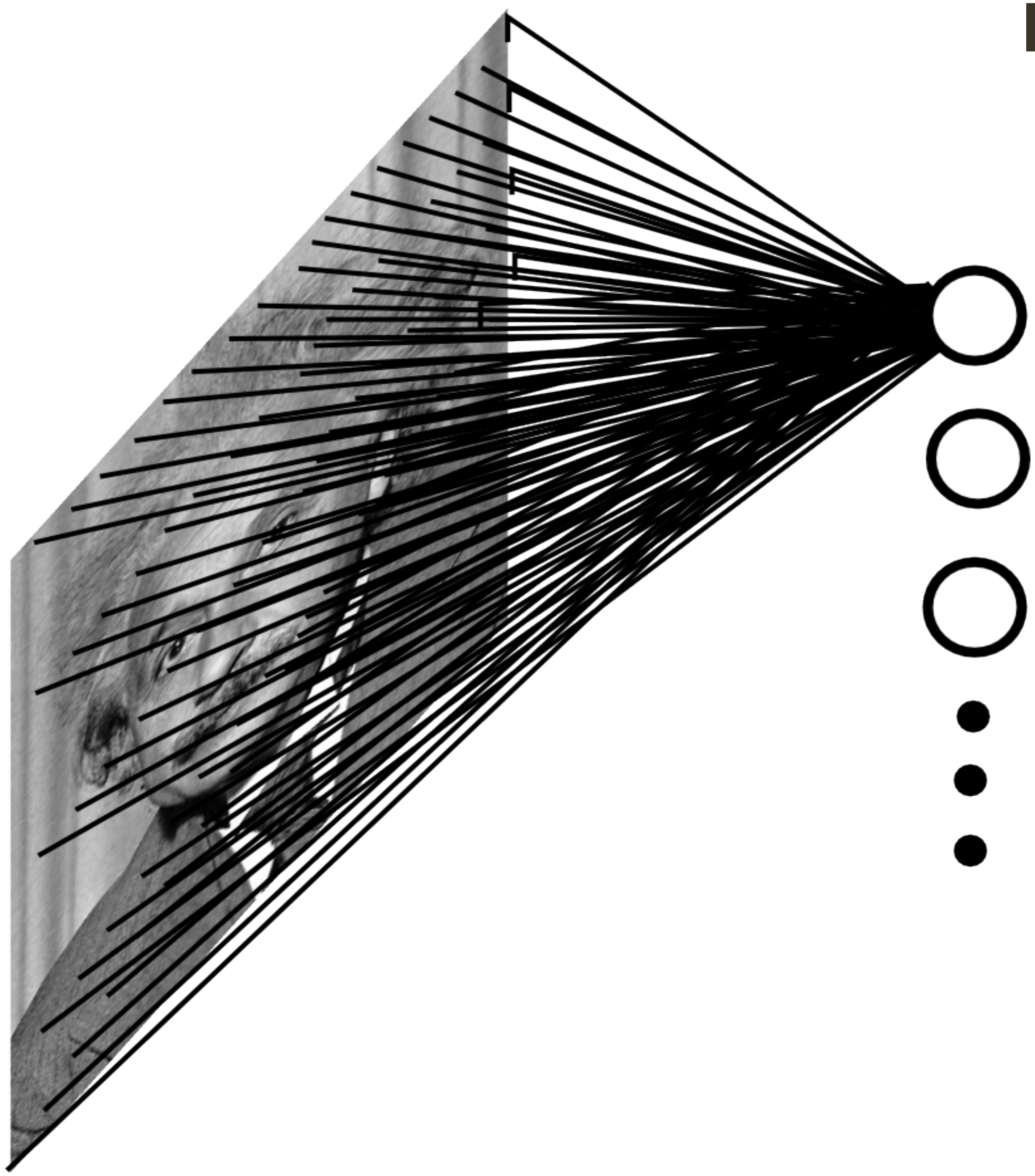
THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 4: Convolutional Neural Networks (Part 1)

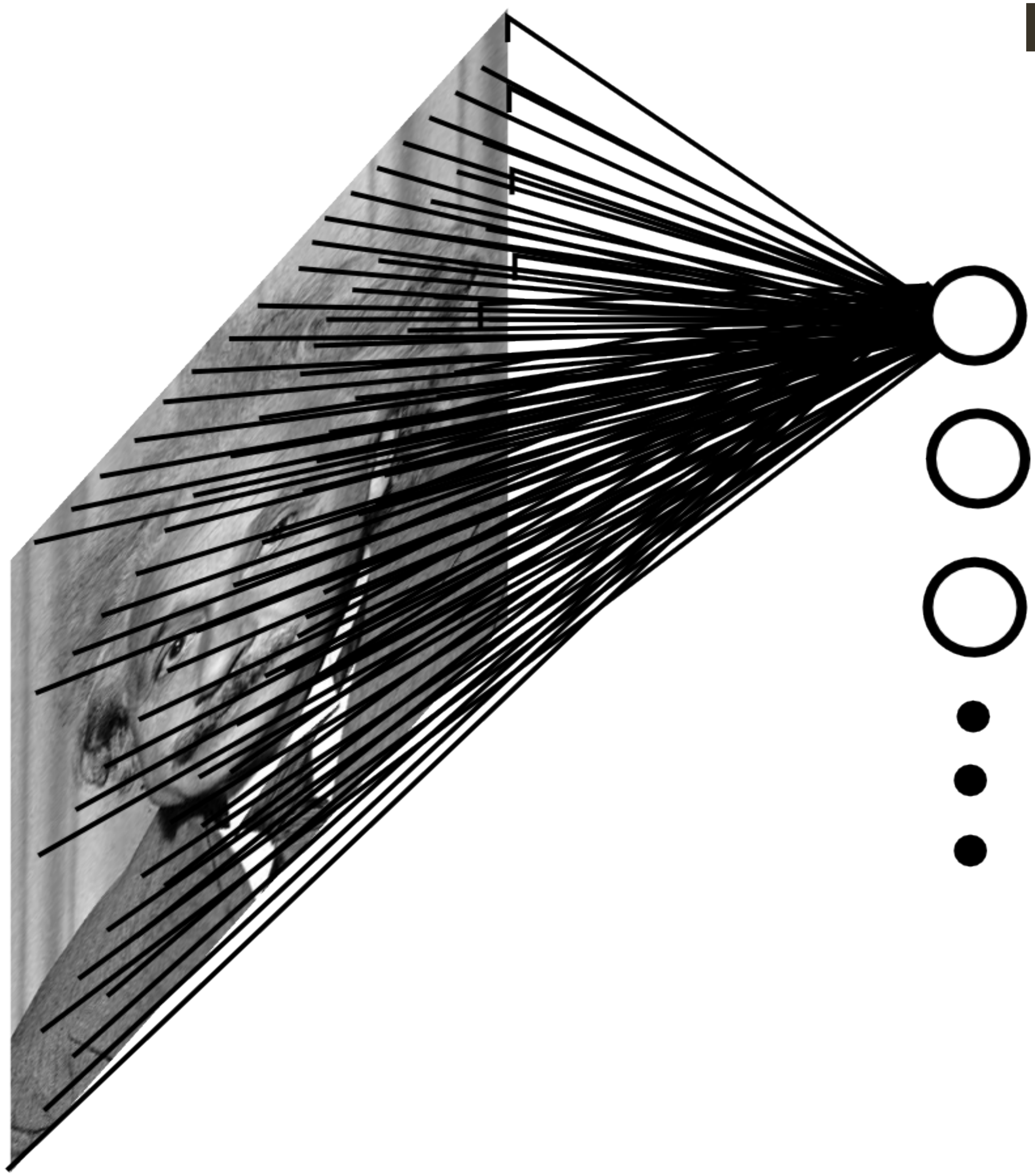
Fully Connected Layer



Example: 200 x 200 image (small)
x 40K hidden units

* slide from Marc'Aurelio Renzato

Fully Connected Layer

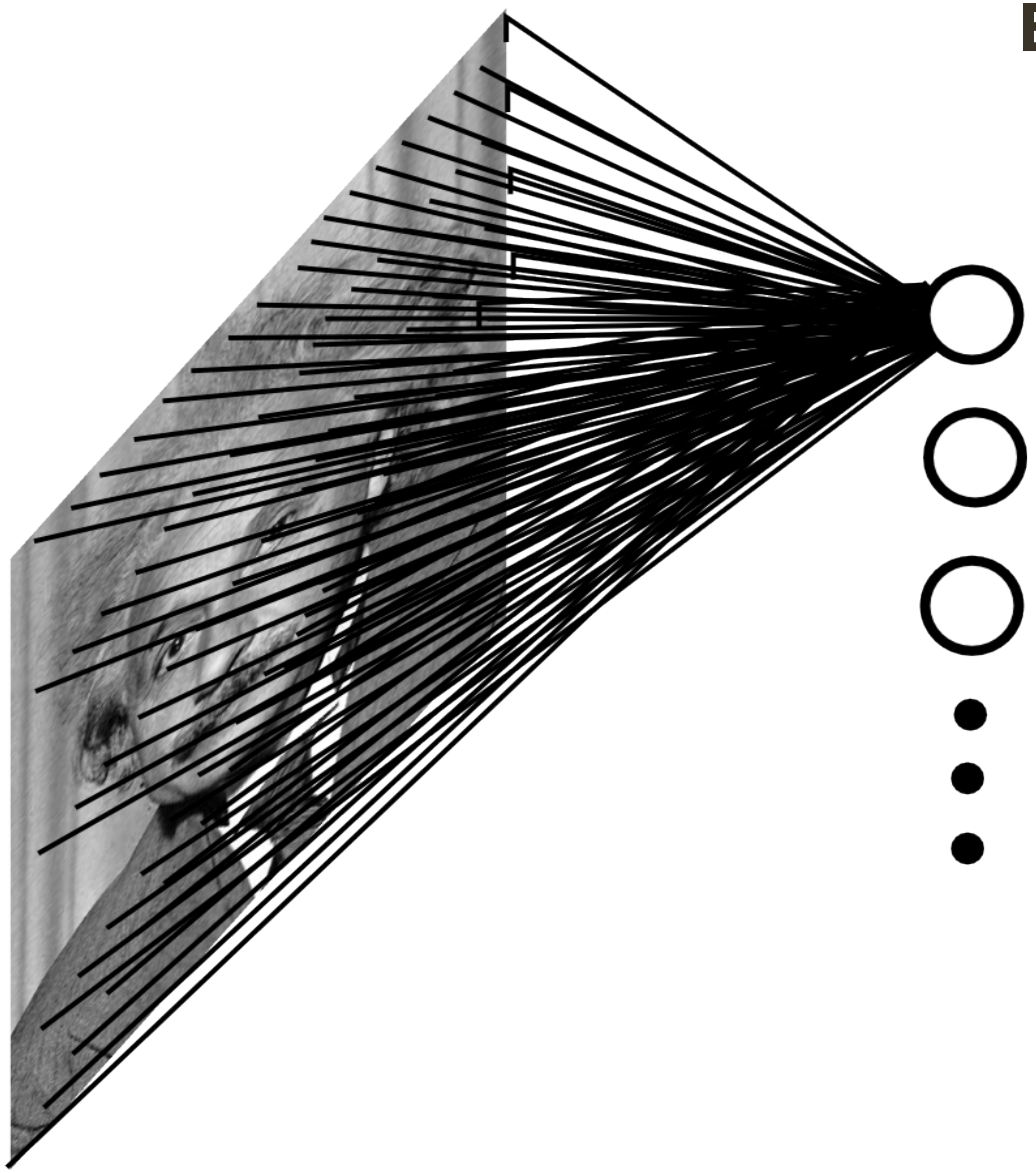


Example: 200 x 200 image (small)
x 40K hidden units

= ~ **2 Billion** parameters (for one layer!)

* slide from Marc'Aurelio Renzato

Fully Connected Layer



Example: 200 x 200 image (small)
x 40K hidden units

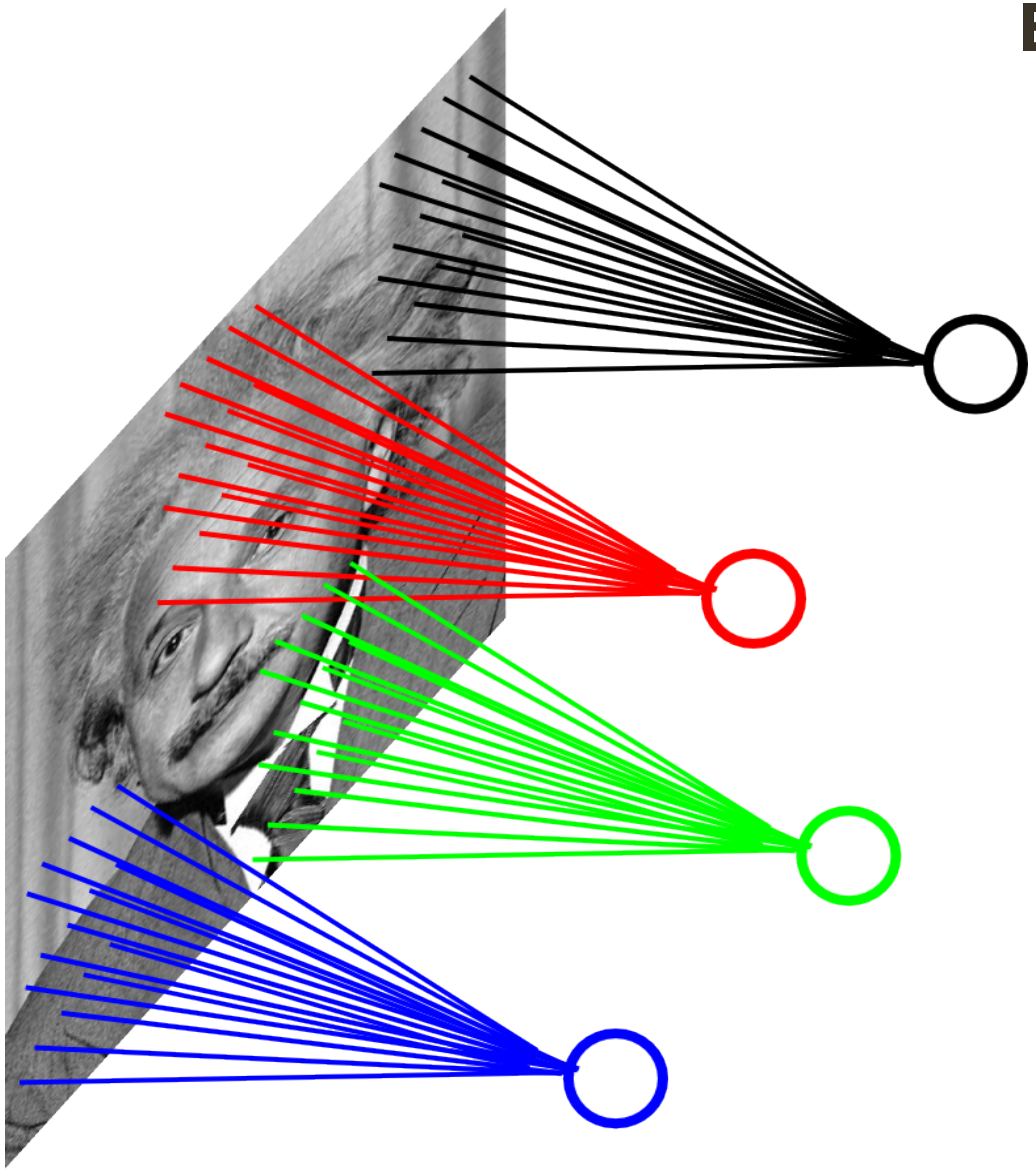
= ~ **2 Billion** parameters (for one layer!)

Spatial correlations are generally local

Waste of resources + we don't have enough data to train networks this large

* slide from Marc'Aurelio Renzato

Locally Connected Layer



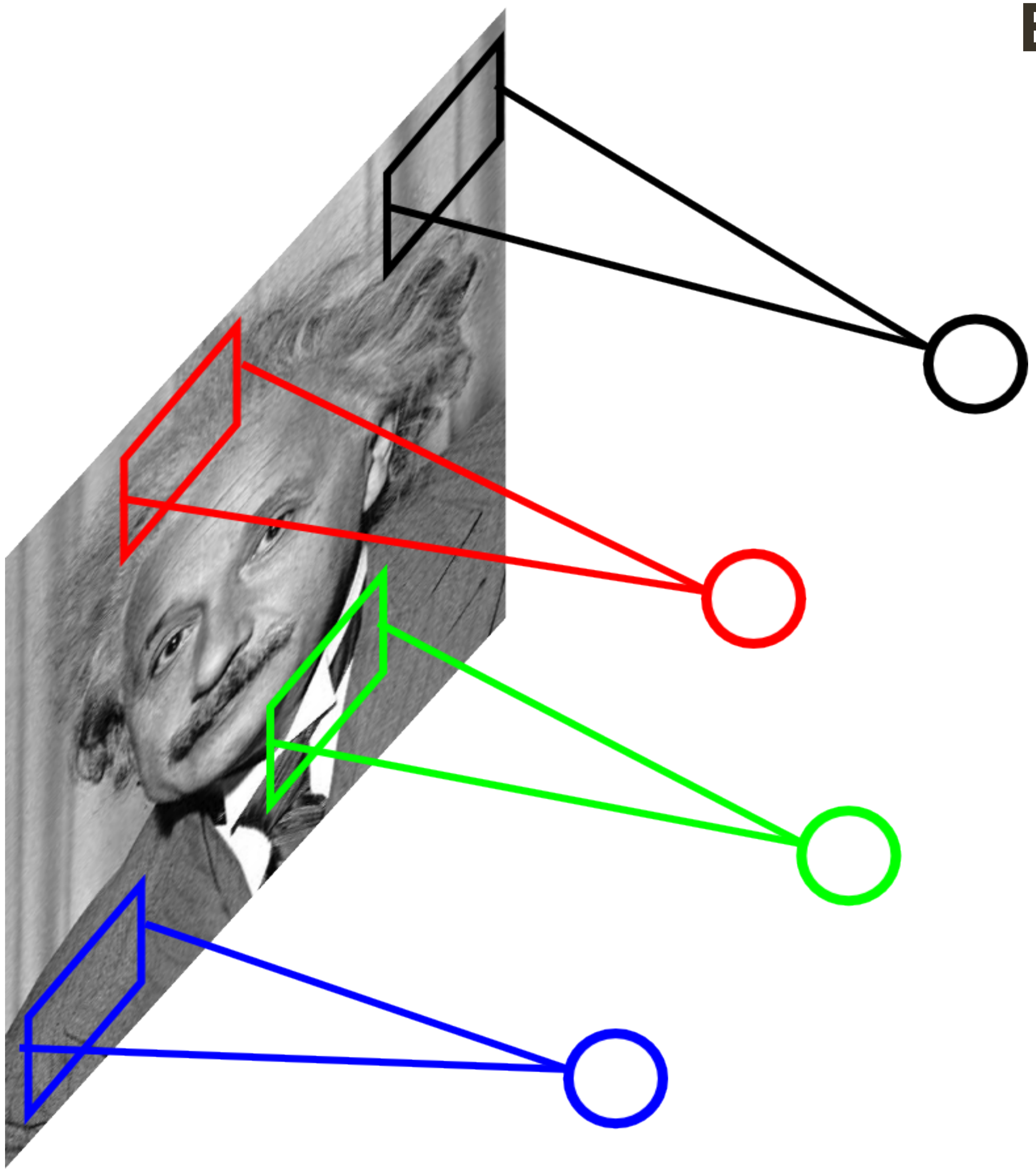
Example: 200 x 200 image (small)

Filter size: 10 x 10

= ~ **4 Million** parameters

* slide from Marc'Aurelio Renzato

Locally Connected Layer



Example: 200 x 200 image (small)

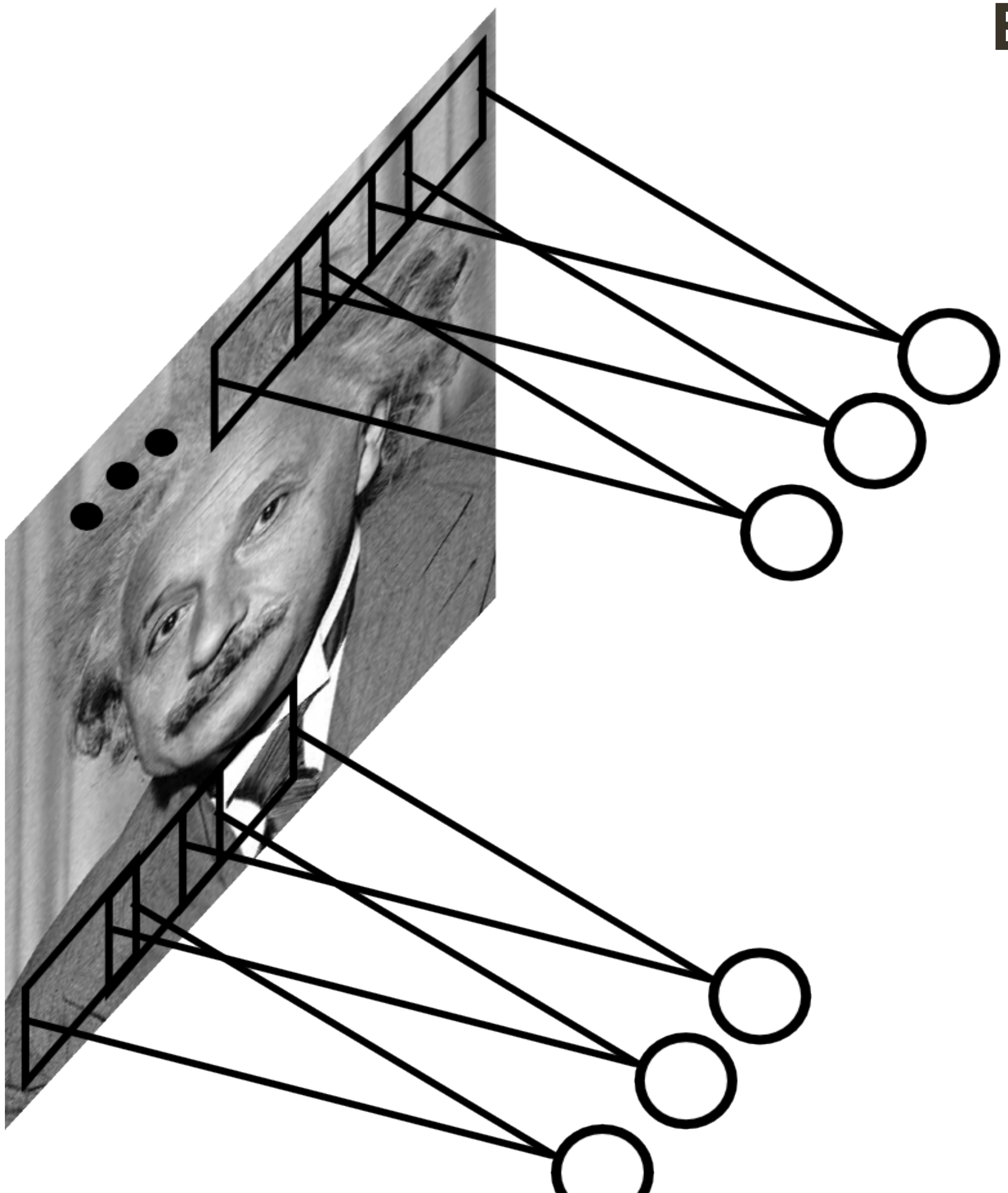
Filter size: 10 x 10

= ~ **4 Million** parameters

Stationarity — statistics is similar at different locations

* slide from Marc'Aurelio Renzato

Convolutional Layer



Example: 200 x 200 image (small)

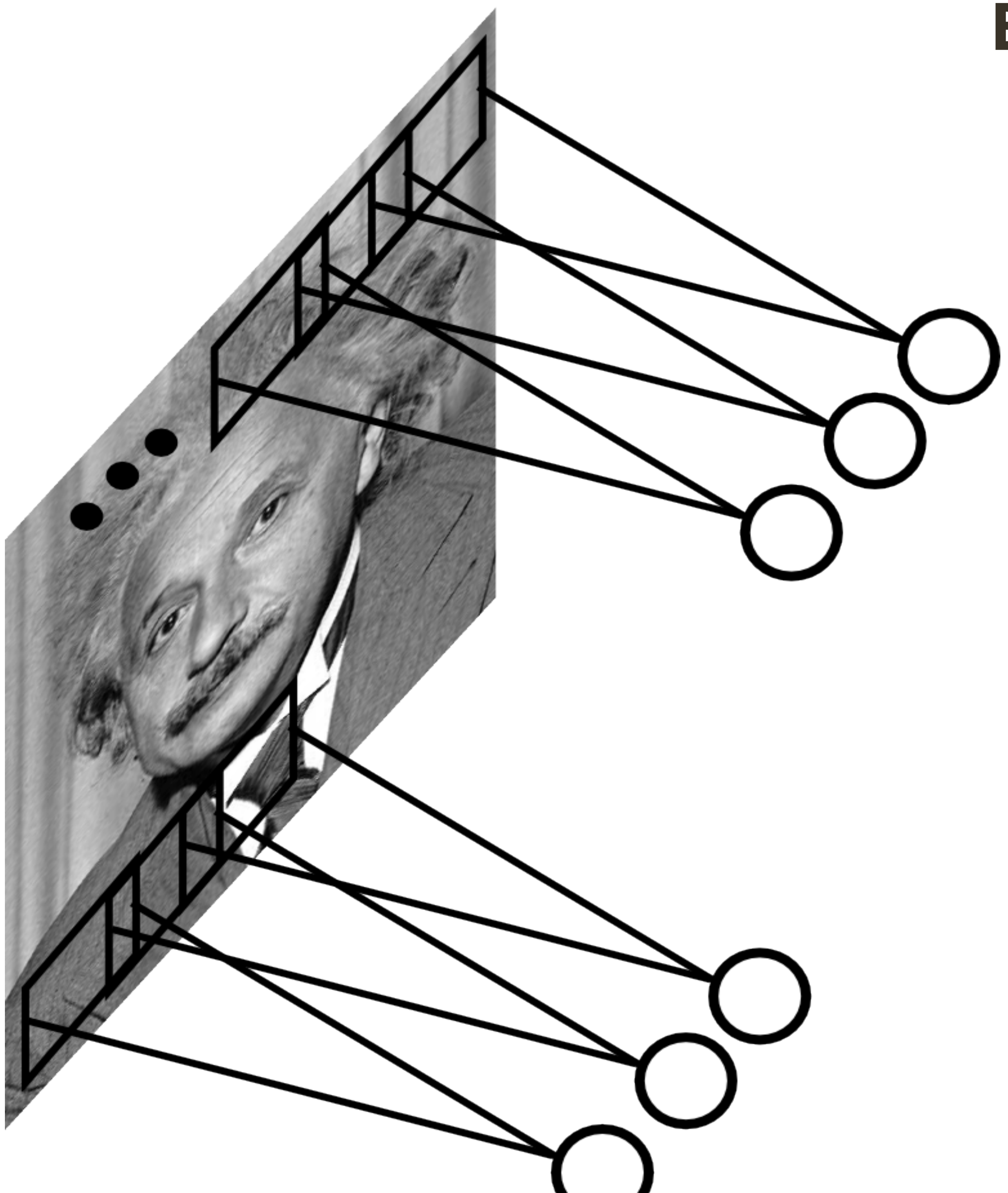
Filter size: 10 x 10

= ~ **4 Million** parameters

Share the same parameters across the locations (assuming input is stationary)

* slide adopted from Marc'Aurelio Renzato

Convolutional Layer



Example: 200 x 200 image (small)

Filter size: 10 x 10

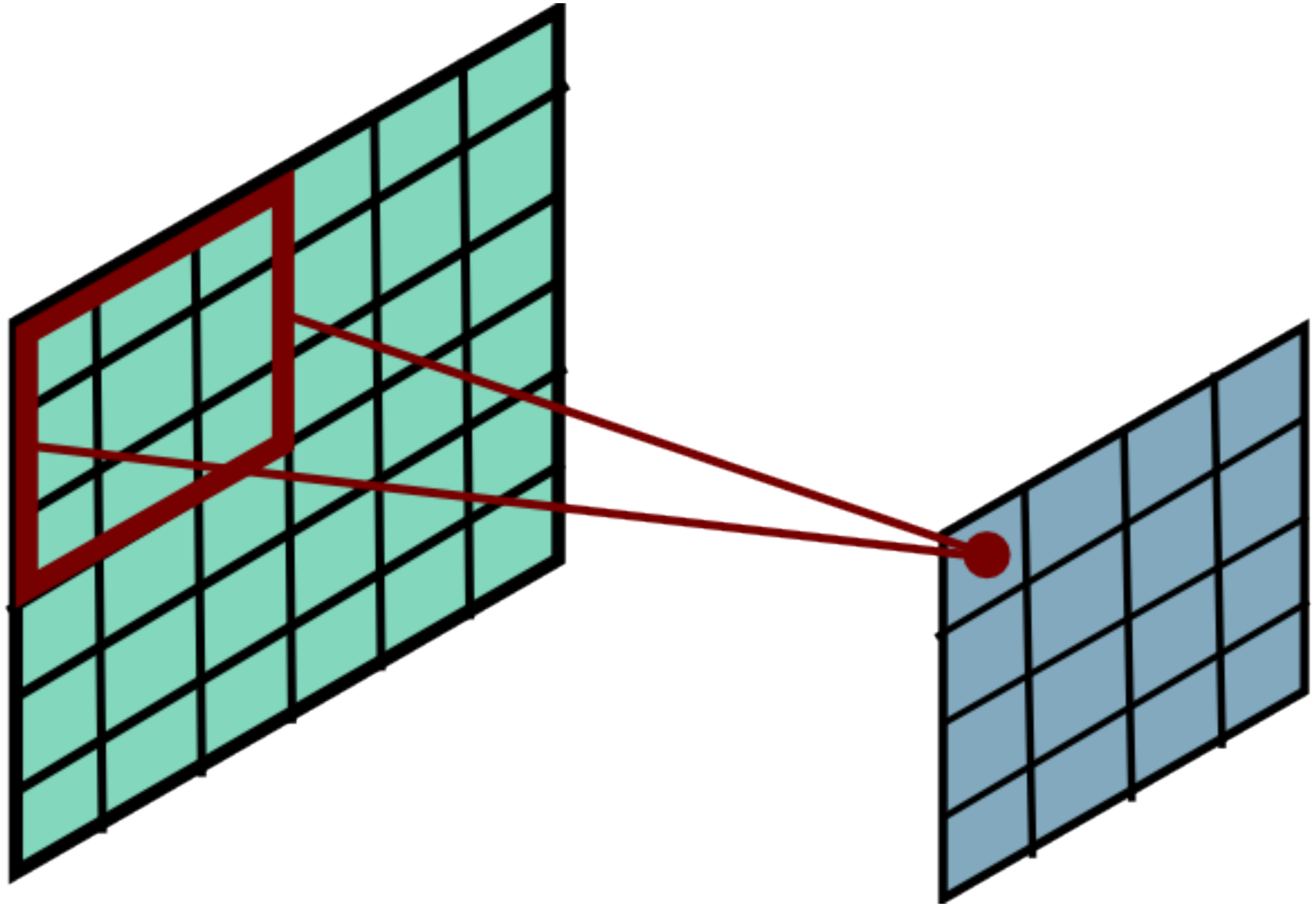
= ~ **~~4 Million~~** parameters

= 100 parameters

Share the same parameters across the locations (assuming input is stationary)

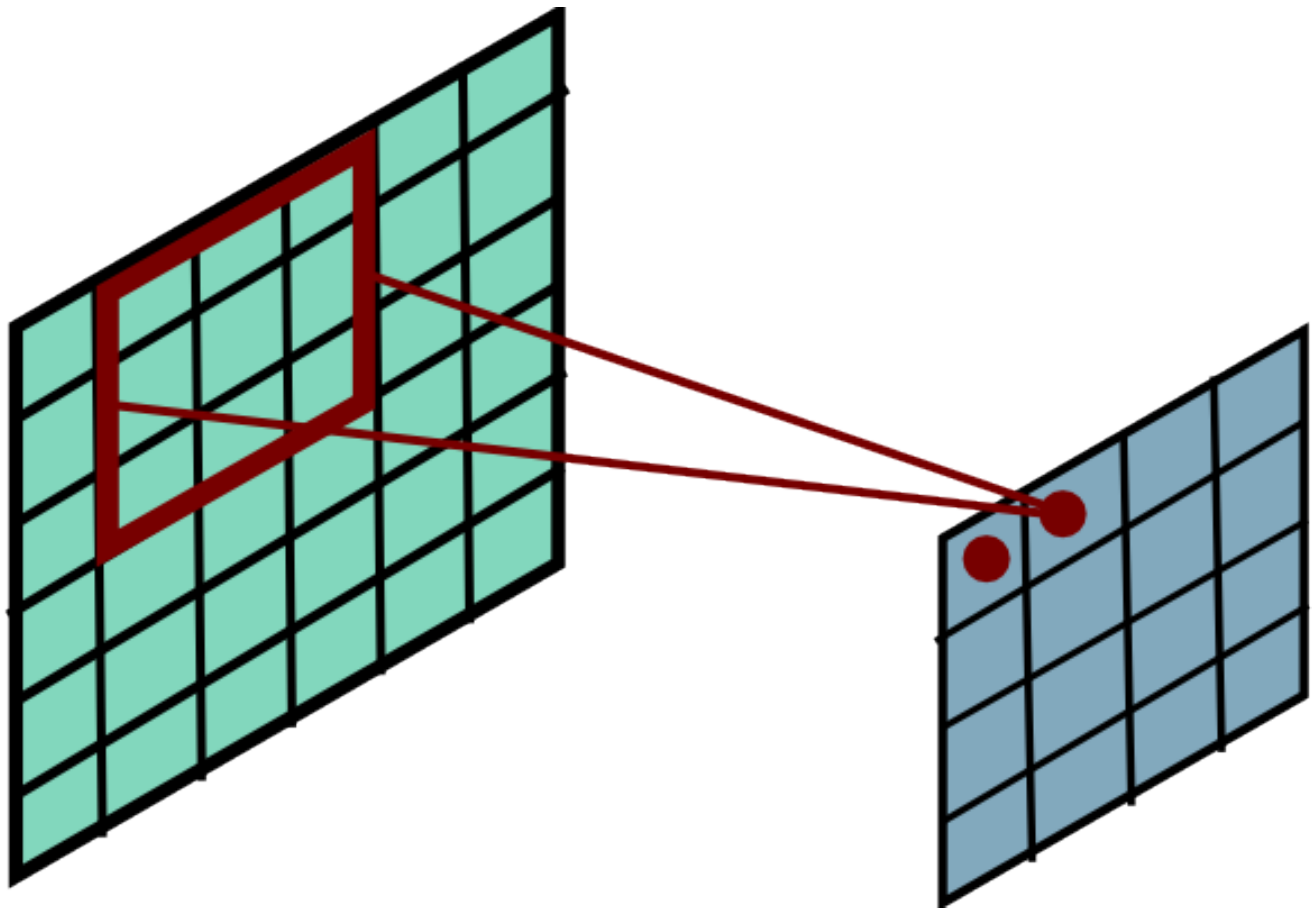
* slide adopted from Marc'Aurelio Renzato

Convolutional Layer



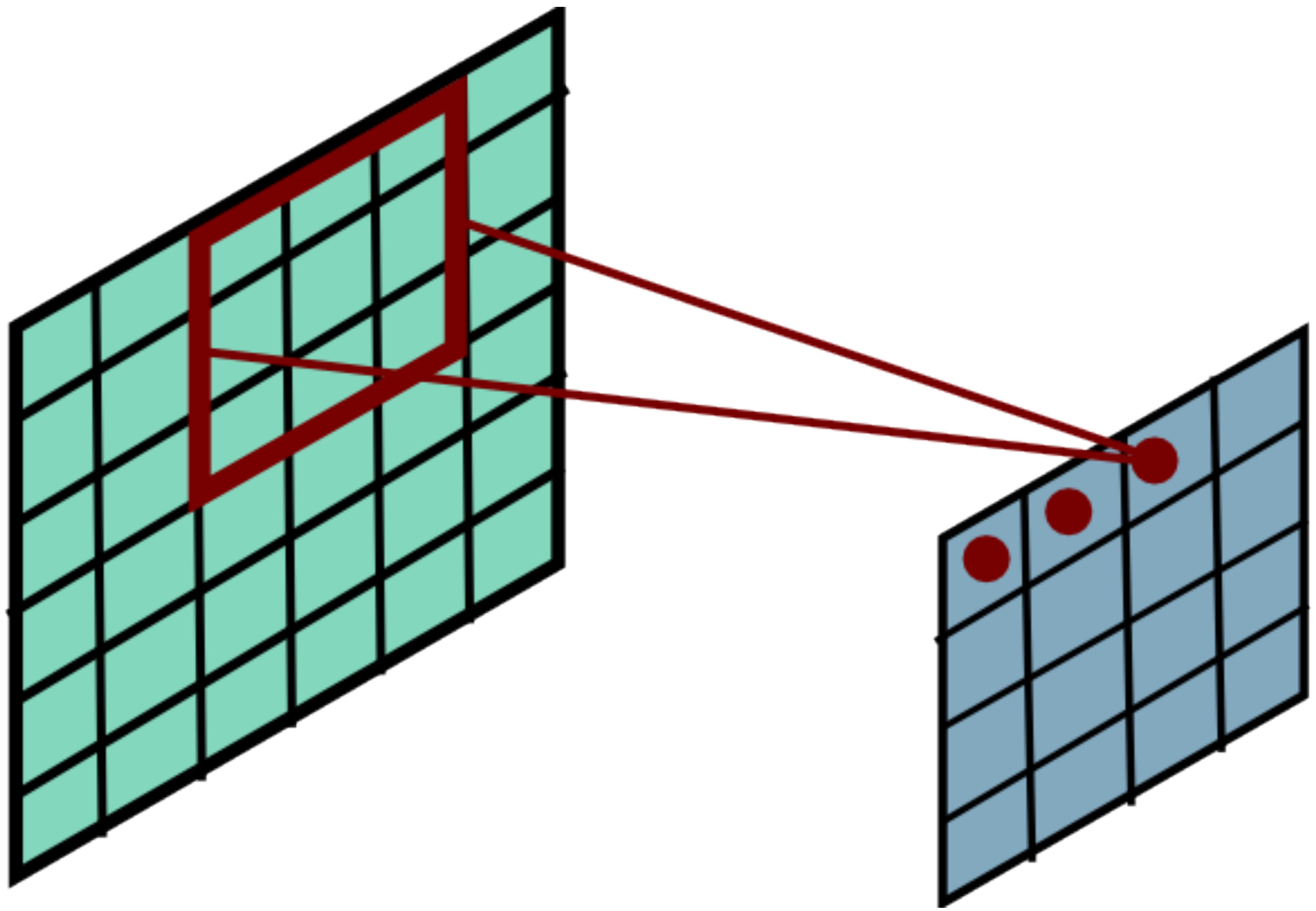
* slide from Marc'Aurelio Renzato

Convolutional Layer



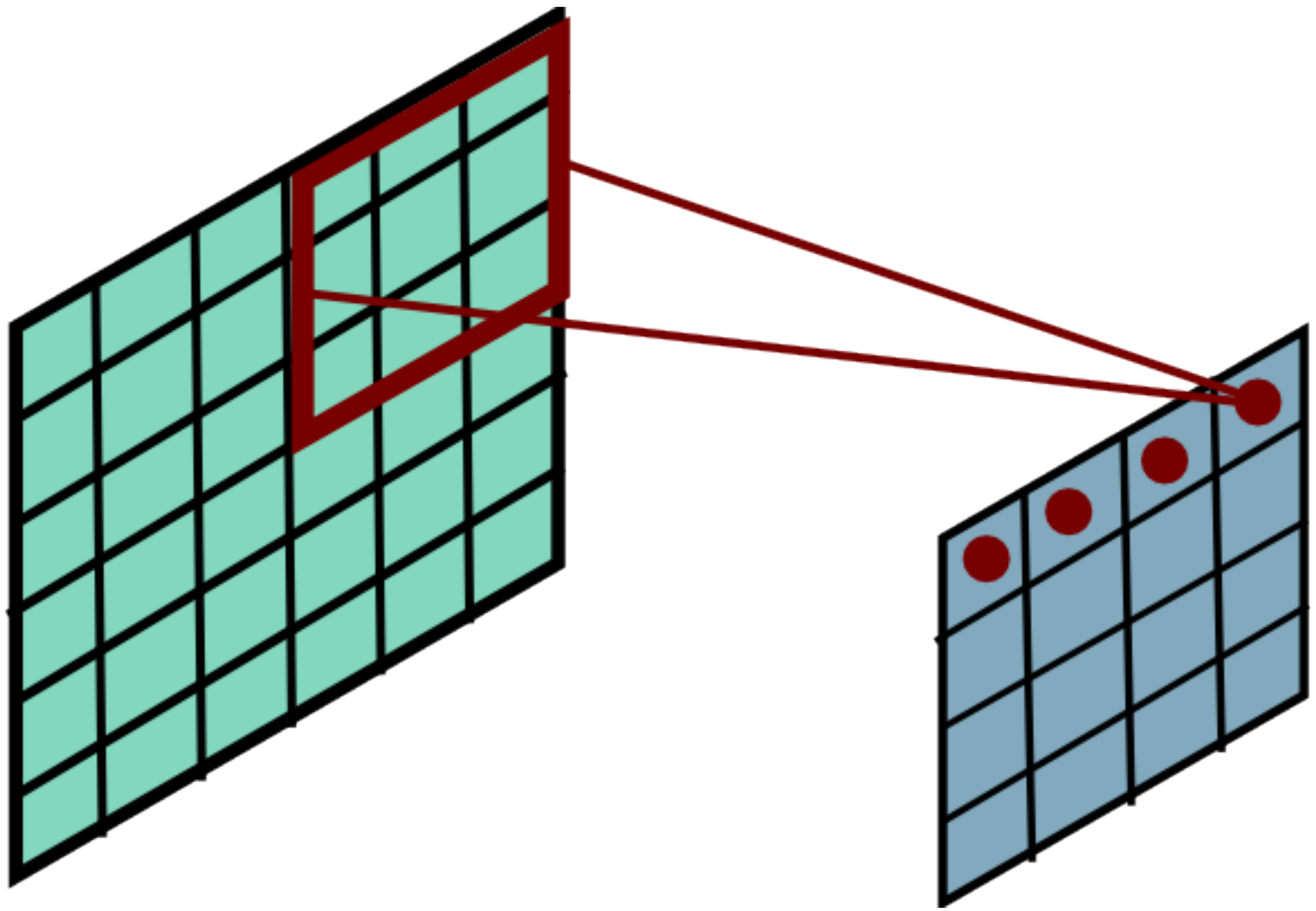
* slide from Marc'Aurelio Renzato

Convolutional Layer



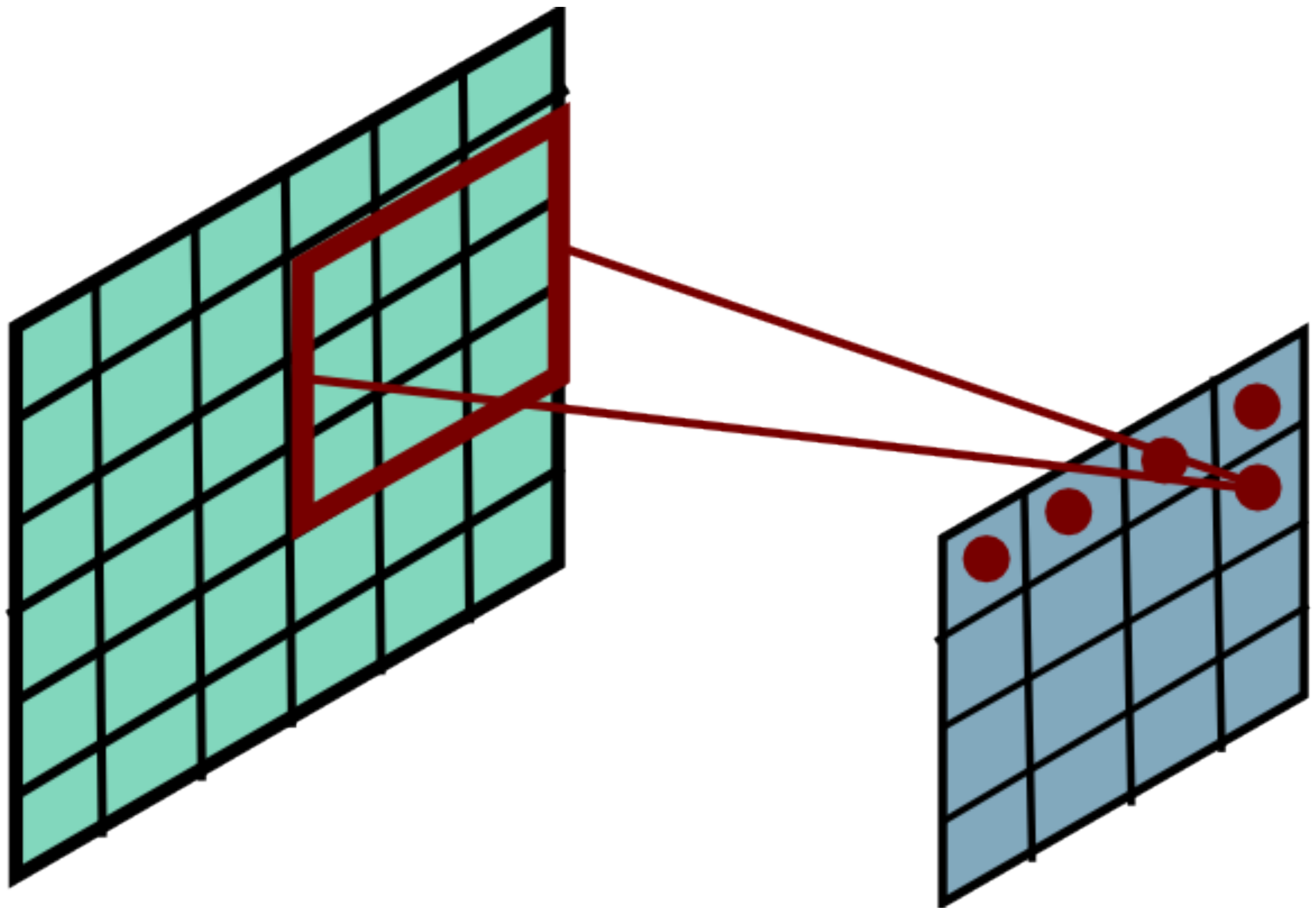
* slide from Marc'Aurelio Renzato

Convolutional Layer



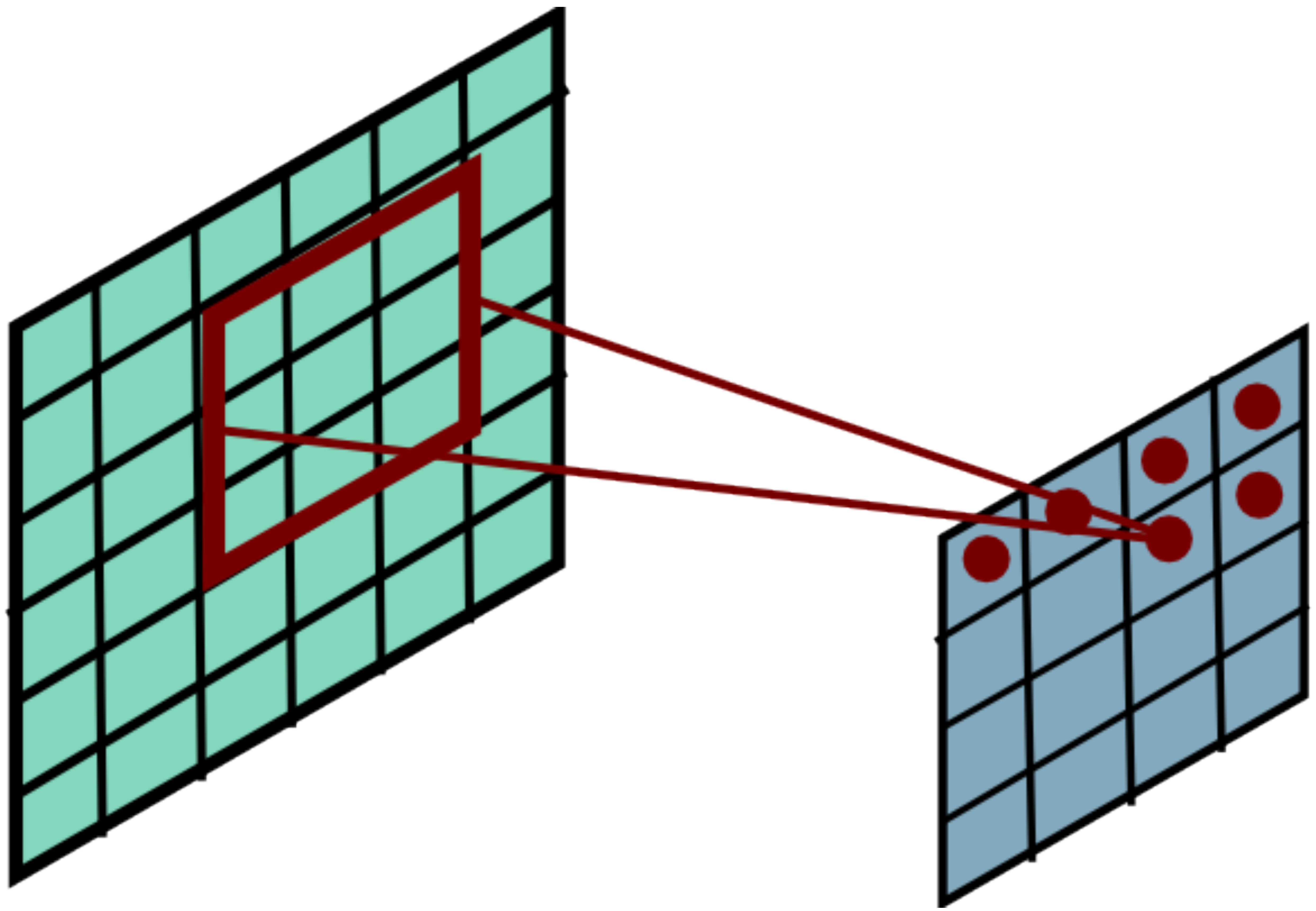
* slide from Marc'Aurelio Renzato

Convolutional Layer



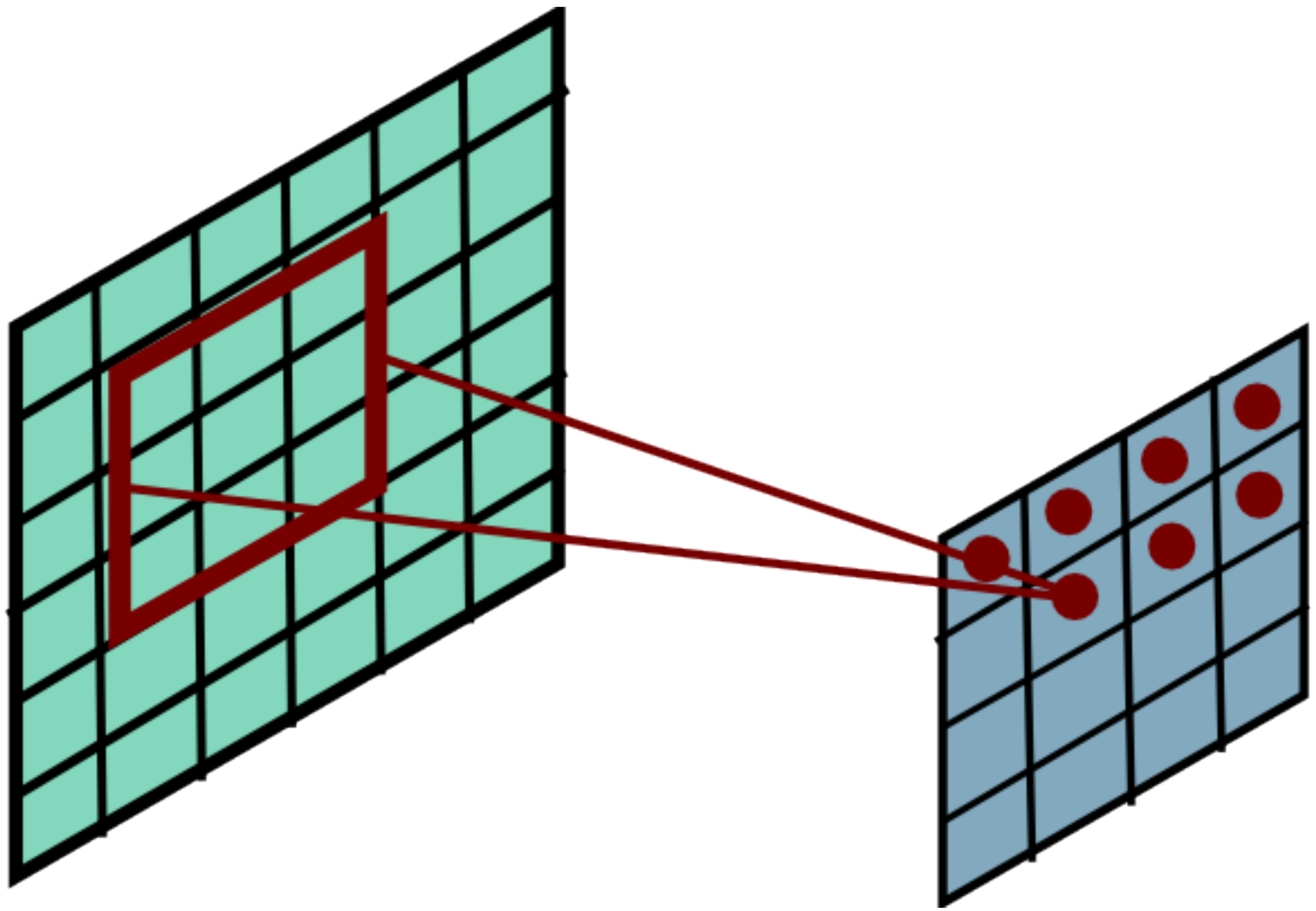
* slide from Marc'Aurelio Renzato

Convolutional Layer



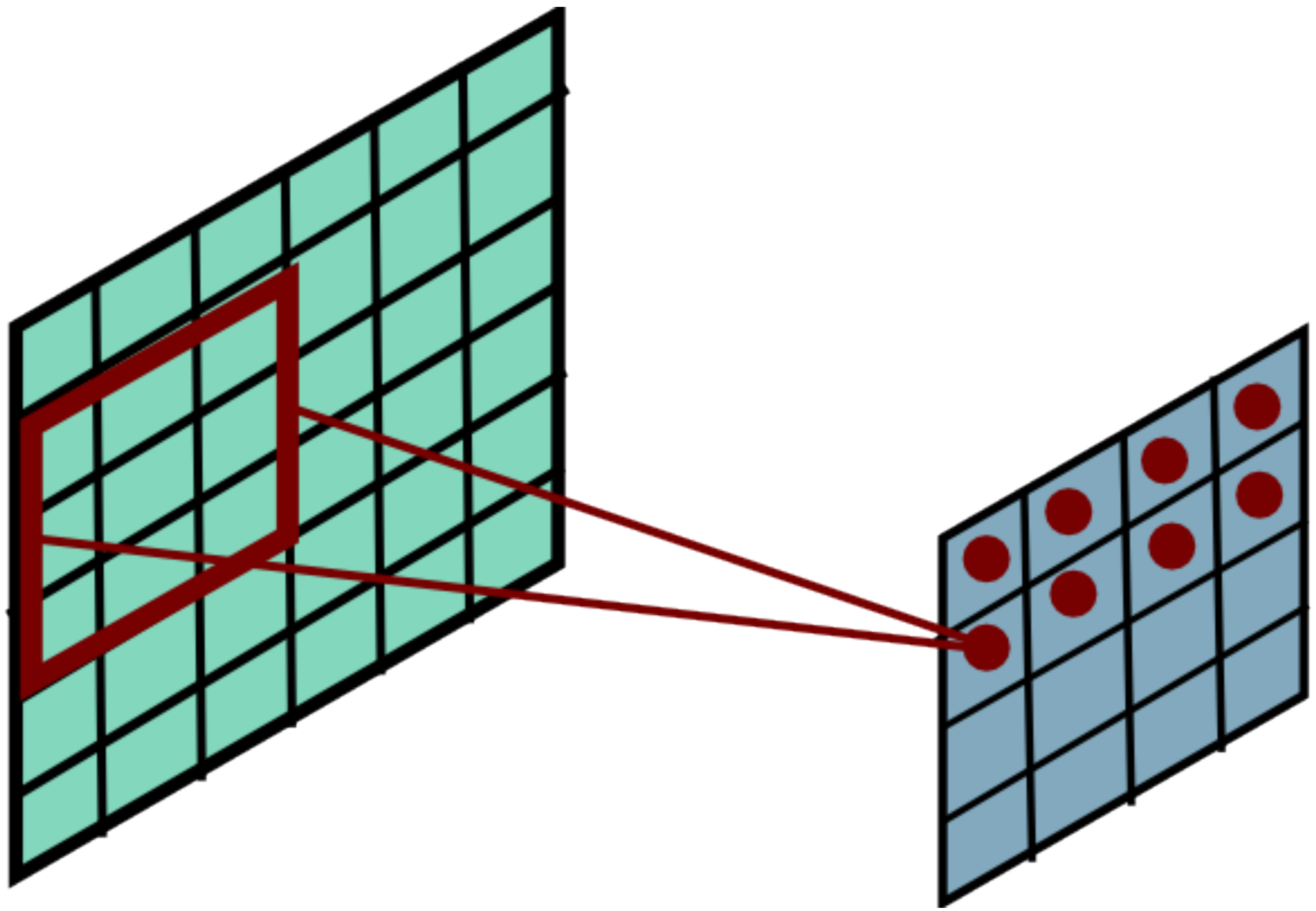
* slide from Marc'Aurelio Renzato

Convolutional Layer



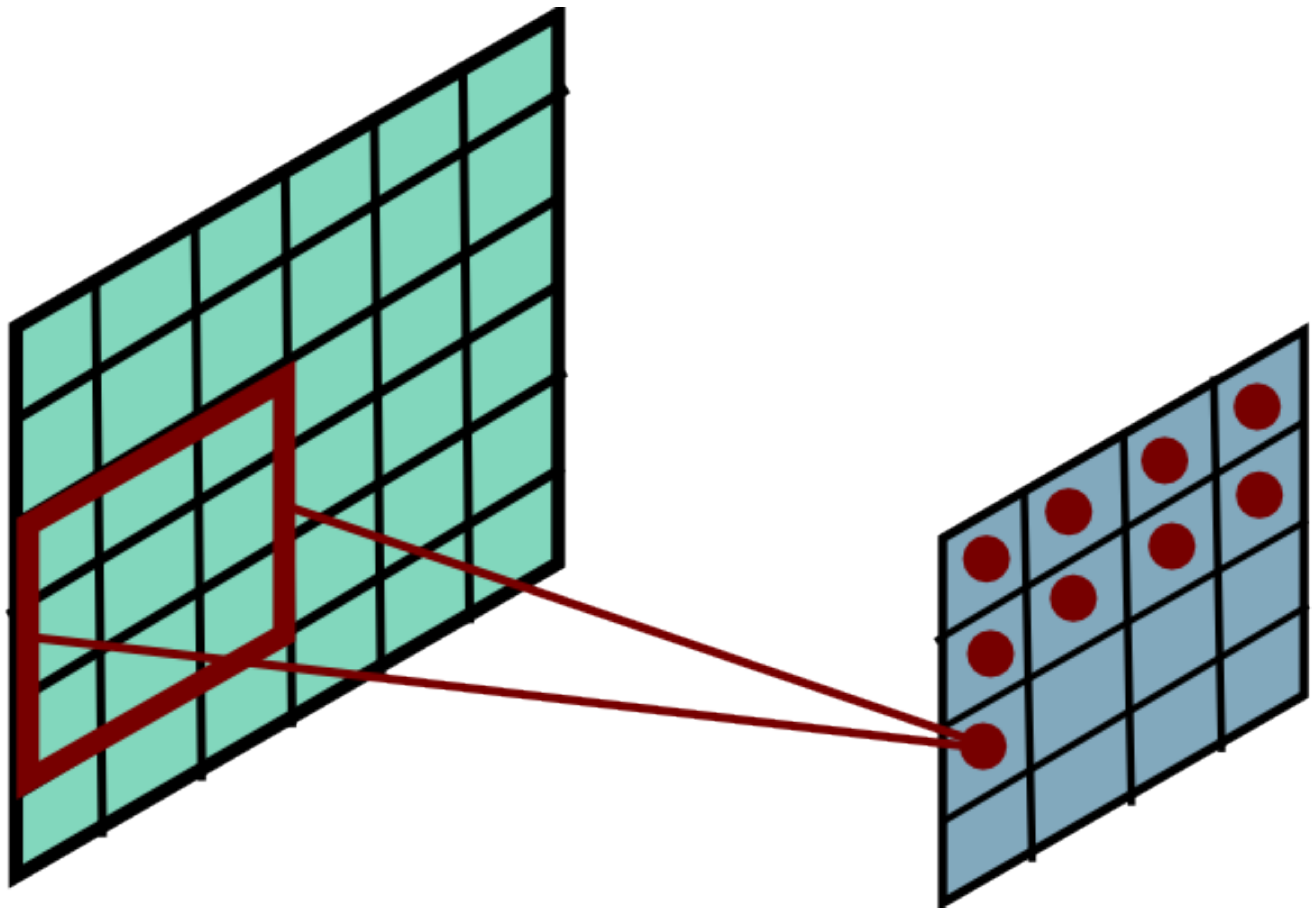
* slide from Marc'Aurelio Renzato

Convolutional Layer



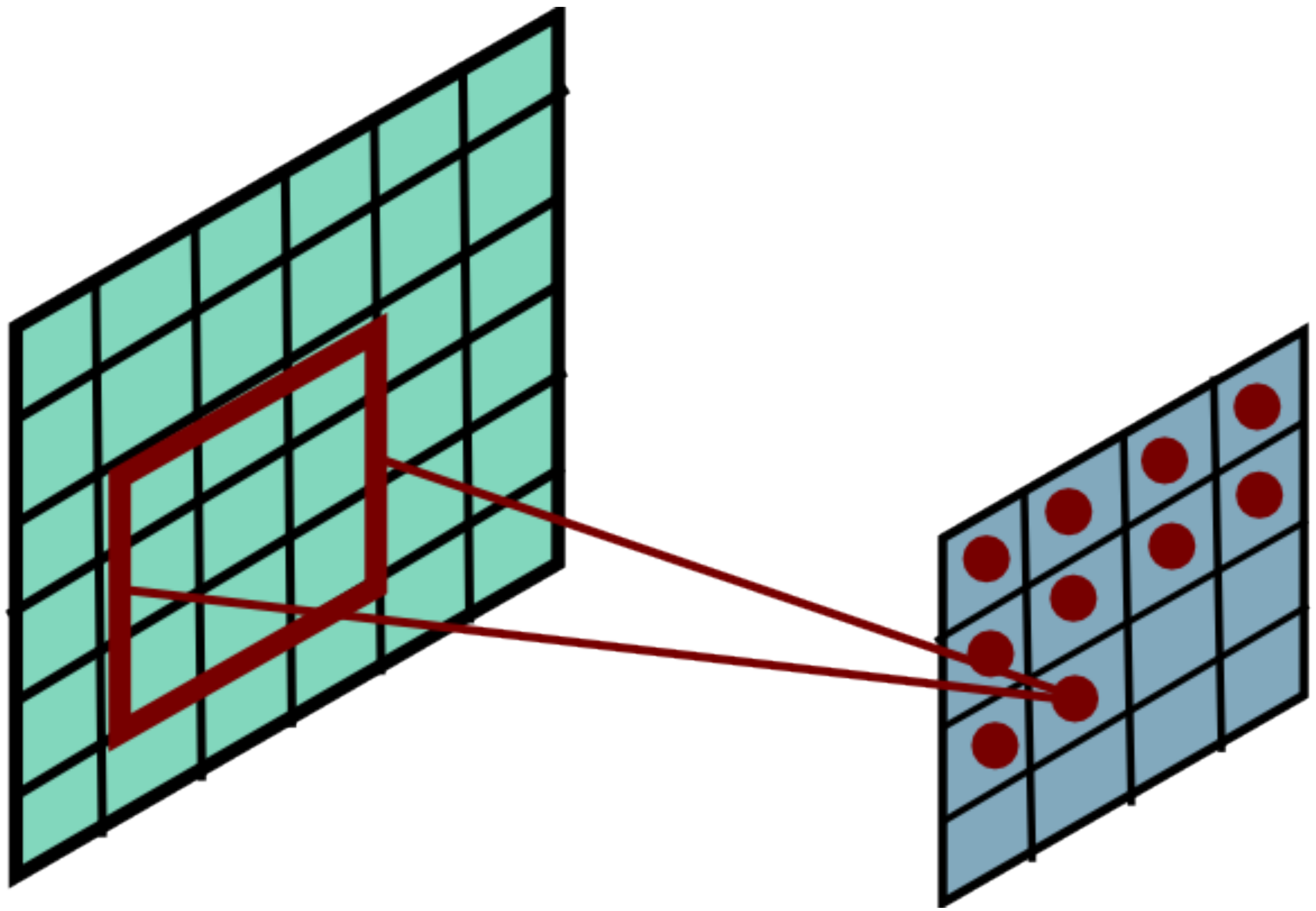
* slide from Marc'Aurelio Renzato

Convolutional Layer



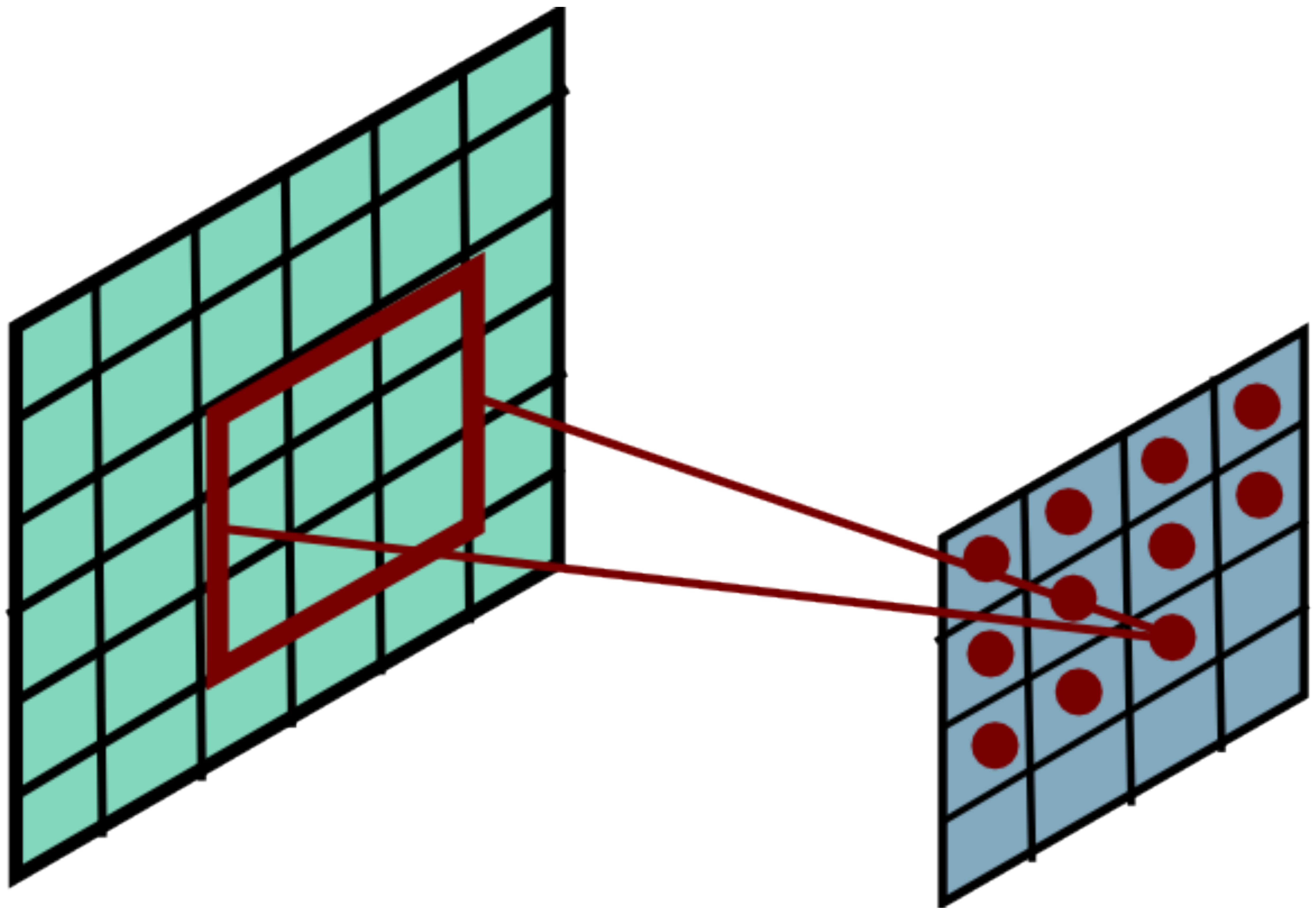
* slide from Marc'Aurelio Renzato

Convolutional Layer



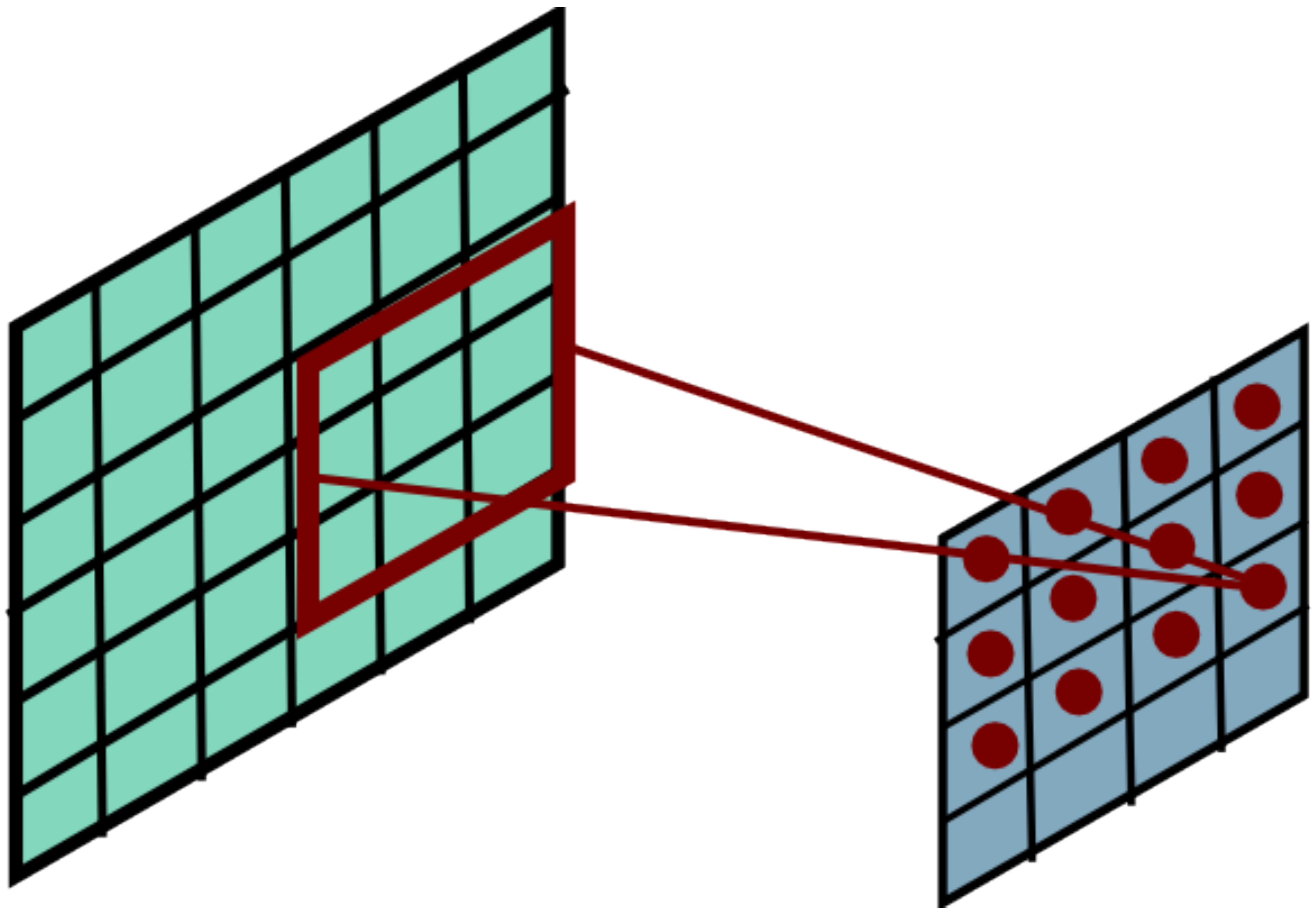
* slide from Marc'Aurelio Renzato

Convolutional Layer



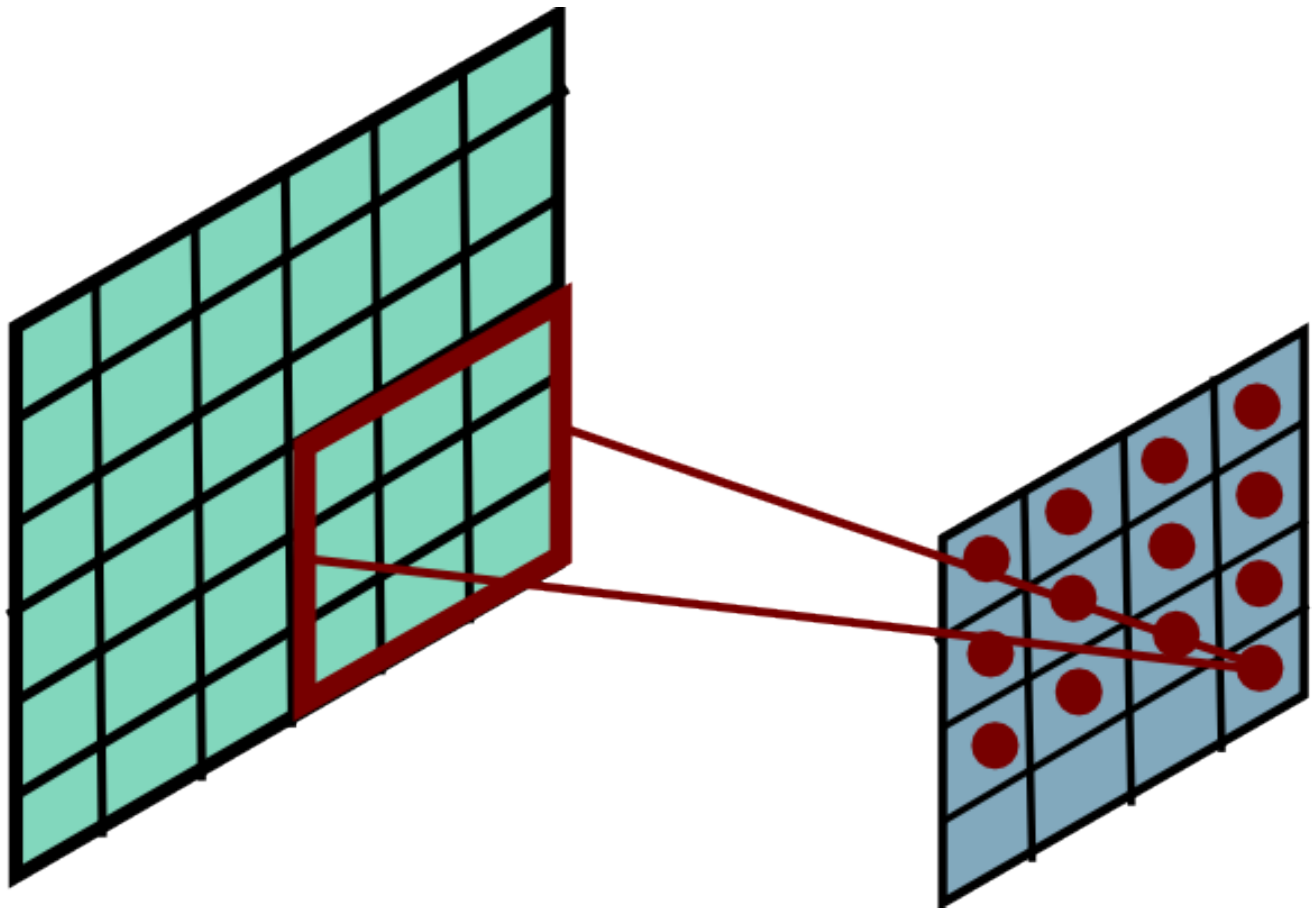
* slide from Marc'Aurelio Renzato

Convolutional Layer



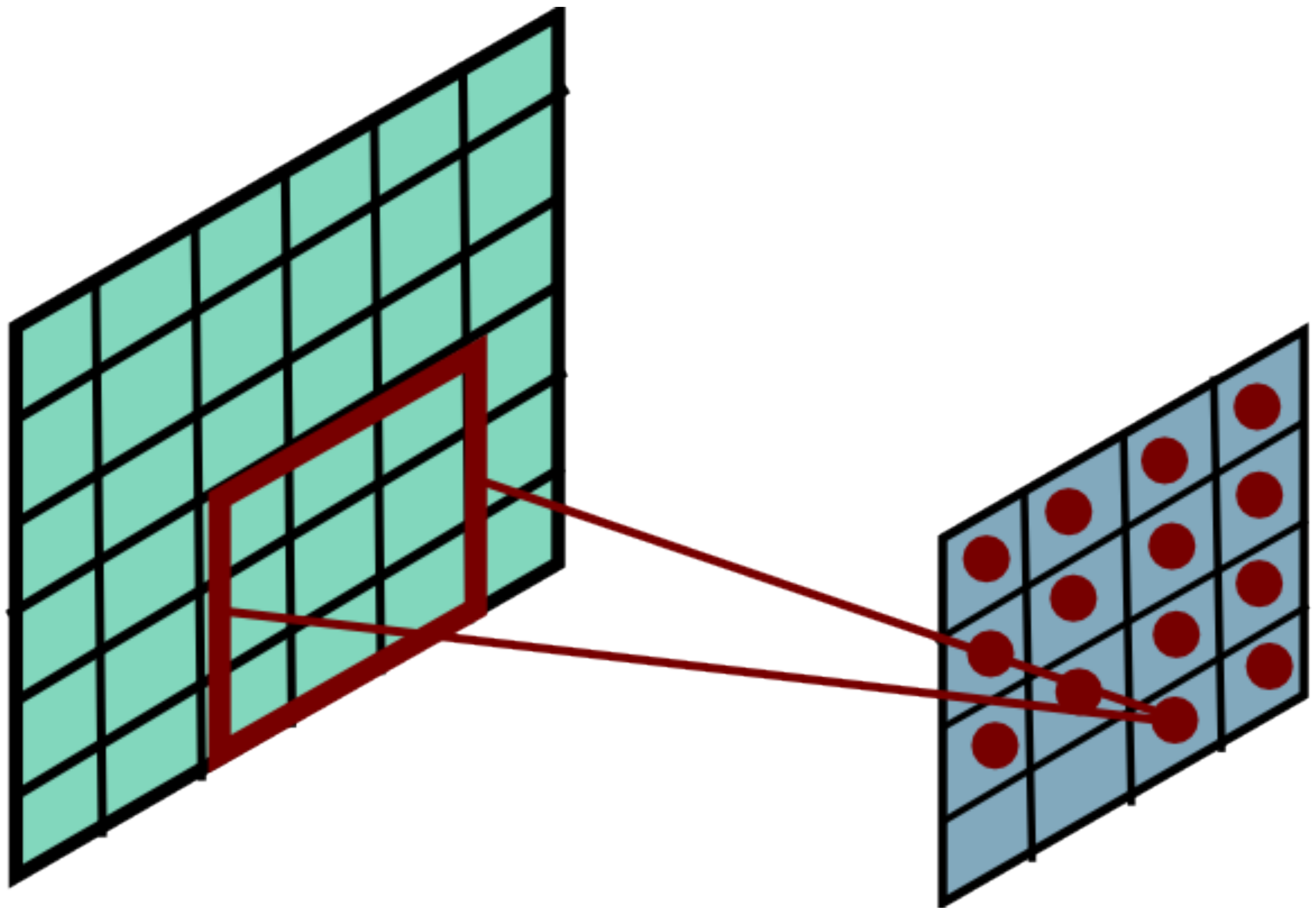
* slide from Marc'Aurelio Renzato

Convolutional Layer



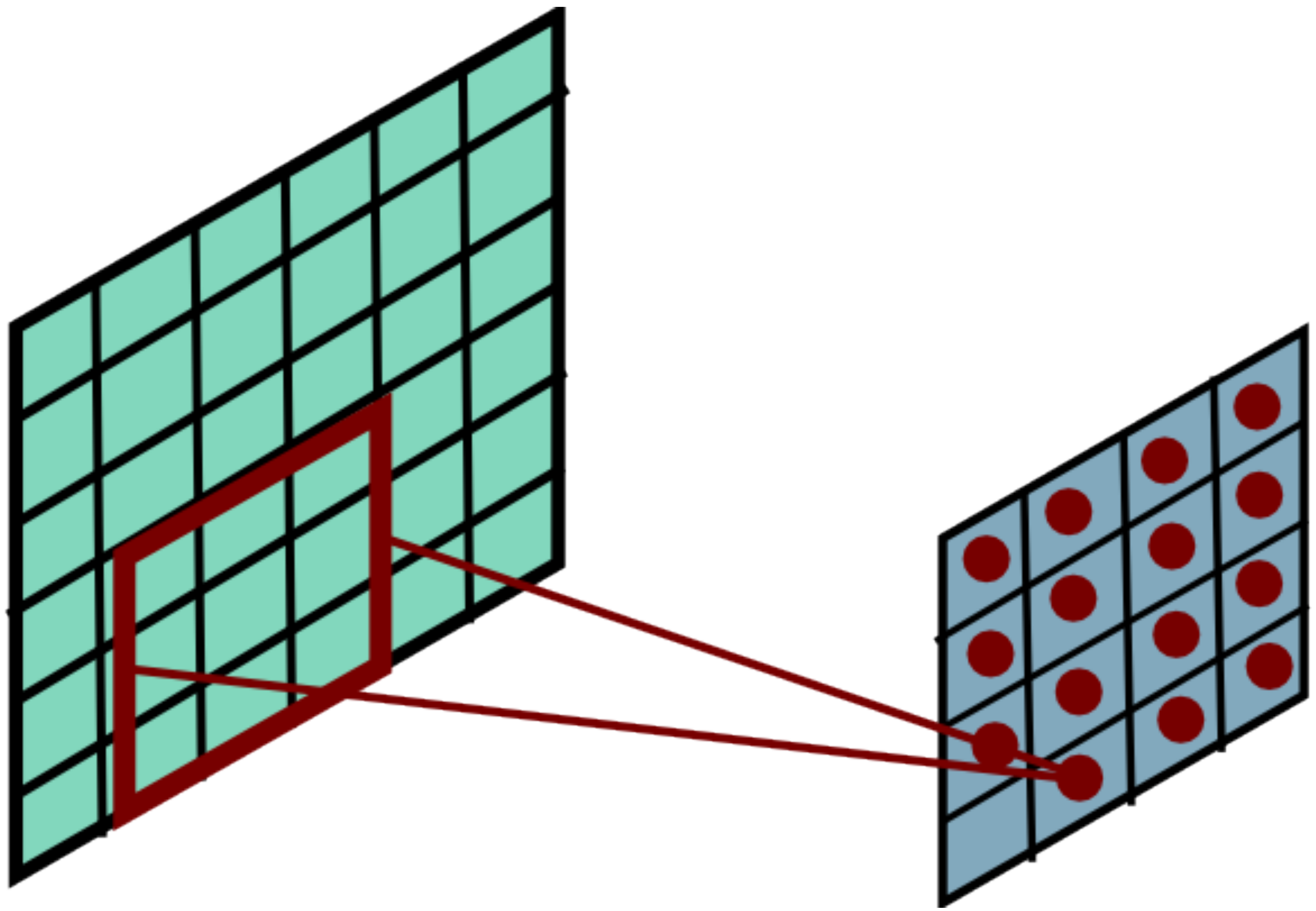
* slide from Marc'Aurelio Renzato

Convolutional Layer



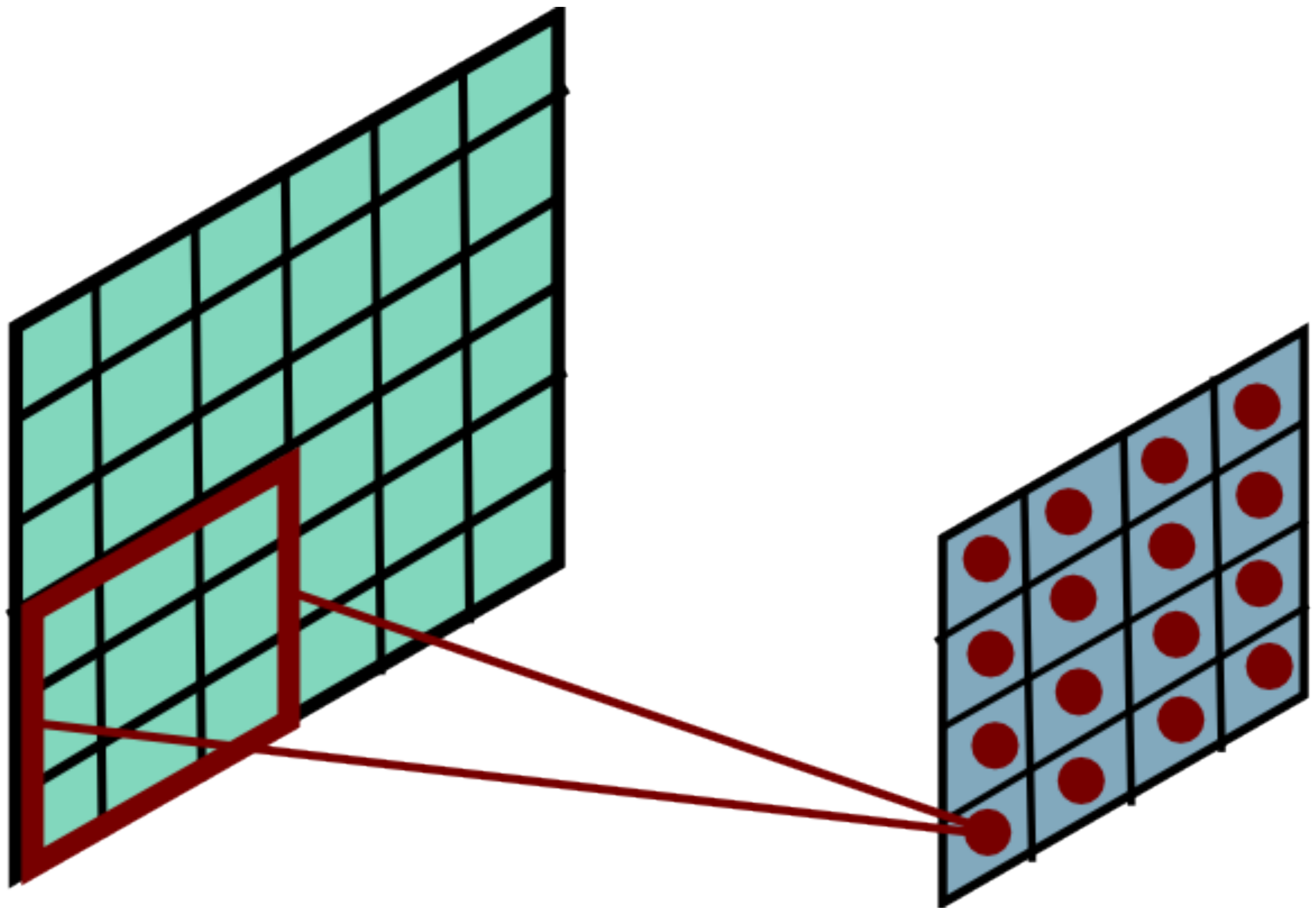
* slide from Marc'Aurelio Renzato

Convolutional Layer



* slide from Marc'Aurelio Renzato

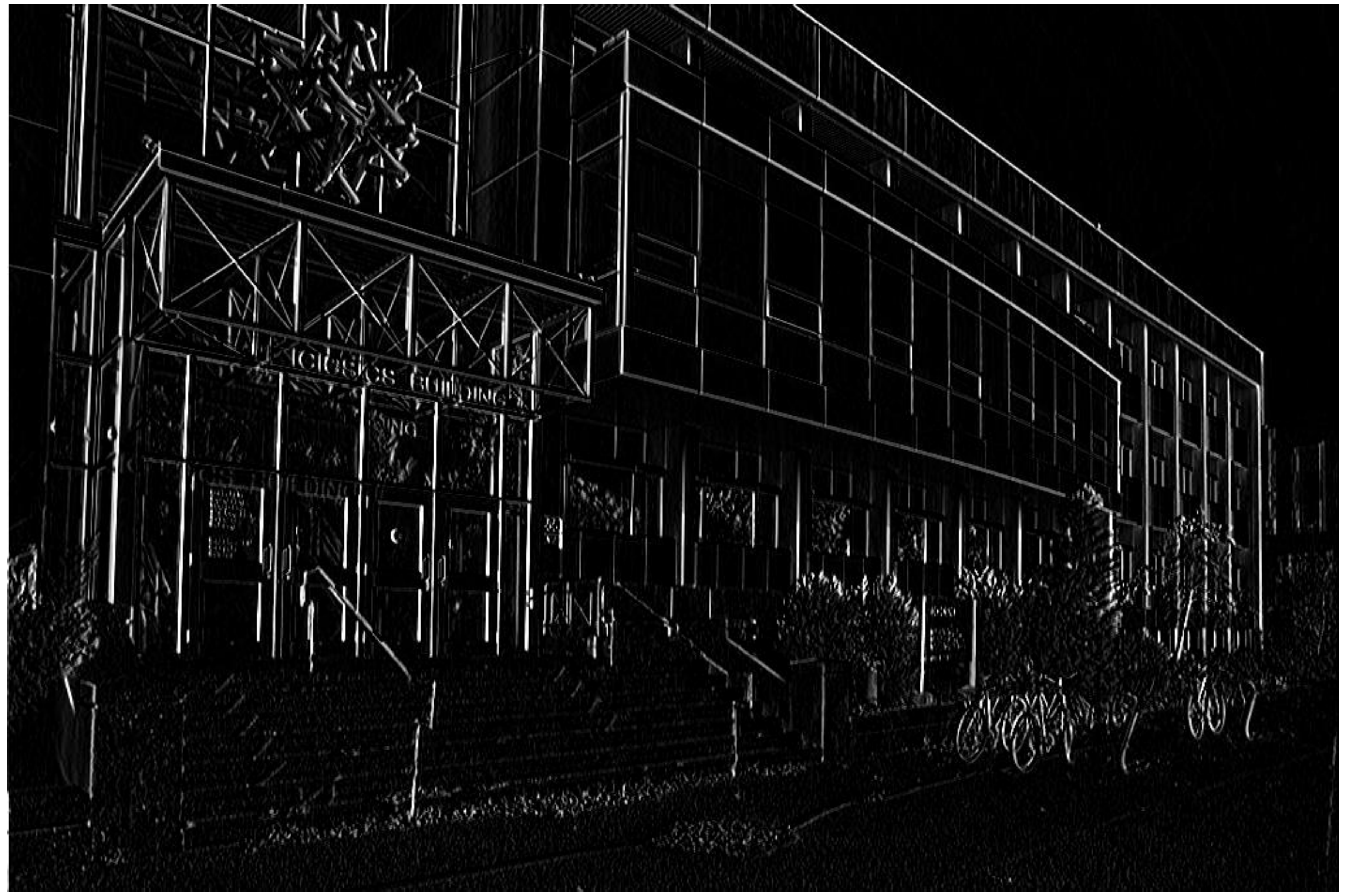
Convolutional Layer



* slide from Marc'Aurelio Renzato

Convolution Layer

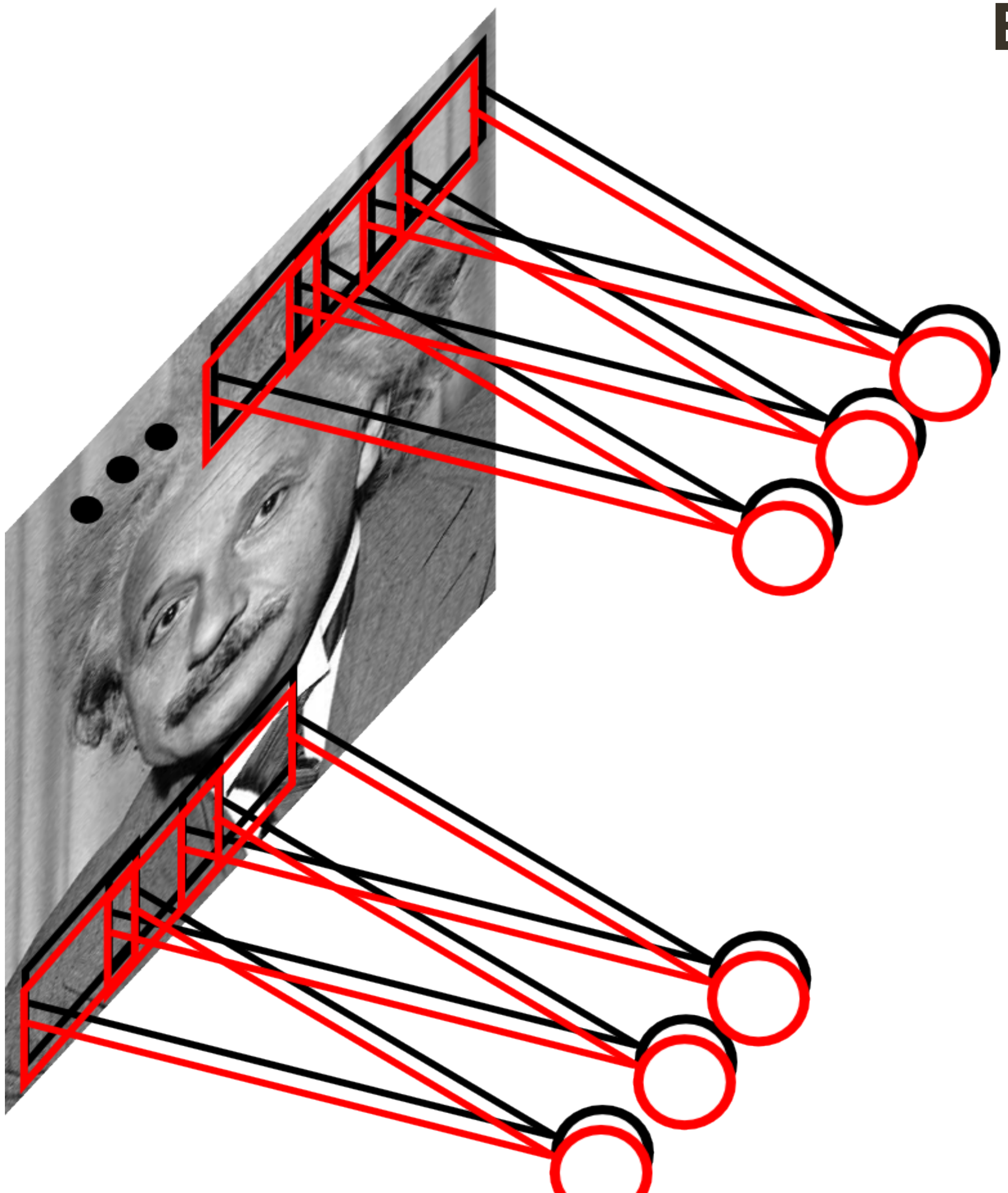
$$\star \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \longrightarrow$$



Convolution Layer

$$\star \begin{bmatrix} 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \end{bmatrix} \rightarrow$$

Convolutional Layer



Example: 200 x 200 image (small)
x 40K hidden units

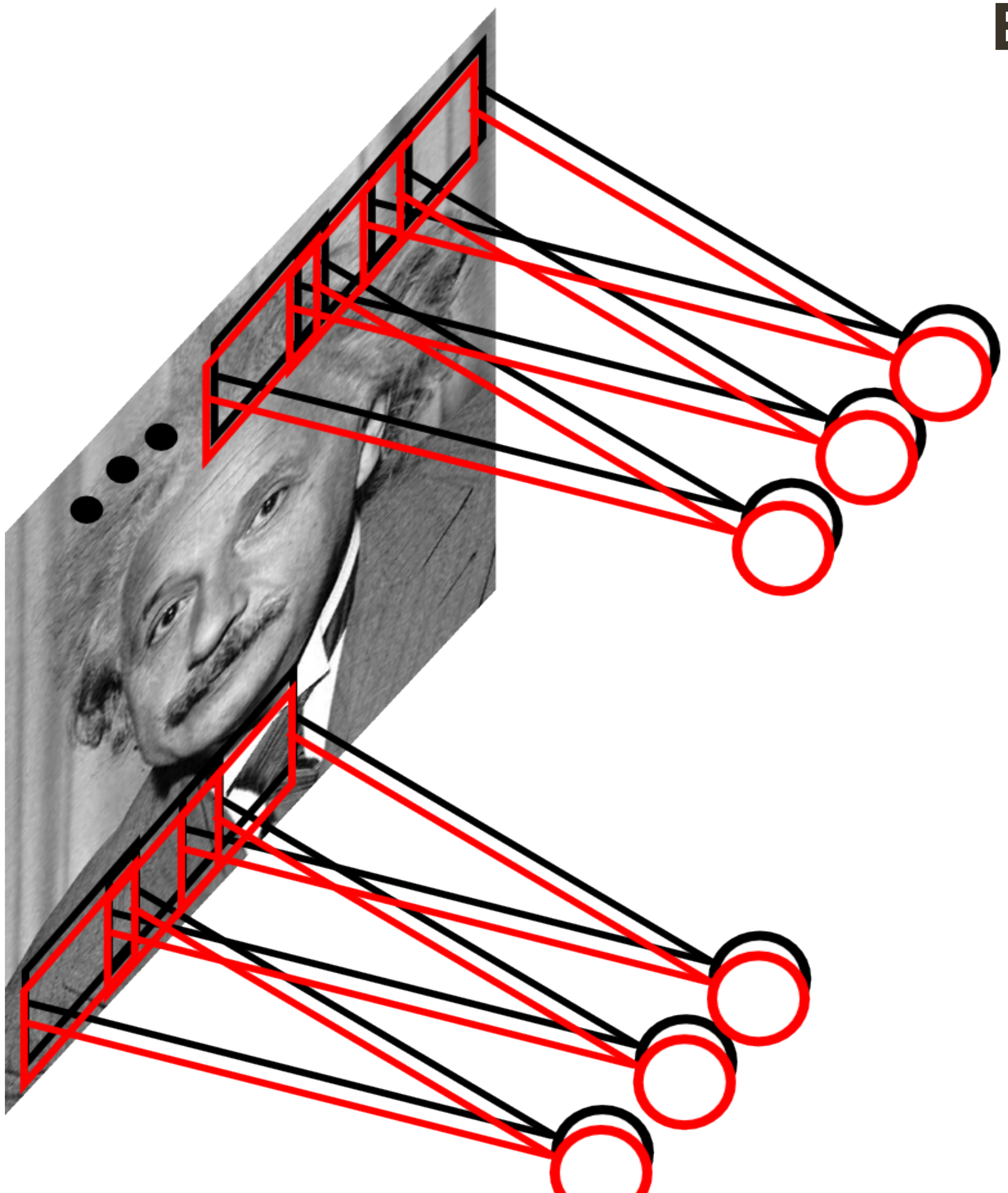
Filter size: 10 x 10

# of filters: 20

Learn **multiple filters**

* slide from Marc'Aurelio Renzato

Convolutional Layer



Example: 200 x 200 image (small)
x 40K hidden units

Filter size: 10 x 10

# of filters: 20

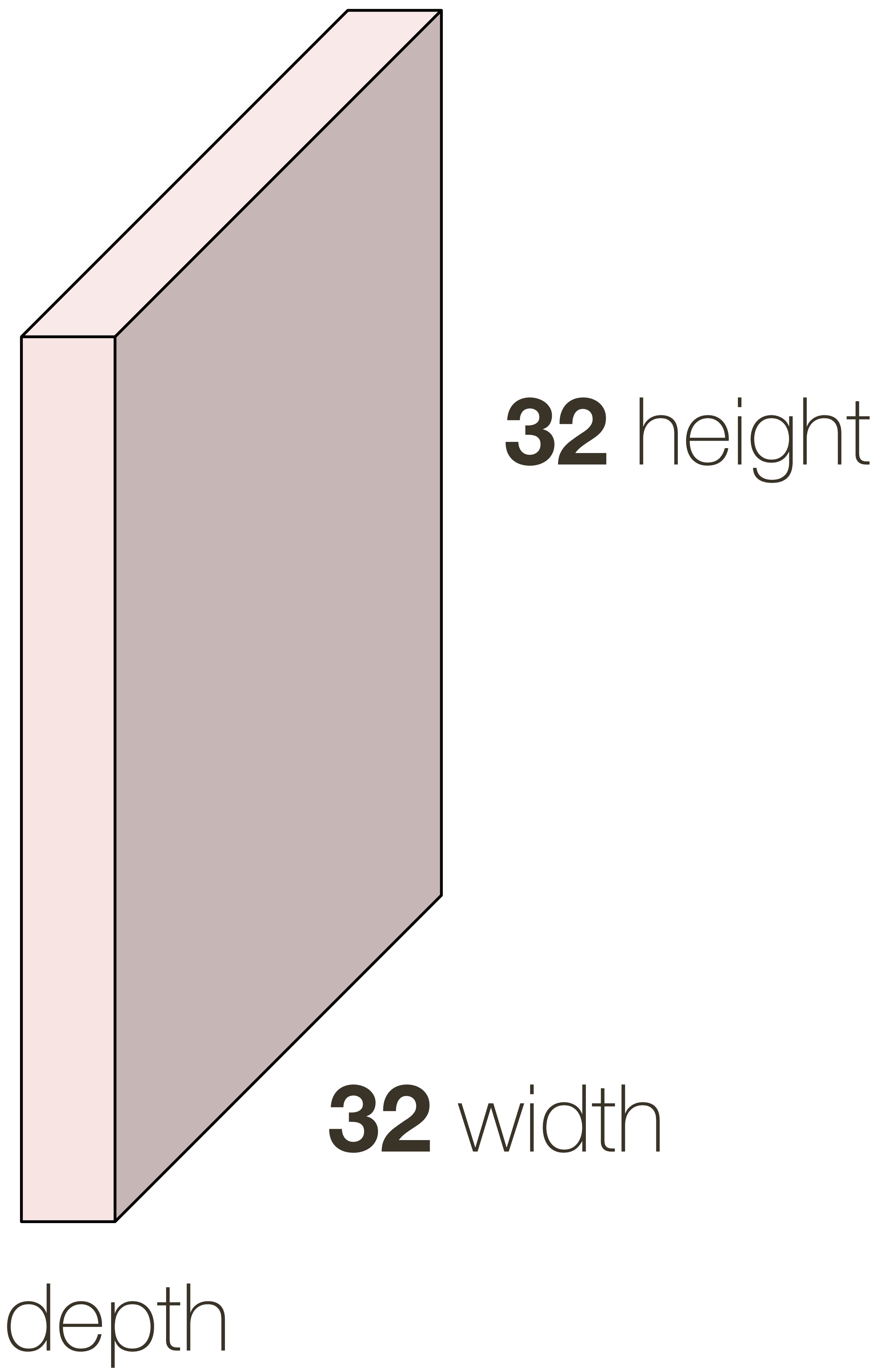
= 2000 parameters

Learn **multiple filters**

* slide from Marc'Aurelio Renzato

Convolutional Layer

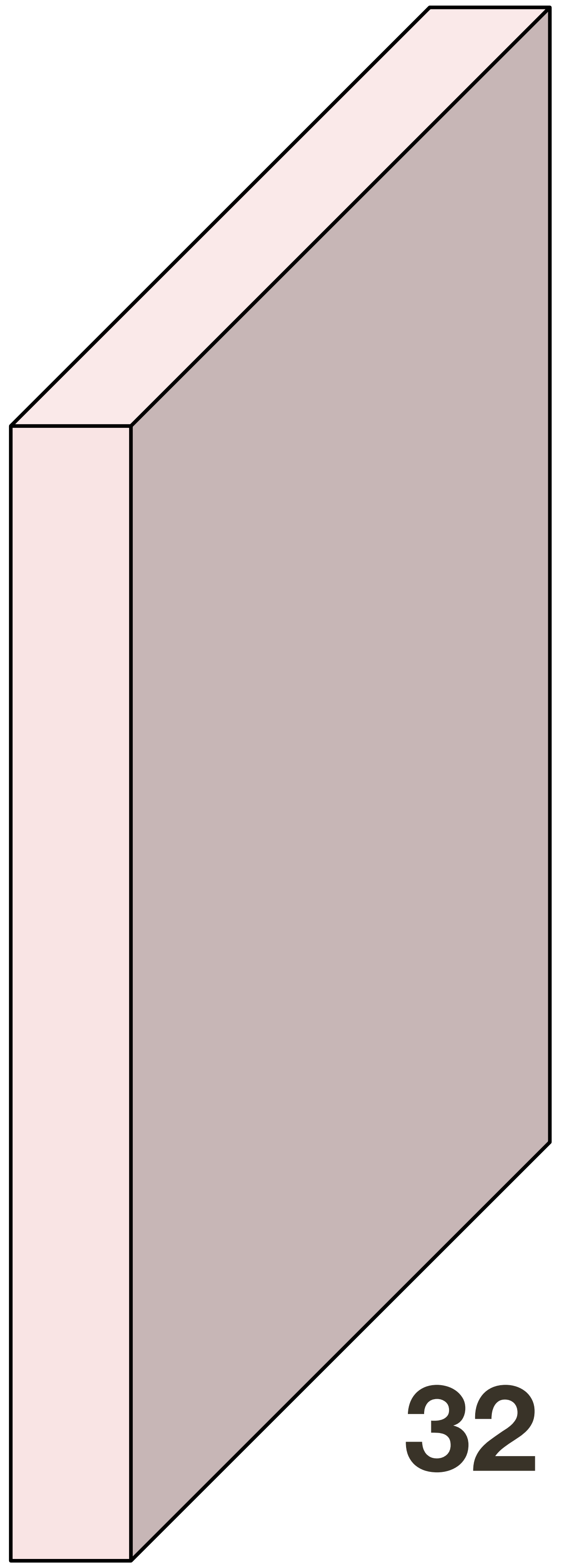
32 x 32 x 3 **image** (note the image preserves spatial structure)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer

32 x 32 x 3 **image**

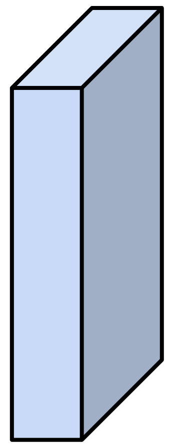


32 height

32 width

3 depth

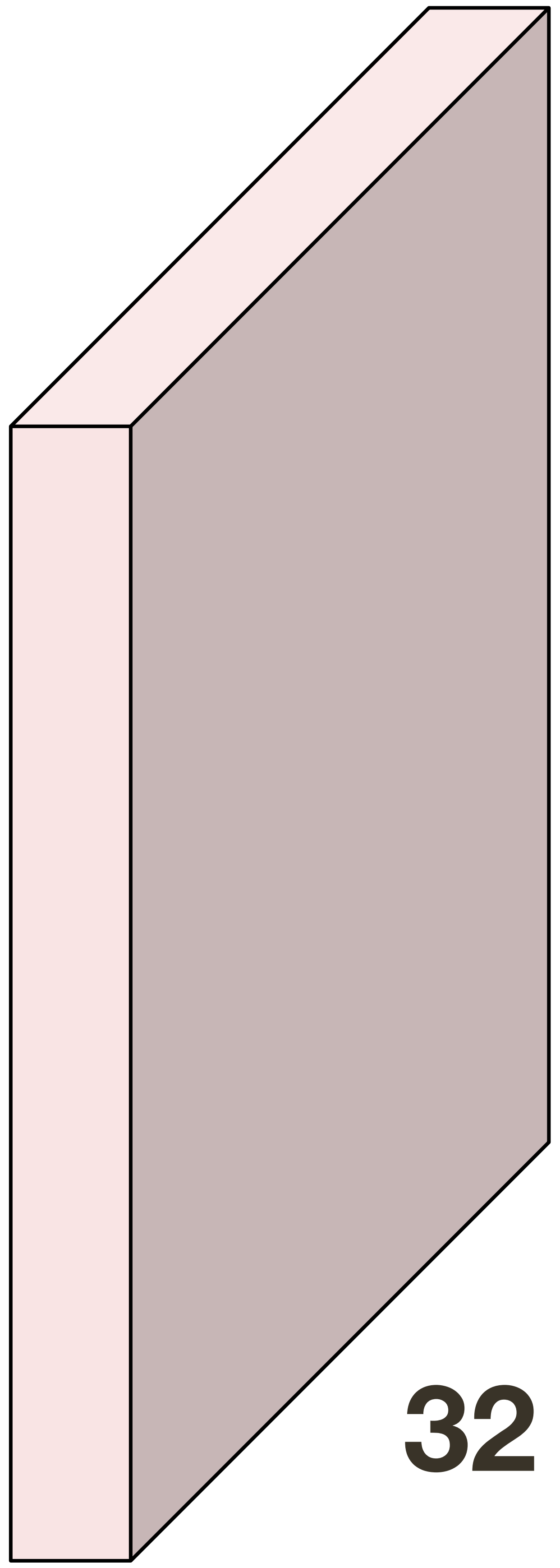
5 x 5 x 3 **filter**



Convolve the filter with the image
(i.e., “slide over the image spatially,
computing dot products”)

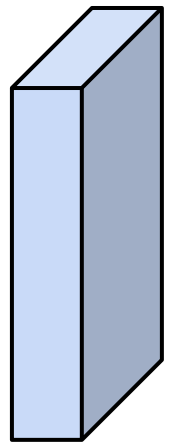
Convolutional Layer

32 x 32 x 3 **image**



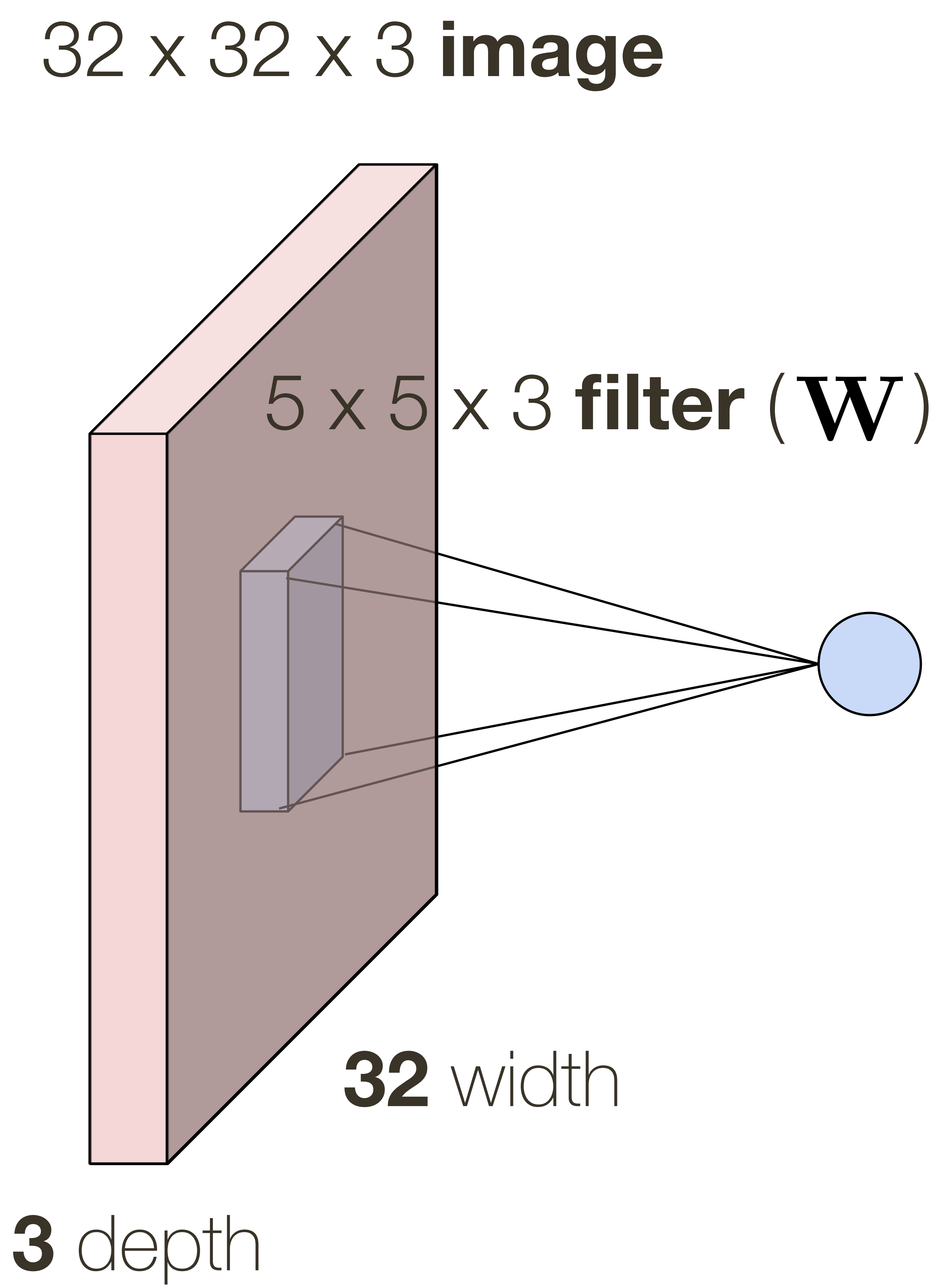
Filters always extend the full depth of the input volume

5 x 5 x 3 **filter**



Convolve the filter with the image (i.e., “slide over the image spatially, computing dot products”

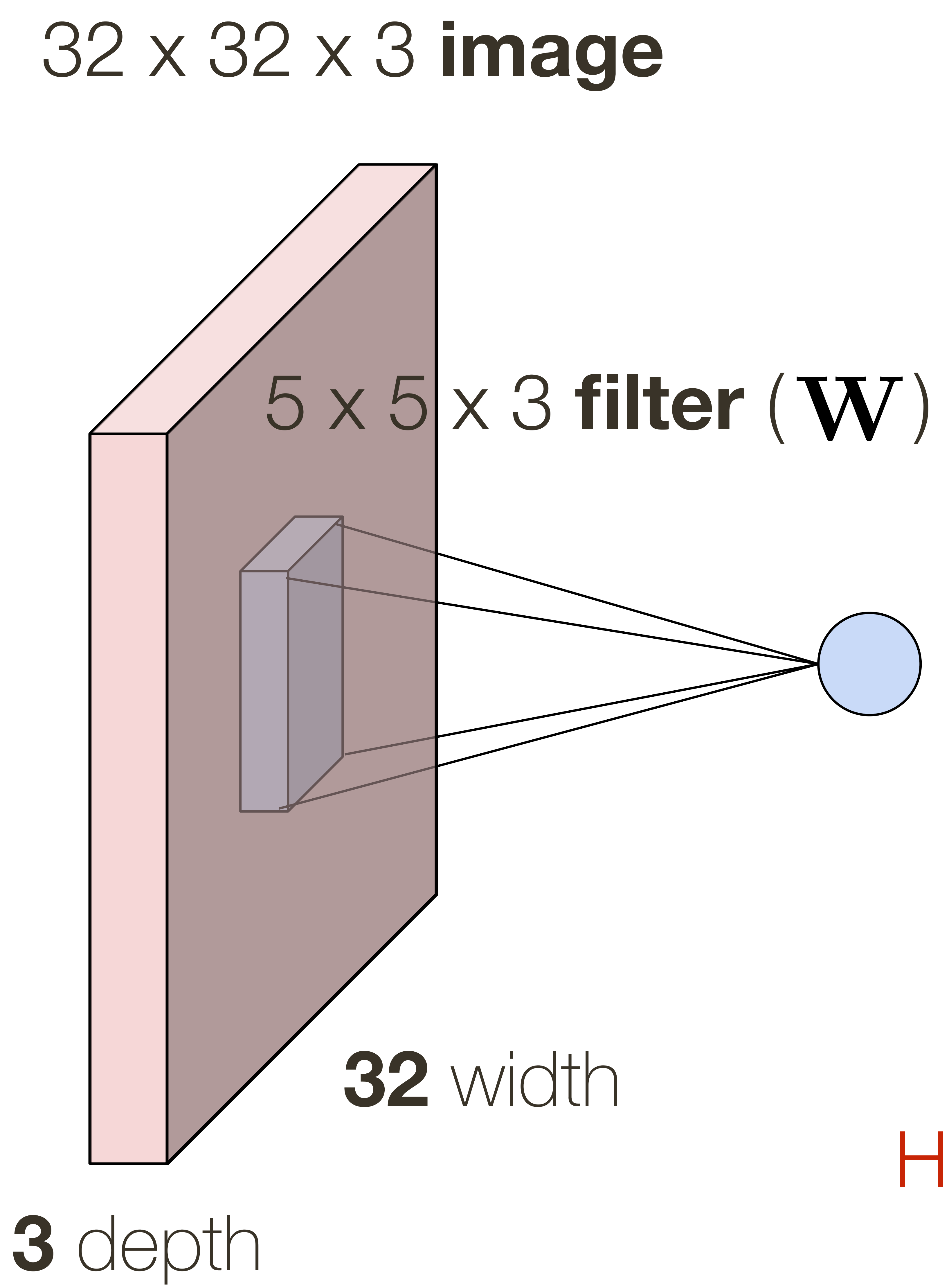
Convolutional Layer



1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

$$\mathbf{W}^T\mathbf{x} + b, \text{where } \mathbf{W}, \mathbf{x} \in \mathbb{R}^{75}$$

Convolutional Layer

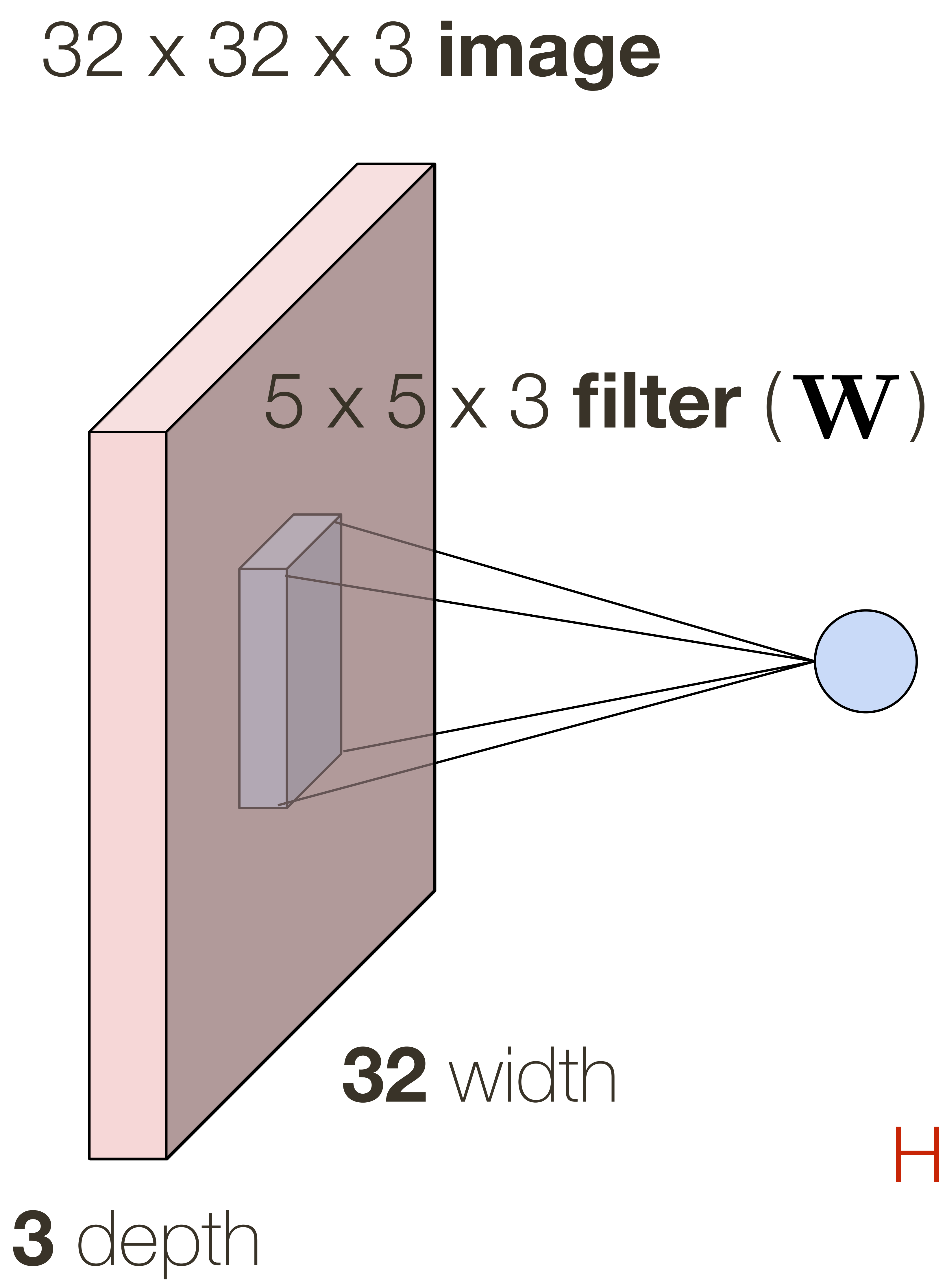


1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

$$\mathbf{W}^T\mathbf{x} + b, \text{where } \mathbf{W}, \mathbf{x} \in \mathbb{R}^{75}$$

How many **parameters** does the layer have?

Convolutional Layer



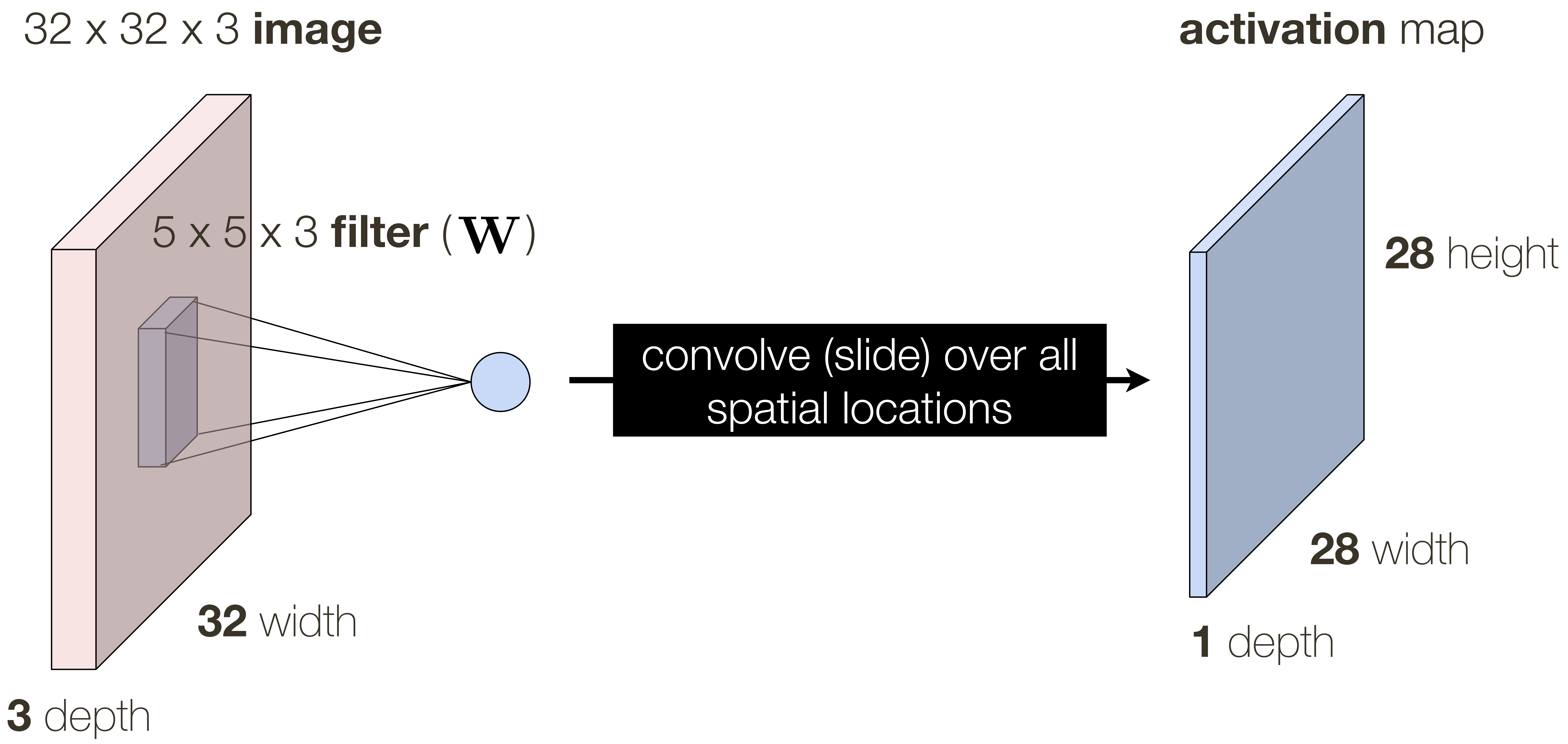
1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

$$\mathbf{W}^T\mathbf{x} + b, \text{where } \mathbf{W}, \mathbf{x} \in \mathbb{R}^{75}$$

How many **parameters** does the layer have? **76**

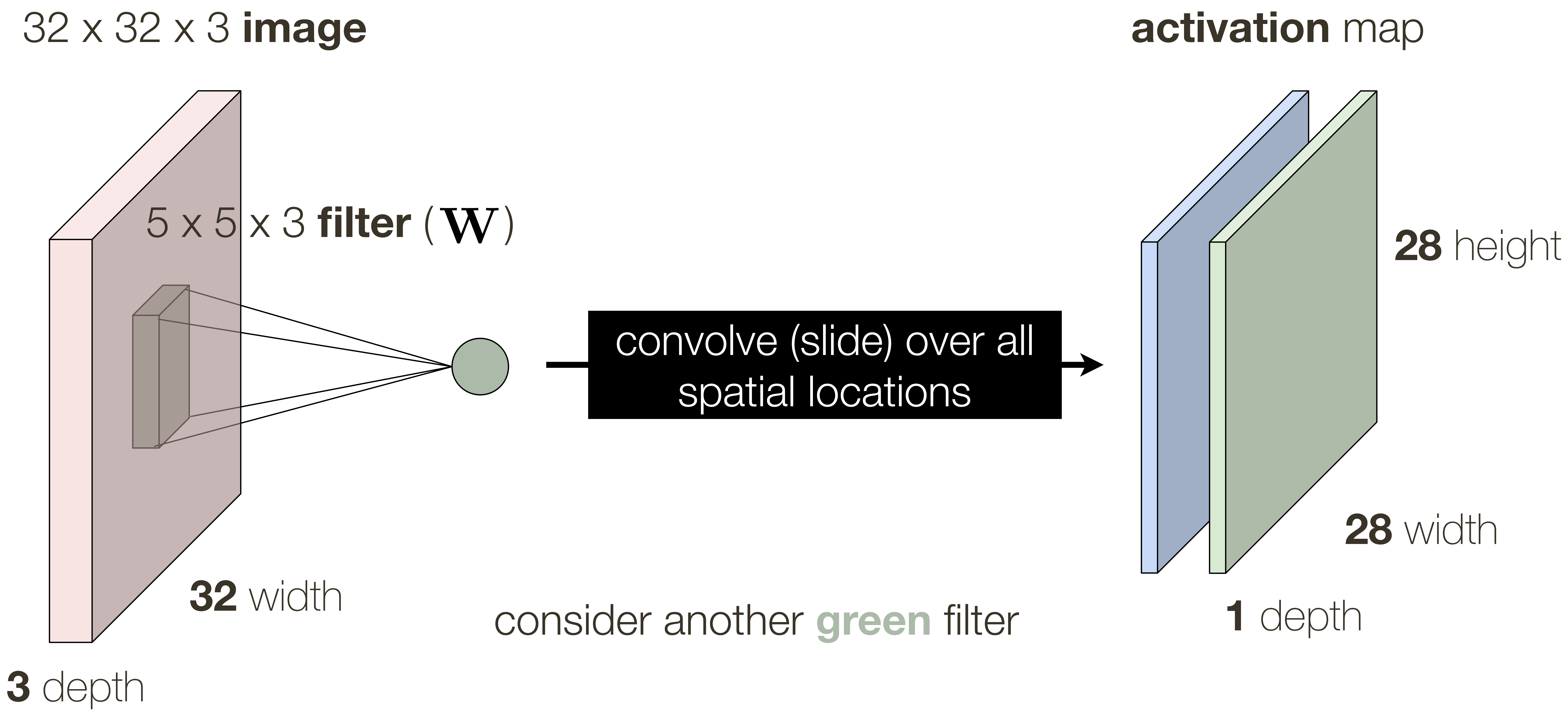
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

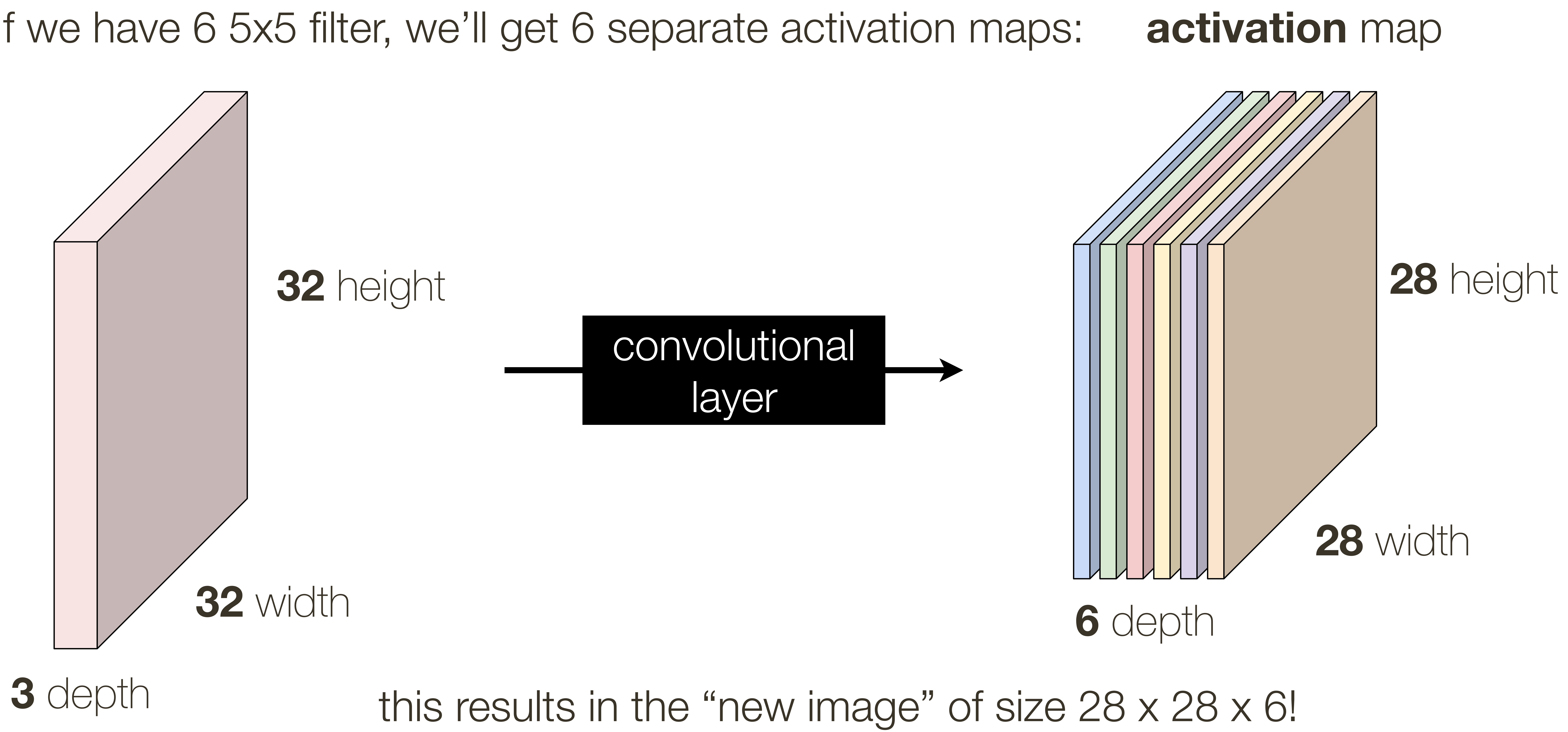
Convolutional Layer



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer

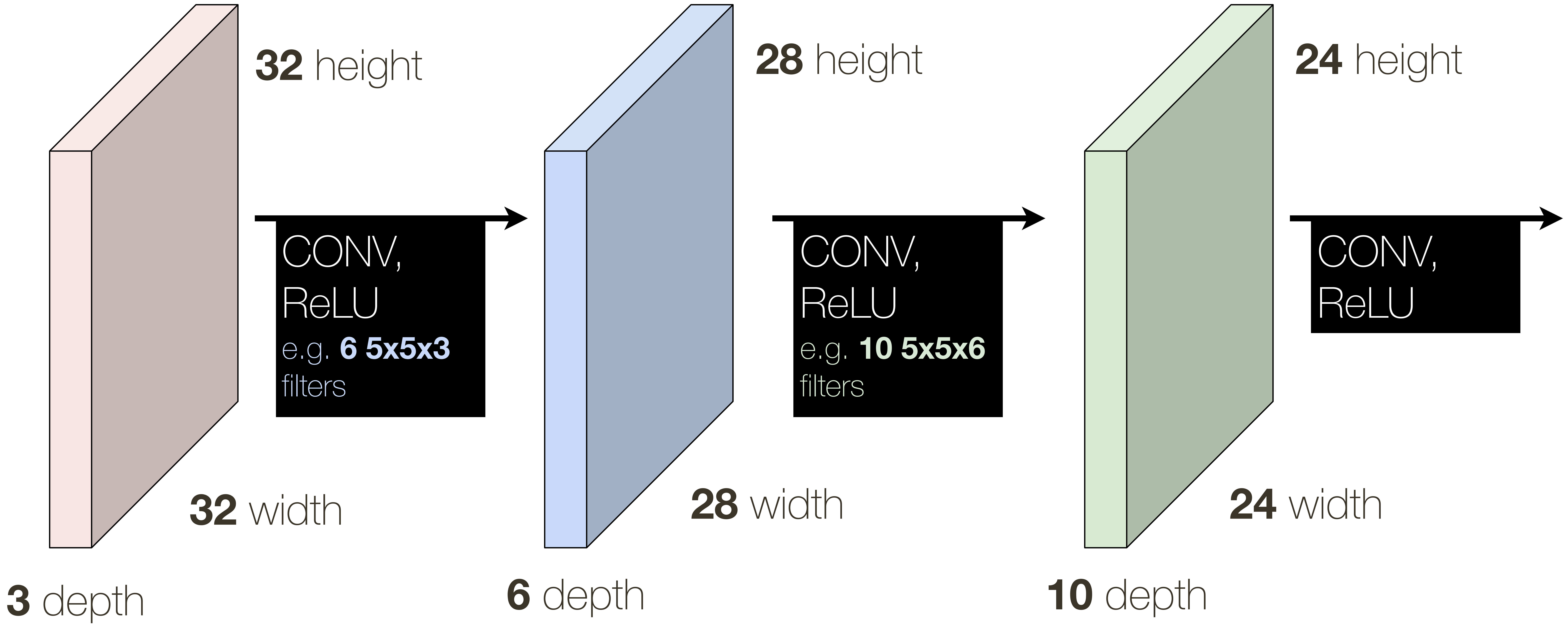
If we have 6 5x5 filter, we'll get 6 separate activation maps:



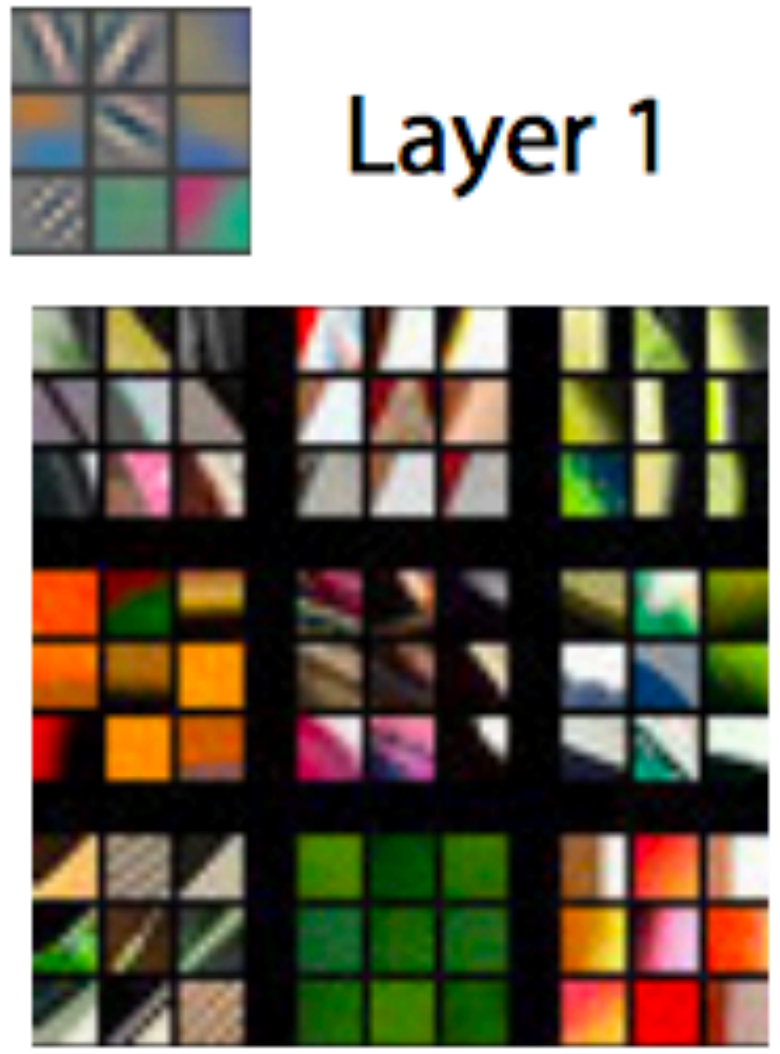
this results in the "new image" of size 28 x 28 x 6!

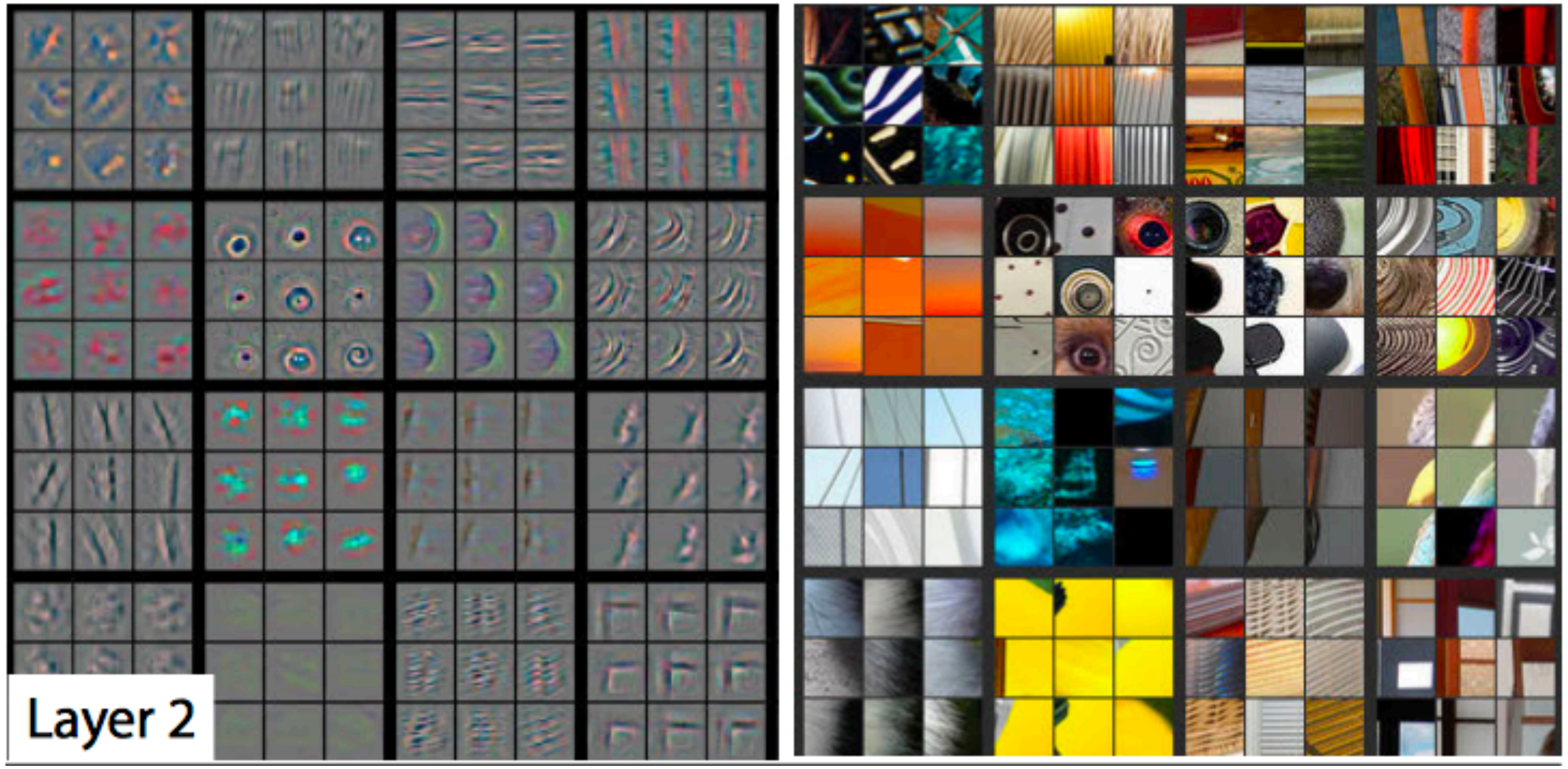
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



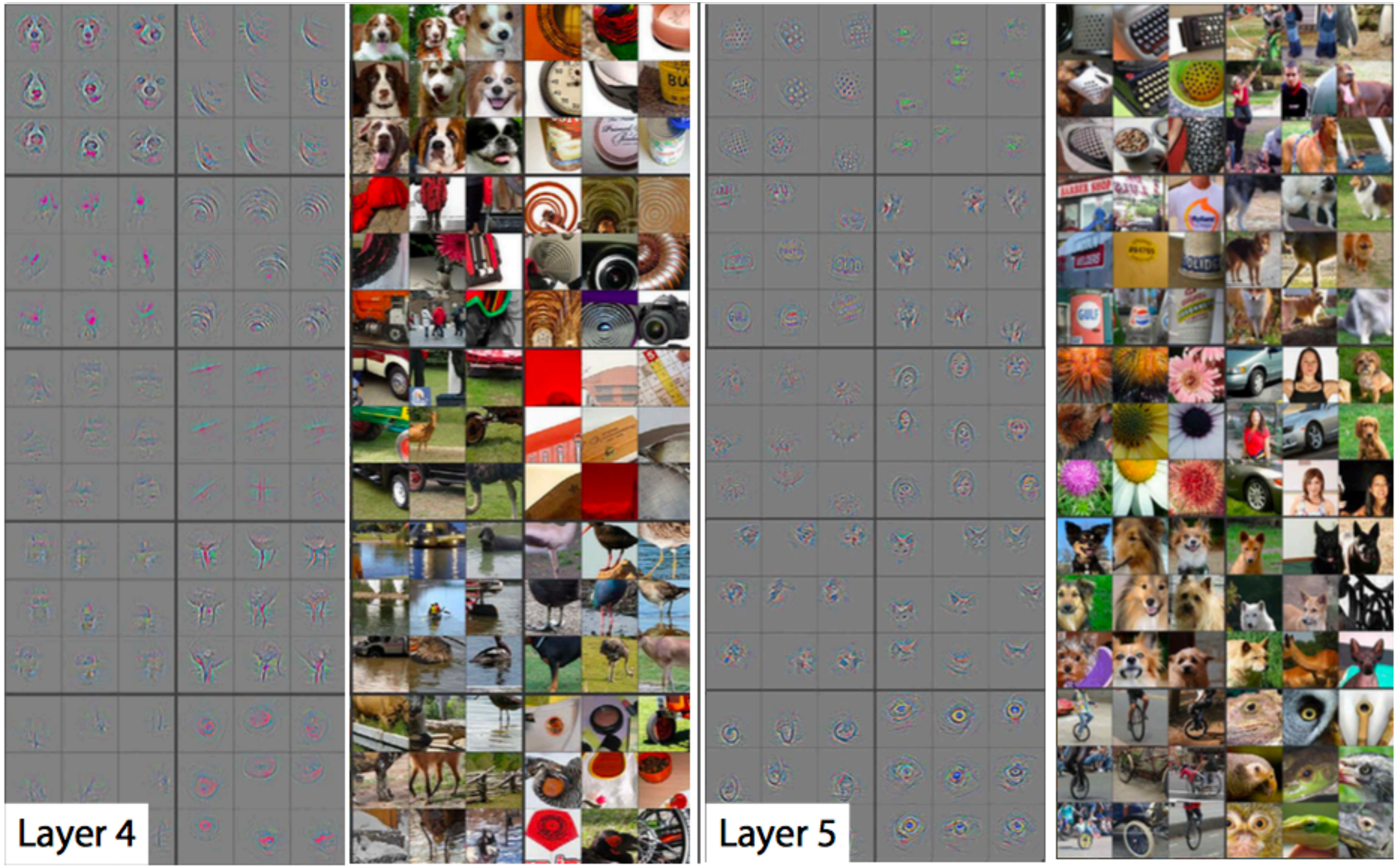
What **filters** do networks learn?





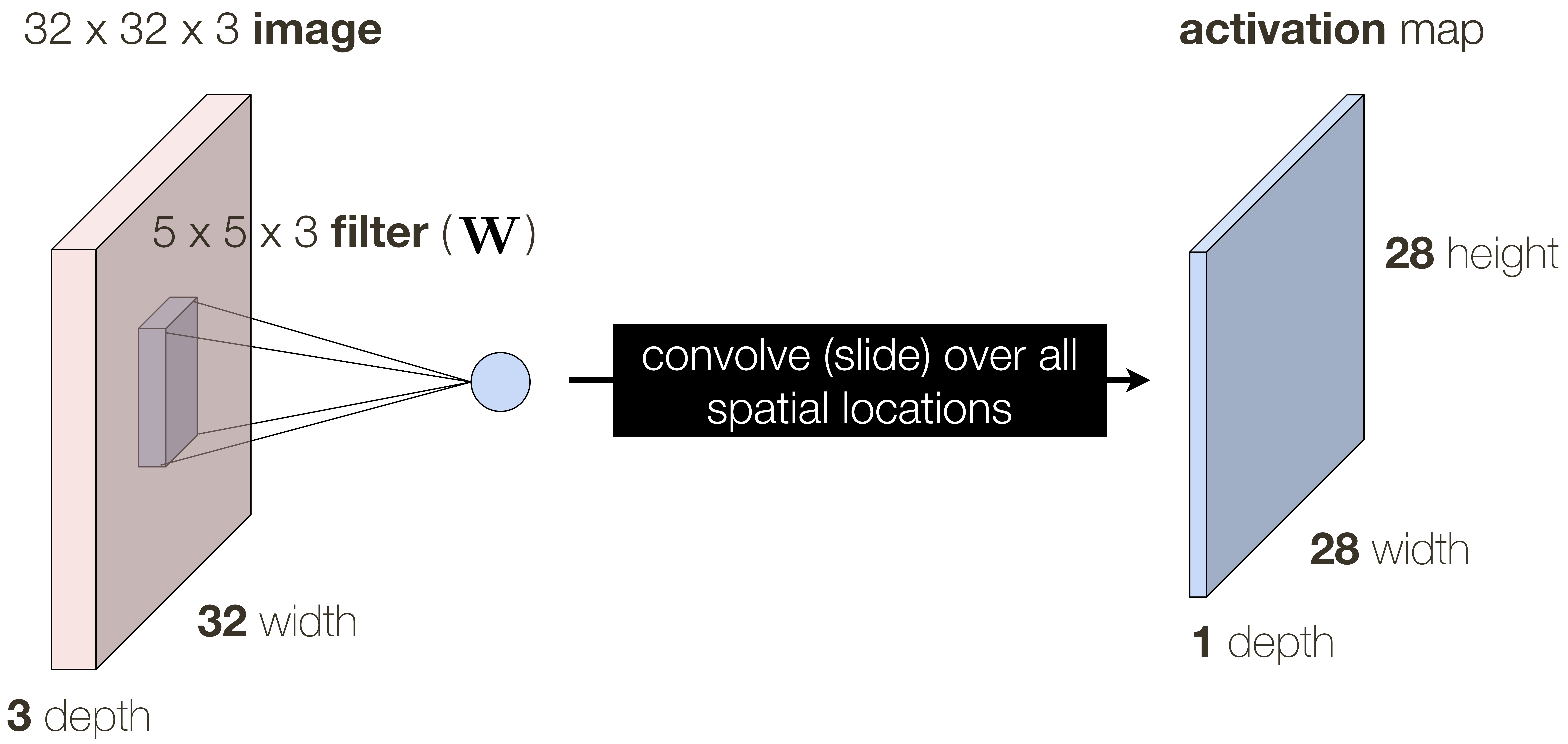
[Zeiler and Fergus, 2013]

What **filters** do networks learn?



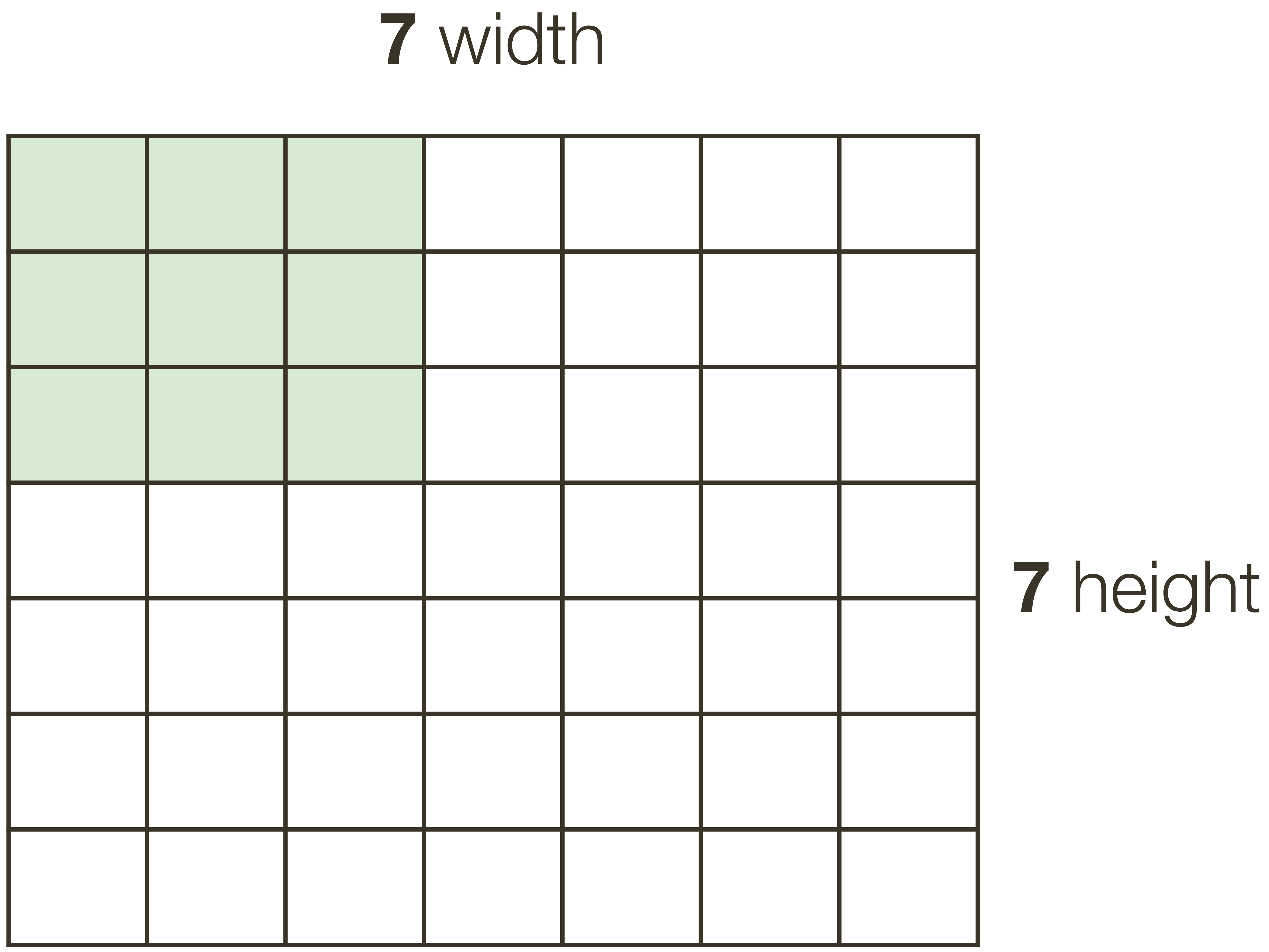
[Zeiler and Fergus, 2013]

Convolutional Layer: Closer Look at **Spatial Dimensions**



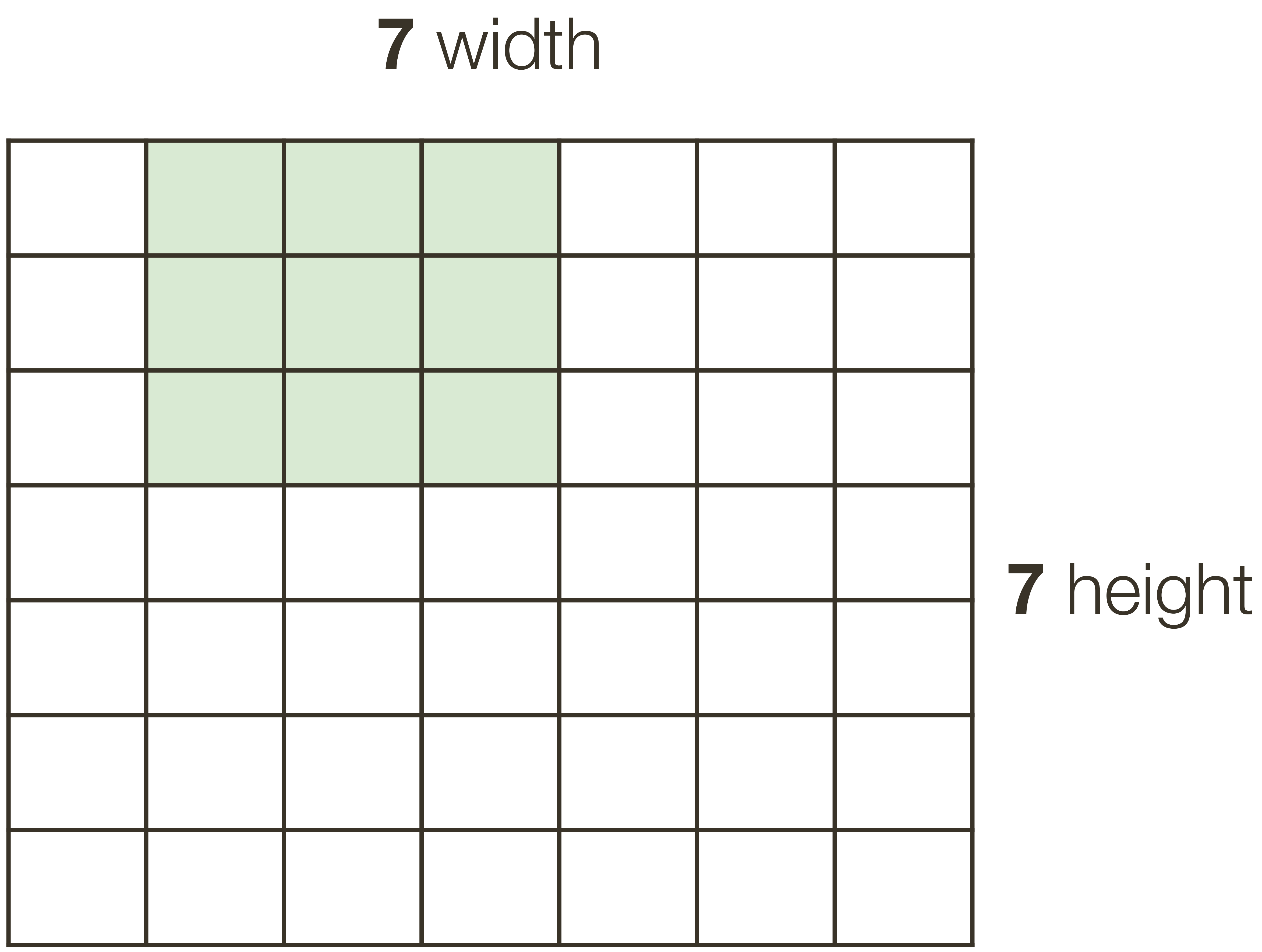
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer: Closer Look at **Spatial Dimensions**



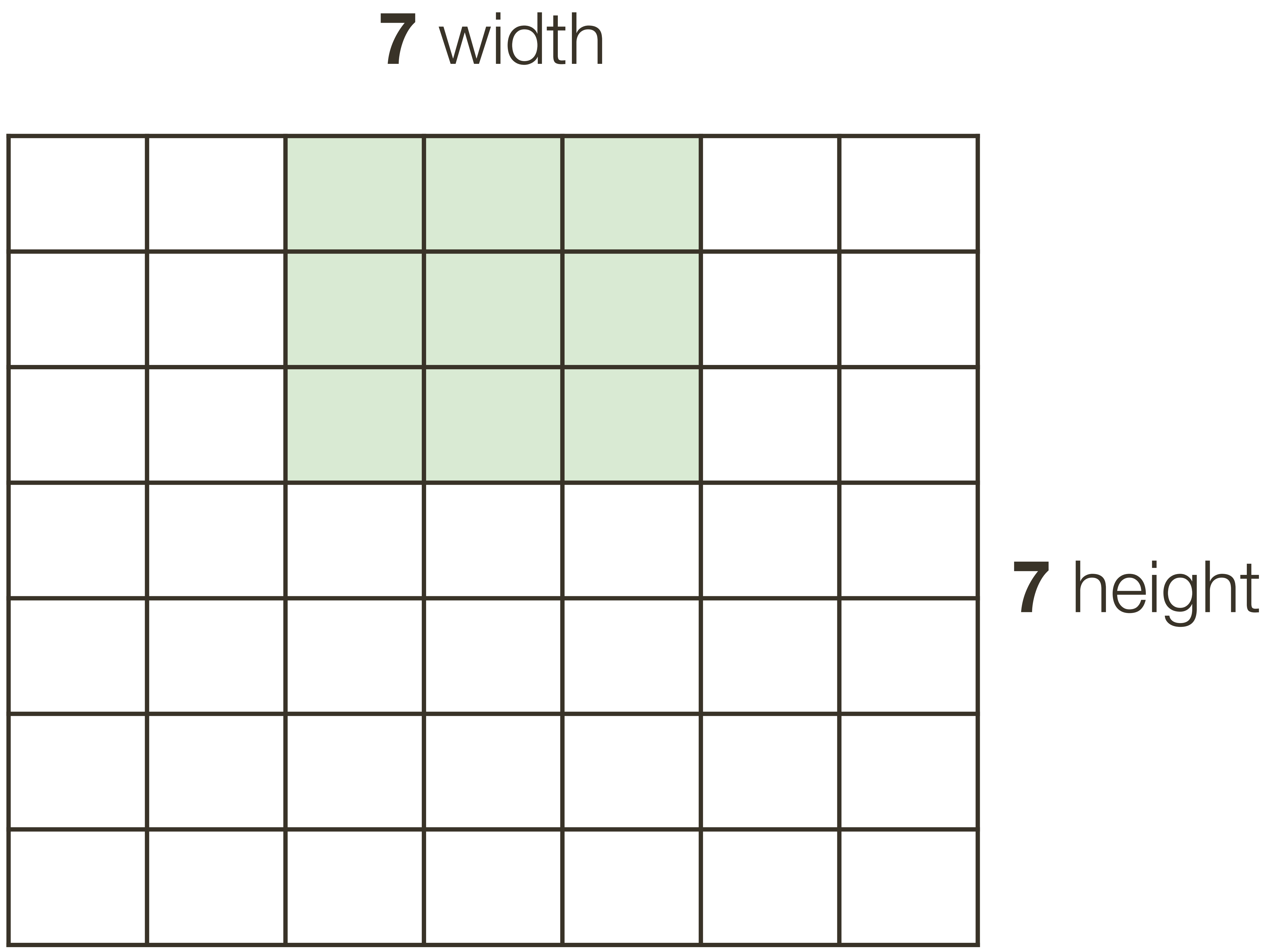
7 x 7 input image (spatially)
3 x 3 filter

Convolutional Layer: Closer Look at **Spatial Dimensions**



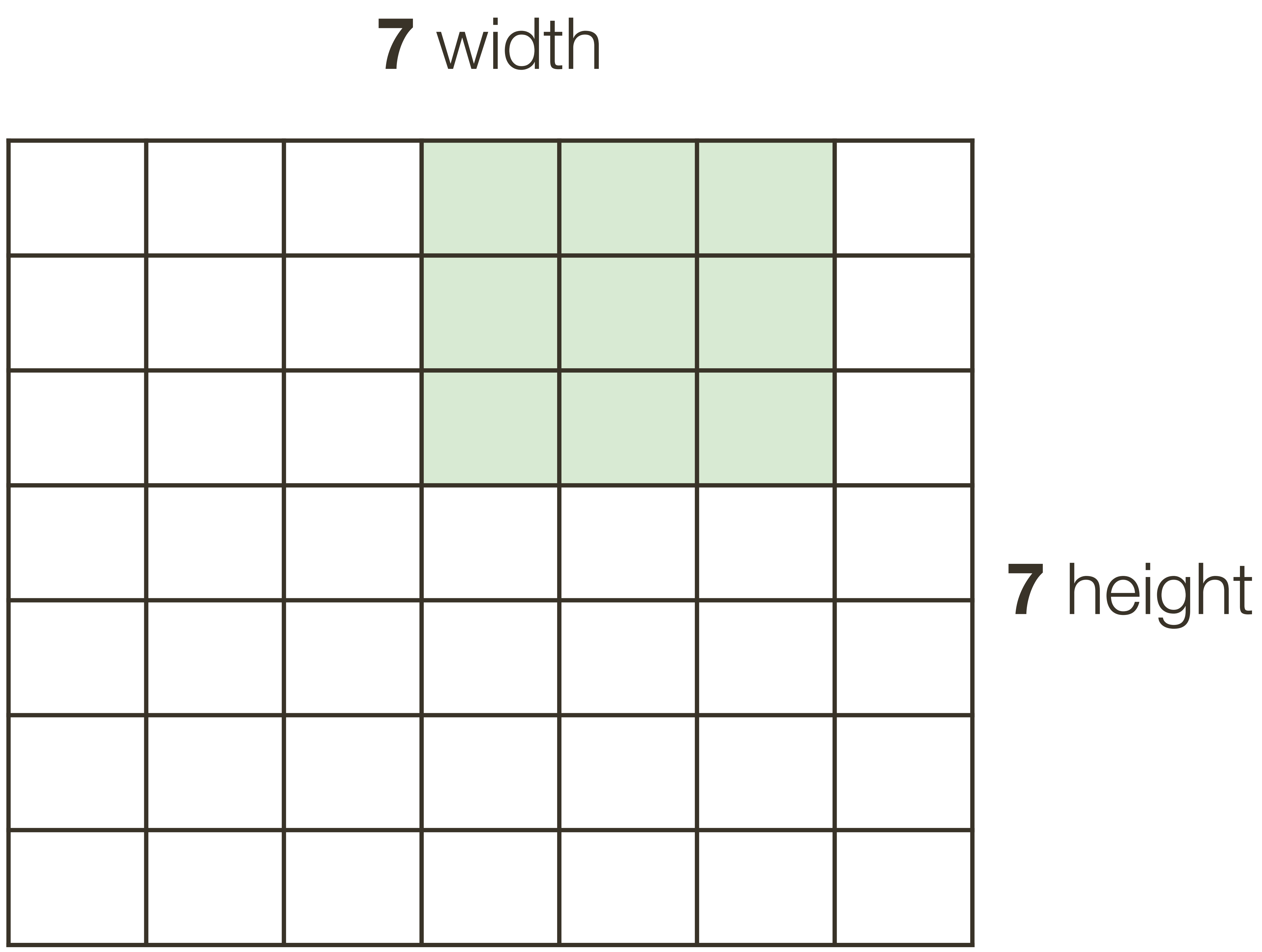
7 x 7 input image (spatially)
3 x 3 filter

Convolutional Layer: Closer Look at **Spatial Dimensions**



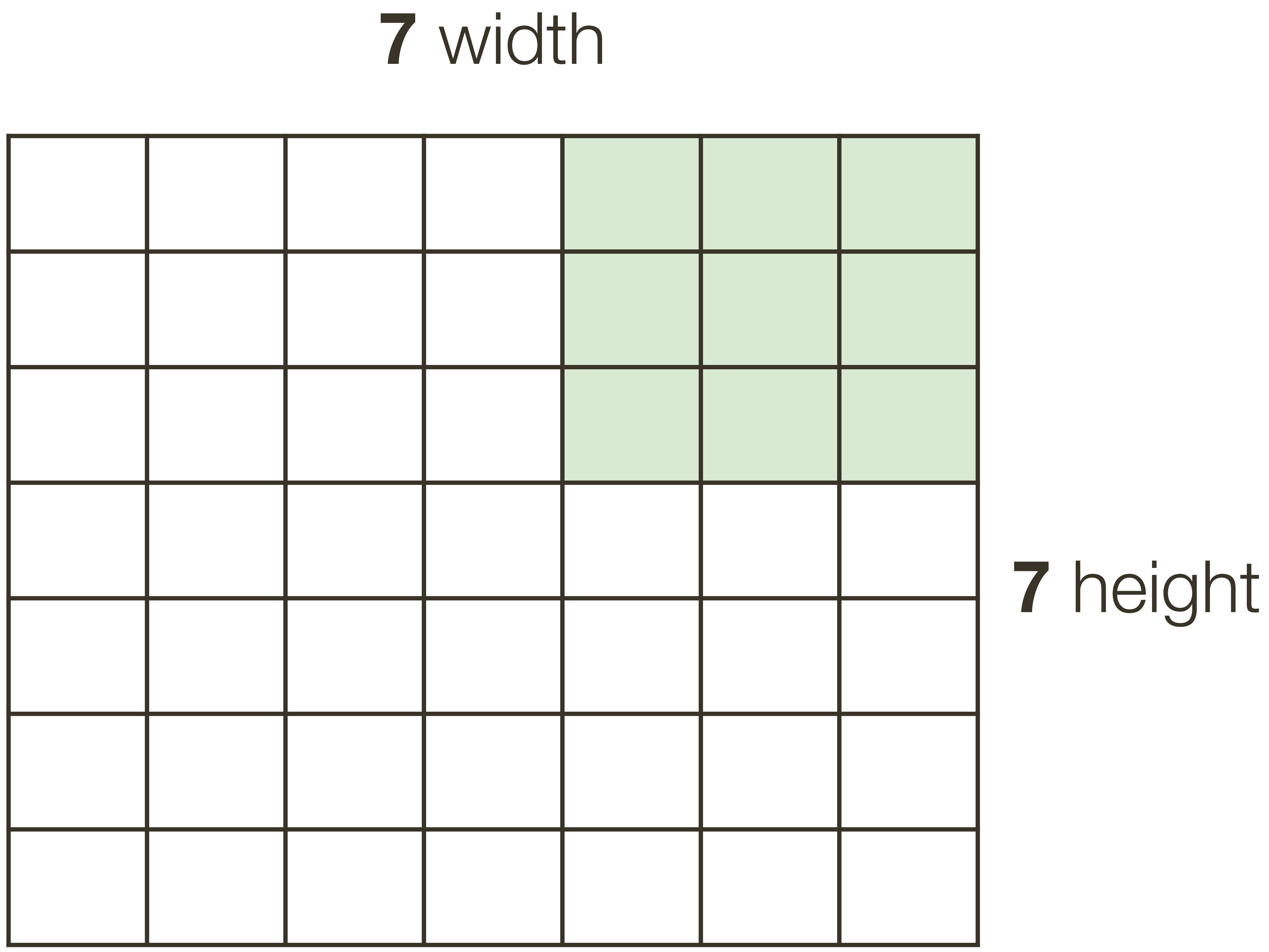
7 x 7 input image (spatially)
3 x 3 filter

Convolutional Layer: Closer Look at **Spatial Dimensions**



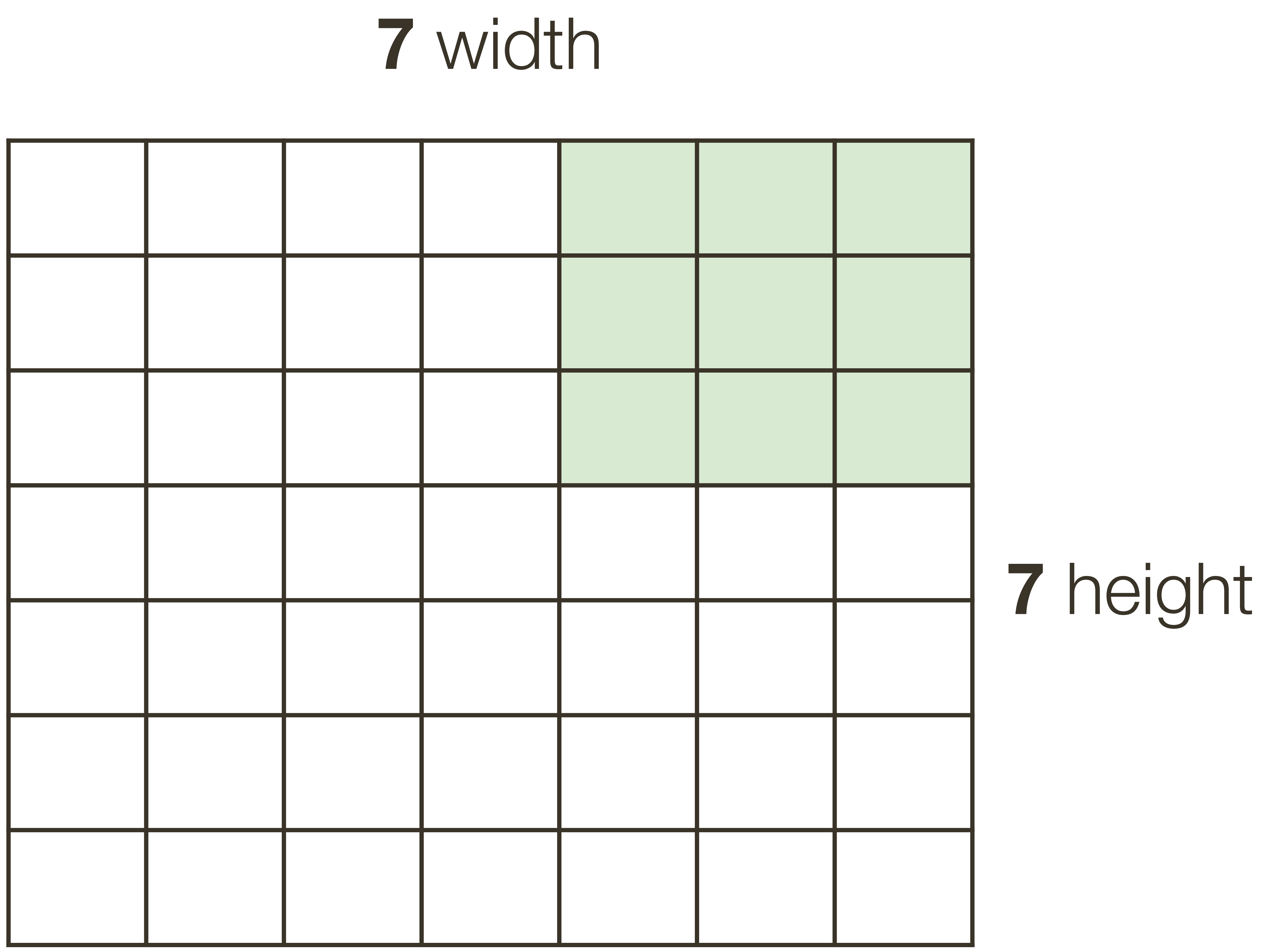
7 x 7 input image (spatially)
3 x 3 filter

Convolutional Layer: Closer Look at **Spatial Dimensions**



7 x 7 input image (spatially)
3 x 3 filter

Convolutional Layer: Closer Look at **Spatial Dimensions**

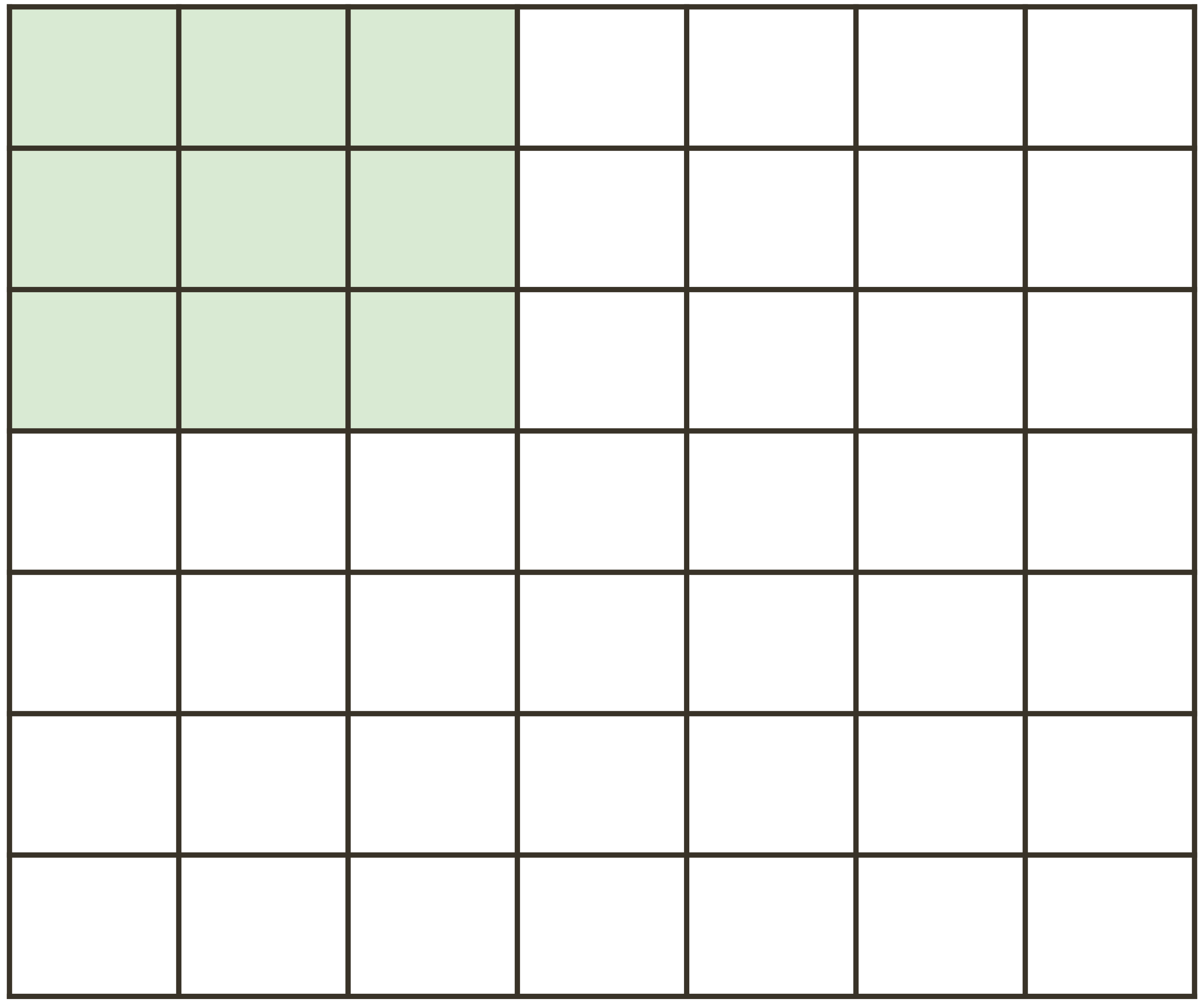


7 x 7 input image (spatially)
3 x 3 filter

=> **5 x 5 output**

Convolutional Layer: Closer Look at **Spatial Dimensions**

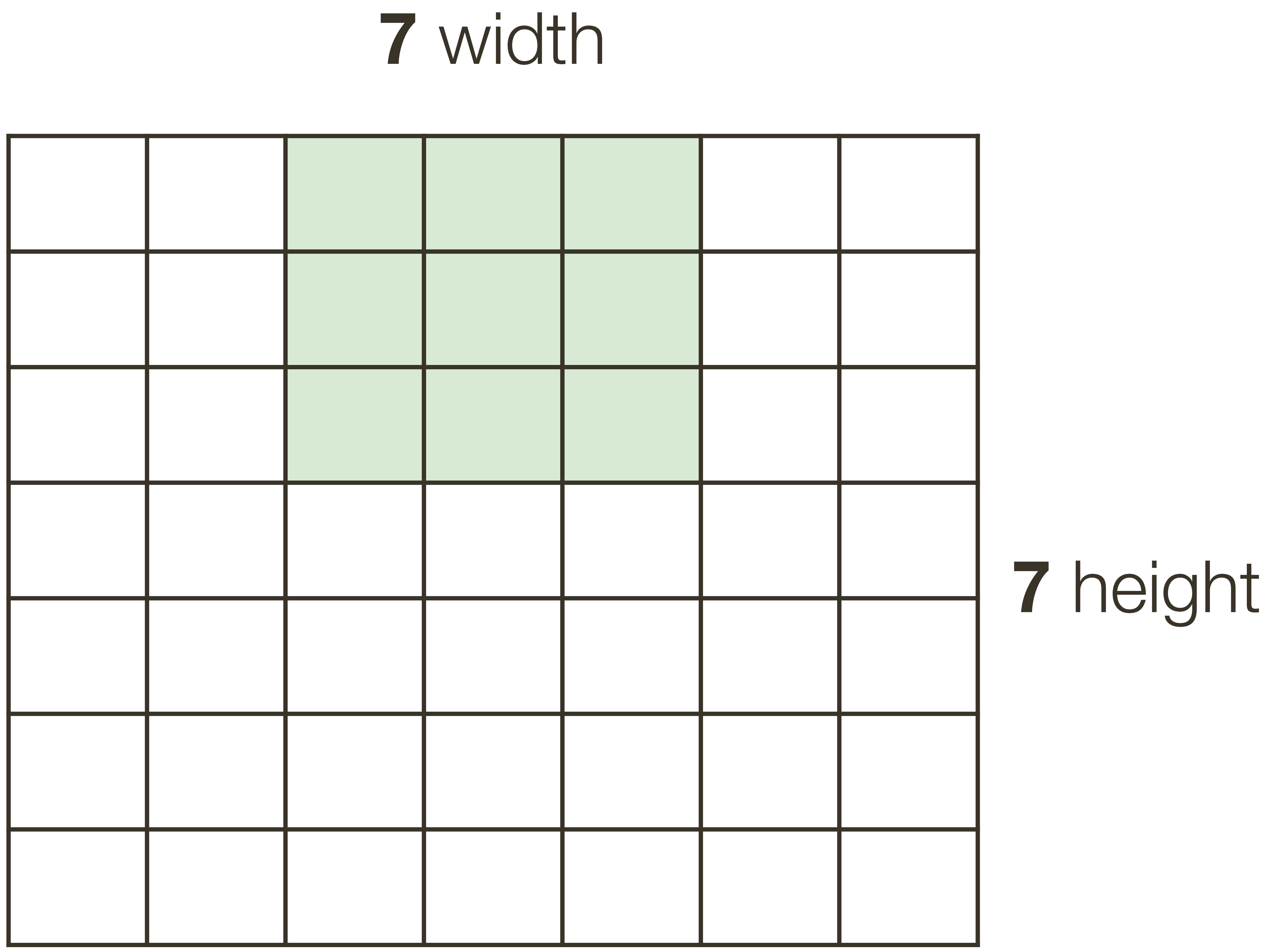
7 width



7 height

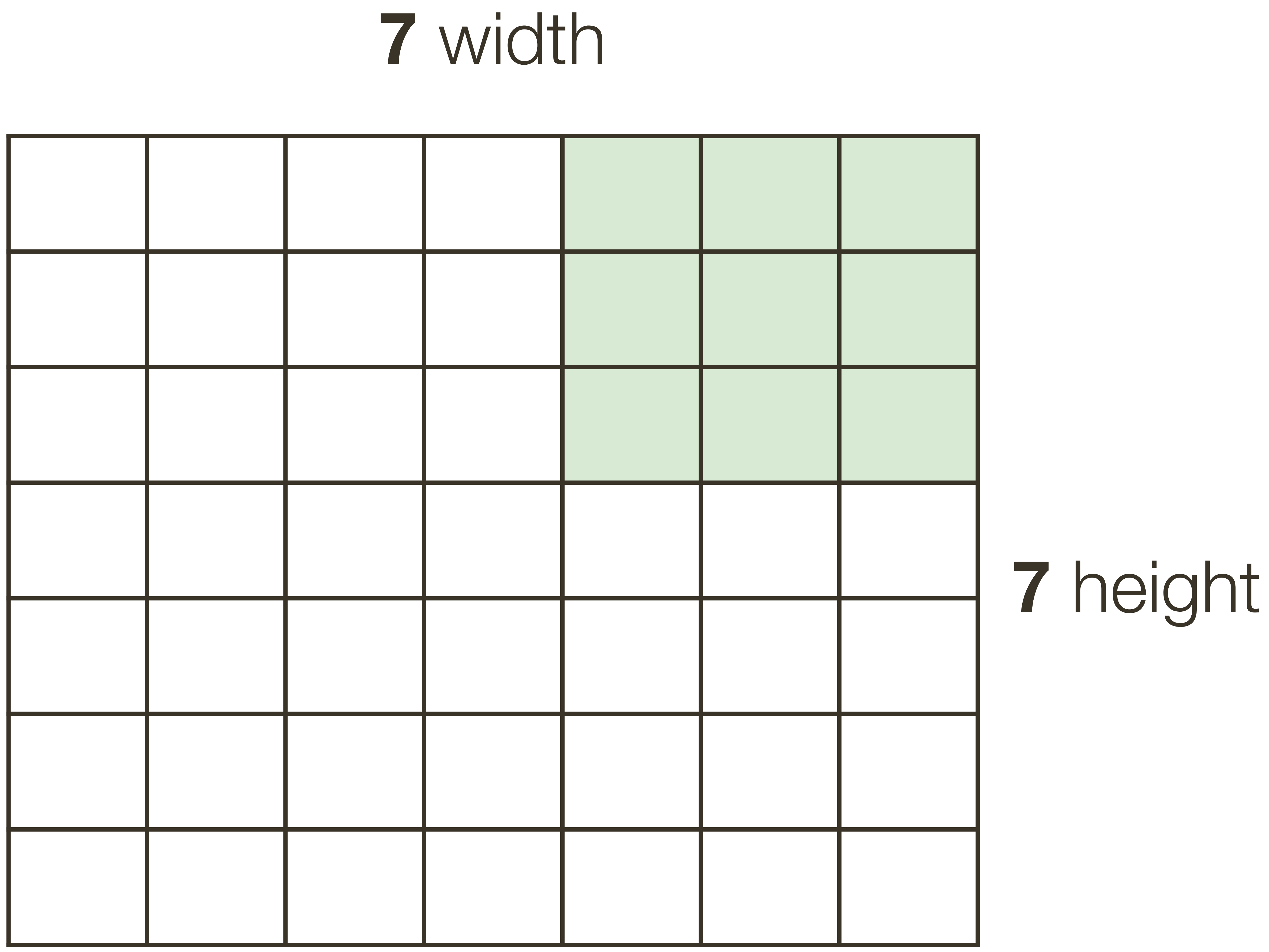
7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

Convolutional Layer: Closer Look at **Spatial Dimensions**



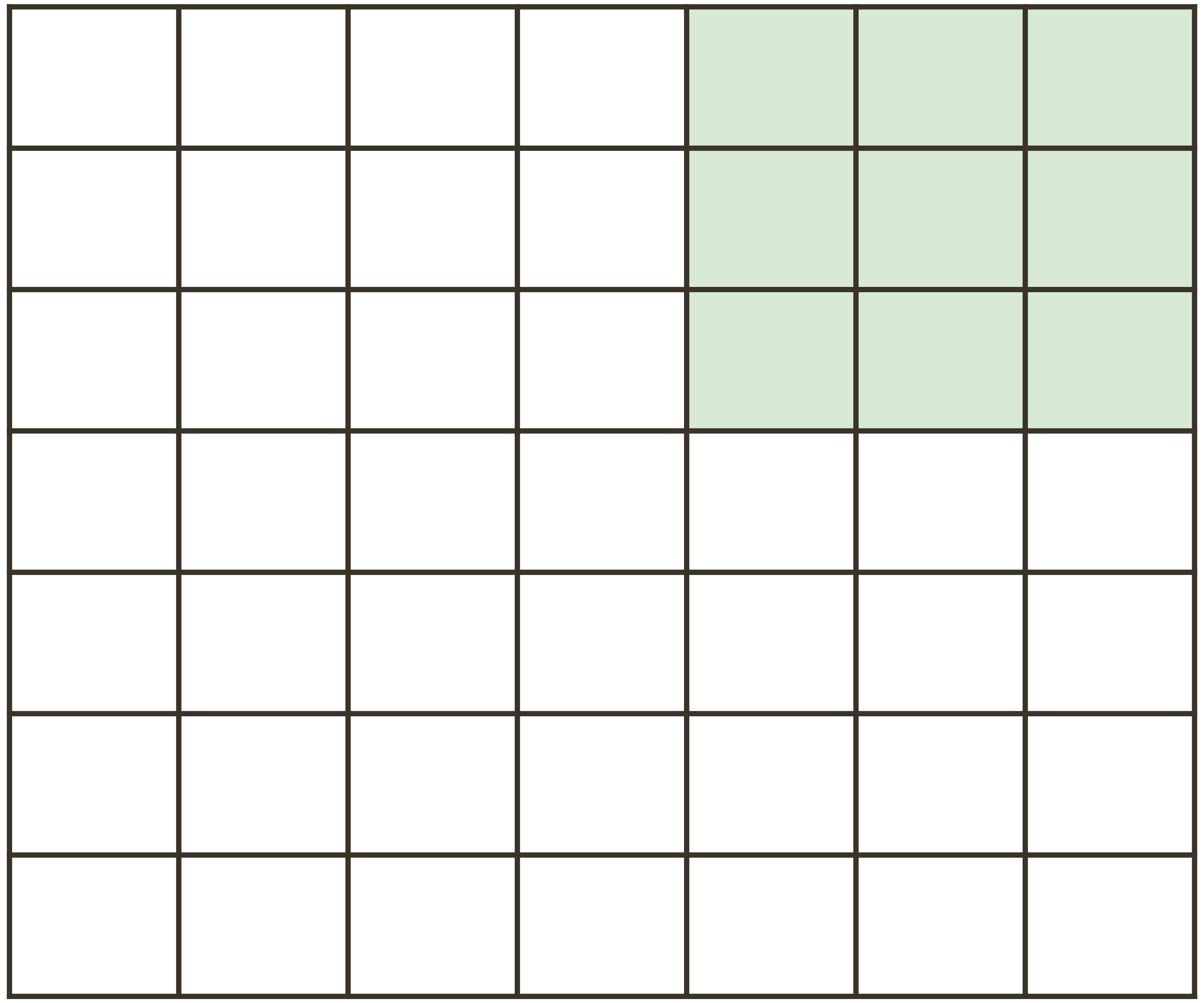
7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

Convolutional Layer: Closer Look at **Spatial Dimensions**



7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

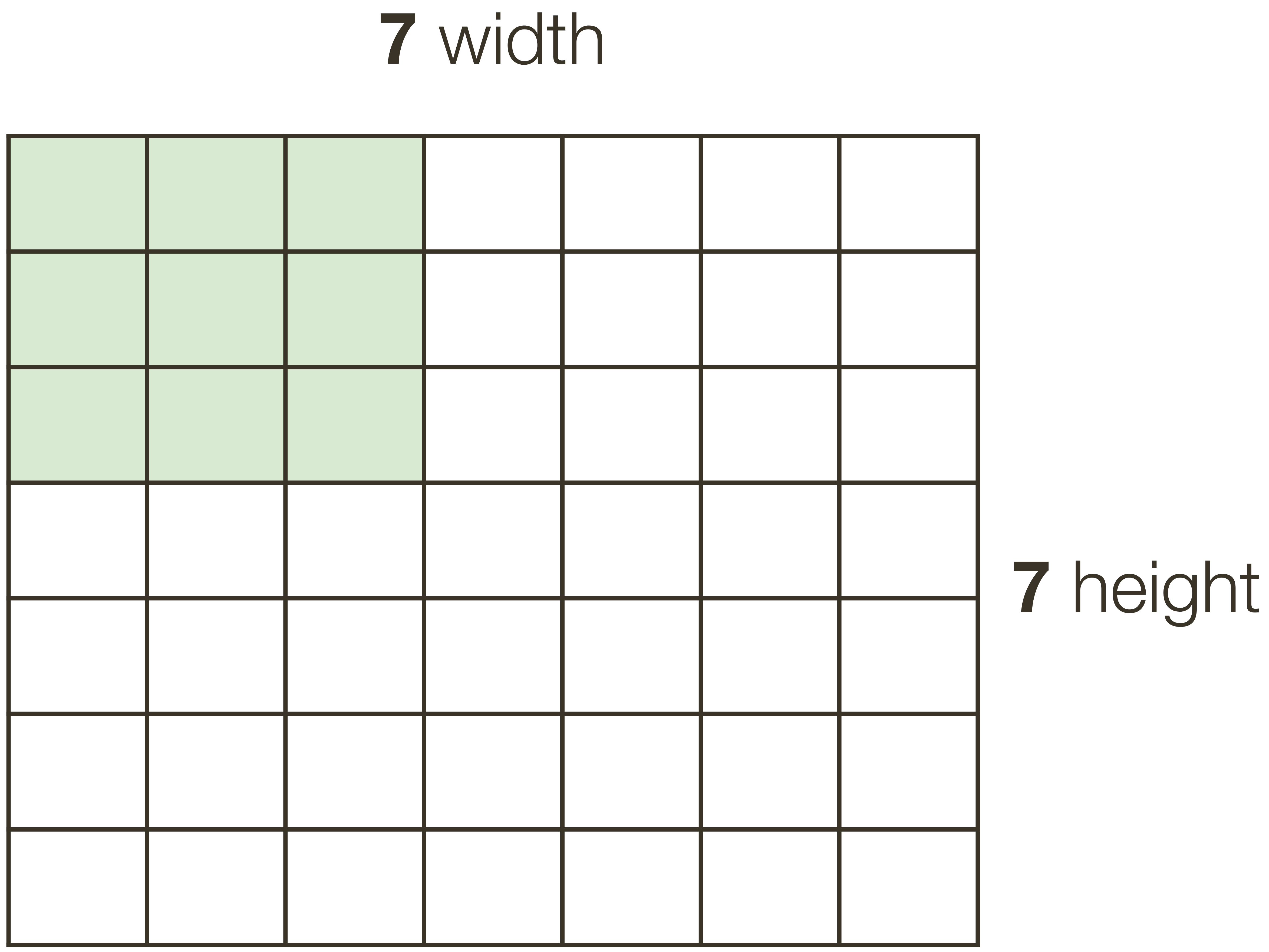
Convolutional Layer: Closer Look at **Spatial Dimensions**



7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

=> **3 x 3 output**

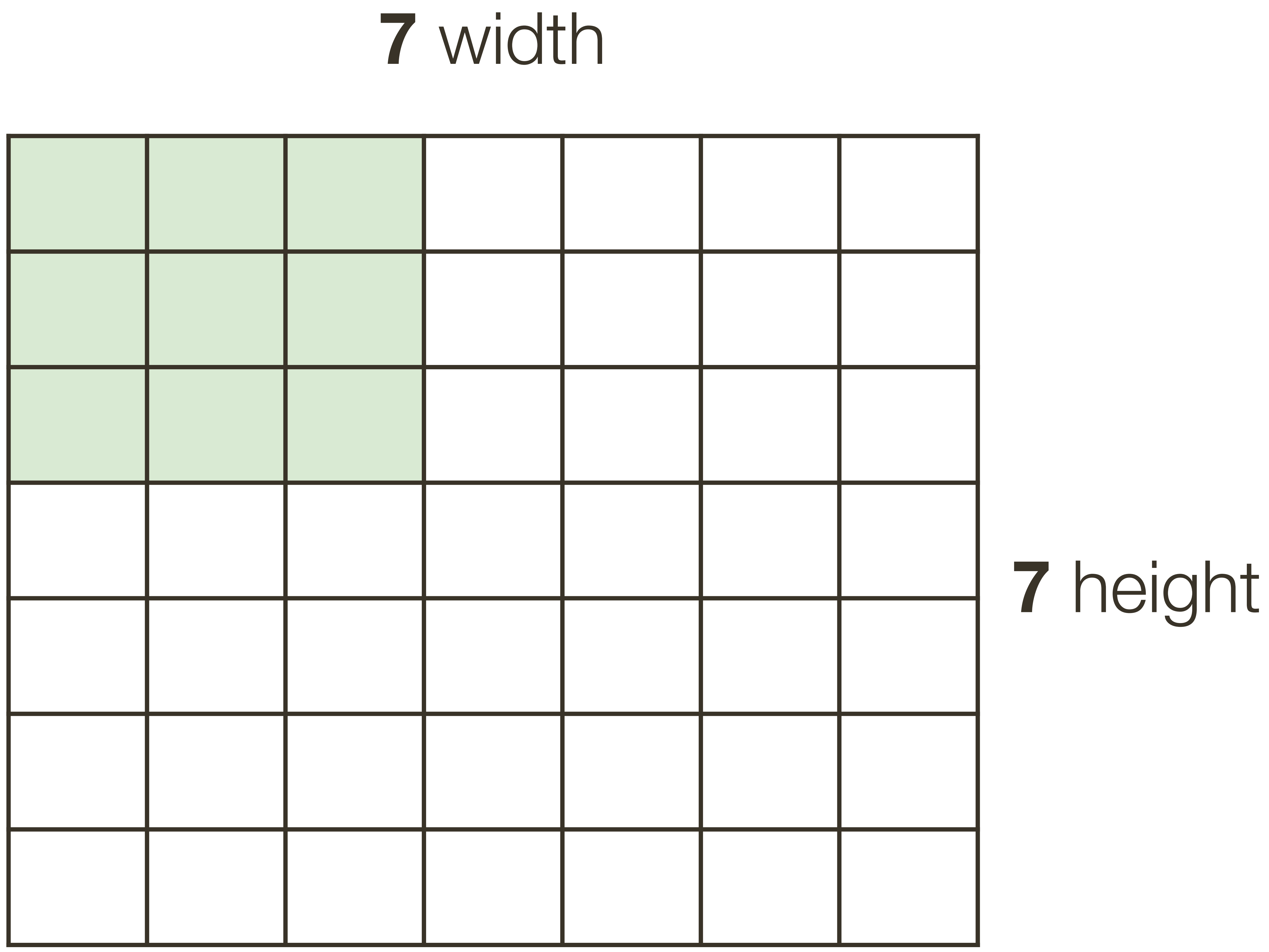
Convolutional Layer: Closer Look at **Spatial Dimensions**



7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 3**)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

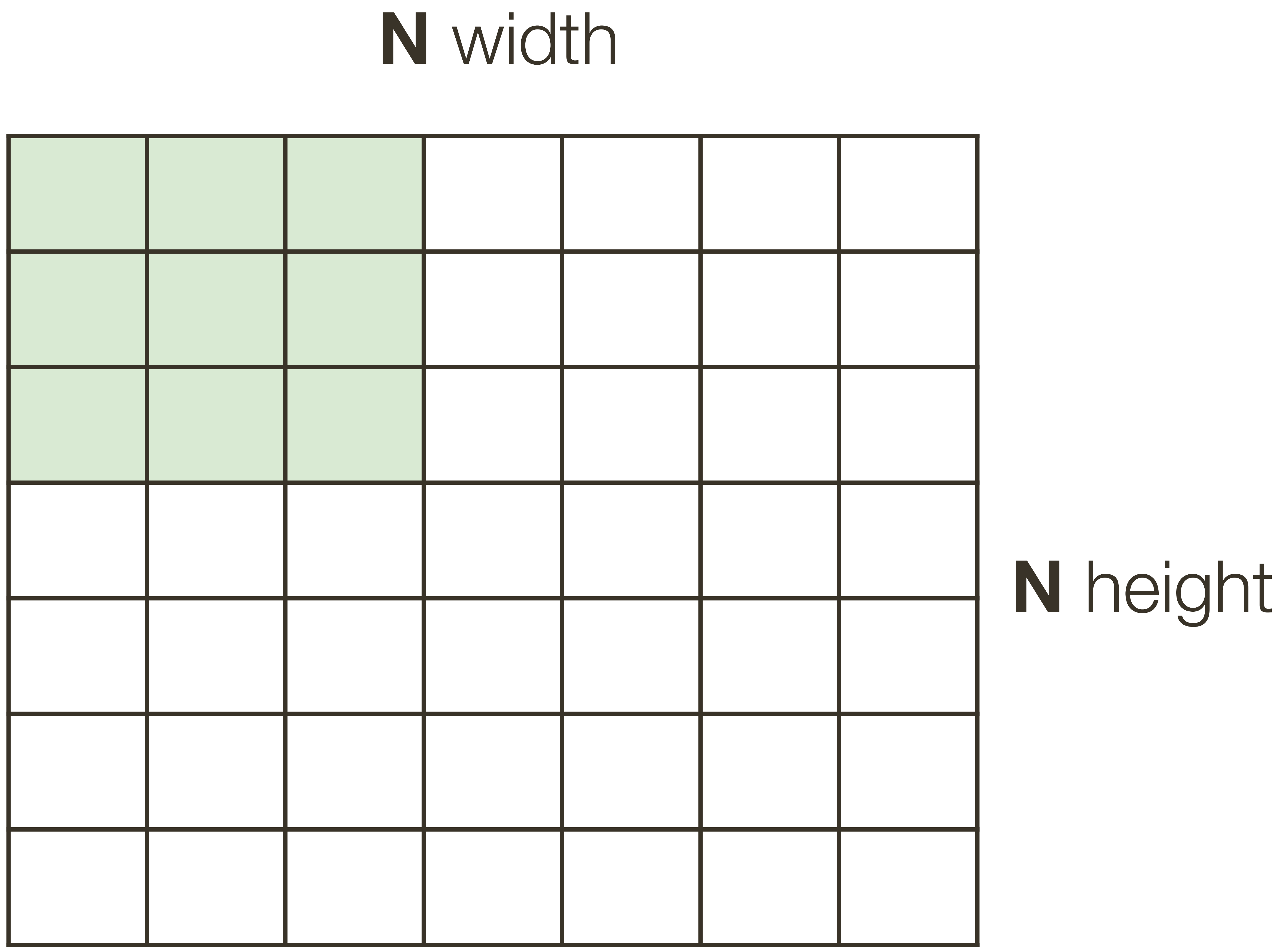
Convolutional Layer: Closer Look at **Spatial Dimensions**



7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 3**)

Does not fit! **Cannot apply** 3 x 3 filter on 7 x 7 image with stride 3

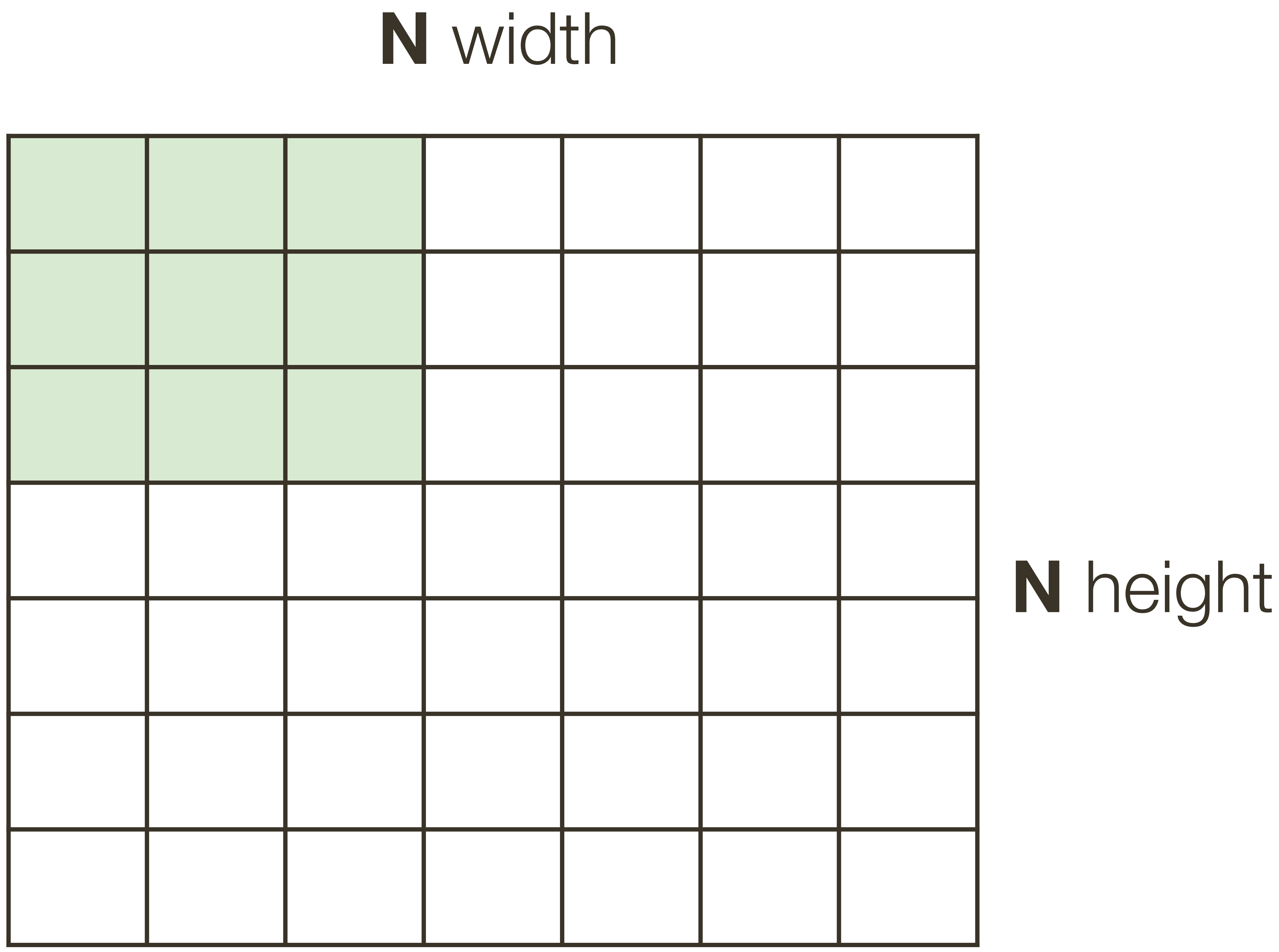
Convolutional Layer: Closer Look at **Spatial Dimensions**



N x N input image (spatially)
F x F filter

Output size: (N-F) / stride + 1

Convolutional Layer: Closer Look at **Spatial Dimensions**



N x N input image (spatially)
F x F filter

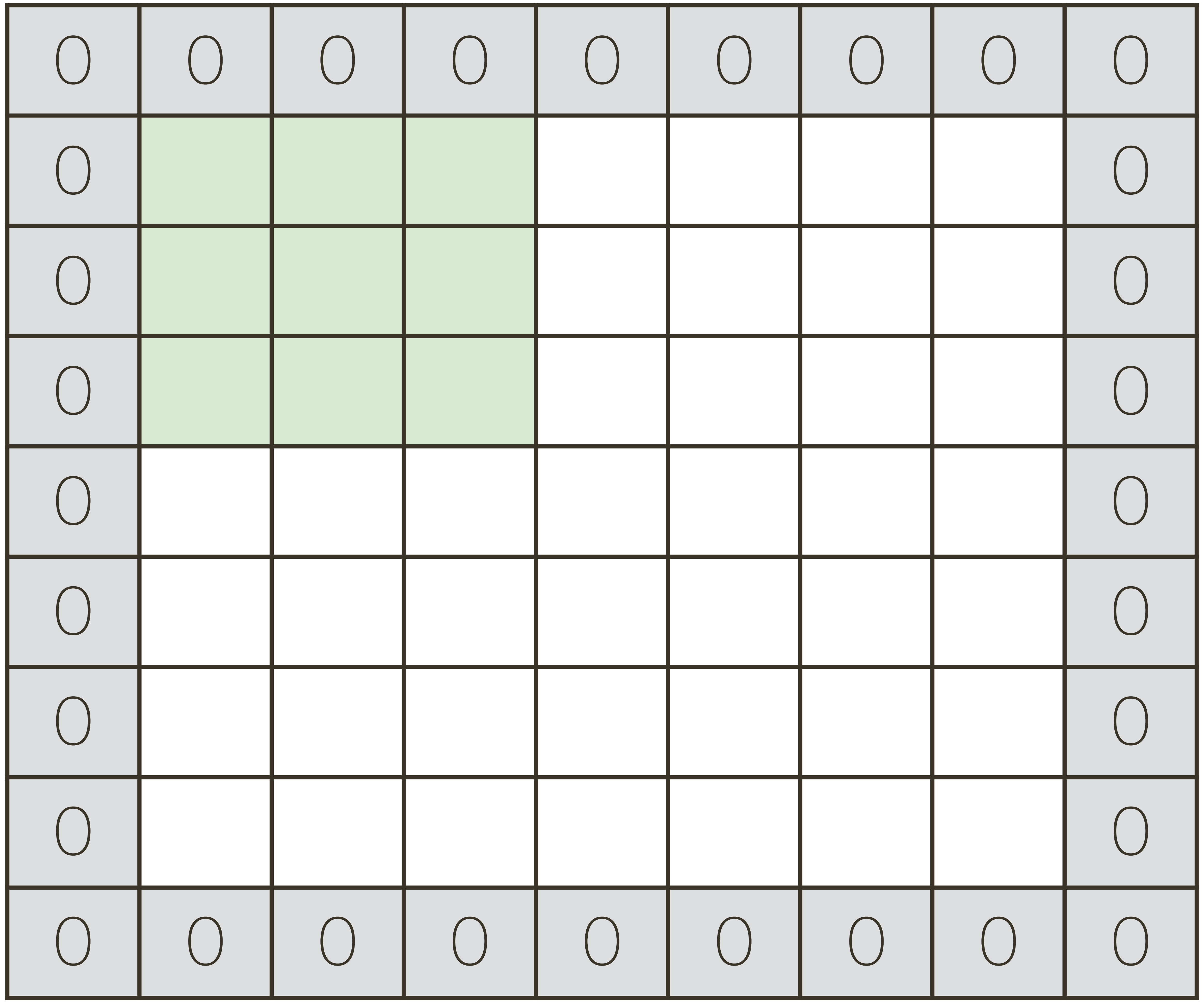
Output size: (N-F) / stride + 1

Example: N = 7, F = 3

stride 1 => (7-3)/1+1 = 5
stride 2 => (7-3)/2+1 = 3
stride 3 => (7-3)/3+1 = **2.33**

Convolutional Layer: **Border padding**

7 width



7 height

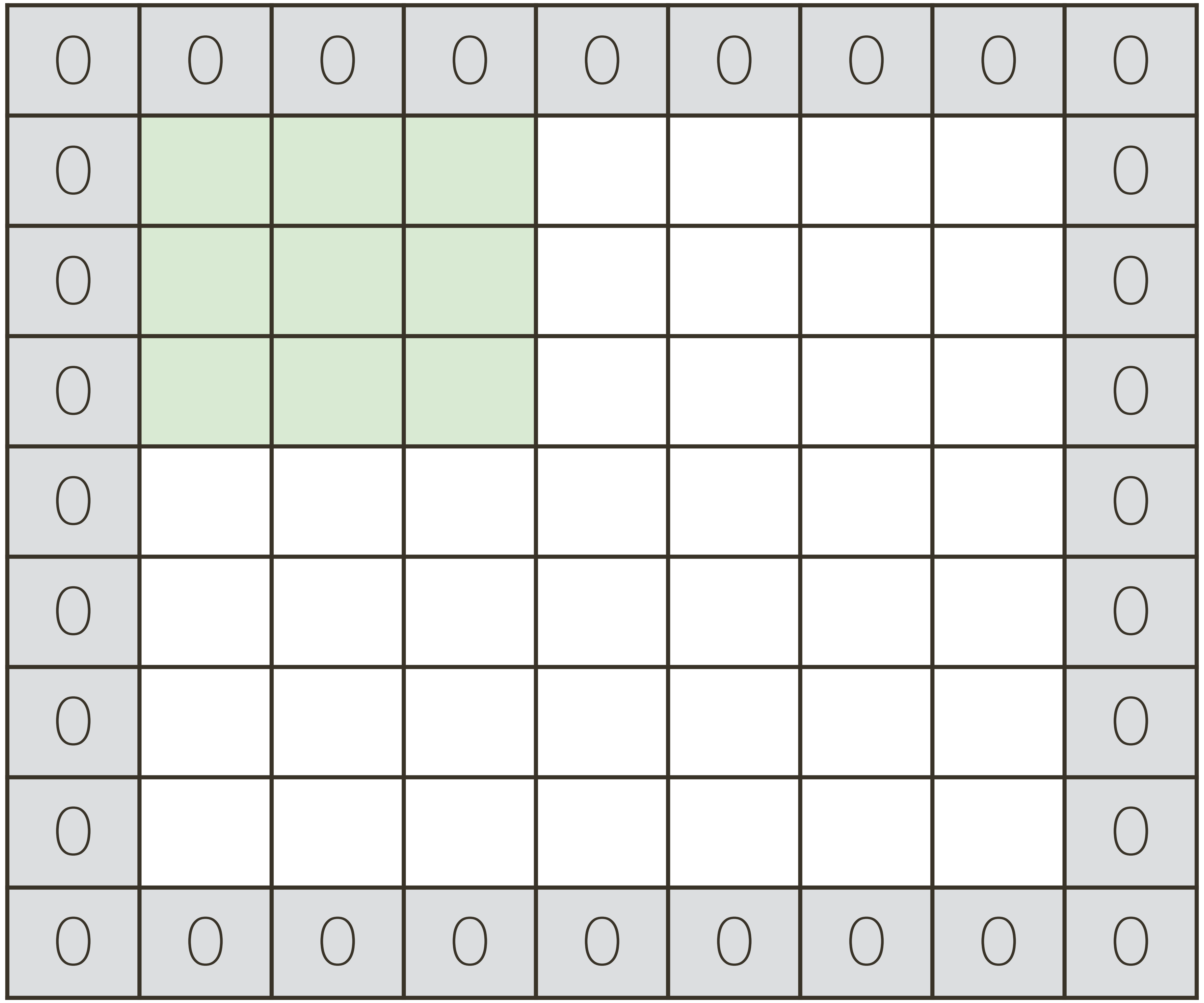
7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 1**)

pad with 1 pixel border

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer: **Border padding**

7 width



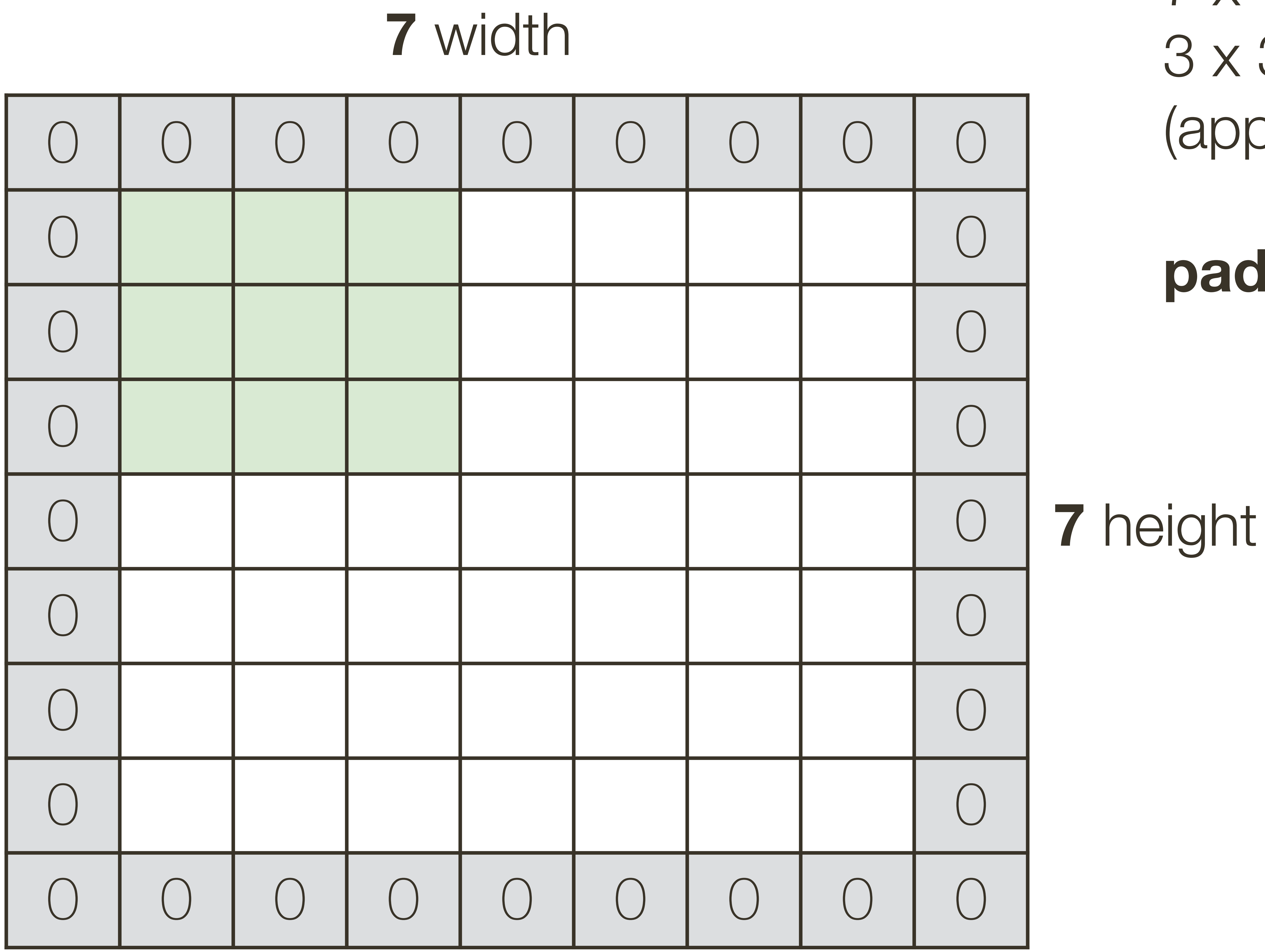
7 height

7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 1**)

pad with 1 pixel border

Output size: 7 x 7

Convolutional Layer: **Border padding**

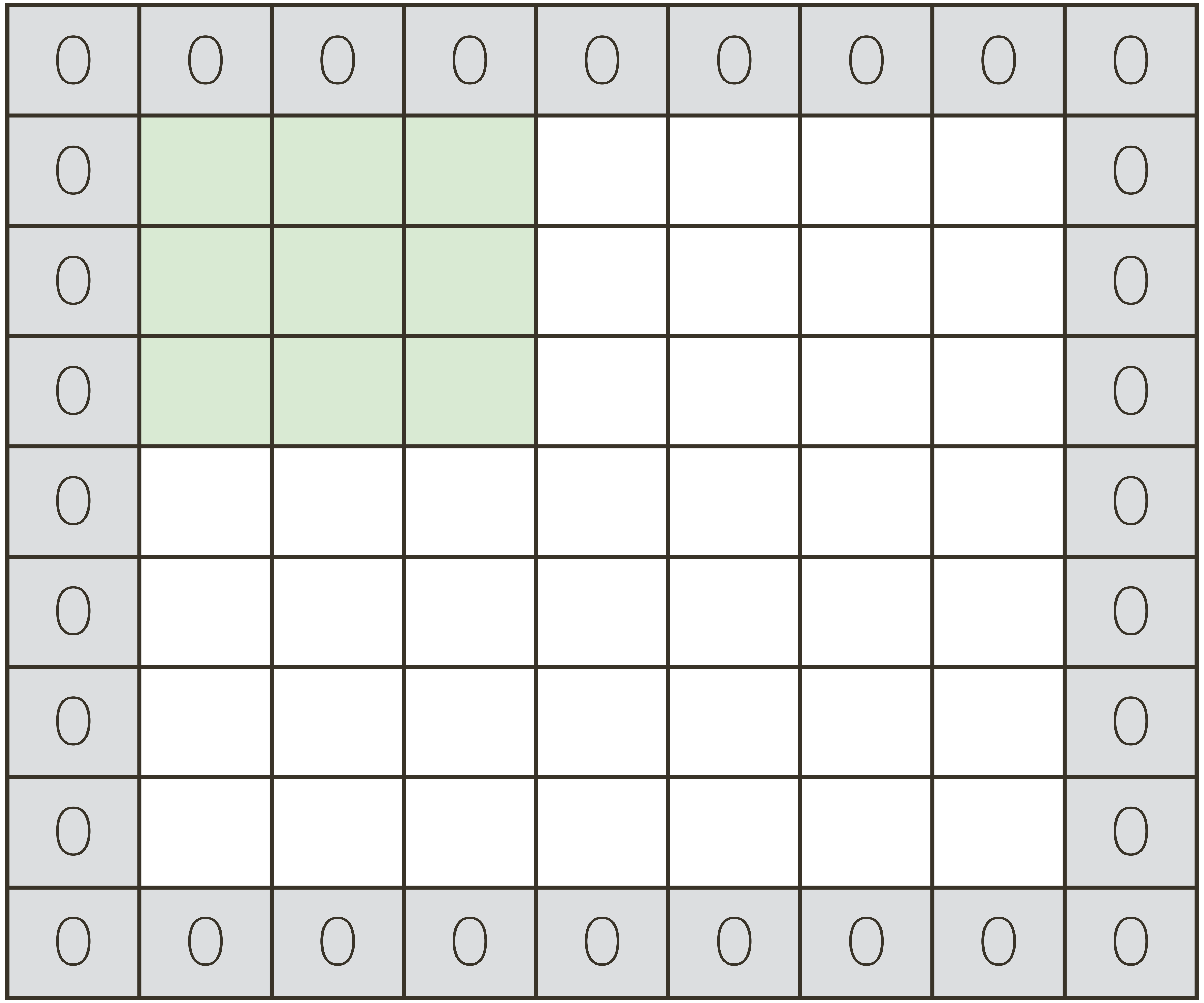


7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 3**)

pad with 1 pixel border

Convolutional Layer: **Border padding**

7 width



7 height

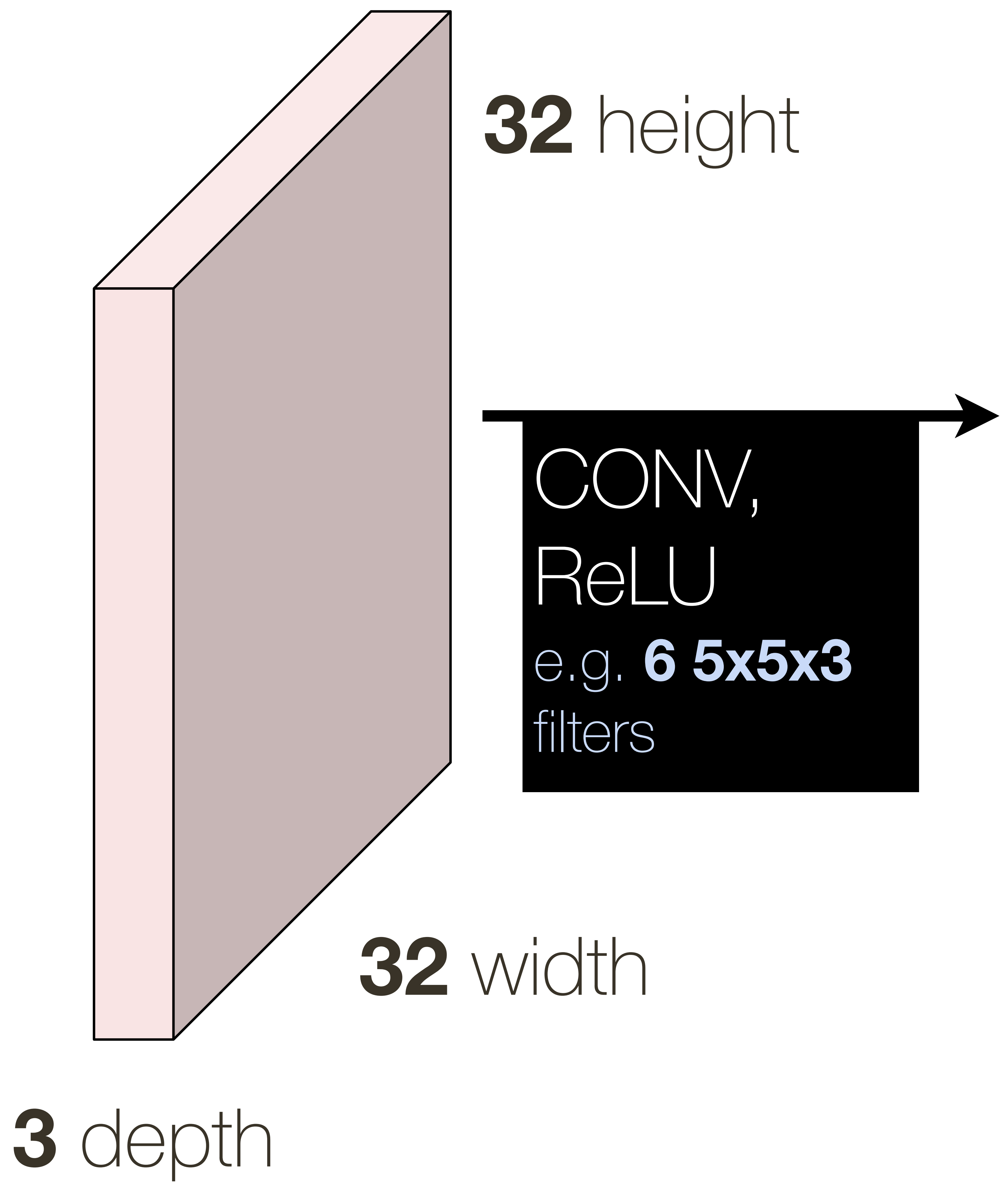
7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 3**)

pad with 1 pixel border

Example: N = 7, F = 3

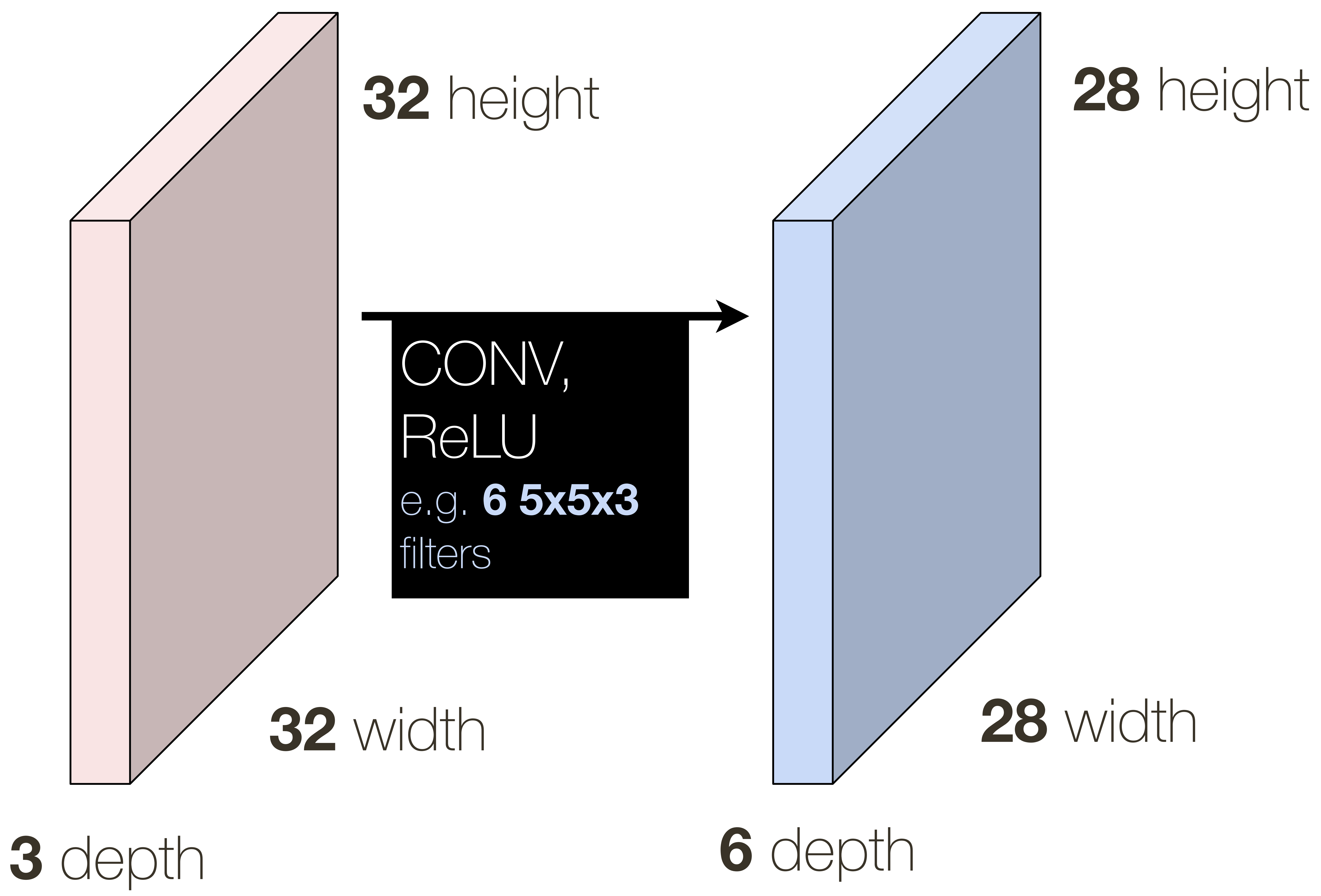
stride 1 => (9-3)/1+1 = 7
stride 2 => (9-3)/2+1 = 4
stride 3 => (9-3)/3+1 = **3**

Convolutional Neural Network (ConvNet)



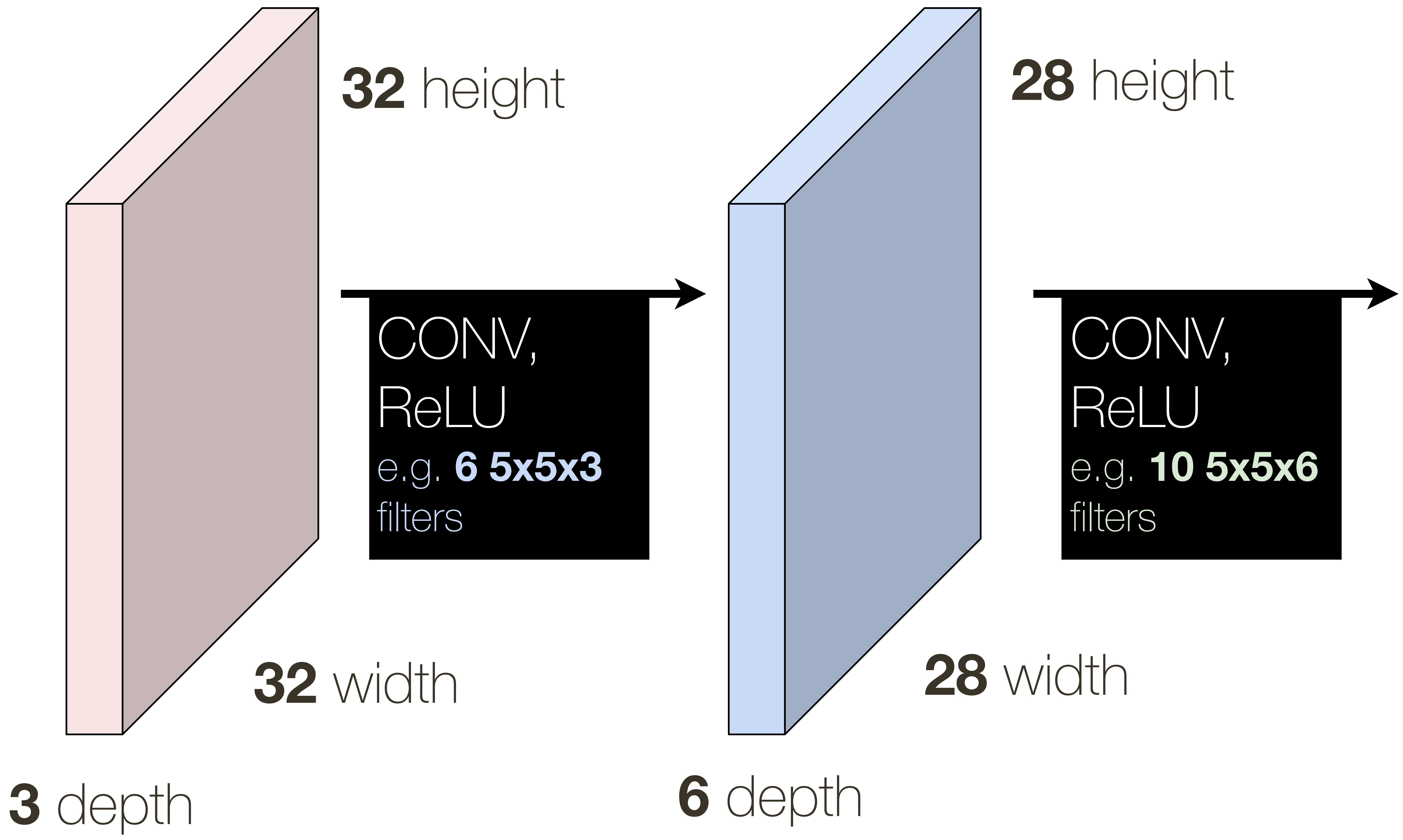
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



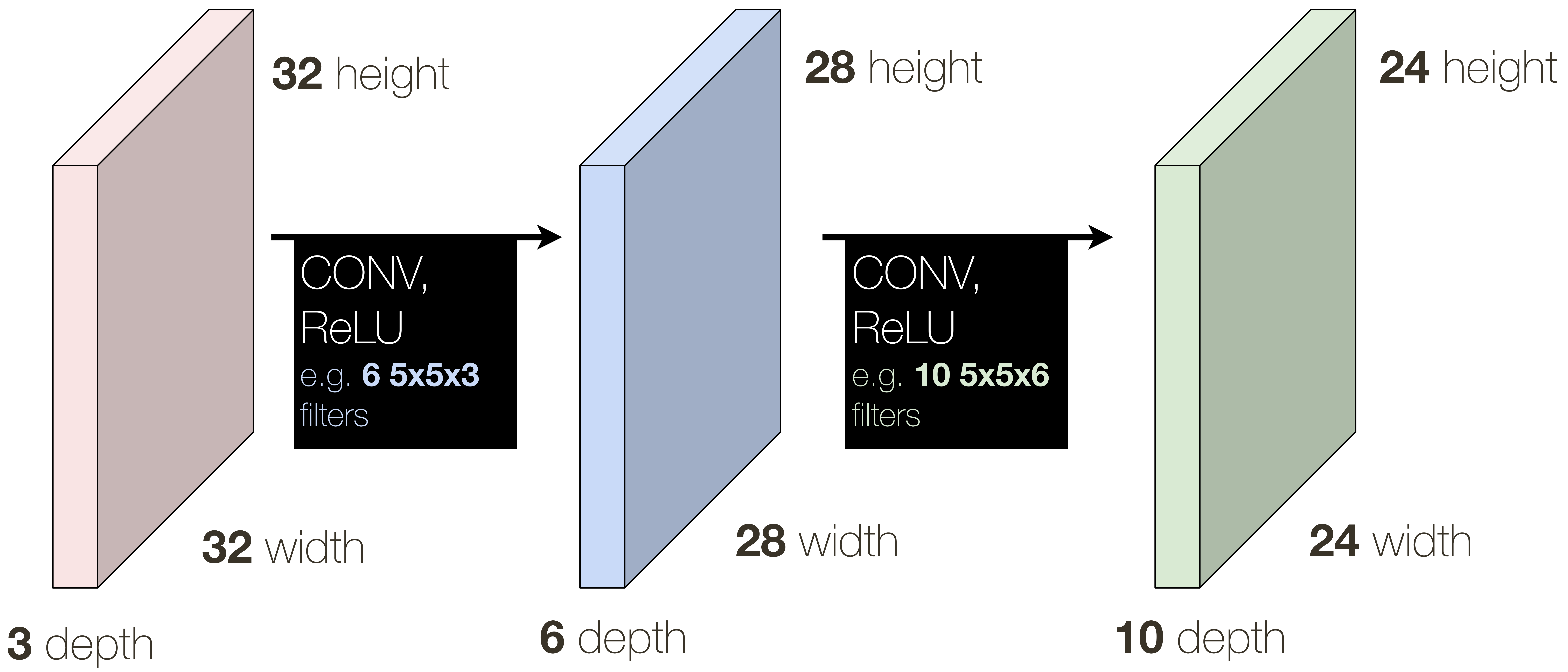
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



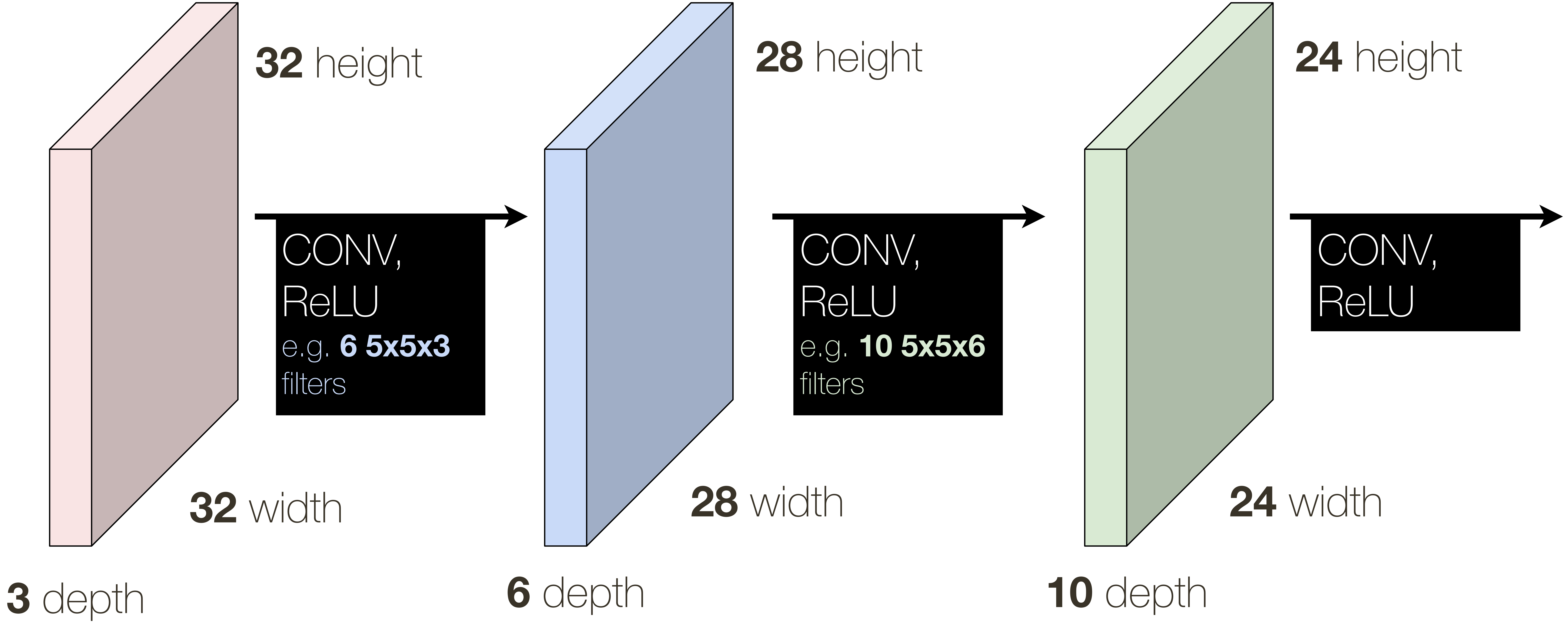
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

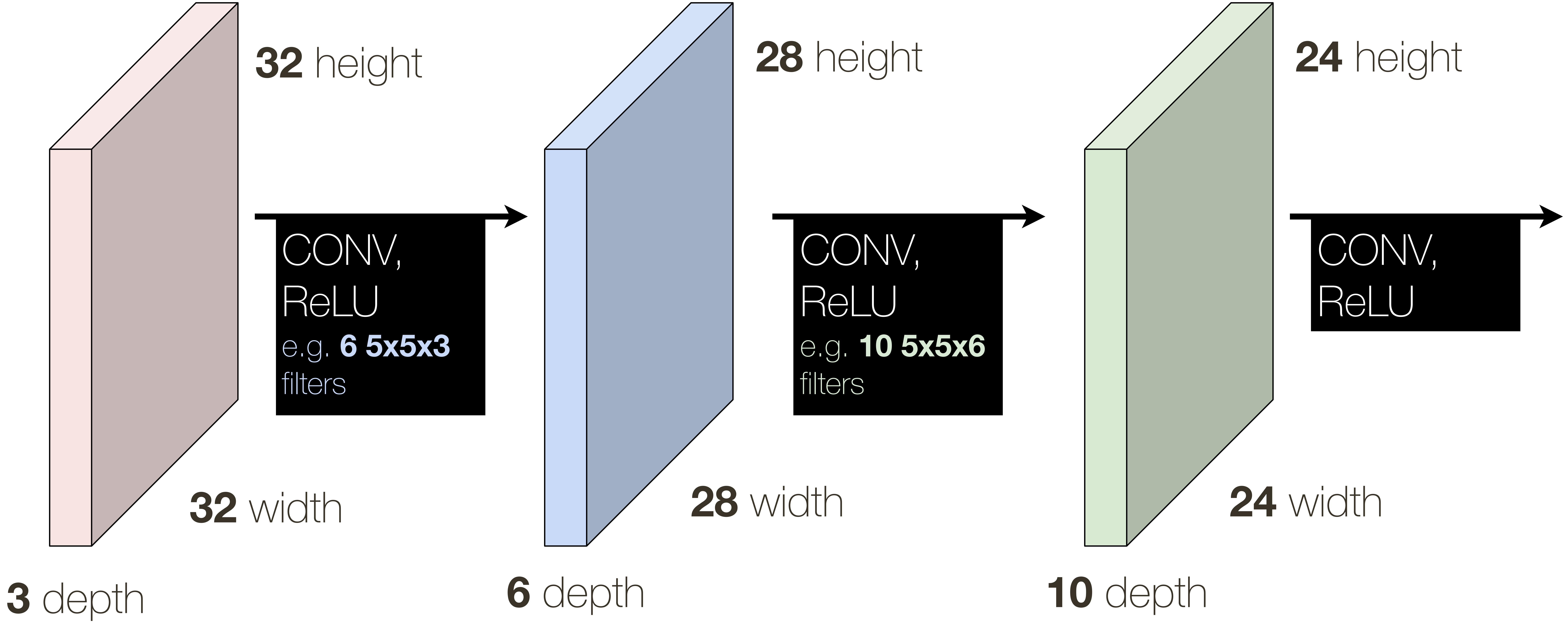
Convolutional Neural Network (ConvNet)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

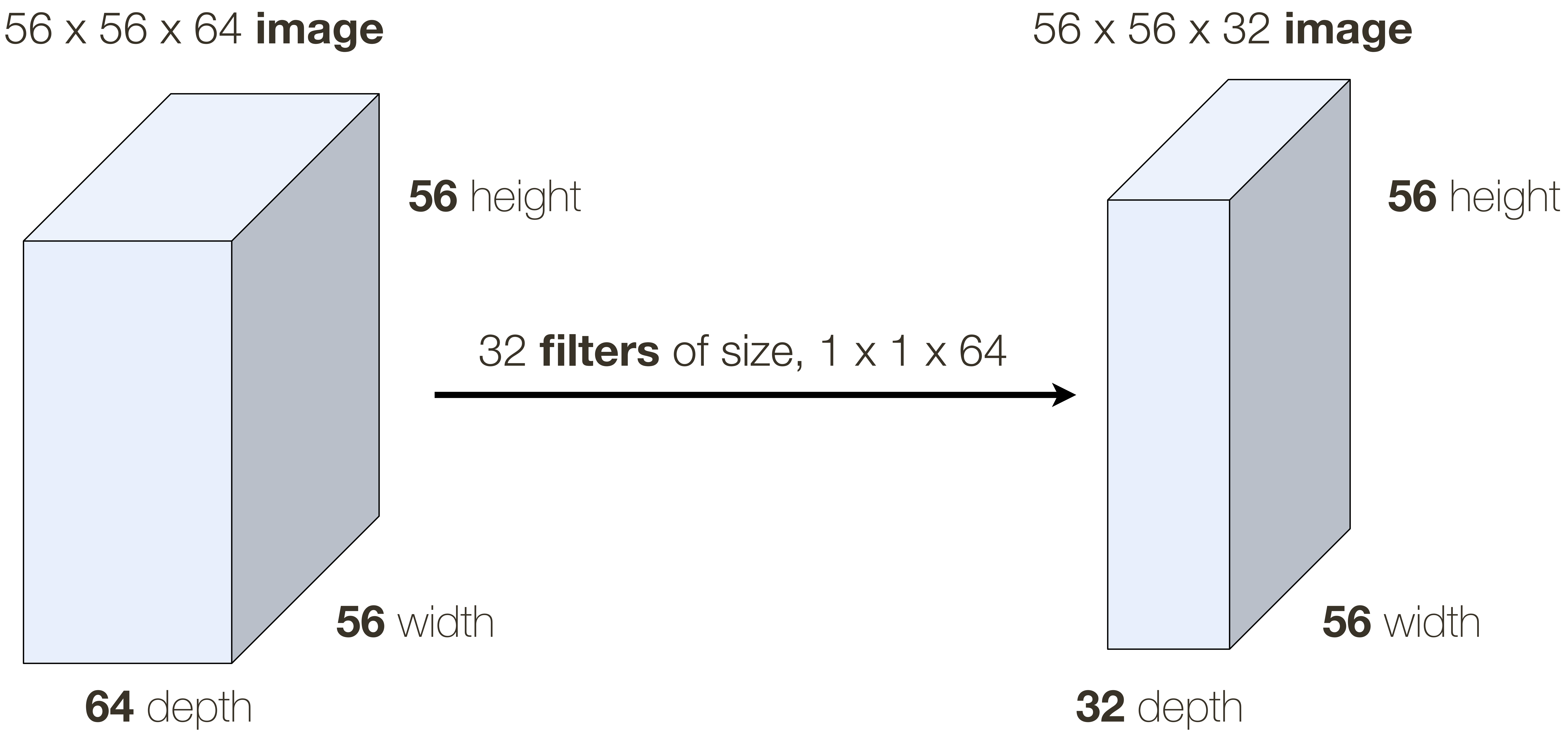
Convolutional Neural Network (ConvNet)

With padding we can achieve no shrinking (32 -> 28 -> 24); shrinking quickly (which happens with larger filters) doesn't work well in practice



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer: **1x1 convolutions**



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Requires hyperparameters:

— Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)

— Stride of application: S (for a typical network $S \in \{1, 2\}$)

— Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Requires hyperparameters:

— Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)

— Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)

— Stride of application: S (for a typical network $S \in \{1, 2\}$)

— Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Requires hyperparameters:

- Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)
- Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)
- Stride of application: S (for a typical network $S \in \{1, 2\}$)
- Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$

$$W_o = (W_i - F + 2P)/S + 1 \qquad H_o = (H_i - F + 2P)/S + 1 \qquad D_o = K$$

Convolutional Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Requires hyperparameters:

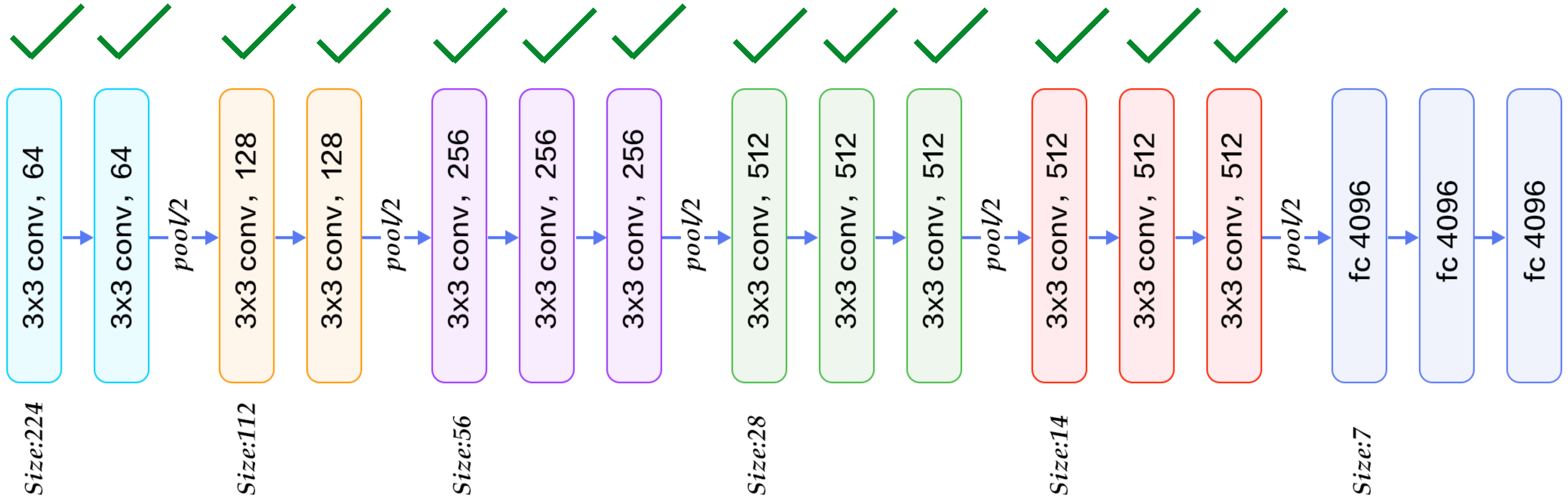
- Number of filters: K (for typical networks $K \in \{32, 64, 128, 256, 512\}$)
- Spatial extent of filters: F (for a typical networks $F \in \{1, 3, 5, ...\}$)
- Stride of application: S (for a typical network $S \in \{1, 2\}$)
- Zero padding: P (for a typical network $P \in \{0, 1, 2\}$)

Produces a volume of size: $W_o \times H_o \times D_o$

$$W_o = (W_i - F + 2P)/S + 1 \qquad H_o = (H_i - F + 2P)/S + 1 \qquad D_o = K$$

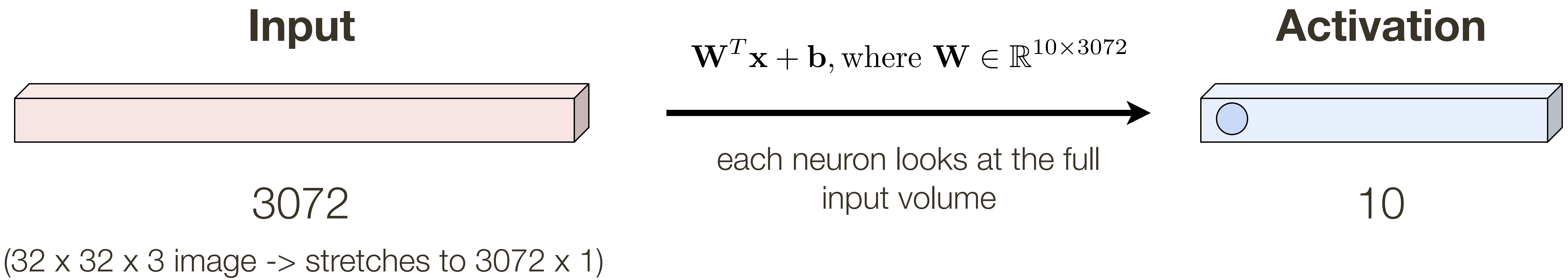
Number of total learnable parameters: $(F \times F \times D_i) \times K + K$

Convolutional Neural Networks



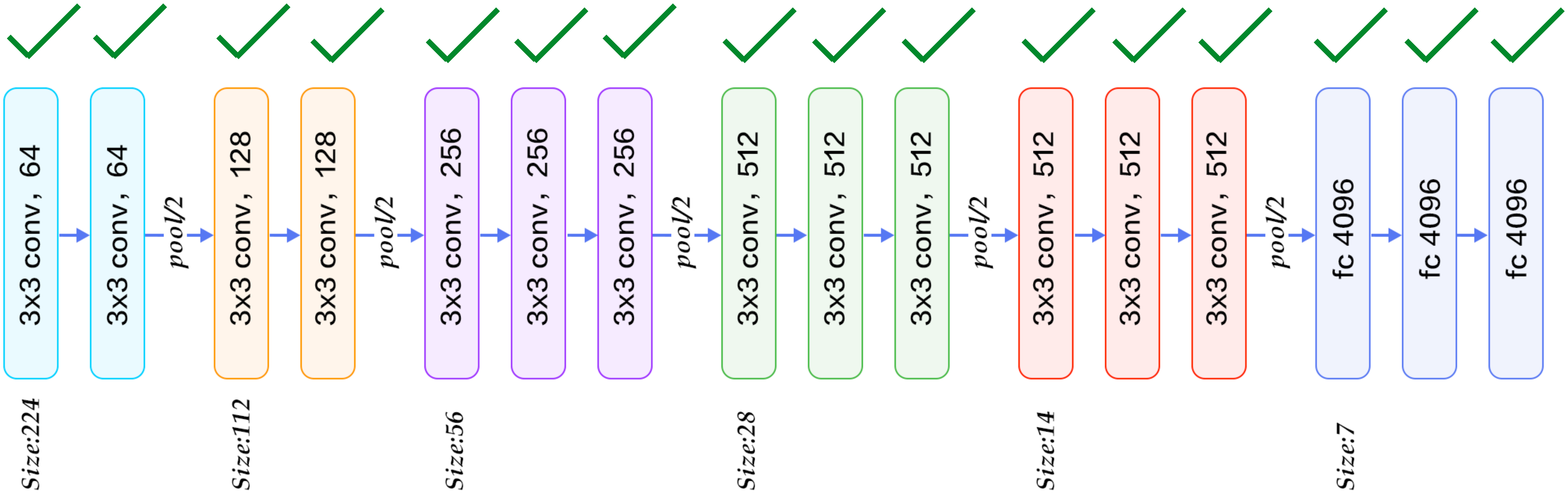
VGG-16 Network

CNNs: Reminder Fully Connected Layers



* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Networks



VGG-16 Network

CNNs: Reminder Fully Connected Layers

CNNs: Reminder Fully Connected Layers

Input

$$\mathbf{W}^T\mathbf{x} + \mathbf{b}, \text{ where } \mathbf{W} \in \mathbb{R}^{25{,}088 \times 4{,}096}$$

Activation

25,088

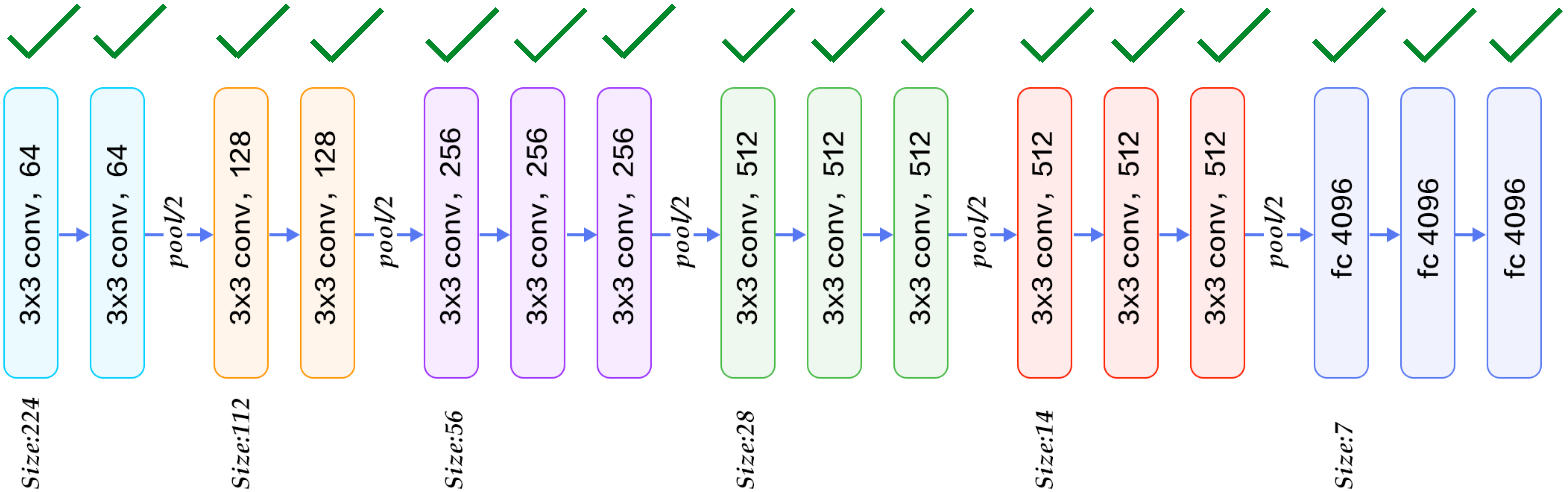
(7 x 7 x 512 image -> stretches to 25,088 x 1)

each neuron looks at the full
input volume

4,096

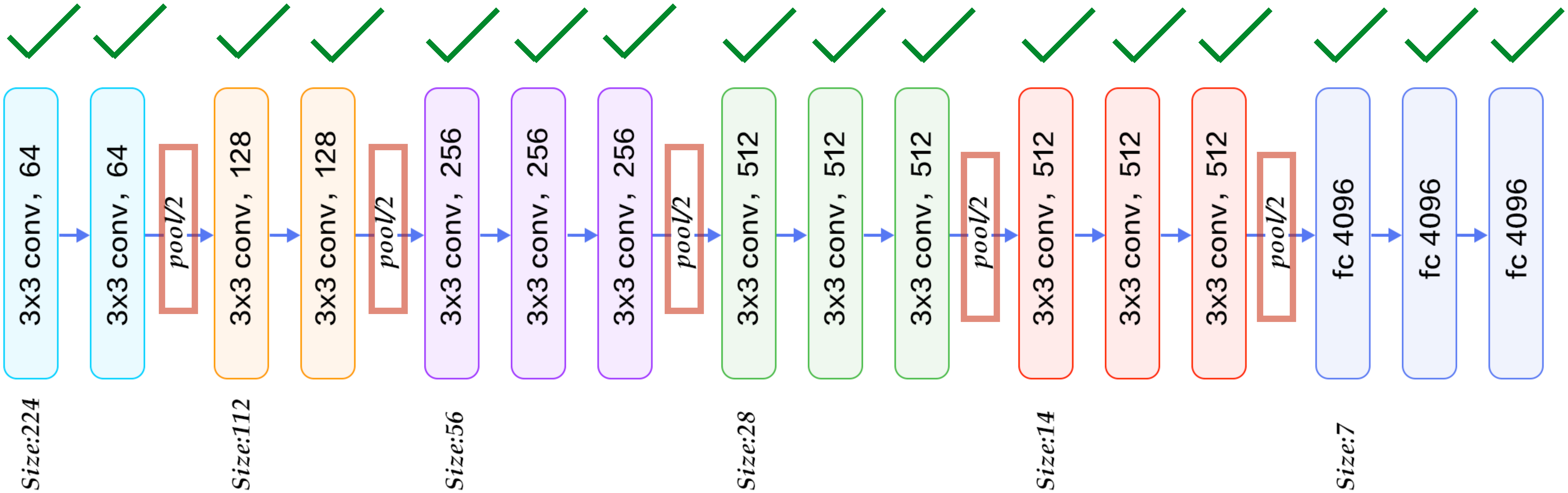
102,760,448 parameters!

Convolutional Neural Networks



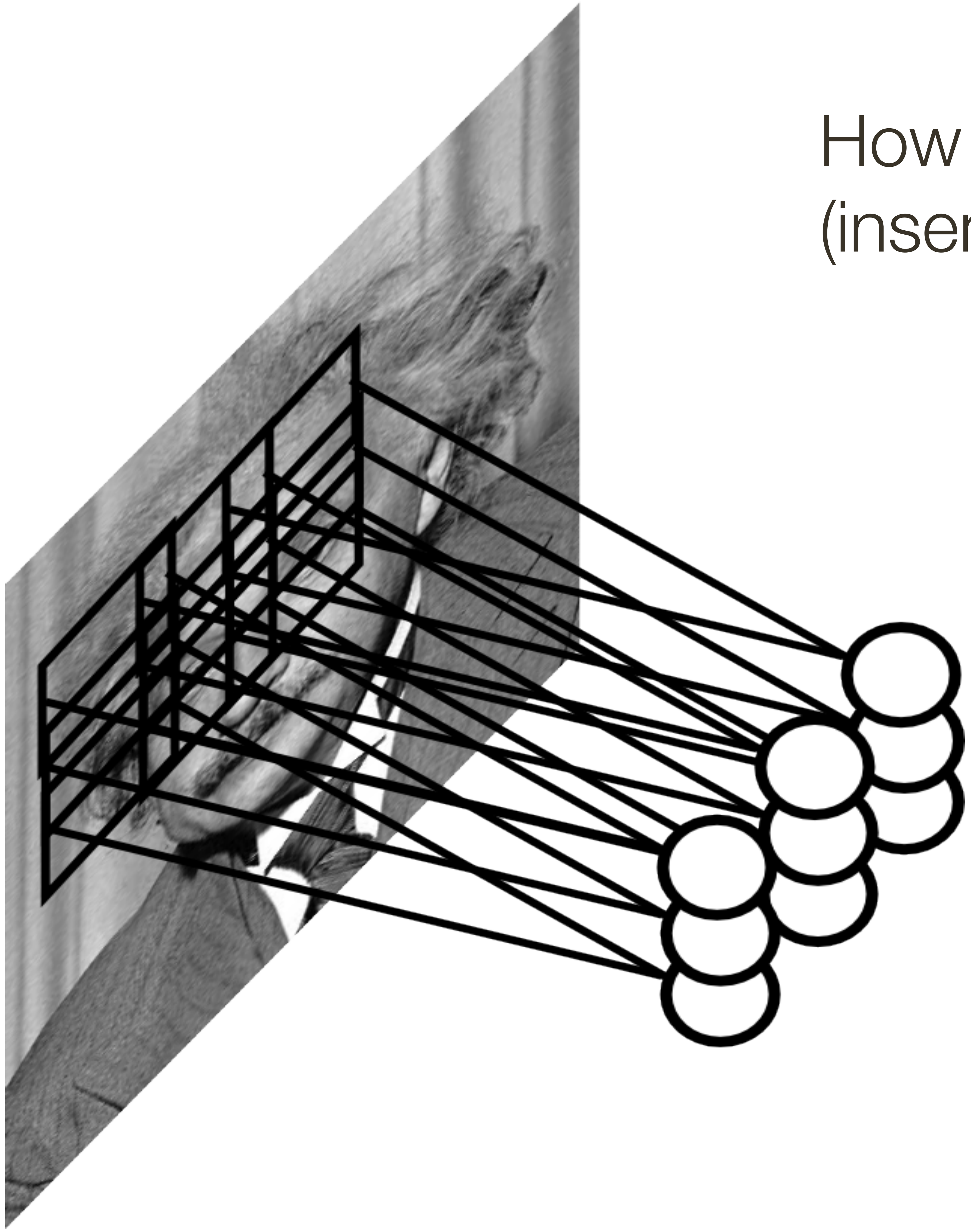
VGG-16 Network

Convolutional Neural Networks



VGG-16 Network

Pooling Layer

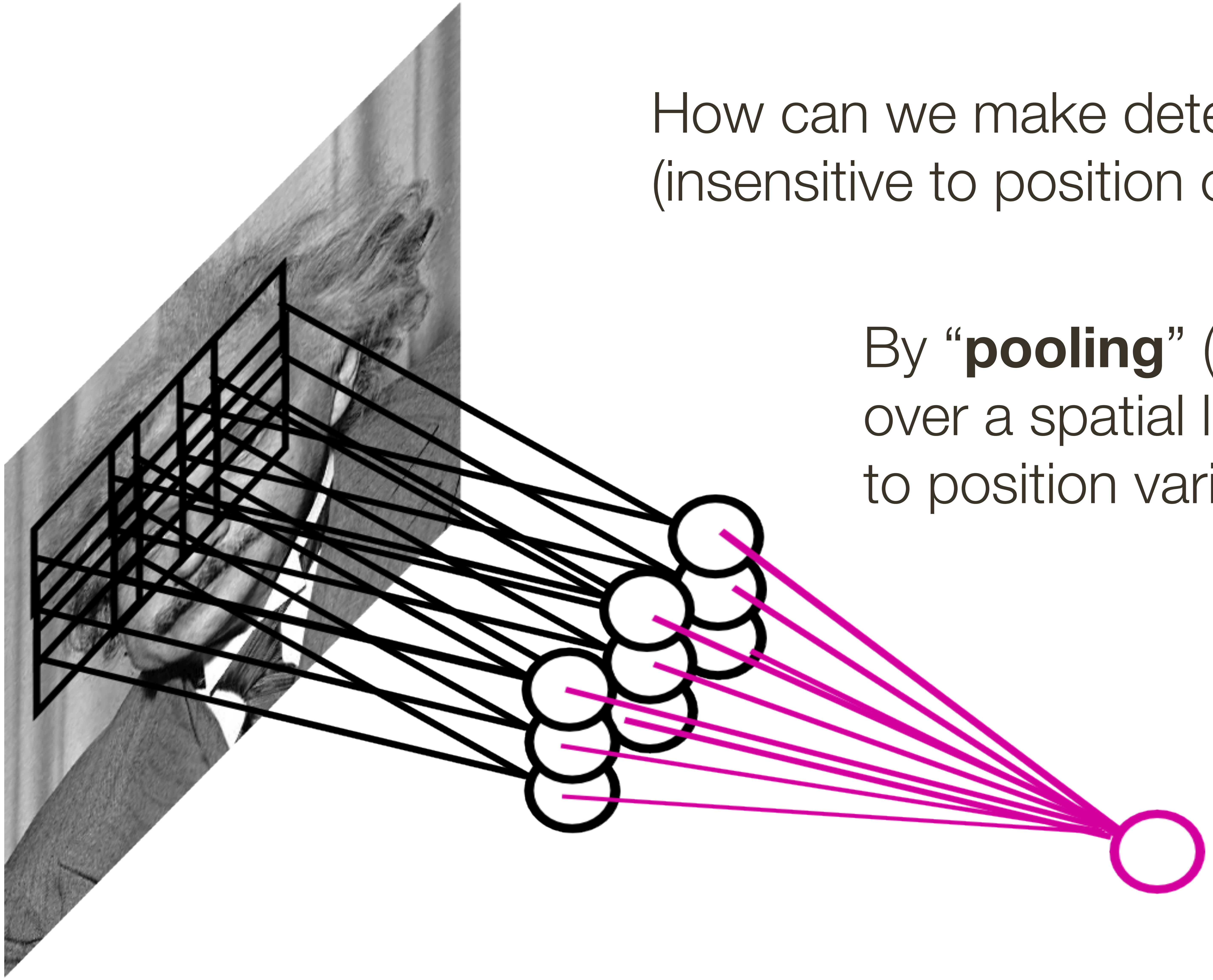


Let us assume the filter is an “eye” detector

How can we make detection spatially invariant (insensitive to position of the eye in the image)

* slide from Marc’Aurelio Renzato

Pooling Layer



Let us assume the filter is an “eye” detector

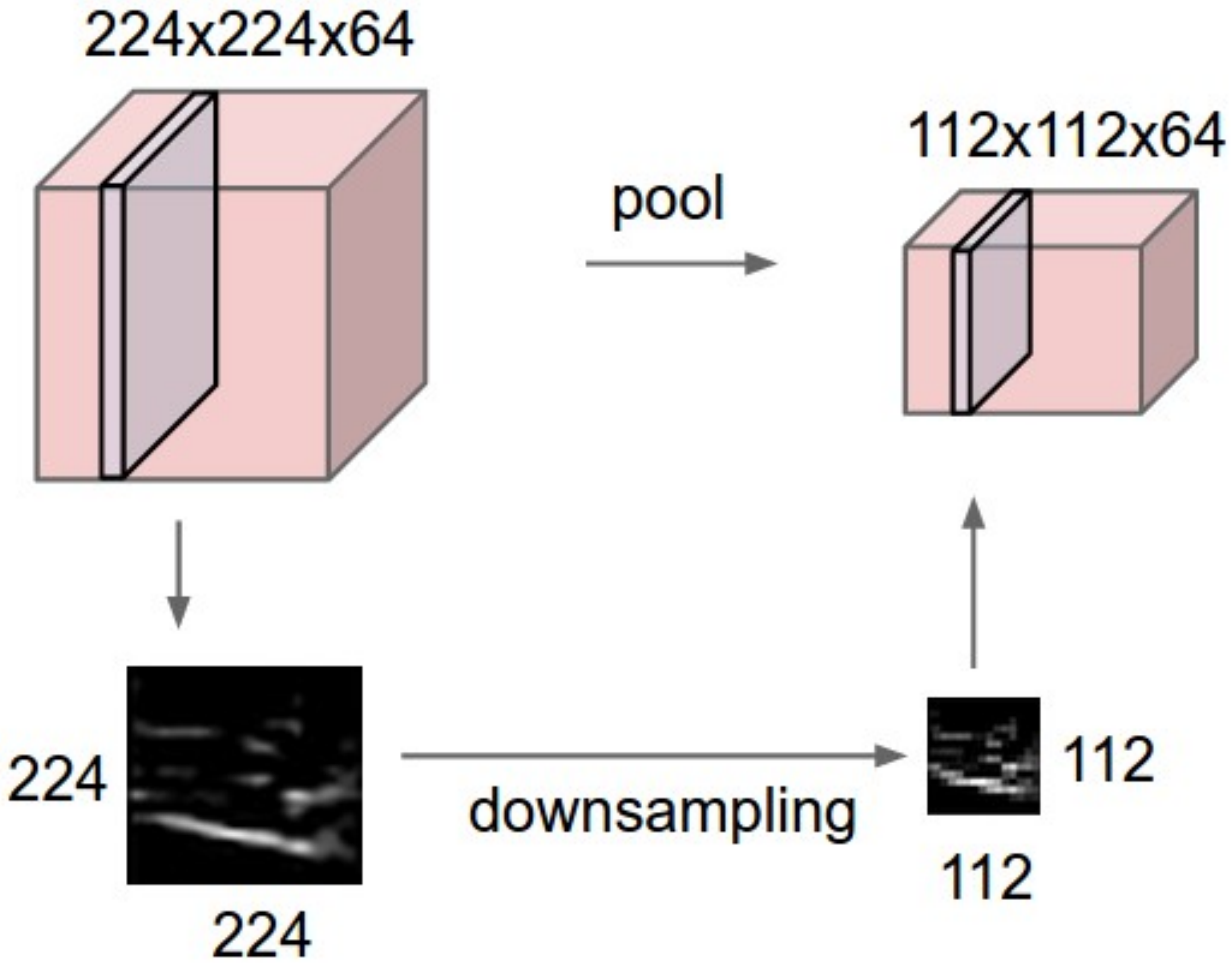
How can we make detection spatially invariant (insensitive to position of the eye in the image)

By “**pooling**” (e.g., taking a max) response over a spatial locations we gain robustness to position variations

* slide from Marc’Aurelio Renzato

Pooling Layer

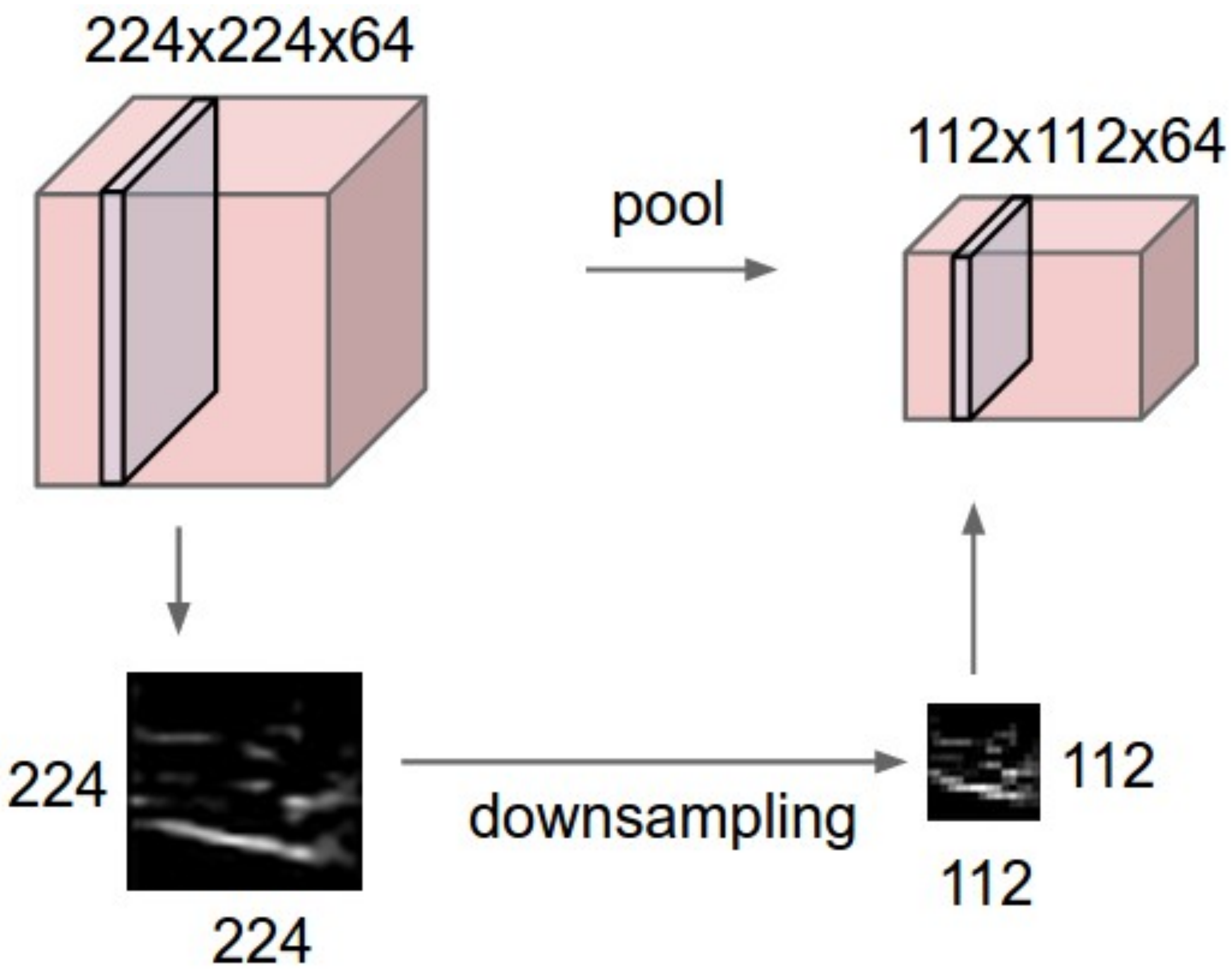
- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

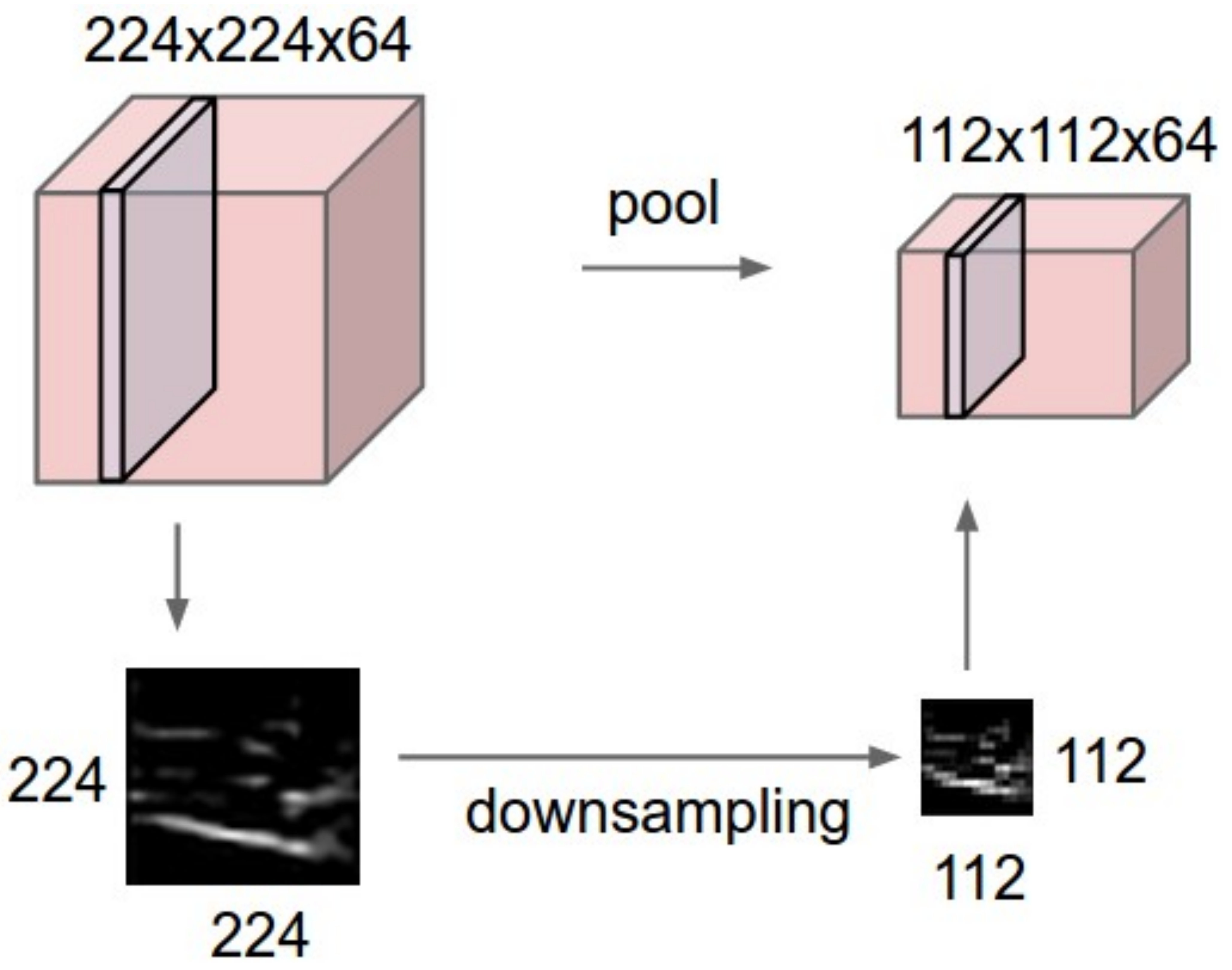


How many **parameters**?

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Pooling Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently

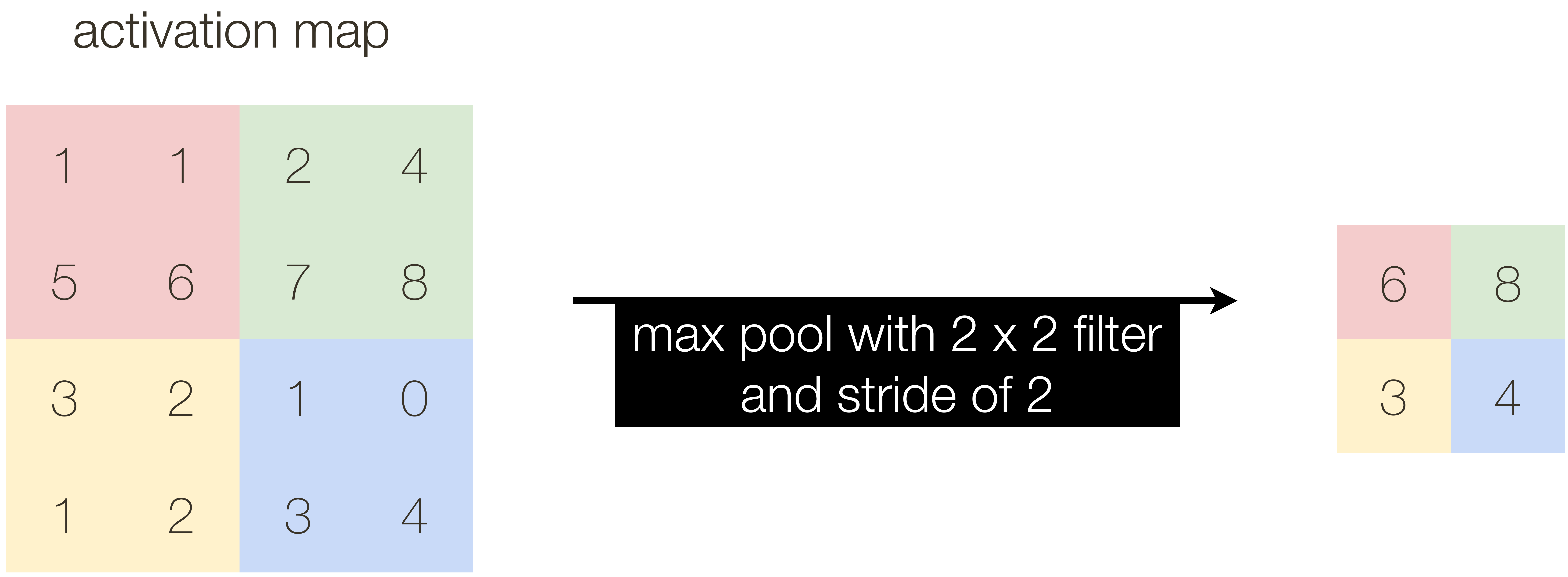


How many **parameters**?

None!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Max **Pooling**


activation map
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2 x 2 filter
and stride of 2
6 8
3 4

Average **Pooling**

activation map

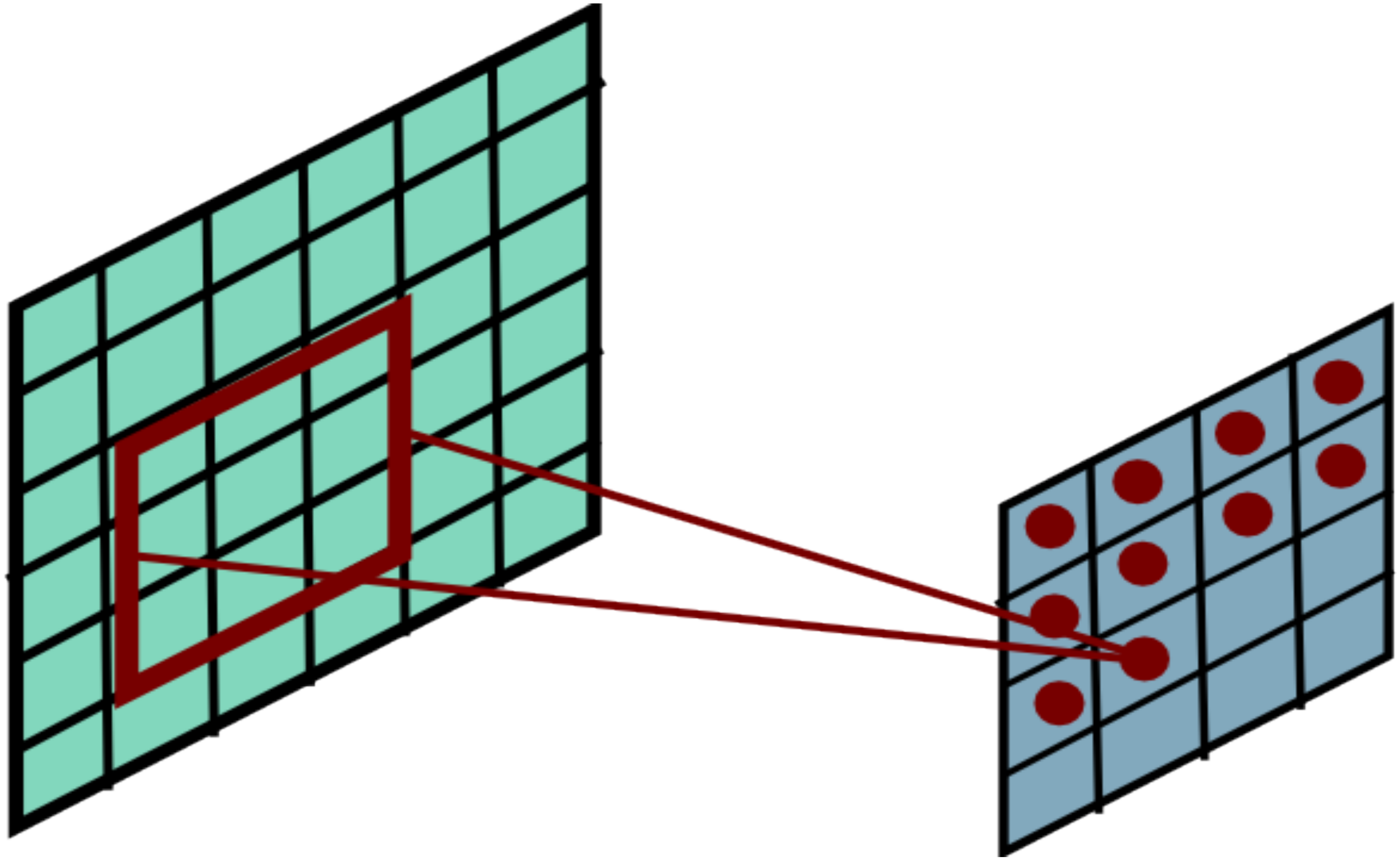
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

avg pool with 2 x 2 filter
and stride of 2

3.25	5.25
2	2

Pooling Layer **Receptive Field**

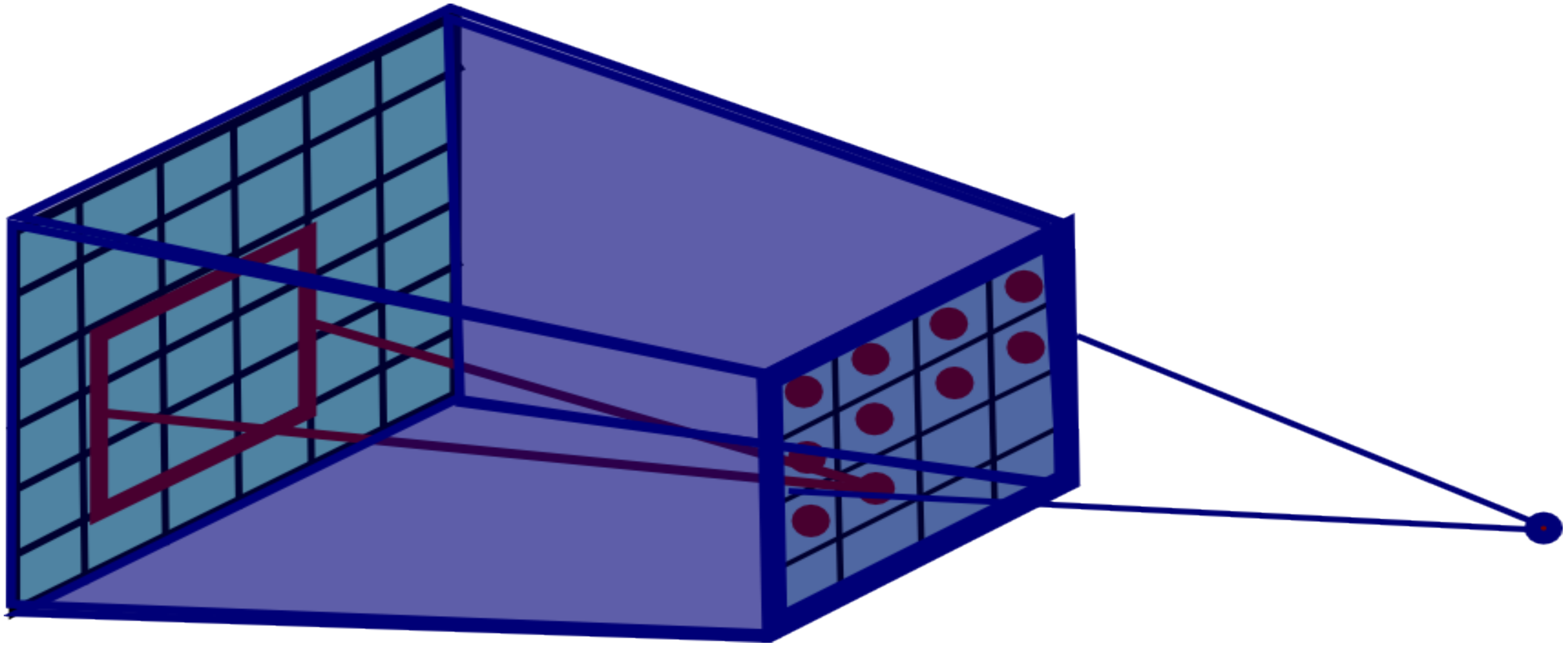
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: **(P+K-1)x(P+K-1)**



* slide from Marc'Aurelio Renzato

Pooling Layer **Receptive Field**

If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: **(P+K-1)x(P+K-1)**



* slide from Marc'Aurelio Renzato

Pooling Layer **Summary**

Accepts a volume of size: $W_i \times H_i \times D_i$

Requires hyperparameters:

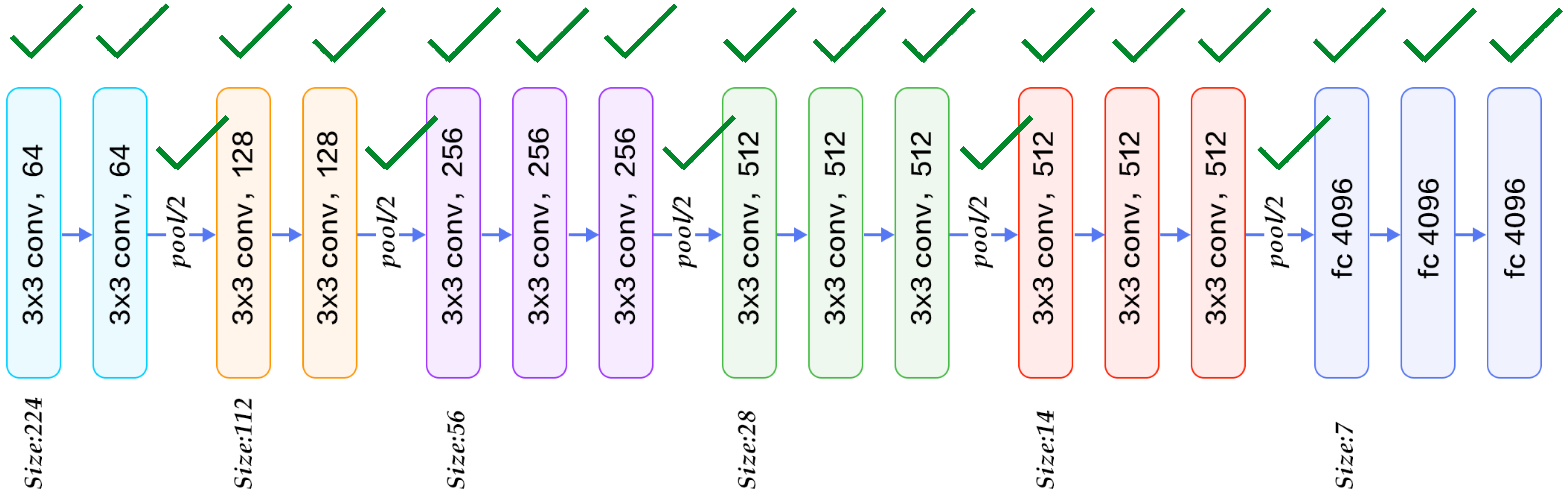
- Spatial extent of filters: K
- Stride of application: F

Produces a volume of size: $W_o \times H_o \times D_o$

$$W_o = (W_i - F)/S + 1 \qquad H_o = (H_i - F)/S + 1 \qquad D_o = D_i$$

Number of total learnable parameters: 0

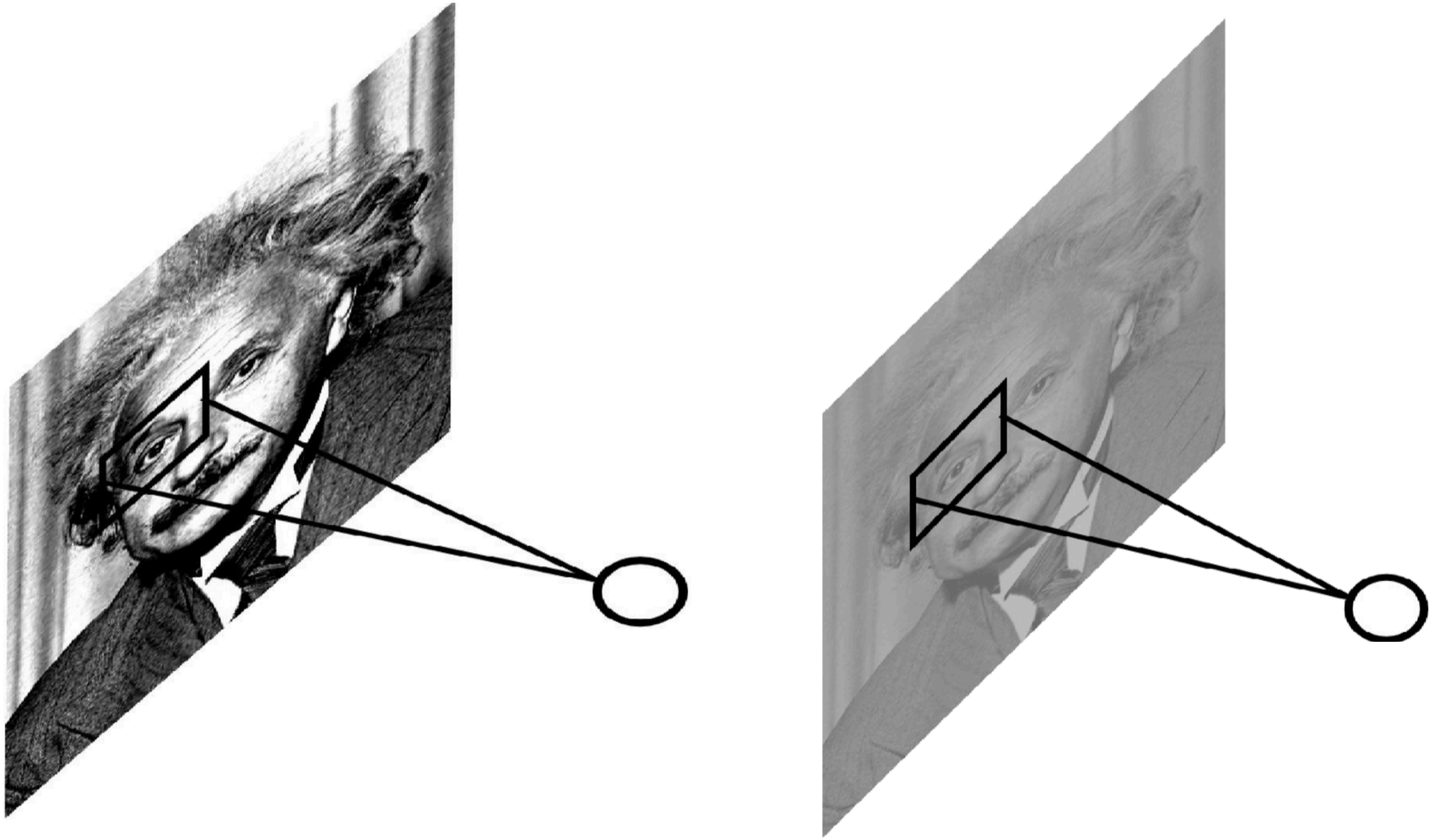
Convolutional Neural Networks



VGG-16 Network

Local **Contrast Normalization** Layer

ensures response is the same in both case (details omitted, no longer popular)



* images from Marc'Aurelio Renzato