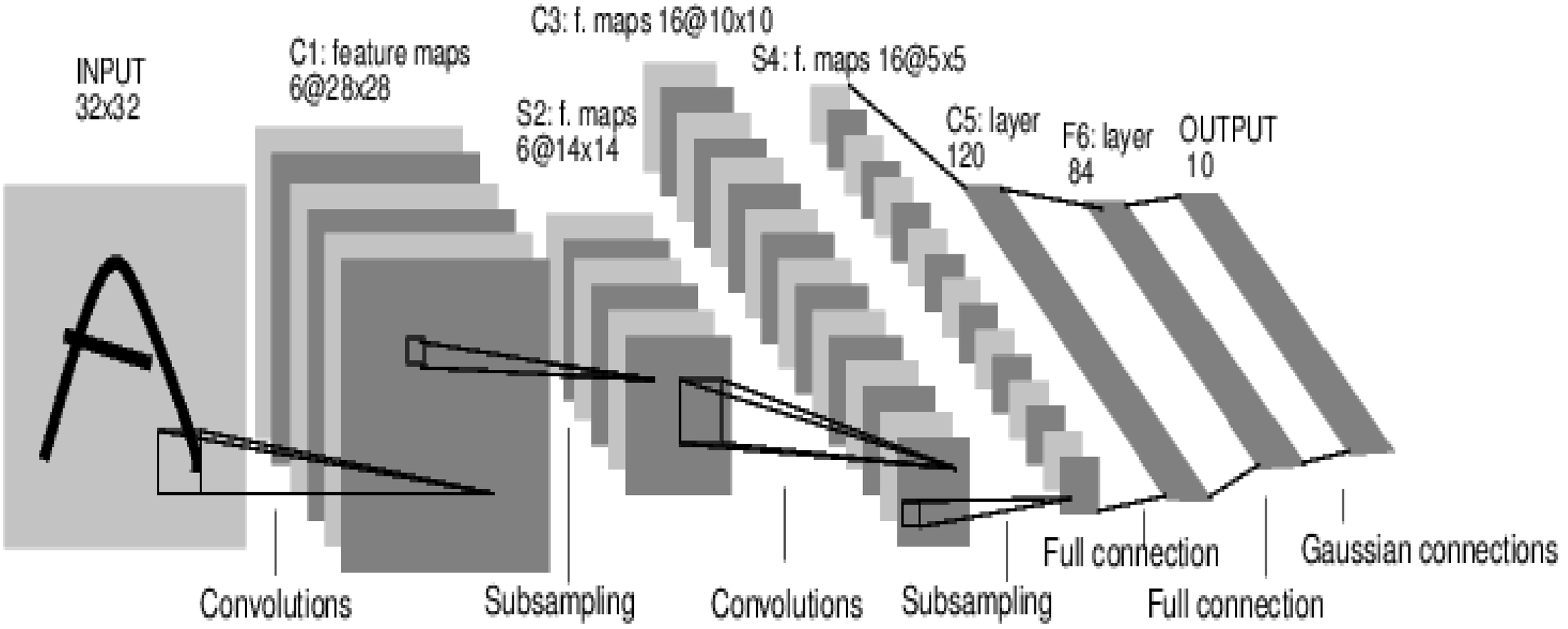
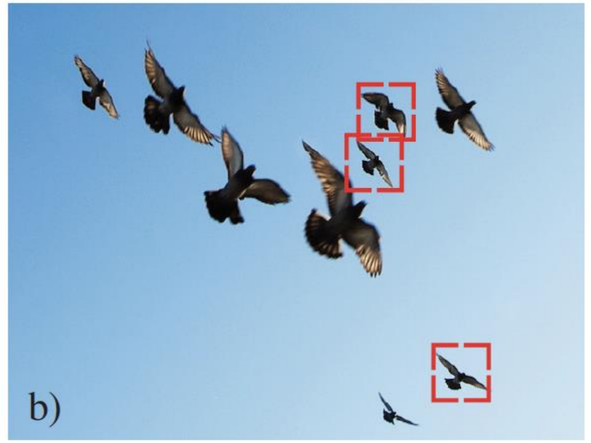
ConvNets, then and now

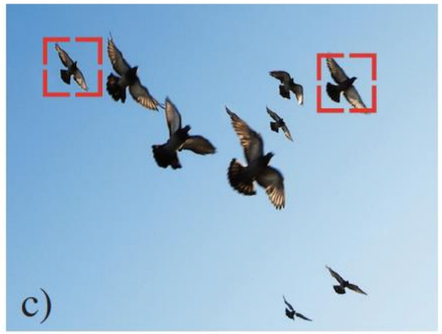


Pooling and downsampling

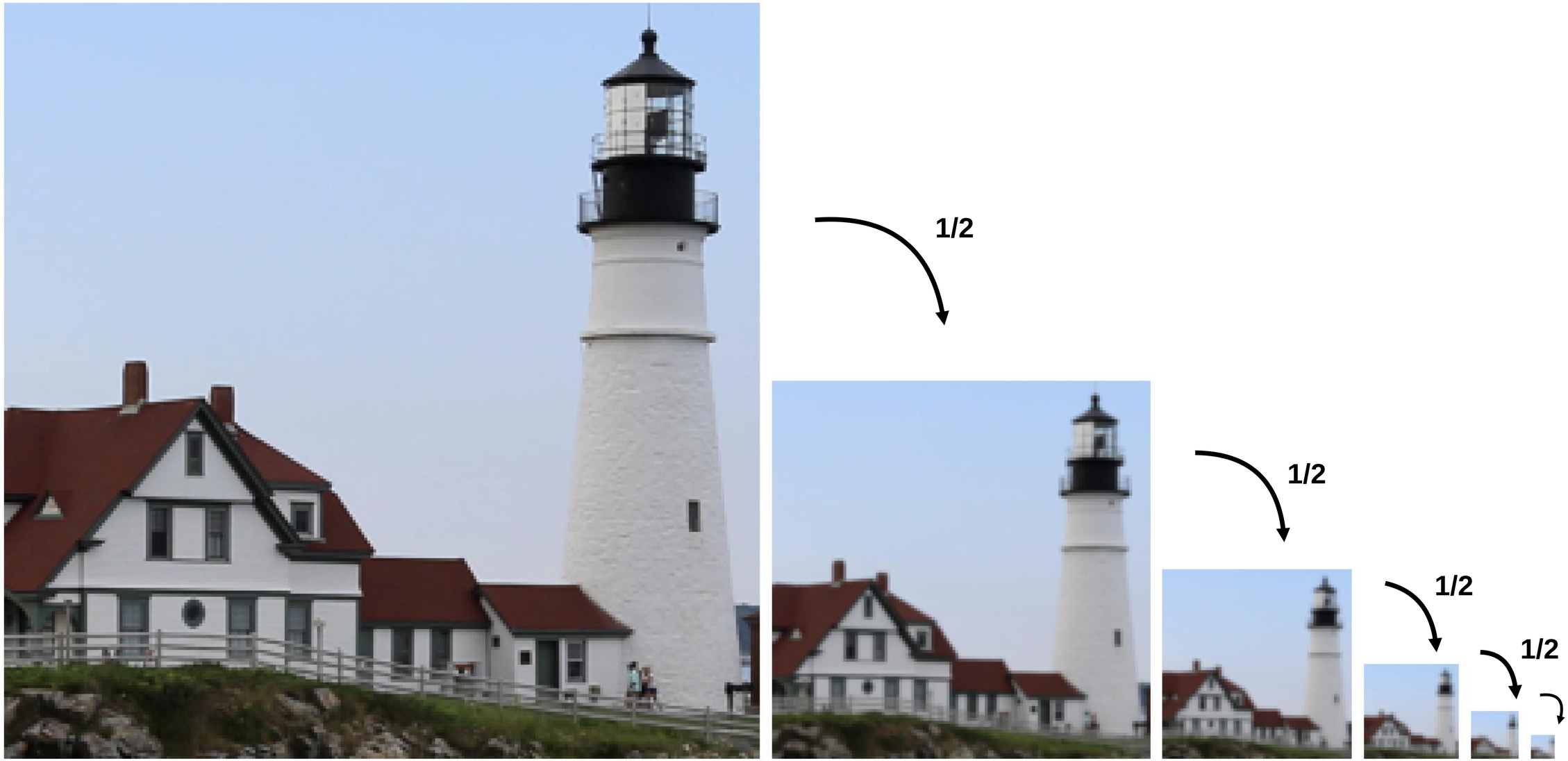
We need translation and **scale** invariance

Image pyramids

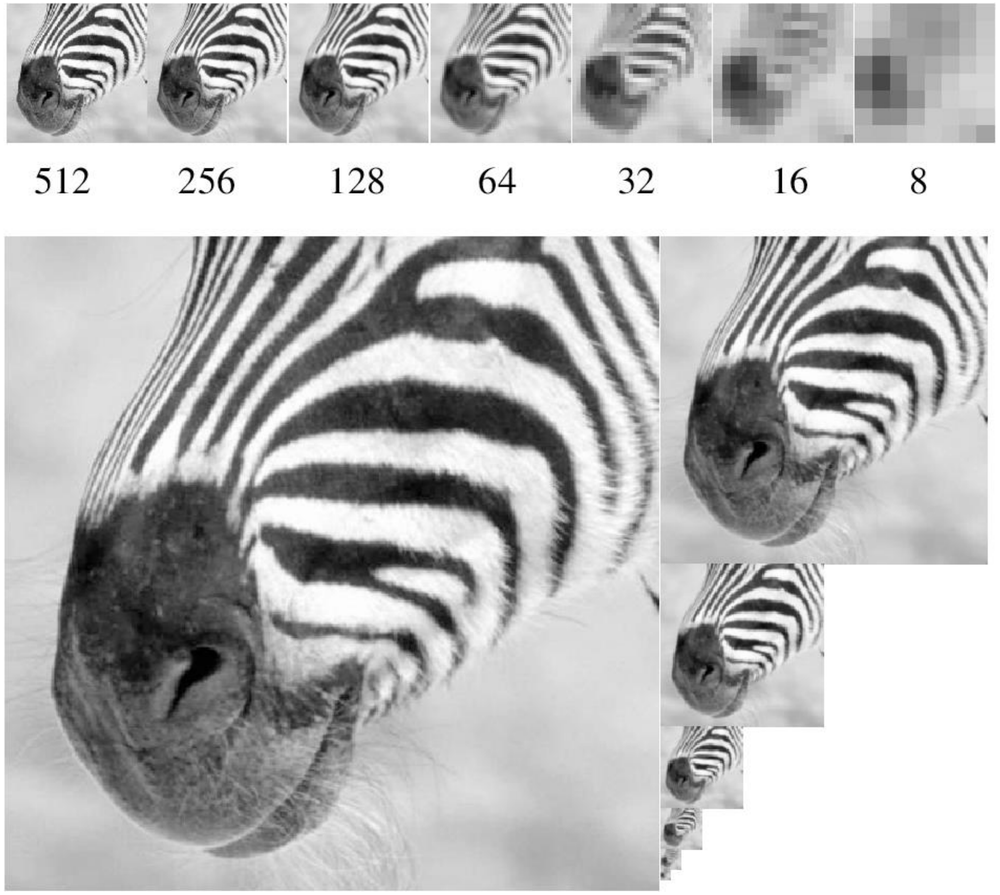




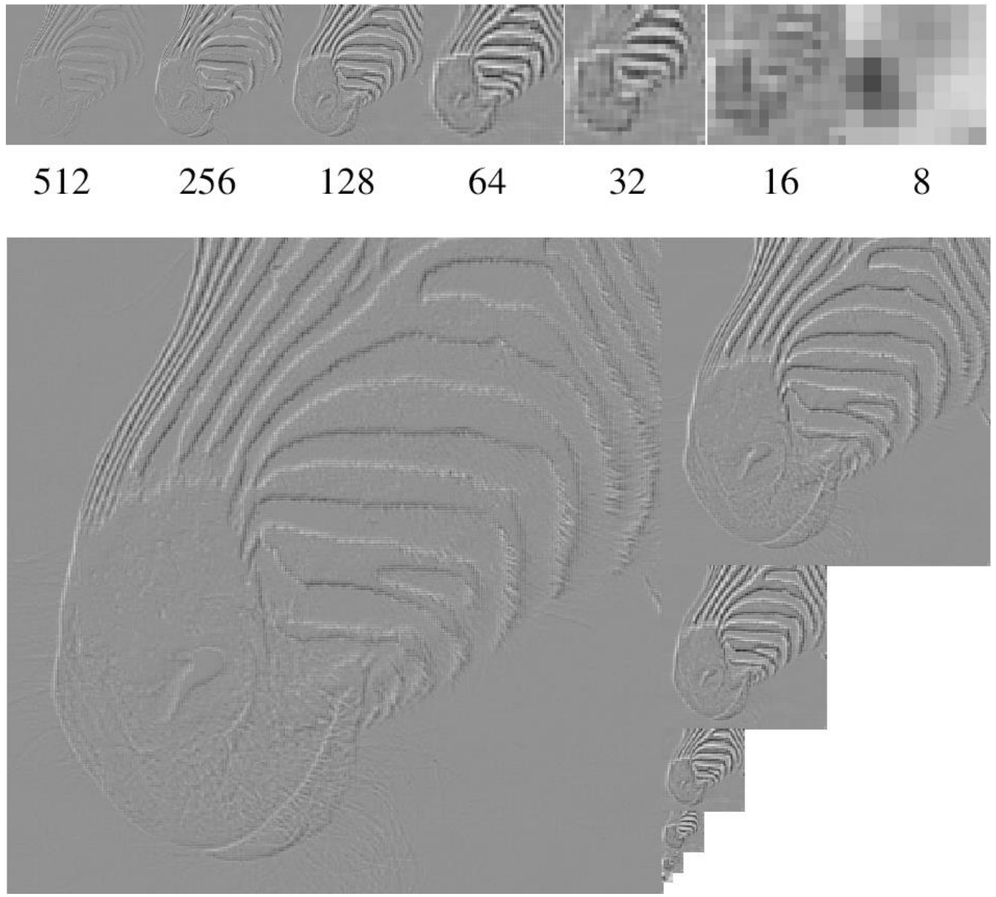
Gaussian Pyramid



Multiscale representations are great!



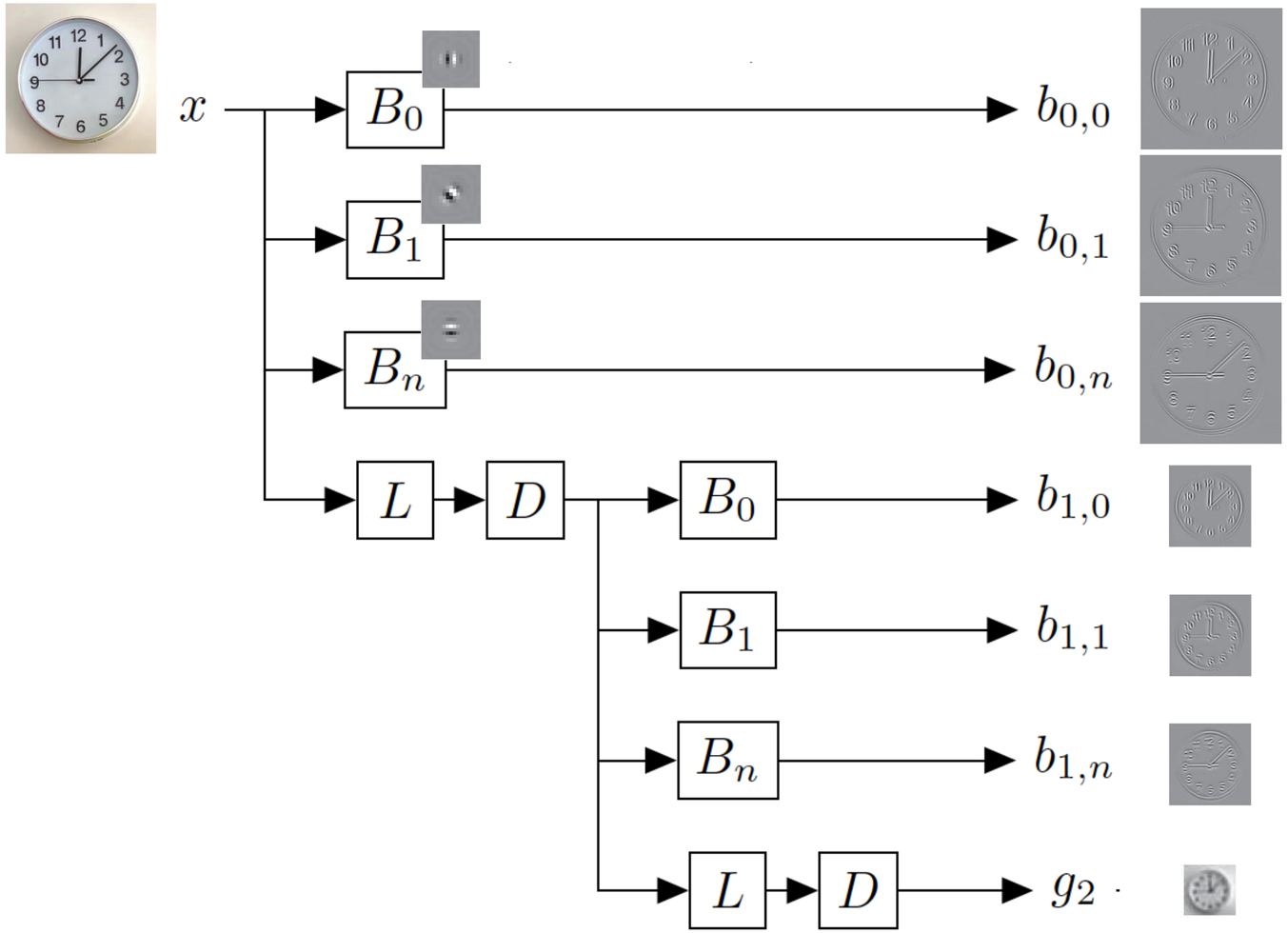
Gaussian Pyr

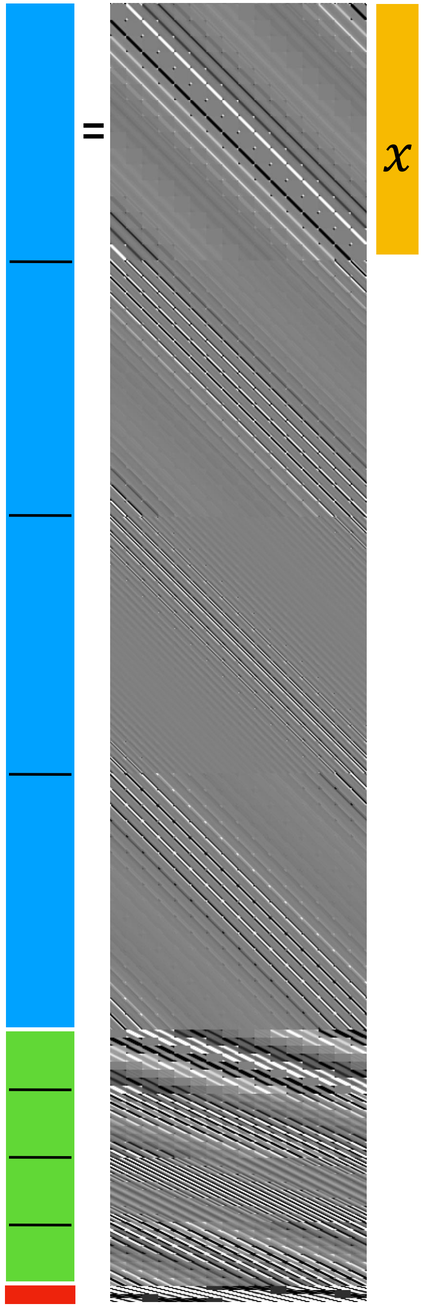


Laplacian Pyr

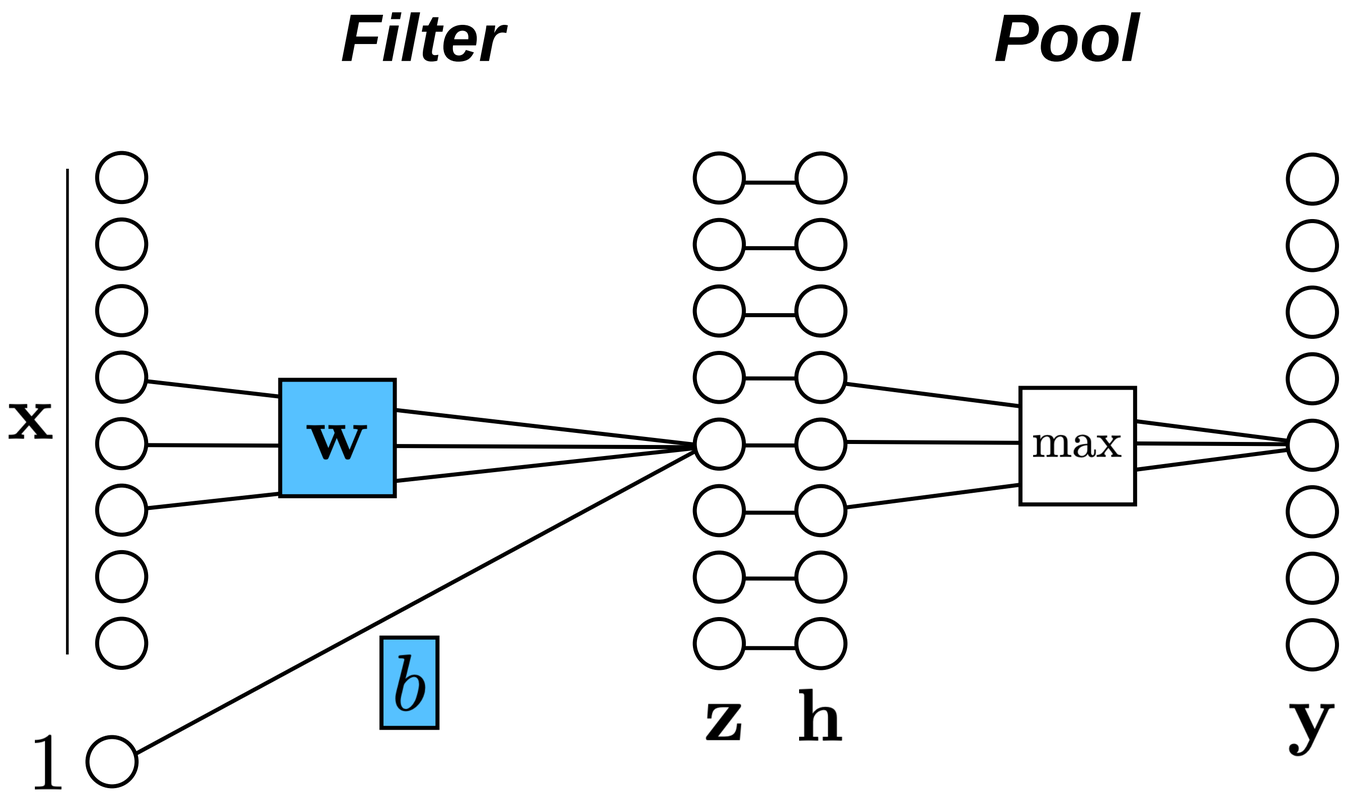
How can we use multi-scale modeling in Convnets?

Steerable Pyramid





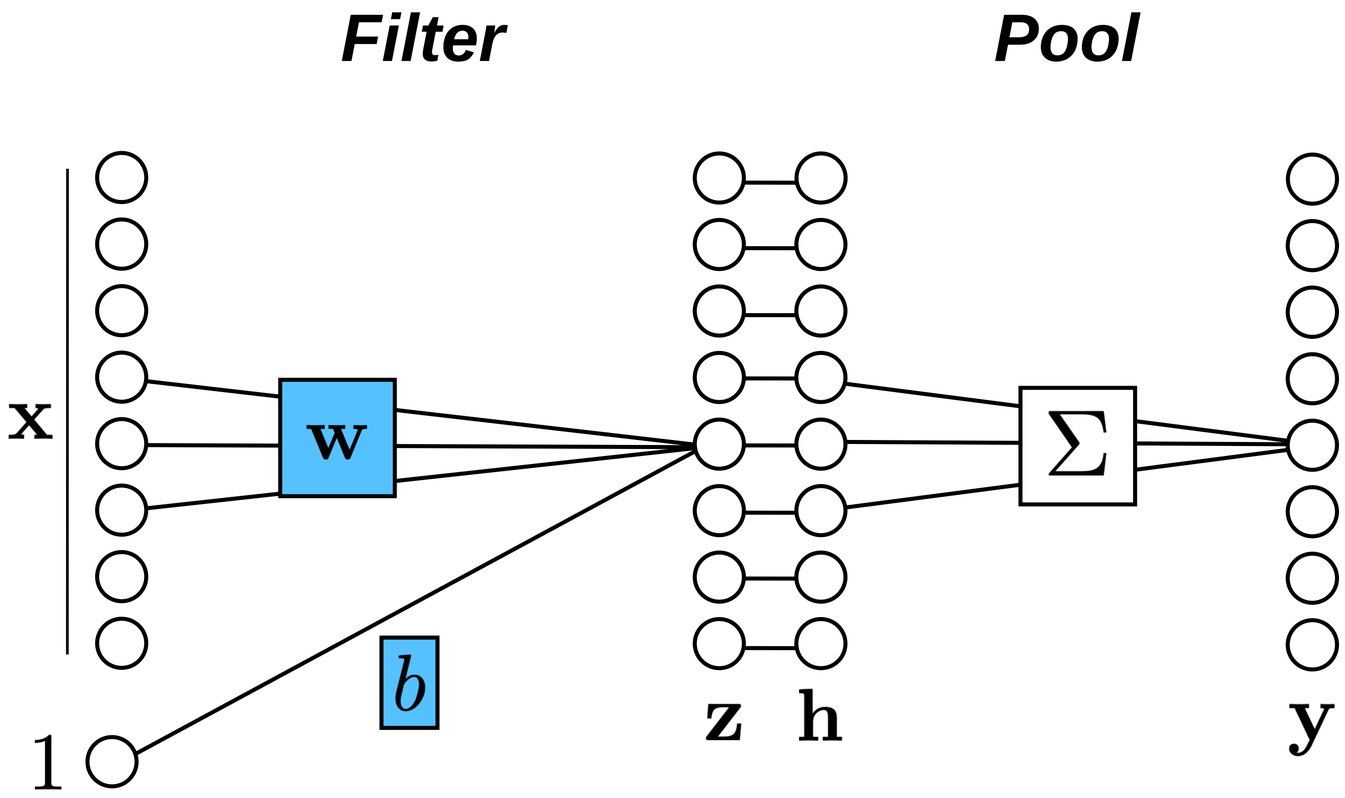
Pooling



Max pooling

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

Pooling



Max pooling

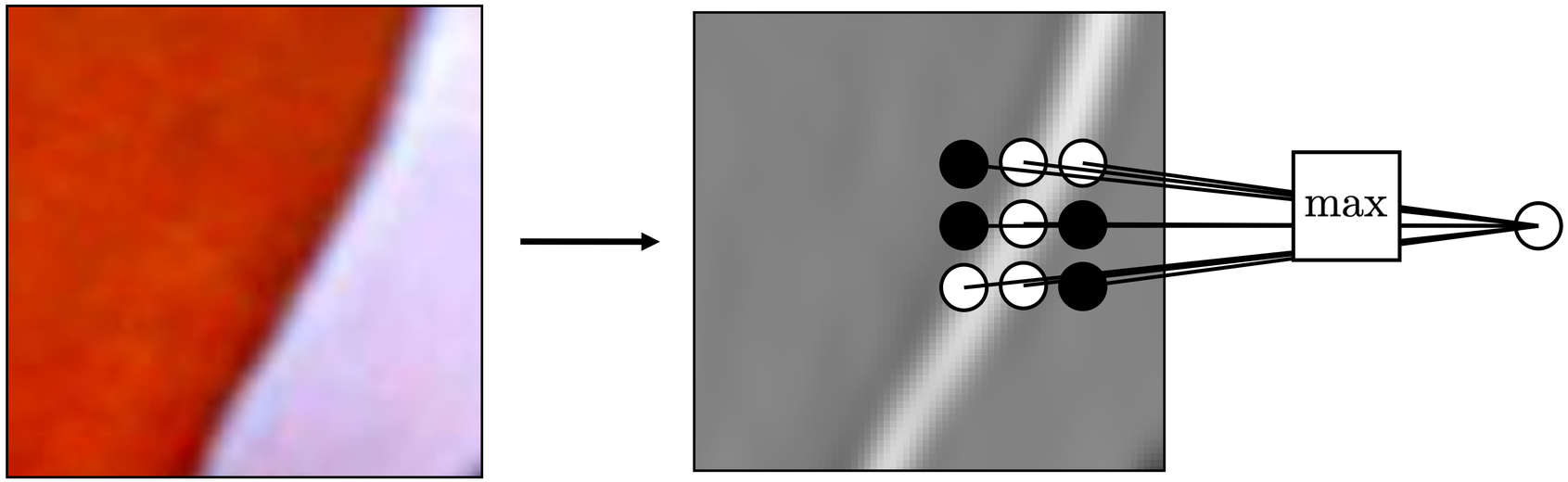
$$y_j = \max_{j \in \mathcal{N}(j)} h_j$$

Mean pooling

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

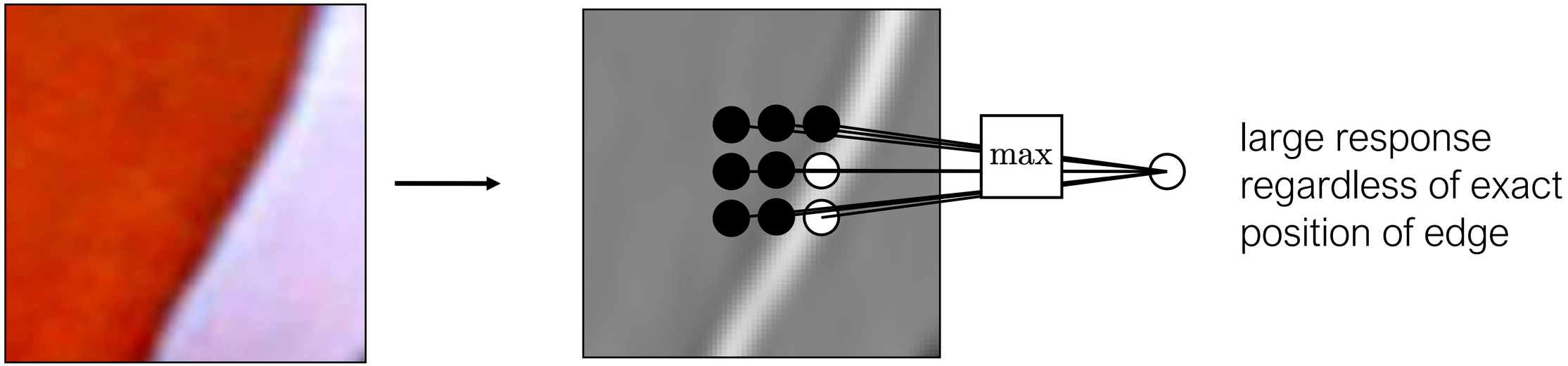
Pooling — Why?

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



Pooling — Why?

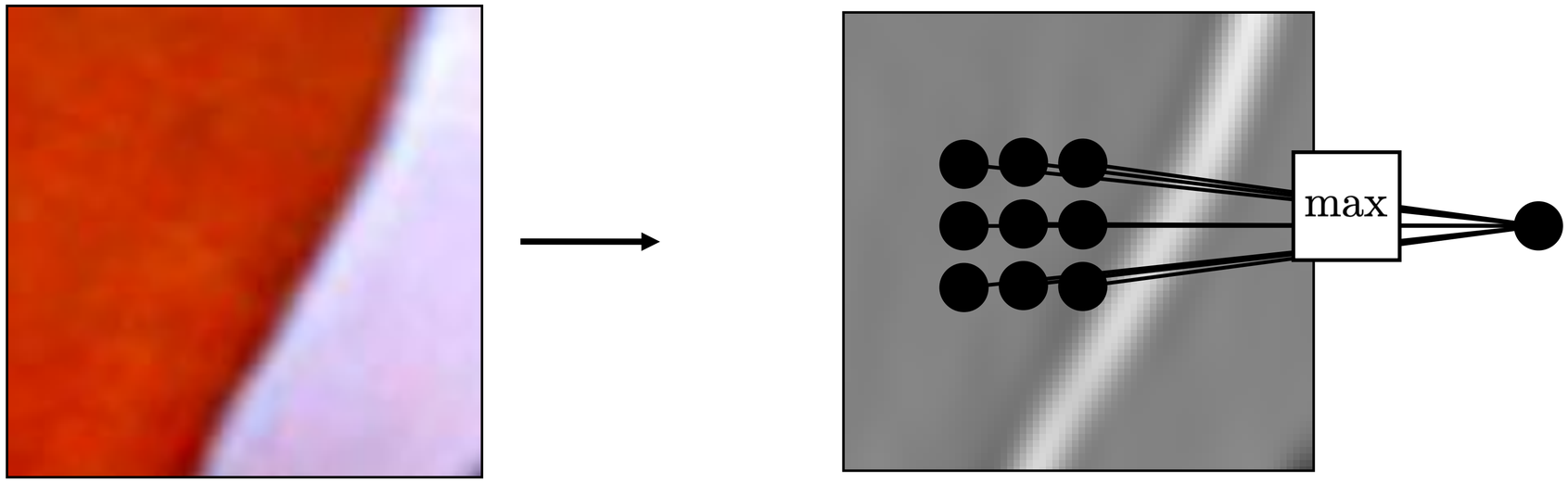
Pooling across spatial locations achieves stability w.r.t. small translations:



large response regardless of exact position of edge

Pooling — Why?

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

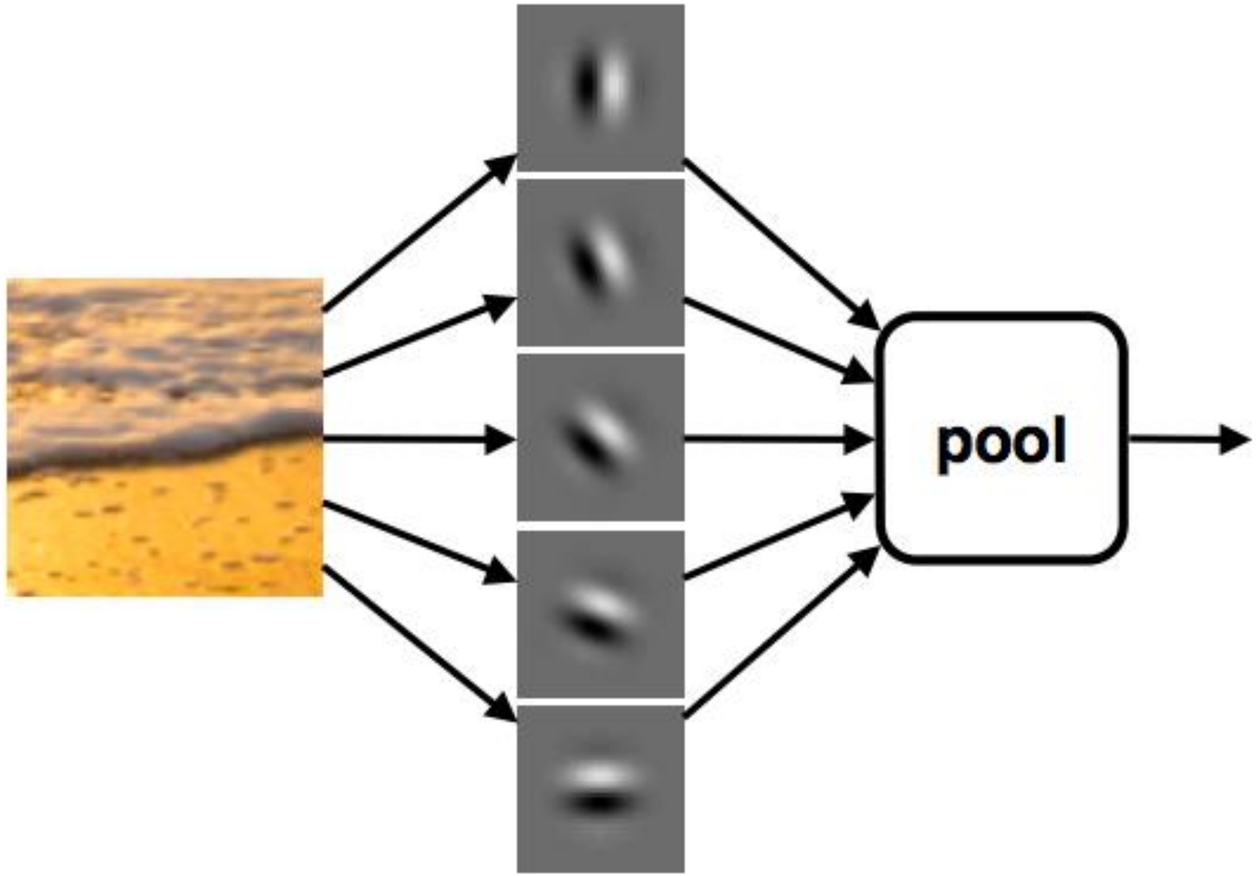


CNNs are stable w.r.t. diffeomorphisms

[“Unreasonable effectiveness of Deep Features as a Perceptual Metric”, Zhang et al. 2018]

Pooling *across channels* — Why?

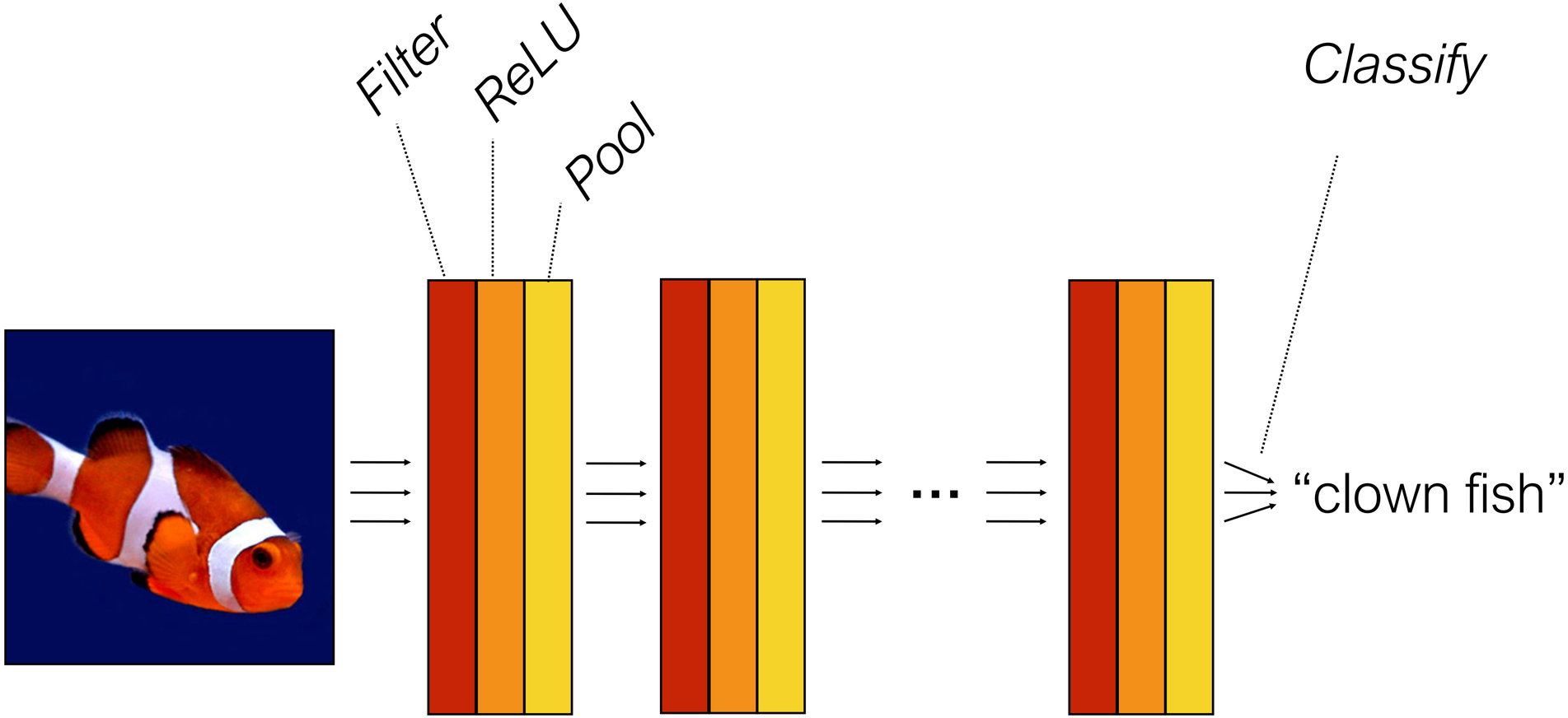
Pooling across feature channels (filter outputs)
can achieve other kinds of invariances:



large response
for any edge,
regardless of its
orientation

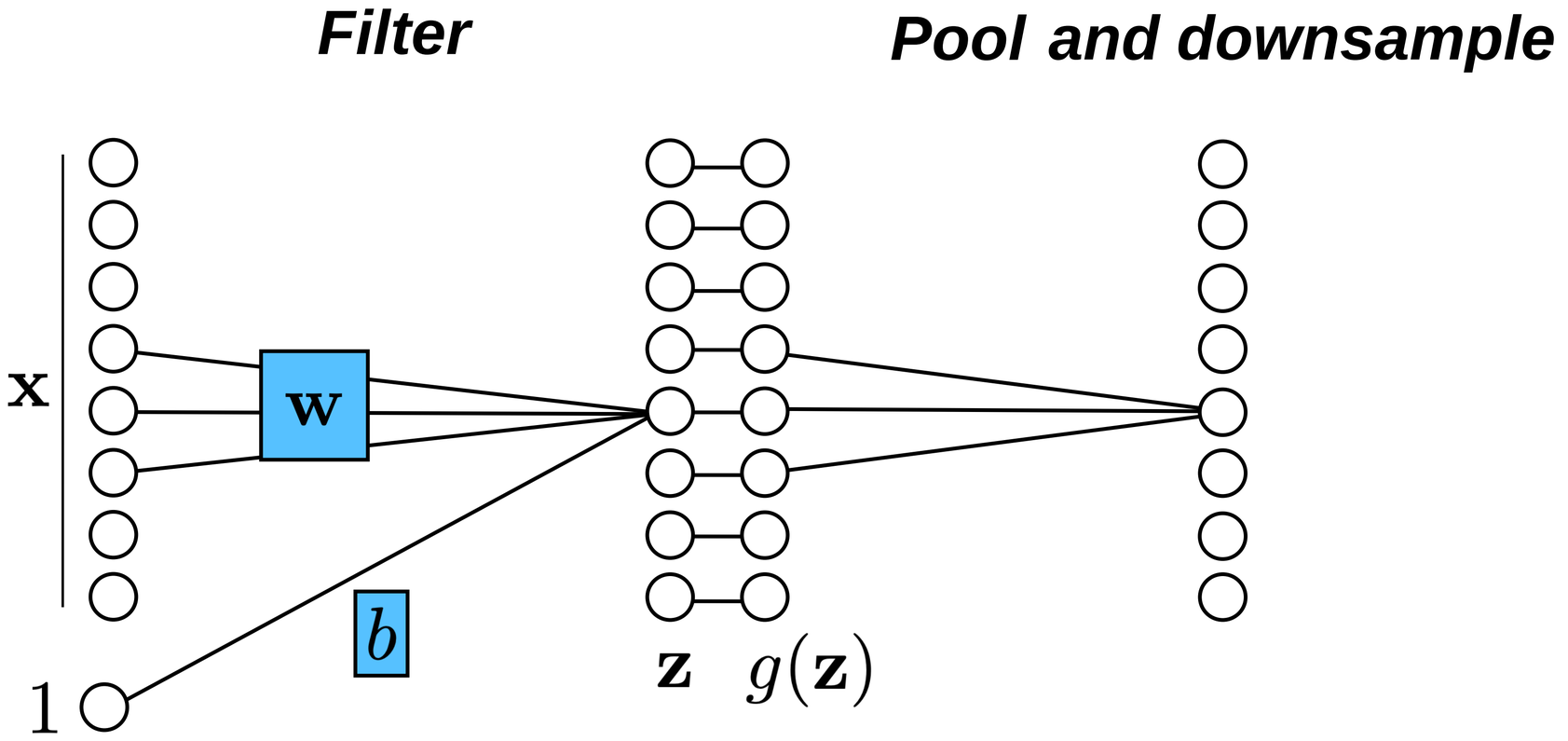
[Derived from slide by Andrea Vedaldi]

Computation in a neural net

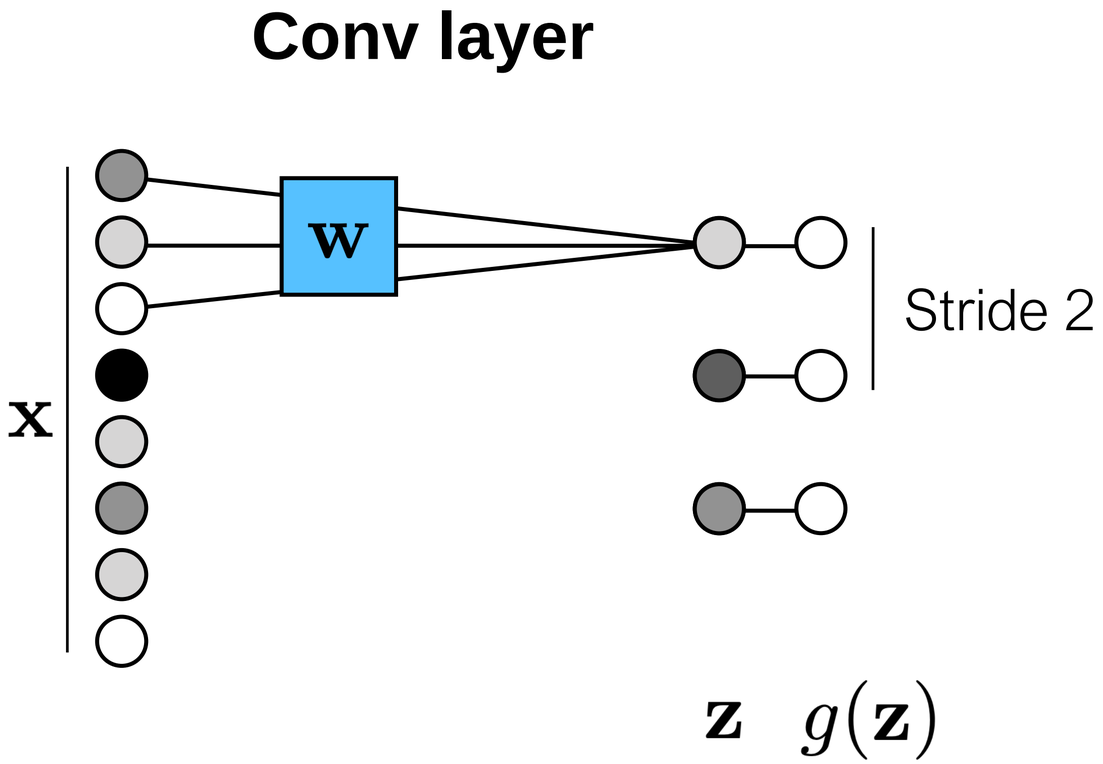


$$f(\mathbf{x}) = f_L(\dots f_2(f_1(\mathbf{x})))$$

Downsampling

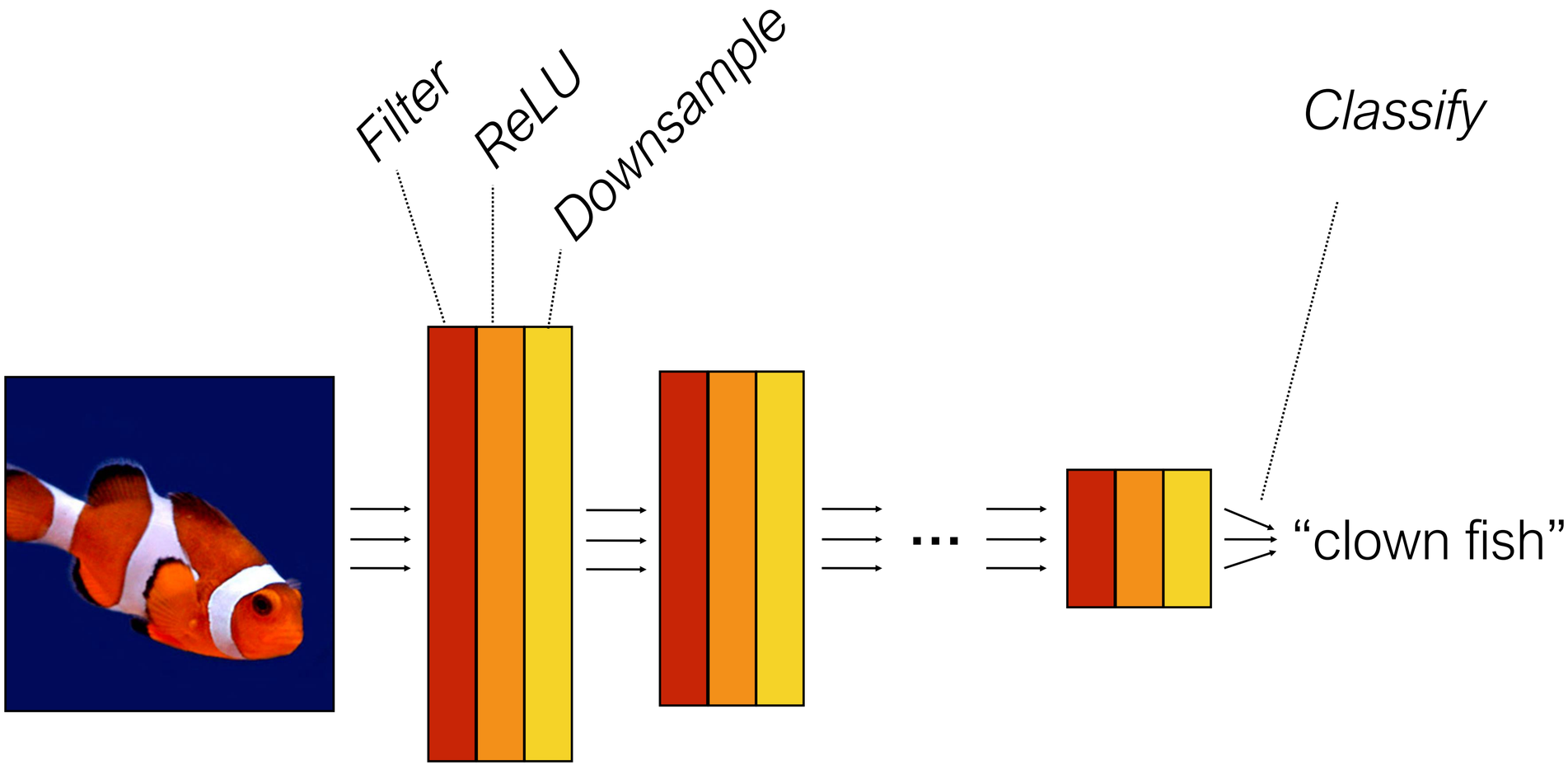


Strided operations



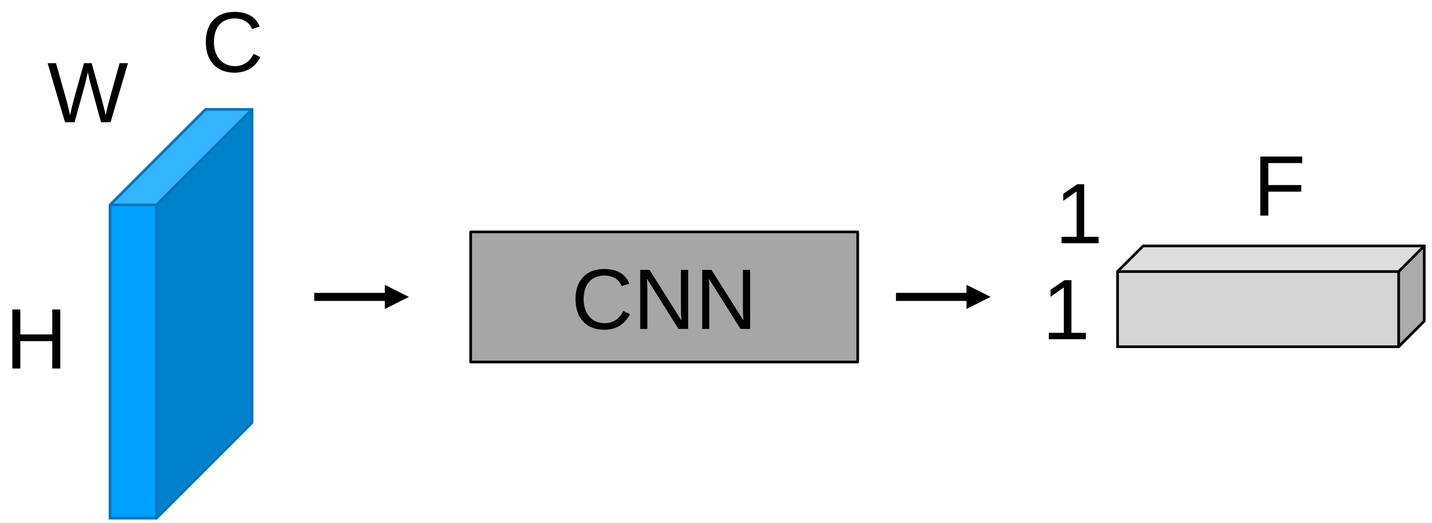
Strided operations combine a given operation (convolution or pooling) and downsampling into a single operation.

Computation in a neural net



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

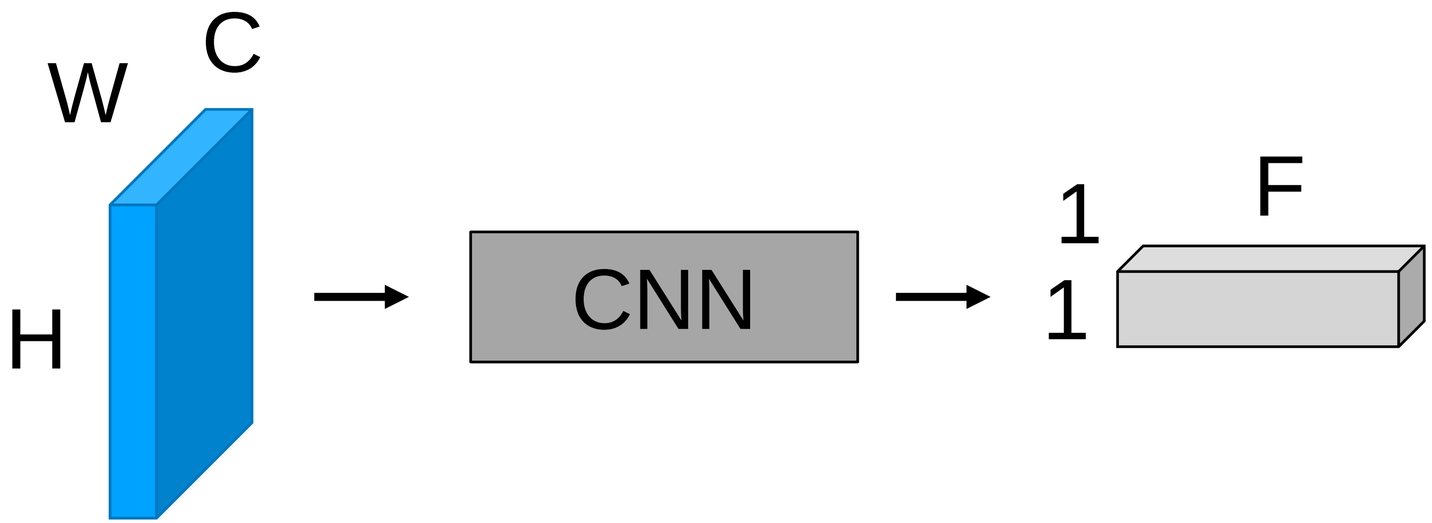
CNN as a classifier



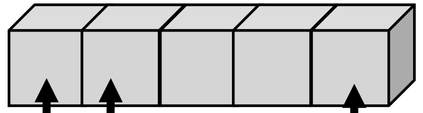
Convert HxW image into a F-dimensional vector

- What's the probability this image is a cat (F=1)
- Which of 1000 categories is this image? (F=1000)
- At what GPS coord was this image taken? (F=2)
- Identify the X,Y coordinates of 28 body joints of an image of a human (F=56)

CNN as a classifier



Running example:
image classification

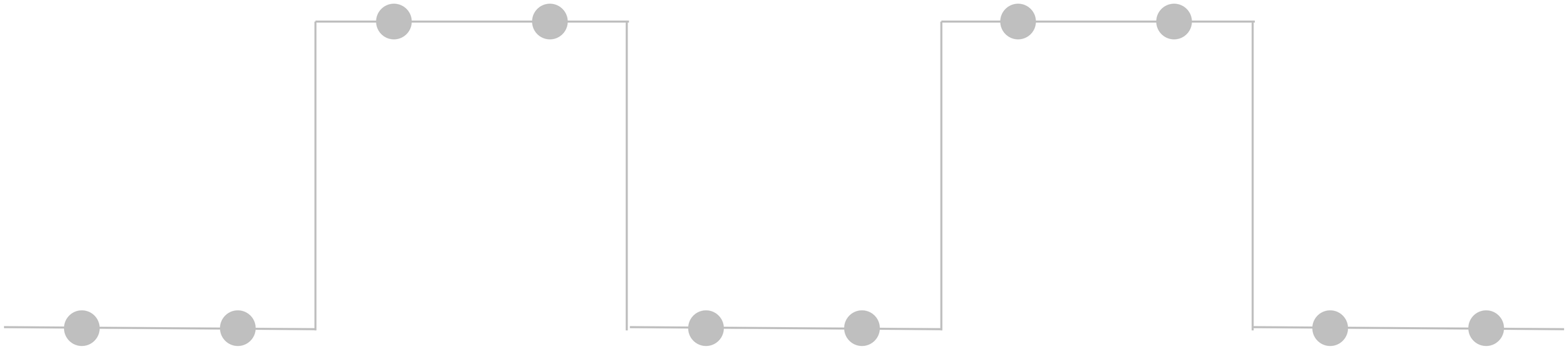


P(image is class #1)

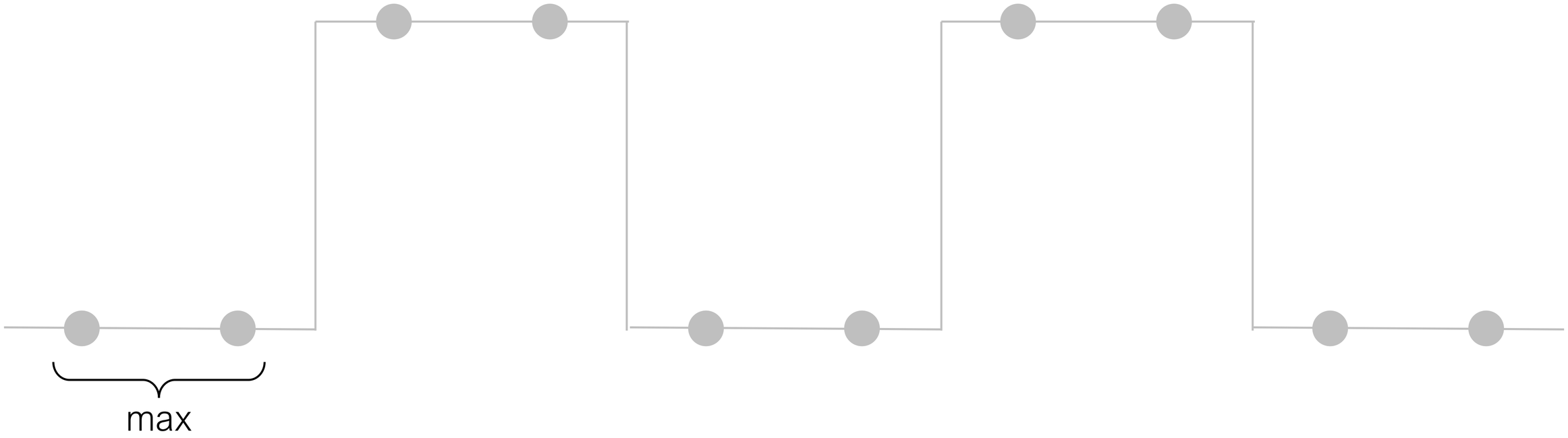
P(image is class #2)

P(image is class #F)

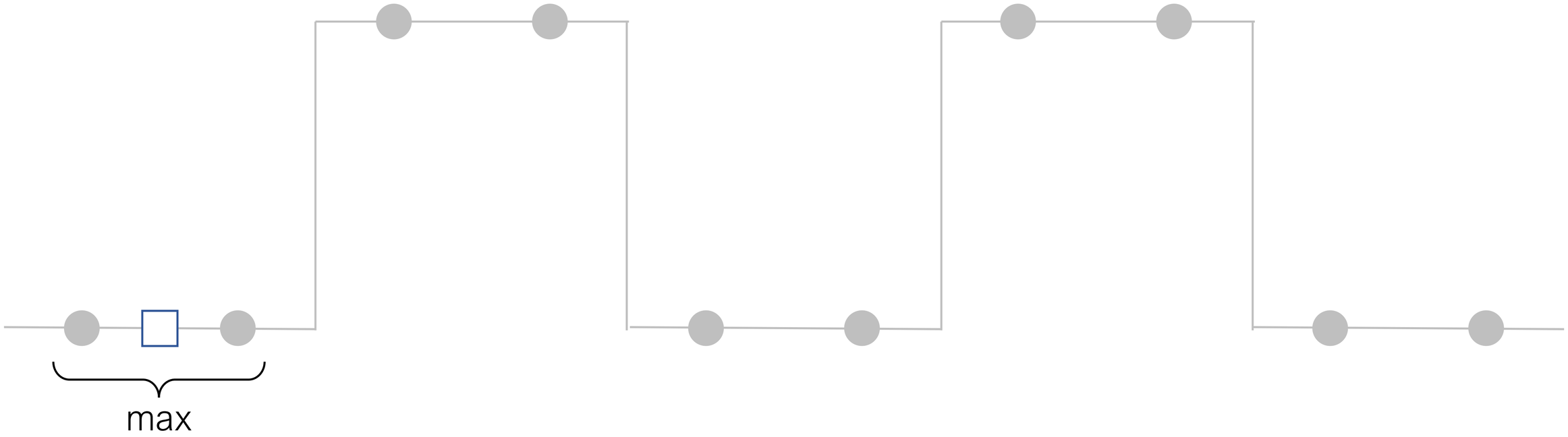
Re-examining Max-Pooling



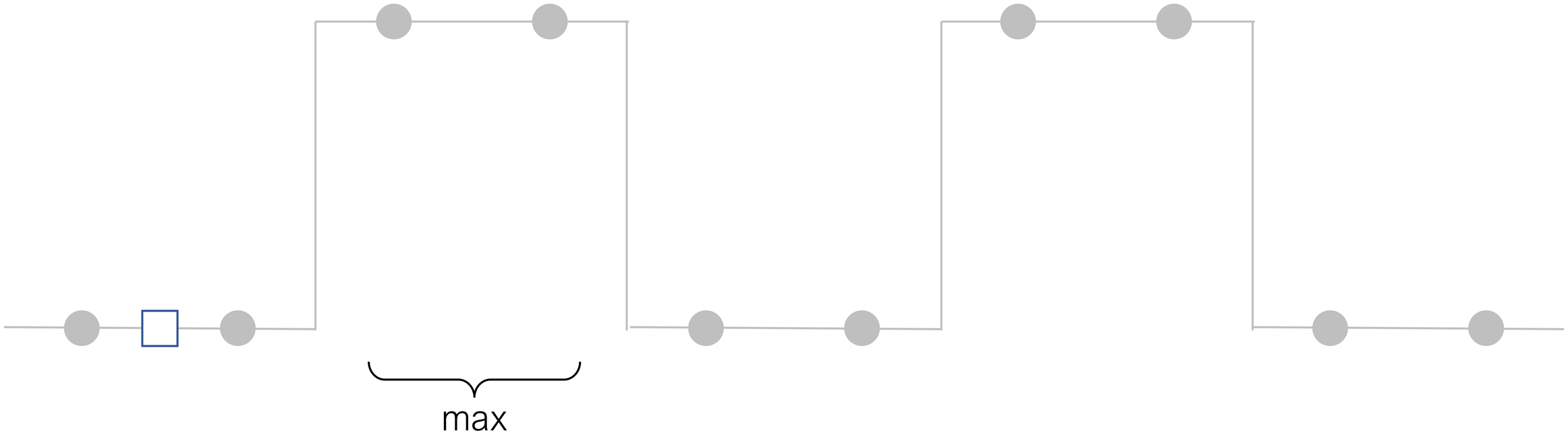
Re-examining Max-Pooling



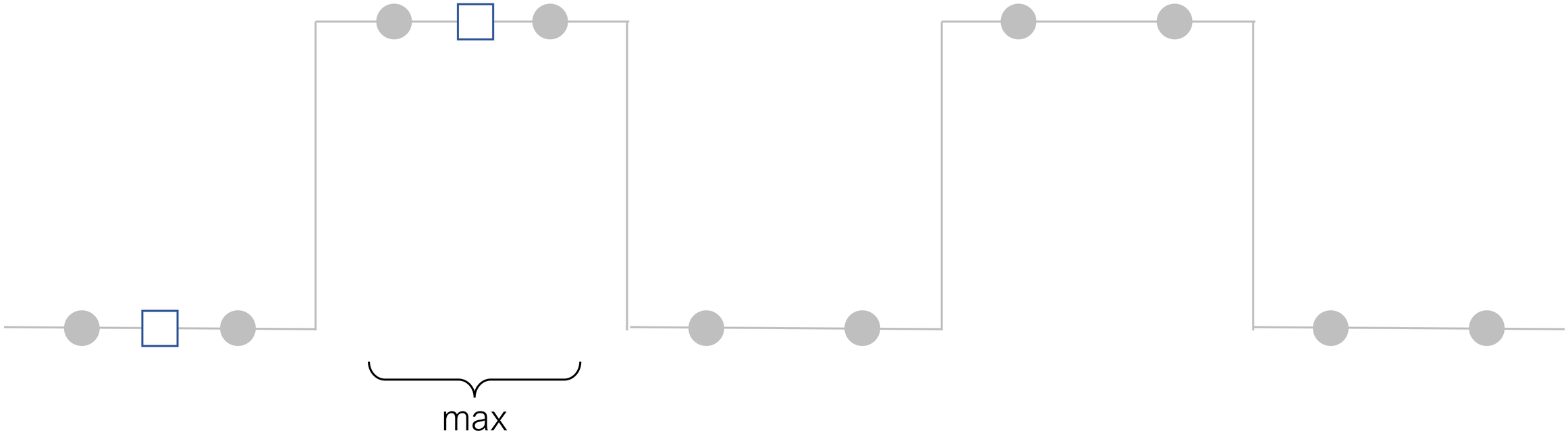
Re-examining Max-Pooling



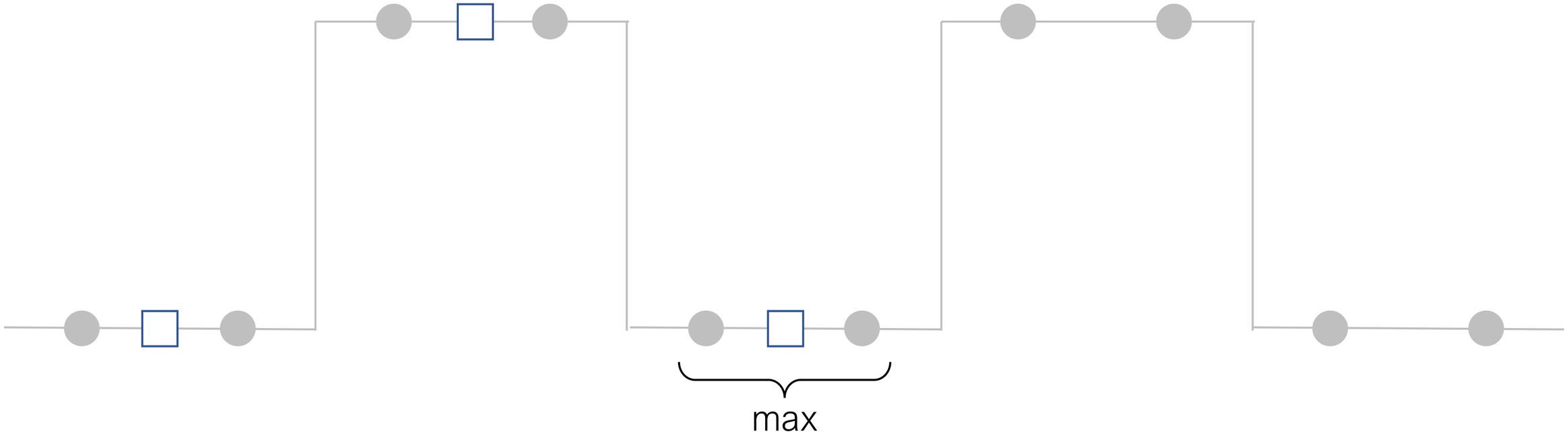
Re-examining Max-Pooling



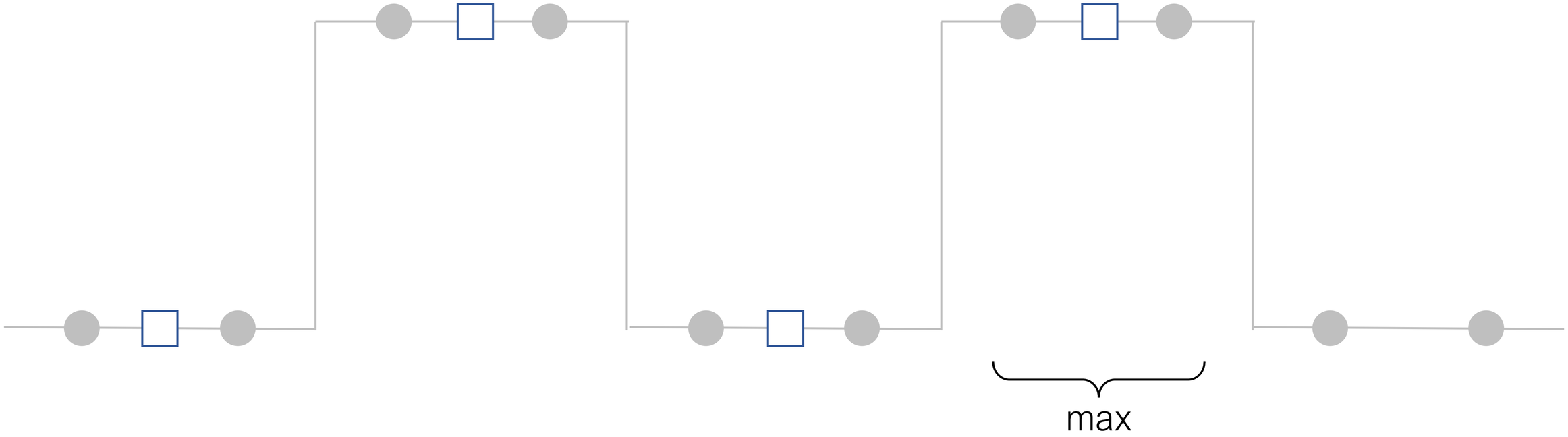
Re-examining Max-Pooling



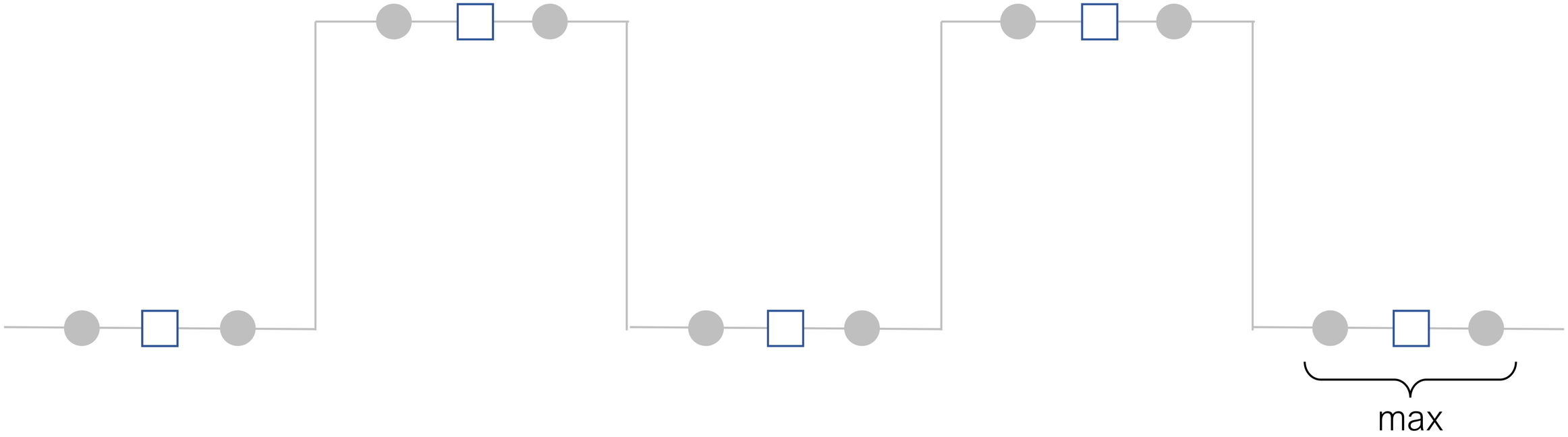
Re-examining Max-Pooling



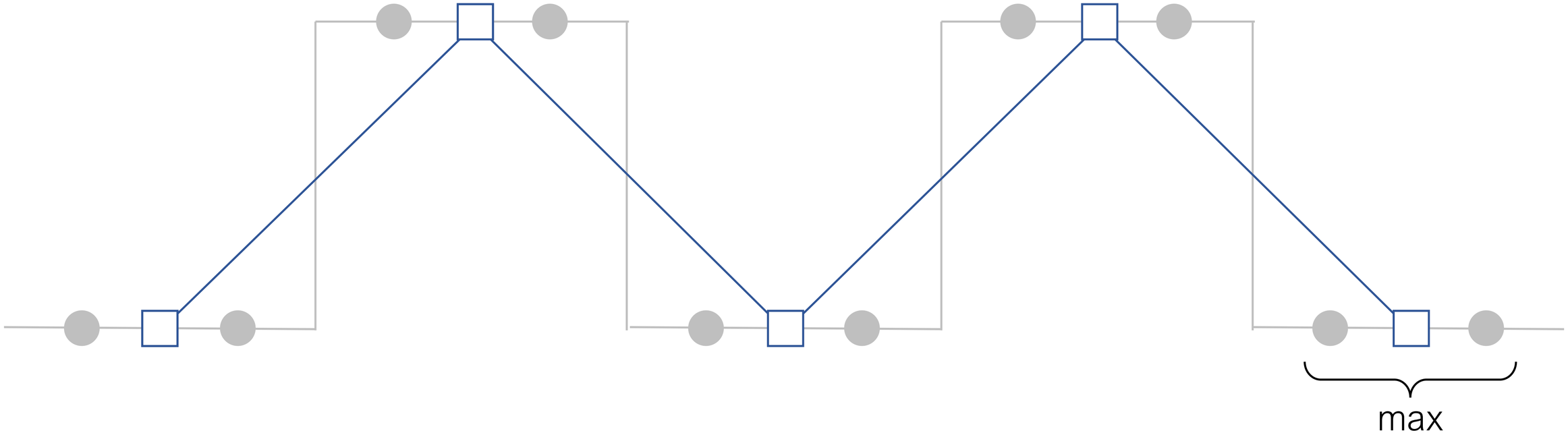
Re-examining Max-Pooling



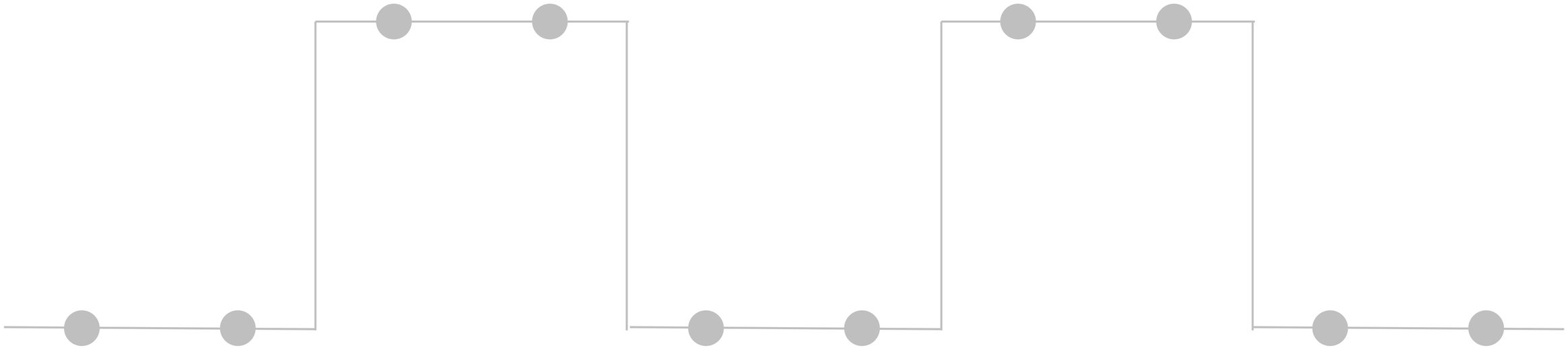
Re-examining Max-Pooling



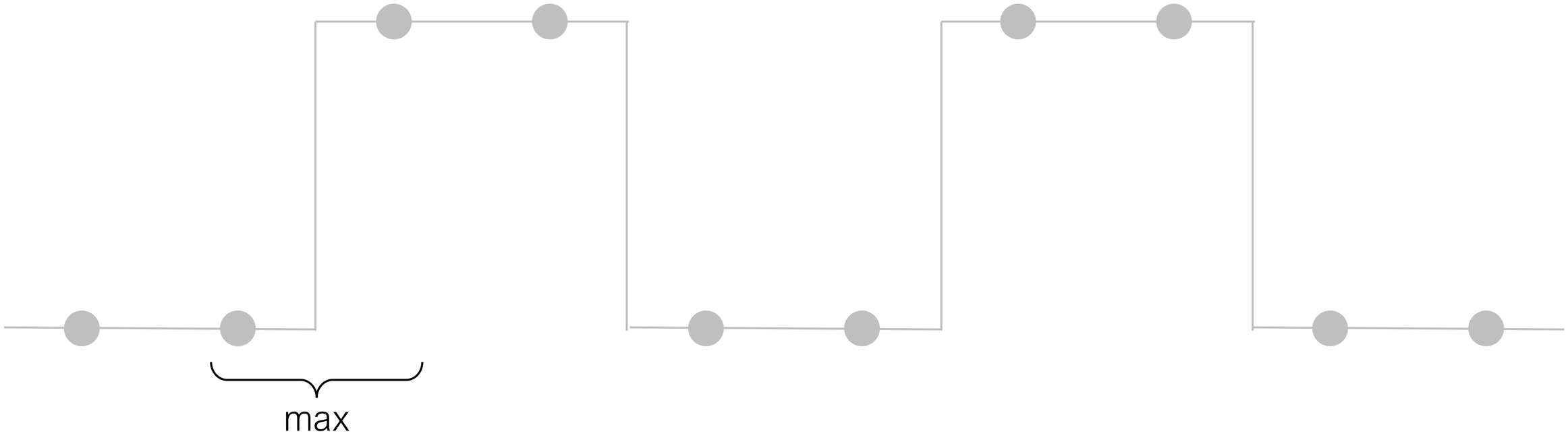
Re-examining Max-Pooling



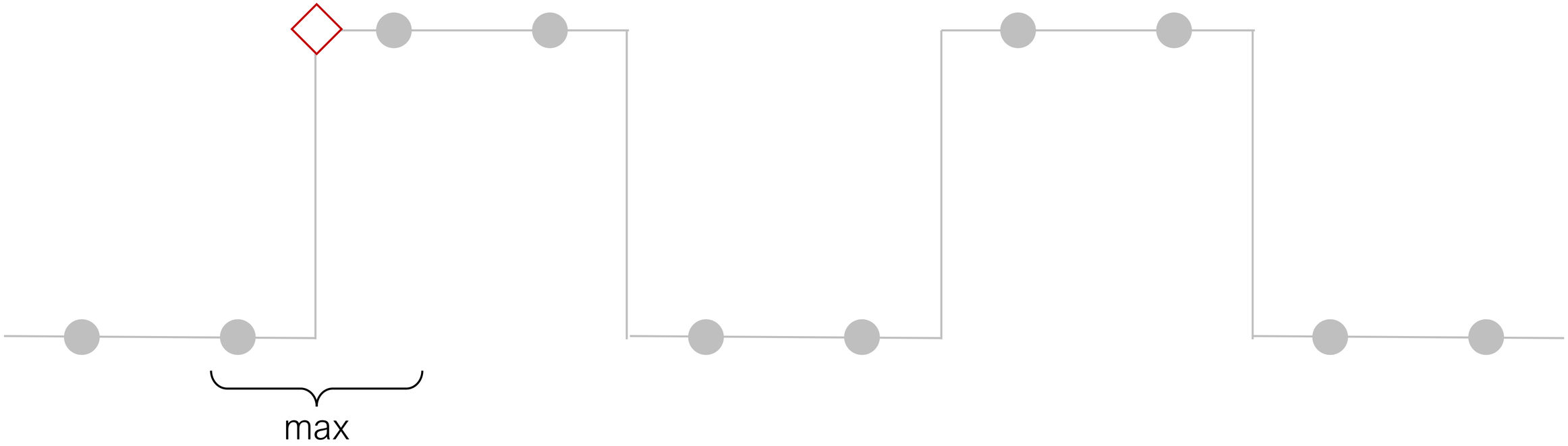
Re-examining Max-Pooling



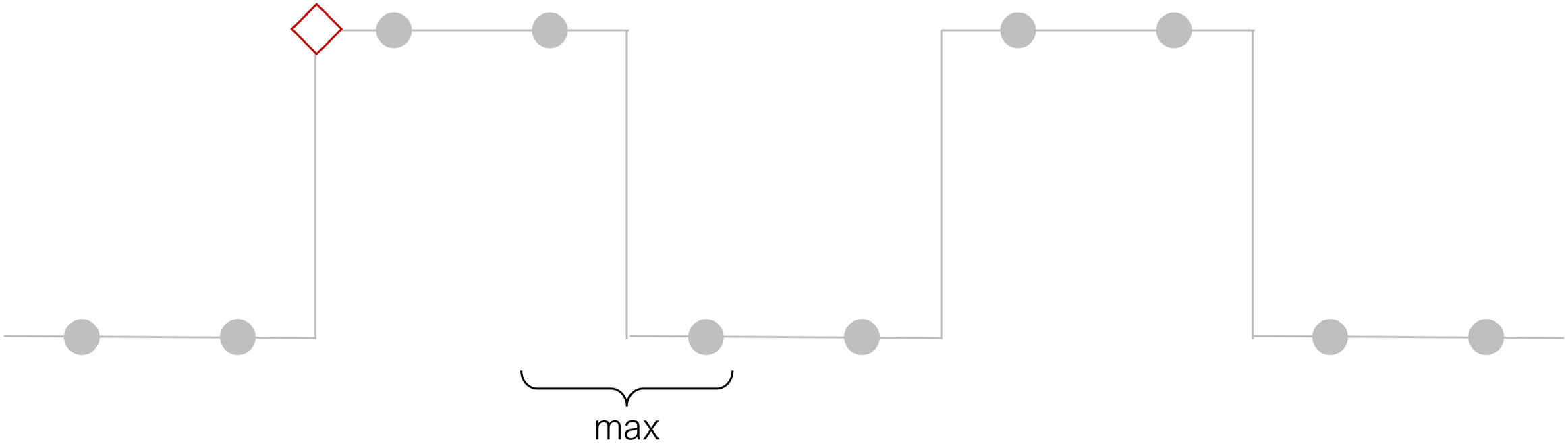
Re-examining Max-Pooling



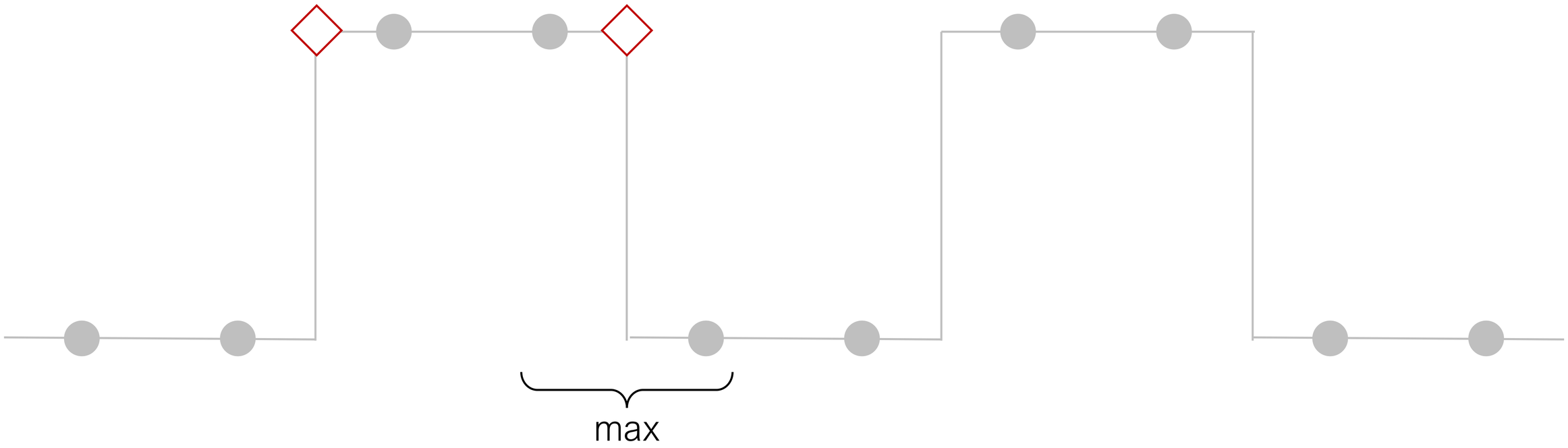
Re-examining Max-Pooling



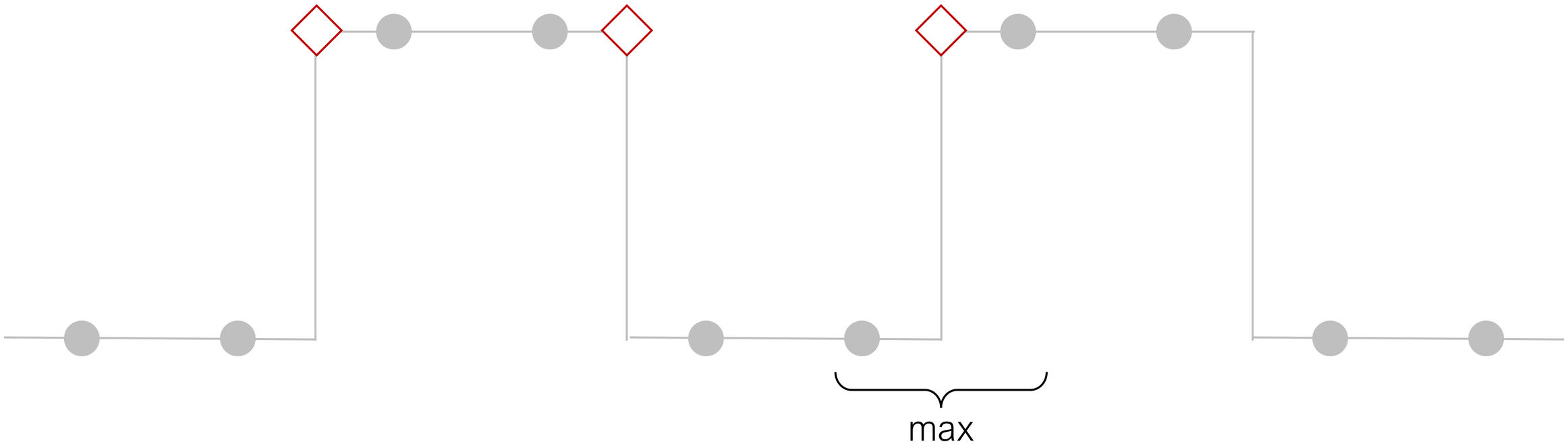
Re-examining Max-Pooling



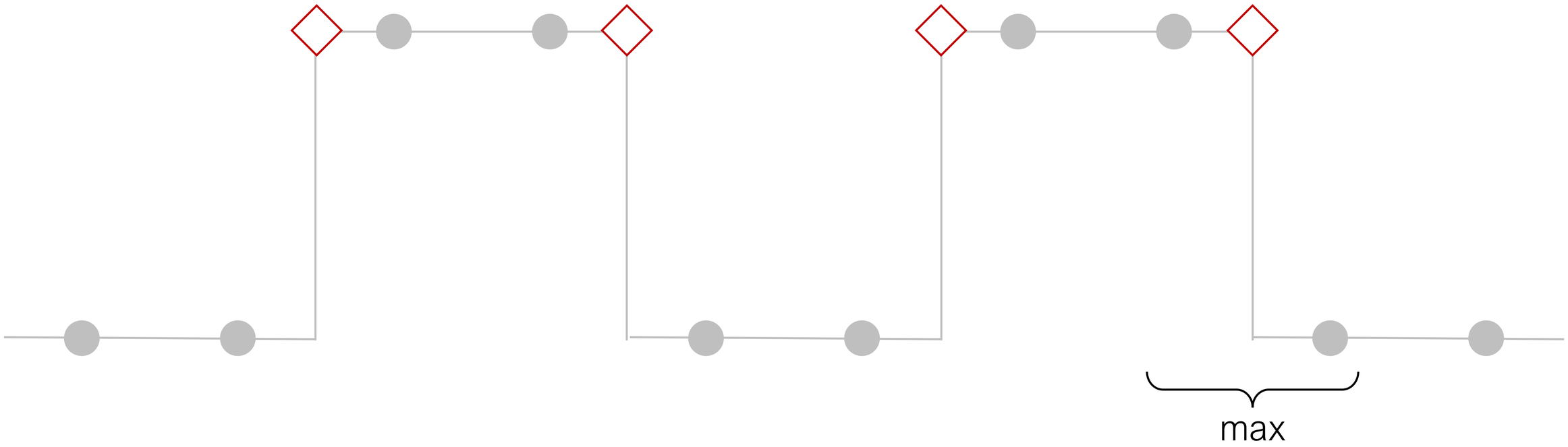
Re-examining Max-Pooling



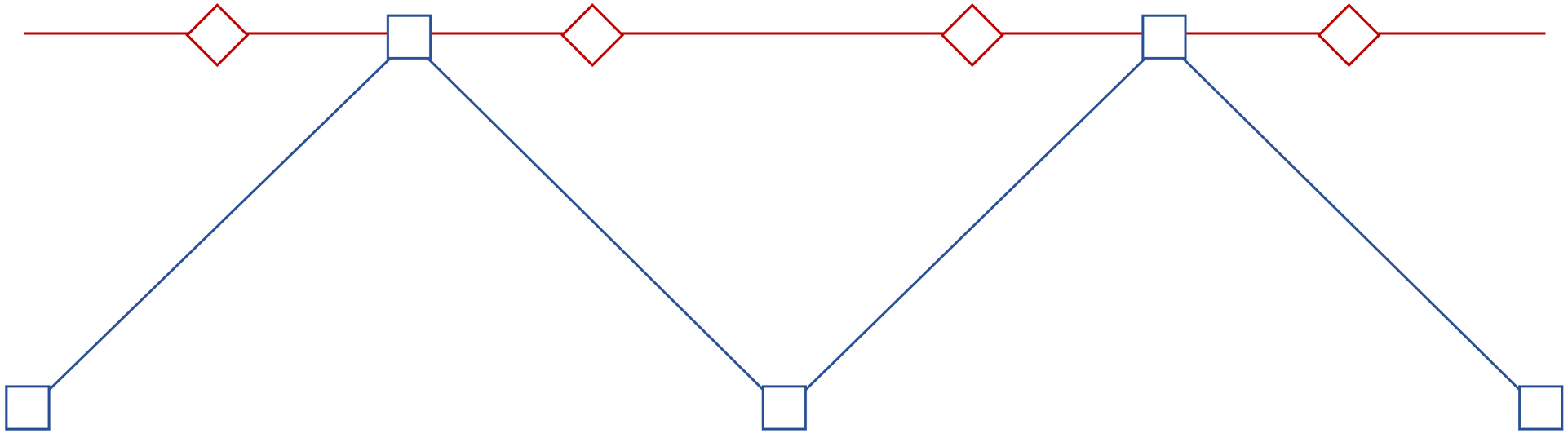
Re-examining Max-Pooling



Re-examining Max-Pooling



Re-examining Max-Pooling



Max-pooling **breaks** shift-equivariance

Aliasing → Loss of shift-equivariance

Shift to the right

Naive Prefiltered

from Jaakko Lehtinen https://twitter.com/jaakkolehtinen/status/1258102168176951299

Example CNN classifications

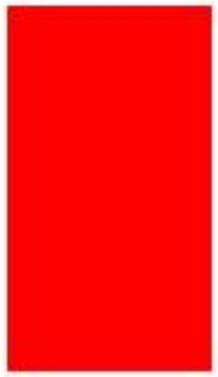
86.7

P(correct class)

69.2

P(correct class)

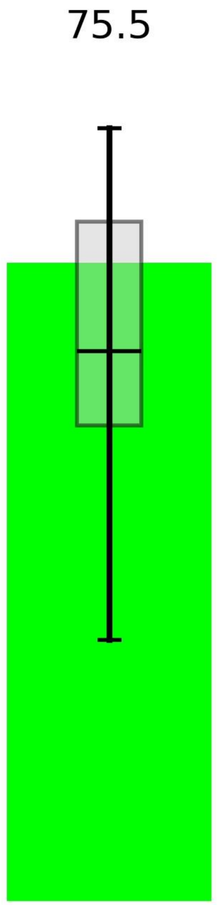
Deep Networks are **not** Shift-Invariant

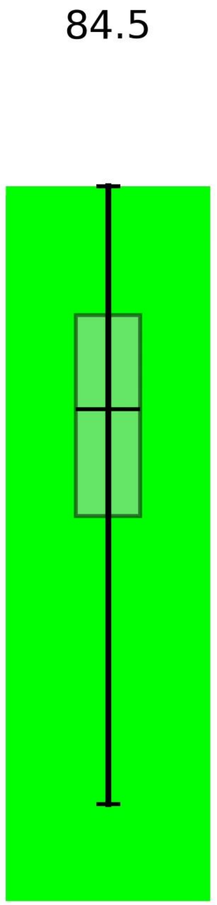


P(correct class)

P(correct class)

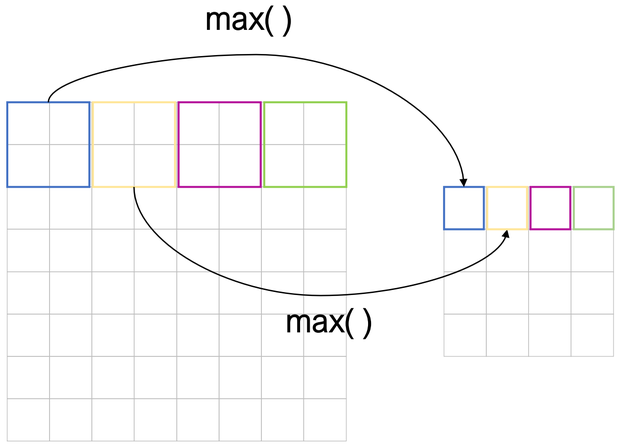
Deep Networks are **not** Shift-Invariant





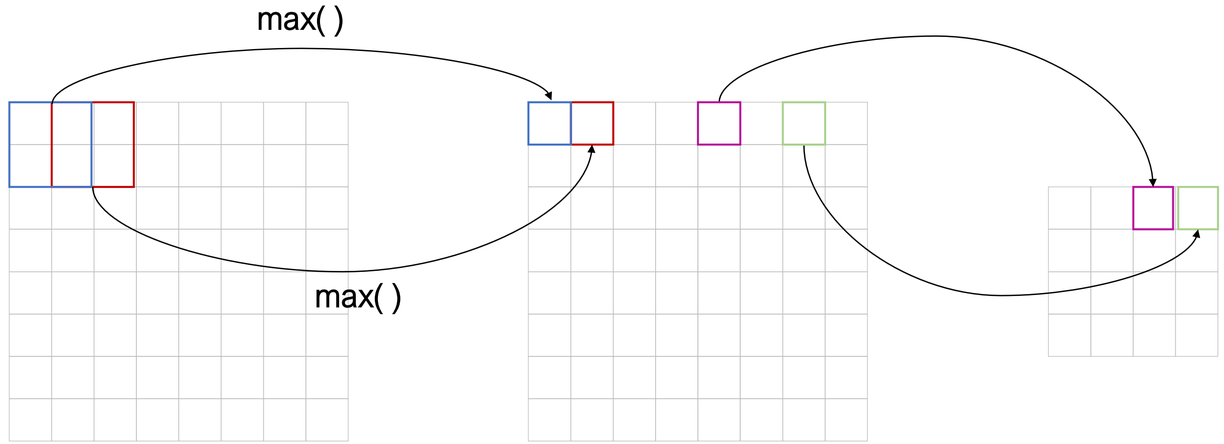
c.f. Azulay and Weiss. **Why do deep convolutional networks generalize so poorly to small image transformations?** Arxiv, 2018. JMLR, 2019.

Engstrom, Tsipras, Schmidt, Madry. **Exploring the Landscape of Spatial Robustness.** Arxiv, 2017. ICML, 2019.



Shift-equivariance lost; heavy aliasing

Strided-MaxPool



Max (densely)

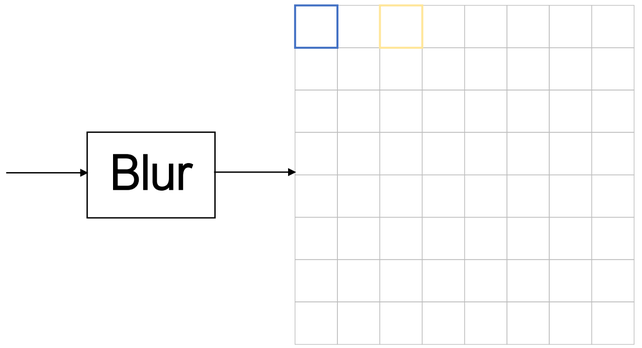
Preserves shift-equivariance

+

Subsampling

Shift-eq. lost; heavy aliasing

Equivalent Interpretation



+

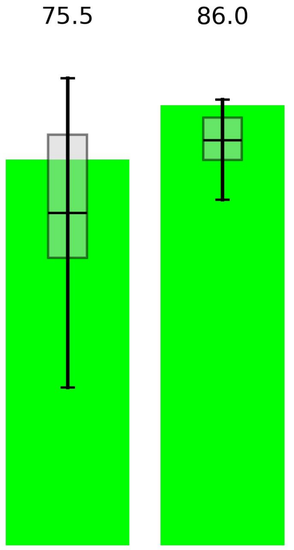
Blur

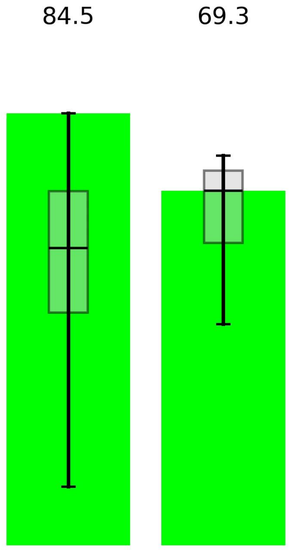
Preserves shift-eq.

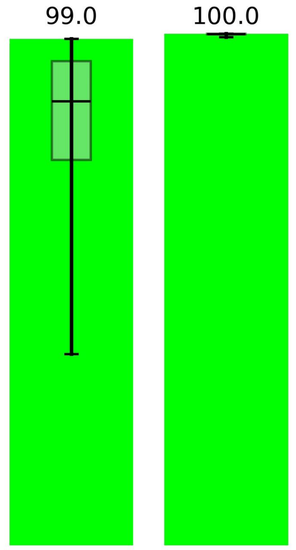
Shift eq. lost, but reduced aliasing

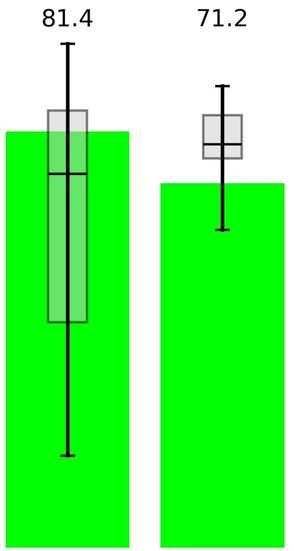
Anti-aliased pooling (MaxBlurPool)

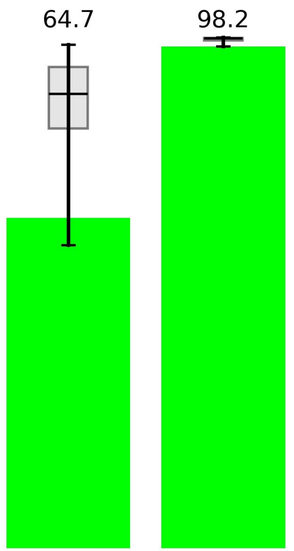
Qualitative examples

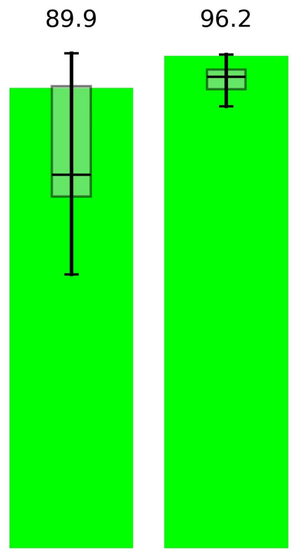




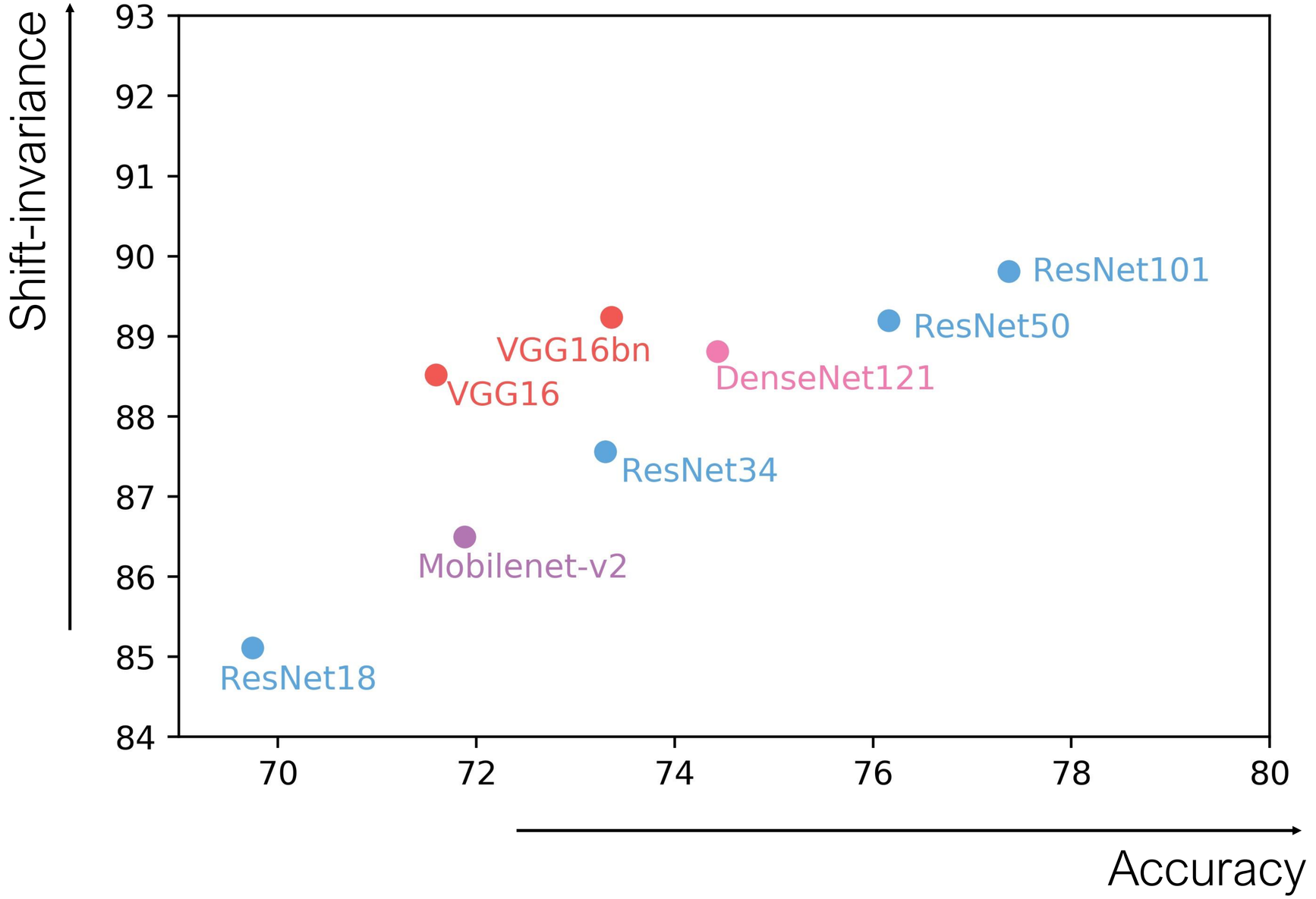




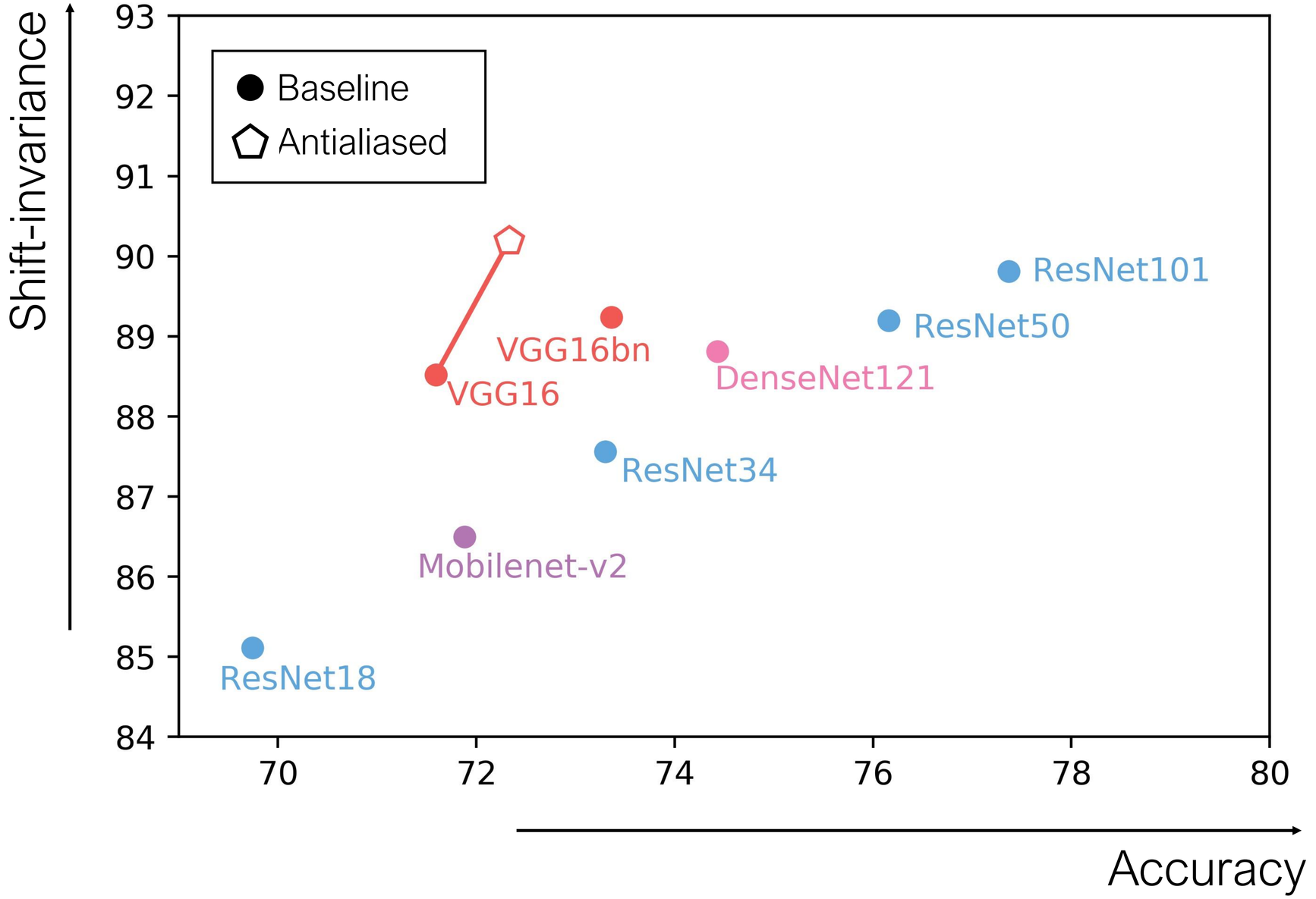




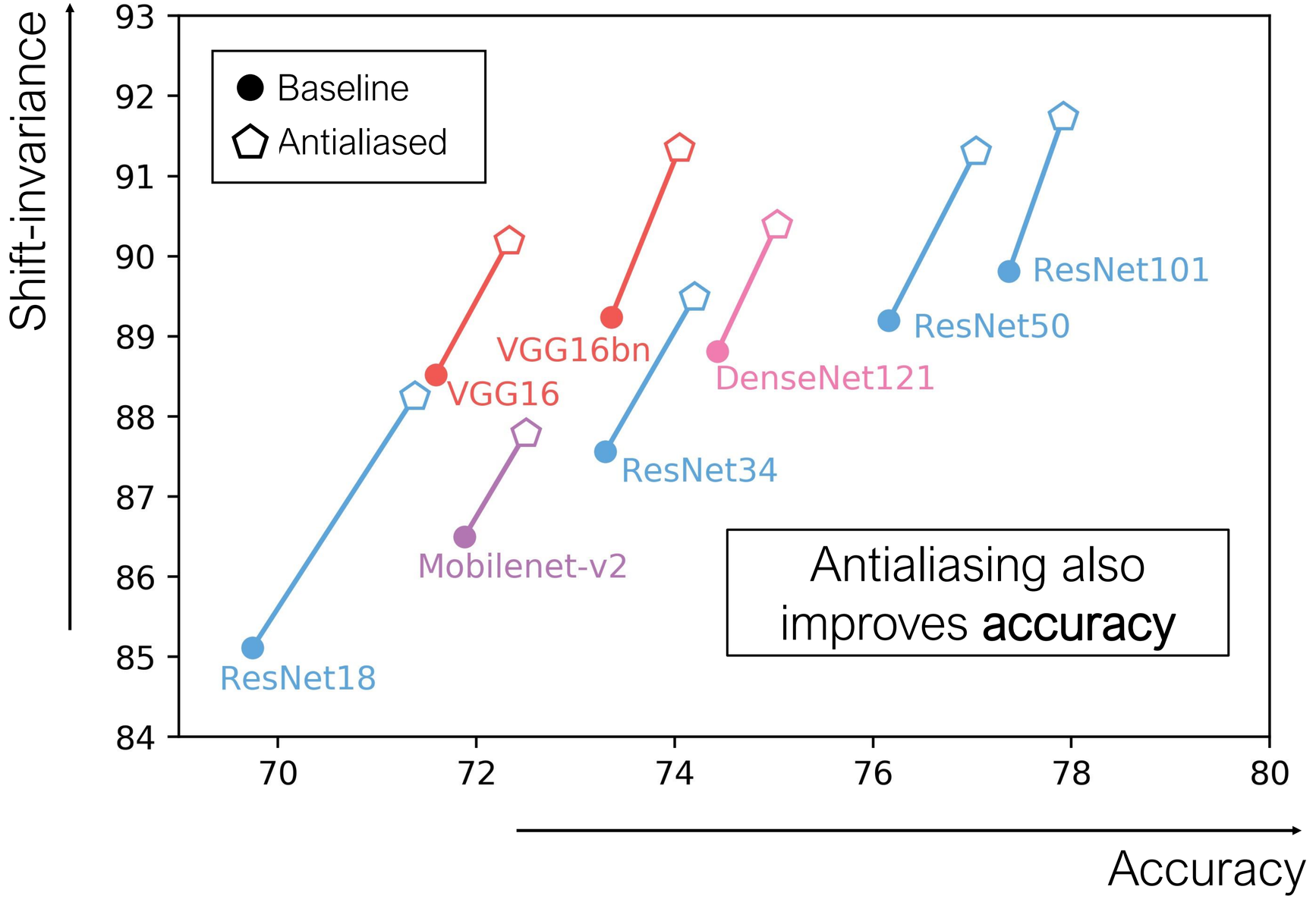
ImageNet


Shift-invariance
93
92
91
90
89
88
87
86
85
84
70
72
74
76
78
80
Accuracy
ResNet101
ResNet50
VGG16bn
DenseNet121
VGG16
ResNet34
Mobilenet-v2
ResNet18

ImageNet


Shift-invariance
93
92
91
90
89
88
87
86
85
84
Baseline
Antialiased
ResNet101
ResNet50
VGG16bn
DenseNet121
VGG16
ResNet34
Mobilenet-v2
ResNet18
70
72
74
76
78
80
Accuracy

ImageNet


Shift-invariance
93
92
91
90
89
88
87
86
85
84
70
72
74
76
78
80
Accuracy
Baseline
Antialiased
ResNet18
VGG16
VGG16bn
Mobilenet-v2
ResNet34
DenseNet121
ResNet50
ResNet101
Antialiasing also improves **accuracy**

Don’t be a doofus – anti-alias your CNNs!

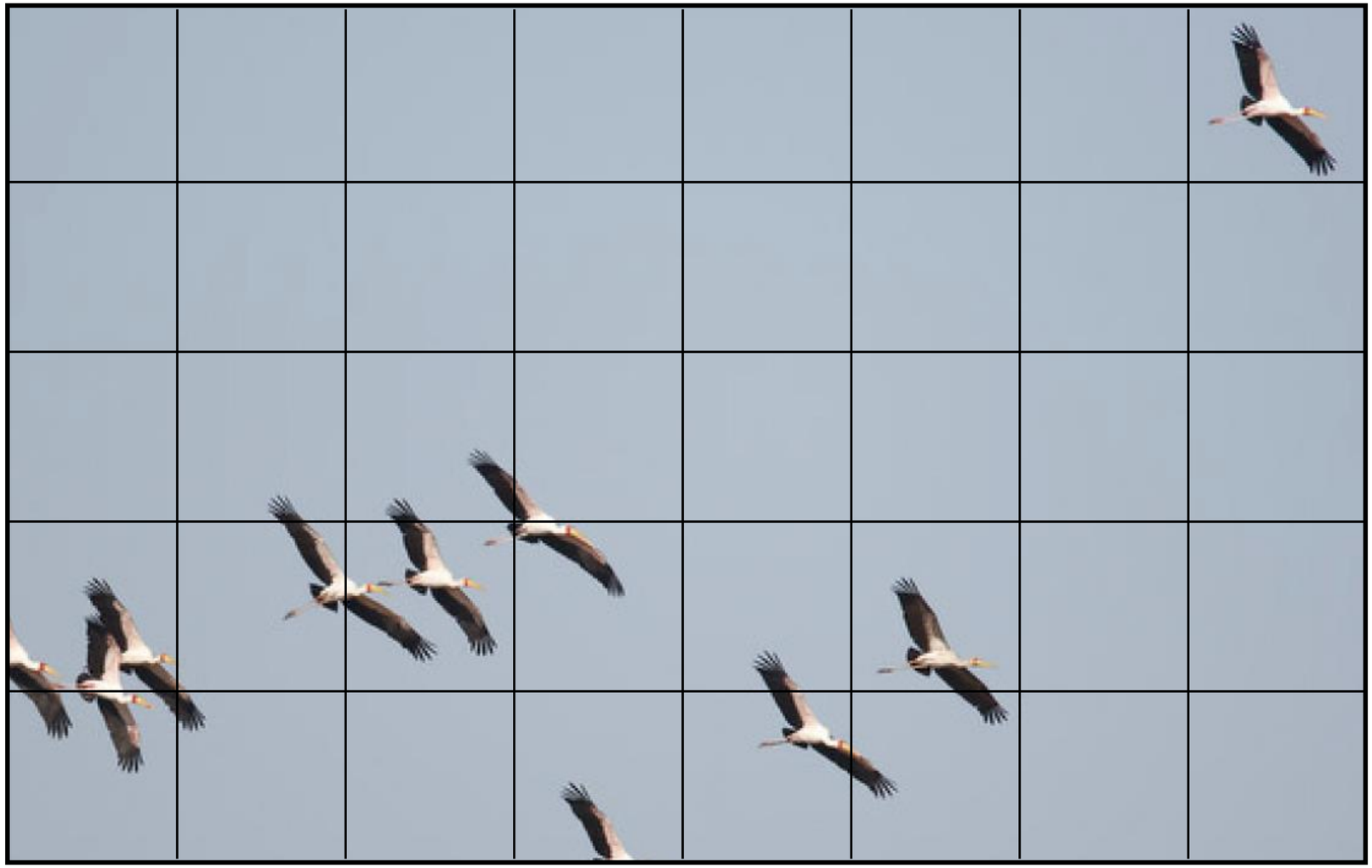
R. Zhang.
Making Convolutional Networks Shift-Invariant Again.
In ICML, 2019.
(hosted on ArXiv)

Code + models: richzhang.github.io/antialiased-cnns/

Run `pip install antialiased-cnns`

```
import antialiased_cnns
model = antialiased_cnns.resnet50(pretrained=True)
```

Receptive fields



Receptive fields

Pool and downsample by 2

3x1 Filter

Pool and downsample by 2

Sky

Sky

Bird

Sky

RF = RF*2

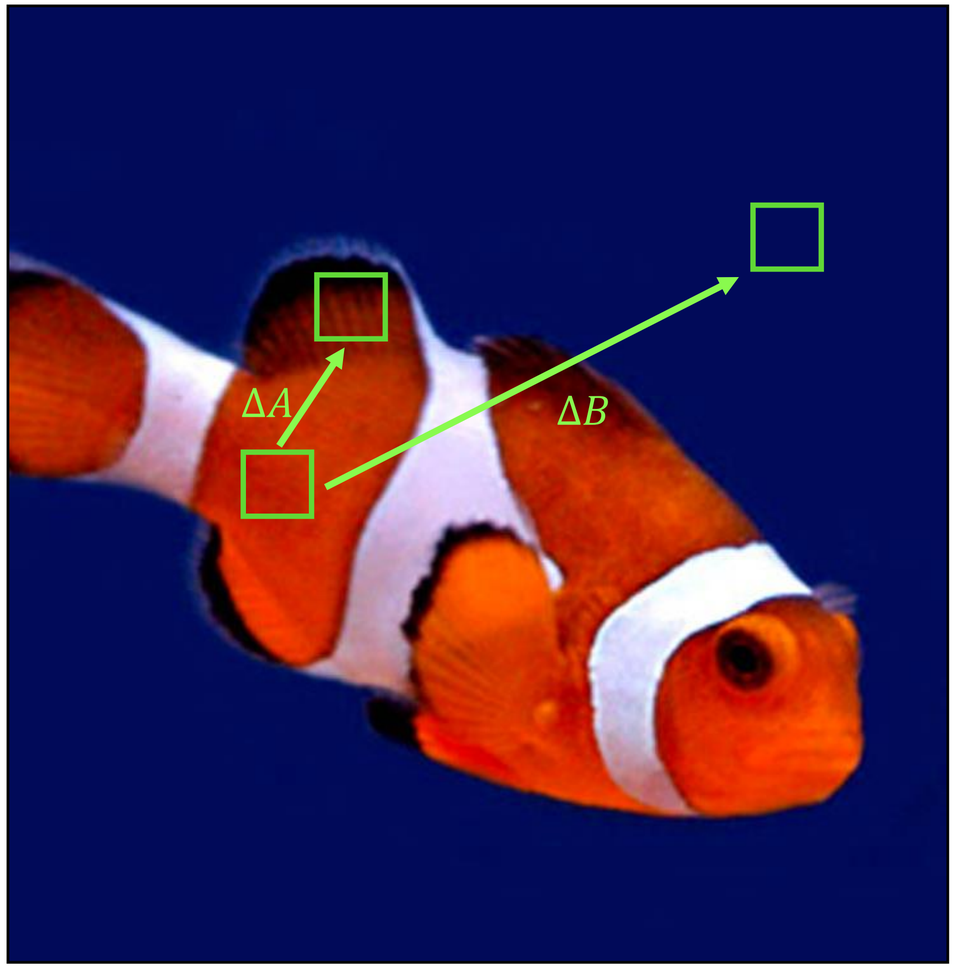
RF = RF + floor(3/2)*2

RF = RF*2

kernel size

downsample factor

Local vs. global processing



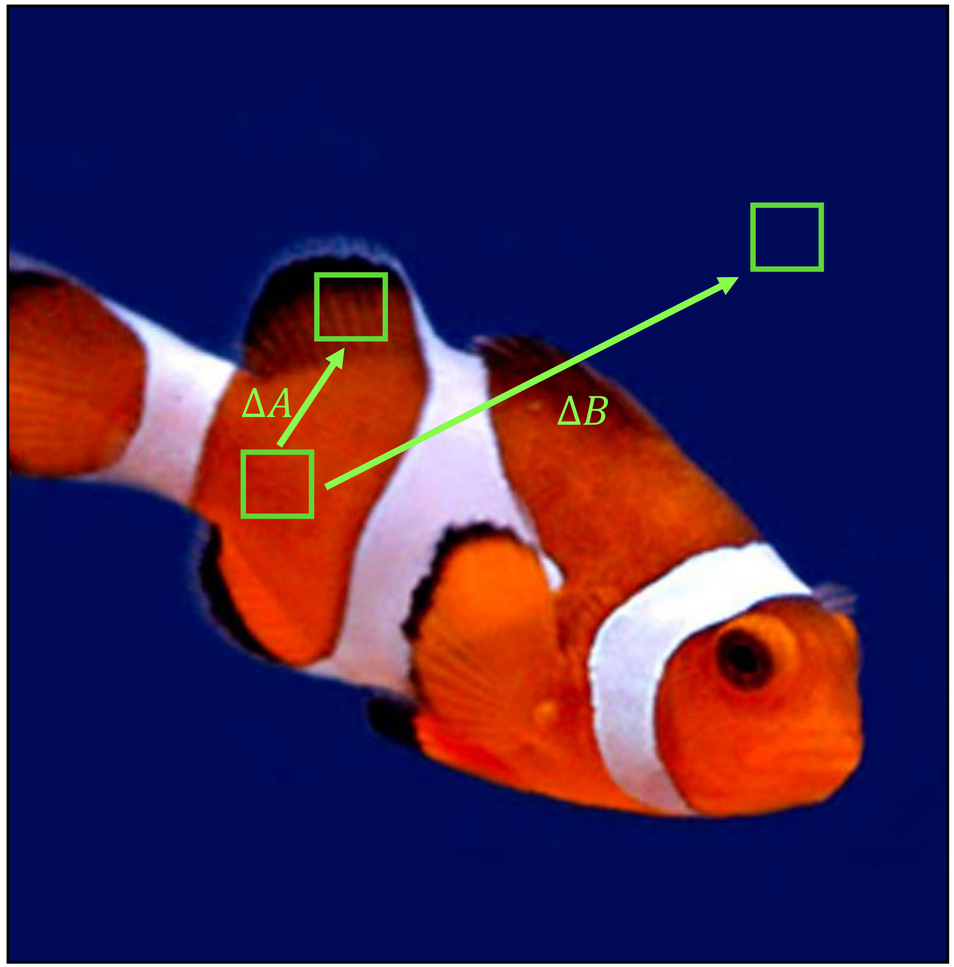
Across all images, which is higher:

(1) correlation between points with distance ΔA

(2) correlation between points with distance ΔB

(3) can't say

Local vs. global processing



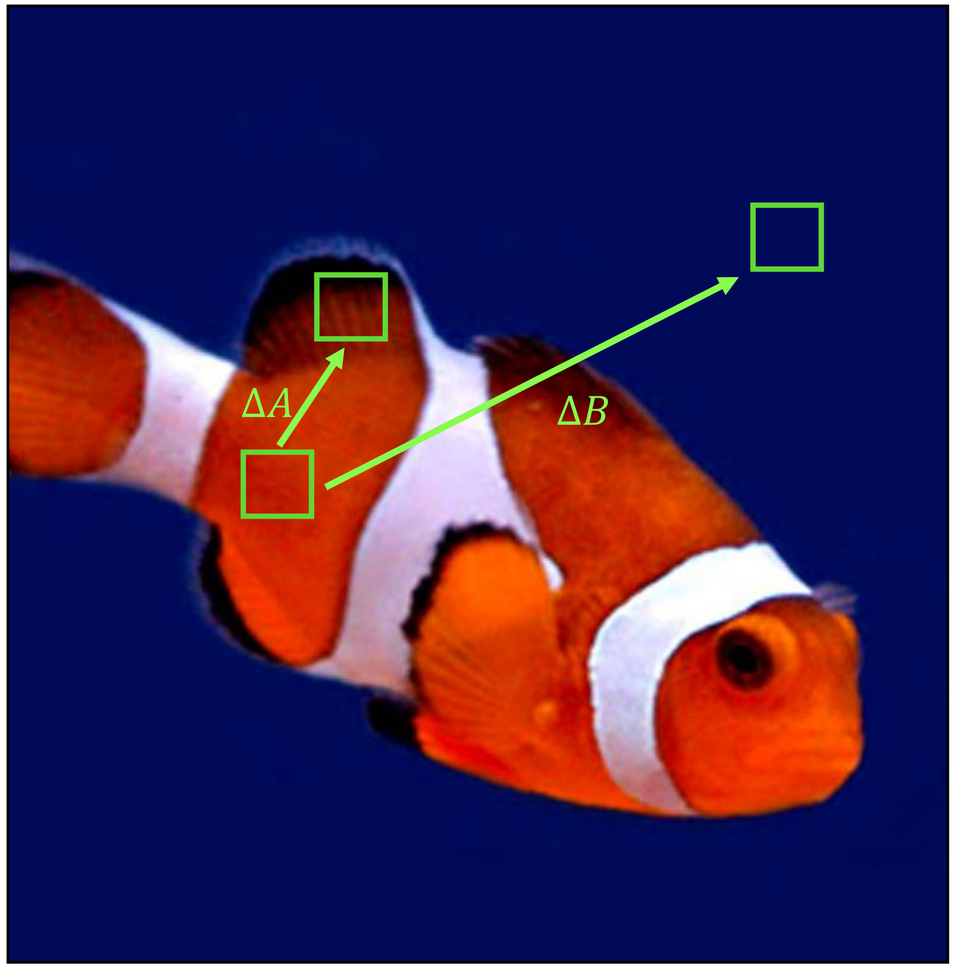
Across all images, which is higher:

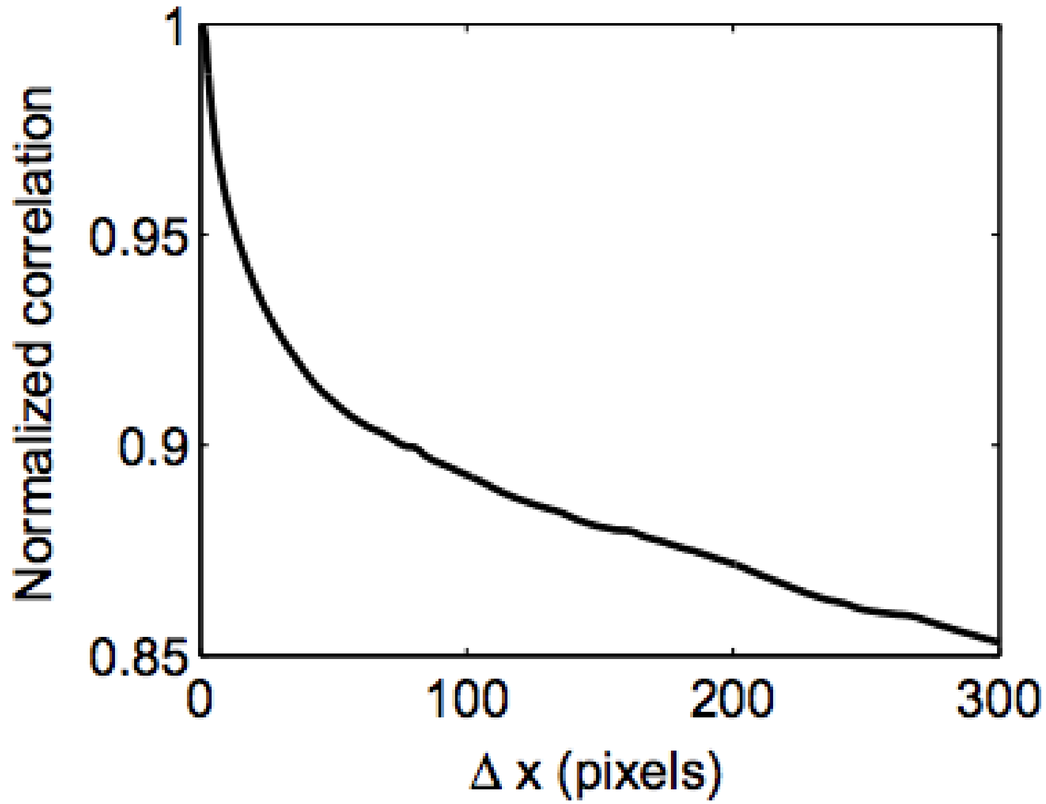
(1) correlation between points with distance ΔA

(2) correlation between points with distance ΔB

(3) can’t say

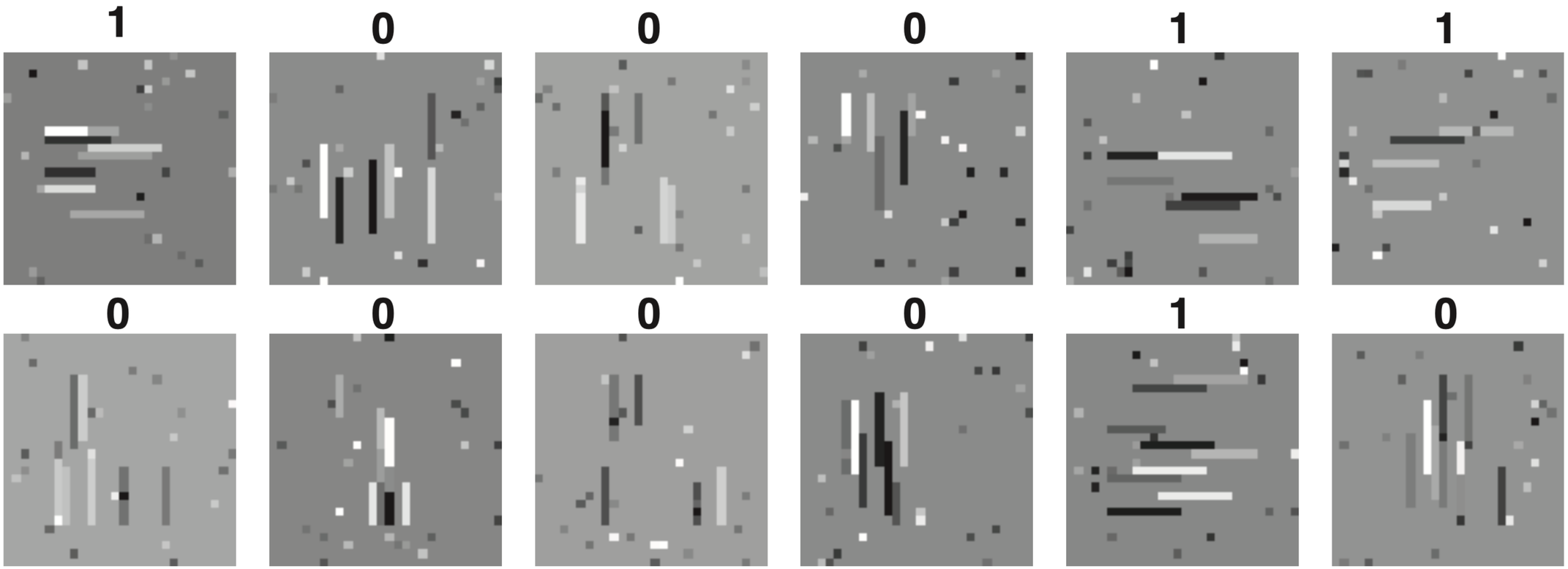
Local vs. global processing





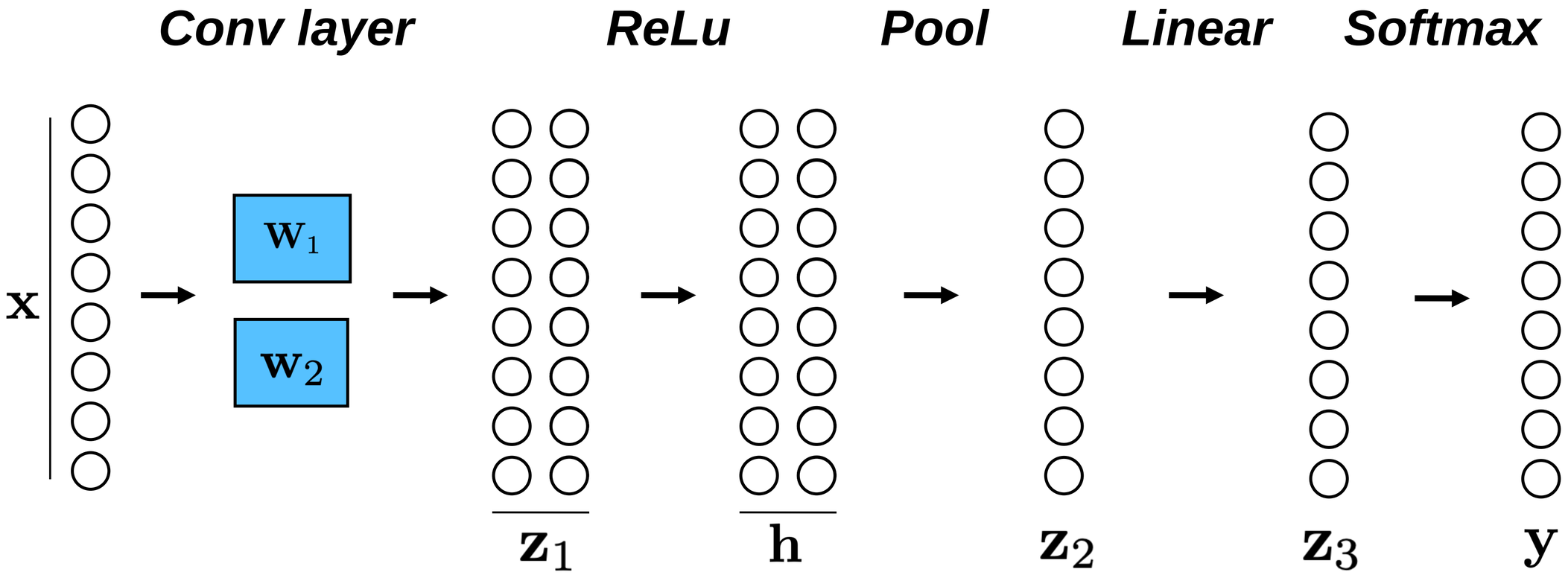
[Simoncelli: Statistical Modelling of Photographic Images, 2005]

Example: simple line classification



Class 0: vertical lines
Class 1: horizontal lines

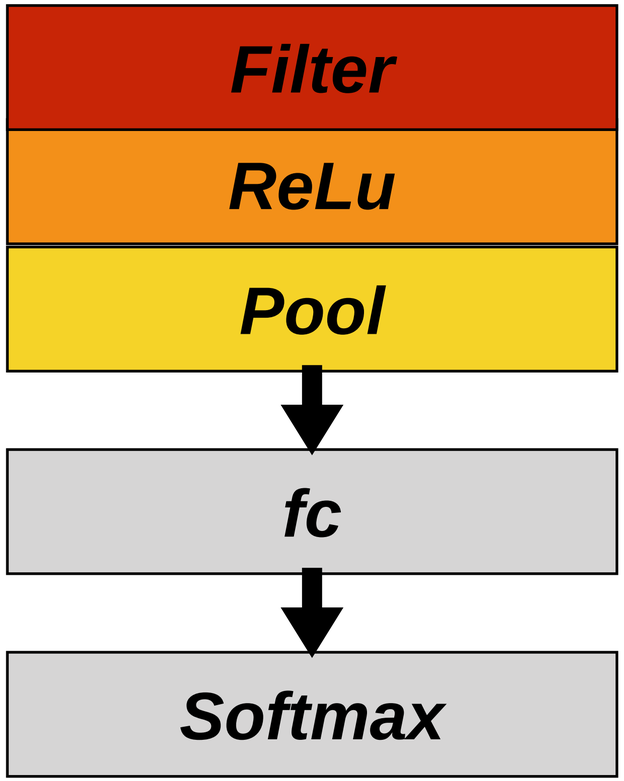
Example: simple line classification



$$\mathbf{z}_{1_i} = \mathbf{w}_i \circ \mathbf{x} + \mathbf{b}_i$$

$$\mathbf{h}_i = \max(\mathbf{z}_{1_i}, 0)$$

Example: simple line classification



$$\mathbf{z}_{1_i} = \mathbf{w}_i \circ \mathbf{x} + \mathbf{b}_i$$

with 2 learned kernels $\mathbf{w}_1, \mathbf{w}_2$

$$\mathbf{h}_i = \max(\mathbf{z}_{1_i}, 0)$$

$$\mathbf{z}_{2_i} = \frac{1}{NM} \sum_{n,m} \mathbf{h}_i[n, m]$$

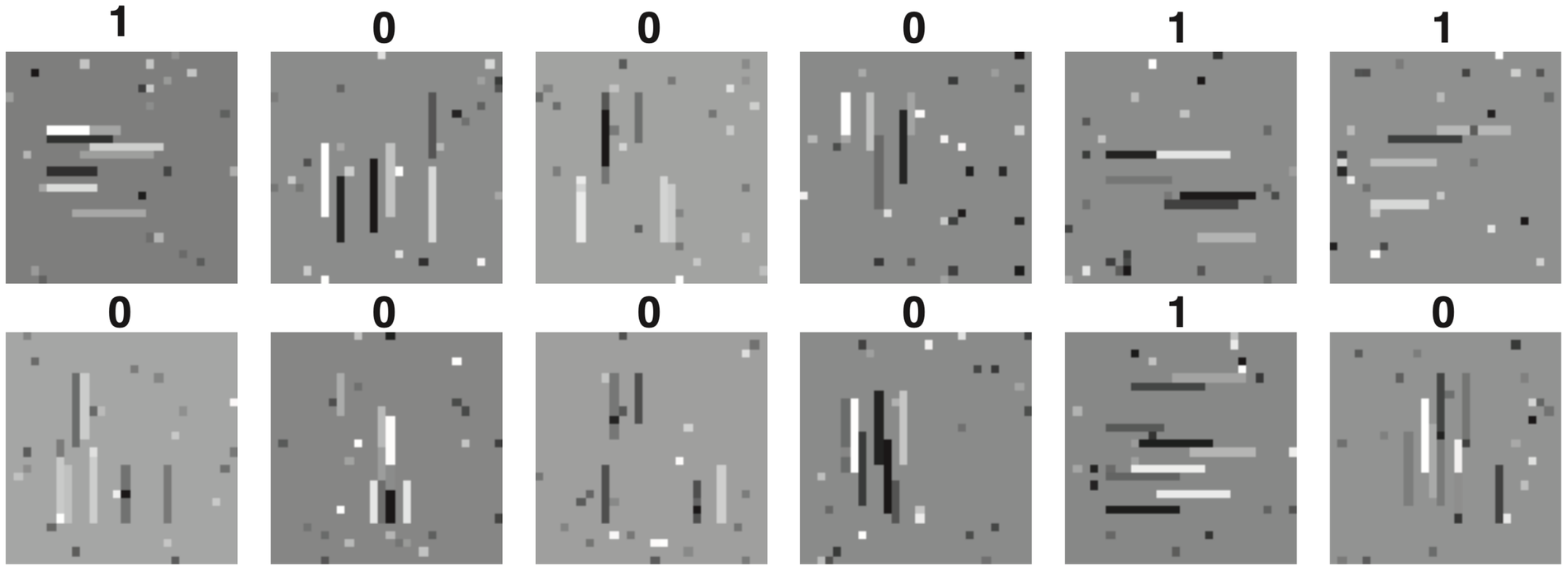
$$\mathbf{z}_3 = \mathbf{W}\mathbf{z}_2 + \mathbf{c}$$

$$\mathbf{y}_i = \frac{e^{-\tau \mathbf{z}_{3_i}}}{\sum_{k=1}^{K} e^{-\tau \mathbf{z}_{3_k}}}$$

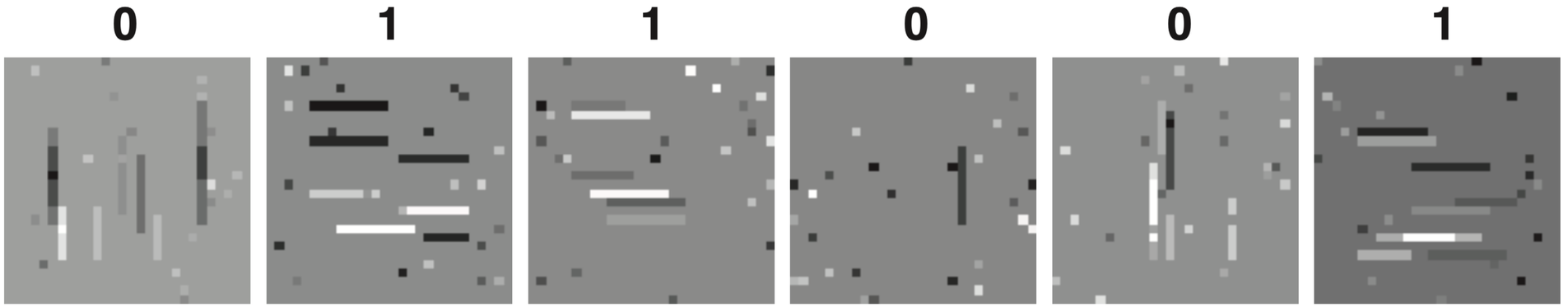
parameters $\mathbf{w}_1, \mathbf{w}_2, \mathbf{b}_1, \mathbf{b}_2, \mathbf{W}, \mathbf{c}$

Network training and evaluation

Training

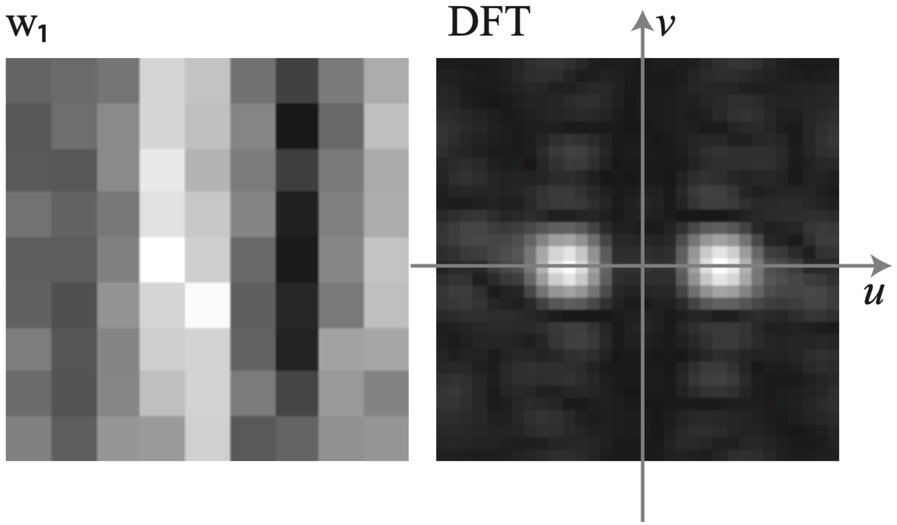


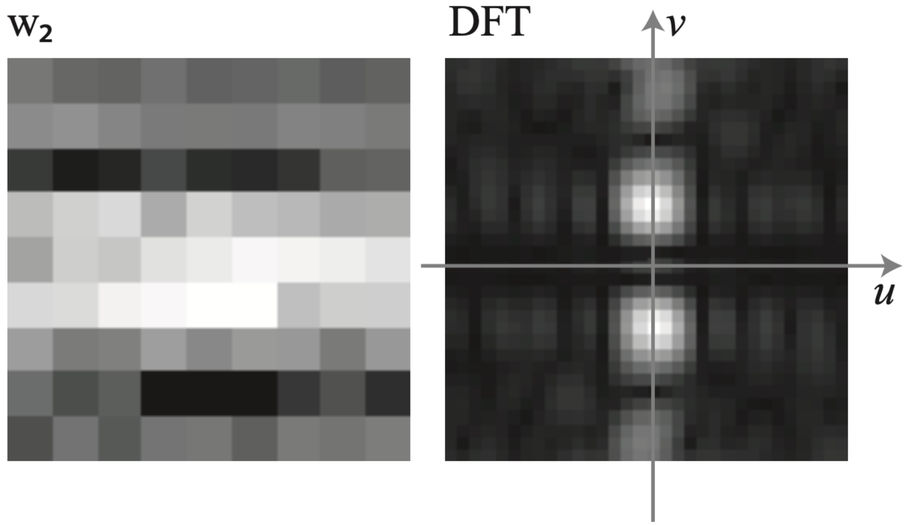
Testing
3/10000 misclassified



Network visualization

9x9 learned kernels:



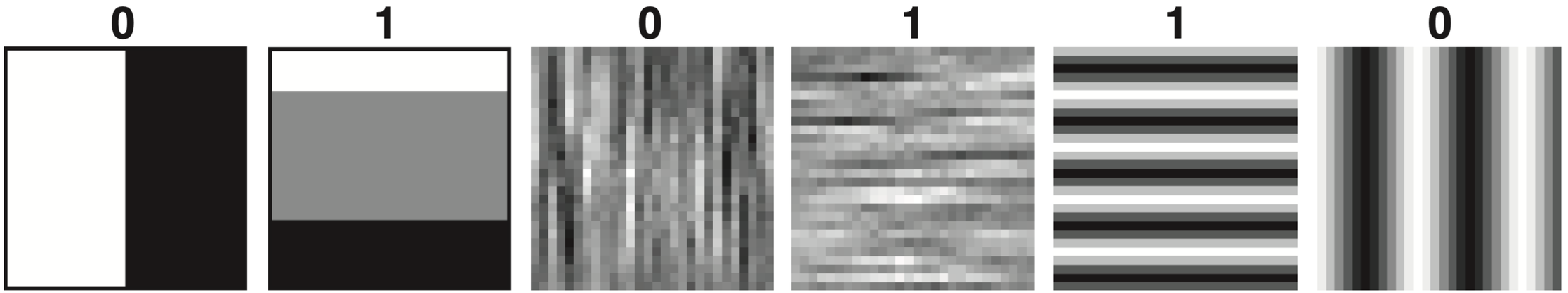


fc layer learned weight:

$$\mathbf{W} = \begin{bmatrix} 2.83 & -2.36 \\ -0.60 & 1.14 \end{bmatrix}$$

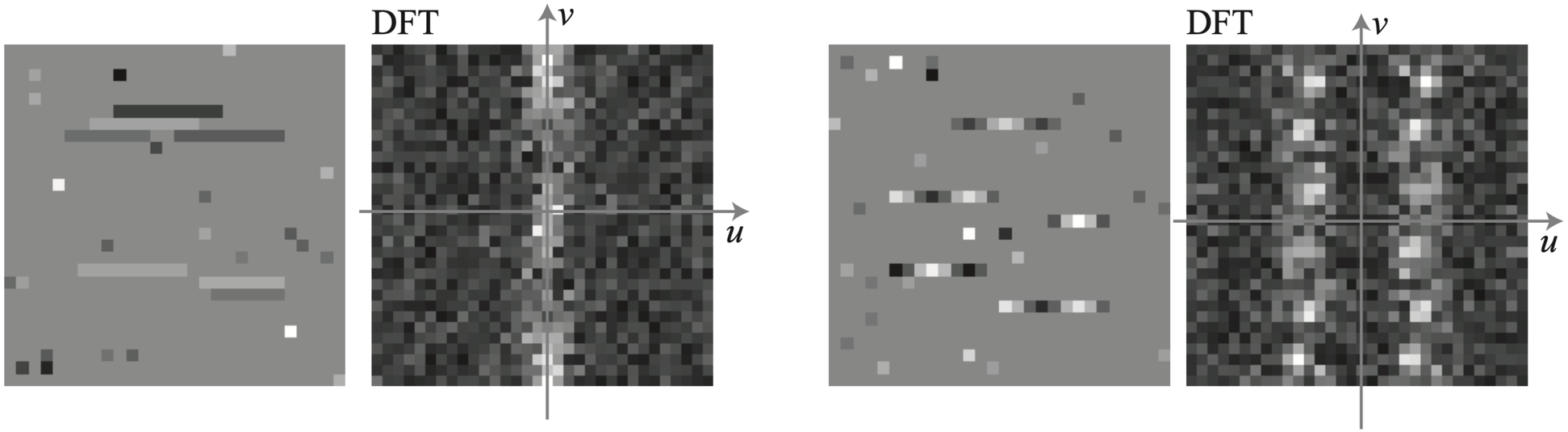
Out of domain generalization

Out of domain test samples are classified correctly

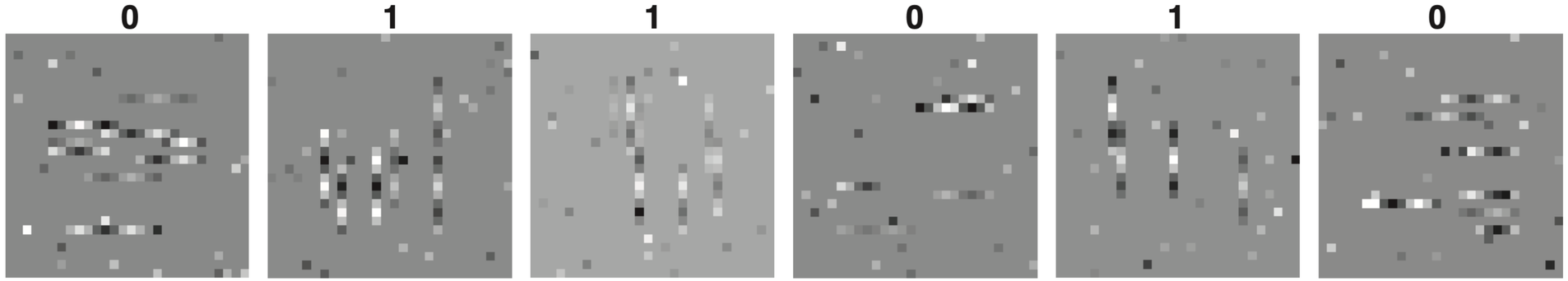


Identifying vulnerability

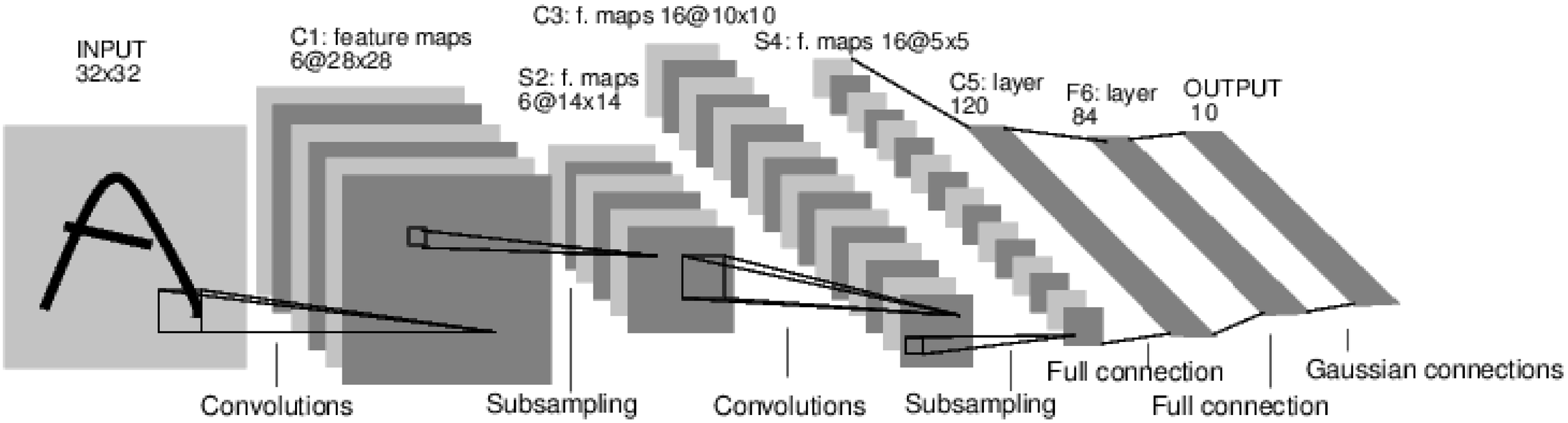
- Modulation: multiplying image with sinusoidal wave moves spectral content horizontally



- All lines with sinusoidal texture are misclassified

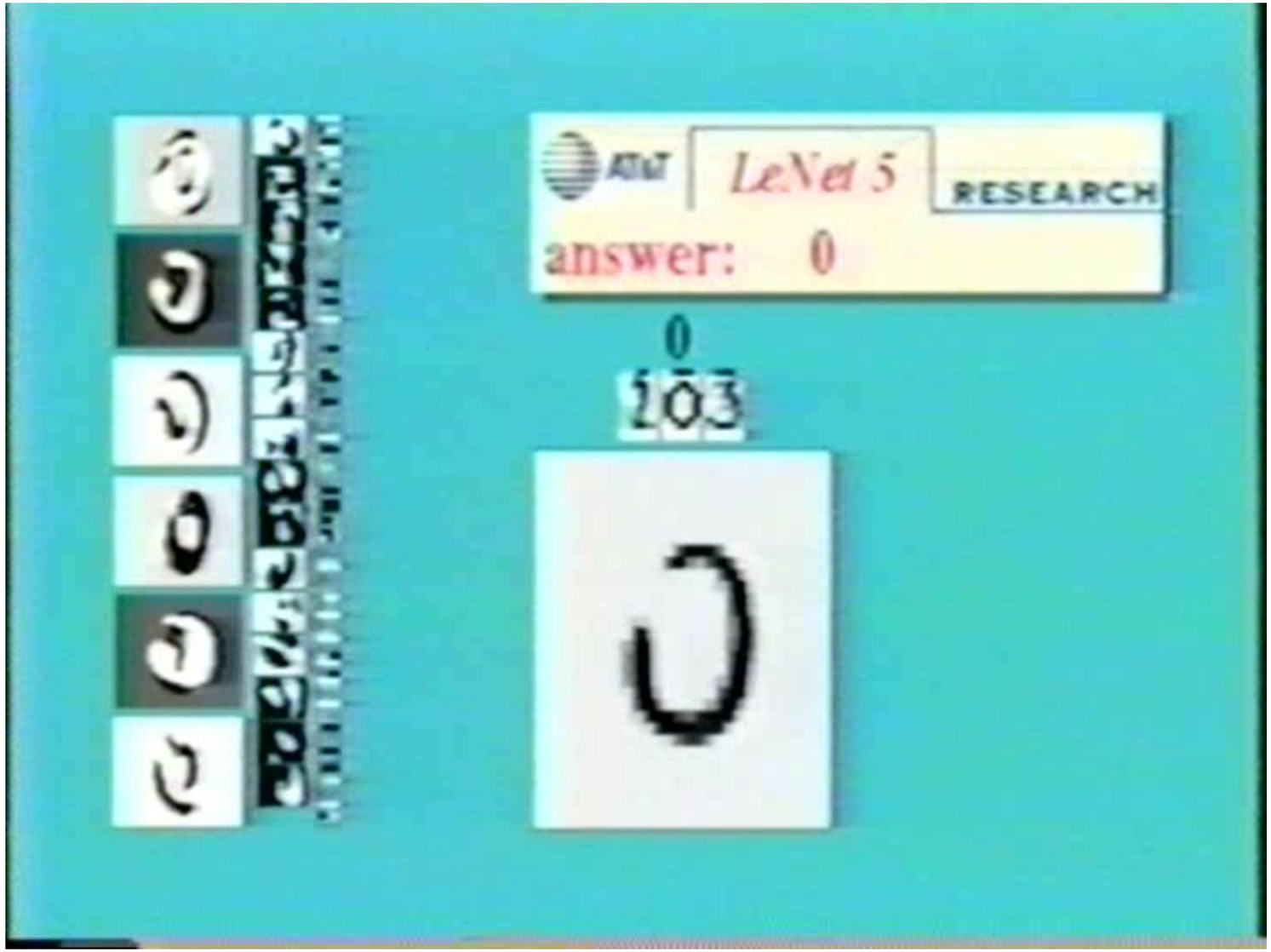


LeNet (LeCun et al., 1998)



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

LeNet in action!



[Courtesy of Yann LeCun]

Looking At Landmark Networks

- We’ll look at 3 landmark networks, each trained to solve a 1000-way classification output (ImageNet)
  - Alexnet (2012)
  - VGG-16 (2014)
  - Resnet (2015)

Dataset – ILSVRC

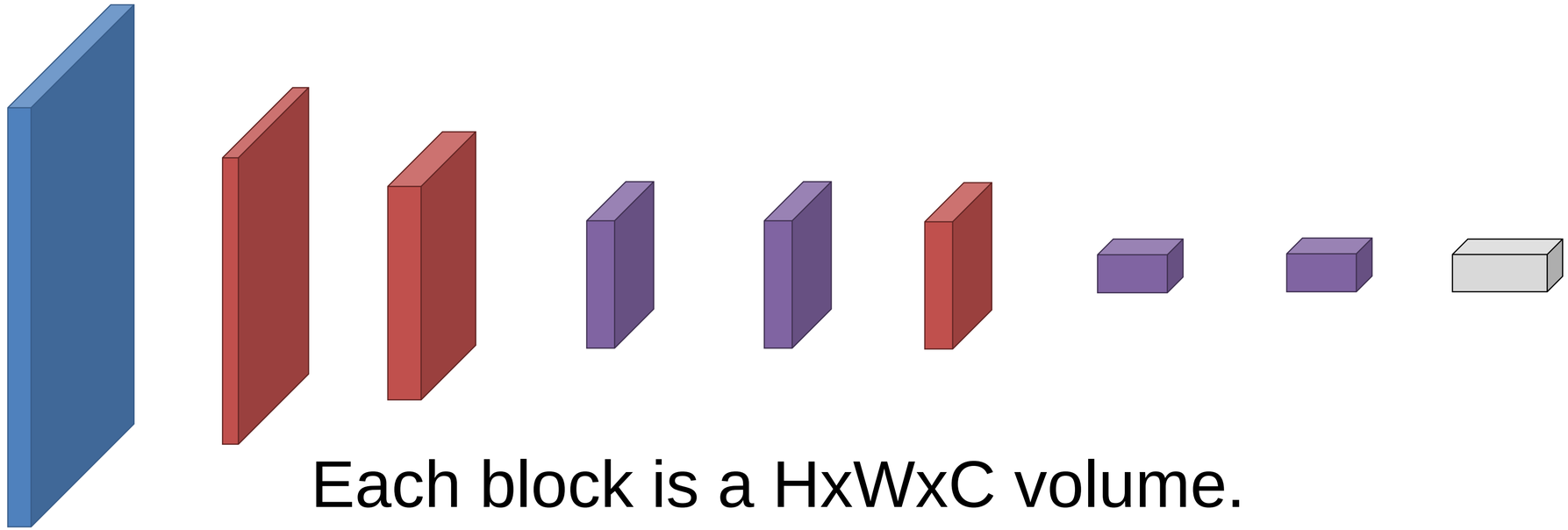
- ImageNet Largescale Visual Recognition Challenge
- 1.4M images
- 1000 Categories, often ridiculously precise

Dataset – ILSVRC

Figure Credit: O. Russakovsky

AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x2273	55x5596	27x27256	13x13384	13x13384	13x13256	1x14096	1x14096	1x11000

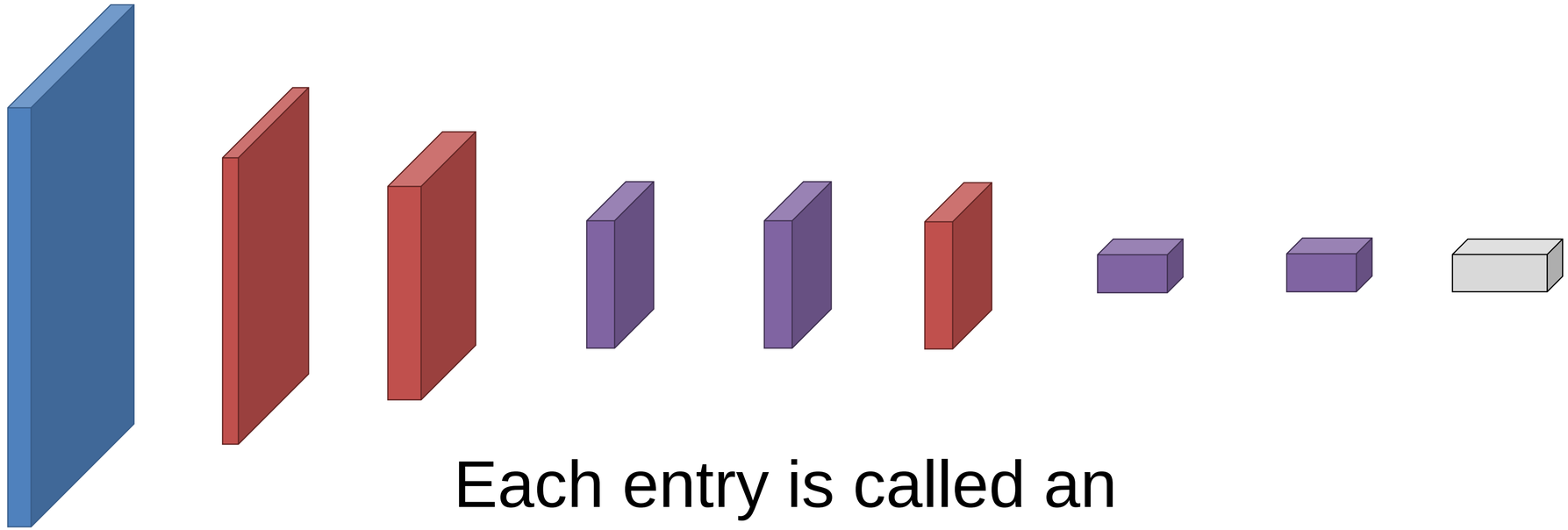


Each block is a HxWxC volume.

You transform one volume to another with convolution

CNN Terminology

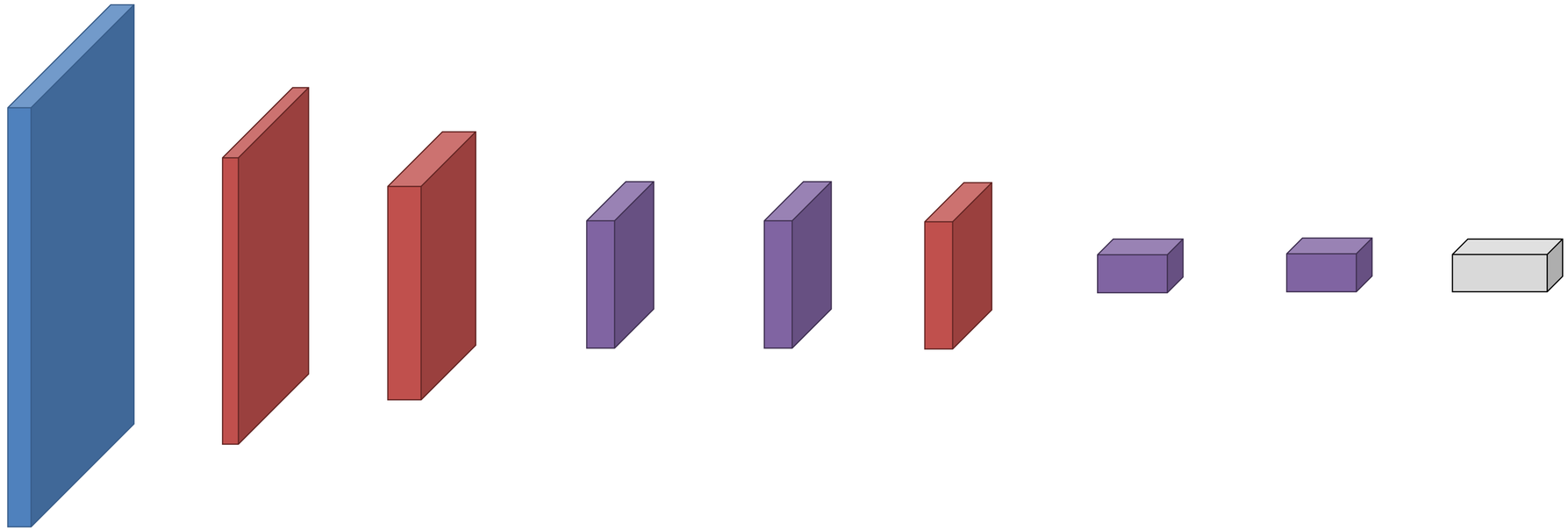
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x2273	55x5596	27x27256	13x13384	13x13384	13x13256	1x14096	1x14096	1x11000



Each entry is called an “activation”/“neuron”/“feature”

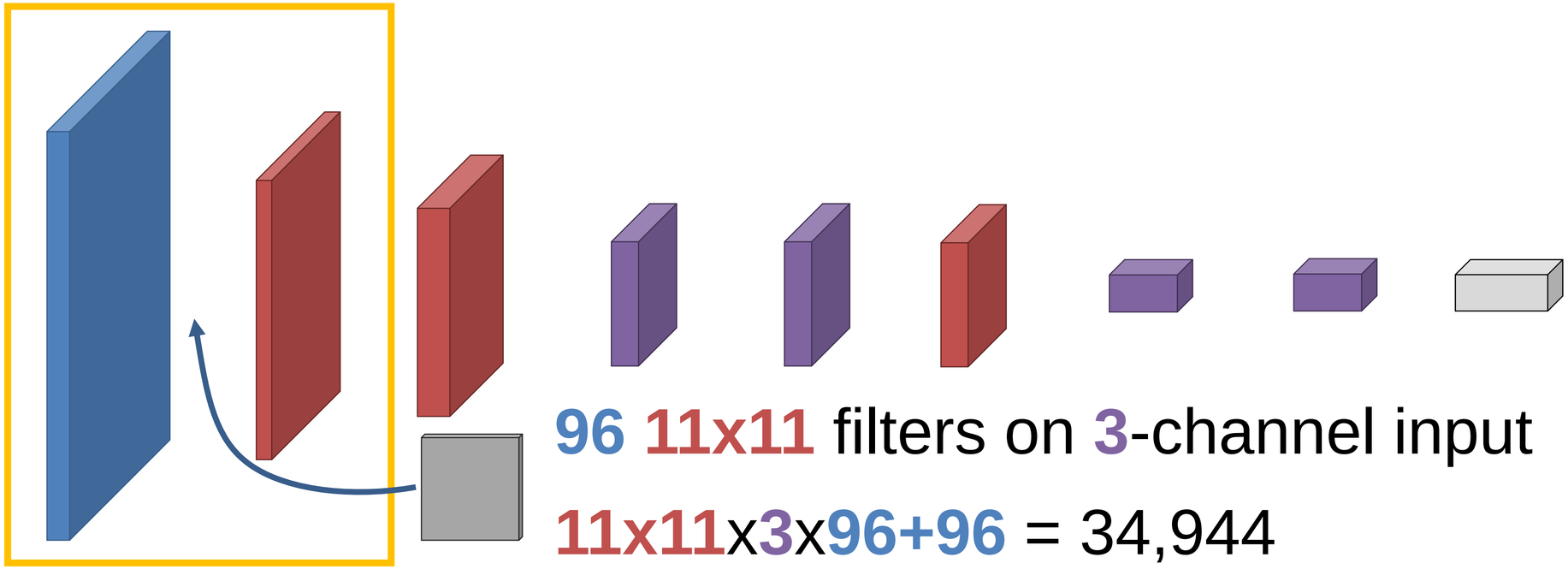
Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x2273	55x5596	27x27256	13x13384	13x13384	13x13256	1x14096	1x14096	1x11000



Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x2273	55x5596	27x27256	13x13384	13x13384	13x13256	1x14096	1x14096	1x11000



96 11x11 filters on **3**-channel input

11x11x**3**x**96+96** = 34,944

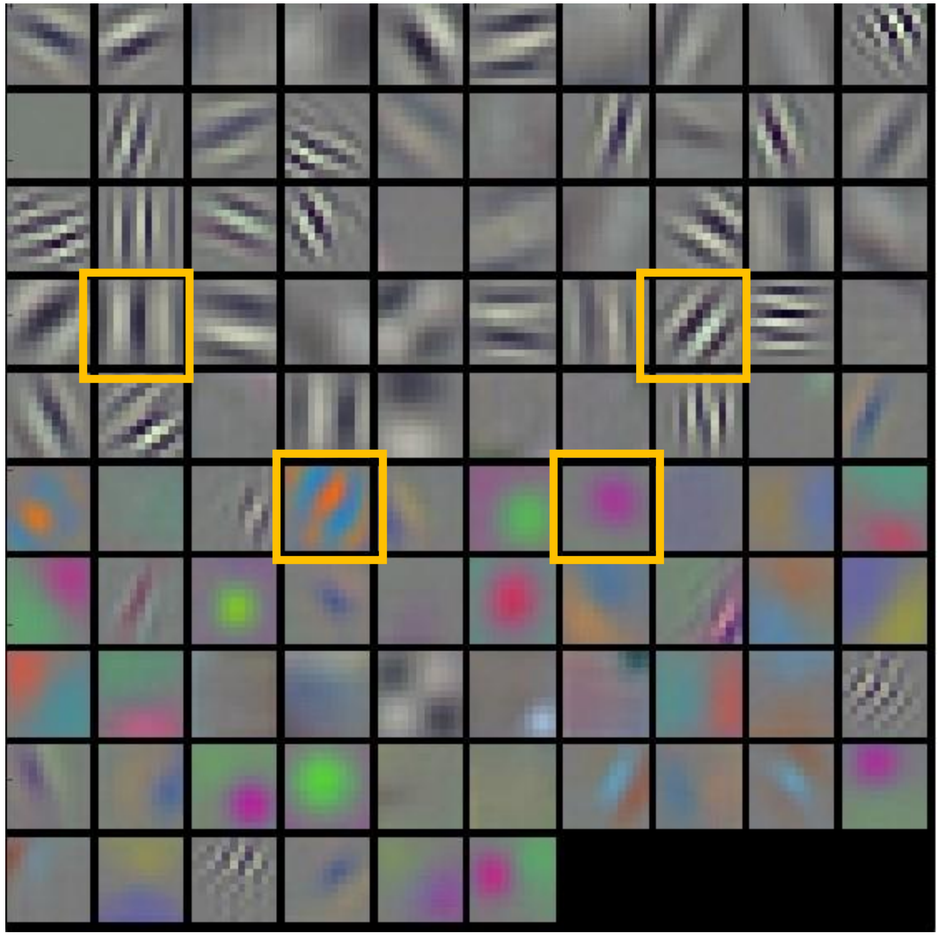
Alexnet – How Many Parameters?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x2273	55x5596	27x27256	13x13384	13x13384	13x13256	1x14096	1x14096	1x11000

4096 **1x1** filters on **4096**-channel input

1x1x**4096**x**4096+4096** = 17 million

What's Learned



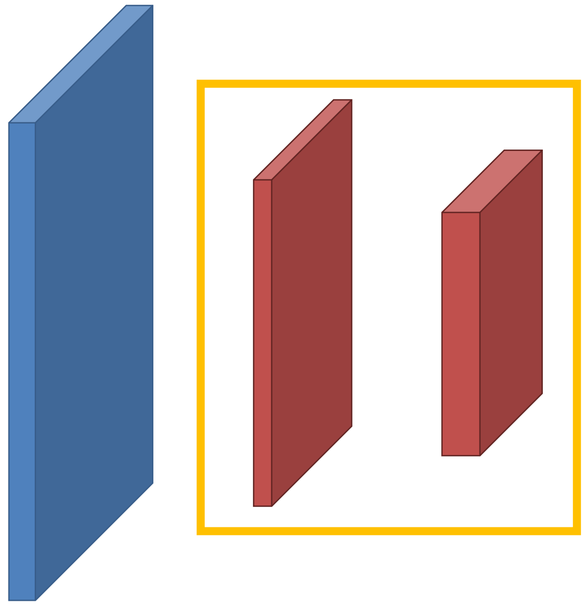
First layer filters of a network trained to distinguish 1000 categories of objects

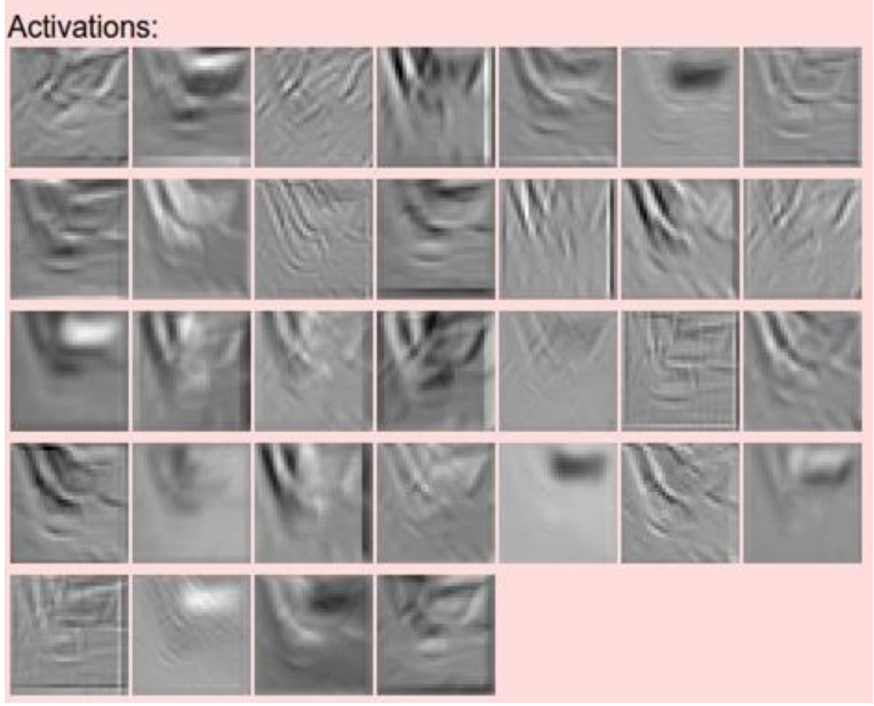
Remember these filters go over color.

Figure Credit: Karpathy and Fei-Fei

Visualizing Later Filters

Input	Conv 1	Conv 2
227x2273	55x5596	27x27256





Conv 2 Filters

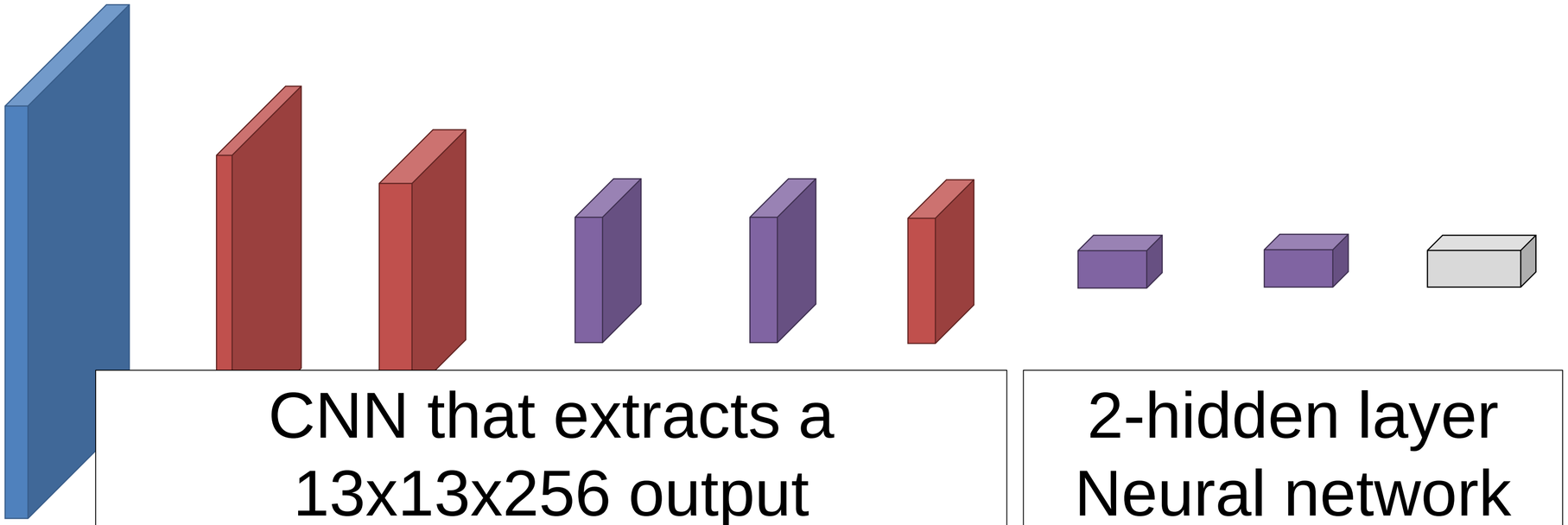
- **Q. How many input dimensions?**
  - A: 96…. hmmm
  - **What does the input mean?**
- Uh, the outputs of filter layer of filters

Visualizing Later Filters

- Understanding the meaning of the later filters *from their values* is typically impossible: too many input dimensions, not even clear what the input means.

Understanding Later Filters

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x559 6	27x272 56	13x133 84	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000

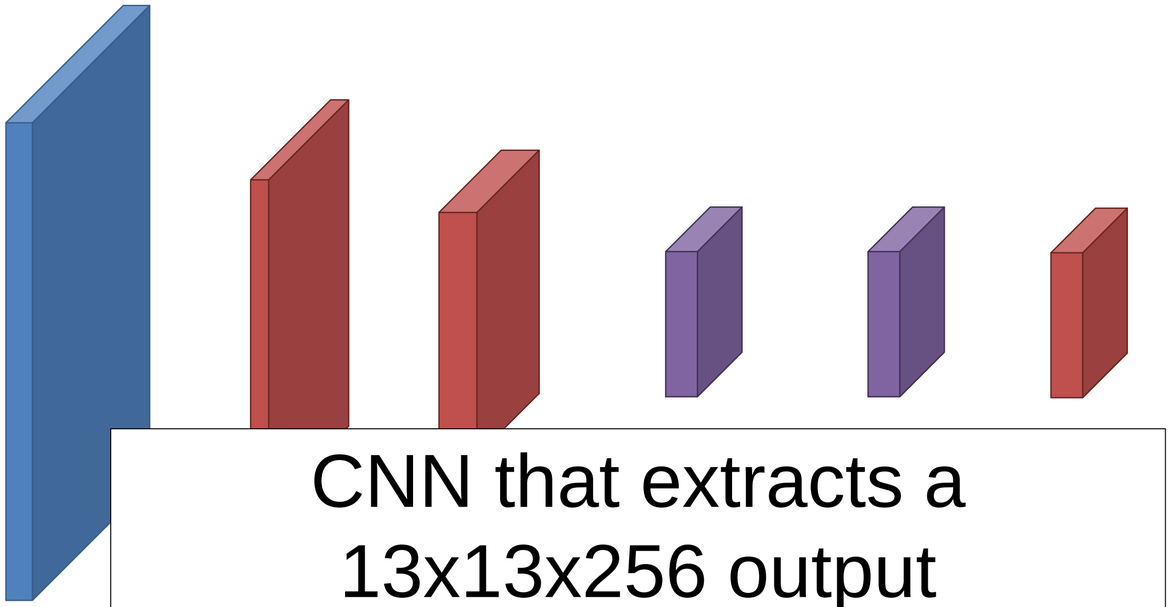


CNN that extracts a 13x13x256 output

2-hidden layer Neural network

Understanding Later Filters

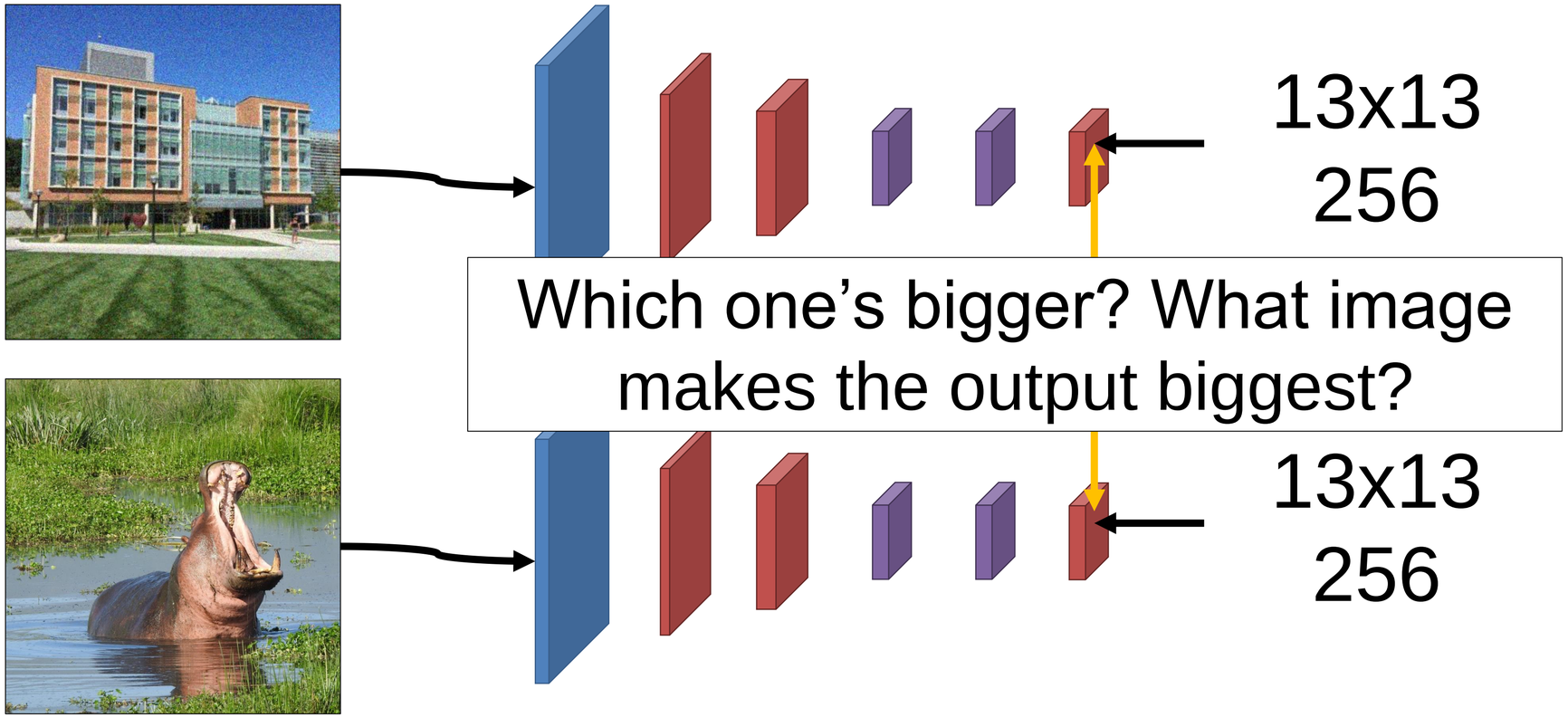
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5
227x227 3	55x559 6	27x272 56	13x133 84	13x13 384	13x13 256



CNN that extracts a 13x13x256 output

Understanding Later Filters

Feed an image in, see what score the filter gives it. A more pleasant version of a real neuroscience procedure.



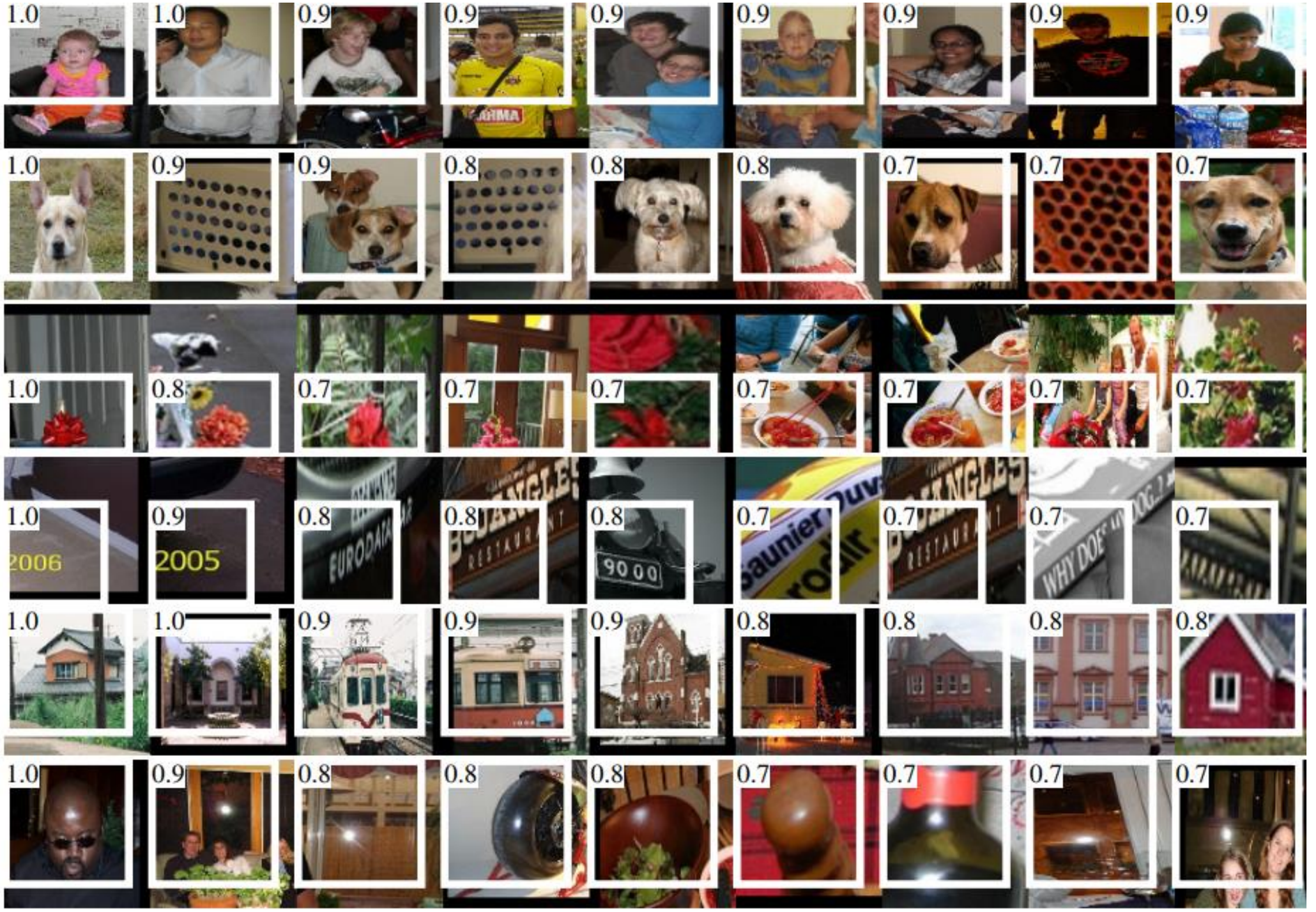
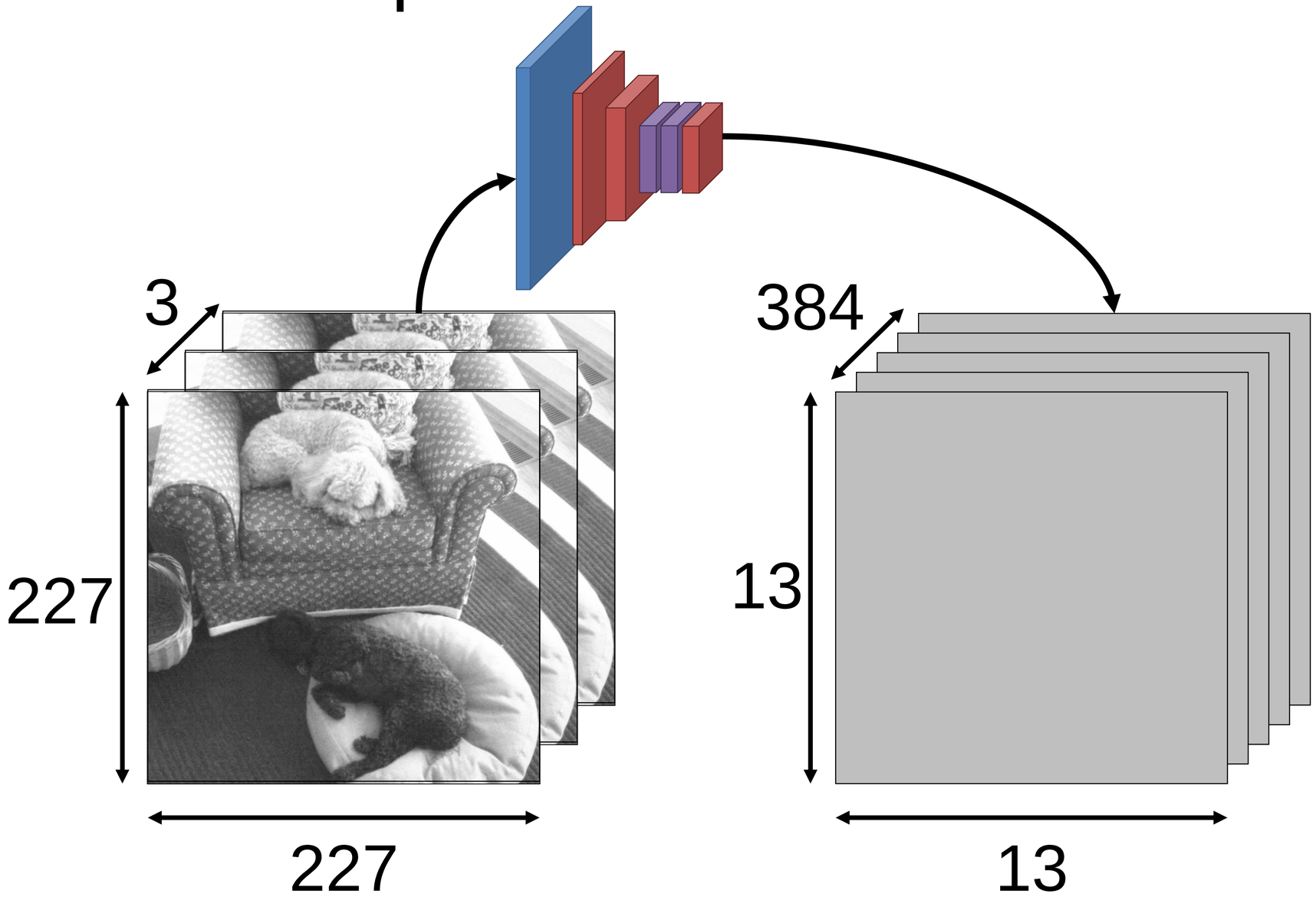
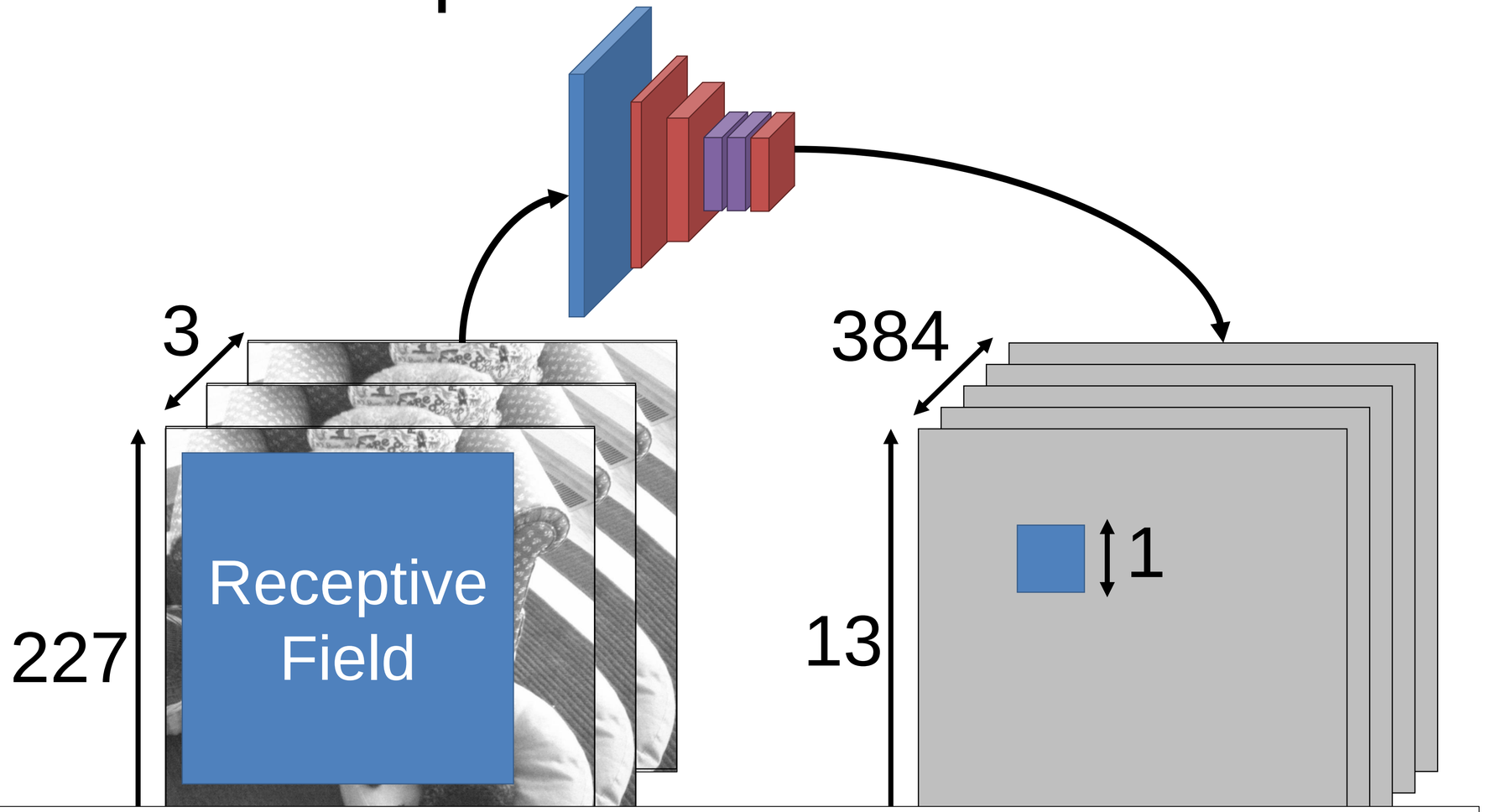


Figure Credit: Girschick et al. CVPR 2014.

What's Up With the White Boxes?

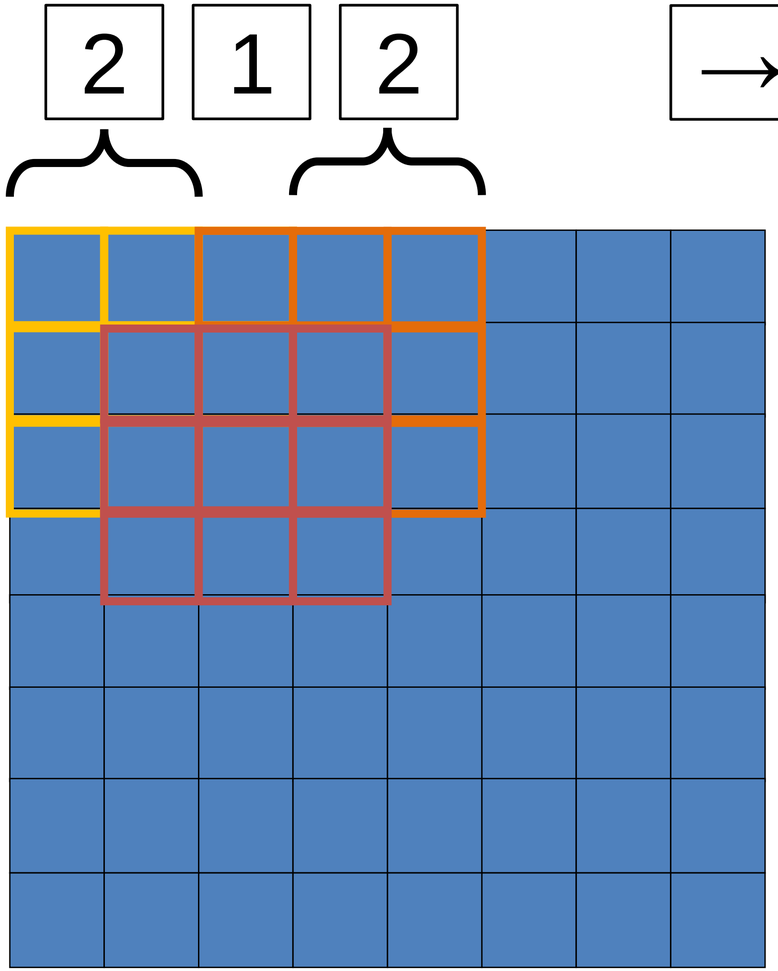


What's Up With the White Boxes?



Due to convolution, each later layer's value depends on / "sees" only a fraction of the input image.

VGG Net

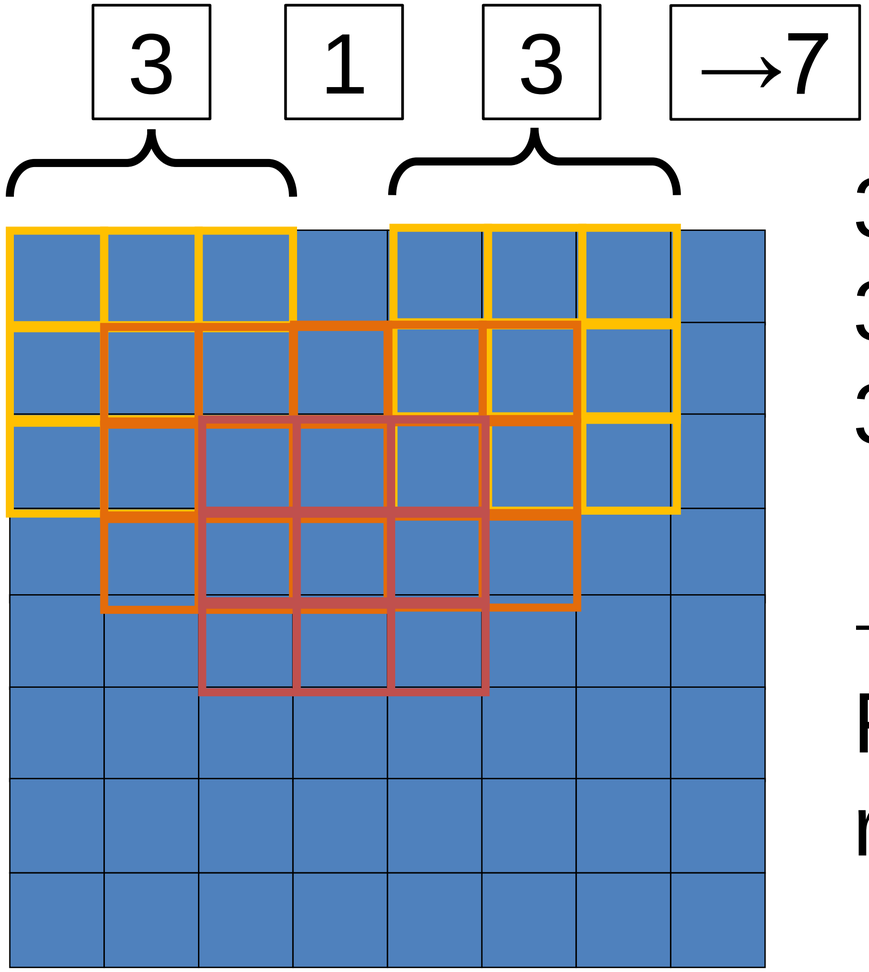


3x3 filter followed by
3x3 filter

→

Filter with 5x5
receptive field

Key Idea – 3x3 Filters

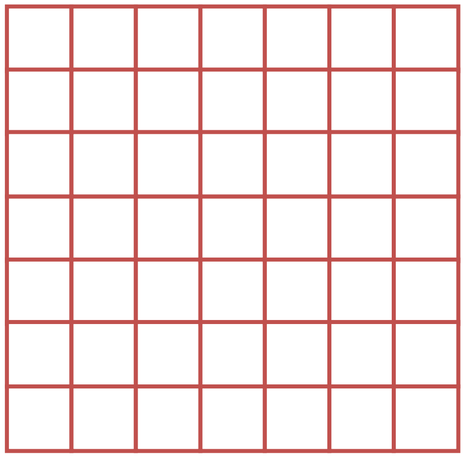


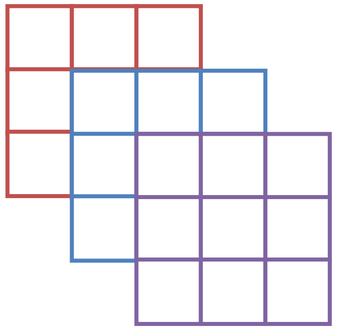
3x3 filter followed by
3x3 filter followed by
3x3 filter

→

Filter with 7x7
receptive field

Why Does This Make A Difference?

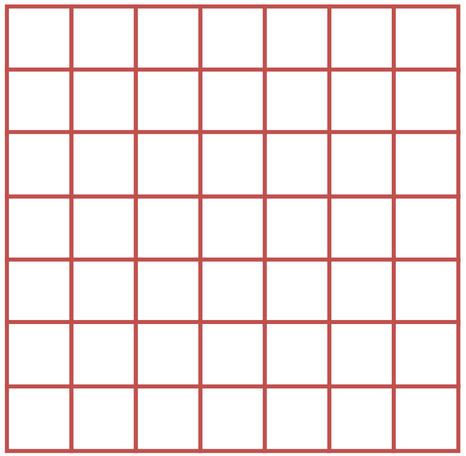




Empirically, repeated 3x3 filters do better compared to a 7x7 filter.

Why?

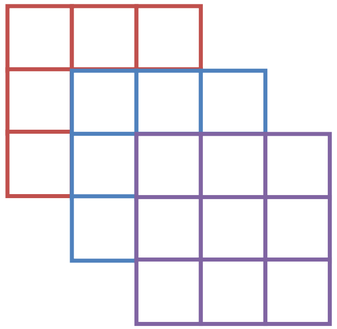
Key Idea – 3x3 Filters



Receptive Field: 7x7 pixels

Parameters/channel: 49

Number of ReLUs: 1



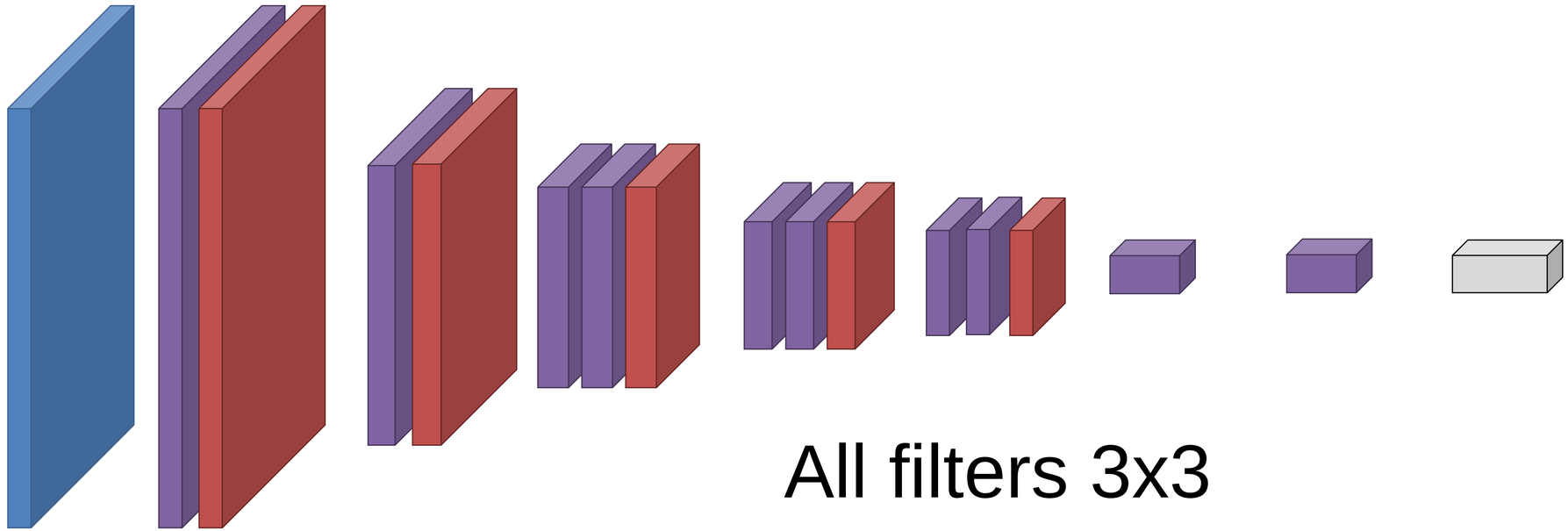
Receptive Field: 7x7 pixels

Parameters/channel: 3x3x3=**27**

Number of ReLUs: **3**

VGG16

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
224x224 3	224x224 64	112x112 128	56x56 256	28x28 512	14x14 512	1x1 4096	1x1 4096	1x1 1000



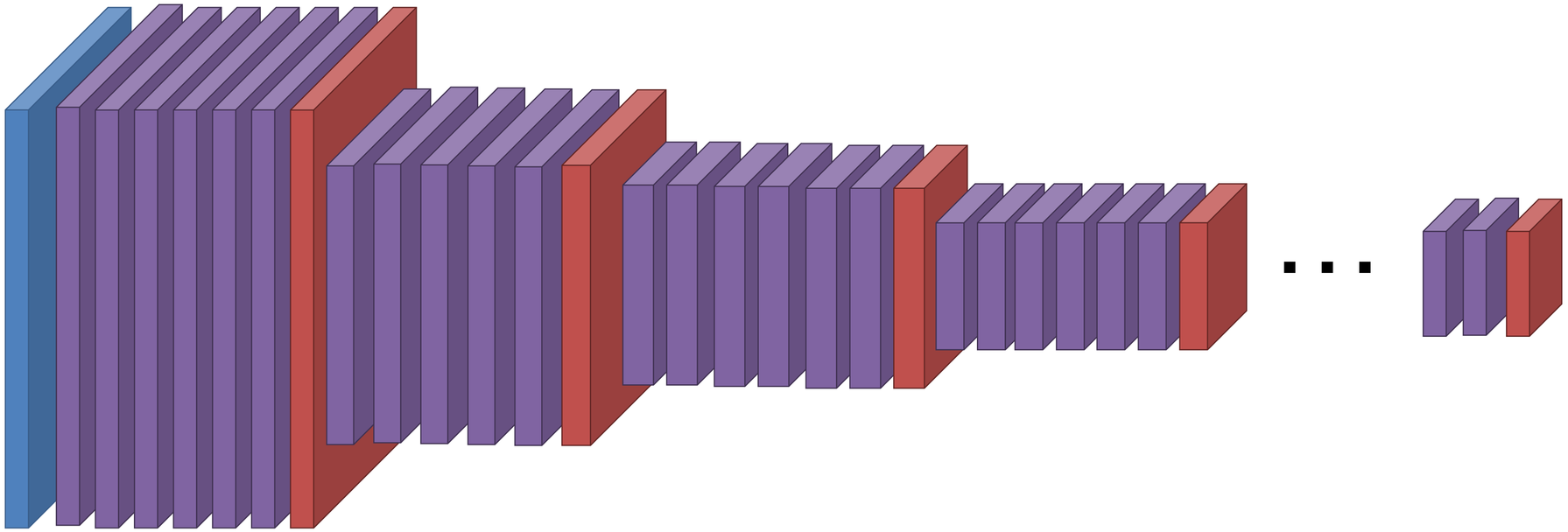
All filters 3x3

All filters followed by ReLU

Training Deeper Networks

Why not just stack continuously?

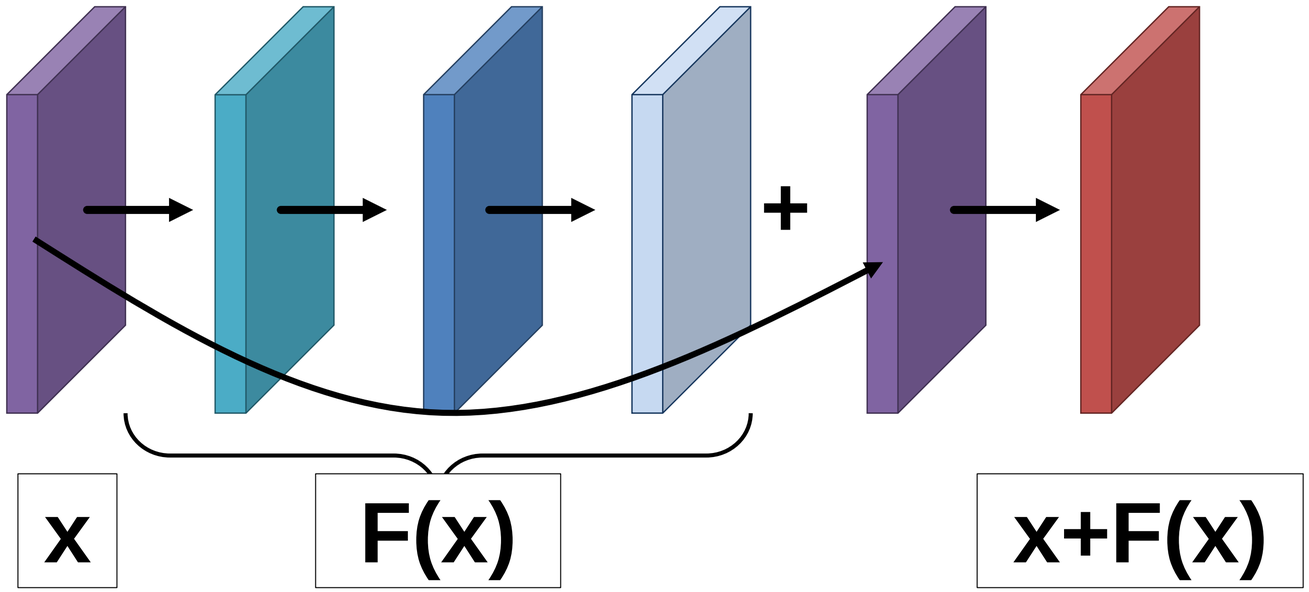
What will happen to gradient going back?



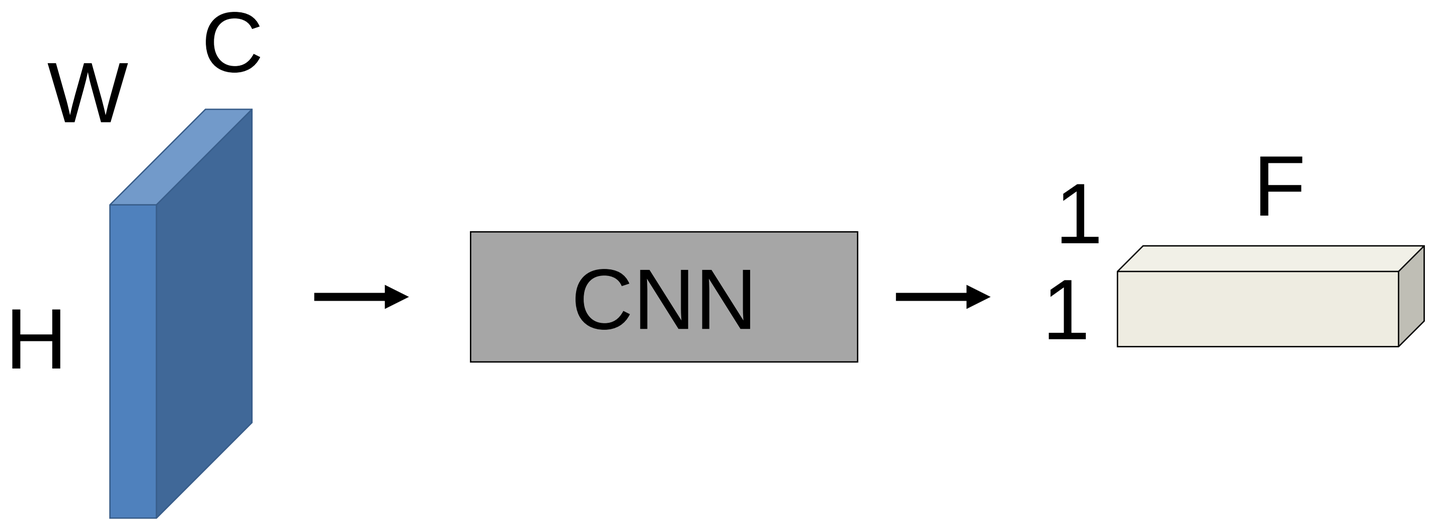
ResNet: Residual Learning

New Building Block: $\boldsymbol{x} + F(\boldsymbol{x})$

Lets you train networks with 100s of layers.



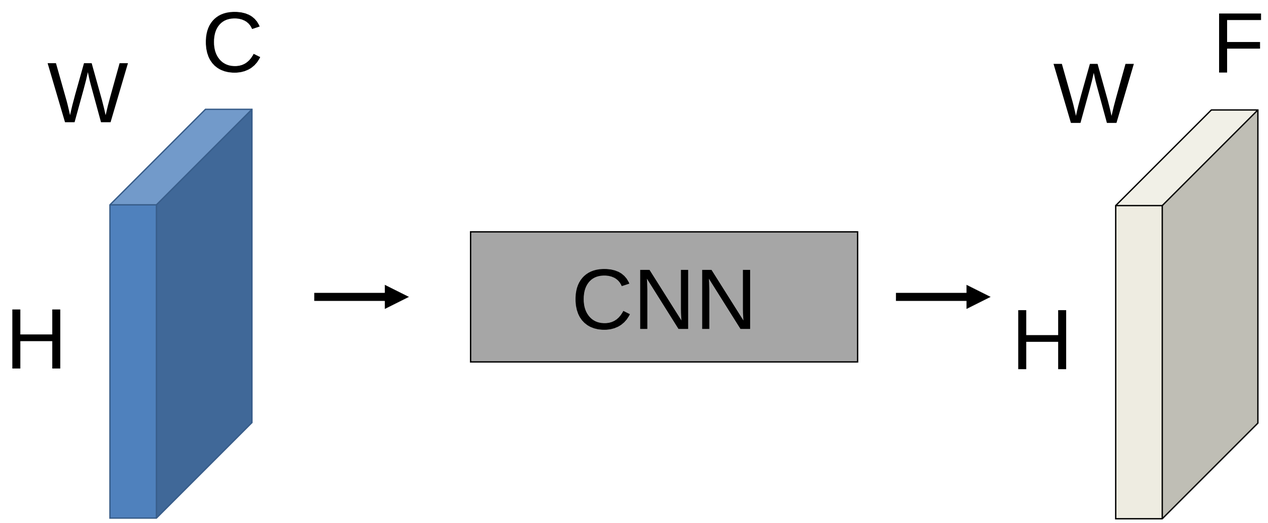
So Far…



Convert HxW image into a F-dimensional vector

Is this image a cat?
At what distance was this photo taken?
Is this image fake?

Pixel Labeling



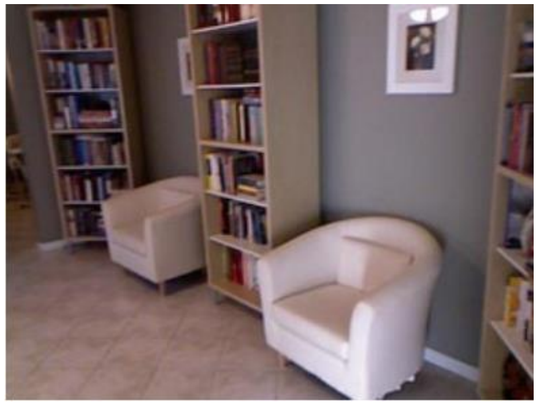
Convert HxW image into a F-dimensional vector

Which pixels in this image are a cat?
How far is each pixel away from the camera?
Which pixels of this image are fake?

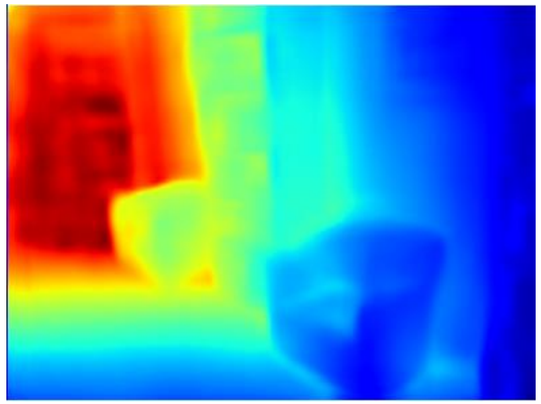
e.g. Depth Prediction

Instead: give label of depthmap, train network to do regression (e.g., $\|z_i - \hat{z}_i\|$ where z_i is the ground-truth and $\hat{z}_i$ the prediction of the network at pixel i).

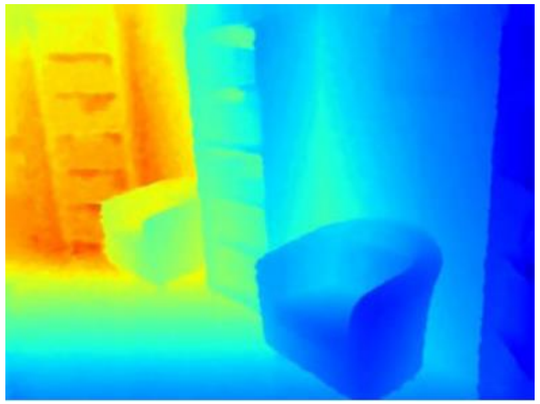
Input HxWx3
RGB Image



Output HxWx1
Depth Image

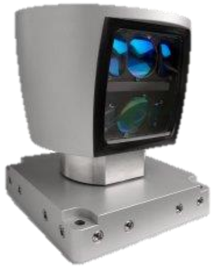


True HxWx1
Depth Image

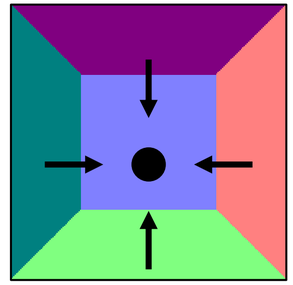


Result credit: Eigen and Fergus, ICCV 2015

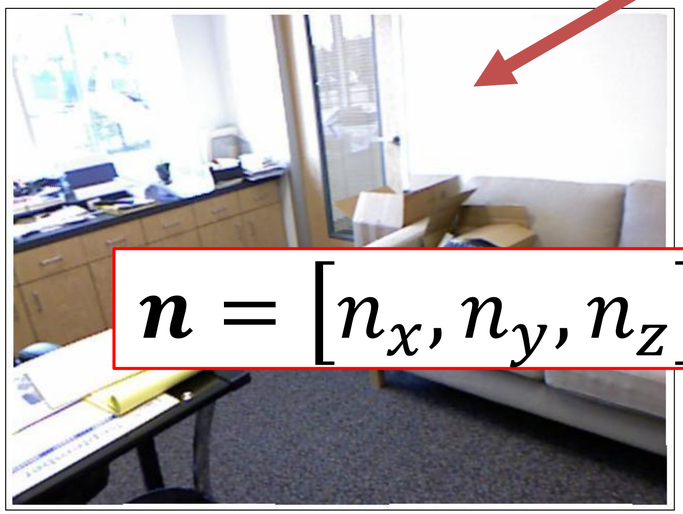
Surface Normals



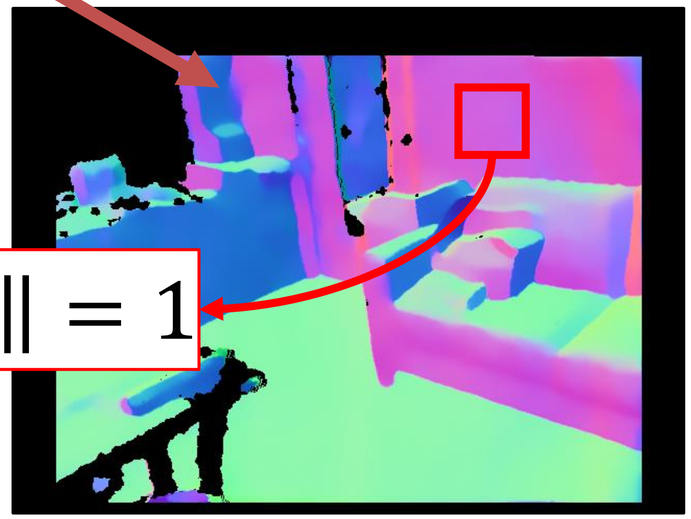
Room



Legend



$$\boldsymbol{n} = [n_x, n_y, n_z], \|\boldsymbol{n}\| = 1$$



Color Image

Normals

Image credit: NYU Dataset, Silberman et al. ECCV 2012

Surface Normals

Instead: train normal network to minimize $\|\boldsymbol{n}_i - \widehat{\boldsymbol{n}_i}\|$
where $\boldsymbol{n}_i$ is ground-truth and $\widehat{\boldsymbol{n}_i}$ prediction at pixel i.

Input: HxWx3
RGB Image

Output: HxWx3
Normals

Result credit: X. Wang, D. Fouhey, A. Gupta, *Designing Deep Networks for Surface Normal Estimation.* CVPR 2014

“Semantic Segmentation”

Each pixel has label, inc. **background**, and unknown
Usually visualized by colors.
Note: don’t distinguish between object *instances*

Input Label Input Label

Image Credit: Everingham et al. Pascal VOC 2012.

“Semantic Segmentation”

“Semantic”: a usually meaningless word.
Meant to indicate here that we’re **naming** things.

Input Label Input Label

Human Pose Estimation

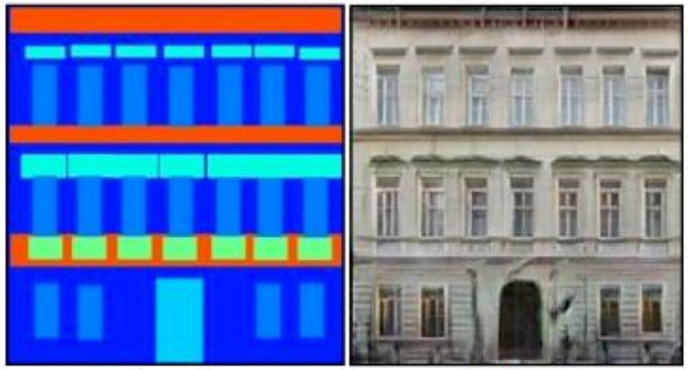
Result credit: Z. Cao et al. *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. CVPR 2017.

Generic Image-to-Image Translation

Labels to Street Scene

input output

Labels to Facade



input output

BW to Color

input output

Aerial to Map

input output

Day to Night

input output

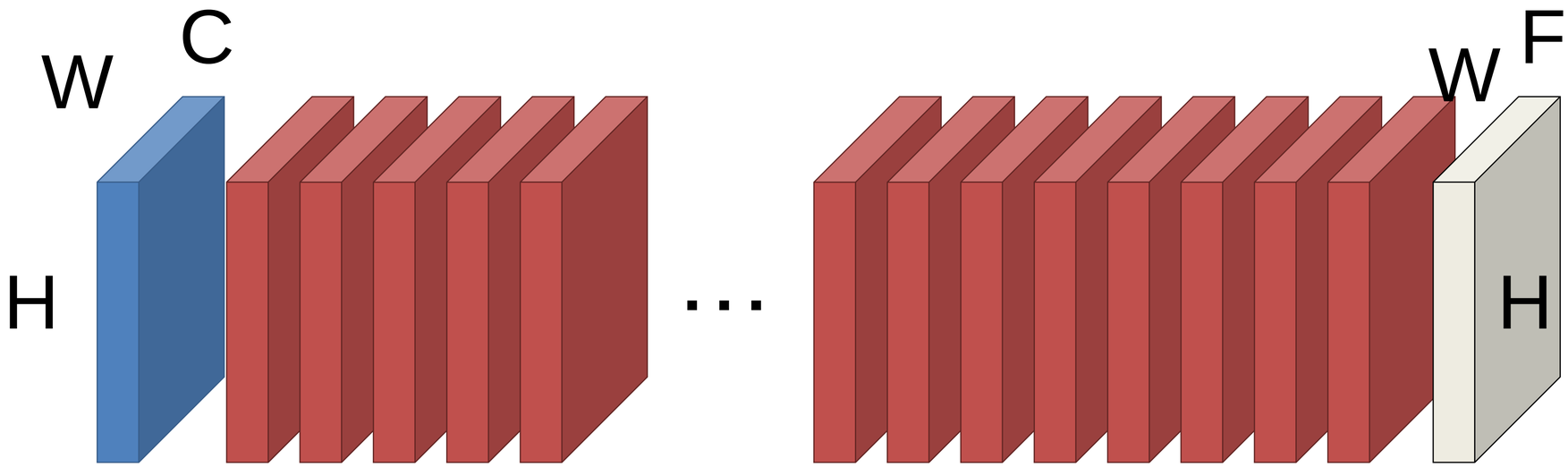
Edges to Photo

input output

First – Two “Wrong” Ways

- It’s helpful to see two “wrong” ways to do this.

Why Not Stack Convolutions?

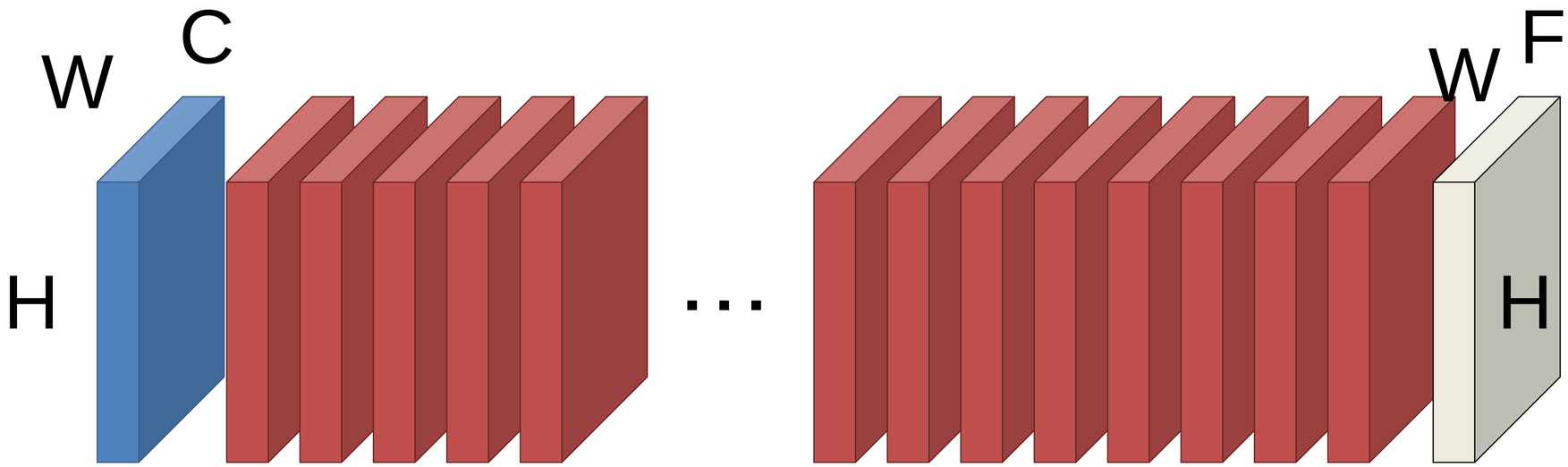


n 3x3 convs have a receptive field of 2n+1 pixels

How many convolutions until >=200 pixels?

100

Why Not Stack Convolutions?



Suppose 200 3x3 filters/layer, H=W=400

Storage/layer/image: 200 * 400 * 400 * 4 bytes = 122MB

Uh oh!*

*100 layers, batch size of 20 = 238GB of memory!

Idea #2

Crop out every sub-window and predict the label in the middle.

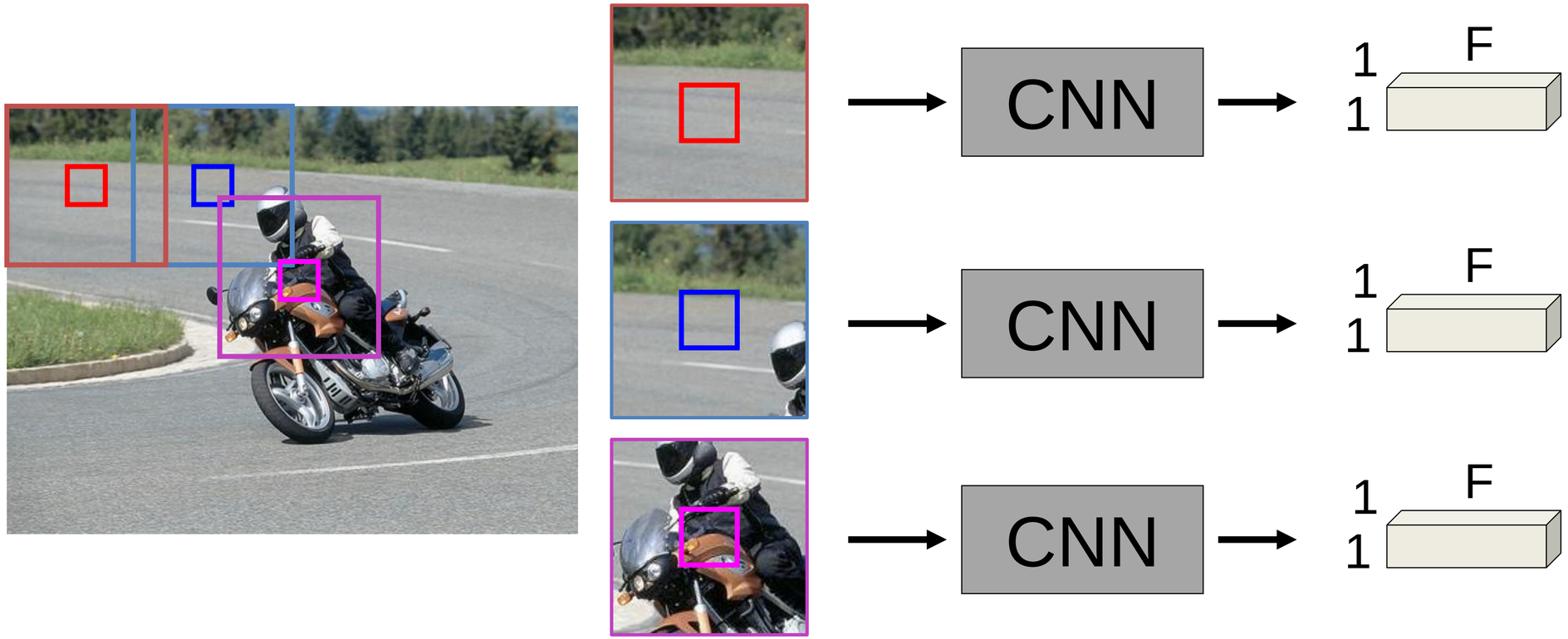
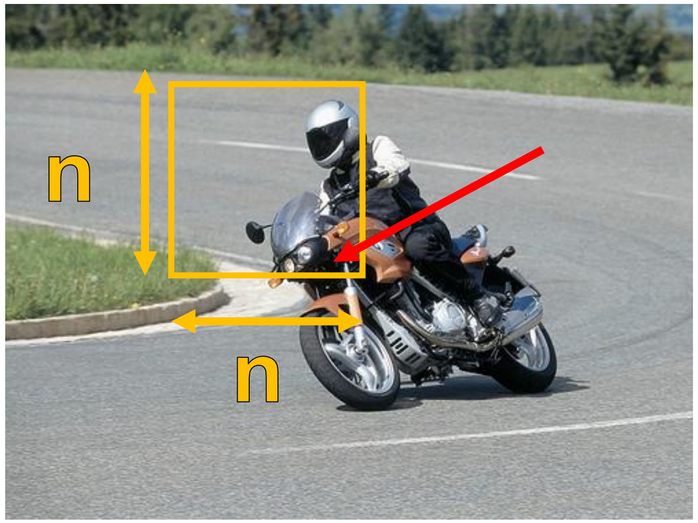


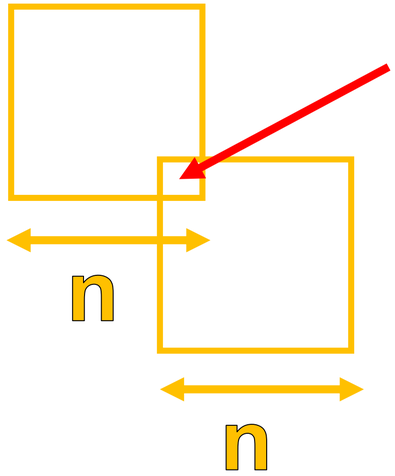
Image credit: PASCAL VOC, Everingham et al.

Idea #2

Meet “Gabor”. We extract NxN patches and do independent CNNs. **How many times does Gabor filter the red pixel?**

Gabor





Answer:
(2n-1)*(2n-1)

Image credit: PASCAL VOC, Everingham et al.

The Big Issue

We need to:

1. Have large receptive fields to figure out what we're looking at
2. Not waste a ton of time or memory while doing so

These two objectives are in total conflict