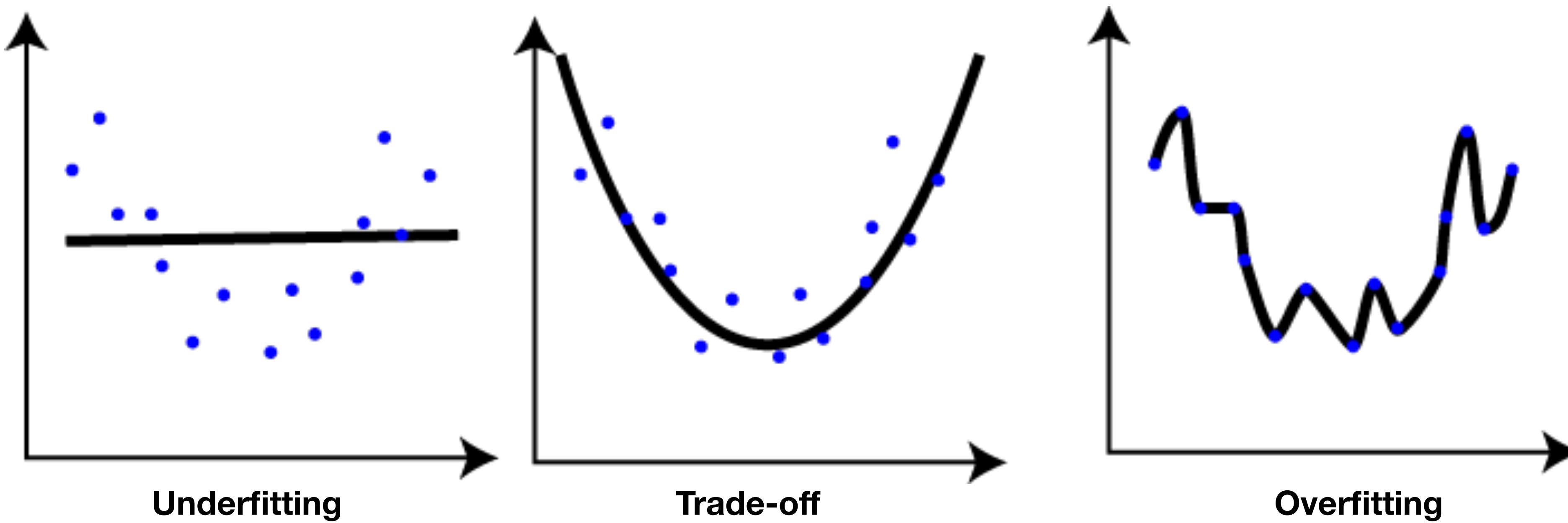


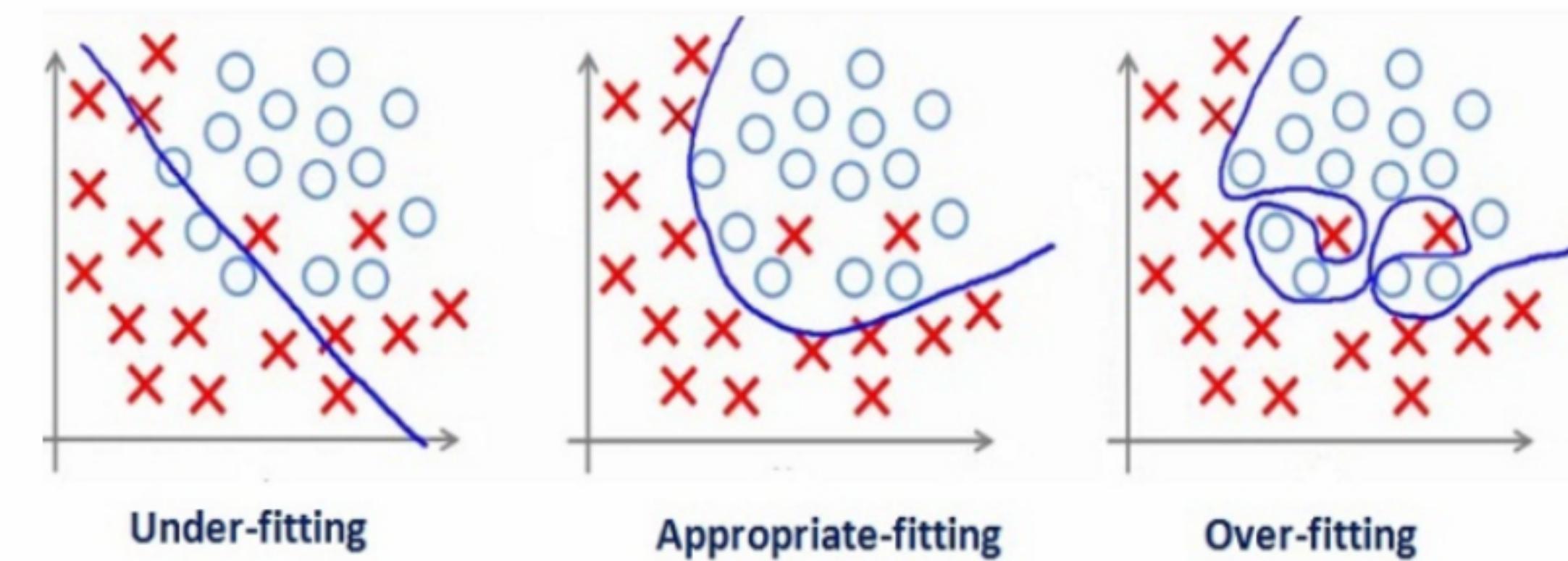
Last time..

Generalization

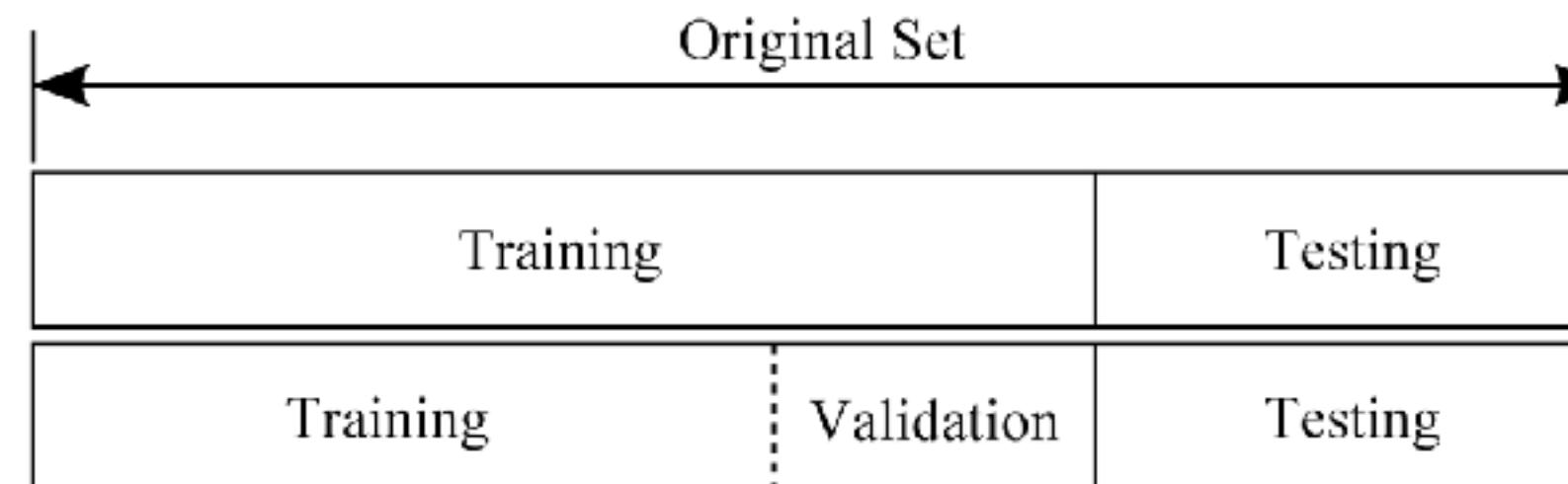
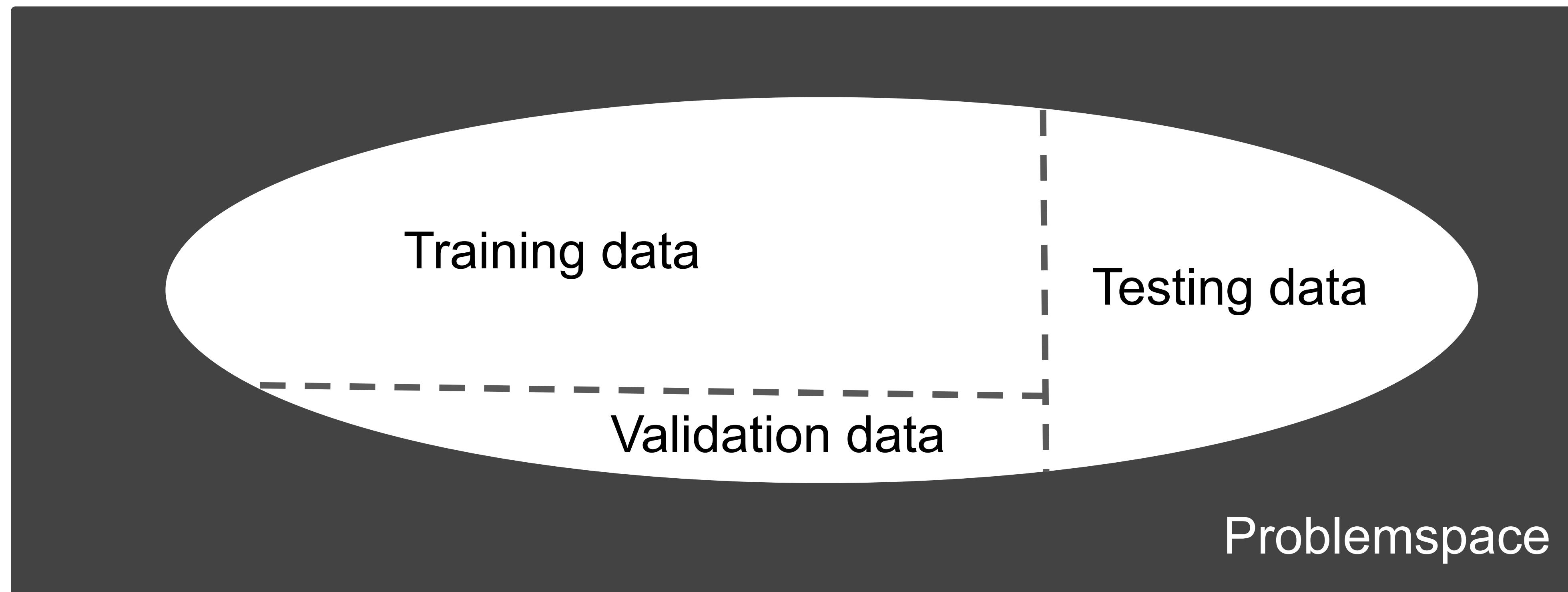
Generalization (vs. memorization)



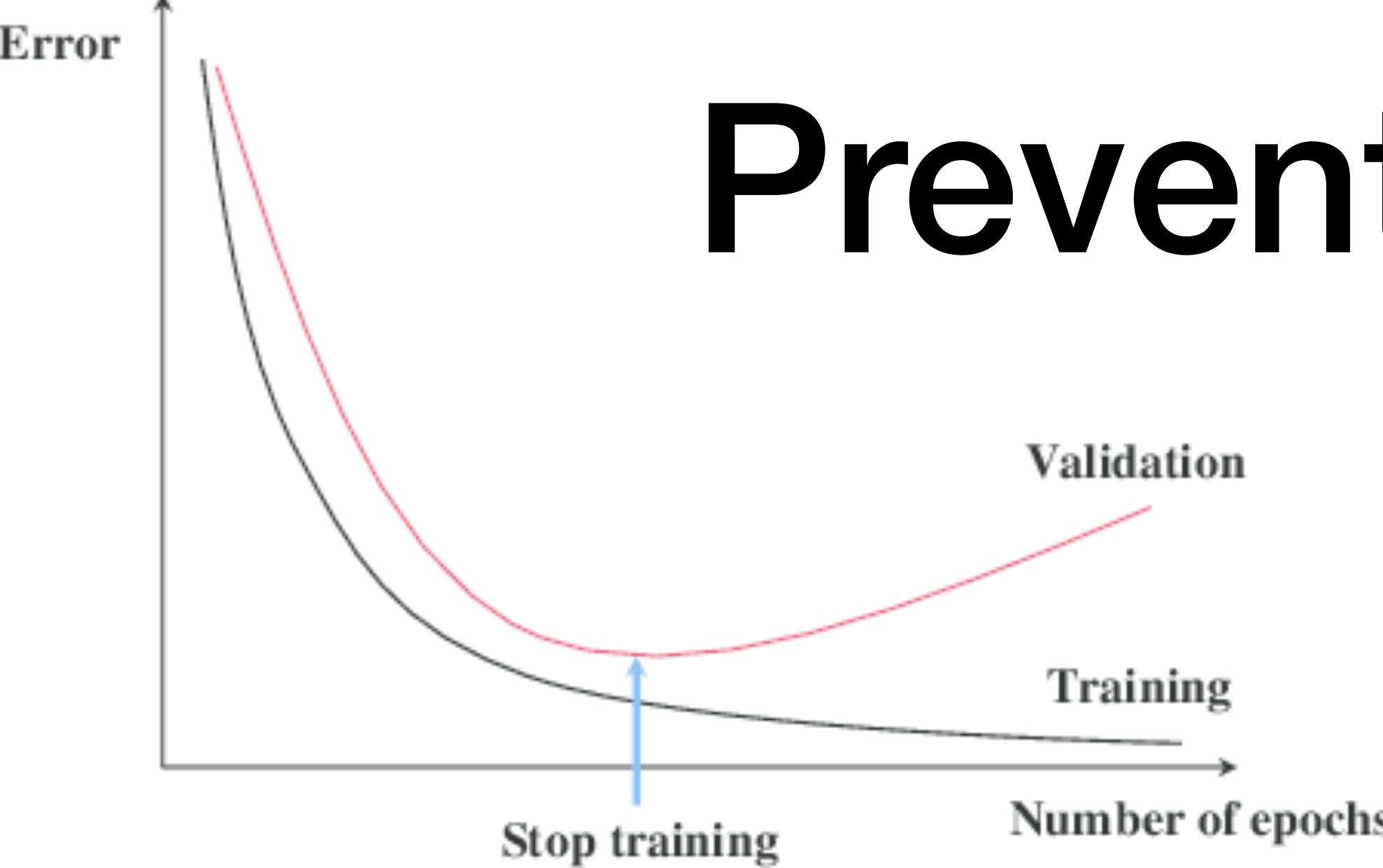
Goal:
Perform well on unseen data



All Data, Labeled data, Training, Validation and Test data

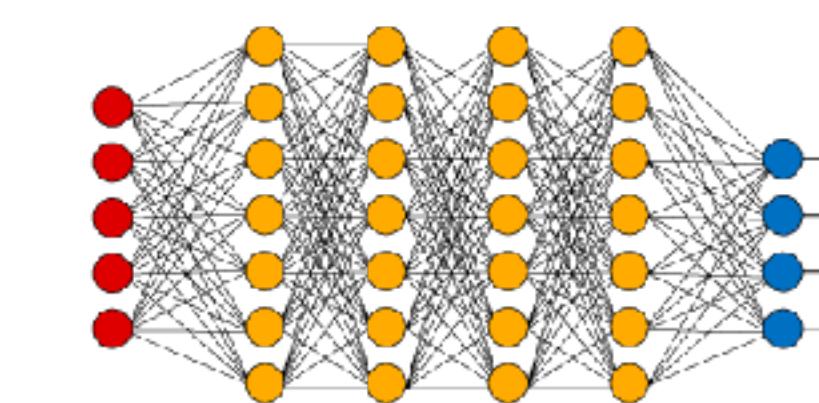


Preventing Overfitting

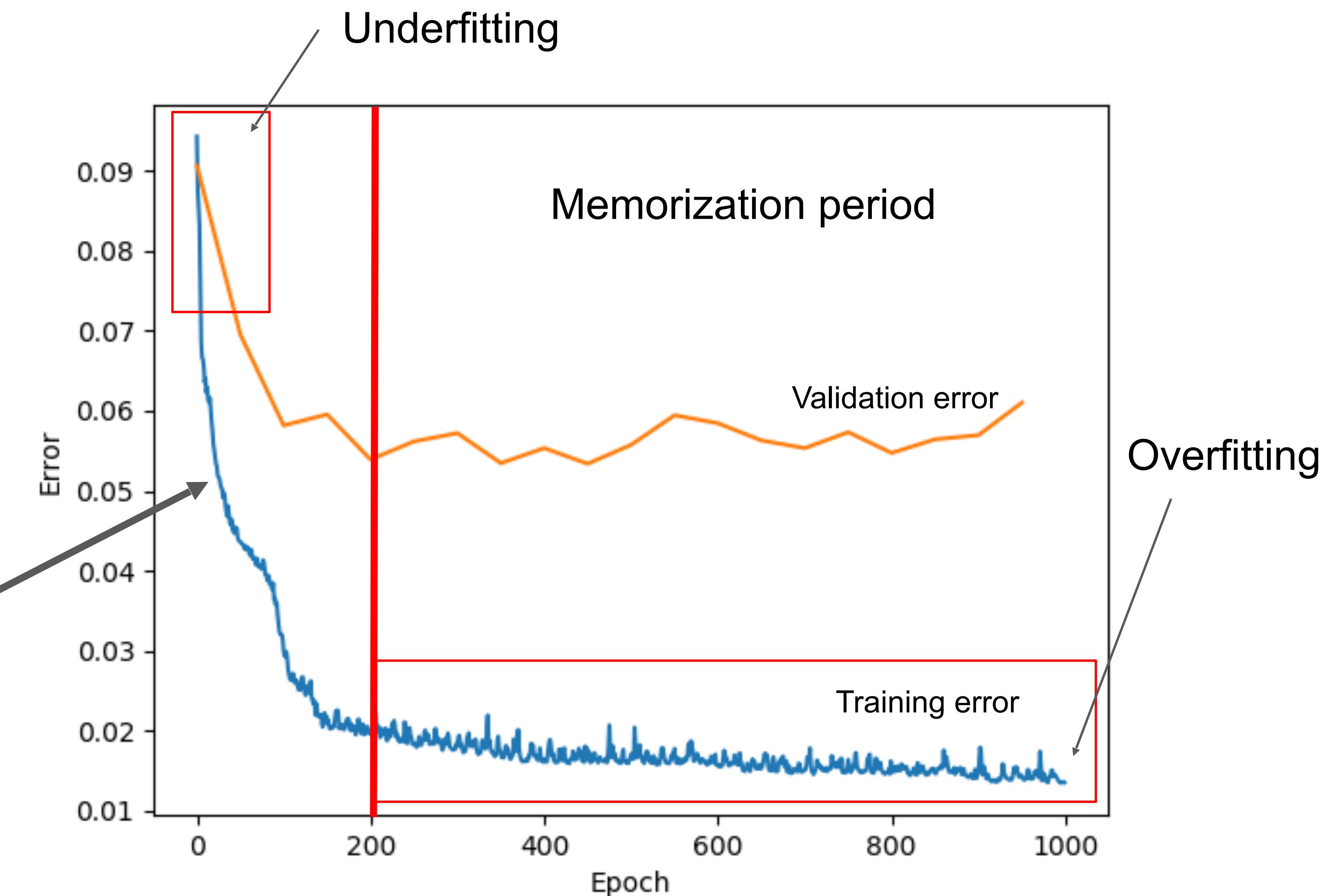


Why overfitting?

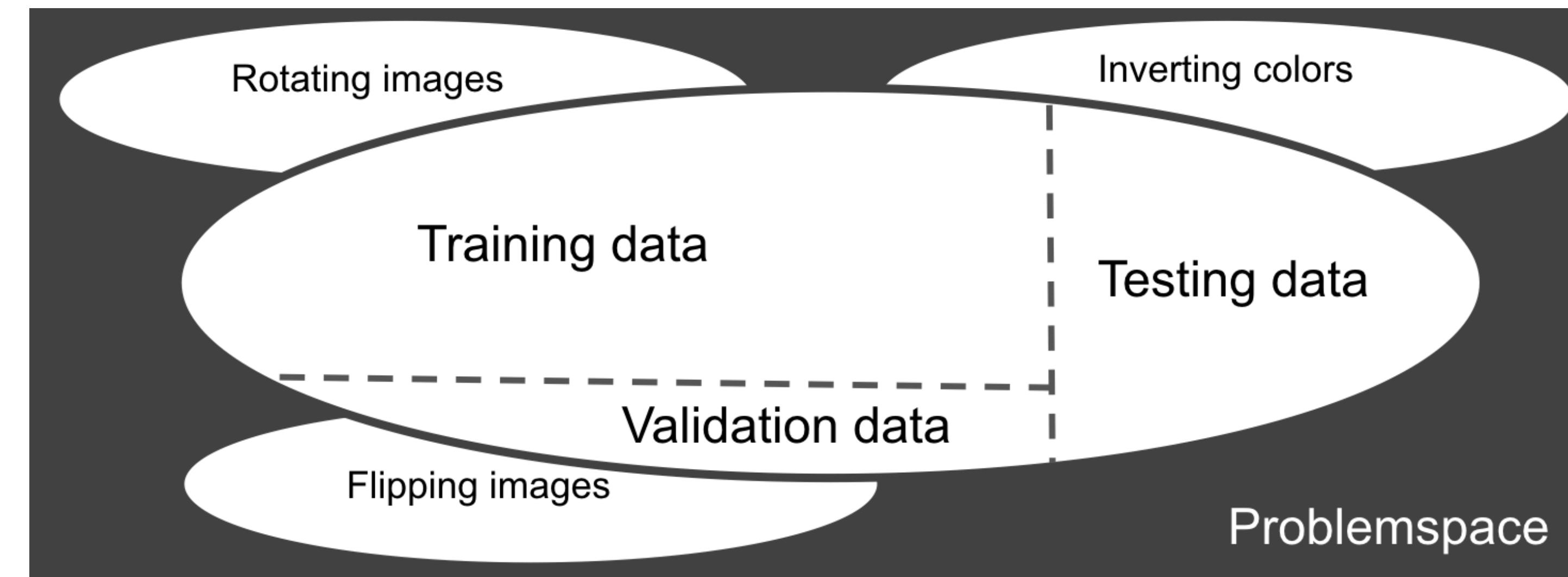
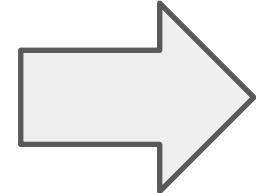
- Many **parameters** (high model capacity)
 - -> Reduce model complexity
- Little training **data**
 - -> Increase training set
- Combination



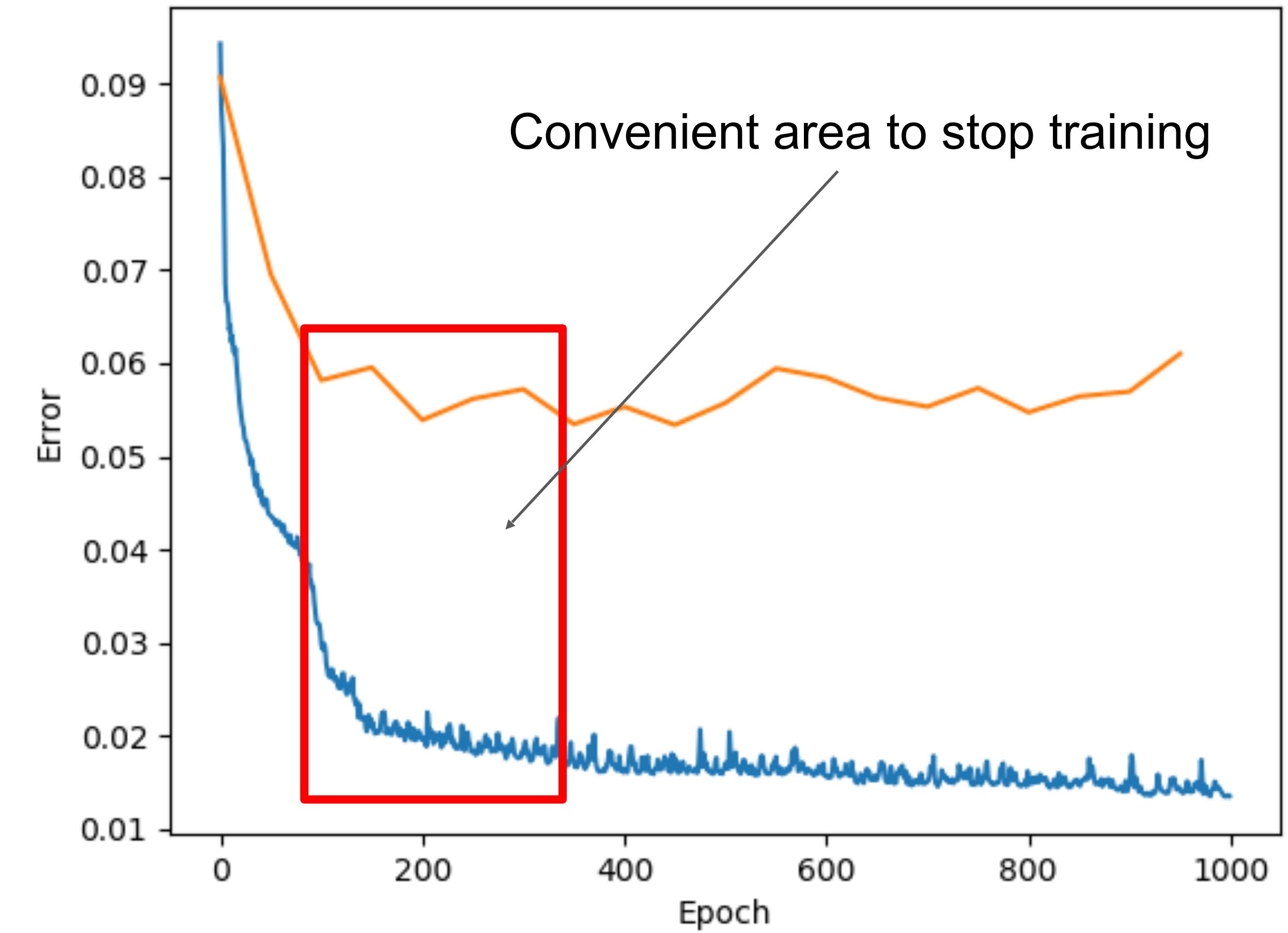
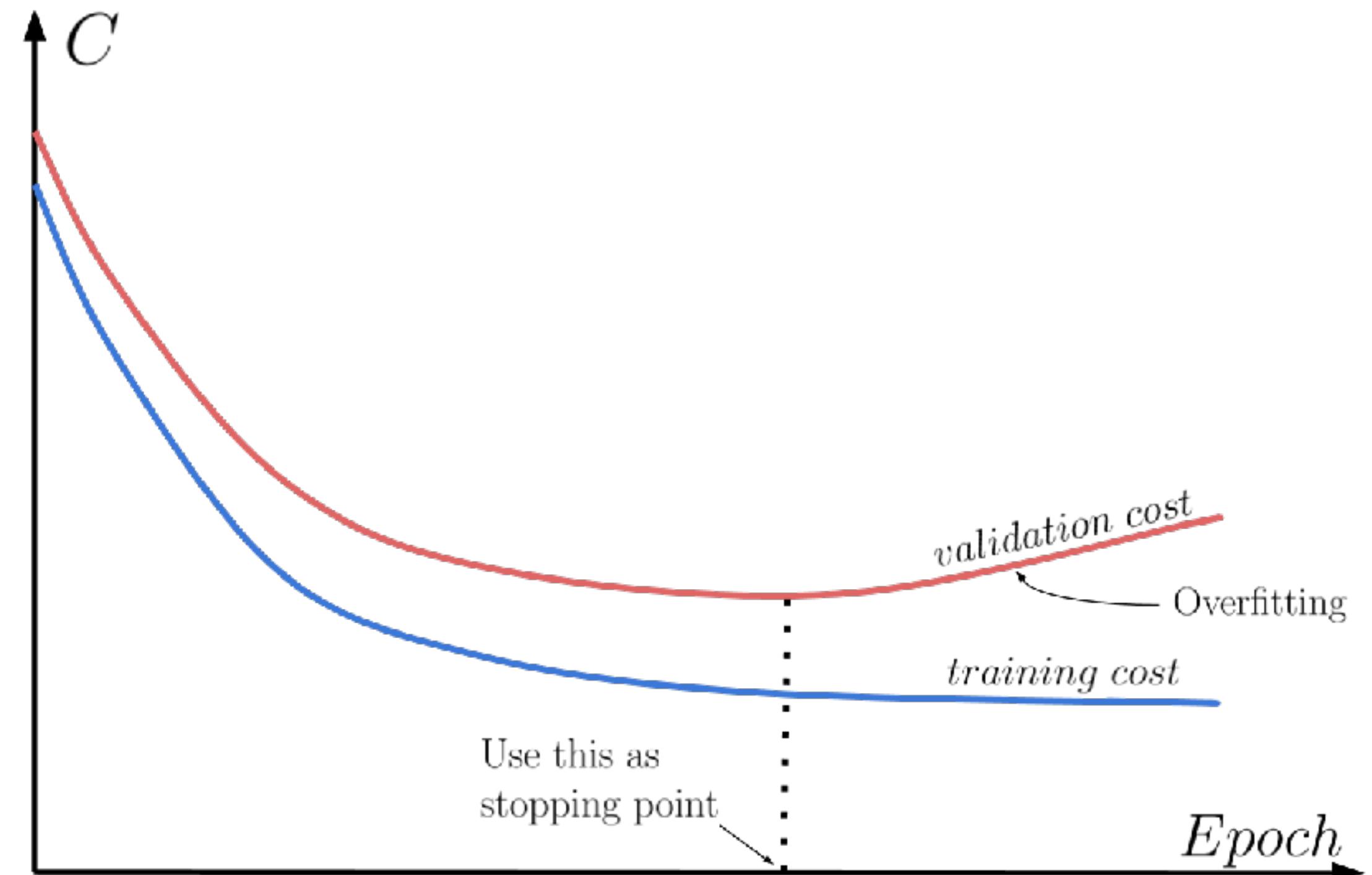
Generalization period



Data augmentation



Early stopping



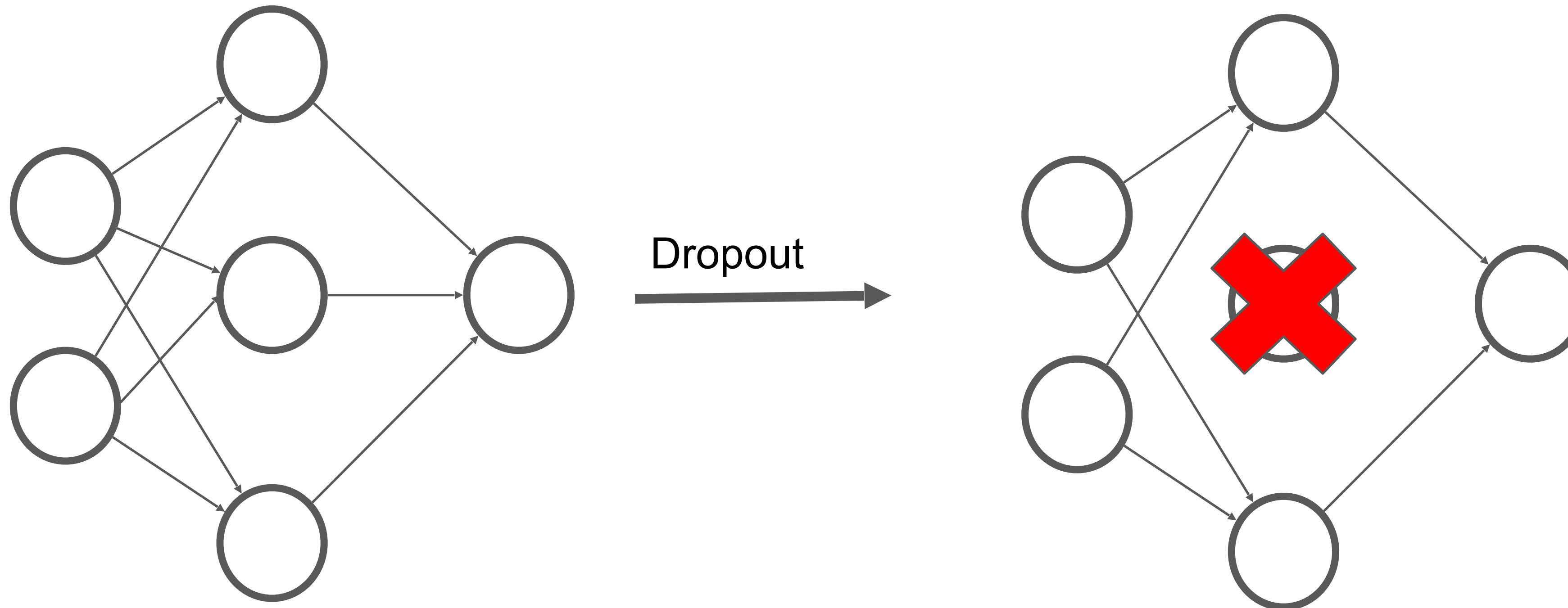
Regularization (Weight decay)

$$\tilde{C}(\theta) = C(\theta) + \alpha \Omega(\theta)$$

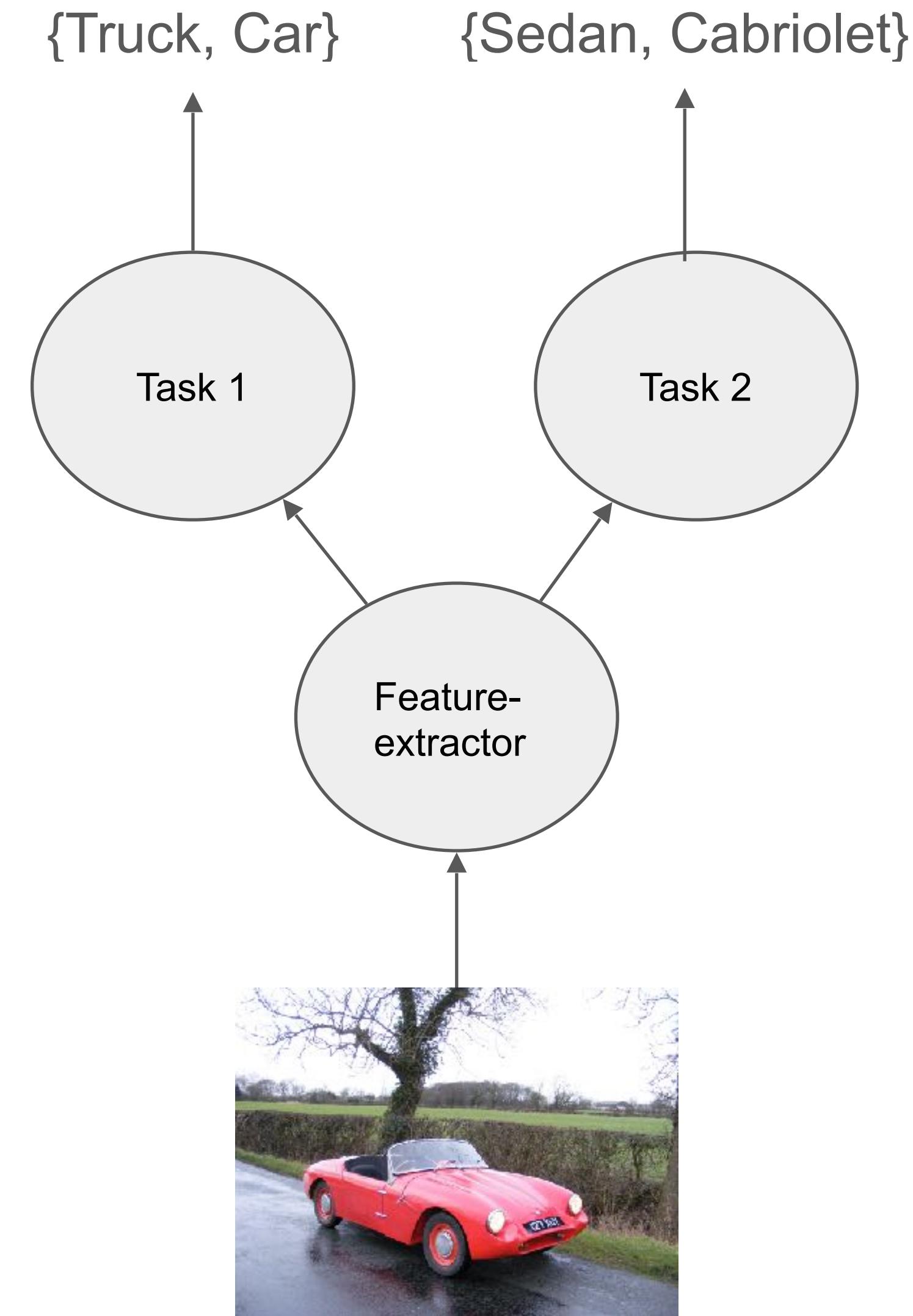
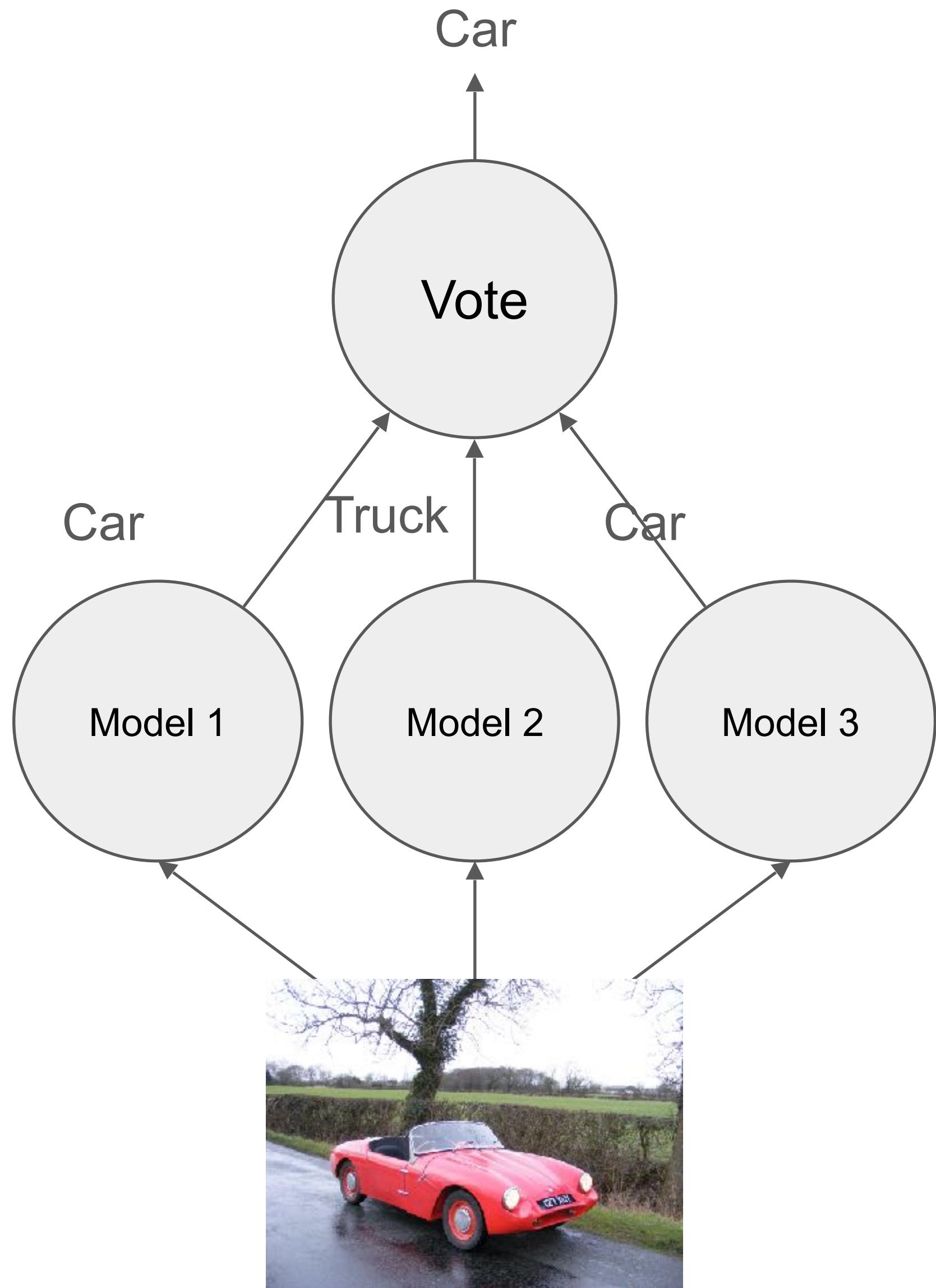
$$\Omega(\theta) = ||w||^2$$

$$\Omega(\theta) = ||w||$$

Dropout

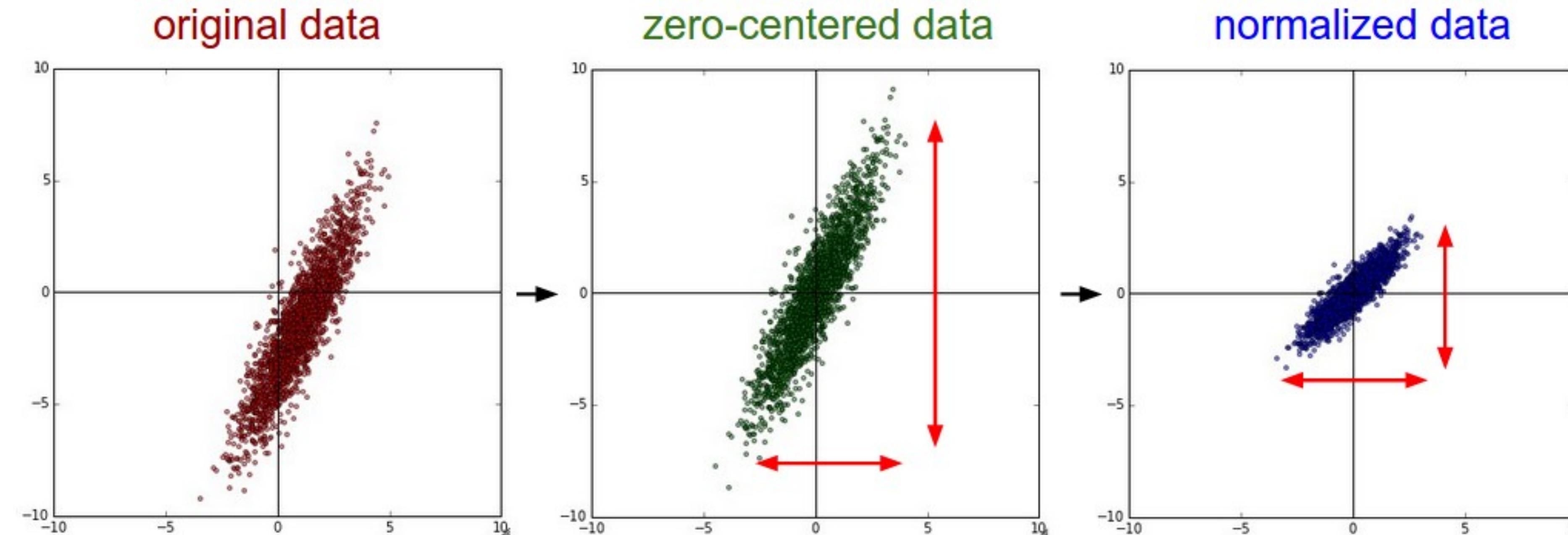


Ensemble learning / Multitask learning

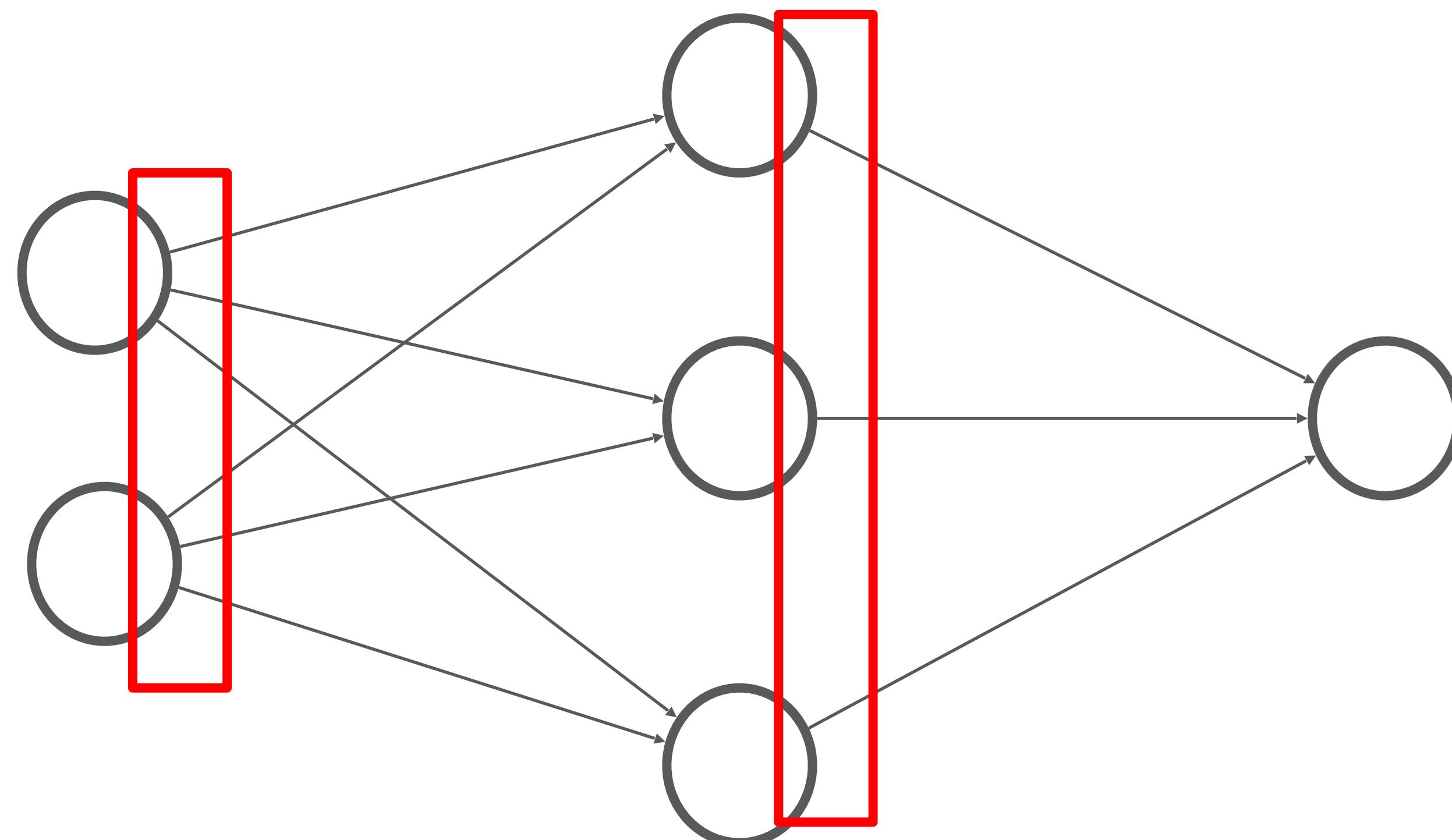


Preprocessing / Input Normalization

- Cleaning and structuring - Garbage in = garbage out
- Zero-Centering of the data - Subtract mean image
- Normalization of the data - Divide by the standard deviation



Batch Normalization



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Parameter (weight) Initialization

In general: biases are initialized with 0 and weights are initialized with random numbers (mean: 0, std.dev: <1)

He initialization

$$\sqrt{\frac{2}{\text{size}^{[l-1]}}}$$

$$W^{[l]} = \text{np.random.randn}(\text{size_l}, \text{size_l-1}) * \text{np.sqrt}(2/\text{size_l-1})$$

Xavier initialization

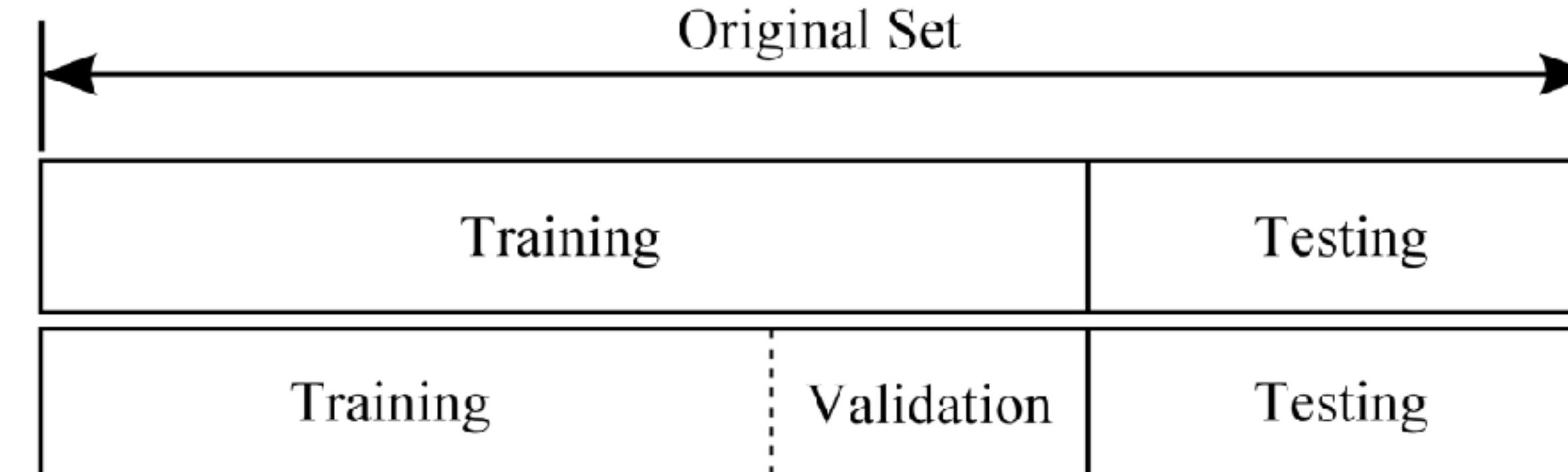
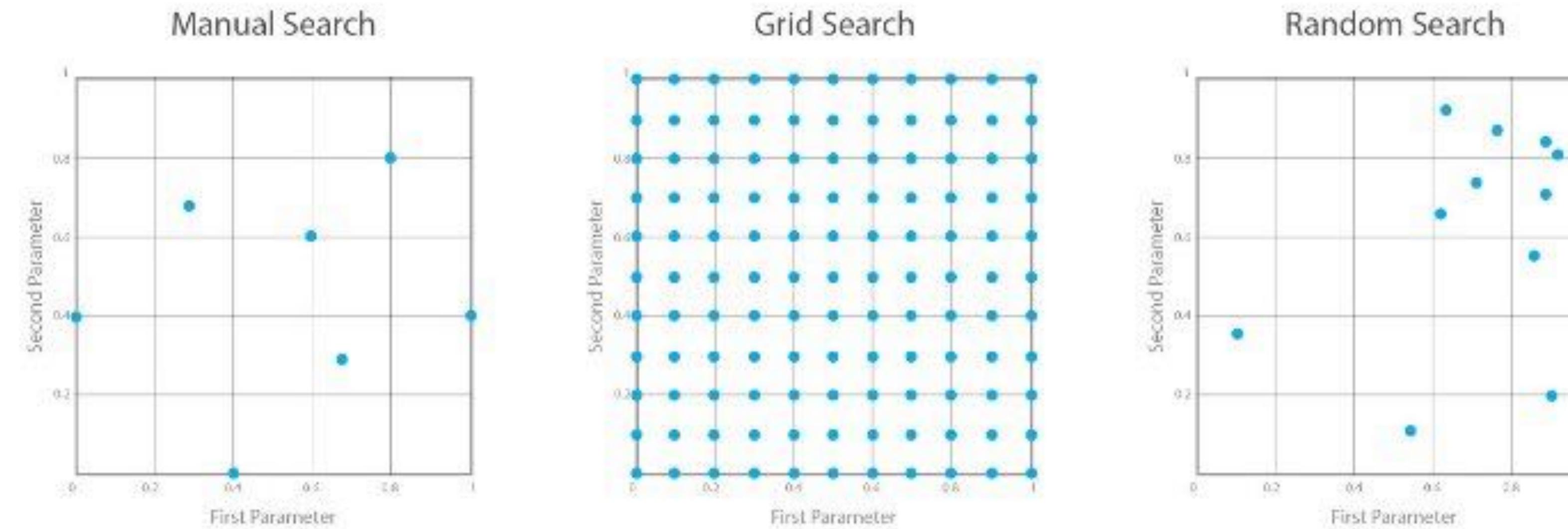
$$\sqrt{\frac{1}{\text{size}^{[l-1]}}}$$

$$W^{[l]} = \text{np.random.randn}(\text{size_l}, \text{size_l-1}) * \text{np.sqrt}(1/\text{size_l-1})$$

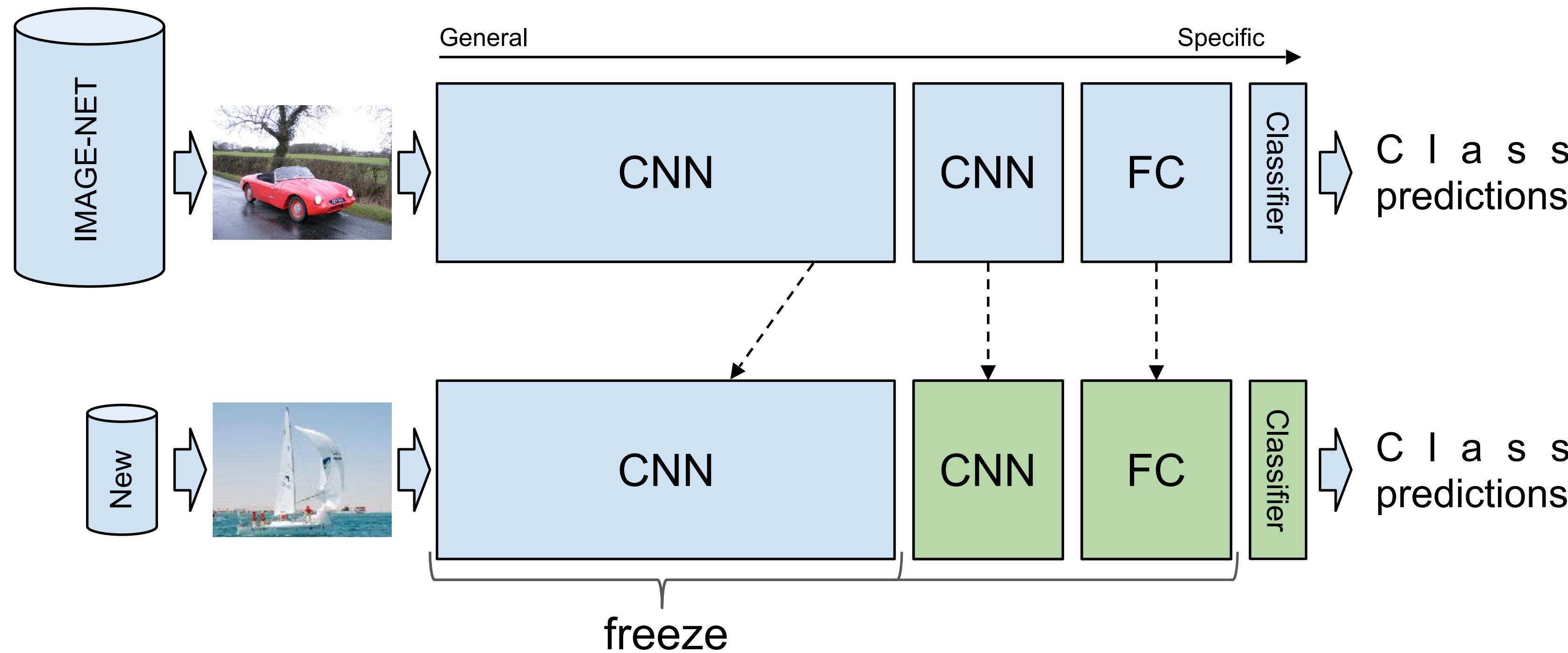
$$\sqrt{\frac{2}{\text{size}^{[l-1]} + \text{size}^{[l]}}}$$

$$W^{[l]} = \text{np.random.randn}(\text{size_l}, \text{size_l-1}) * \text{np.sqrt}(2 / (\text{size_l-1} + \text{size_l}))$$

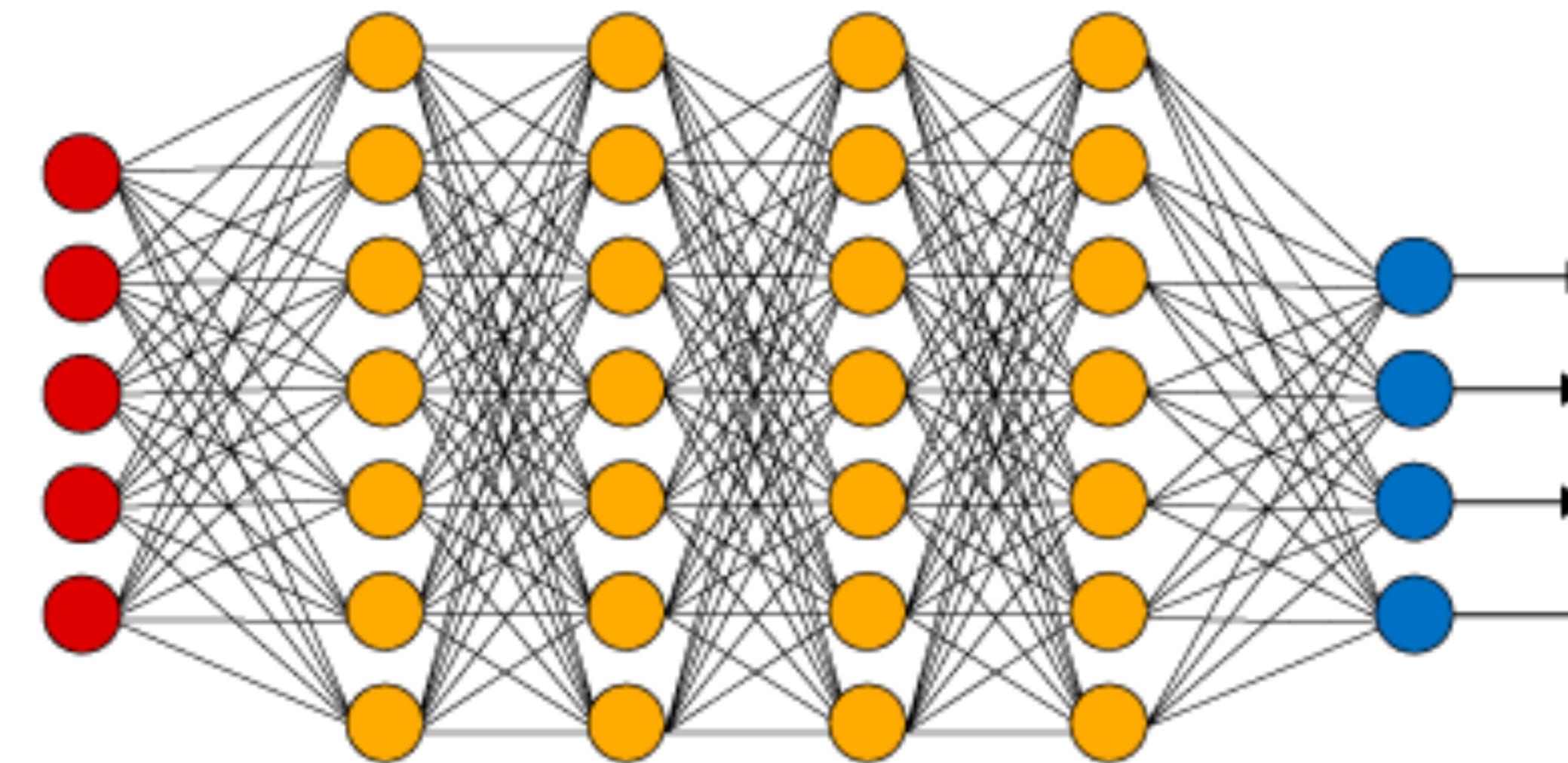
Hyper-parameter Initialization and optimization



Transfer learning



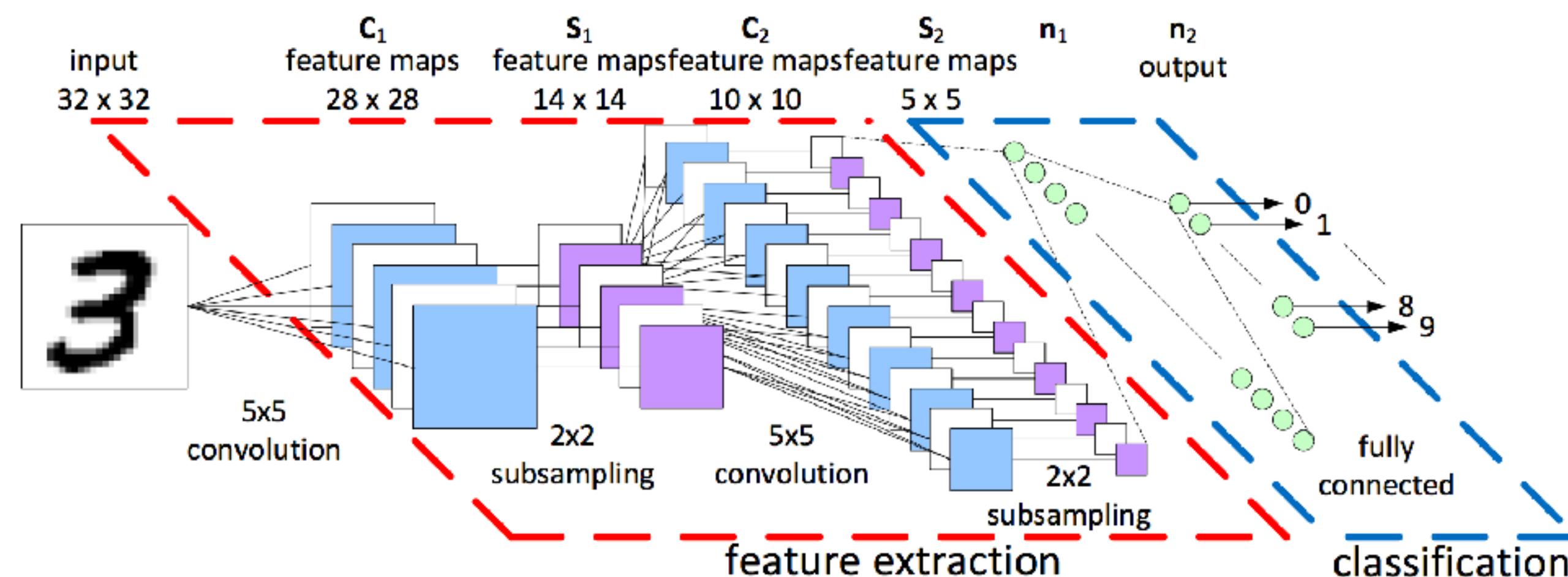
Challenges with Deep Neural Networks



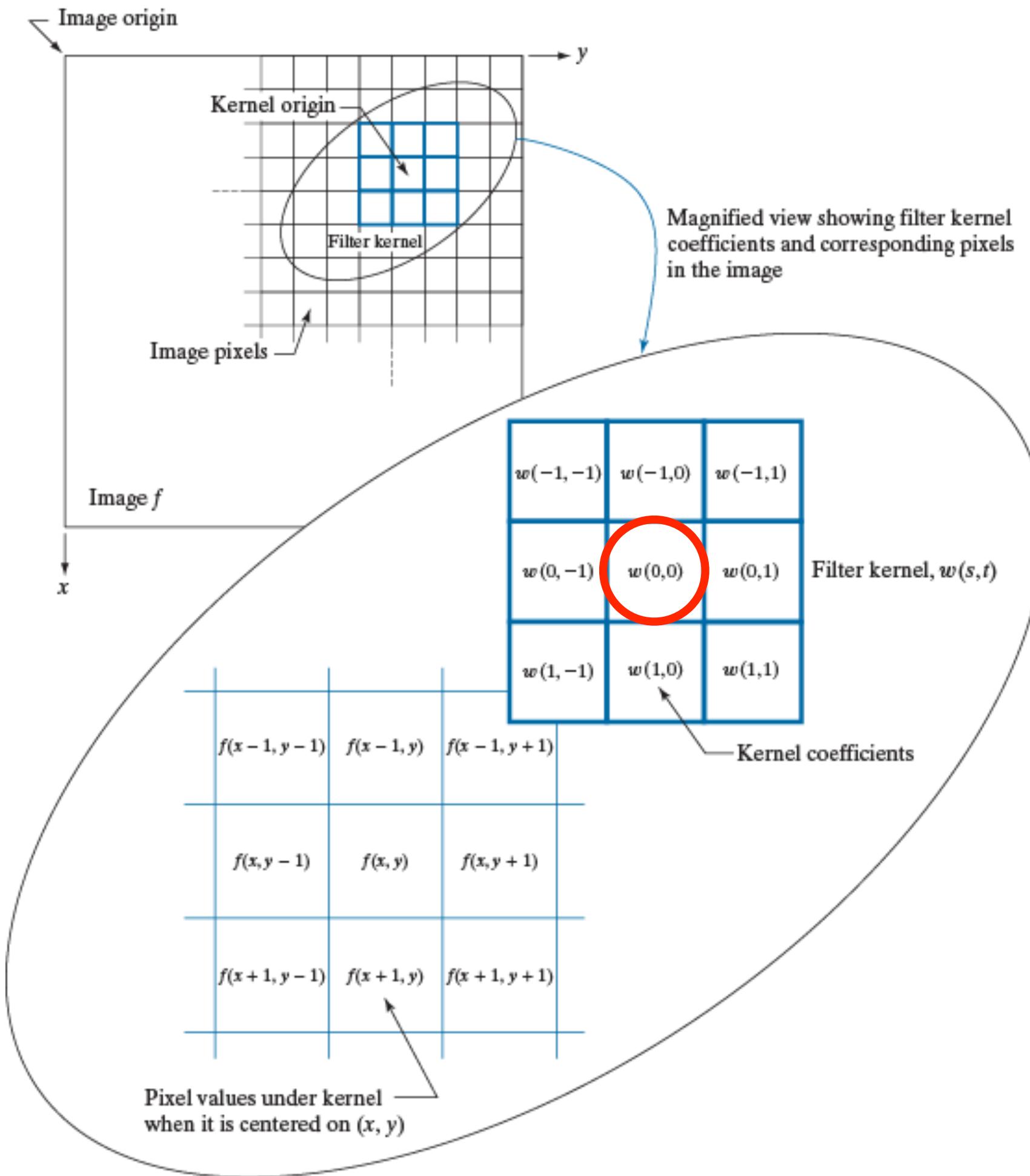
Vanishing, Exploding and Unstable gradients

Today: DL-based CV: CNNs & ImClass

Frank Lindseth, IDI, NTNU



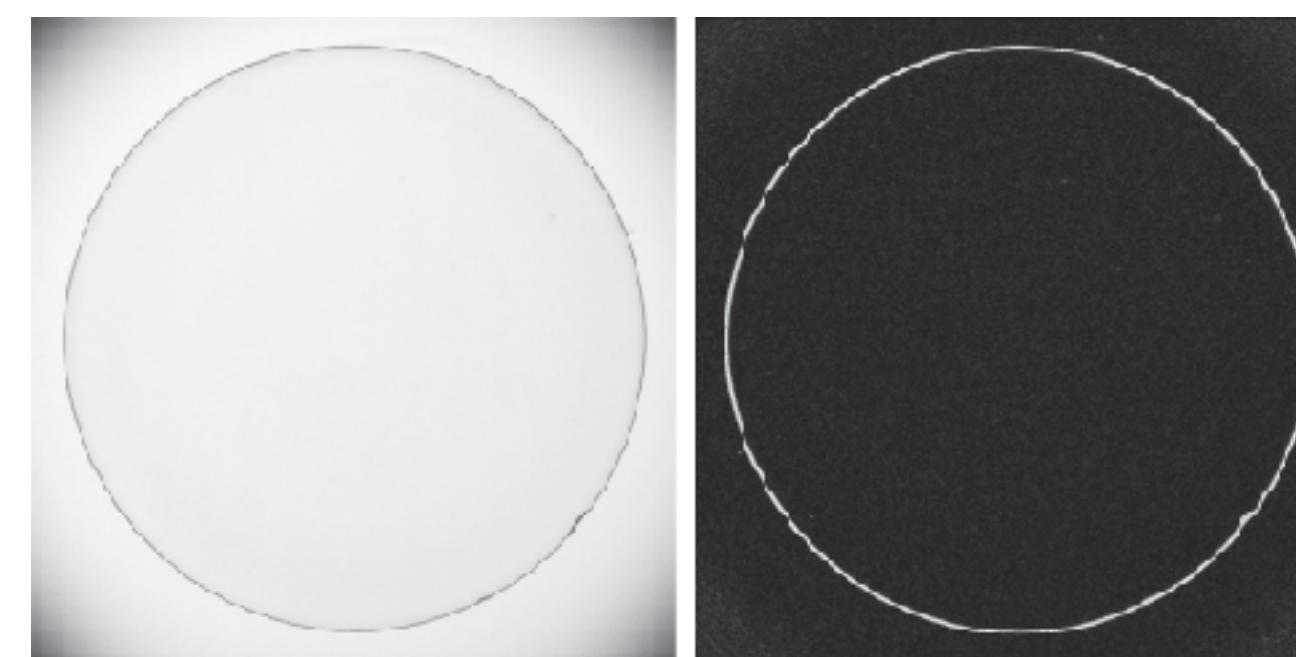
Motivation: Old school



kernel = [1 0 -1]

-1	-2	-1	
0	0	0	
1	2	1	

-1	0	1
-2	0	2
-1	0	1



								Padded f
	Origin	f						0 0 0 0 0 0 0
	0 0 0 0 0							0 0 0 0 0 0 0
	0 0 0 0 0		w					0 0 0 0 0 0 0
	0 0 1 0 0	1	2	3				0 0 0 0 0 0 0
	0 0 0 0 0	4	5	6				0 0 0 0 0 0 0
	0 0 0 0 0	7	8	9				0 0 0 0 0 0 0

← Rotated w	Convolution result				
9 8 7	0	0	0	0	
6 5 4	0	0	0	0	0 0 0 0 0
3 2 1	0	0	0	0	0 1 2 3 0
0 0 0	1	0	0	0	0 4 5 6 0
0 0 0	0	0	0	0	0 7 8 9 0
0 0 0	0	0	0	0	0 0 0 0 0
0 0 0	0	0	0	0	0 0 0 0 0

Full convolution result						
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	2	3	0	0
0	0	4	5	6	0	0
0	0	7	8	9	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Motivation: Old school

255 x 0	255 x 0	255 x 0	114	114	114	114
255 x 1	255 x 0	255 x -1	114	114	114	114
255 x 0	255 x 0	255 x 0	114	114	114	114
255	255	255	114	114	114	114
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

0				

255	255	255	114	114	114	114
255	255	255	x 0	x 0	x 0	x 0
255	255	255	x 1	x 0	x -1	
255	255	255	x 0	x 0	x 0	x 0
255	255	255	255	114	114	114
255	255	255	255	255	255	255
255	255	255	255	255	255	255

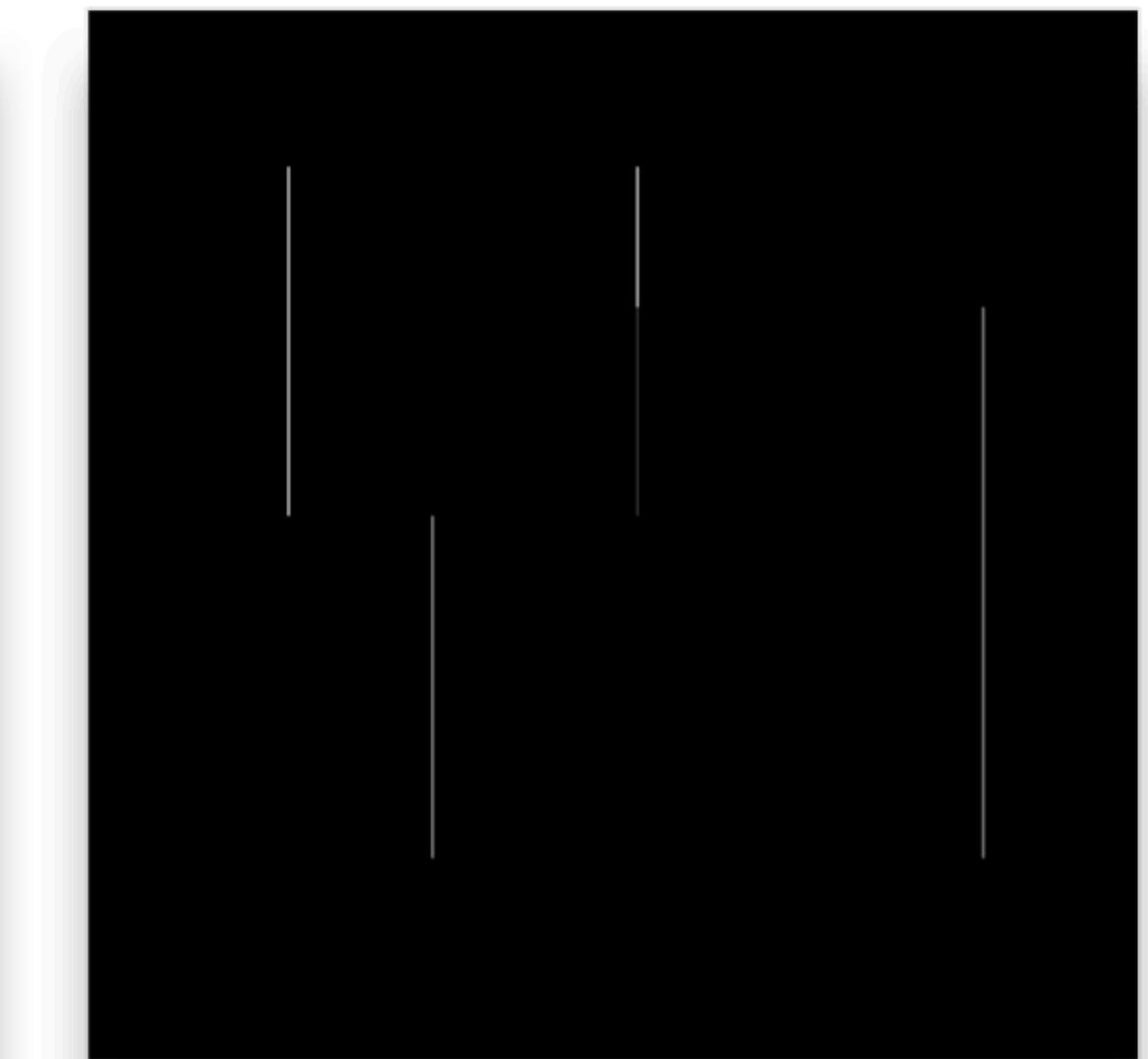
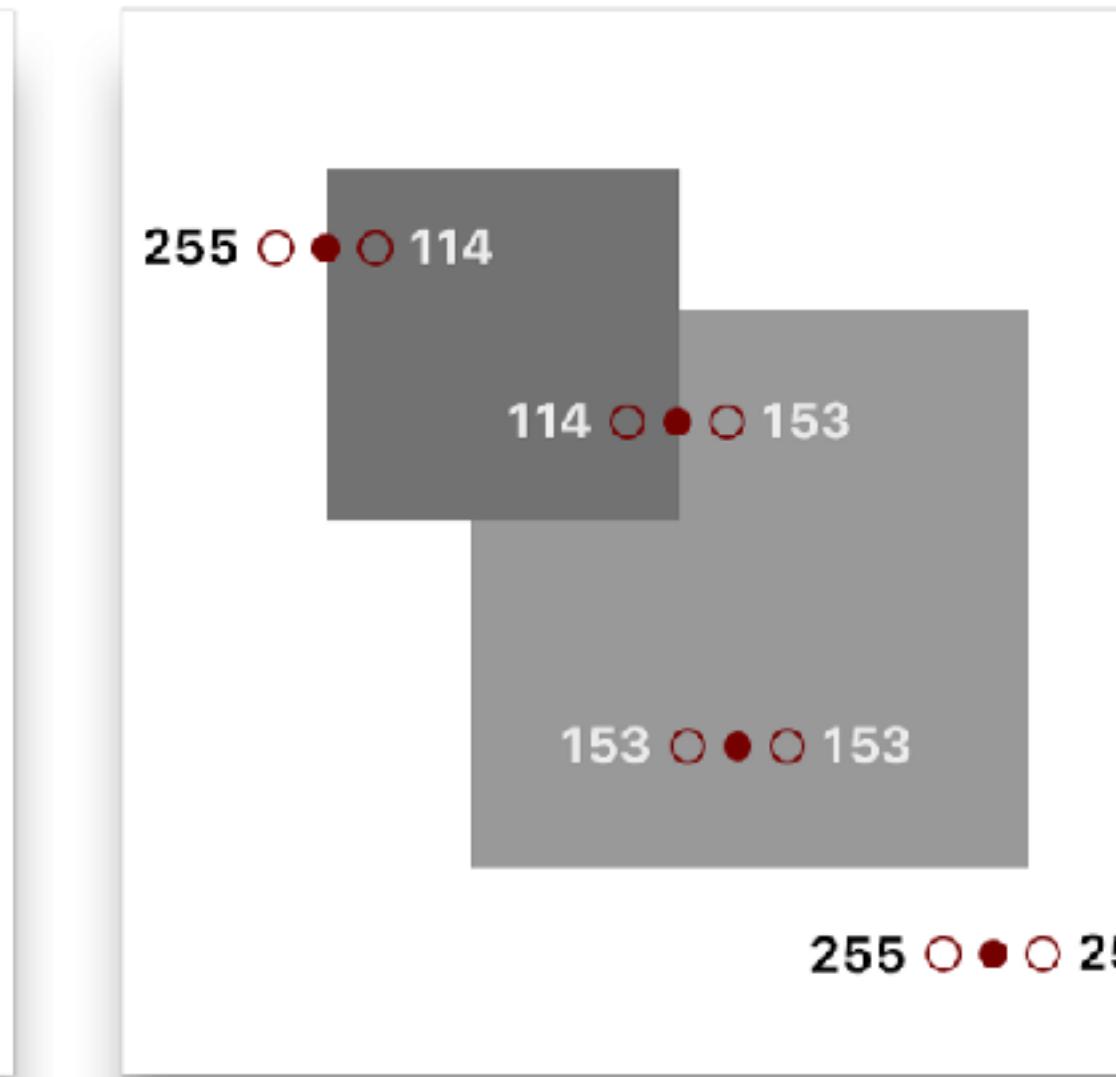
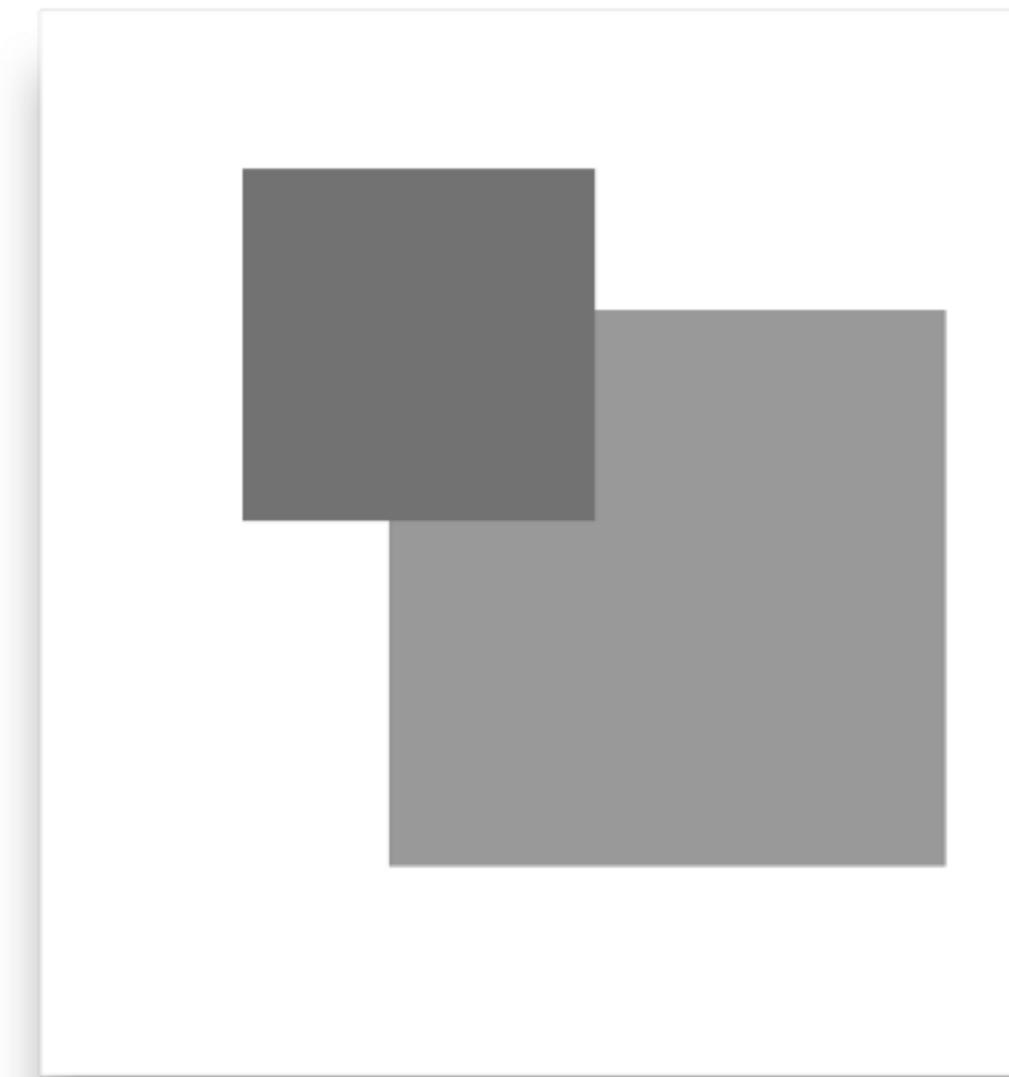
255	255	255	114	114	114	114
255	255	255	x 0	x 0	x 0	x 0
255	255	255	x 1	x 0	x -1	
255	255	255	x 0	x 0	x 0	x 0
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

0	141	141	0	0
0	141	141	0	0
0	141	141	0	0

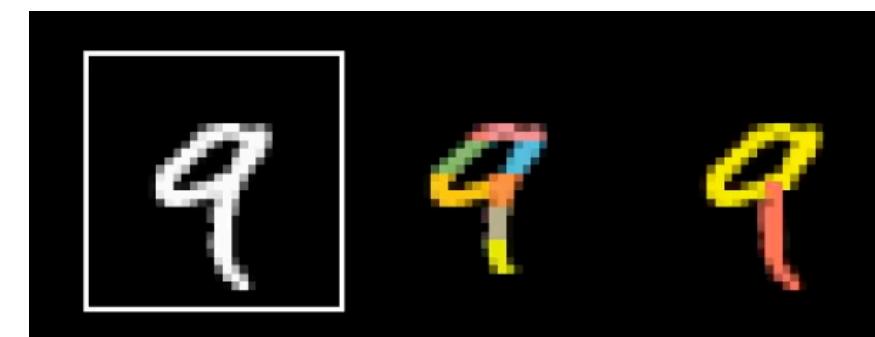
Motivation: Old school

```
kernel = [1  0 -1]
```

```
kernel = [  
    [0  0  0]  
    [1  0 -1]  
    [0  0  0]  
]
```



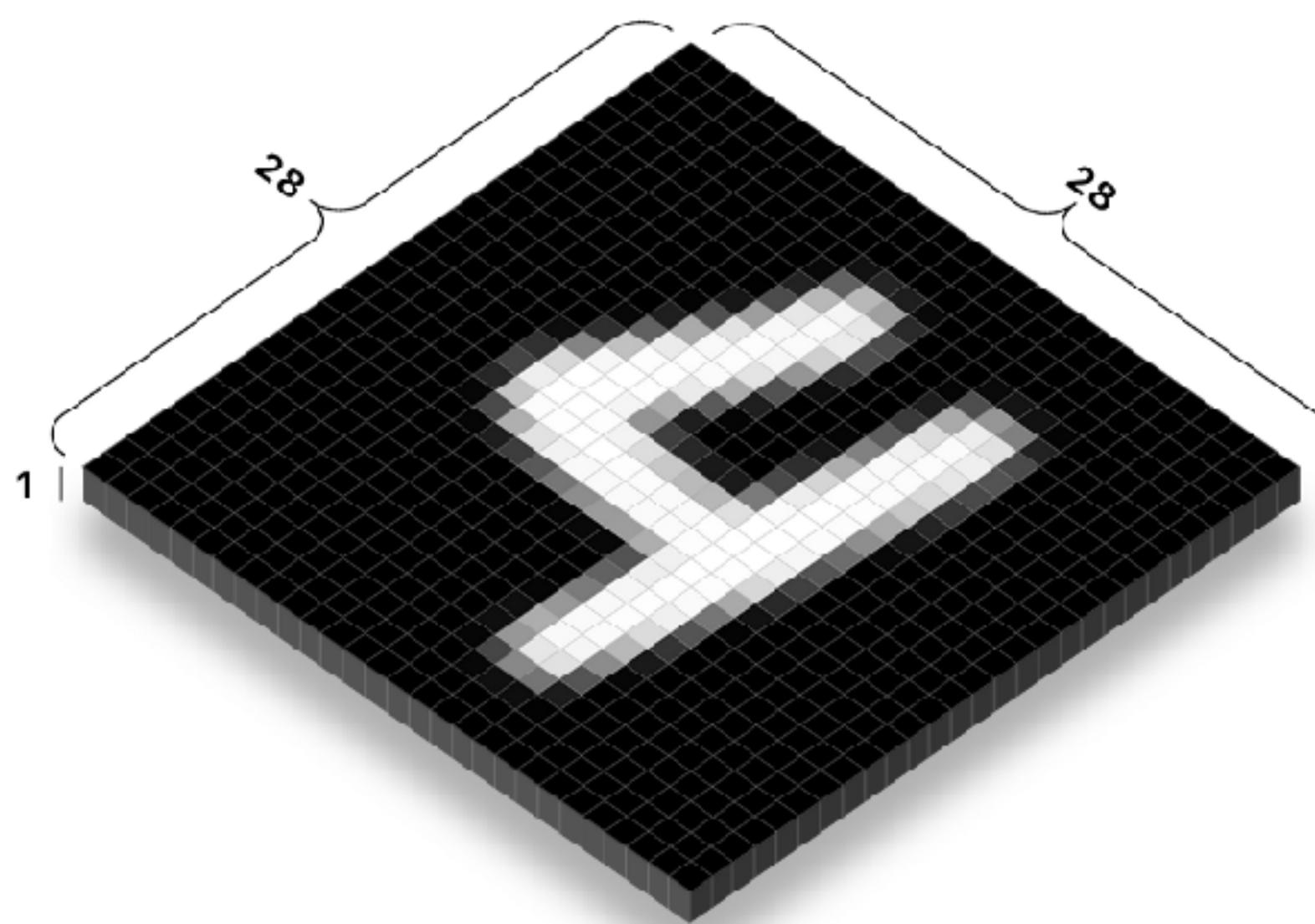
Motivation: New school



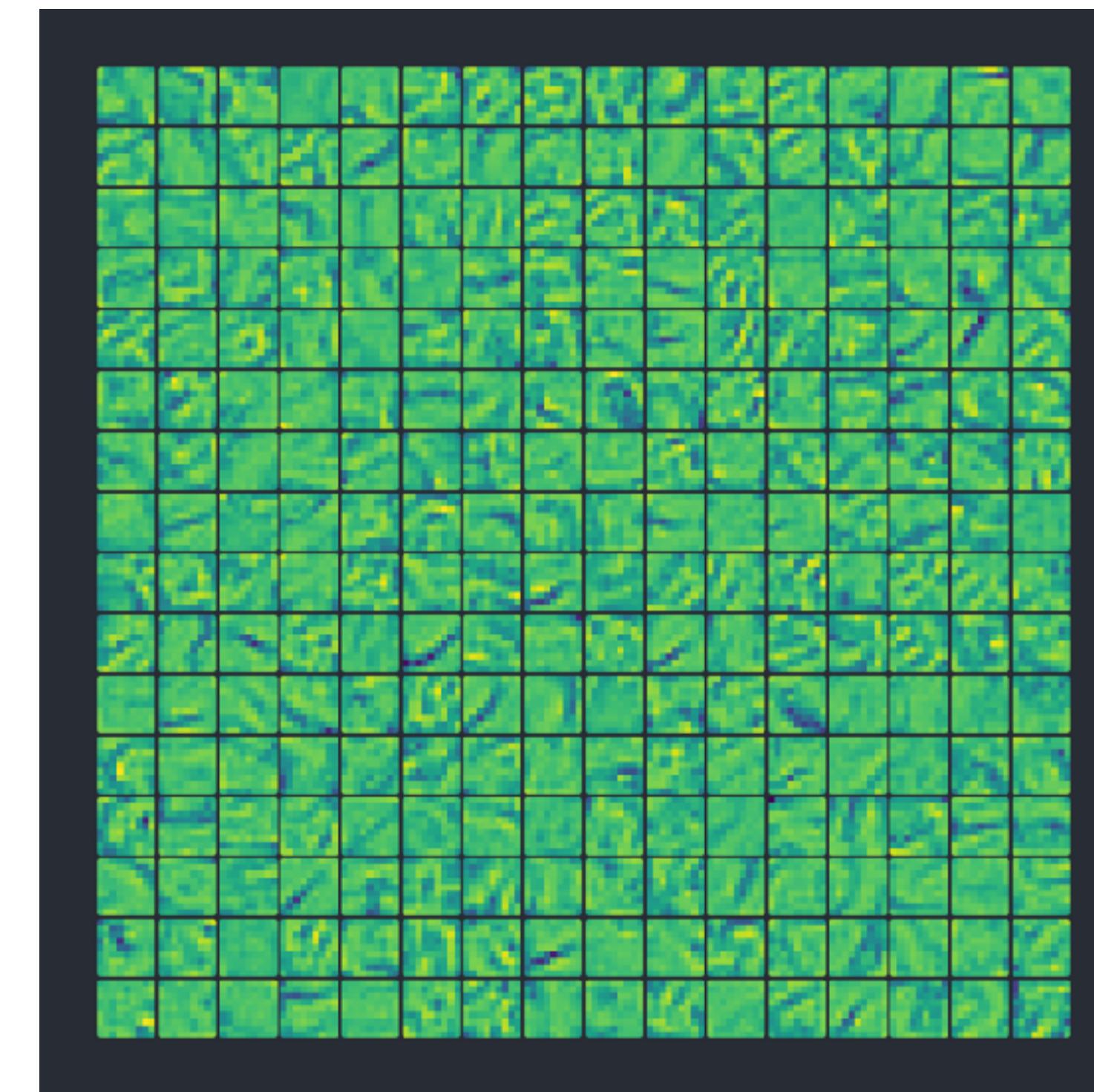
$$\text{9} = \text{o} + \text{l}$$

$$\text{o} = \text{pen} + \text{hand} + \text{ring} + \text{dot} + \text{tail}$$

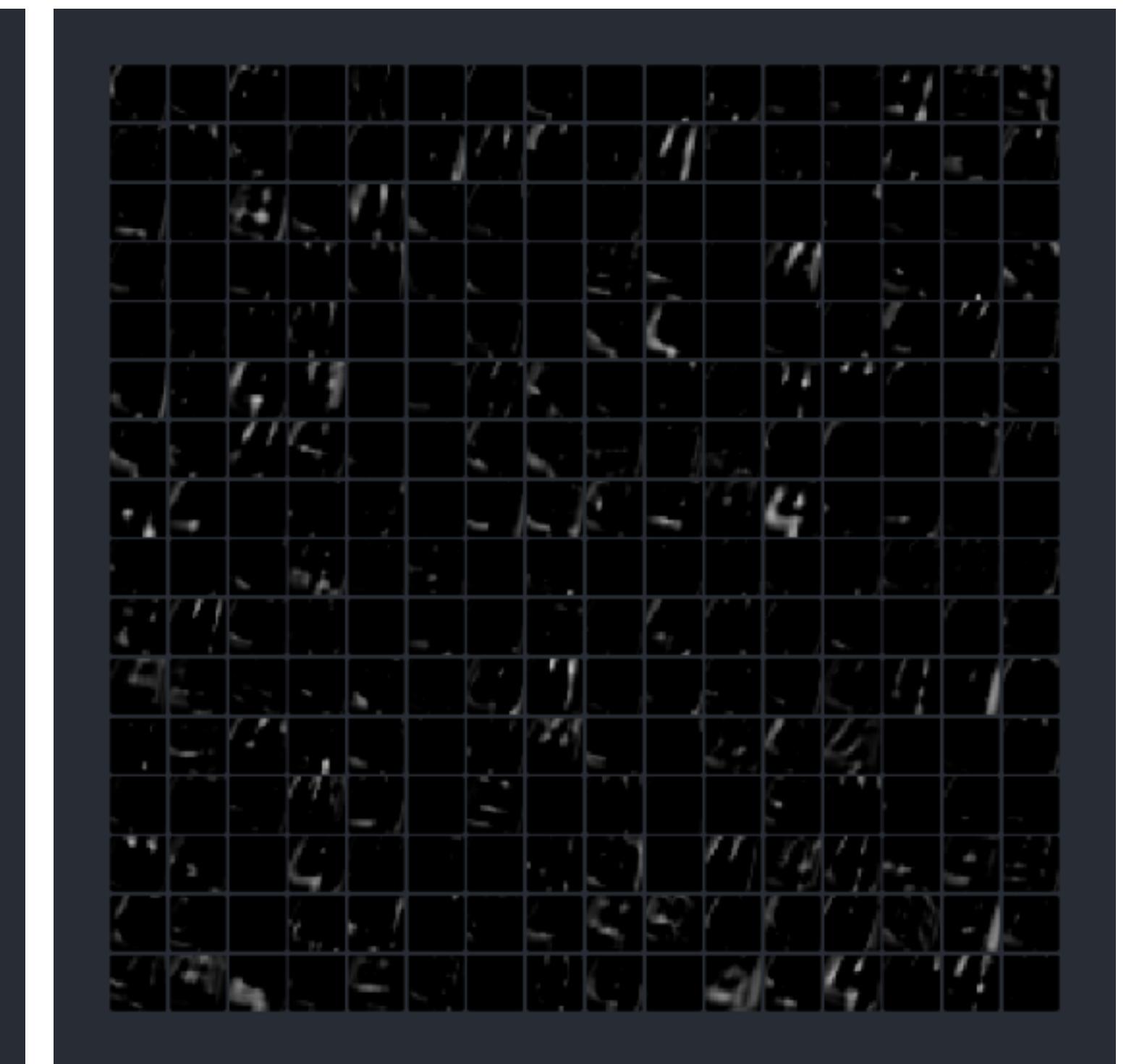
$$\text{l} = \text{hand} + \text{tail} + \text{dot}$$



```
kernel = [
    [ 0.02 -0.01  0.01 -0.05 -0.08 -0.14 -0.16 -0.22 -0.02]
    [ 0.01  0.02  0.03  0.02  0.00 -0.06 -0.14 -0.28  0.03]
    [ 0.03  0.01  0.02  0.01  0.03  0.01 -0.11 -0.22 -0.08]
    [ 0.03 -0.01 -0.02  0.01  0.04  0.07 -0.11 -0.24 -0.05]
    [-0.01 -0.02 -0.02  0.01  0.06  0.12 -0.13 -0.31  0.04]
    [-0.05 -0.02  0.00  0.05  0.08  0.14 -0.17 -0.29  0.08]
    [-0.06  0.02  0.00  0.07  0.07  0.04 -0.18 -0.10  0.05]
    [-0.06  0.01  0.04  0.05  0.03 -0.01 -0.10 -0.07  0.00]
    [-0.04  0.00  0.04  0.05  0.02 -0.04 -0.02 -0.05  0.04]
]
```

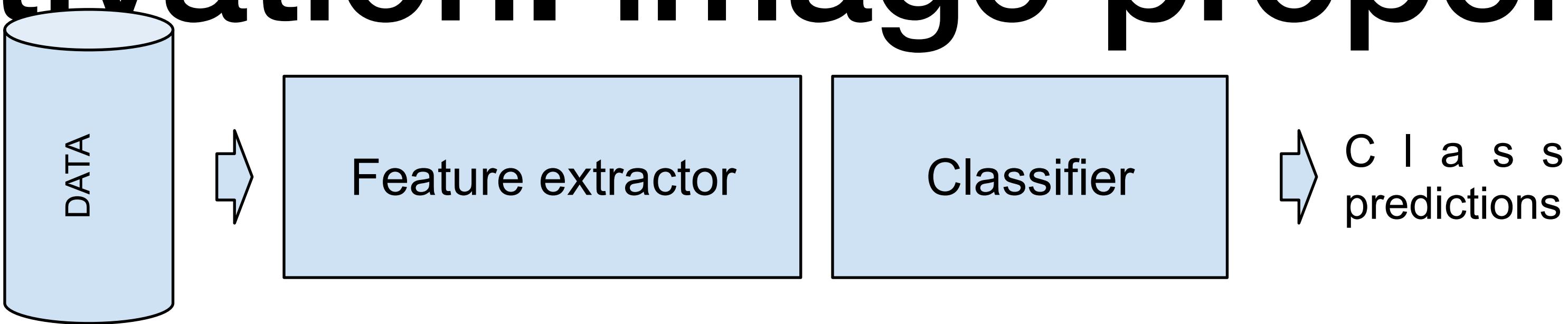


Kernels / filters (after training)

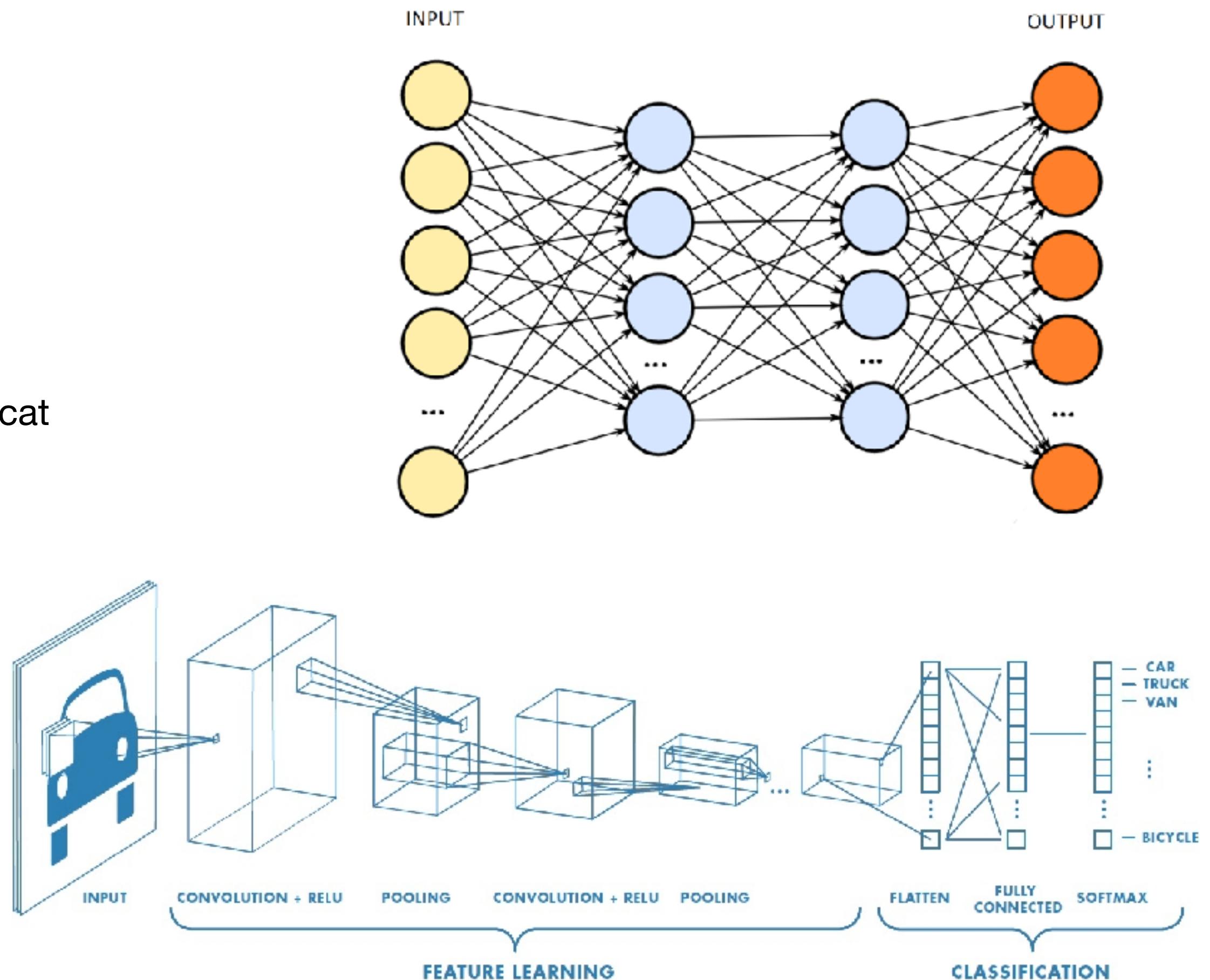


Activations (Conv + ReLU)

Motivation: image properties

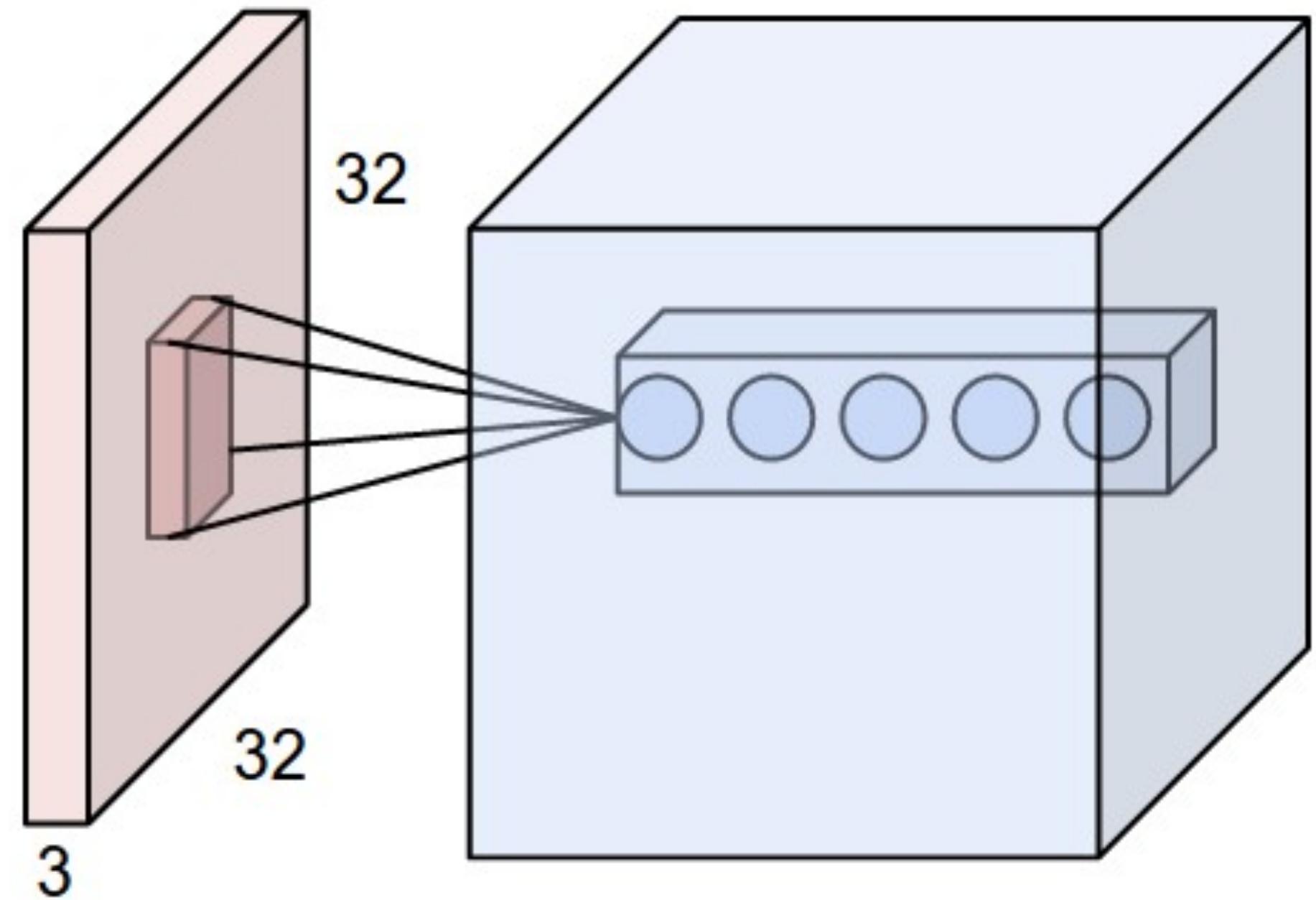


- Neural networks can be powerful **feature extractors** and classifiers
- Feature extraction is important for **images classification**.
- We observe that image classification have some “**properties**”
 - **Spatial structure** - the pixels position in relation to other pixels matters
 - **Translation invariant** - the position of a cat does not change the fact that it is a cat
 - **Large** - images are generally quite large
 - **Rich** - there are a lot of information in images
- Some of these properties makes FC layers **inadequate**
 - Expensive: Too many parameters
 - Do not consider spatial structures
- These properties guides the **design** of CNNs

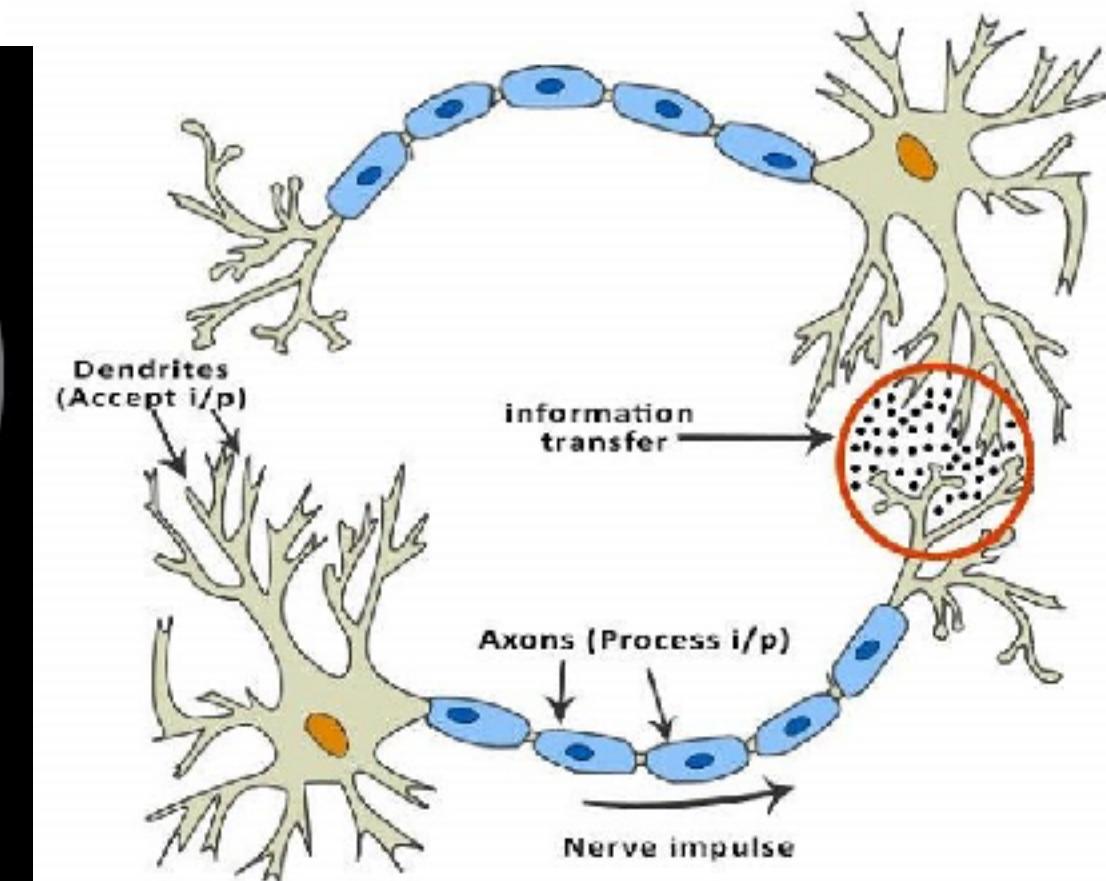
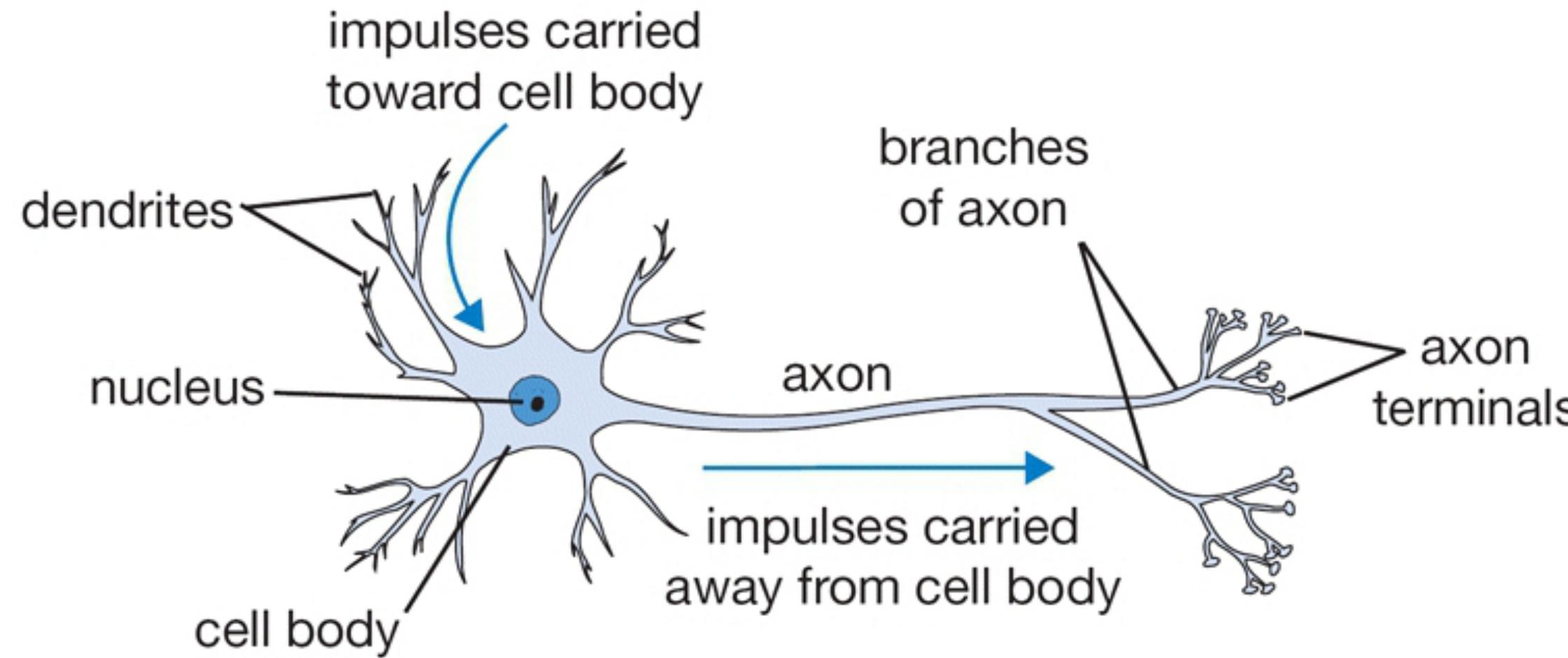


Motivation: image properties

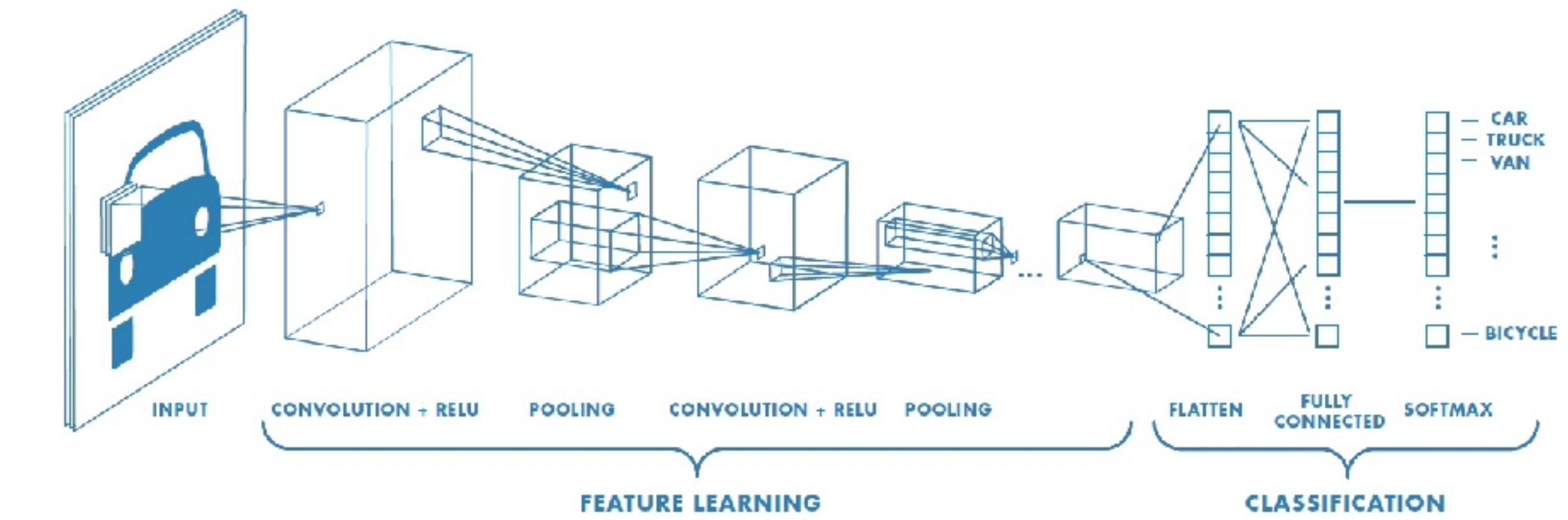
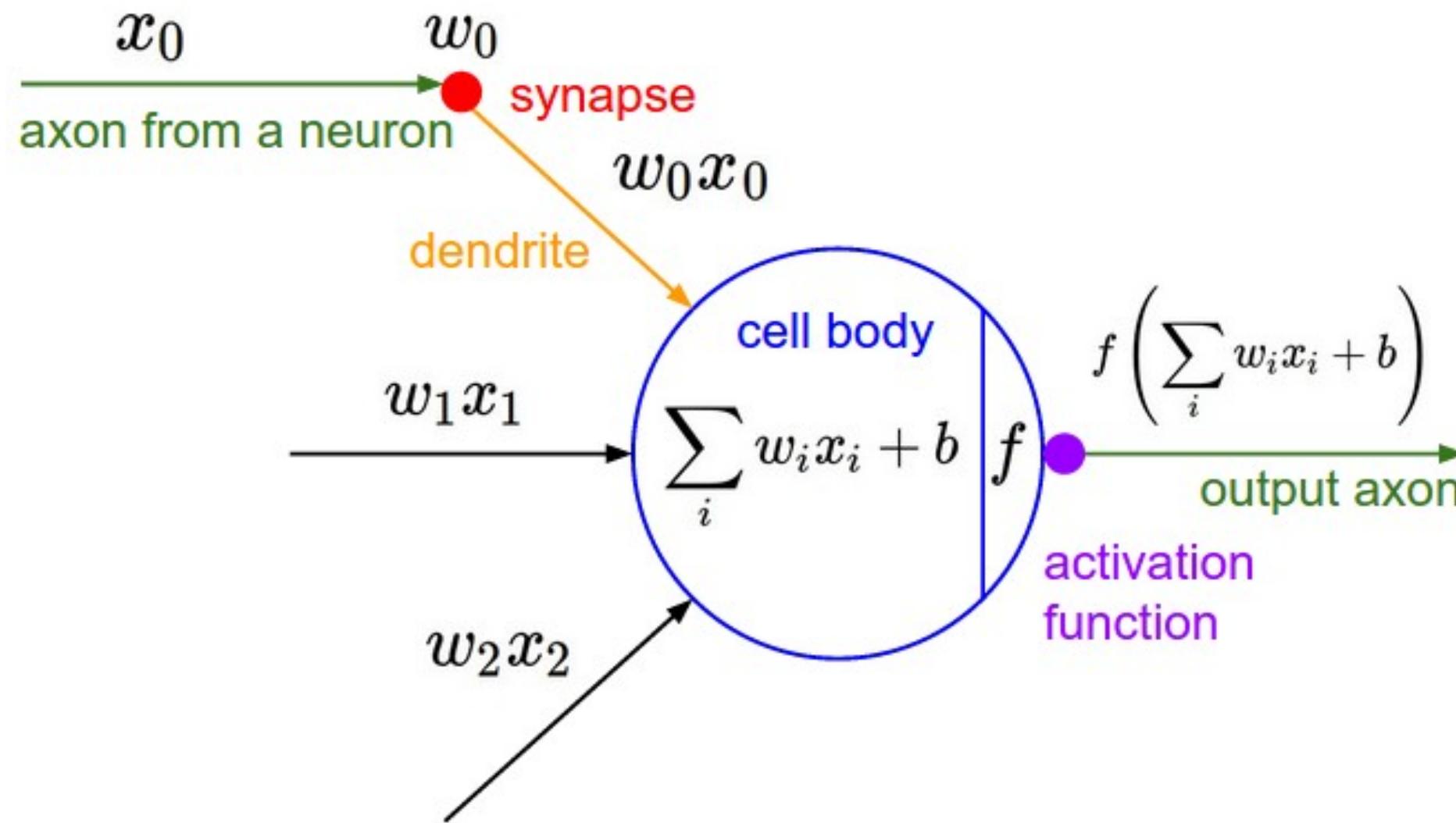
- **Spatial structure:** Work on image data directly and produce maps of where features are in a image
 - CNNs: Work on volumes (tensors)
- **Translation invariant:** Look for the same features everywhere
 - CNNs: Use the same filter that looks for the same feature at all spatial positions
- **Large:** Exploit the translation invariance to reduce the number of parameters
 - CNNs: The filters looking for the same features naturally share weights, which reduces the parameters needed
- **Rich:**
 - Build up **higher** level features from low level features
 - CNNs: Multiple layers
 - Look for **more** than one type of feature in each layers
 - CNNs: Multiple filters per layer



Motivation: Biological vs. Artificial

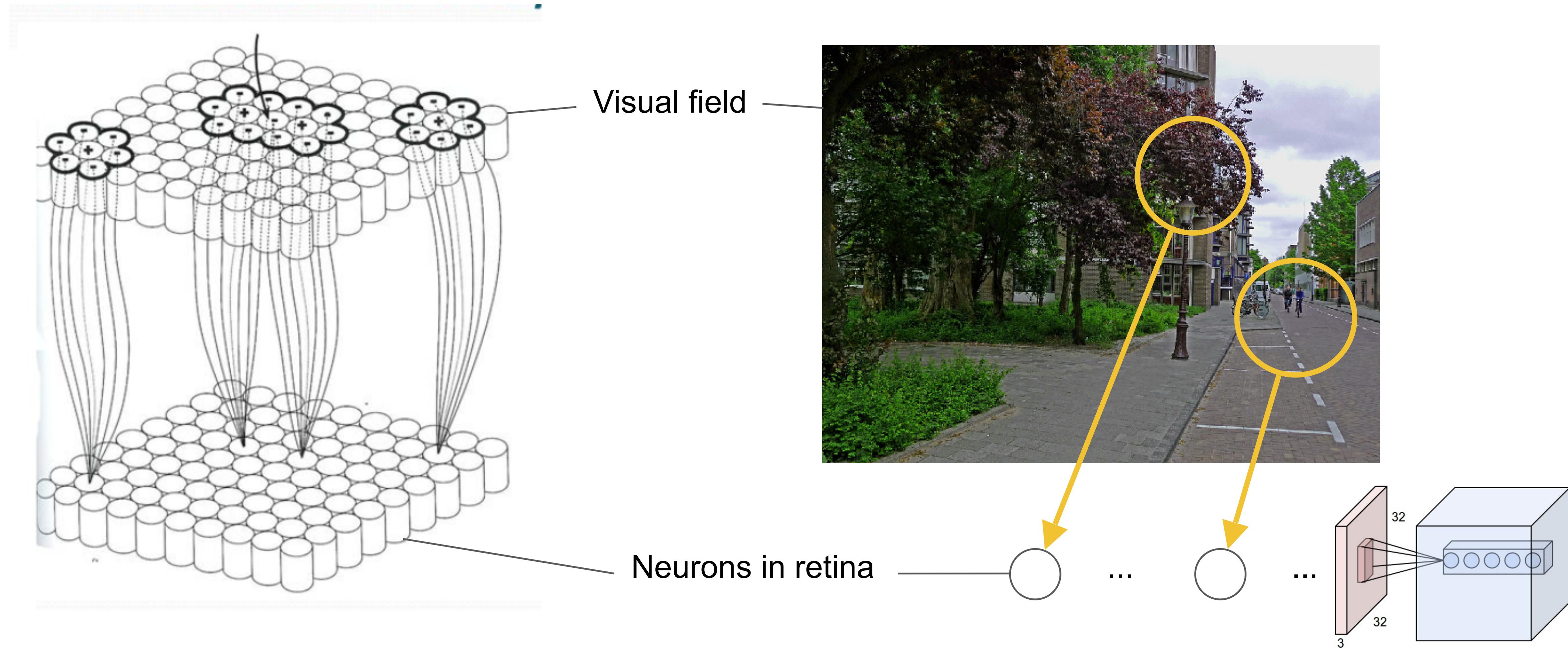


Human Visual system



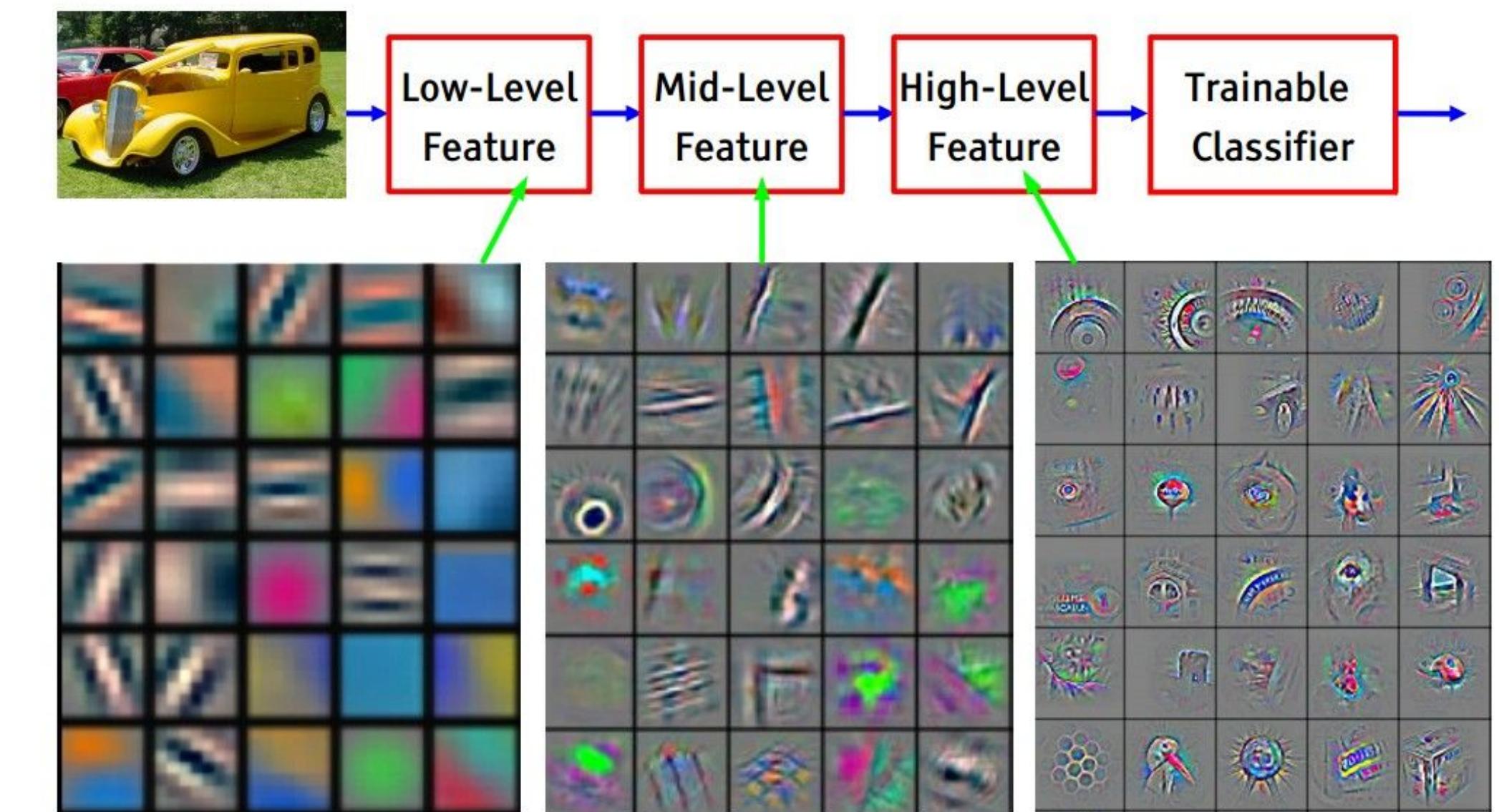
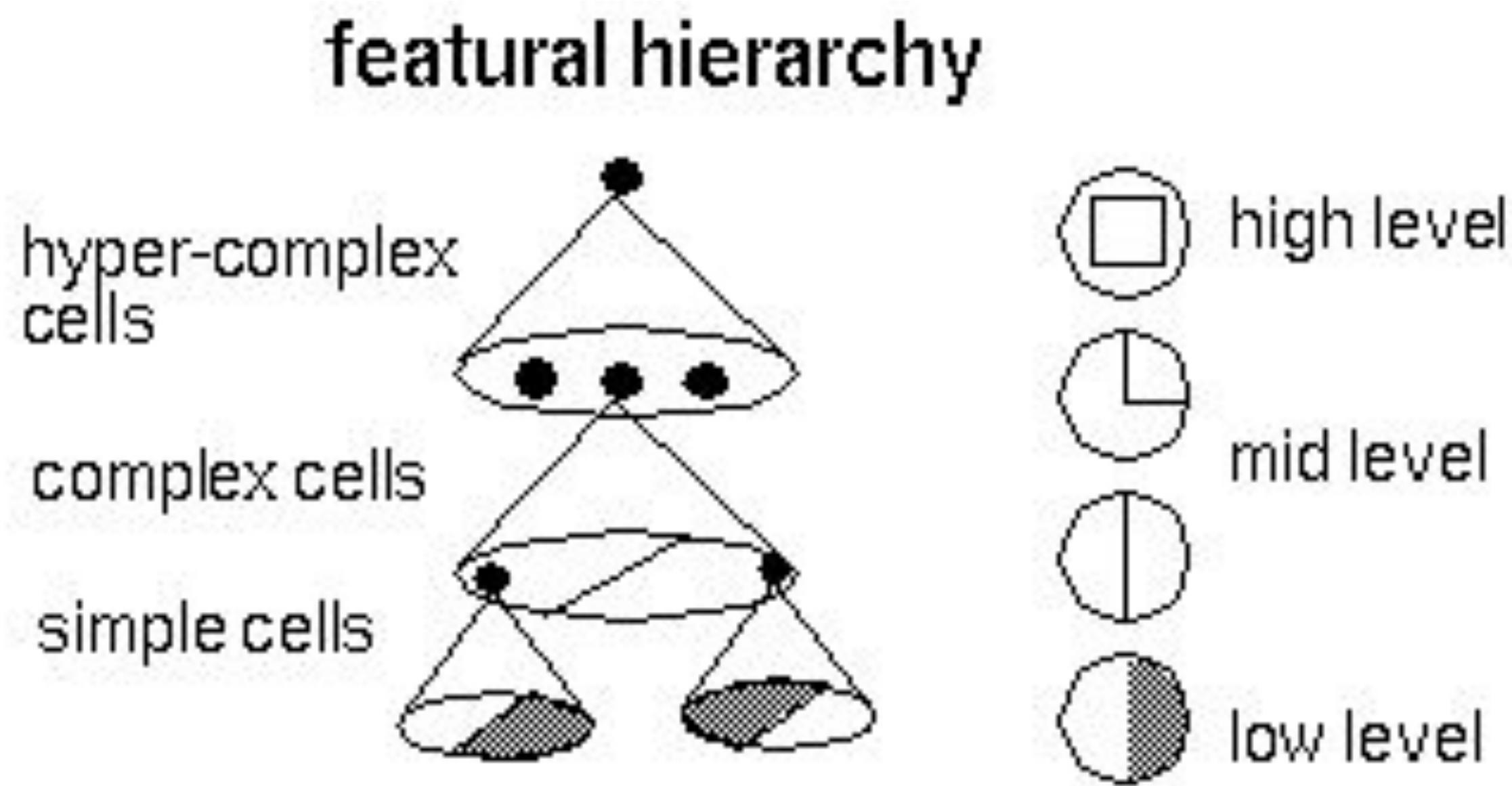
Motivation: Biological Receptive Fields

- Specific neurons in the retina cares about only a small part of the visual field



Motivation: Biological Feature Hierarchies

- Neurons later in the processing “pipeline” are able to extract more complex patterns



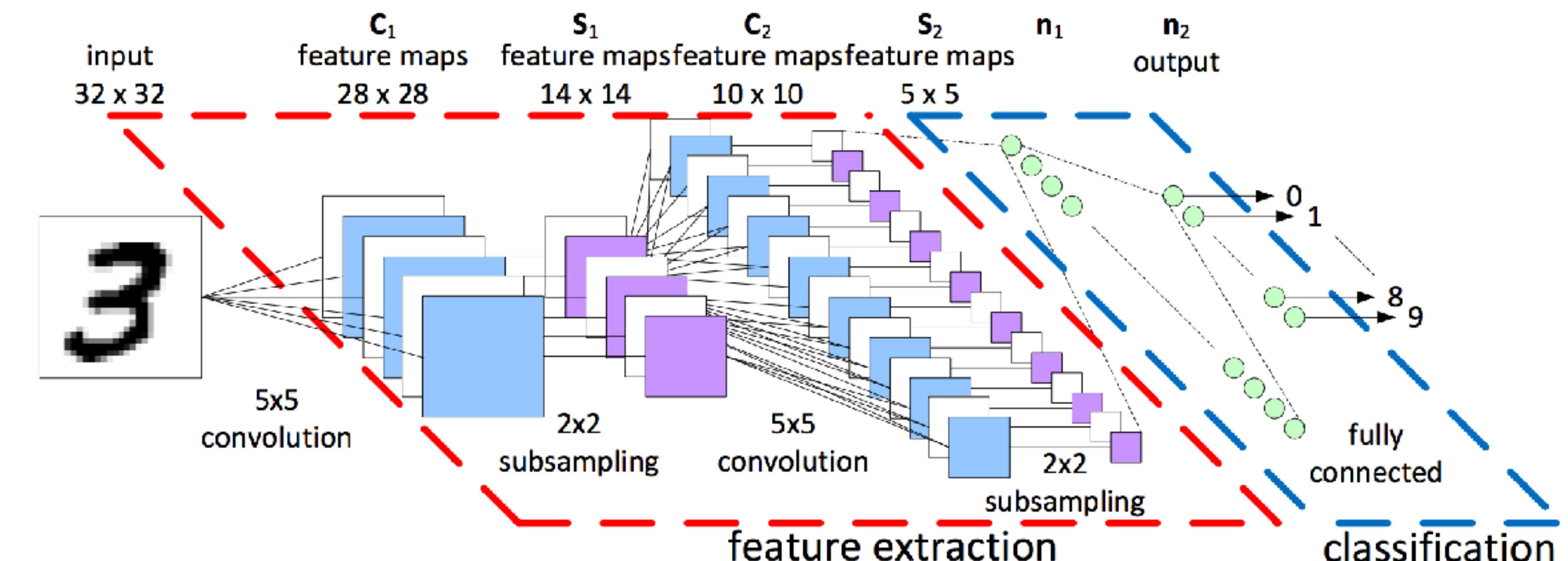
CNNs & ImClass

- **CNNs** (Spatial structure): Feature learning

- convolutional layer
 - local receptive fields
 - shared weights
- pooling layer

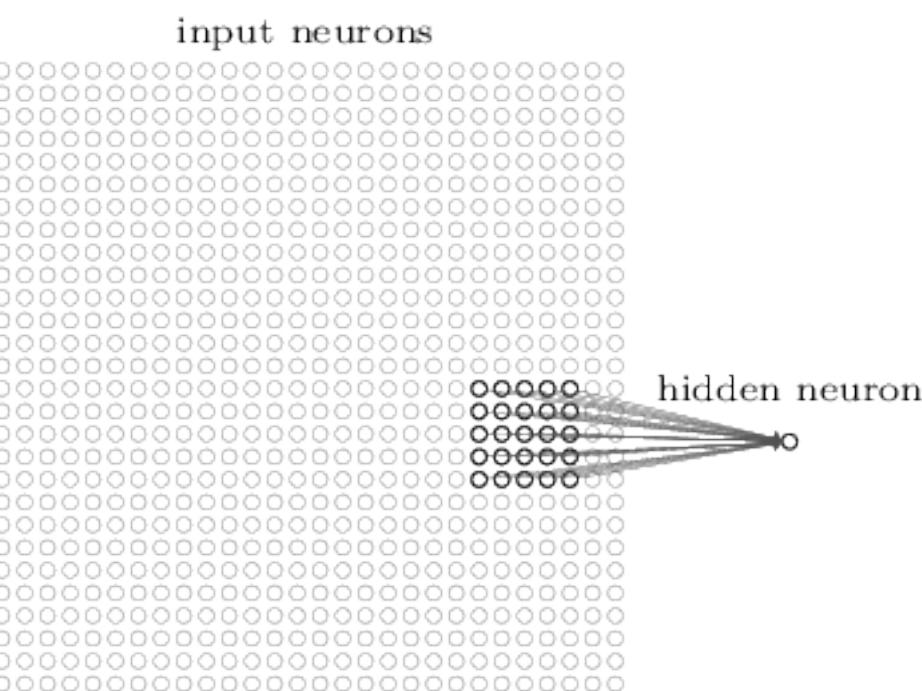
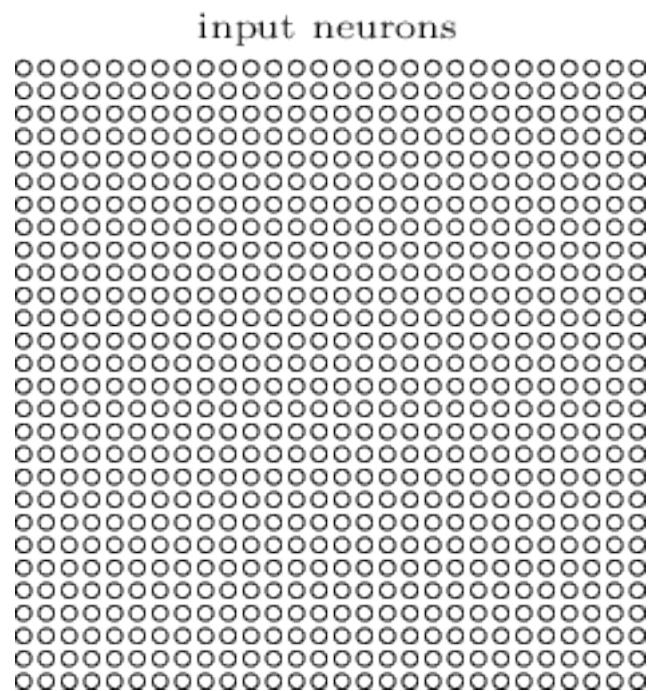
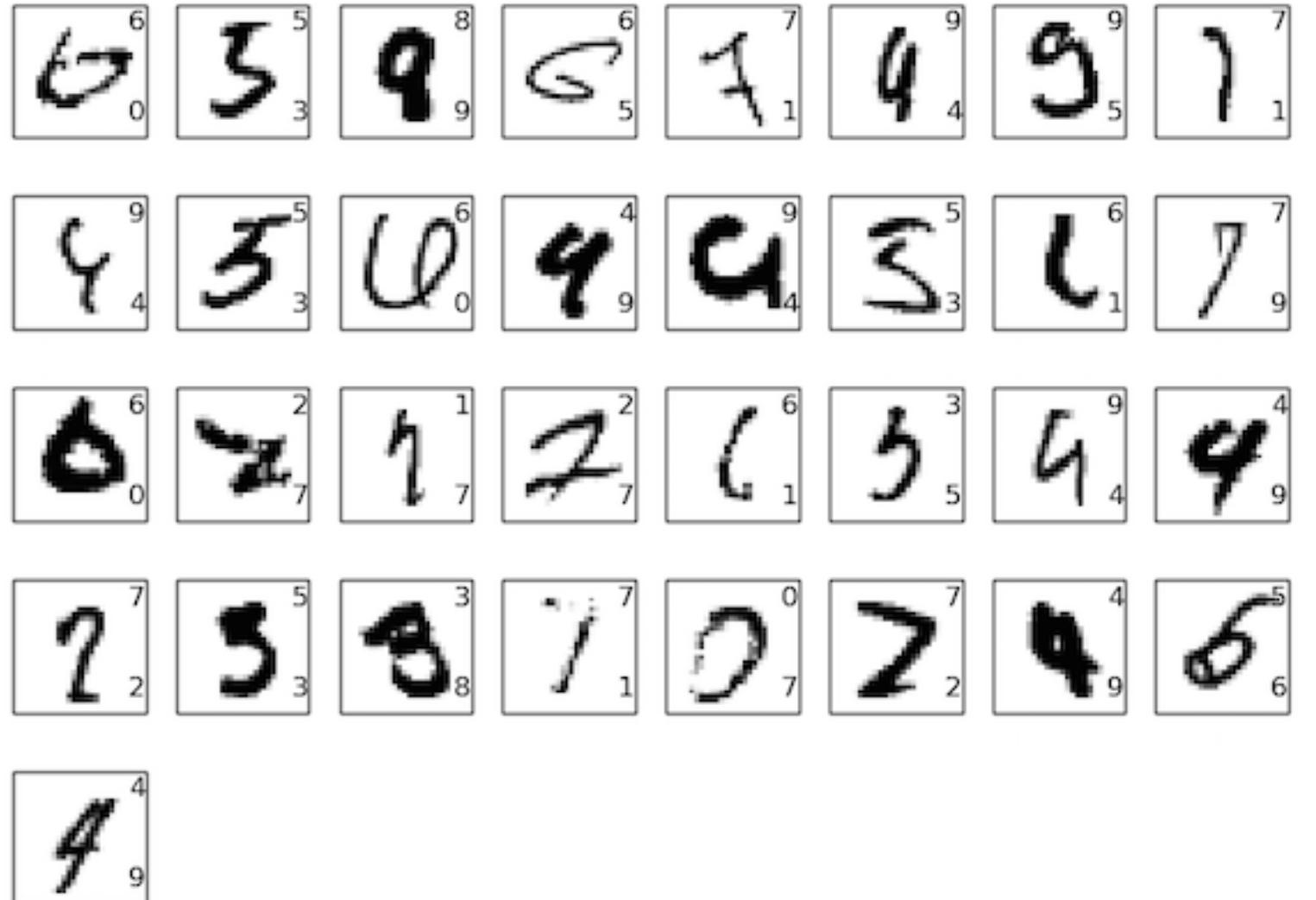
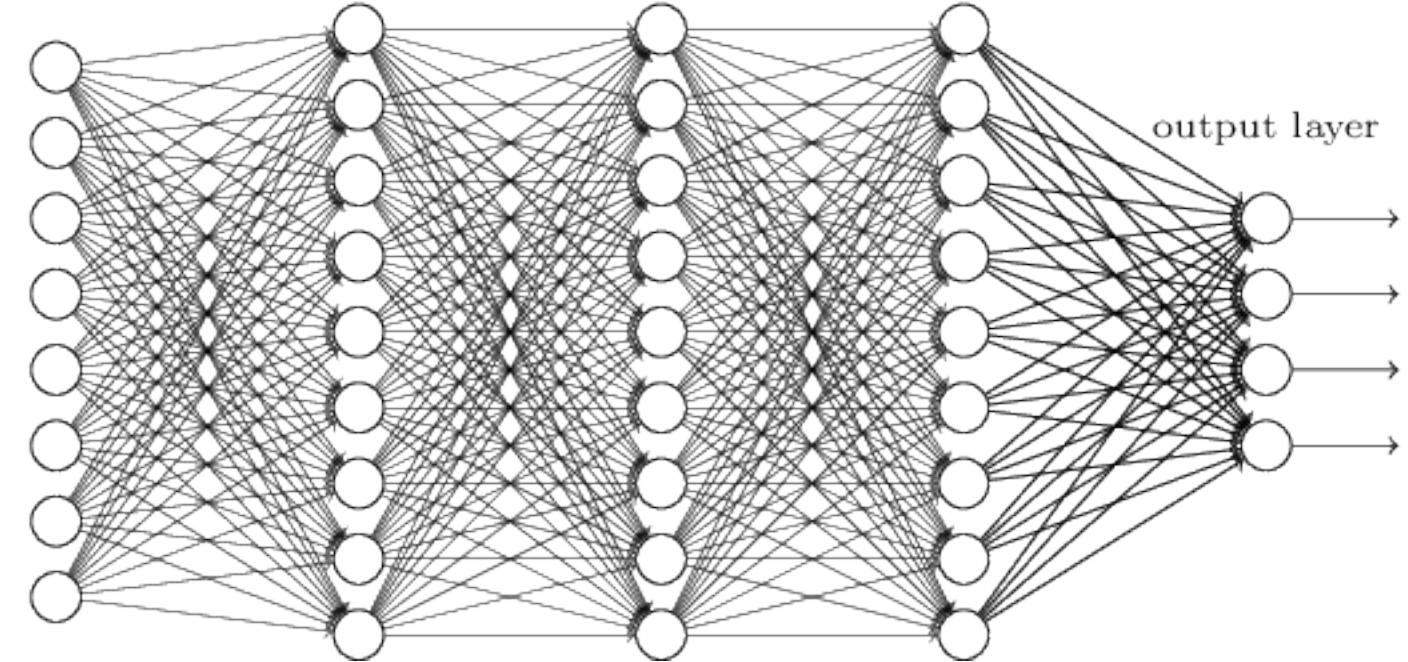
- **Image Classification:**

- Feature extraction / learning
- Classification



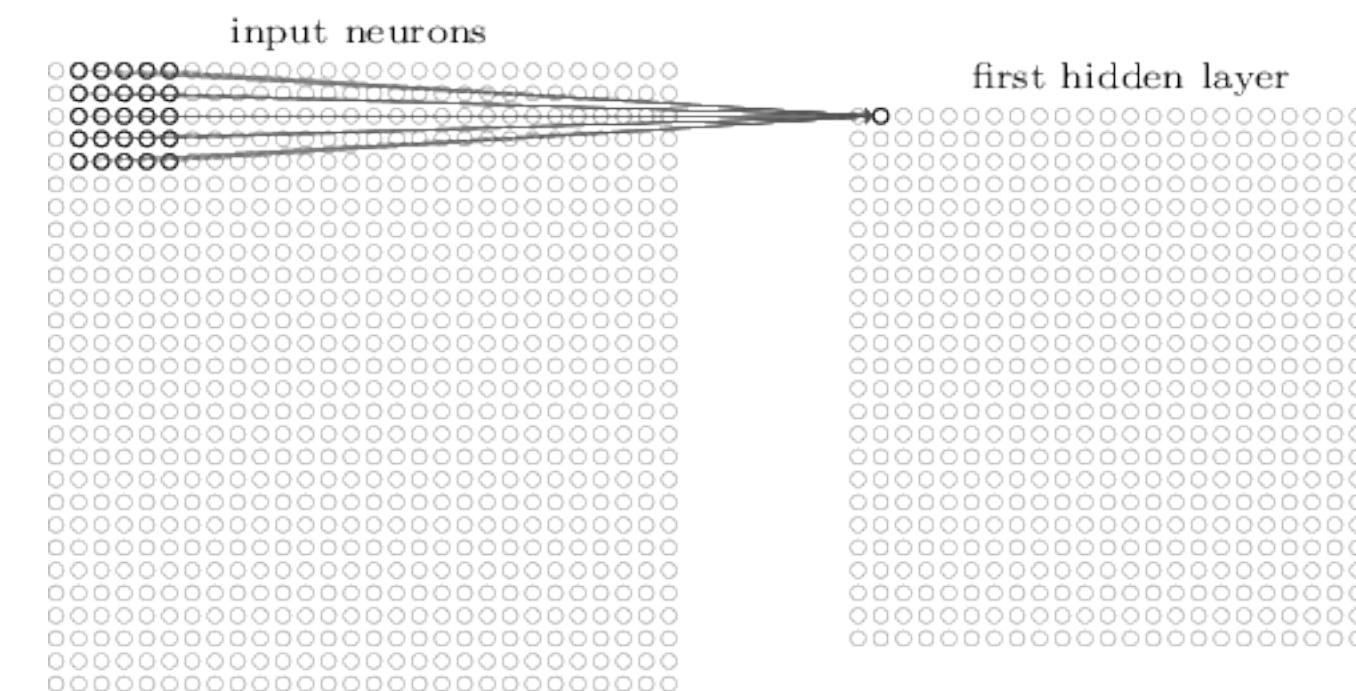
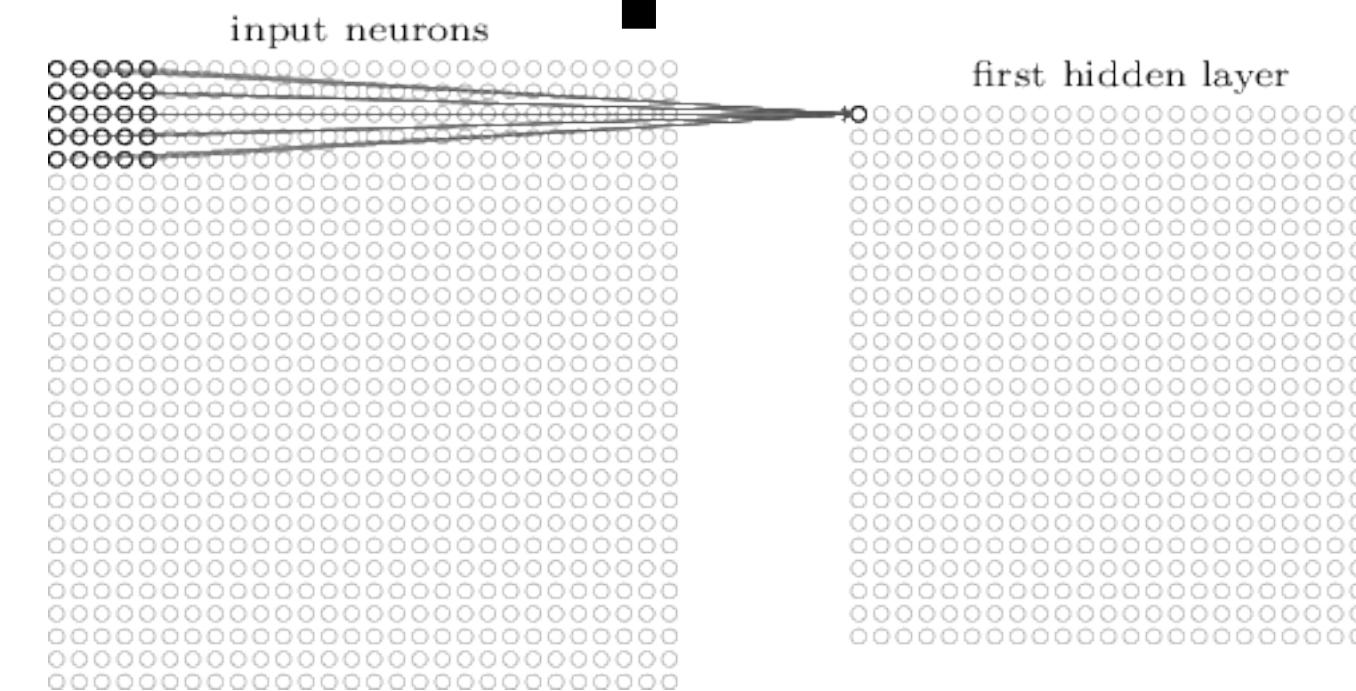
CNNs: Basic concepts

input layer hidden layer 1 hidden layer 2 hidden layer 3



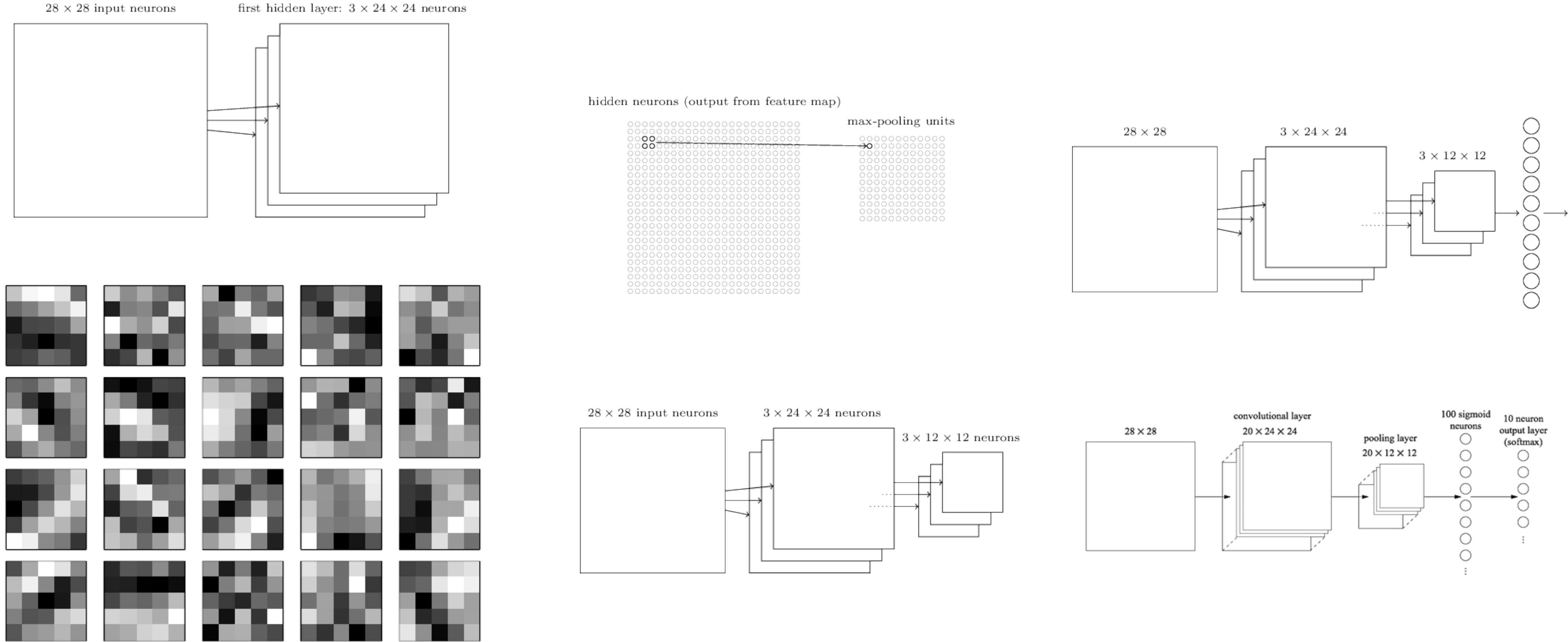
$$f\left(b + \sum_{l=0}^4 \sum_{m=0}^4 w_{l,m} a_{j+l,k+m} \right)$$

$$a^1 = f(b + w * a^0)$$



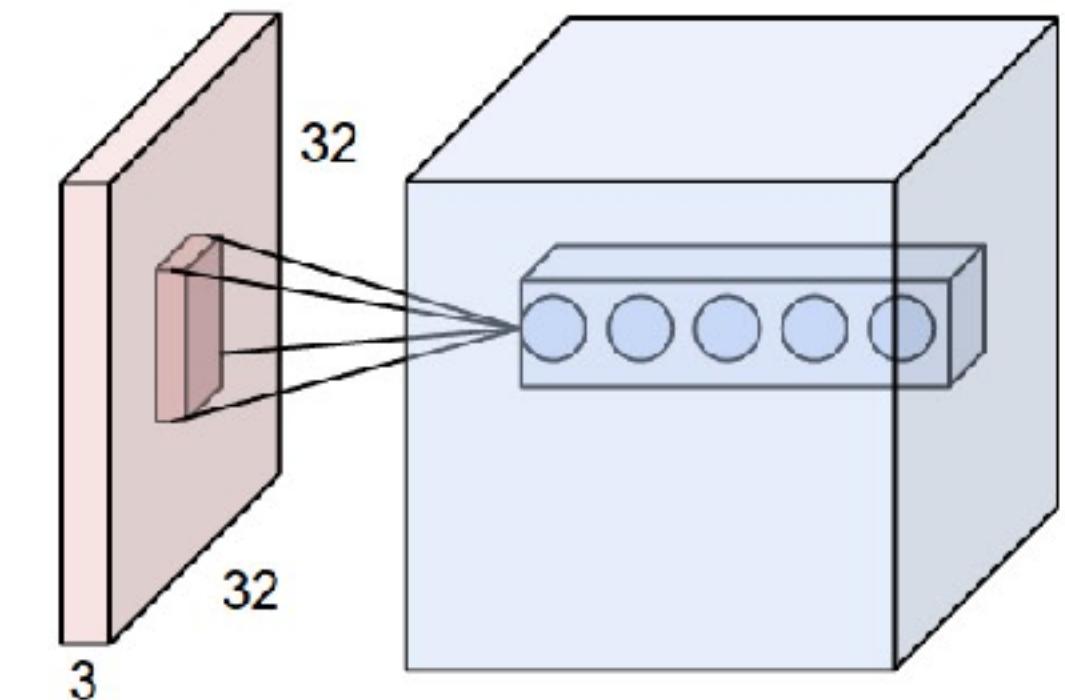
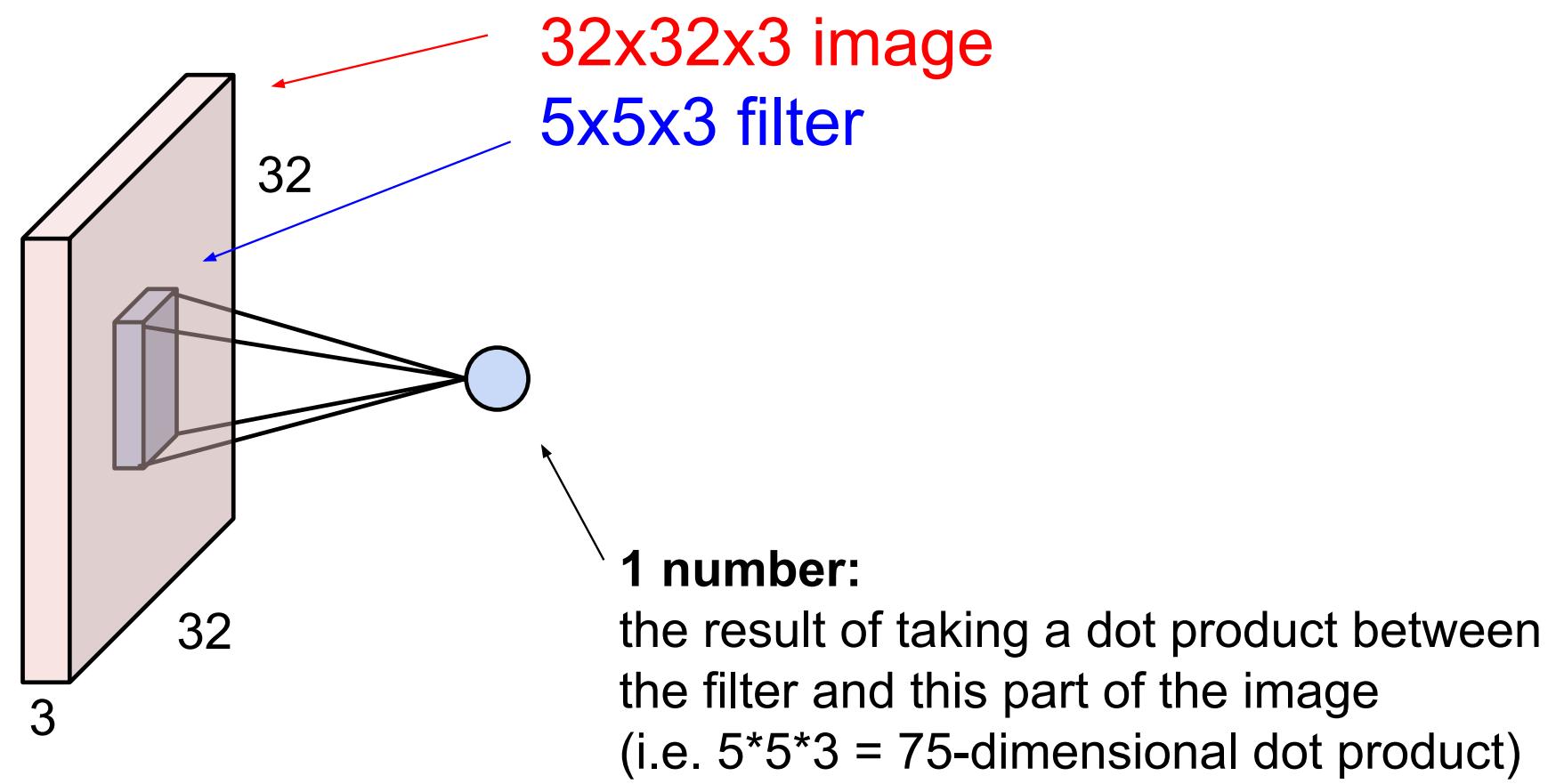
- **CNNs (Spatial structure)**
 - local receptive fields
 - shared weights
 - pooling

CNNs: Basic concepts (2)

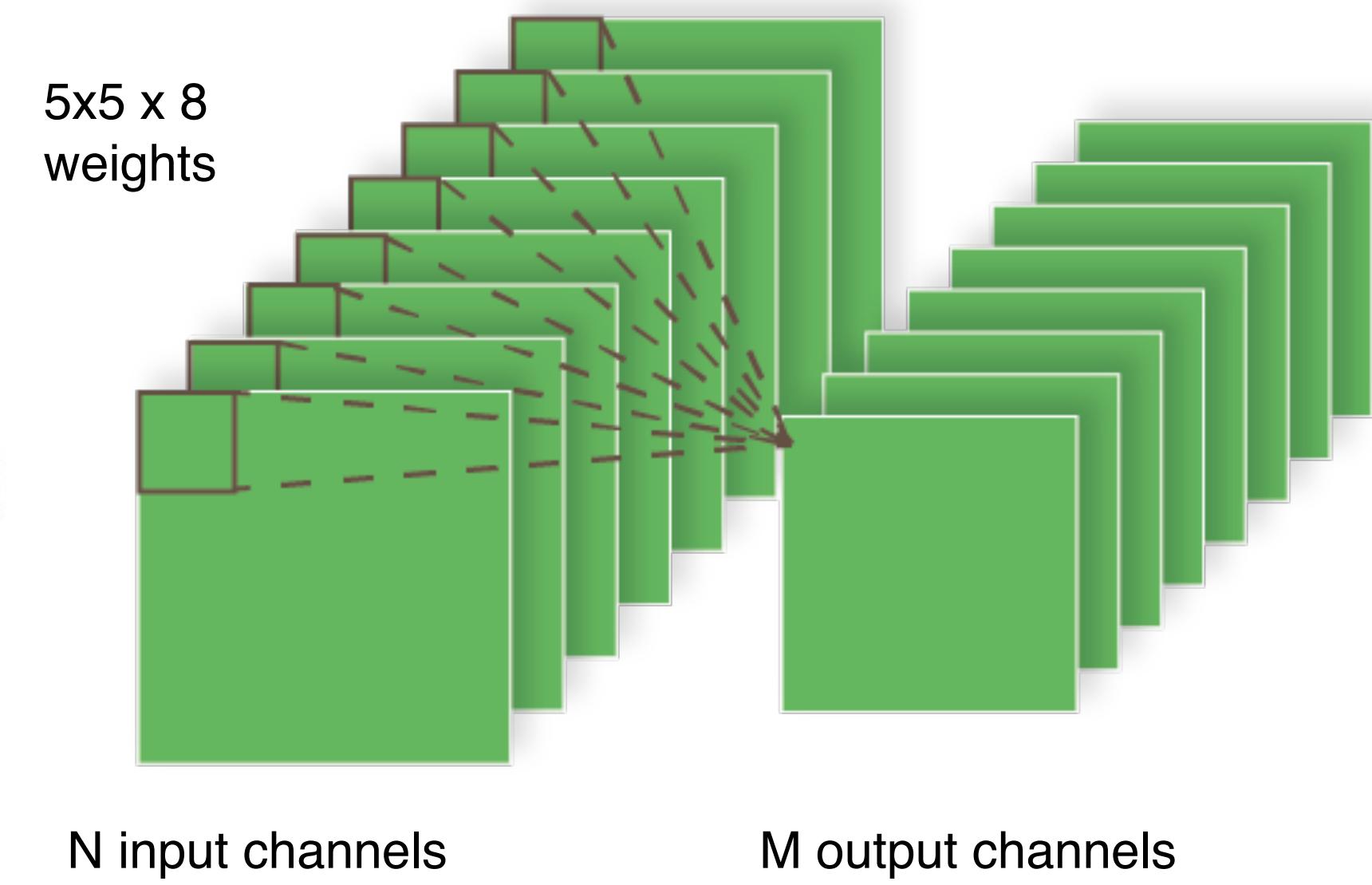
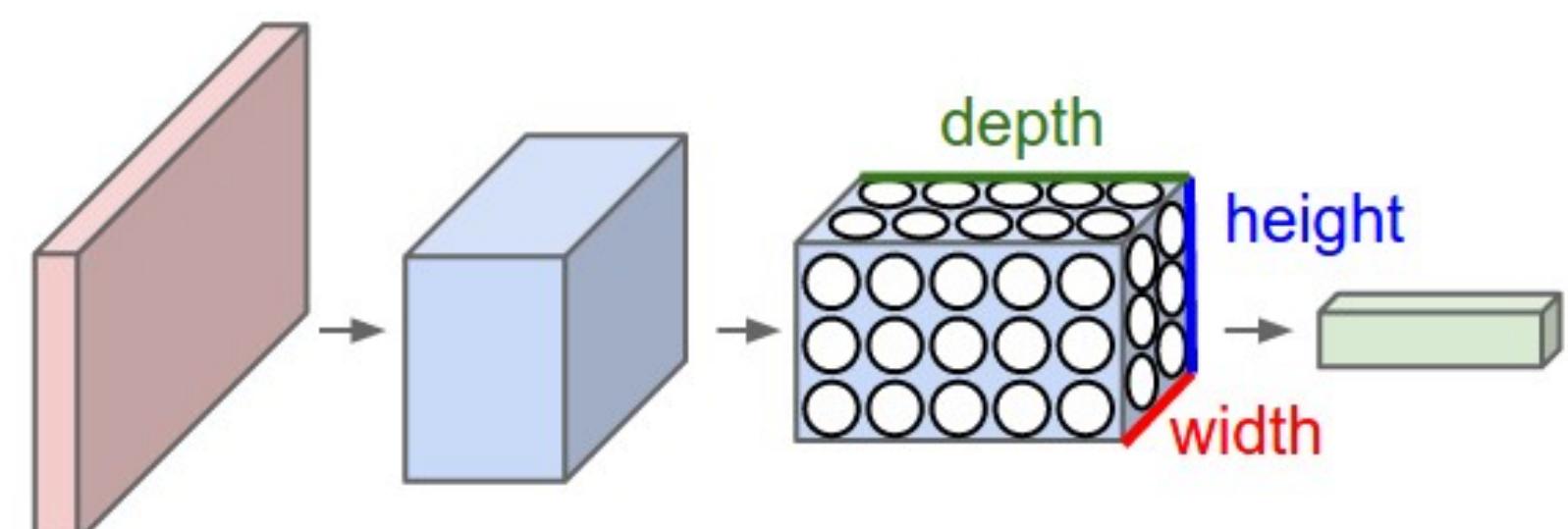


Convolution

- Filter kernels
 - Small spatial size
 - Same depth
- The filter convolves over each location
- One set of parameters per filter
 - Looks for the same patterns over all spatial locations
- A form of dot product
- Result called **feature maps** (response map)



Input channels (RGB, features, ...) are combined



All filters have $\#w = F \times F \times D_{in}$ weights (+ 1 bias)

Total number of weights in one layer is $\#w \times K$

Convolution: Filters

All filters have $F \times F \times D_{in} + 1$ weights

So the total number of parameters in one layer is $F \times F \times D_{in} \times K$

$$W_{out} = \frac{W_{in}-F+2P}{S} + 1$$

$$H_{out} = \frac{H_{in}-F+2P}{S} + 1$$

$$D_{out} = K$$

K number of filters

F spatial size of filter

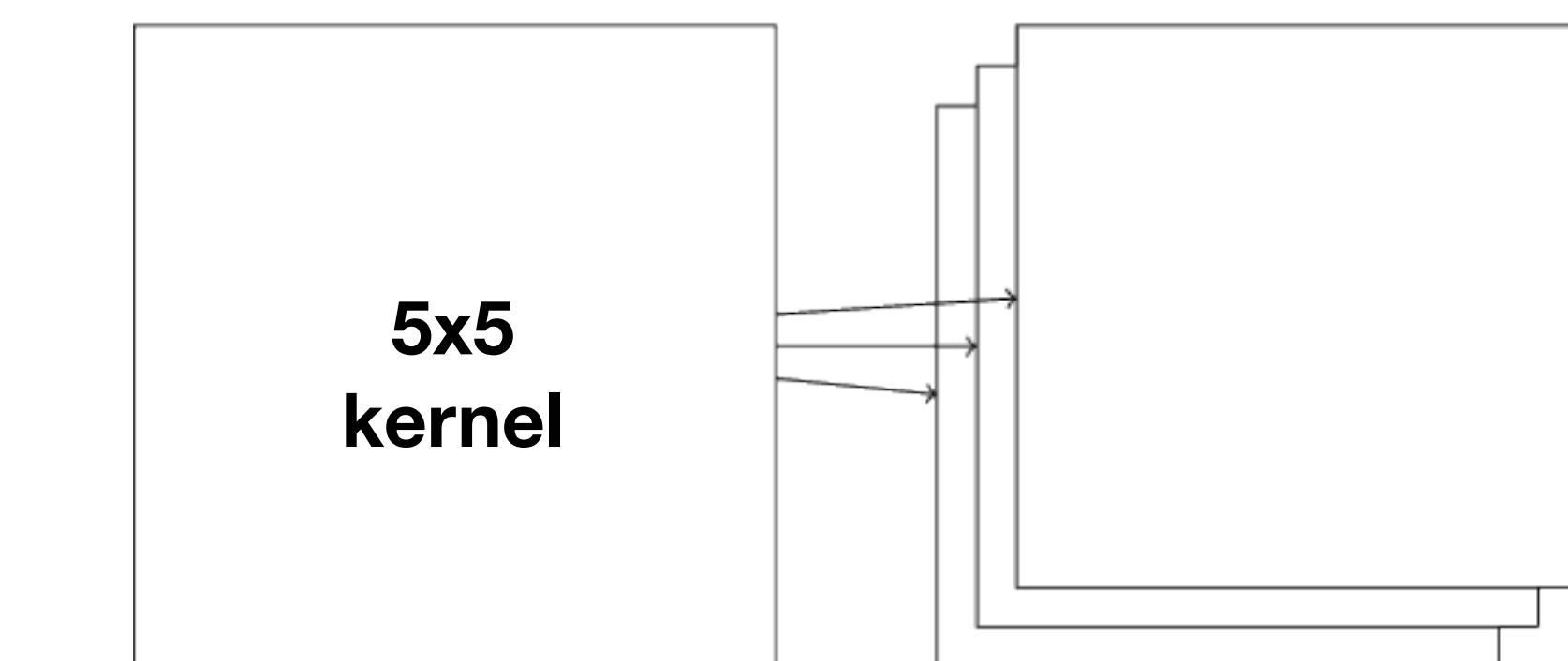
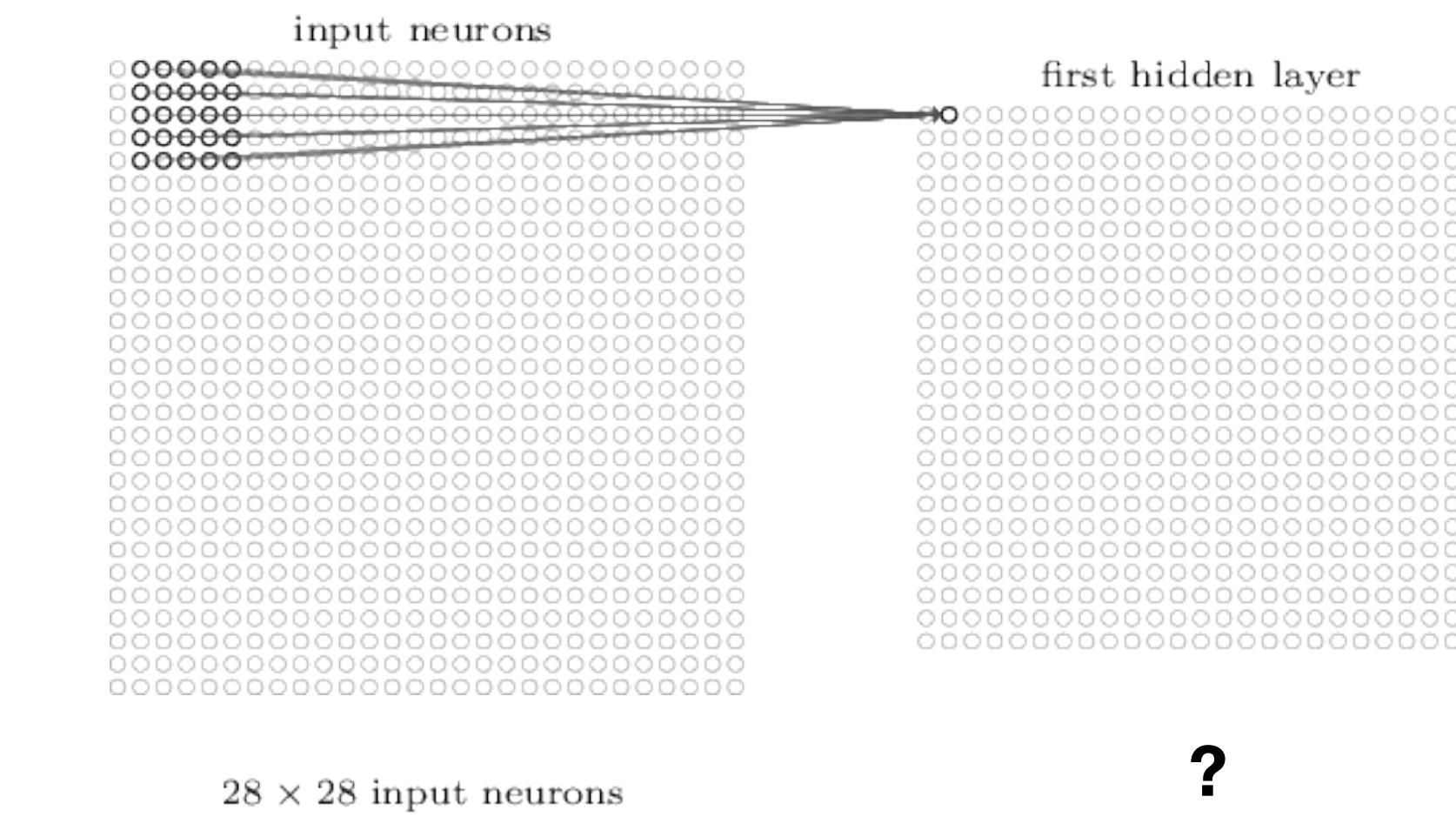
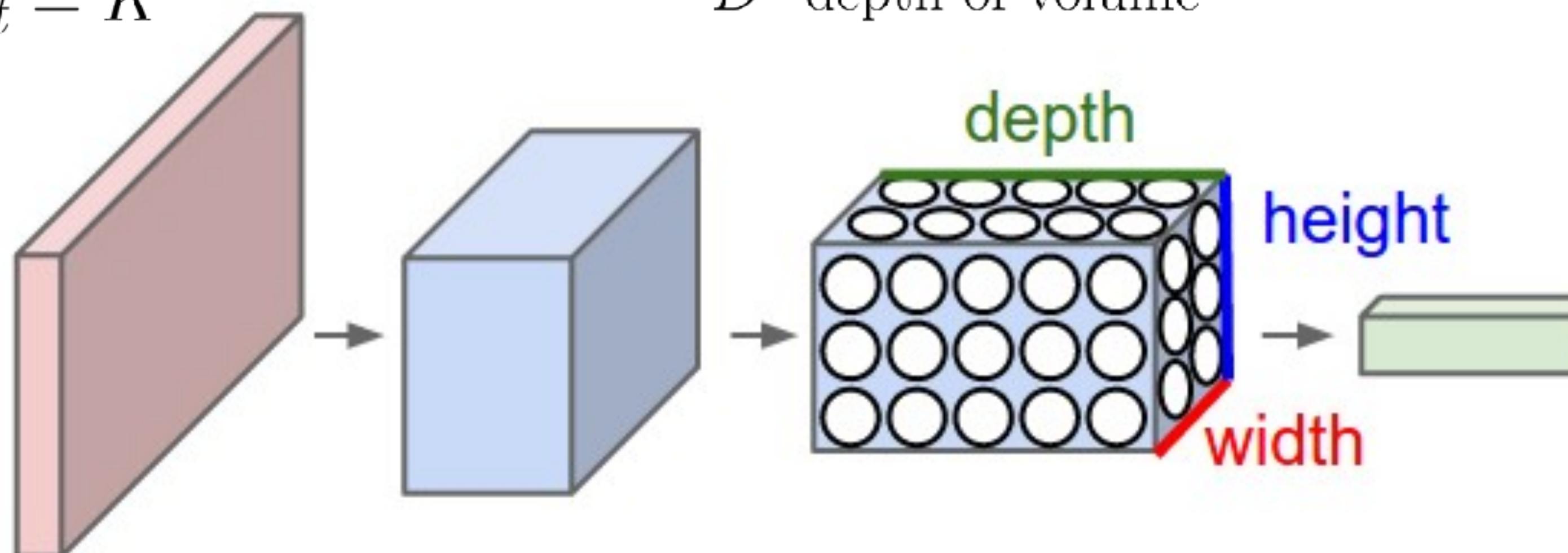
S stride

P amount of padding

W width of volume

H height of volume

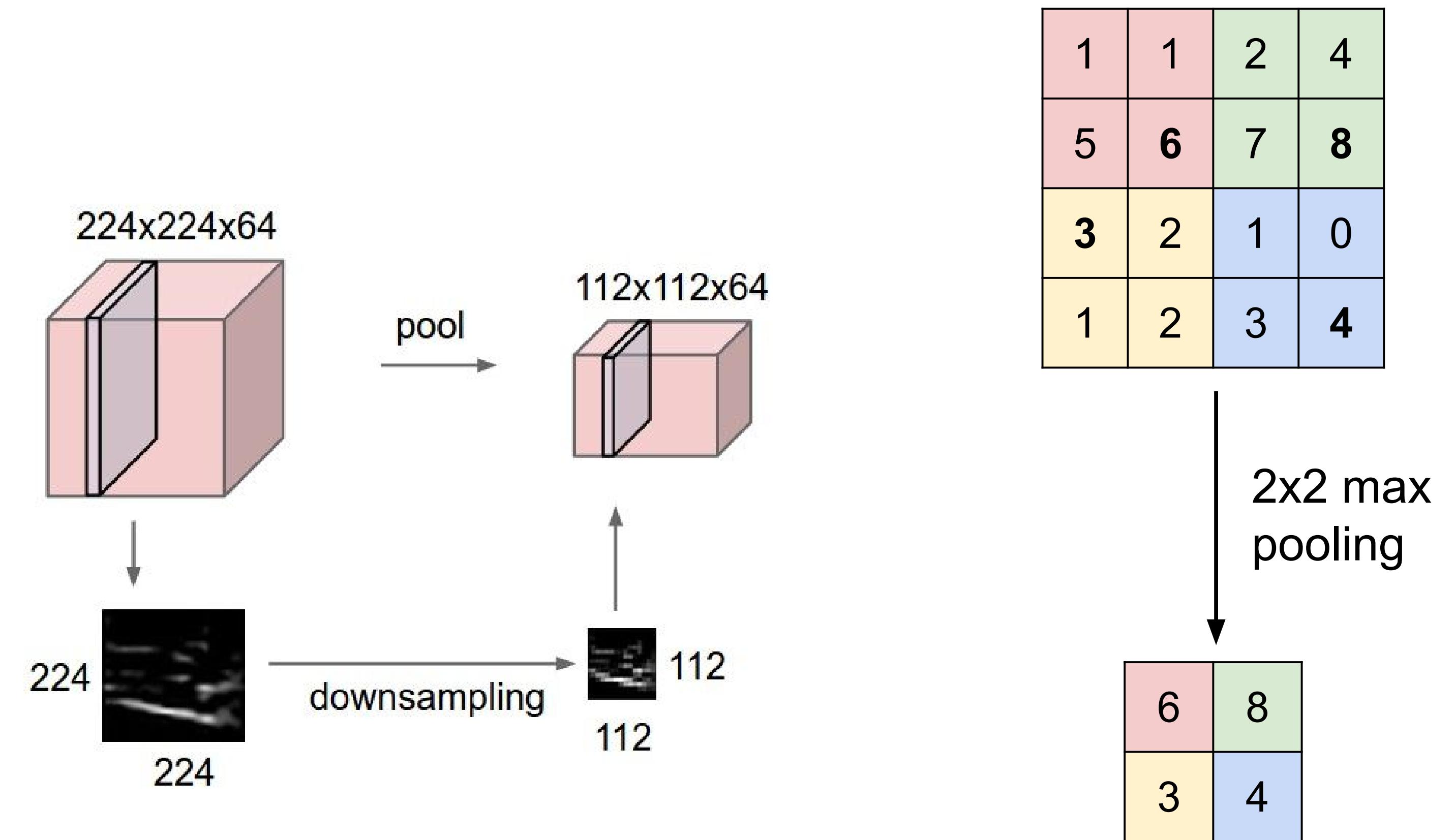
D depth of volume



$$(28 - 5 + 2*0) / 1 + 1 = 23/1 + 1 = 24$$

Pooling

- Downsampling
- Reduces spatial size
 - Aids faster computation
 - Regularization effect



Pooling

The pooling layers is a parameter less function

There are different kinds of pooling max, avg, L2-norm.

Max-pooling have been show to work good in practice.

$$W_{out} = \frac{W_{in}-F}{S} + 1 \quad (4 - 2) / 2 + 1 = 1 + 1 = 2$$

$$H_{out} = \frac{H_{in}-F}{S} + 1$$

$$D_{out} = D_{in}$$

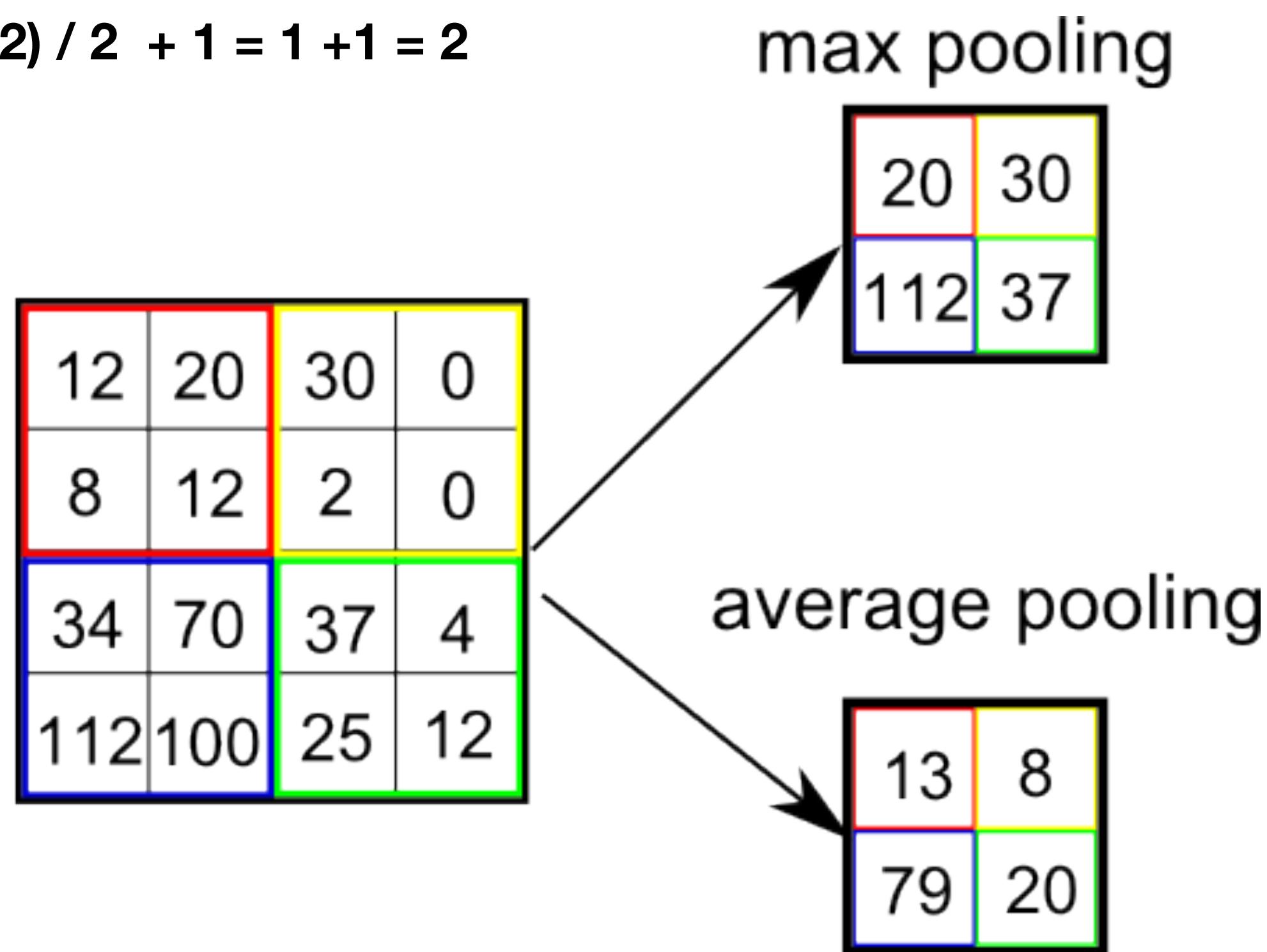
F spatial size

S stride

W width of volume

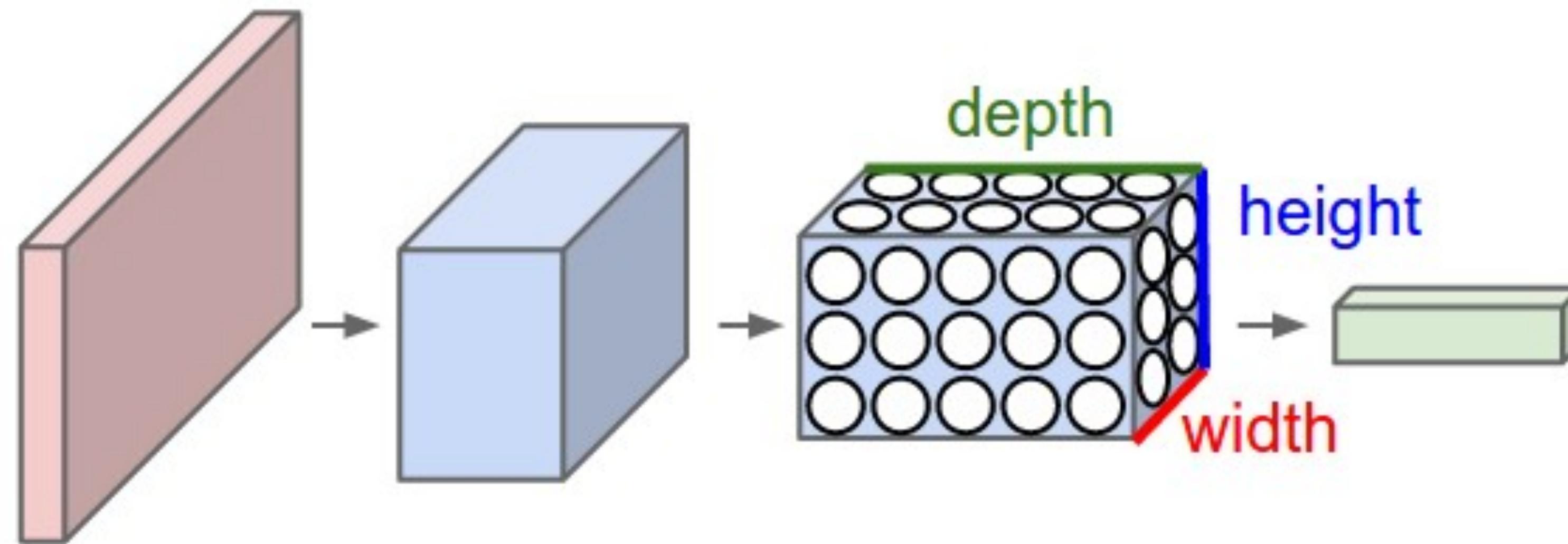
H height of volume

D depth of volume



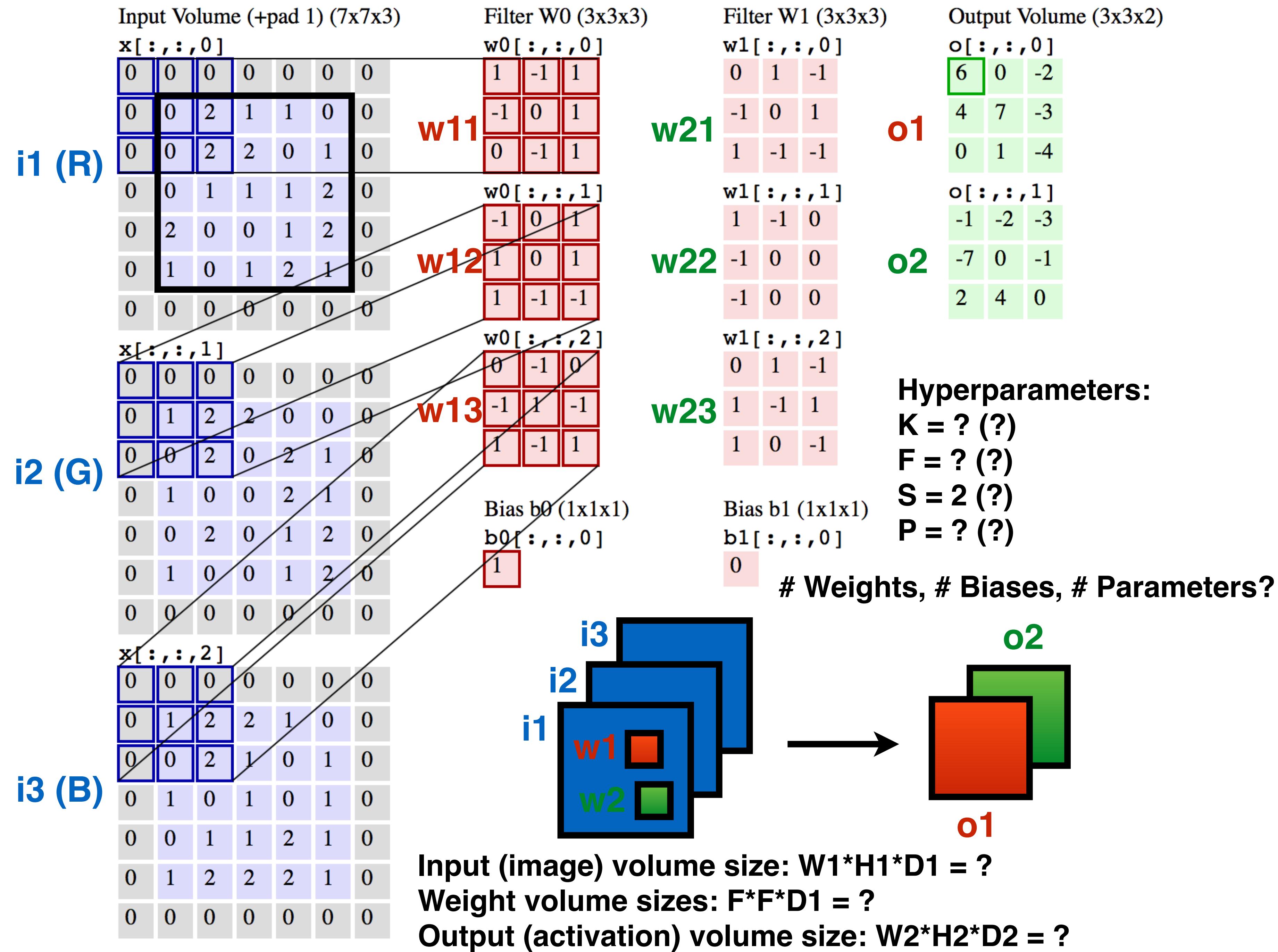
Intuition: cone shape of CNNs

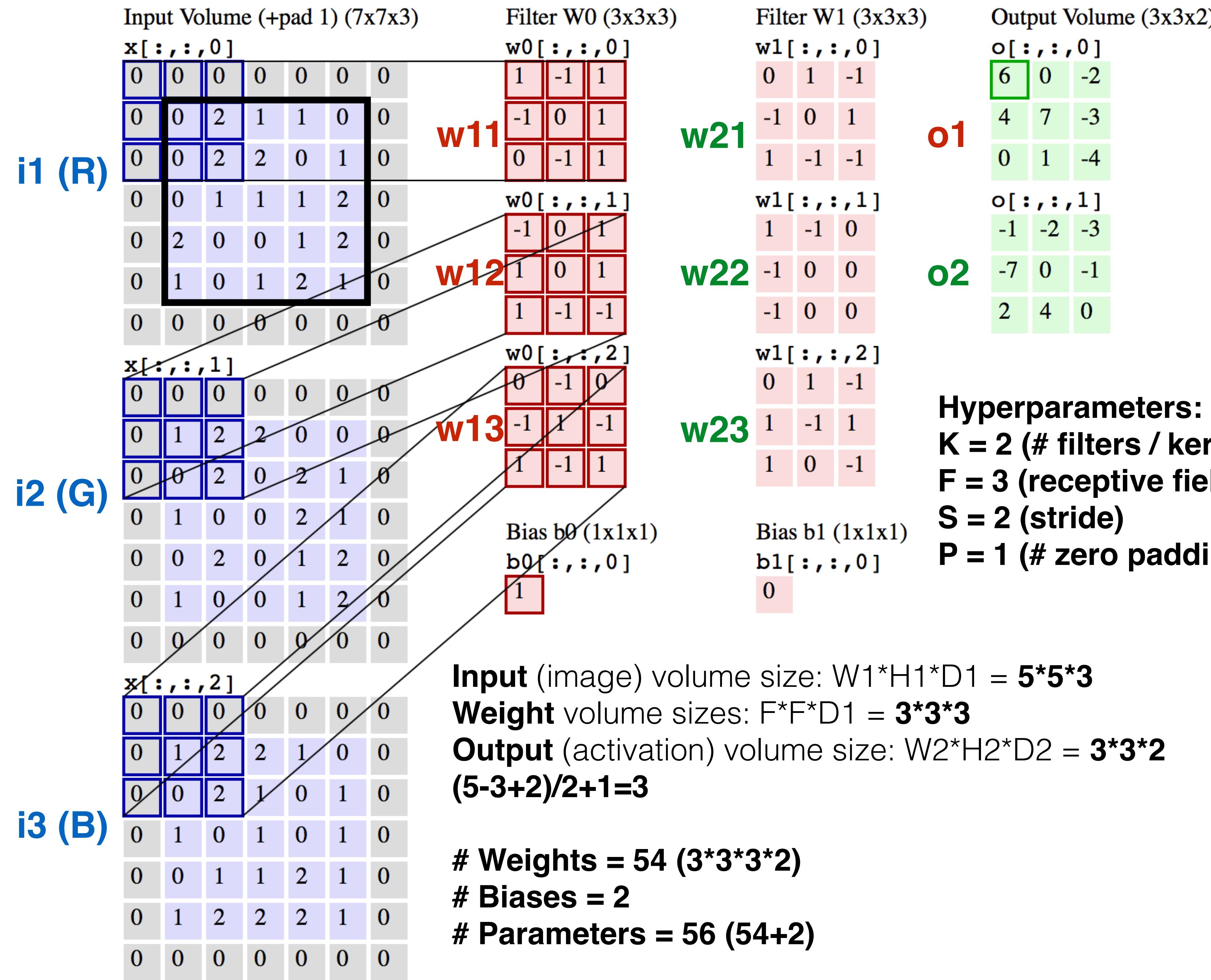
- More filters allow for more features
- Lower spatial dimensions reflect spatial invariance
- Gradually so the network does not need to infer classes directly from edge positions
- This hierarchical extraction of features is what our brain does so well



Numerical Example 1:

CNNs and Image Classification





Parameters

Hyperparameters:

K = 2 (# filters / kernels)

F = 3 (receptive field)

S = 2 (stride)

P = 1 (# zero paddings)

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d -th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.

A common setting of the hyperparameters is $F = 3, S = 1, P = 1$. However, there are common conventions and rules of thumb that motivate these hyperparameters.

Input (image) volume size: $W_1 \times H_1 \times D_1 = 5 \times 5 \times 3$

Weight volume sizes: $F \times F \times D_1 = 3 \times 3 \times 3$

Output (activation) volume size: $W_2 \times H_2 \times D_2 = 3 \times 3 \times 2$

$(5-3+2)/2+1 = 3 = (W_1-F+2P)/S + 1$

Weights = 54 = $(3 \times 3 \times 3) \times 2 = (F \times F \times D_1) \times K$

Biases = 2 = K

Parameters = 56 (54+2)

AlexNet (winner of the 2012 ImageNet com.):

- Input volume: $227 \times 227 \times 3$
- $K = 96, F = 11, S = 4, P = 0$
- Output Volume = ??

Parameters

Hyperparameters:

K = 2 (# filters / kernels)

F = 3 (receptive field)

S = 2 (stride)

P = 1 (# zero paddings)

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d -th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.

A common setting of the hyperparameters is $F = 3, S = 1, P = 1$. However, there are common conventions and rules of thumb that motivate these hyperparameters.

Input (image) volume size: $W_1 \times H_1 \times D_1 = 5 \times 5 \times 3$

Weight volume sizes: $F \times F \times D_1 = 3 \times 3 \times 3$

Output (activation) volume size: $W_2 \times H_2 \times D_2 = 3 \times 3 \times 2$

$(5-3+2)/2+1 = 3 = (W_1-F+2P)/S + 1$

Weights = 54 = (3*3*3)*2 = (F*F*D1)*K

Biases = 2 = K

Parameters = 56 (54+2)

AlexNet (winner of the 2012 ImageNet com.):

- Input volume: $227 \times 227 \times 3$
- $K = 96, F = 11, S = 4, P = 0$
- Output Volume = $55 \times 55 \times 96 = (227-11)/4 + 1 = 55$
- **Depth column = ?, Size = ?, Connected to = ?**
- **Parameter Sharing ?**

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	0
0	0	2	2	0	1	0	0

4

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	0
0	0	2	0	2	1	0	0

1

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	0
0	0	2	1	0	1	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w0[:, :, 2]$

0	-1	0
-1	1	-1
1	-1	1

 $W^*i + b$

1*0	-1*0	1*0
-1*0	0*0	1*2
0*0	-1*0	1*2

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Bias b0 (1x1x1)

 $b0[:, :, 0]$

0

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

$w(-1,-1) * i(-1,-1)$		
	$w(0,0) * i(0,0)$	
		$w(1,1) * i(1,1)$

-1*0	0*0	1*0
1*0	0*1	1*2
1*0	-1*0	-1*2

0*0	-1*0	0*0
-1*0	1*1	-1*2
1*0	-1*0	1*2

1

$$4(2+2) + 0(2-2) + 1(1-2+2) + 1 = 6$$

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w0[:, :, 2]$

0	-1	0
-1	1	-1
1	-1	1

Bias b0 (1x1x1)
 $b0[:, :, 0]$

1

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4

 $o[:, :, 1]$

-1	-2	-3
-7	0	-1
2	4	0

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

 $w(-1,-1) * i(-1,-1)$ $w(0,0) * i(0,0)$ $w(1,1) * i(1,1)$

$$\begin{array}{c}
 \begin{array}{ccc|c}
 ? & ? & ? & \\
 \hline
 ? & ? & ? & \\
 ? & ? & ? & \\
 \hline
 ? & ? & ? & \\
 \end{array} +
 \begin{array}{ccc|c}
 ? & ? & ? & \\
 \hline
 ? & ? & ? & \\
 ? & ? & ? & \\
 \hline
 ? & ? & ? & \\
 \end{array} +
 \begin{array}{ccc|c}
 ? & ? & ? & \\
 \hline
 ? & ? & ? & \\
 ? & ? & ? & \\
 \hline
 ? & ? & ? & \\
 \end{array} =
 \begin{array}{c}
 ? \\
 \end{array}
 \end{array}$$

Input Volume (+pad 1) (7x7x3)

```
x[ :, :, 0]
```

$$\begin{array}{ccccccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 2 & 1 & 1 & 0 & 0 & -3 \\ 0 & 0 & 2 & 2 & 0 & 1 & 0 & \end{array}$$

3

Filter W0 (3x3x3)

```
w0 [ :, :, 0 ]
```

1	-1	1
-1	0	1
0	-1	1
-1	0	1
1	0	1
1	-1	-1

```
x[ :, :, 1]
```

0

Bias b0 (1x1x1)
b0 [:, :, 0]

1

$$w^*i + b$$

Filter W1 (3x3x

w1[:, :, 0]

0	1	-1		6	0	-2
-1	0	1		4	7	-3
1	-1	-1		0	1	-4
w1[:, :, 1]				o[:, :, 1]		
1	-1	0		-1	-2	-3
-1	0	0		-7	0	-1
-1	0	0		2	4	0

```
w1[ :, :, 2]
```

0	1	-1
1	-1	1
1	0	-1

Bias b1 (1x1x1)

b1[:, :, 0]

$w(-1,-1) * i(-1,-1)$		
	$w(0,0) * i(0,0)$	
		$w(1,1) * i(1,1)$

*0 | 0*0 | 1*0

*2 0*2 1*0

Output Volume (3x3x2)

`o[:, :, 0]`

6	0	-2
4	7	-3
0	1	-4
○[:	,	1
-1	-2	-3
-7	0	-1
2	4	0

$$-3(-2+1-2) + 2(2+2-2) + 0(-2+2-1+2-1) + 1 = 0$$

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	0
0	0	2	2	0	1	0	0
0	0	1	1	1	2	0	0
0	2	0	0	1	2	0	0
0	1	0	1	2	1	0	0
0	0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w0[:, :, 2]$

0	1	0
-1	1	-1
1	-1	1

 $b0[:, :, 0]$

1

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	0
0	0	2	0	2	1	0	0
0	1	0	0	2	1	0	0
0	0	2	0	1	2	0	0
0	1	0	0	1	2	0	0
0	0	0	0	0	0	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	0
0	0	2	1	0	1	0	0
0	1	0	1	0	1	0	0
0	0	1	1	2	1	0	0
0	1	2	2	2	1	0	0
0	0	0	0	0	0	0	0

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Output Volume (3x3x2)

 $o[:, :, 0]$

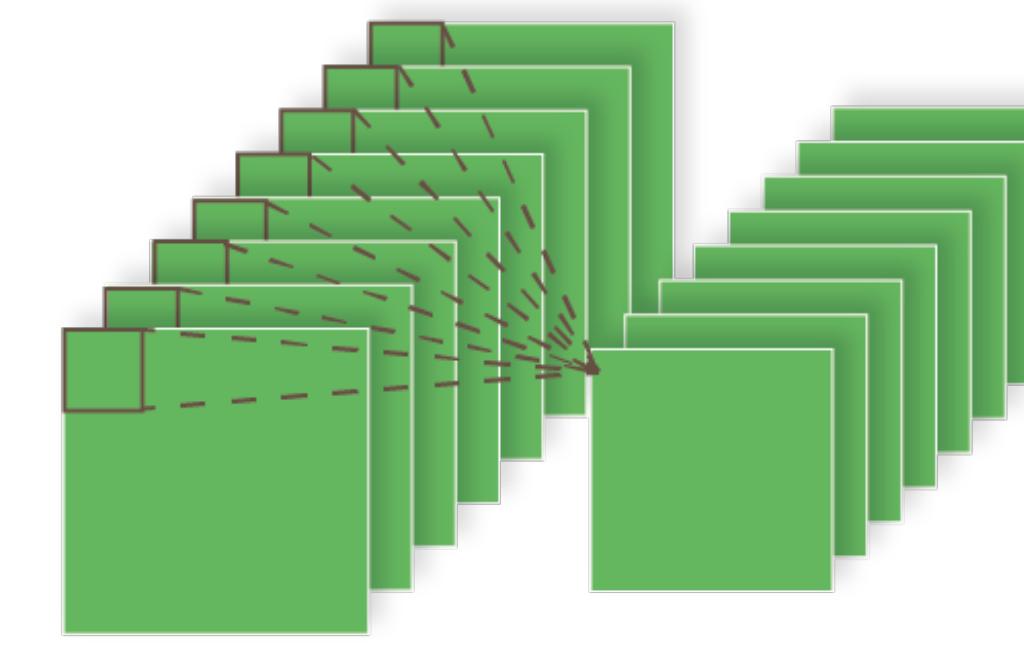
6	0	-2
4	7	-3
0	1	-4

 $o[:, :, 1]$

-1	-2	-3
-7	0	-1
2	4	0

 $b1[:, :, 0]$

0



Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

Bias b0 (1x1x1)
 $b0[:, :, 0]$

1

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	
0	0	2	0	2	1	0	
0	1	0	0	2	1	0	
0	0	2	0	1	2	0	
0	1	0	0	1	2	0	
0	0	0	0	0	0	0	

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	
0	0	2	1	0	1	0	
0	1	0	1	0	1	0	
0	0	1	1	2	1	0	
0	1	2	2	2	1	0	
0	0	0	0	0	0	0	

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

Bias b1 (1x1x1)
 $b1[:, :, 0]$

0

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4

 $o[:, :, 1]$

-1	-2	-3
-7	0	-1
2	4	0

Bias b1 (1x1x1)
 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w0[:, :, 2]$

0	-1	0
-1	1	-1
1	-1	1

Bias b0 (1x1x1)

 $b0[:, :, 0]$

1

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3
-7	0	-1
2	4	0

 $o[:, :, 1]$

0	1	-1
1	-1	1
1	0	-1

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	
0	0	2	0	2	1	0	
0	1	0	0	2	1	0	
0	0	2	0	1	2	0	
0	1	0	0	1	2	0	
0	0	0	0	0	0	0	

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	
0	0	2	1	0	1	0	
0	1	0	1	0	1	0	
0	0	1	1	2	1	0	
0	1	2	2	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

Bias b0 (1x1x1)
 $b0[:, :, 0]$

1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4
1	-1	0
-1	-2	-3
-7	0	-1
2	4	0

 $o[:, :, 1]$

0	1	-1
1	-1	1
1	0	-1

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w0[:, :, 2]$

0	-1	0
1	1	-1
1	-1	1

 $b0[:, :, 0]$

1

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	0	1	2
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $w1[:, :, 0]$

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

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

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

 $b1[:, :, 0]$

0

Output Volume (3x3x2)

 $o[:, :, 0]$

6 0 -2

4 7 -3

0 1 -4

 $o[:, :, 1]$

-1 -2 -3

-7 0 -1

2 4 0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	0
0	0	2	2	0	1	0	0
0	0	1	1	1	2	0	0
0	2	0	0	1	2	0	0
0	1	0	1	2	1	0	0
0	0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w0[:, :, 2]$

0	-1	0
-1	1	-1
1	-1	1

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	0
0	0	2	0	2	1	0	0
0	1	0	0	2	1	0	0
0	0	2	0	1	2	0	0
0	1	0	0	1	2	0	0
0	0	0	0	0	0	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	0
0	0	2	1	0	1	0	0
0	1	0	1	0	1	0	0
0	0	1	1	2	1	0	0
0	1	2	2	2	1	0	0
0	0	0	0	0	0	0	0

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4

 $o[:, :, 1]$

-1	-2	-3
-7	0	-1
2	4	0

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	
0	0	2	0	2	1	0	
0	1	0	0	2	1	0	
0	0	2	0	1	2	0	
0	1	0	0	1	2	0	
0	0	0	0	0	0	0	

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	
0	0	2	1	0	1	0	
0	1	0	1	0	1	0	
0	0	1	1	2	1	0	
0	1	2	2	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

Bias b0 (1x1x1)
 $b0[:, :, 0]$

1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	1	-1
1	-1	1
1	0	-1

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4

 $o[:, :, 1]$

-1	-2	-3
-7	0	-1
2	4	0

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

Output Volume (3x3x2)

6	0	-2
4	7	-3
0	1	-4
$\circ[:, :, 1]$		
-1	-2	-3
-7	0	-1
2	4	0

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	0	0	0	0	0	0

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	1	0
-1	0	0
-1	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Bias b0 (1x1x1)

 $b0[:, :, 0]$

1

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

Output Volume (3x3x2)

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3
-7	0	-1
2	4	0

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

Bias b0 (1x1x1)

 $b0[:, :, 0]$

1

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Input Volume (+pad 1) (7x7x3)

```
x[:, :, 0]
```

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0

Filter W0 (3x3x3)

w0[:, :, 0]

Filter W1 (3x3x3)

w1[:, :, :, 0]

Output Volume (3x3x2)

`o[:, :, 0]`

6	0	-2
4	7	-3
0	1	-4

```
x[, :, 1]
```

0 0 0 0 0 0 0

0	1	2	2	0	0	0
0	0	2	0	2	1	0

```
w0[:, :, 2]
```

0	-1	0
-1	1	-1
1	-1	1

w1 [; ; ;]

0	1	-1
1	-1	1
1	0	-1

~~Bias b0 (1x1x1)~~

```
b0[:, :, 0]  
1
```

~~Bias b1 (1x1x1)~~

b1 [

```
x[, :, 2]
```

0 0 0 0 0 0 0

0	1	2	2	1	0	0
0	0	2	1	0	1	0

~~Bias b0 (1x1)~~

1

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

Output Volume (3x3x2)

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3
-7	0	-1
2	4	0

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $w0[:, :, 1]$

0	-1	0
-1	1	-1
1	-1	1

 $w1[:, :, 1]$

0	1	1
1	-1	1
1	0	-1

 $w0[:, :, 2]$

0	-1	0
-1	1	-1
1	-1	1

 $w1[:, :, 2]$

0	1	1
1	-1	1
1	0	-1

 $b0[:, :, 0]$

1

 $b1[:, :, 0]$

0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1
-1	0	1
1	-1	1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1
1	-1	0
-1	0	0
-1	0	0

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3
-7	0	-1
2	4	0

 $o[:, :, 1]$

-1	-2	-3
-7	0	-1
2	4	0

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $w0[:, :, 1]$

0	-1	0
-1	1	-1
1	-1	1

 $w1[:, :, 1]$

0	1	-1
1	-1	1
1	0	-1

Bias b0 (1x1x1)

 $b0[:, :, 0]$

1

Bias b1 (1x1x1)

 $b1[:, :, 0]$

0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0
0	0	2	1	1	0	0
0	0	2	2	0	1	0
0	0	1	1	1	2	0
0	2	0	0	1	2	0
0	1	0	1	2	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

Output Volume (3x3x2)

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3
-7	0	-1
2	4	0

 $x[:, :, 1]$

0	0	0	0	0	0	0
0	1	2	2	0	0	0
0	0	2	0	2	1	0
0	1	0	0	2	1	0
0	0	2	0	1	2	0
0	1	0	0	1	2	0
0	0	0	0	0	0	0

 $w0[:, :, 1]$

0	1	0
-1	1	-1
1	-1	1

 $w1[:, :, 1]$

0	1	-1
1	1	1
1	0	-1

 $bias b0 (1x1x1)$ $b0[:, :, 0]$

1

 $bias b1 (1x1x1)$ $b1[:, :, 0]$

0

 $x[:, :, 2]$

0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	0	2	1	0	1	0
0	1	0	1	0	1	0
0	0	1	1	2	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1
-1	0	1
1	-1	-1

Output Volume (3x3x2)

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3
-7	0	-1
2	4	0

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	
0	0	2	0	2	1	0	
0	1	0	0	2	1	0	
0	0	2	0	1	2	0	
0	1	0	0	1	2	0	
0	0	0	0	0	0	0	

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
1	0	0
-1	0	0

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	
0	0	2	1	0	1	0	
0	1	0	1	0	1	0	
0	0	1	1	2	1	0	
0	1	2	2	2	1	0	
0	0	0	0	0	0	0	

 $b0[:, :, 0]$

1

 $b1[:, :, 0]$

0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1
-1	0	1
0	-1	1

 $w0[:, :, 1]$

-1	0	1
1	0	1
1	-1	-1

 $w1[:, :, 1]$

1	-1	0
-1	0	0
-1	0	0

 $w1[:, :, 2]$

0	-1	0
-1	1	-1
1	-1	1

 $w1[:, :, 2]$

0	1	-1
1	1	1
1	0	-1

 $w1[:, :, 2]$

1

 $b0[:, :, 0]$

1

 $b1[:, :, 0]$

0

 $b1[:, :, 0]$ $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	
0	0	2	1	0	1	0	
0	1	0	1	0	1	0	
0	0	1	1	2	1	0	
0	1	2	2	2	1	0	
0	0	0	0	0	0	0	

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2
4	7	-3
0	1	-4
-1	-2	-3

 $o[:, :, 1]$

-7	0	-1
2	4	0

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1	
-1	0	1	
0	-1	1	
			?

 $w0[:, :, 1]$

-1	0	1	
1	0	1	
1	-1	-1	
			?

 $w0[:, :, 2]$
 $b0[:, :, 0]$

0	-1	0	
-1	1	-1	
1	-1	1	
			?

 $W^*i + b$

 $? (?)$

+

 $? (?)$

+

 $? (?)$

+

 $? (?)$

=

 $?$

Output Volume (3x3x2)

 $o[:, :, 0]$

6	0	-2	
4	7	-3	
0	1	-4	
			$o[:, :, 1]$
			-1 -2 -3
			-7 0 -1
	2	4	0

 $o[:, :, 1]$

-1	-2	-3	
-7	0	-1	
2	4	0	

 $Bias b1 (1x1x1)$
 $b1[:, :, 0]$

0		

$w(-1, -1) * i(-1, -1)$		
	$w(0, 0) * i(0, 0)$	
		$w(1, 1) * i(1, 1)$

 $?$

Input Volume (+pad 1) (7x7x3)

 $x[:, :, 0]$

0	0	0	0	0	0	0	0
0	0	2	1	1	0	0	
0	0	2	2	0	1	0	
0	0	1	1	1	2	0	
0	2	0	0	1	2	0	
0	1	0	1	2	1	0	
0	0	0	0	0	0	0	

Filter W0 (3x3x3)

 $w0[:, :, 0]$

1	-1	1	
-1	0	1	
0	-1	1	
-1	0	1	
1	0	1	
1	-1	1	
1	-1	0	
-1	0	0	
-1	0	0	

Filter W1 (3x3x3)

 $w1[:, :, 0]$

0	1	-1	
-1	0	1	
1	-1	-1	
1	-1	0	
-1	0	0	
-1	0	0	
0	1	-1	
1	-1	1	
1	0	-1	

Output Volume (3x3x2)

6	0	-2	
4	7	-3	
0	1	-4	
-1	-2	-3	
-7	0	-1	
2	4	0	

 $x[:, :, 1]$

0	0	0	0	0	0	0	0
0	1	2	2	0	0	0	
0	0	2	0	2	1	0	
0	1	0	0	2	1	0	
0	0	2	0	1	2	0	
0	1	0	0	1	2	0	
0	0	0	0	0	0	0	

 $w0[:, :, 2]$

0	-1	0	
-1	1	-1	
1	-1	1	
1	-1	1	
1	0	-1	
1	0	-1	

 $w1[:, :, 2]$

0	1	-1	
1	-1	1	
1	0	-1	
0	1	-1	
1	0	-1	
1	0	-1	

 $x[:, :, 2]$

0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	
0	0	2	1	0	1	0	
0	1	0	1	0	1	0	
0	0	1	1	2	1	0	
0	1	2	2	2	1	0	
0	0	0	0	0	0	0	

 0^*1 1^*2 -1^*0 1^*1 -1^*2 0^*0 0^*2 1^*1 -1^*0 1^*2 -1^*1 1^*0

$$0 (2-2) + -2 (1-2-1) + 2 (1+2-1) + 0 = 0$$

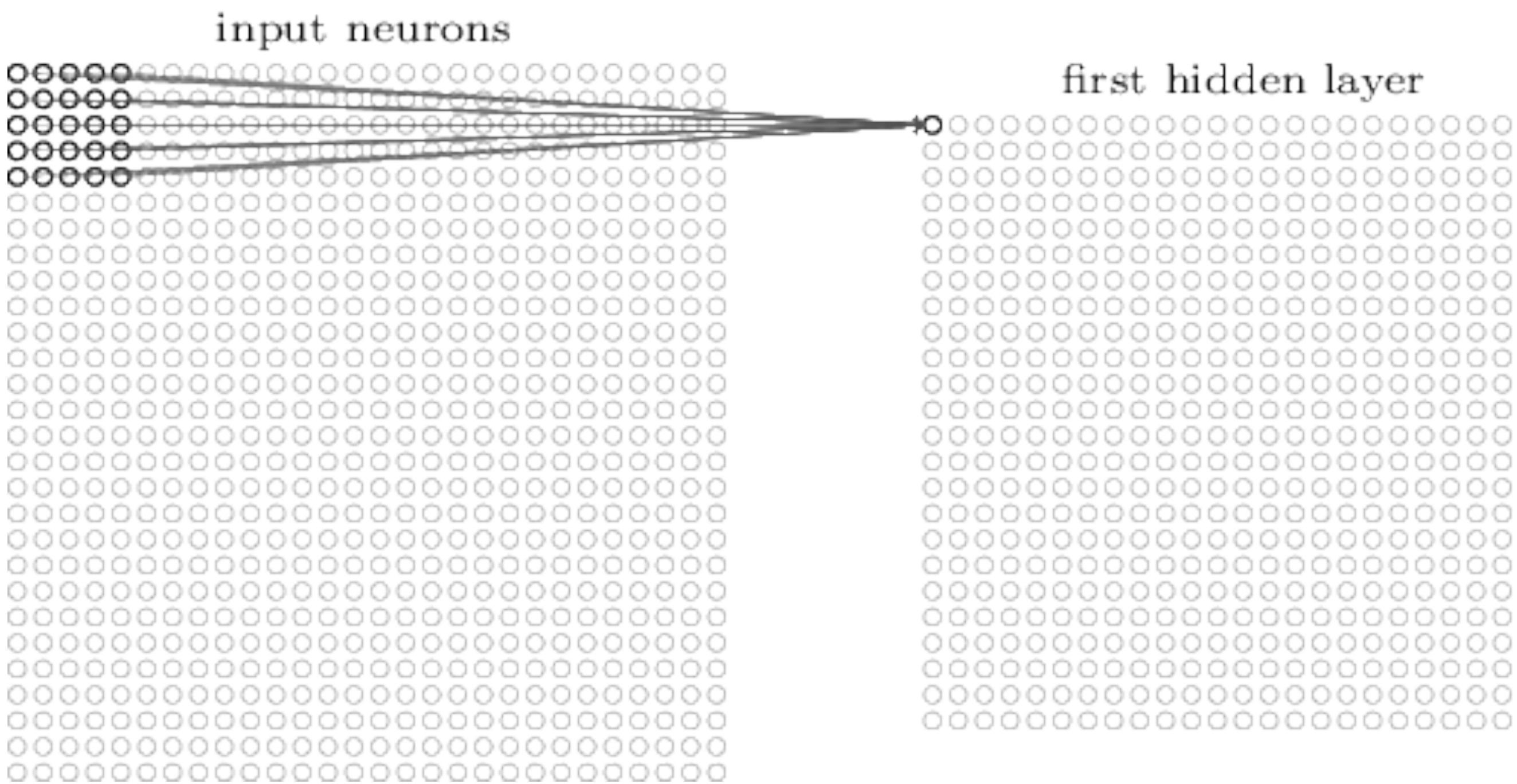
0

Numerical Example 2:

CNNs and Image Classification

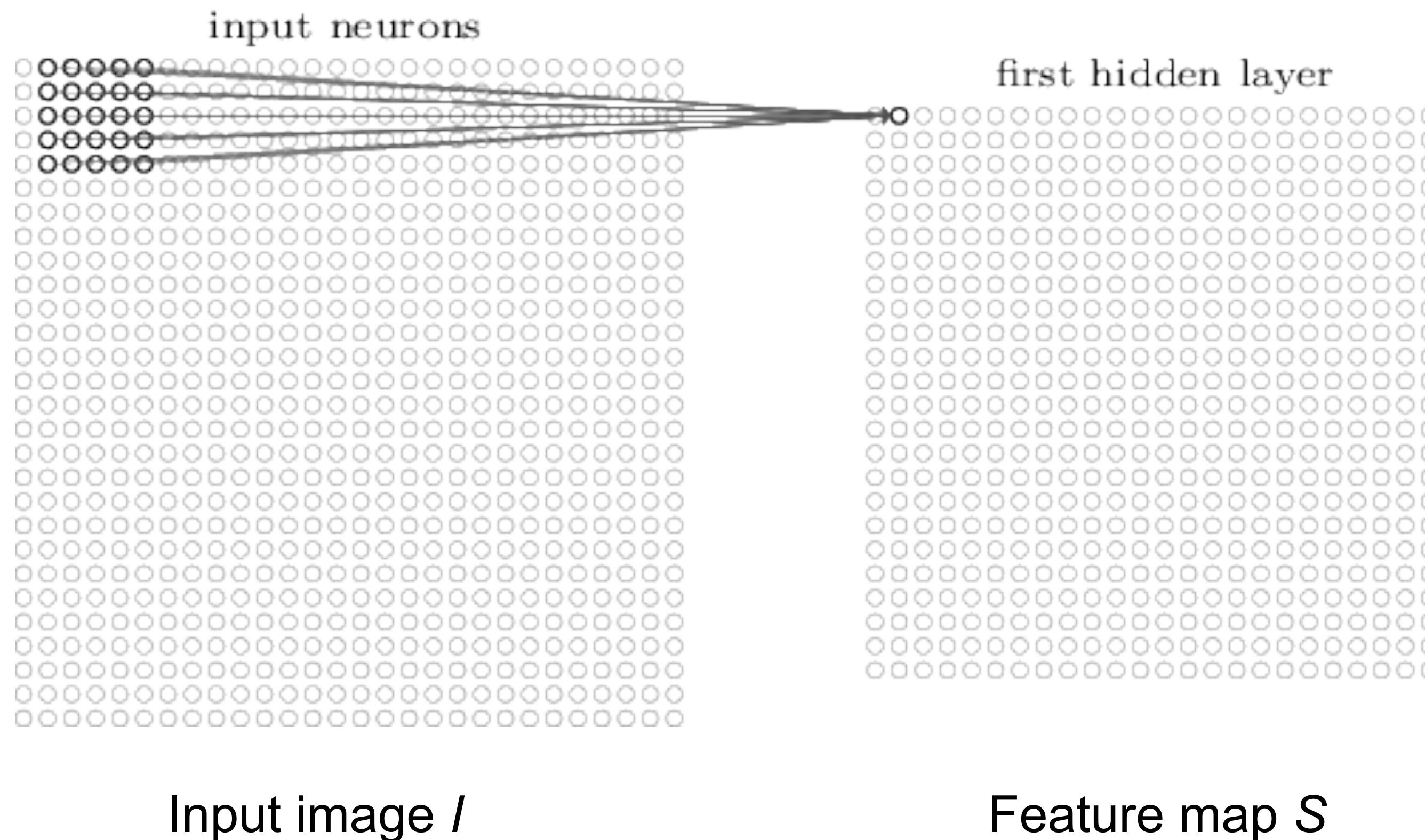
Convolution: Local Receptive Fields

- Each neuron is connected to a region in the previous layer
- 5×5 local receptive field



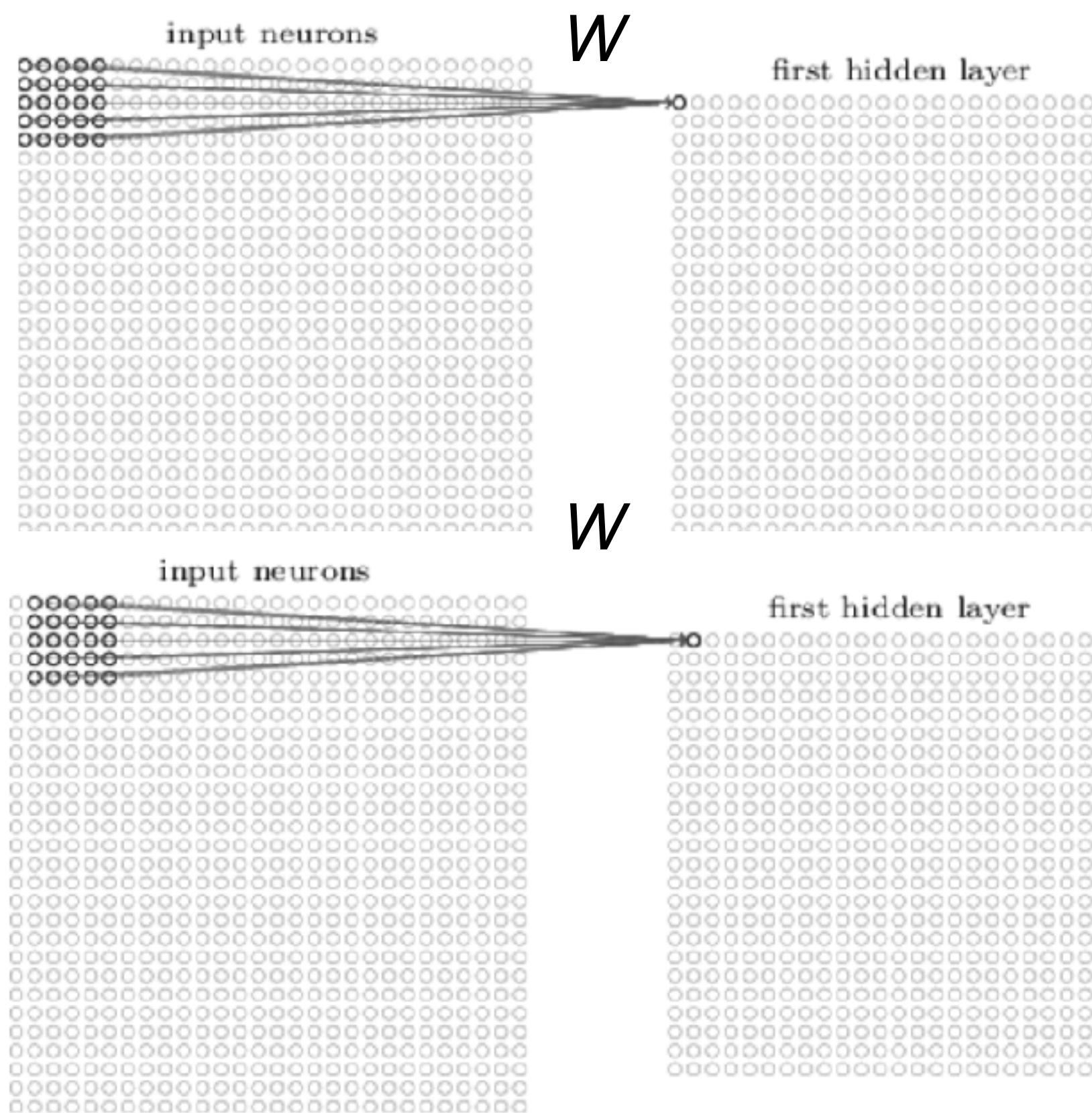
Convolution: Local Receptive Fields

- Each neuron is connected to a region in the previous layer
- 5x5 local receptive field



Convolution: Shared Weights

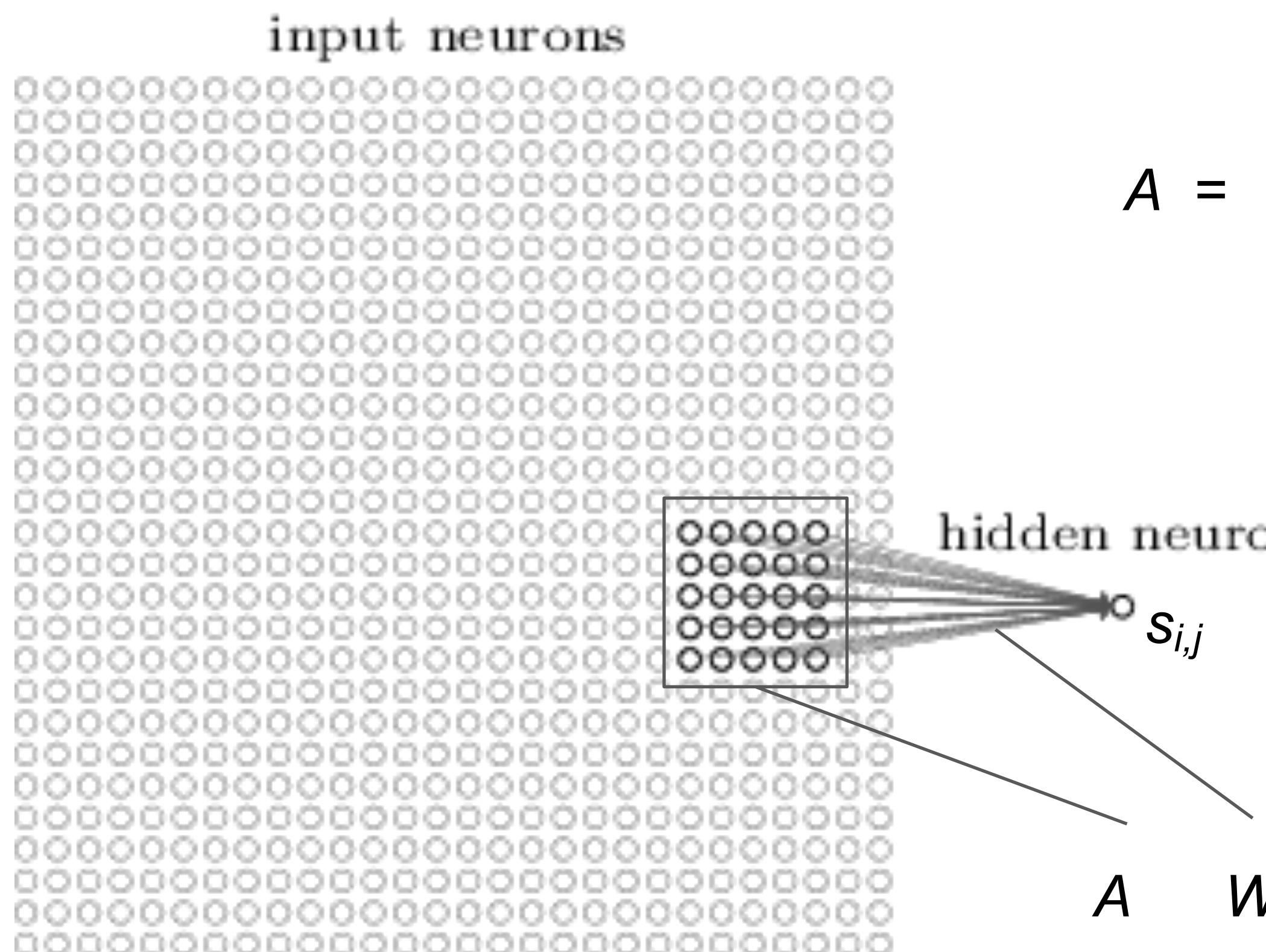
- The weights W connecting the neurons in the local receptive field and the hidden neuron are the same across the feature map



$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & w_{4,5} \\ w_{5,1} & w_{5,2} & w_{5,3} & w_{5,4} & w_{5,5} \end{bmatrix}$$

Convolution: Convolution Operation

- The values of the neurons ($s_{i,j}$) in feature map S are calculated with convolution. Given input image I , local receptive field A , shared weights W and shared bias b , we have:



$$s_{i,j} = (I * W)_{i,j} + b = \sum_m \sum_n i_{i+m, j+n} w_{m,n} + b$$

$$s_{i,j} = \langle A, W \rangle + b = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

Convolution: Example

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{1:3,1:3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$s_{1,1} = (1 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 2 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 3) - 2 =$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \left[\begin{array}{c} \boxed{\quad} \\ \vdots \\ \boxed{\quad} \end{array} \right]$$

Convolution: Example

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{1:3,1:3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$s_{1,1} = (1 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 2 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 3) - 2 = 4$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 4 \end{bmatrix}$$

Convolution: Example (2)

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{1:3,2:4} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$s_{1,2} = (0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 2 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot 3) - 2 =$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \left[\begin{array}{c} 4 \rightarrow \boxed{} \\ \vdots \end{array} \right]$$

Convolution: Example (2)

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{1:3,2:4} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$s_{1,2} = (0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 2 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 1 \cdot 3) - 2 = 1$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$S = \left[\begin{array}{c} 4 \rightarrow 1 \\ \vdots \\ \vdots \\ \vdots \end{array} \right]$$

Convolution: Example (3)

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{1:3,3:5} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$s_{1,3} = (1 \cdot 2 + 1 \cdot 3) - 2 =$$

$$S = \begin{bmatrix} 4 & 1 & \square \\ & & \end{bmatrix}$$

Convolution: Example (3)

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2 \quad s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b \quad \text{kernel size = 3x3, stride = 1}$$

$$A = I_{1:3,3:5} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 & \boxed{0 & 1 & 1} & 0 & 0 & 1 \\ 0 & 1 & \boxed{0 & 1 & 1} & 0 & 0 & 1 \\ 0 & 0 & \boxed{1 & 1 & 1} & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$s_{1,3} = (1 \cdot 2 + 1 \cdot 3) - 2 = 3$$

A number line starting at -4 and ending at 3. The integers -4, 1, and 3 are labeled. A red arrow points to the number 3.

Convolution: Example (4)

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{6:8,6:8} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$s_{6,6} = (1 \cdot 1 + 1 \cdot 2 + 1 \cdot 3) - 2 =$$

$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & -1 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & \square \end{bmatrix}$$

Convolution: Example (4)

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, b = -2$$

$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b$$

kernel size = 3x3, stride = 1

$$A = I_{6:8,6:8} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

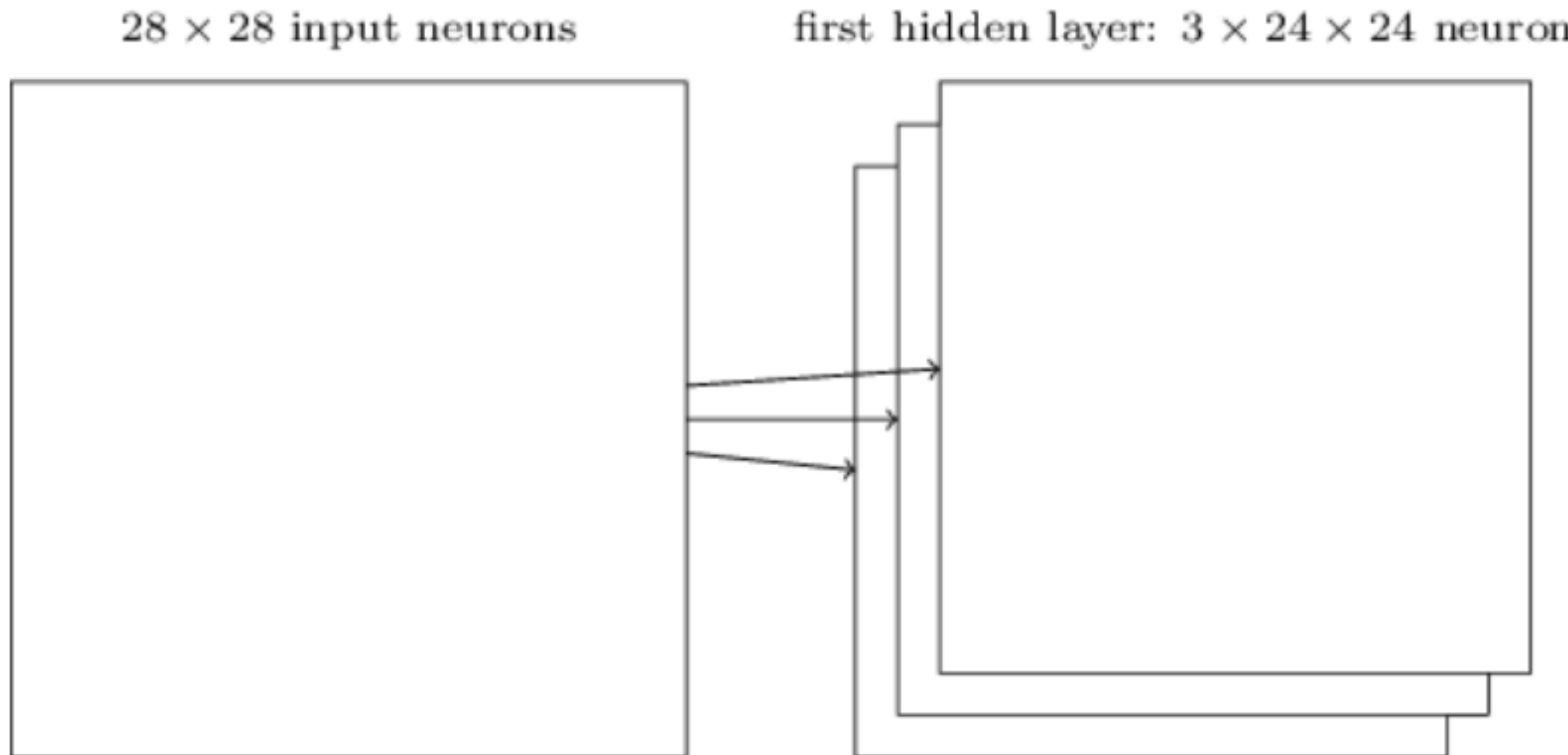
$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$s_{6,6} = (1 \cdot 1 + 1 \cdot 2 + 1 \cdot 3) - 2 = 4$$

$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & -1 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

Convolution: Feature Maps

- 3 5x5 kernels, each with its own set of weights, results in 3 feature maps



Convolution: Comparison

- Compared with fully-connected layer
 - **20x20** input neurons
 - Fully-connected layer with 10 neurons: $(20 * 20) * 10 = \text{4 000 weights}$
 - Convolutional layer with 10 kernels of size 3x3: $(3 * 3) * 10 = \text{90 weights}$
 - **200x200** input neurons
 - Fully-connected layer with 10 neurons: $(200 * 200) * 10 = \text{400 000 weights}$
 - Convolutional layer with 10 kernels of size 3x3: $(3 * 3) * 10 = \text{90 weights}$

Rectified Linear Unit (ReLU)

- Following convolution a non-linearity operation is performed
 - Non-linearity to learn non-linear patterns
- ReLU
 - Replace all negative values with 0

$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

ReLU: Example

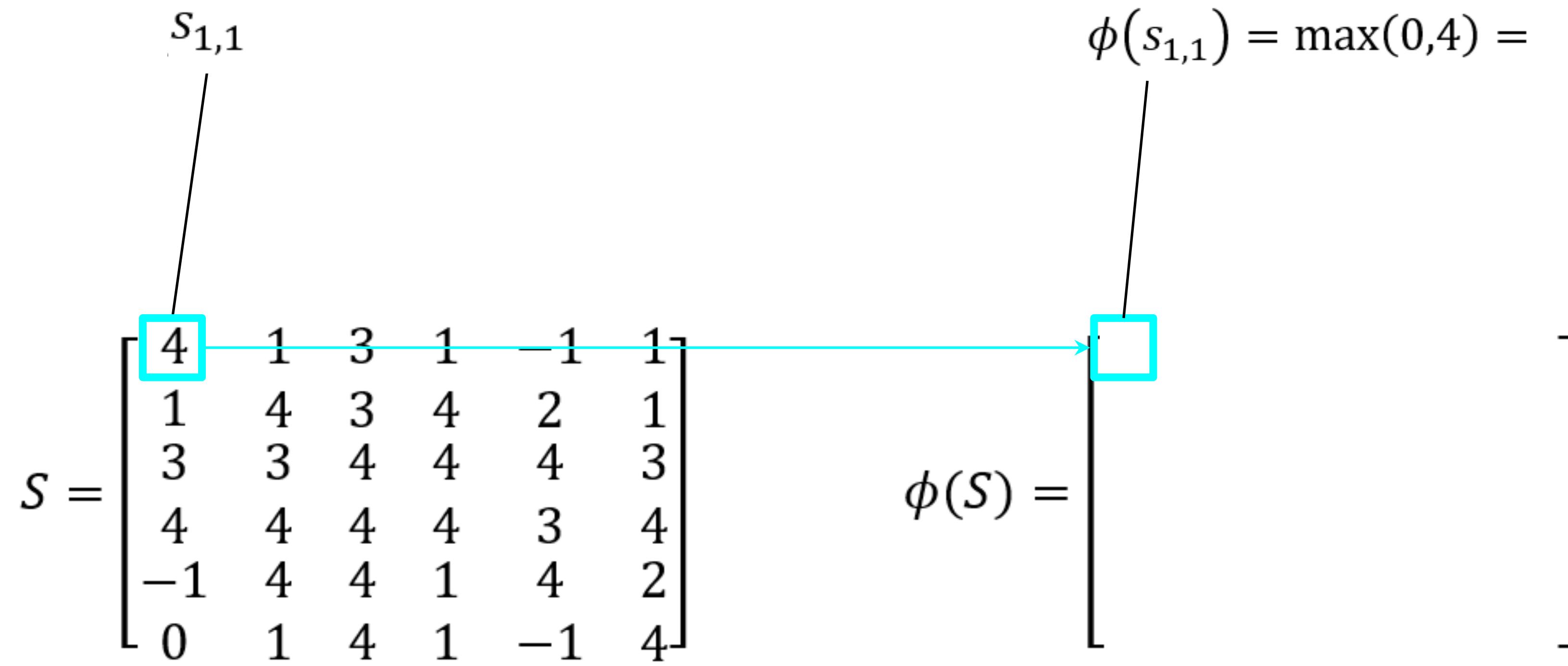
$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & -1 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

$s_{1,1}$

$\phi(s_{1,1}) = \max(0, 4) =$

$\phi(S) =$



ReLU: Example

$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

$$S = \begin{bmatrix} s_{1,1} & 1 & 3 & 1 & -1 & 1 \\ 4 & 1 & 4 & 3 & 4 & 2 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

$\phi(s_{1,1}) = \max(0, 4) = 4$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & -1 & 1 \\ 4 & 1 & 4 & 3 & 4 & 2 \\ 4 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 4 & 4 & 4 & 1 & 4 & 2 \\ 4 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

ReLU: Example (2)

$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & \boxed{-1} & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

$s_{1,5}$

$\phi(s_{1,5}) = \max(0, -1) =$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & \boxed{\square} & \end{bmatrix}$$

ReLU: Example (2)

$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & \boxed{-1} & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

$s_{1,5}$

$\phi(s_{1,5}) = \max(0, -1) = 0$

$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & \boxed{0} & \\ & & & & & \end{bmatrix}$

ReLU: Example (3)

$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & -1 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

$$\phi(s_{6,6}) = \max(0, 4) =$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & \square \end{bmatrix}$$

ReLU: Example (3)

$$\phi(s_{i,j}) = \max(0, s_{i,j})$$

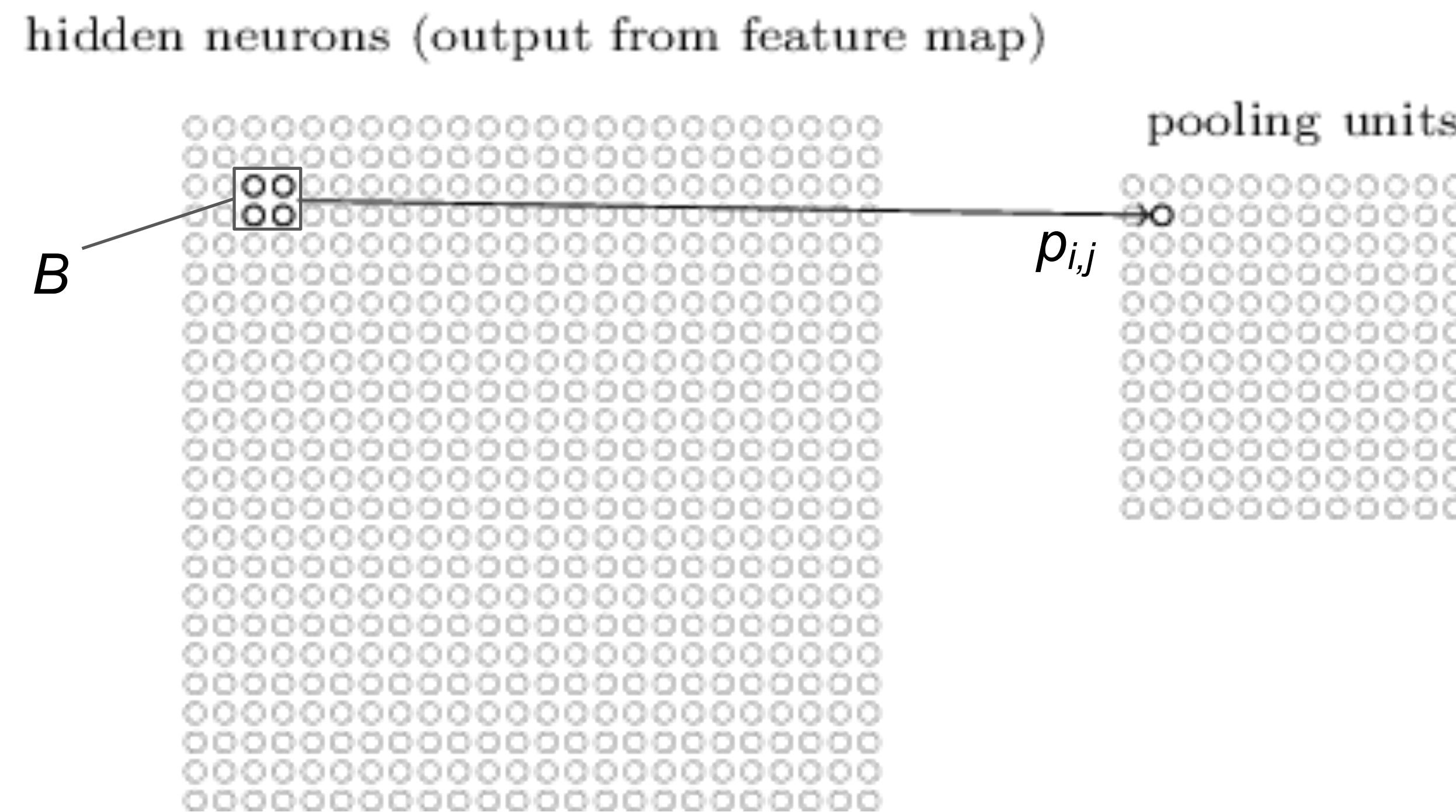
$$S = \begin{bmatrix} 4 & 1 & 3 & 1 & -1 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ -1 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & -1 & 4 \end{bmatrix}$$

$$\phi(s_{6,6}) = \max(0, 4) = 4$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

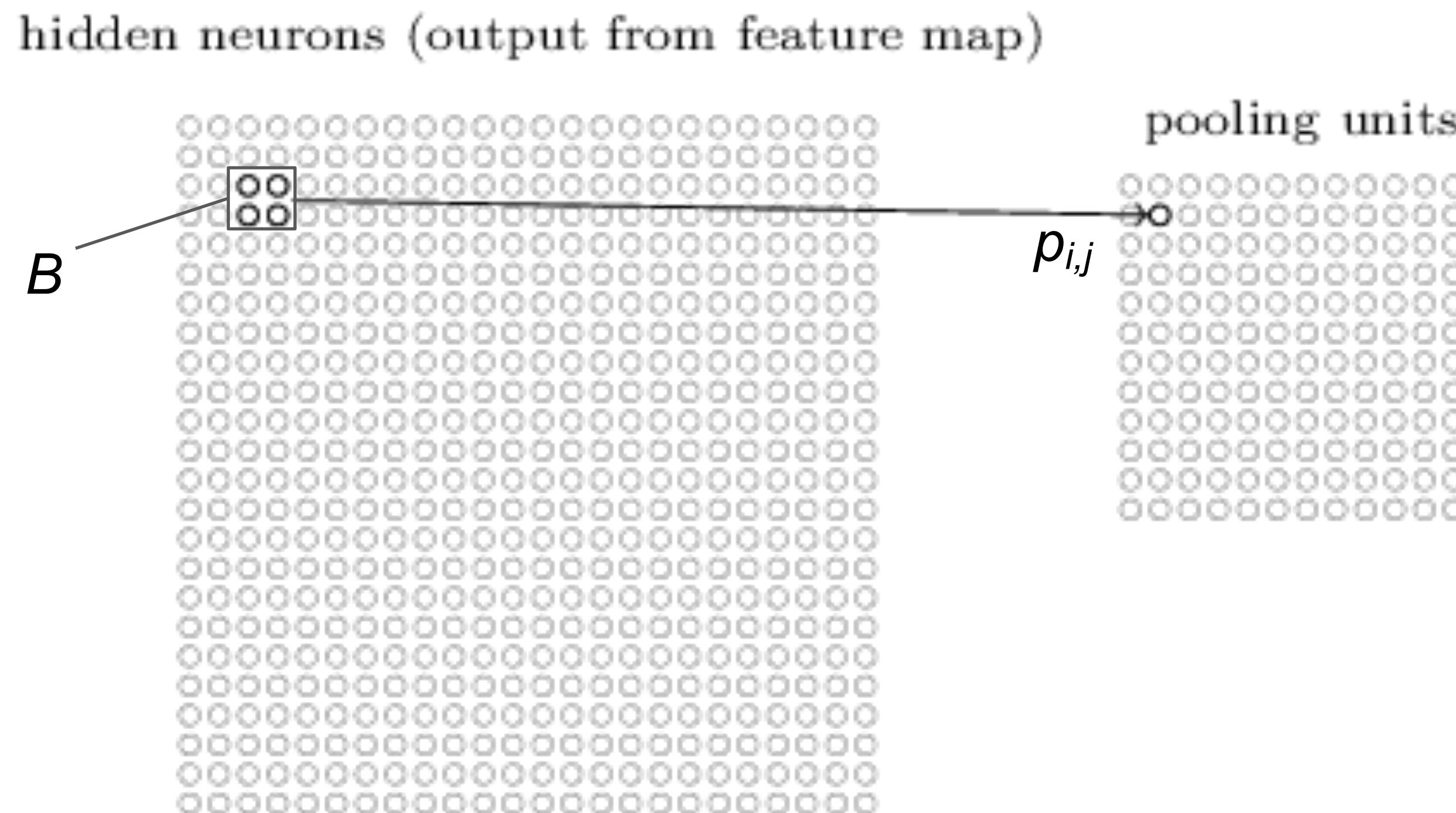
Pooling

- Reduce dimension of feature map while preserving the most important information



Pooling

- The values of the neurons ($p_{i,j}$) in the pooled feature map P are calculated as a function of the region B :



$$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Max pooling: $p_{i,j} = \max(B) = \max(b_{1,1}, b_{1,2}, b_{2,1}, b_{2,2})$

Avg pooling: $p_{i,j} = \text{avg}(B) = \frac{b_{1,1} + b_{1,2} + b_{2,1} + b_{2,2}}{4}$

L2 pooling: $p_{i,j} = \text{l2}(B) = \sqrt{b_{1,1}^2 + b_{1,2}^2 + b_{2,1}^2 + b_{2,2}^2}$

Max Pooling: Example

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{1:2,1:2} = \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}$$

$$p_{1,1} = \max(B) = \max\left(\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}\right) =$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$P = \begin{bmatrix} \end{bmatrix}$$

Max Pooling: Example

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{1:2,1:2} = \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}$$

$$p_{1,1} = \max(B) = \max\left(\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix}\right) = 4$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$P = \begin{bmatrix} 4 \\ \end{bmatrix}$$

Max Pooling: Example (2)

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{1:2,3:4} = \begin{bmatrix} 3 & 1 \\ 3 & 4 \end{bmatrix}$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & \boxed{3} & \boxed{1} & 0 & 1 \\ 1 & 4 & \boxed{3} & \boxed{4} & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$p_{1,2} = \max\left(\begin{bmatrix} 3 & 1 \\ 3 & 4 \end{bmatrix}\right) =$$

$$P = \begin{bmatrix} 4 \\ \square \end{bmatrix}$$

Max Pooling: Example (2)

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{1:2,3:4} = \begin{bmatrix} 3 & 1 \\ 3 & 4 \end{bmatrix}$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & \boxed{3} & \boxed{1} & 0 & 1 \\ 1 & 4 & \boxed{3} & \boxed{4} & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$p_{1,2} = \max\left(\begin{bmatrix} 3 & 1 \\ 3 & 4 \end{bmatrix}\right) = 4$$

$$P = \begin{bmatrix} 4 & 4 \\ & \end{bmatrix}$$

Max Pooling: Example (3)

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{1:2,5:6} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$p_{1,3} = \max\left(\begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}\right) =$$

$$P = \begin{bmatrix} 4 & 4 \\ & \end{bmatrix}$$

Max Pooling: Example (3)

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{1:2,5:6} = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$p_{1,3} = \max\left(\begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}\right) = 2$$

$$P = \begin{bmatrix} 4 & 4 \\ & 2 \end{bmatrix}$$

Max Pooling: Example (4)

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{5:6,5:6} = \begin{bmatrix} 4 & 2 \\ 0 & 4 \end{bmatrix}$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

$$p_{3,3} = \max\left(\begin{bmatrix} 4 & 2 \\ 0 & 4 \end{bmatrix}\right) =$$

$$P = \begin{bmatrix} 4 & 4 & 2 \\ 4 & 4 & 4 \\ 4 & 4 & \square \end{bmatrix}$$

Max Pooling: Example (4)

$$p_{i,j} = \max(B)$$

region size = 2x2, stride = 1

$$B = \phi(S)_{5:6,5:6} = \begin{bmatrix} 4 & 2 \\ 0 & 4 \end{bmatrix}$$

$$\phi(S) = \begin{bmatrix} 4 & 1 & 3 & 1 & 0 & 1 \\ 1 & 4 & 3 & 4 & 2 & 1 \\ 3 & 3 & 4 & 4 & 4 & 3 \\ 4 & 4 & 4 & 4 & 3 & 4 \\ 0 & 4 & 4 & 1 & 4 & 2 \\ 0 & 1 & 4 & 1 & 0 & 4 \end{bmatrix}$$

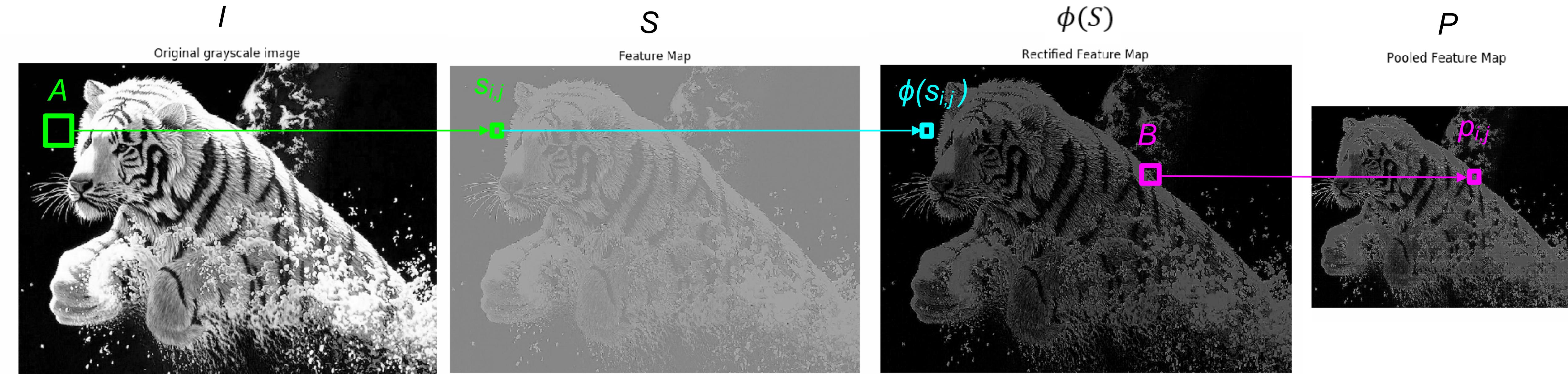
$$p_{3,3} = \max\left(\begin{bmatrix} 4 & 2 \\ 0 & 4 \end{bmatrix}\right) = 4$$

$$P = \begin{bmatrix} 4 & 4 & 2 \\ 4 & 4 & 4 \\ 4 & 4 & 4 \end{bmatrix}$$

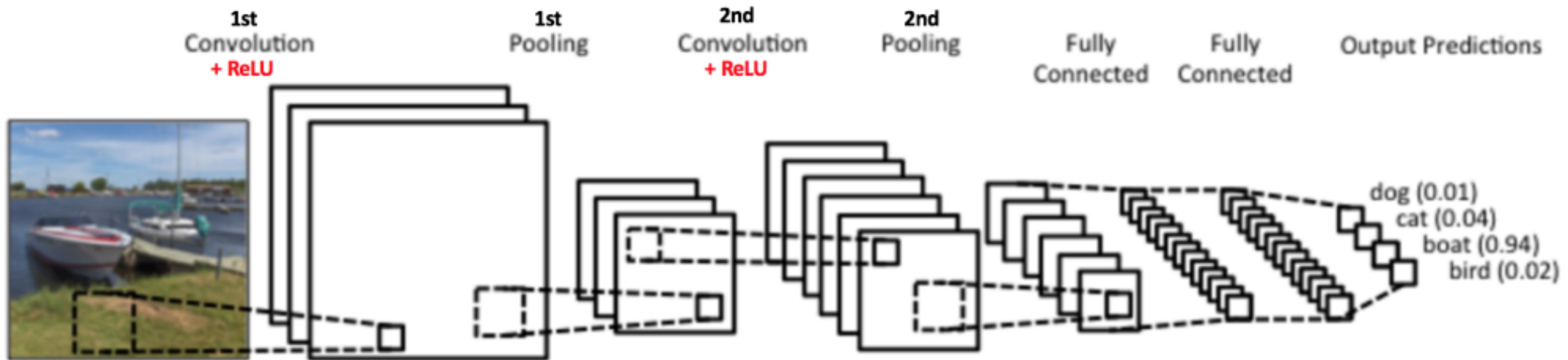
Putting the Pieces Together

$$W = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}, b = 0$$

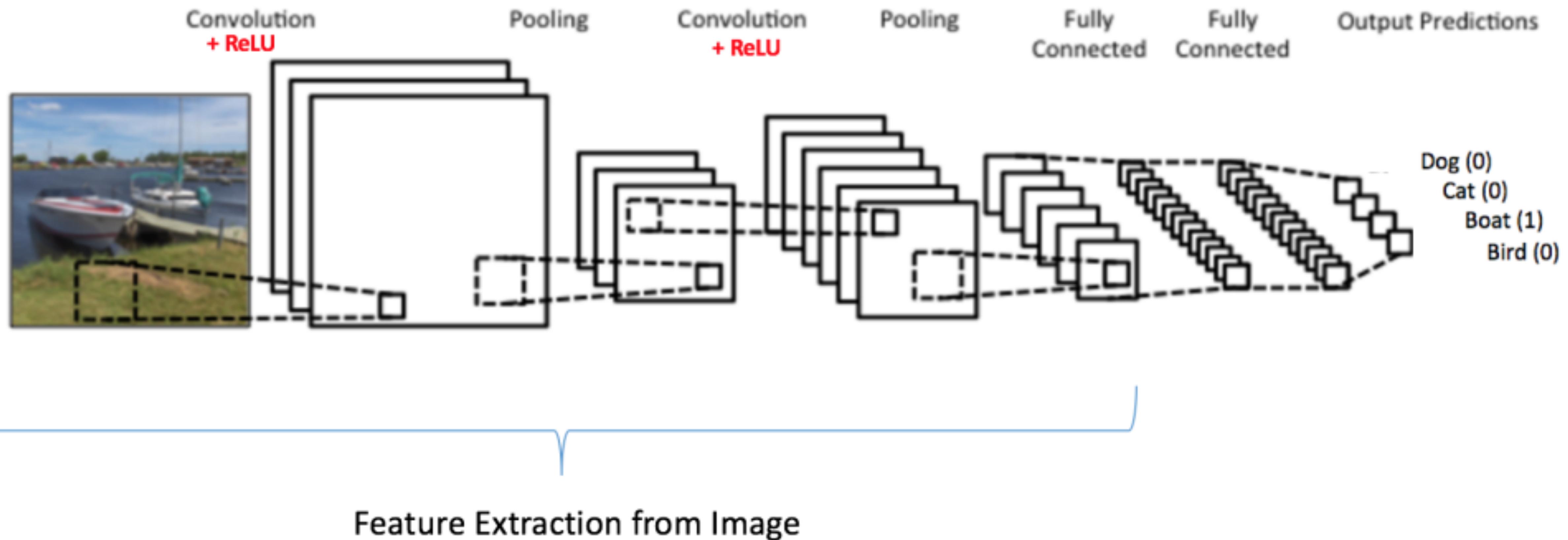
$$s_{i,j} = \sum_m \sum_n a_{m,n} w_{m,n} + b \quad \phi(s_{i,j}) = \max(0, s_{i,j}) \quad p_{i,j} = \max(B)$$



Putting the Pieces Together

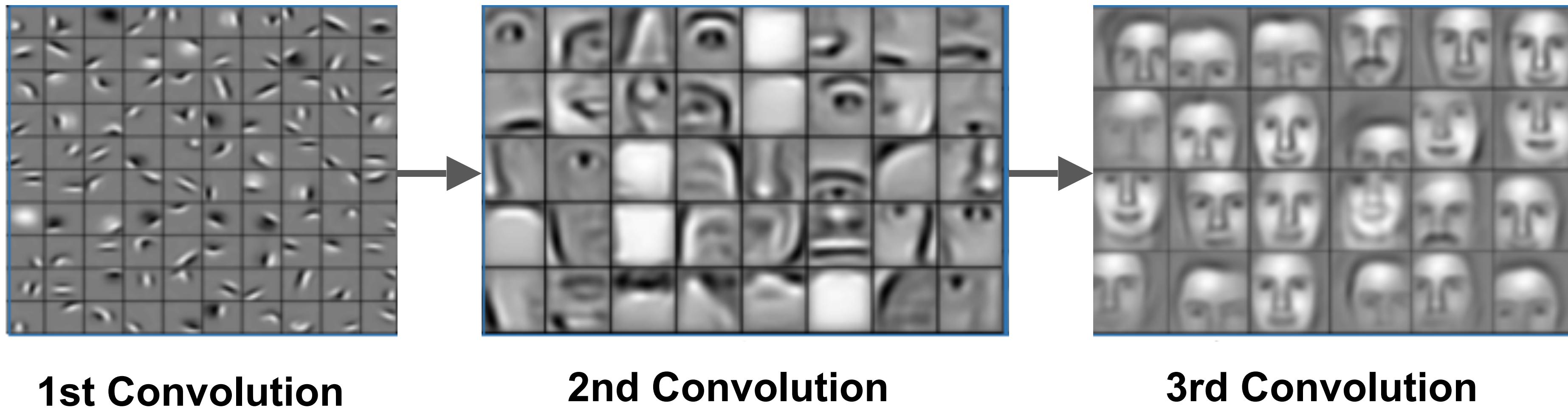


Putting the Pieces Together



Putting the Pieces Together: Feature Extractor

- Combining several convolutional layers
- The deeper in the network, the more complex structures of the image can be learnt



Putting the Pieces Together: Classification

- Fully-connected layers combined with softmax in the output layer

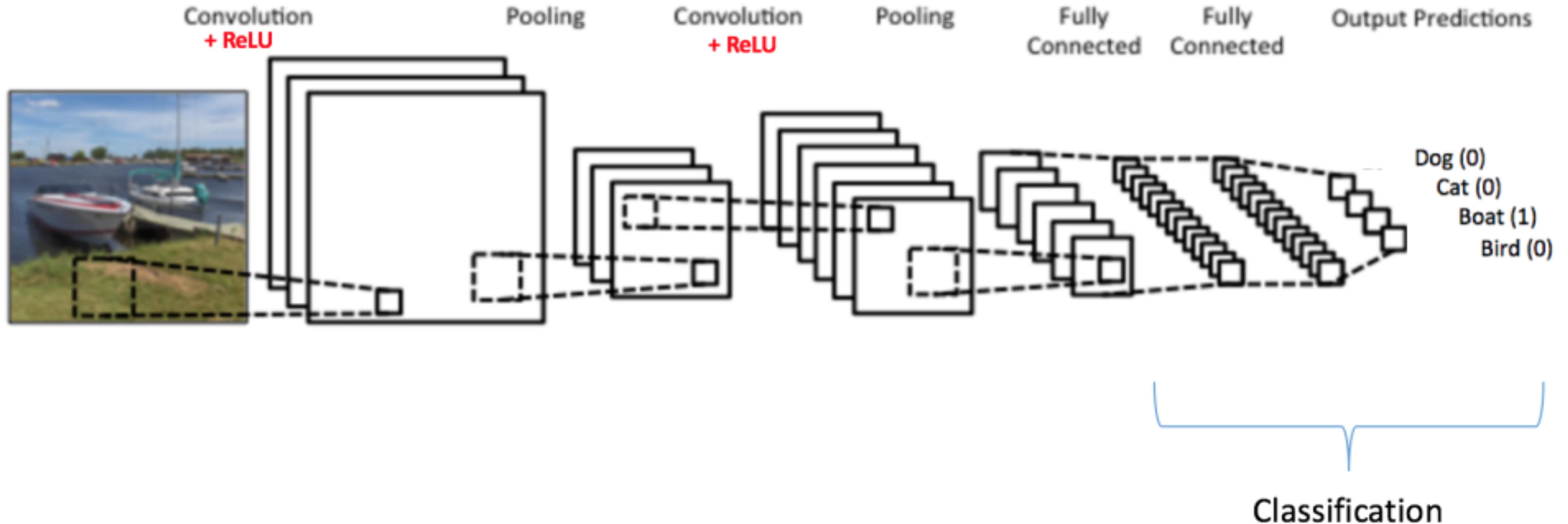


Image Classification

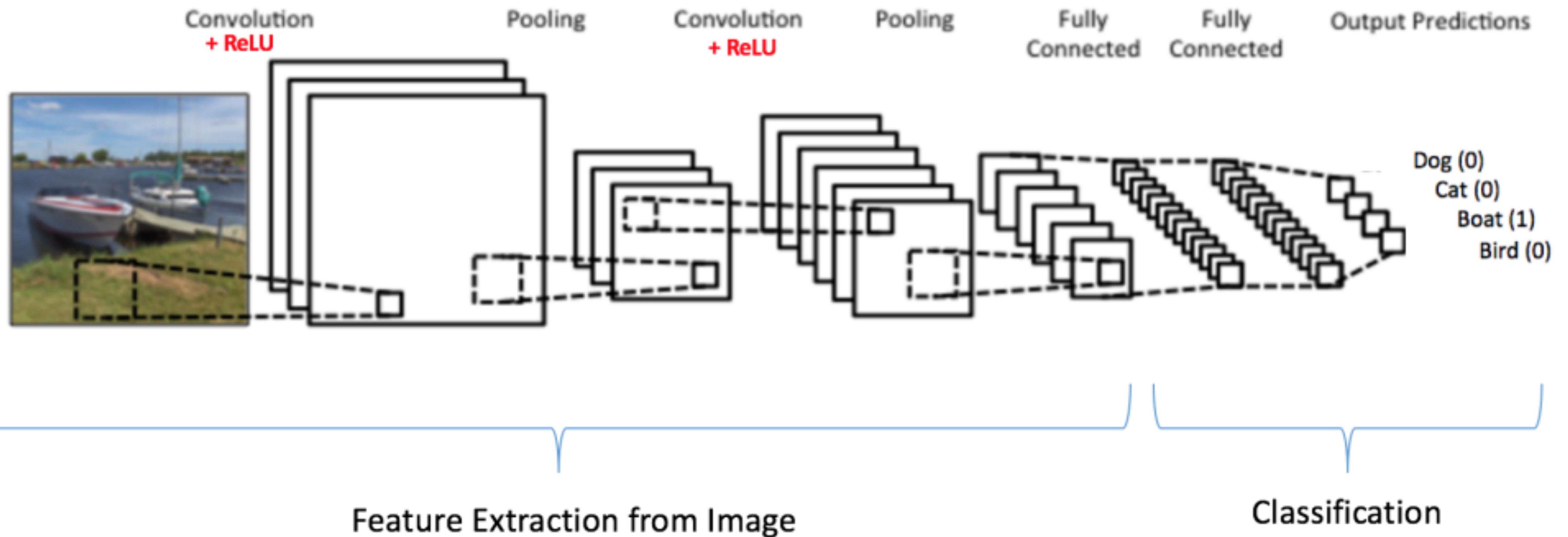


Image Classification: Example

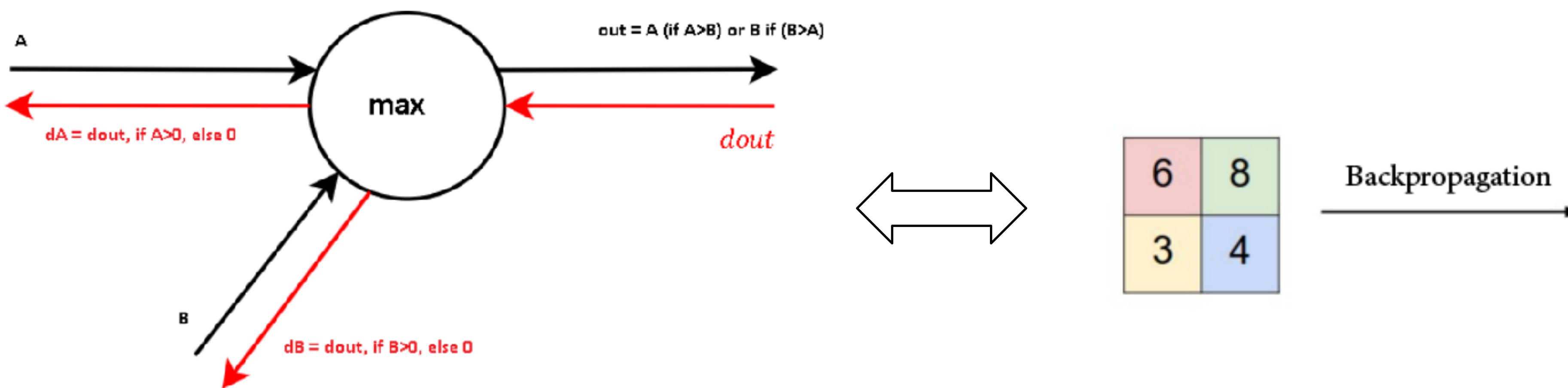
- MNIST
 - Predict what number (0-9) an image contains



CNNs: Learning

- Updates performed using backpropagation similarly to regular NNs
- Some modifications
 - Account for weight sharing, pooling

Backpropagation in case of max pooling



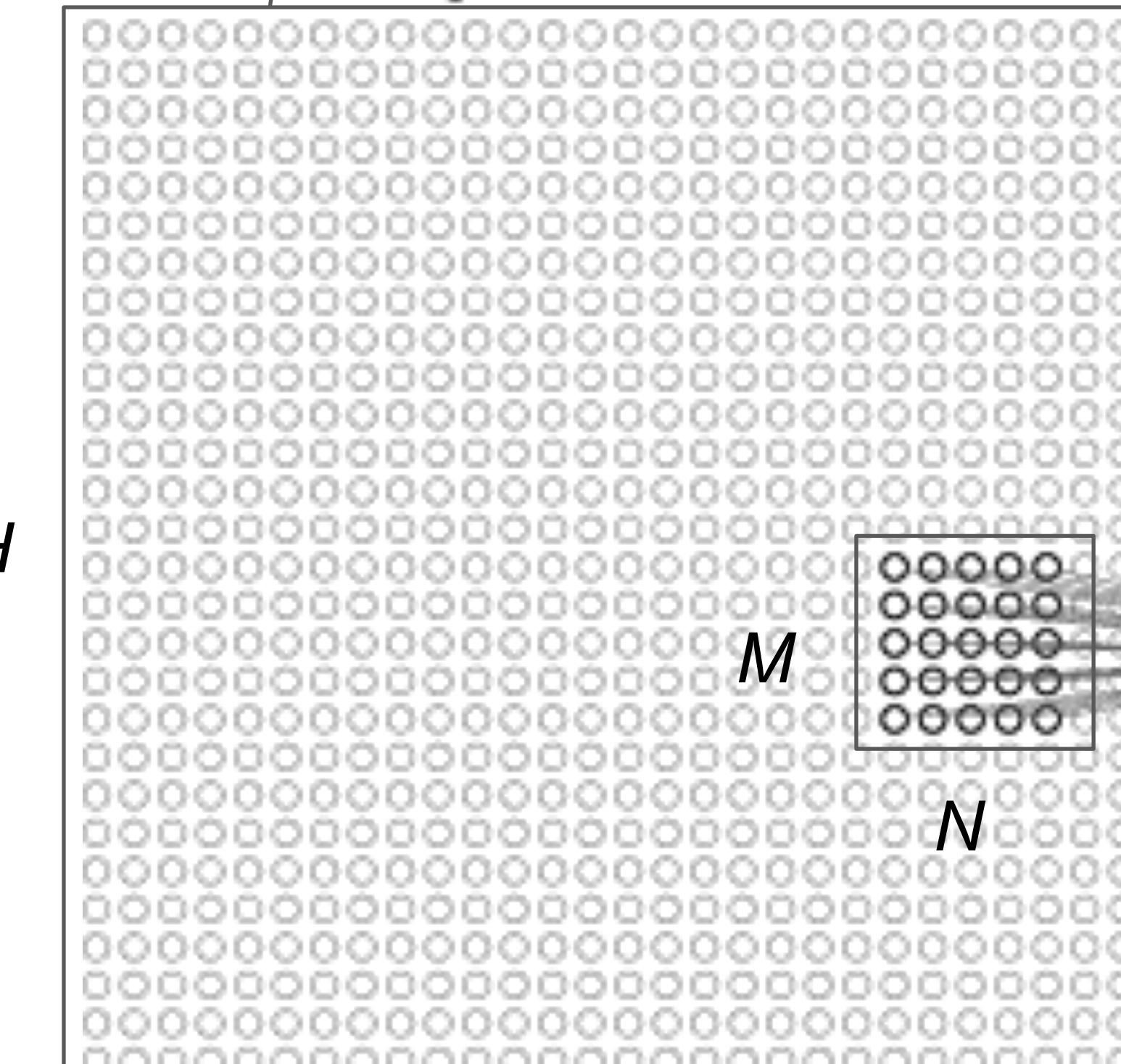
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

0	0	0	0
0	$dout$	0	$dout$
$dout$	0	0	0
0	0	0	$dout$

CNNs: Backpropagation

$$I \in R^{H \times W}$$

input neurons



- Given input map I , output map S , kernel W
- How to update a weight $w_{m',n'}$ in W ?

$$\frac{\partial E}{\partial w_{m',n'}} = \sum_{i=0}^{H-M+1} \sum_{j=0}^{W-N+1} \frac{\partial E}{\partial s_{i,j}} \frac{\partial s_{i,j}}{\partial w_{m',n'}}$$

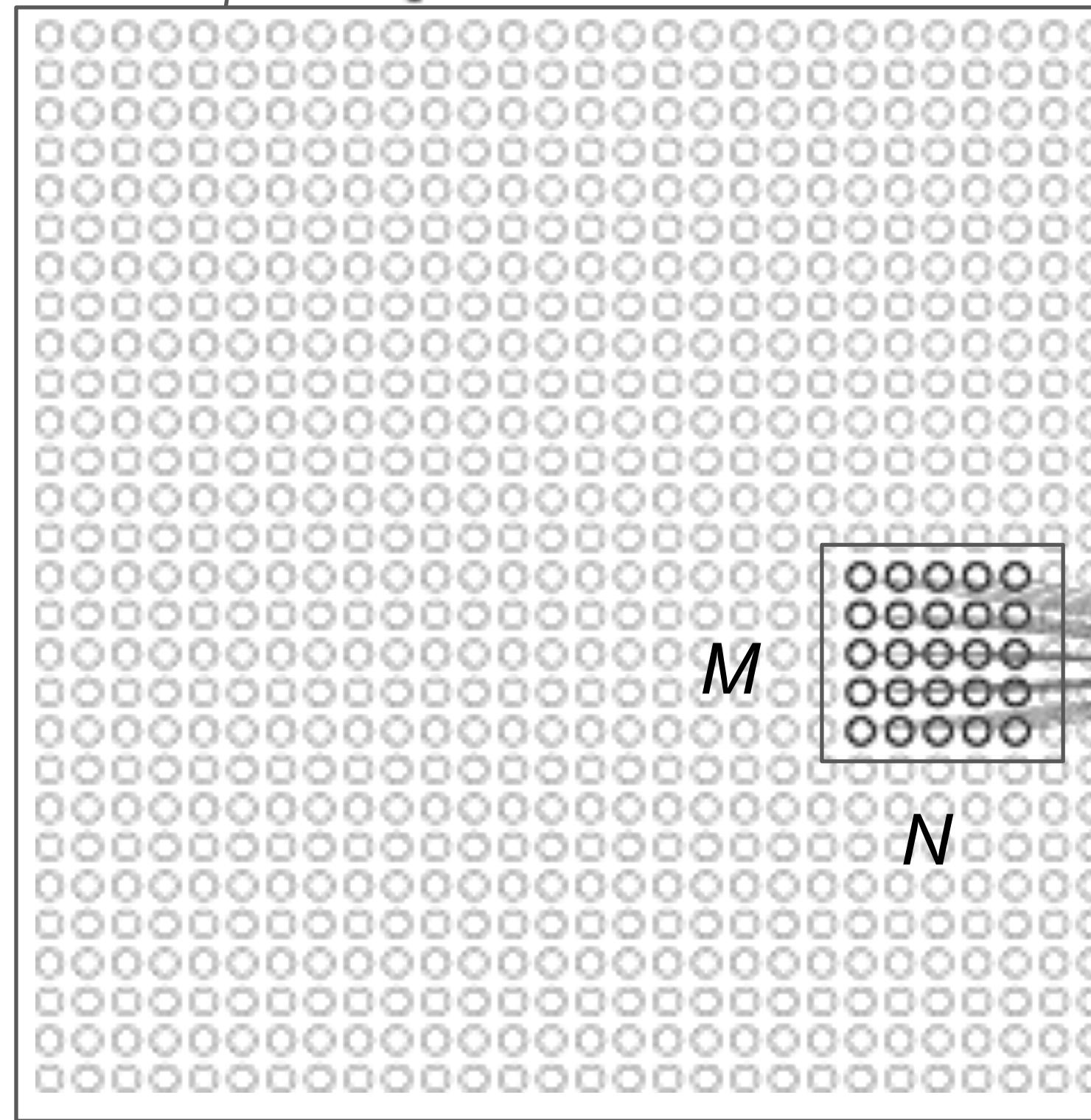
$\frac{\partial E}{\partial s_{i,j}}$ is known, how to calculate $\frac{\partial s_{i,j}}{\partial w_{m',n'}}$

$$\frac{\partial C}{\partial w_{kj}^L} = \frac{\partial C}{\partial a_k^L} f'(z_k^L) \frac{\partial z_k^L}{\partial w_{kj}^L} = \delta_k^L a_j^{L-1}$$

CNNs: Backpropagation (2)

$$I \in R^{H \times W}$$

input neurons



$$W \in R^{M \times N}$$

hidden neuron
 $s_{i,j}$

$$\frac{\partial C}{\partial w_{kj}^L} = \frac{\partial C}{\partial a_k^L} f'(z_k^L) \frac{\partial z_k^L}{\partial w_{kj}^L} = \delta_k^L a_j^{L-1}$$

- How to calculate $\frac{\partial s_{i,j}}{\partial w_{m',n'}}$?

$$s_{i,j} = (I * W)_{i,j} + b = \sum_{m=1}^M \sum_{n=1}^N i_{i+m,j+n} w_{m,n} + b$$

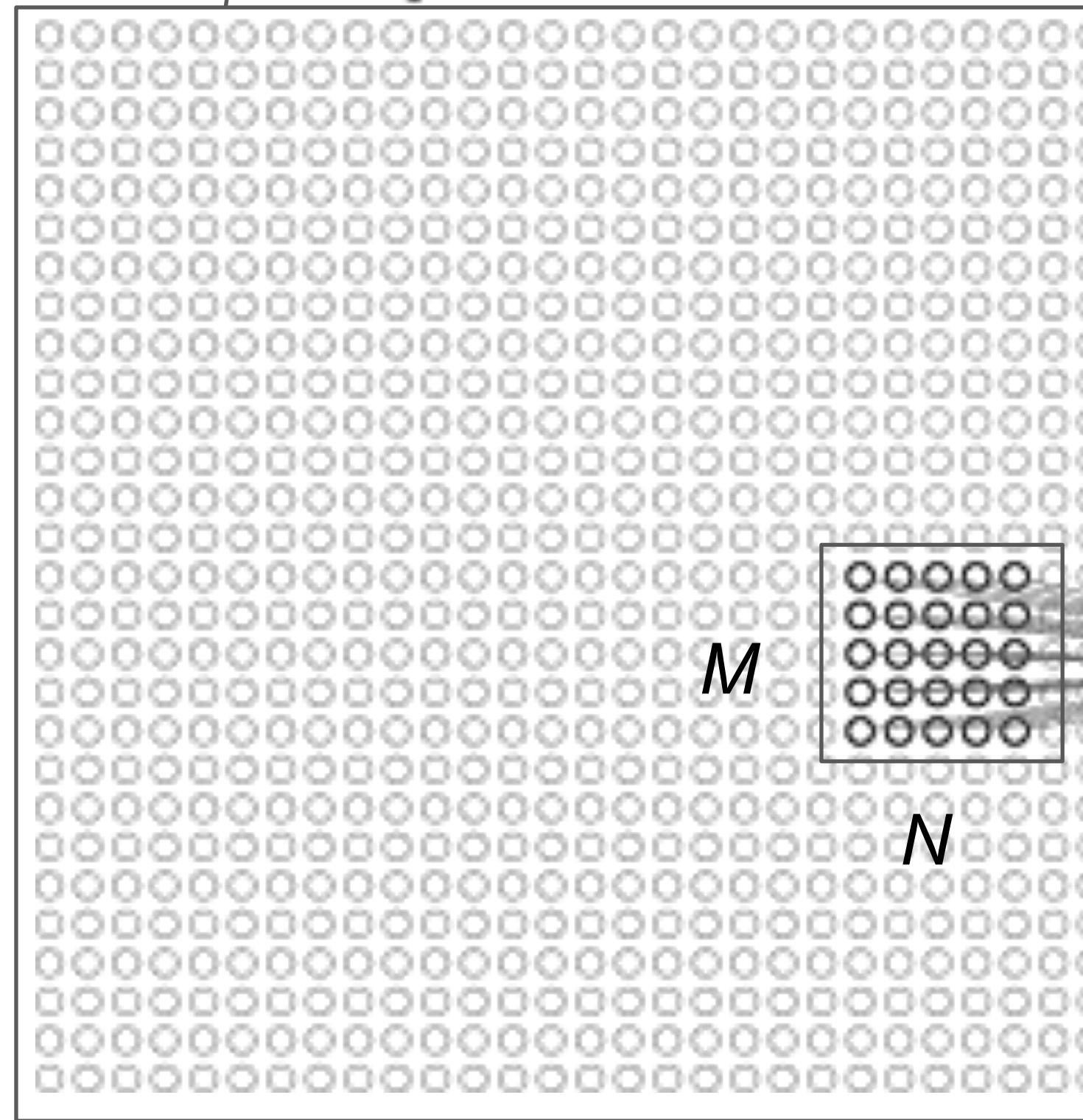
$$\frac{\partial s_{i,j}}{\partial w_{m',n'}} = \frac{\partial}{\partial w_{m',n'}} \sum_m \sum_n i_{i+m,j+n} w_{m,n} + b$$

$$= \frac{\partial}{\partial w_{m',n'}} (i_{i+m',j+n'} w_{m',n'}) = i_{i+m',j+n'}$$

CNNs: Backpropagation (3)

$$I \in R^{H \times W}$$

input neurons



$$W \in R^{M \times N}$$

hidden neuron
 $s_{i,j}$

$$\frac{\partial C}{\partial w_{kj}^L} = \frac{\partial C}{\partial a_k^L} f'(z_k^L) \frac{\partial z_k^L}{\partial w_{kj}^L} = \delta_k^L a_j^{L-1}$$

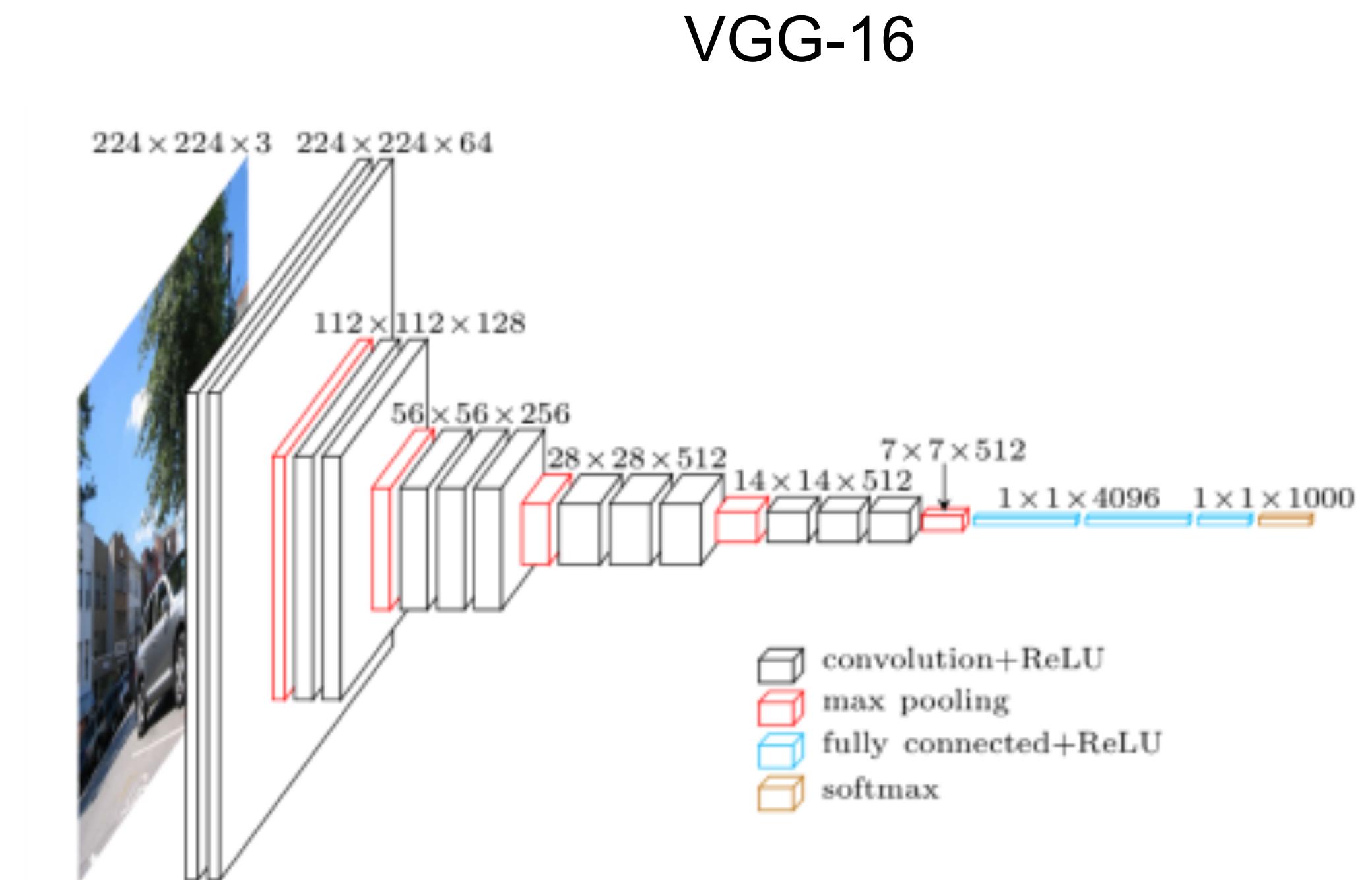
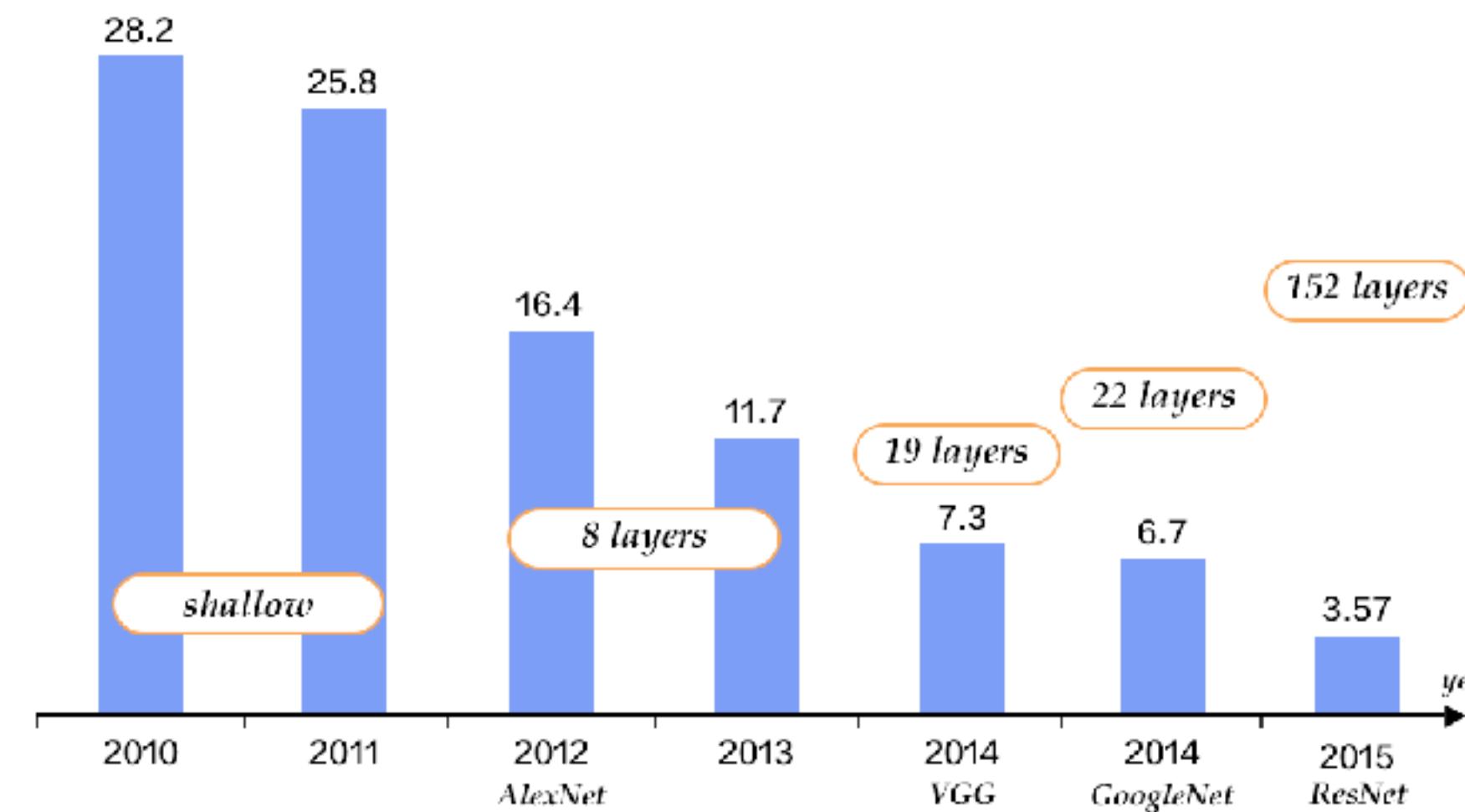
- We know that $\frac{\partial s_{i,j}}{\partial w_{m',n'}} = i_{i+m',j+n'}$

$$\frac{\partial E}{\partial w_{m',n'}} = \sum_{i=1}^{H-M+1} \sum_{j=1}^{W-N+1} \frac{\partial E}{\partial s_{i,j}} \frac{\partial s_{i,j}}{\partial w_{m',n'}}$$

$$= \sum_{i=1}^{H-M+1} \sum_{j=1}^{W-N+1} \frac{\partial E}{\partial s_{i,j}} i_{i+m',j+n'}$$

Recent Progress

- GPUs accelerates training of deep neural networks
 - Parallel computations
- Larger datasets
 - ImageNet = Millions of images
- ImageNet competition (ILSVRC)
 - Development of high performing CNNs
 - From AlexNet to ResNet



Recent Progress (2)

- More complex tasks
 - Predict 1000 categories
 - Localization, Object detection and Segmentation

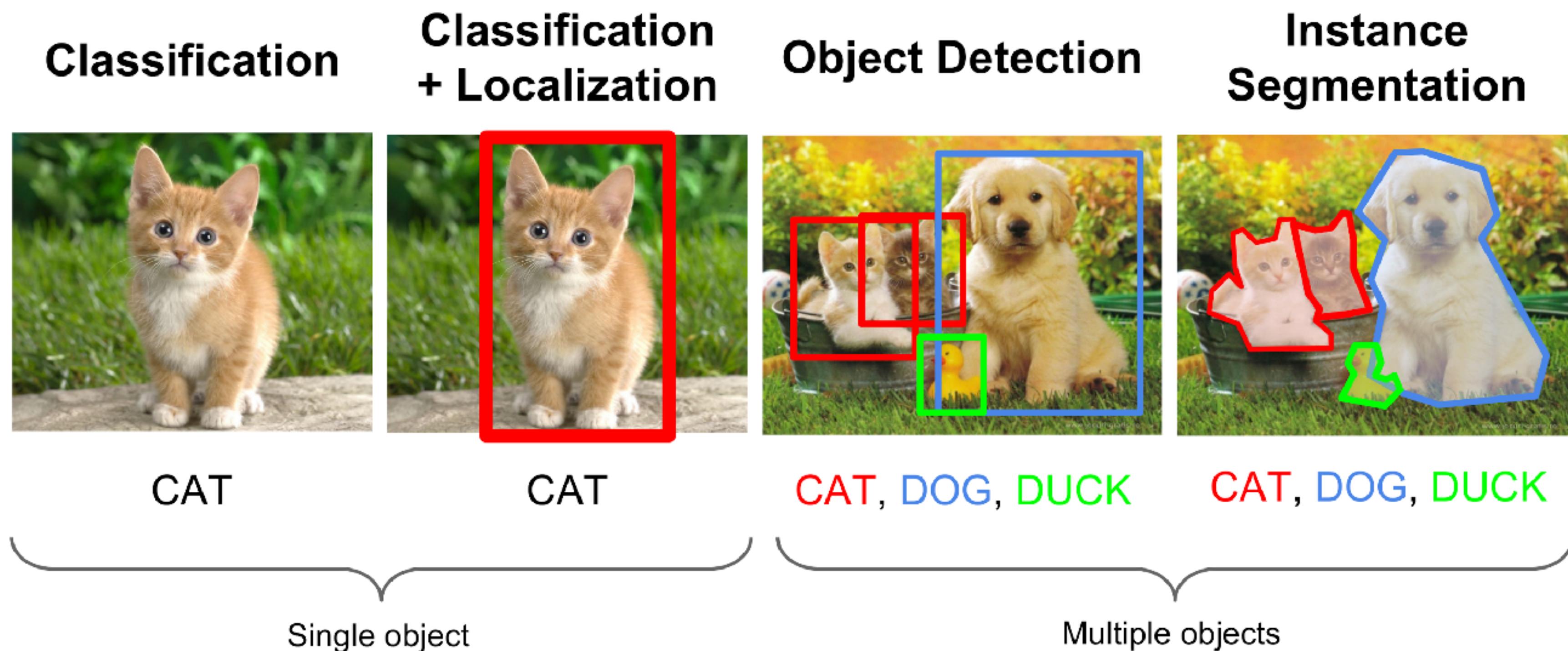
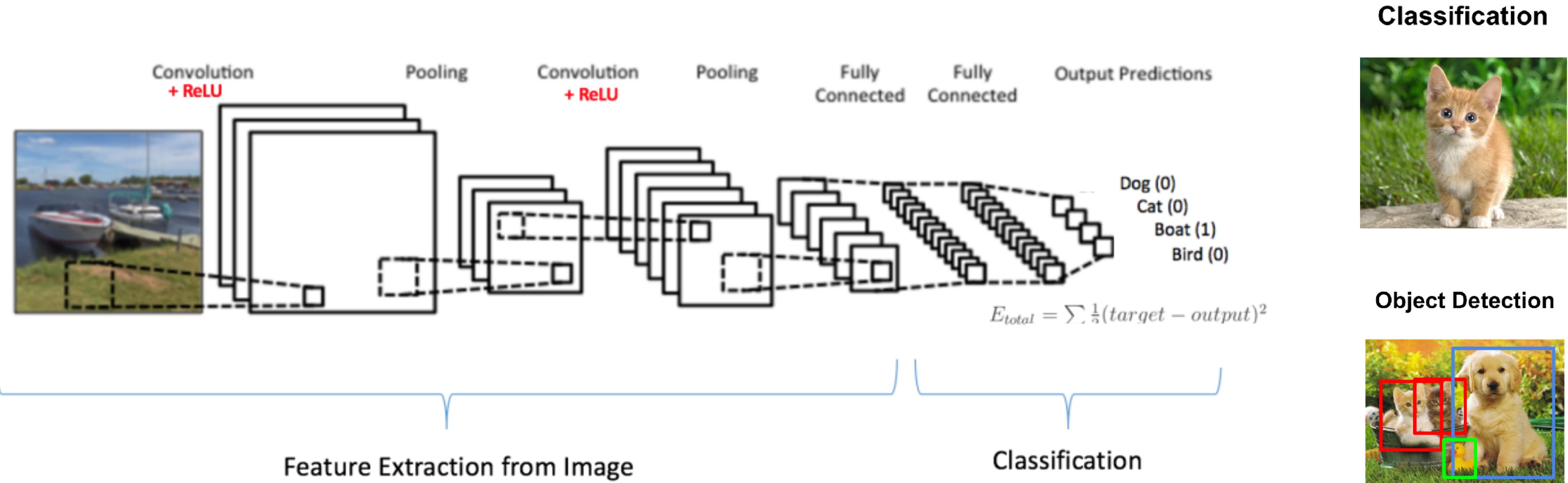


Image Classification (base net)

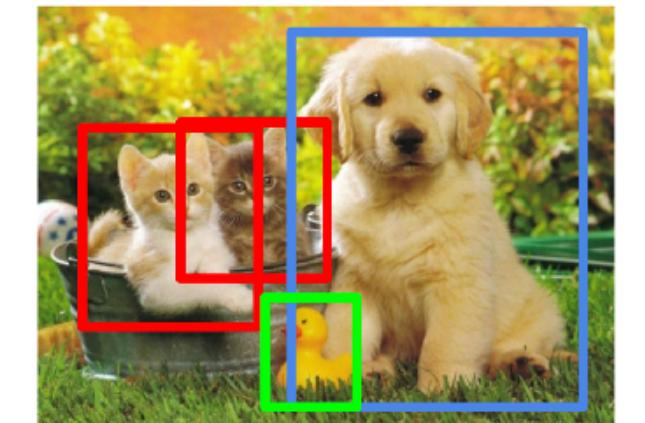


Object detection: Faster R-CNN (SSD, YOLO2, etc.)

Image segmentation: Mask R-CNN (FCN, U-net, etc.)



Object Detection



Instance Segmentation



Next time..

Practical