THE UNIVERSITY OF BRITISH COLUMBIA

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

Lecture 4: Convolutional Neural Networks (Part 1)

Course **Logistics**

— **Azure credits** have been distributed

— Instructions for using Azure are on Piazza

— **Office hours** (Shikib) on Wednesday (January 17th) 1-2pm

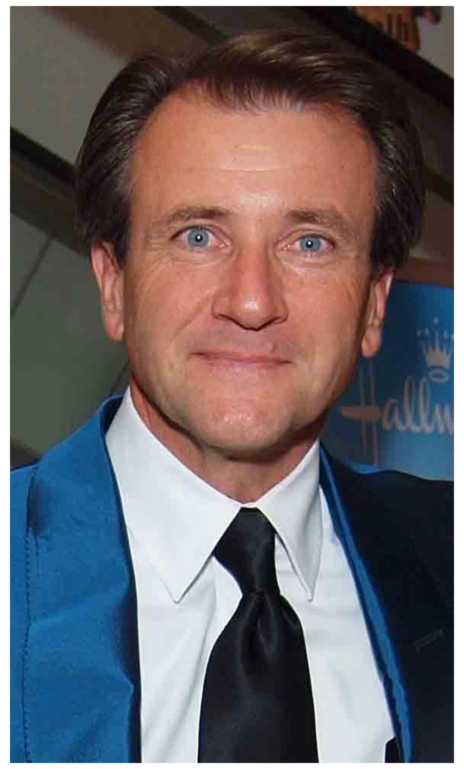
— Assignment 1 grading

— Assignment 2 will be out today (on CNNs)

— Start thinking about **papers**

— Start thinking about **project** and forming groups (proposal is in ~month)

Research **Projects**

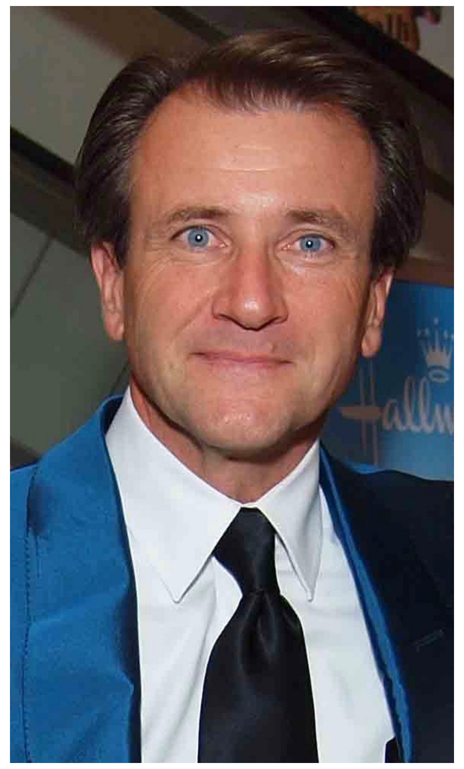


Robert Herjavec
(Canadian businessman and investor)
Shark Tank

Research **Projects**

Business

“A good **entrepreneur** can take a mediocre **idea** and make it great, a bad entrepreneur will take the best idea and run it into a ground.”

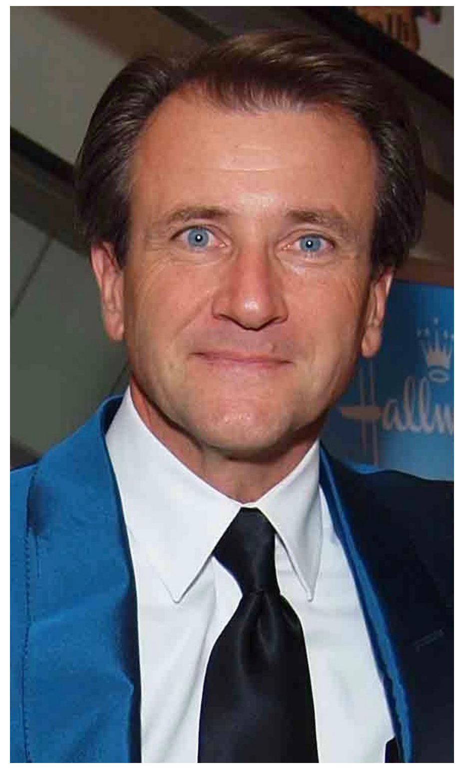


Robert Herjavec
(Canadian businessman and investor)
Shark Tank

Research **Projects**

Business

“A good **entrepreneur** can take a mediocre **idea** and make it great, a bad entrepreneur will take the best idea and run it into a ground.”



Robert Herjavec
(Canadian businessman and investor)
Shark Tank

Research

“A good **idea** can make a mediocre **project** great, a bad idea will take the best project and run it into a ground.” — Me

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

Marvin Minsky

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

- This is excellent if **X** is something no one has done or thought about and is important (guaranteed success)
- Requires **forward thinking**, knowledge of the field
- A sure way to get tenure
- Difficult to do as the field matures

Rosland Picard, MIT — Affective Computing

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

— More incremental; a lot of science is incremental (“standing on the shoulders of giants”)

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

— More incremental; a lot of science is incremental ("standing on the shoulders of giants")

— **Retrospective:** compare existing approaches see why they work what is missing (guaranteed success)

— **Perspective:** come up with an idea or the insight that you truly believe and test it

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

- More incremental; a lot of science is incremental ("standing on the shoulders of giants")
- **Retrospective:** compare existing approaches see why they work what is missing (guaranteed success)
- **Perspective:** come up with an idea or the insight that you truly believe and test it
- Requires through knowledge of the sub-field (lots of reading)

Research **Projects** Ideas Levels of Abstraction

Geoffrey Hinton

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

— More incremental; a lot of science is incremental (“standing on the shoulders of giants”)

— **Retrospective:** compare existing approaches see why they work what is missing (guaranteed success)

— **Perspective:** come up with an idea or the insight that you truly believe and test it

— Requires through knowledge of the sub-field (lots of reading)

— Requires strong intuition and high level (intuitive) thinking

Research **Projects** Ideas Levels of Abstraction

Geoffrey Hinton

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

- More incremental; a lot of science is incremental ("standing on the shoulders of giants")
- **Retrospective:** compare existing approaches see why they work what is missing (guaranteed success)
- **Perspective:** come up with an idea or the insight that you truly believe and test it
- Requires through knowledge of the sub-field (lots of reading)
- Requires strong intuition and high level (intuitive) thinking
- Requires understanding of the mathematical tools and formulations to know what maybe possible

Research **Projects** Ideas Levels of Abstraction

Geoffrey Hinton

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

— More incremental; a lot of science is incremental (“standing on the shoulders of giants”)

— **Retrospective:** compare existing approaches see why they work what is missing (guaranteed success)

— **Perspective:** come up with an idea or the insight that you truly believe and test it

— Requires through knowledge of the sub-field (lots of reading)

— Requires strong intuition and high level (intuitive) thinking

— Requires understanding of the mathematical tools and formulations to know what maybe possible

— Helps to bringing knowledge from other fields (field cross pollination)

Research **Projects** Ideas Levels of Abstraction

I want to **solve vision** / language / etc.

I want to do X (e.g., **image captioning**)

I think the **right way to solve** (or improve) X is Y

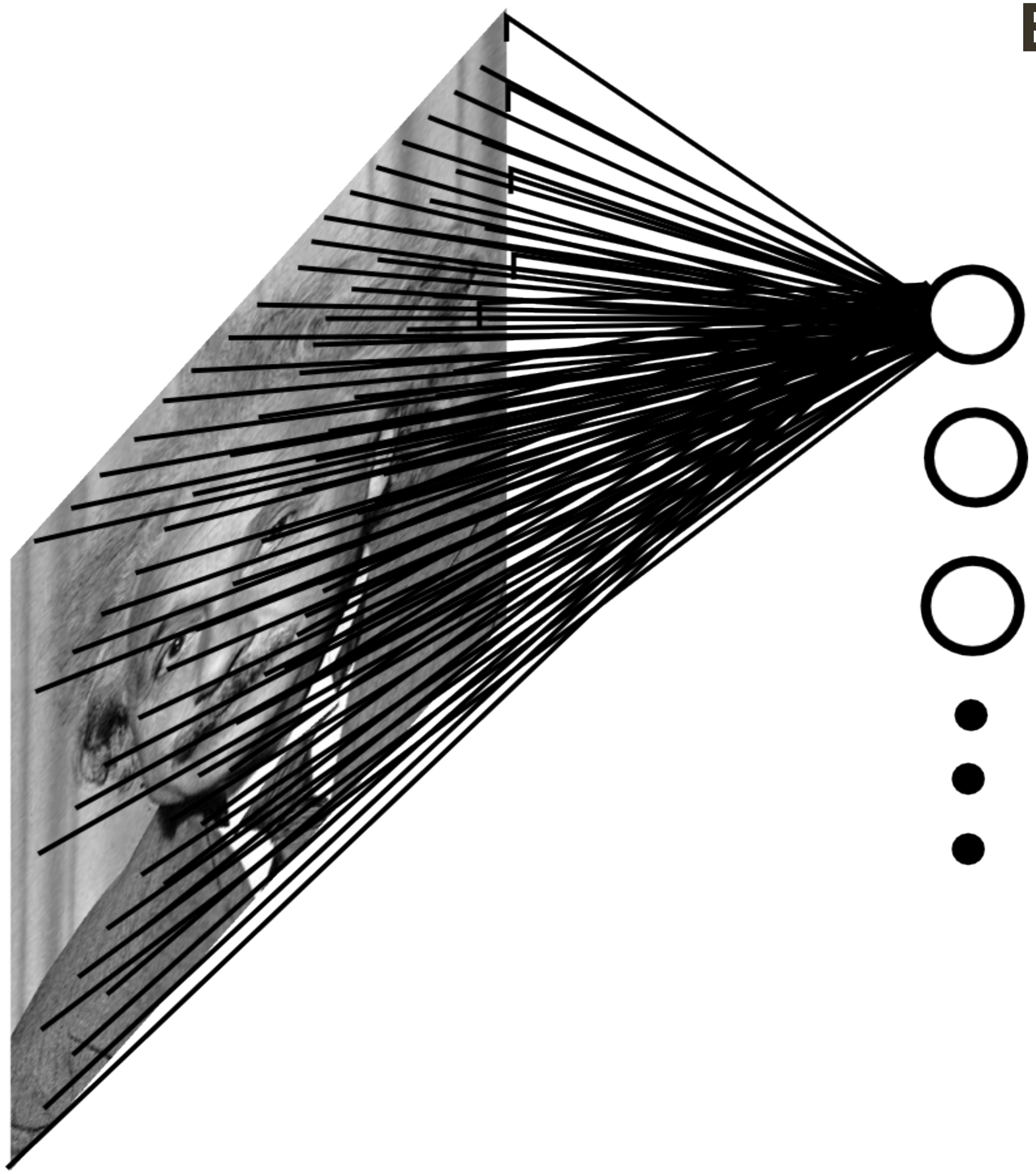
Mathematical **formulation**

Implementation / engineering

Experimental testing

On to **todays** lecture …

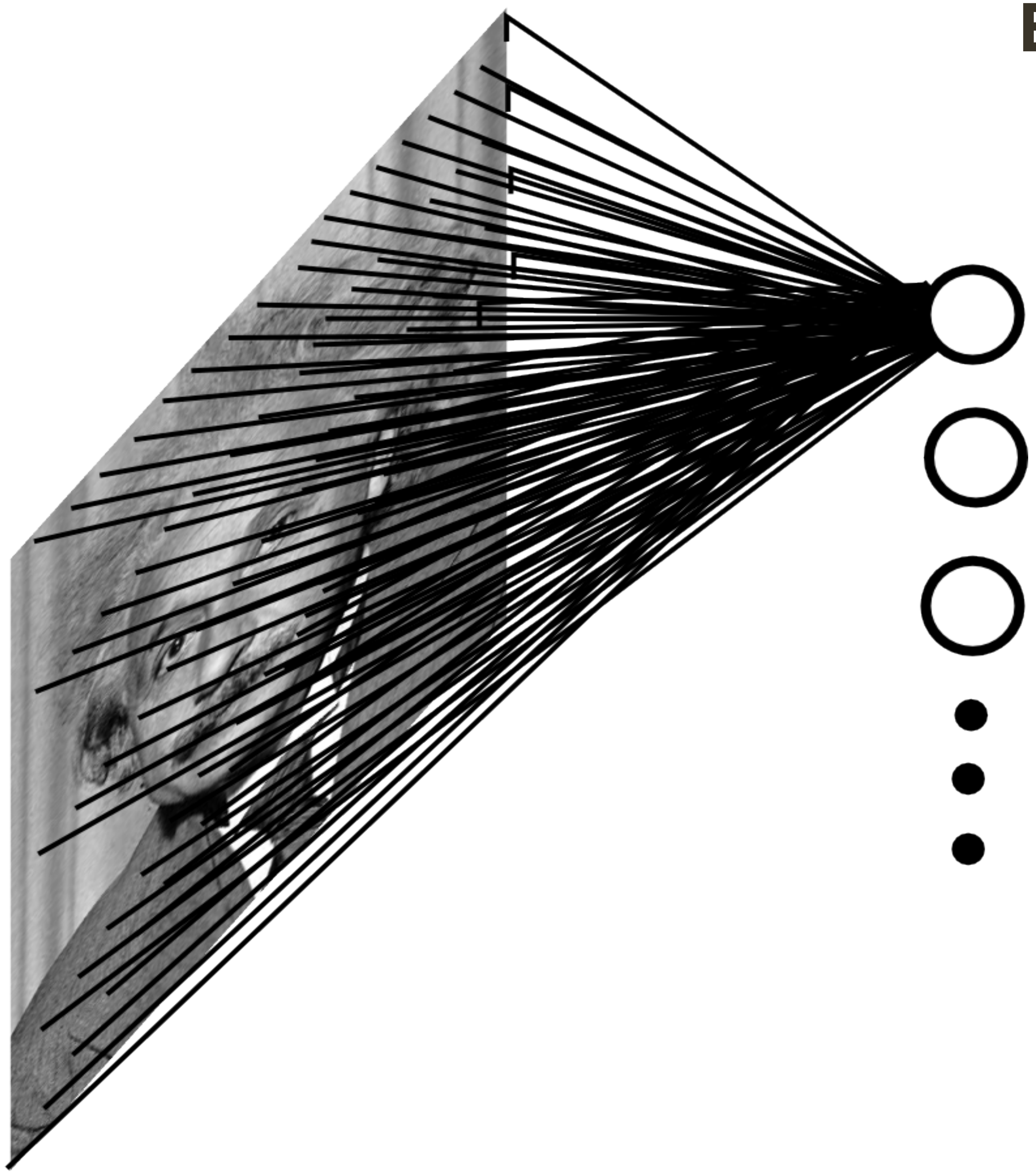
Fully Connected Layer



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

* slide from Marc'Aurelio Renzato

Fully Connected Layer

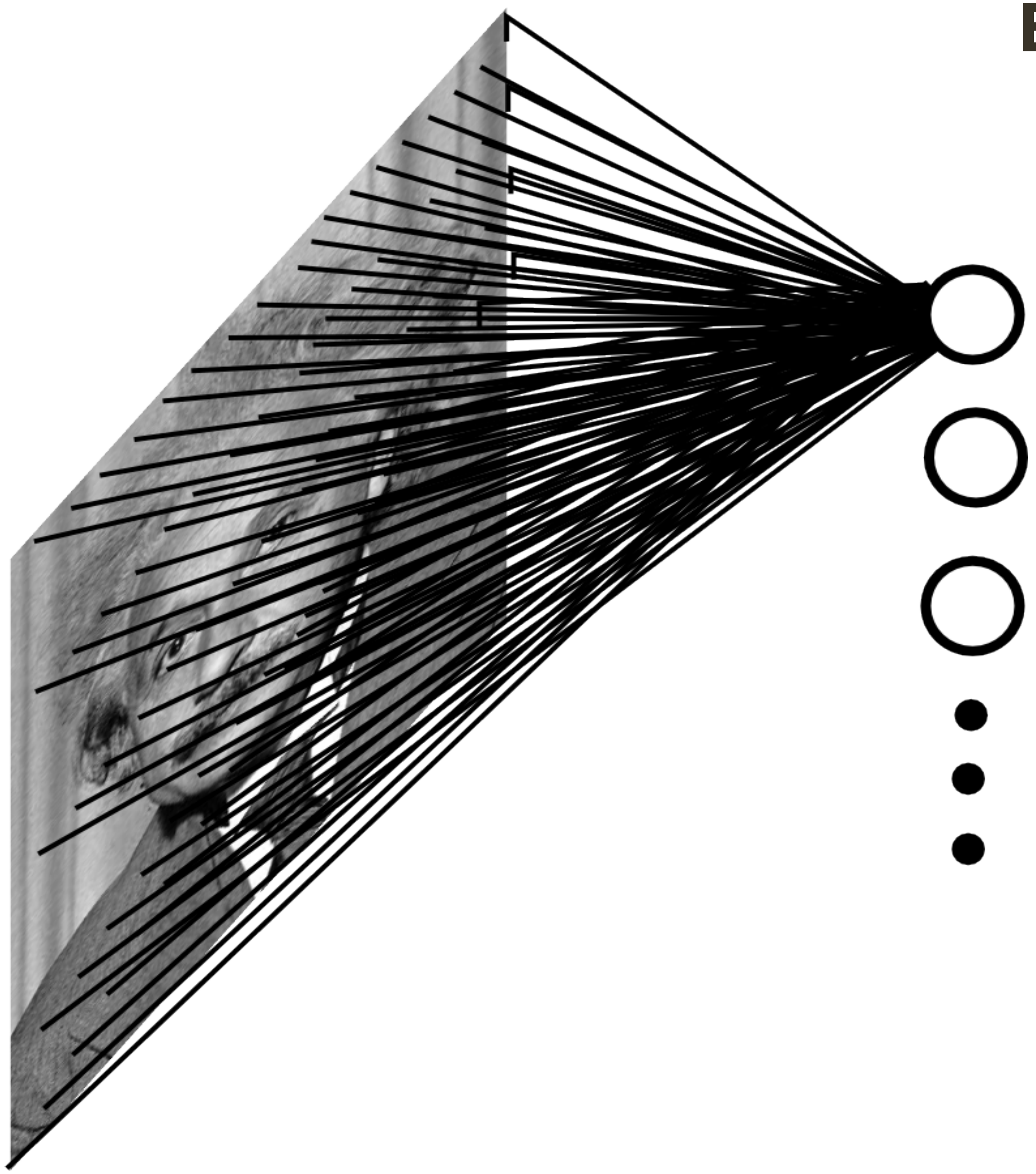


Example: 200 x 200 image (small)
x 40K hidden units

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

* slide from Marc'Aurelio Renzato

Fully Connected Layer



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

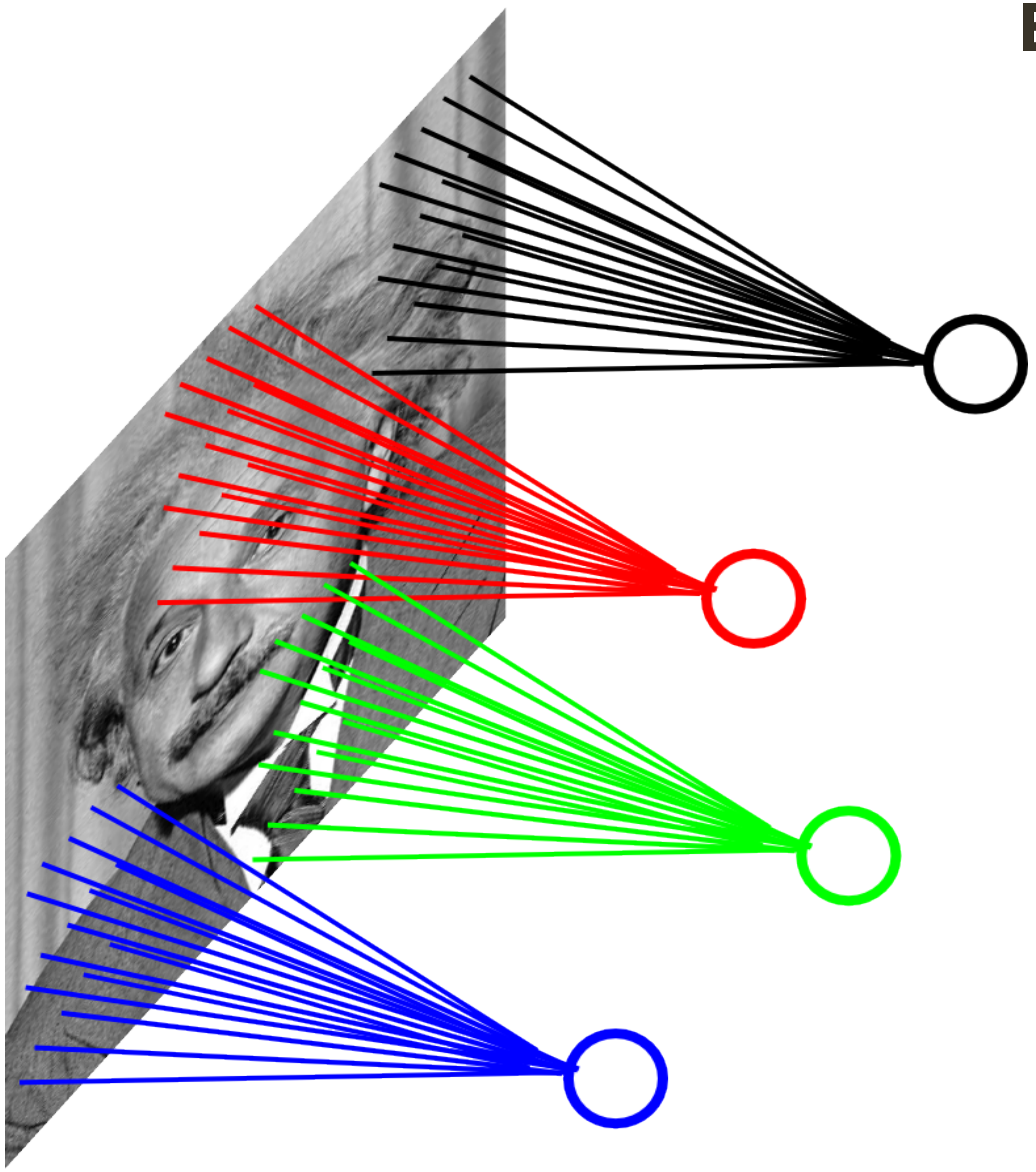
= ~ **2 Billion** parameters (for one layer!)

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



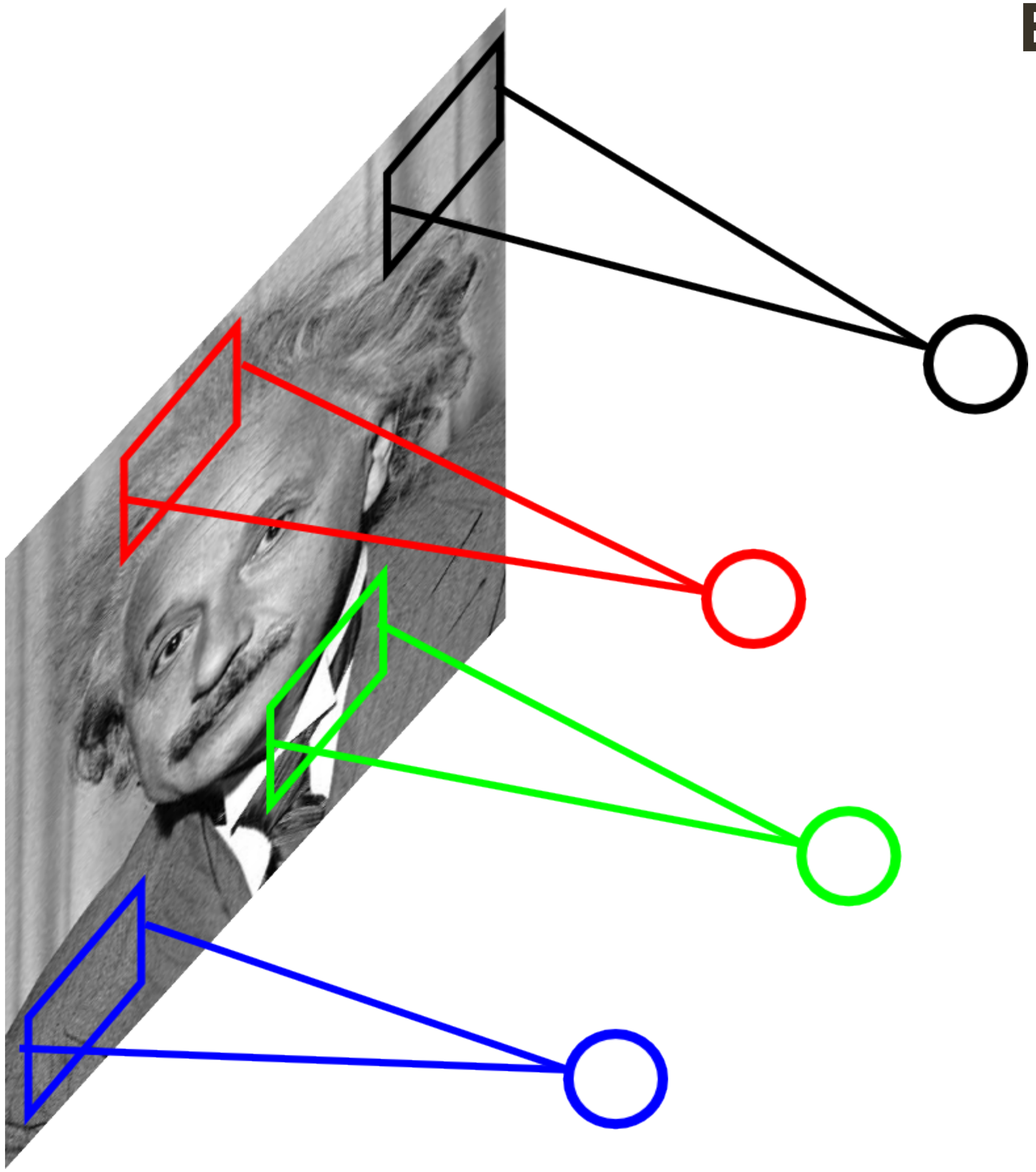
Example: 200 x 200 image (small)
x 40K hidden units

Filter size: 10 x 10

= ~ **4 Million** parameters

* slide from Marc'Aurelio Renzato

Locally Connected Layer



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

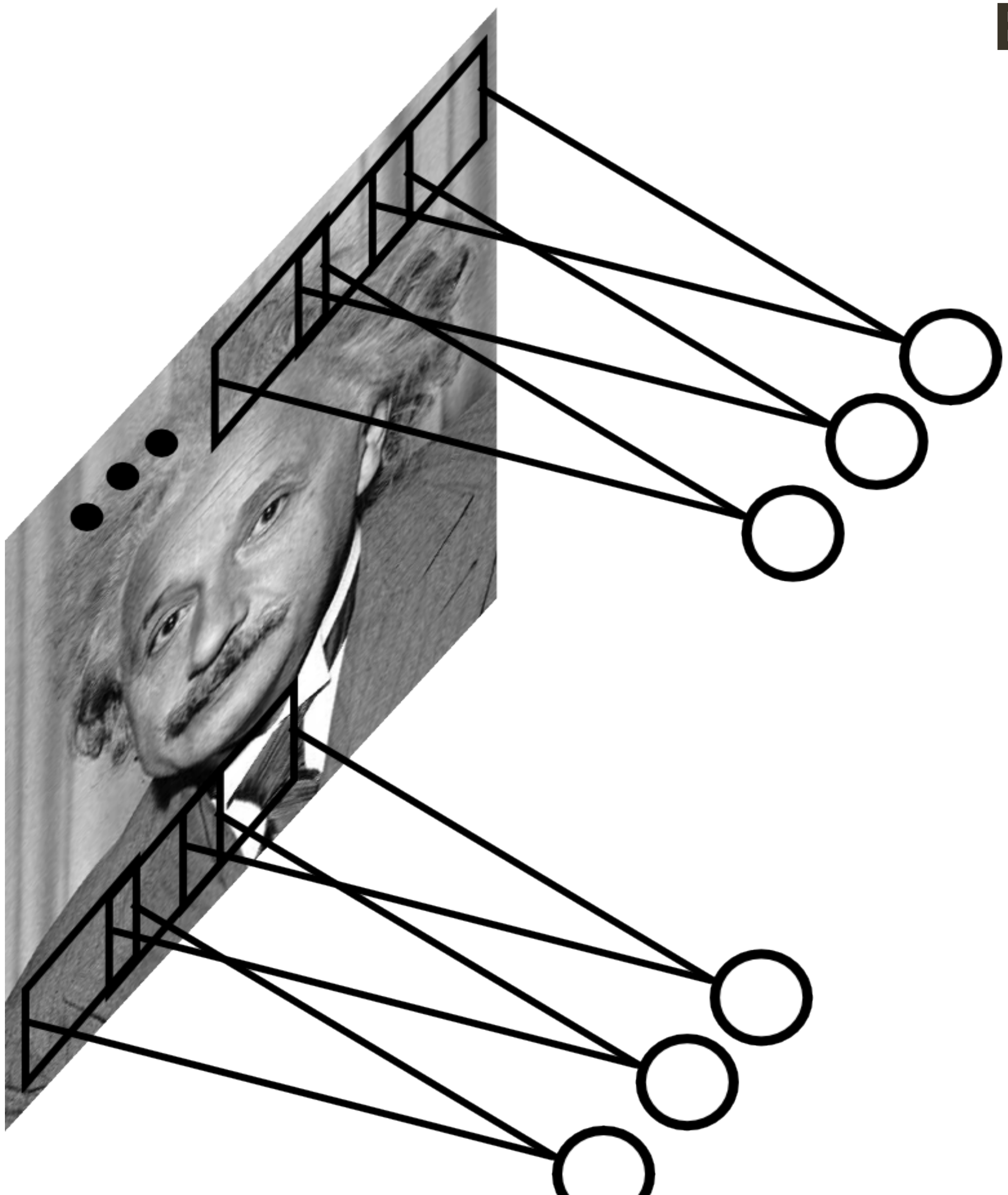
Filter size: 10 x 10

= ~ **4 Million** parameters

Stationarity — statistics is similar at different locations

* slide from Marc'Aurelio Renzato

Convolutional Layer



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

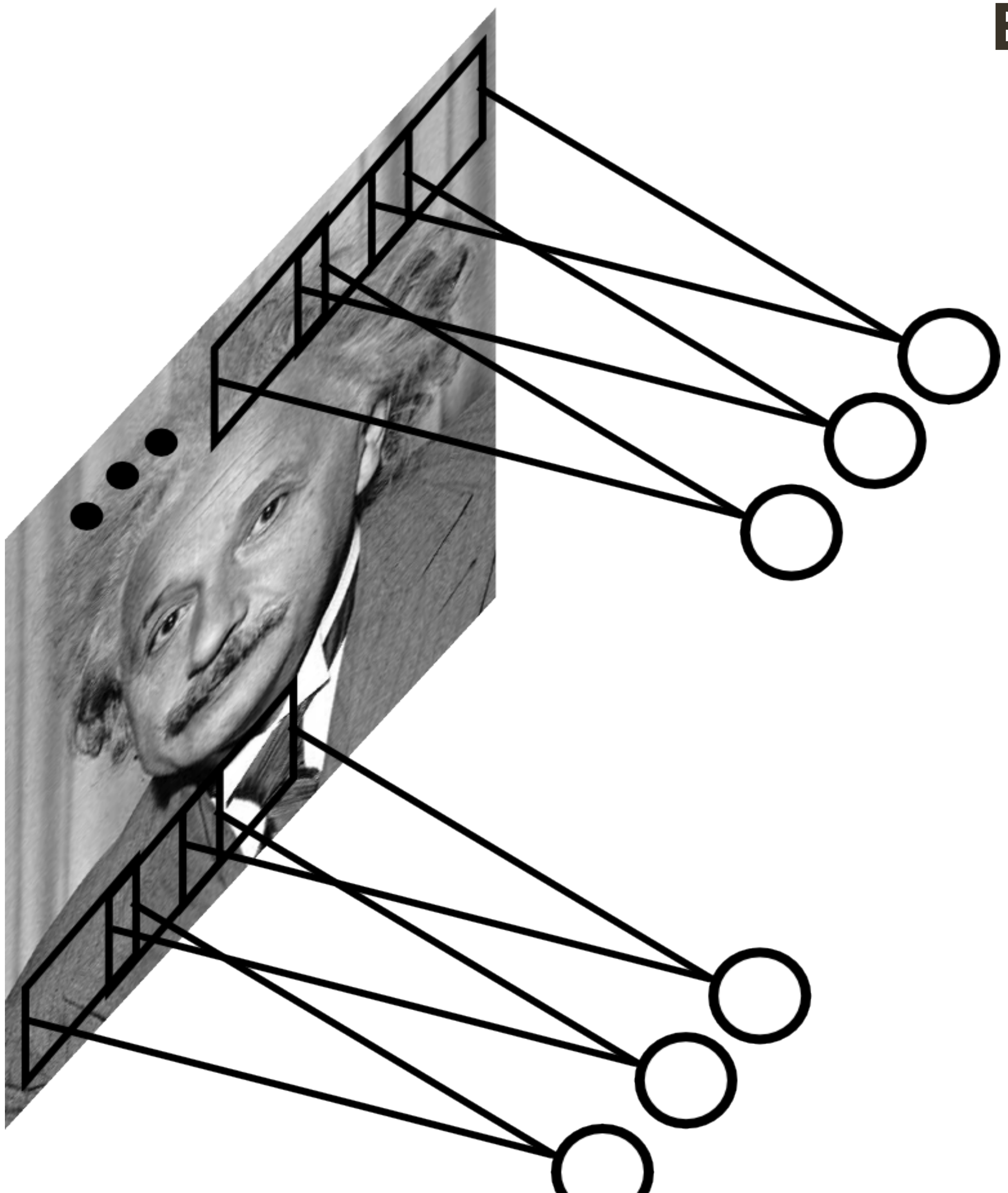
Filter size: 10 x 10

= ~ **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)
x 40K hidden units

Filter size: 10 x 10

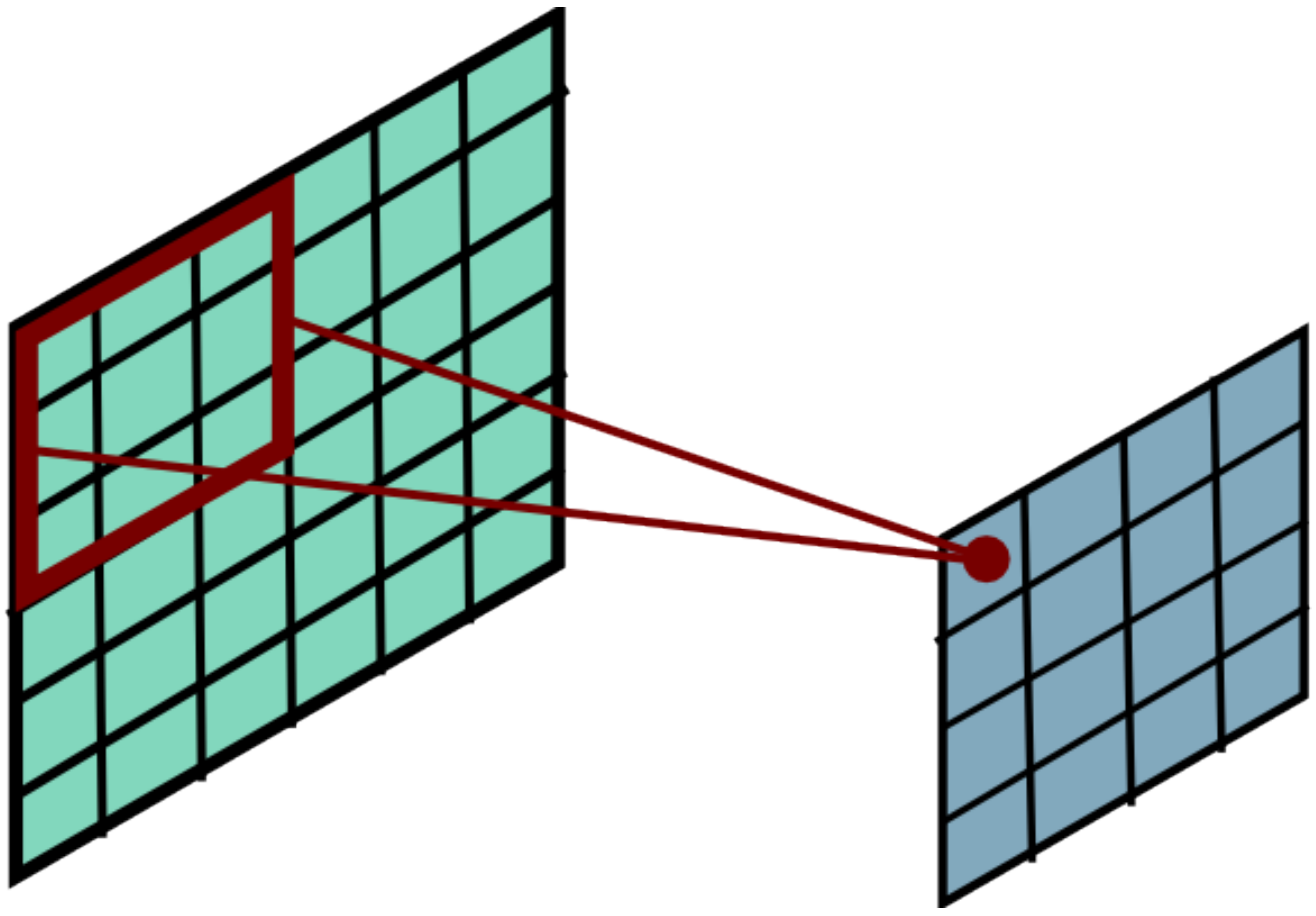
= ~ **~~4 Million~~** parameters

= 100 parameters

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

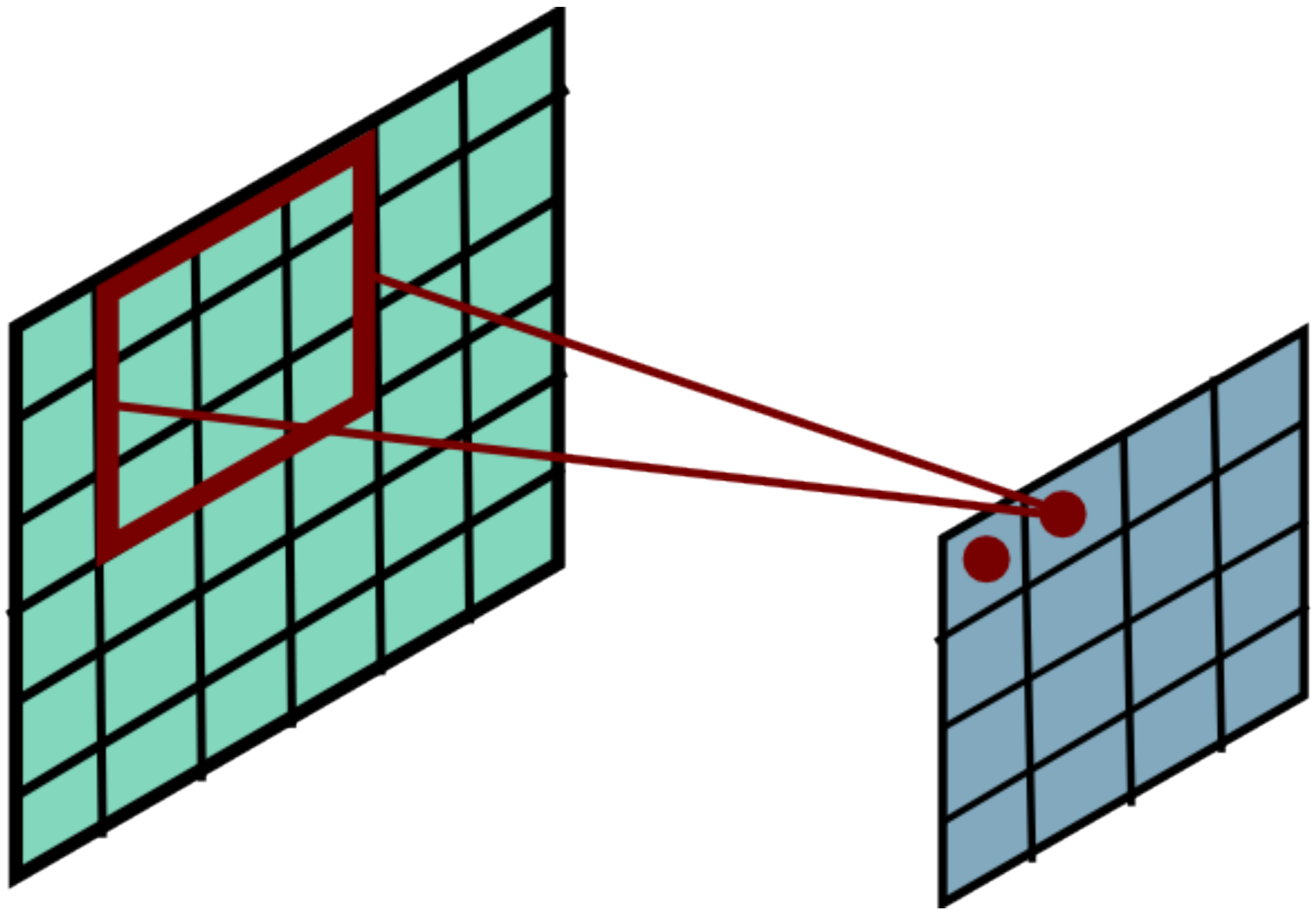
* slide adopted from Marc'Aurelio Renzato

Convolutional Layer



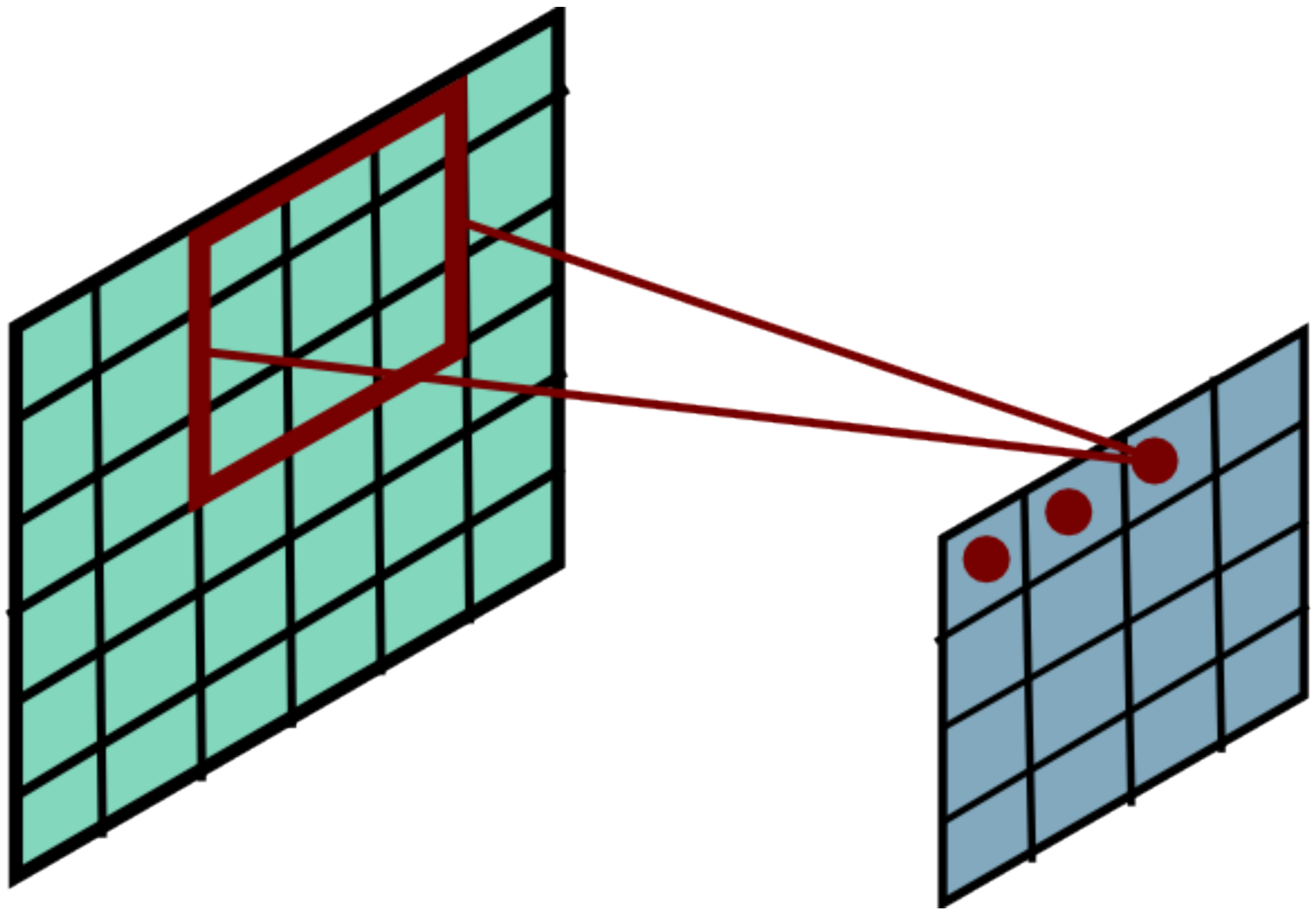
* slide from Marc'Aurelio Renzato

Convolutional Layer



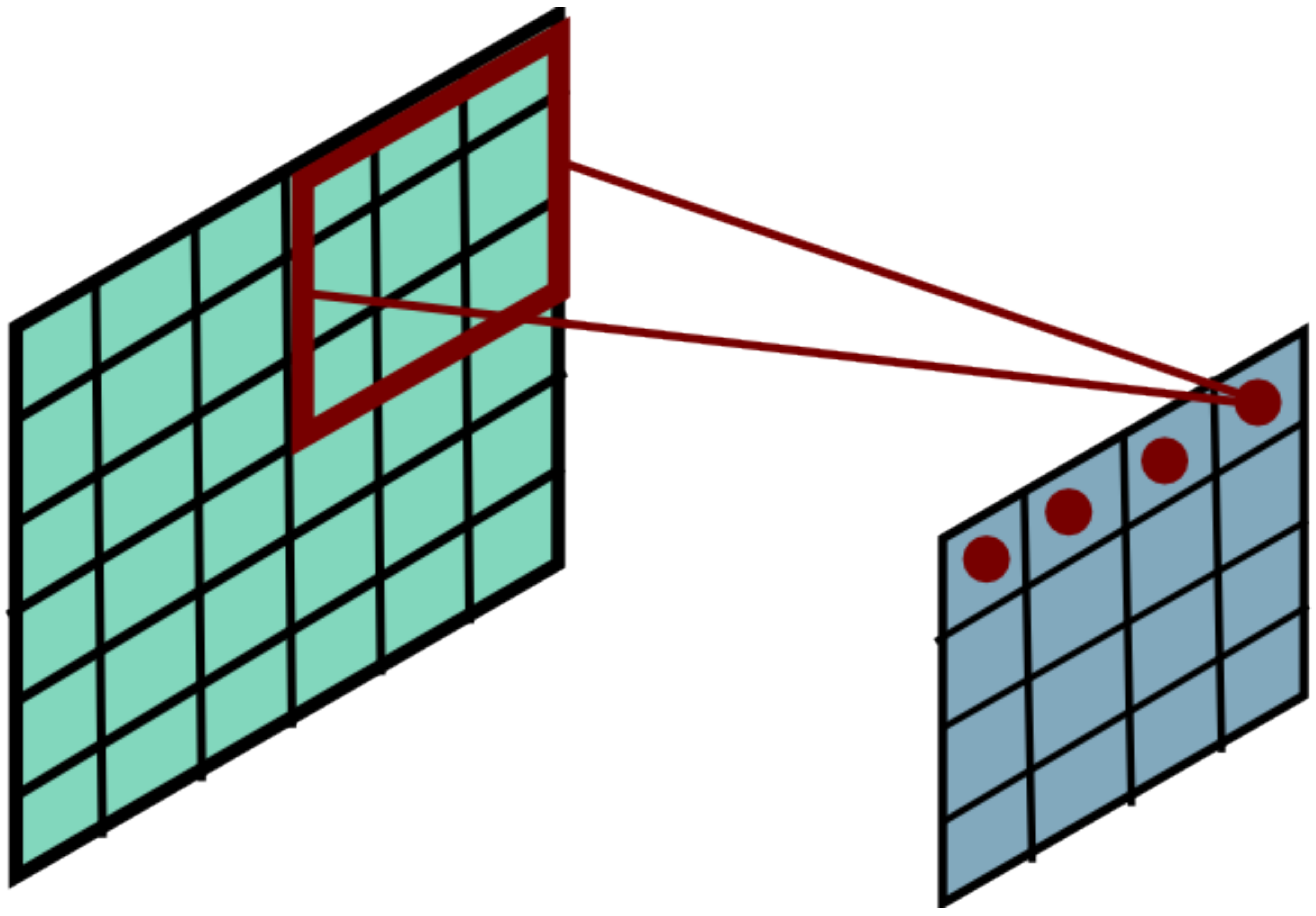
* slide from Marc'Aurelio Renzato

Convolutional Layer



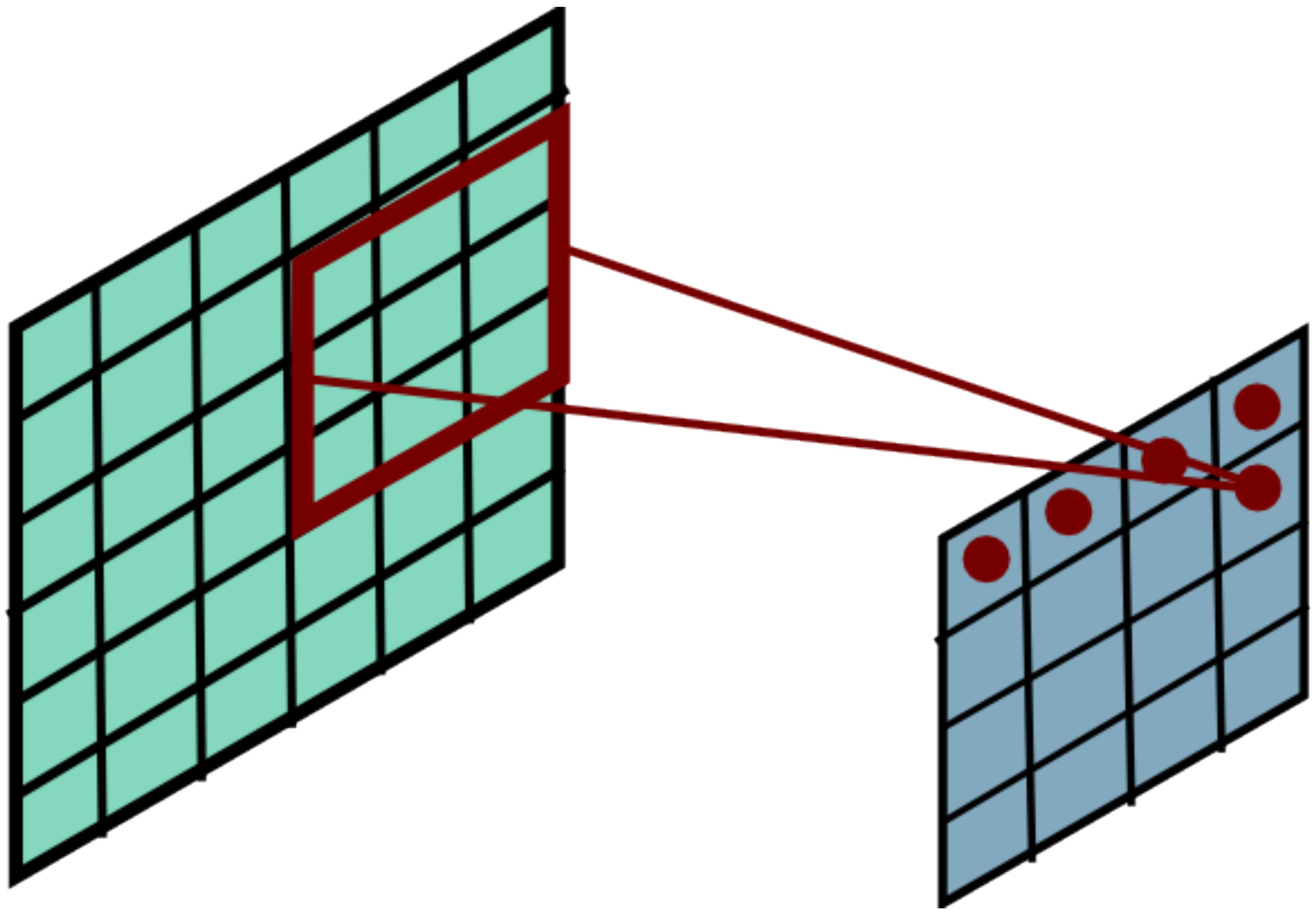
* slide from Marc'Aurelio Renzato

Convolutional Layer



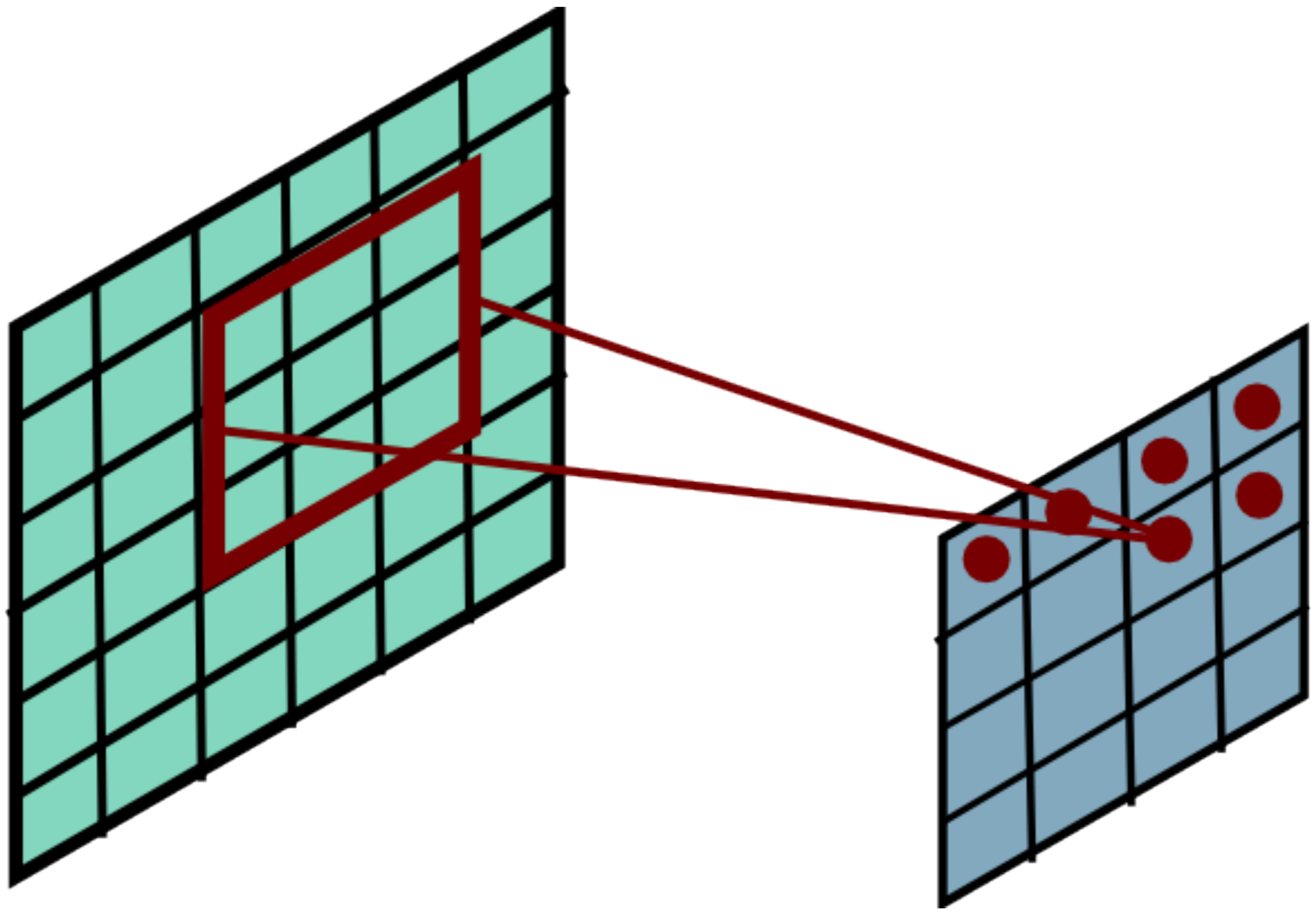
* slide from Marc'Aurelio Renzato

Convolutional Layer



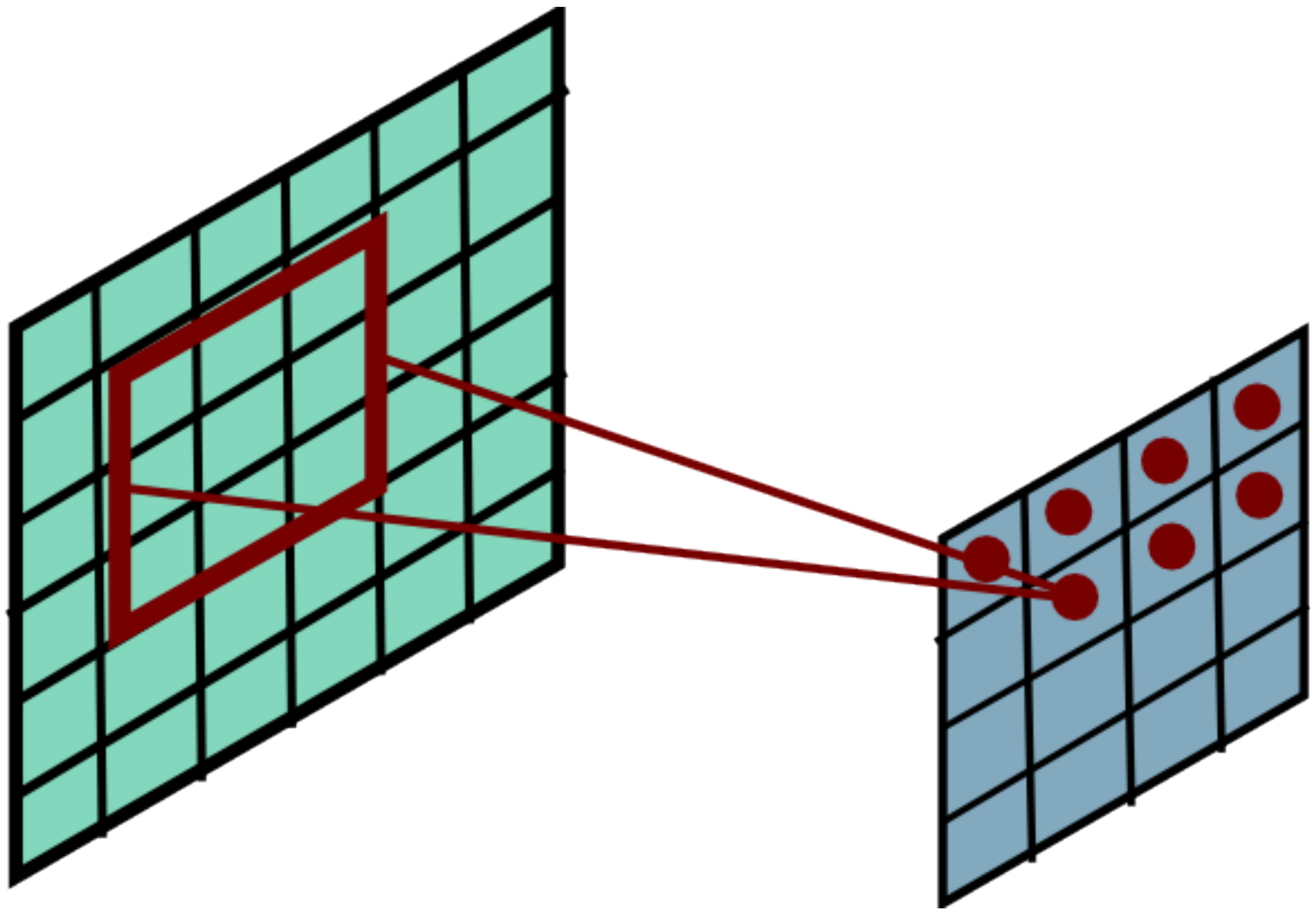
* slide from Marc'Aurelio Renzato

Convolutional Layer



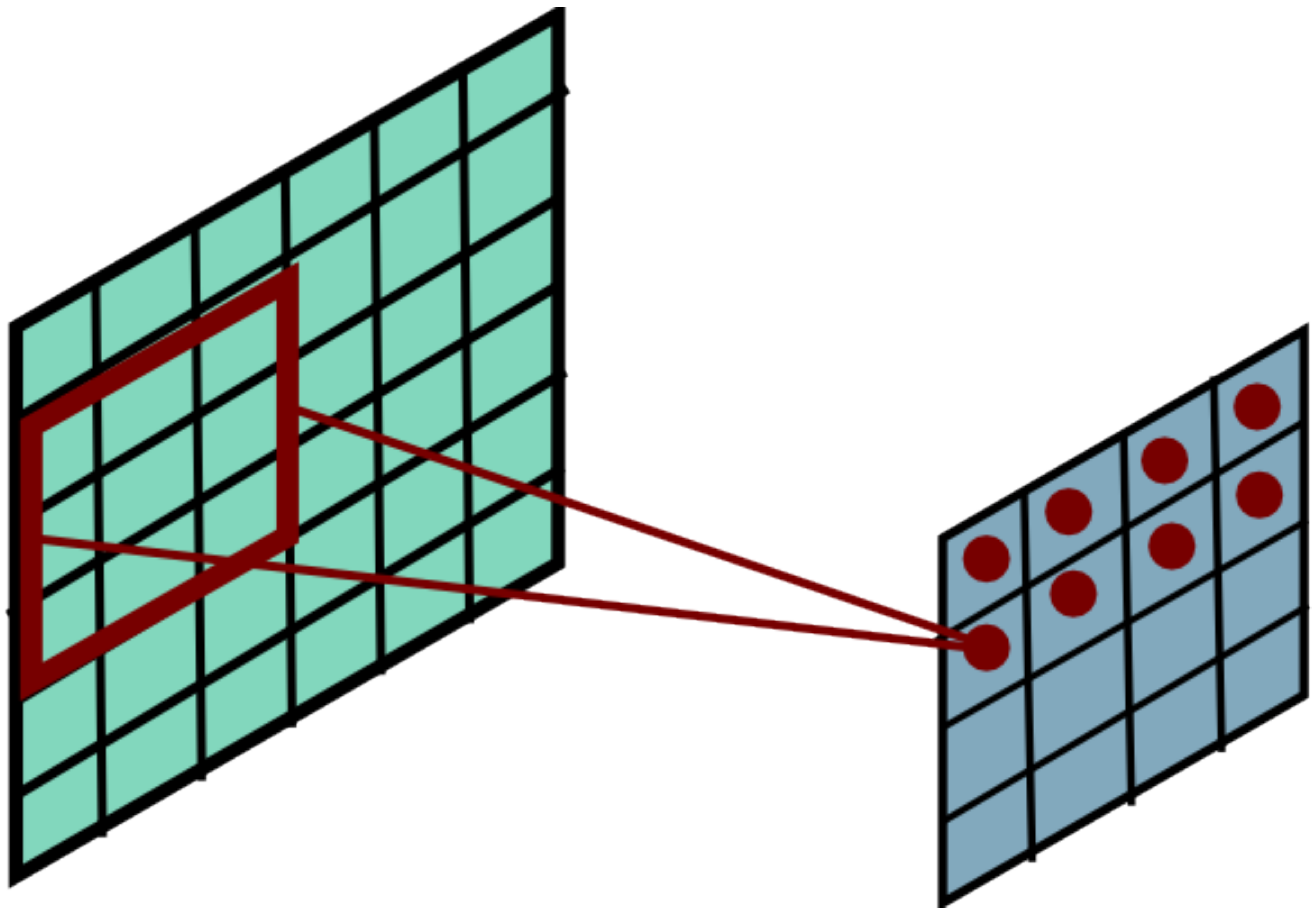
* slide from Marc'Aurelio Renzato

Convolutional Layer



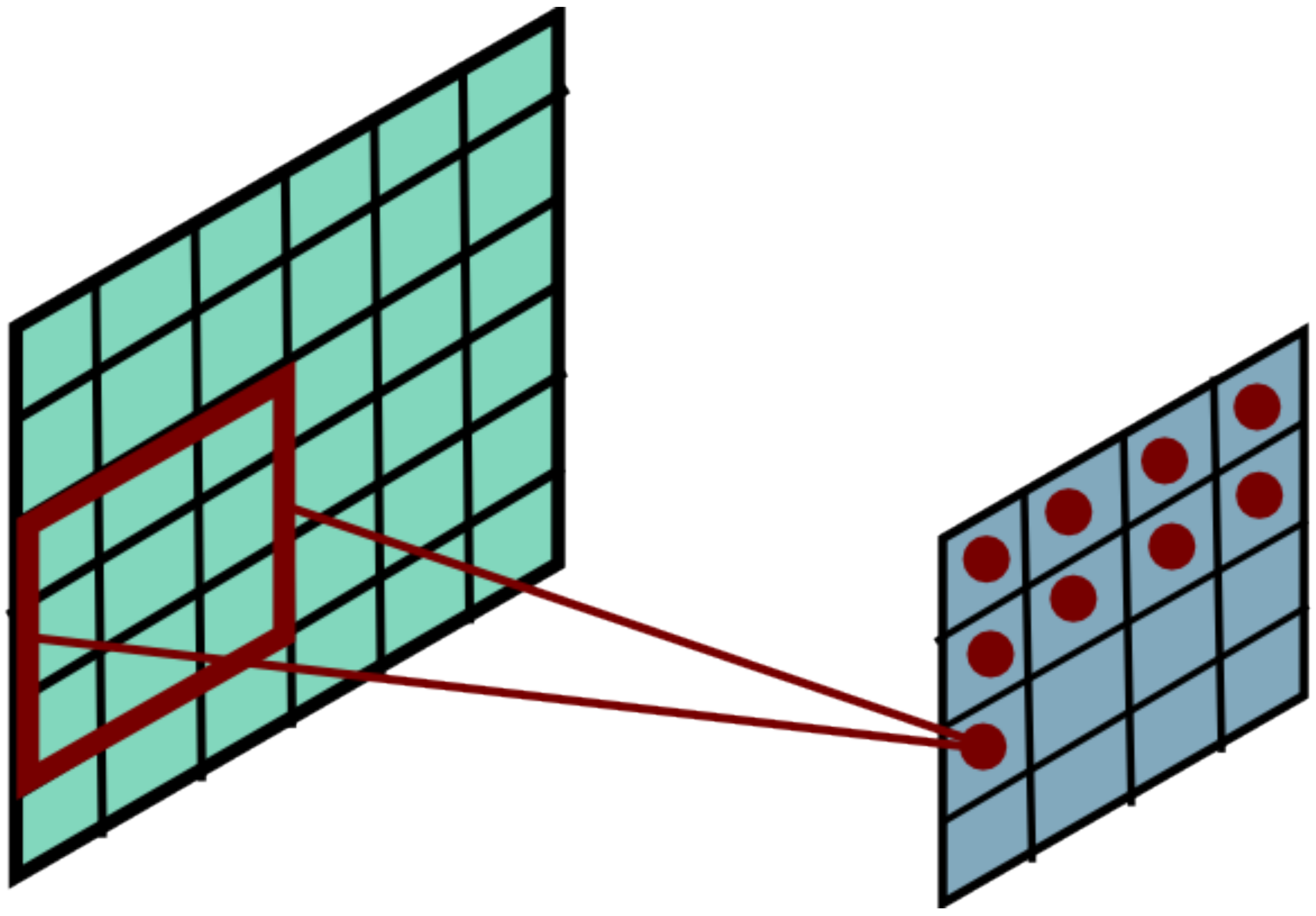
* slide from Marc'Aurelio Renzato

Convolutional Layer



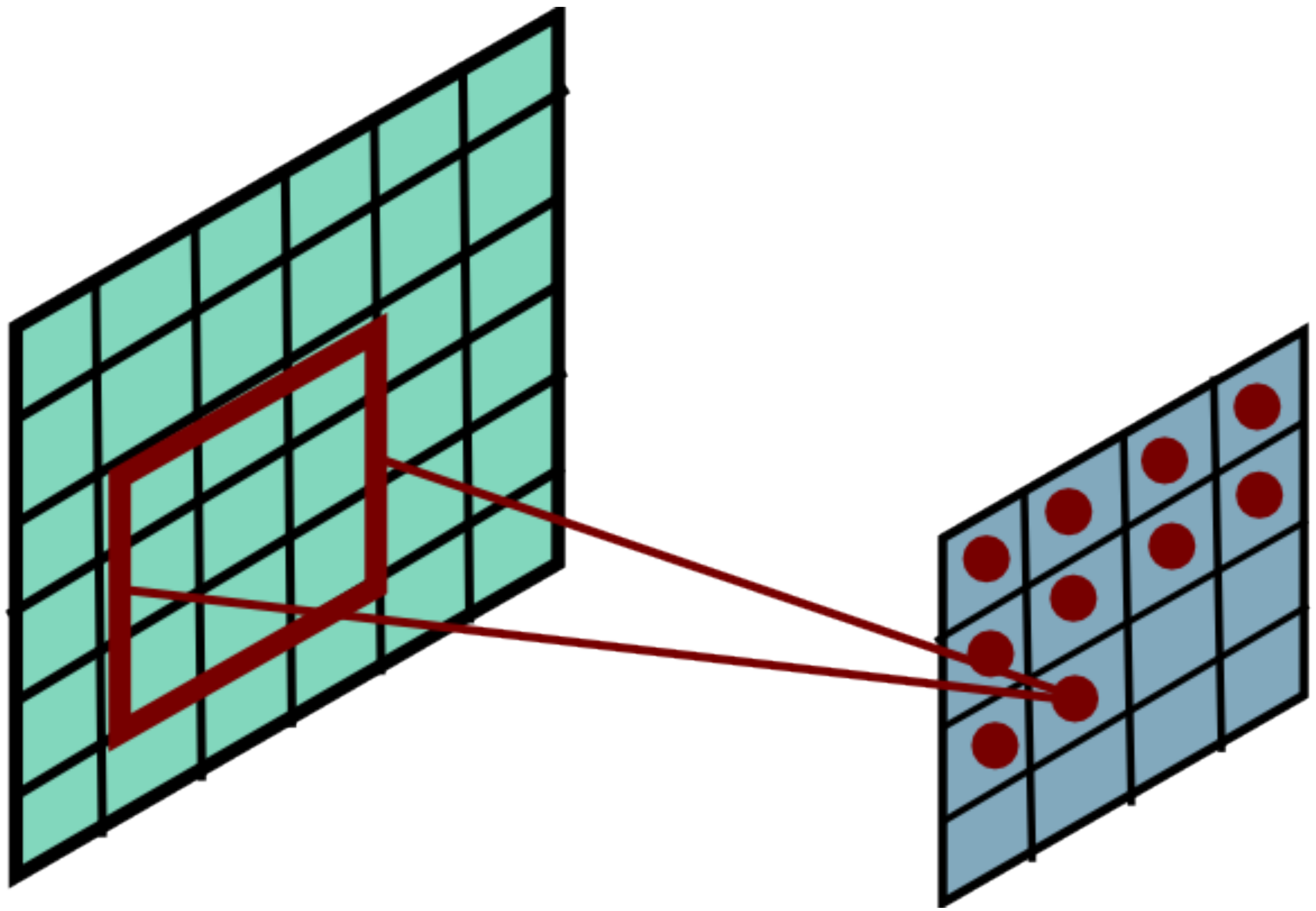
* slide from Marc'Aurelio Renzato

Convolutional Layer



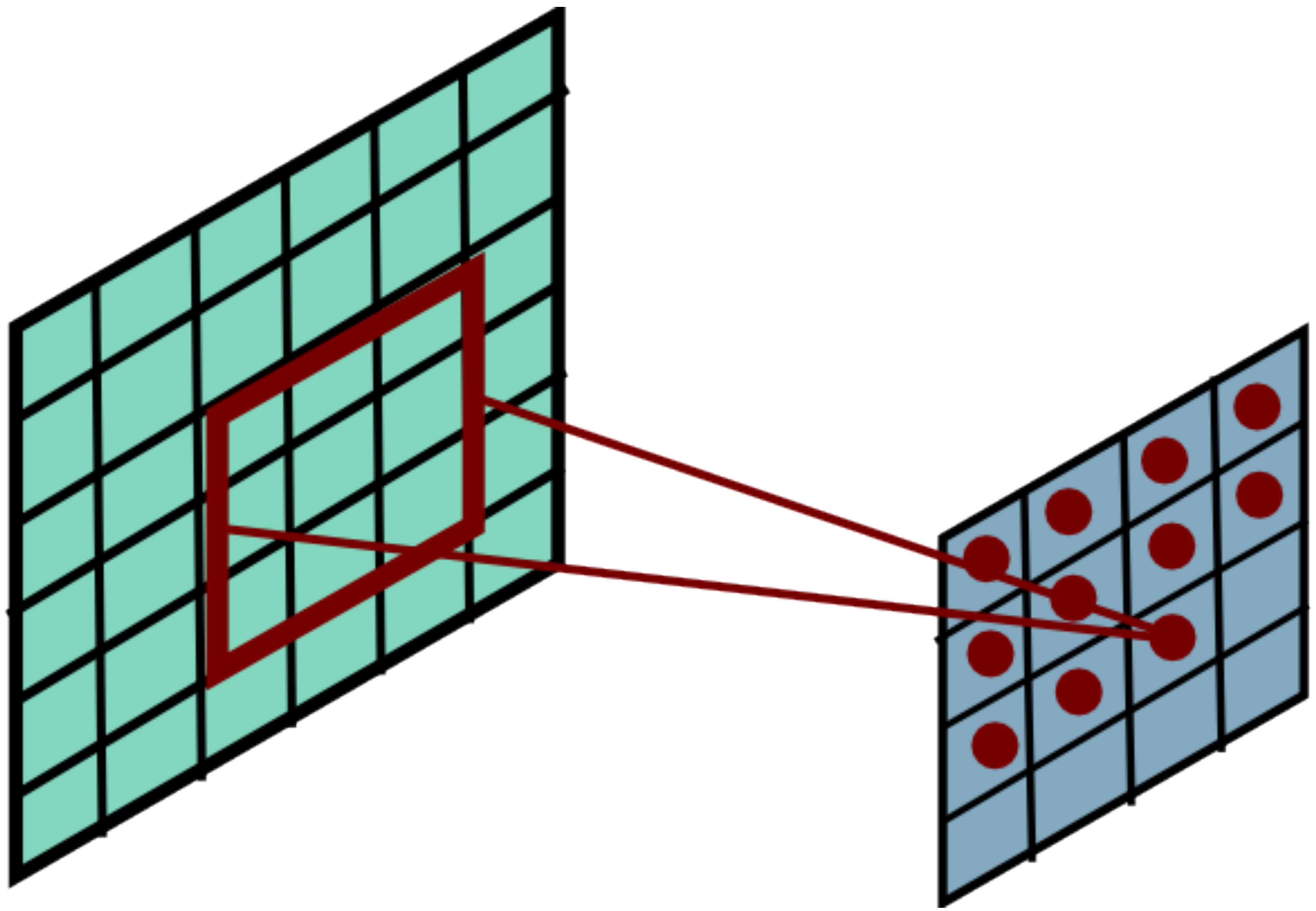
* slide from Marc'Aurelio Renzato

Convolutional Layer



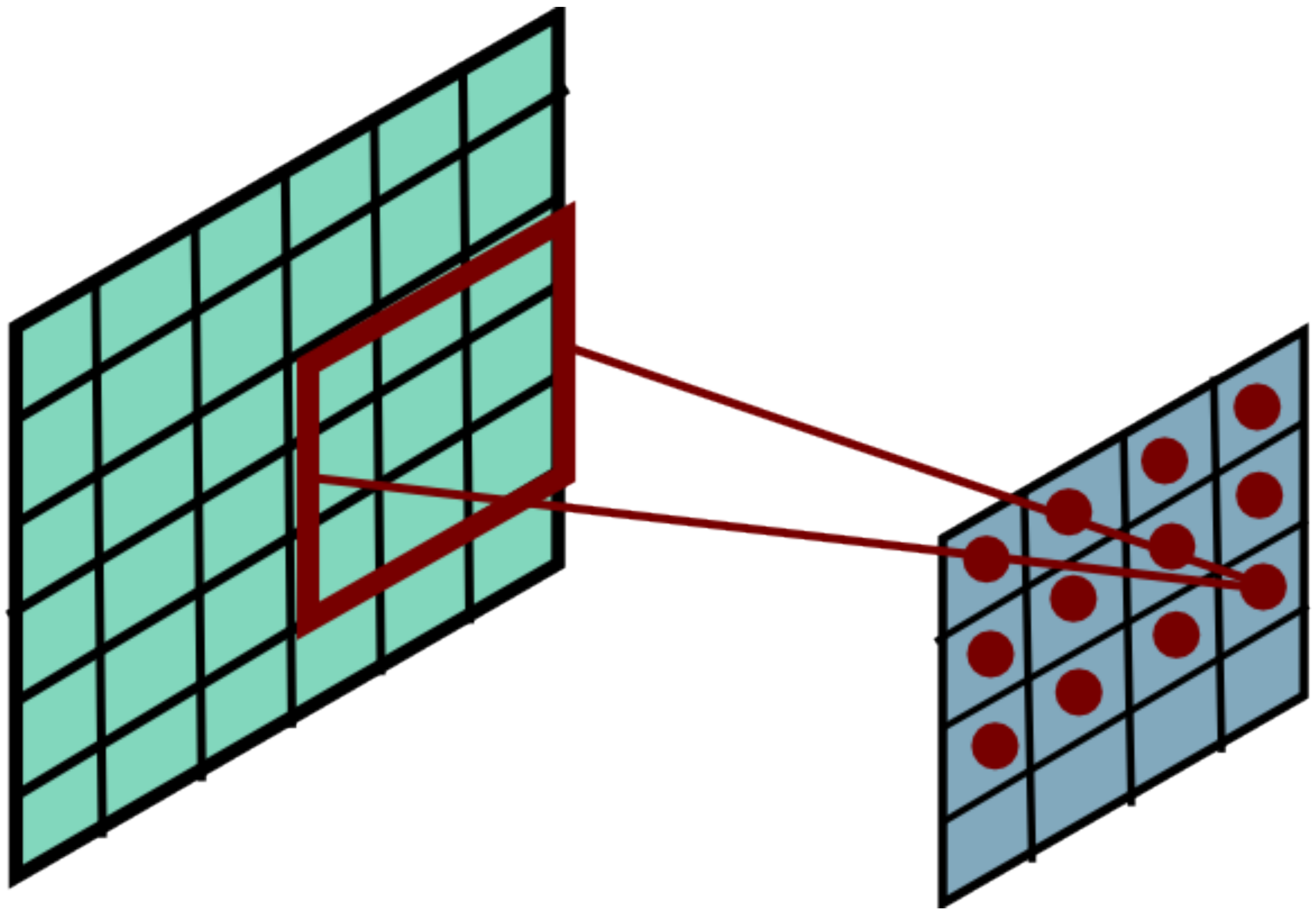
* slide from Marc'Aurelio Renzato

Convolutional Layer



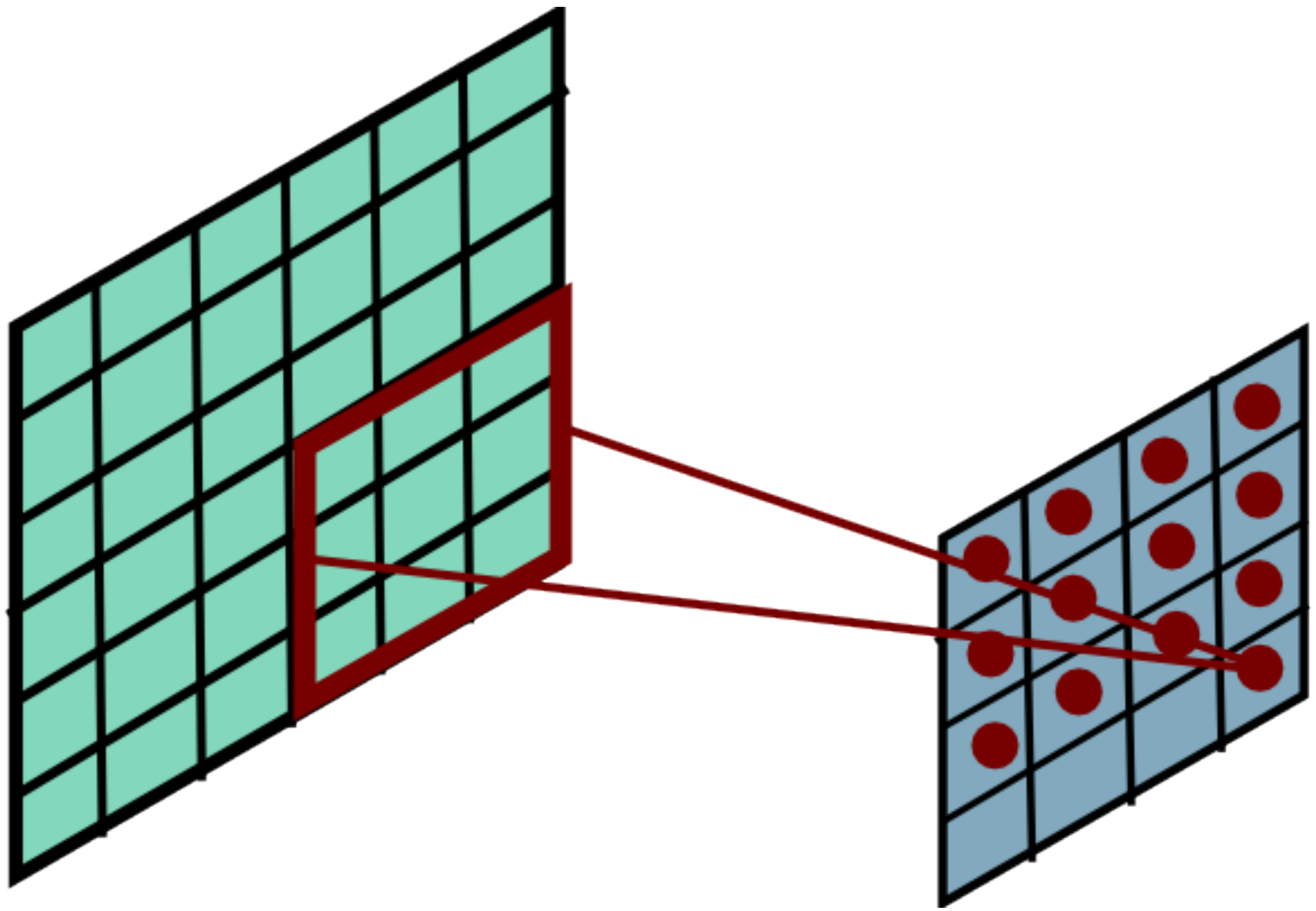
* slide from Marc'Aurelio Renzato

Convolutional Layer



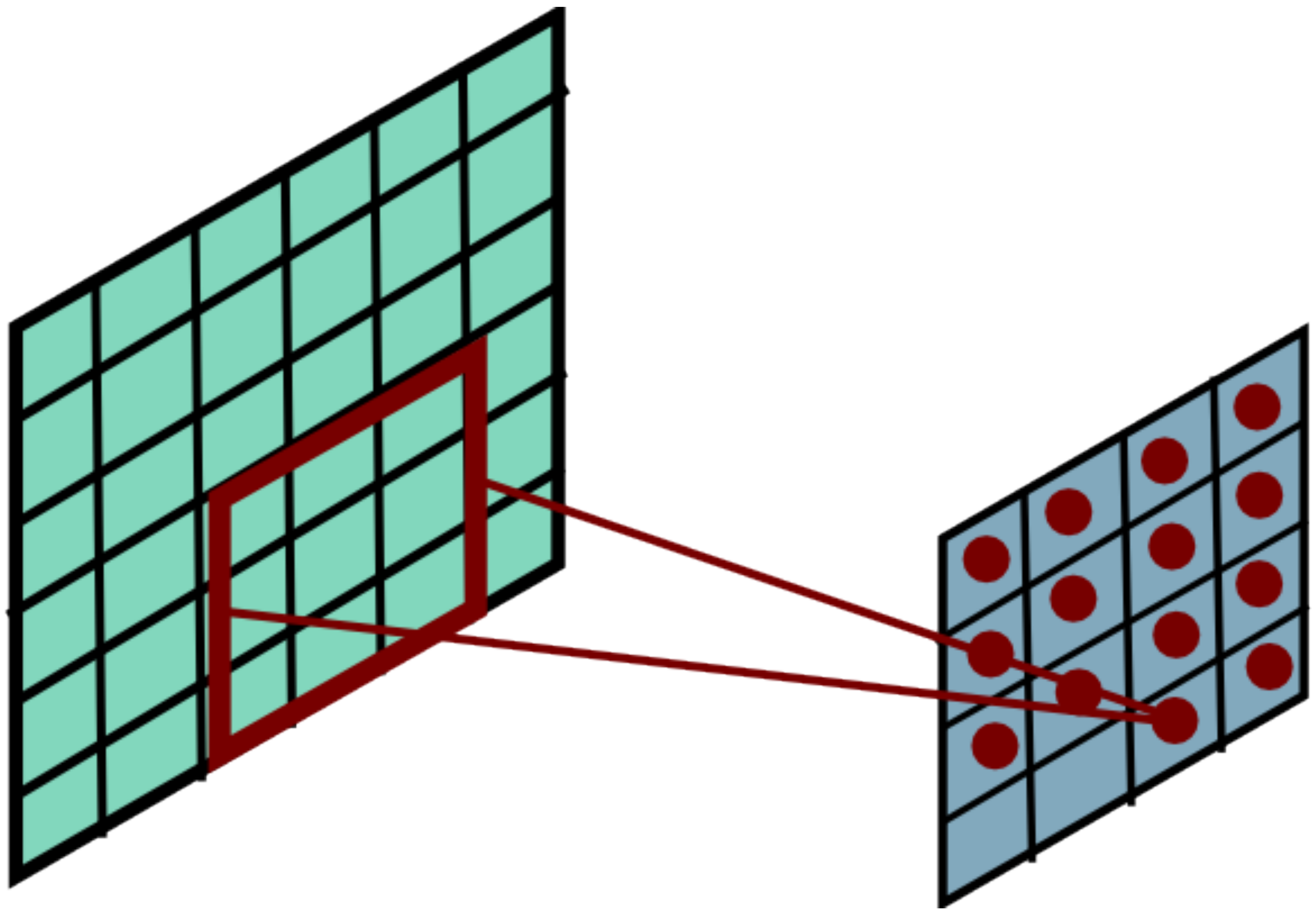
* slide from Marc'Aurelio Renzato

Convolutional Layer



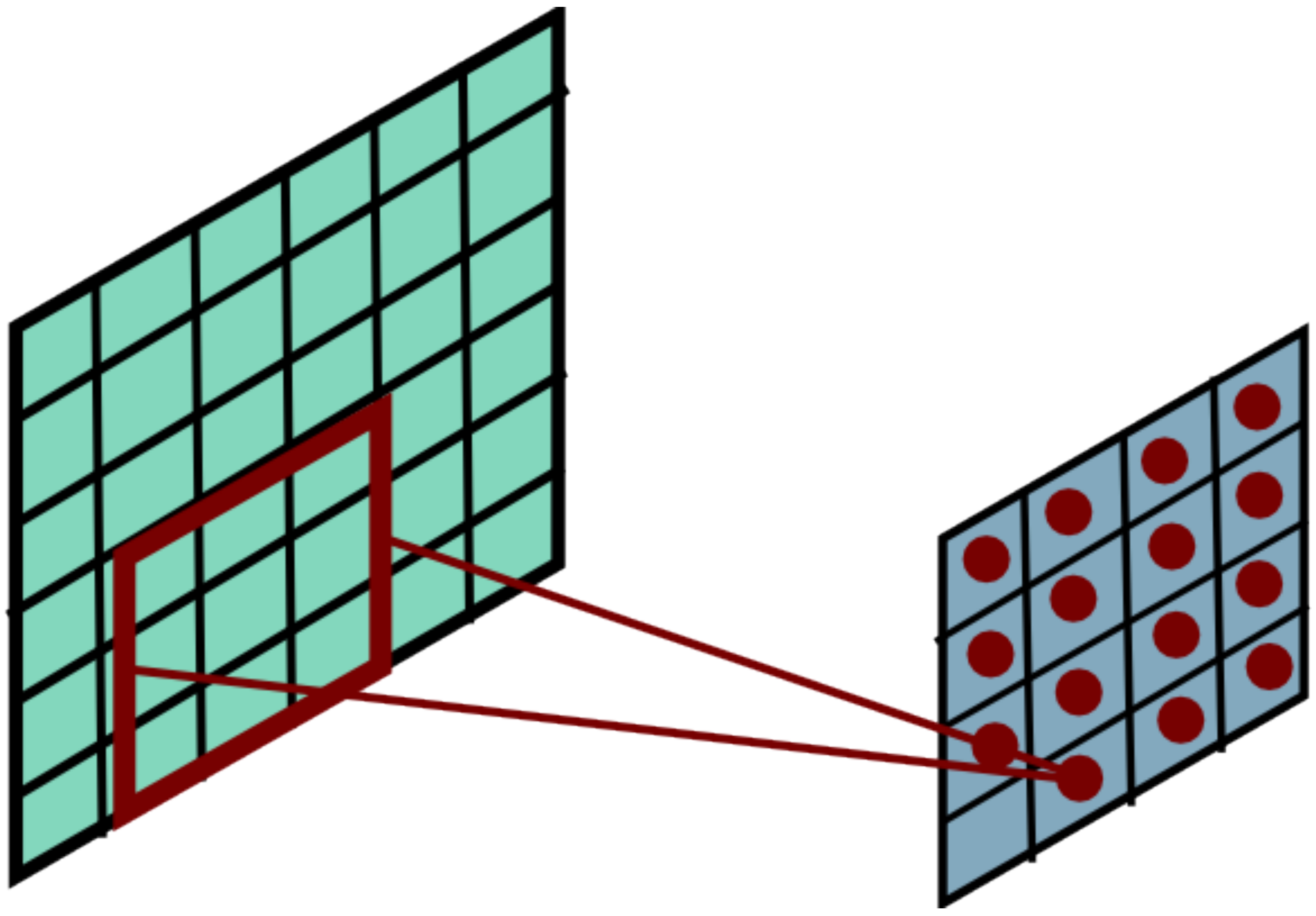
* slide from Marc'Aurelio Renzato

Convolutional Layer



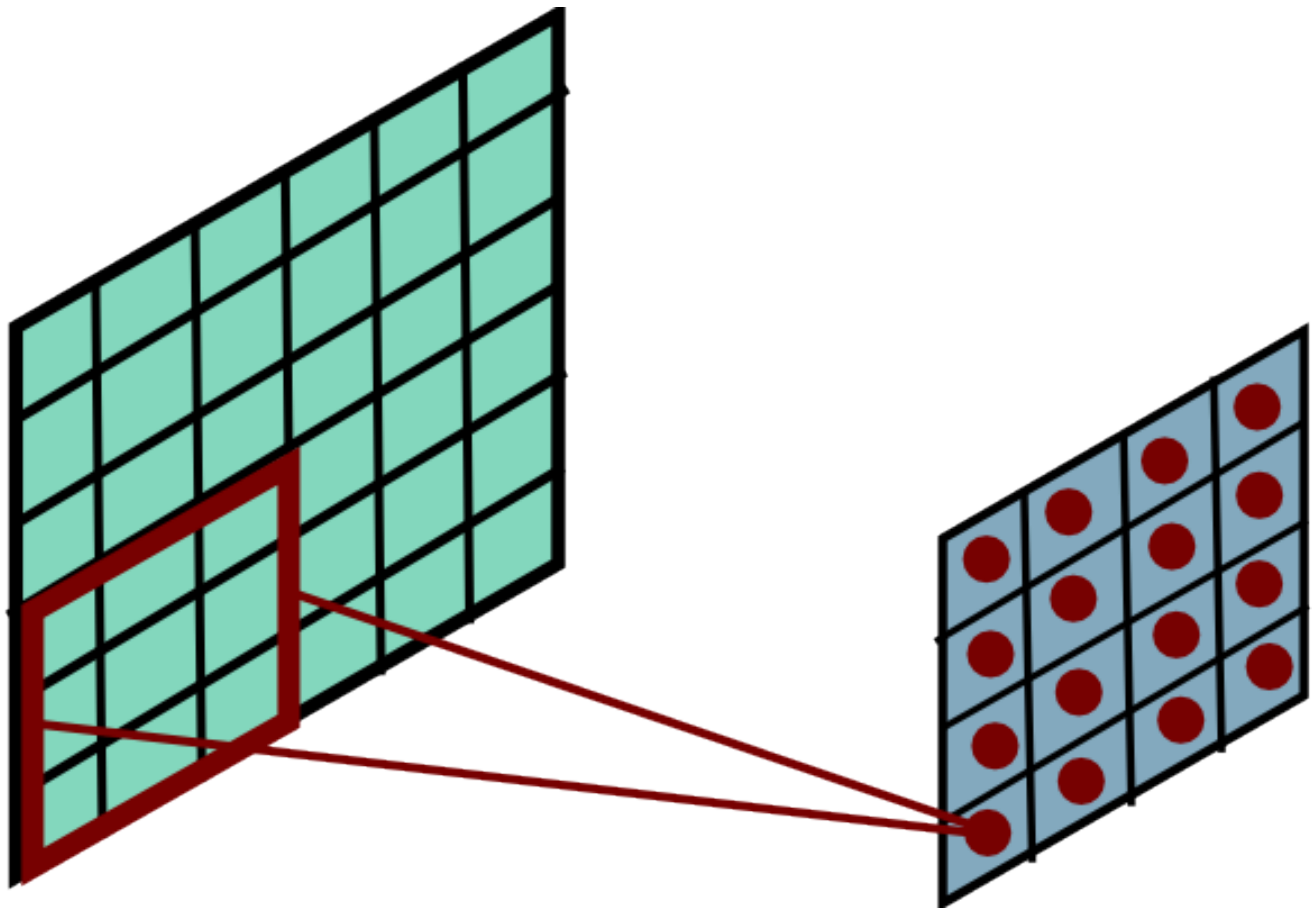
* slide from Marc'Aurelio Renzato

Convolutional Layer



* slide from Marc'Aurelio Renzato

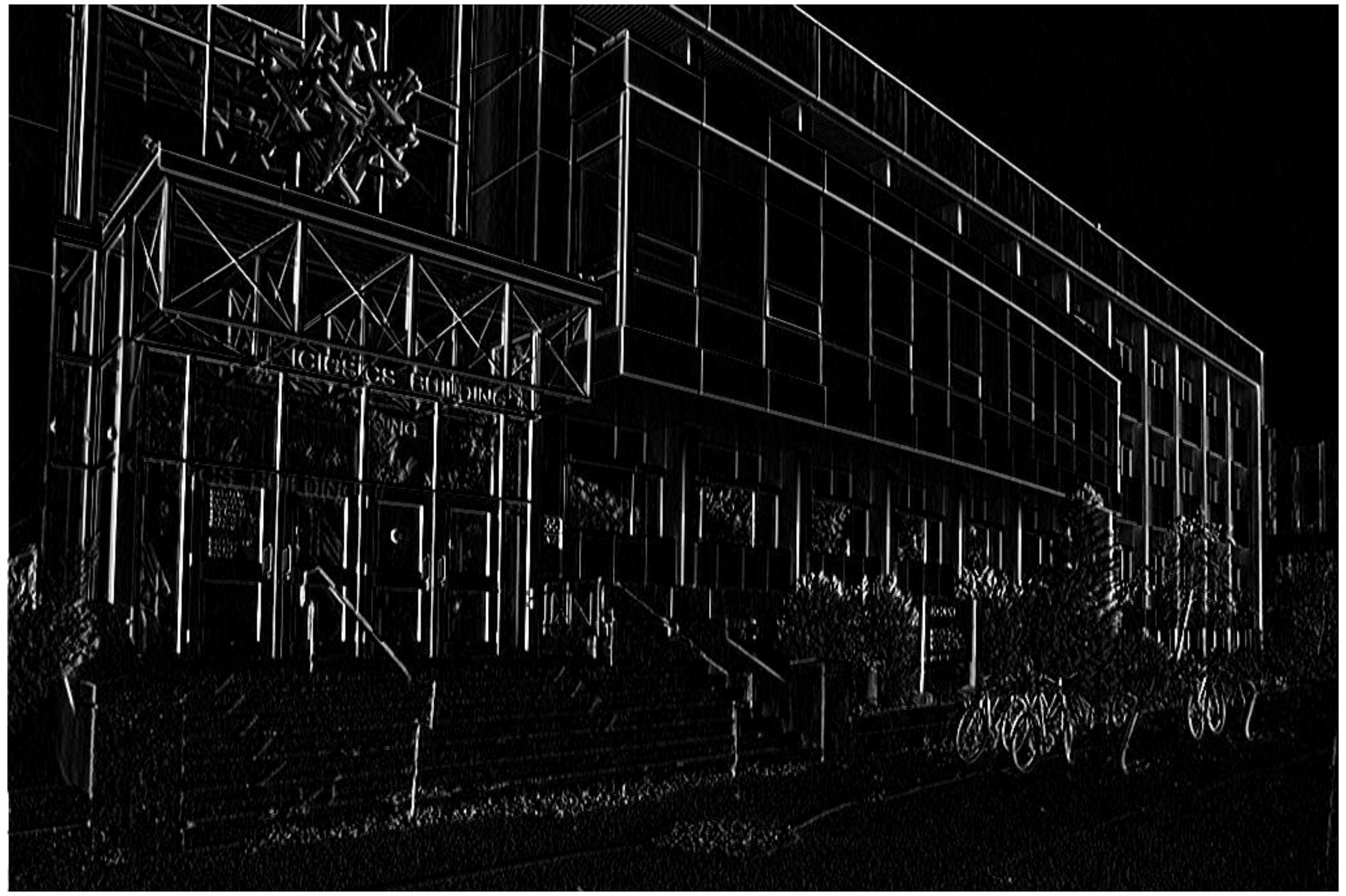
Convolutional Layer



* slide from Marc'Aurelio Renzato

Convolution Layer

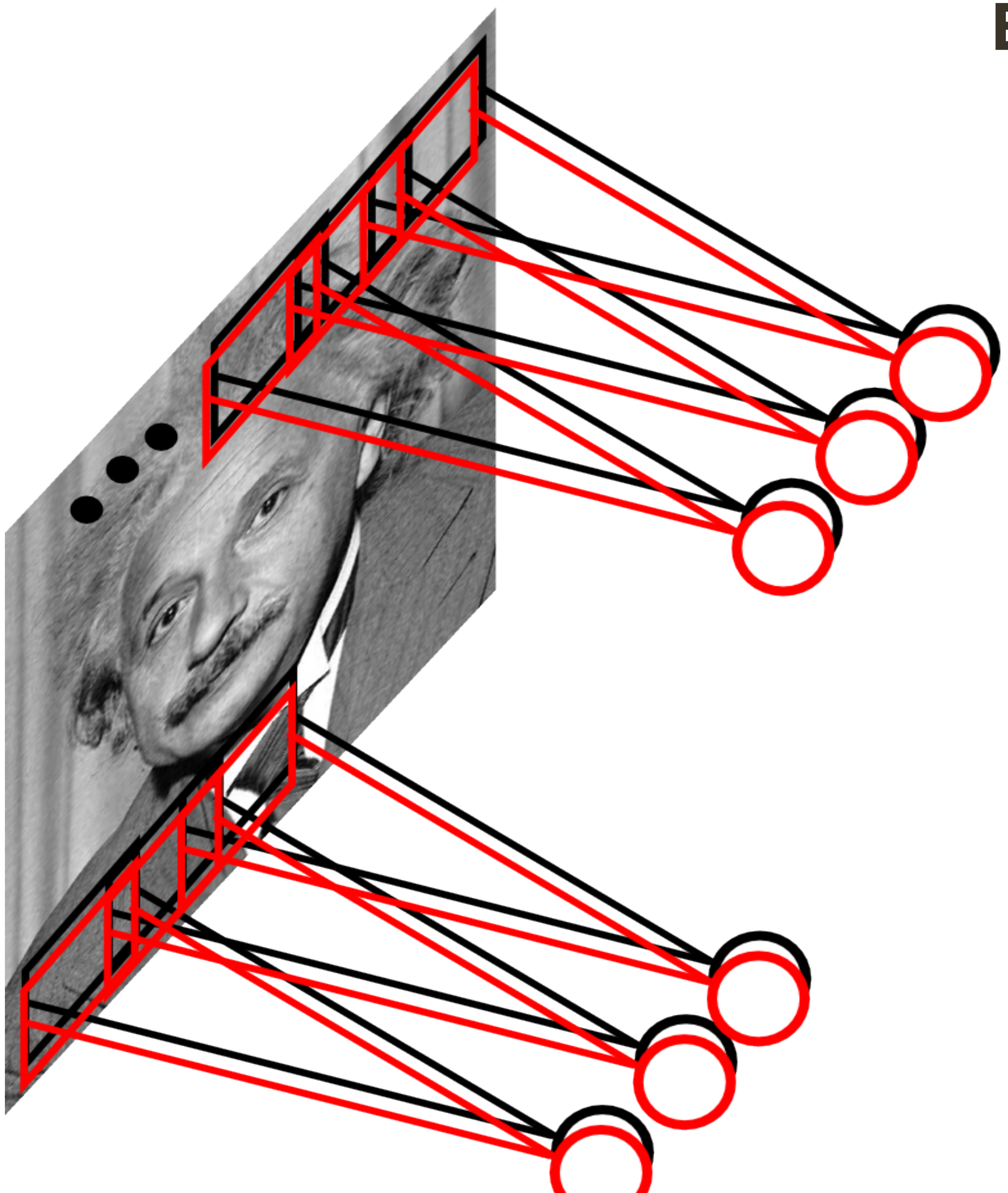
$$\star \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \longrightarrow$$



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

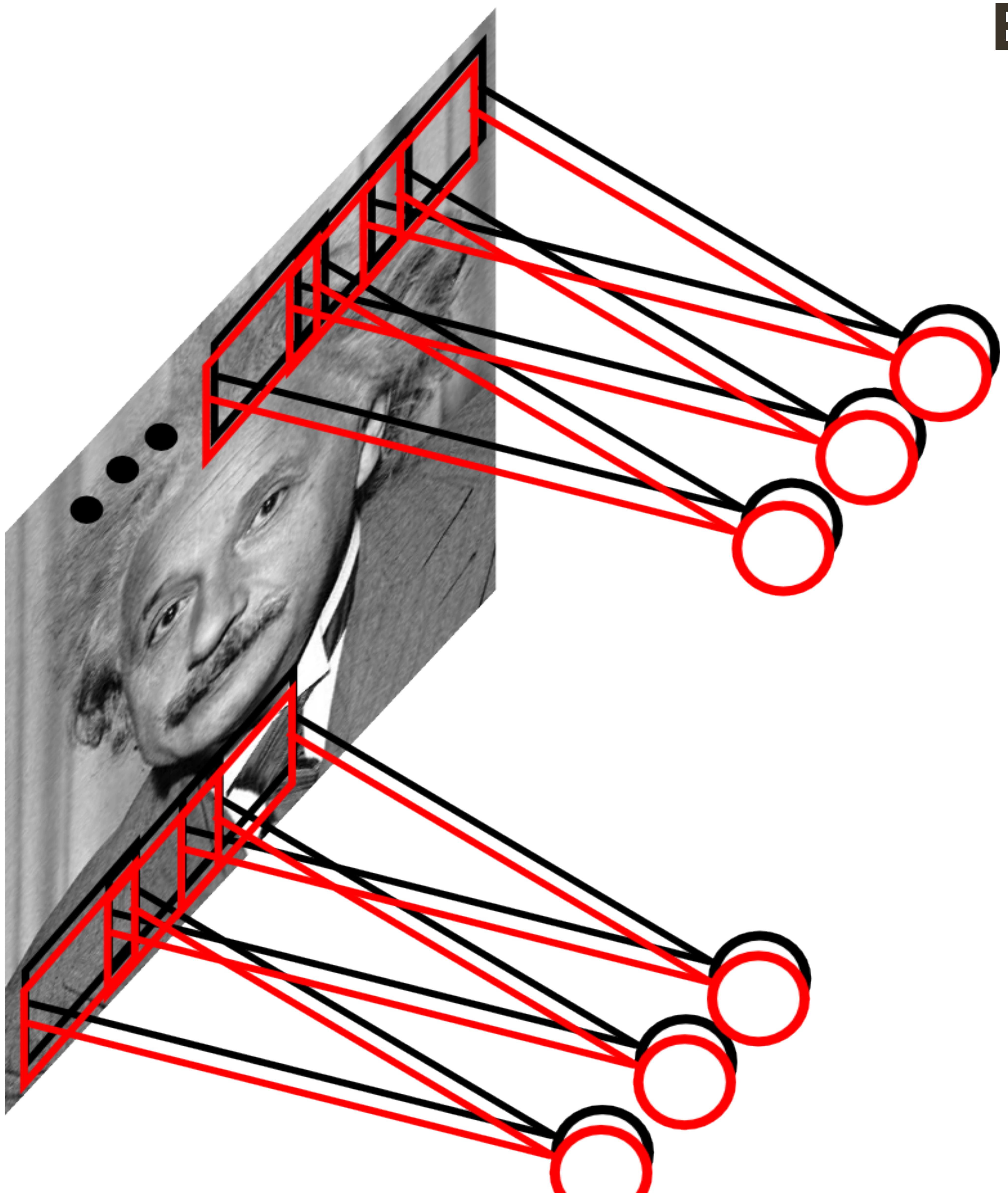
Filter size: 10 x 10

# 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

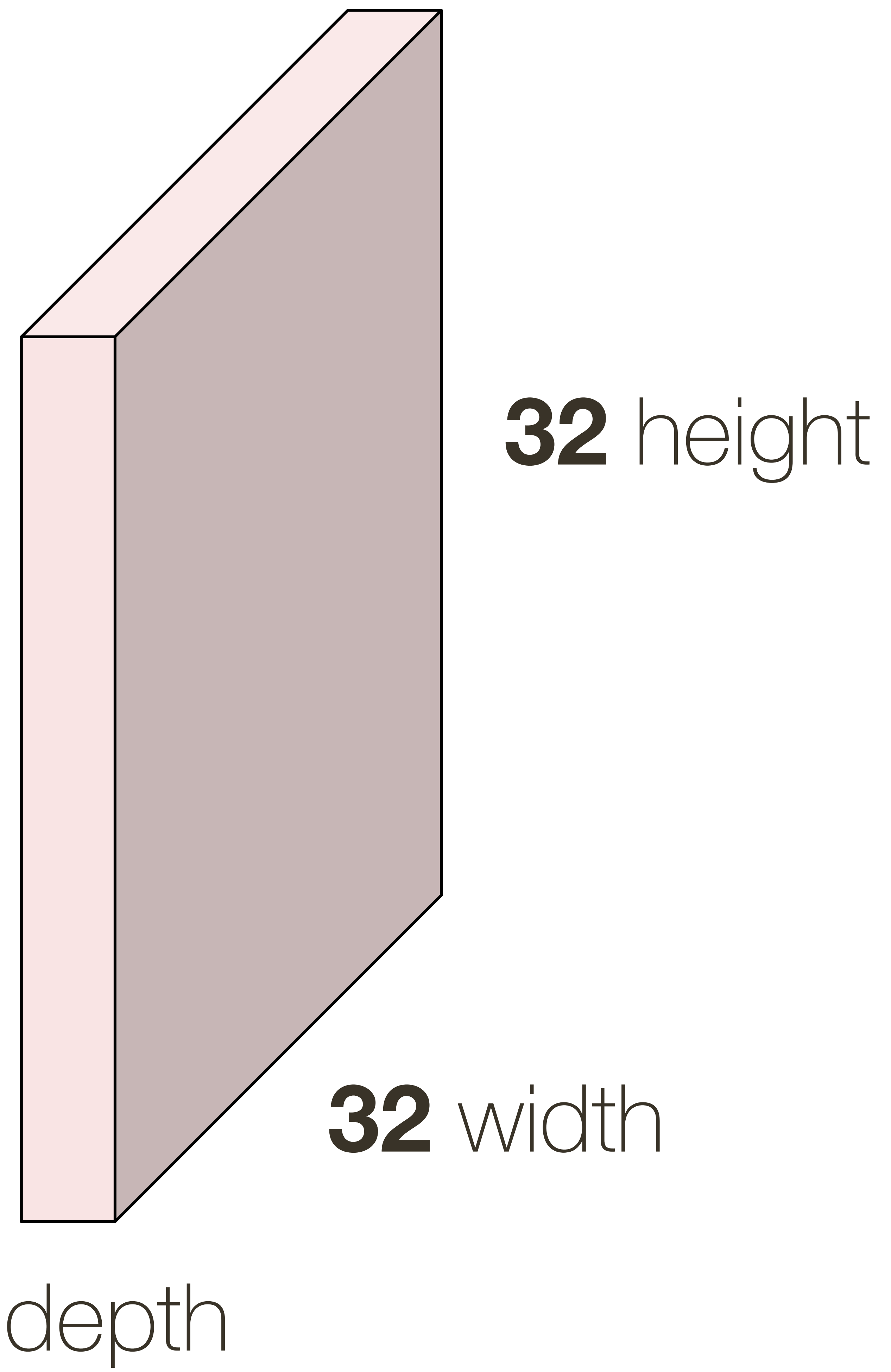
= 2000 parameters

Learn **multiple filters**

* slide from Marc'Aurelio Renzato

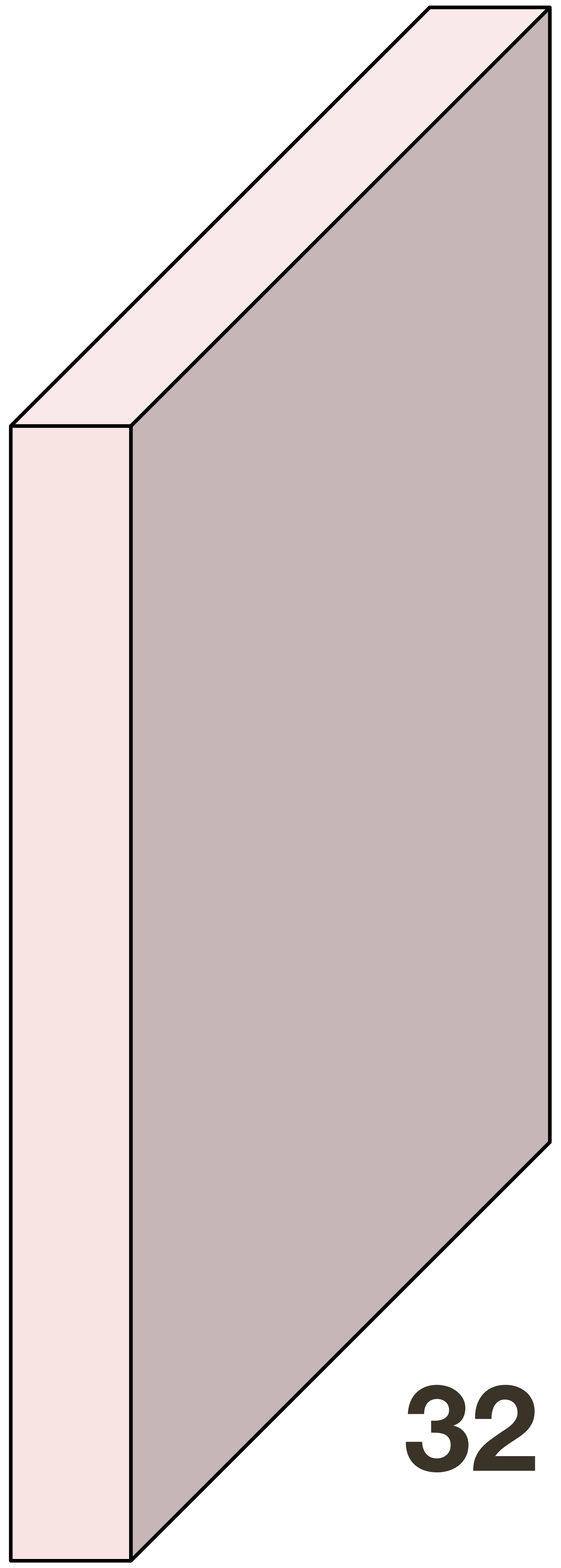
Convolutional Layer

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



Convolutional Layer

32 x 32 x 3 **image**

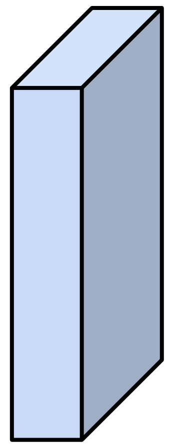


32 height

32 width

3 depth

5 x 5 x 3 **filter**

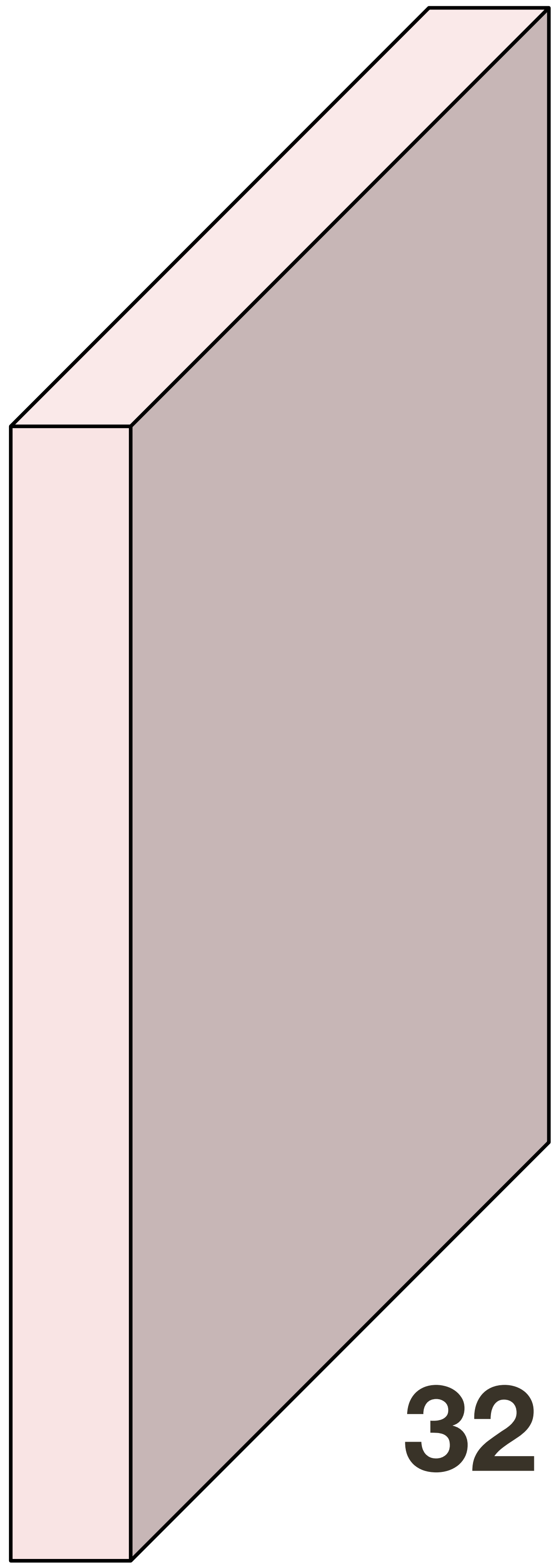


Convolve the filter with the image
(i.e., “slide over the image spatially,
computing dot products”)

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

Convolutional Layer

32 x 32 x 3 **image**



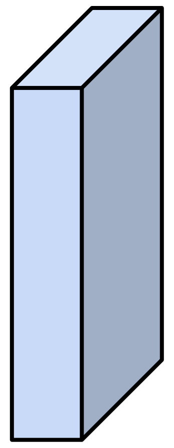
32 height

32 width

3 depth

Filters always extend the full depth of the input volume

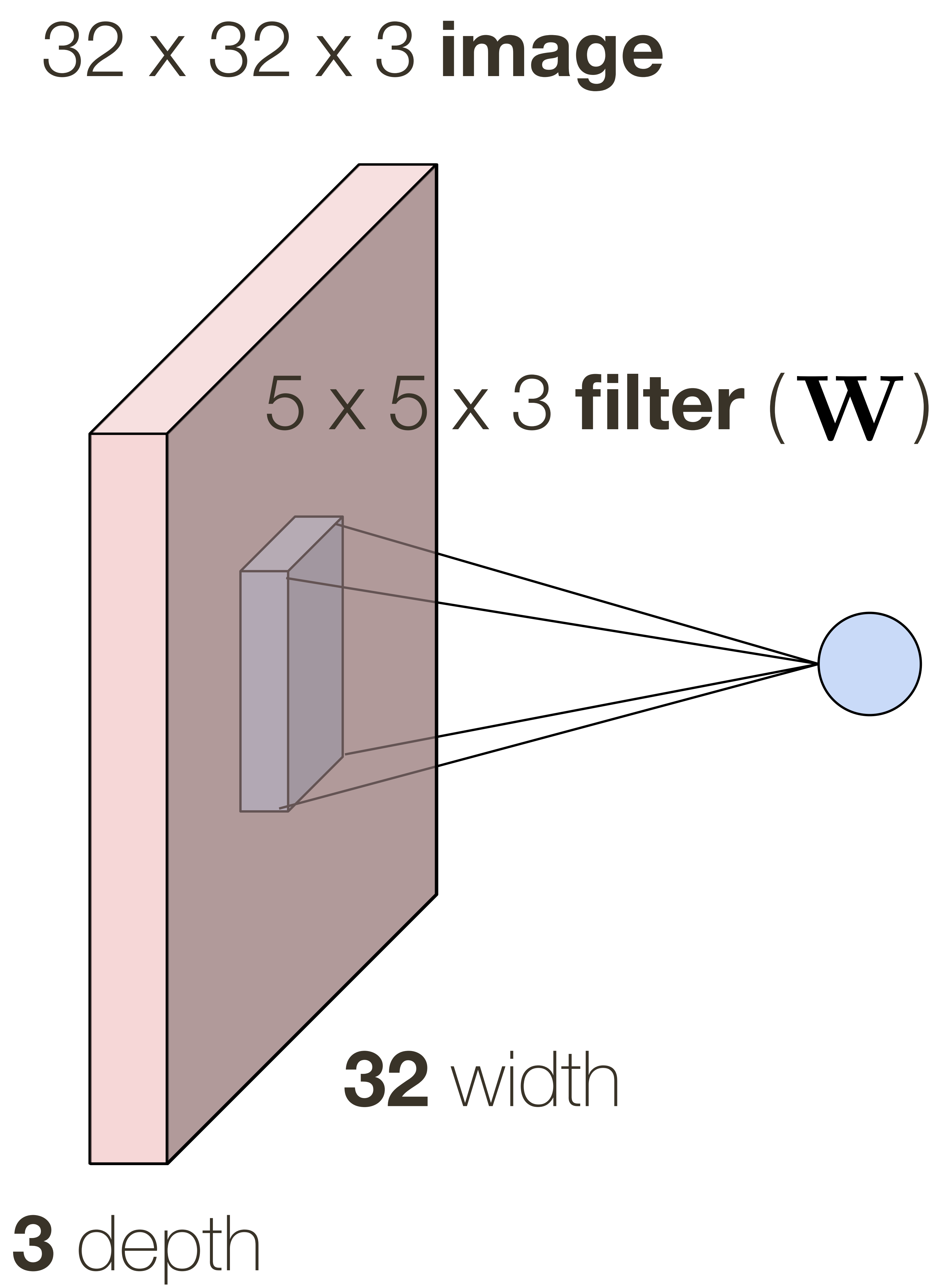
5 x 5 x 3 **filter**



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

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

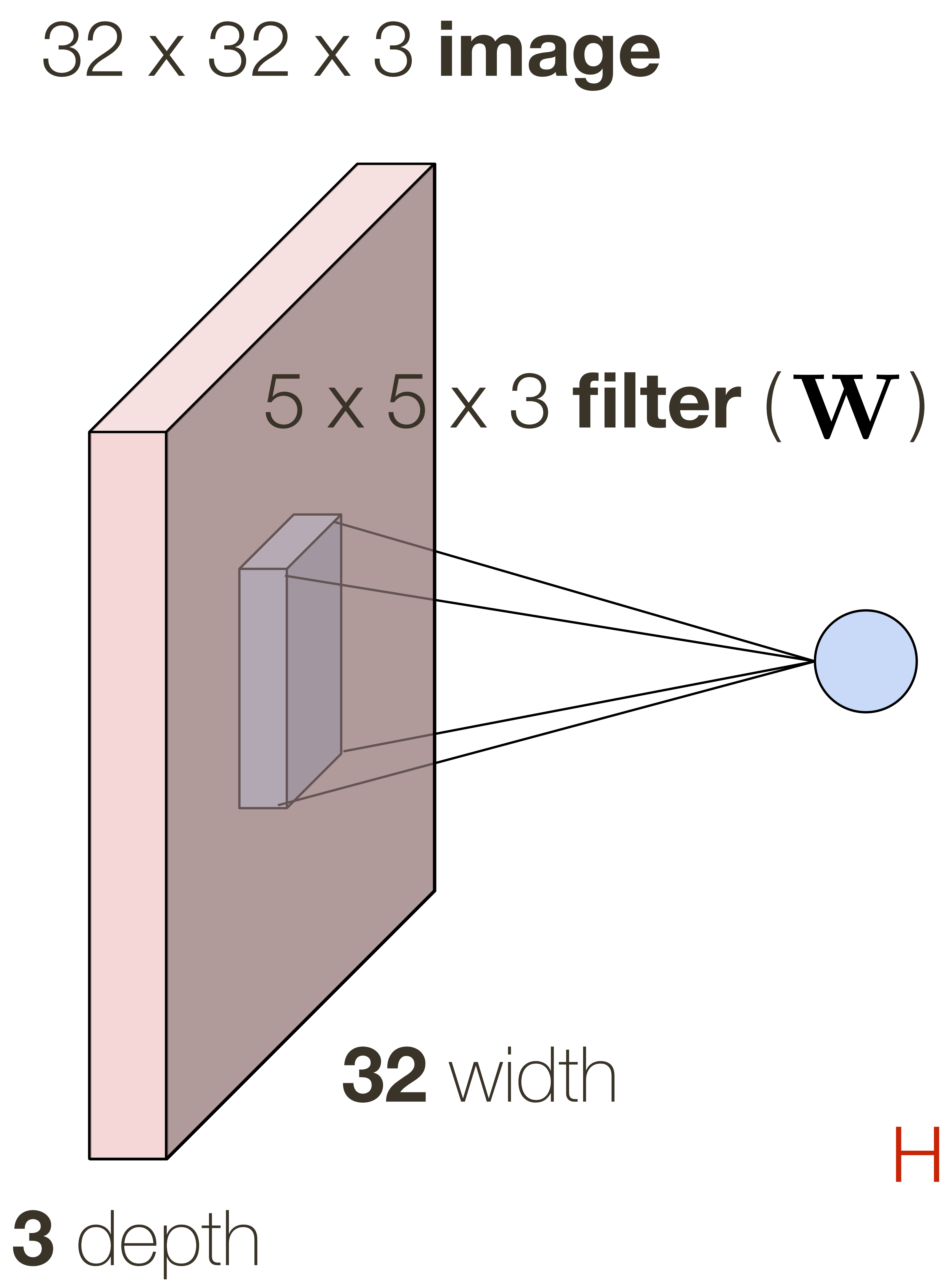
Convolutional Layer



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



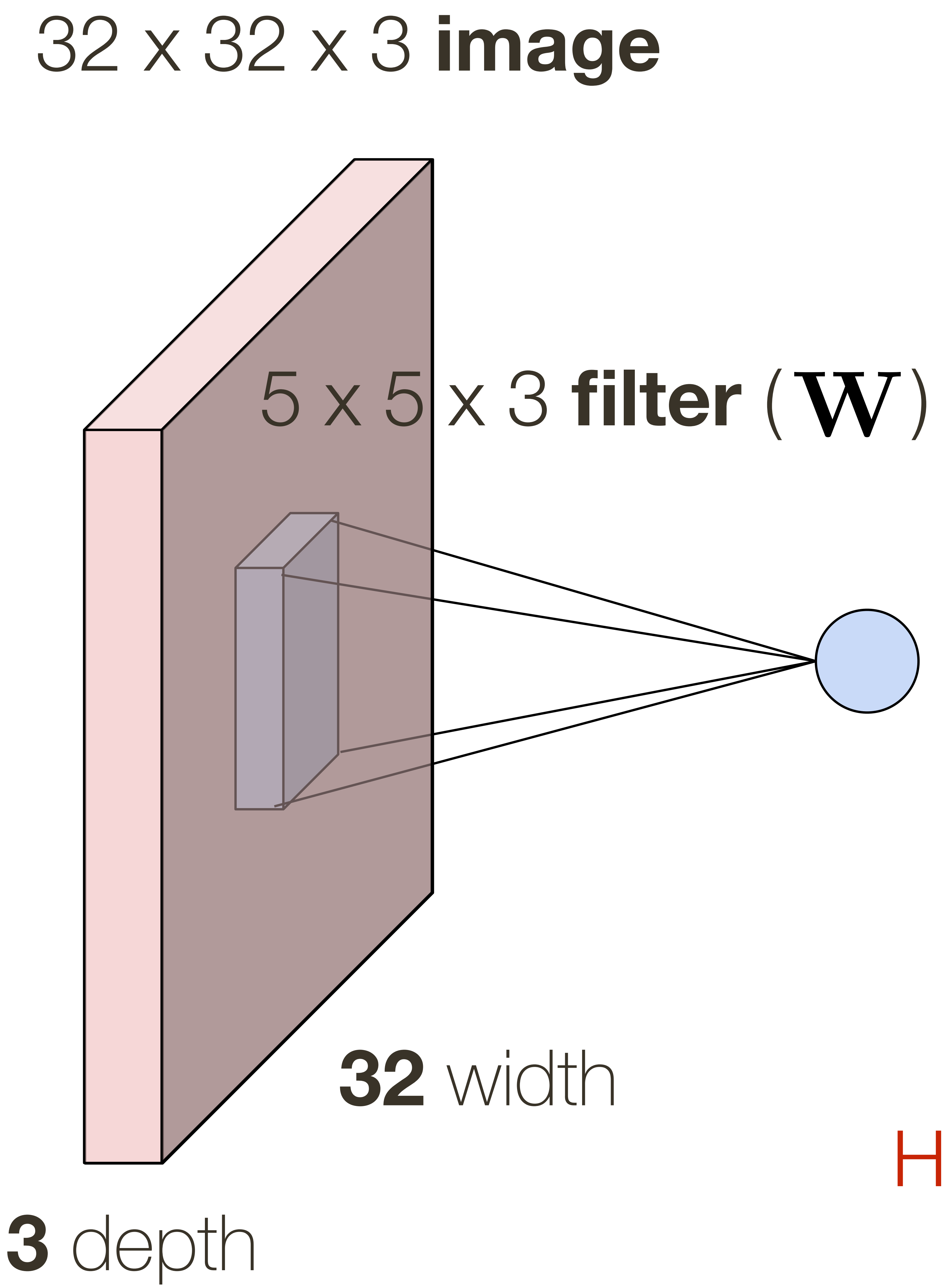
1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

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

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

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

Convolutional Layer



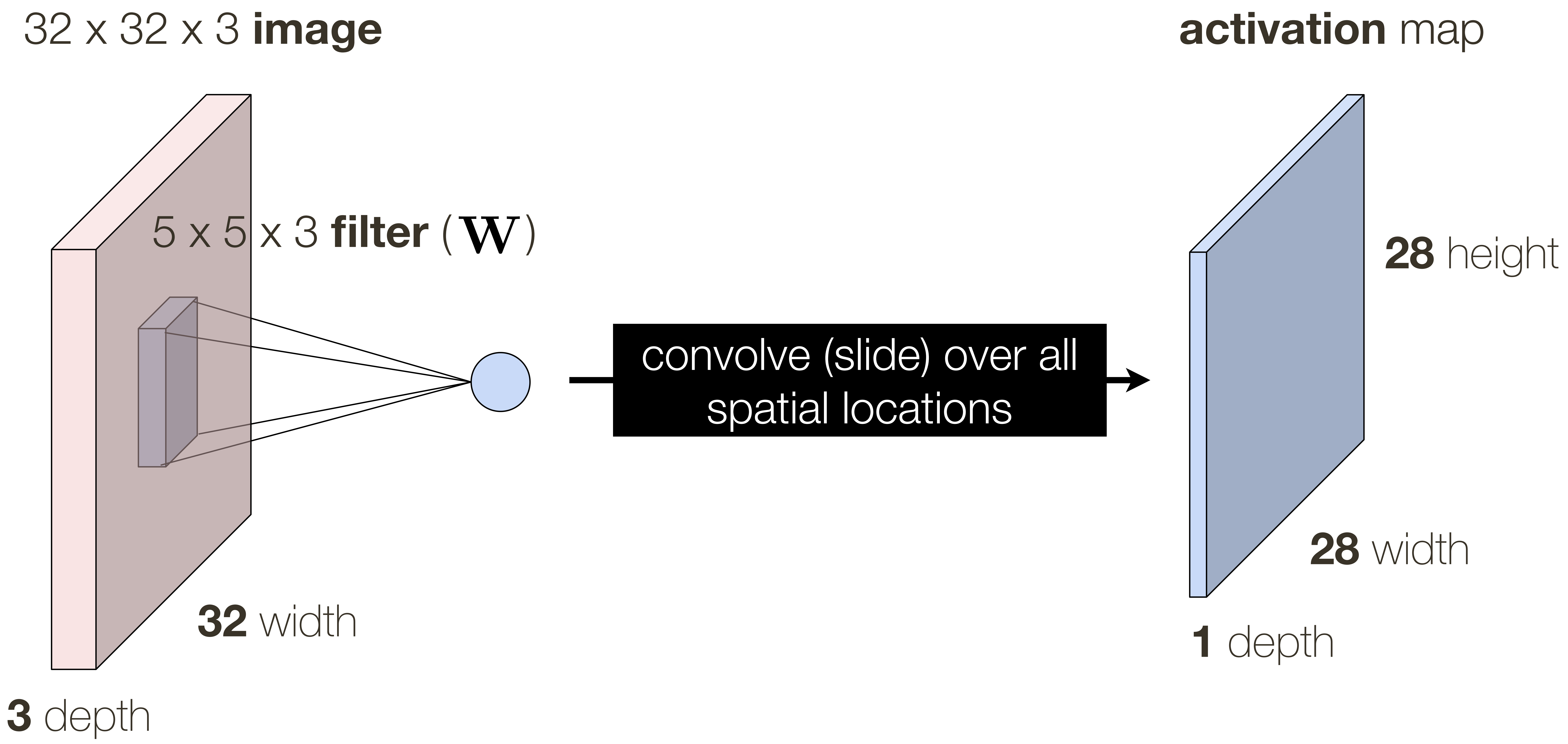
1 number: the result of taking a dot product between the filter and a small 5 x 5 x 3 part of the image

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

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

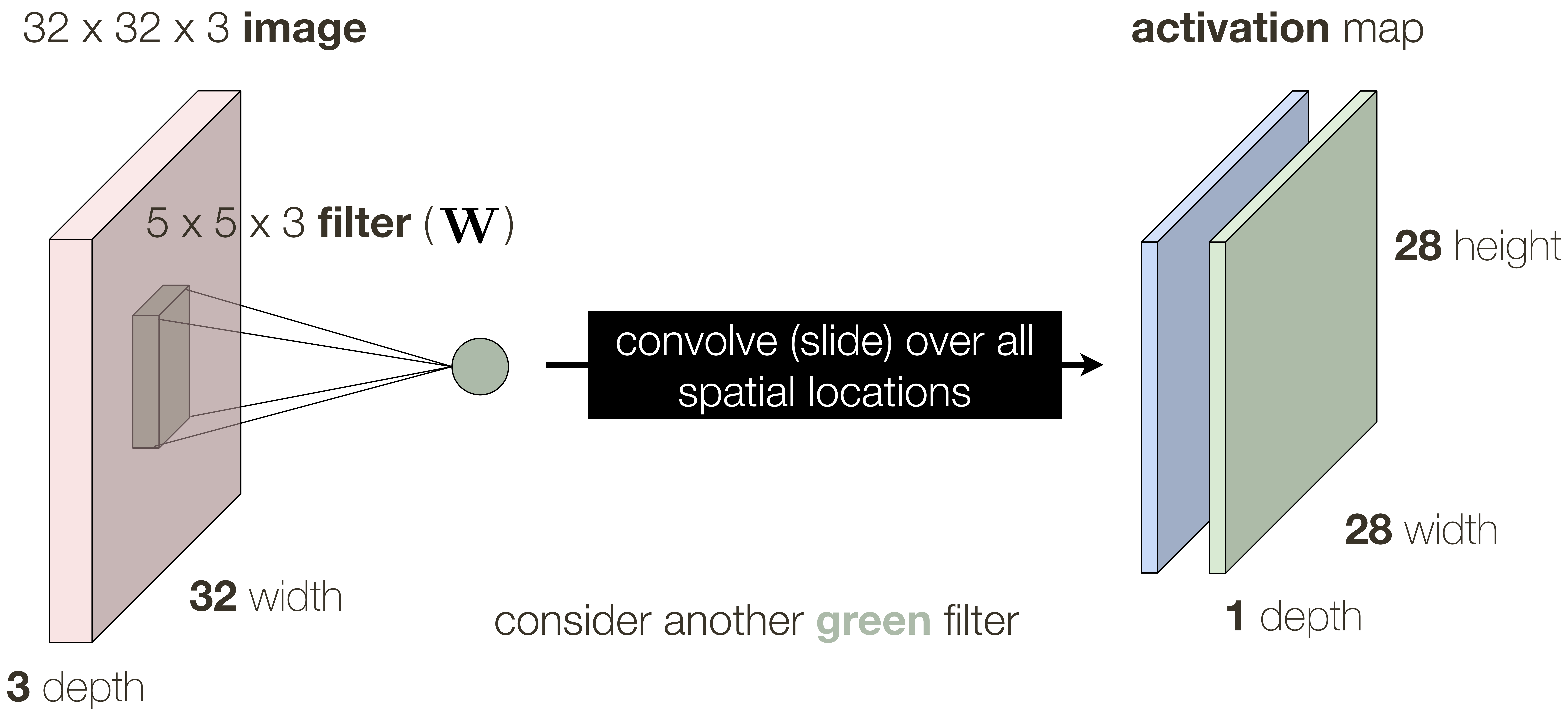
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer



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

Convolutional Layer

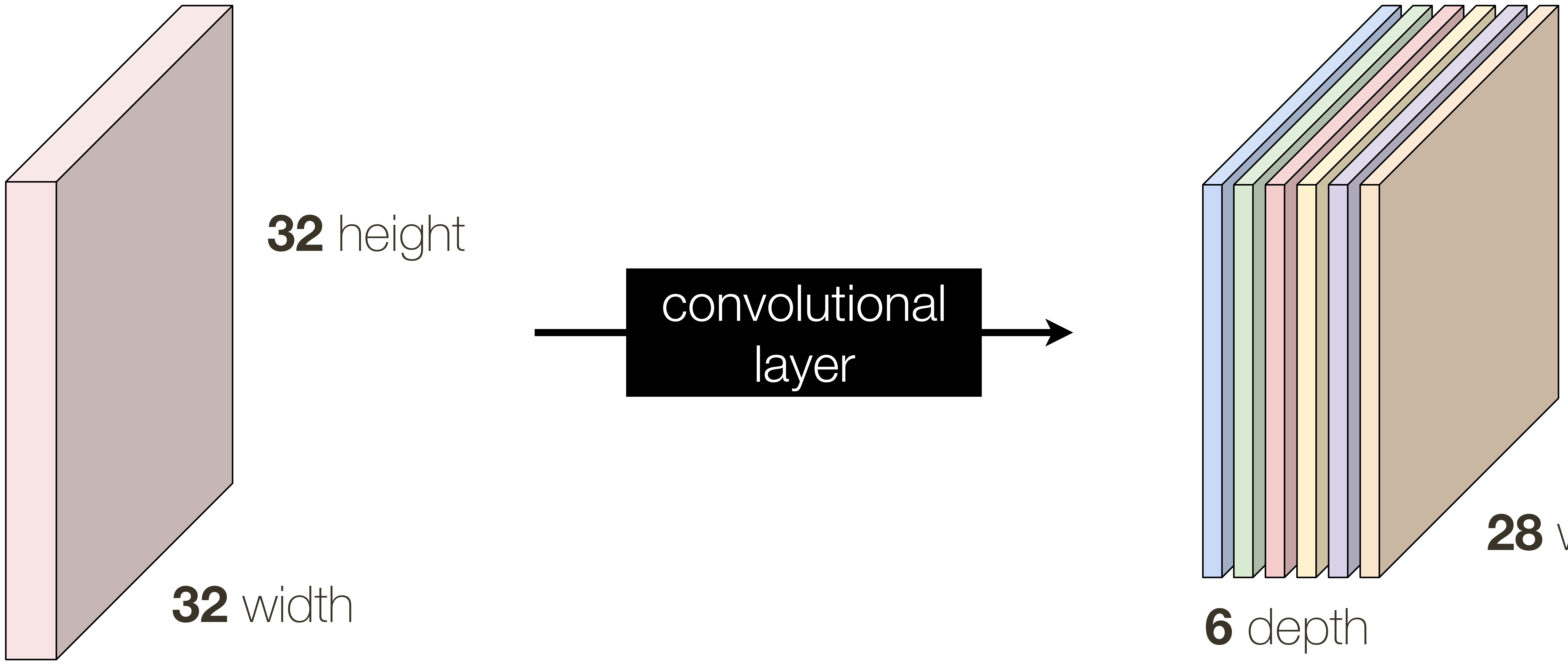


consider another **green** filter

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

Convolutional Layer

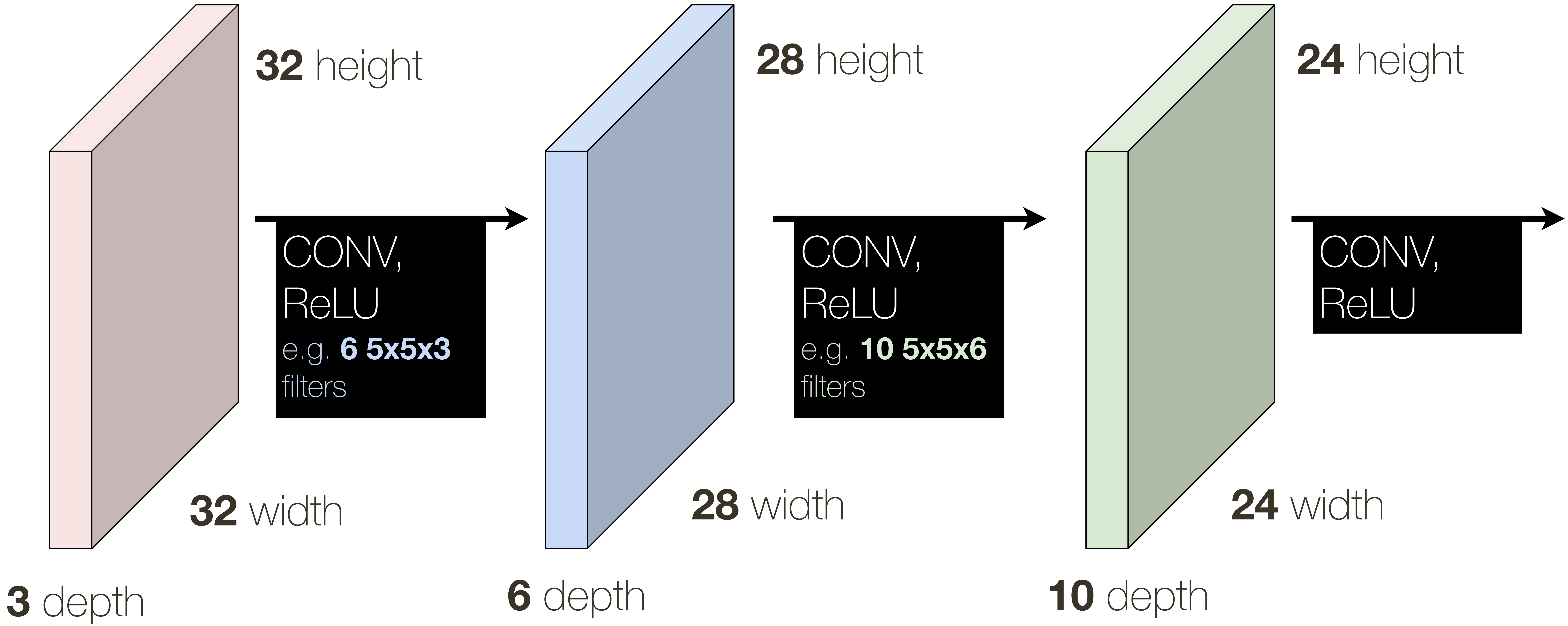
If we have 6 5x5 filter, we'll get 6 separate activation maps: **activation** map



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

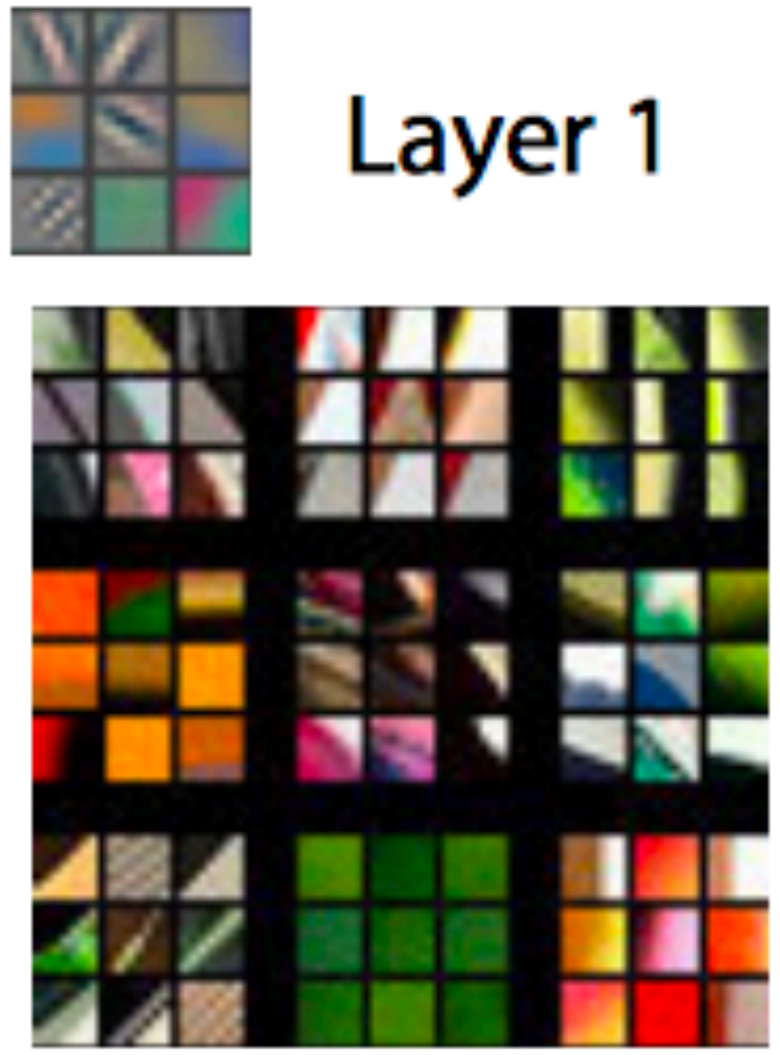
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

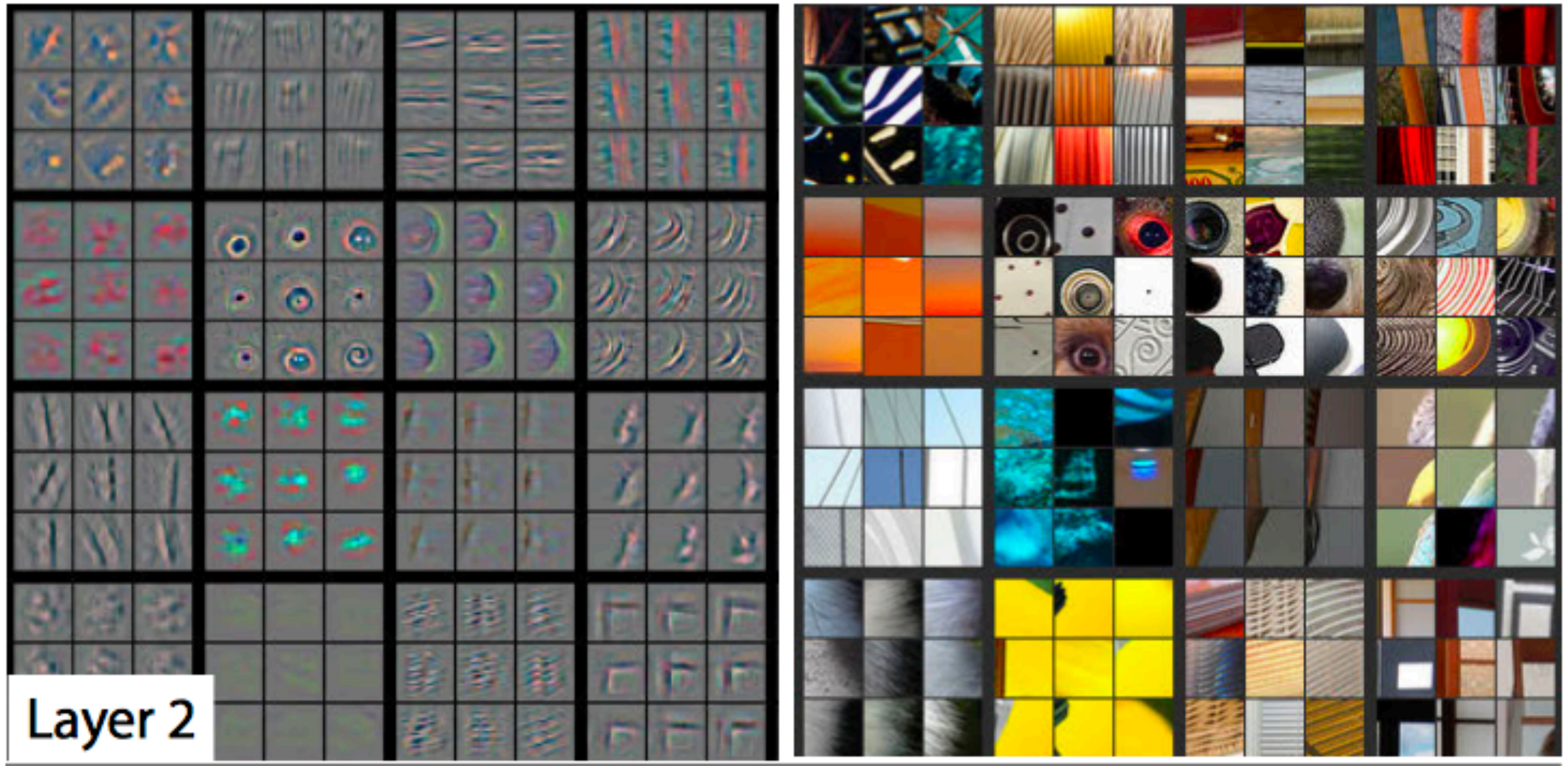
Convolutional Neural Network (ConvNet)



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

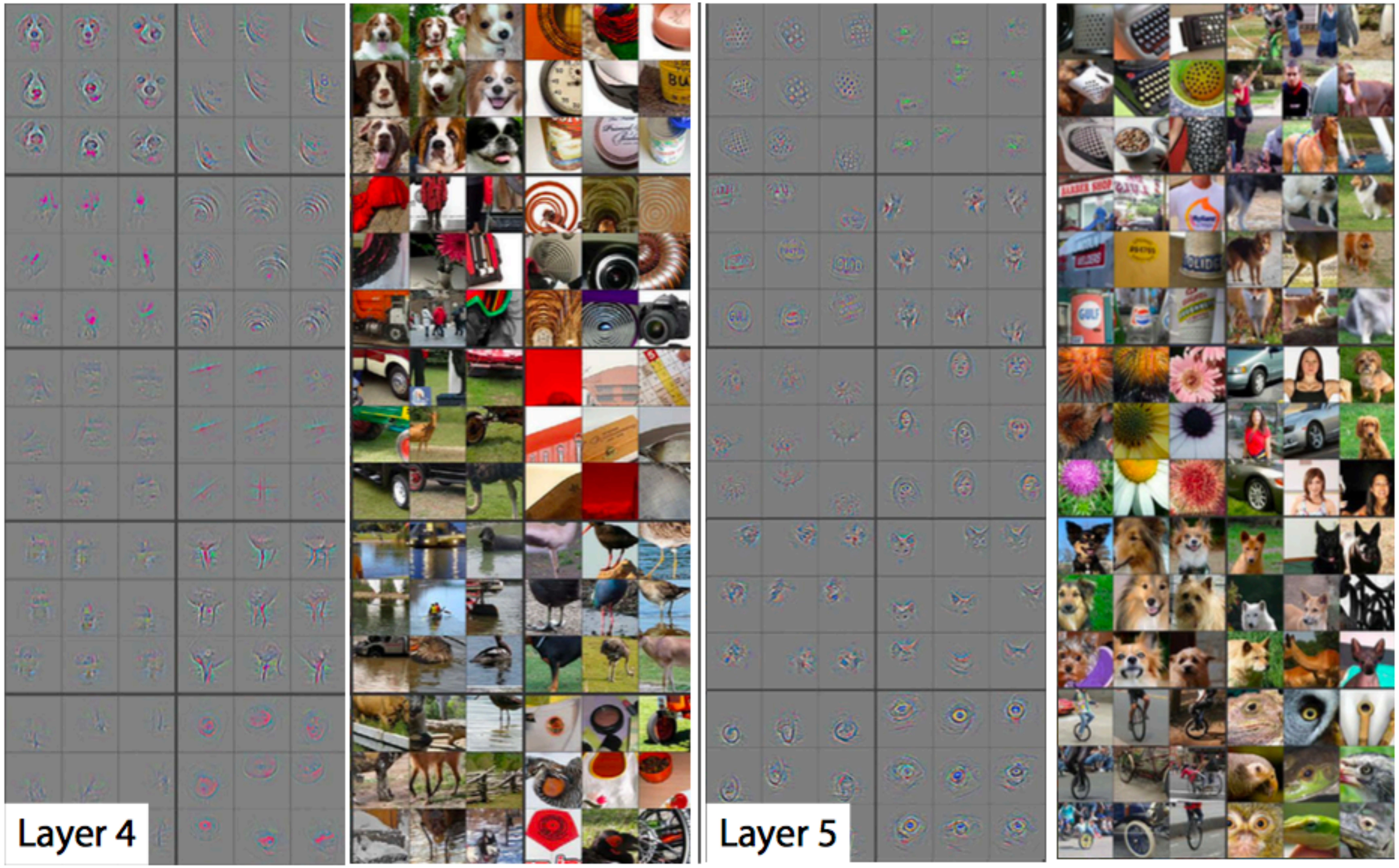
What **filters** do networks learn?





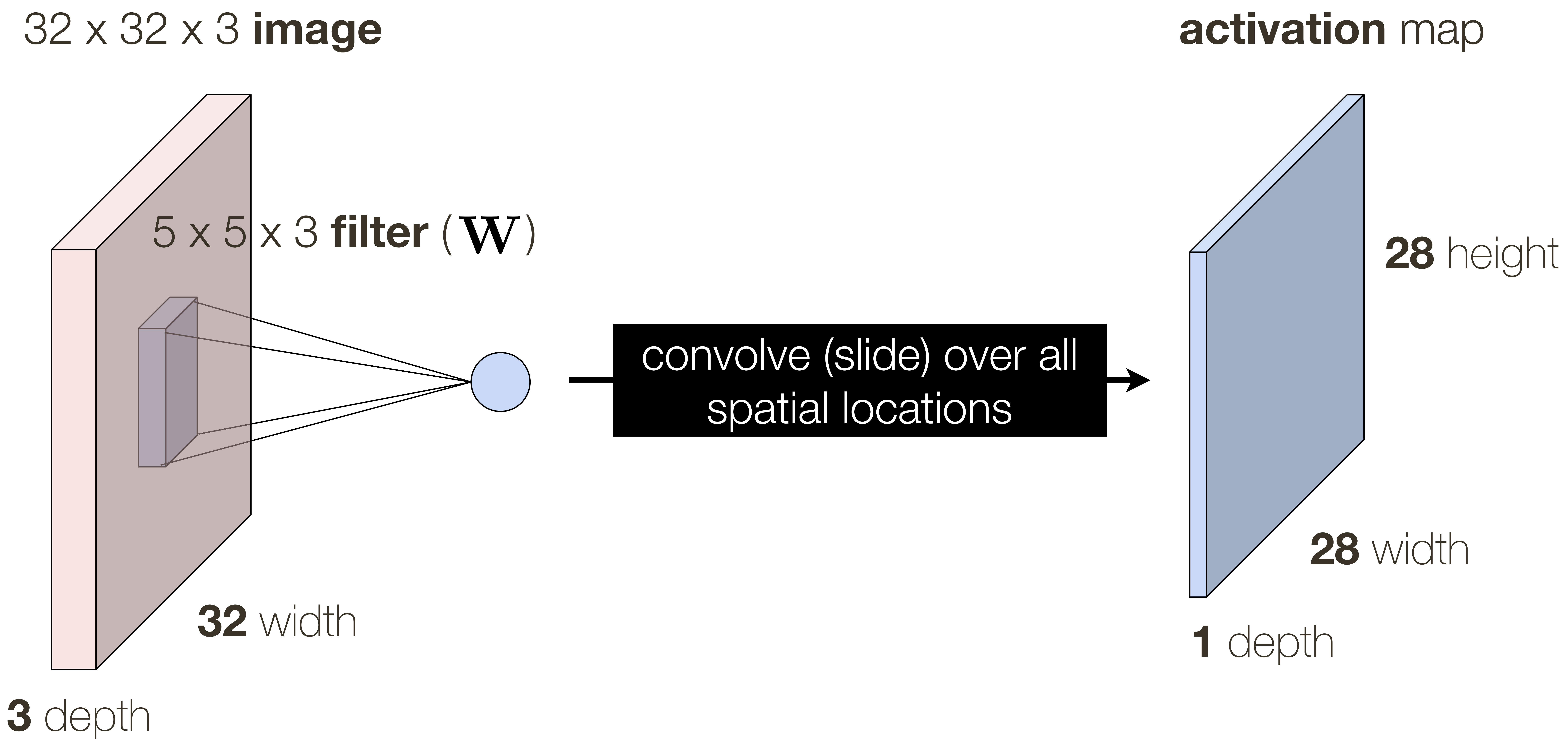
[Zeiler and Fergus, 2013]

What **filters** do networks learn?



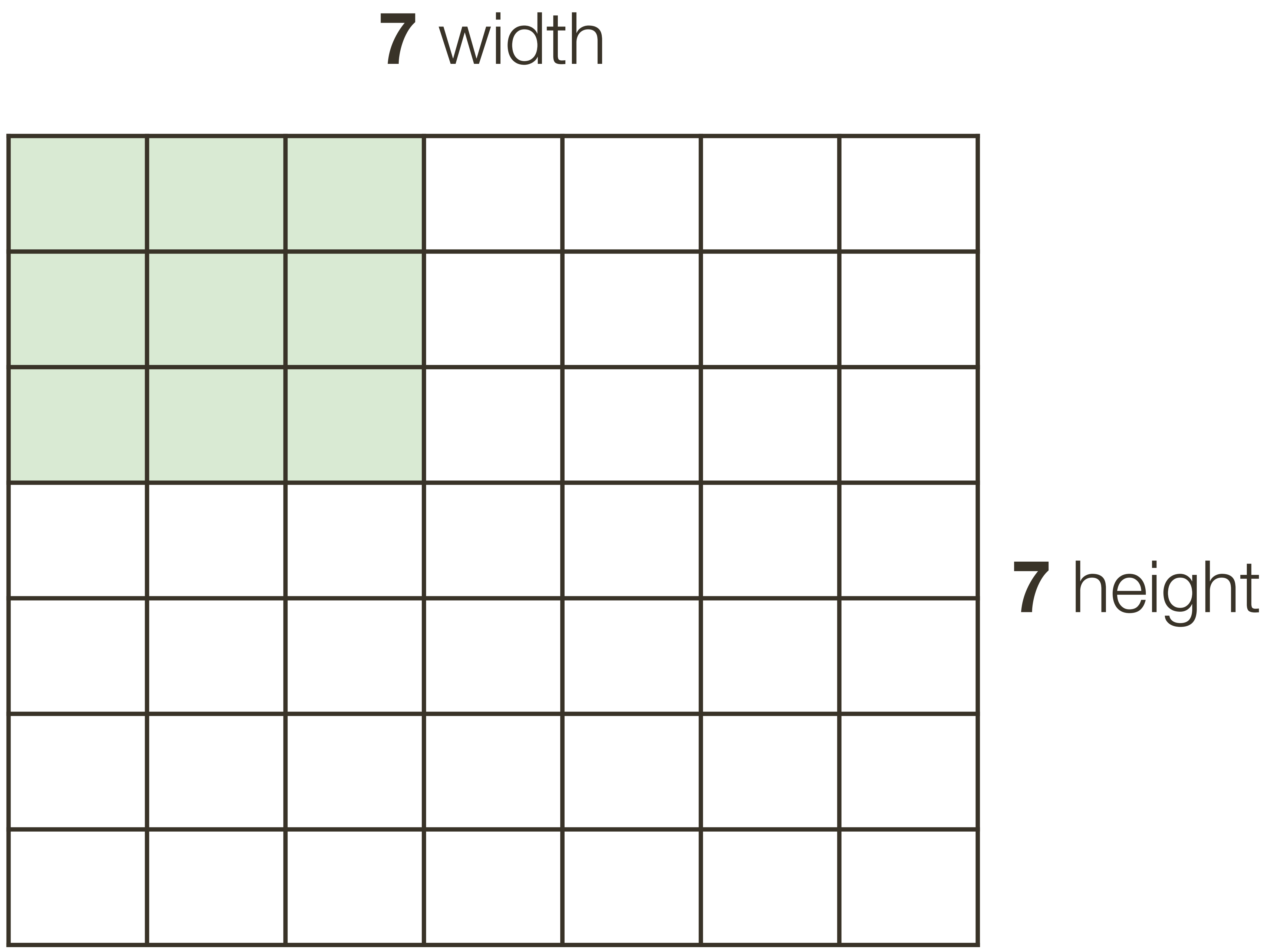
[Zeiler and Fergus, 2013]

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



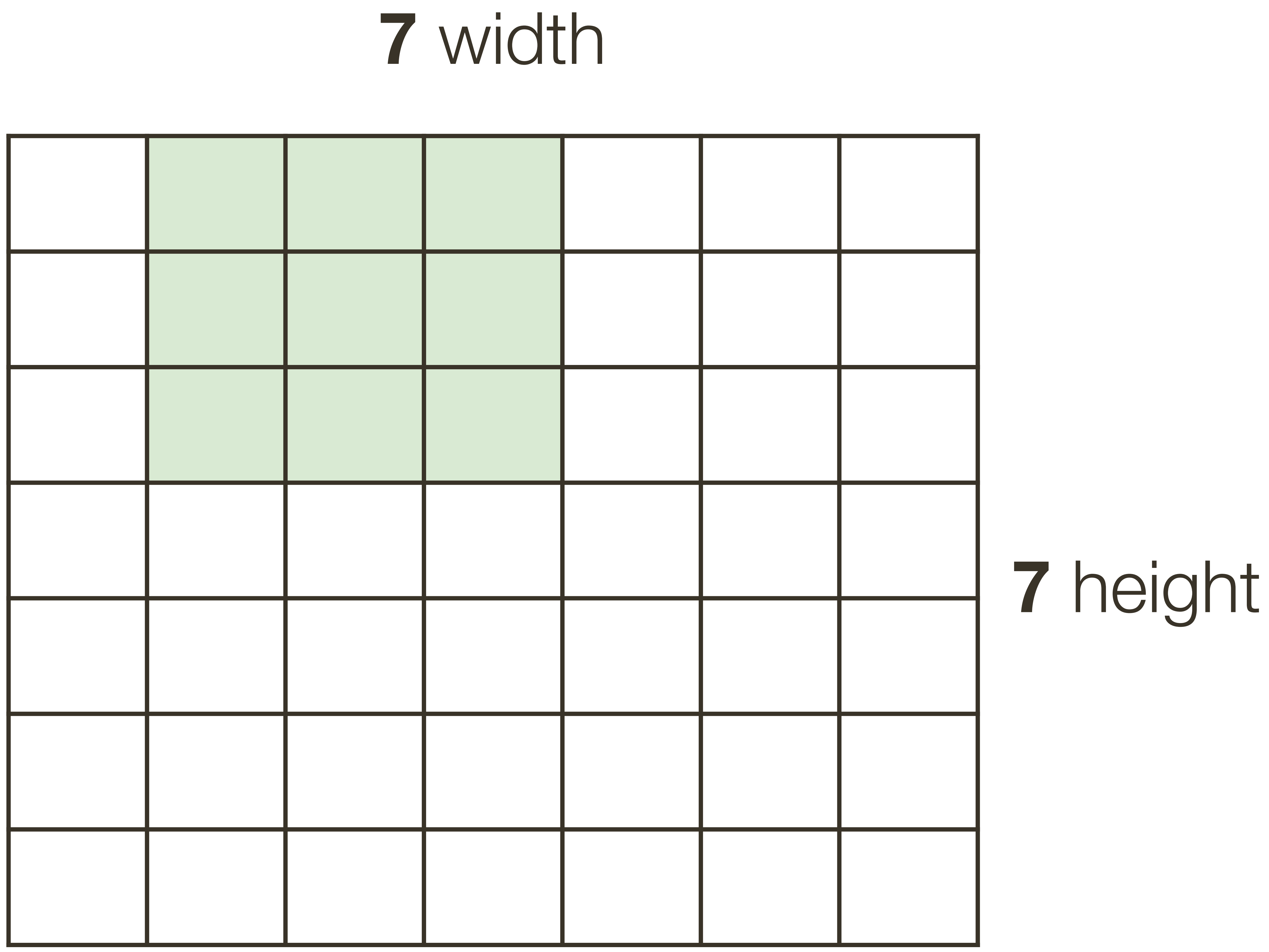
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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



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

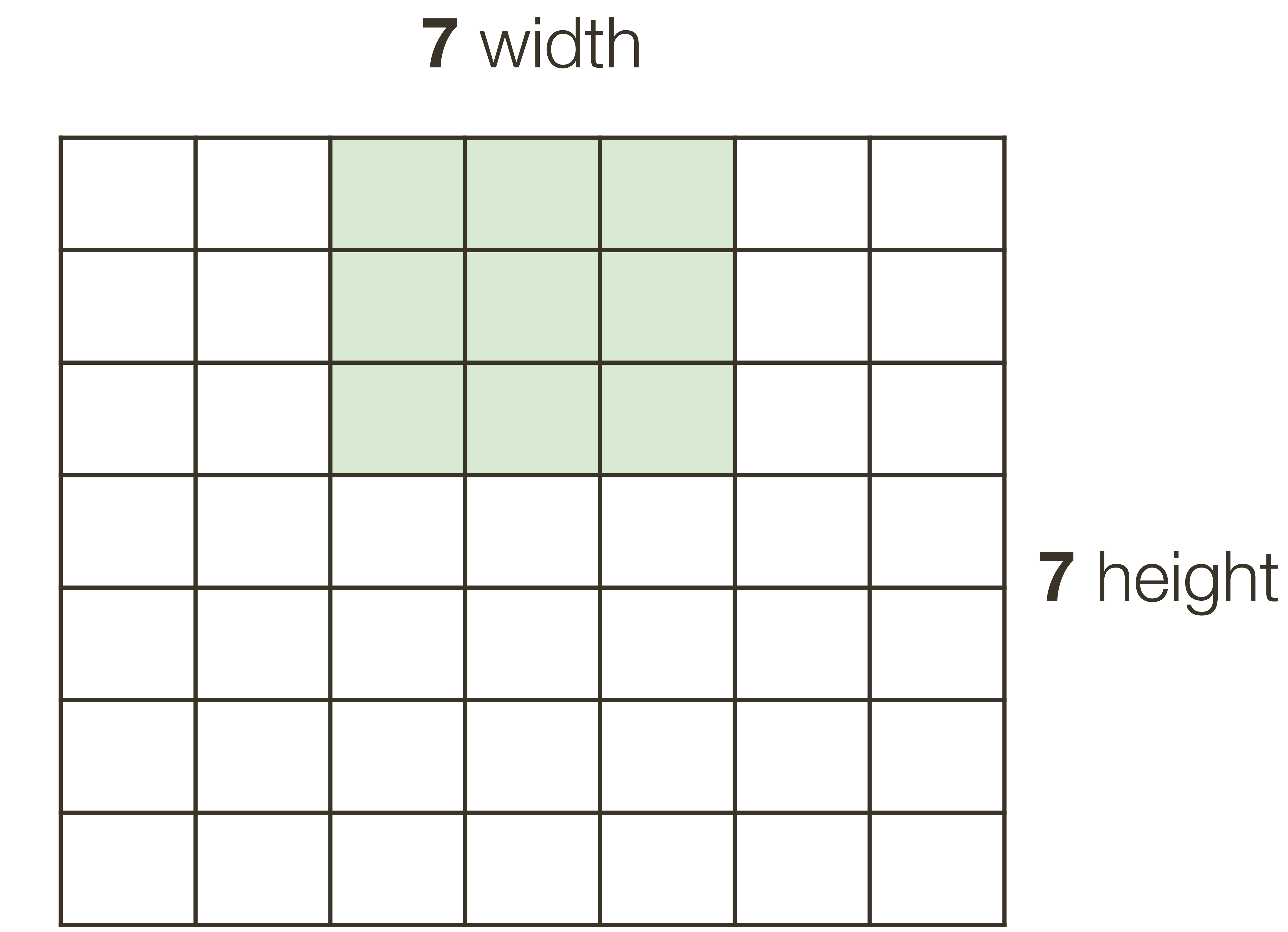
Convolutional Layer: Closer Look at **Spatial Dimensions**



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

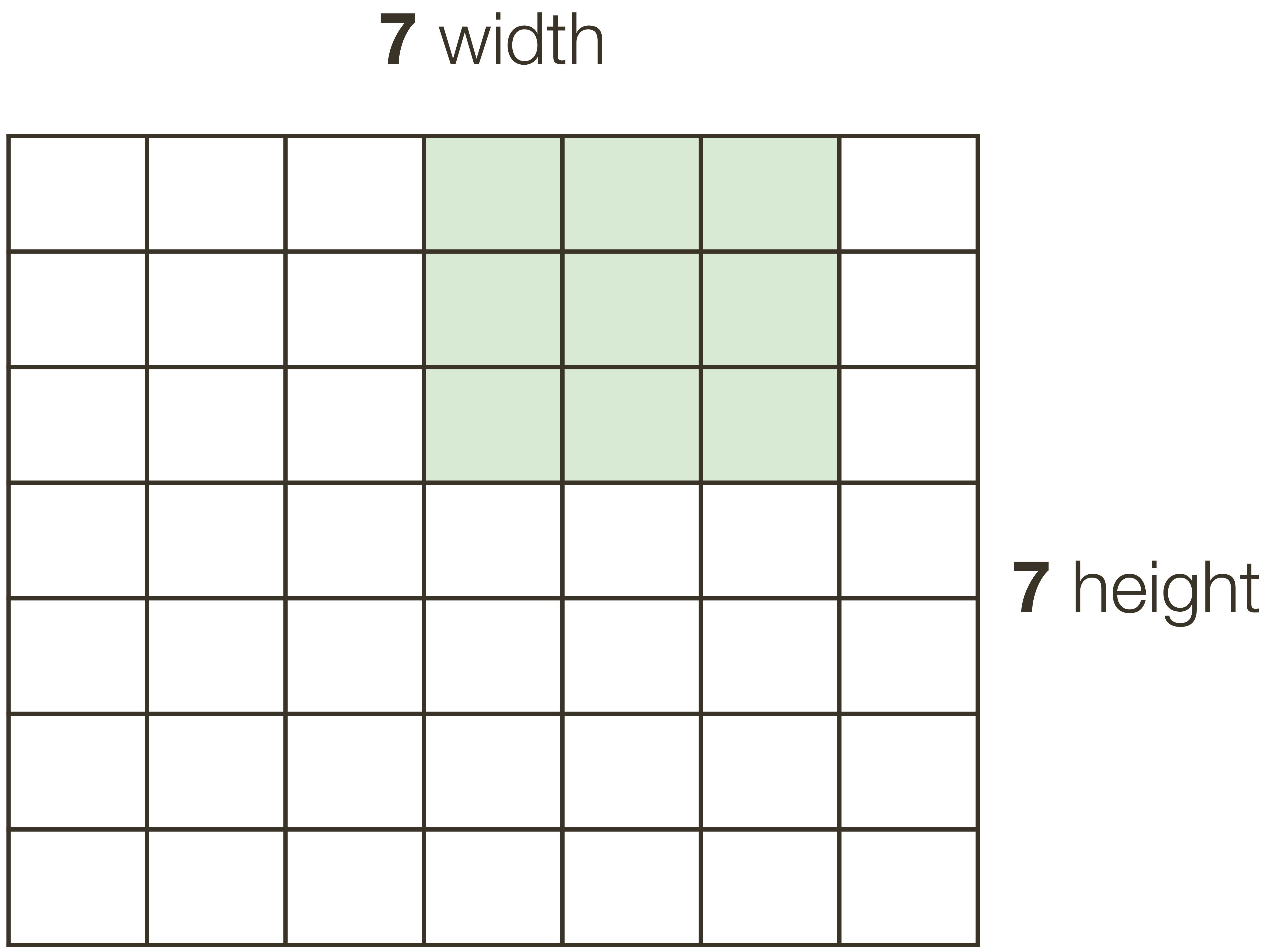
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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



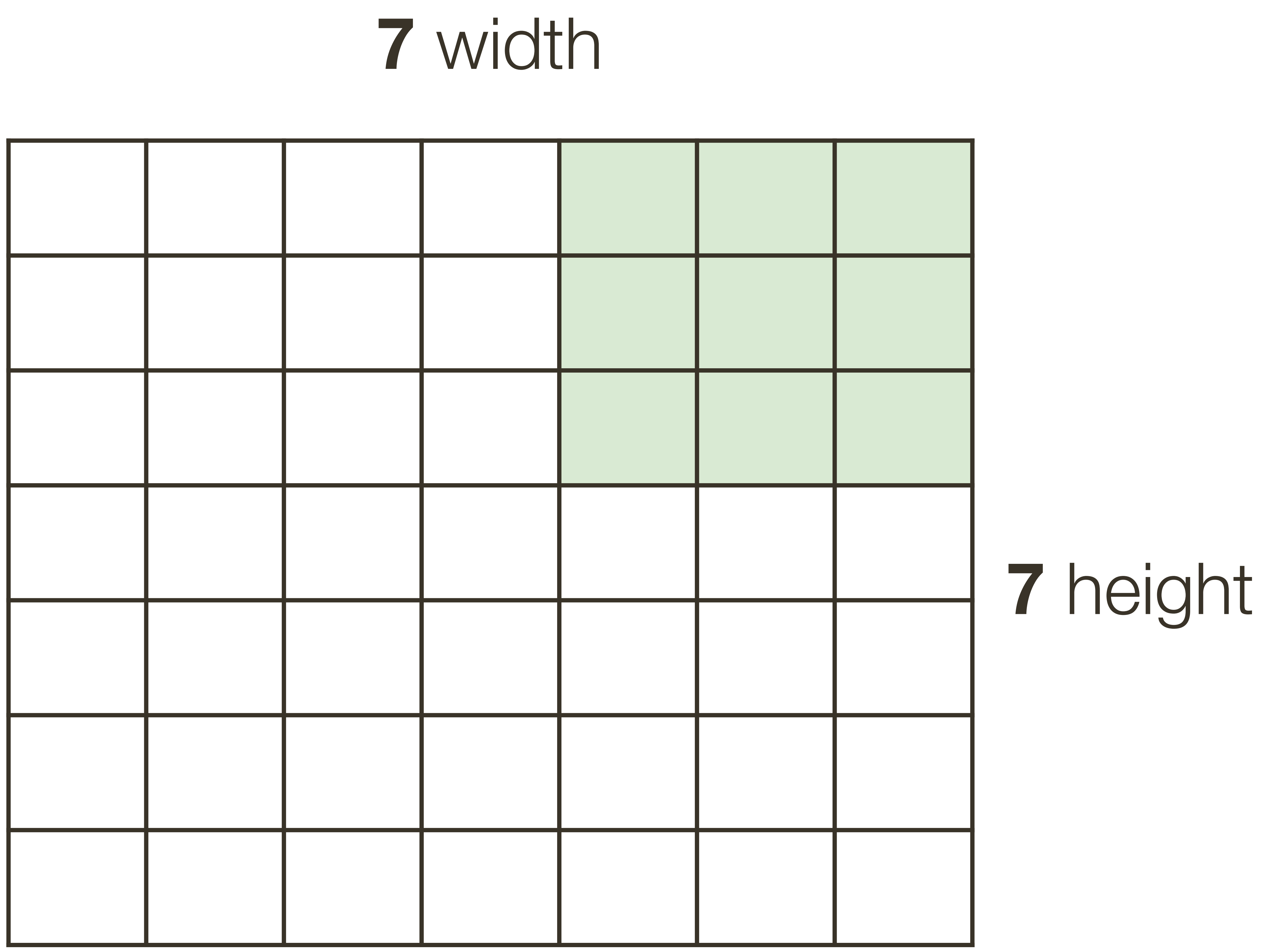
7 x 7 input image (spatially)
3 x 3 filter

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



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

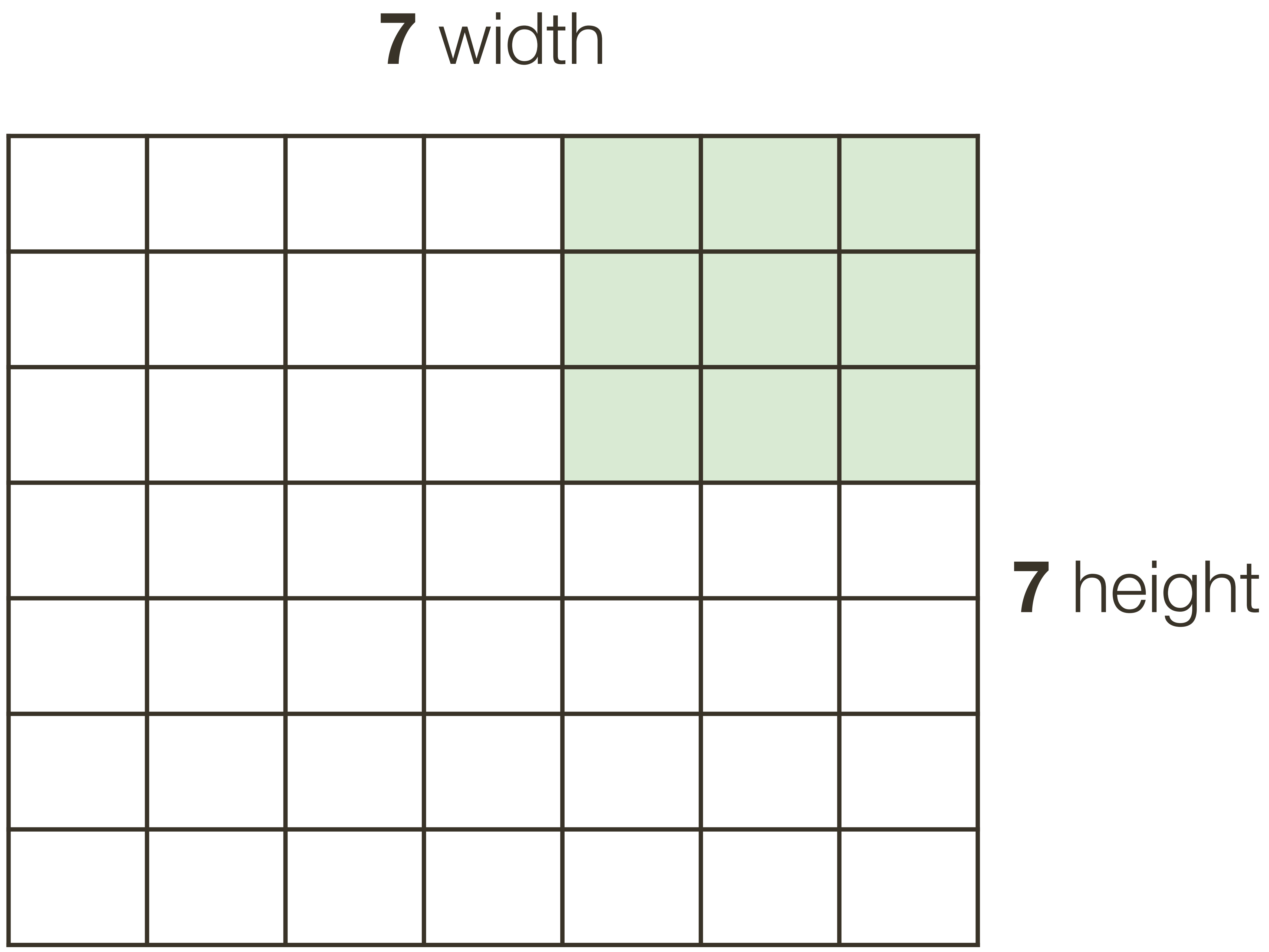
Convolutional Layer: Closer Look at **Spatial Dimensions**



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

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

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

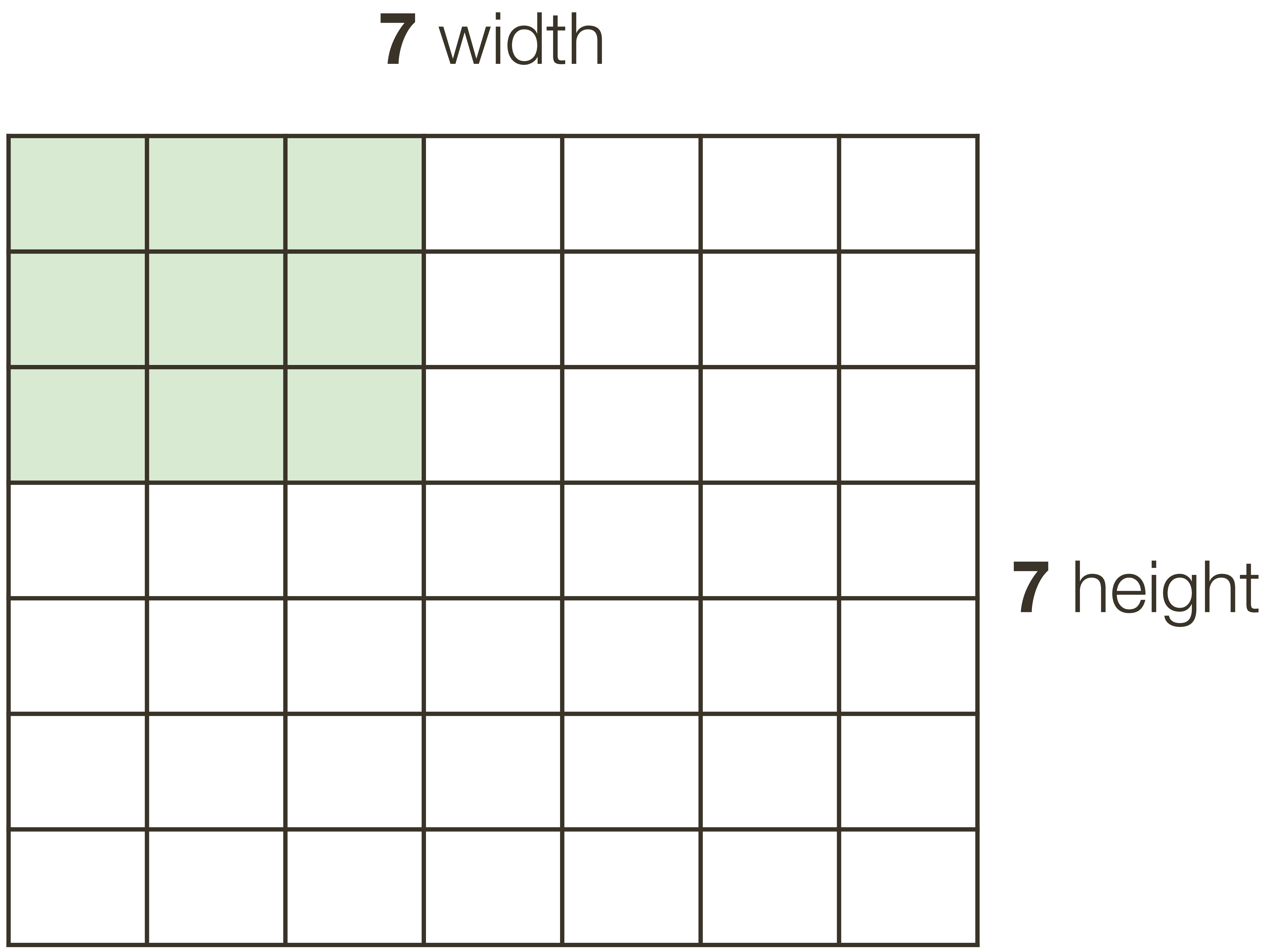


7 x 7 input image (spatially)
3 x 3 filter

=> **5 x 5 output**

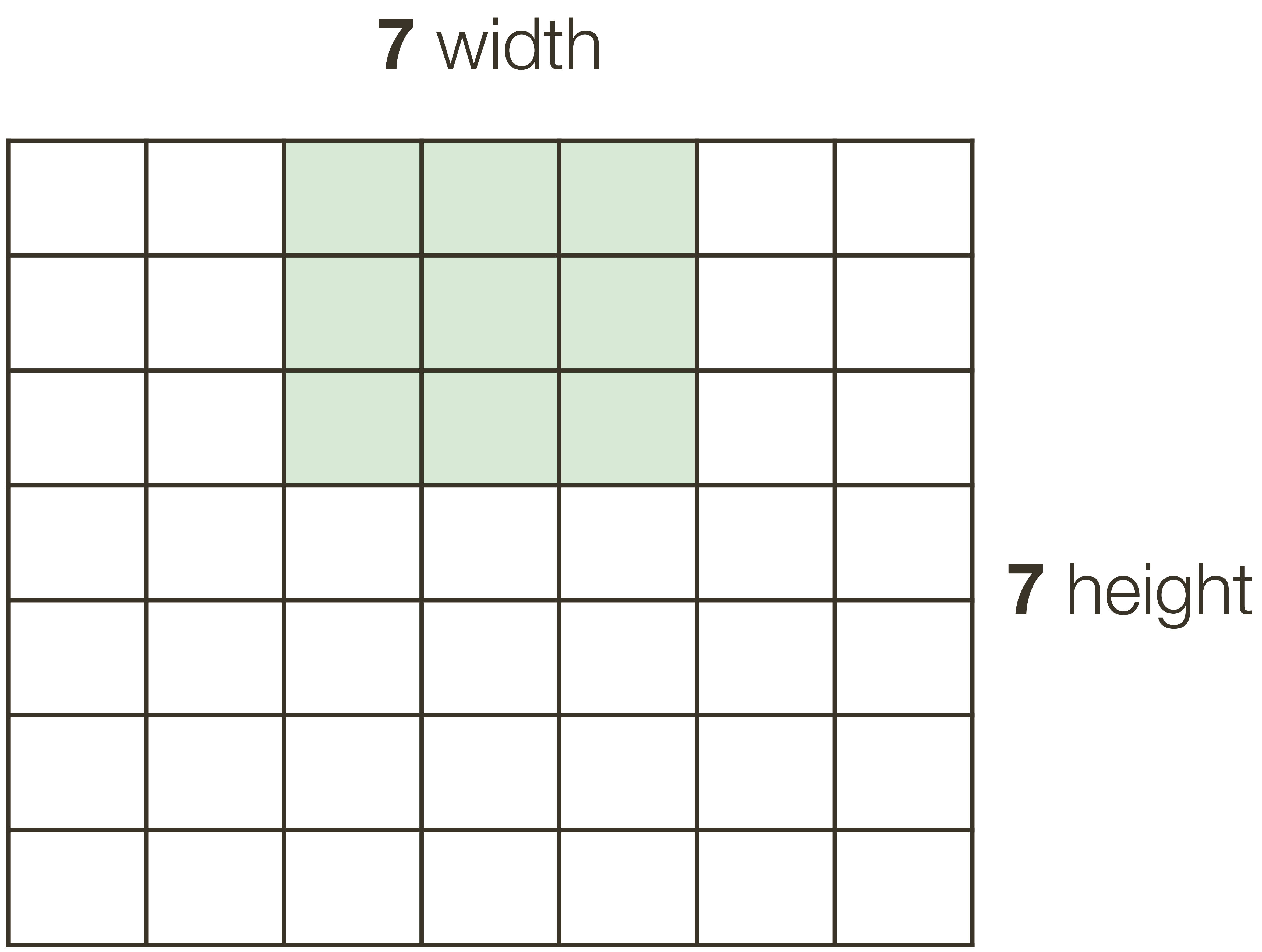
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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



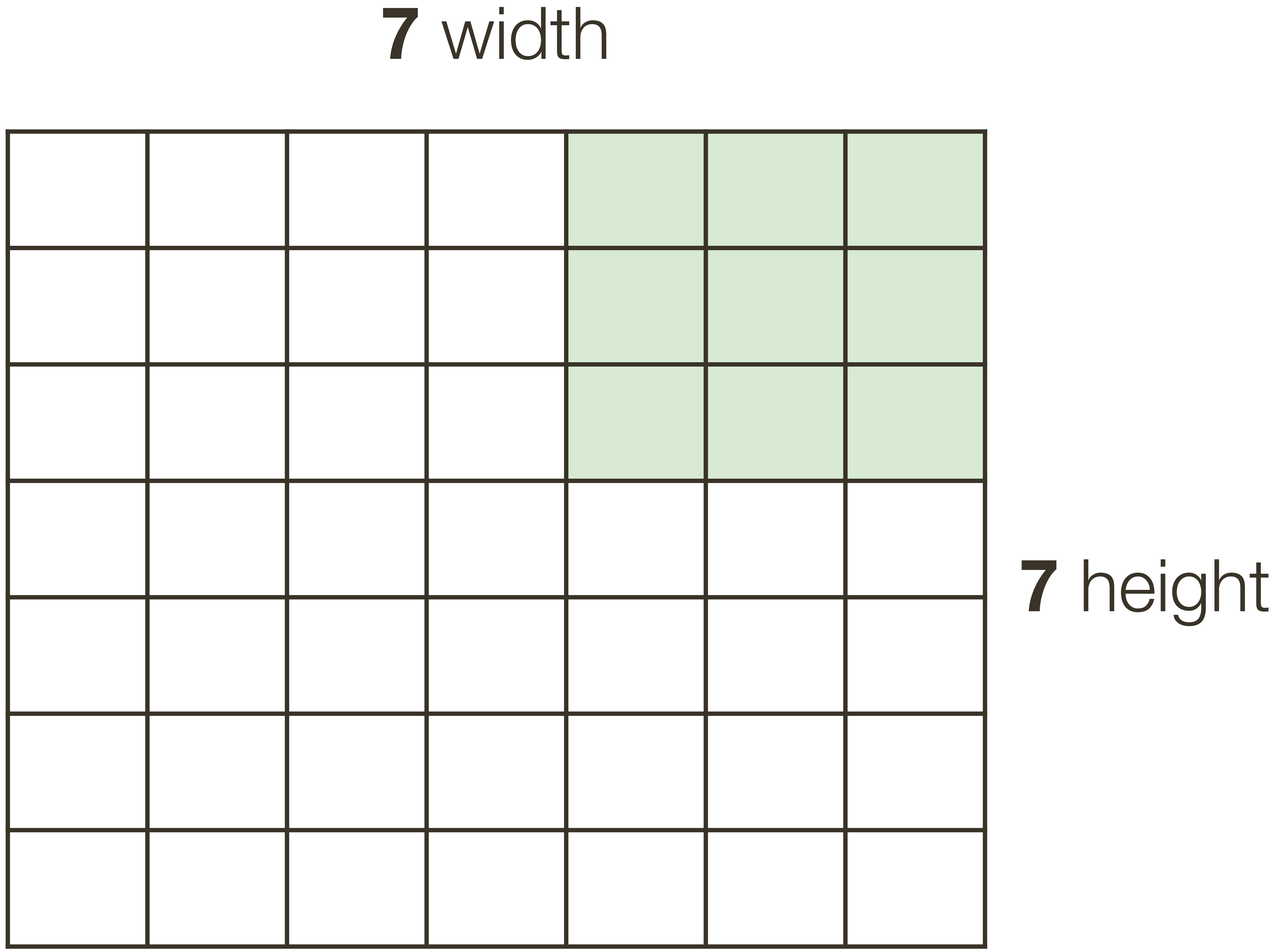
7 x 7 input image (spatially)
3 x 3 filter
(applied with **stride 2**)

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



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

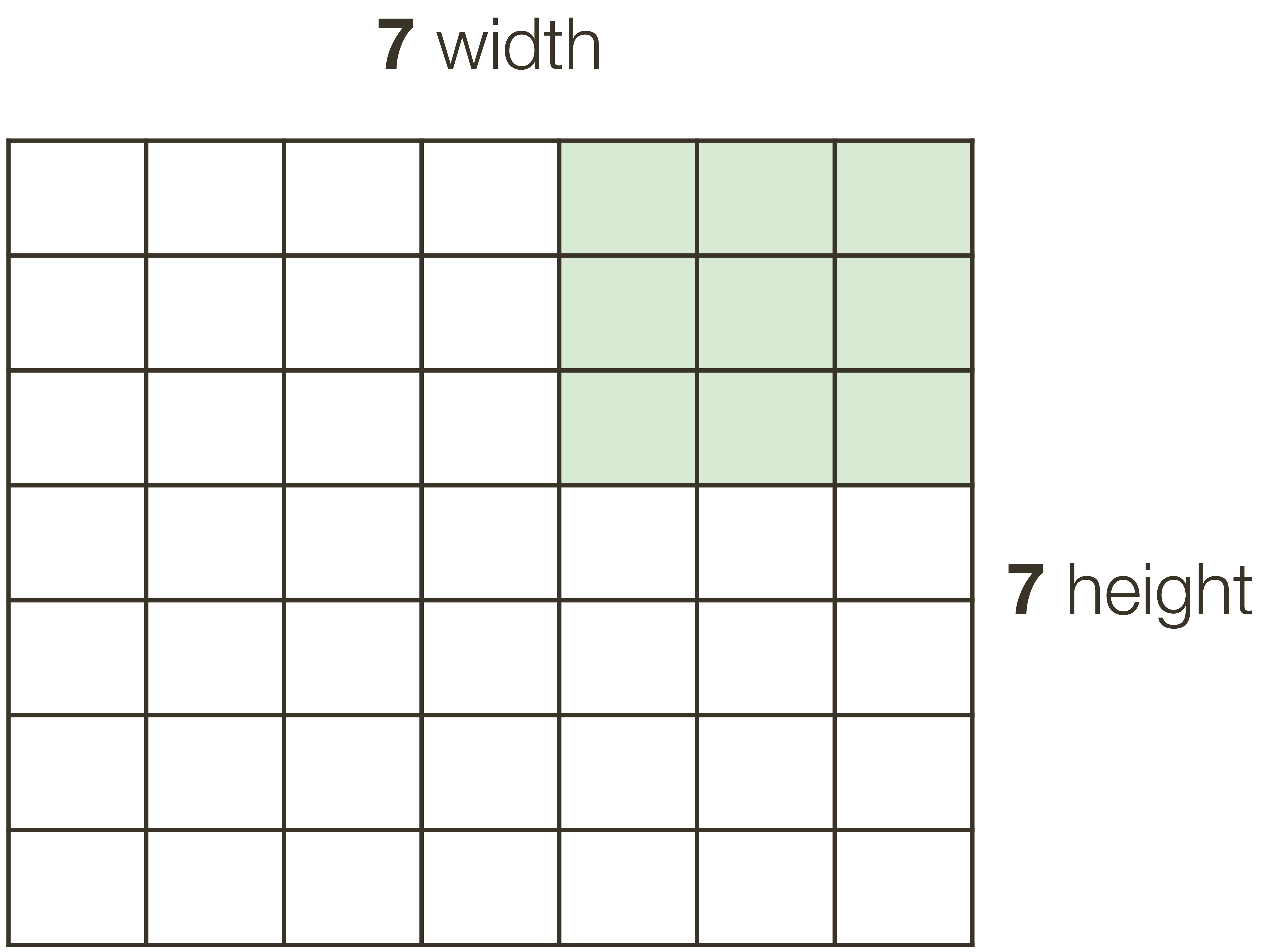
Convolutional Layer: Closer Look at **Spatial Dimensions**



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

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

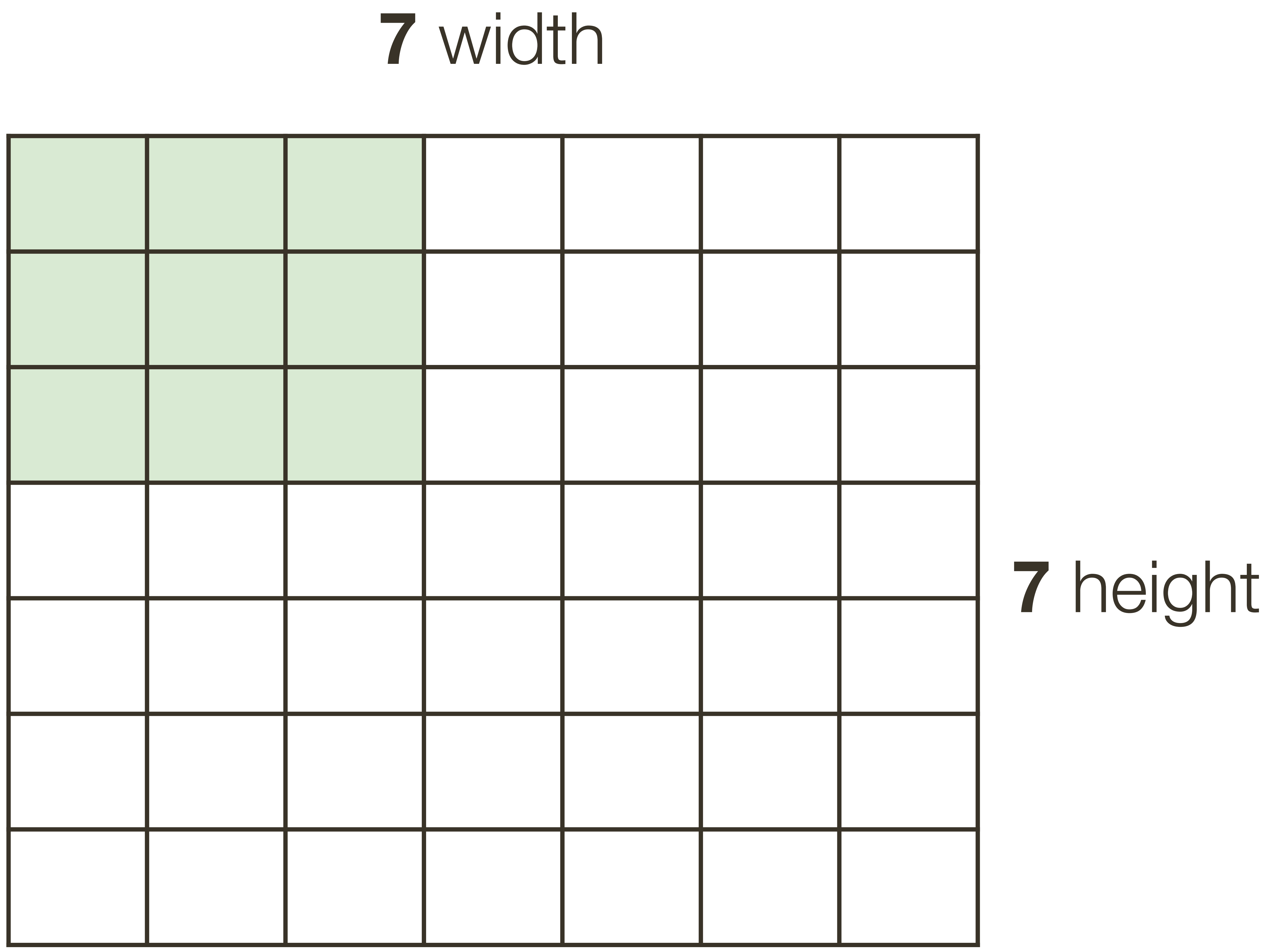
Convolutional Layer: Closer Look at **Spatial Dimensions**



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

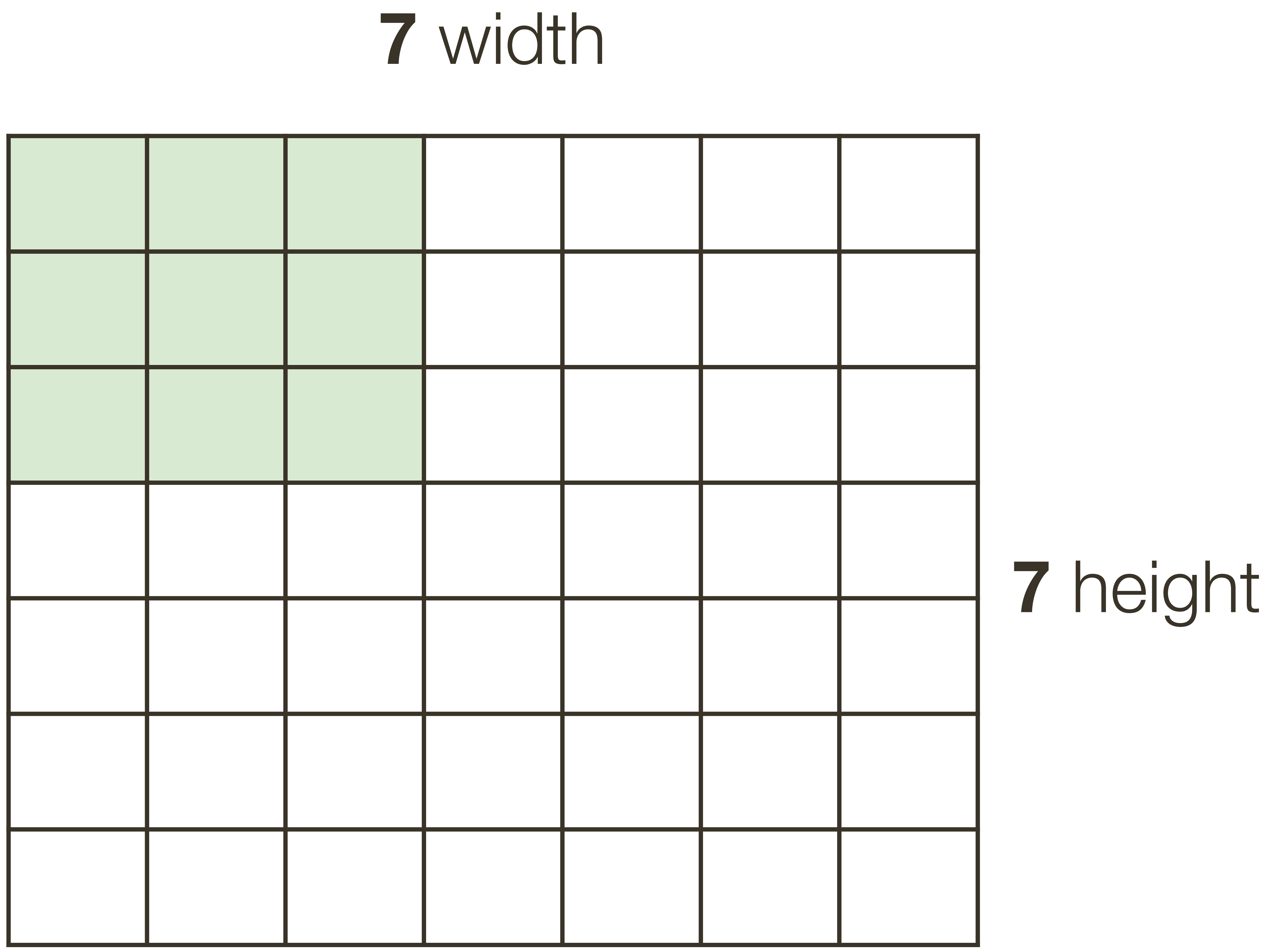
=> **3 x 3 output**

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



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

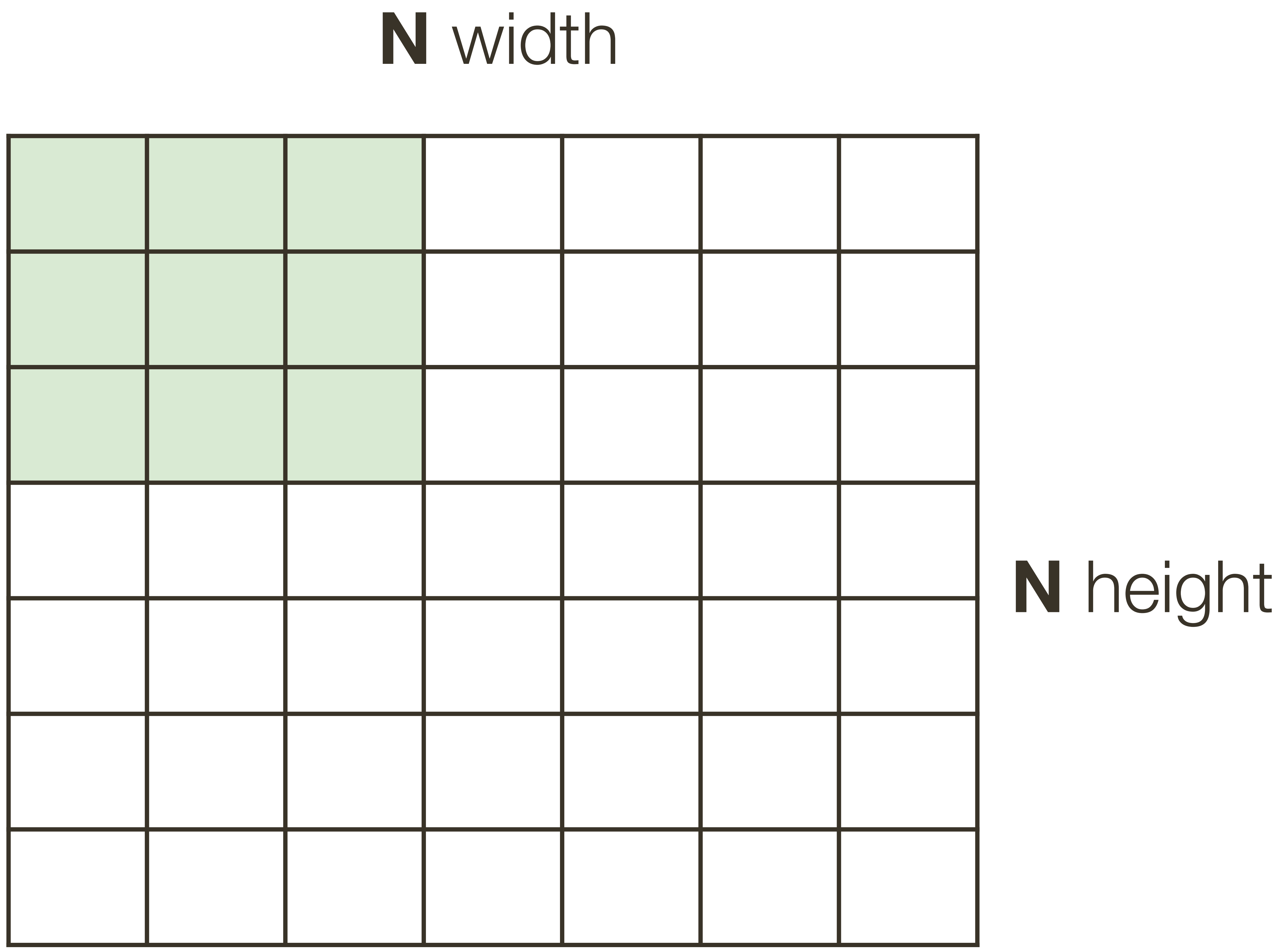
Convolutional Layer: Closer Look at **Spatial Dimensions**



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**

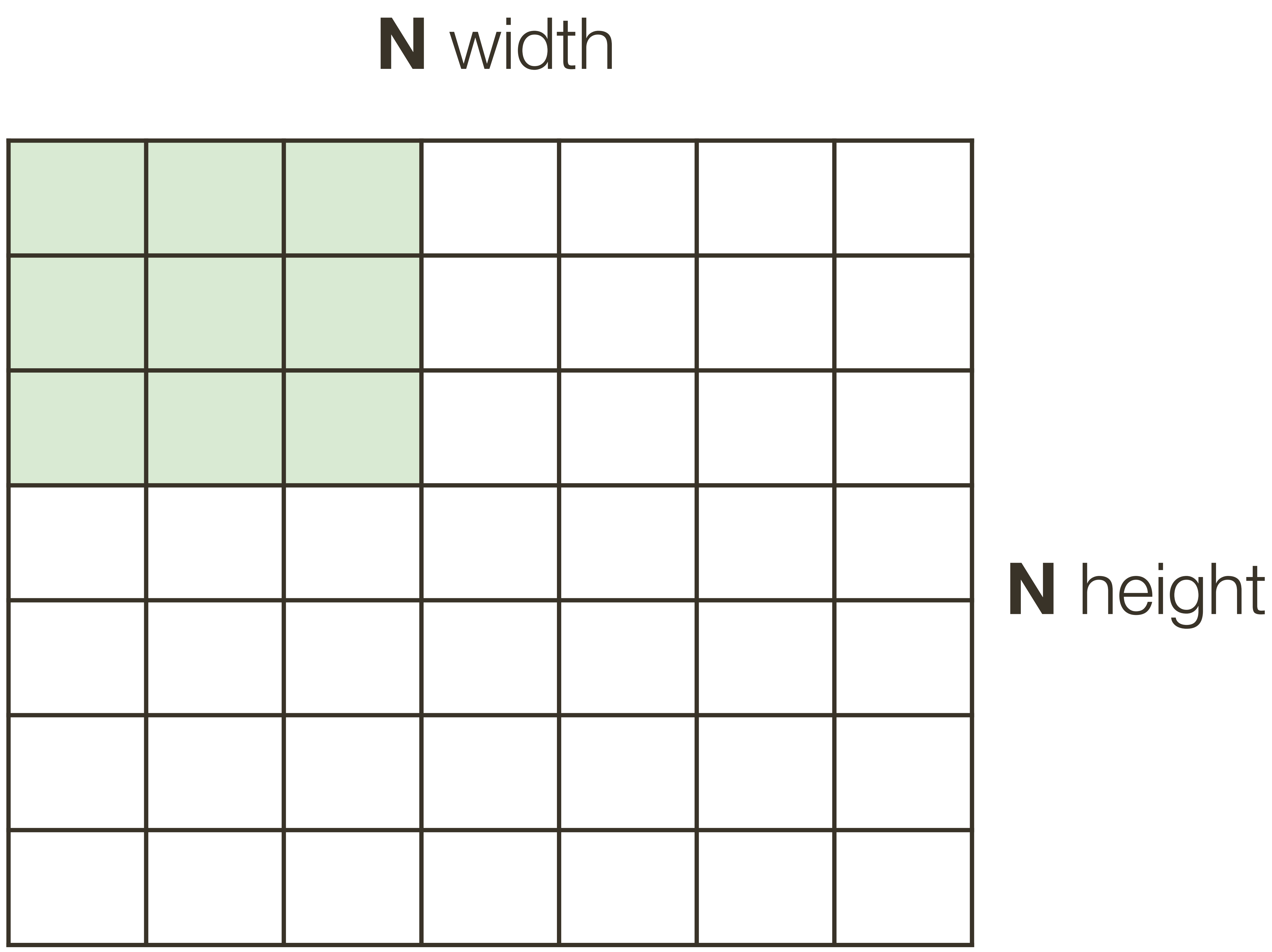


N x N input image (spatially)
F x F filter

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

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

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



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

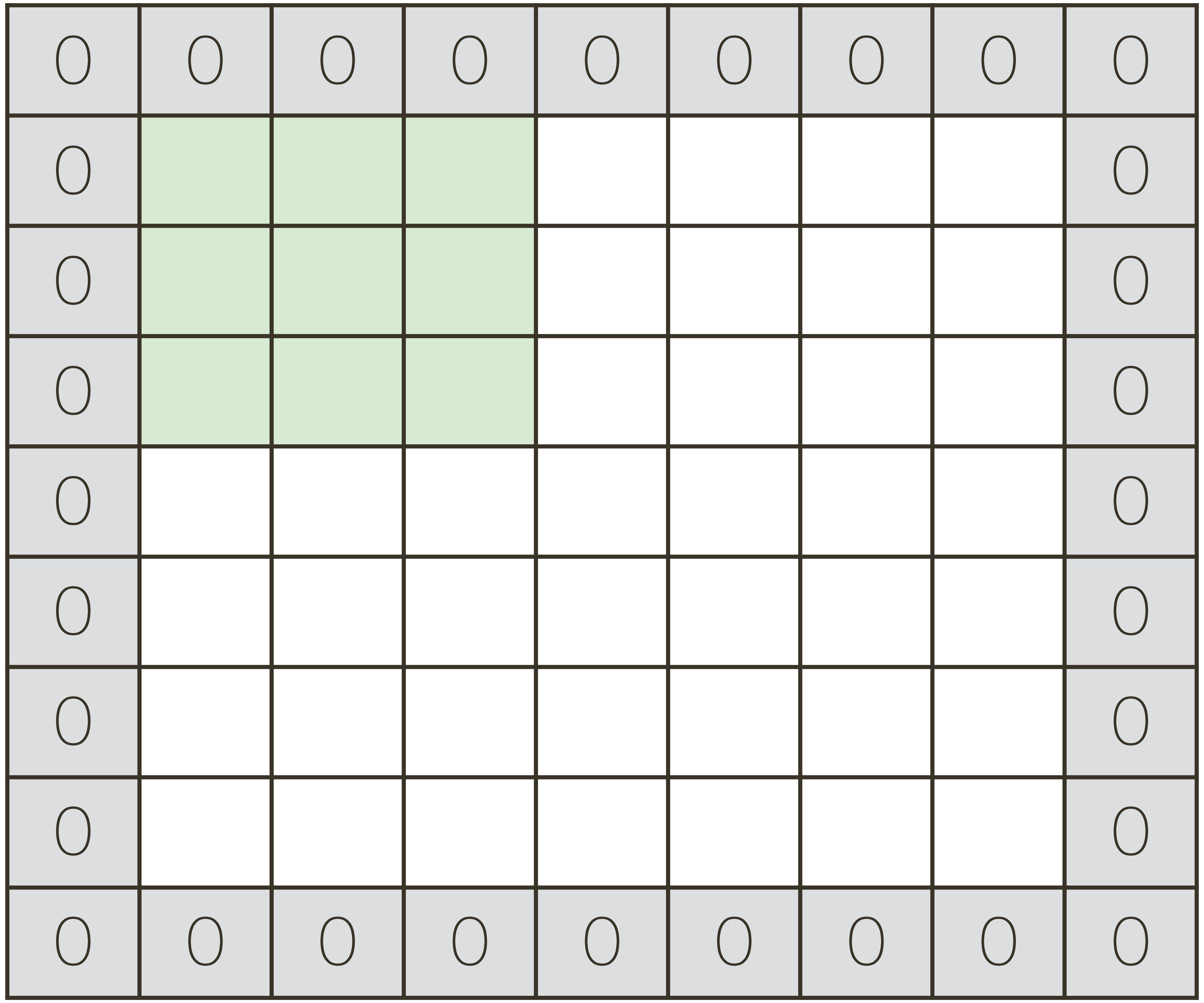
Output size: (N-F) / stride + 1

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

5 x 5 input image (spatially)
3 x 3 filter
(applied with **stride 1**)

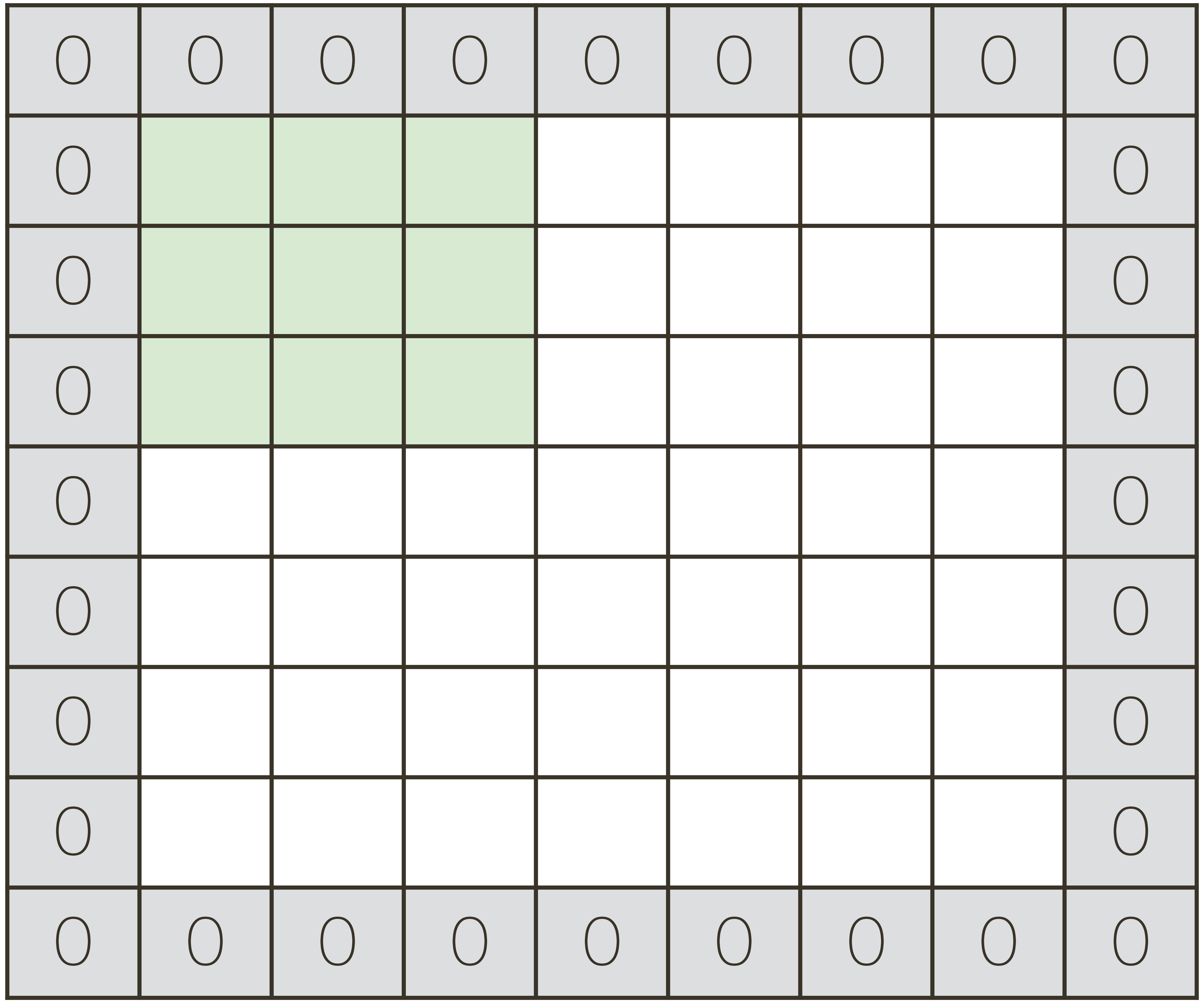
pad with 1 pixel border

Output size: 7 x 7

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

Convolutional Layer: **Border padding**

7 width



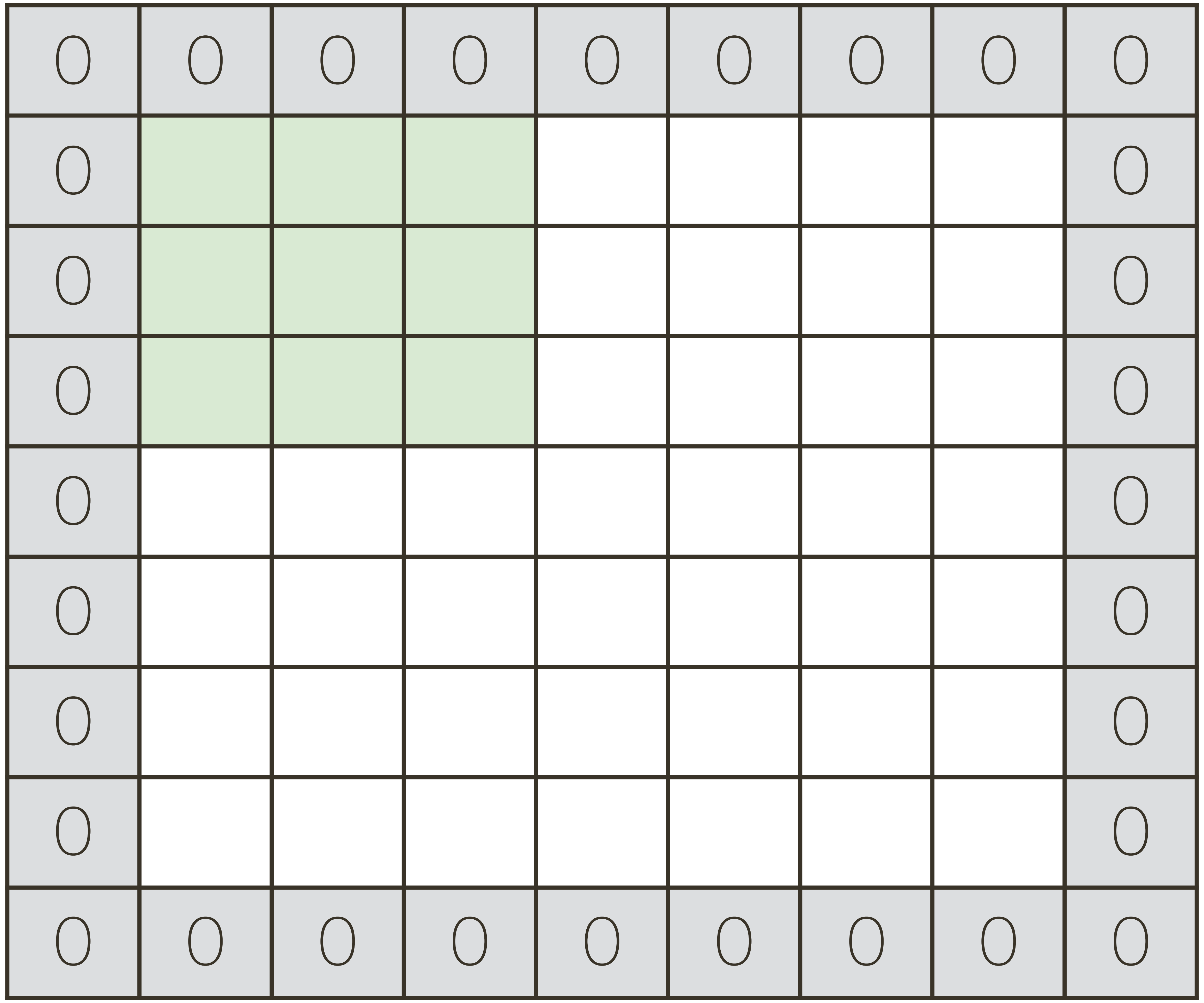
7 height

5 x 5 input image (spatially)
3 x 3 filter
(applied with **stride 3**)

pad with 1 pixel border

Convolutional Layer: **Border padding**

7 width



7 height

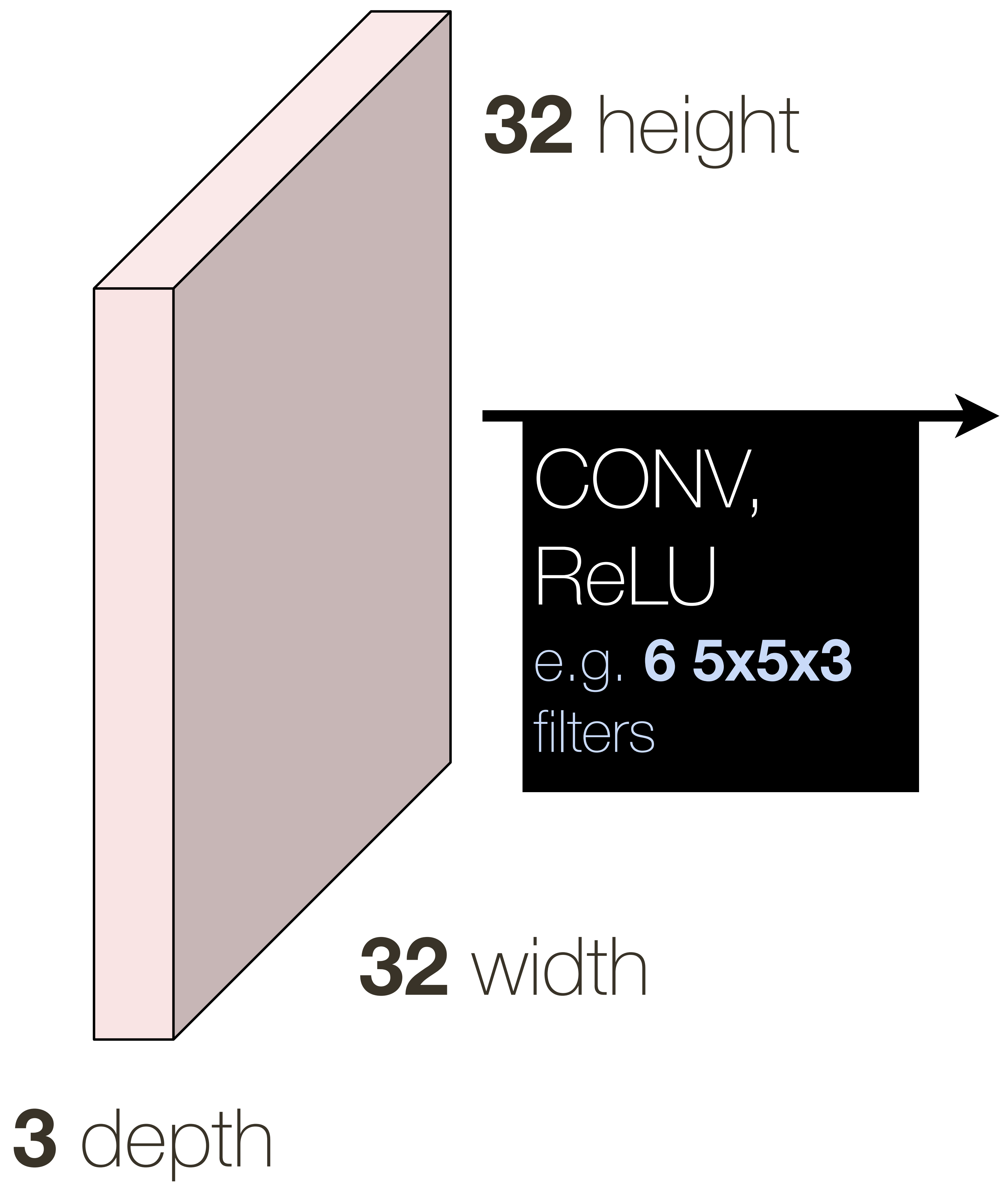
5 x 5 input image (spatially)
3 x 3 filter
(applied with **stride 3**)

pad with 1 pixel border

Example: N = 7, F = 3

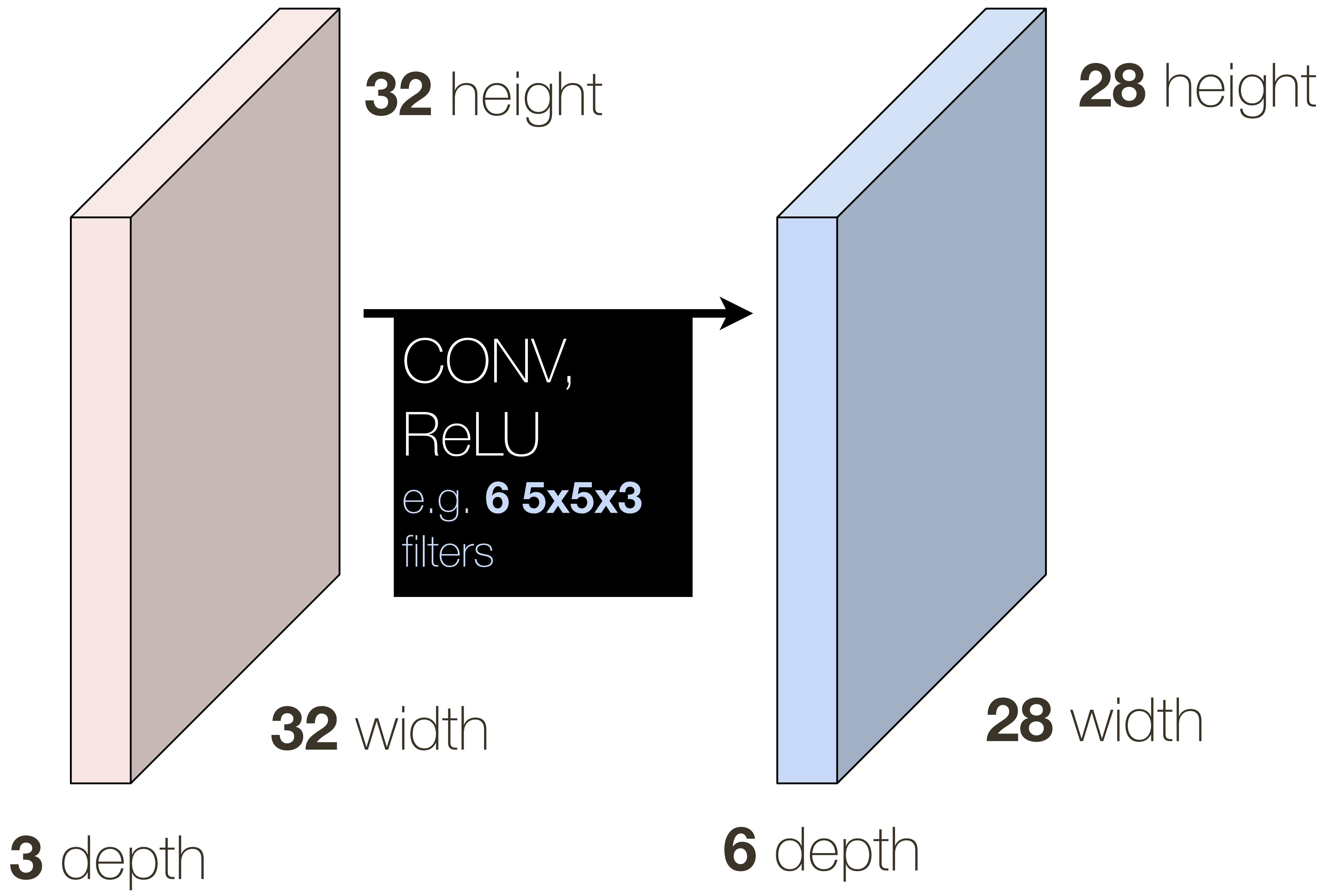
stride 1 => (9-3)/1+1 = 7
stride 2 => (9-3)/2+1 = 4
stride 3 => (9-3)/3+1 = **3**

Convolutional Neural Network (ConvNet)



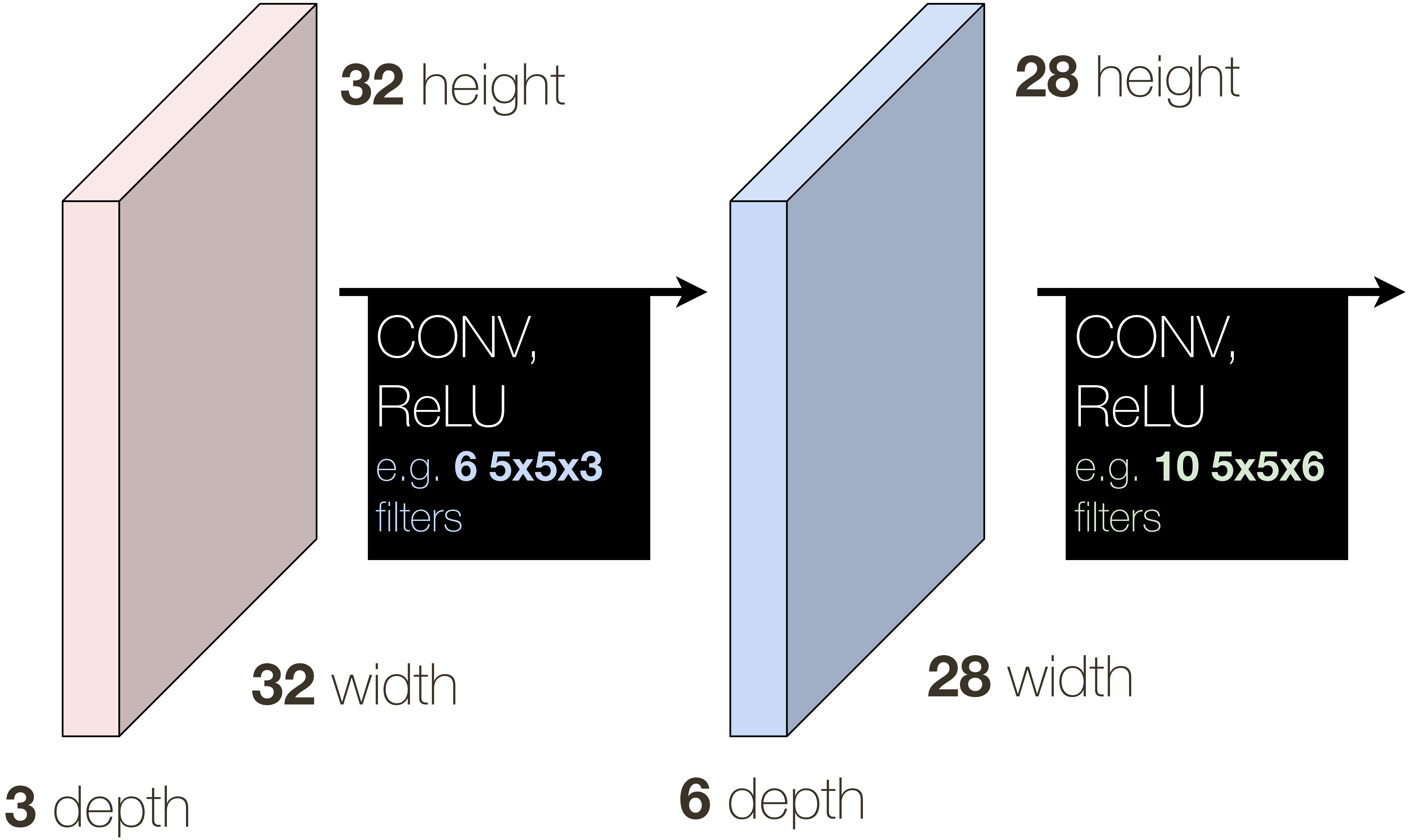
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



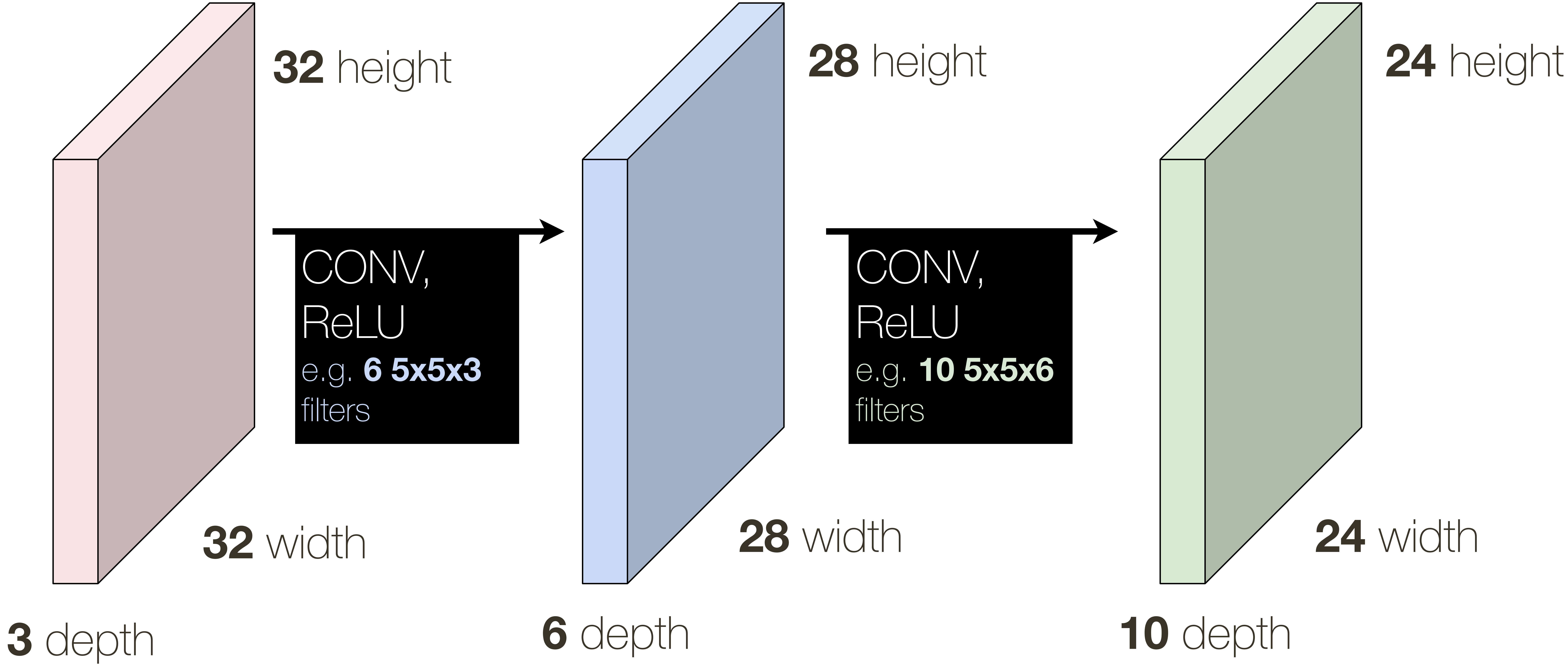
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



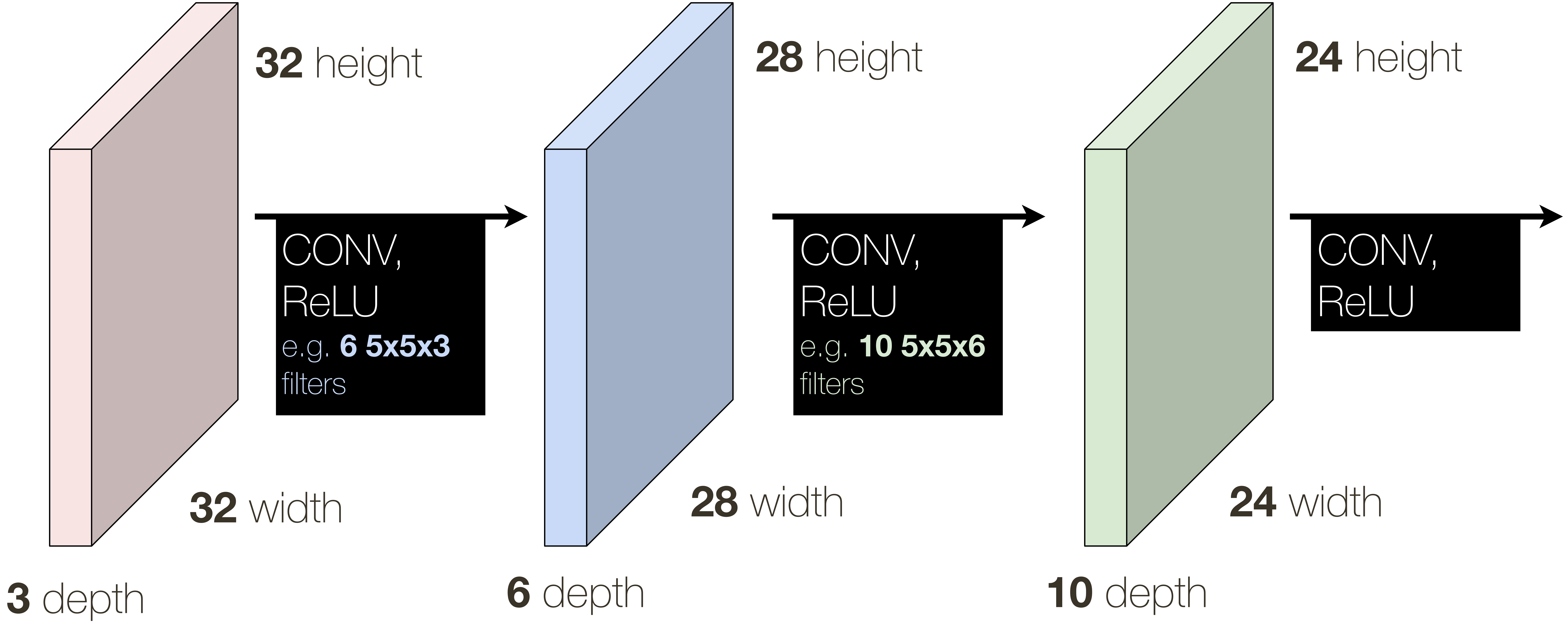
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



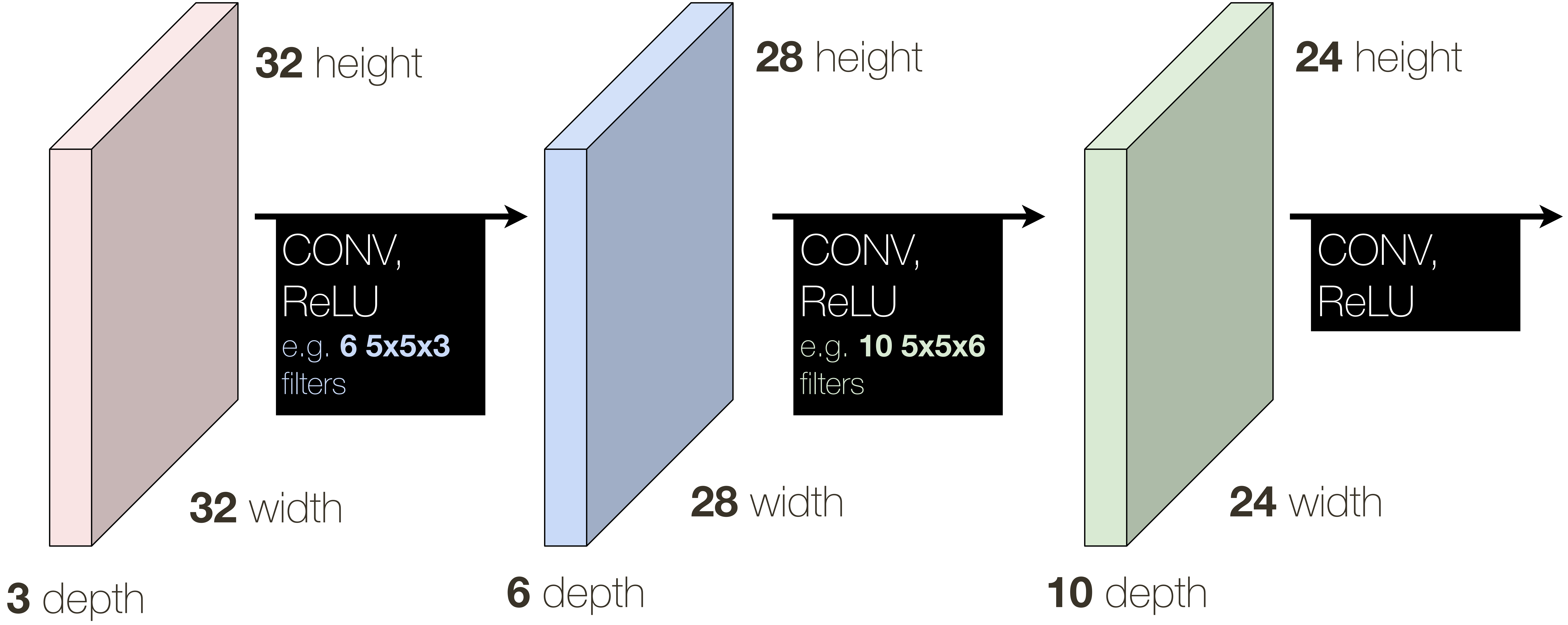
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Network (ConvNet)



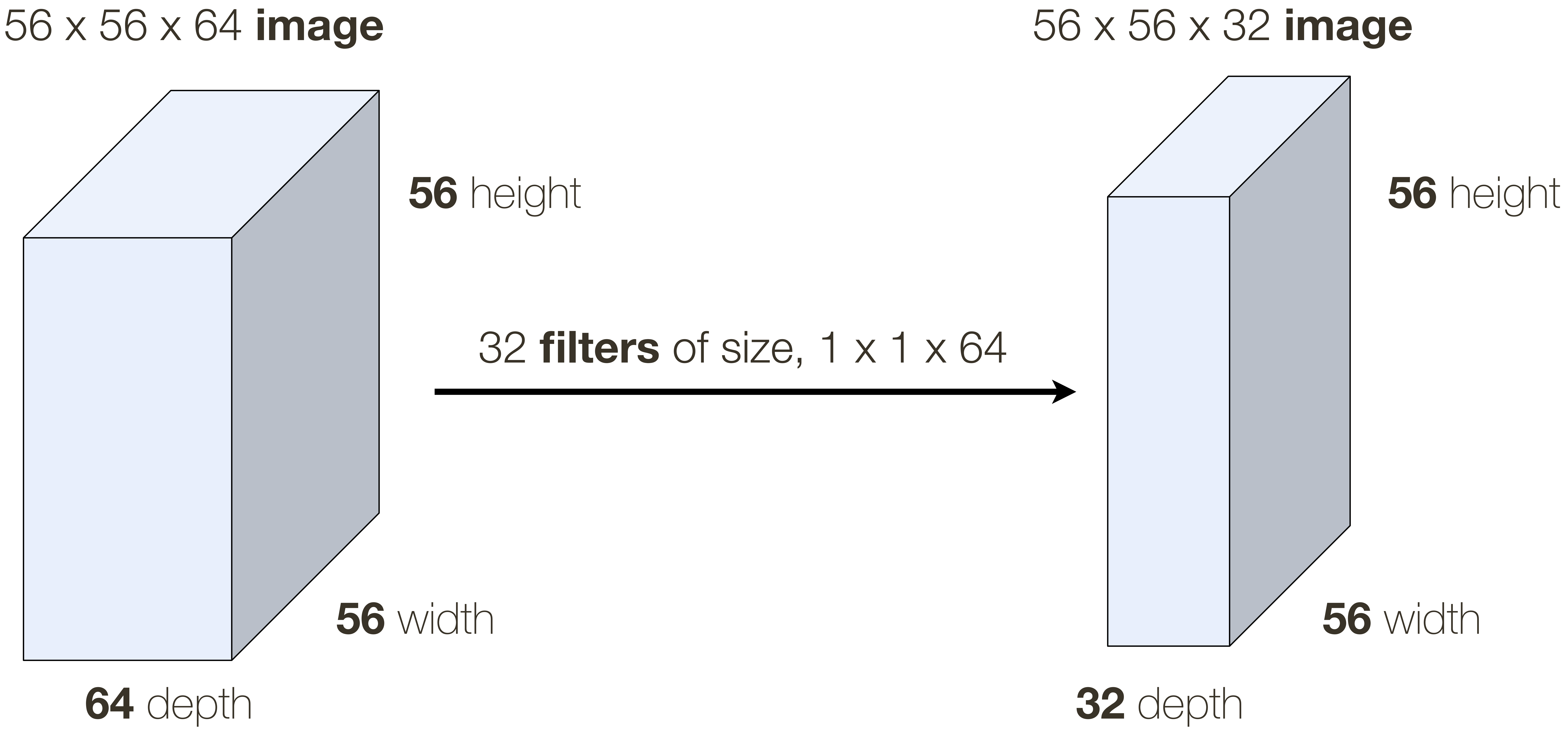
Convolutional Neural Network (ConvNet)

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



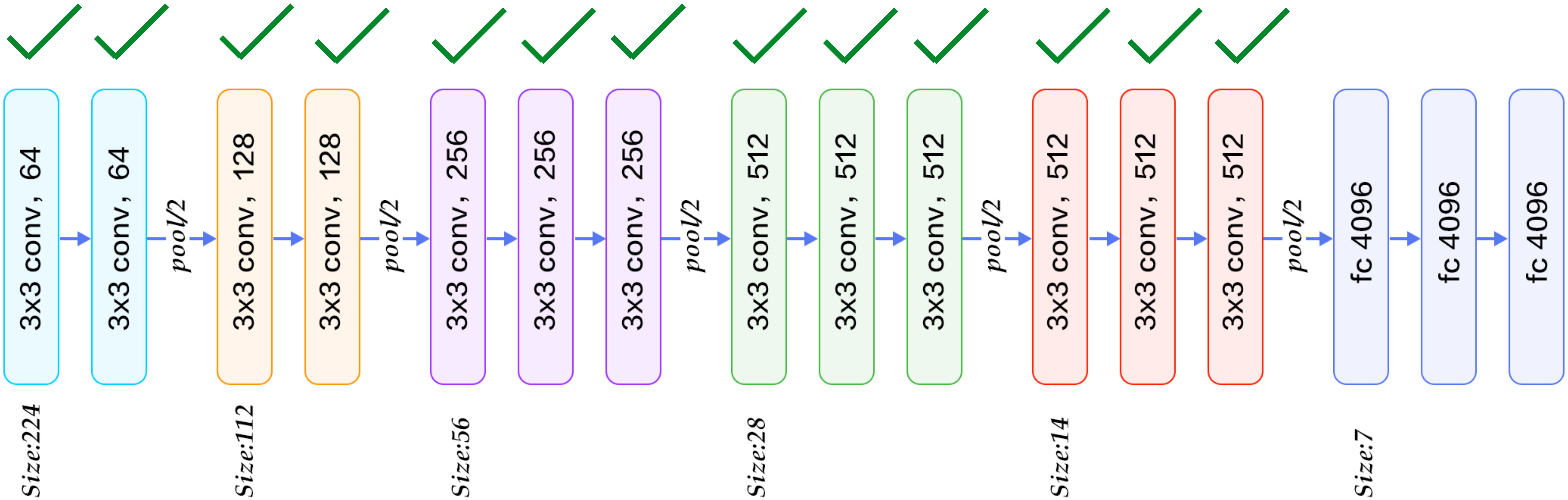
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Layer: **1x1 convolutions**



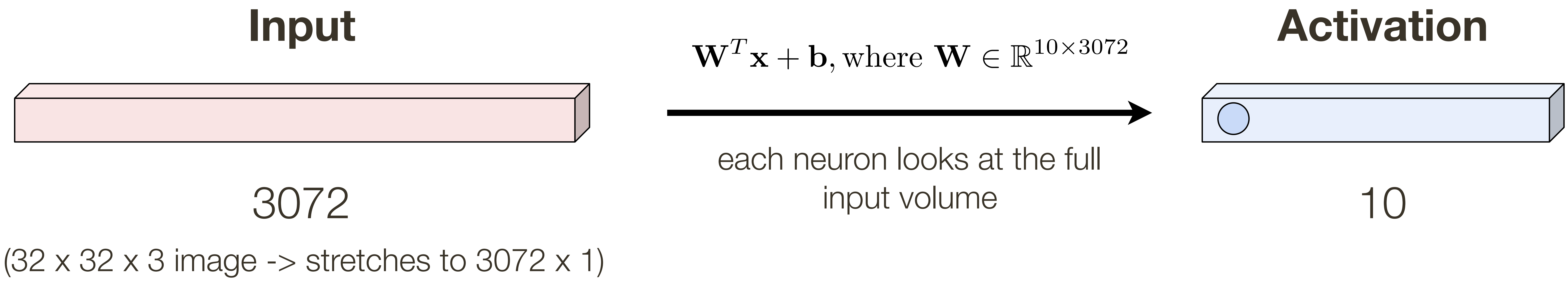
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Networks

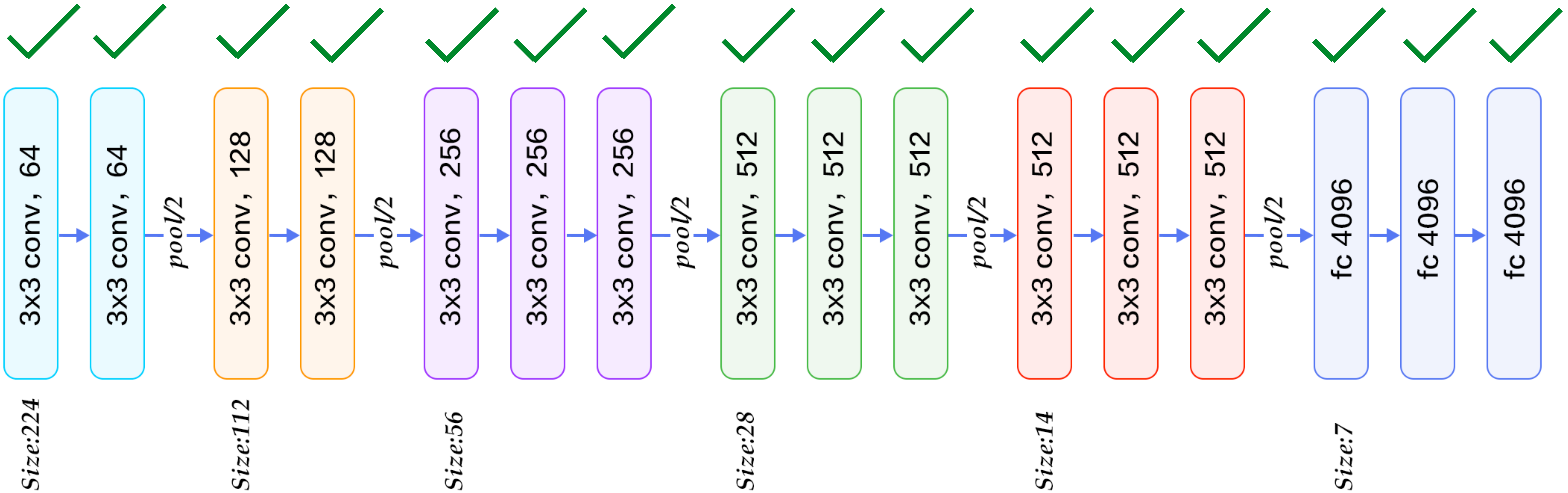


VGG-16 Network

CNNs: Reminder Fully Connected Layers

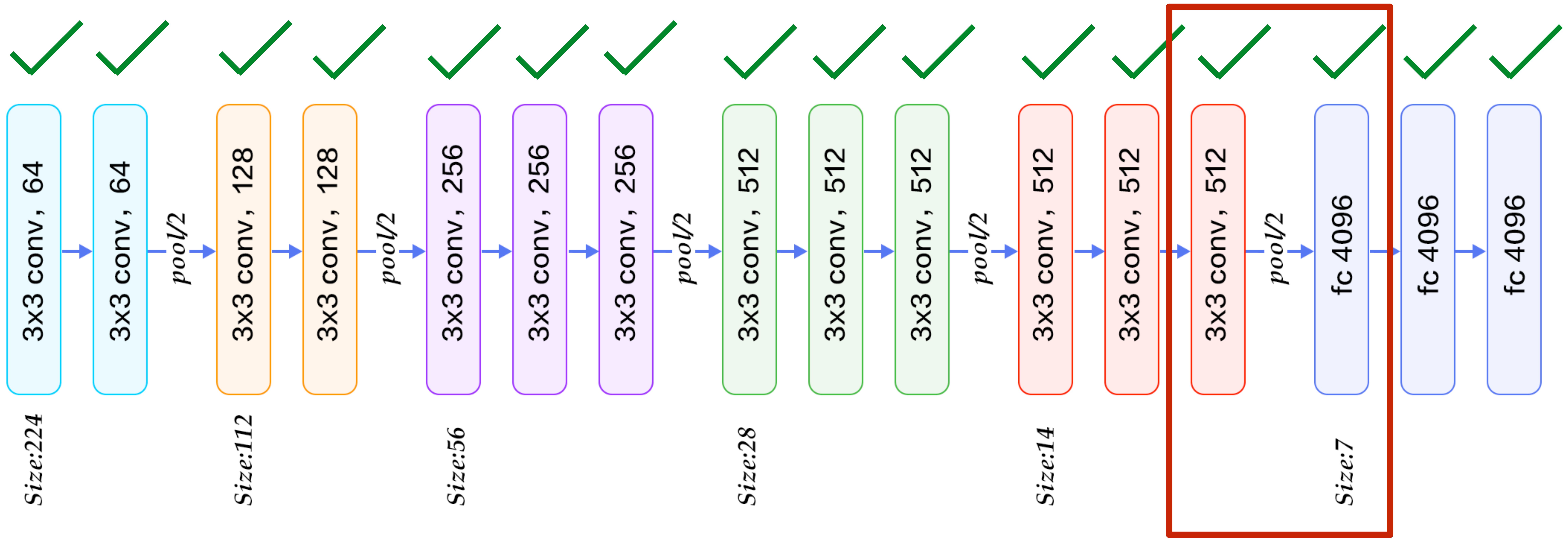


Convolutional Neural Networks



VGG-16 Network

Convolutional Neural Networks



VGG-16 Network

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

CNNs: Reminder Fully Connected Layers

Input

Activation

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

each neuron looks at the full
input volume

25,088

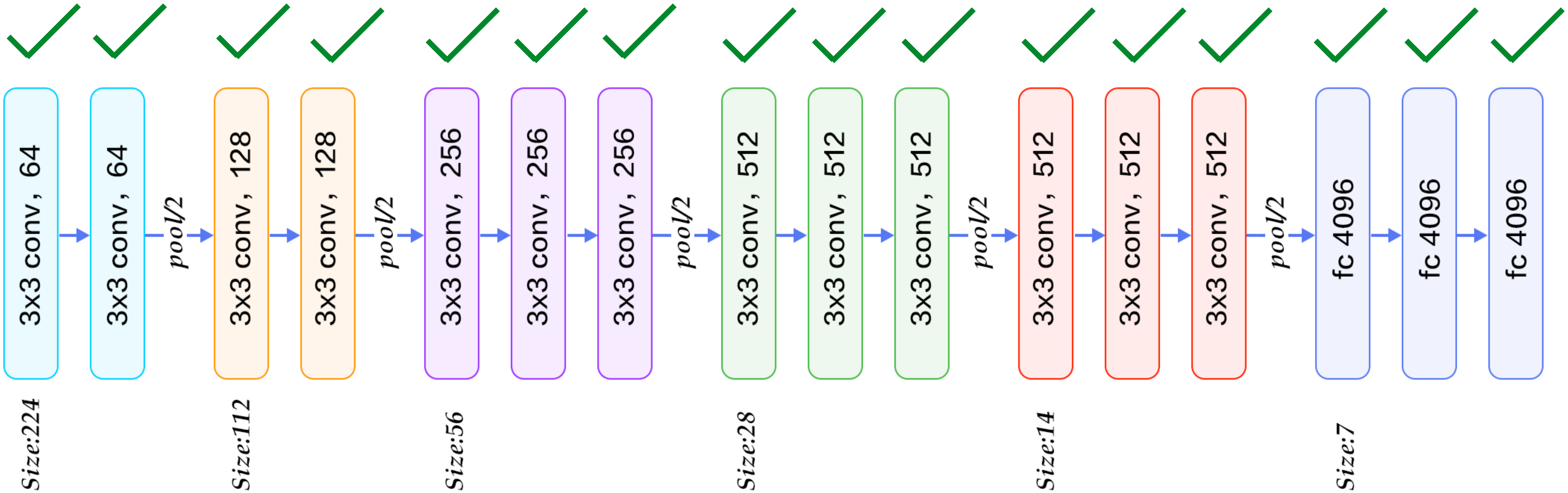
(7 x 7 x 512 image -> stretches to 25,088 x 1)

4,096

102,760,448 parameters!

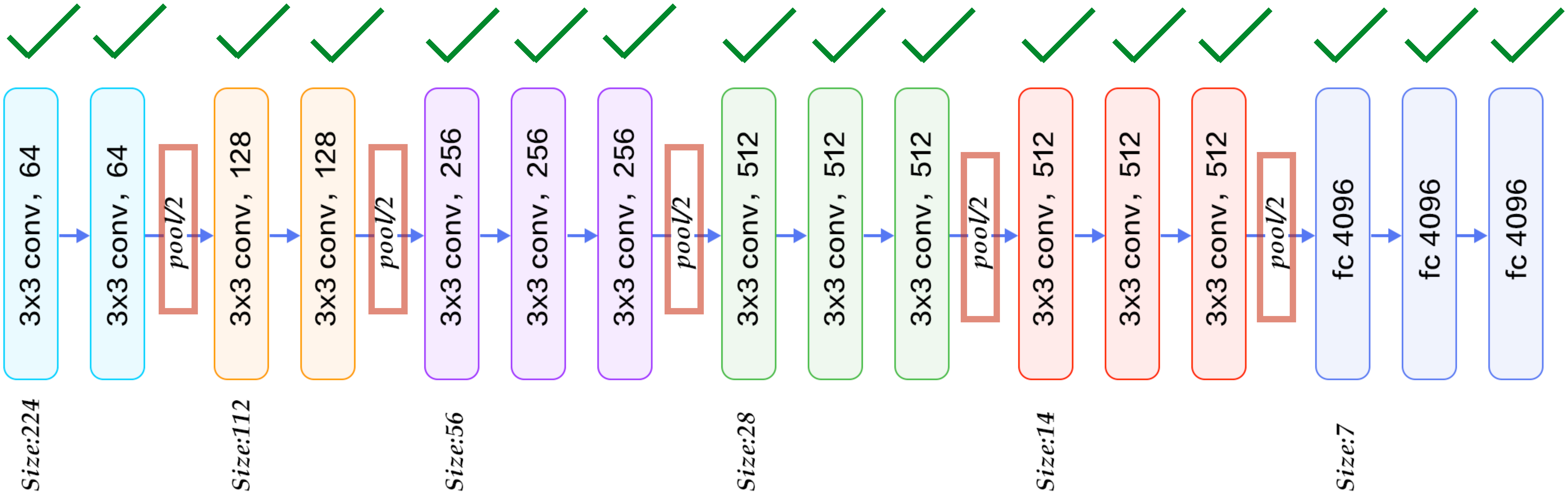
* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Convolutional Neural Networks



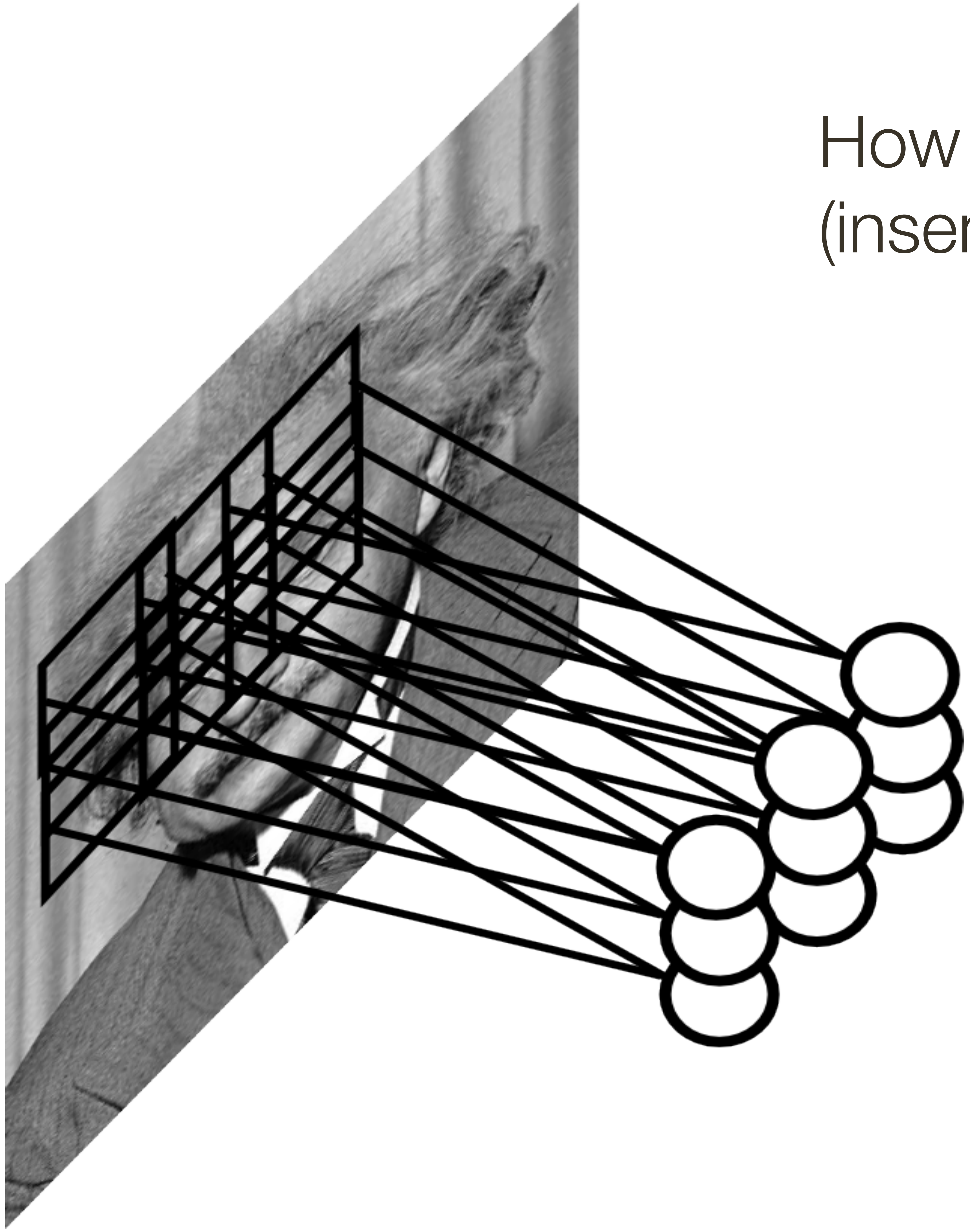
VGG-16 Network

Convolutional Neural Networks



VGG-16 Network

Pooling Layer

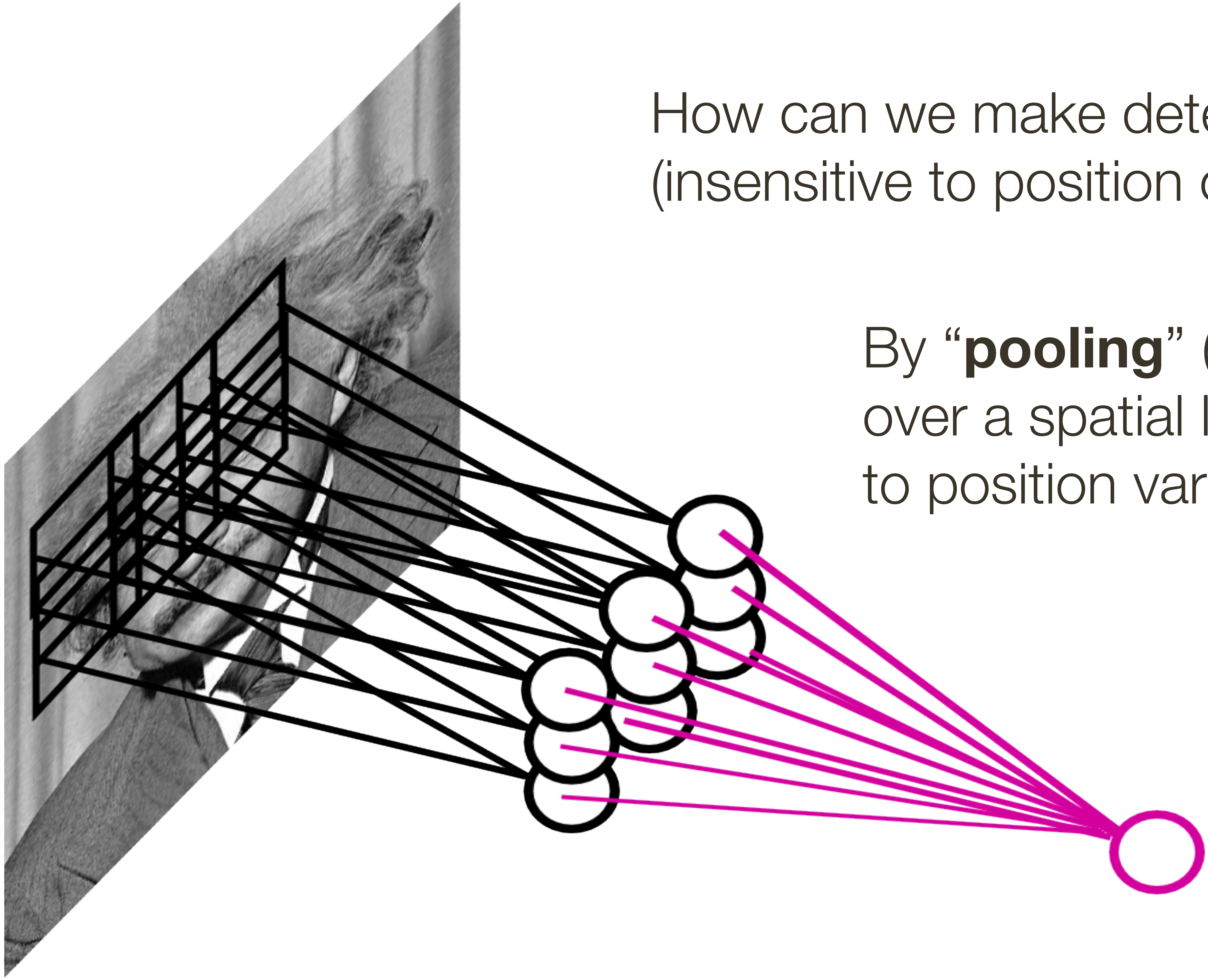


Let us assume the filter is an “eye” detector

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

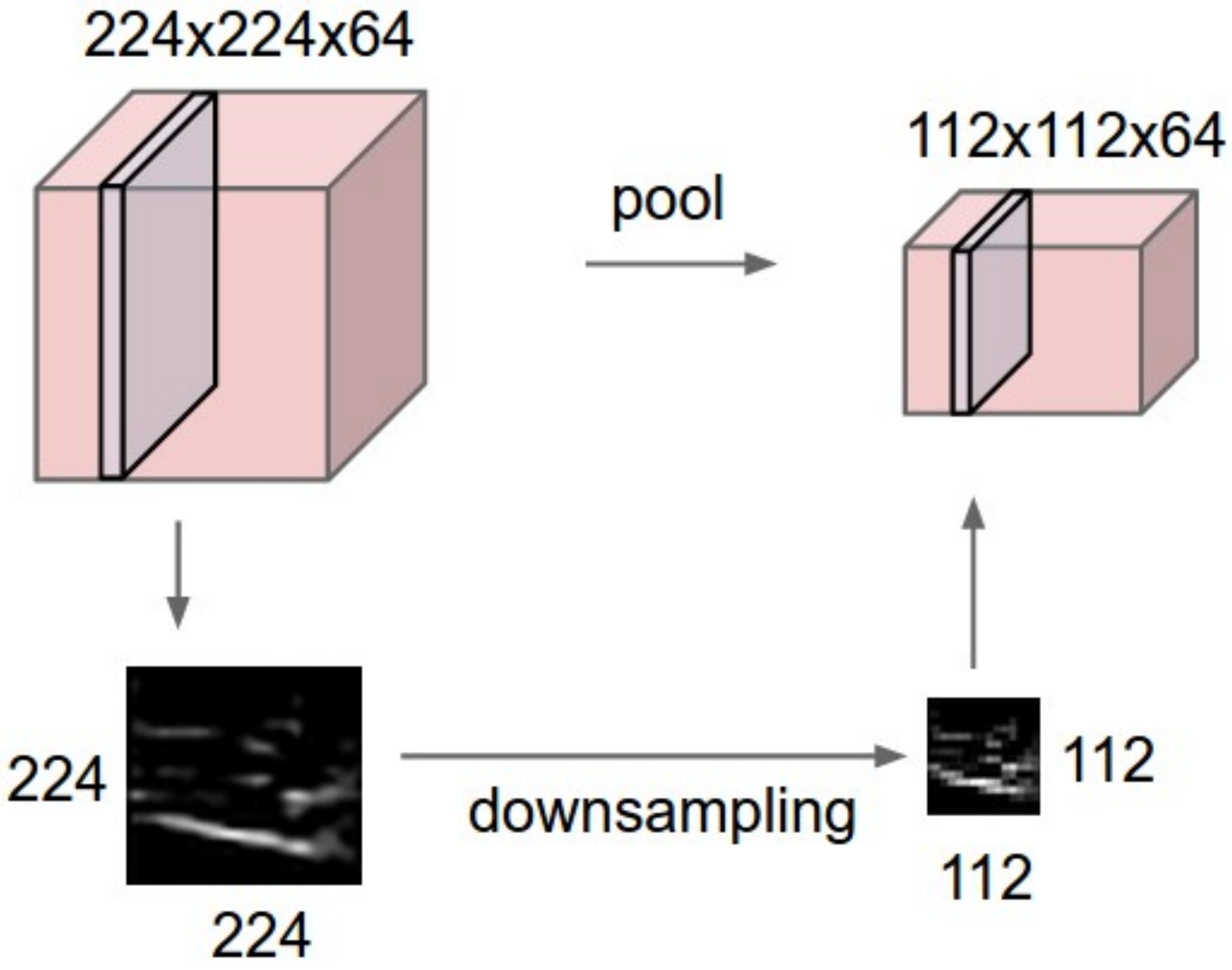
How can we make detection spatially invariant (insensitive to position of the eye in the image)

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

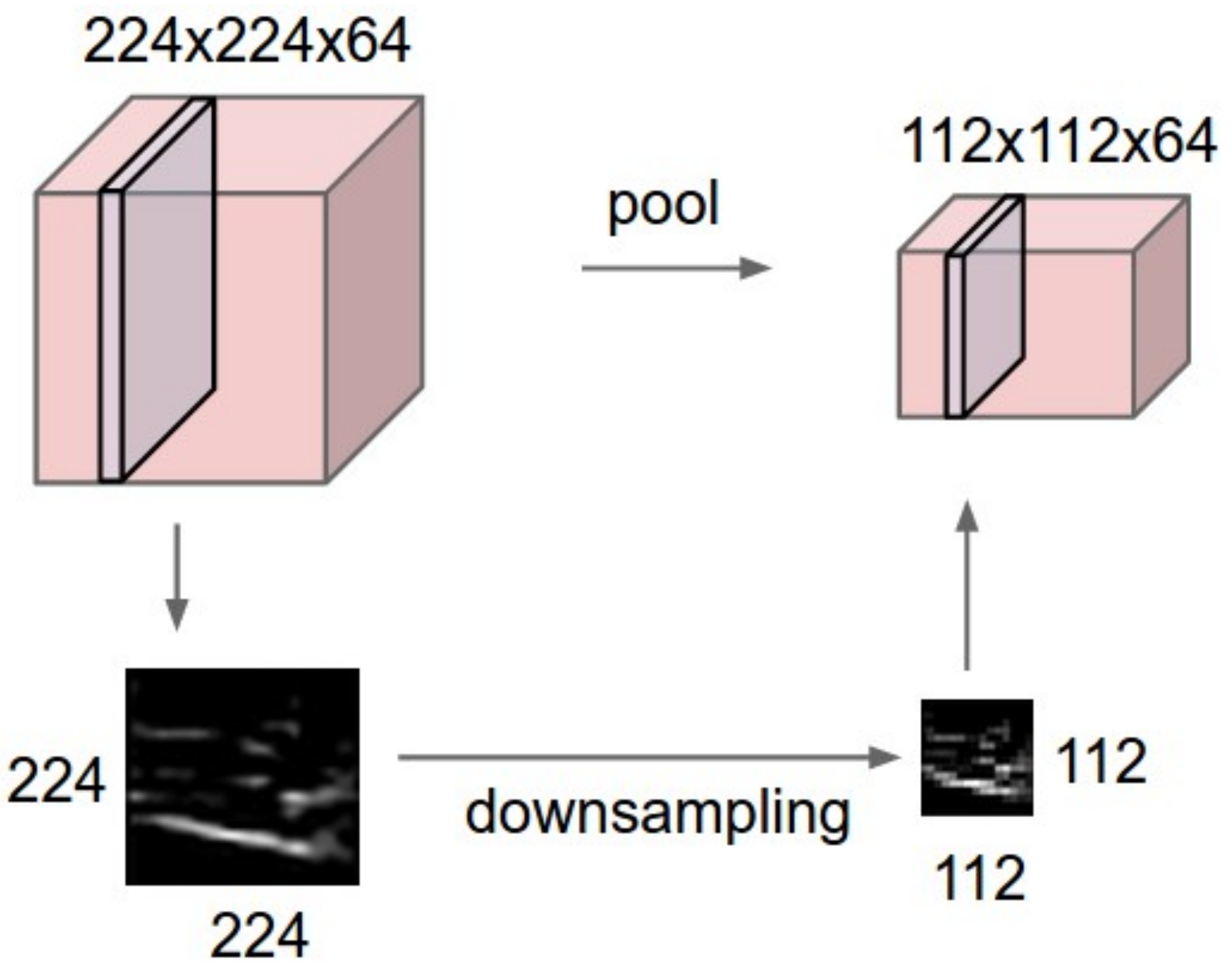
- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently



* 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

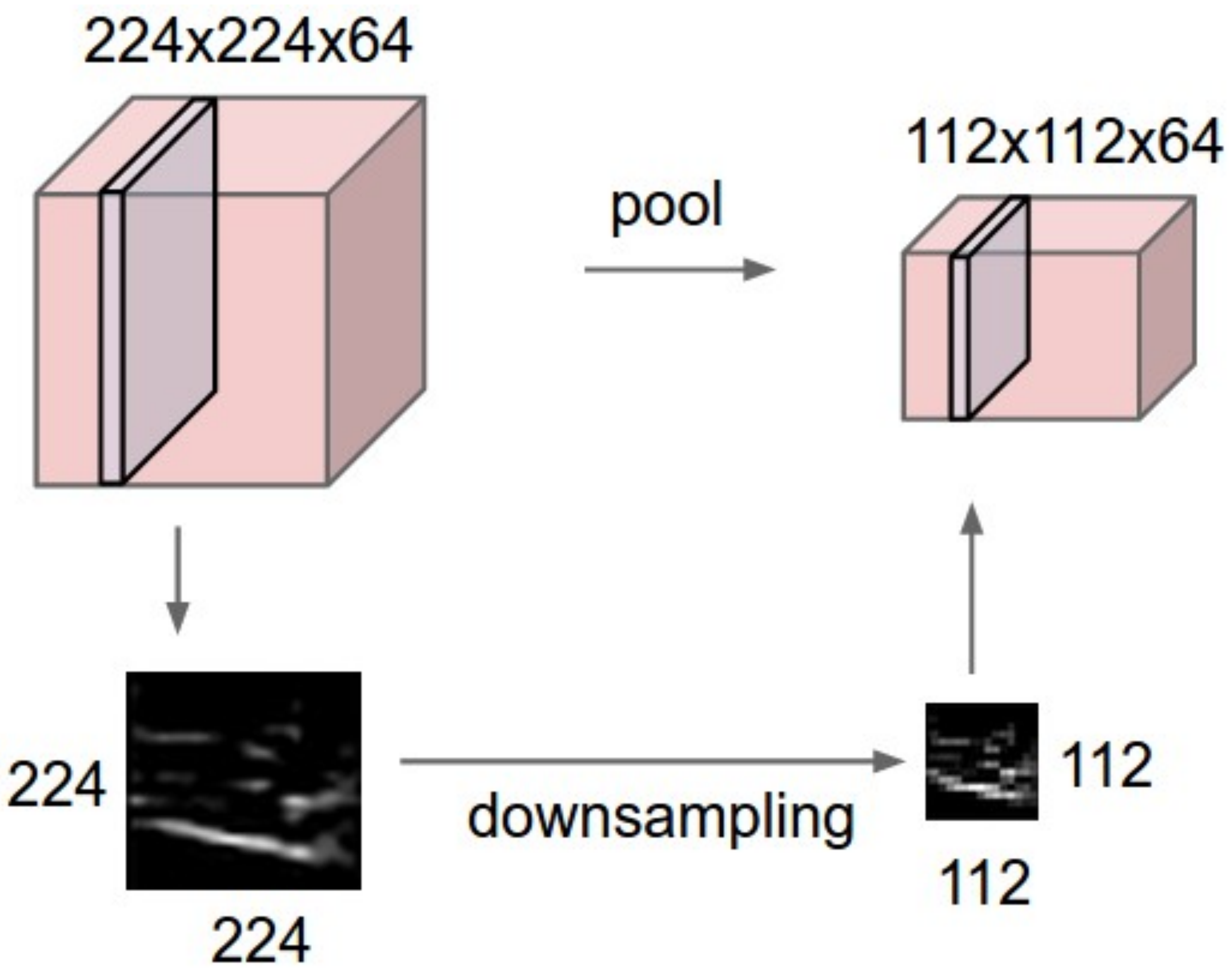


How many **parameters**?

* 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!

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

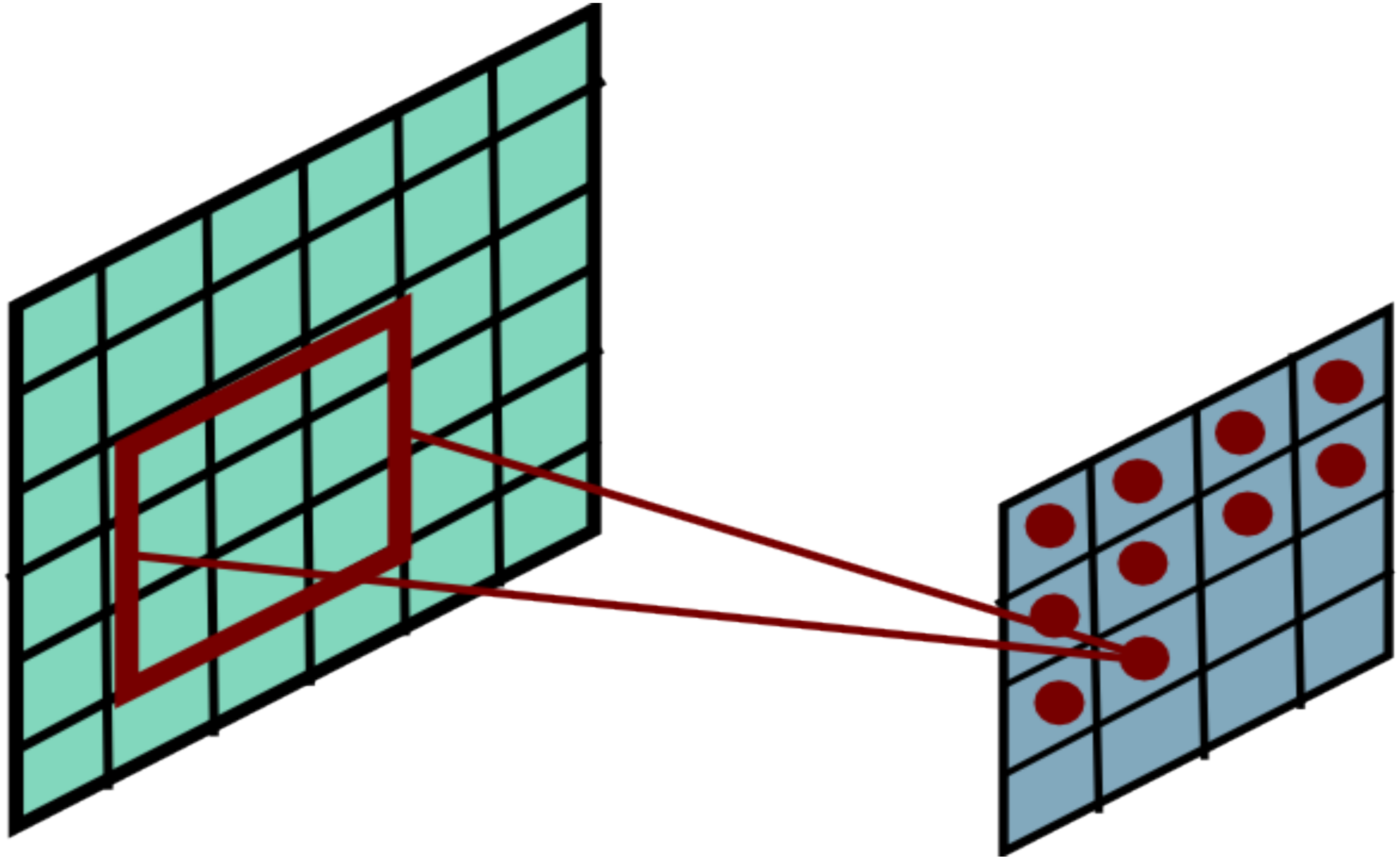
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

avg pool with 2 x 2 filter
and stride of 2

3.25	5.25
2	2

Pooling Layer **Receptive Field**

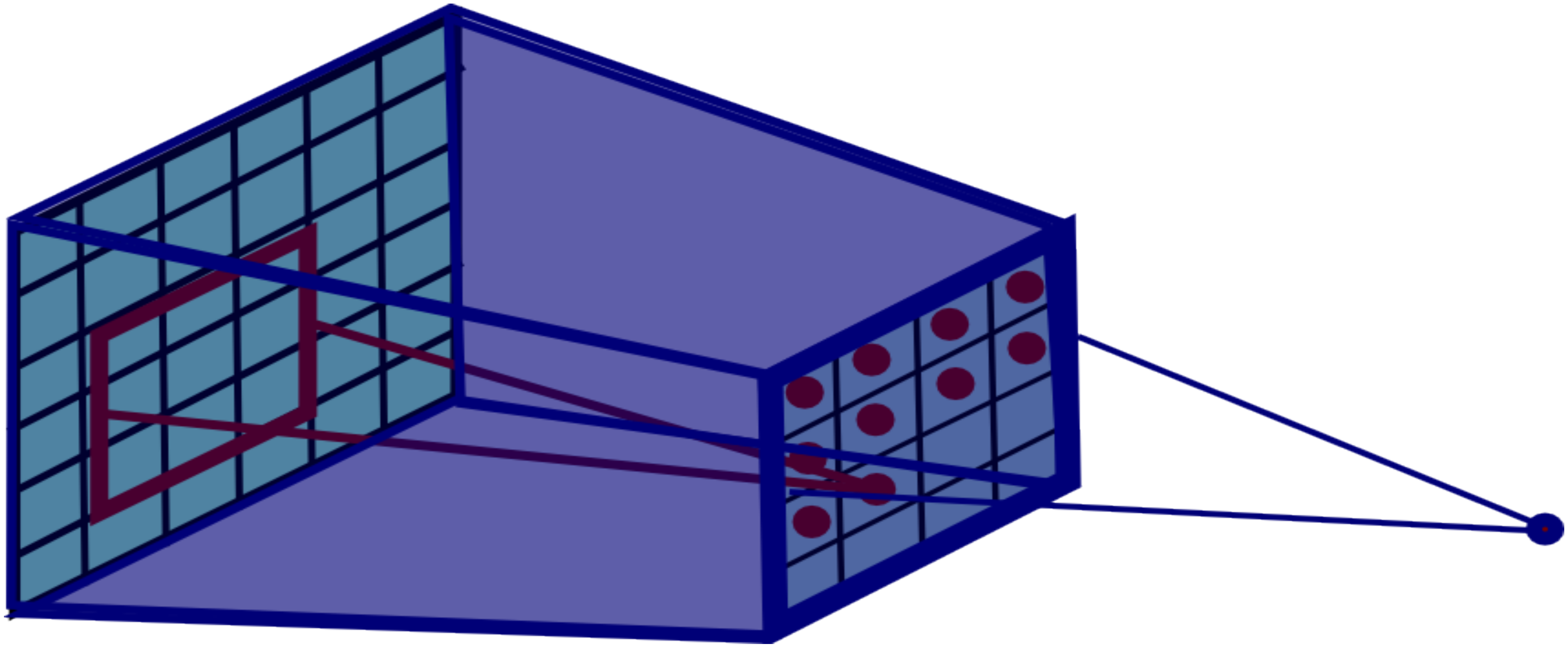
If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: **(P+K-1)x(P+K-1)**



* slide from Marc'Aurelio Renzato

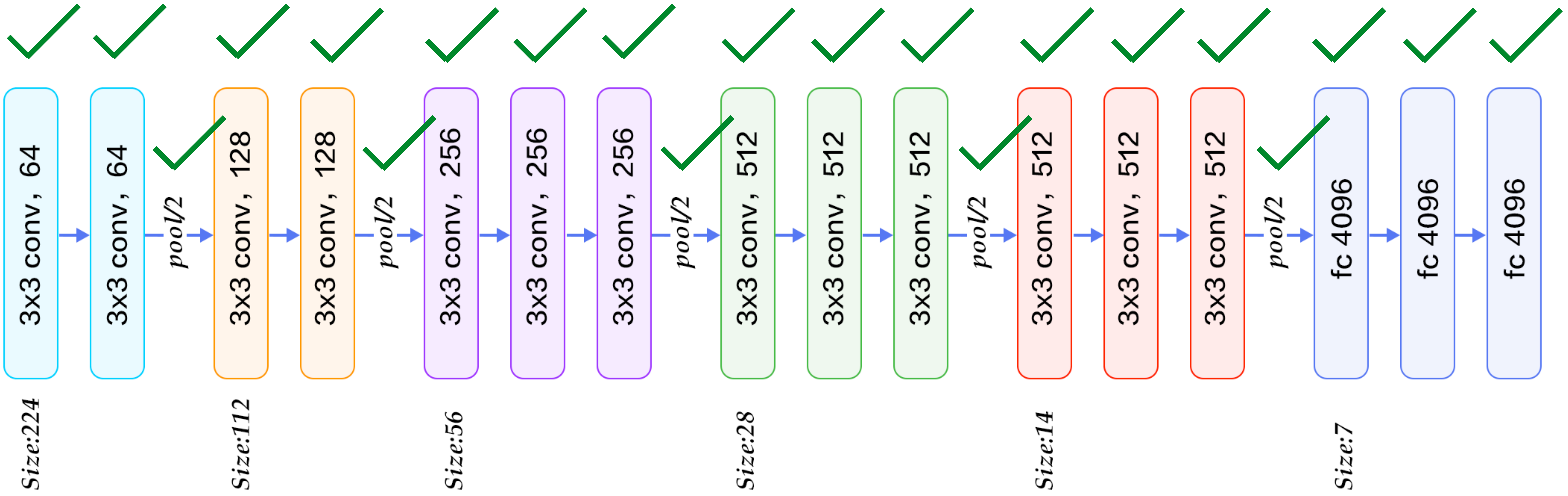
Pooling Layer **Receptive Field**

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

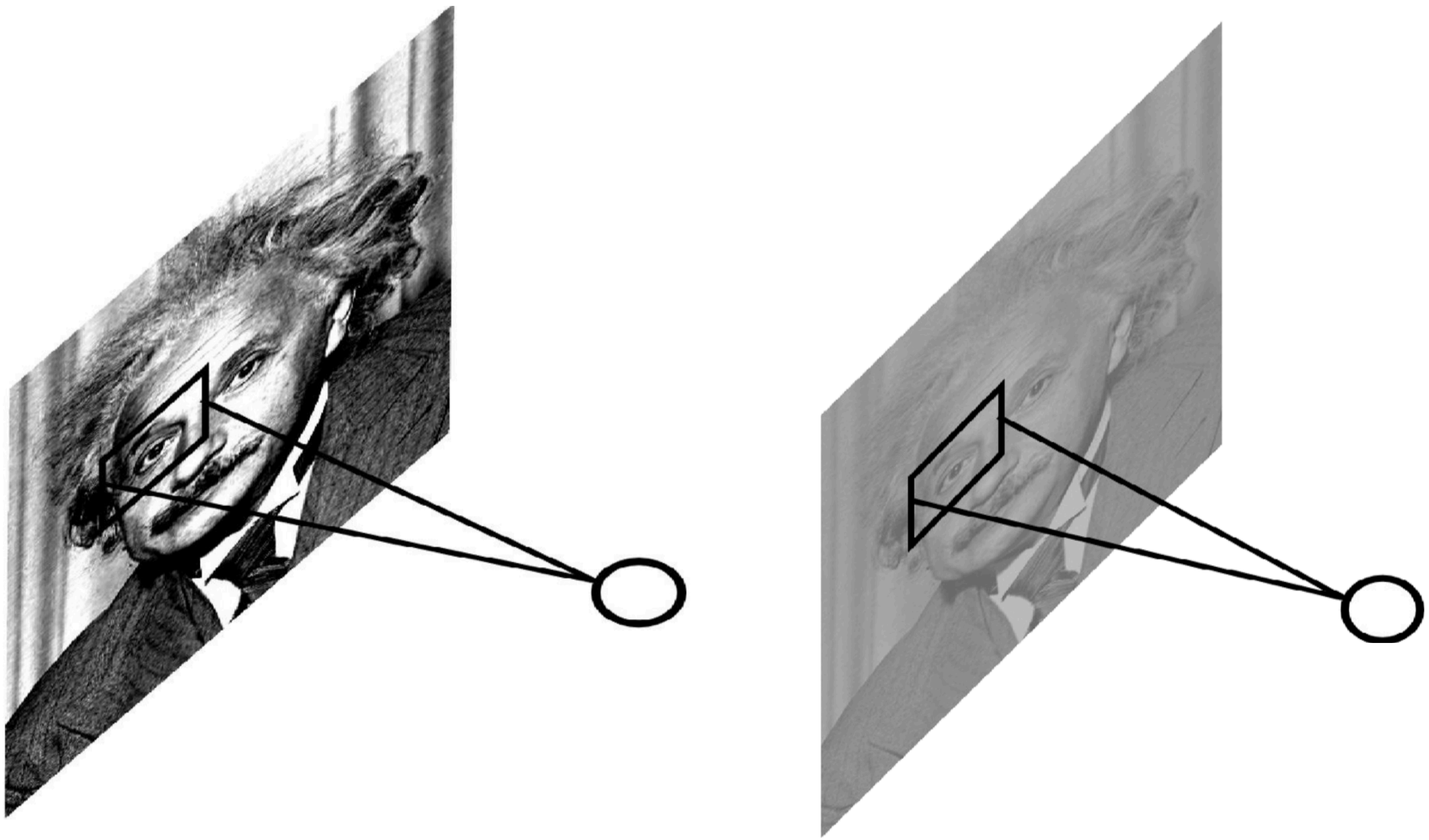
Convolutional Neural Networks



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

Improving **Single Model**

Regularization

- L2, L1
- Dropout / Inverted Dropout
- Data augmentation

Improving **Single Model**

Regularization

- L2, L1
- Dropout / Inverted Dropout
- Data augmentation

L2 Regularization: Learn a more (dense) distributed representation

$$R(\mathbf{W}) = ||\mathbf{W}||_2 = \sum_i \sum_j \mathbf{W}_{i,j}^2$$

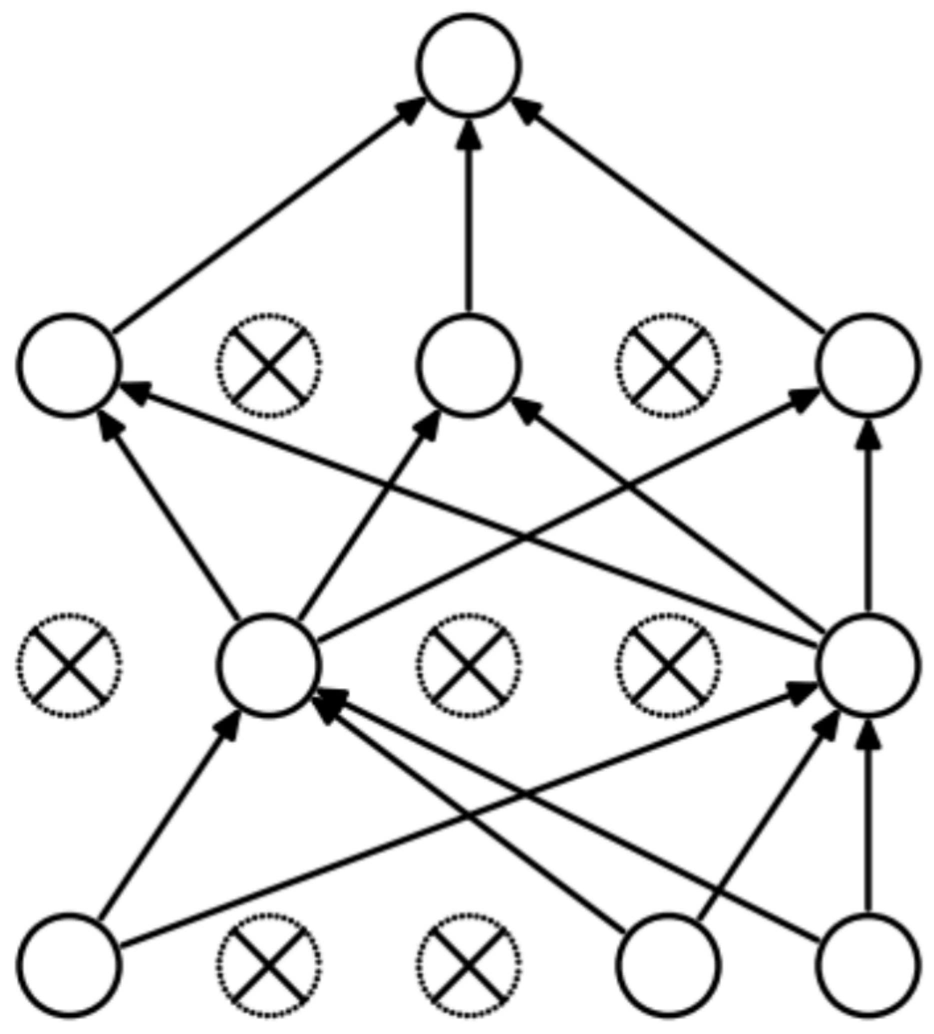
L1 Regularization: Learn a sparse representation (few non-zero wight elements)

$$R(\mathbf{W}) = ||\mathbf{W}||_1 = \sum_i \sum_j |\mathbf{W}_{i,j}|$$

Improving **Single Model**

Regularization

- L2, L1
- Dropout / Inverted Dropout
- Data augmentation



Dropout

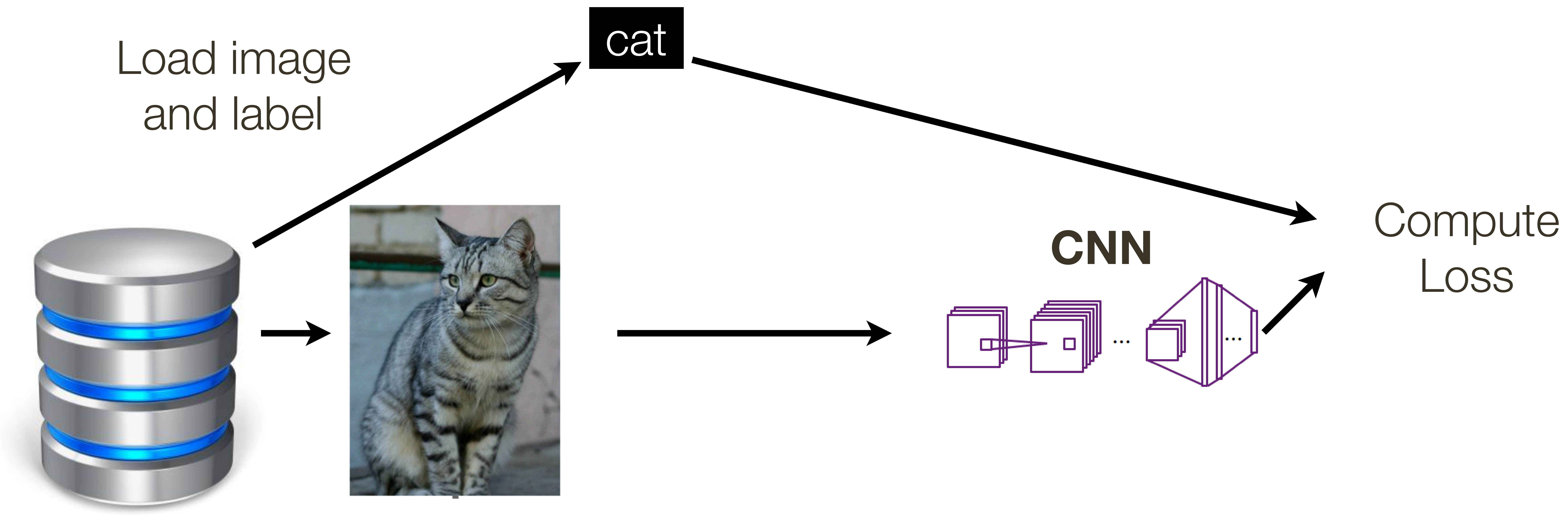
L2 Regularization: Learn a more (dense) distributed representation

$$R(\mathbf{W}) = ||\mathbf{W}||_2 = \sum_i \sum_j \mathbf{W}_{i,j}^2$$

L1 Regularization: Learn a sparse representation (few non-zero wight elements)

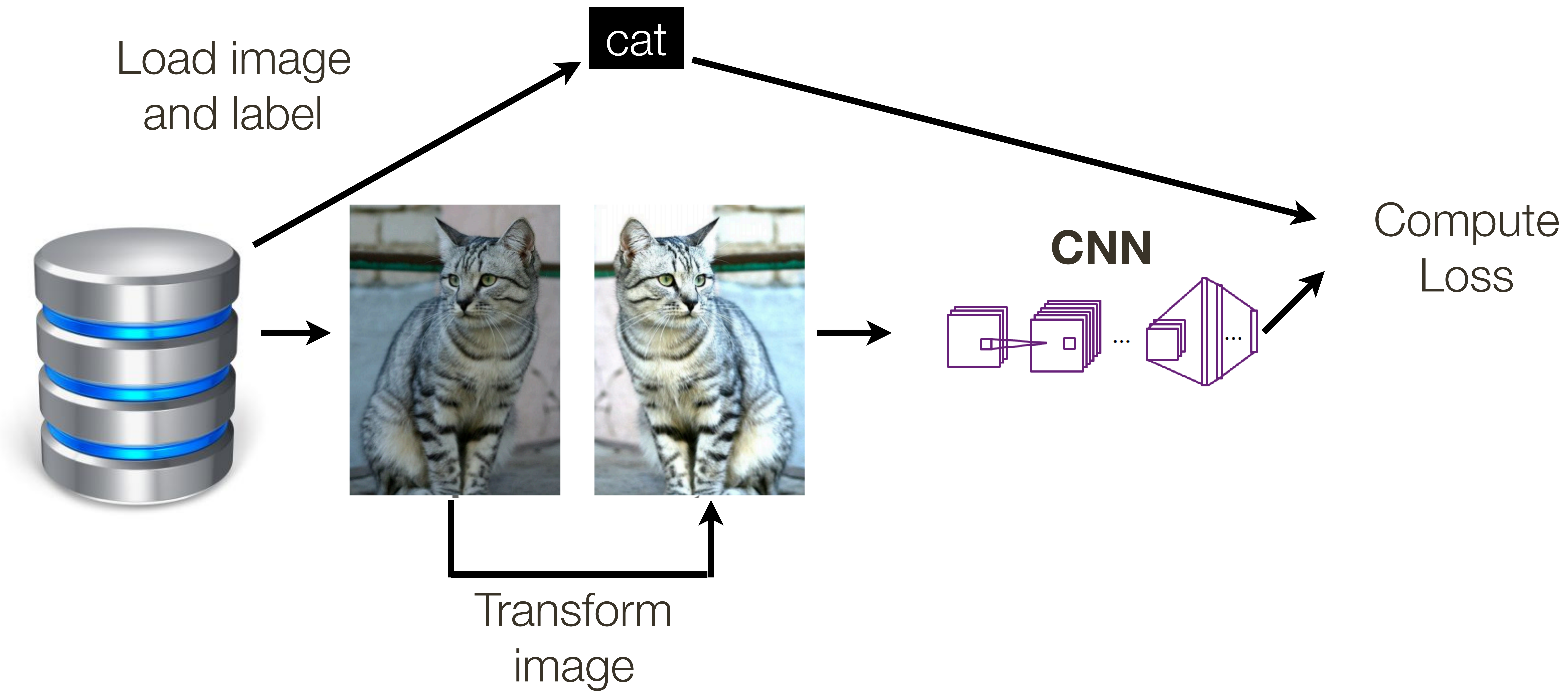
$$R(\mathbf{W}) = ||\mathbf{W}||_1 = \sum_i \sum_j |\mathbf{W}_{i,j}|$$

Regularization: Data Augmentation



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

Regularization: Data Augmentation



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

Regularization: Data Augmentation

Horizontal flips

Random crops & scales

Color Jitter

Regularization: Data Augmentation

Horizontal flips | Random crops & scales | Color Jitter

→

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

Regularization: Data Augmentation

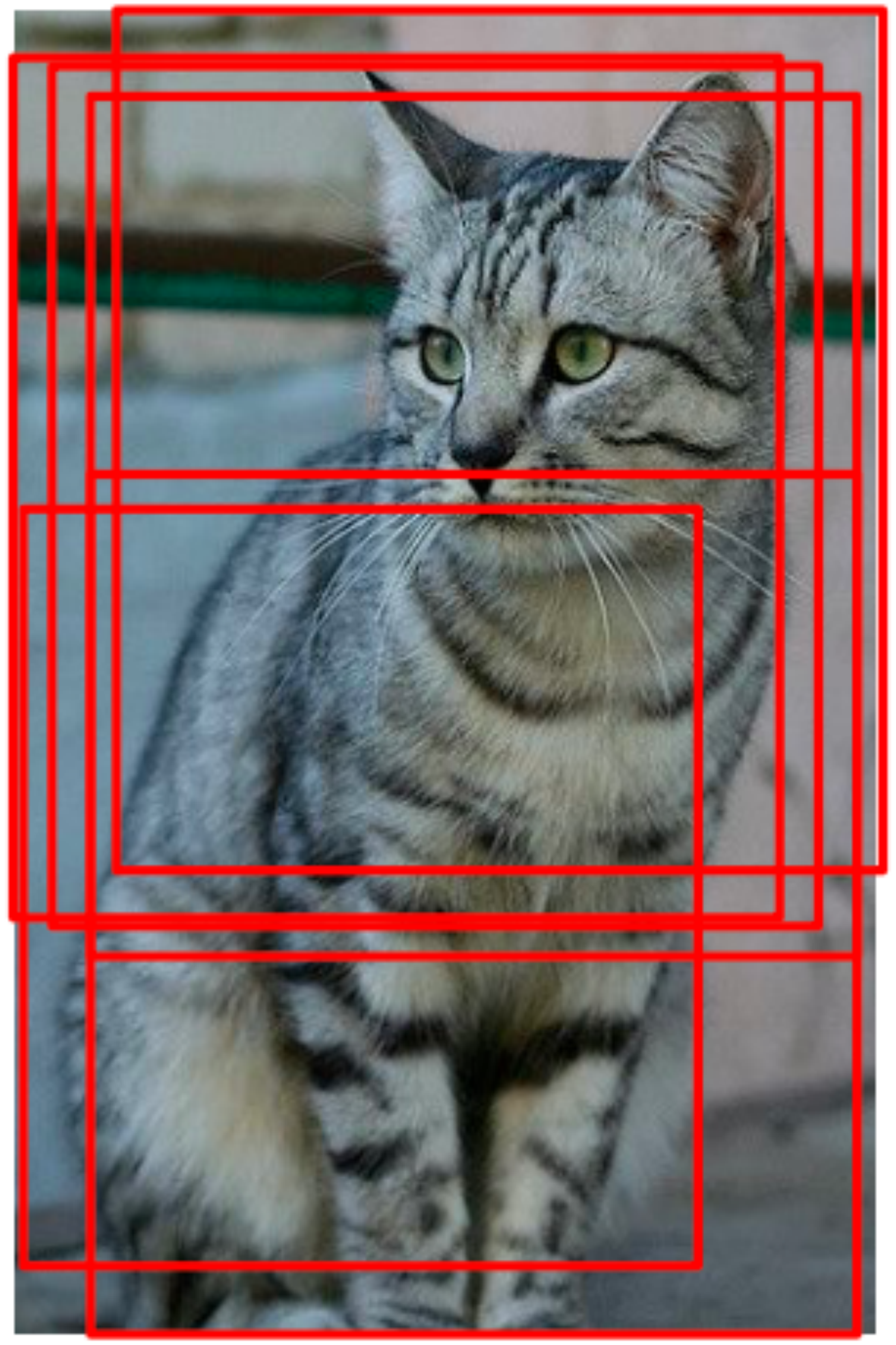
Horizontal flips | **Random crops & scales** | Color Jitter

Training: sample random crops and scales
e.g., ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short size = L
3. Sample random 224x224 patch

Testing: average a fix set of crops
e.g., ResNet:

1. Resize image to 5 scales (224, 256, 384, 480, 640)
2. For each image use 10 224x224 crops: 4 corners + center, + flips



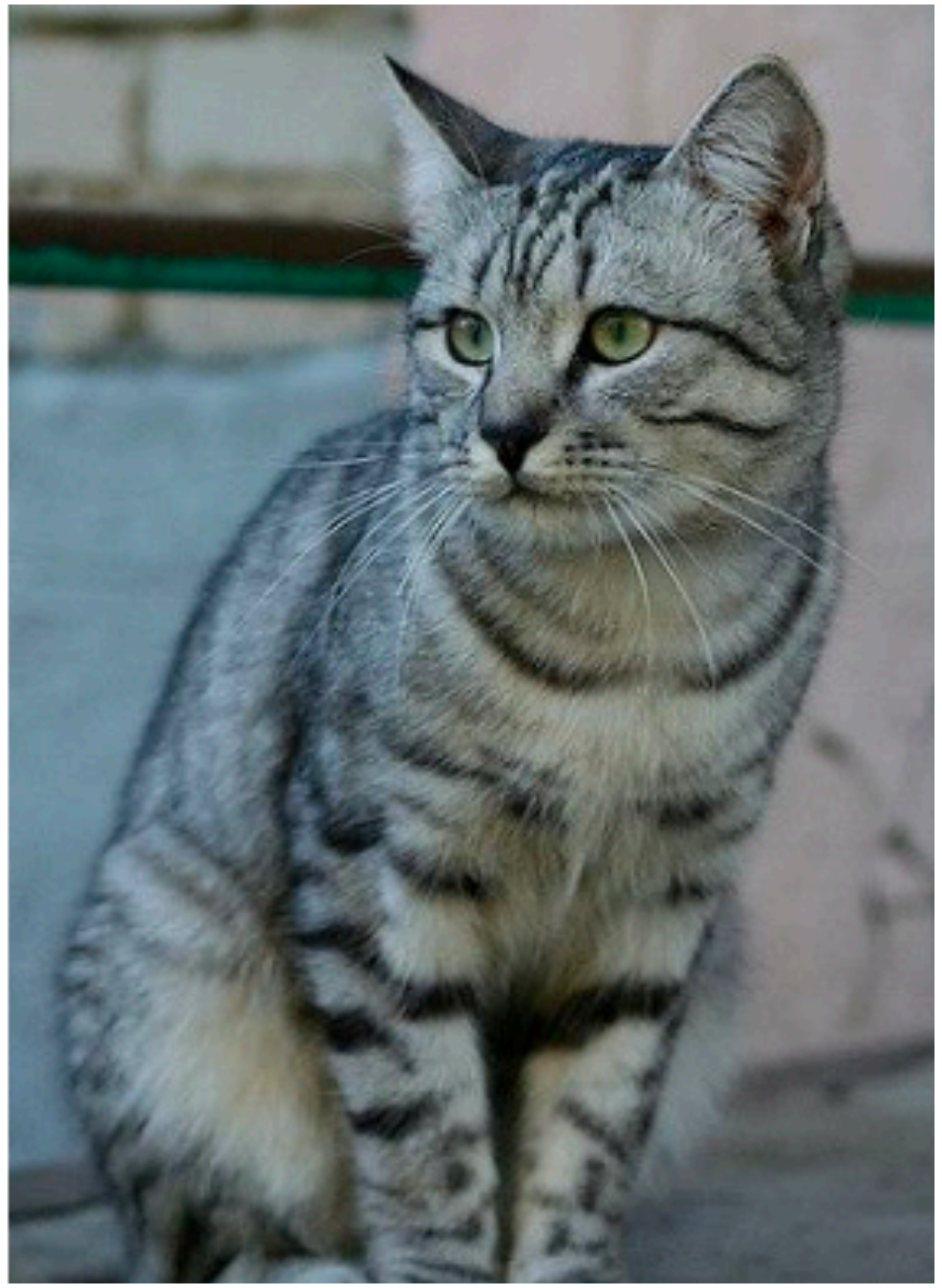
Regularization: Data Augmentation

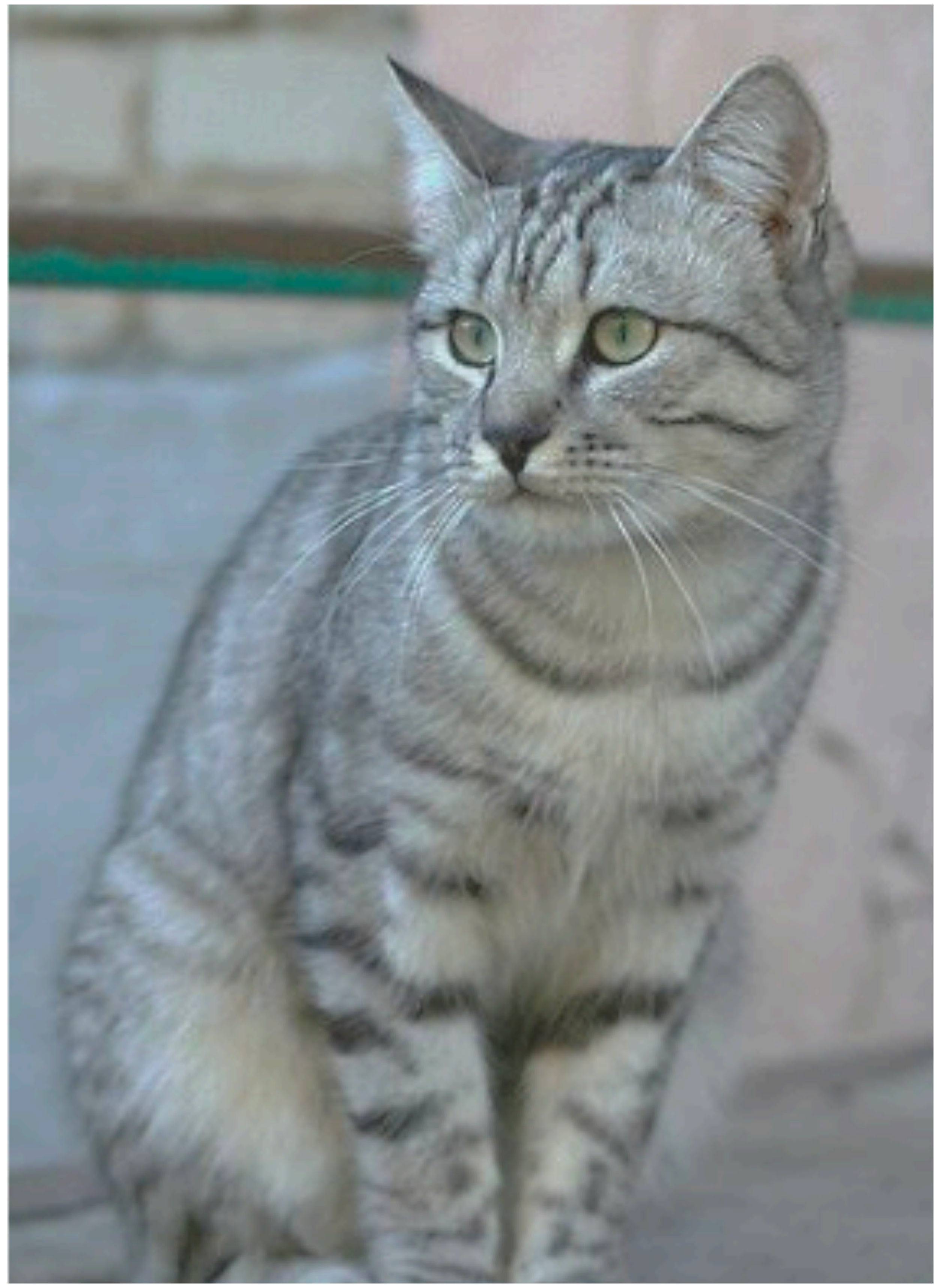
Horizontal flips

Random crops & scales

Color Jitter

Random perturbations in contrast and brightness





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

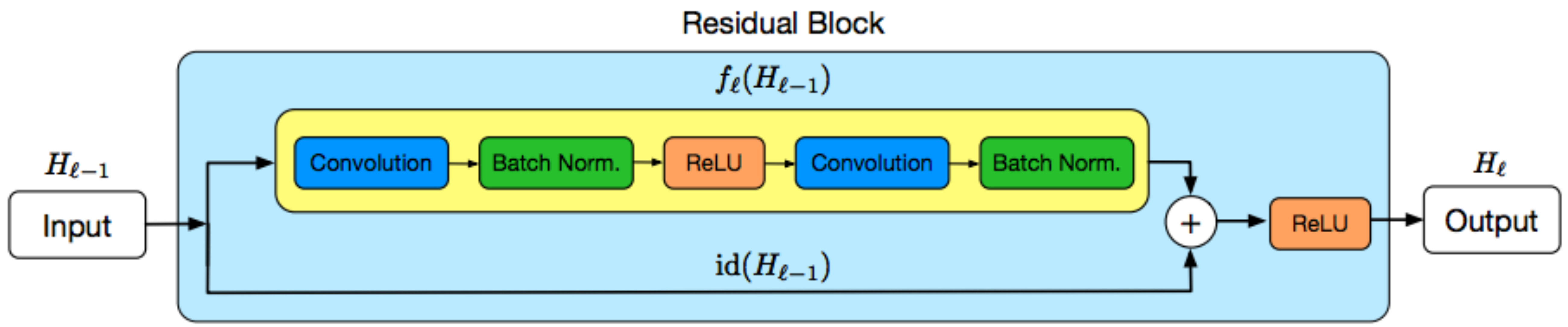
Regularization: Stochastic Depth

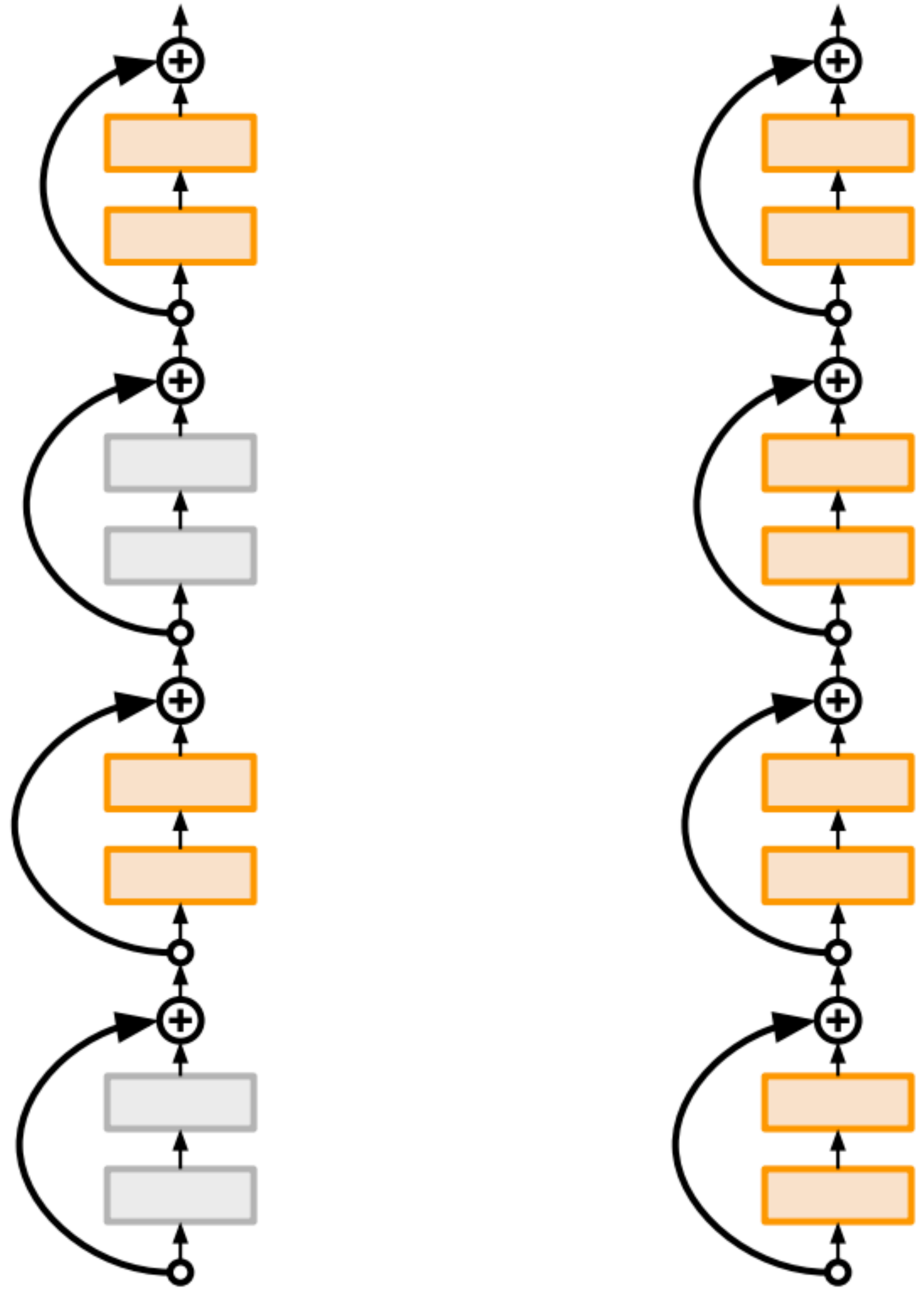
[Huang et al., ECCV 2016]

Effectively "dropout" but for layers

Stochastically with some probability **turn off some layer** (for each batch)

Effectively trains a collection of neural networks





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

Transfer Learning with CNNs

Common “Wisdom”: You need a lot of data to train a CNN

Transfer Learning with CNNs

Common "Wisdom": You need a lot of data to train a CNN

Solution: Transfer learning — taking a model trained on the task that has lots of data and adopting it to the task that may not

Transfer Learning with CNNs

Common "Wisdom": You need a lot of data to train a CNN

Solution: Transfer learning — taking a model trained on the task that has lots of data and adopting it to the task that may not

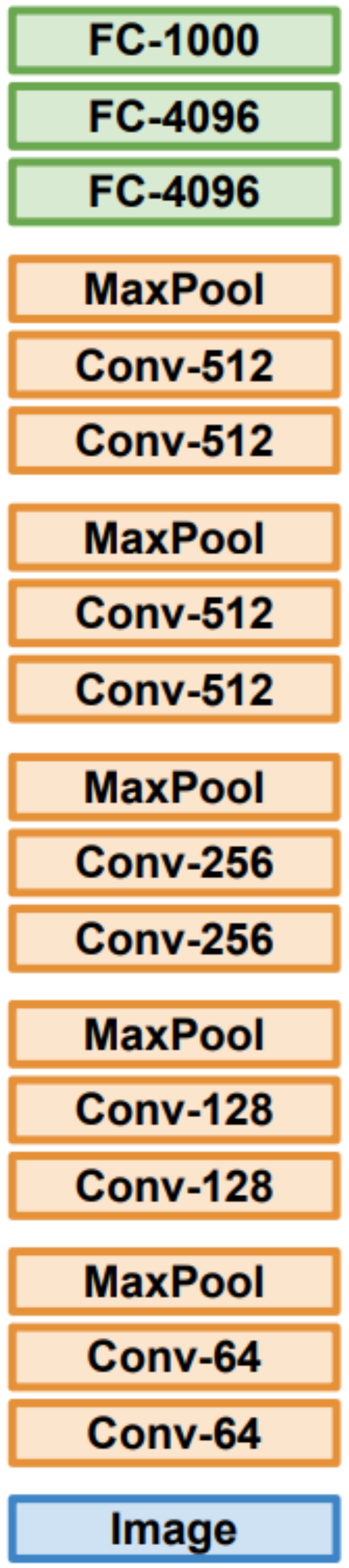
This strategy is PERVASIVE.

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

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

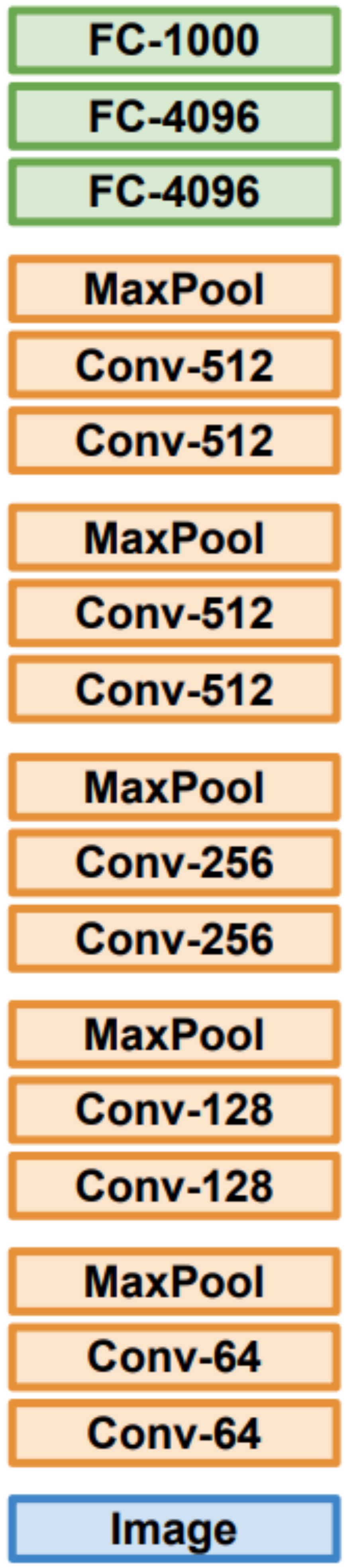
Train on **ImageNet**



Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet**

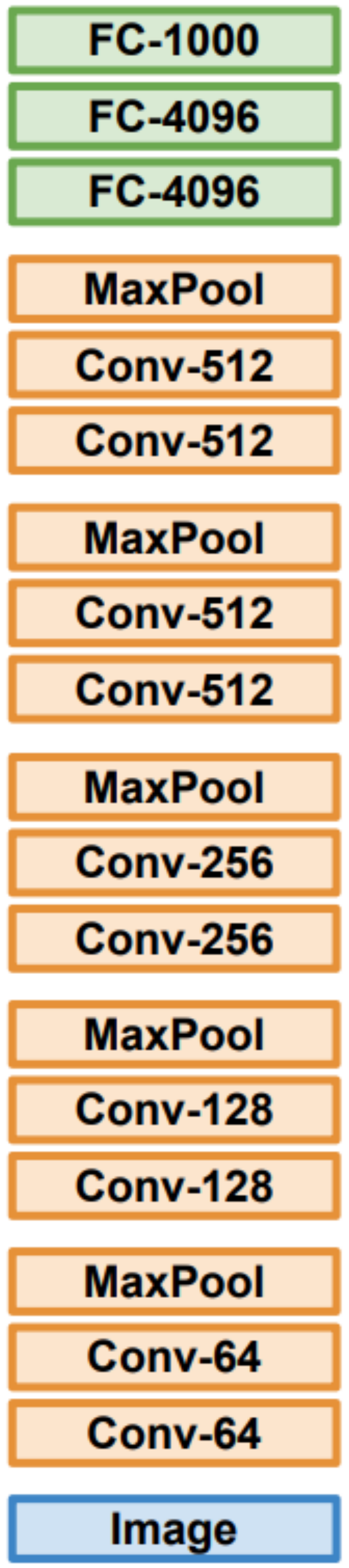


Why on **ImageNet**?

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet**



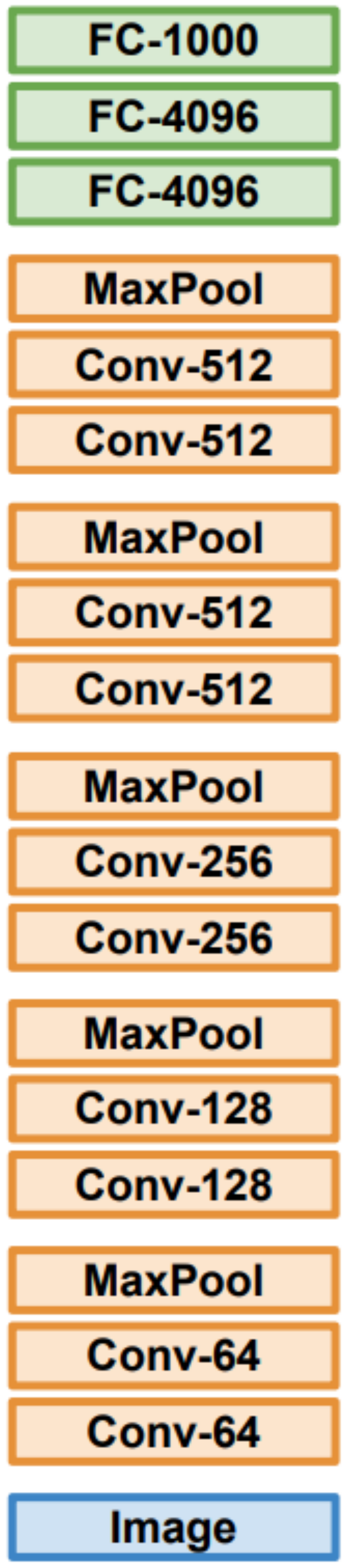
Why on **ImageNet**?

- Convenience, lots of **data**
- We know how to **train these well**

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet**



Why on **ImageNet**?

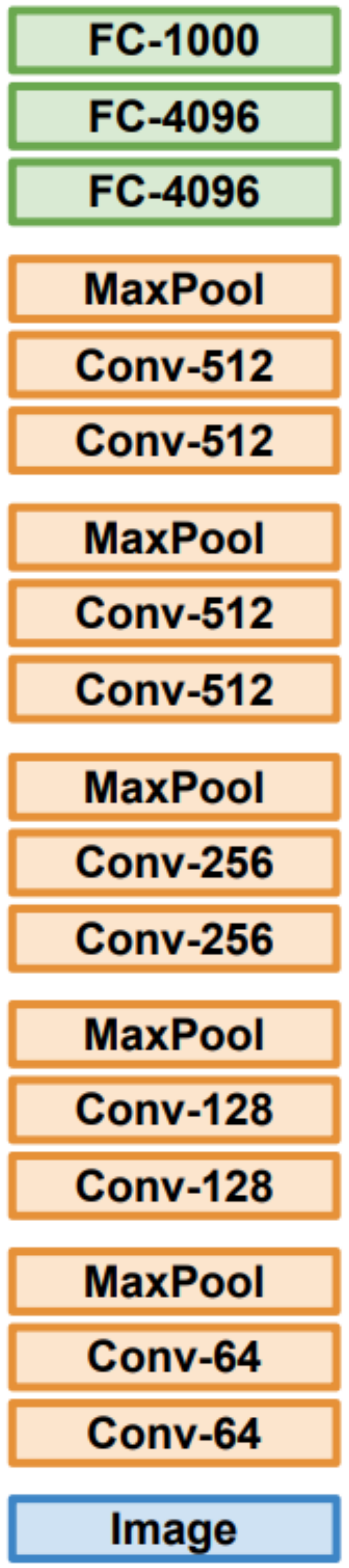
- Convenience, lots of **data**
- We know how to **train these well**

However, for some tasks we would need to start with something else (e.g., videos for optical flow)

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet**

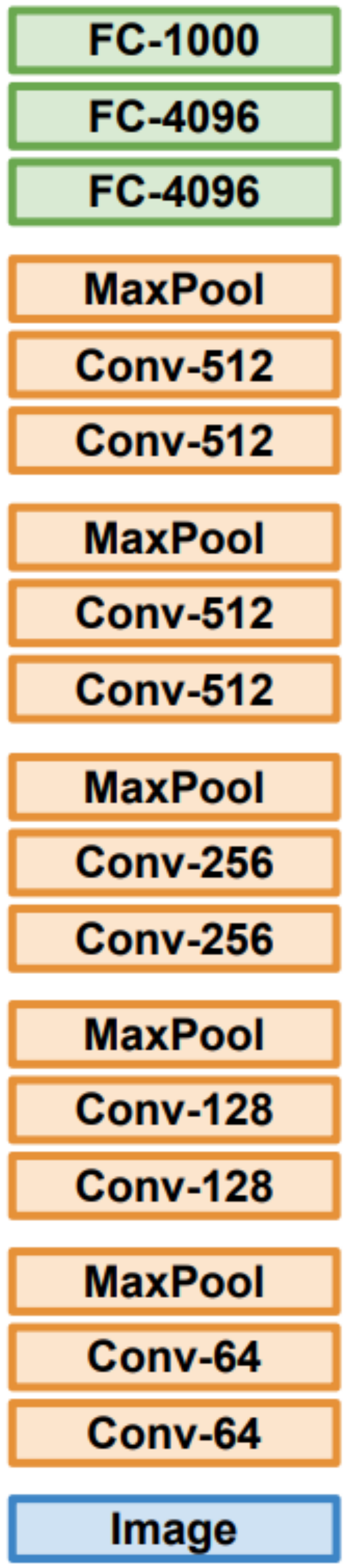


Small dataset with C classes

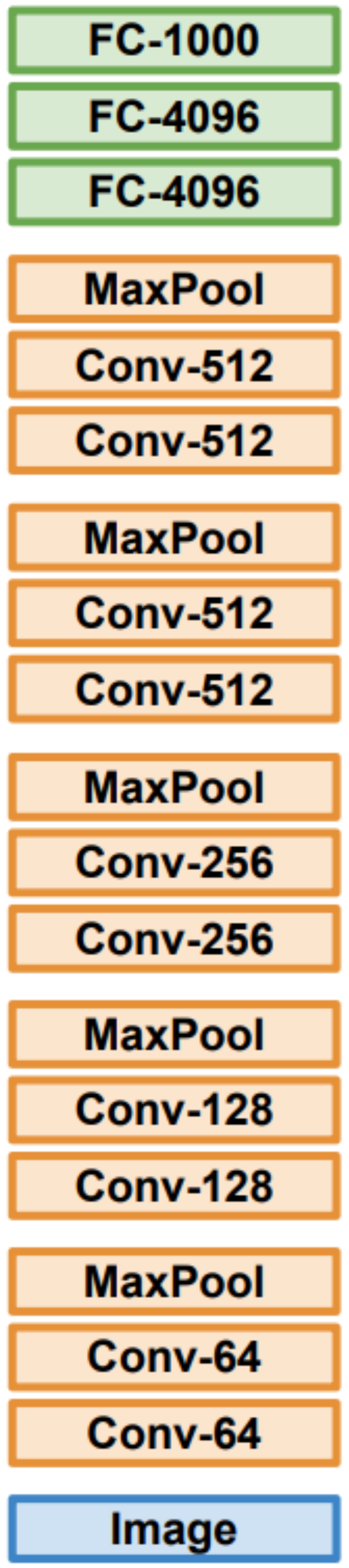
Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet**



Small dataset with C classes

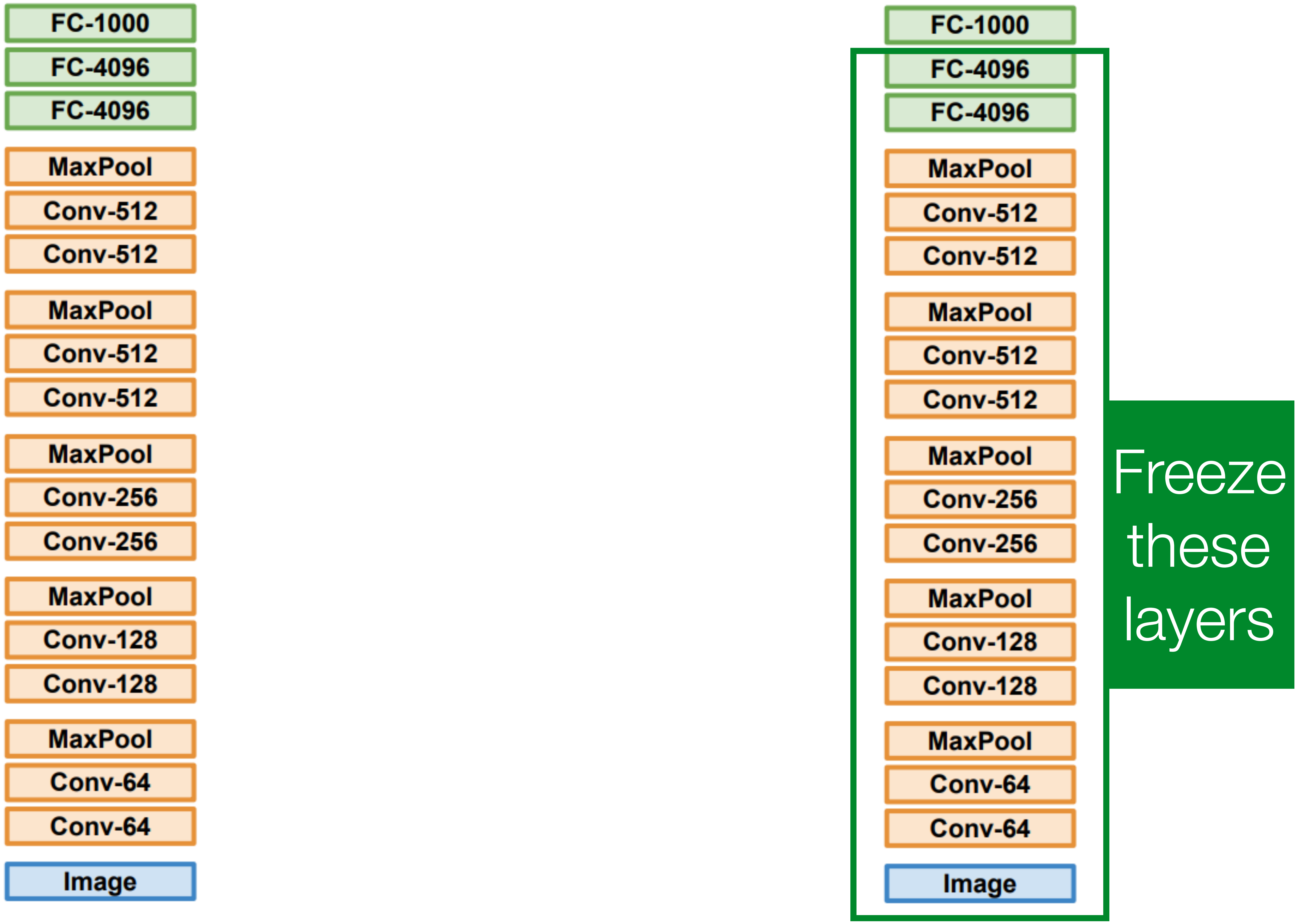


* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

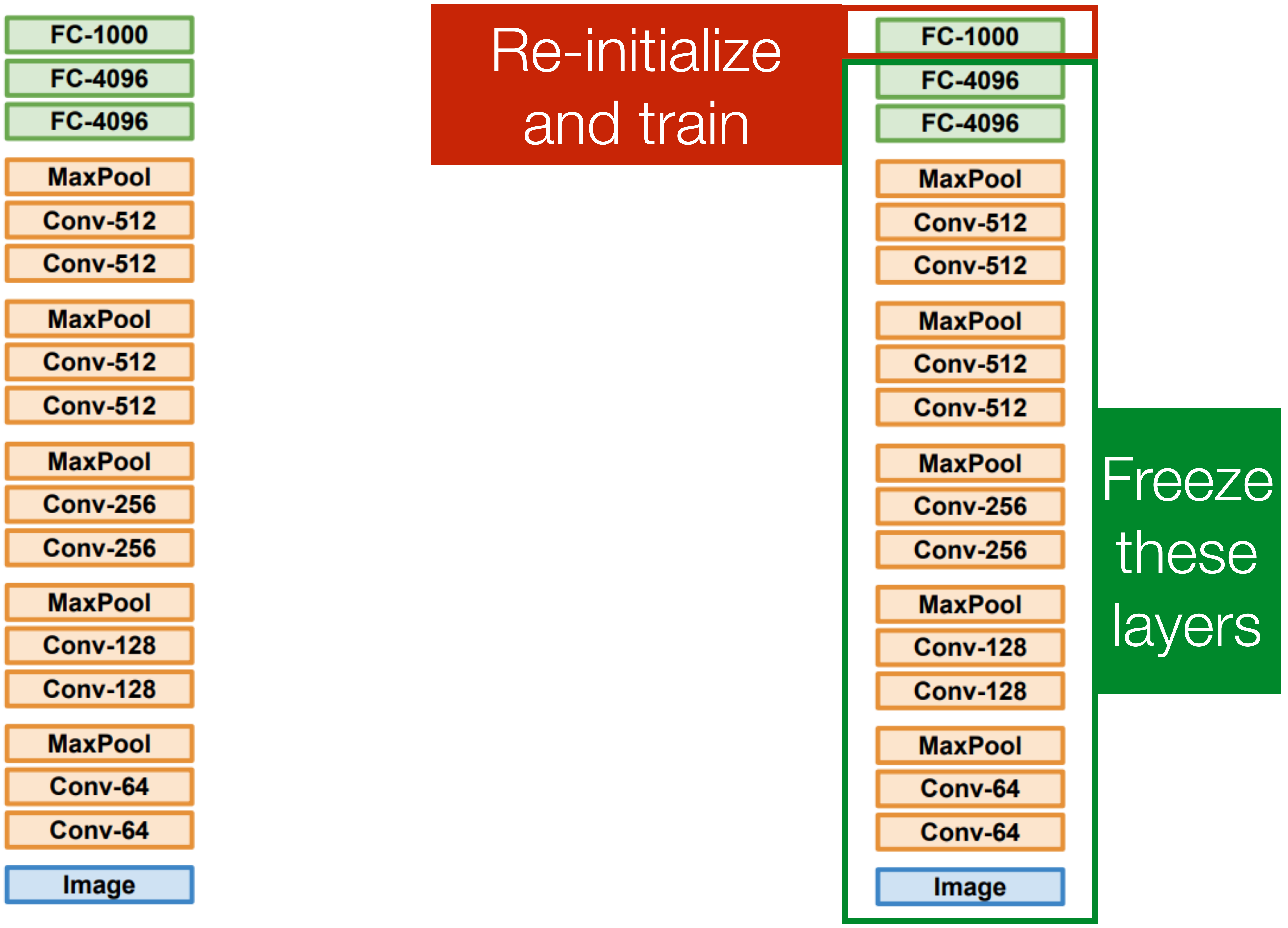
Train on **ImageNet** | **Small dataset** with C classes



Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet** | **Small dataset** with C classes



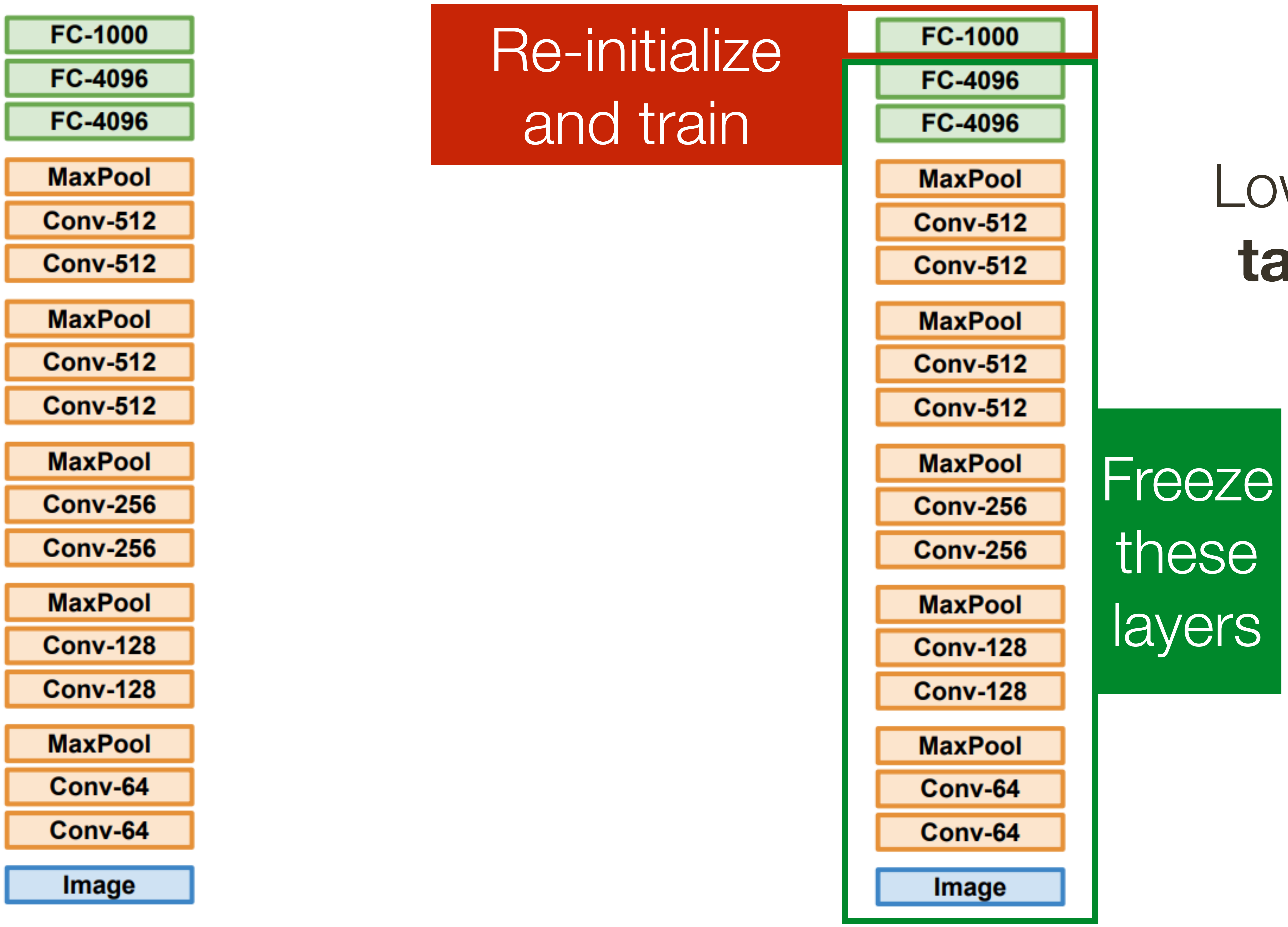
* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet**

Small dataset with C classes

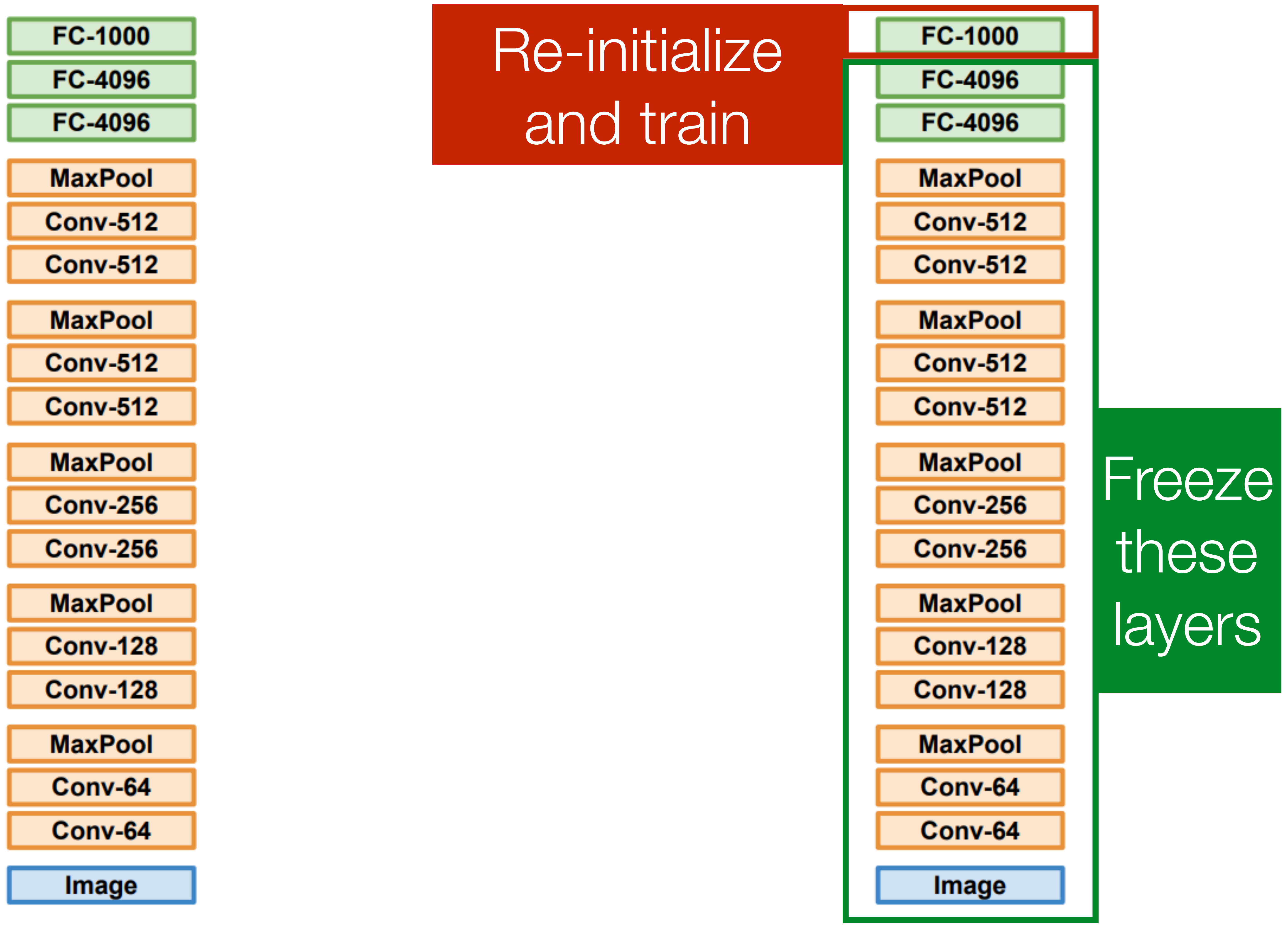


Lower levels of the CNN are at **task independent** anyways

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

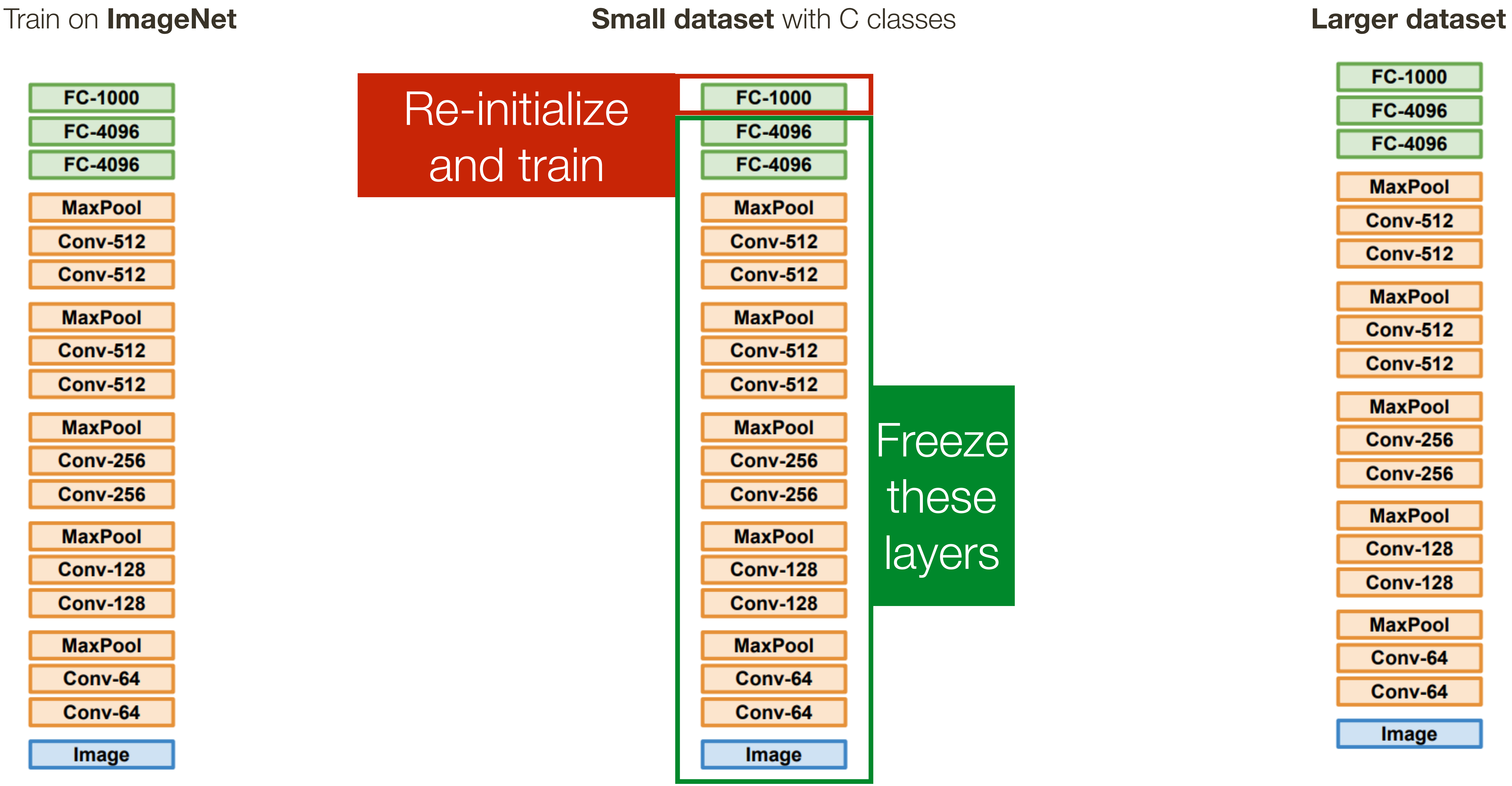
Train on **ImageNet** | **Small dataset** with C classes | **Larger dataset**



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

Transfer Learning with CNNs

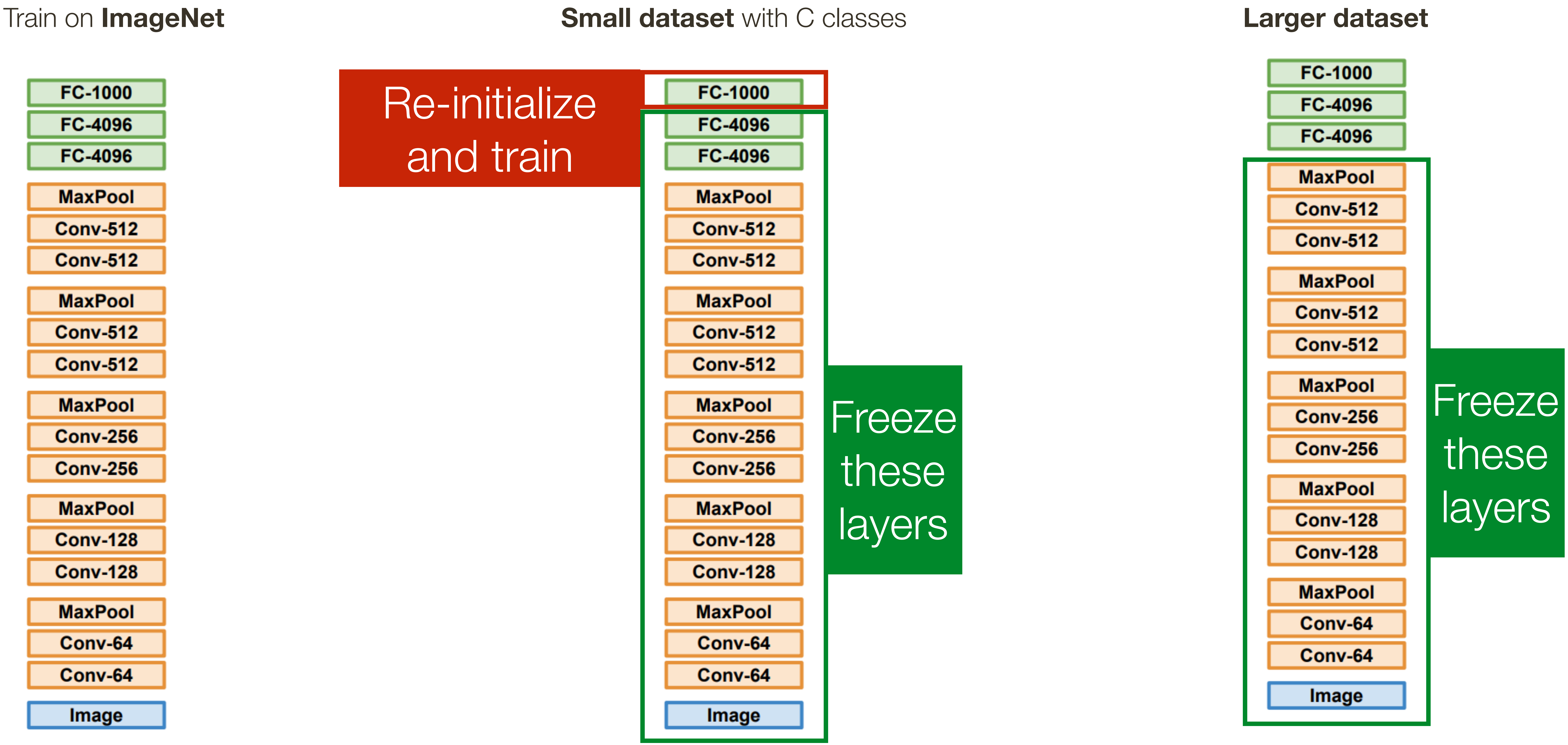
[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]



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

Transfer Learning with CNNs

[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

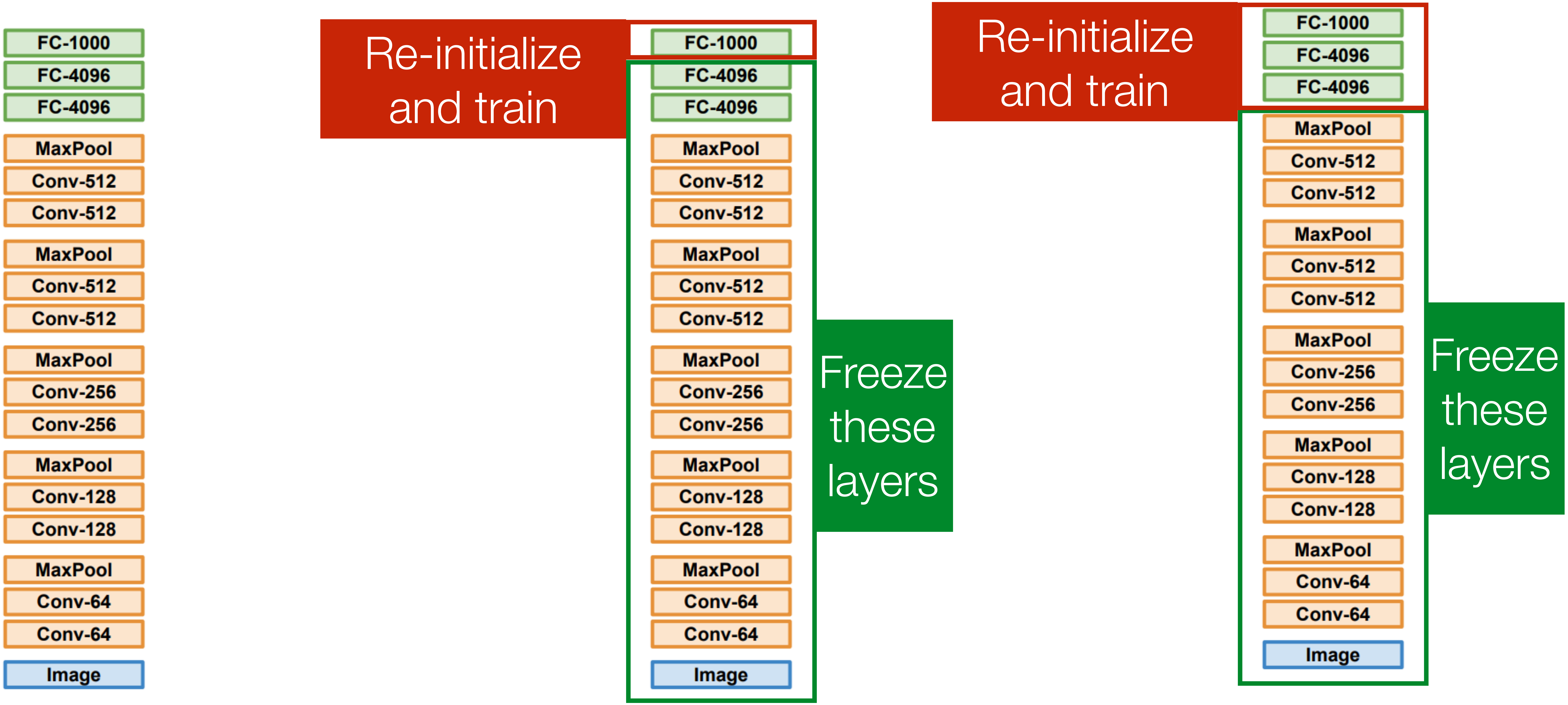


* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Transfer Learning with CNNs

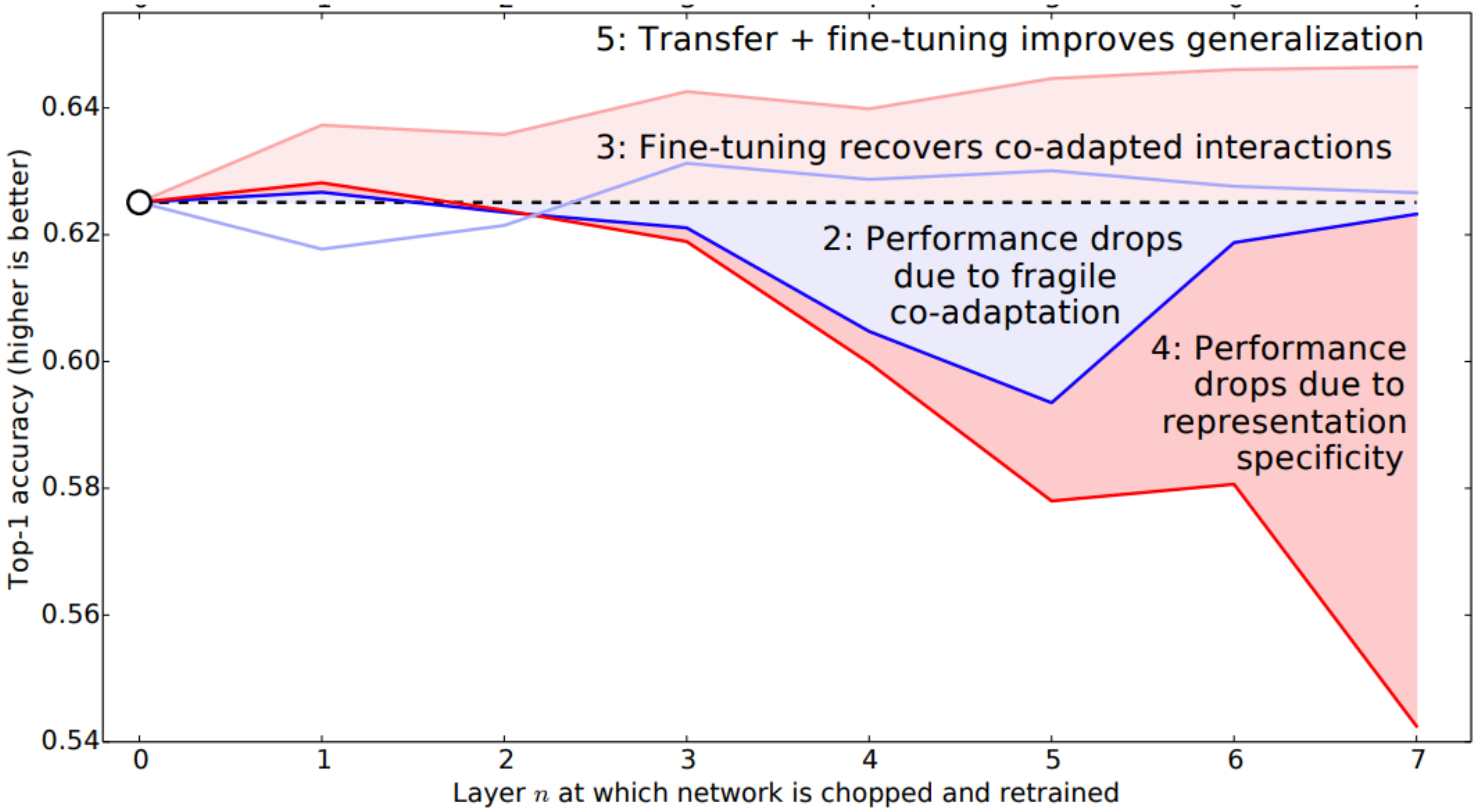
[Yosinski et al., NIPS 2014]
[Donahue et al., ICML 2014]
[Razavian et al., CVPR Workshop 2014]

Train on **ImageNet** | **Small dataset** with C classes | **Larger dataset**



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

Transfer Learning with CNNs



[Yosinski et al., NIPS 2014]

Model **Ensemble**

Training: Train multiple independent models

Test: Average their results

Model **Ensemble**

Training: Train multiple independent models

Test: Average their results

~ 2% improved performance in practice

Model **Ensemble**

Training: Train multiple independent models

Test: Average their results

~ 2% improved performance in practice

Alternative: Multiple snapshots of the single model during training!

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

Model **Ensemble**

Training: Train multiple independent models

Test: Average their results

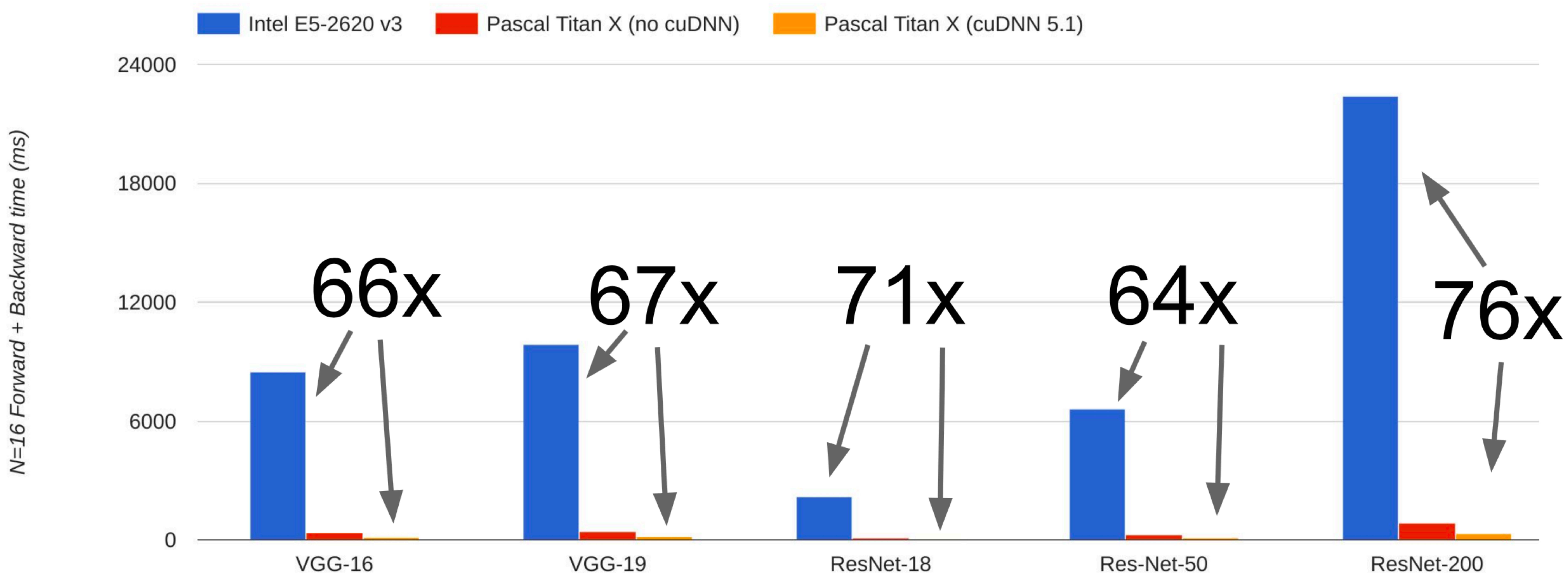
~ 2% improved performance in practice

Alternative: Multiple snapshots of the single model during training!

Improvement: Instead of using the actual parameter vector, keep a moving average of the parameter vector and use that at test time (Polyak averaging)

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

CPU vs. **GPU** (Why do we need Azure?)



Data from https://github.com/jcjohnson/cnn-benchmarks

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

Frameworks: Super quick overview

1. Easily **build computational graphs**

2. Easily **compute gradients** in computational graphs

3. **Run it all efficiently** on a GPU (weap cuDNN, cuBLAS, etc.)

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

Frameworks: Super quick overview

Core DNN Frameworks		
Caffe (UC Berkeley)	Caffe 2 (Facebook)	Puddle (Baidu)
Torch (NYU/Facebook)	**PyTorch** (Facebook)	CNTK (Microsoft)
Theano (U Montreal)	**TensorFlow** (Google)	MXNet (Amazon)

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

Frameworks: Super quick overview

Core DNN Frameworks		
Caffe (UC Berkeley)	Caffe 2 (Facebook)	Puddle (Baidu)
Torch (NYU/Facebook)	**PyTorch** (Facebook)	CNTK (Microsoft)
Theano (U Montreal)	**TensorFlow** (Google)	MXNet (Amazon)

Wrapper Libraries
Keras
TFLearn
TensorLayer
tf.layers
TF-Slim
tf.contrib.learn
Pretty Tensor

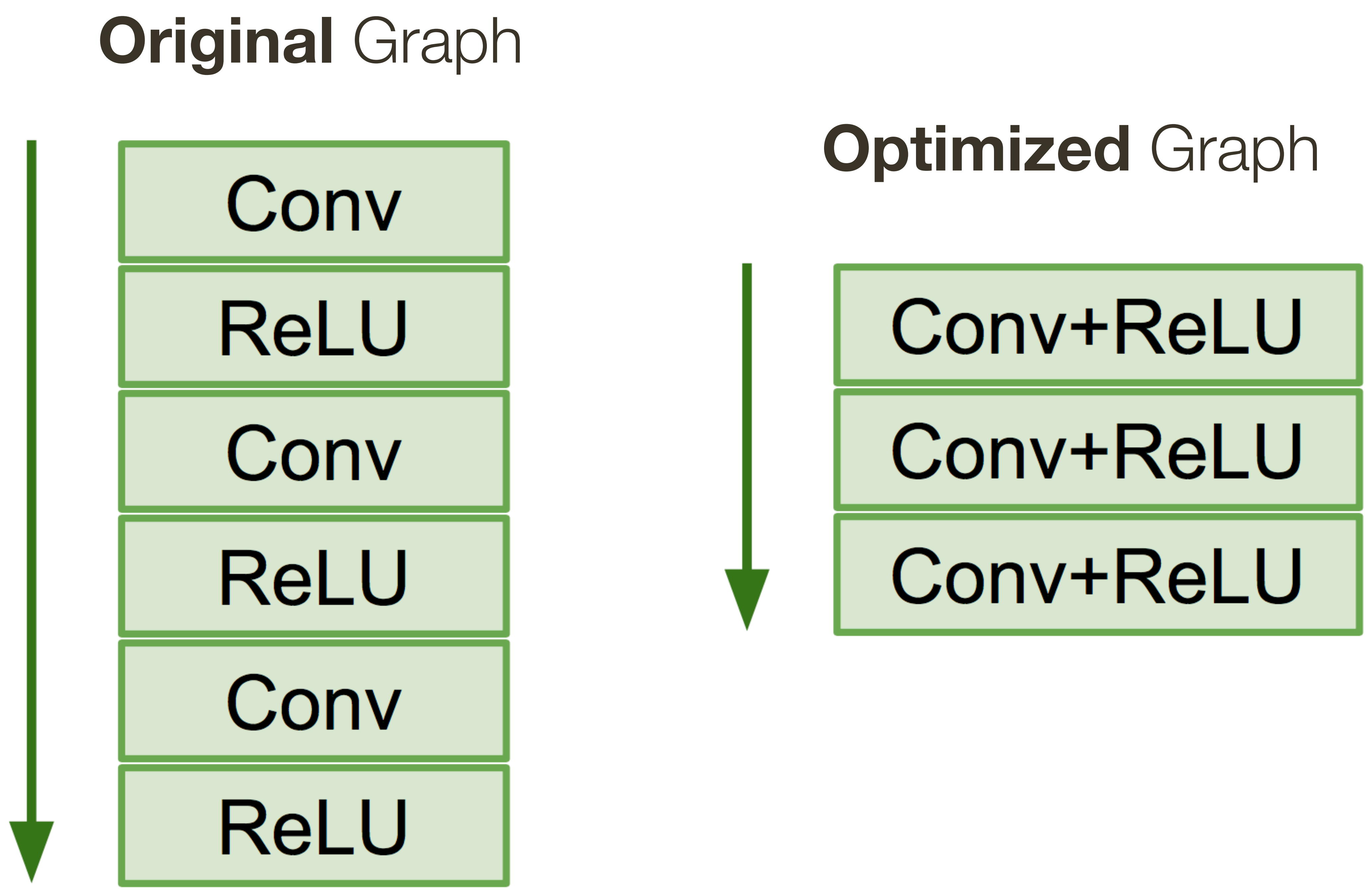
Frameworks: PyTorch vs. TensorFlow

Dynamic vs. Static computational graphs

Frameworks: PyTorch vs. TensorFlow

Dynamic vs. **Static** computational graphs

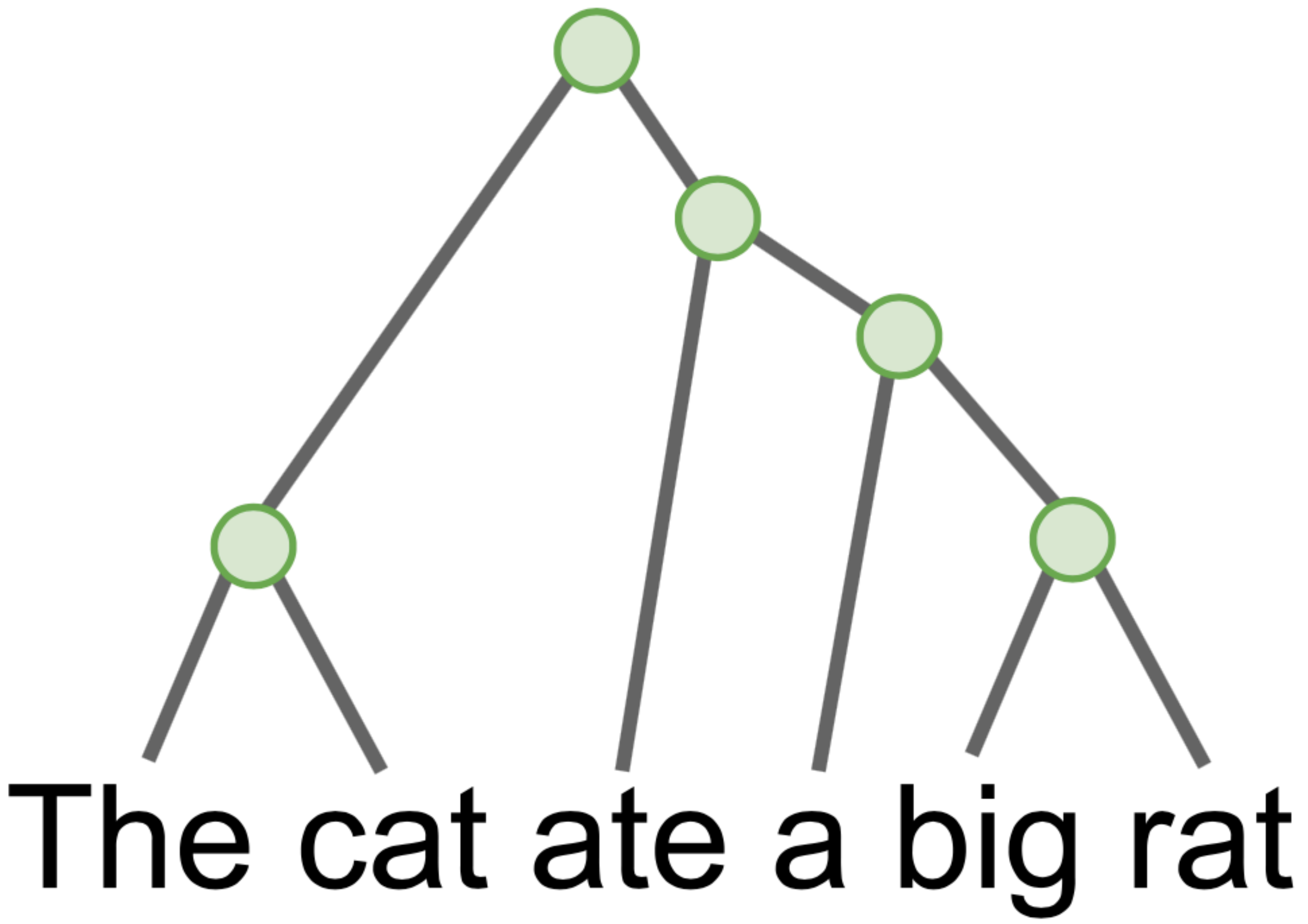
With static graphs, framework can **optimize** the graph for you before it runs!



Frameworks: PyTorch vs. TensorFlow

Dynamic vs. Static computational graphs

Graph building and execution is intertwined. Graph can be different for every sample.



PyTorch: Three levels of abstraction

Tensor: Imperative ndarray, but runs on GPU

Variable: Node in a computational graph; stores data and gradients

Module: A neural network layer; may store state or learnable weights

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