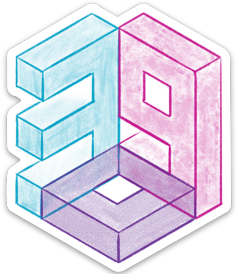


https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 7: Convolutional Neural Networks

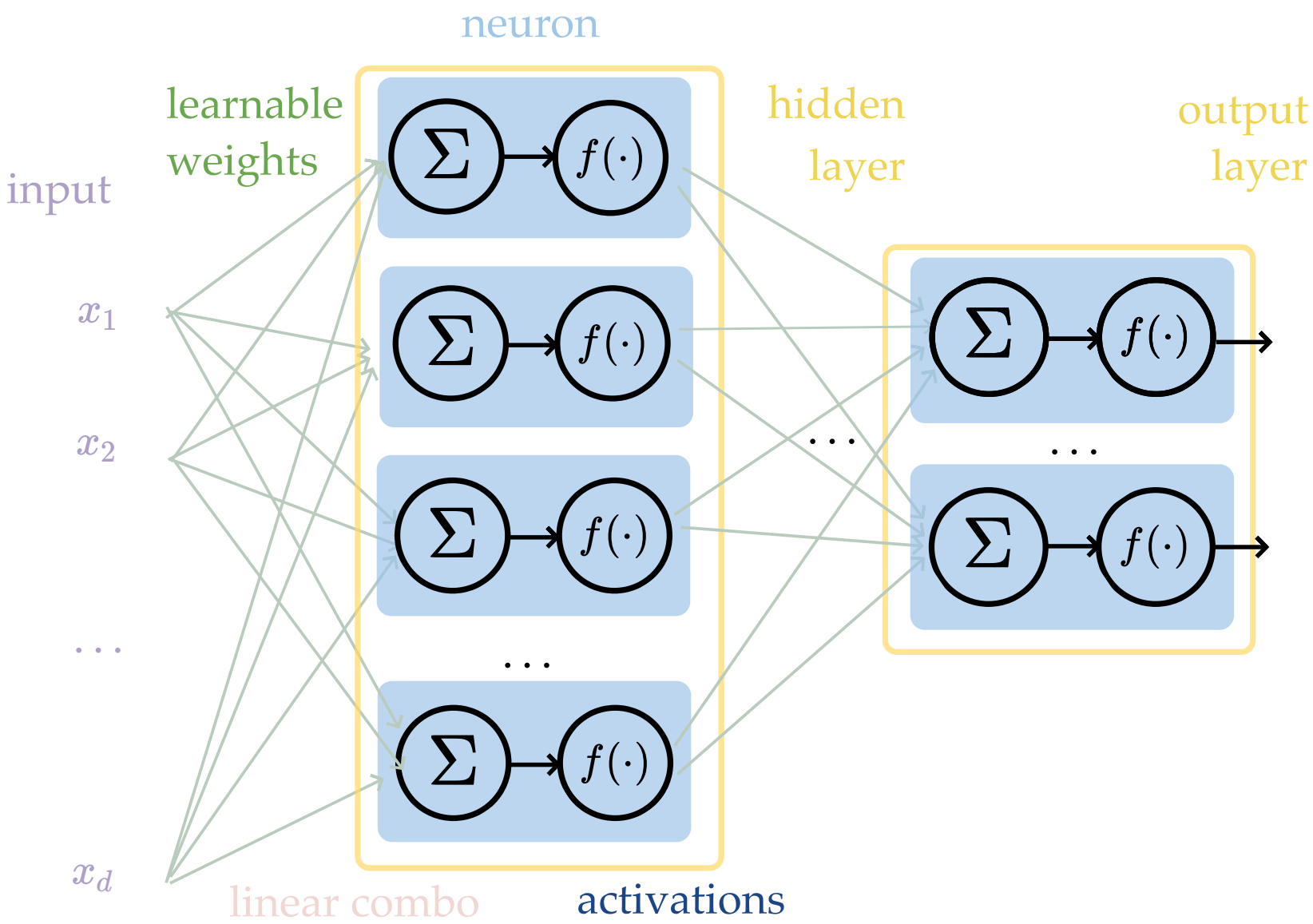
Shen Shen

Oct 16, 2025

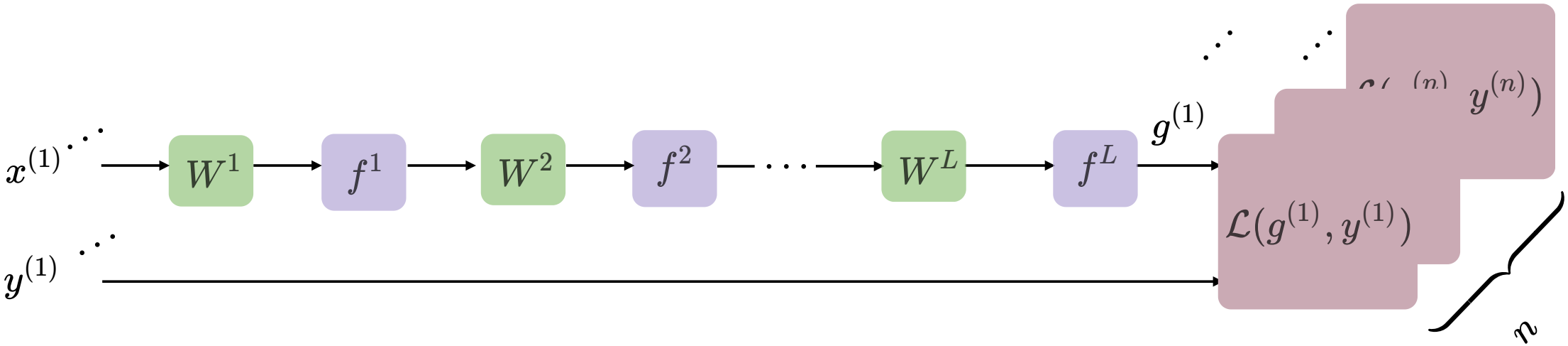
11am, Room 10-250

Interactive Slides and Lecture Recording

Recap: A (fully-connected, feed-forward) neural network



Forward pass: evaluate, *given* the current parameters



- the model outputs $g^{(i)} = f^L\left(\ldots f^2\left(\ f^1(\mathbf{x}^{(i)}; \mathbf{W}^1); \mathbf{W}^2\right); \ldots \mathbf{W}^L\right)$
- the loss incurred on the current data $\mathcal{L}(g^{(i)}, y^{(i)})$
- the training error $J = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(g^{(i)}, y^{(i)})$

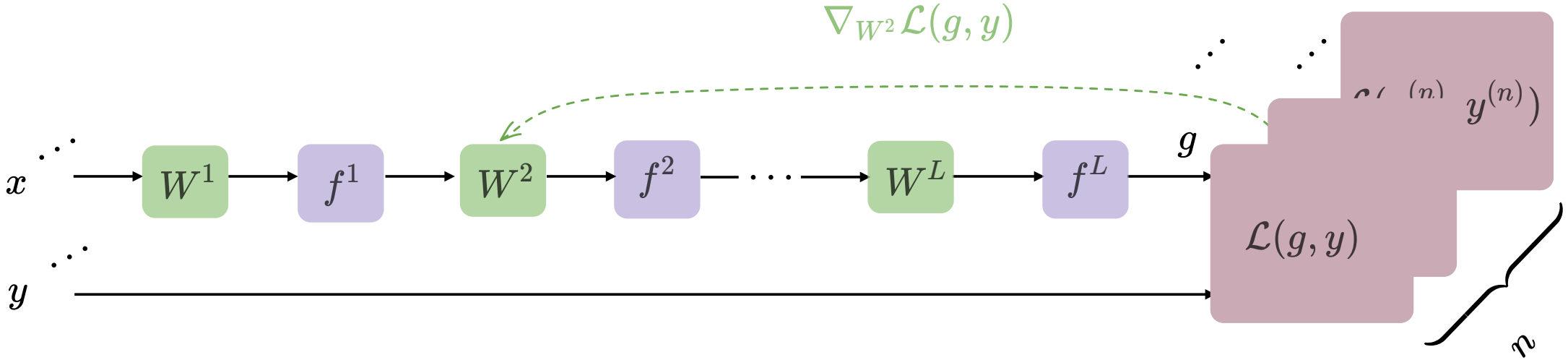
linear combination

(nonlinear) activation

loss function

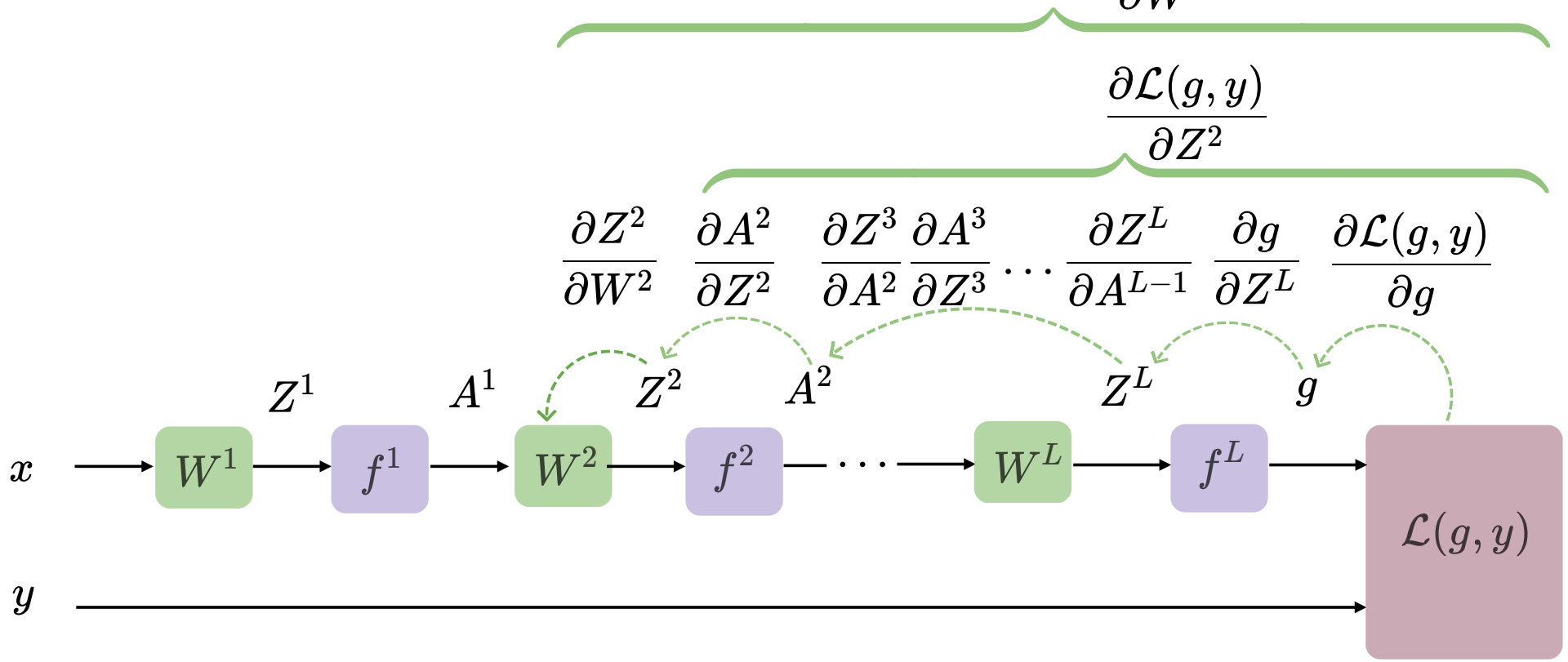
`Backward pass: run SGD to learn all parameters`

e.g. to update W^2

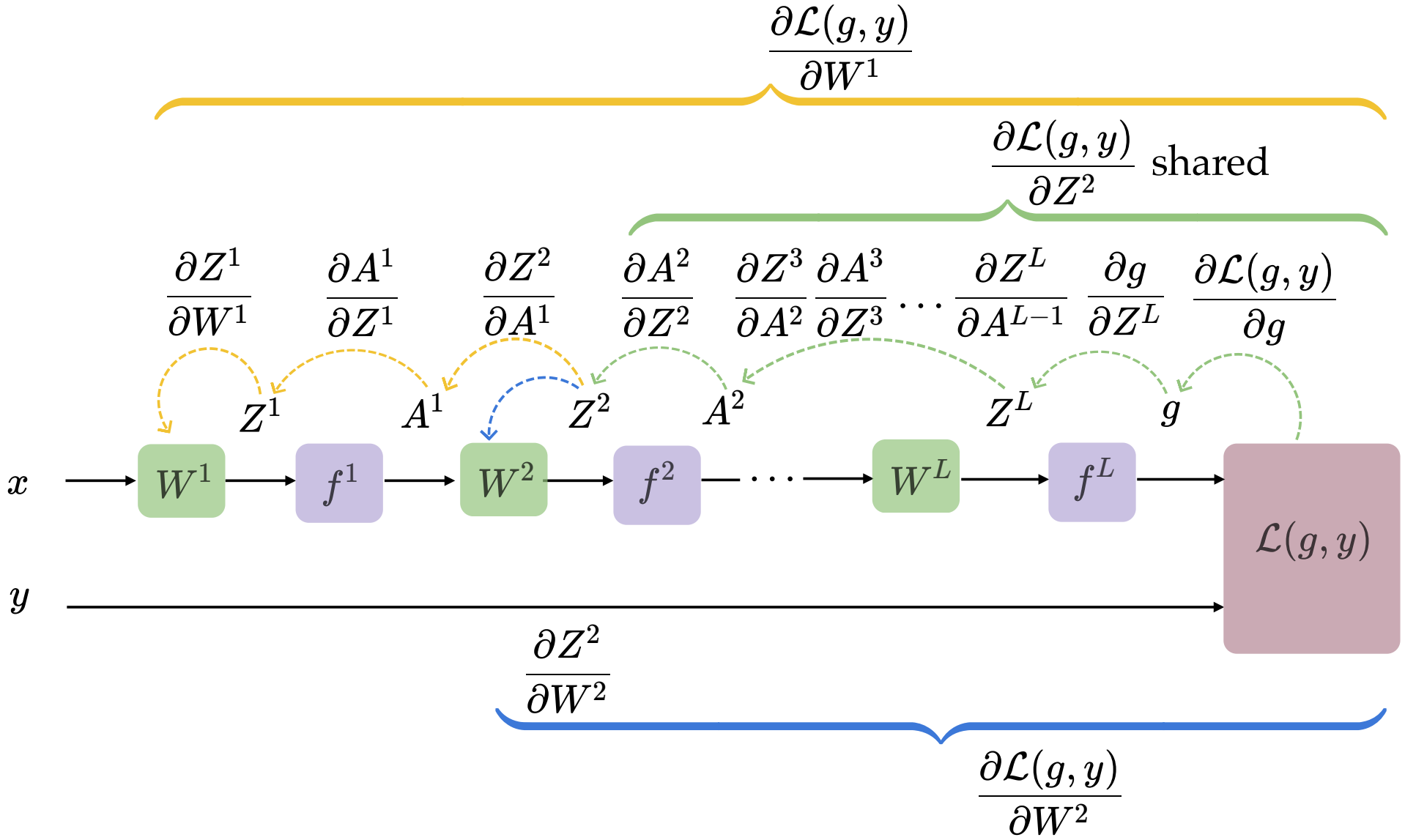


- Randomly pick a data point (x, y)
- Evaluate the gradient $\nabla_{W^2}\mathcal{L}(g, y)$
- Update the weights $W^2 \leftarrow W^2 - \eta\nabla_{W^2}\mathcal{L}(g, y)$

suppose we sampled a particular (x, y), how to find $\frac{\partial \mathcal{L}(g,y)}{\partial W^2}$?

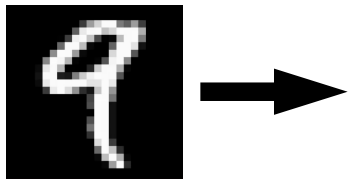


back propagation: reuse of computation



(The demo won't embed in PDF. But the direct link below works.)

https://shenshen.mit.edu/demos/nn/playground/

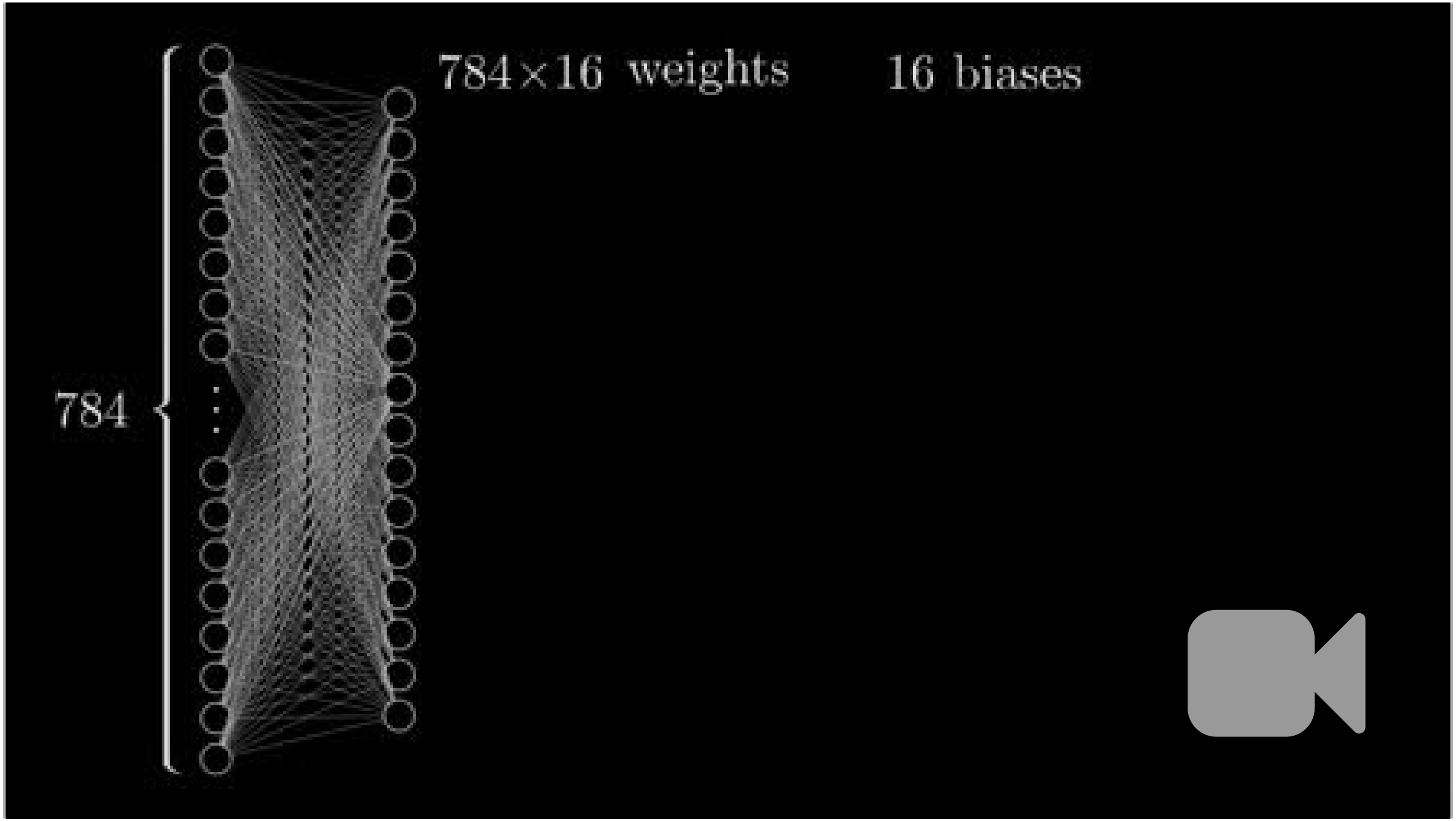


convolutional neural networks → 9

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
784

426-by-426
grayscale image

Use the same 2 hidden-layer network to predict what top-10 engineering school seal this image is, need to learn ~3M parameters.

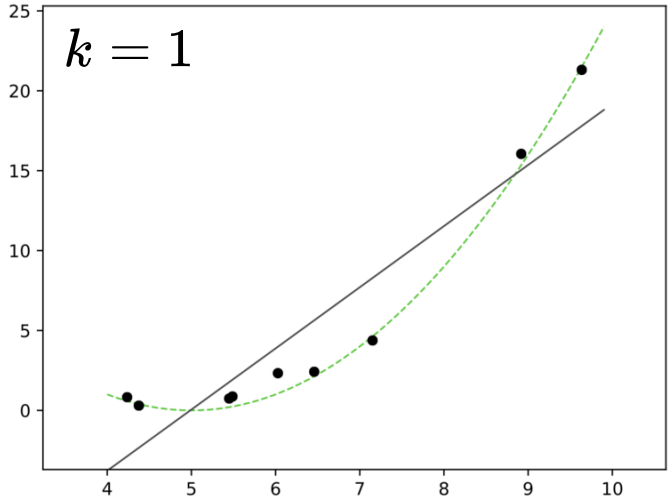
For higher-resolution images, or more complex tasks, or larger networks, the number of parameters can grow very fast.

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

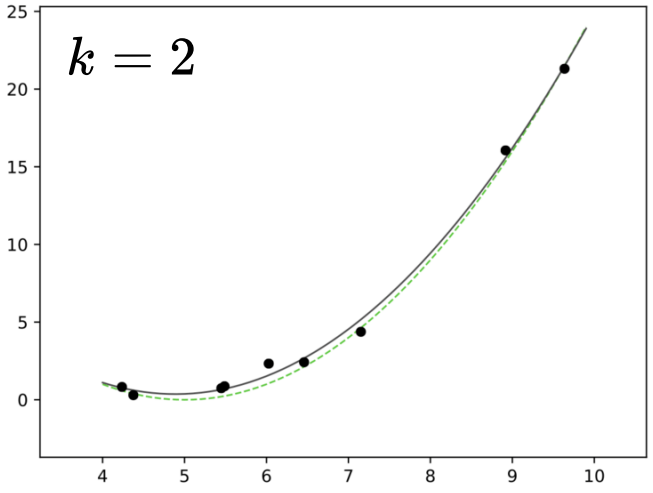
- Partly, fully-connected nets don't scale well for vision tasks
- More importantly, a carefully chosen hypothesis class helps fight overfitting

Recall, models with needless parameters tend to overfit

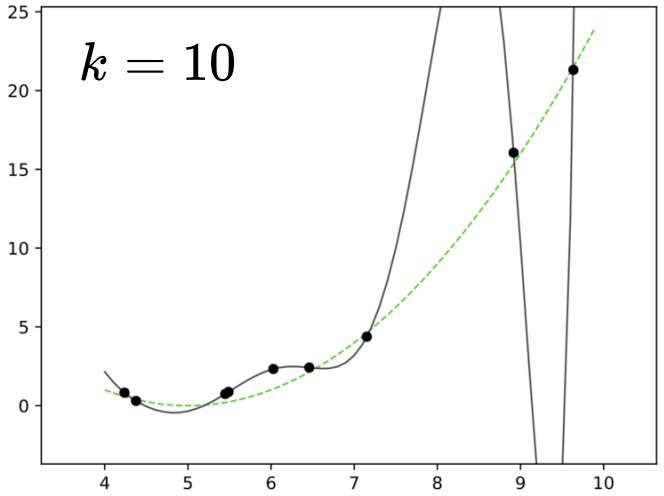
Underfitting



Appropriate



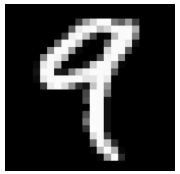
Overfitting



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?

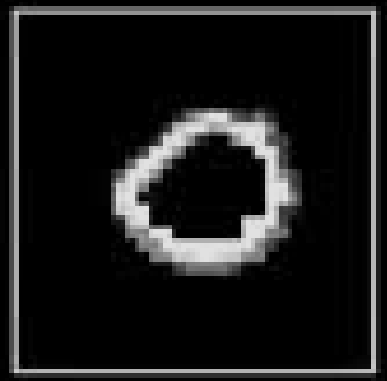
Why do we humans think



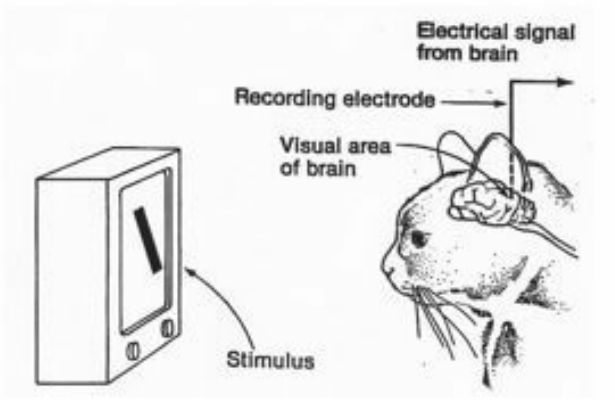
is a 9?

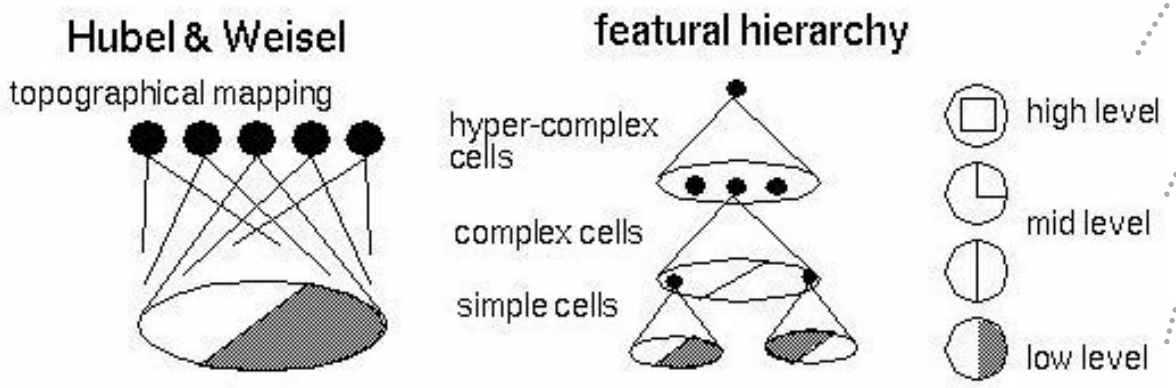
Why do we think any of

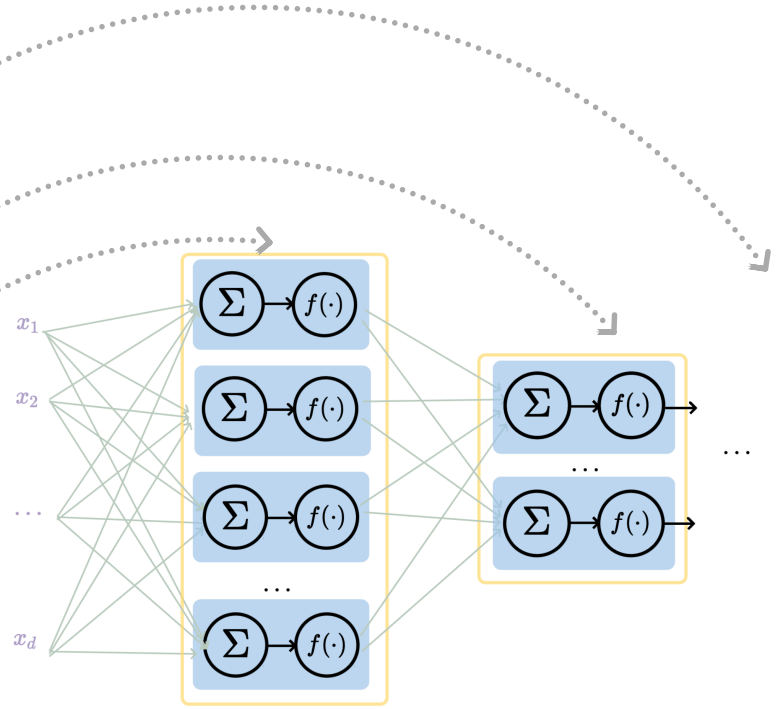
is a 9?



- Visual hierarchy

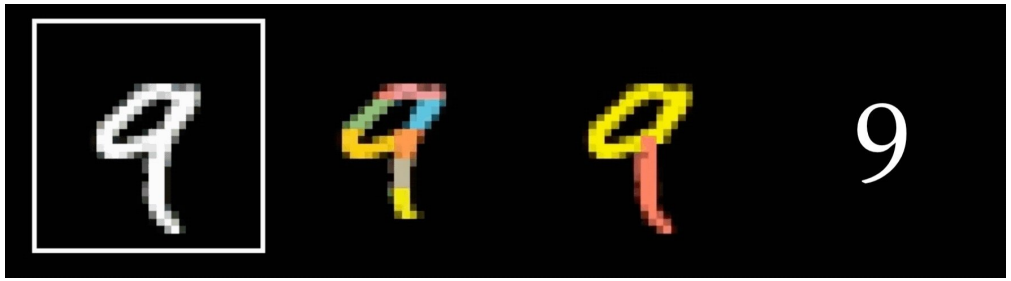




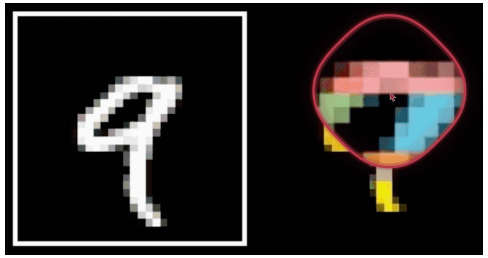


Layered structure are well-suited to model this hierarchical processing.

- Visual hierarchy



- Spatial locality



- Translational invariance

CNN exploits

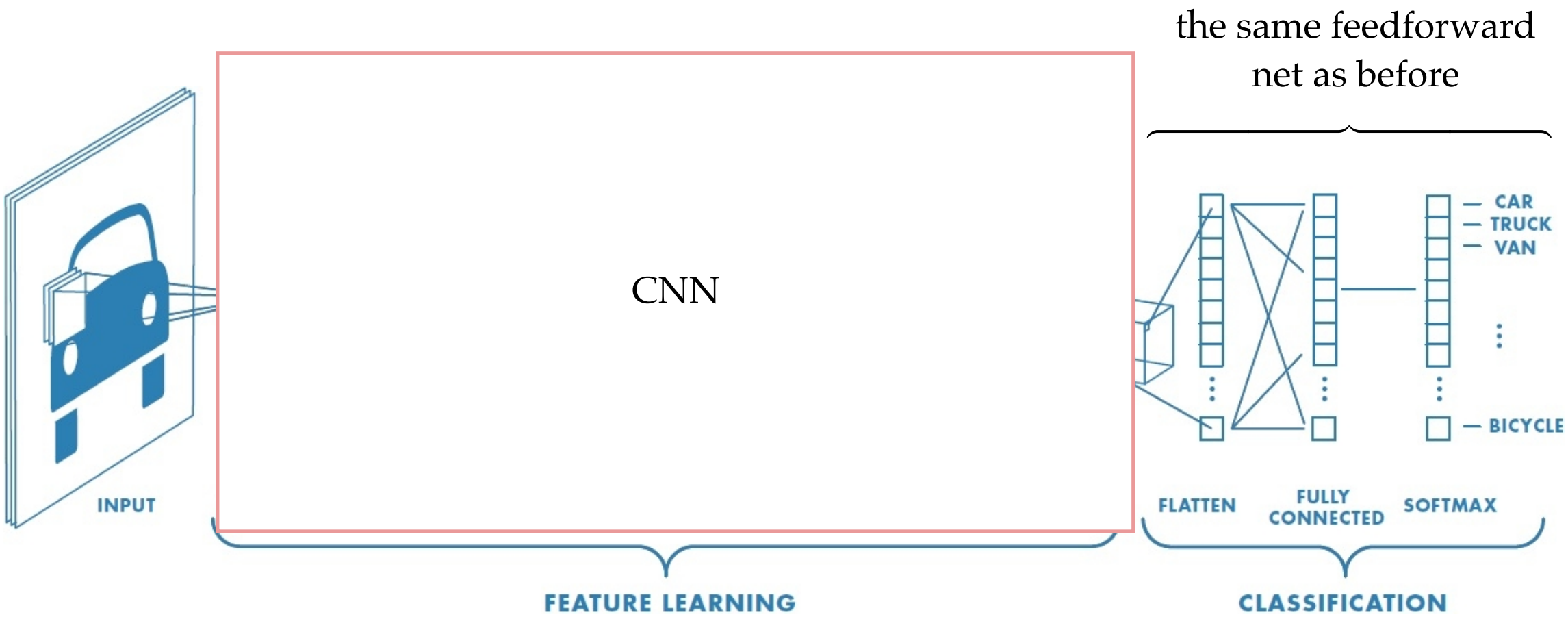
- Visual hierarchy
- Spatial locality
- Translational invariance

via

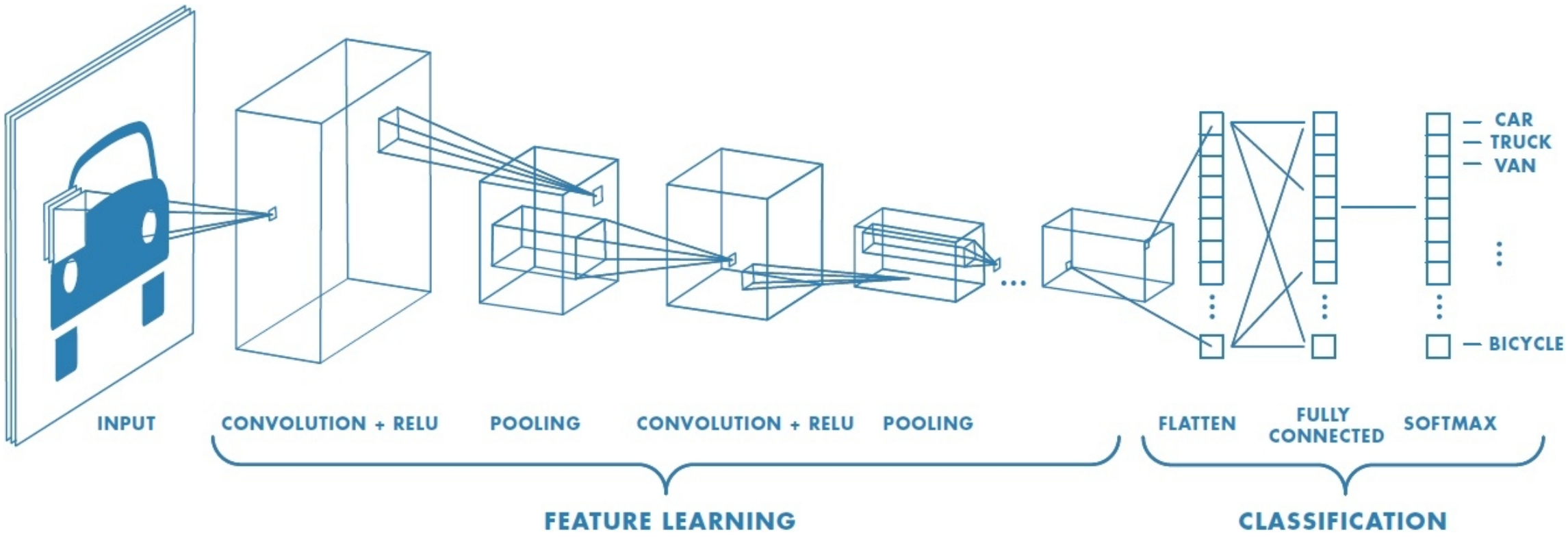
- layered structure
- convolution
- pooling

to handle images efficiently and sensibly.

typical CNN architecture for image classification



typical CNN structure for image classification



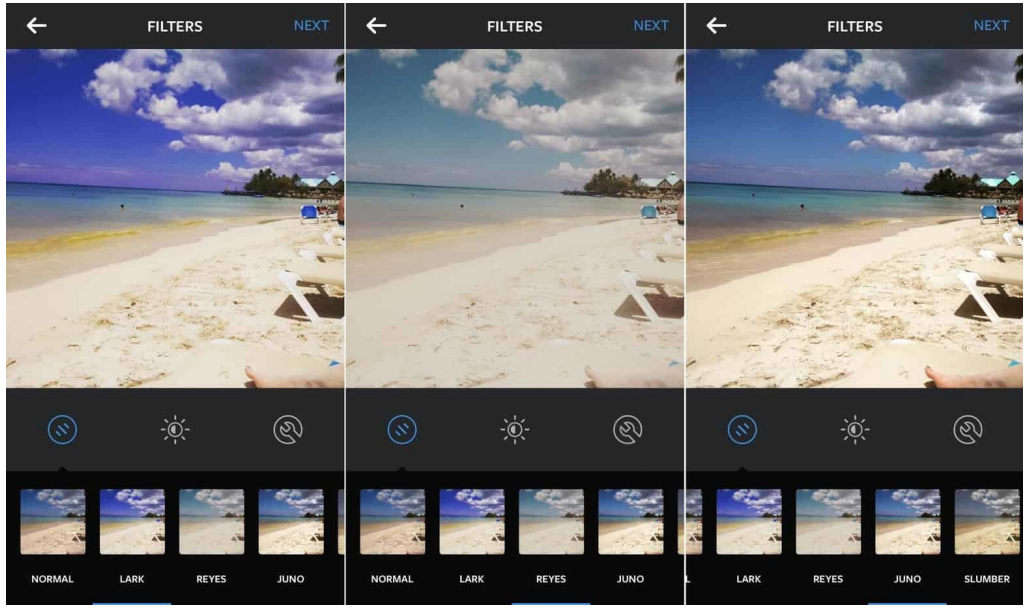
Outline

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

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

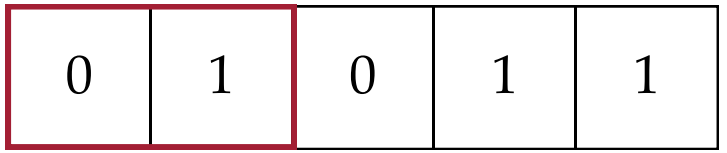
Layer	Forward pass, *do*	Backward pass, *learn*	Design choices
fully-connected	dot-product, activation	neuron weights	neuron count, etc.
convolutional	convolution, activation	filter weights	conv specs, etc.

Convolution result:

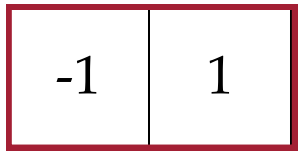


example: 1-dimensional convolution

input

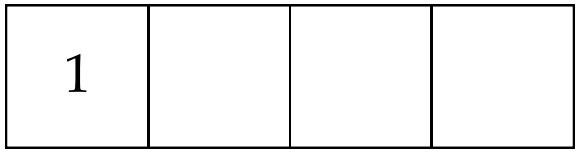


filter



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

convolved output



example: 1-dimensional convolution

input

0	1	0	1	1

filter

-1	1

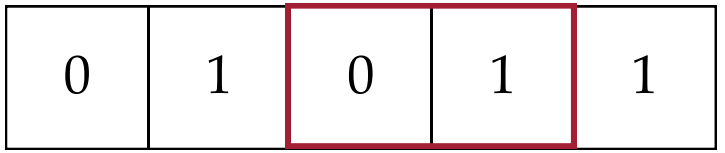
$$(1 * -1) + (0 * 1) = -1$$

convolved output

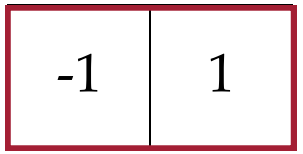
1	-1		

example: 1-dimensional convolution

input

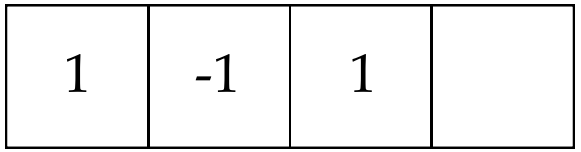


filter



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

convolved output



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: "look" *locally* through the filter (sparse-connected layer)

input

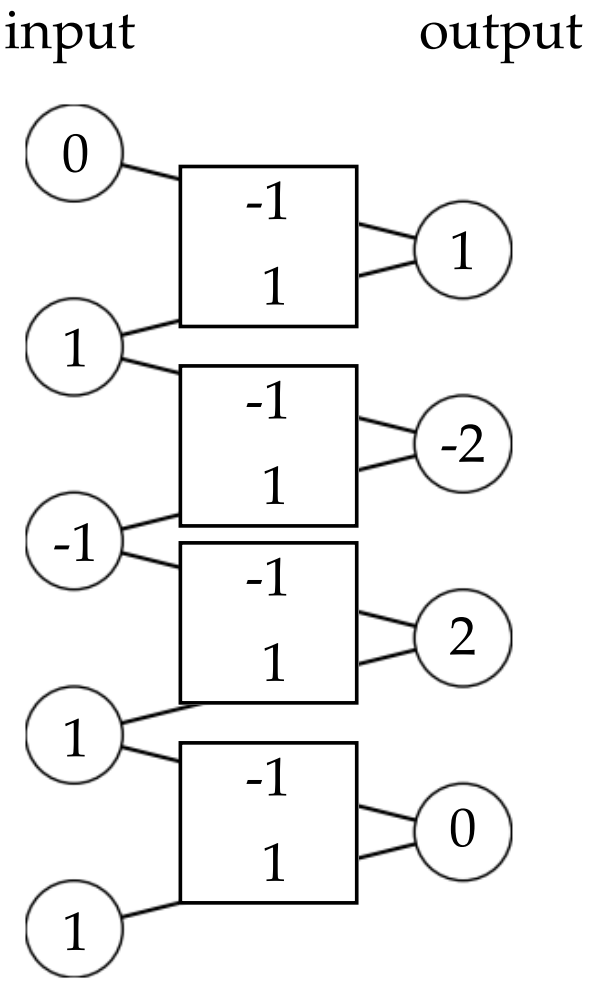
0	1	-1	1	1

filter

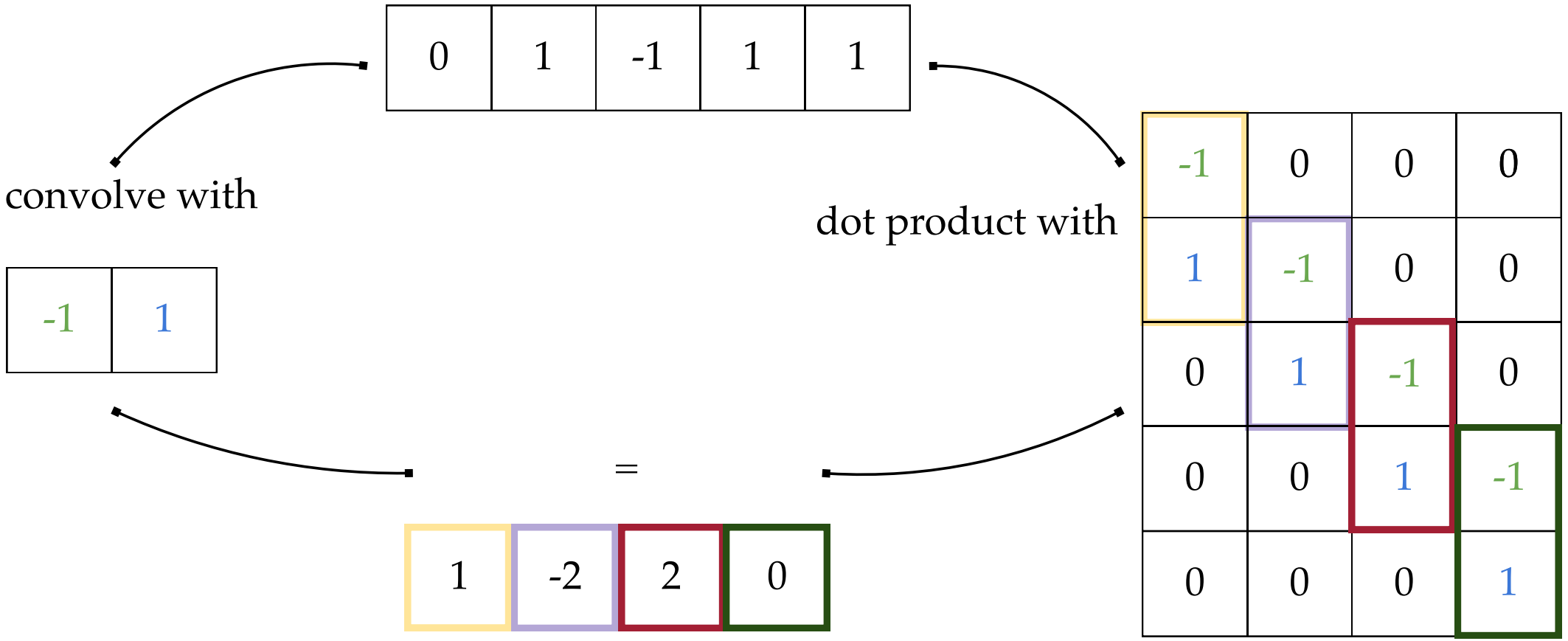
-1	1

convolved output

1	-2	2	0



convolution interpretation 3: sparse-connected layer with parameter sharing



0	1	0	1	1

convolve with

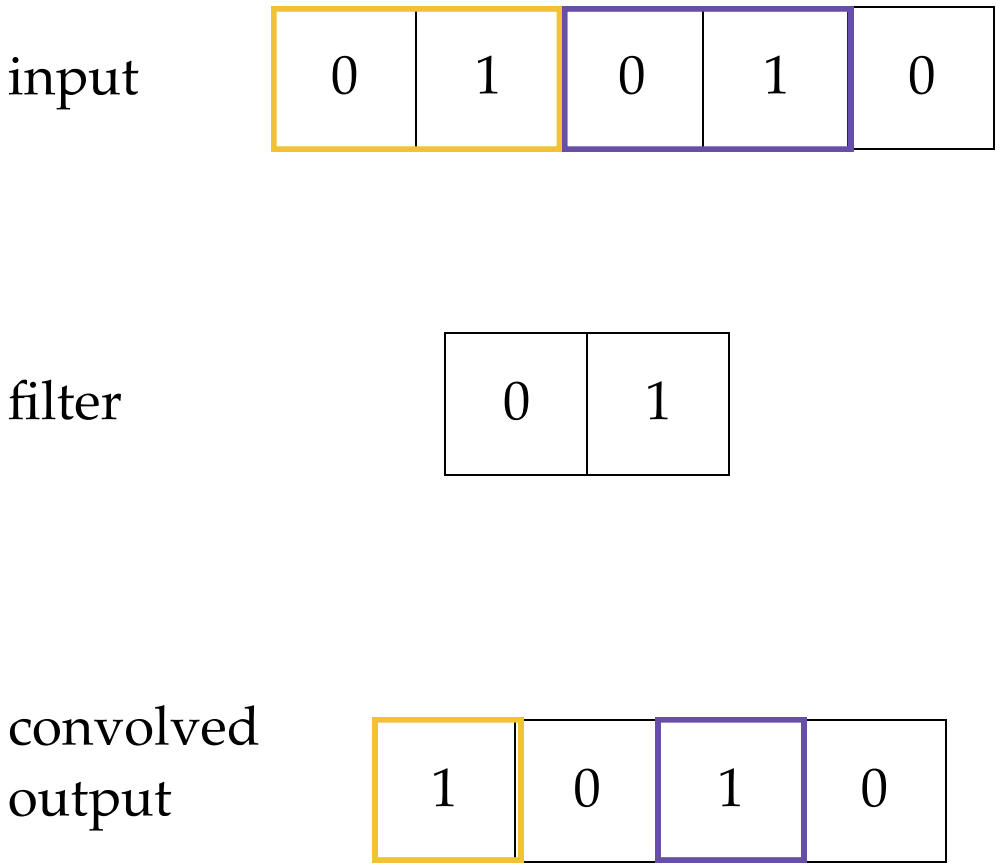
1

dot product with

$I_{5\times5}$

0	1	0	1	1

convolution interpretation 4: translational equivariance



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

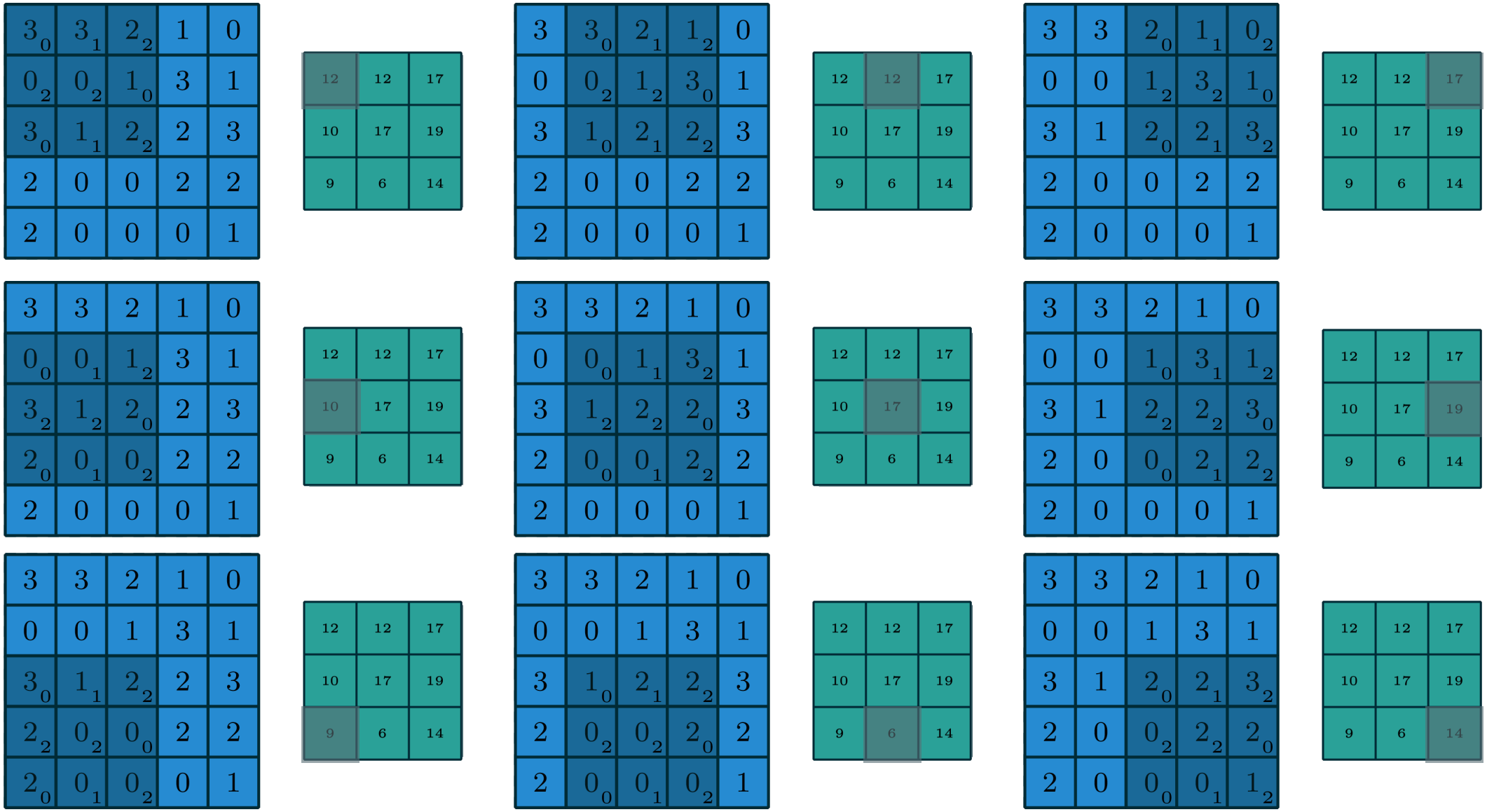
filter

0	1	2
2	2	0
0	1	2

convolved output

12	12	17
10	17	19
9	6	14

[image edited from vdumoulin]



[image edited from vdumoulin]

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

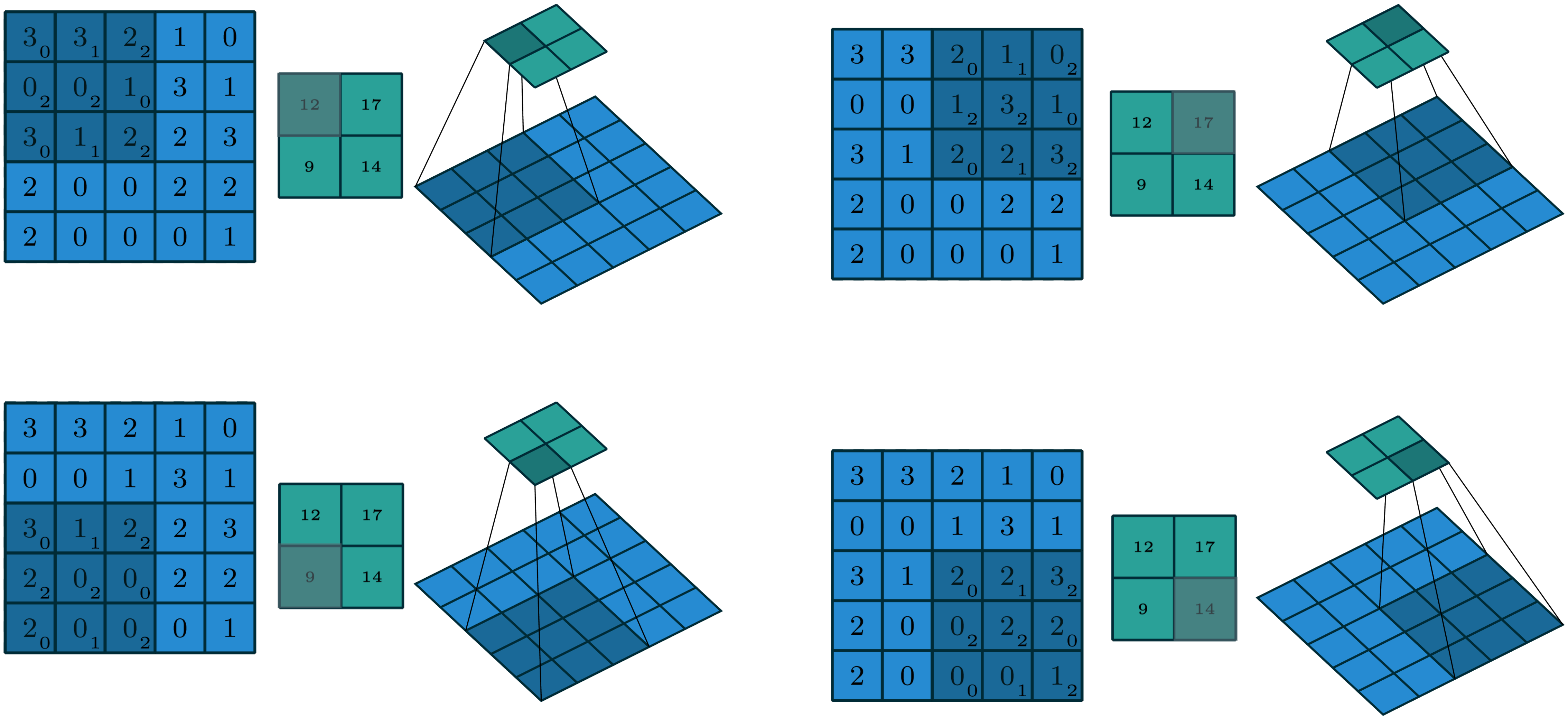
0	1	2
2	2	0
0	1	2

output

12	17
9	14

[image edited from vdumoulin]

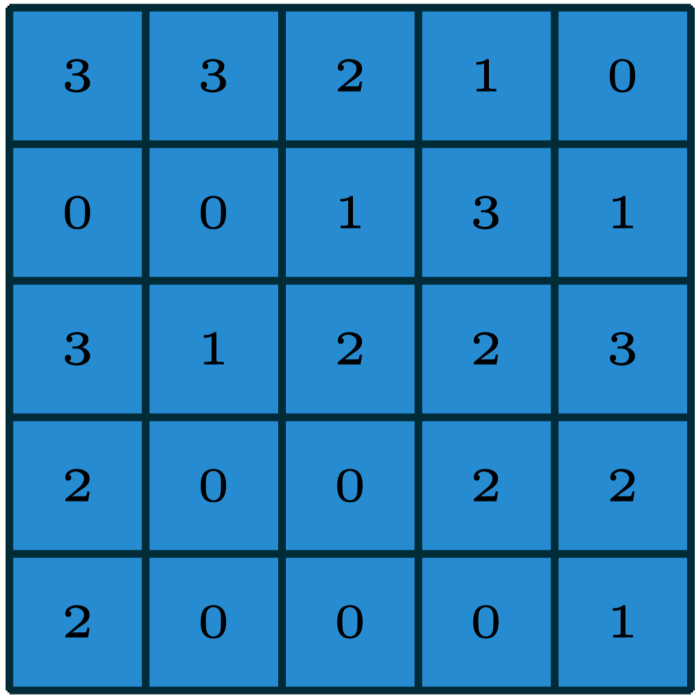
stride of 2



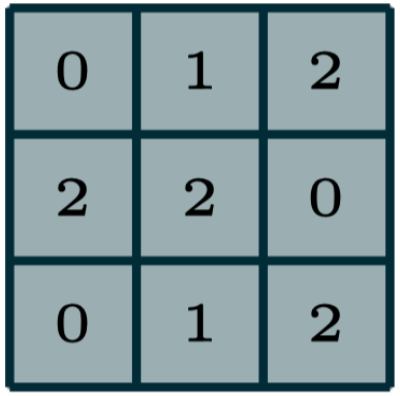
[image edited from vdumoulin]

stride of 2, with padding of size 1

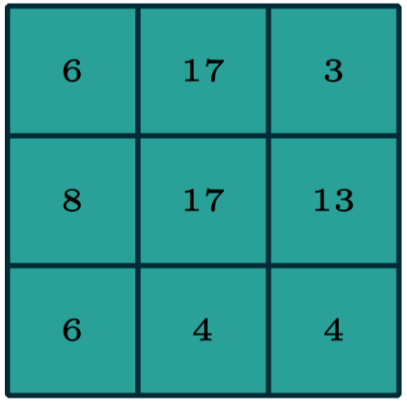
input



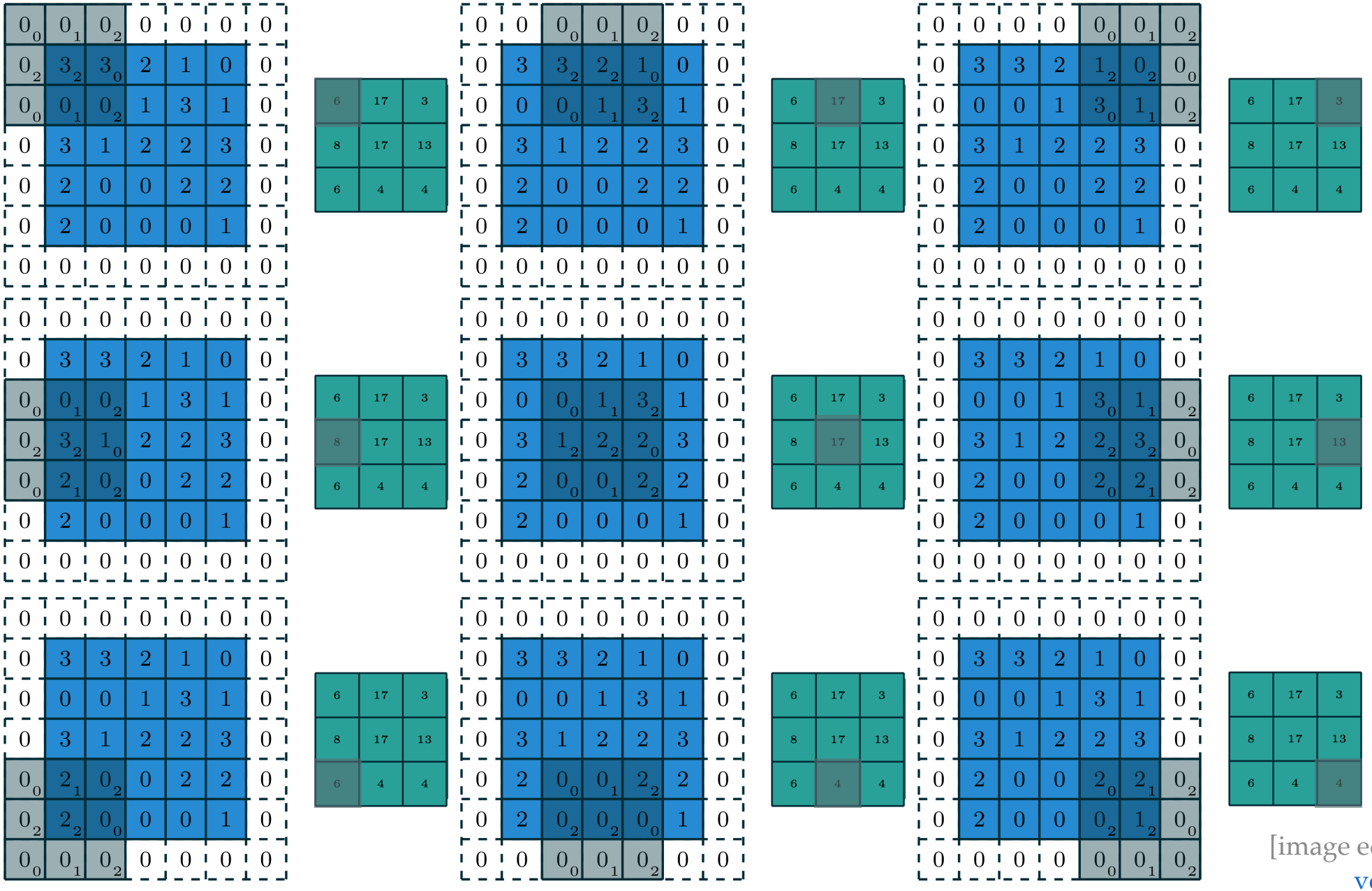
filter



output



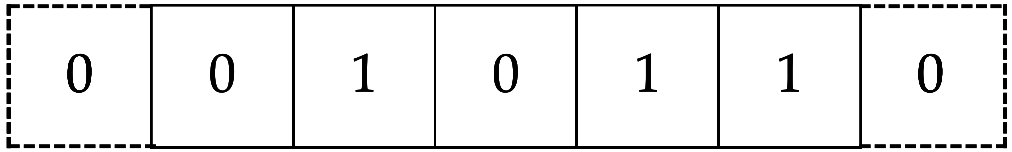
[image edited from vdumoulin]



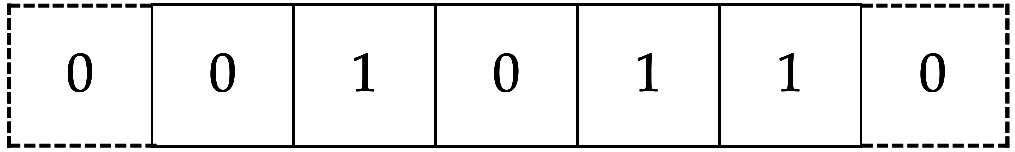
[image edited from vdumoulin]

quick summary: hyperparameters for 1d convolution

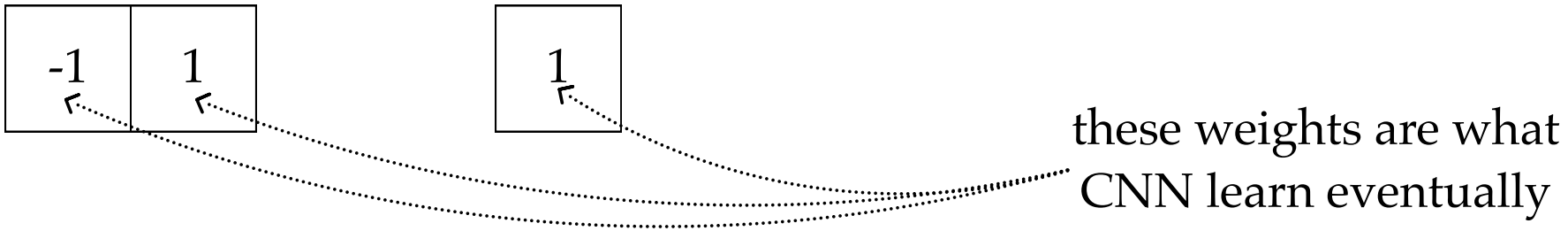
- Zero-padding



- Stride (e.g. stride of 2)



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

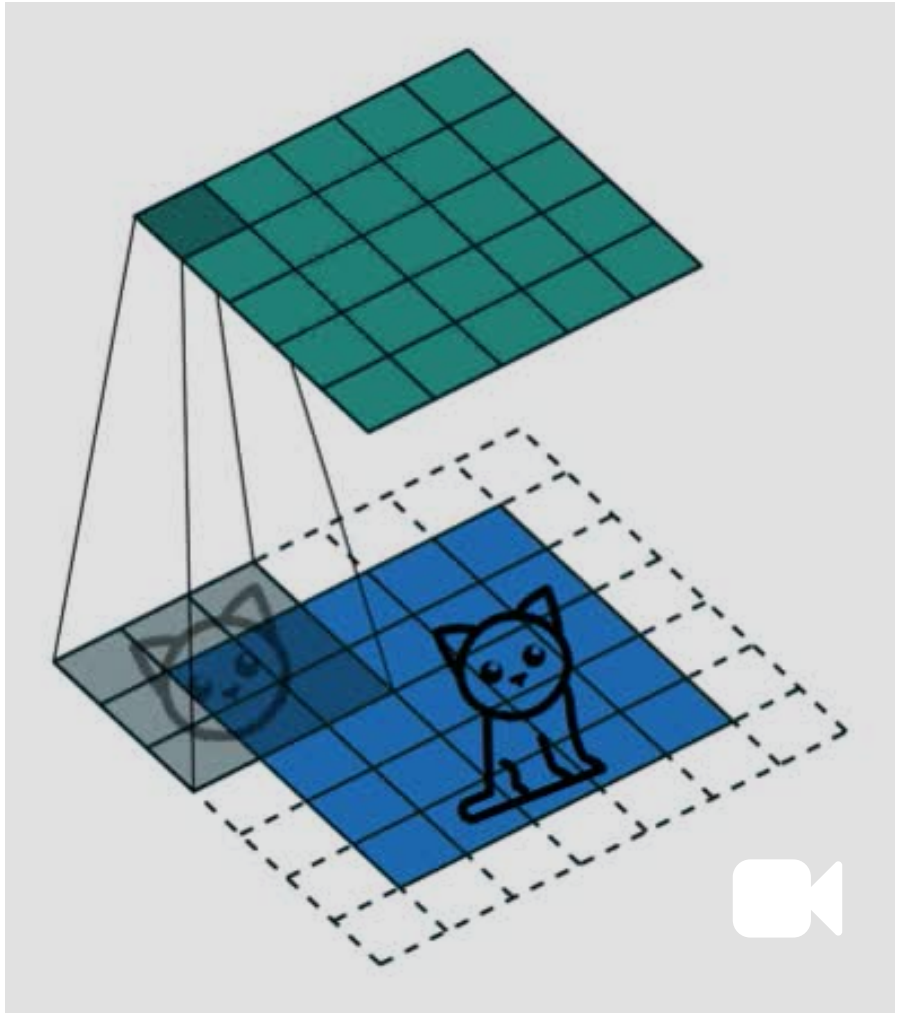


quick summary: hyperparameters for 2d convolution

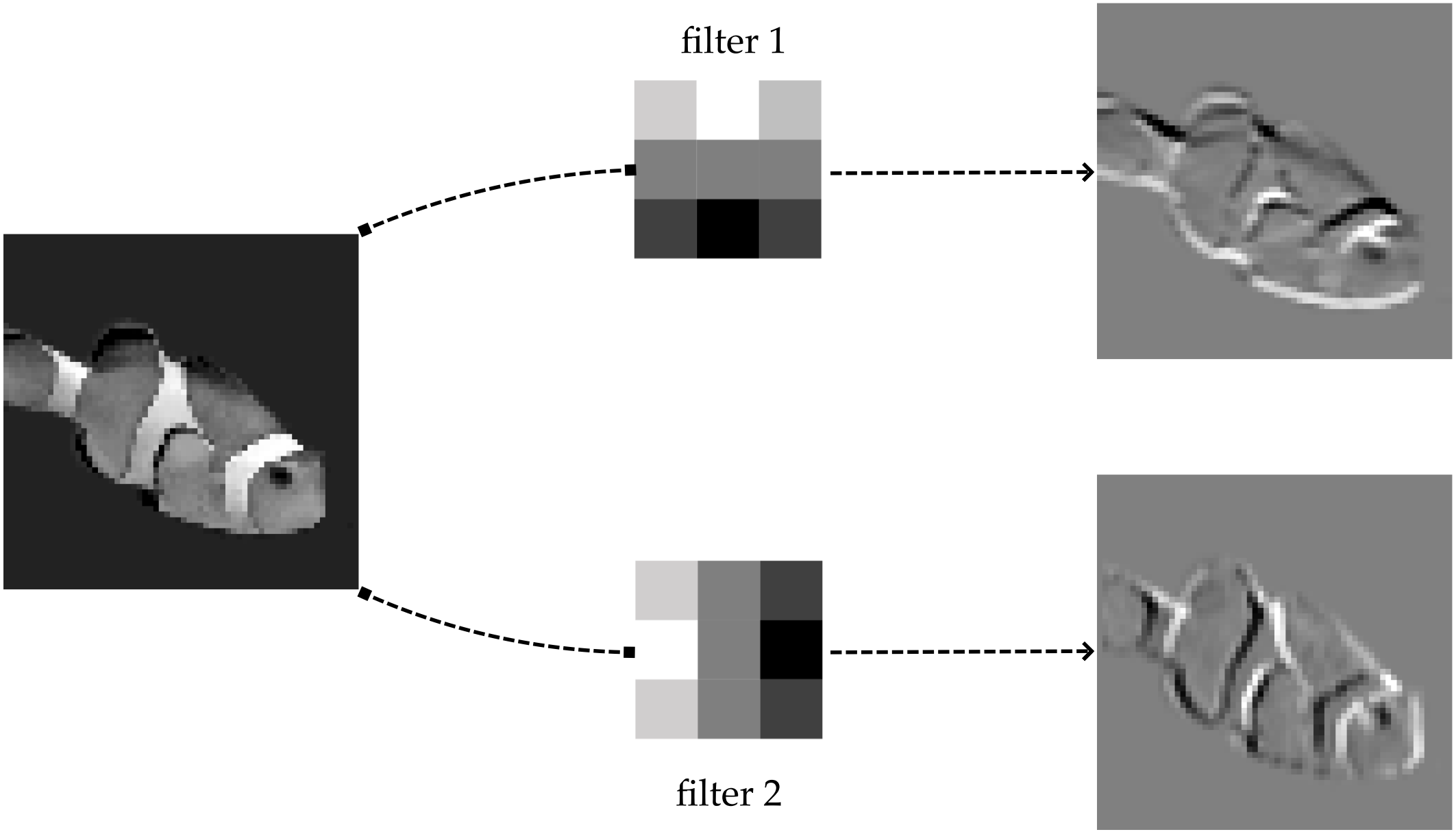
https://shenshen.mit.edu/demos/cnn/hyperparameters.html

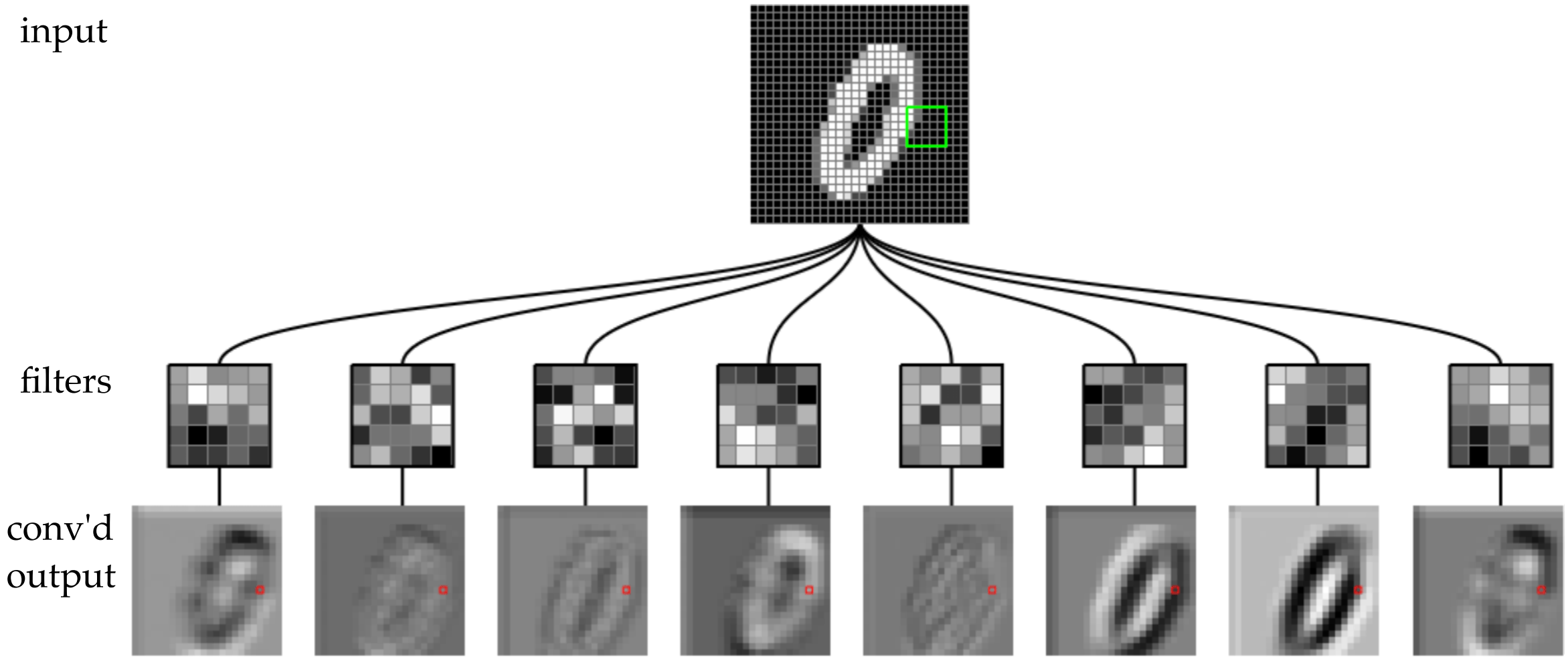
quick summary: convolution interpretation

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



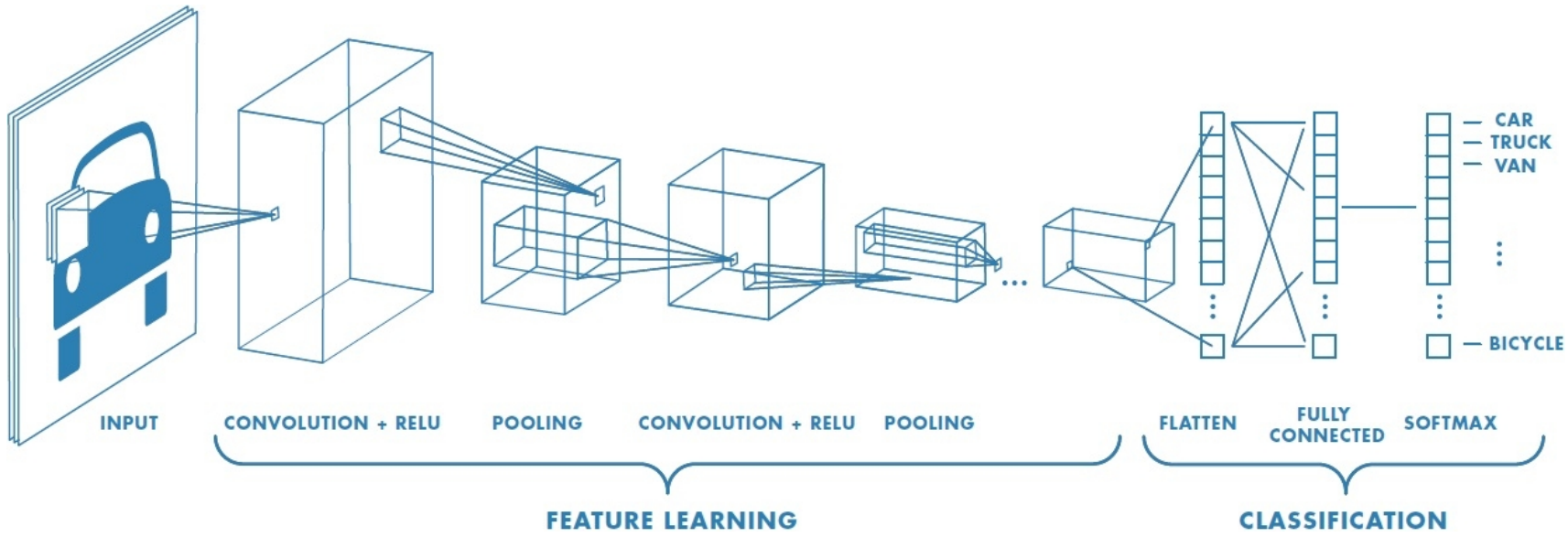
[video credit Lena Voita]


filter 1
filter 2

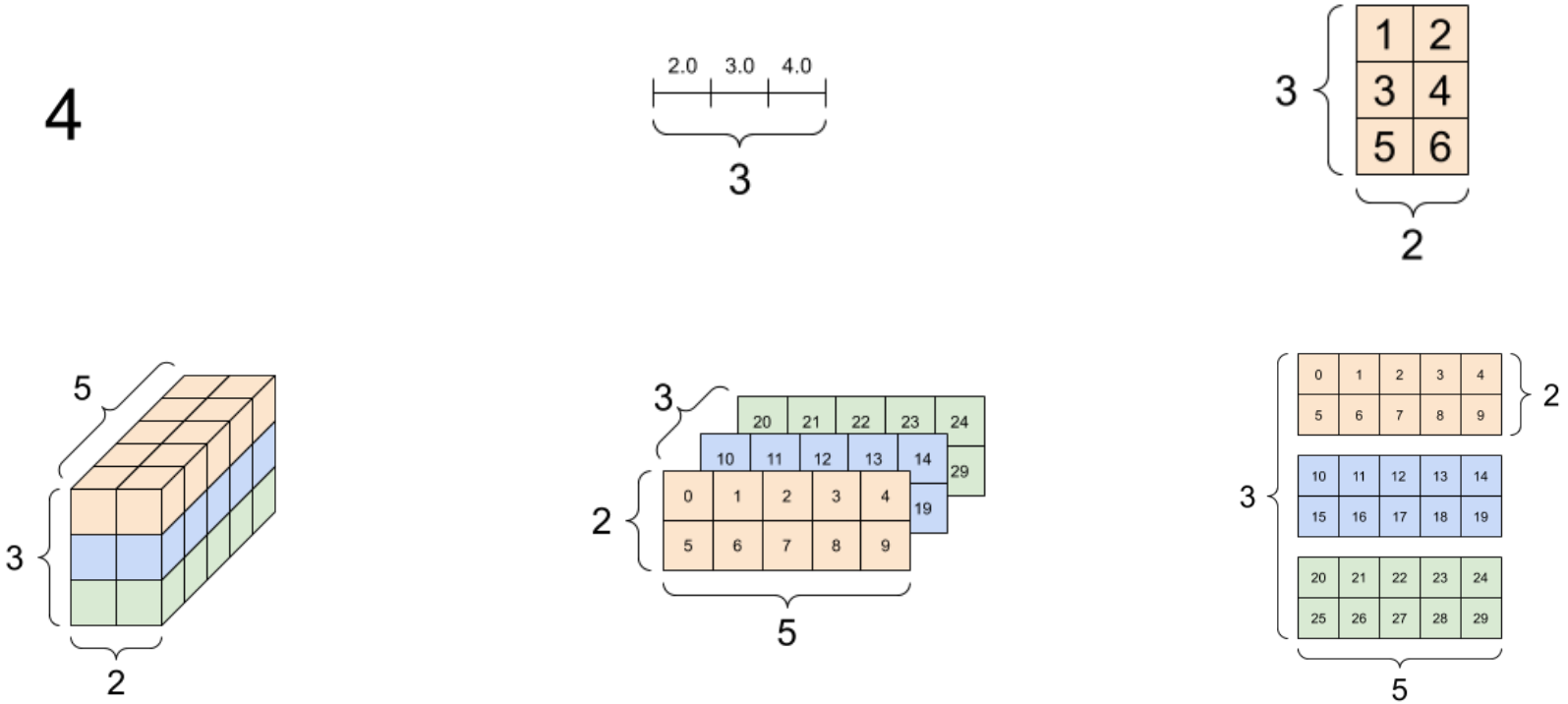

input
filters
conv'd
output

Outline

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



A tender intro to tensor:



[image credit: tensorflow]

color images and channels

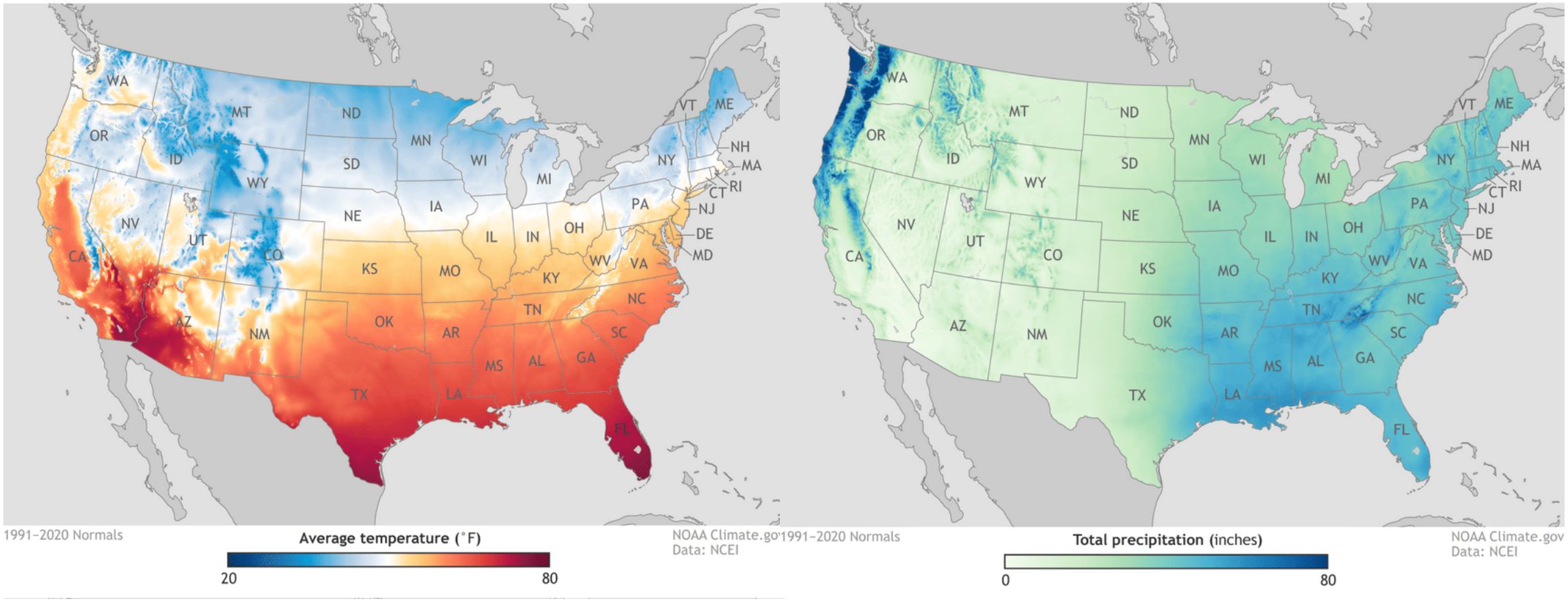
red

green

blue

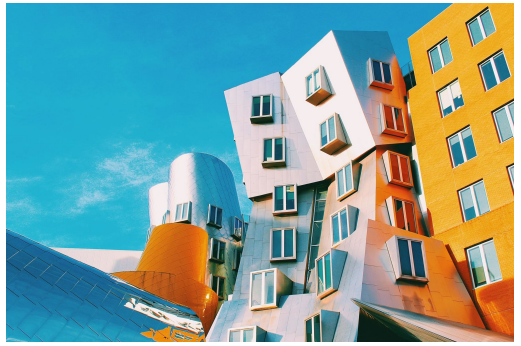
[Photo by Zayn Shah, Unsplash]

each channel encodes a *holistic but independent* perspective of the same image, similar to:



so channels are often referred to as *feature maps*

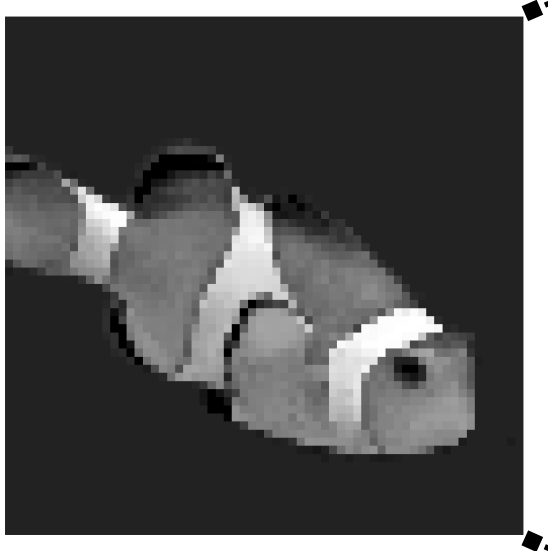
3d tensors from color channels

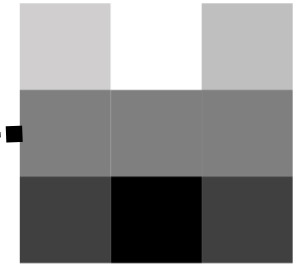


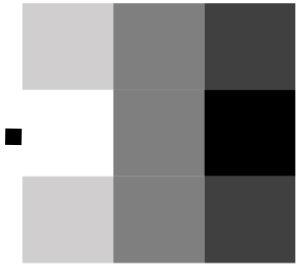
[Photo by Zayn Shah, Unsplash]

3d tensors from multiple filters

filter 1

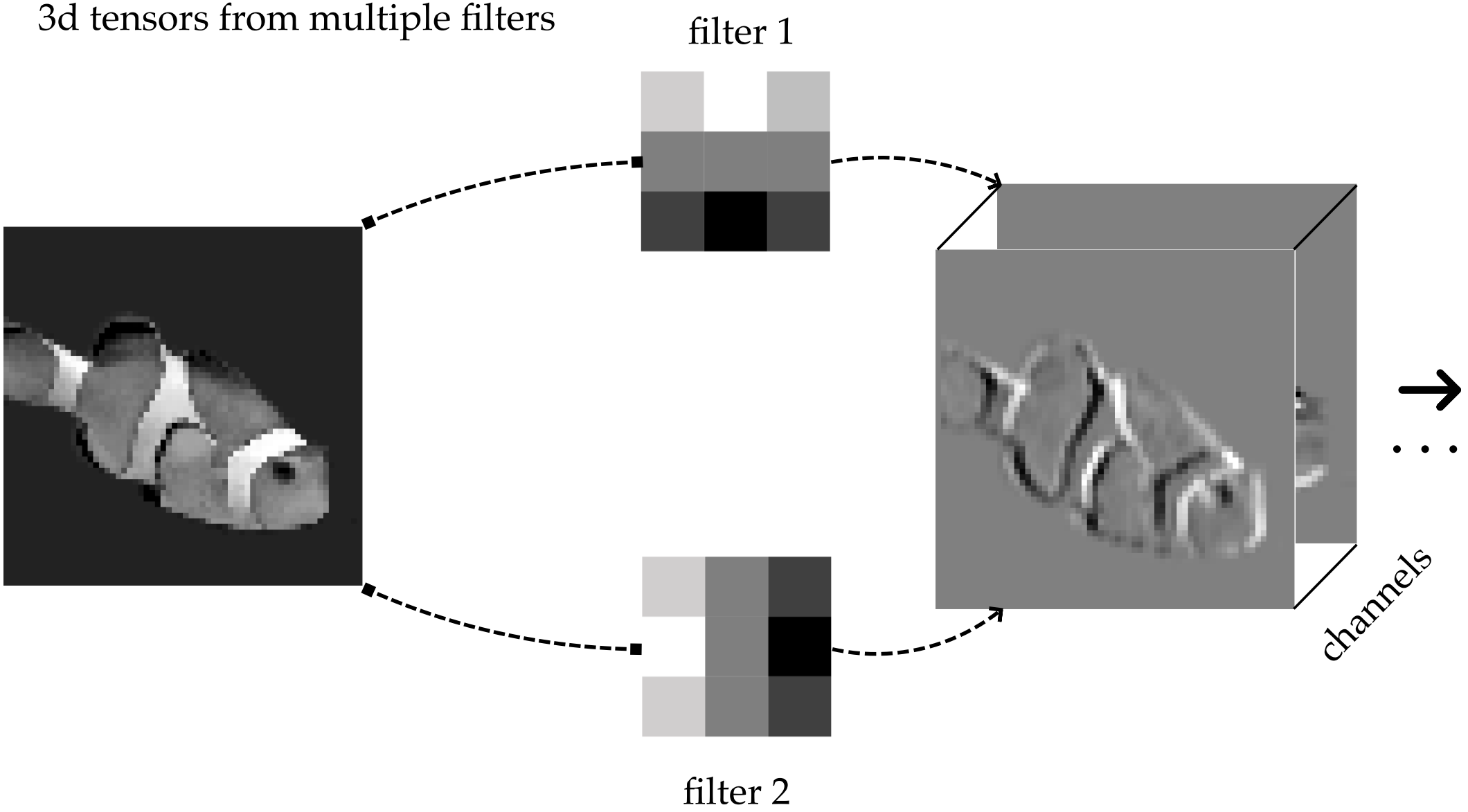






filter 2

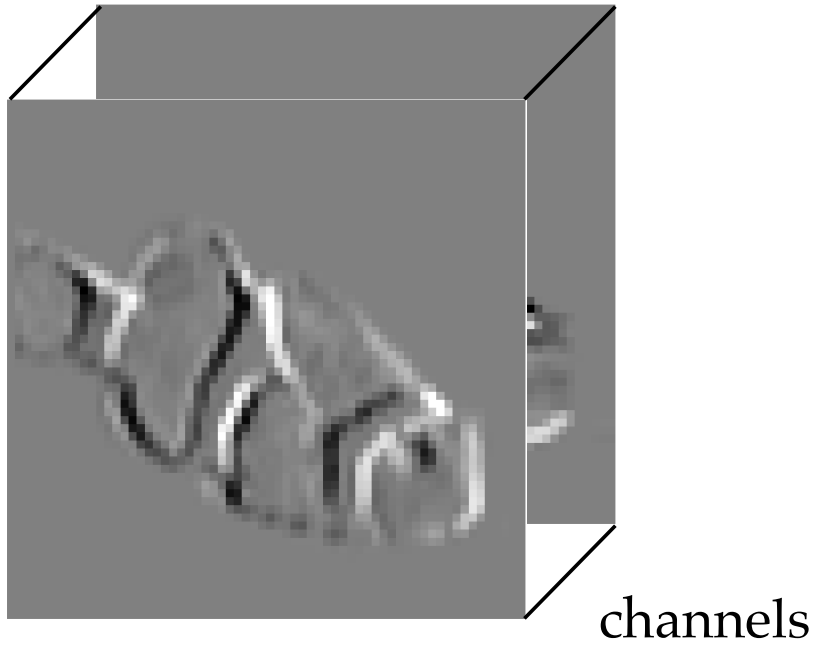
3d tensors from multiple filters



Why 3d tensors:

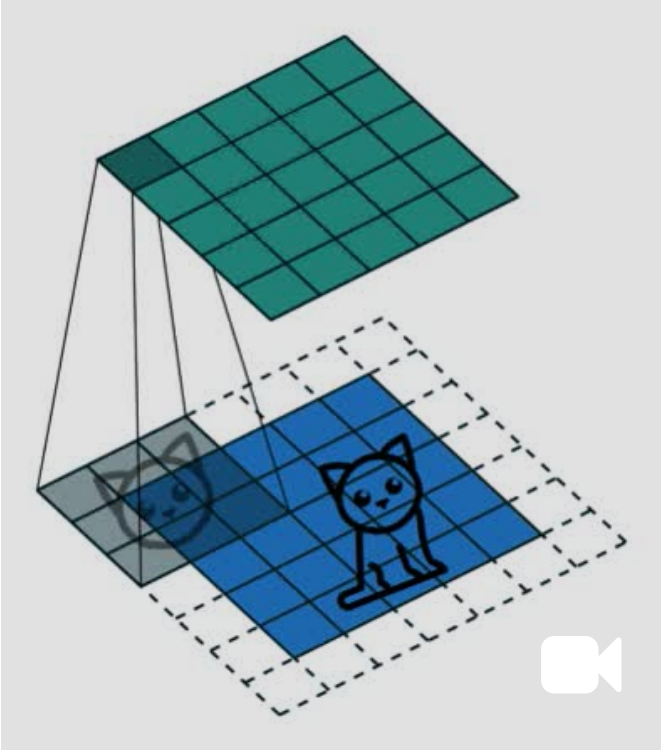
1. color input

2. using multiple filters

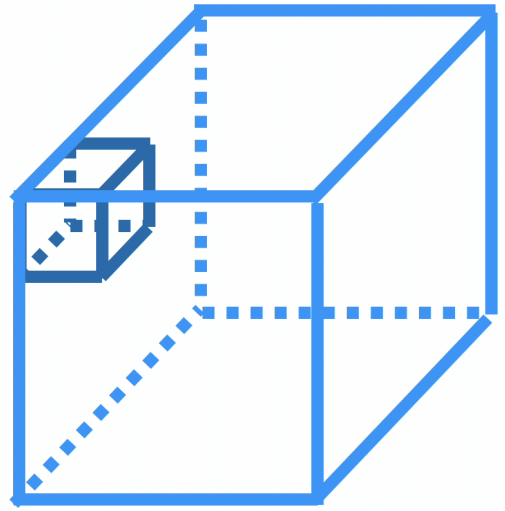


Why we *don't* typically do 3d convolution

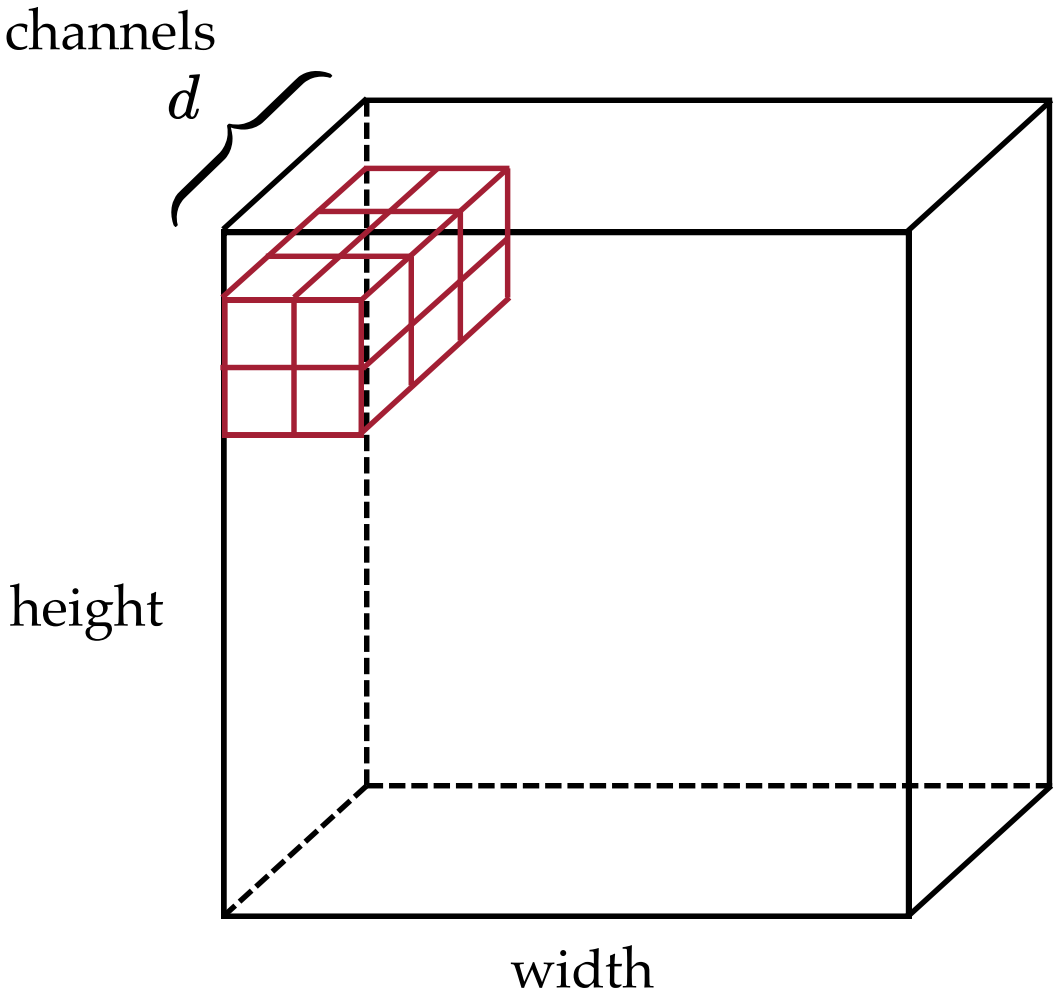
2d convolution



3d convolution

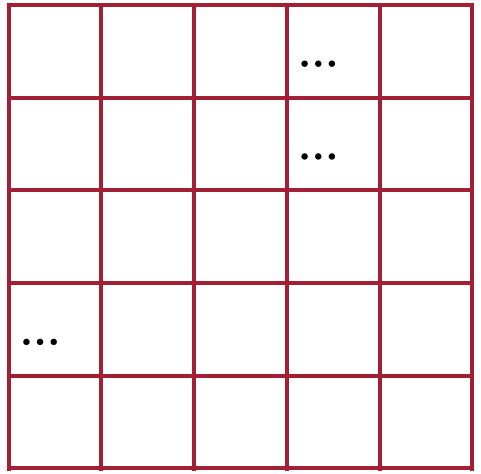


We *don't* typically do 3-dimensional convolution. Instead:

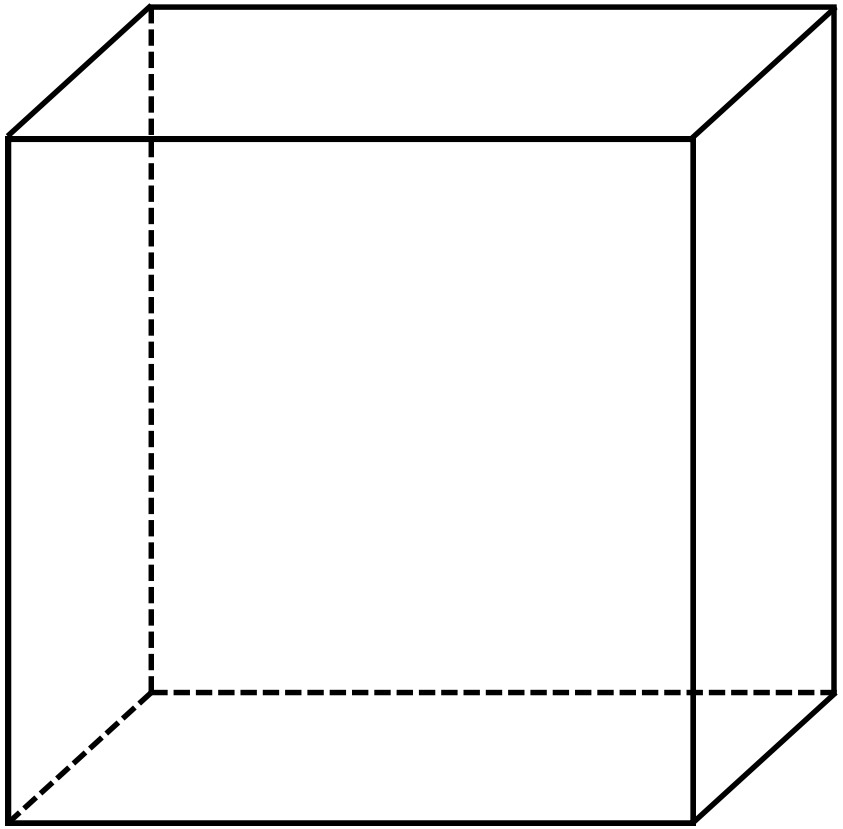


- 3d tensor input, channel d
- 3d tensor filter, channel d
- 2d convolution, 2d output

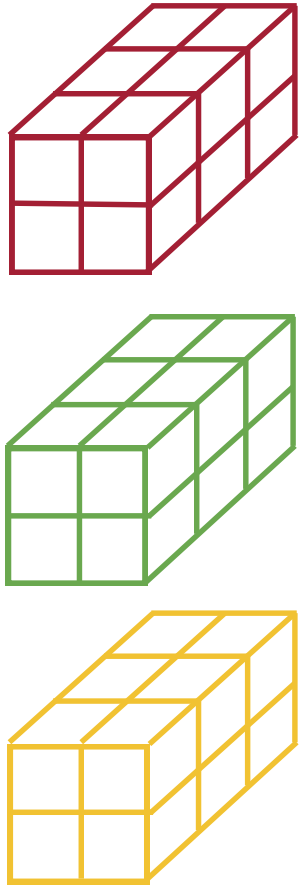
output



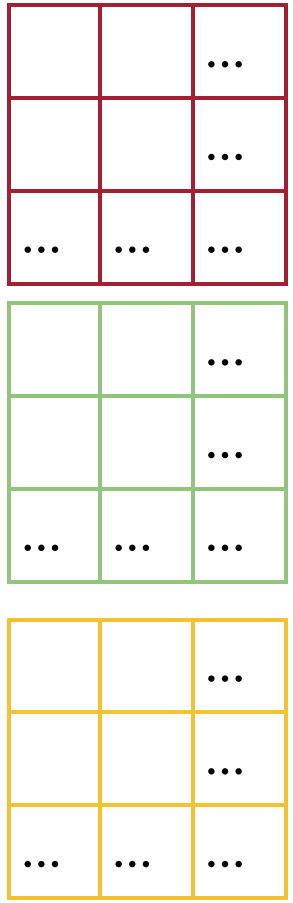
input tensor



multiple filters



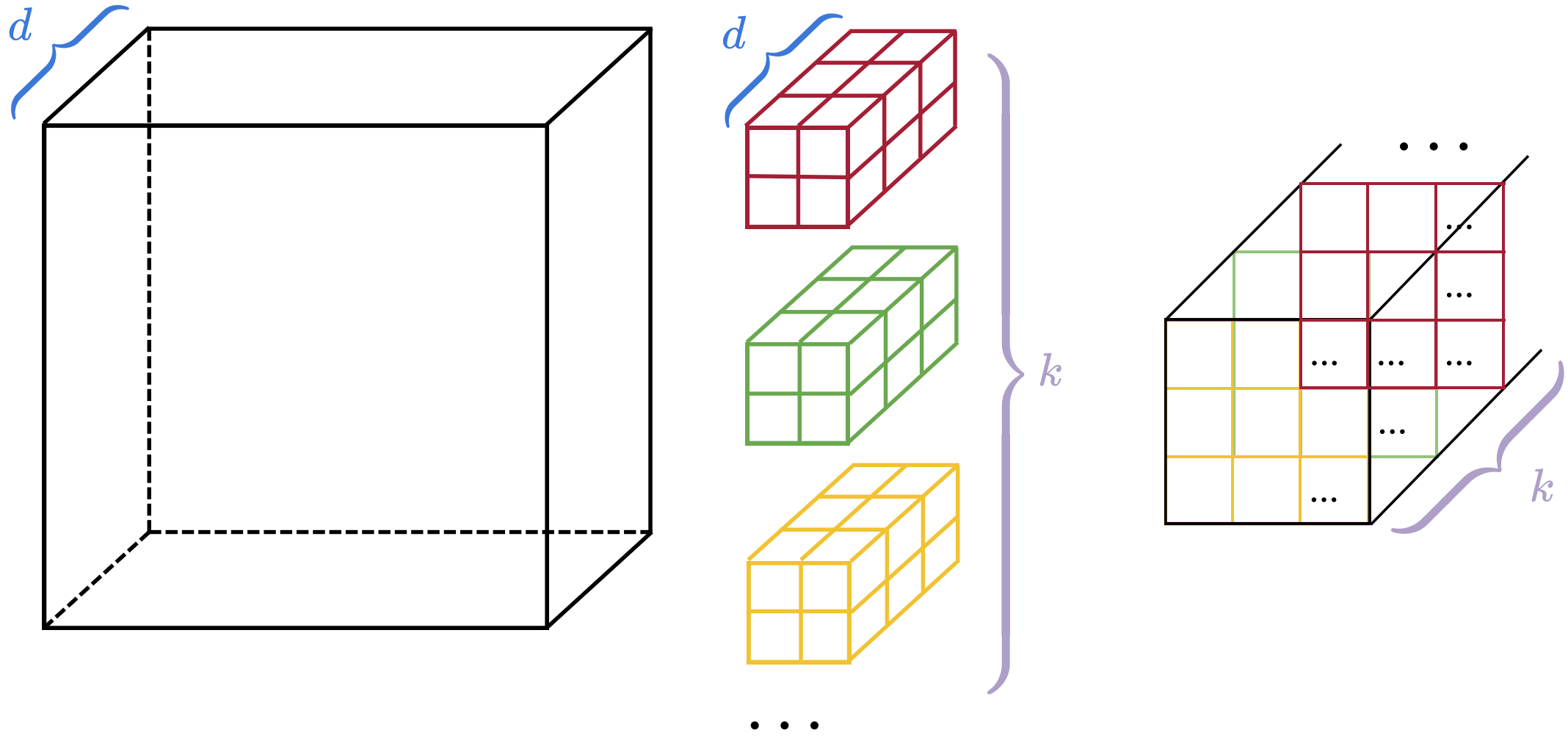
multiple output matrices



input tensor

k filters

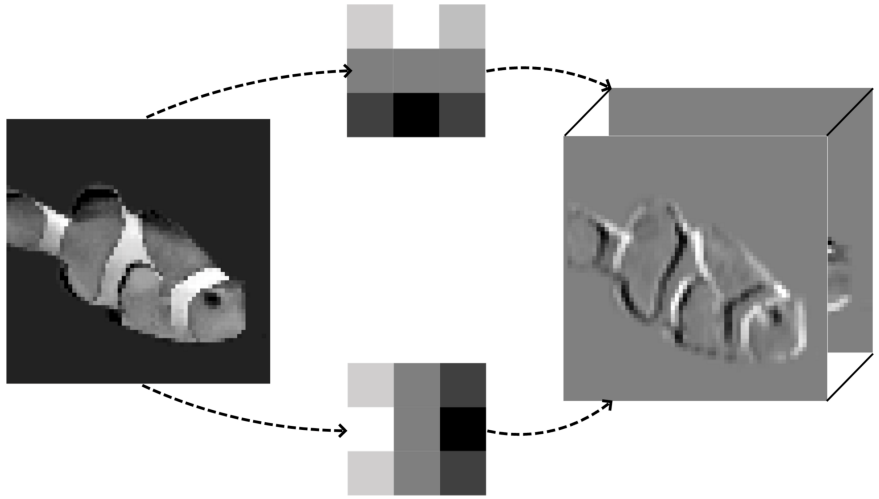
output tensor

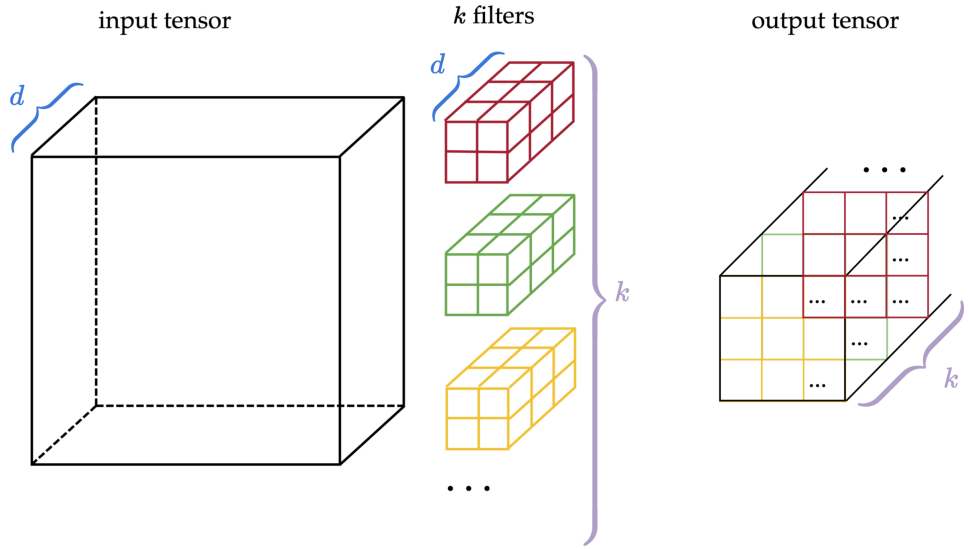


Every convolutional layer works with 3d tensors:

1. color input

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



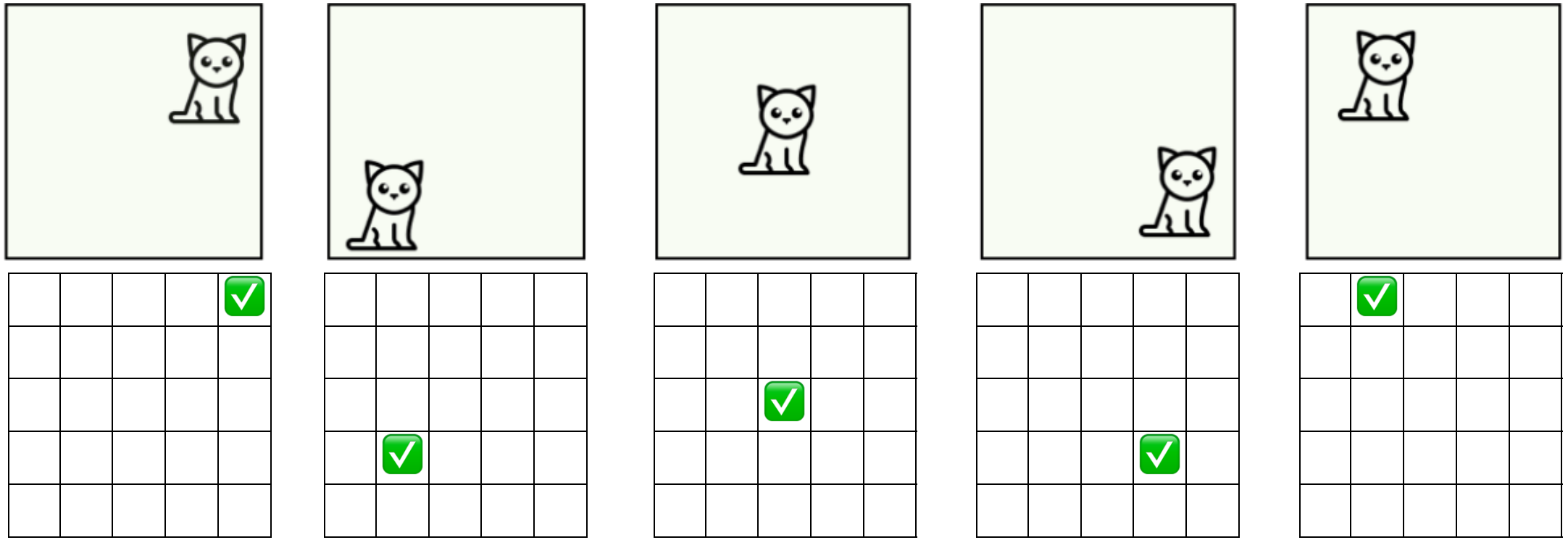


https://distill.pub/2017/feature-visualization/appendix

Outline

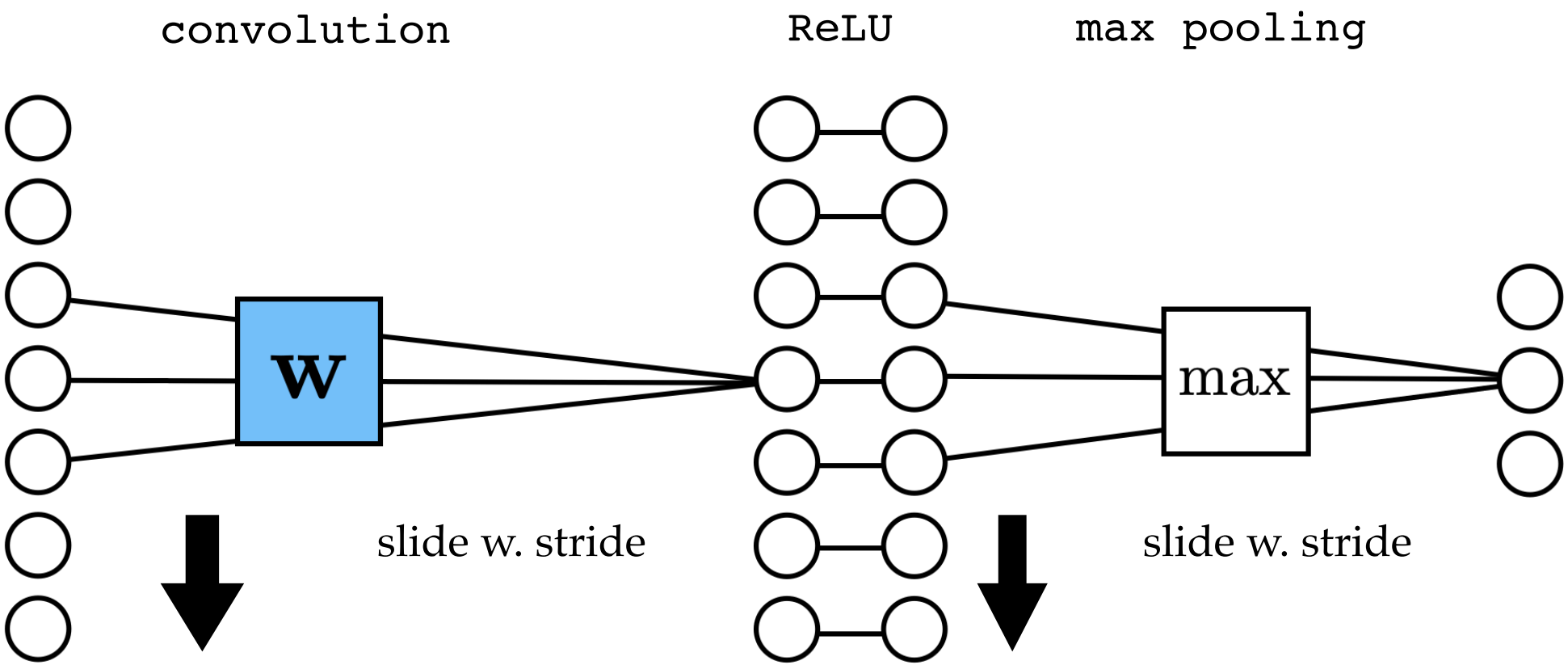
- Vision problem structure
- Convolution
  - 1-dimensional and 2-dimensional *convolution*
  - 3-dimensional *tensors*
- Max pooling
- (Case studies)

convolution helps detect pattern, but ...



cat moves, detection moves

1d max pooling



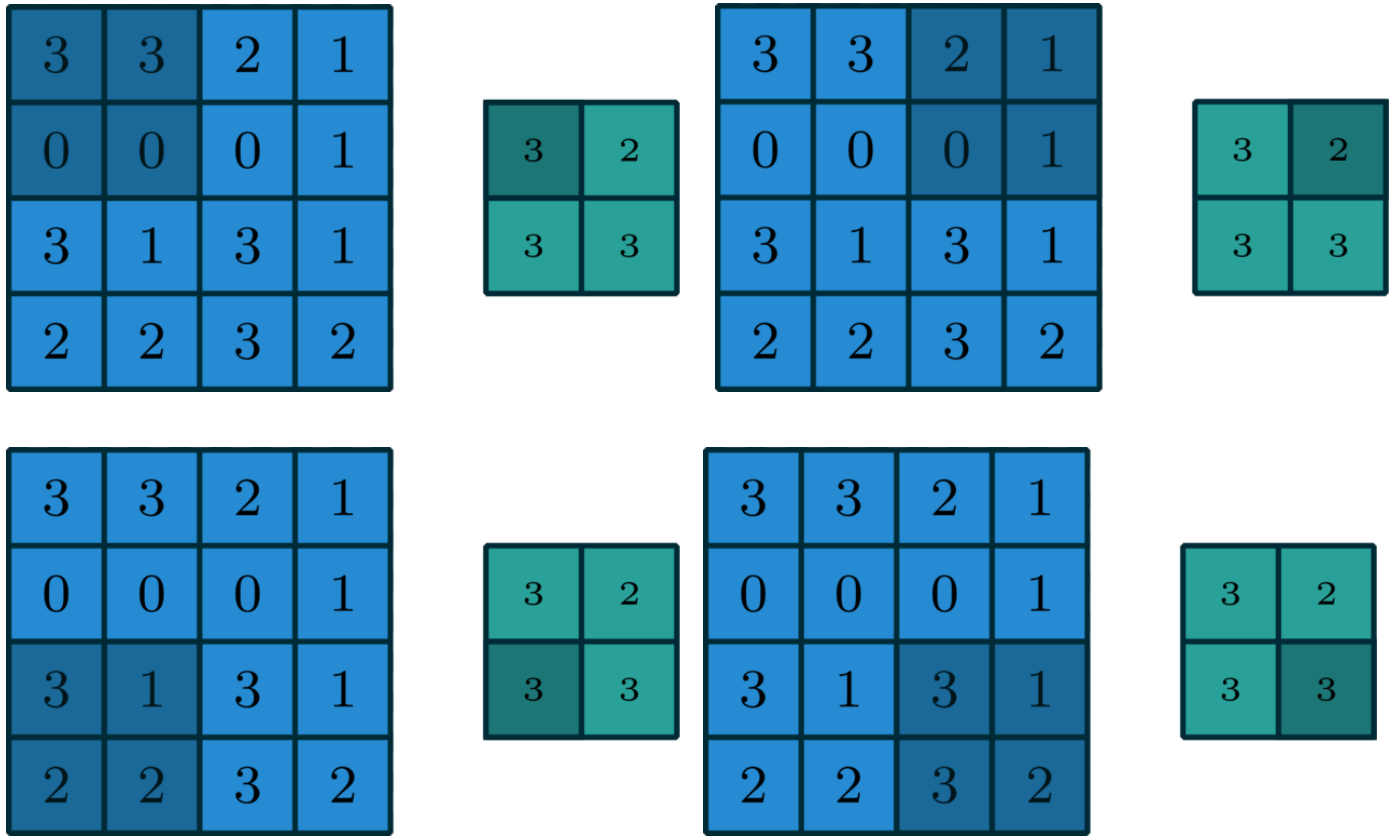
learnable filter weights

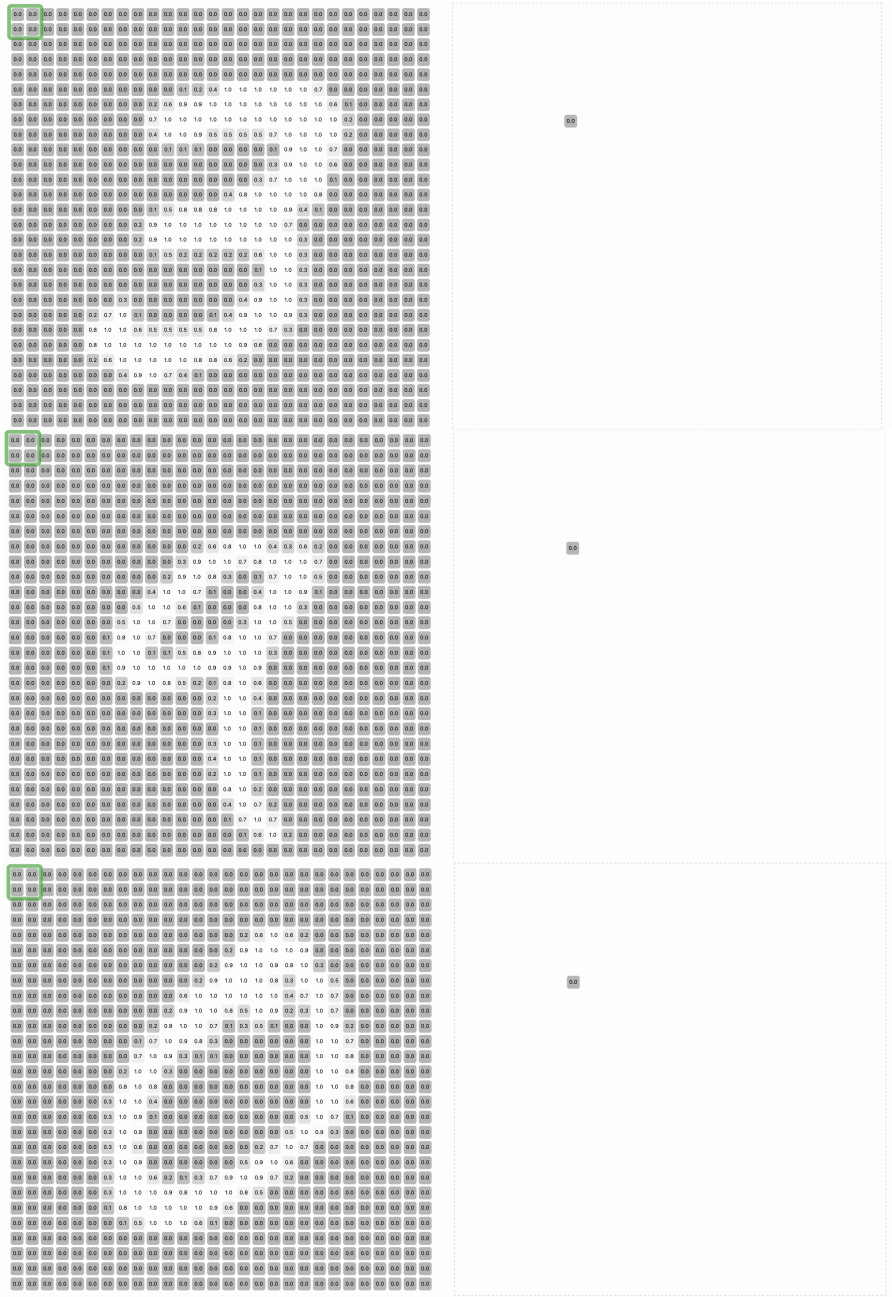
detects pattern

no learnable parameter

summarizes strongest response

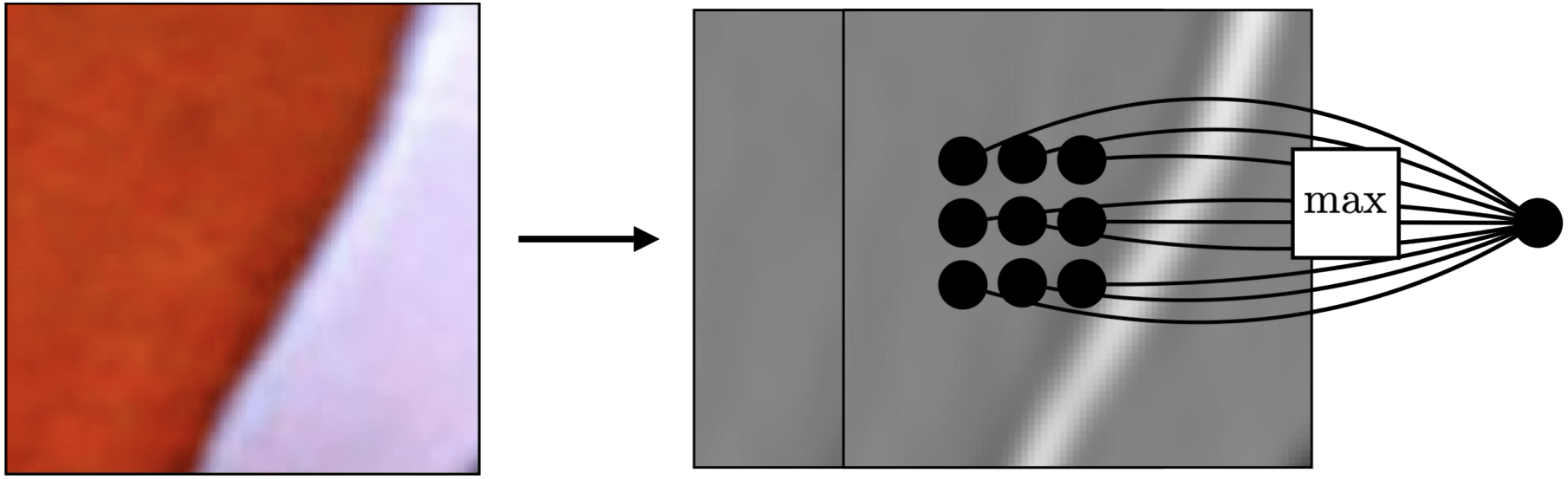
2d max pooling





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

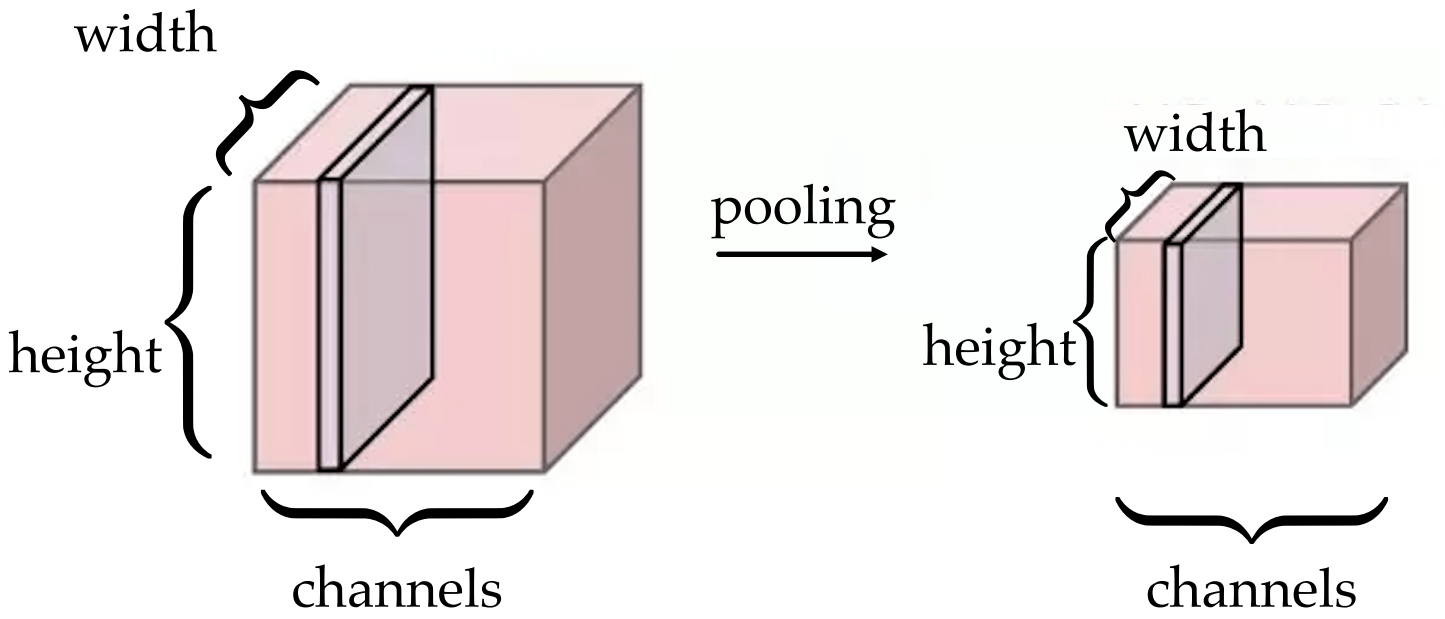
Pooling across *spatial* locations achieves invariance w.r.t. small translations:



large response regardless of exact position of edge

[image credit Philip Isola]

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

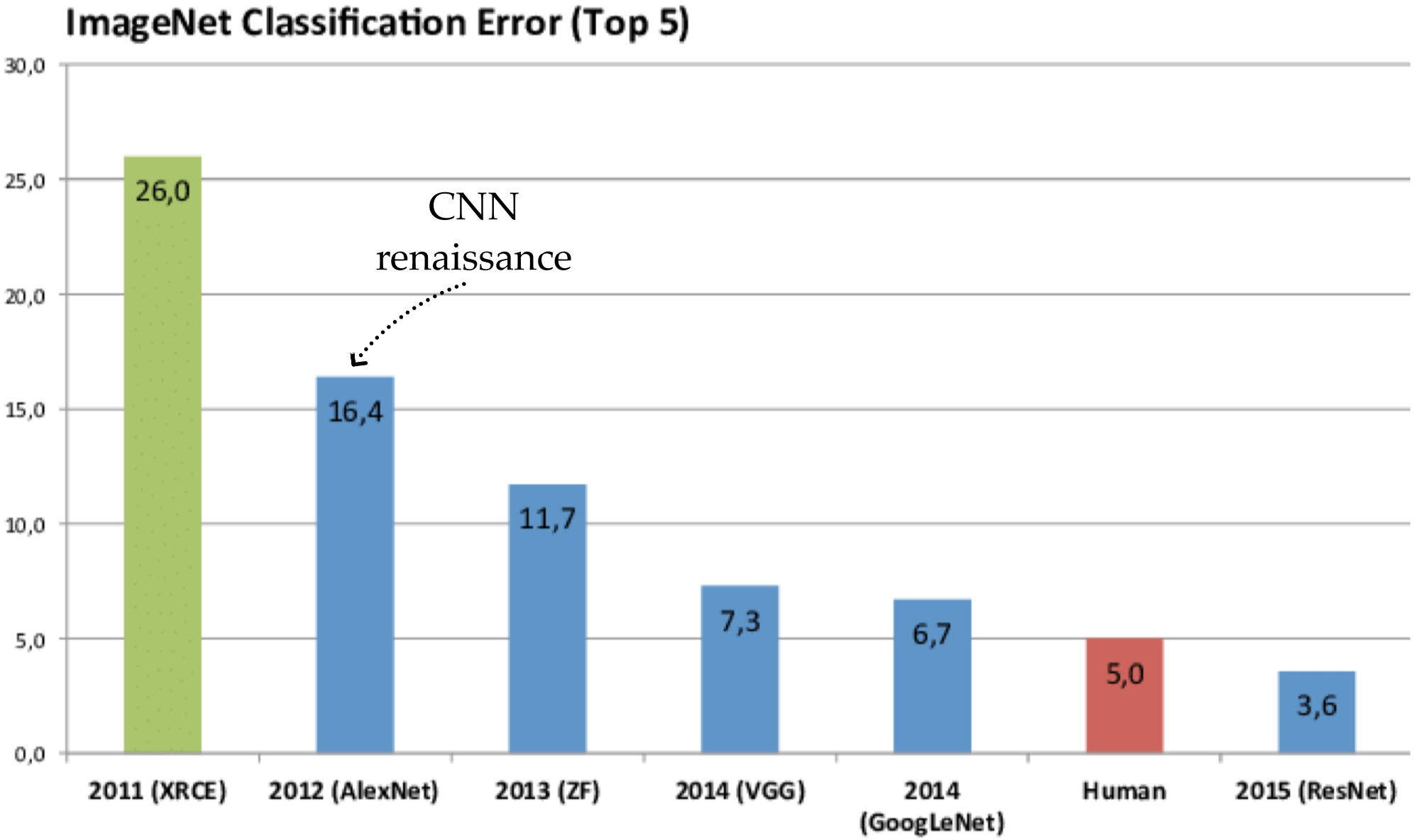


applied independently across all channels

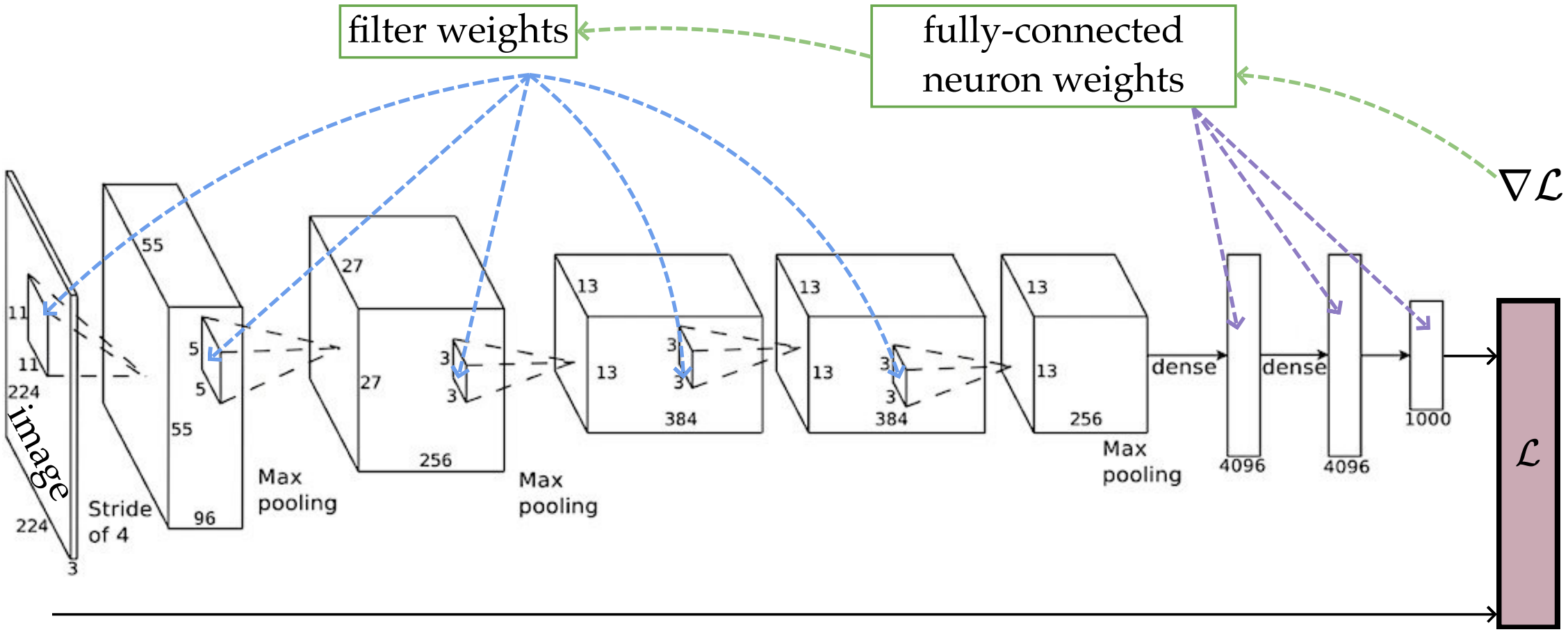
so the *channel* dimension remains *unchanged*

Outline

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



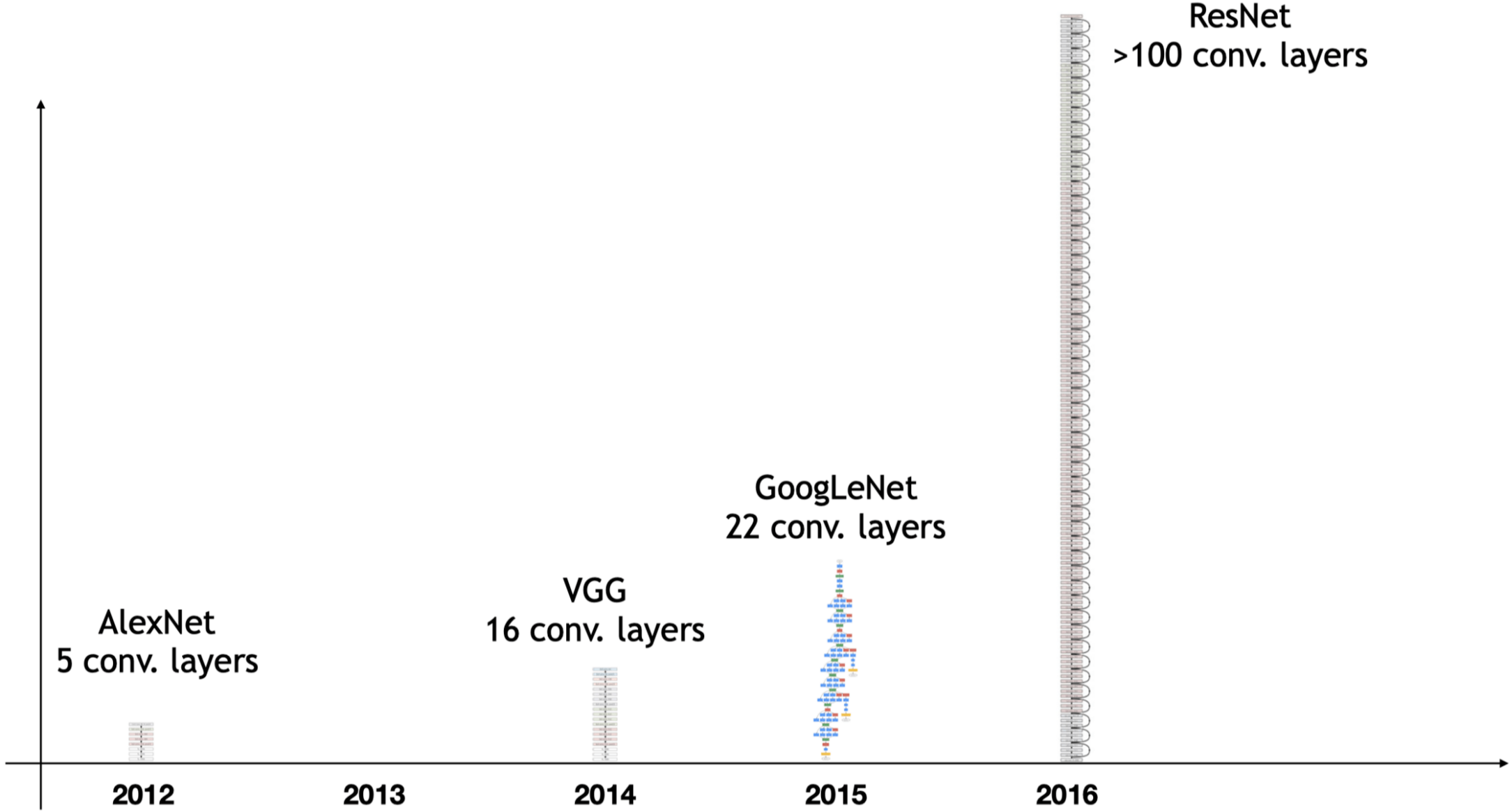
[image credit Philip Isola]



label

[AlexNet paper]

[all max pooling are via 3-by-3 filter, stride of 2]

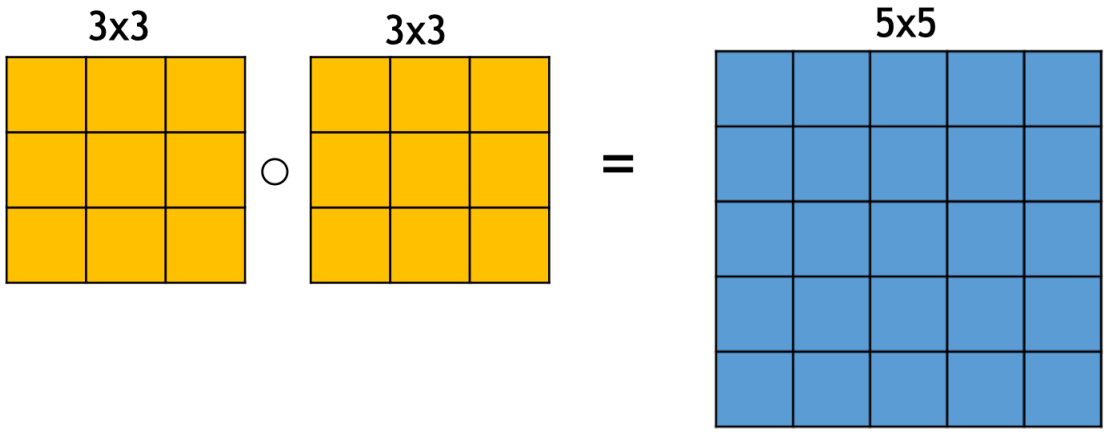


[image credit Philip Isola]

VGG '14

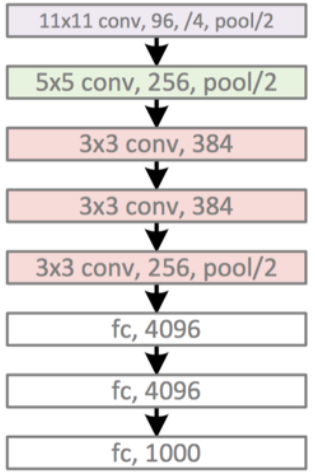
Main developments:

- small convolutional filters: only 3x3

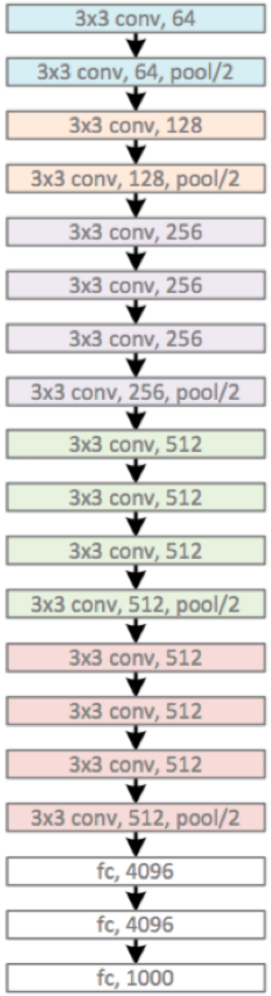


- increased depth: about 16 or 19 layers
- stack the same modules

AlexNet '12



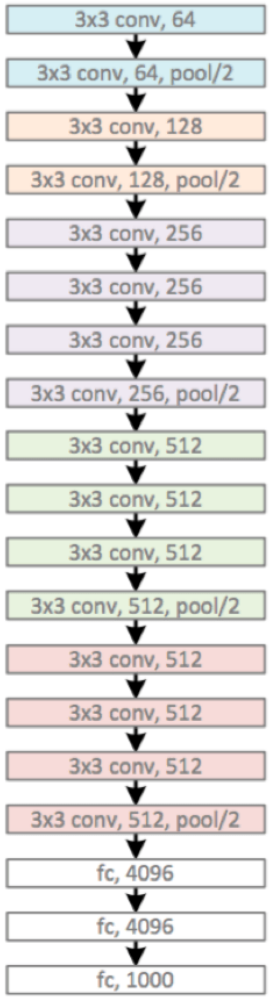
VGG '14



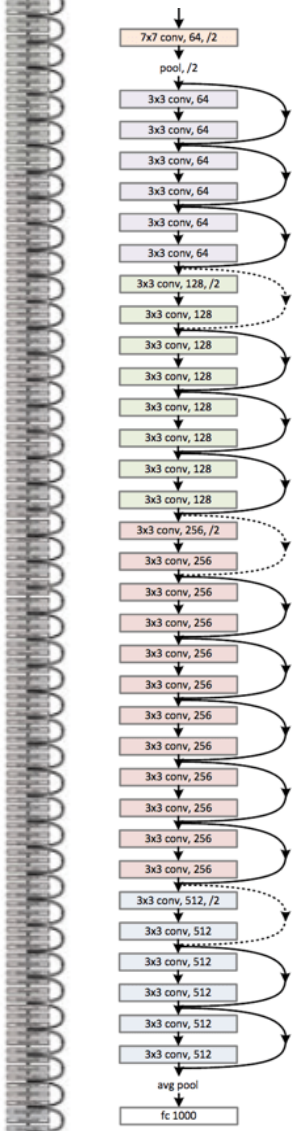
"Very Deep Convolutional Networks for Large-Scale Image Recognition", Simonyan & Zisserman. ICLR 2015

[image credit Philip Isola and Kaiming He]

VGG '14

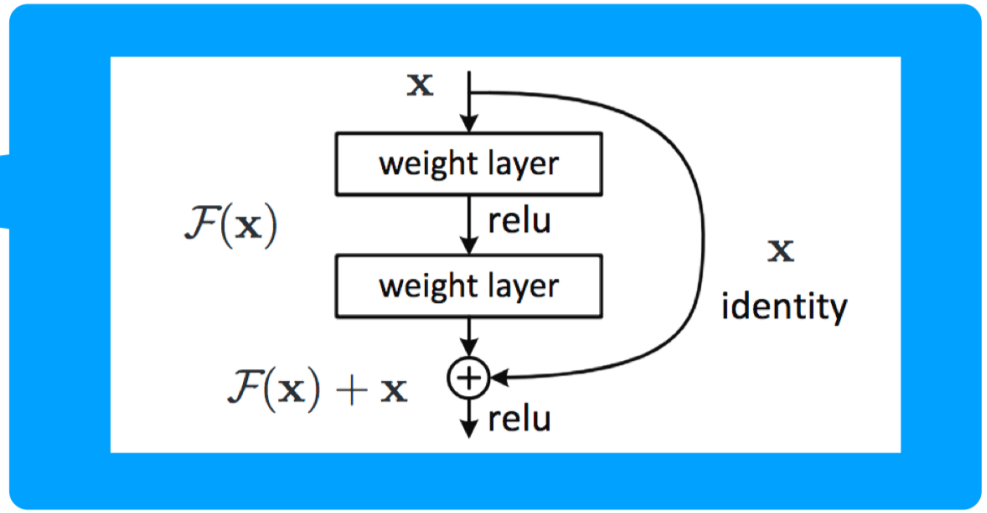


ResNet '16



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]

Summary

- Though NN are technically “universal approximators”, designing the NN structure so that it matches what we know about the underlying structure of the problem can substantially improve generalization ability and computational efficiency.
- Images are a very important input type and they have important properties that we can take advantage of: visual hierarchy, translation invariance, spatial locality.
- Convolution is an important image-processing technique that builds on these ideas. It can be interpreted as locally connected network, with weight-sharing.
- Pooling layer helps aggregate local info effectively, achieving bigger receptive field.
- We can train the parameters in a convolutional filtering function using backprop and combine convolutional filtering operations with other neural-network layers.

https://forms.gle/36SX9pqCTWpp323N8

We'd love to hear
your thoughts.

Thanks!