https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 7: Convolutional Neural Networks

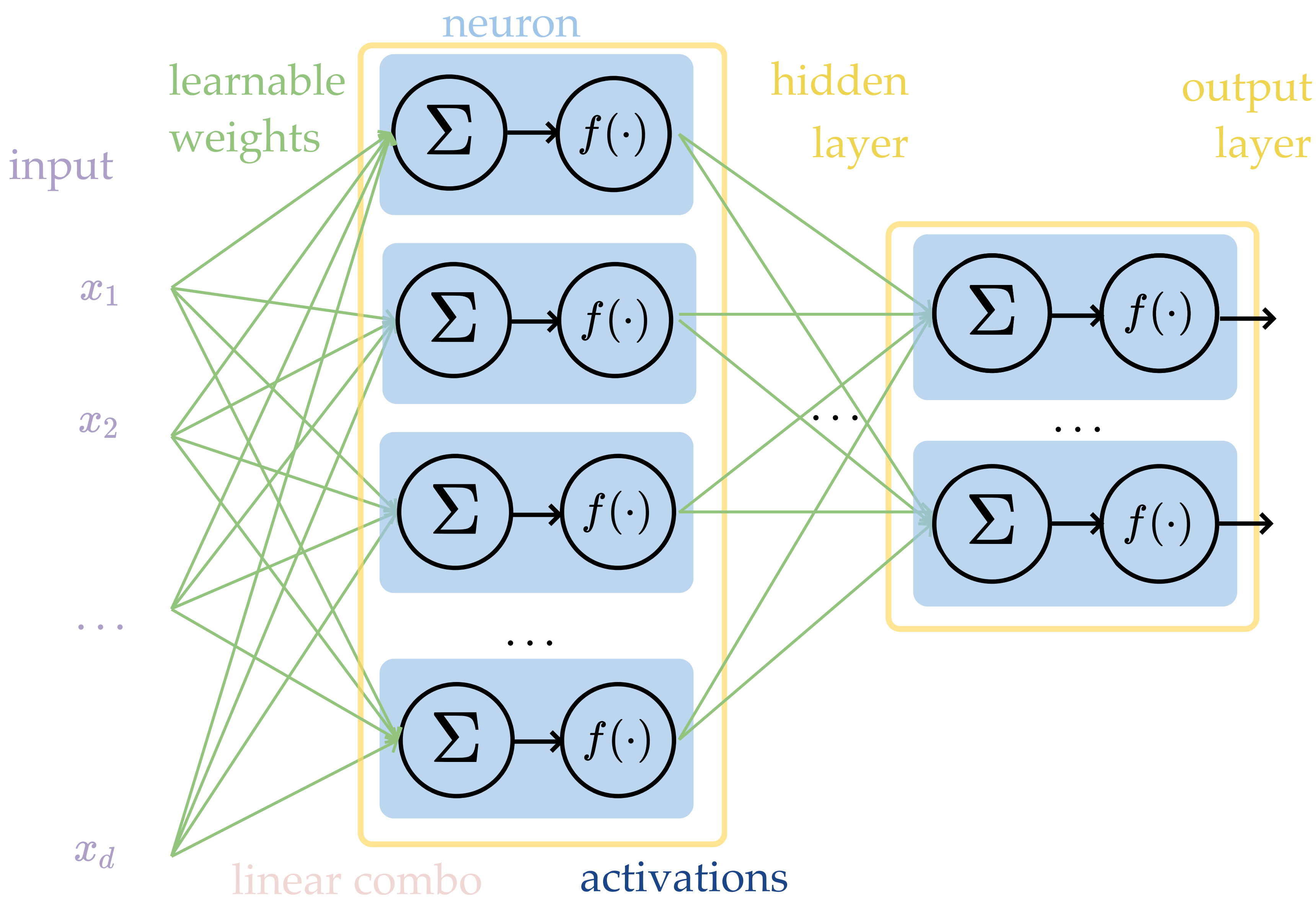
Shen Shen

Mar 16, 2026

3pm, Room 10-250

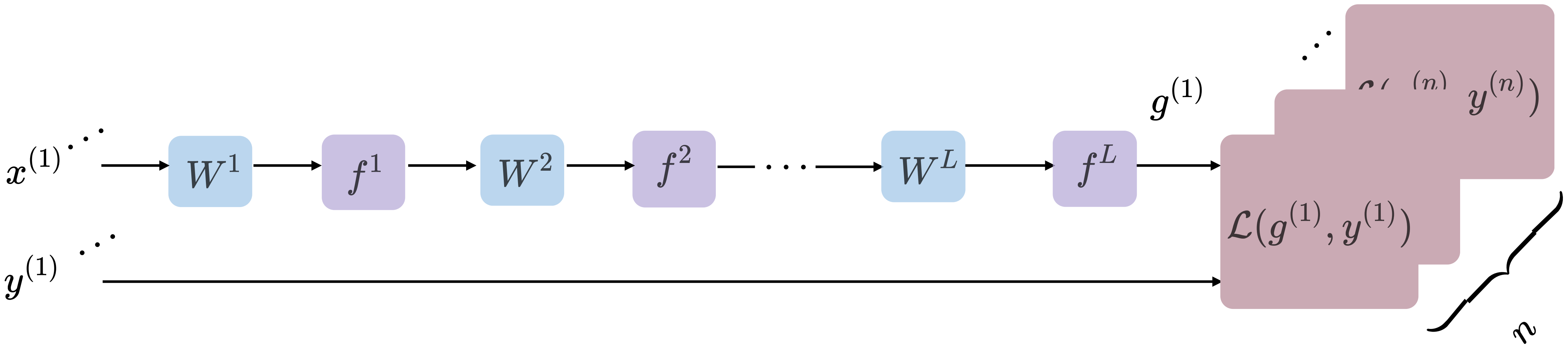
Slides and Lecture Recording

Recap a (fully-connected, feed-forward) neural network



Recap

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

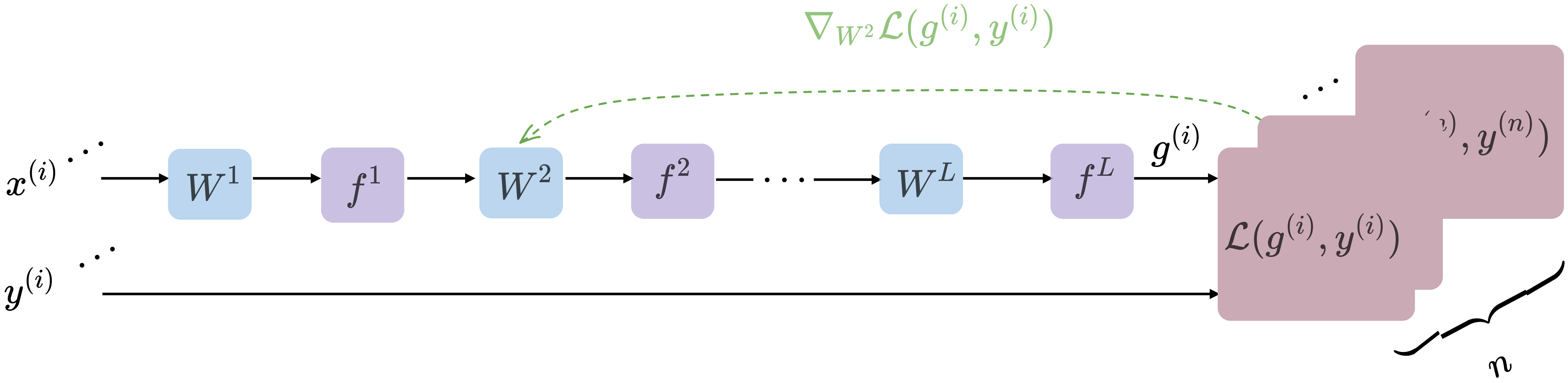
loss function

`Recap`

Backward pass: run SGD to update all parameters

e.g. to update W^2

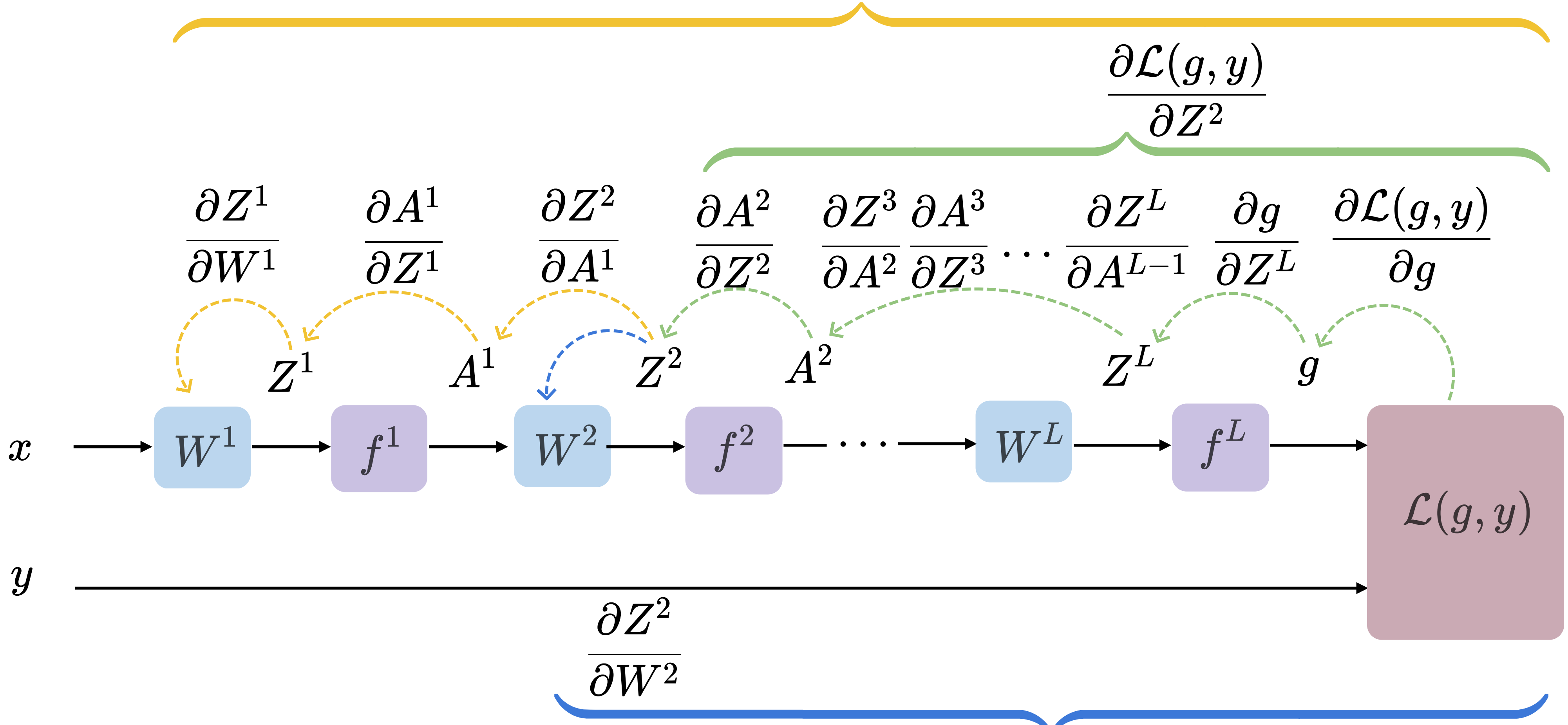
- Randomly pick a data point $(x^{(i)}, y^{(i)})$
- Evaluate the gradient $\nabla_{W^2}\mathcal{L}(g^{(i)}, y^{(i)})$
- Update the weights $W^2 \leftarrow W^2 - \eta\nabla_{W^2}\mathcal{L}(g^{(i)}, y^{(i)})$



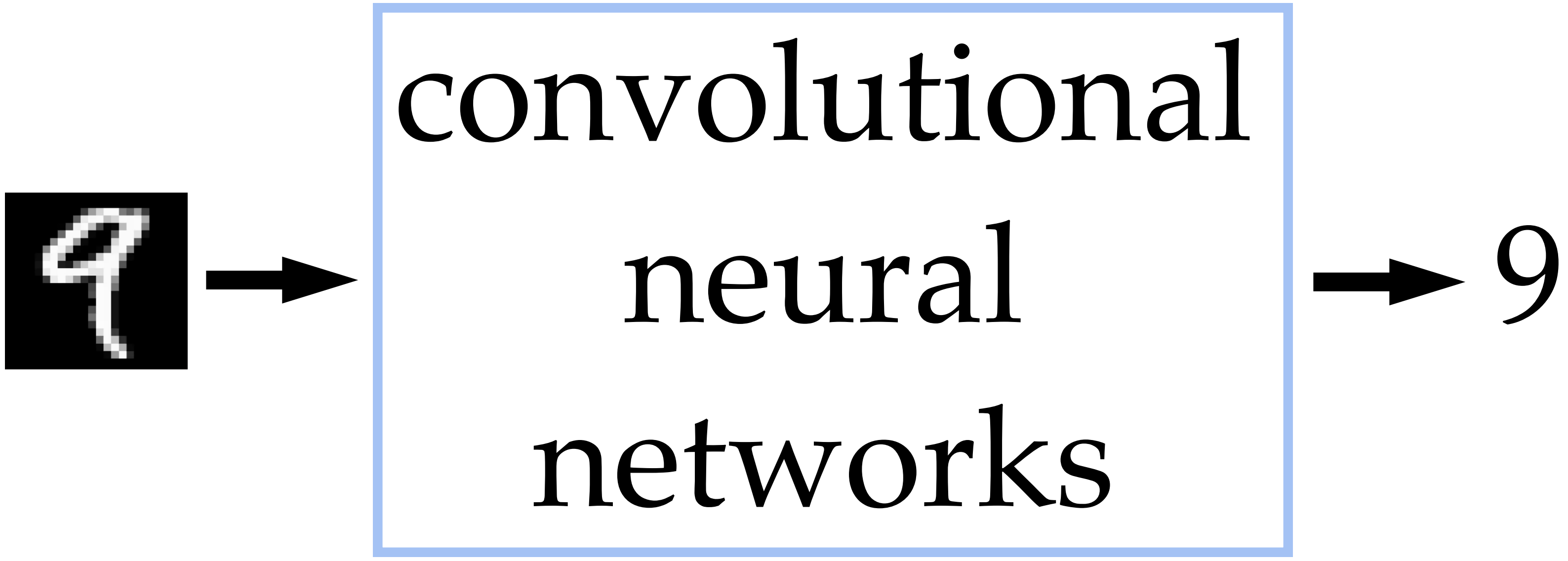
Recap

backpropagation: reuse of computation

$$\frac{\partial \mathcal{L}}{\partial W^1} = \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial Z^2}{\partial A^1} \cdot \frac{\partial \mathcal{L}}{\partial Z^2}$$



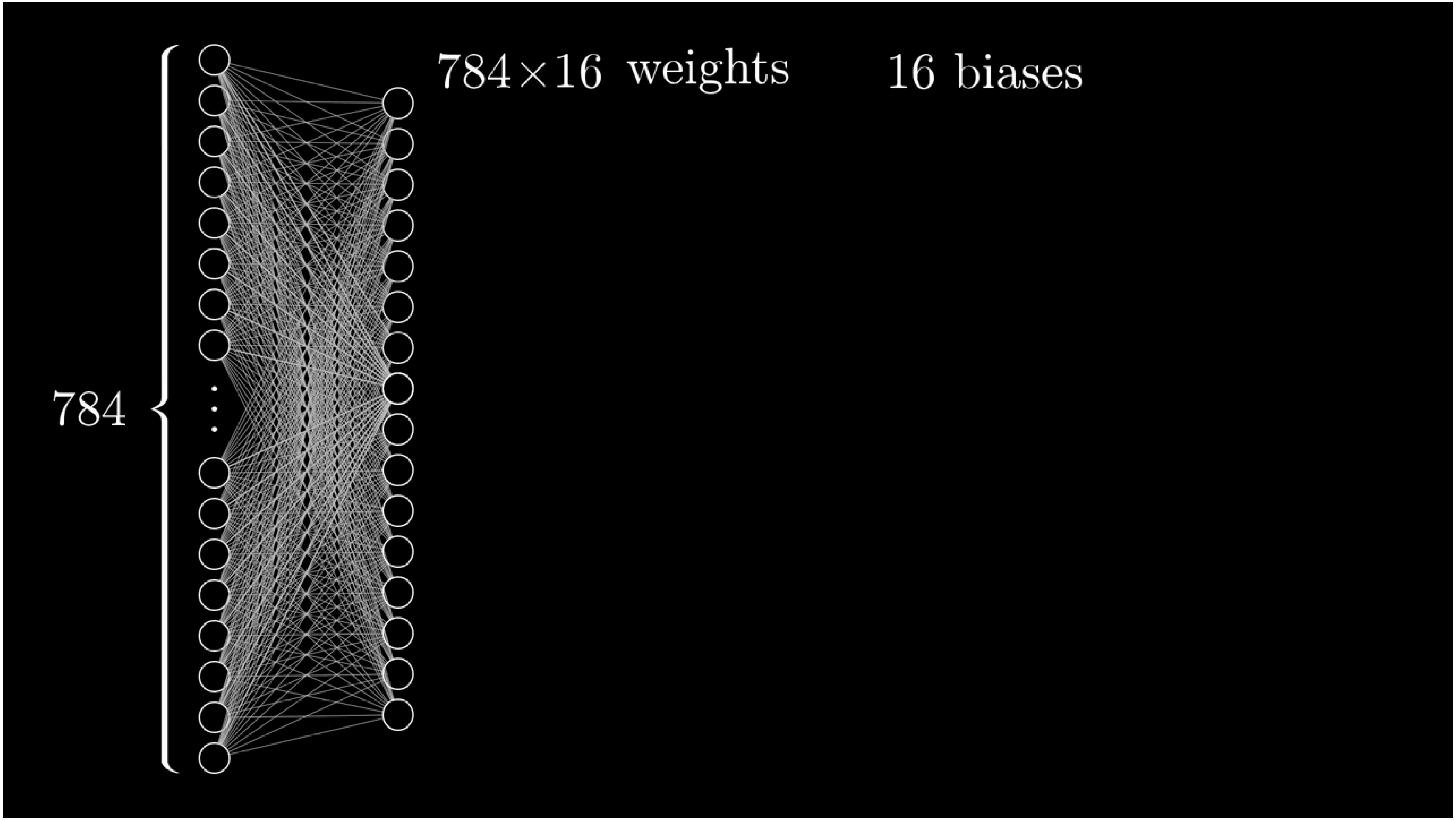
$$\frac{\partial \mathcal{L}}{\partial W^2} = \frac{\partial Z^2}{\partial W^2} \cdot \frac{\partial \mathcal{L}}{\partial Z^2}$$



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.

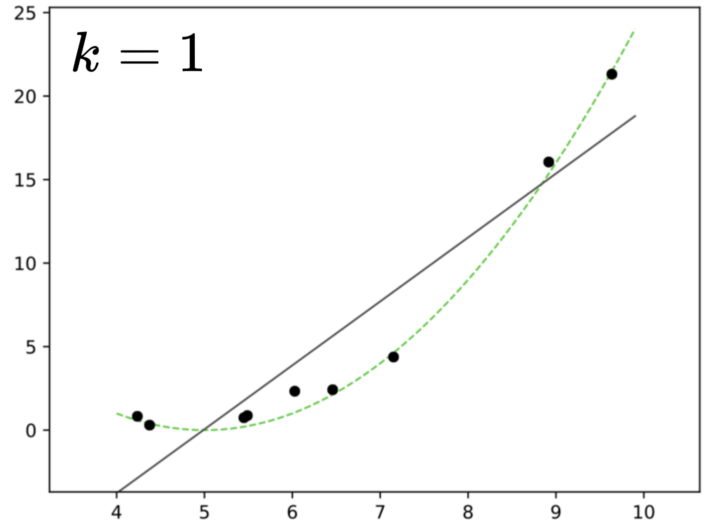
For higher-resolution images, or more complex tasks, or larger networks, the number of parameters can grow very fast.

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

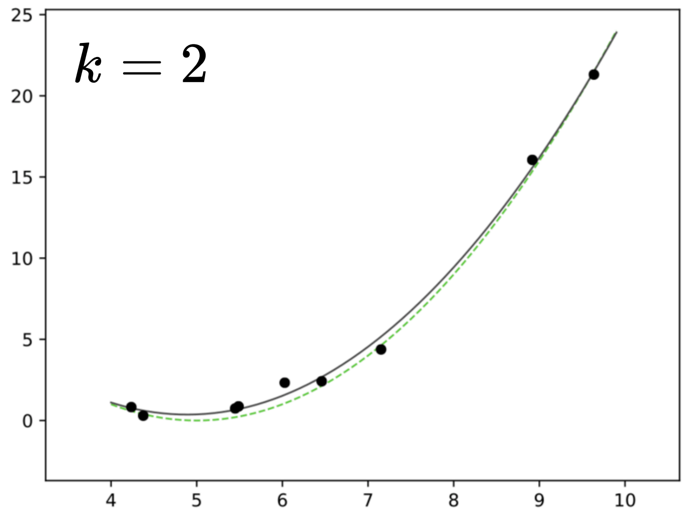
- Partly, fully-connected nets don't scale well for vision tasks
- More importantly, a carefully chosen hypothesis class helps fight overfitting

Recall, models with needless parameters tend to overfit

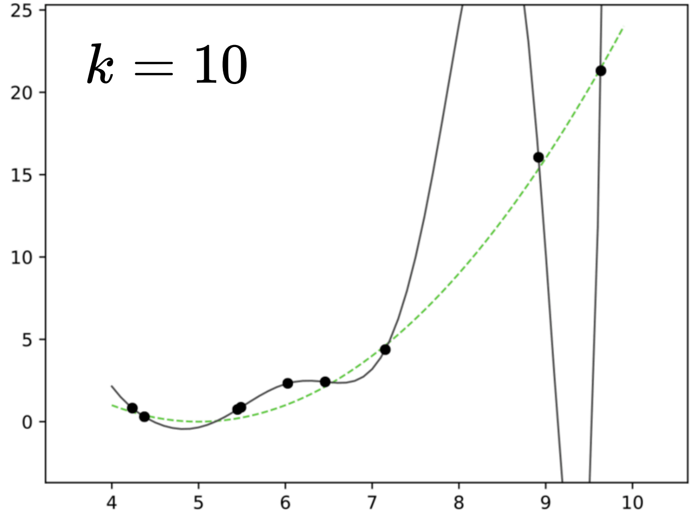
Underfitting



Appropriate



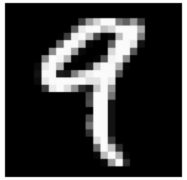
Overfitting



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

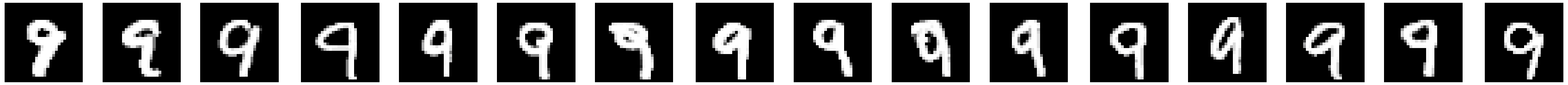
so... do we know anything about vision problems?

Why do we humans think

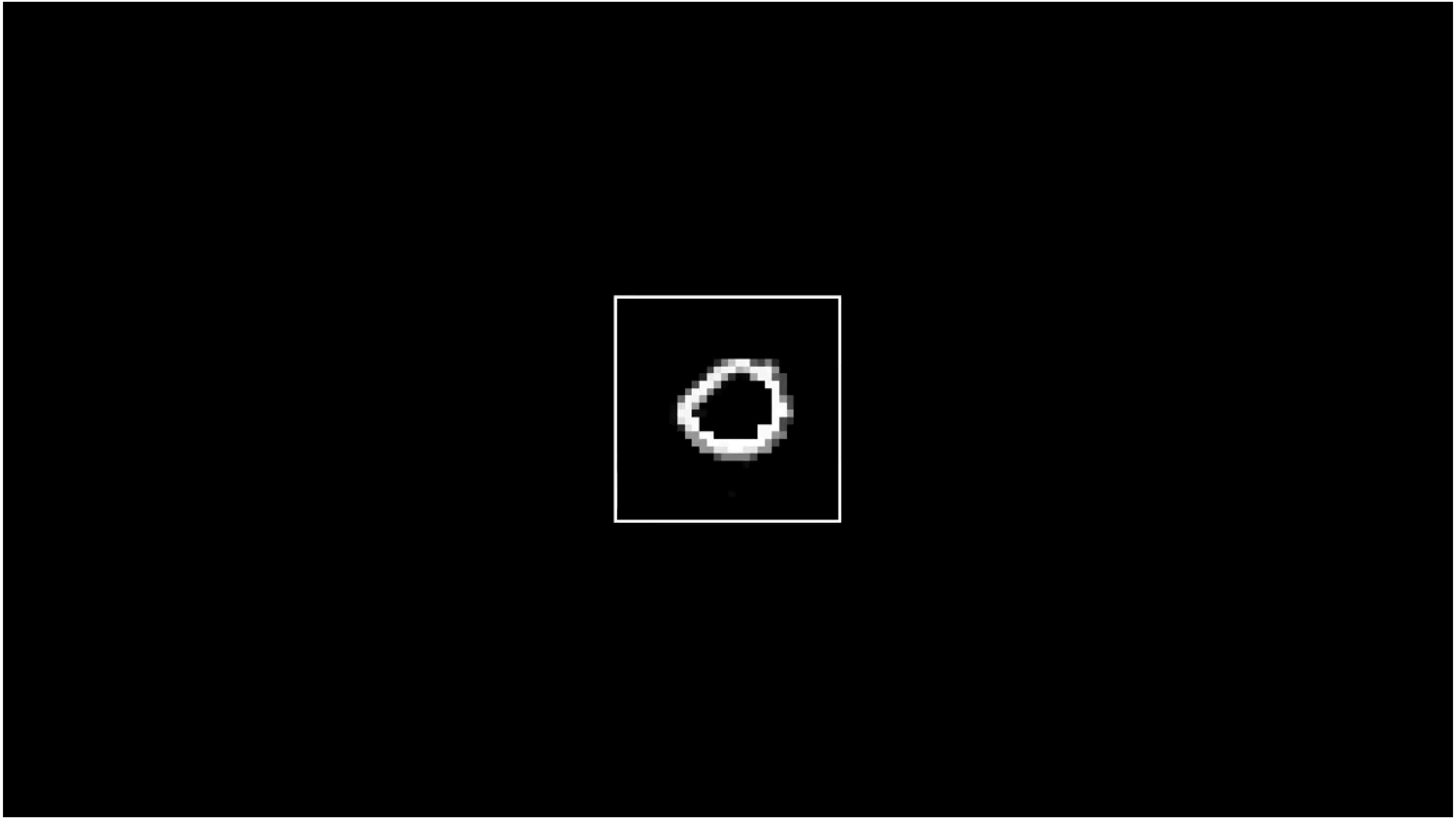


is a 9?

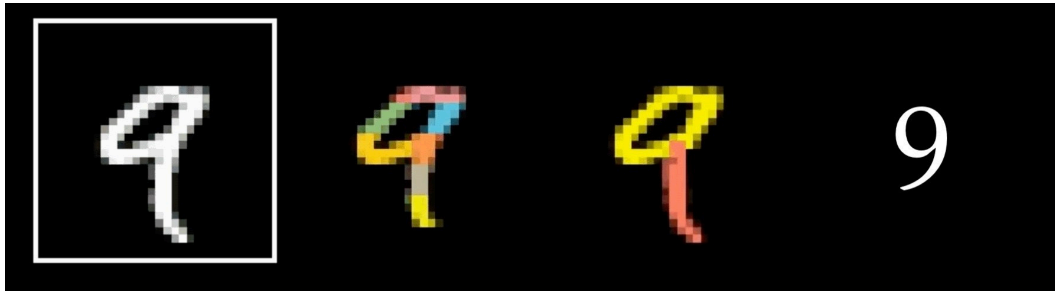
Why do we think any of

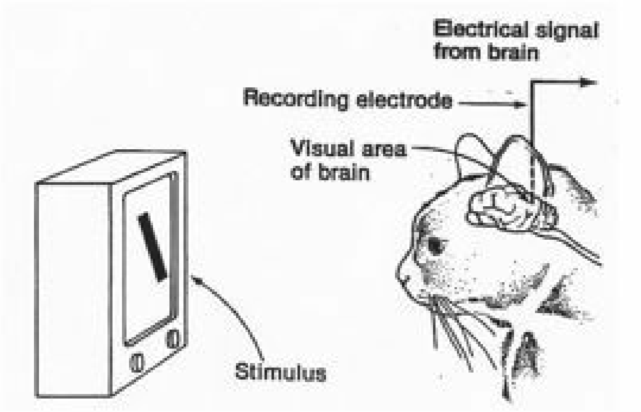


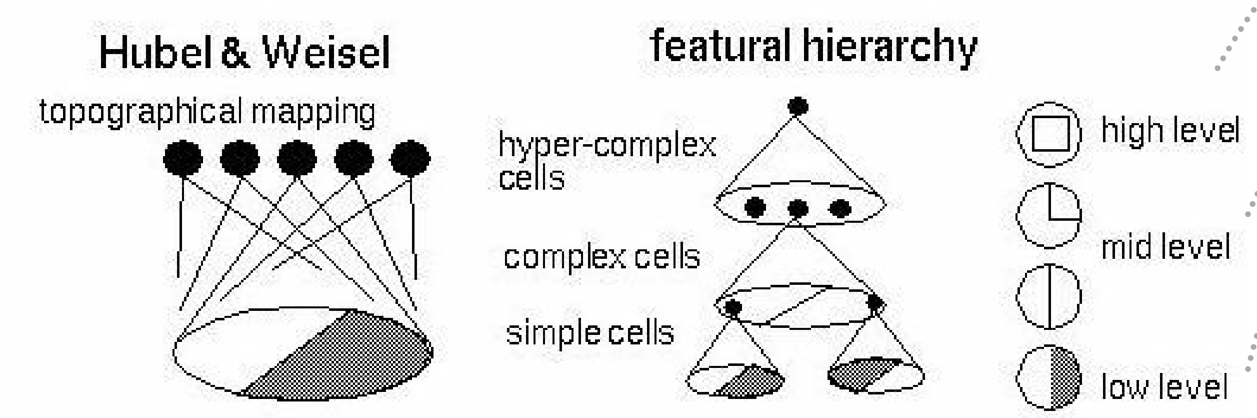
is a 9?

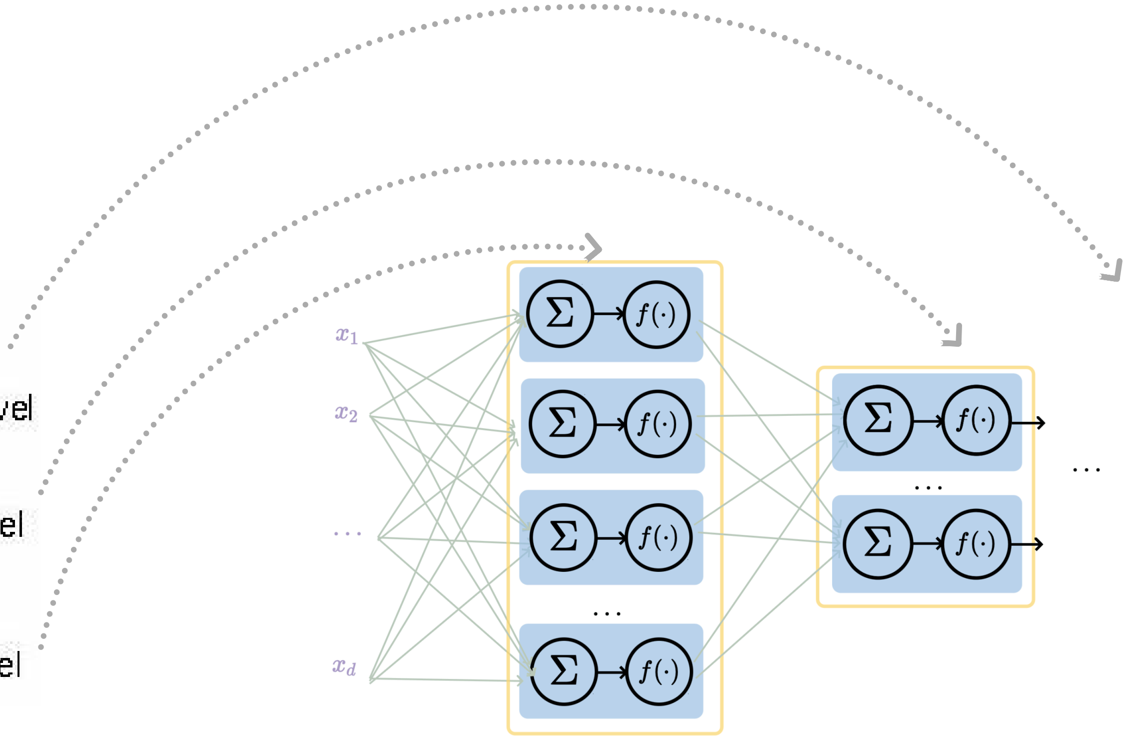


- Visual hierarchy



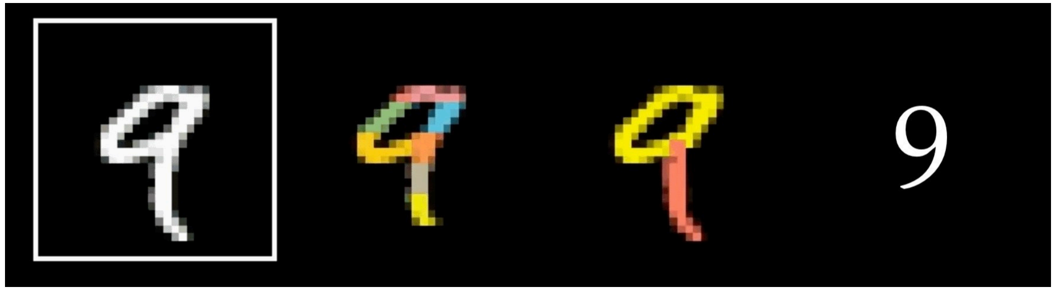




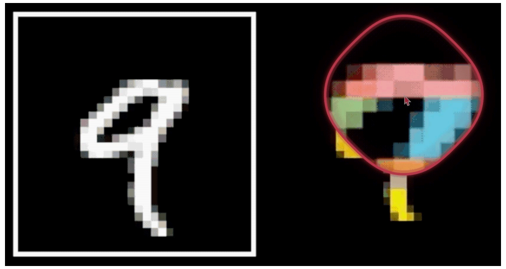


Layered structure are well-suited to model this hierarchical processing.

- Visual hierarchy



- Spatial locality



- Translational invariance

CNN exploits

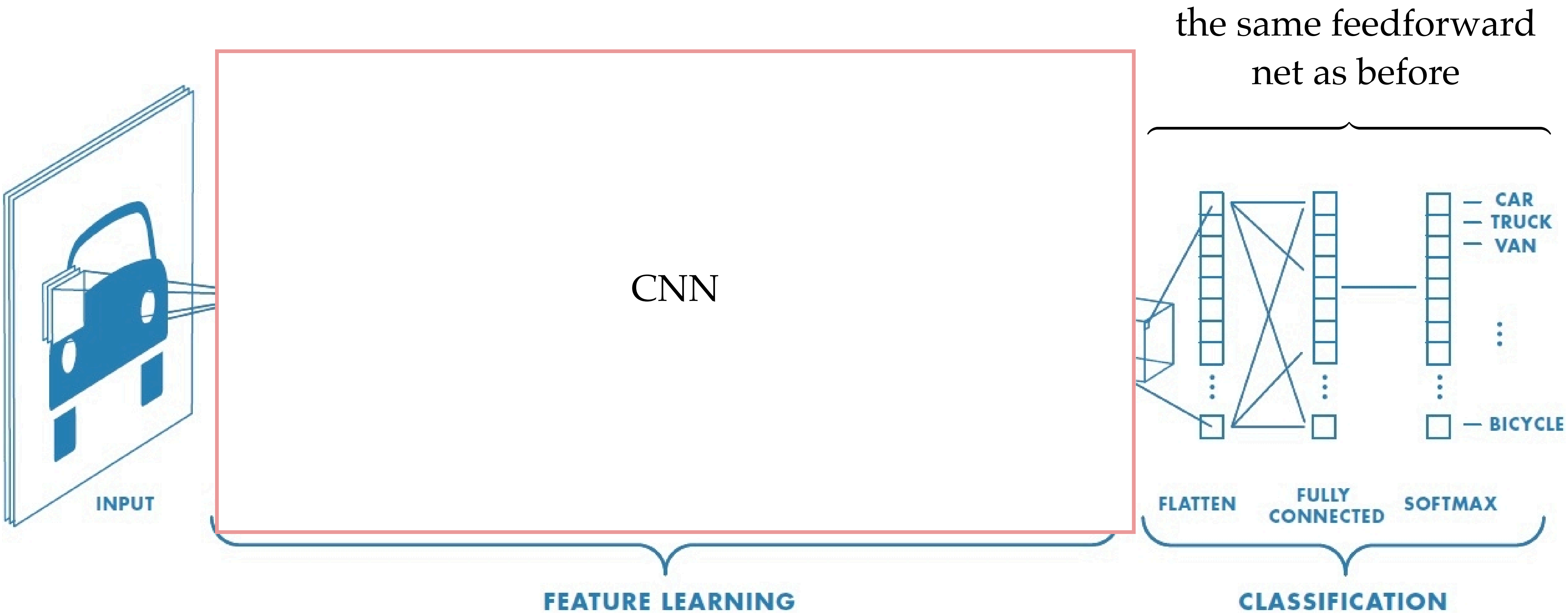
- Visual hierarchy
- Spatial locality
- Translational invariance

via

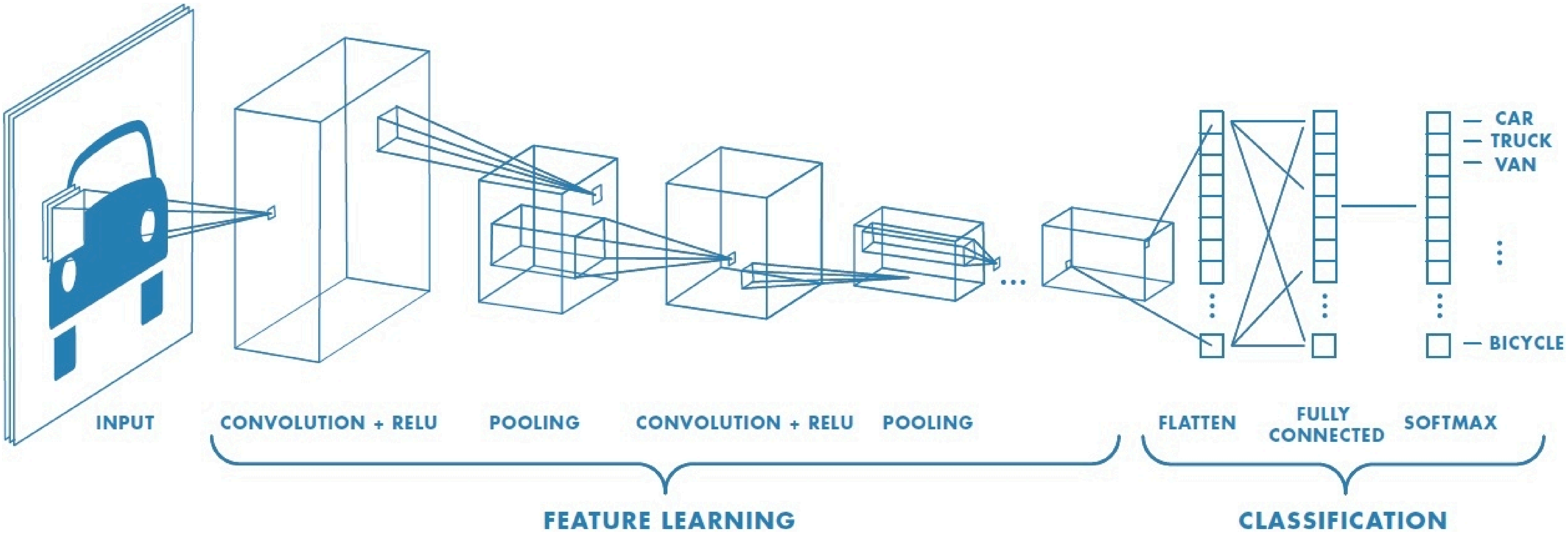
- layered structure
- convolution
- pooling

to handle images efficiently and effectively.

typical CNN architecture for image classification



typical CNN structure for image classification



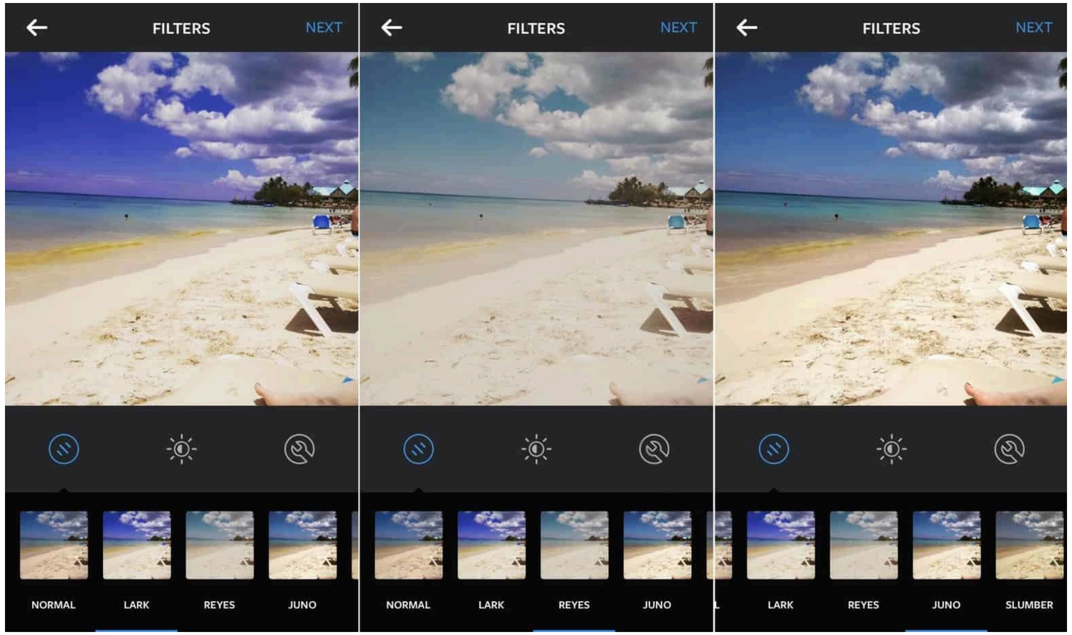
Outline

- 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

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

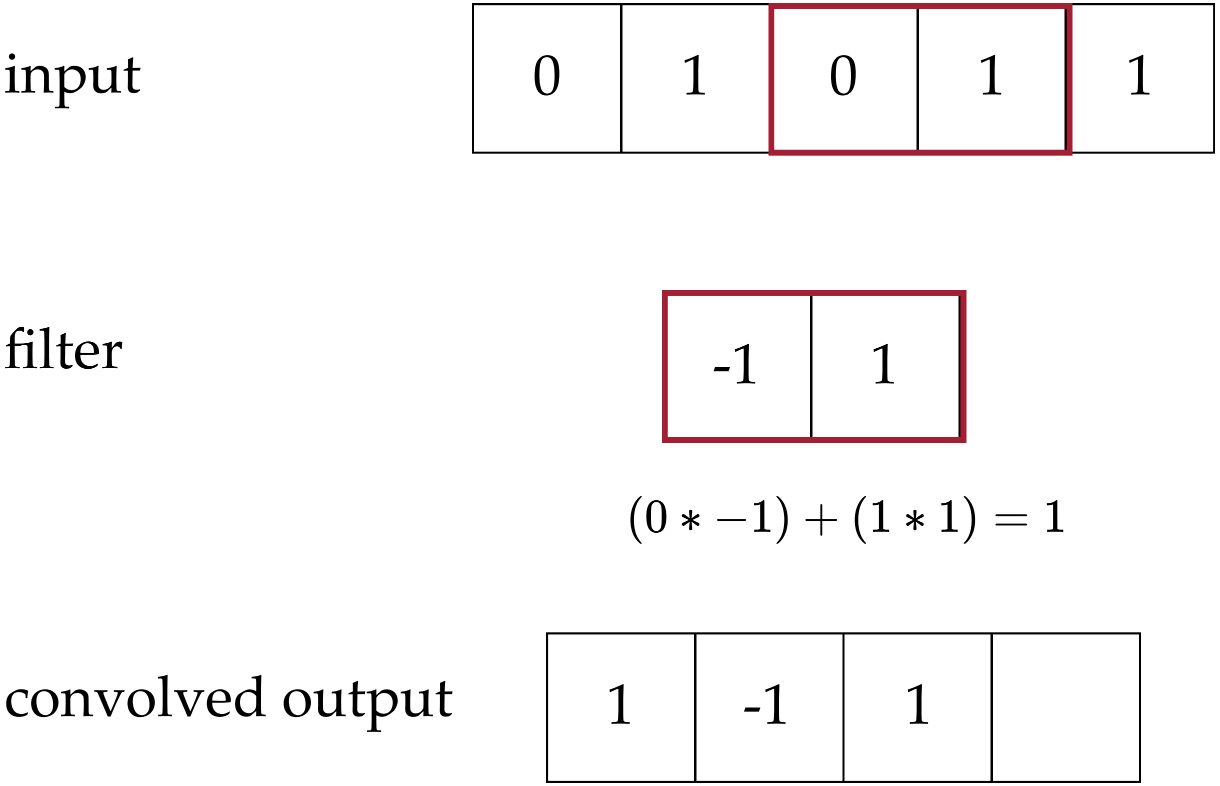
-1	1

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

convolved output

1	-1		

example: 1-dimensional convolution



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

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:

"look" *locally* through the filter

this local region = *receptive field*

input

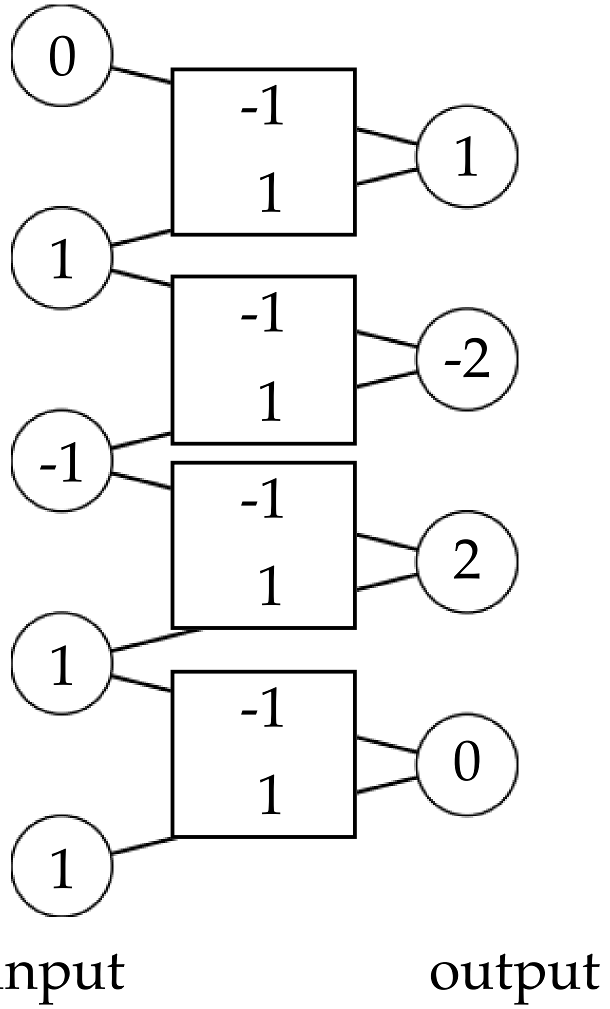
0	1	-1	1	1

filter

-1	1

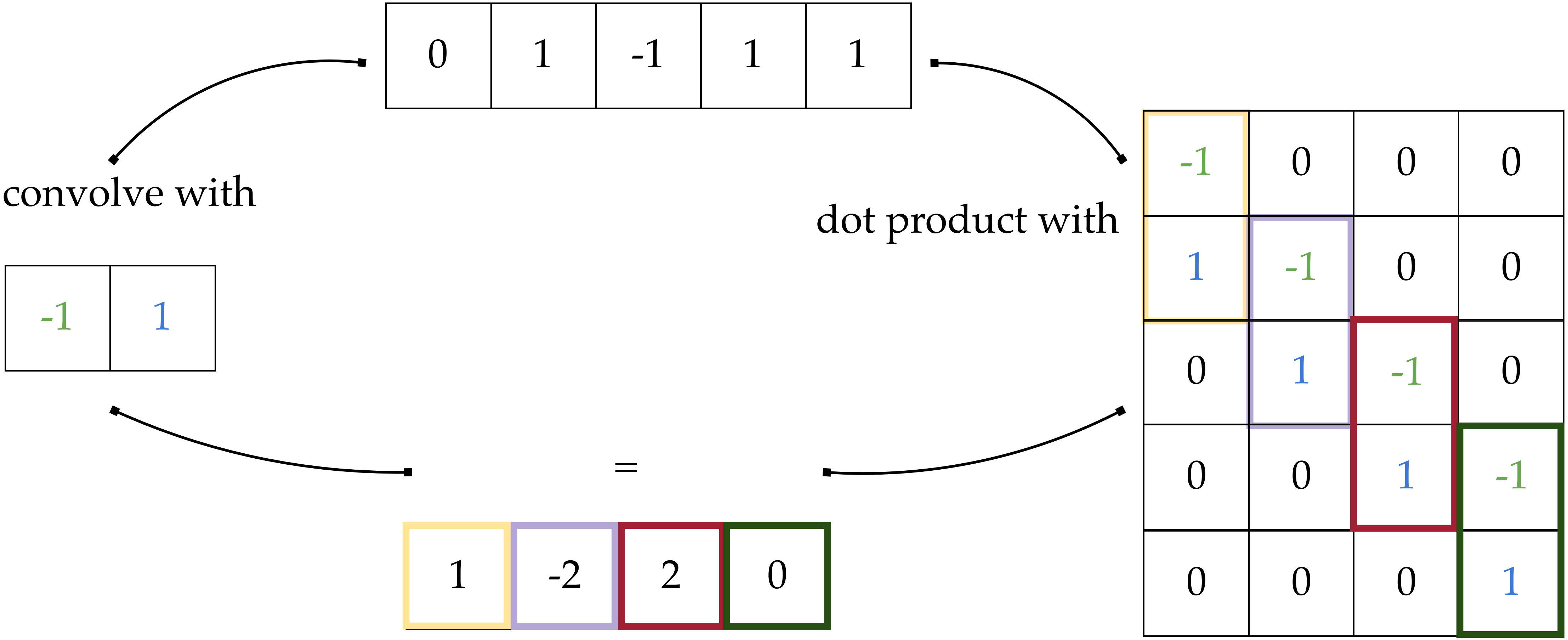
convolved output

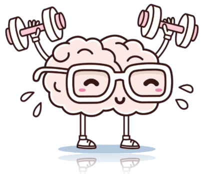
1	-2	2	0

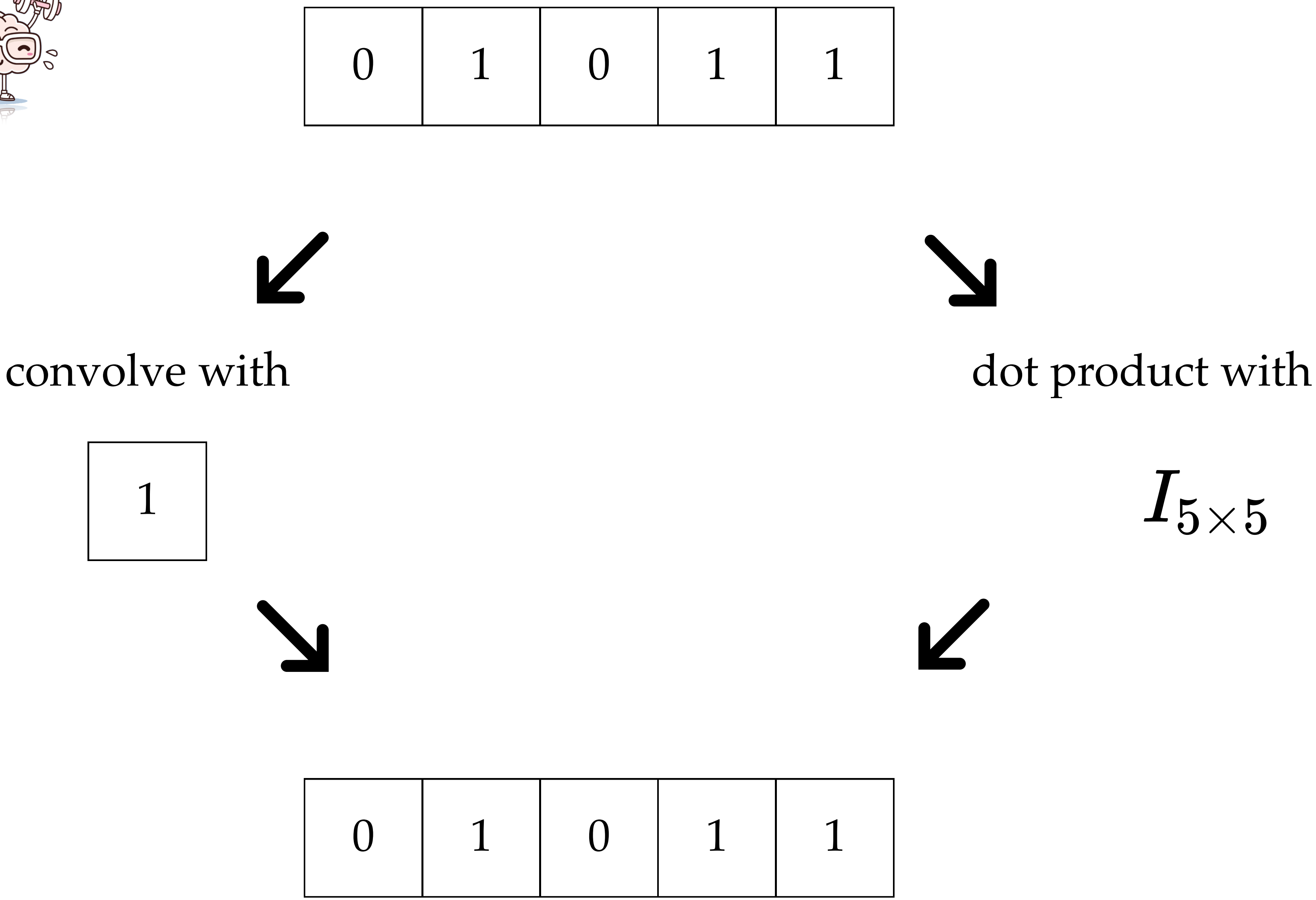


convolution interpretation 3:

sparse-connected layer
with parameter sharing

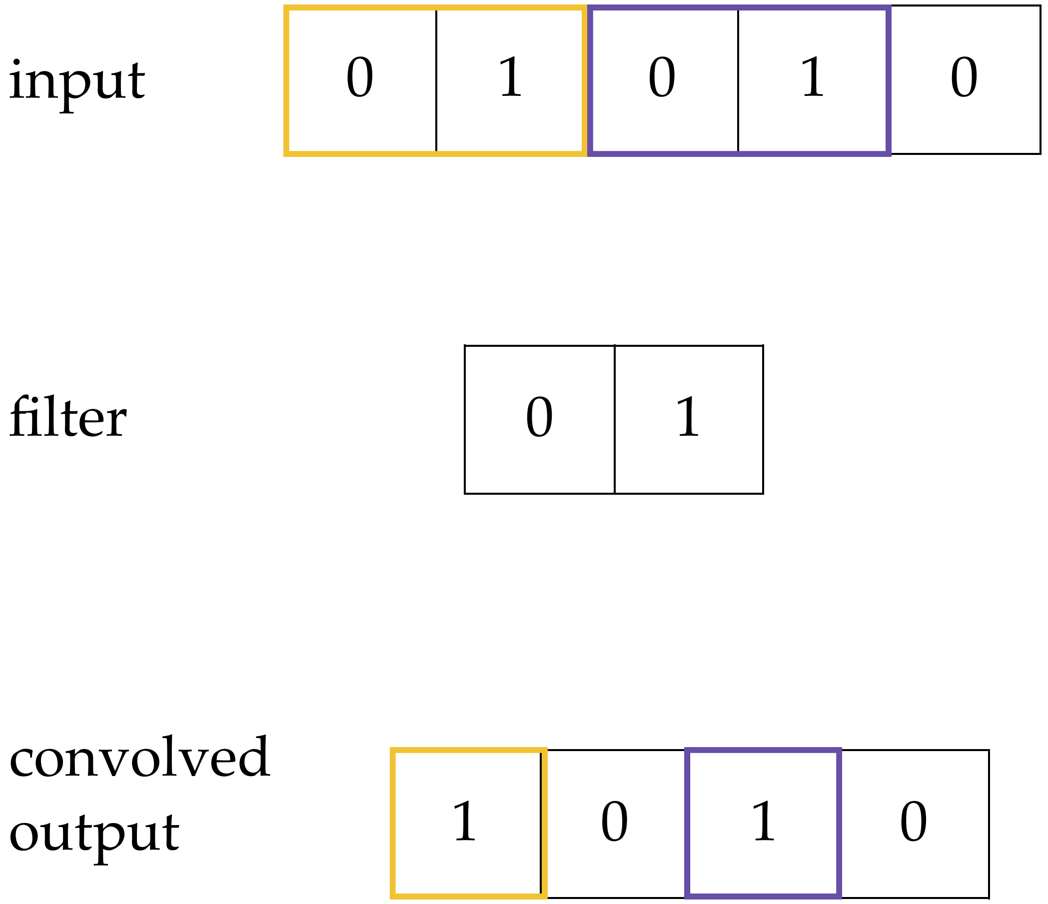






0 1 0 1 1
convolve with
1
dot product with
$I_{5\times5}$
0 1 0 1 1

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

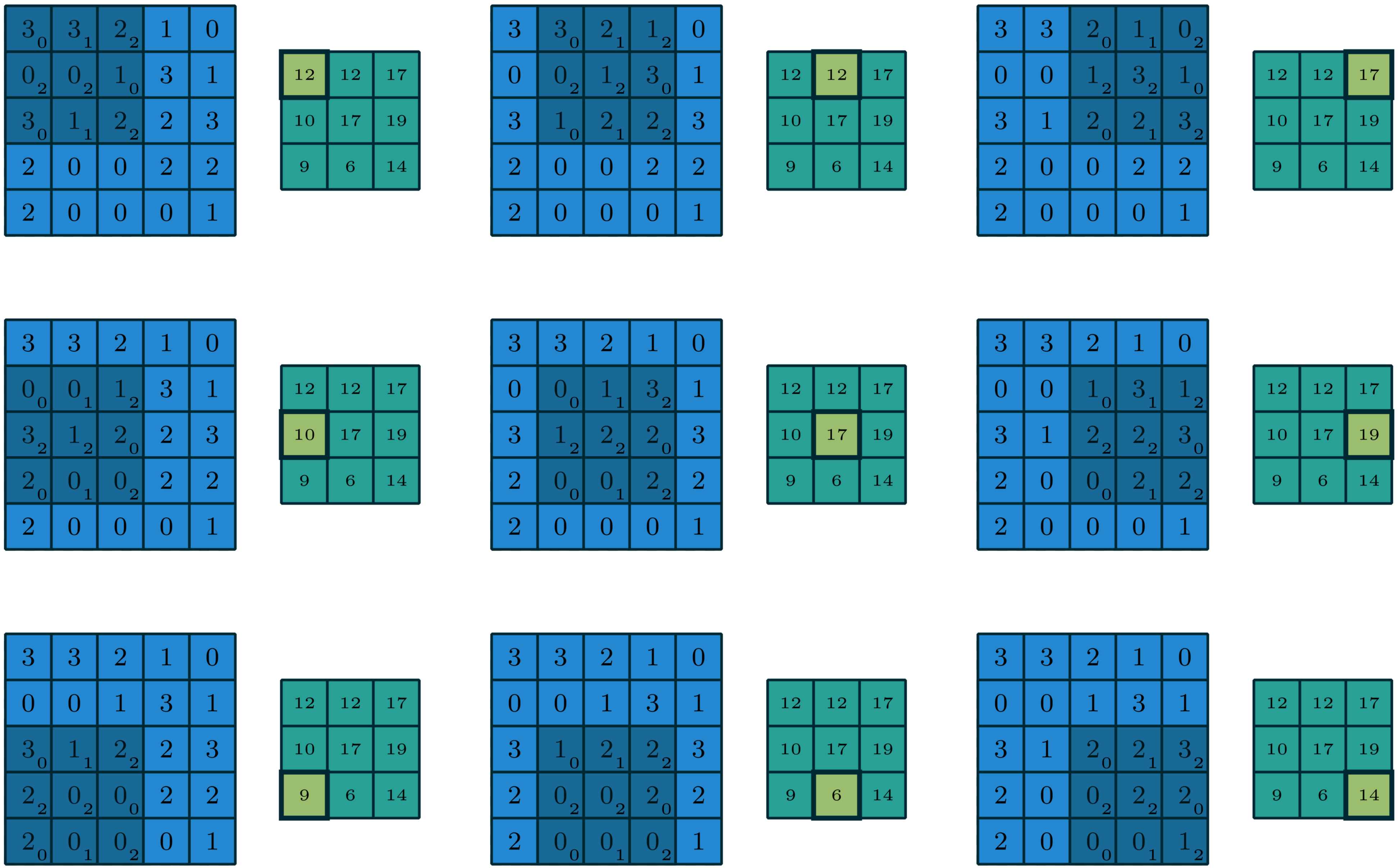
filter

0	1	2
2	2	0
0	1	2

convolved output

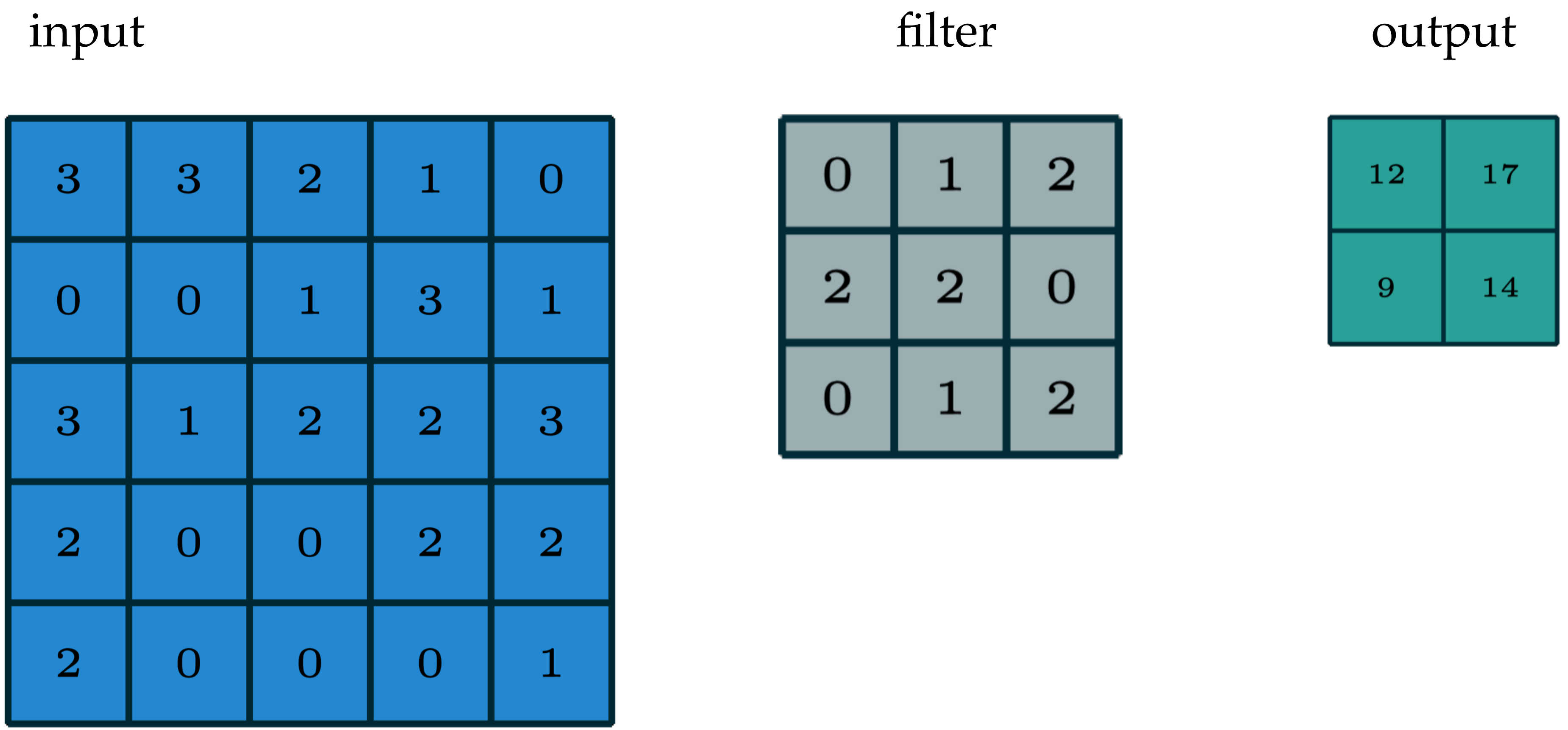
12	12	17
10	17	19
9	6	14

[image edited from vdumoulin]

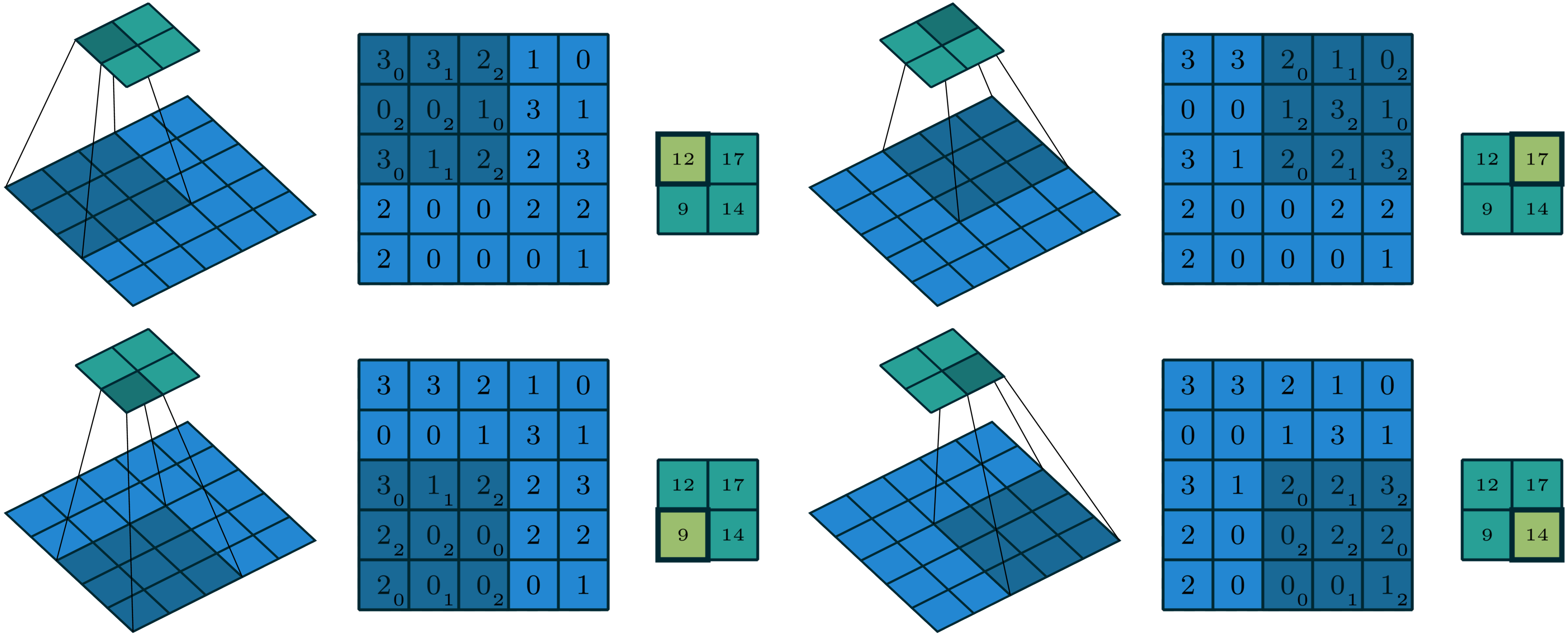


[image edited from vdumoulin]

stride of 2

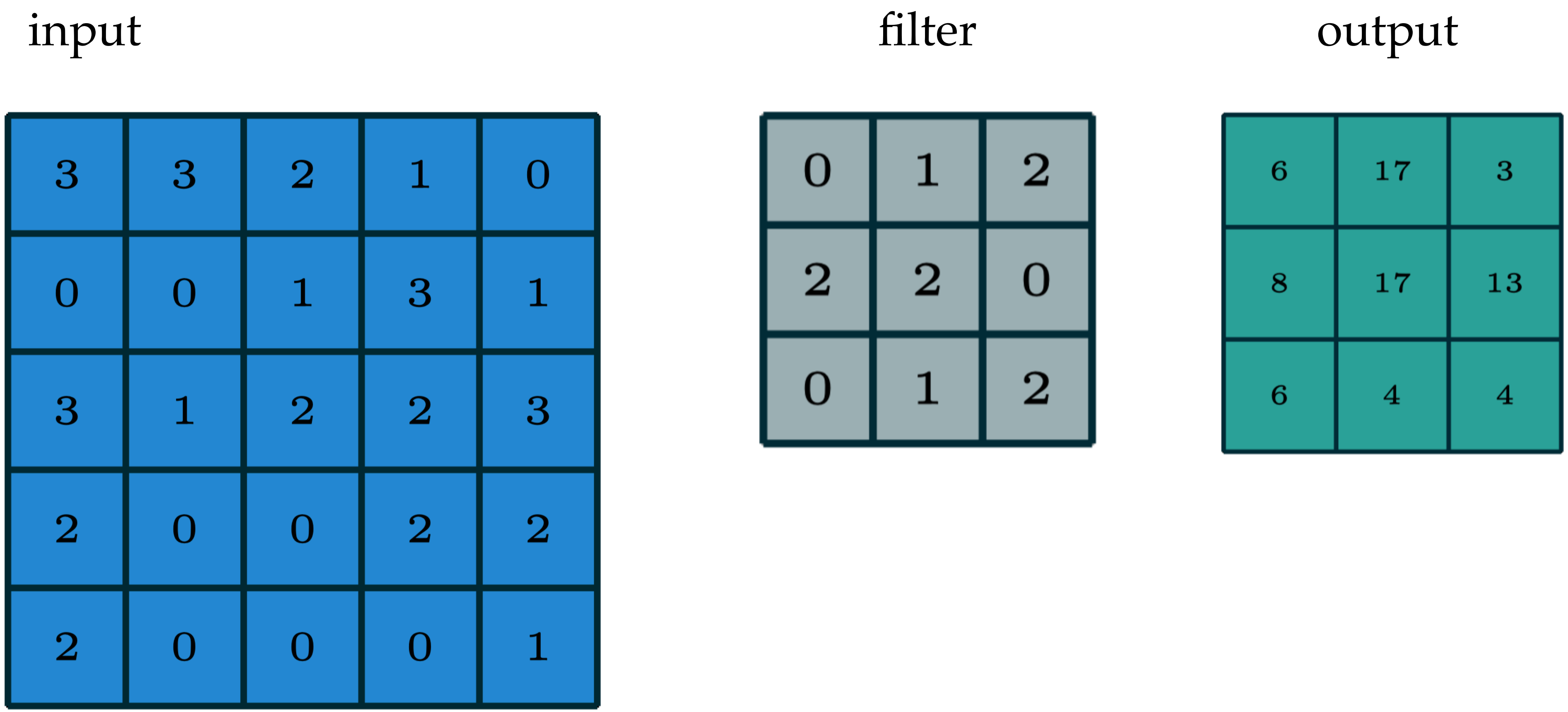


[image edited from vdumoulin]

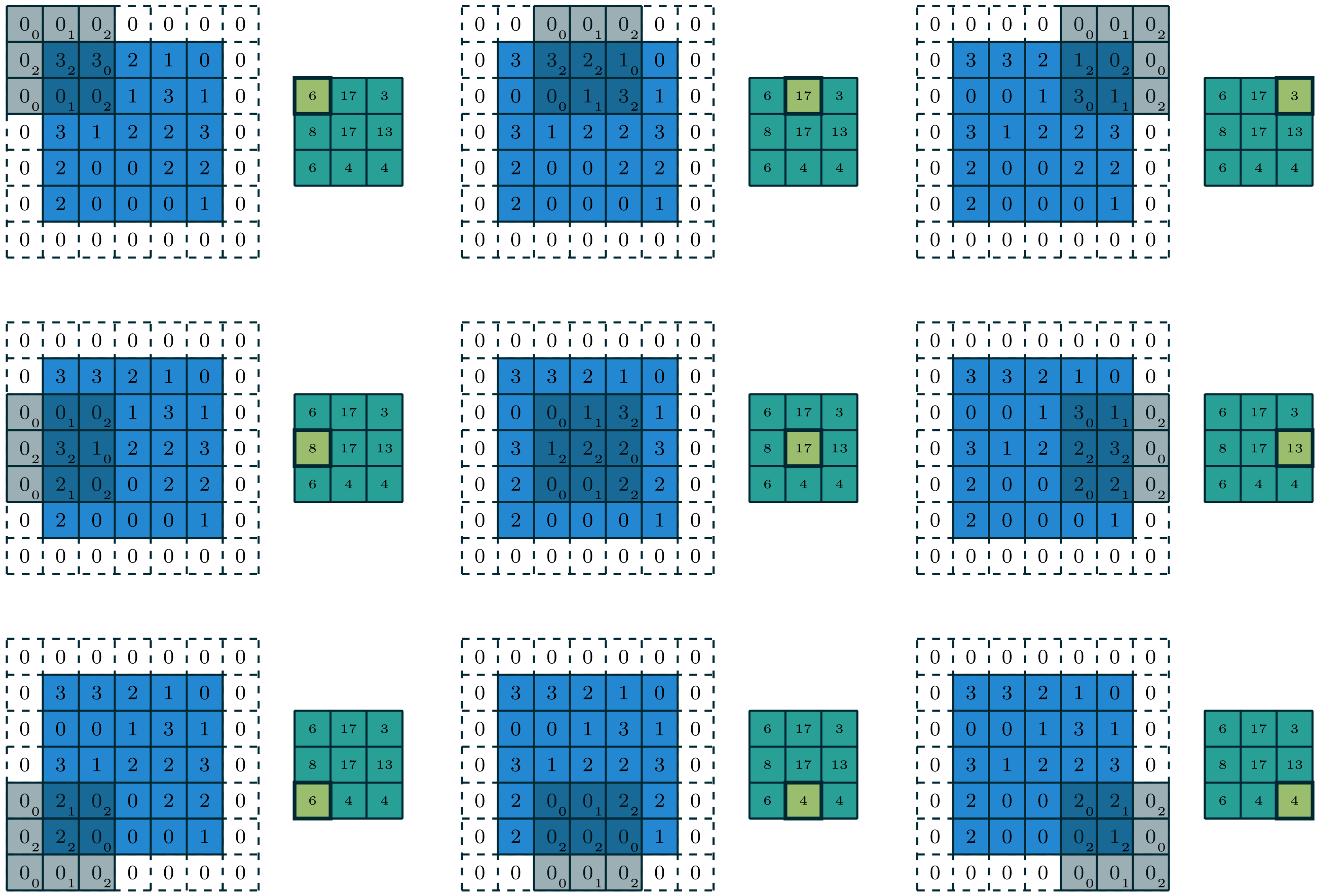


[image edited from vdumoulin]

stride of 2, with padding of size 1



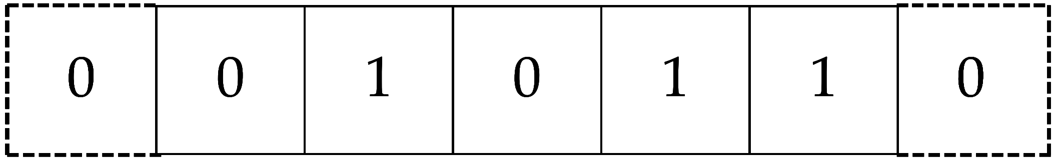
[image edited from vdumoulin]



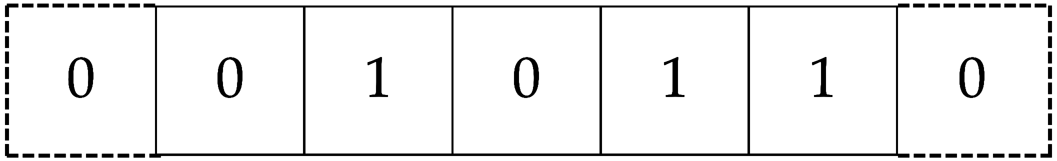
[image edited from vdumoulin]

quick summary: hyperparameters for 1d convolution

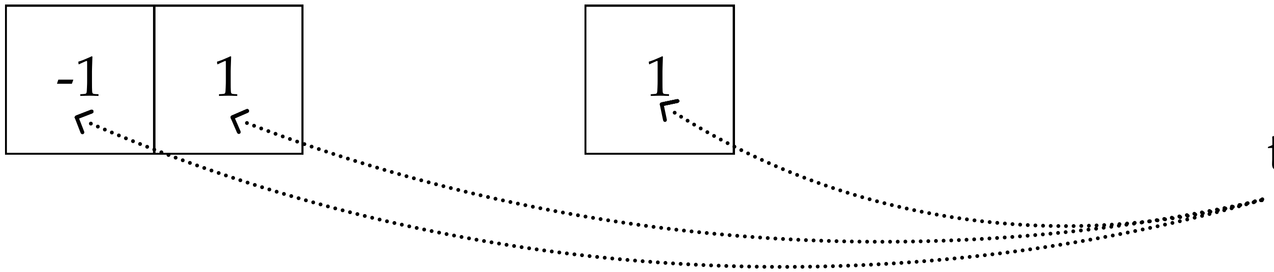
- Zero-padding



- Stride (e.g. stride of 2)

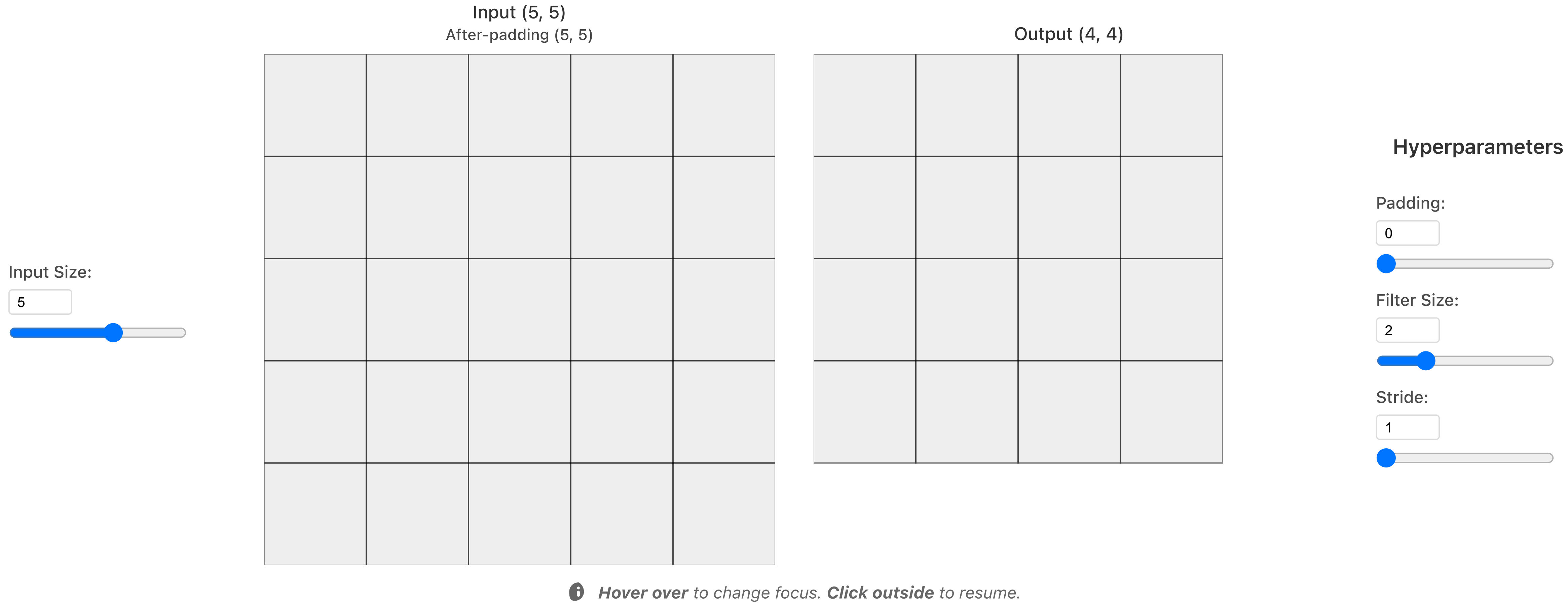


- Filter size (e.g. we saw these two in 1-d)



these weights are what CNN learn eventually

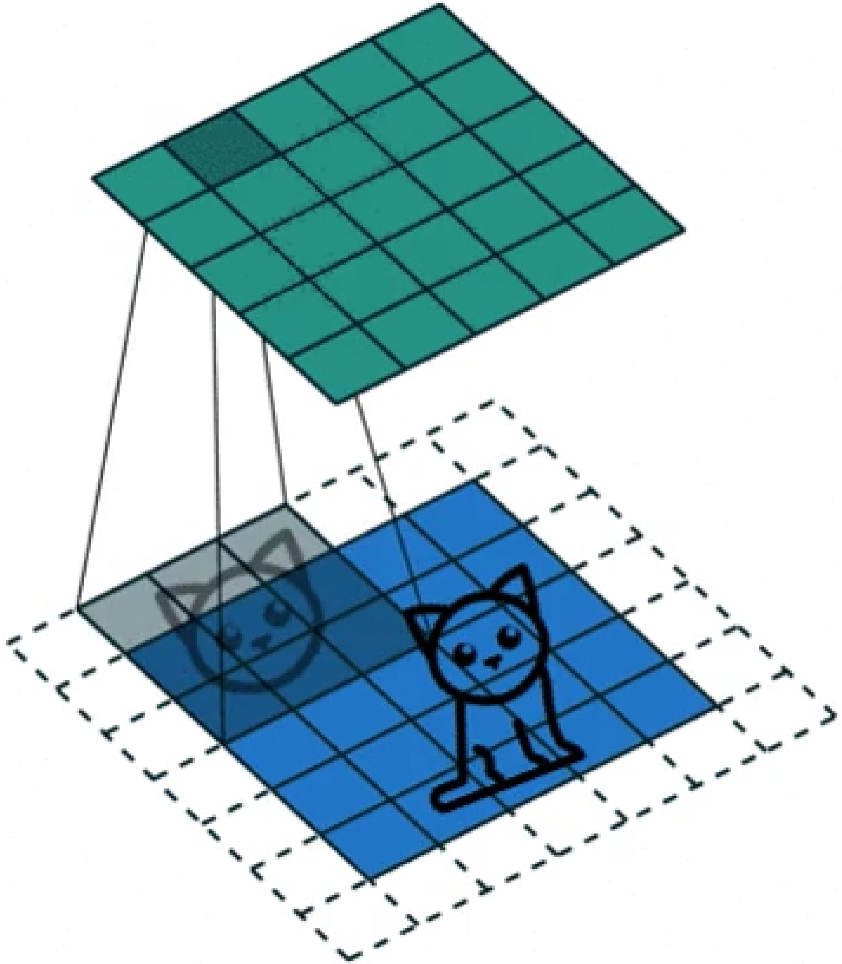
quick summary: hyperparameters for 2d convolution



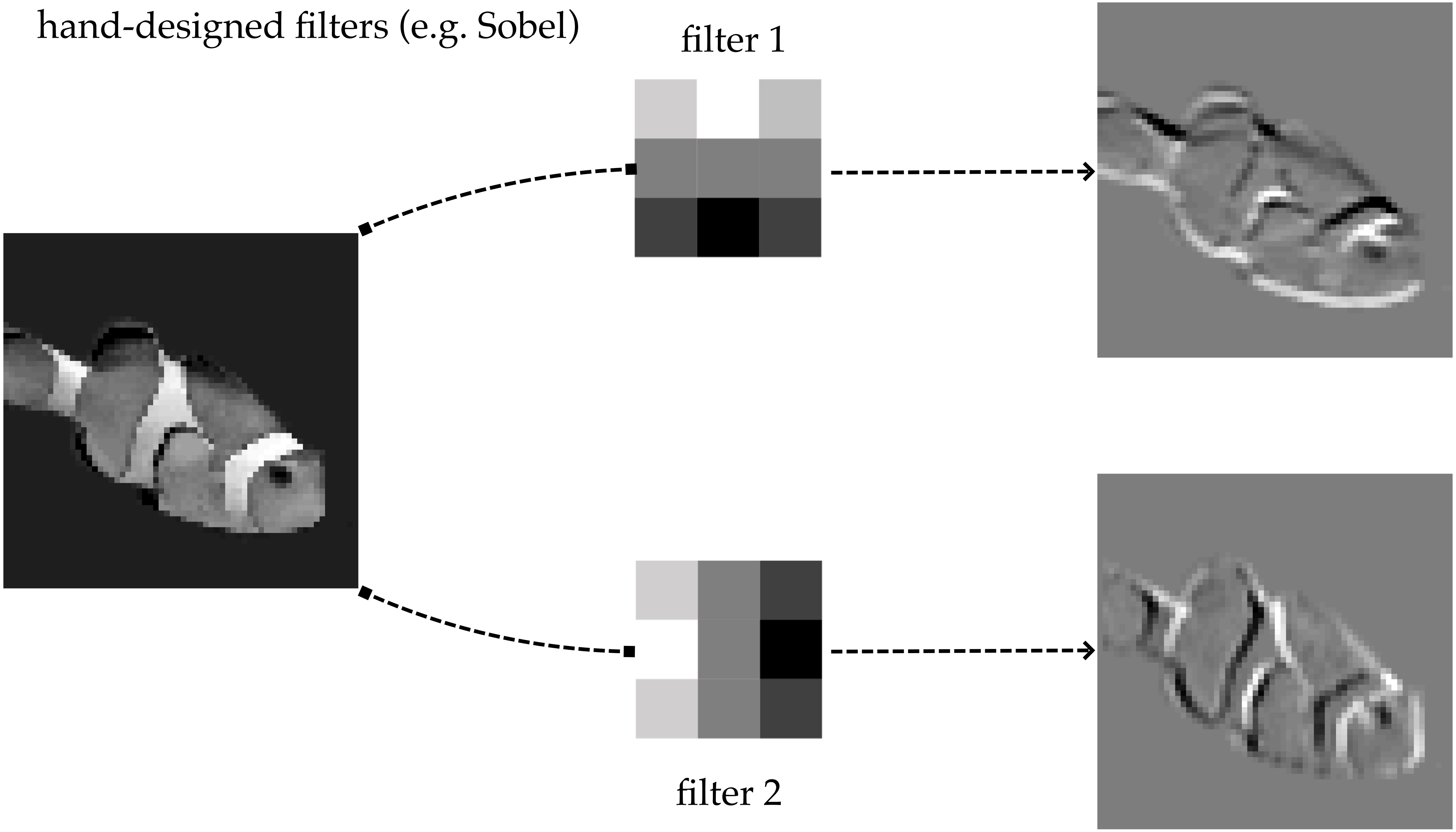
Adapted from CNN Explainer with ❤️ by Shen² | Bug Report

quick summary: convolution interpretation

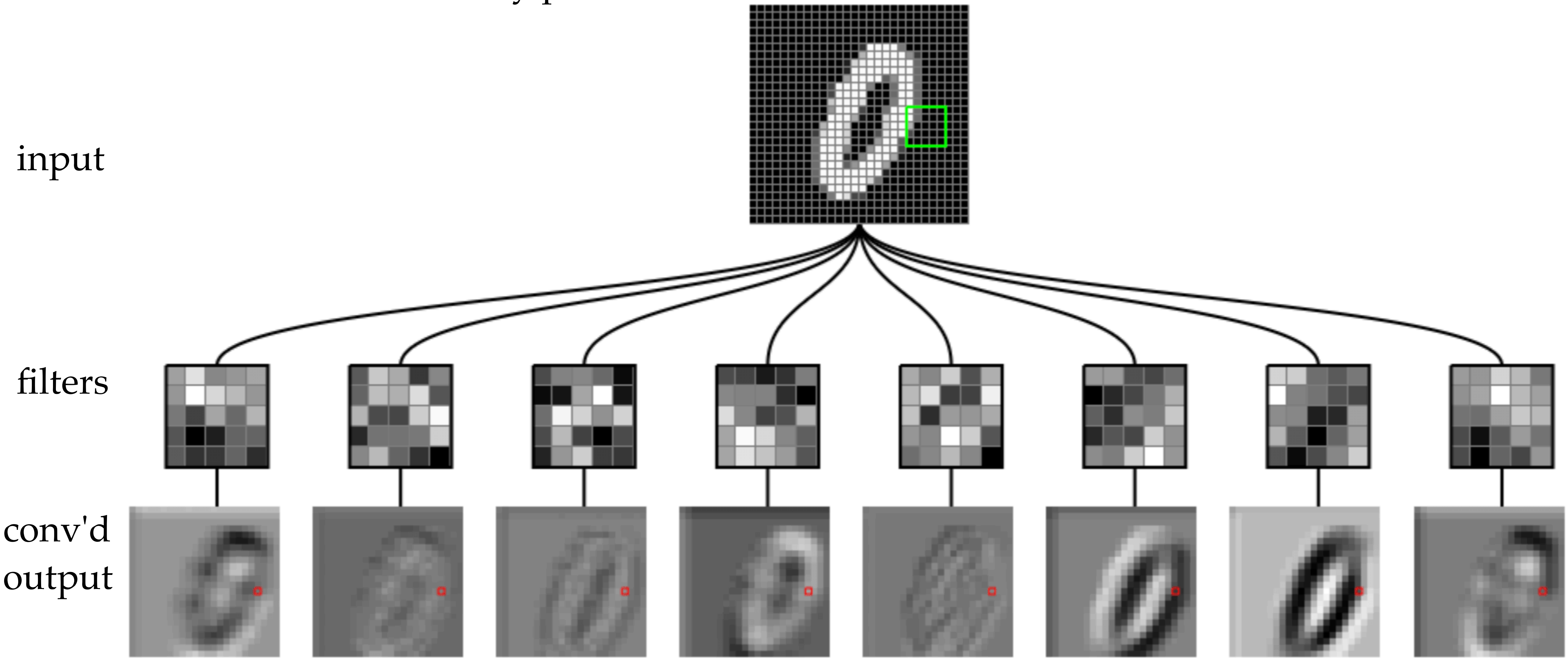
- Look locally (sparse connections)
- Parameter sharing
- Template matching
- Translational equivariance



[video credit Lena Voita]

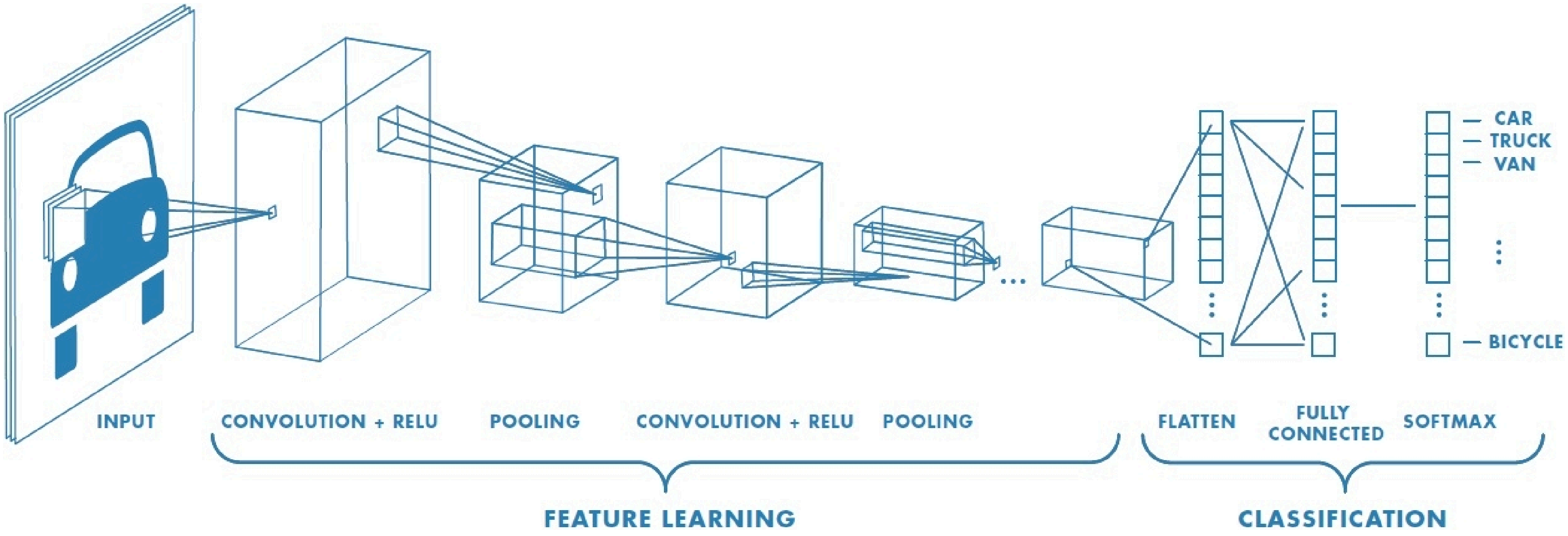

hand-designed filters (e.g. Sobel)
filter 1
filter 2

learned filters detect many patterns

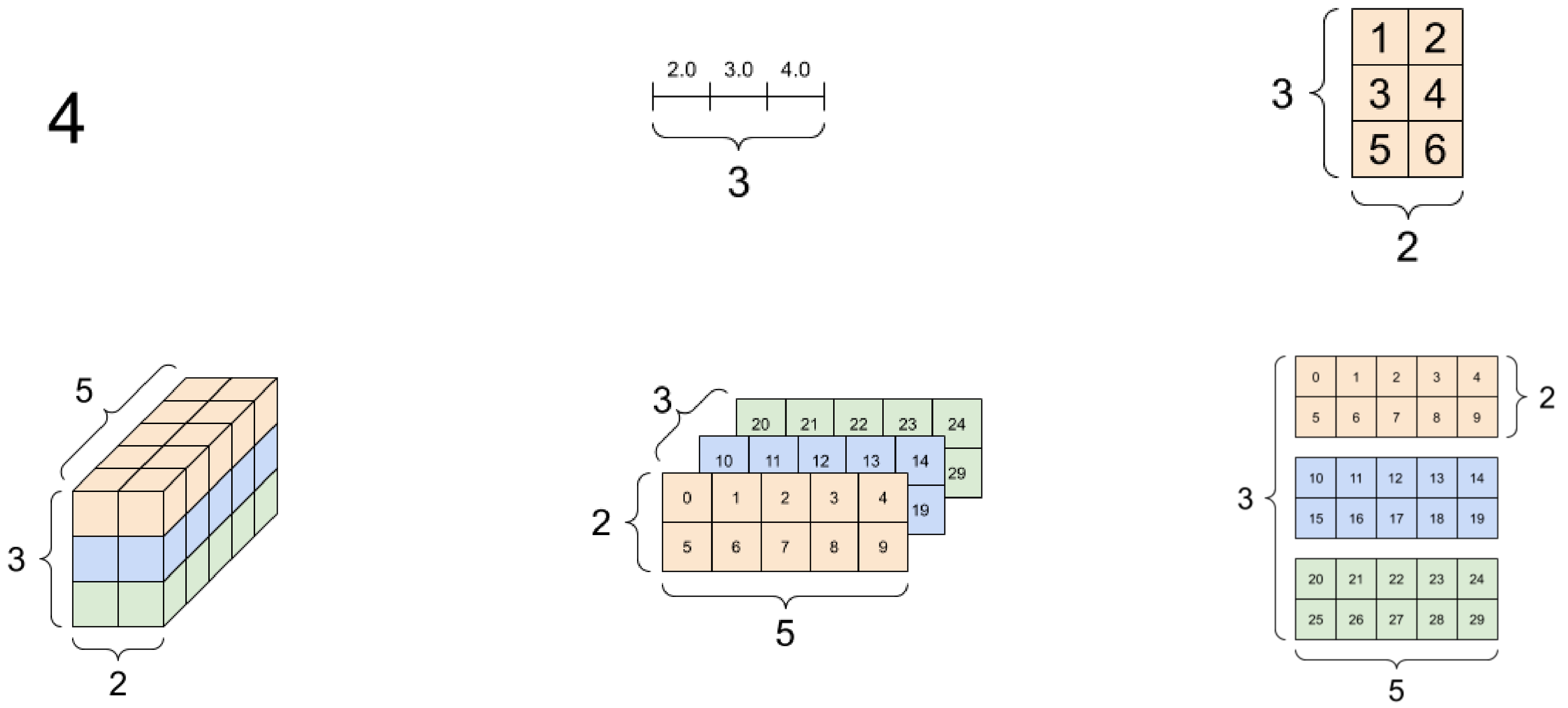


Outline

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

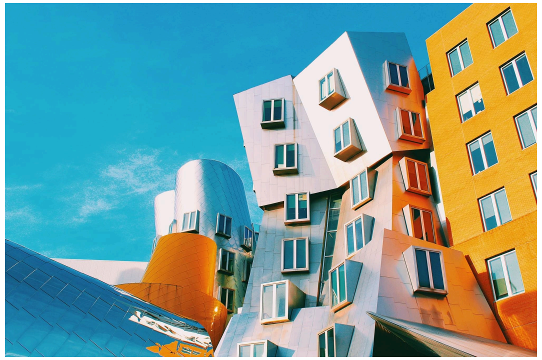


A tender intro to tensor:



[image credit: tensorflow]

color images and channels



red

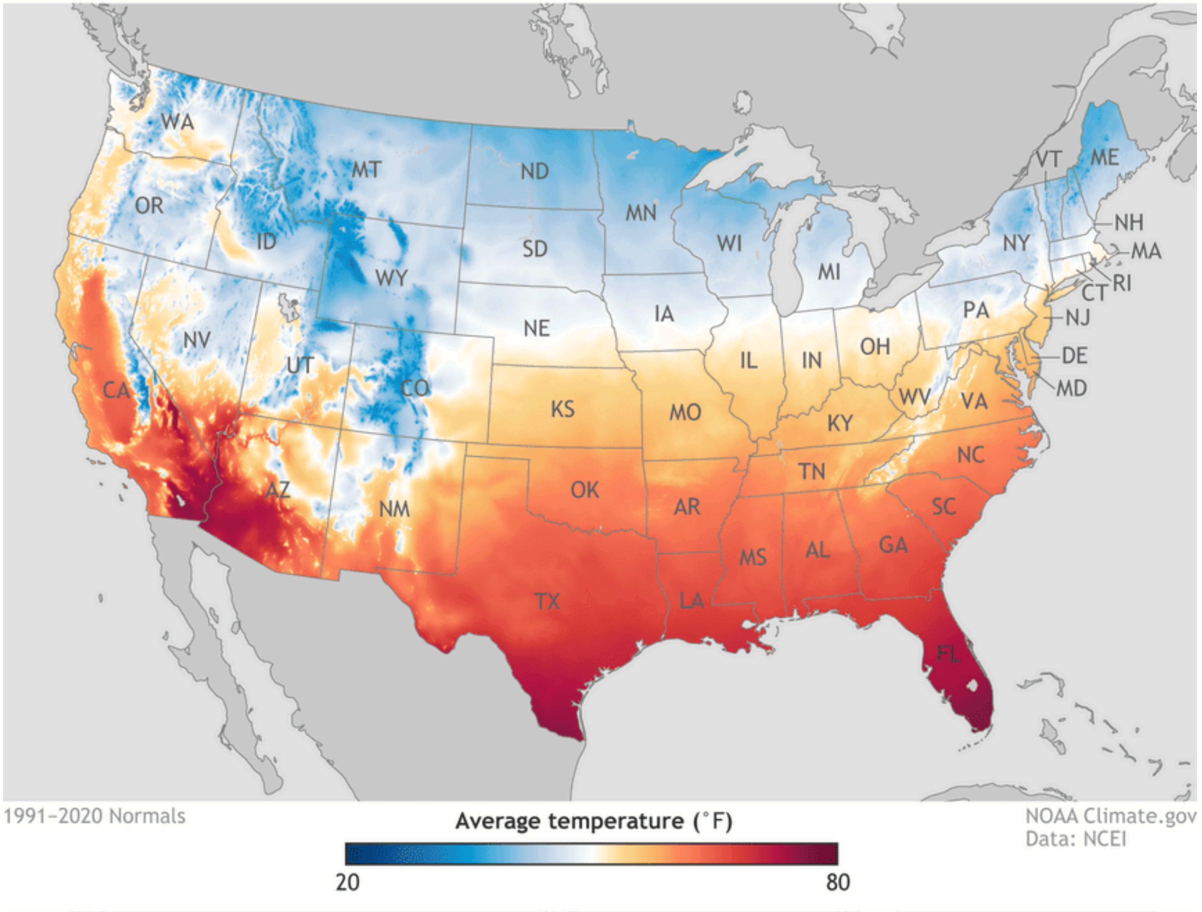
green

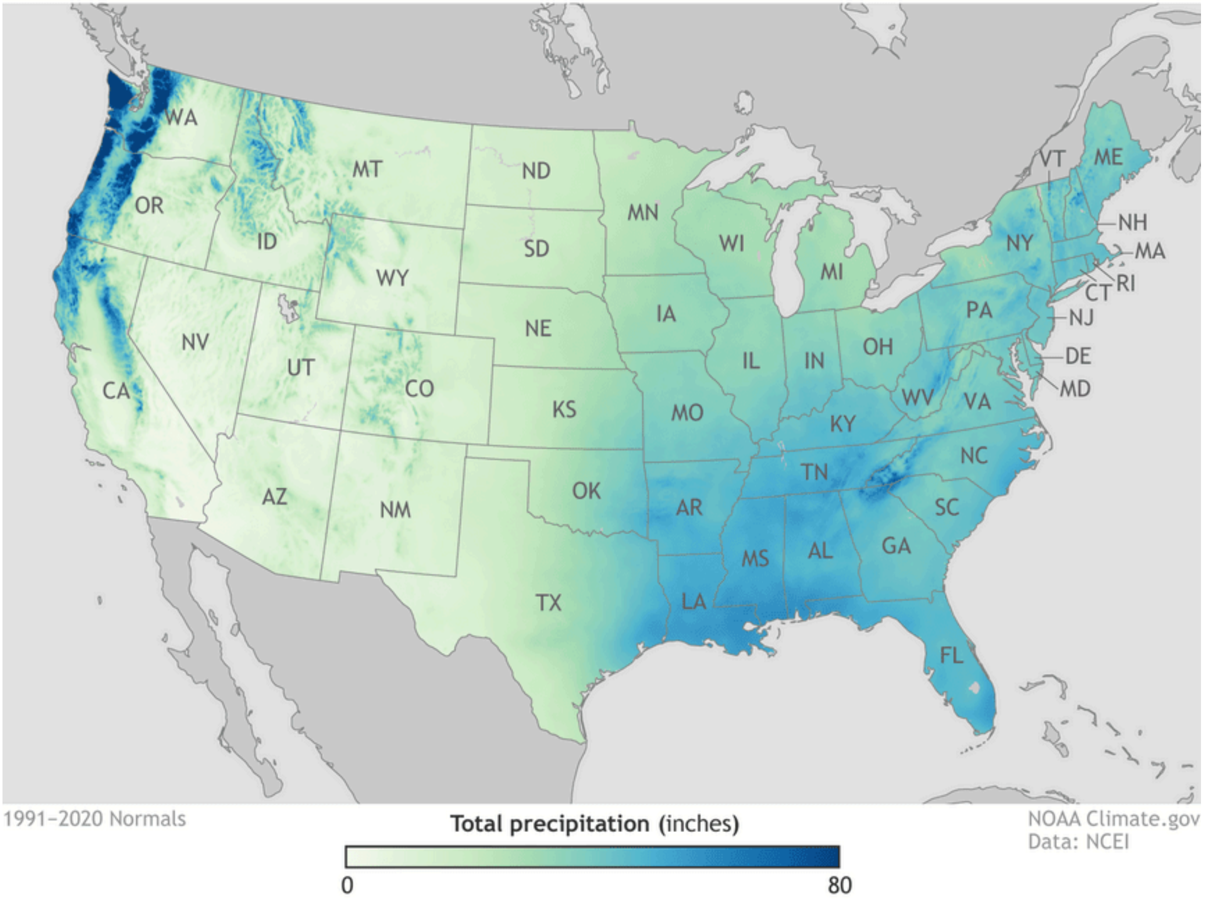
blue

[Photo by Zayn Shah, Unsplash]

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

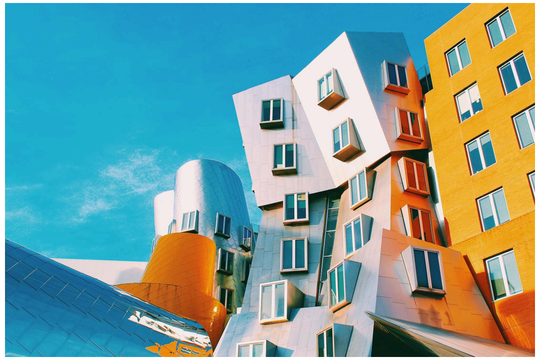
like when we think of weather:





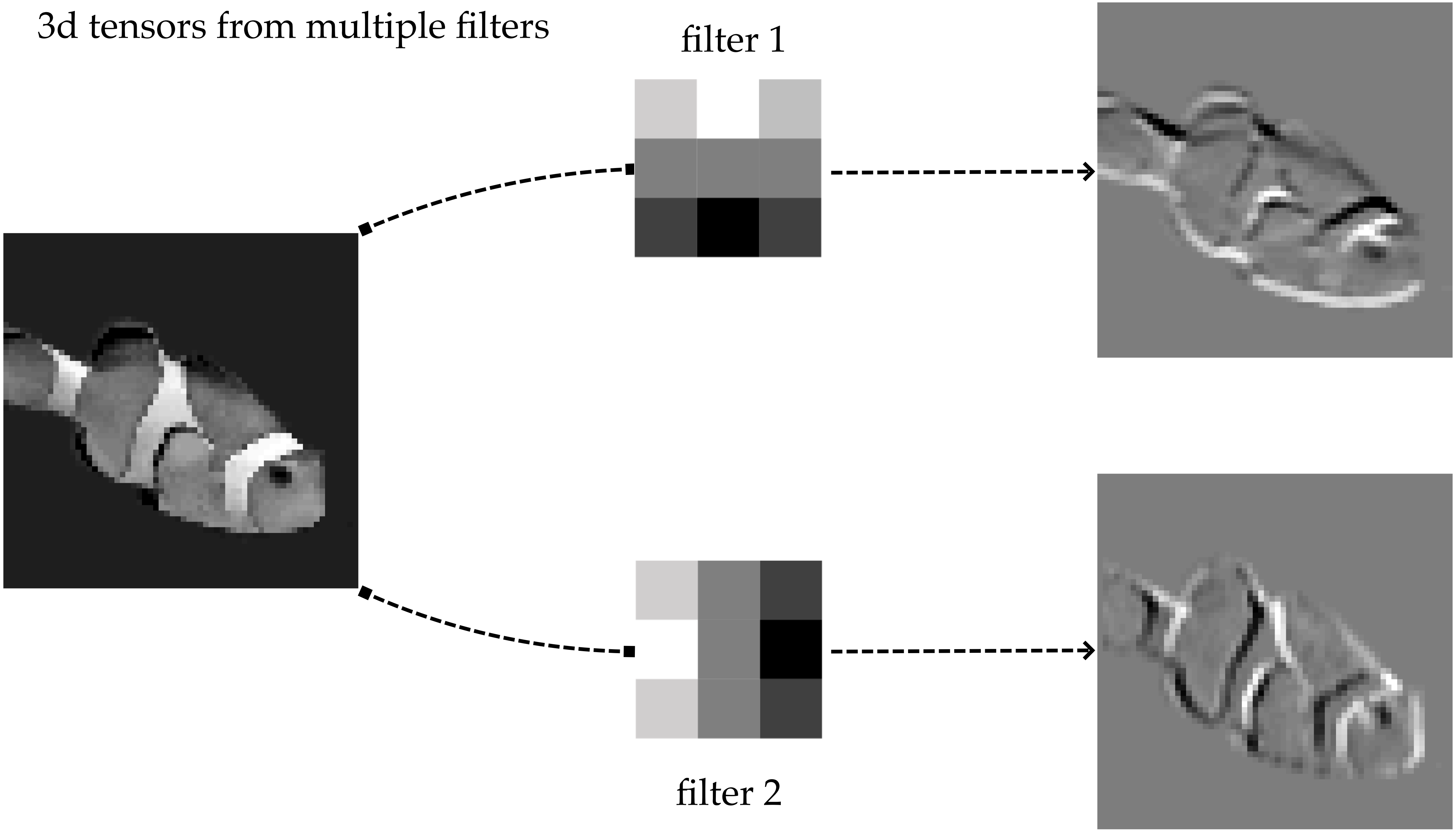
so channels are often referred to as *feature maps*

3d tensors from color channels

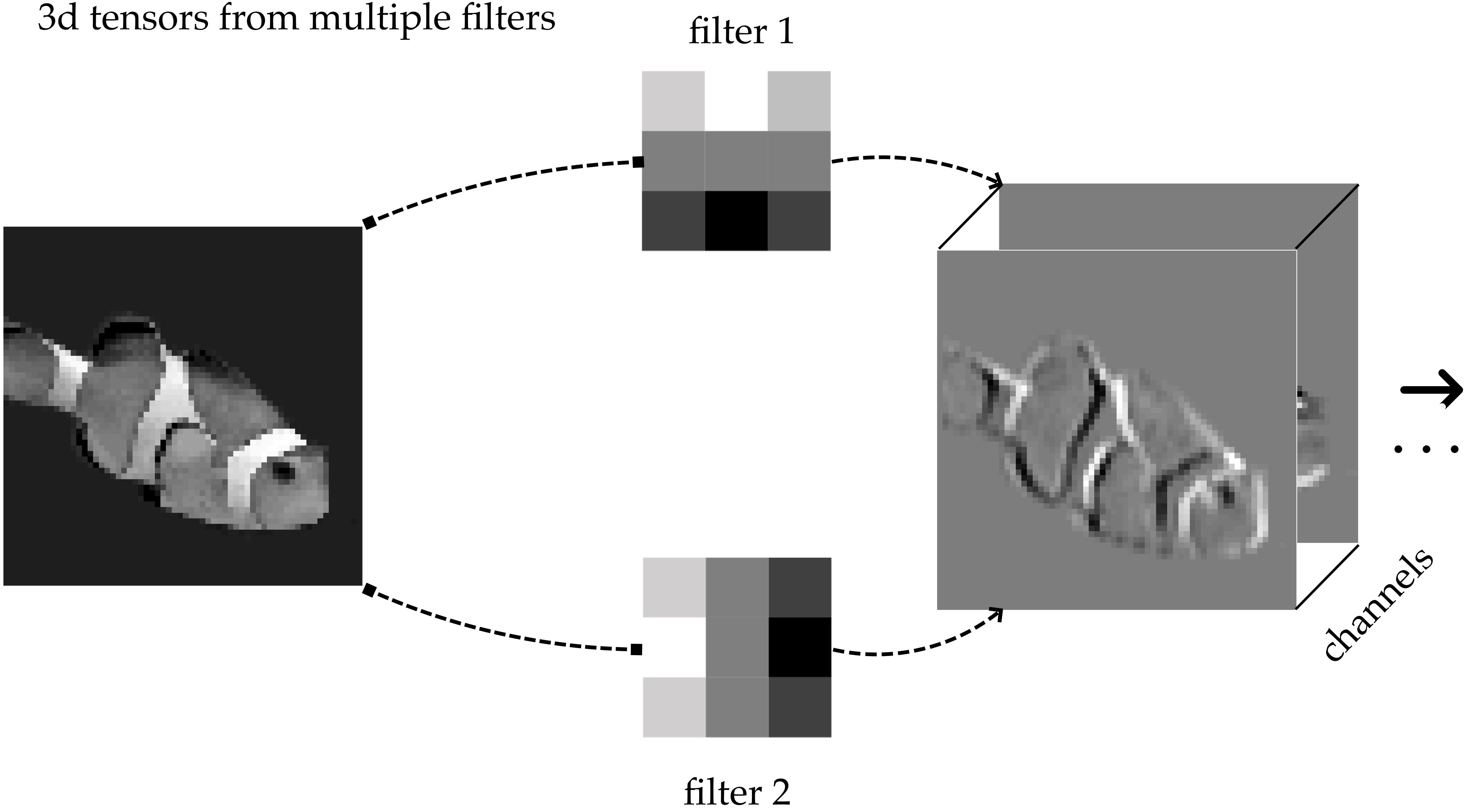


[Photo by Zayn Shah, Unsplash]

3d tensors from multiple filters



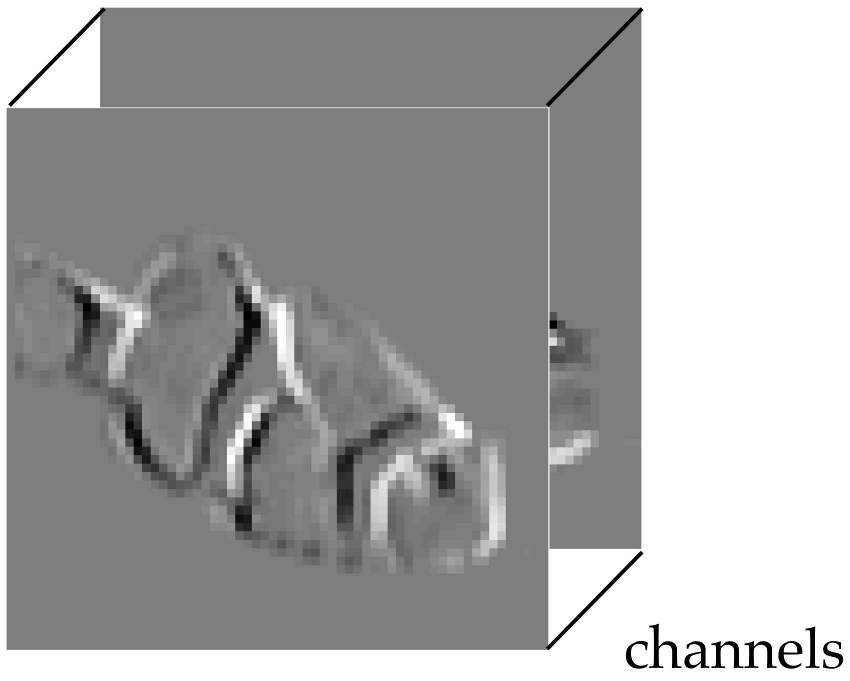
3d tensors from multiple filters



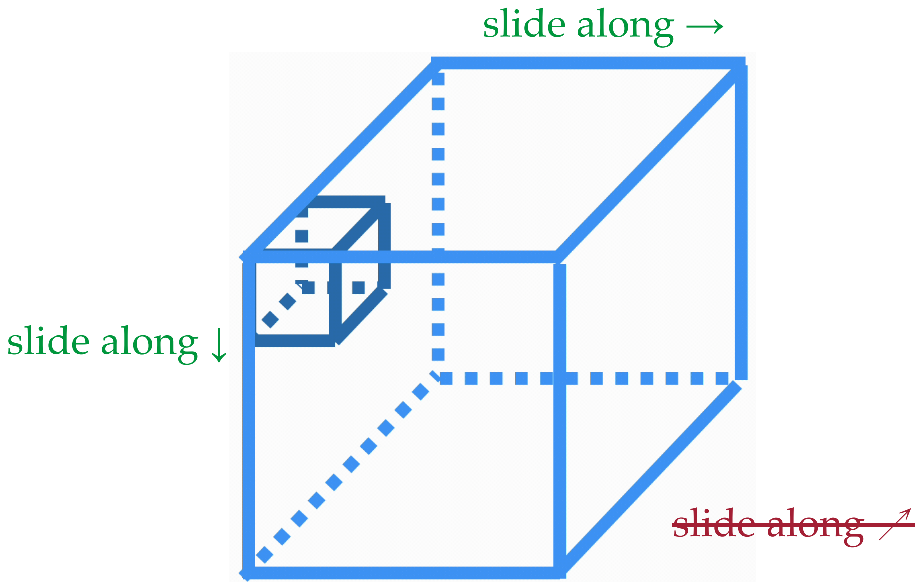
where do channels come from?

1. color input

2. multiple filters → multiple channels



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

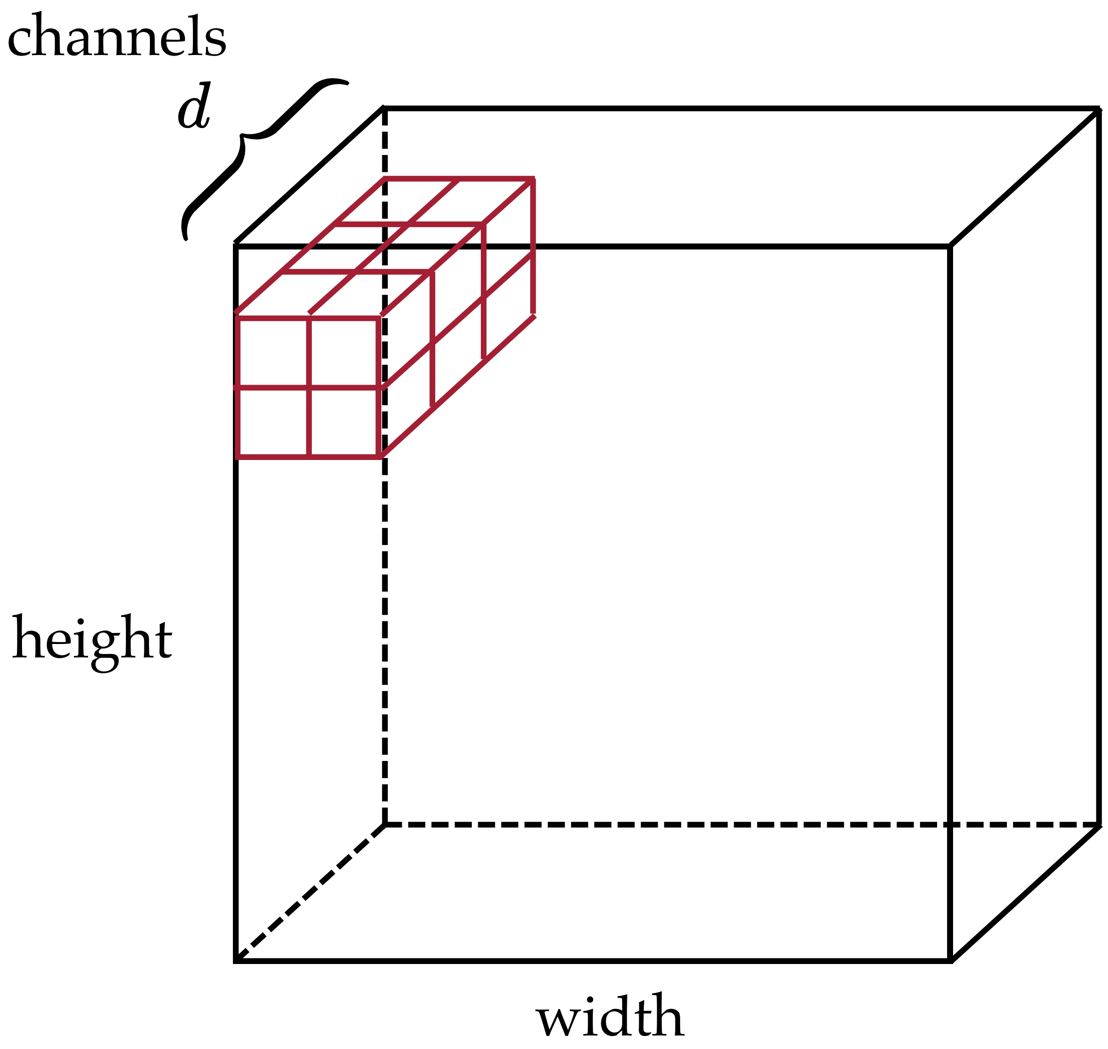


Convolution **shares** weights across **shifted** positions

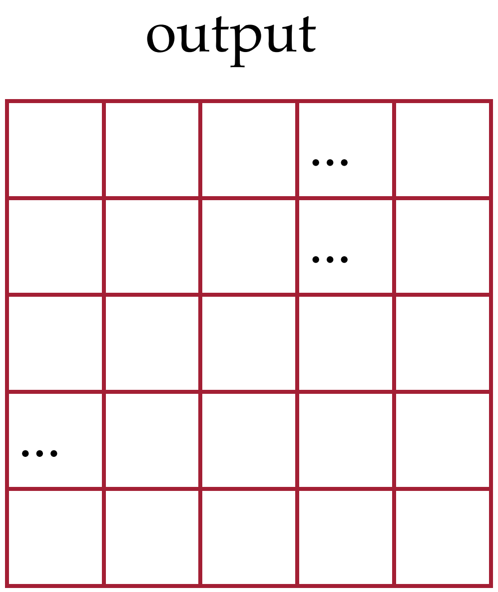
shifting makes sense spatially (a cat can be anywhere)

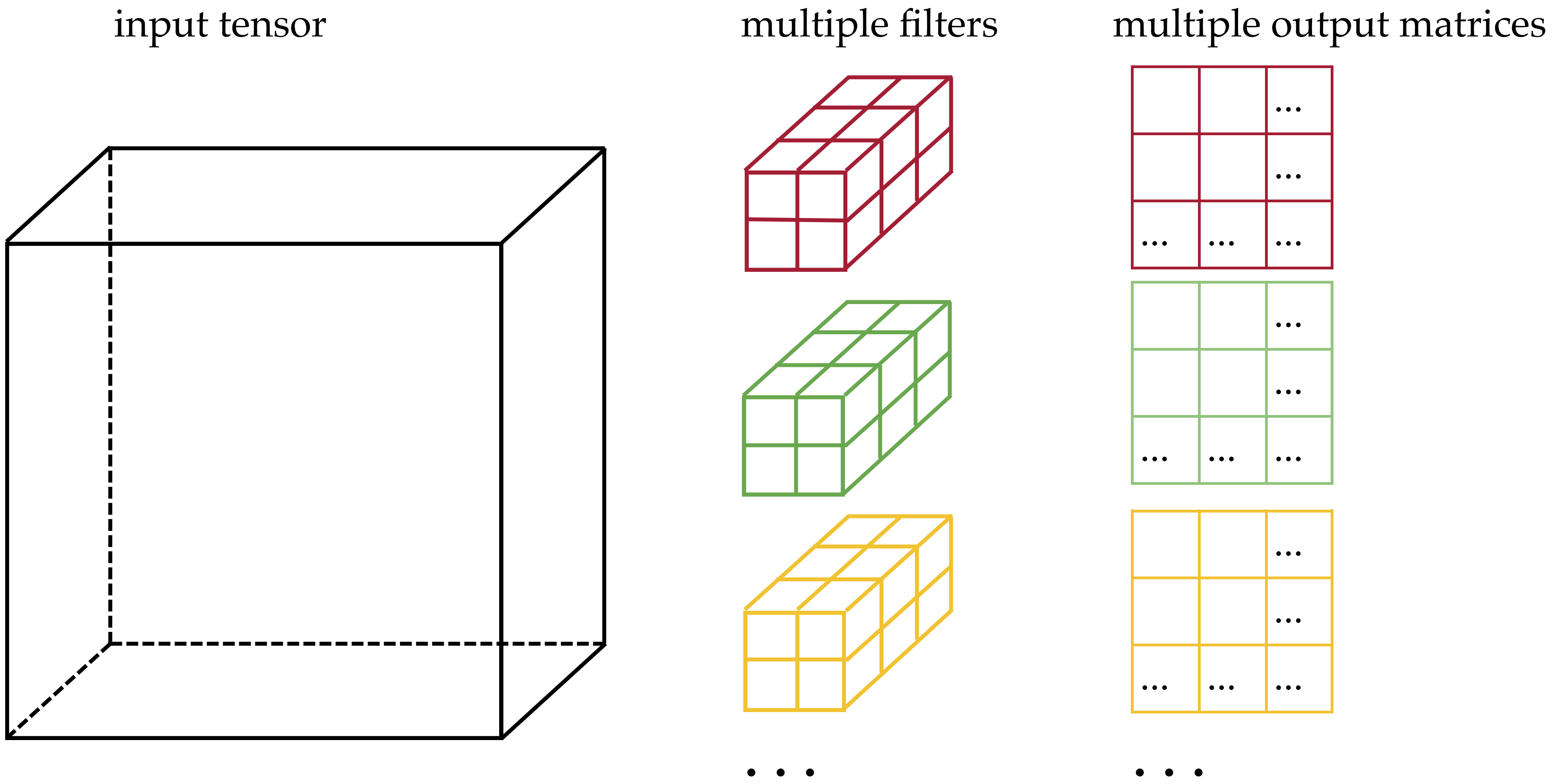
but not across channels (red ≠ shifted green)

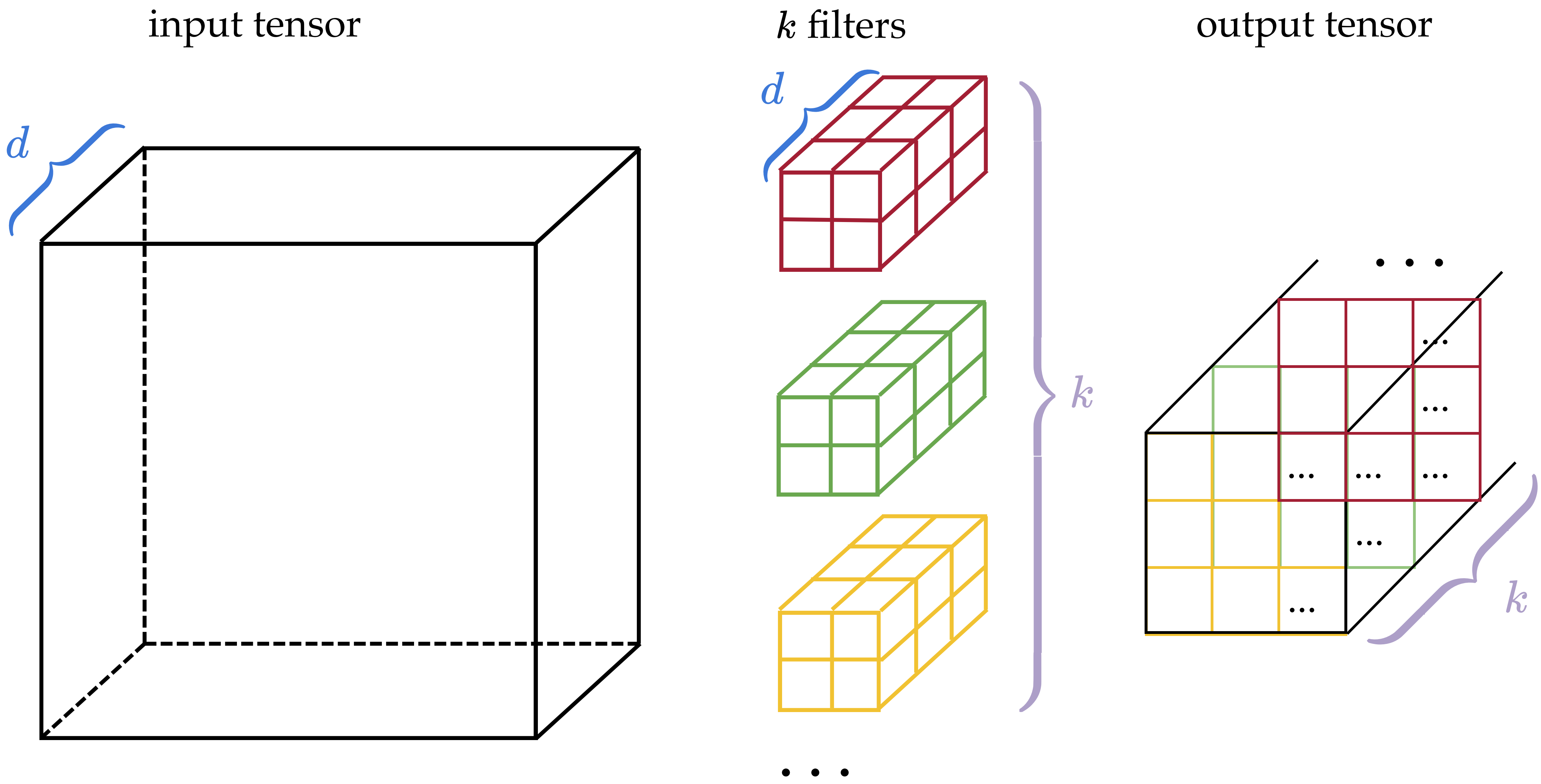
3D conv is used when the third axis is spatial / temporal (MRI, video) full-depth 2D convolution



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




input tensor
multiple filters
multiple output matrices

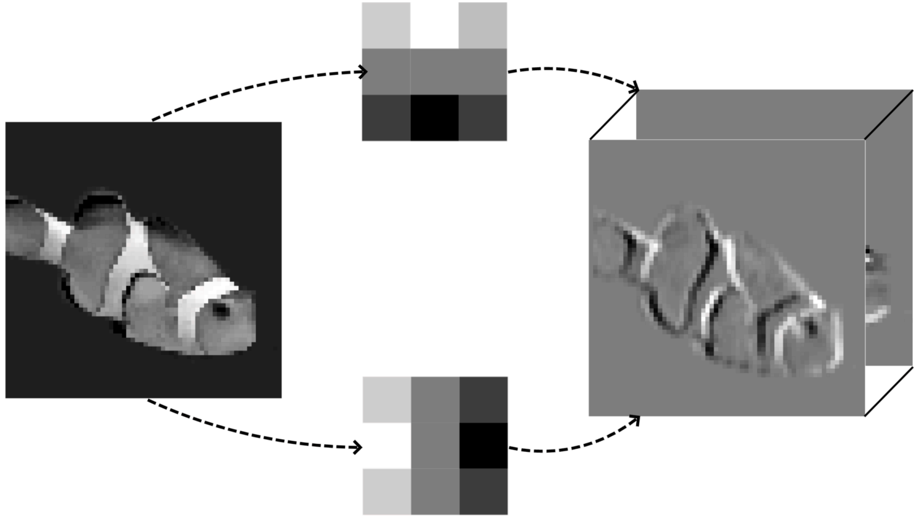


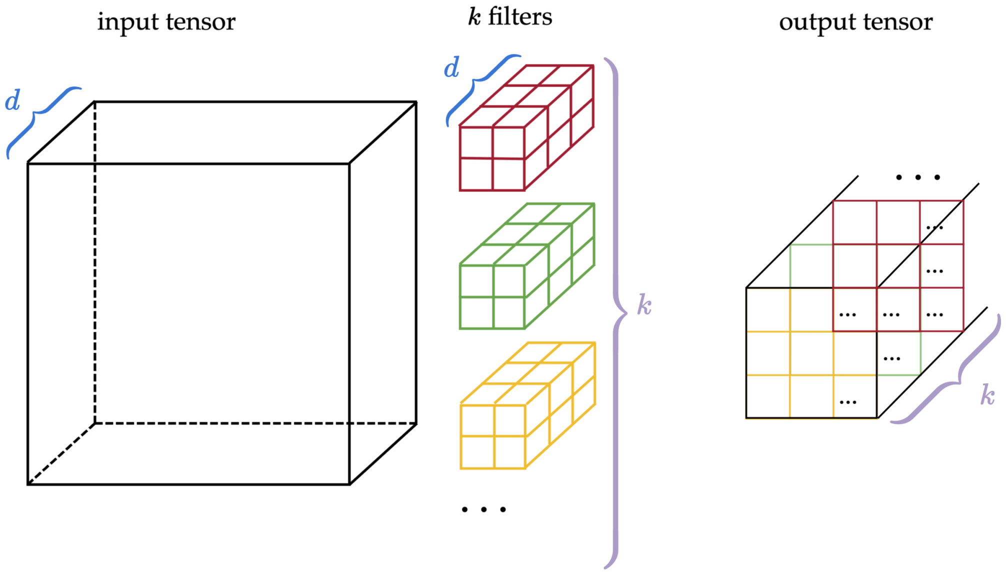
input tensor
k filters
output tensor
d
d
k
k

Every convolutional layer works with 3d tensors:

1. color input

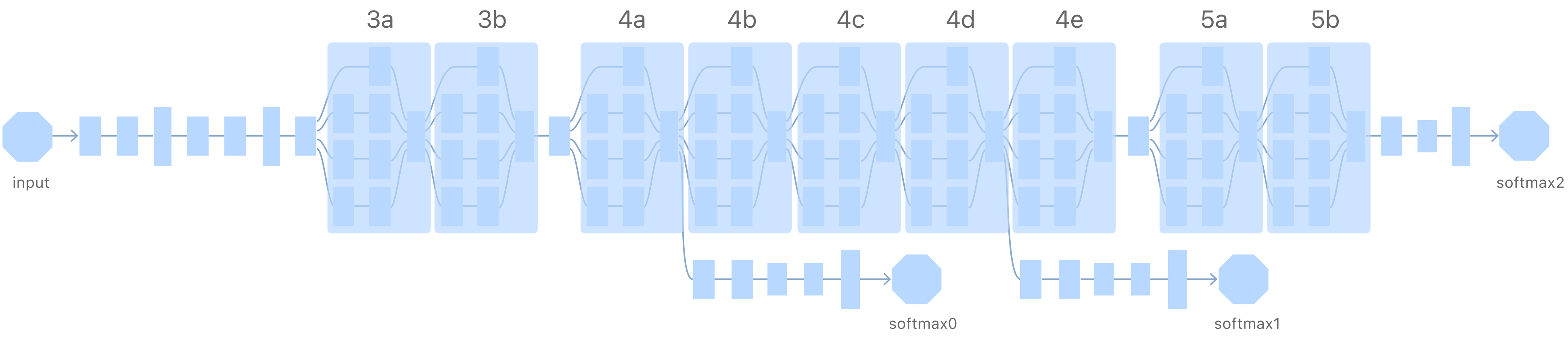
2. the use of multiple filters in doing 2d convolution





Feature Visualization — Appendix

After reading "Feature Visualization" you may be curious what other channels of GoogLeNet look like.

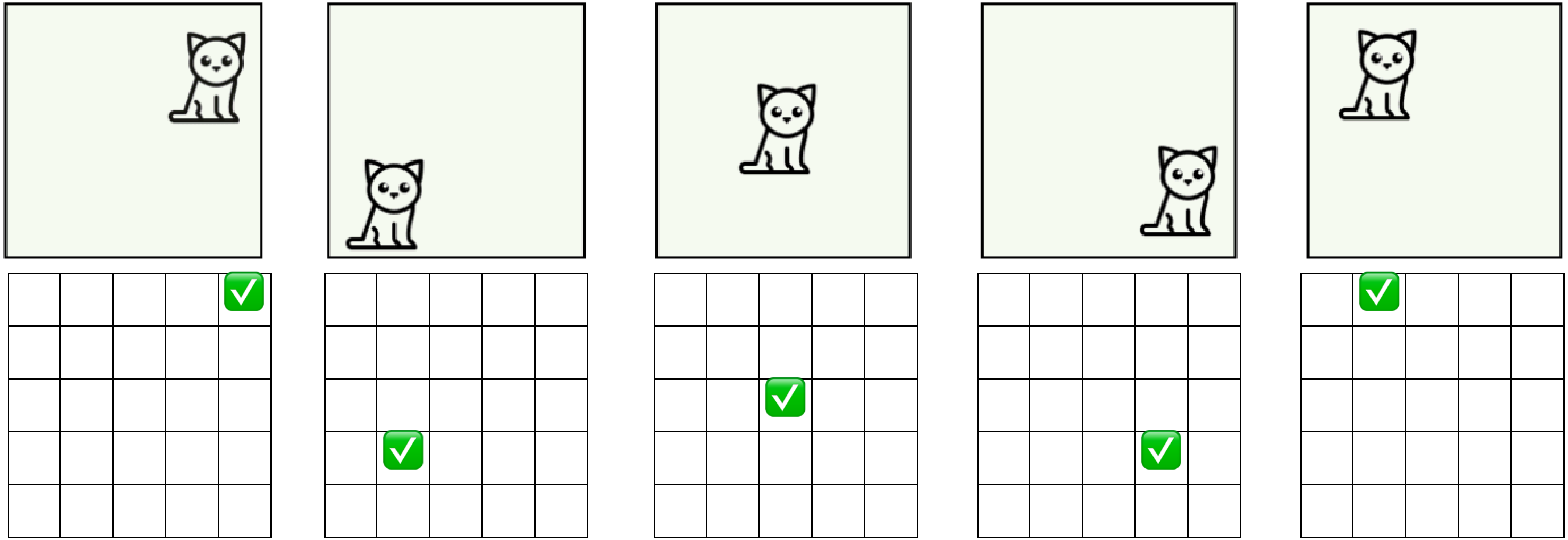


This appendix contains layers 3a through 5b of GoogLeNet.

Outline

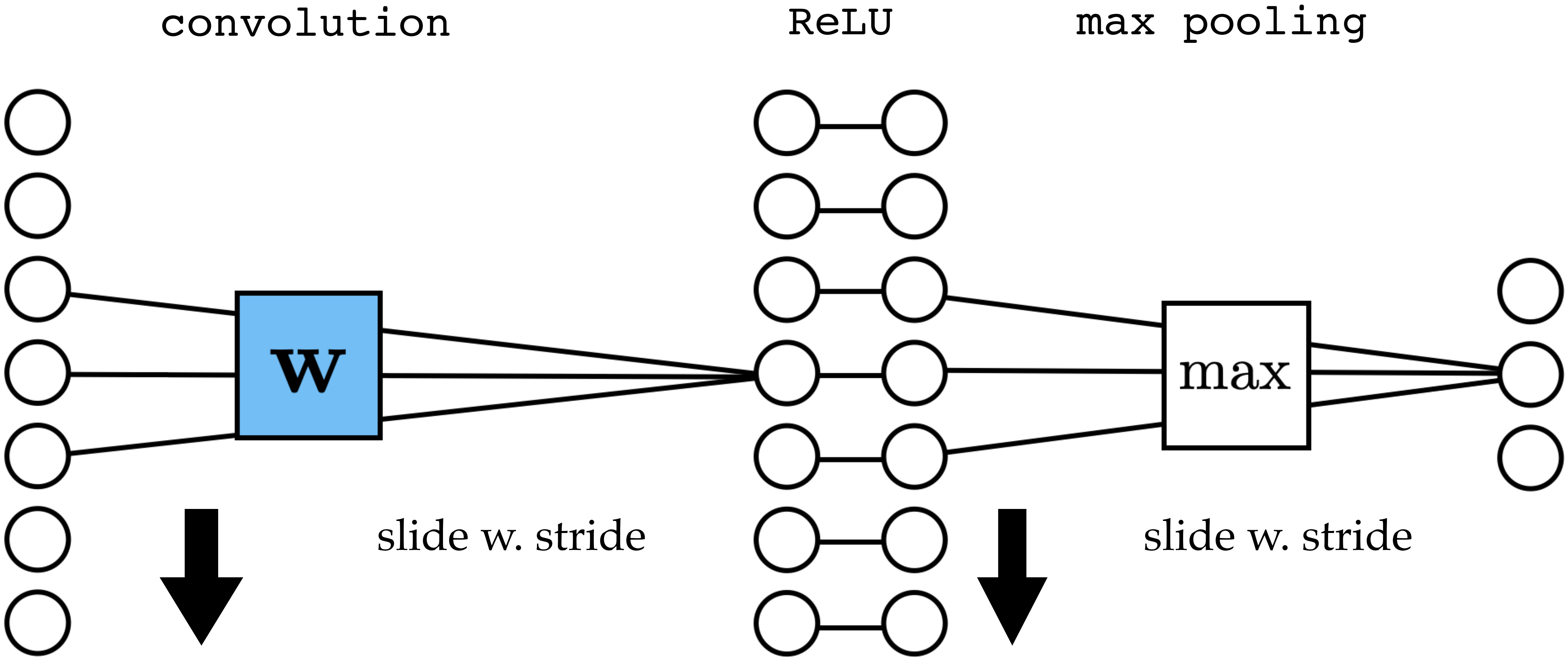
- Vision problem structure
- Convolution
  - 1-dimensional and 2-dimensional *convolution*
  - 3-dimensional *tensors*
- Max pooling
- (Case studies)

convolution helps detect pattern, but ...



cat moves, detection moves

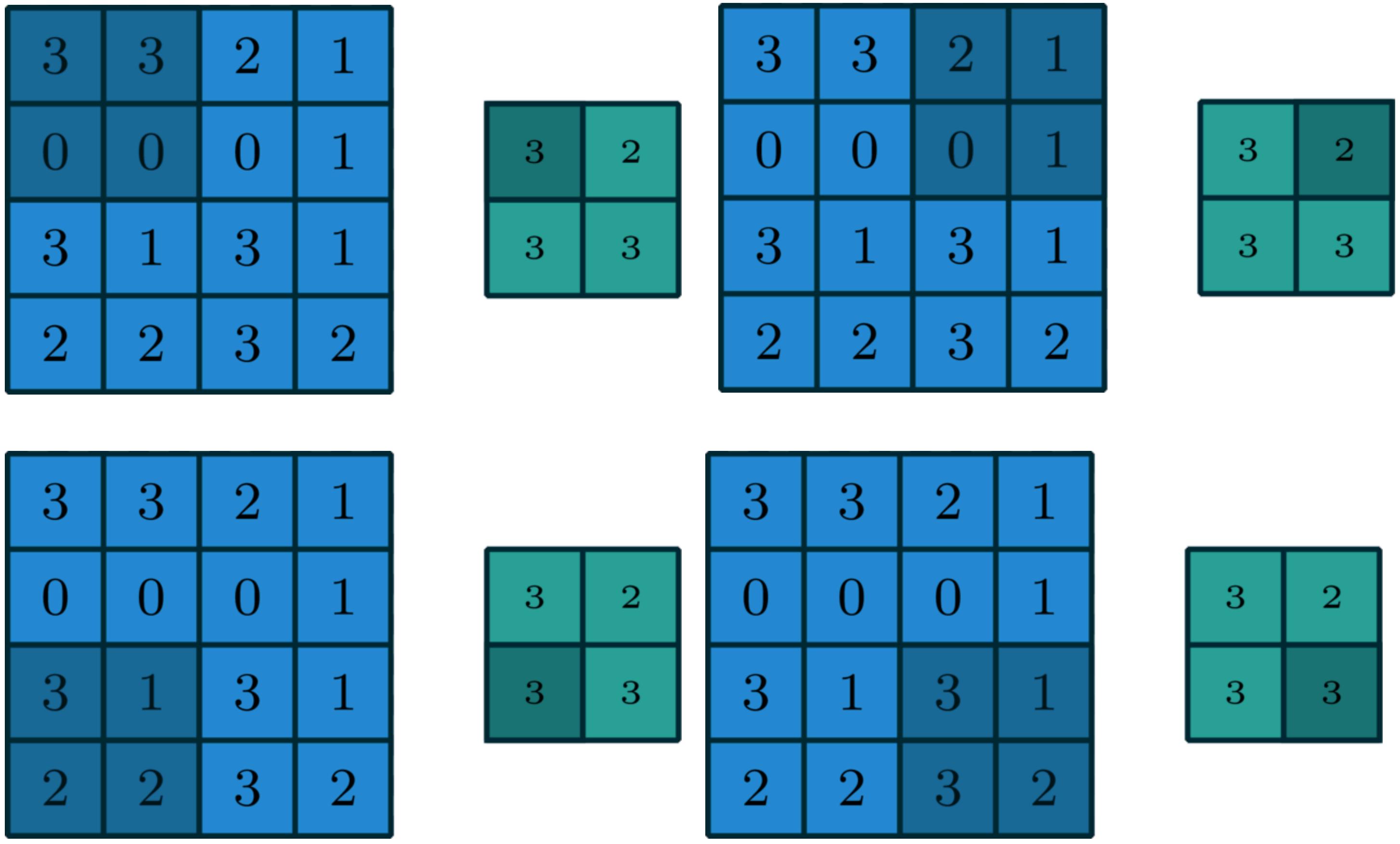
1d max pooling

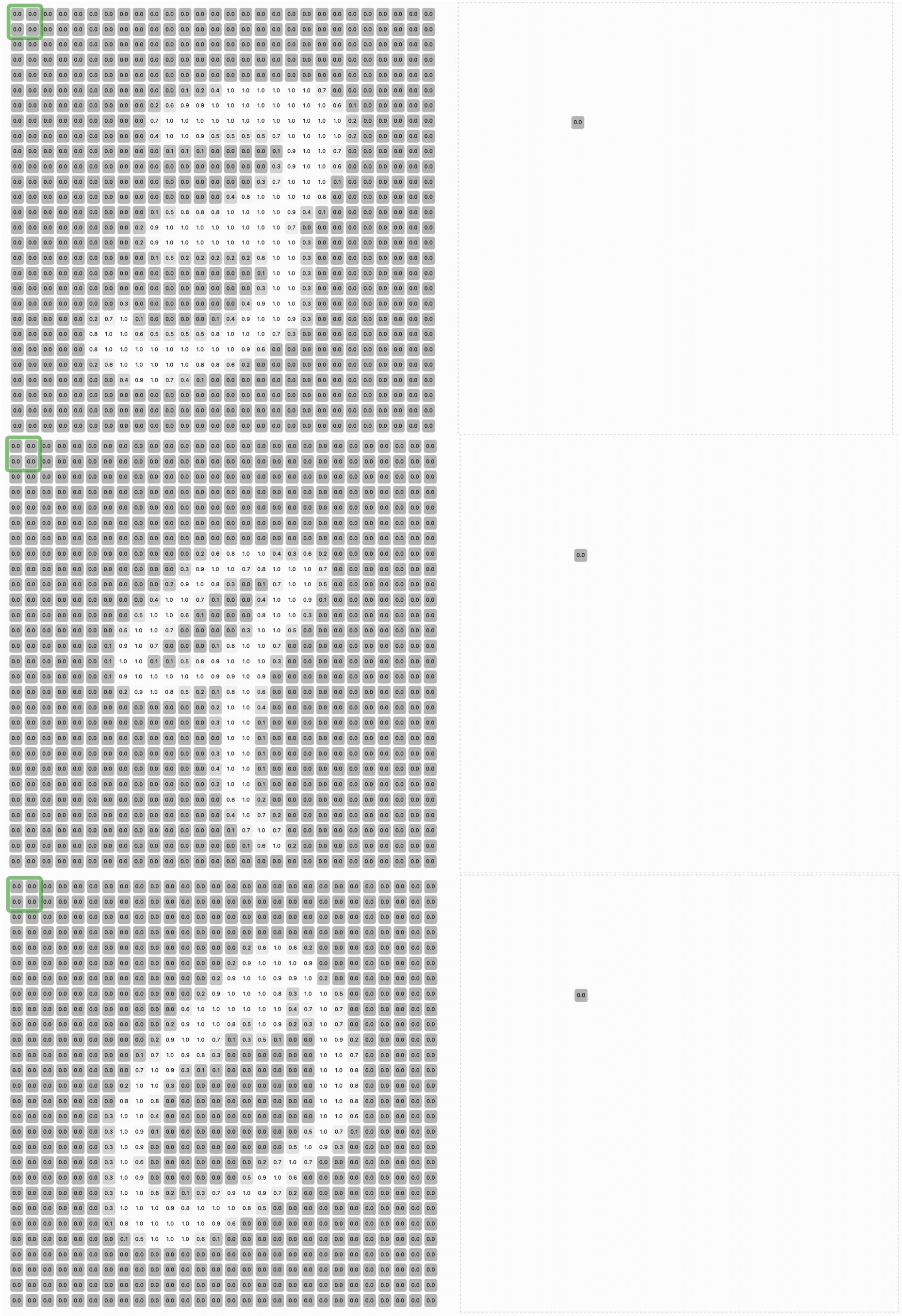


learnable filter weights
detects pattern

no learnable parameter
summarizes strongest response

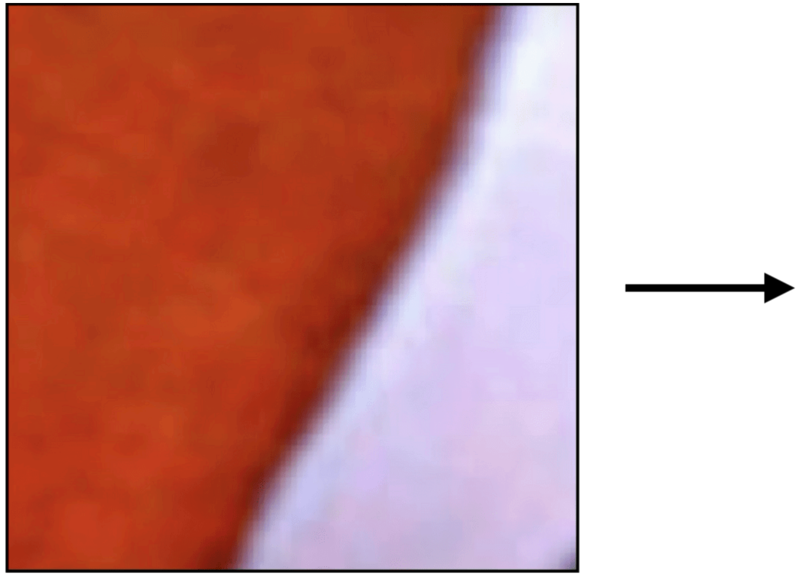
2d max pooling

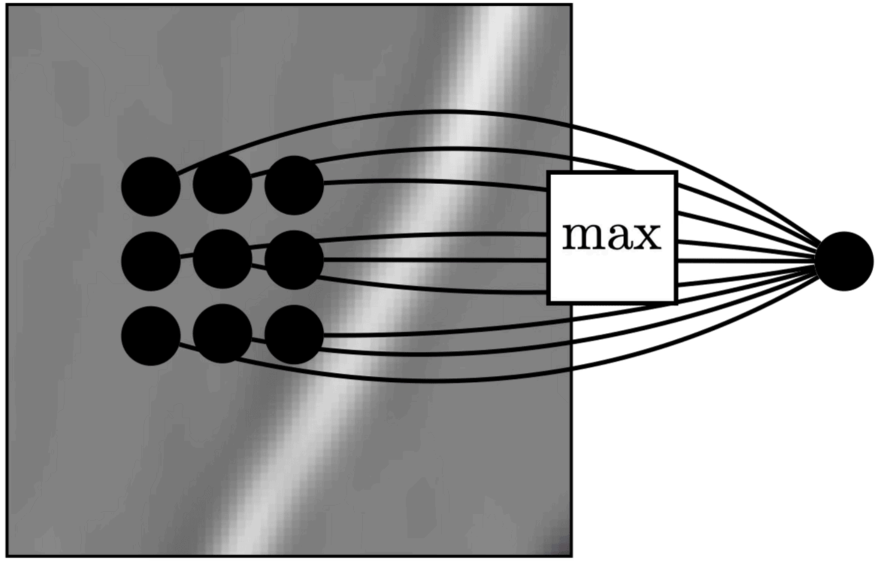




[image edited from vdumoulin, gif adapted from demo source]

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

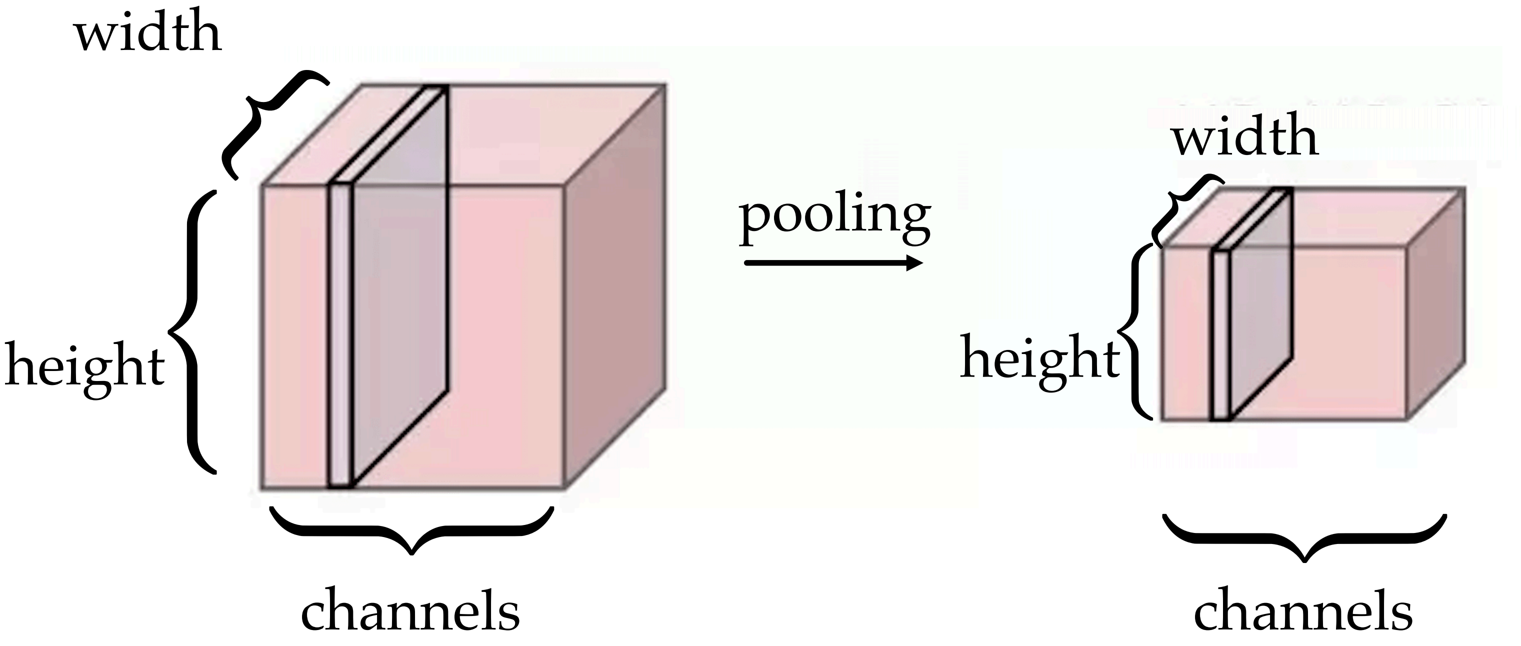




large response regardless of exact position of edge

[image credit Philip Isola]

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

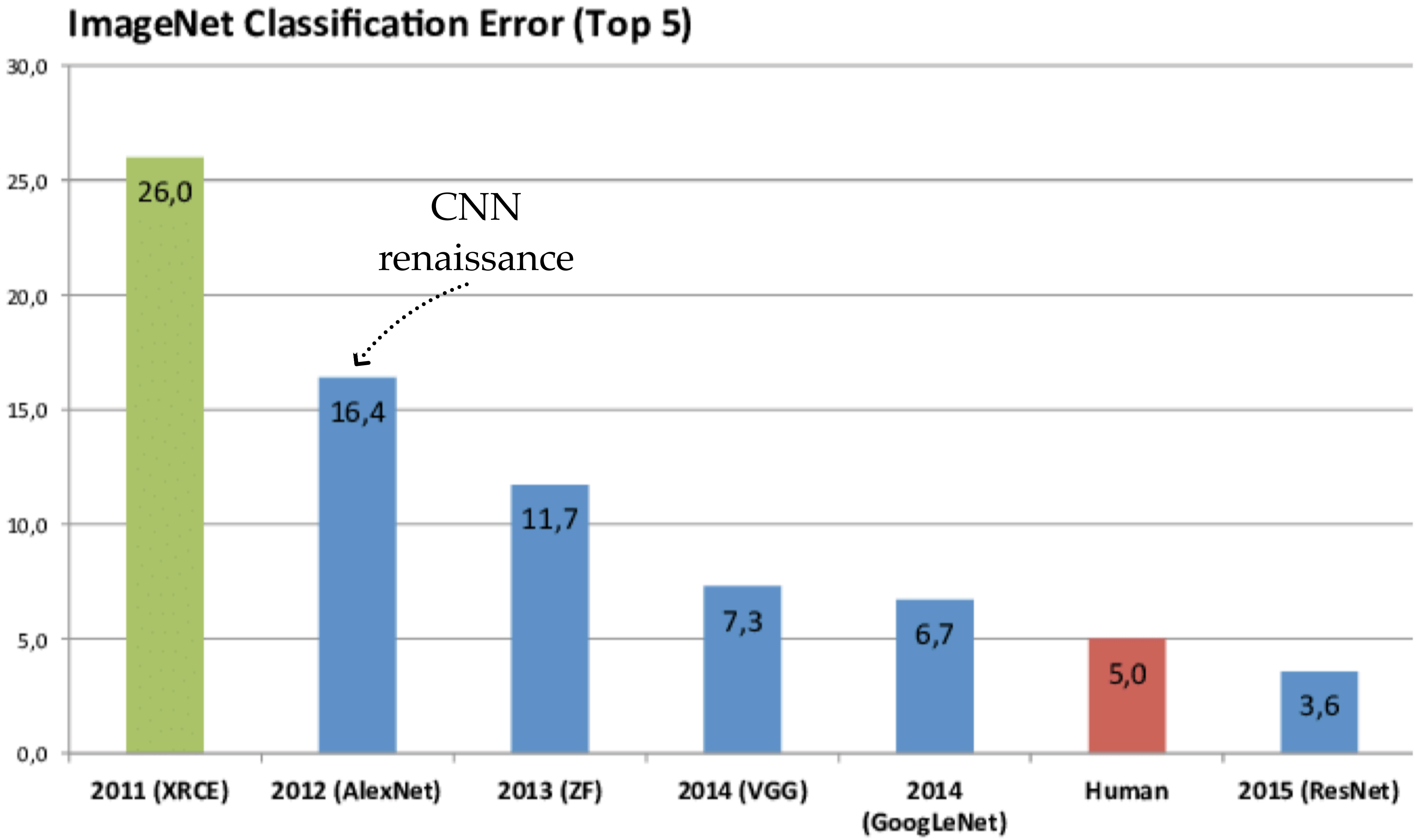


applied independently across all channels

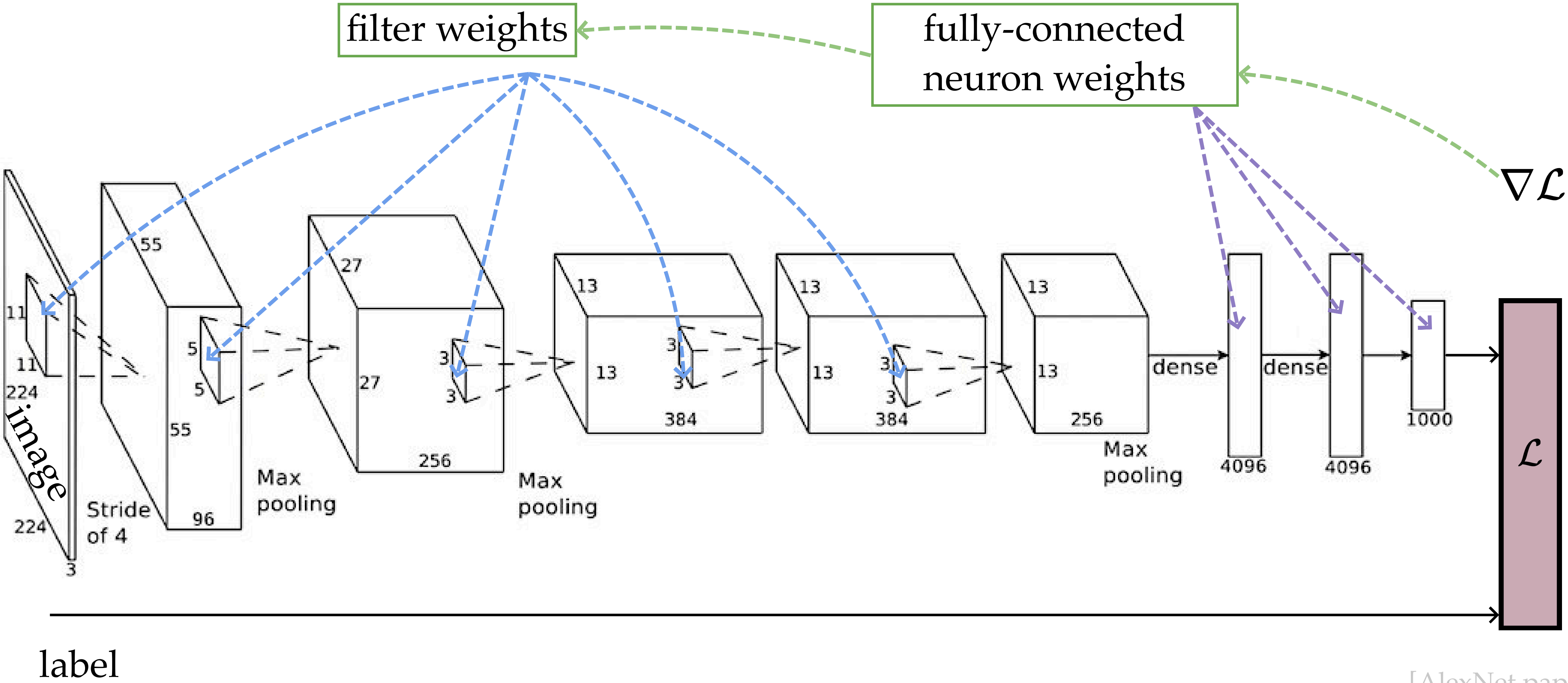
so the *channel* dimension remains *unchanged*

Outline

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

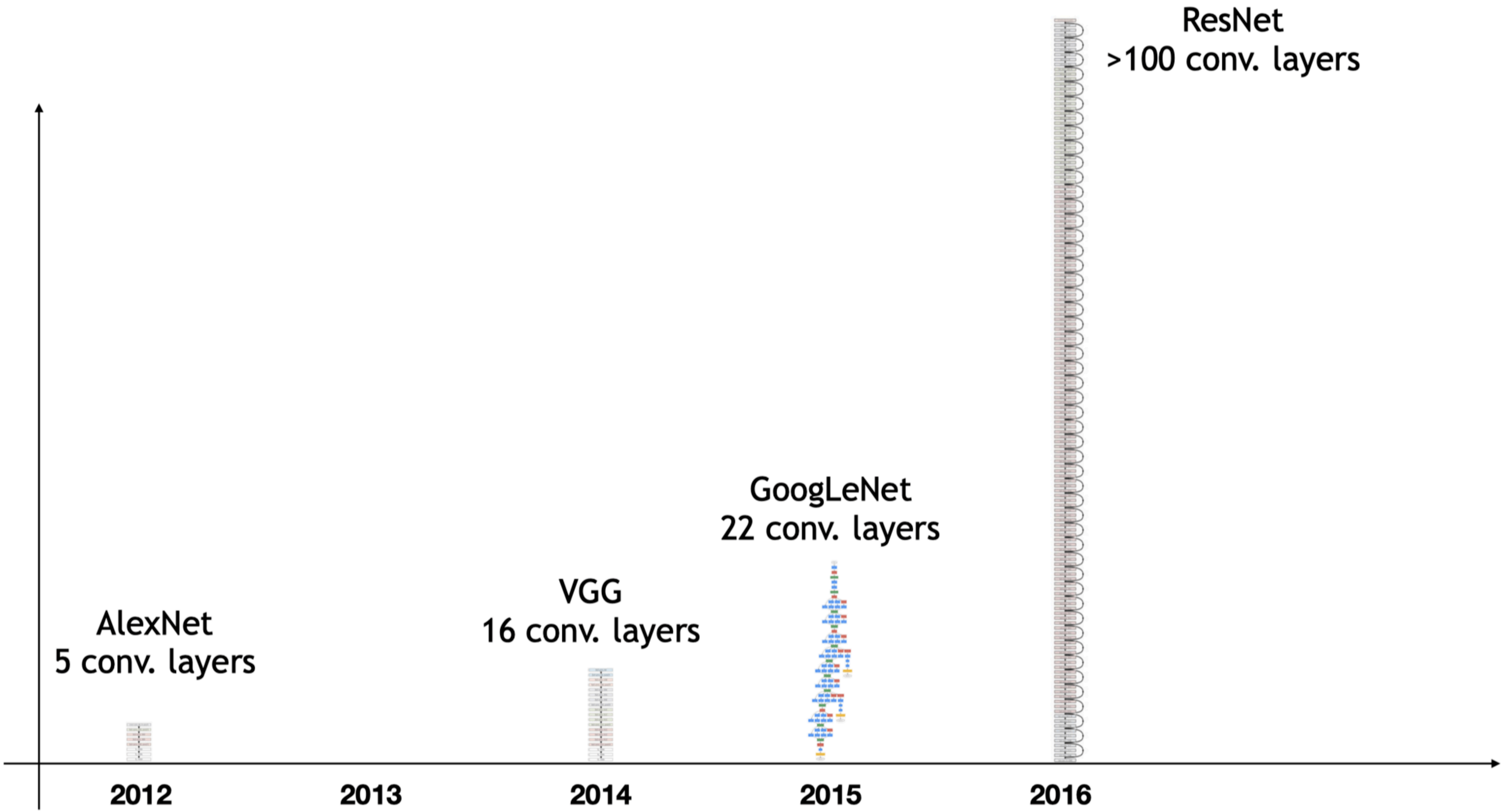


[image credit Philip Isola]



[AlexNet paper]

[all max pooling are 3×3 filter, stride 2; pooled outputs not explicitly shown on diagram: 27×27×96, 13×13×256, 6×6×256]

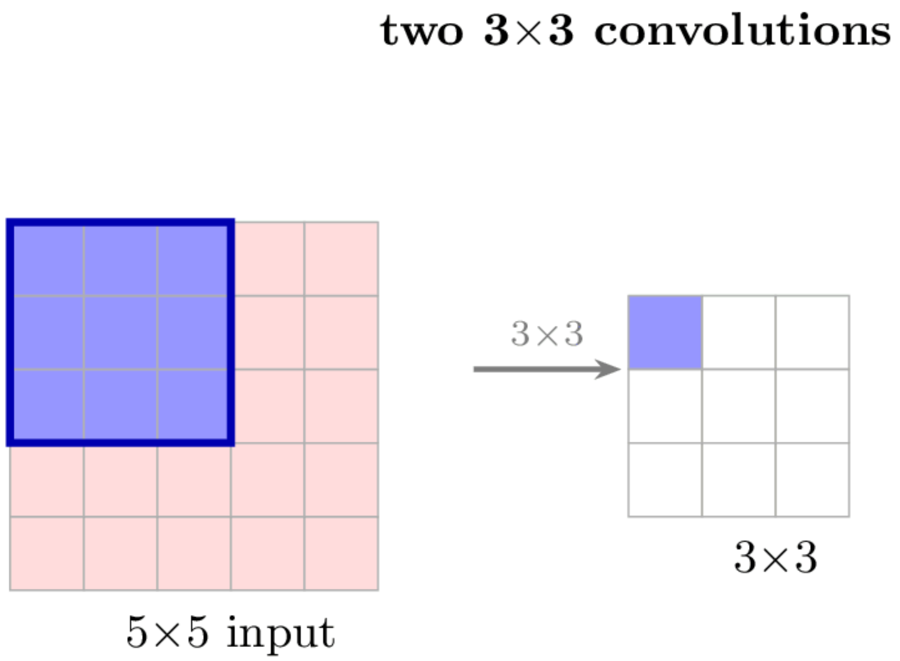


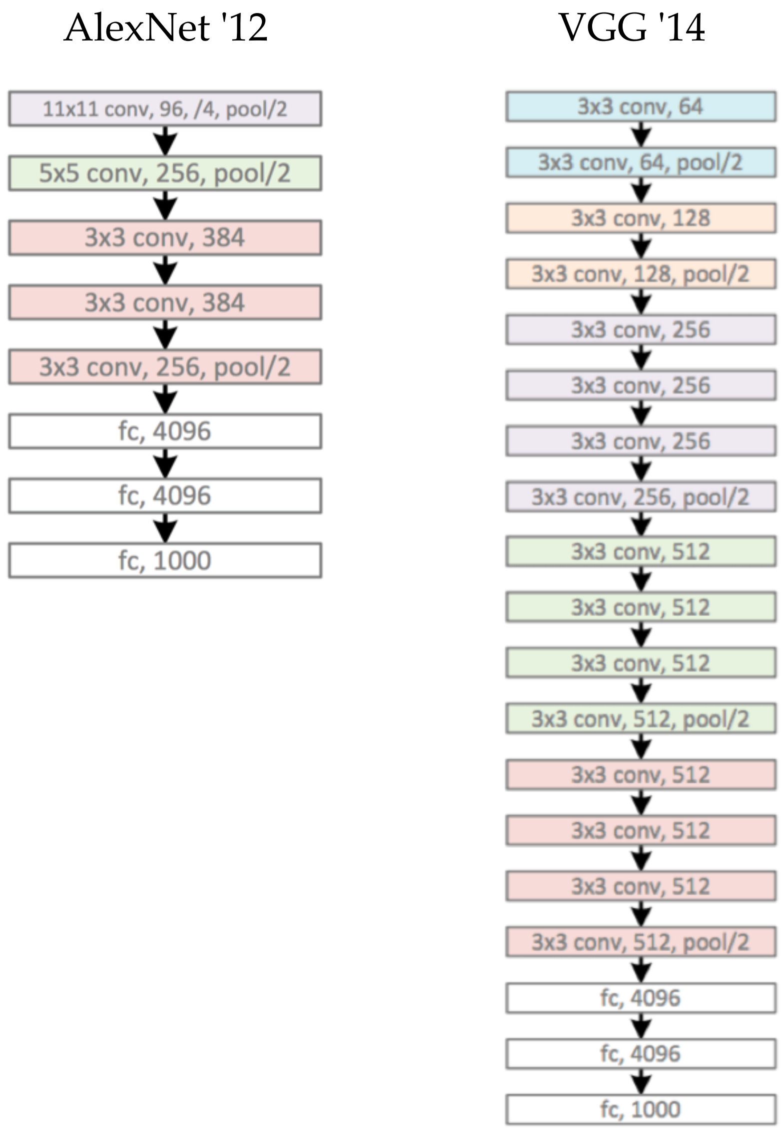
[image credit Philip Isola]

VGG '14

Main developments:

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

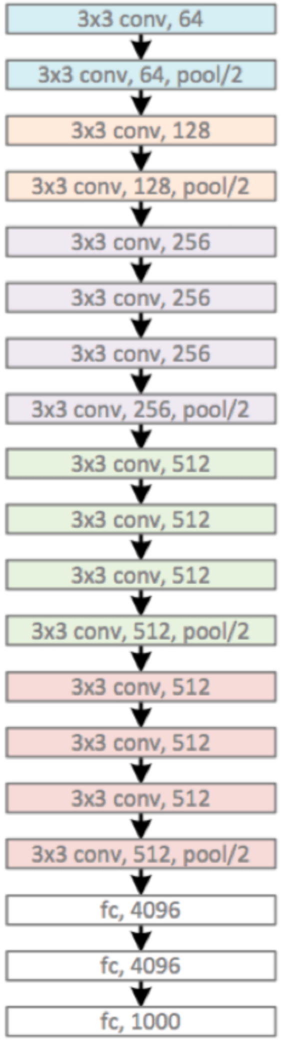




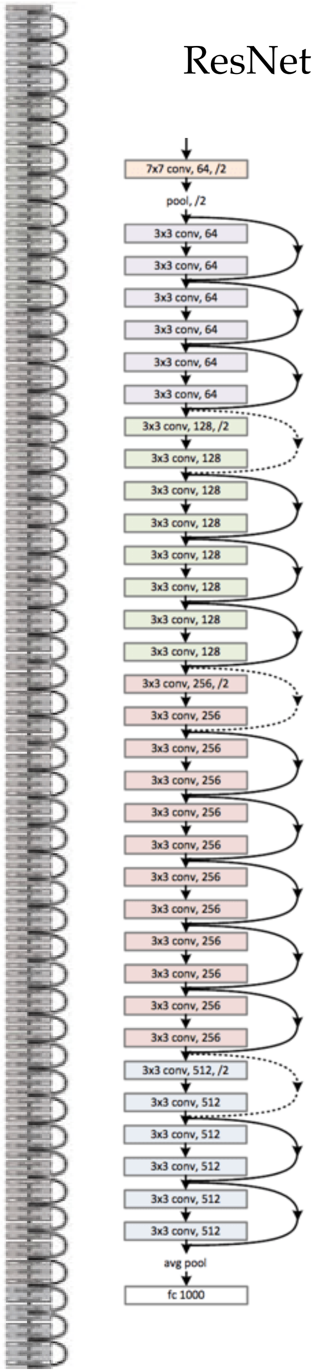
"Very Deep Convolutional Networks for Large-Scale Image Recognition", Simonyan & Zisserman. ICLR 2015

[image credit Philip Isola and Kaiming He]

VGG '14

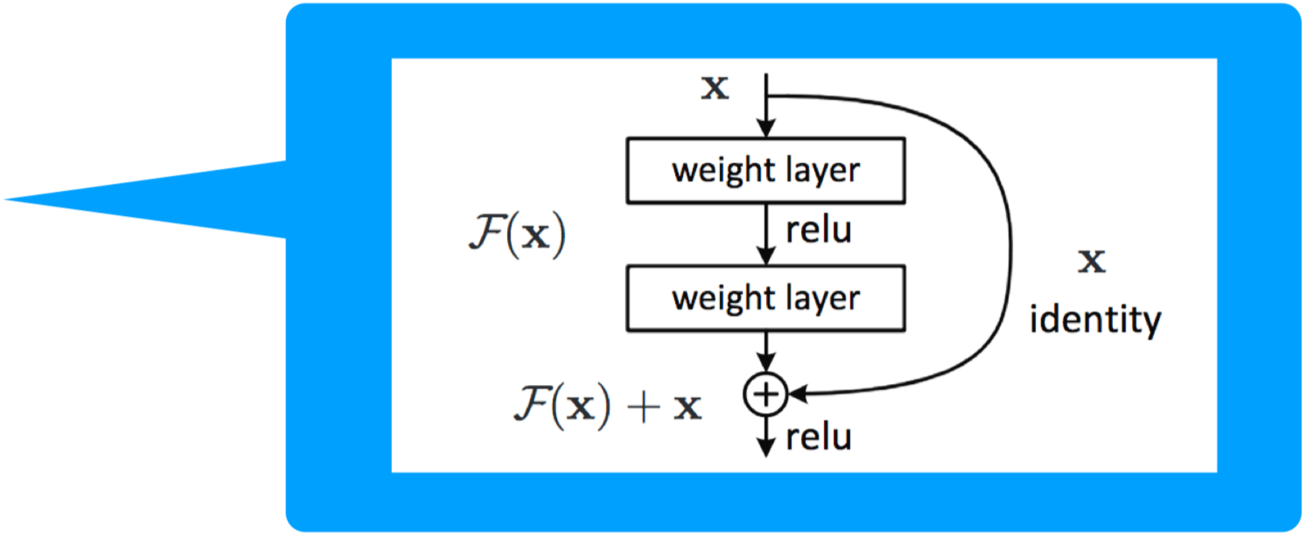


ResNet '16



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

- 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.