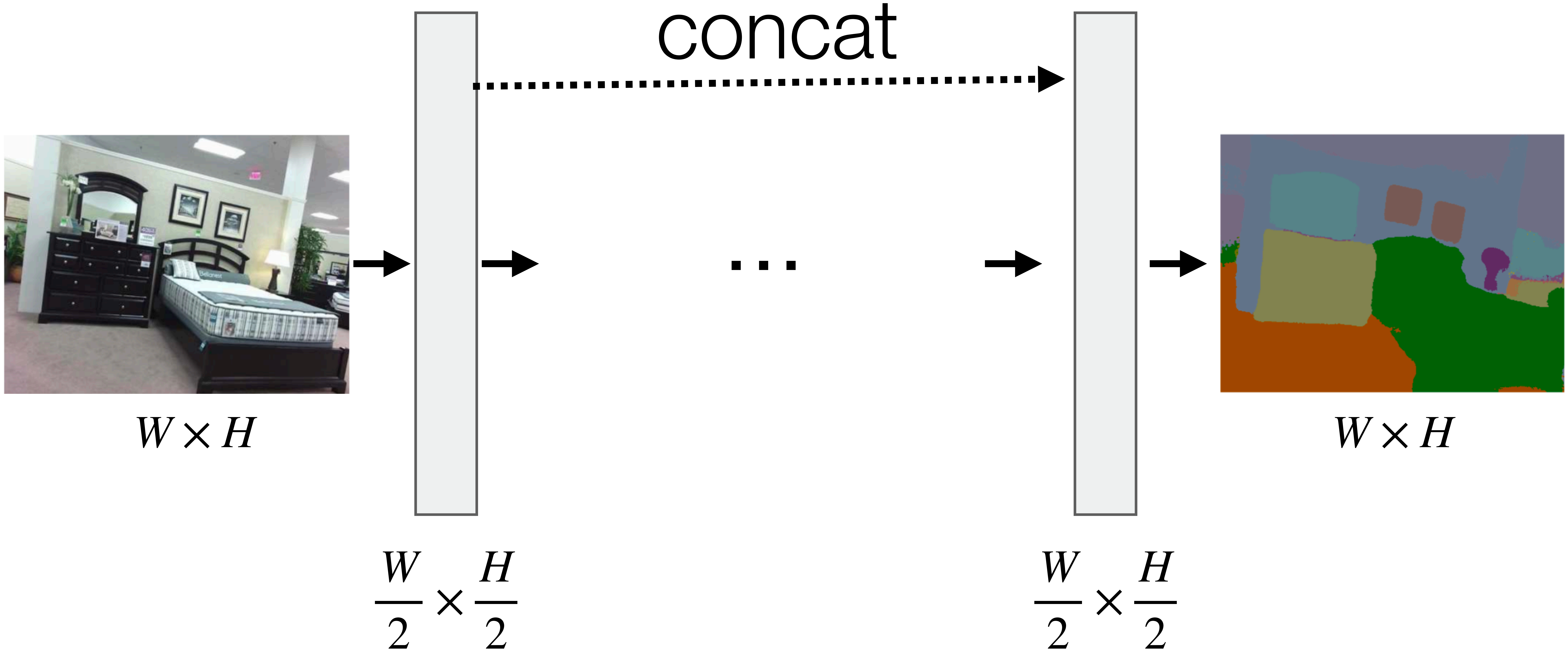
Early layers and late layers have same shape. Concatenate channel-wise!



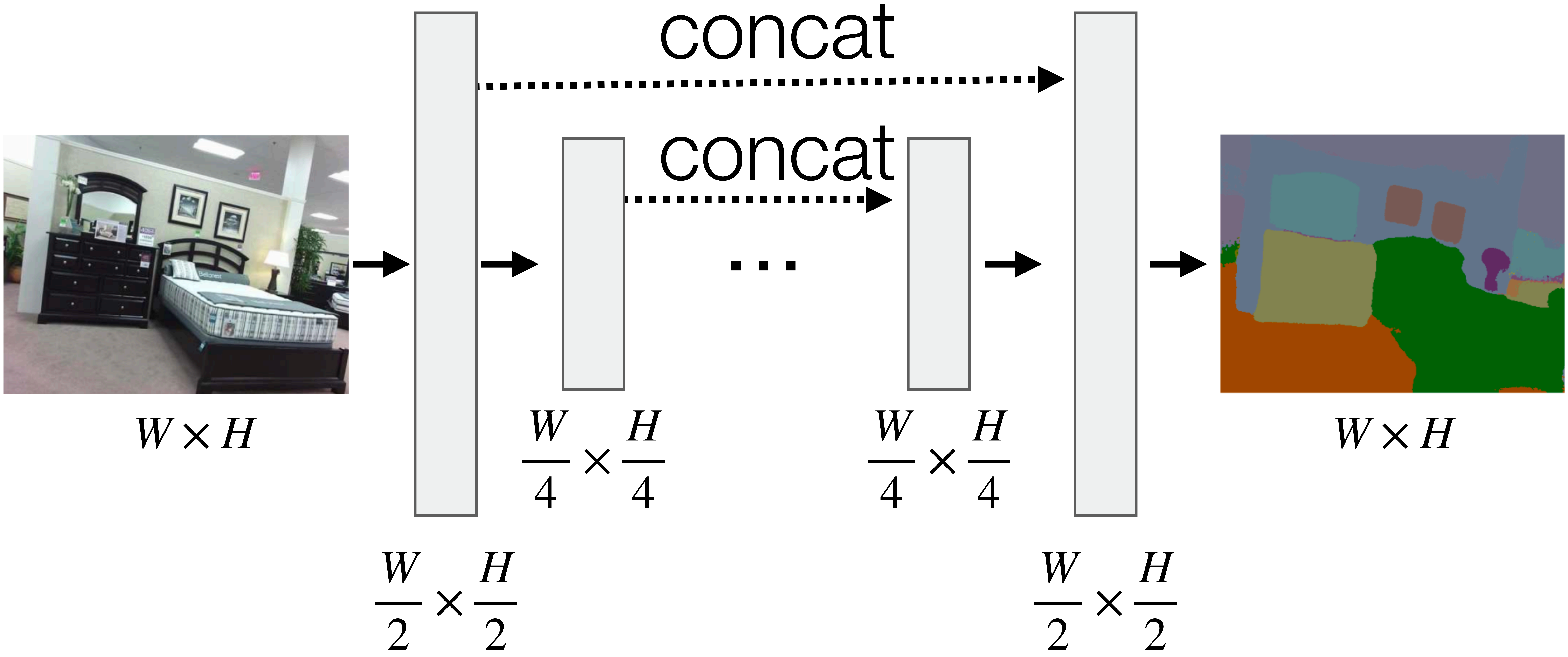
U-Net

Figures from [Isola et al., “Image-to-Image Translation with Conditional Adversarial Networks”, 2017]

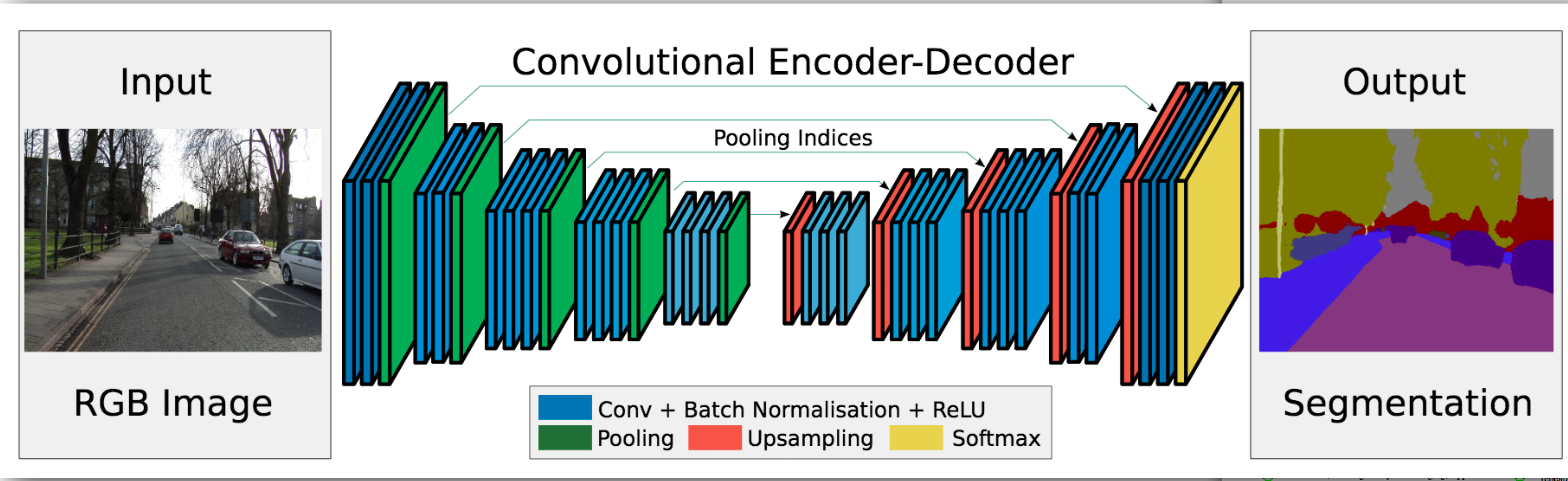
U-net

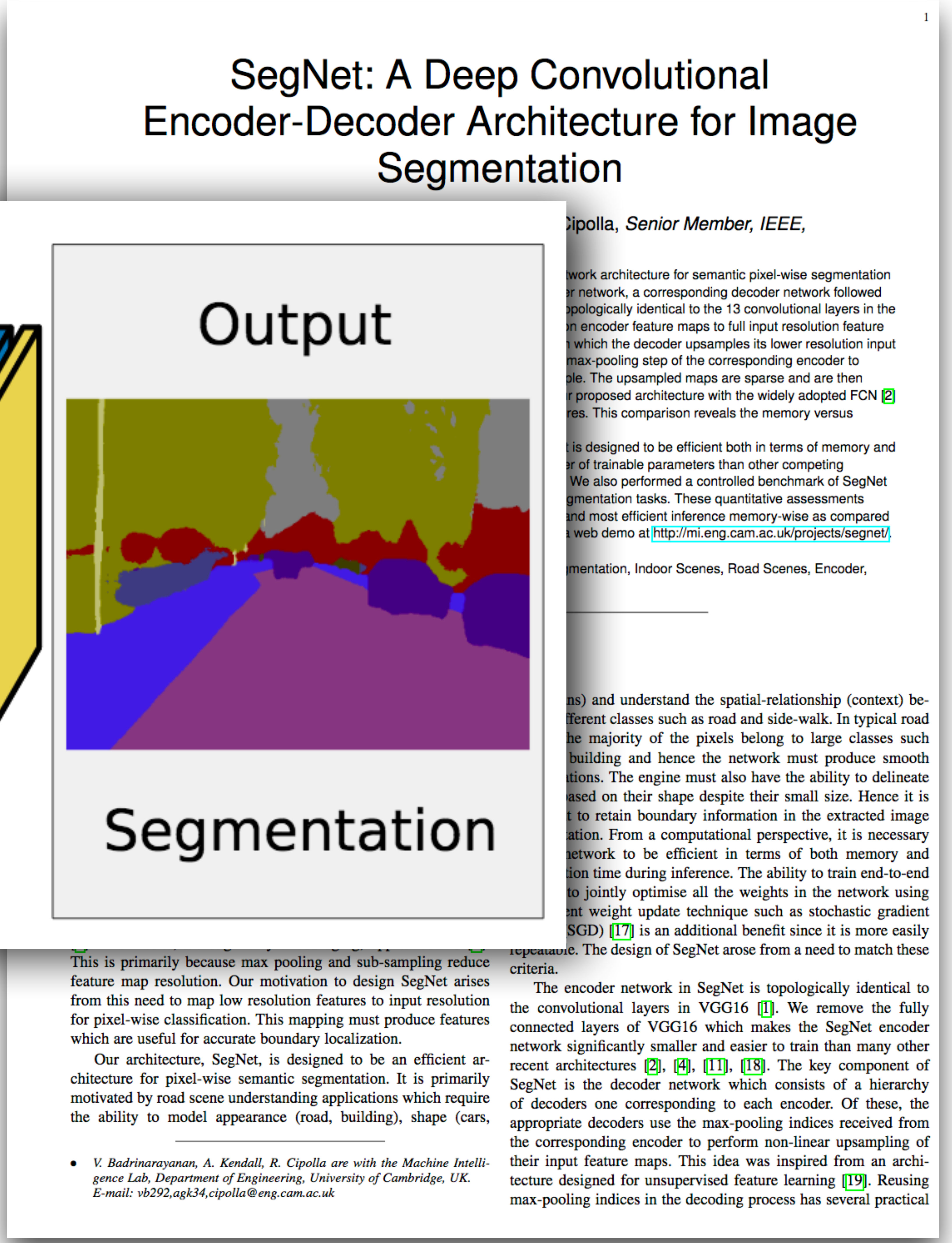

concat
...
$W \times H$
$\frac{W}{2} \times \frac{H}{2}$
$\frac{W}{2} \times \frac{H}{2}$
$W \times H$

U-net


concat
concat
...
$W \times H$
$\frac{W}{2} \times \frac{H}{2}$
$\frac{W}{4} \times \frac{H}{4}$
$\frac{W}{4} \times \frac{H}{4}$
$\frac{W}{2} \times \frac{H}{2}$
$W \times H$

Encoder-decoder architectures





1

SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

...ipolla, *Senior Member, IEEE,*

...work architecture for semantic pixel-wise segmentation
...r network, a corresponding decoder network followed
...pologically identical to the 13 convolutional layers in the
...n encoder feature maps to full input resolution feature
... which the decoder upsamples its lower resolution input
...max-pooling step of the corresponding encoder to
...ple. The upsampled maps are sparse and are then
...r proposed architecture with the widely adopted FCN [2]
...res. This comparison reveals the memory versus

...t is designed to be efficient both in terms of memory and
...r of trainable parameters than other competing
... We also performed a controlled benchmark of SegNet
...gmentation tasks. These quantitative assessments
...and most efficient inference memory-wise as compared
...a web demo at http://mi.eng.cam.ac.uk/projects/segnet/.

...mentation, Indoor Scenes, Road Scenes, Encoder,

This is primarily because max pooling and sub-sampling reduce feature map resolution. Our motivation to design SegNet arises from this need to map low resolution features to input resolution for pixel-wise classification. This mapping must produce features which are useful for accurate boundary localization.

Our architecture, SegNet, is designed to be an efficient architecture for pixel-wise semantic segmentation. It is primarily motivated by road scene understanding applications which require the ability to model appearance (road, building), shape (cars,

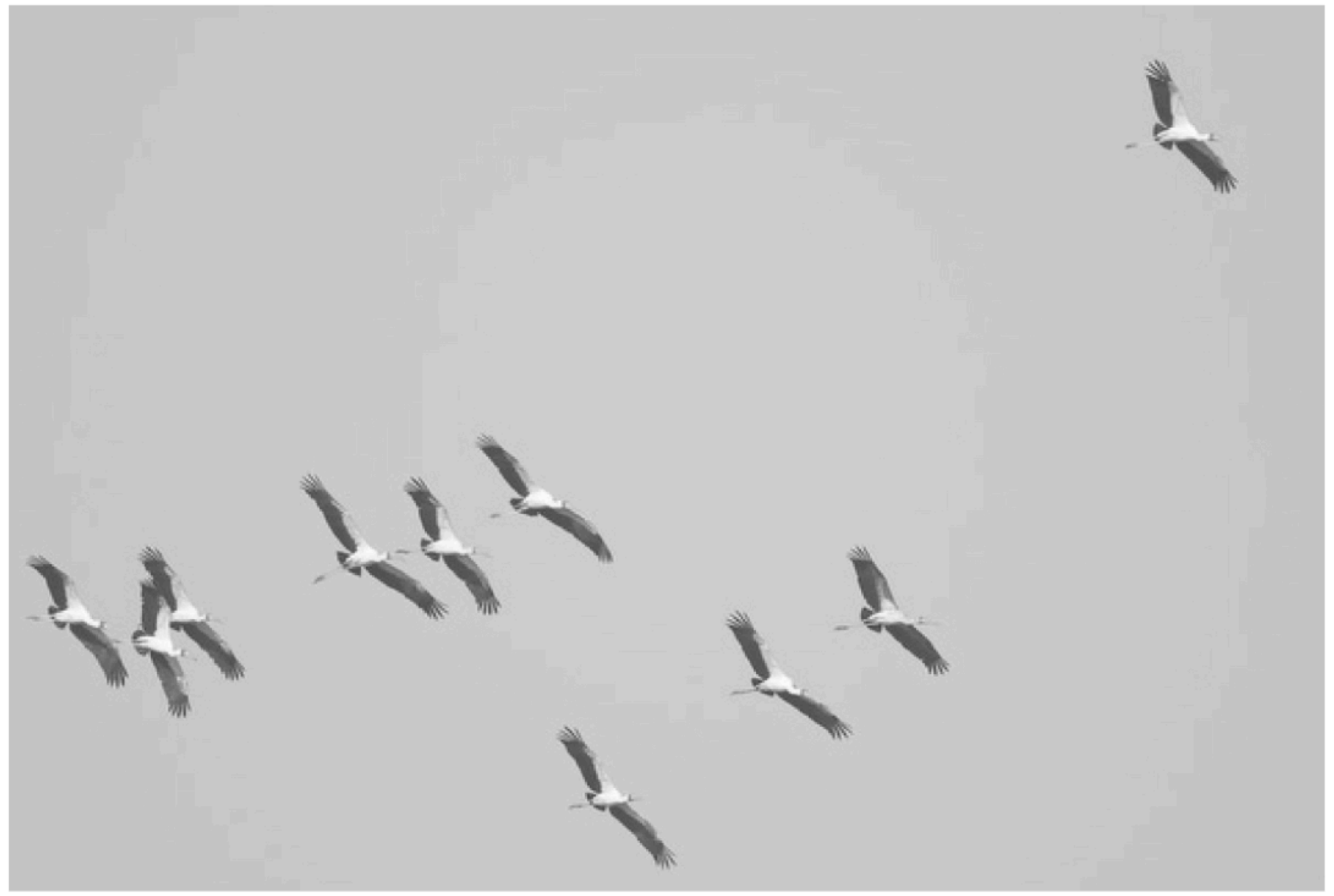
...ns) and understand the spatial-relationship (context) be-
...fferent classes such as road and side-walk. In typical road
...he majority of the pixels belong to large classes such
...building and hence the network must produce smooth
...tions. The engine must also have the ability to delineate
...ased on their shape despite their small size. Hence it is
...t to retain boundary information in the extracted image
...ation. From a computational perspective, it is necessary
...etwork to be efficient in terms of both memory and
...ion time during inference. The ability to train end-to-end
...to jointly optimise all the weights in the network using
...nt weight update technique such as stochastic gradient
...SGD) [17] is an additional benefit since it is more easily repeatable. The design of SegNet arose from a need to match these criteria.

The encoder network in SegNet is topologically identical to the convolutional layers in VGG16 [1]. We remove the fully connected layers of VGG16 which makes the SegNet encoder network significantly smaller and easier to train than many other recent architectures [2], [4], [11], [18]. The key component of SegNet is the decoder network which consists of a hierarchy of decoders one corresponding to each encoder. Of these, the appropriate decoders use the max-pooling indices received from the corresponding encoder to perform non-linear upsampling of their input feature maps. This idea was inspired from an architecture designed for unsupervised feature learning [19]. Reusing max-pooling indices in the decoding process has several practical

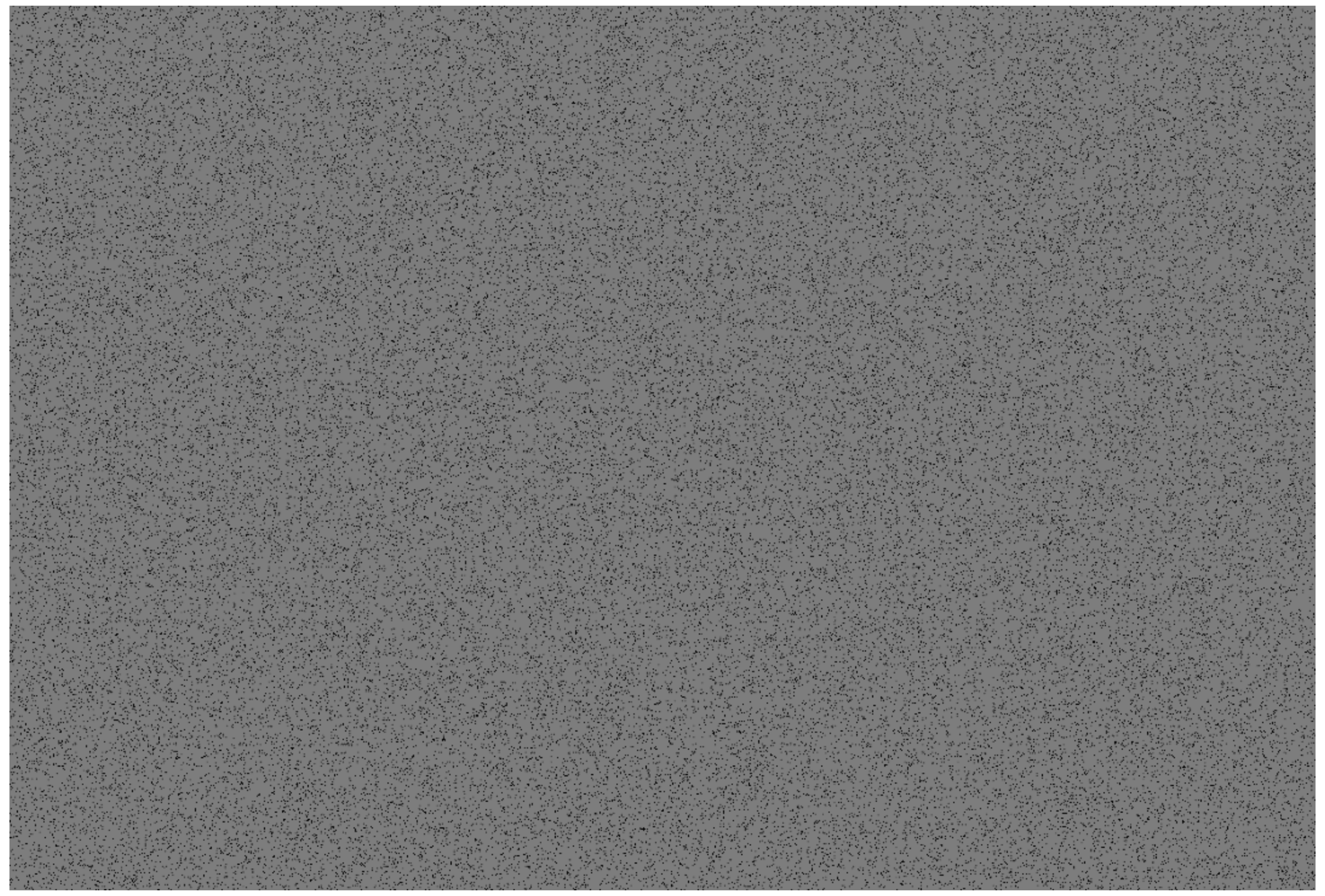
- *V. Badrinarayanan, A. Kendall, R. Cipolla are with the Machine Intelligence Lab, Department of Engineering, University of Cambridge, UK. E-mail: vb292,agk34,cipolla@eng.cam.ac.uk*

Source: Torralba, Freeman, Isola

Other uses for U-nets



+



=

Goal: recover the original image

Recall: denoising problem

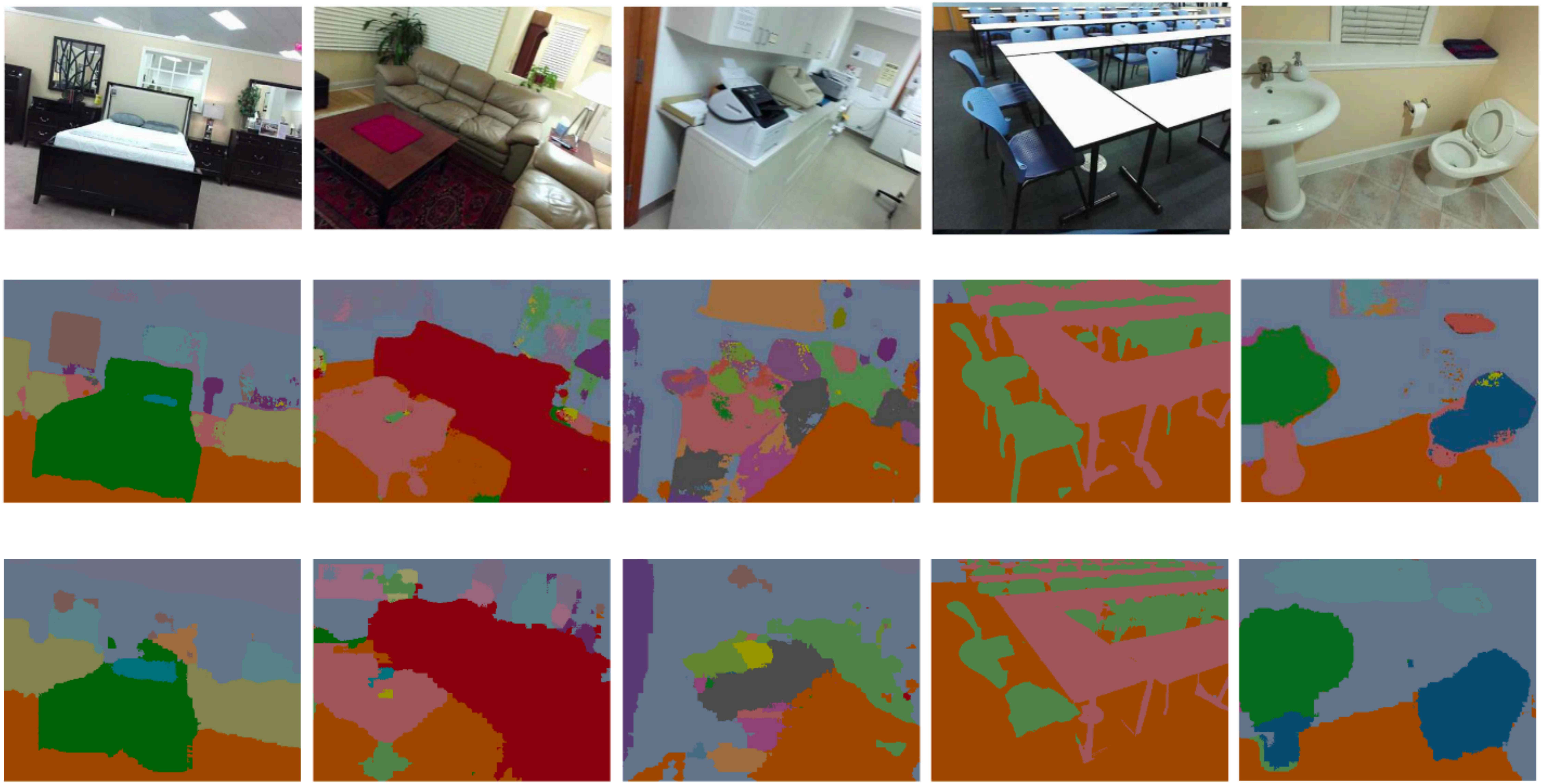
Denoising

Loss: $\|\mathbf{x}_{\text{clean}} - \hat{\mathbf{x}}\|^2$

Input

Segnet

FCN



Source: Torralba, Freeman, Isola

PS5: CNNs

- Train a convolutional network to recognize scenes
- Use PyTorch + autodiff
- Train on GPU

Next class: image generation with GANs