

<https://introml.mit.edu>

6.390 Intro to Machine Learning

Lecture 7: Convolutional Neural Networks

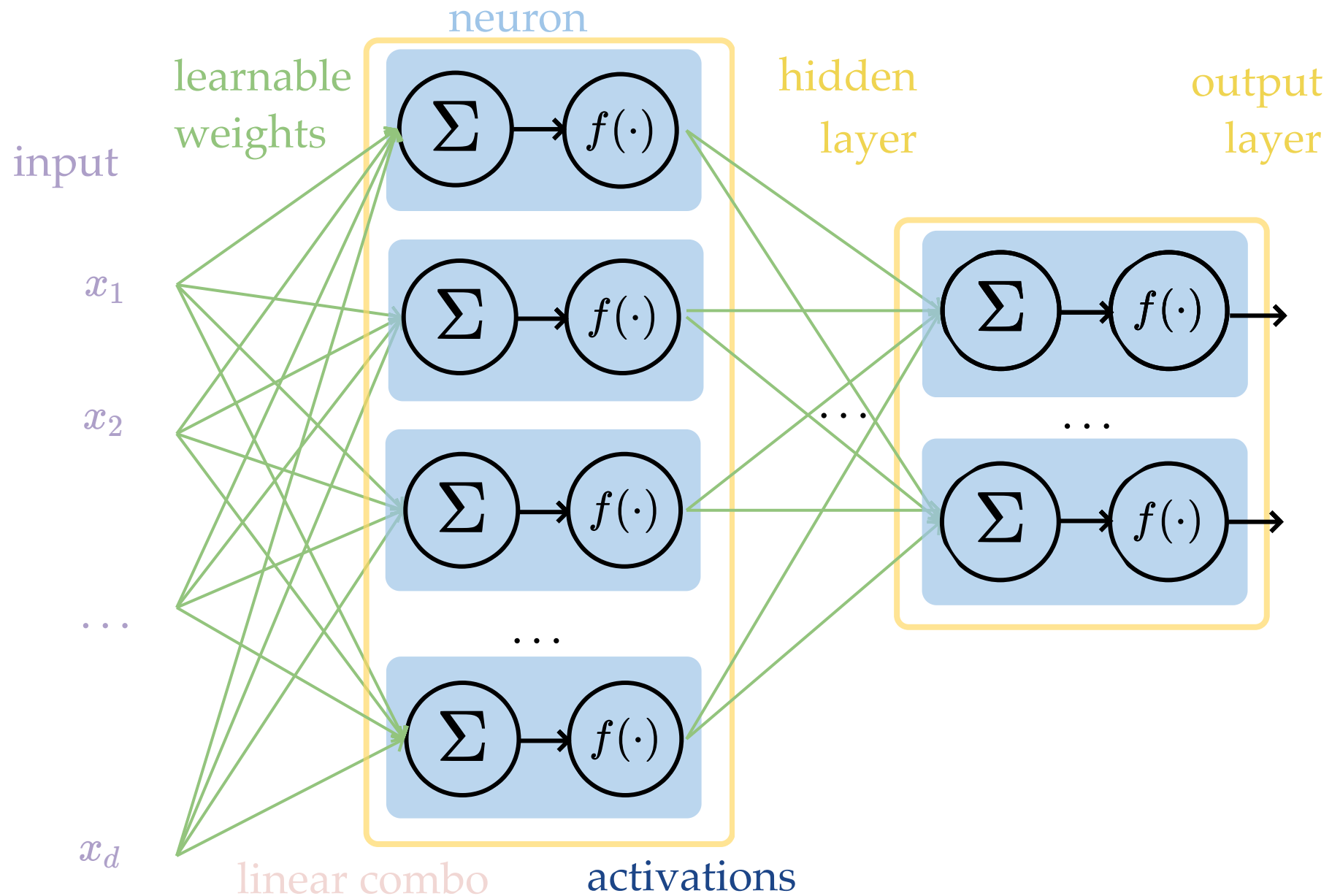
Shen Shen

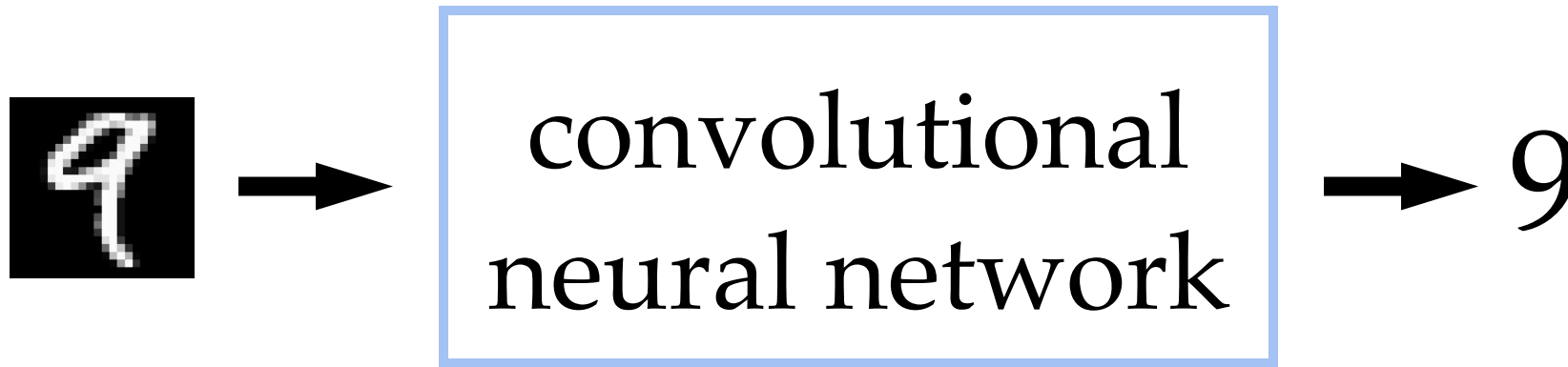
Oct 19, 2026

2:30pm, Room 10-250

[Slides and Lecture Recording](#)

Recall: a (fully-connected, feed-forward) neural network

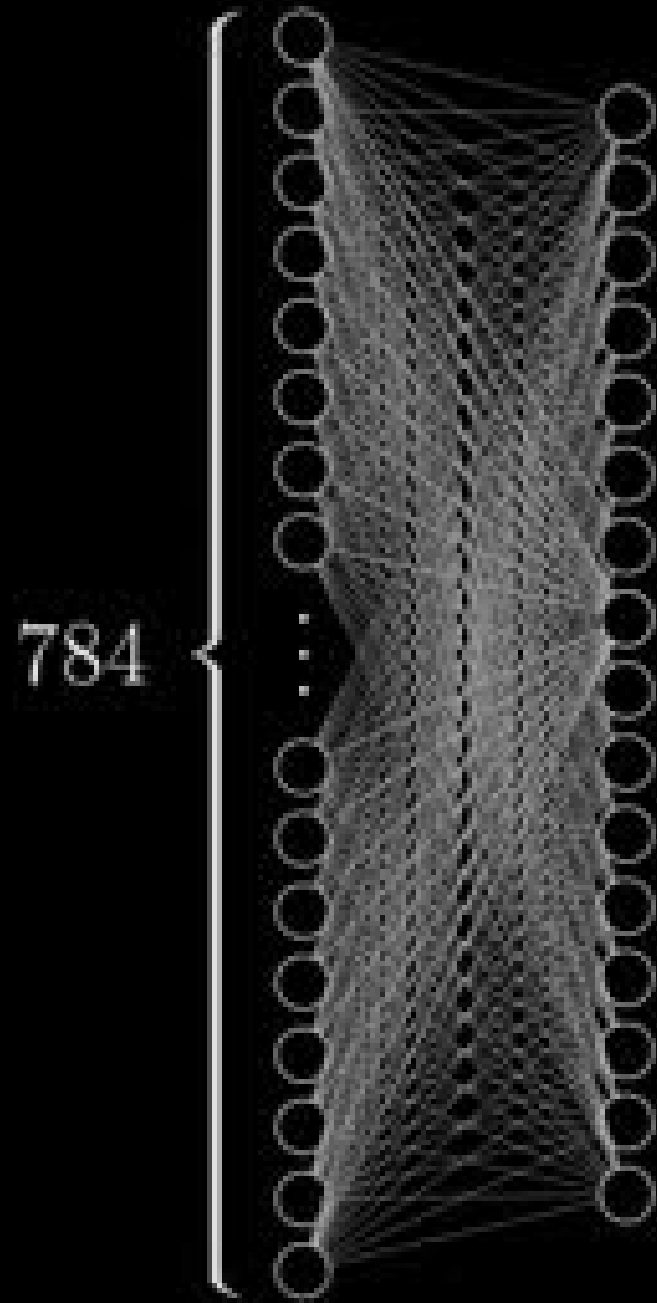




1. Why do we need a special network for images?
2. Why is CNN *the* special network for images?

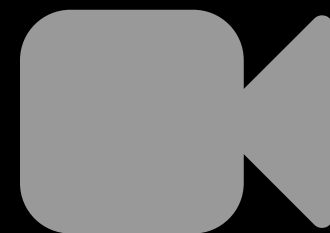
Outline

- Vision problem structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- (Case studies)



784×16 weights

16 biases



Why do we need a specialized network (hypothesis class)?



426-by-426
grayscale image

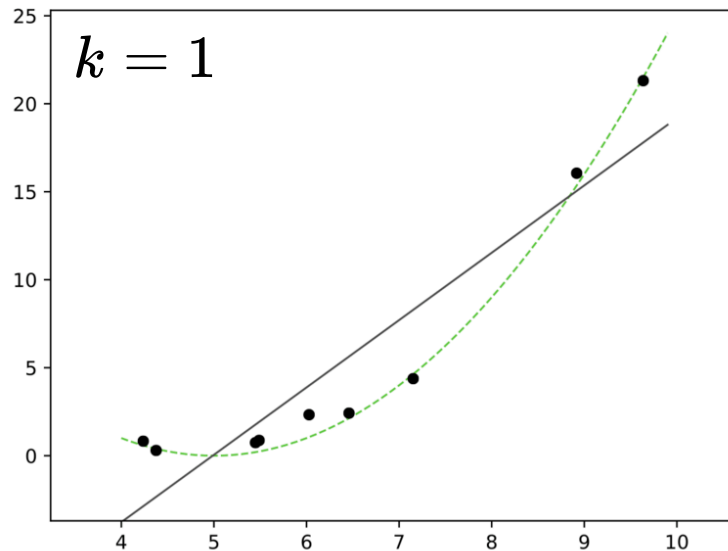
Use the same 2 hidden-layer network to predict what top-10 engineering school seal this image is. That takes $\sim 3\text{M}$ parameters.

For higher-resolution images, or more complex tasks, or larger networks, the number of parameters can grow very fast.

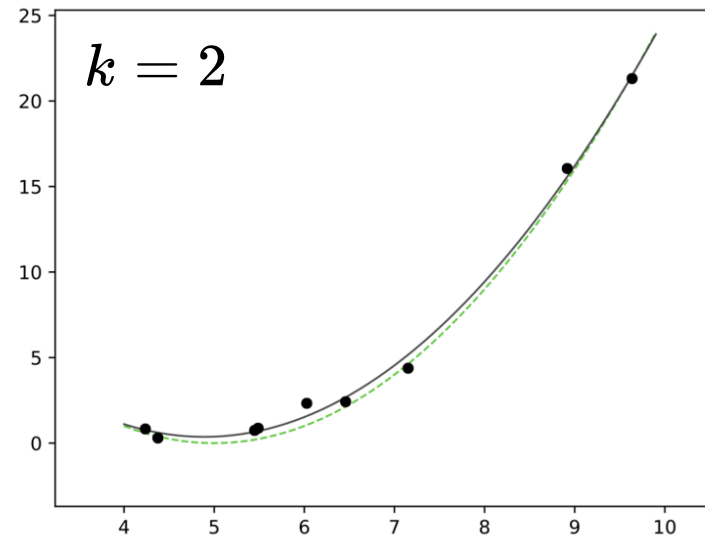
- Partly, fully-connected nets don't scale well for vision tasks
- More importantly, a carefully chosen hypothesis class helps fight overfitting

Recall: models with needless parameters tend to overfit

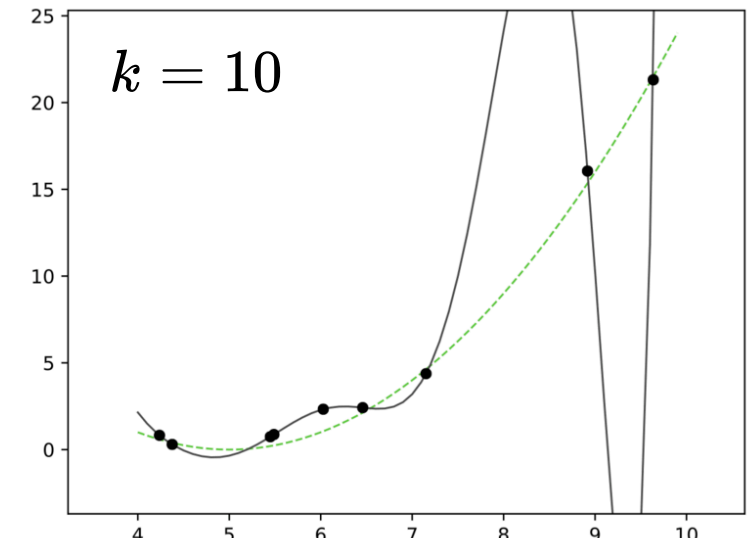
Underfitting



Appropriate



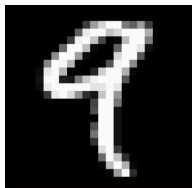
Overfitting



If we know the data is generated by the green curve, it's easy to choose the appropriate quadratic hypothesis class.

so... do we know anything about vision problems?

Why do we humans think

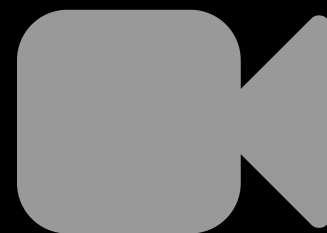
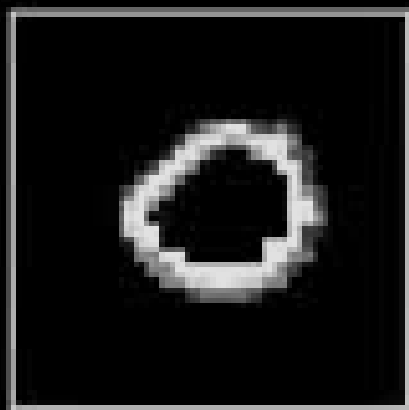


is a 9?

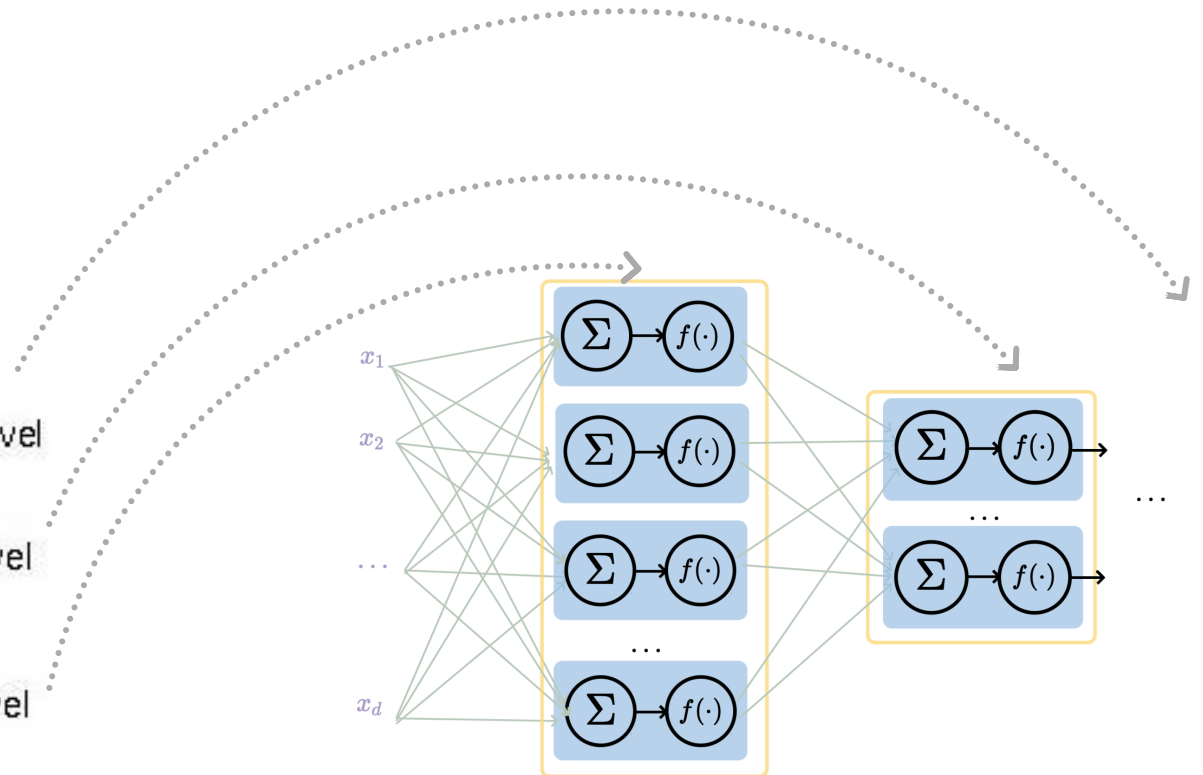
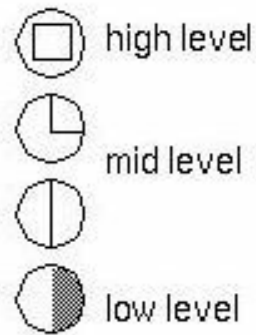
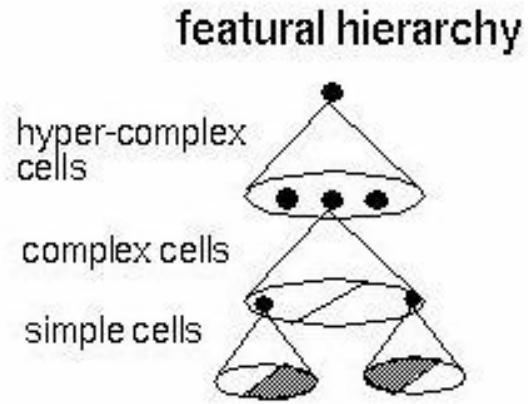
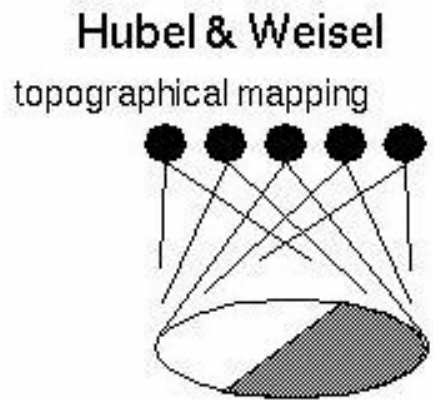
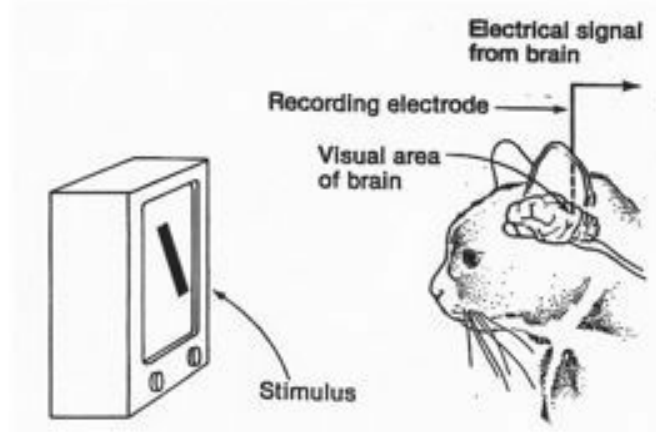
Why do we think any of



is a 9?

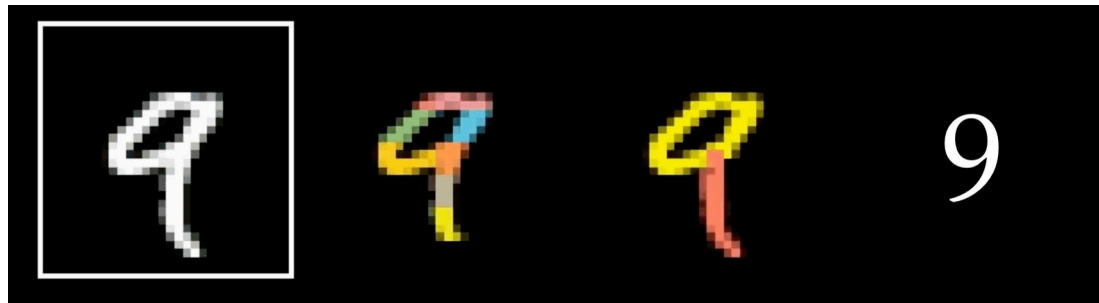


- Visual hierarchy

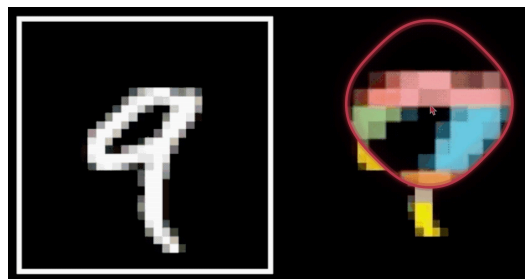


Layered structures are well-suited to model this hierarchical processing₀

- Visual hierarchy



- Spatial locality



- Translational invariance

CNN exploits

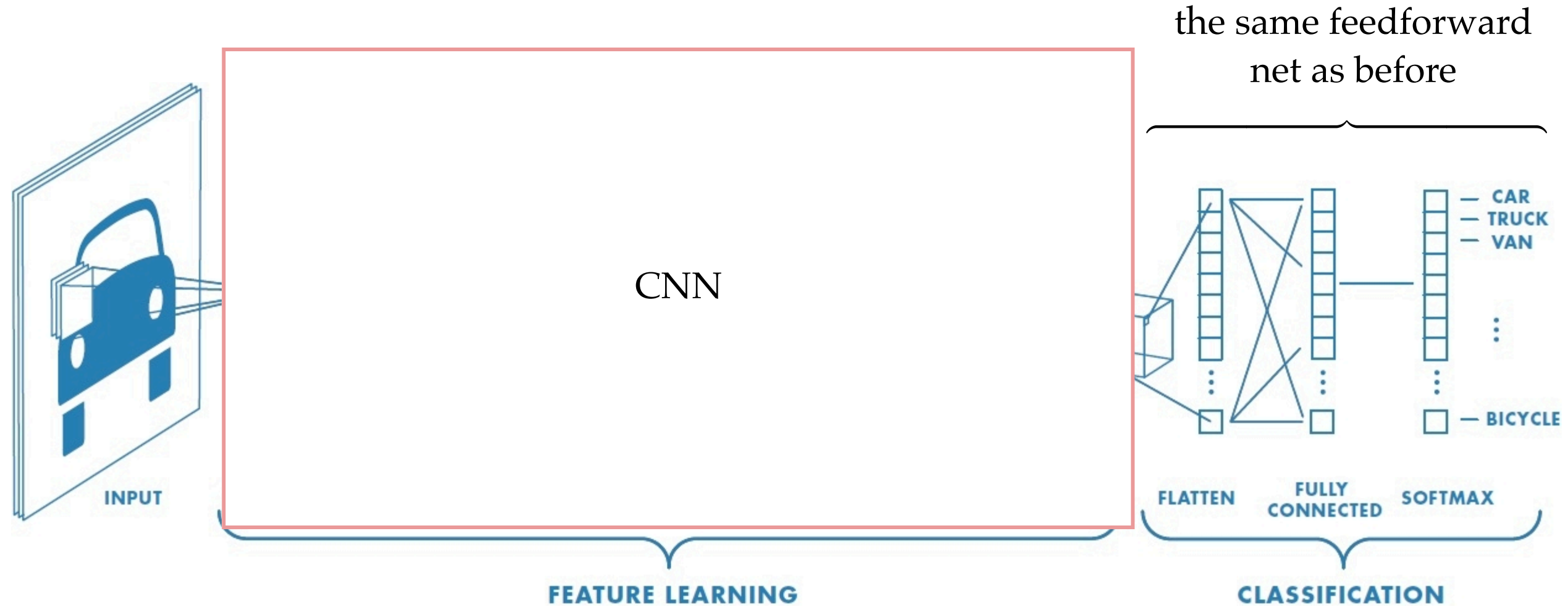
- Visual hierarchy
- Spatial locality
- Translational invariance

via

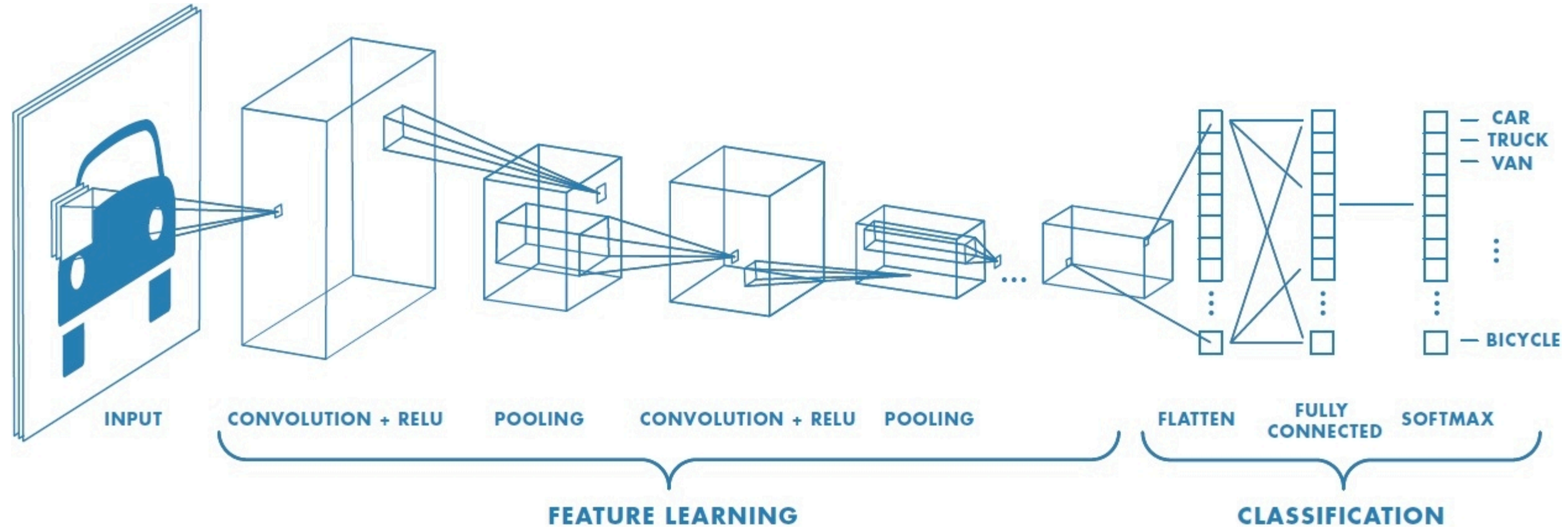
- Layered structure
- Convolution
- Pooling

to handle images efficiently and effectively.

typical CNN architecture for image classification



typical CNN architecture for image classification



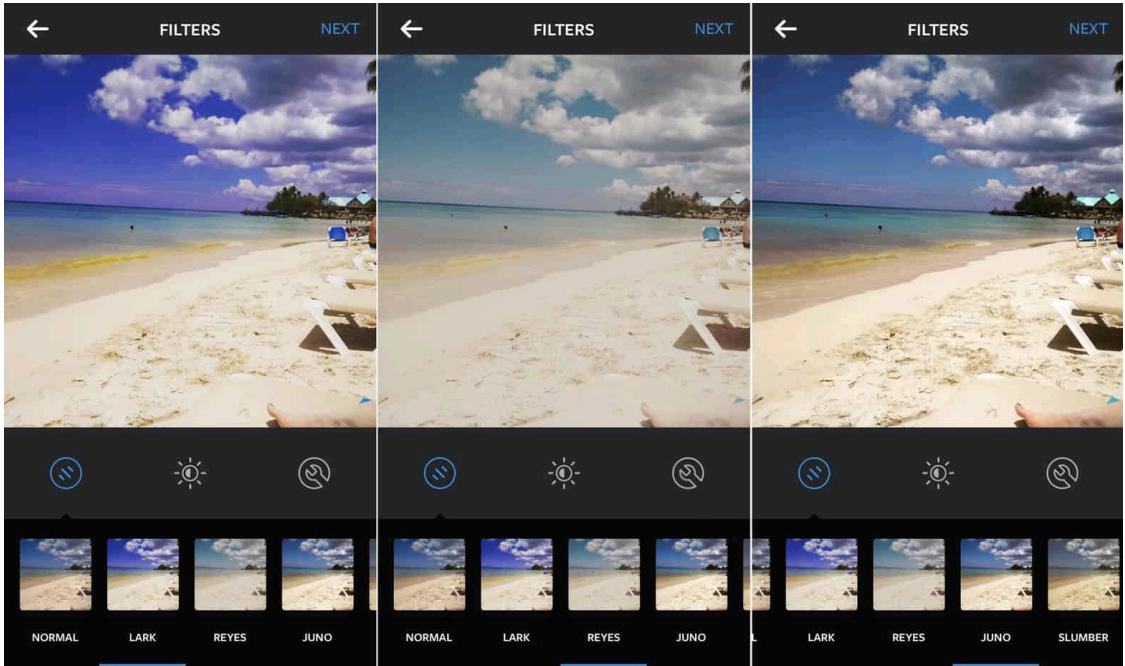
Outline

- Vision problem structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- (Case studies)

A convolutional layer might sound foreign,
but it's very similar to a fully-connected layer

Layer	Forward pass, <i>do</i>	Backward pass, <i>learn</i>	Design choices
fully-connected	dot-product, activation	neuron weights	neuron count, etc.
convolutional	convolution, activation	filter weights	conv specs, etc.

Convolution result:



[a filter is also called a kernel]
[we focus on the forward pass for now]

example: 1-dimensional convolution

input

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

filter

-1	1
----	---

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

convolved output

1			
---	--	--	--

example: 1-dimensional convolution

input

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

filter

-1	1
----	---

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

convolved output

1	-1		
---	----	--	--

example: 1-dimensional convolution

input

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

filter

-1	1
----	---

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

convolved output

1	-1	1	
---	----	---	--

example: 1-dimensional convolution

input

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

filter

-1	1
----	---

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

convolved output

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

convolution interpretation 1:

template matching

input

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

filter

-1	1
----	---

convolved
output

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

convolution interpretation 2:

input

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

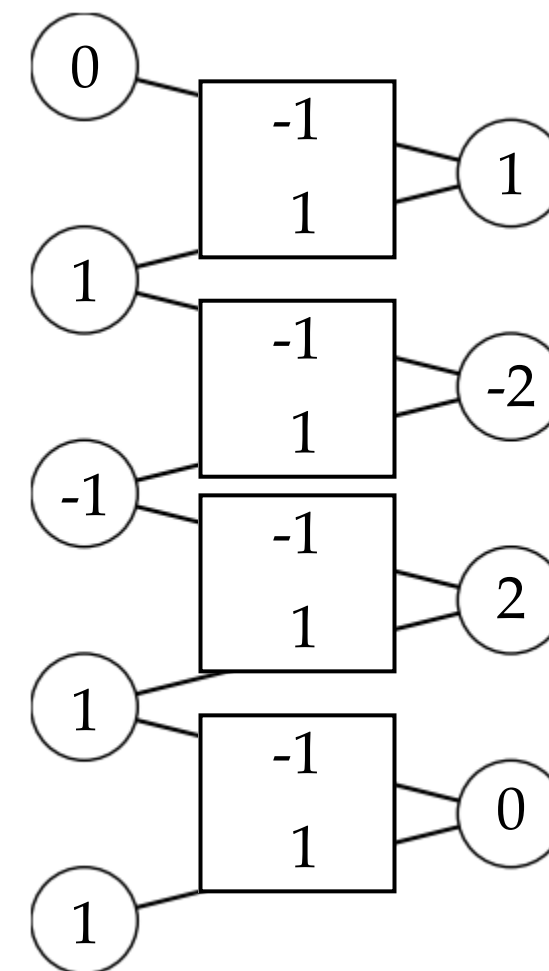
filter

-1	1
----	---

convolved
output

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

"look" *locally* through the filter
this local region = *receptive field*

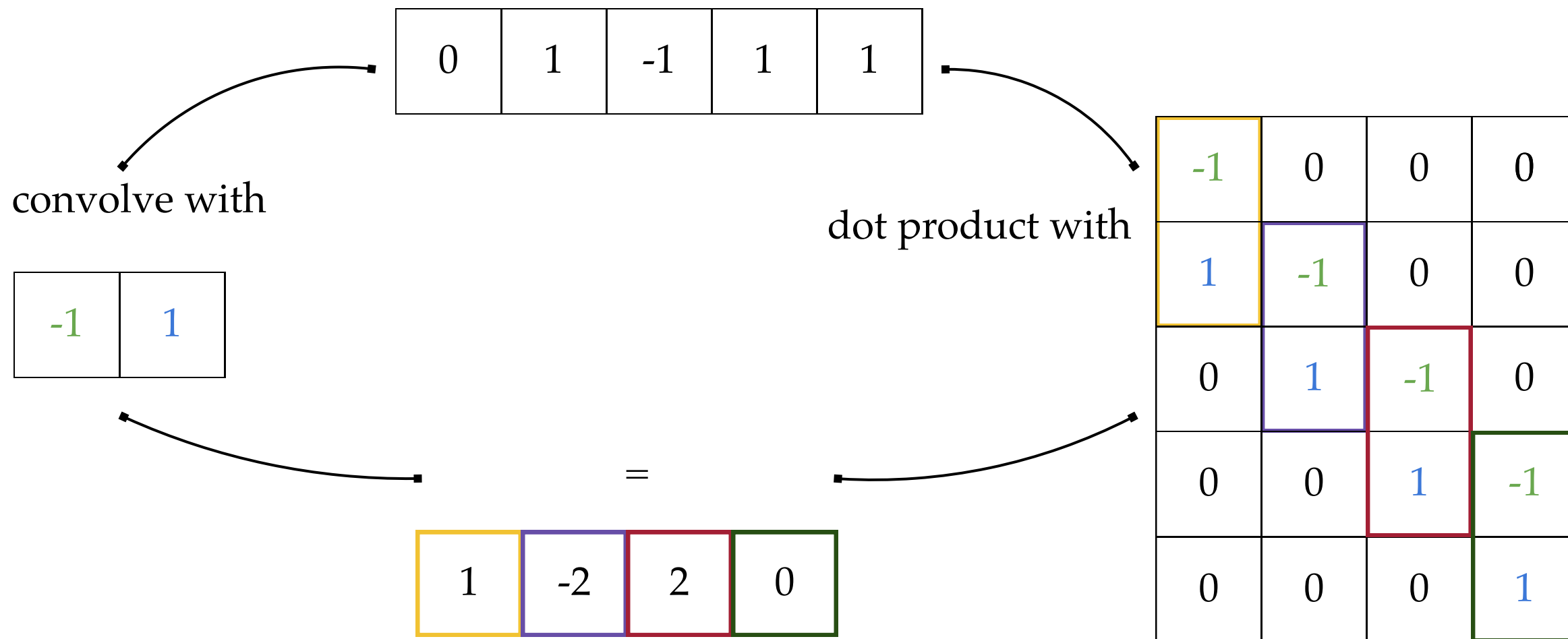


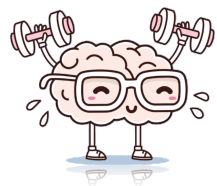
input

output

convolution interpretation 3:

sparsely connected layer
with parameter sharing





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



convolve with

1



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



dot product with

$I_{5 \times 5}$



convolution interpretation 4:

translational equivariance

input

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

filter

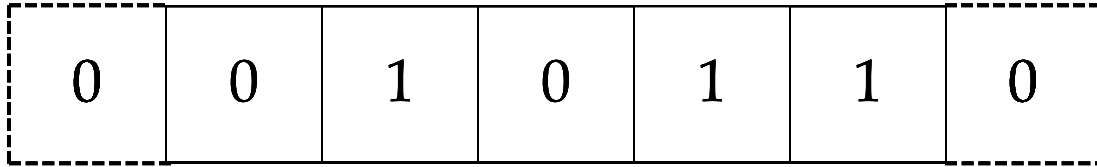
0	1
---	---

convolved
output

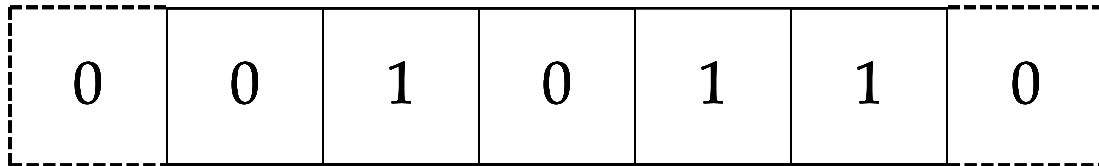
1	0	1	0
---	---	---	---

hyperparameters: padding, stride, filter size

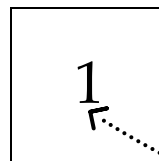
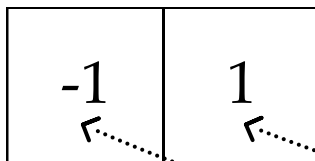
- Zero-padding



- Stride (e.g. stride of 2)



- Filter size (e.g. we saw these two in 1-d)



these weights are what
CNNs learn

example: 2-dimensional convolution

input

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

filter

0	1	2
2	2	0
0	1	2

convolved output

12	12	17
10	17	19
9	6	14

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	3 ₀	2 ₁	1 ₂	0
0	0 ₂	1 ₂	3 ₀	1
3	1 ₀	2 ₁	2 ₂	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	3	2 ₀	1 ₁	0 ₂
0	0	1 ₂	3 ₂	1 ₀
3	1	2 ₀	2 ₁	3 ₂
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	3	2	1	0
0 ₀	0 ₁	1 ₂	3	1
3 ₂	1 ₂	2 ₀	2	3
2 ₀	0 ₁	0 ₂	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	3	2	1	0
0	0 ₀	1 ₁	3 ₂	1
3	1 ₂	2 ₂	2 ₀	3
2	0 ₀	0 ₁	2 ₂	2
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	3	2	1	0
0	0	1 ₀	3 ₁	1 ₂
3	1	2 ₂	2 ₂	3 ₀
2	0	0 ₀	2 ₁	2 ₂
2	0	0	0	1

12	12	17
10	17	19
9	6	14

3	3	2	1	0
0	0	1	3	1
3 ₀	1 ₁	2 ₂	2	3
2 ₂	0 ₂	0 ₀	2	2
2 ₀	0 ₁	0 ₂	0	1

12	12	17
10	17	19
9	6	14

3	3	2	1	0
0	0	1	3	1
3	1 ₀	2 ₁	2 ₂	3
2	0 ₂	0 ₂	2 ₀	2
2	0 ₀	0 ₁	0 ₂	1

12	12	17
10	17	19
9	6	14

3	3	2	1	0
0	0	1	3	1
3	1	2 ₀	2 ₁	3 ₂
2	0	0 ₂	2 ₂	2 ₀
2	0	0 ₀	0 ₁	1 ₂

12	12	17
10	17	19
9	6	14

stride of 2

input

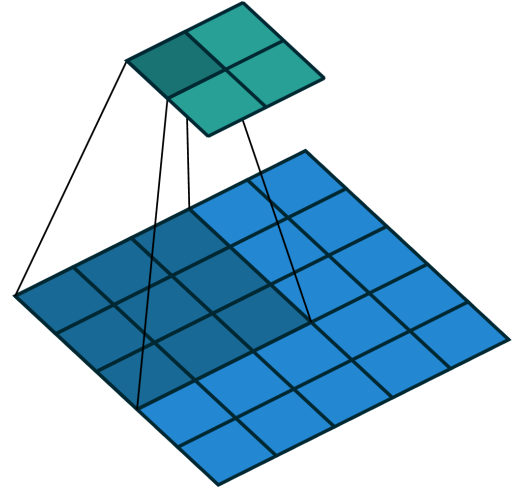
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

filter

0	1	2
2	2	0
0	1	2

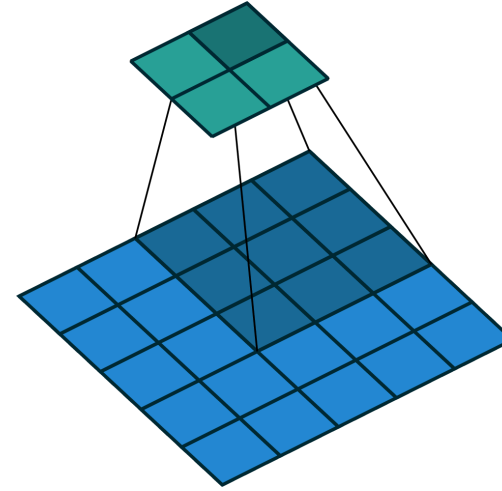
output

12	17
9	14



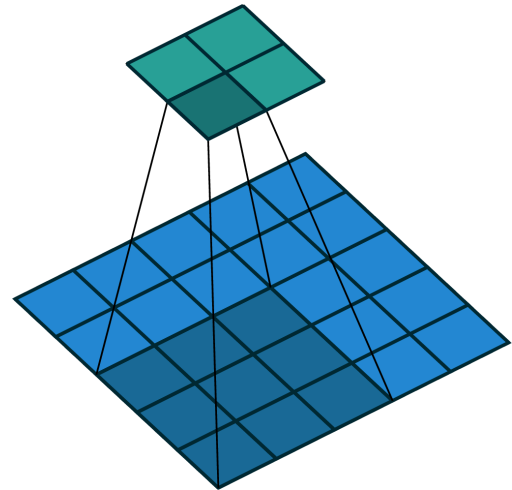
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12	17
9	14



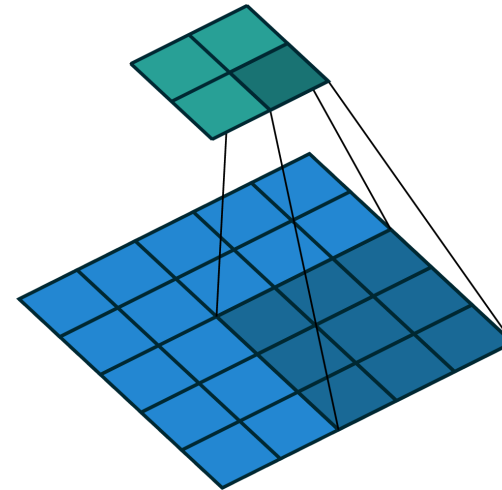
3	3	2_0	1_1	0_2
0	0	1_2	3_2	1_0
3	1	2_0	2_1	3_2
2	0	0	2	2
2	0	0	0	1

12	17
9	14



3	3	2	1	0
0	0	1	3	1
3_0	1_1	2_2	2	3
2_2	0_2	0_0	2	2
2_0	0_1	0_2	0	1

12	17
9	14



3	3	2	1	0
0	0	1	3	1
3	1	2_0	2_1	3_2
2	0	0_2	2_2	2_0
2	0	0_0	0_1	1_2

12	17
9	14

stride of 2, with padding of size 1

input

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

filter

0	1	2
2	2	0
0	1	2

output

6	17	3
8	17	13
6	4	4

0 ₀	0 ₁	0 ₂	0	0	0	0
0 ₂	3 ₂	3 ₀	2	1	0	0
0 ₀	0 ₁	0 ₂	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0 ₀	0 ₁	0 ₂	0	0
0	3	3 ₂	2 ₂	1 ₀	0	0
0	0	0 ₀	1 ₁	3 ₂	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0 ₀	0 ₁	0 ₂
0	3	3	2	1 ₂	0 ₂	0 ₀
0	0	0	1	3 ₀	1 ₁	0 ₂
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0 ₀	0 ₁	0 ₂	1	3	1	0
0 ₂	3 ₂	1 ₀	2	2	3	0
0 ₀	2 ₁	0 ₂	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0 ₀	1 ₁	3 ₂	1	0
0	3	1 ₂	2 ₂	2 ₀	3	0
0	2	0 ₀	0 ₁	2 ₂	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3 ₀	1 ₁	0 ₂
0	3	1	2	2 ₂	3 ₂	0 ₀
0	2	0	0	2 ₀	2 ₁	0 ₂
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0 ₀	2 ₁	0 ₂	0	2	2	0
0 ₂	2 ₂	0 ₀	0	0	1	0
0 ₀	0 ₁	0 ₂	0	0	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0 ₀	0 ₁	2 ₂	2	0
0	2	0 ₂	0 ₂	0 ₀	1	0
0	0	0 ₀	0 ₁	0 ₂	0	0

6	17	3
8	17	13
6	4	4

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2 ₀	2 ₁	0 ₂
0	2	0	0	0 ₂	1 ₂	0 ₀
0	0	0	0	0 ₀	0 ₁	0 ₂

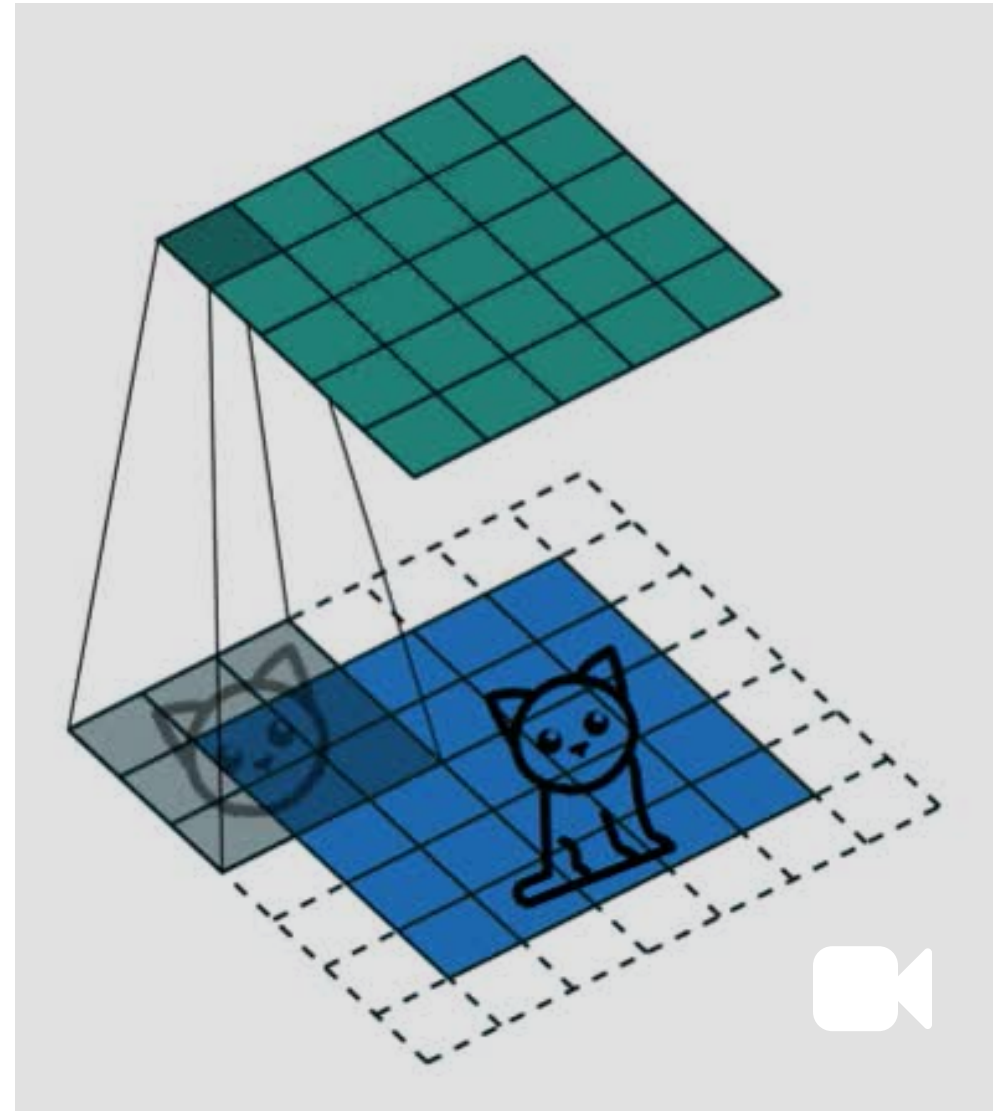
6	17	3
8	17	13
6	4	4

hyperparameters in 2d

<https://shenshen.mit.edu/demos/cnn/hyperparameters.html>

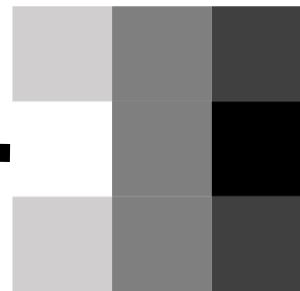
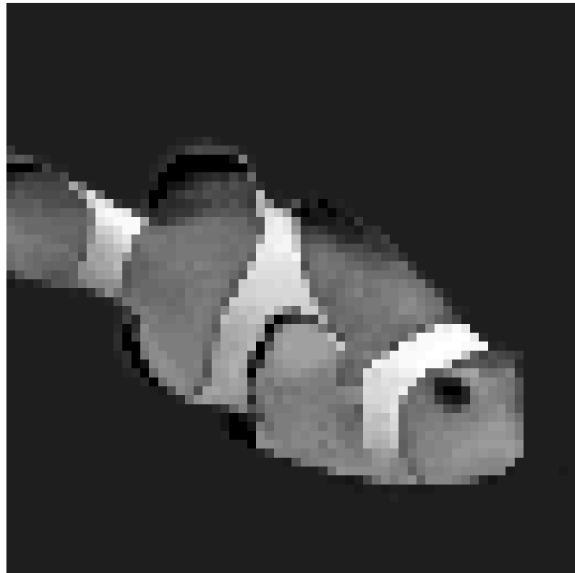
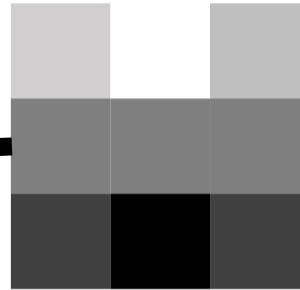
quick summary: convolution interpretations

- Look locally (sparse connections)
- Parameter sharing
- Template matching
- Translational equivariance



hand-designed filters (e.g. Sobel)

filter 1

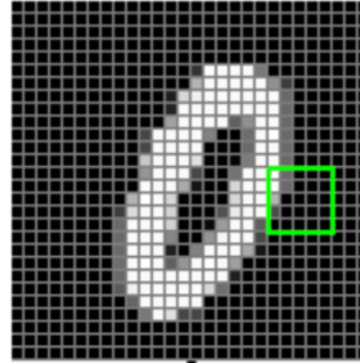


filter 2

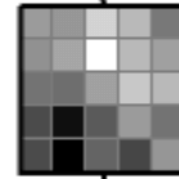
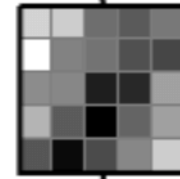
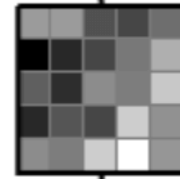
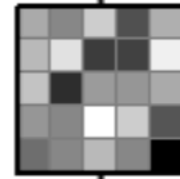
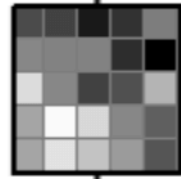
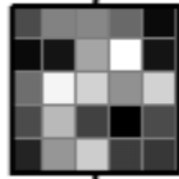
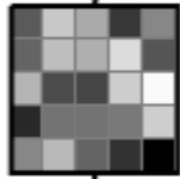


learned filters detect many patterns

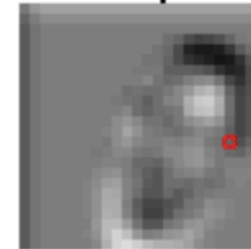
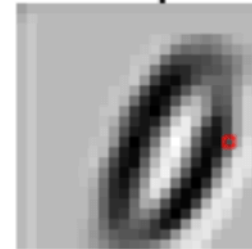
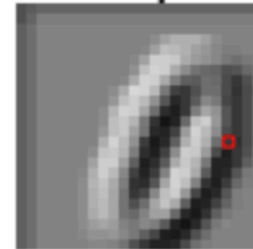
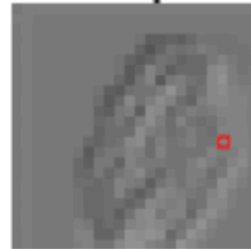
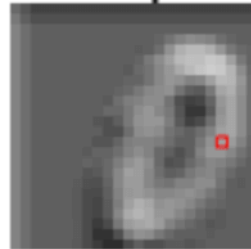
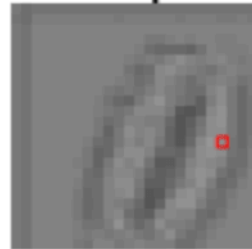
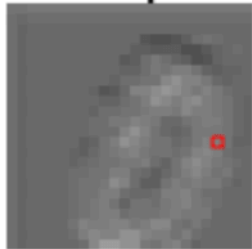
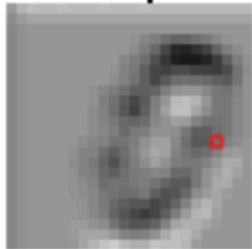
input



filters

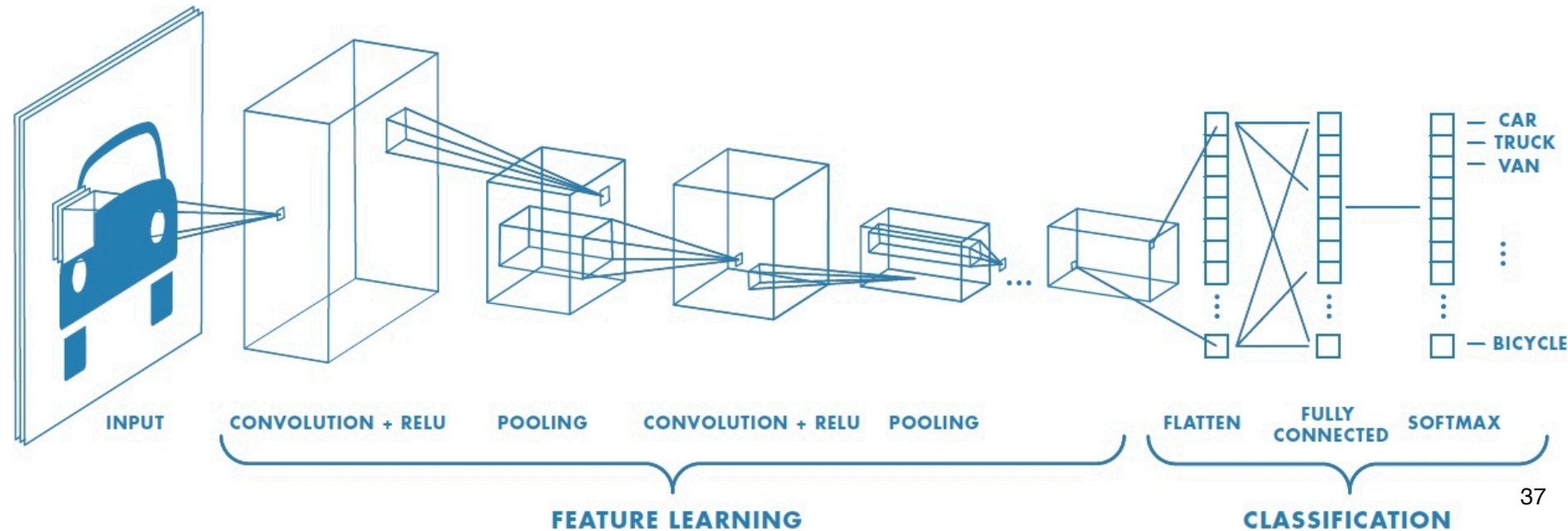


output



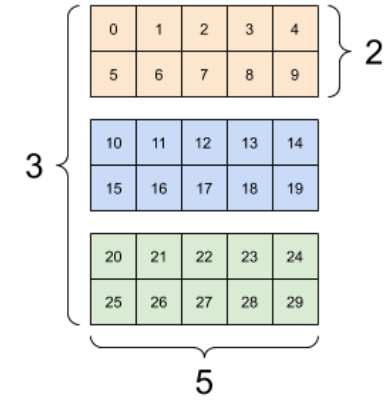
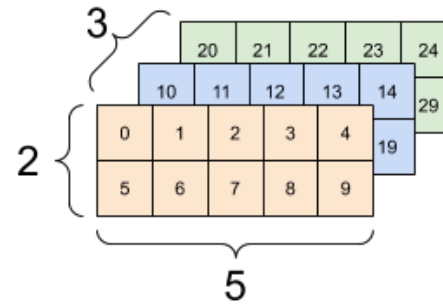
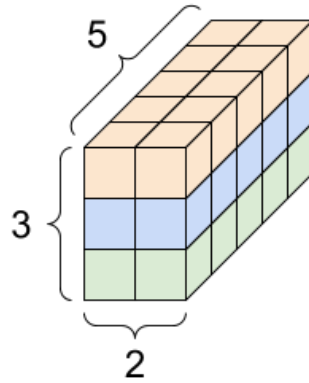
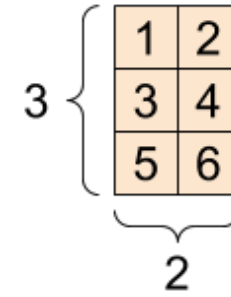
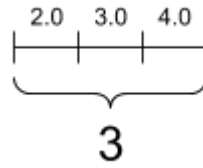
Outline

- Vision problem structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- (Case studies)



A tender intro to tensors:

4



color images and channels



red



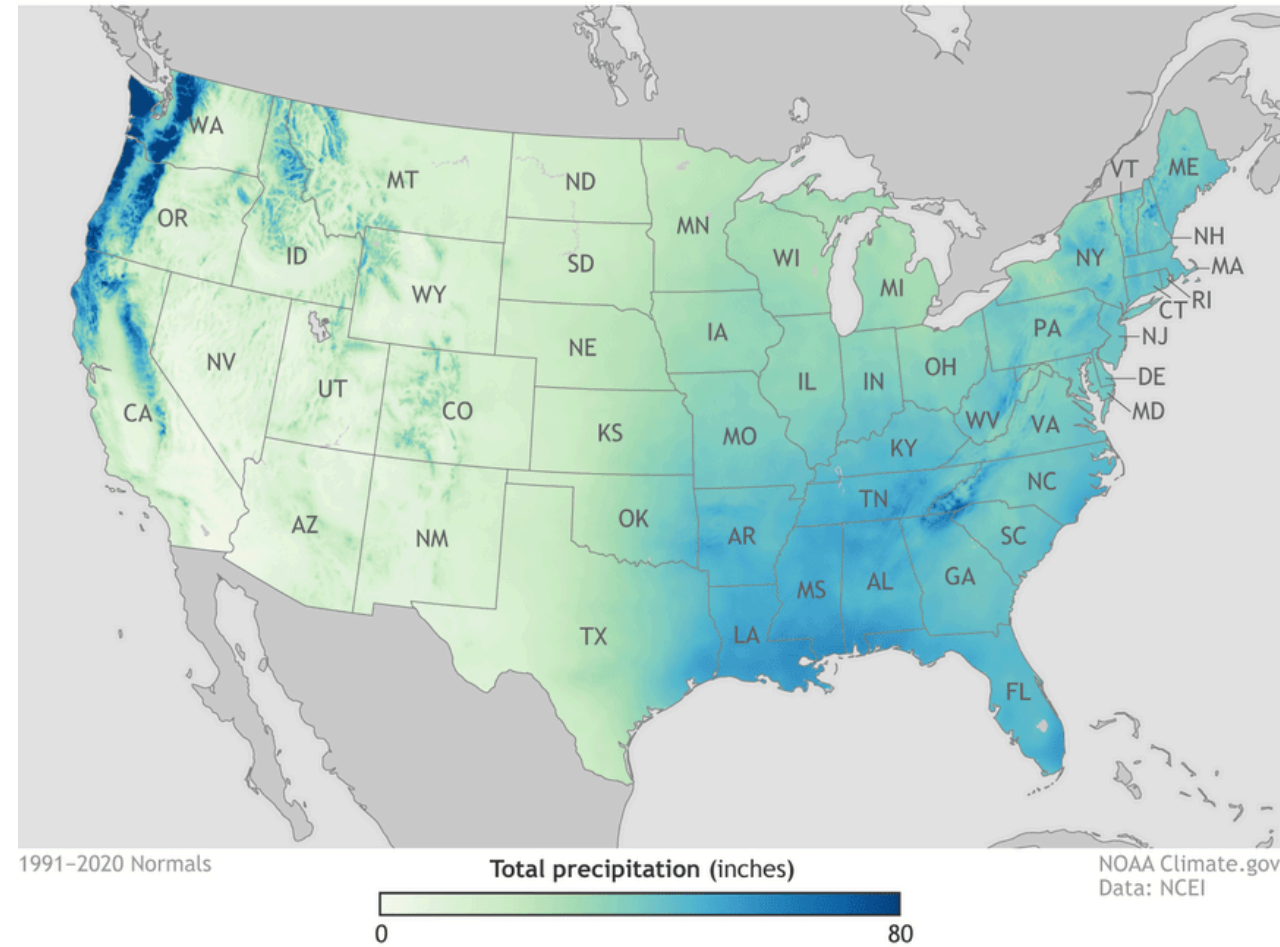
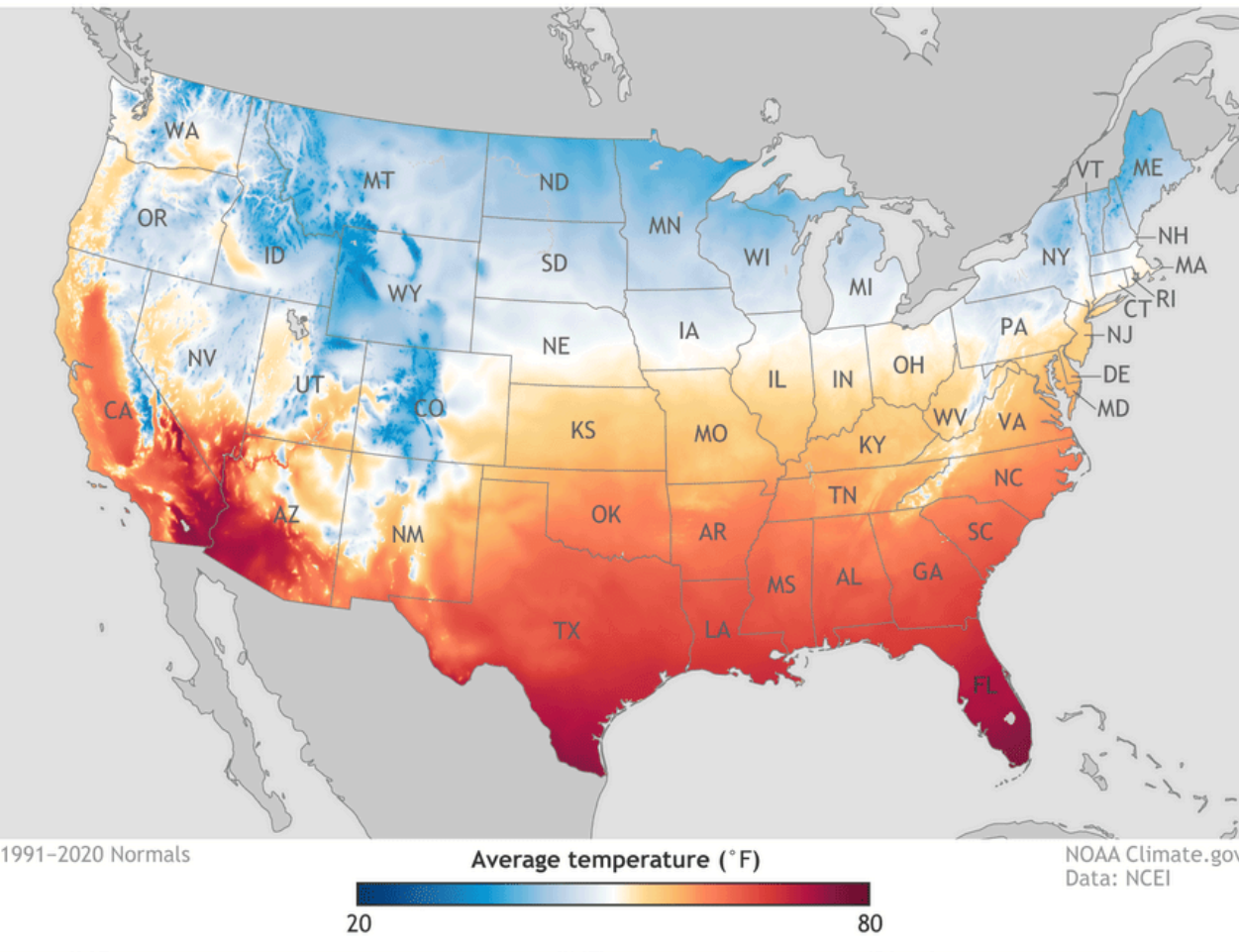
green



blue

each channel is a *complete but independent* view of the same scene

like when we think of weather:



so channels are often referred to as *feature maps*

3d tensors from color channels



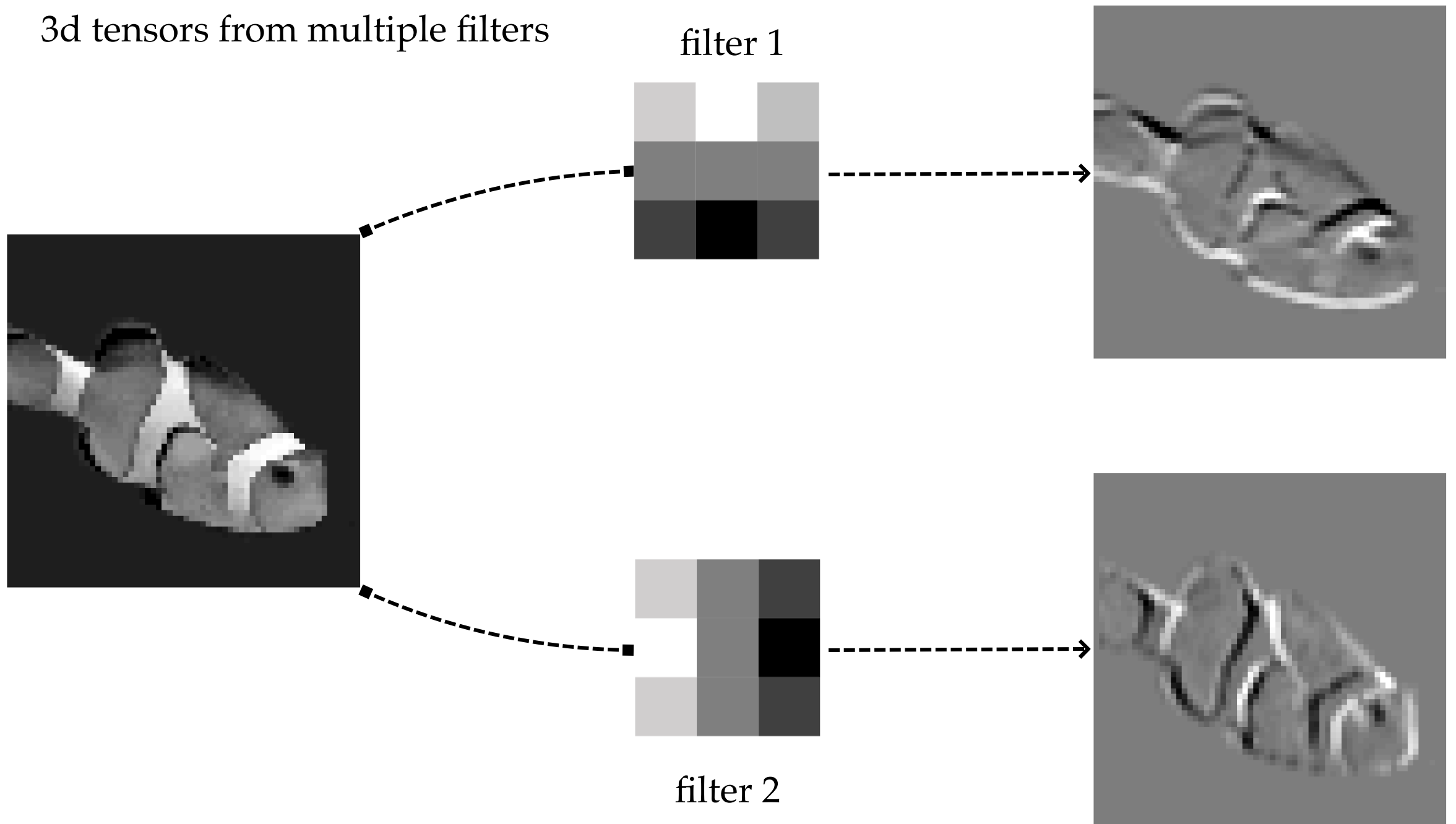
image
height



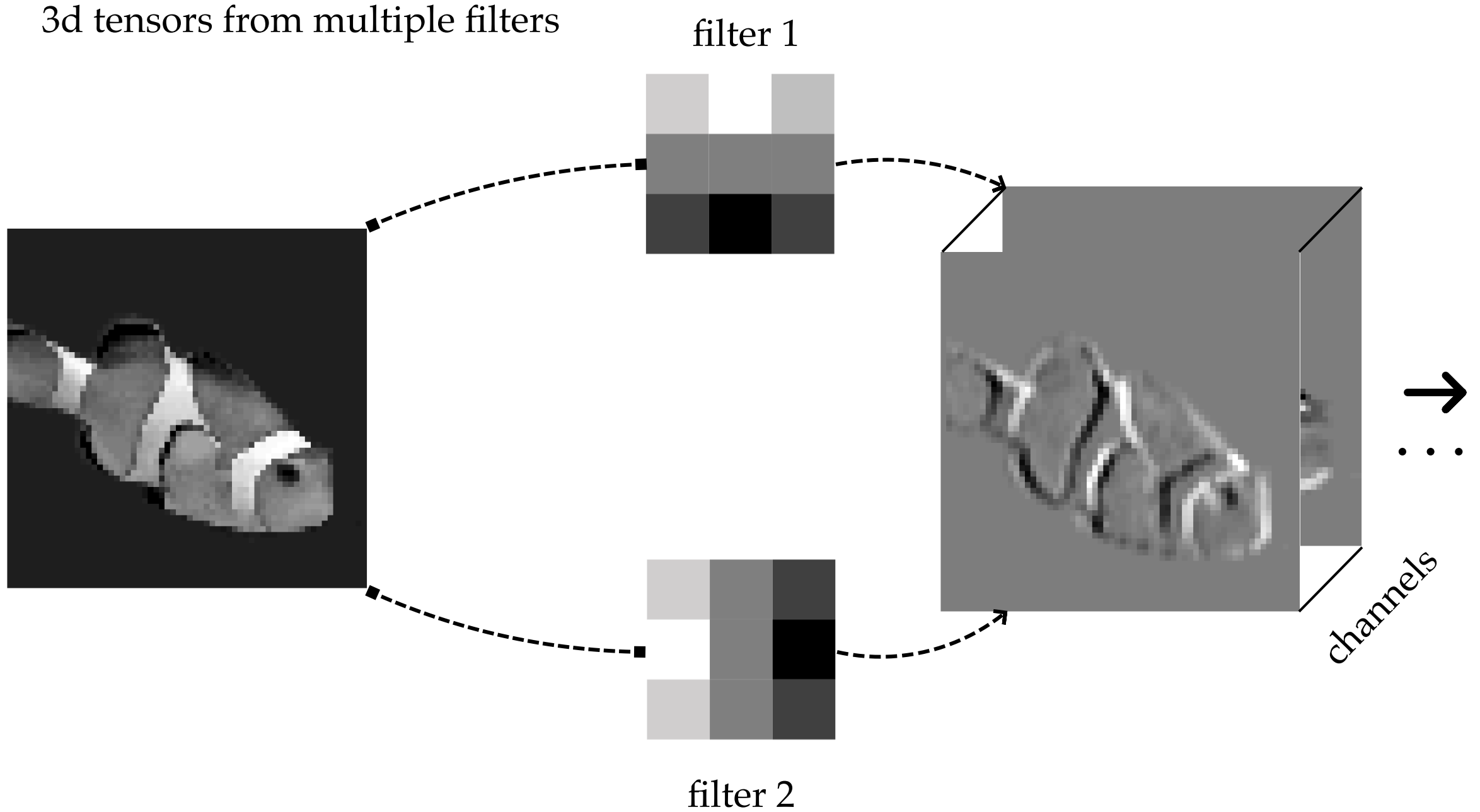
image channels

image width

3d tensors from multiple filters



3d tensors from multiple filters

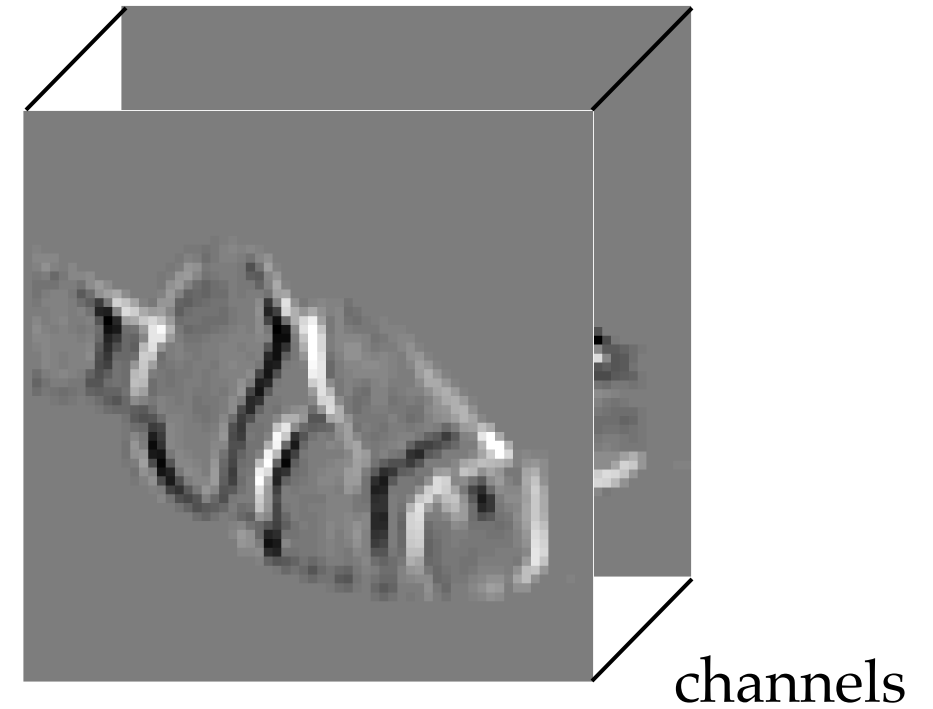


where do channels come from?

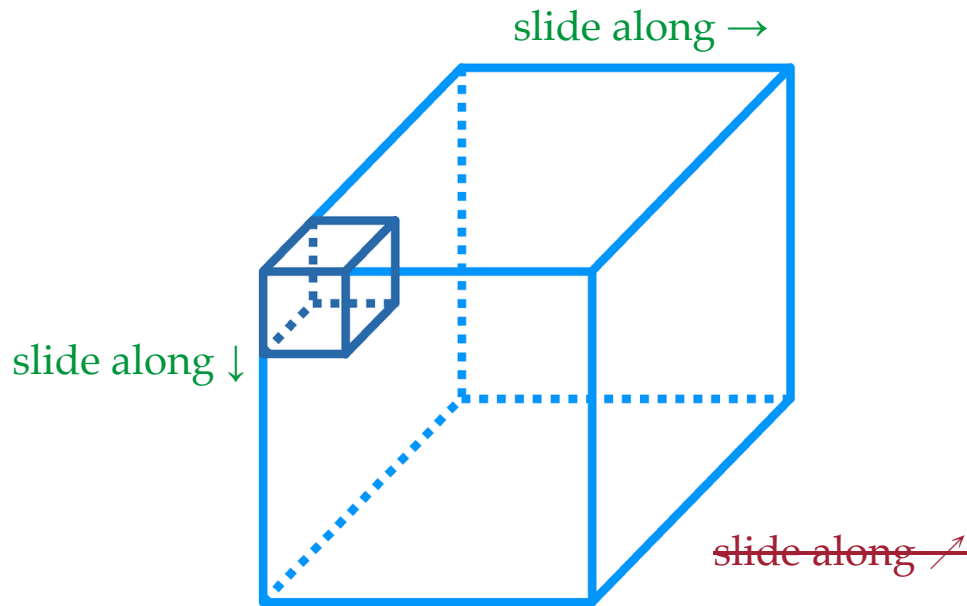
1. color input



2. multiple filters $\rightarrow$ multiple channels

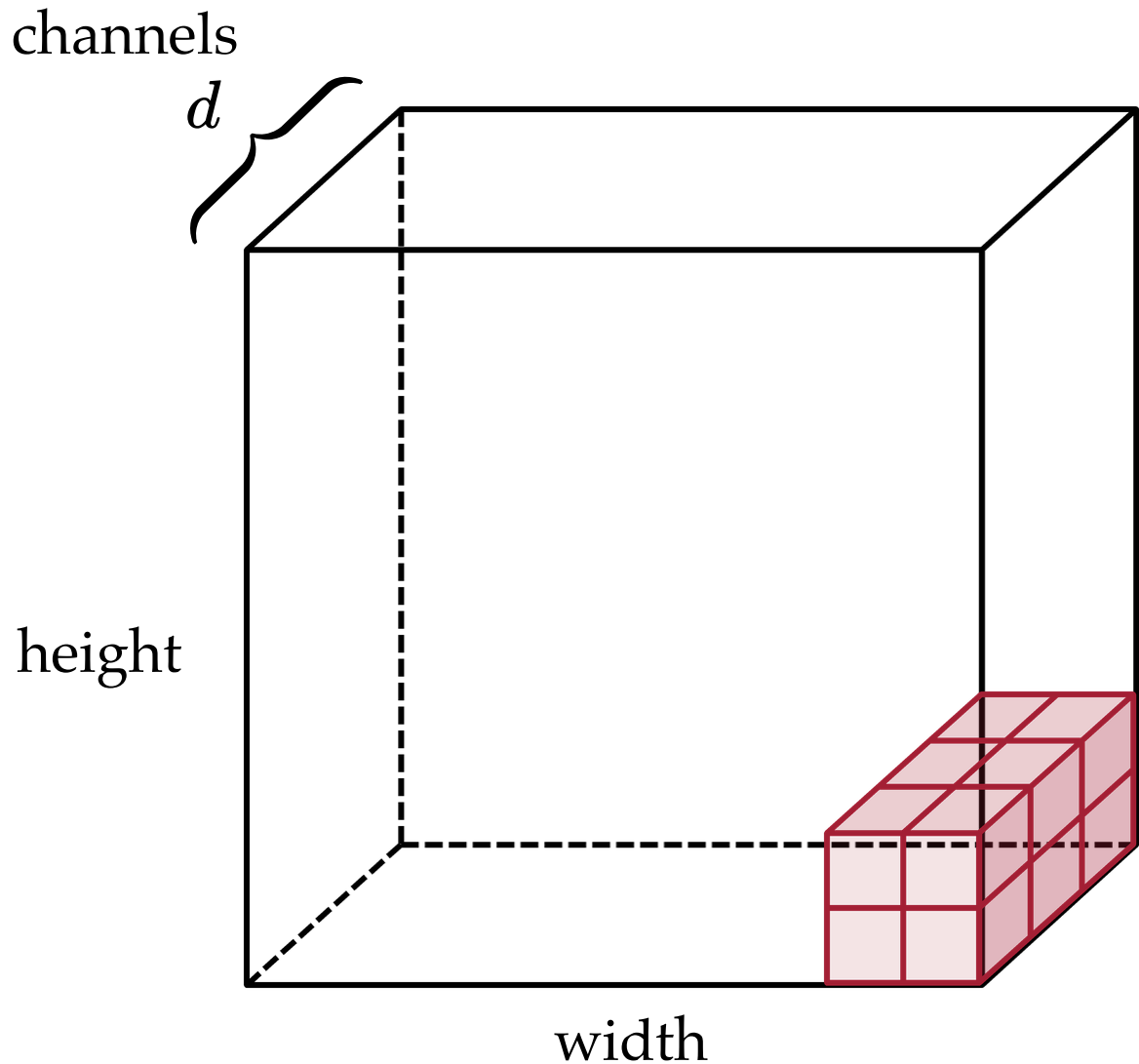


Why we *don't* typically do 3d convolution



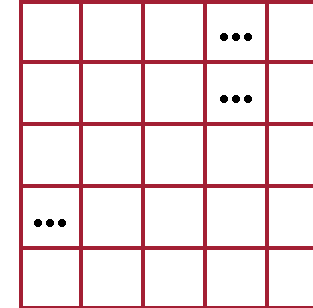
Convolution **shares** weights across **shifted** positions
shifting makes sense spatially (a cat can be anywhere)
but not across channels (red $\neq$ shifted green)

full-depth 2d convolution

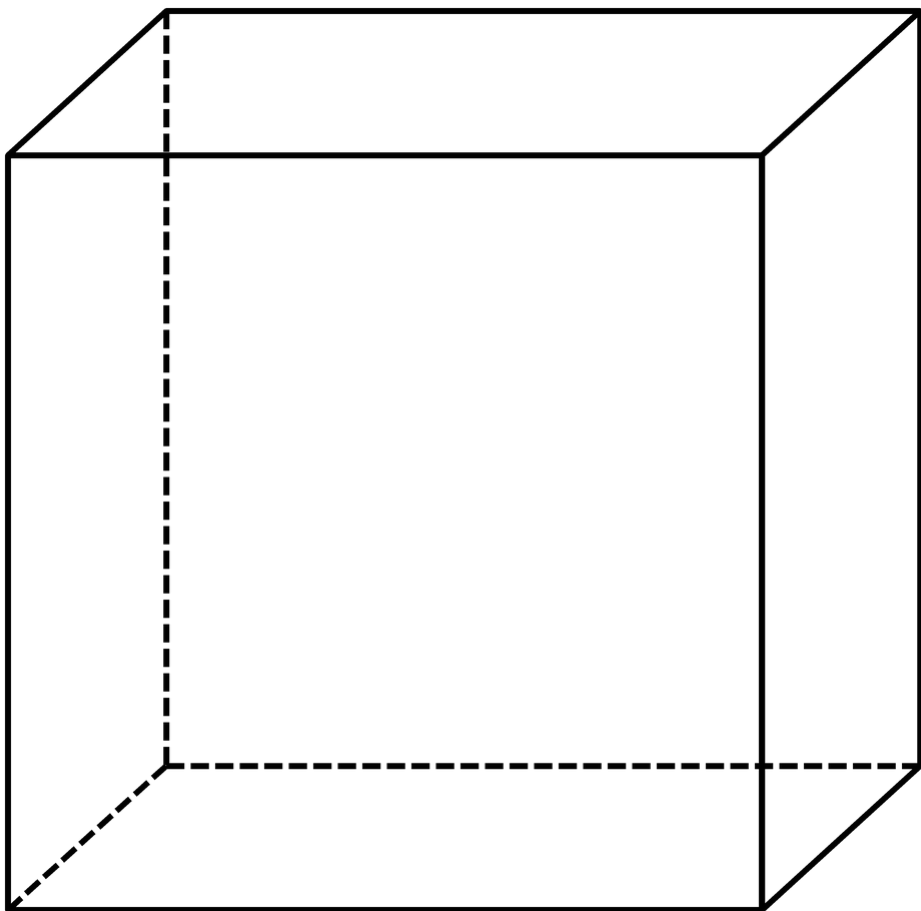


- 3d tensor input, d channels
- 3d tensor filter, same d channels deep
- 2d convolution, 2d output

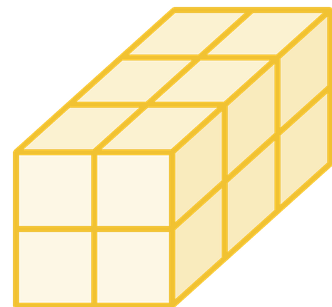
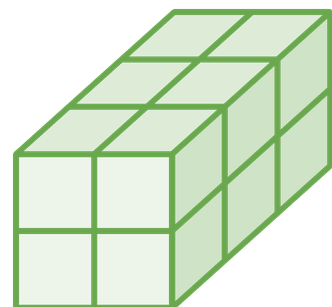
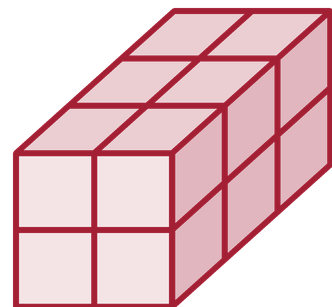
output



input tensor

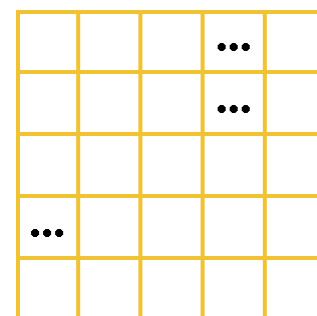
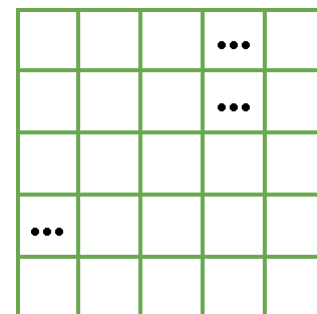
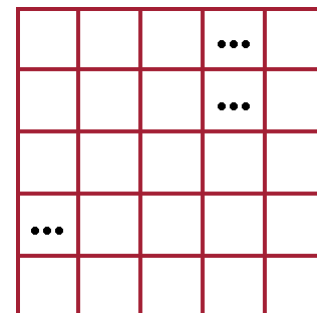


multiple filters



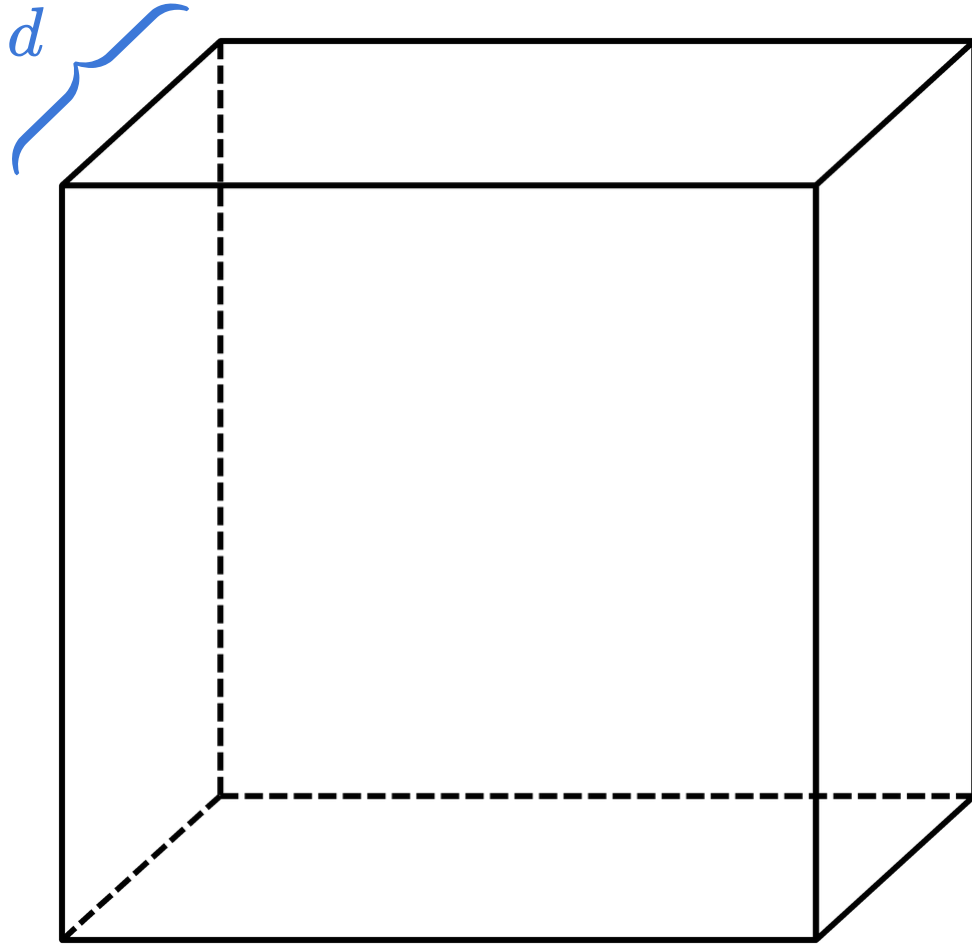
...

multiple output matrices

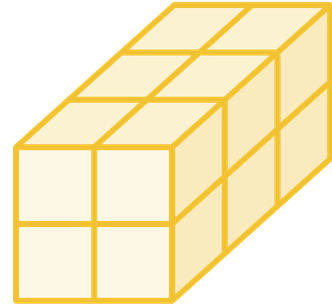
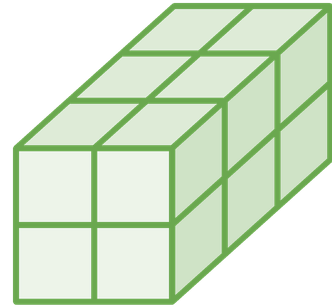
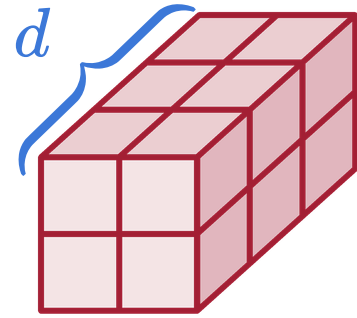


...

input tensor



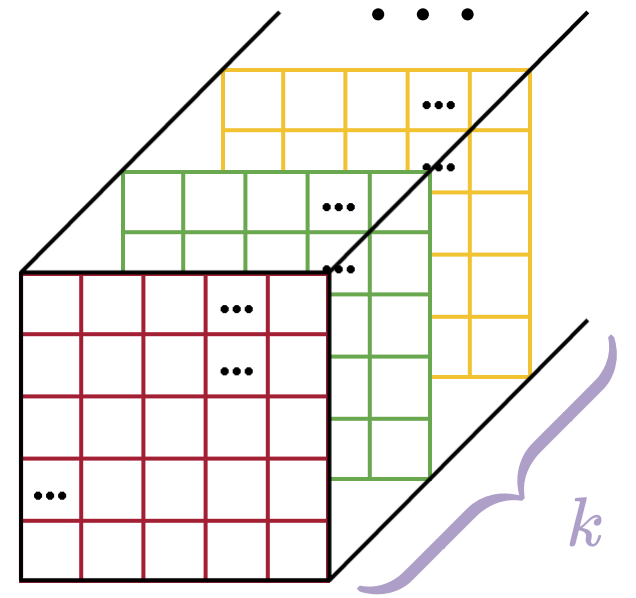
k filters



...



output tensor



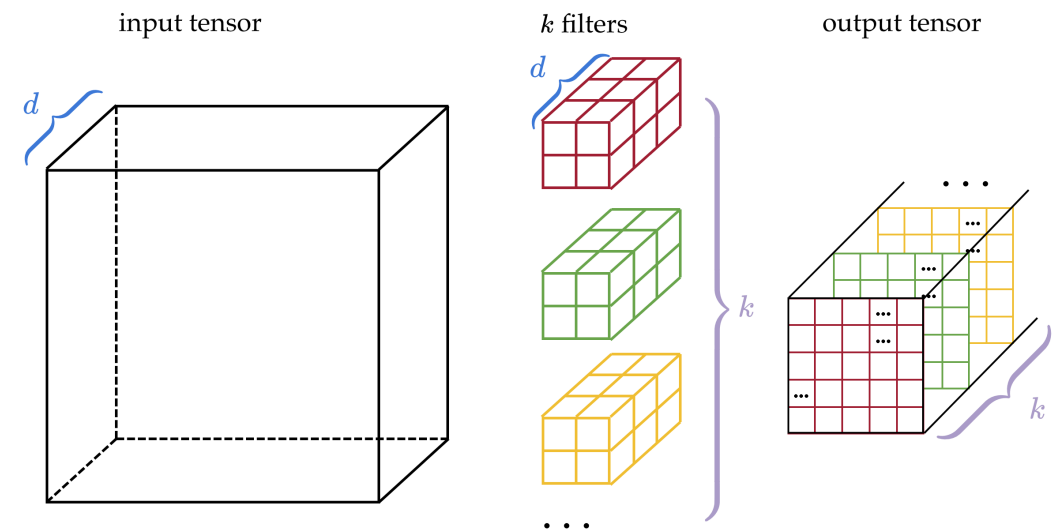
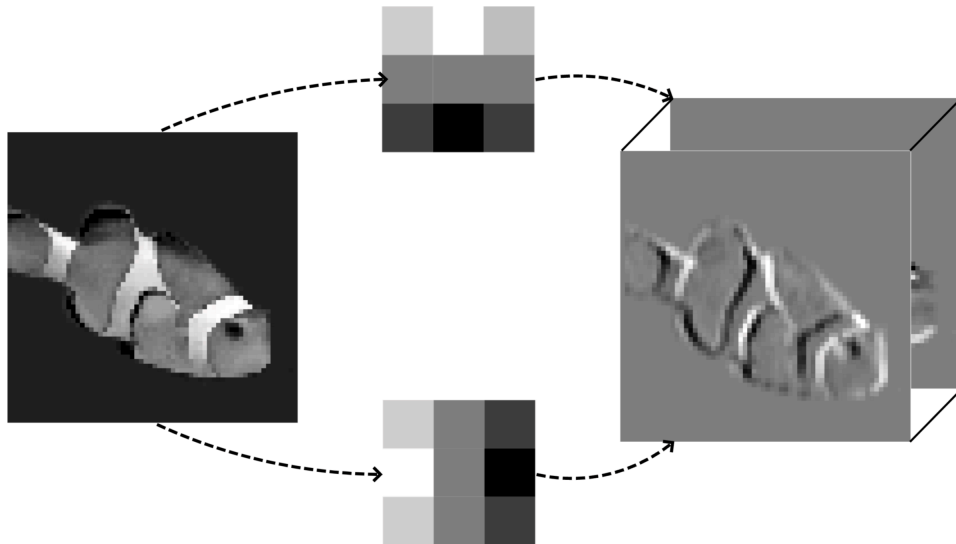
this layer's k becomes
next layer's d

Every convolutional layer works with 3d tensors:

1. color input



2. the use of multiple filters in doing 2d convolution

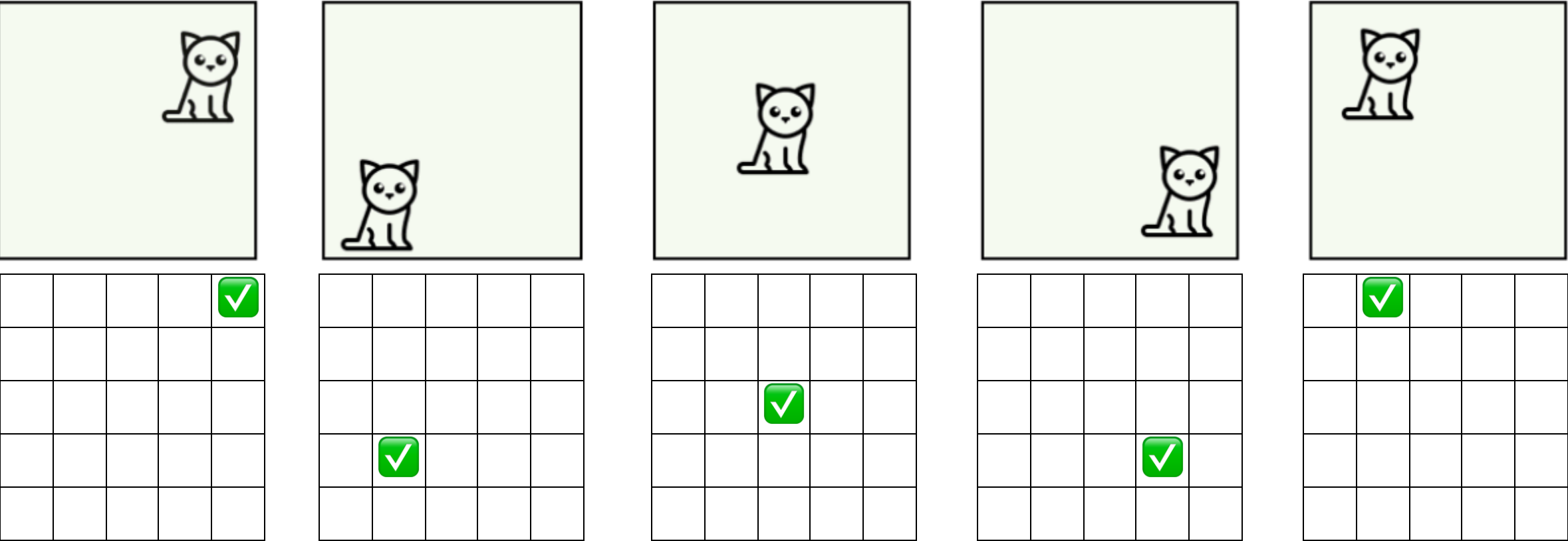


<https://distill.pub/2017/feature-visualization/appendix>

Outline

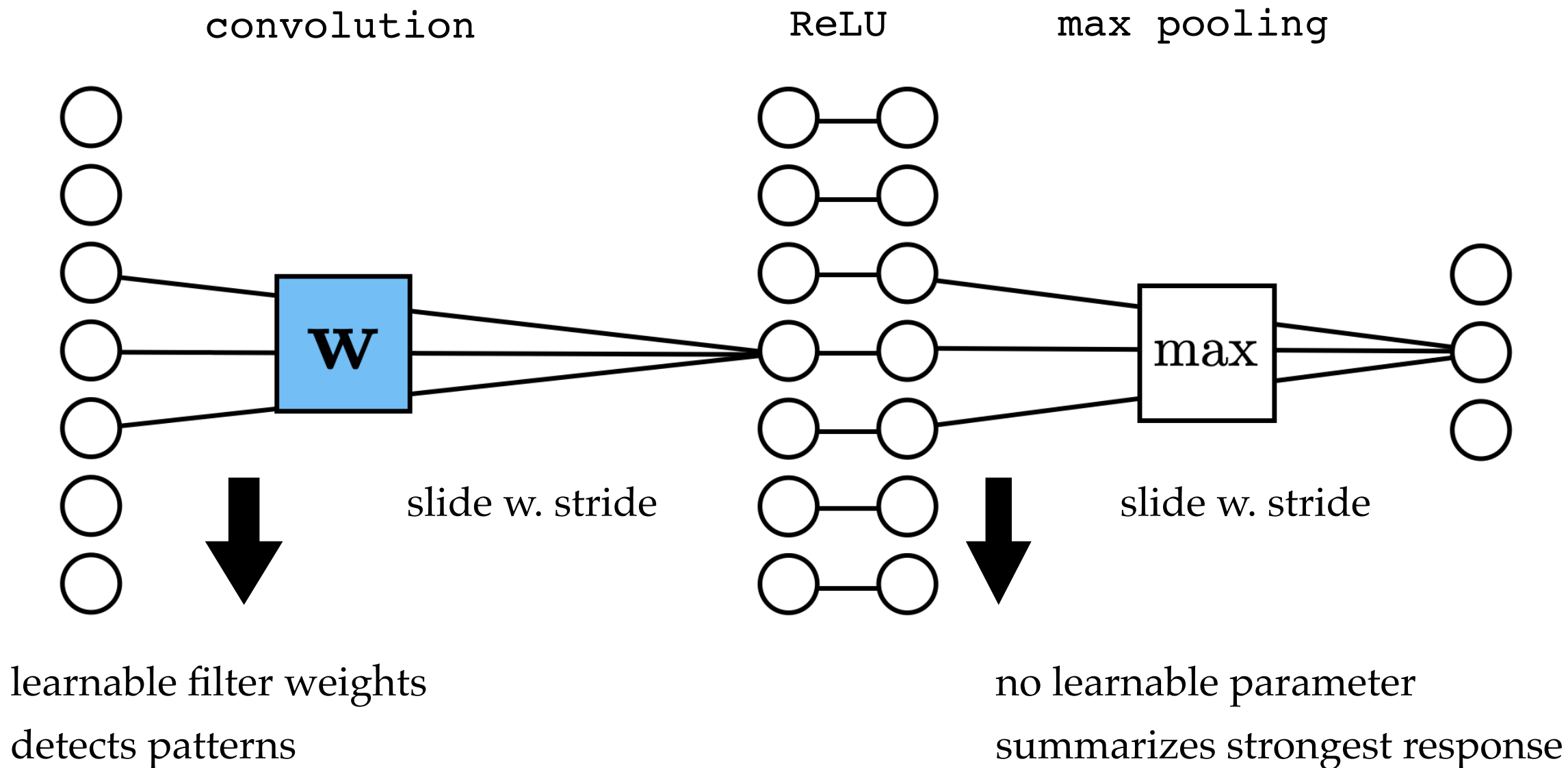
- Vision problem structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- (Case studies)

convolution helps detect patterns, but ...

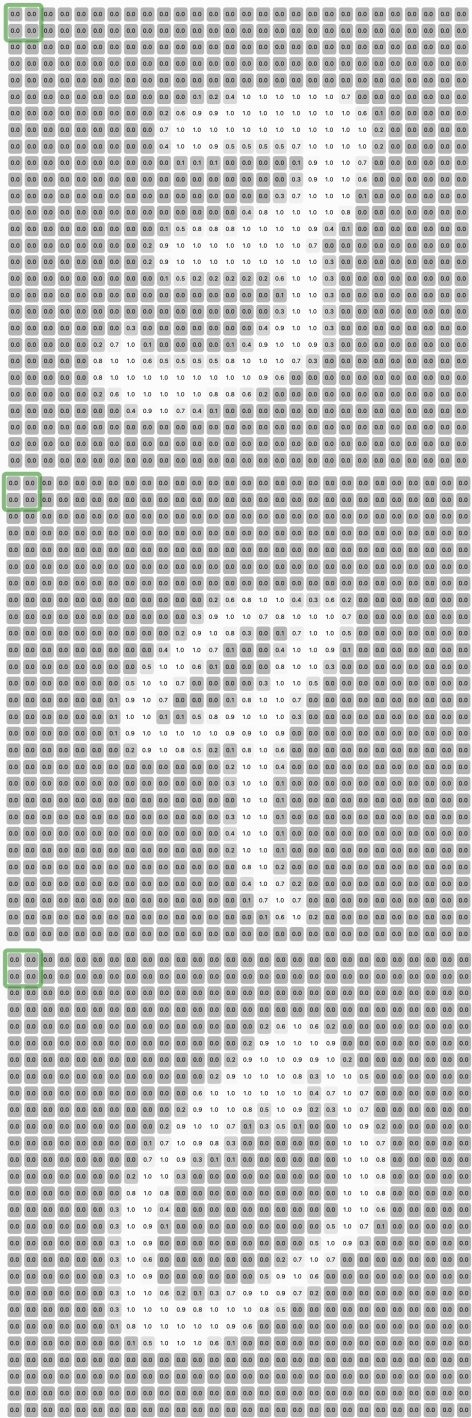
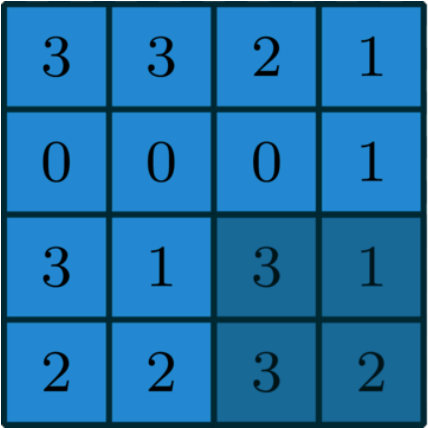
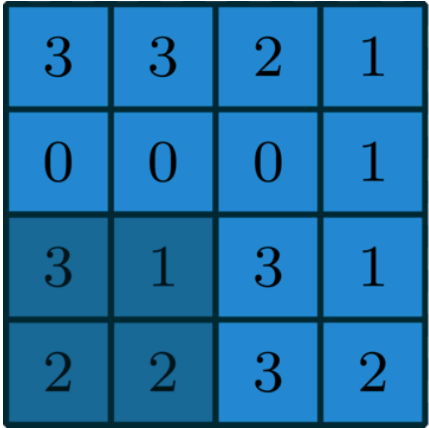
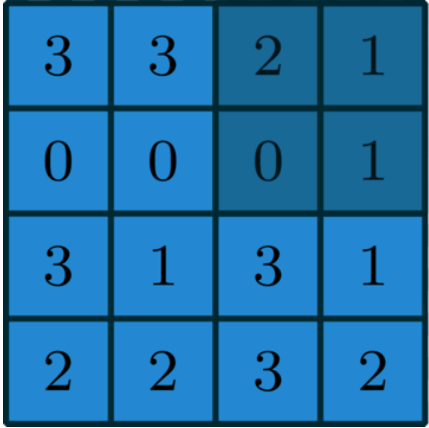
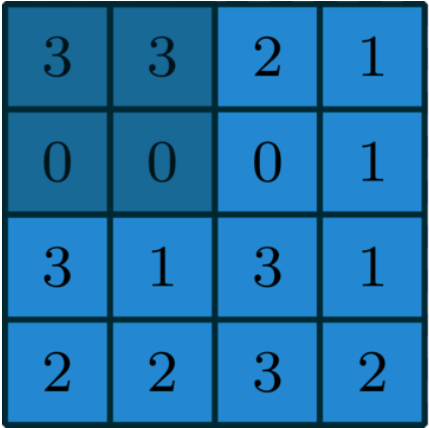


cat moves, detection moves

1d max pooling

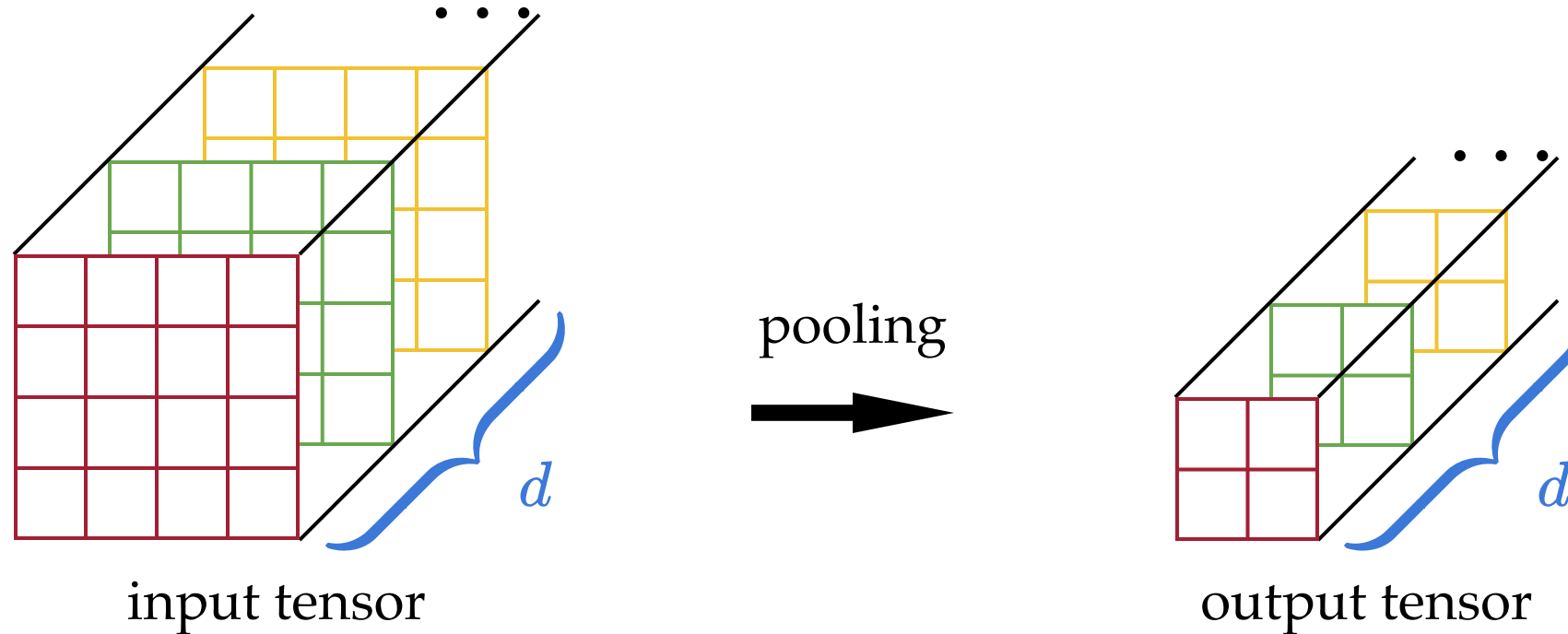


2d max pooling



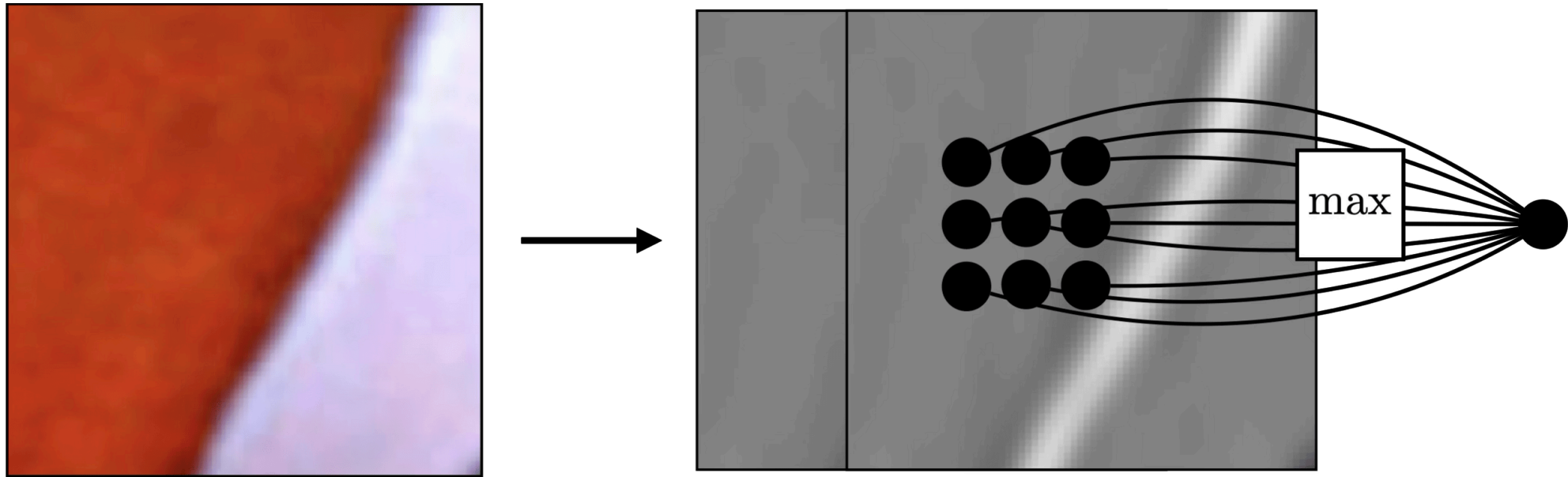
[image edited from vdumoulin, gif adapted from demo source]

max pooling on a 3d tensor



applied independently across all channels
so the *channel* dimension remains *unchanged*

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

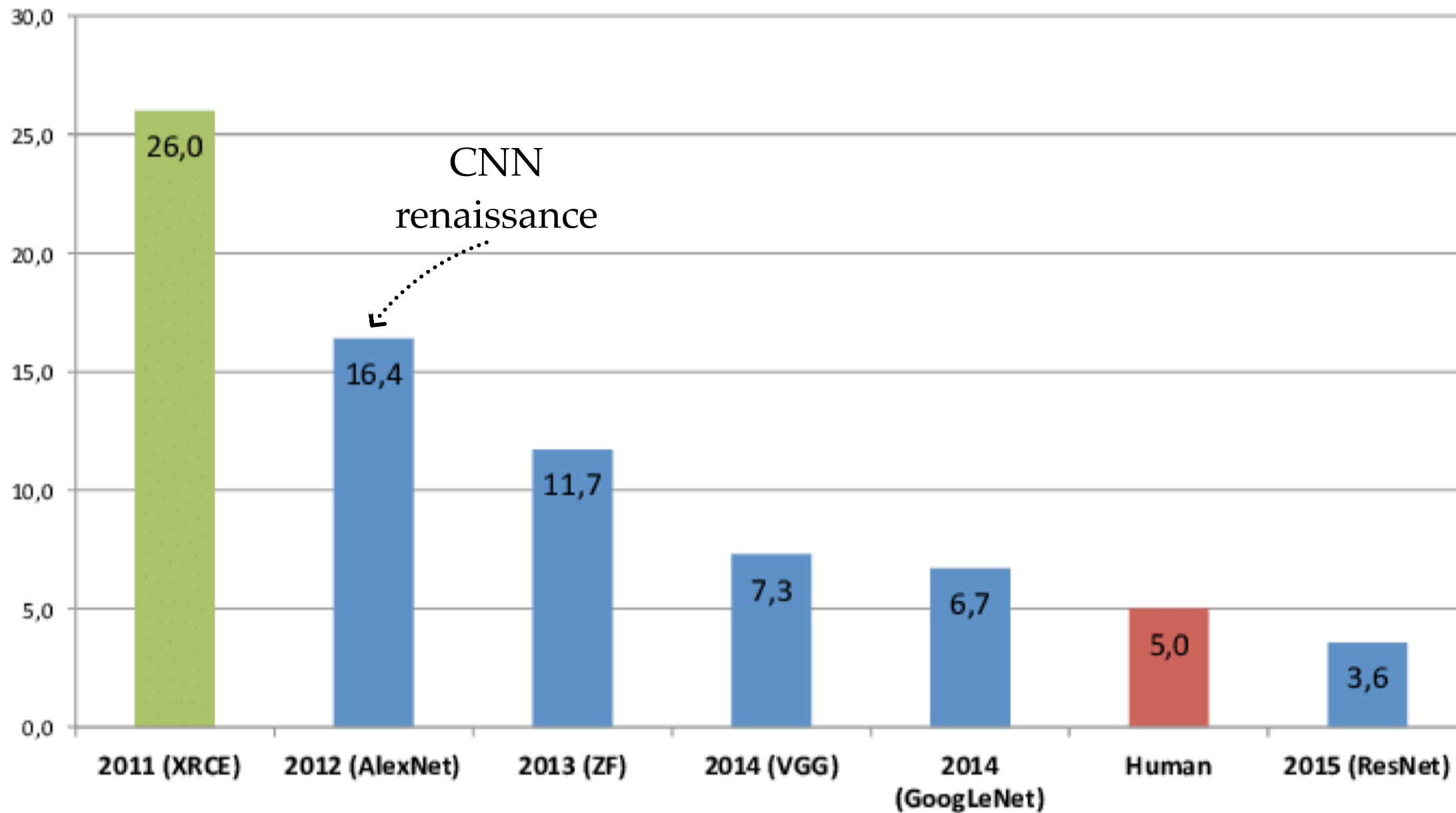


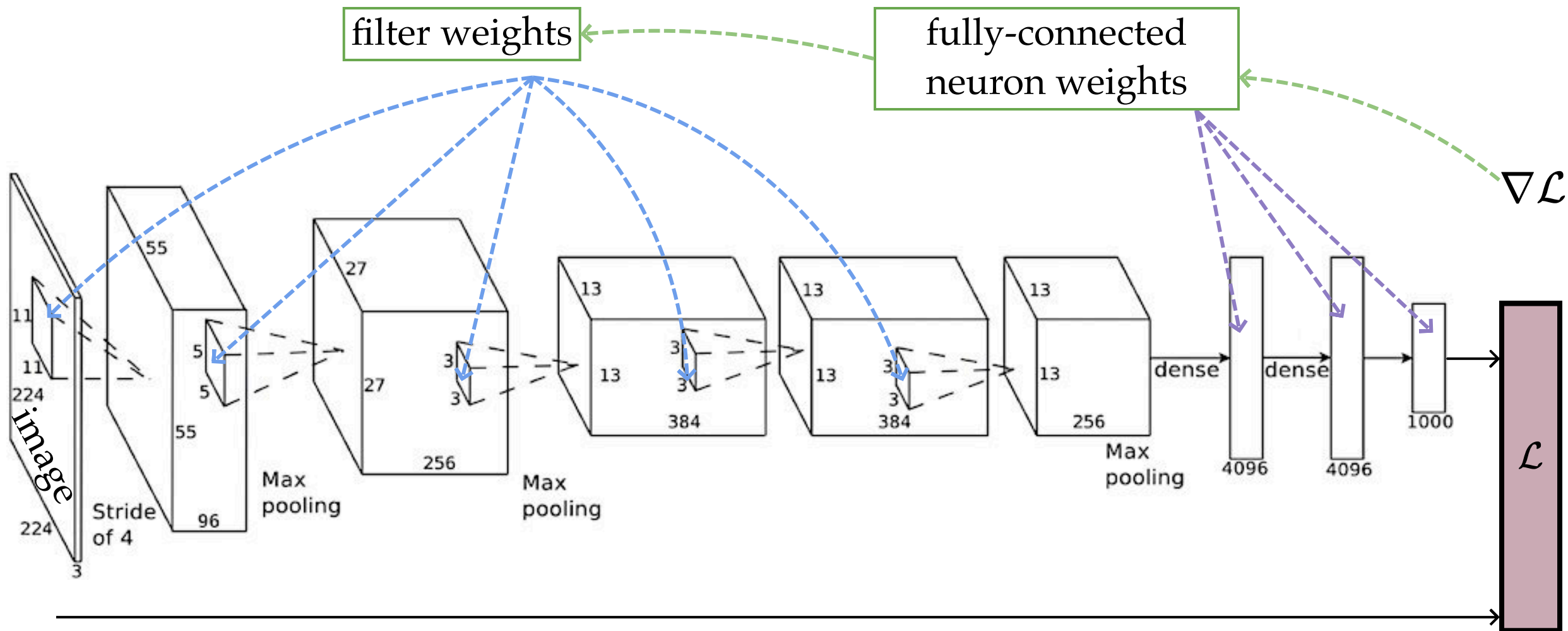
large response regardless of exact position of edge

Outline

- Vision problem structure
- Convolution
 - 1-dimensional and 2-dimensional *convolution*
 - 3-dimensional *tensors*
- Max pooling
- (Case studies)

ImageNet Classification Error (Top 5)

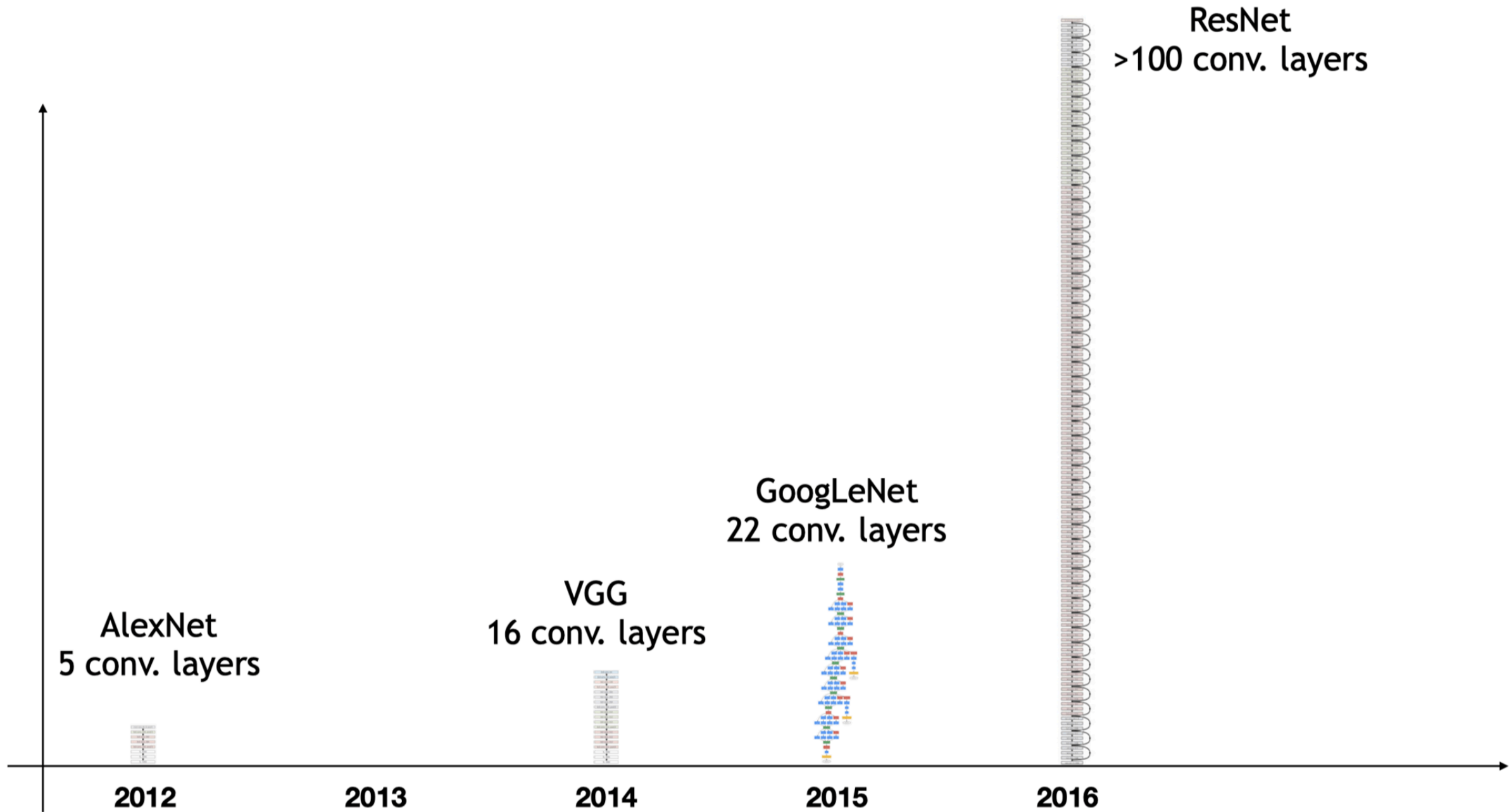




label

[AlexNet paper]

[all max pooling uses a 3x3 window, stride 2; pooled outputs not explicitly shown on diagram: 27x27x96, 13x13x256, 6x6x256]

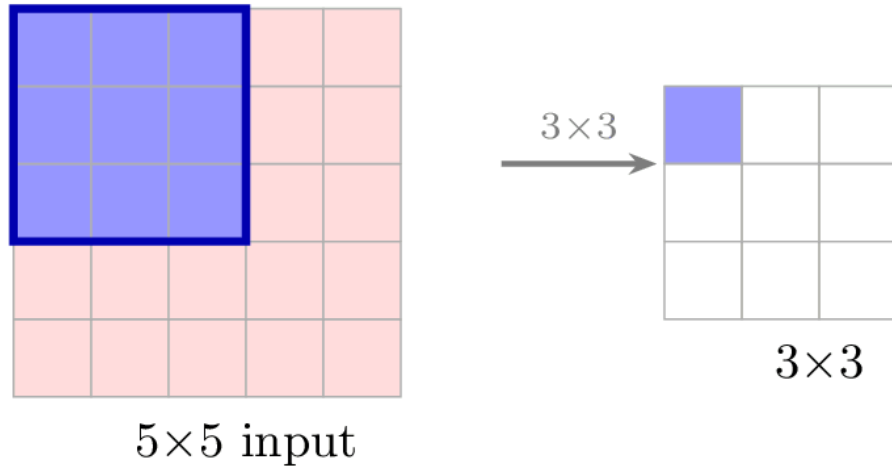


VGG '14

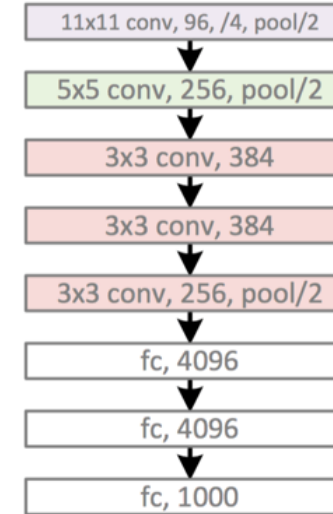
Main developments:

- small convolutional filters: only 3x3
- increased depth: about 16 or 19 layers of the same modules

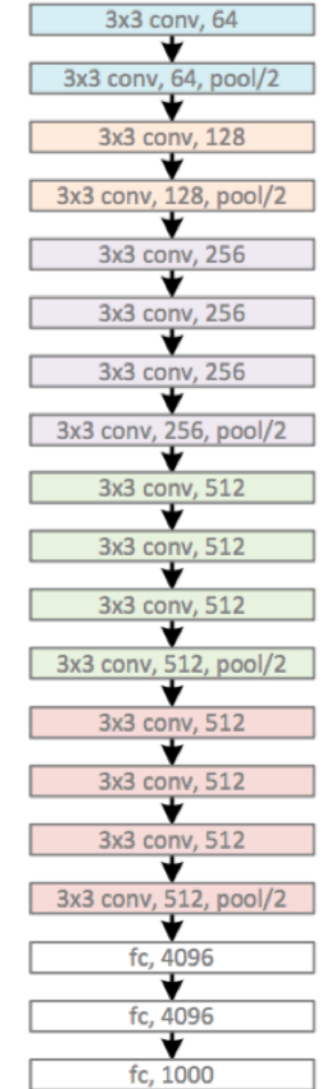
two 3×3 convolutions



AlexNet '12

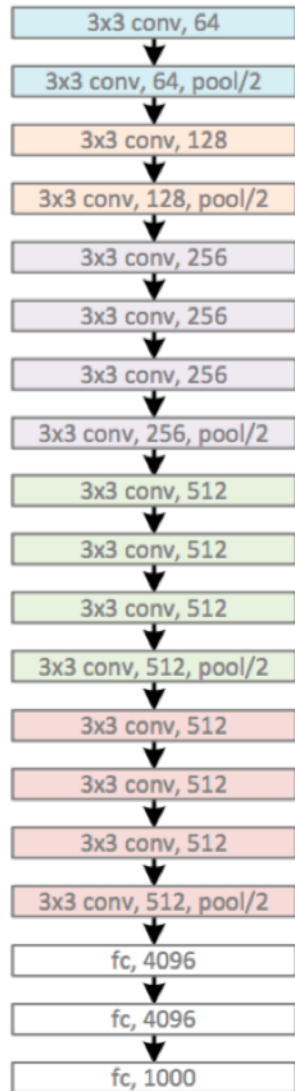


VGG '14

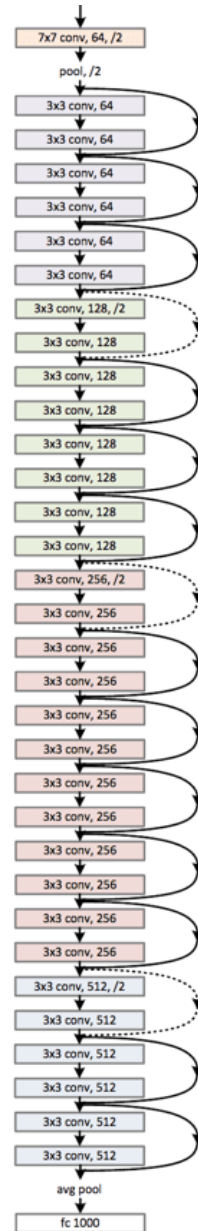


“Very Deep Convolutional Networks for Large-Scale Image Recognition”, Simonyan & Zisserman. ICLR 2015

VGG '14

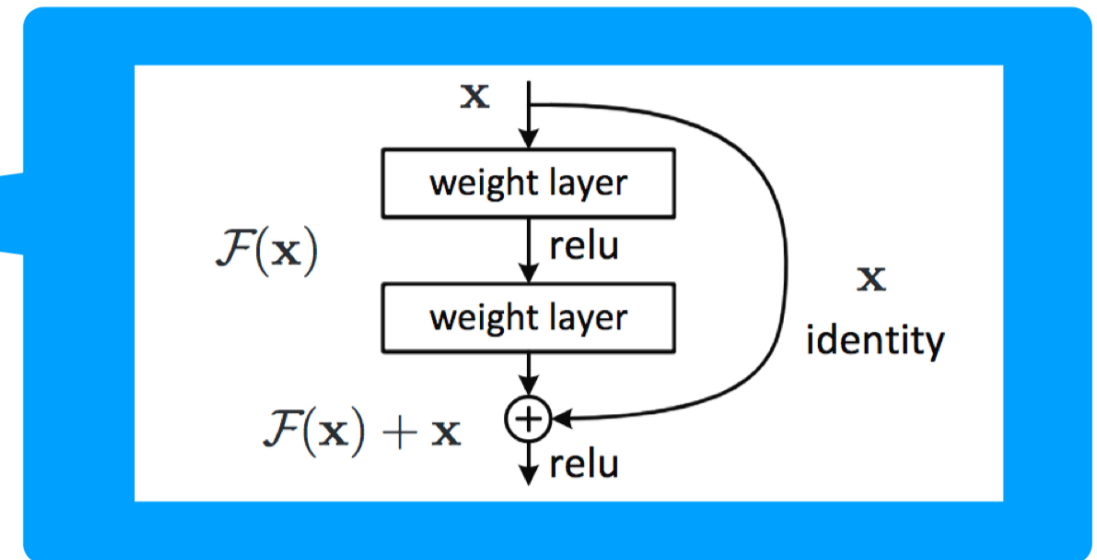


ResNet '16



Main developments:

- Residual block -- gradients can propagate faster (via the identity mapping)
- increased depth: > 100 layers



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

[image credit Philip Isola and Kaiming He] 62

Summary

- Even though NNs are universal approximators, matching the architecture to problem structure — visual hierarchy, locality, translational invariance — improves generalization and efficiency.
- Convolution slides a small learned filter across the input, detecting local patterns with shared weights — sparse and efficient.
- Max pooling summarizes spatial information: "did a pattern occur?" rather than "where exactly?"
- Filter weights are learned end-to-end; convolutional layers extract features, fully connected layers classify.