https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 7: Convolutional Neural Networks

Shen Shen

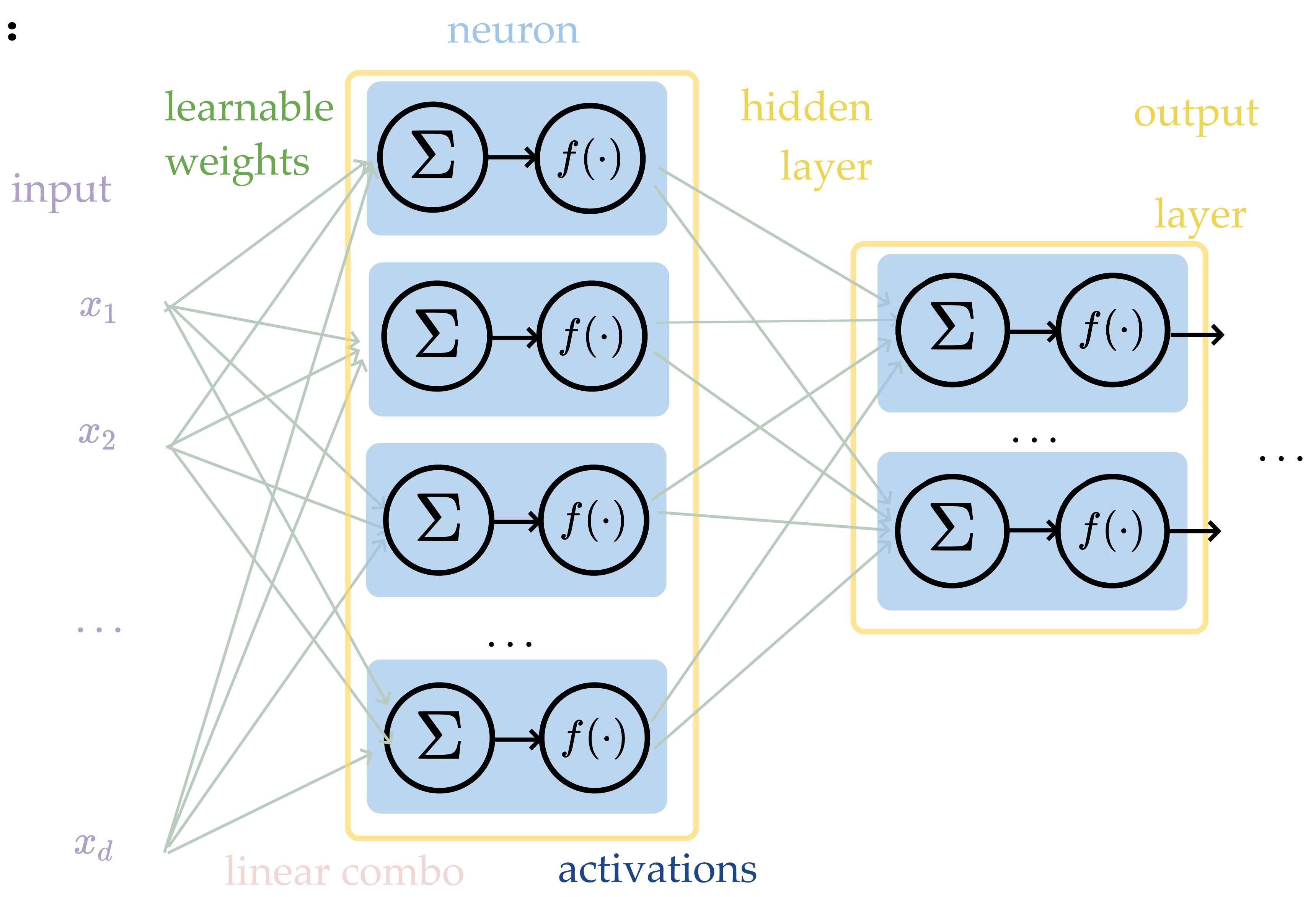
March 21, 2025

11am, Room 10-250

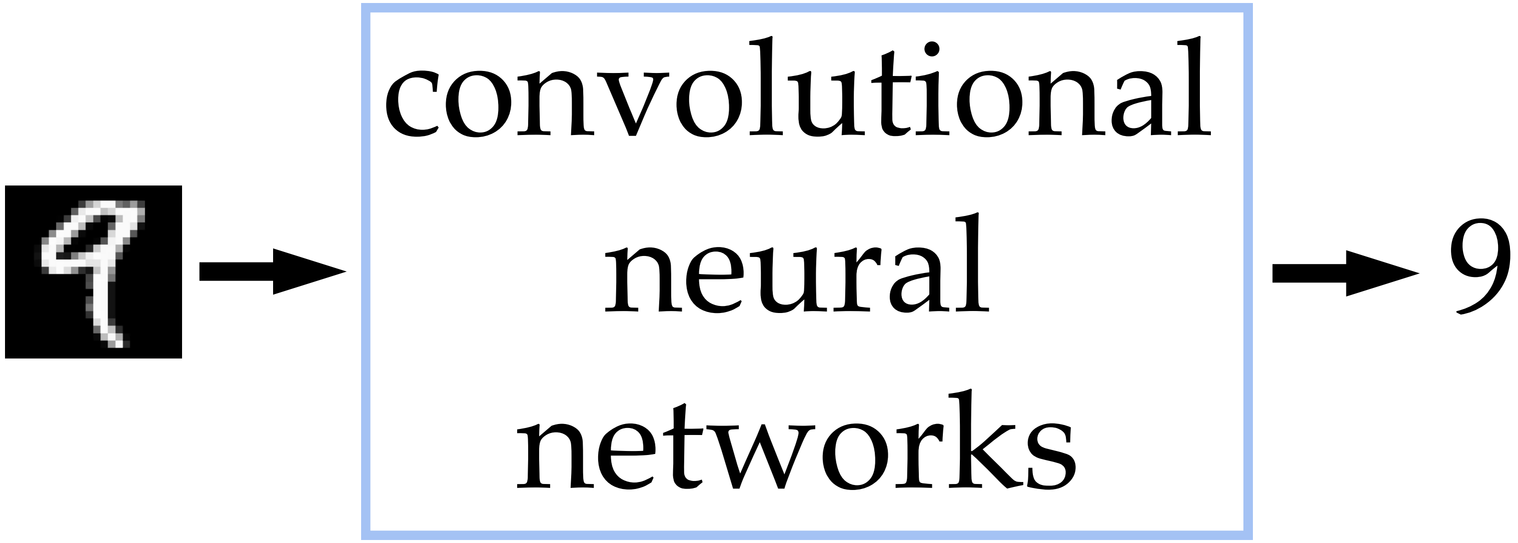
Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
    - 1-dimensional and 2-dimensional *convolution*
    - 3-dimensional *tensors*
- Max pooling
- (Case studies)

Recap:

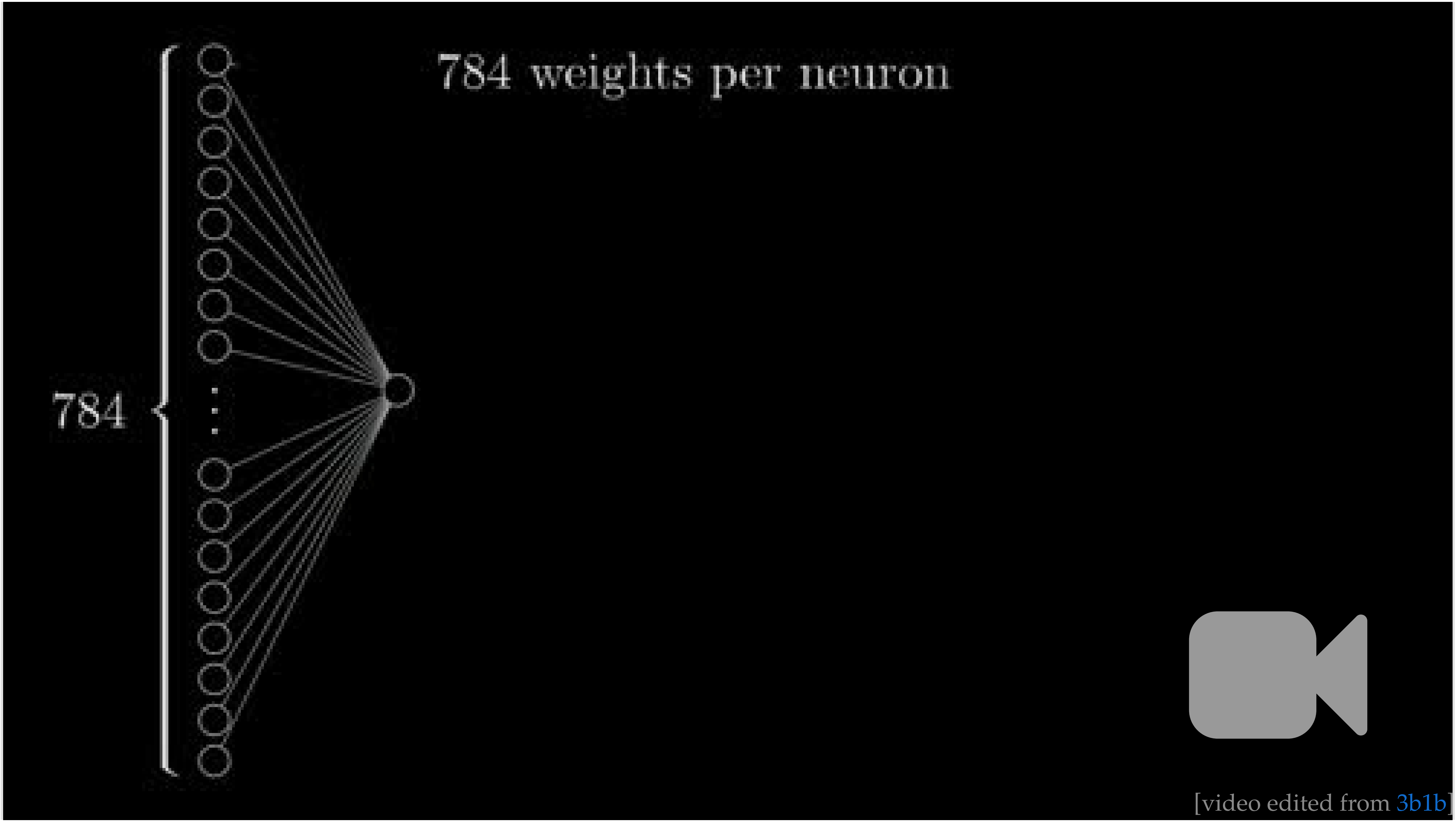


neuron
learnable weights
input
x_1
x_2
$\cdots$
x_d
hidden layer
output layer
Σ
$f(\cdot)$
linear combo
activations

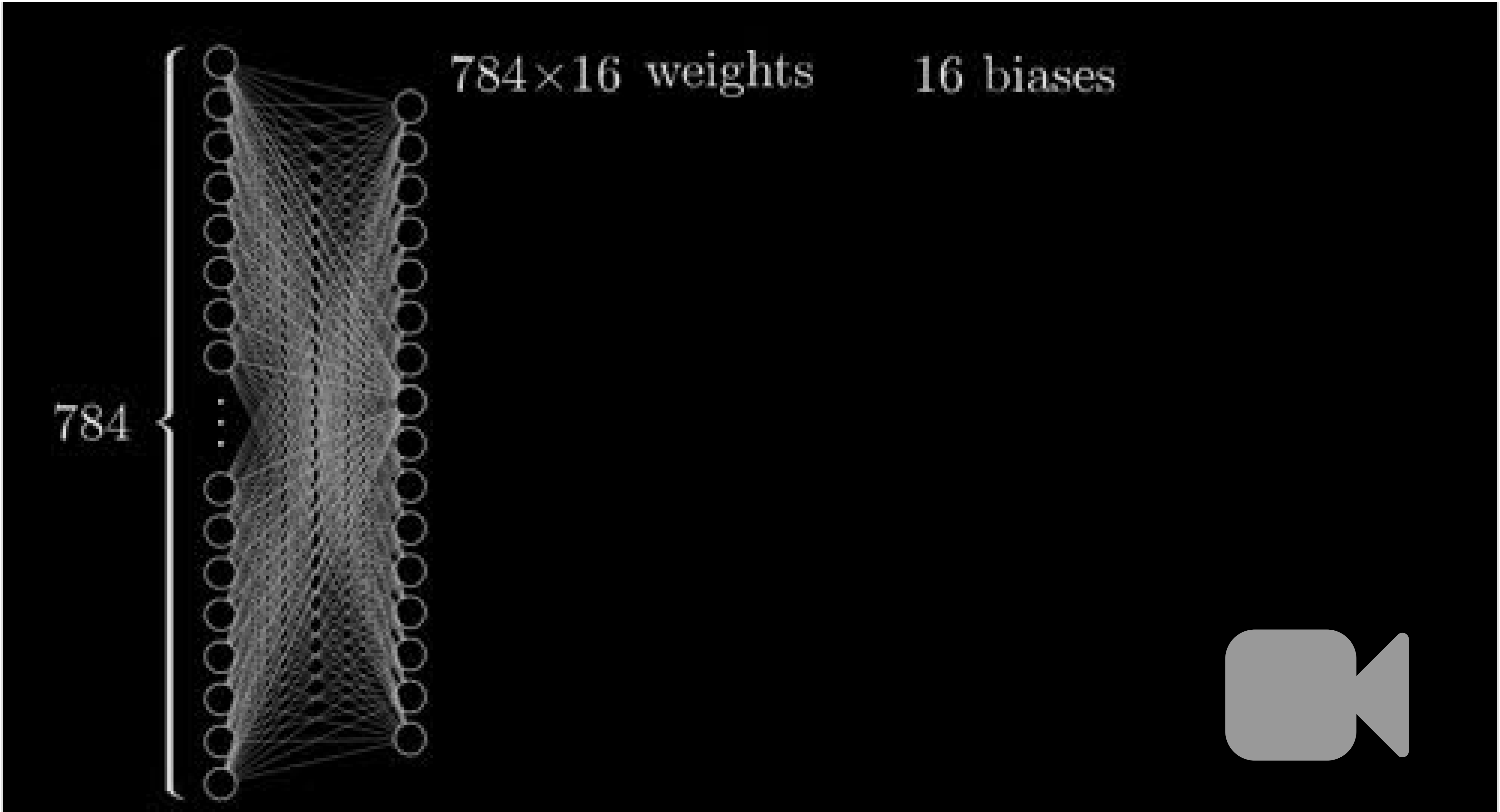


1. Why do we need a special network for images?
2. Why is CNN (the) special network for images?

[video edited from 3b1b]



[video edited from 3b1b]



[video edited from 3b1b]

426-by-426
grayscale image

Use the same 2 hidden-layer network, need to learn ~3M parameters.

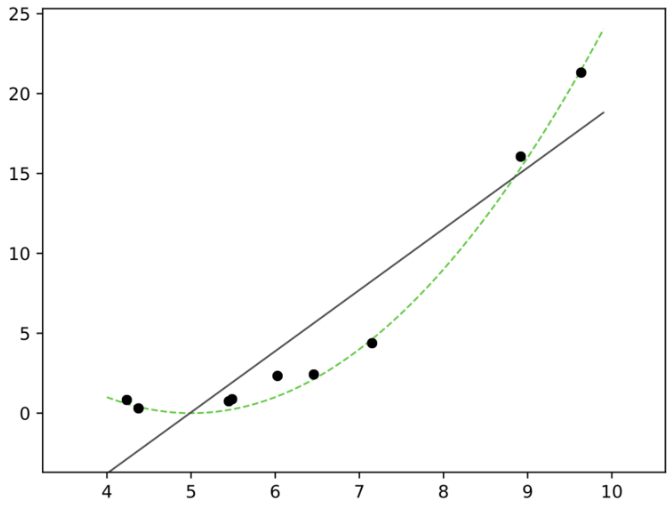
For higher-resolution images, or more complex tasks, or larger networks, the number of parameters can just grow very fast.

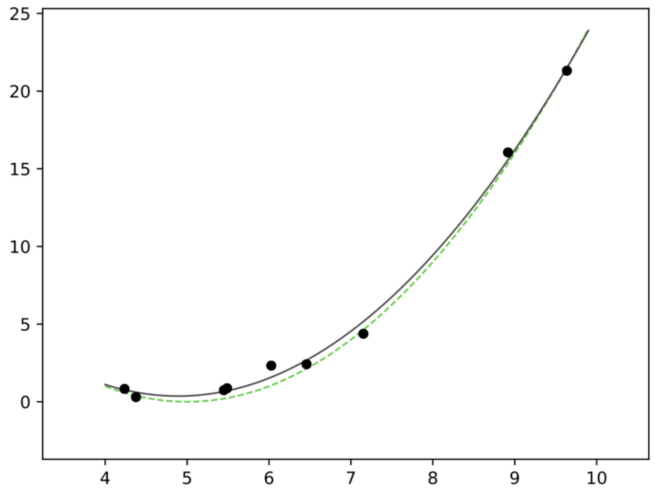
Why do we need a specialized network (hypothesis class)?

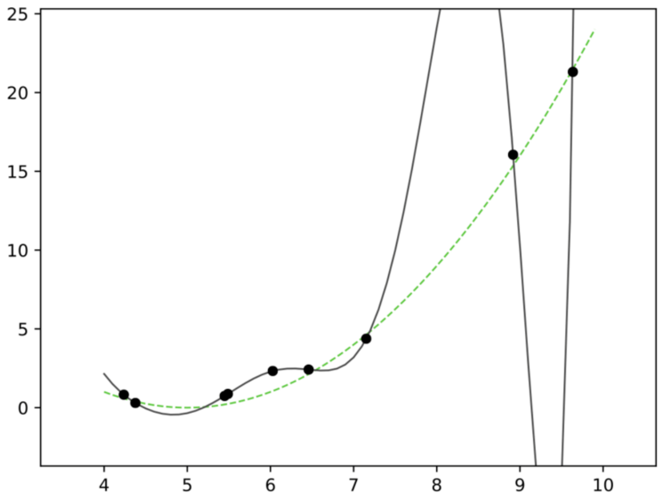
Partly, fully-connected nets don't scale well for vision tasks, but more importantly ...

Recall, more powerful models also tend to overfit

Underfitting	Appropriate model	Overfitting
$k = 1$	$k = 2$	$k = 10$







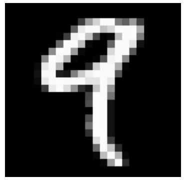
Recall, more powerful models also tend to overfit

https://playground.tensorflow.org/

Outline

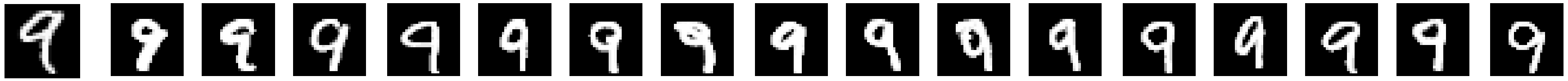
- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
    - 1-dimensional and 2-dimensional *convolution*
    - 3-dimensional *tensors*
- Max pooling
- (Case studies)

Why do we think

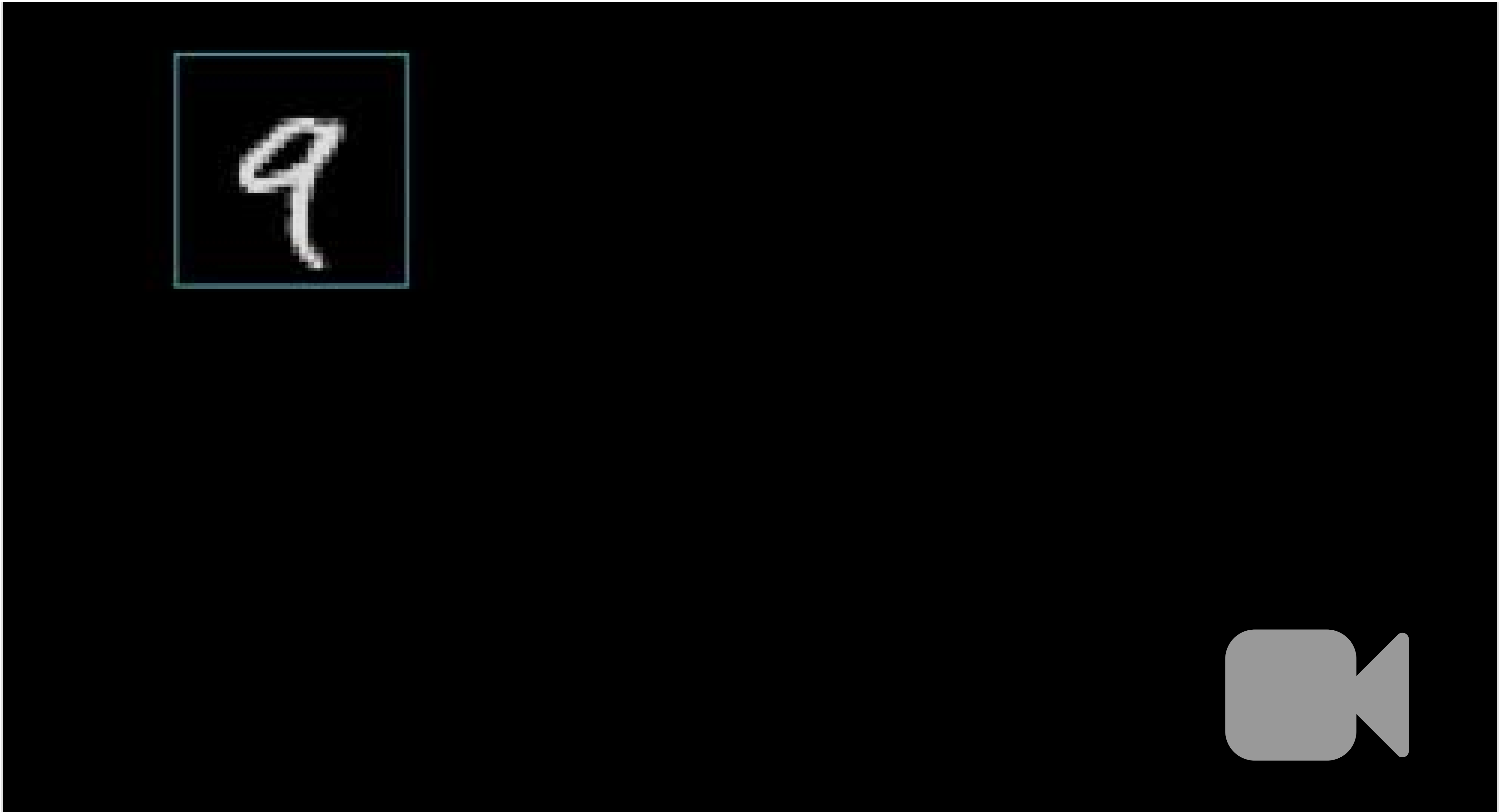


is 9?

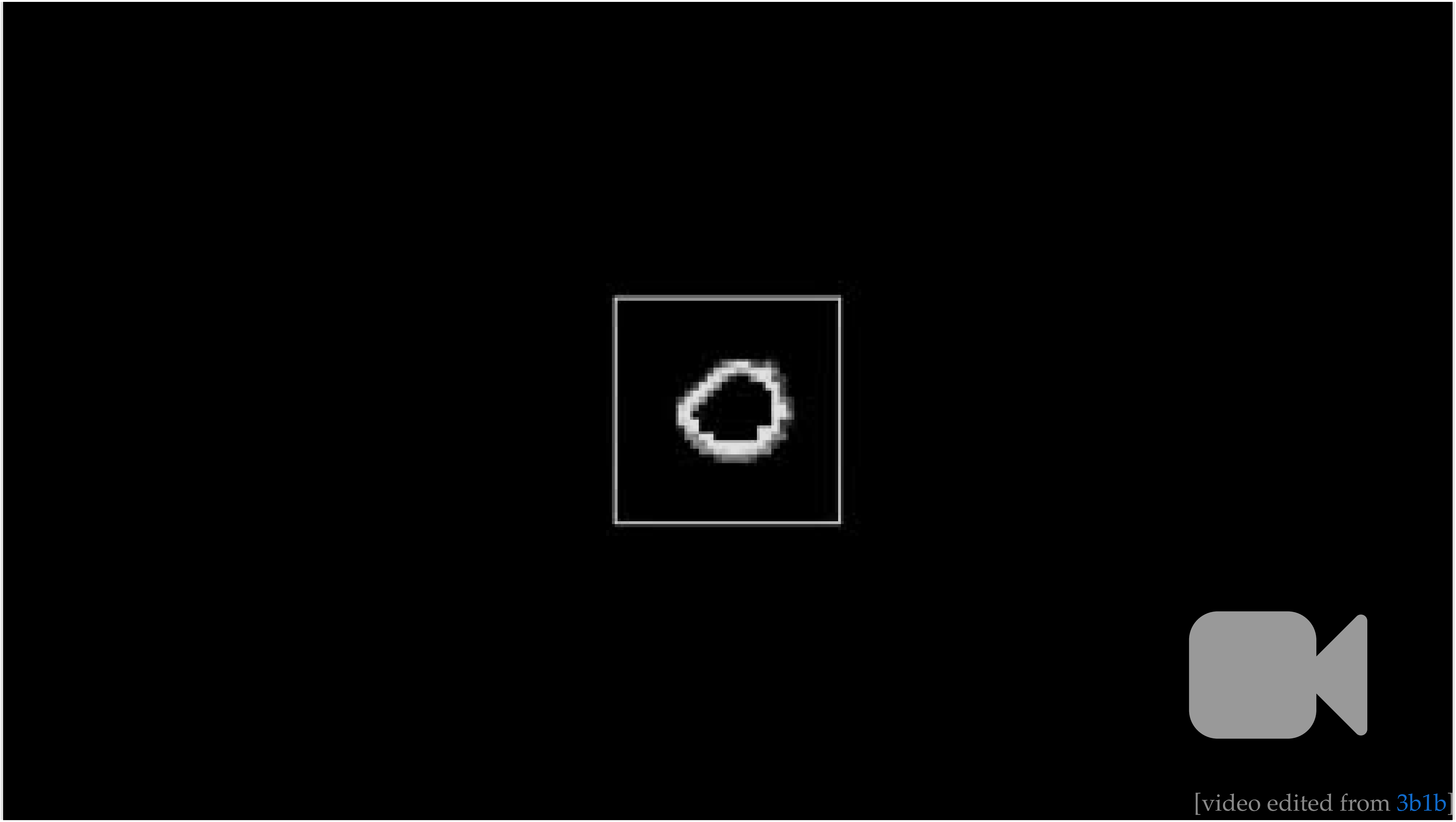
Why do we think any of

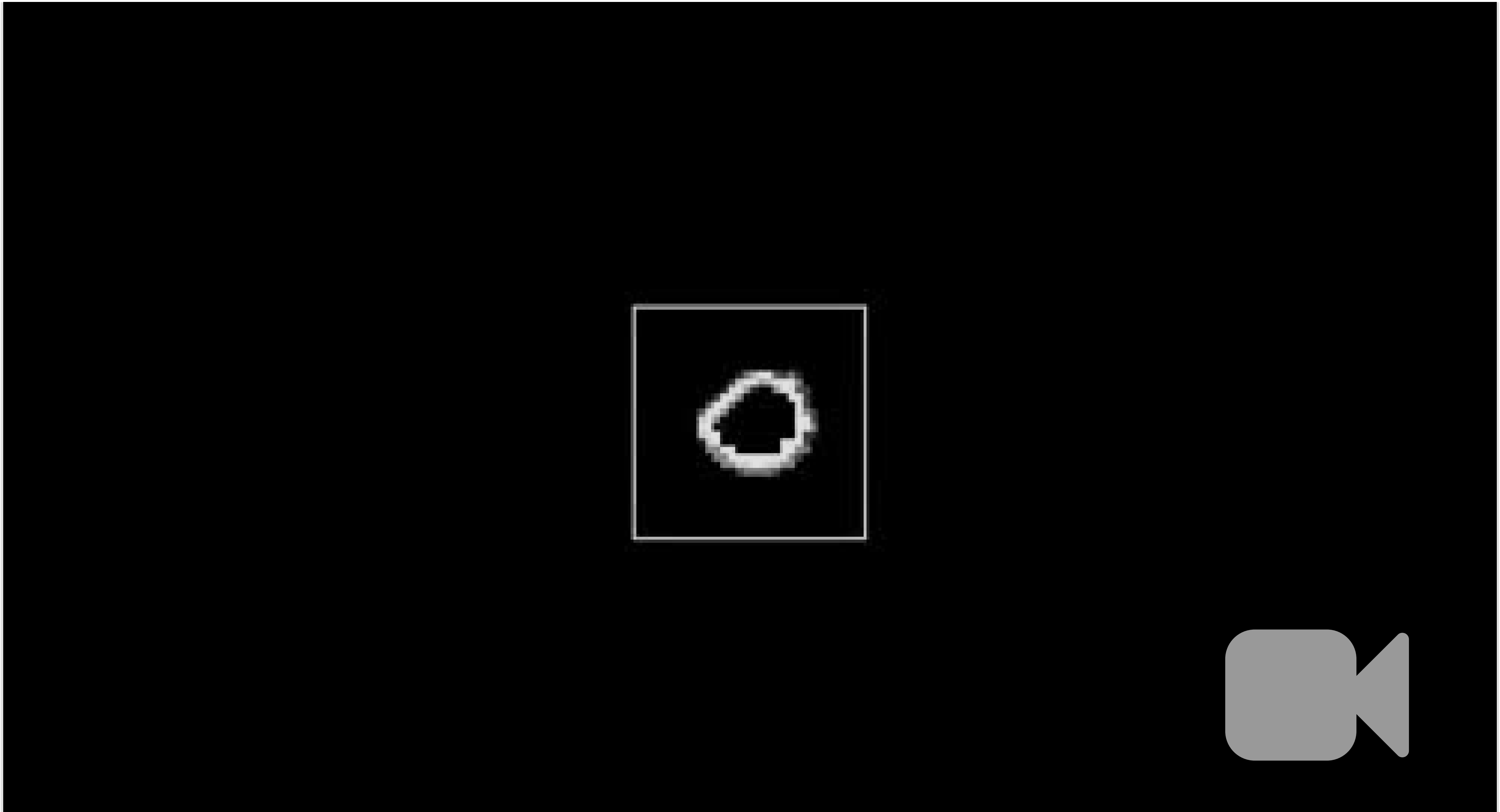


is 9?



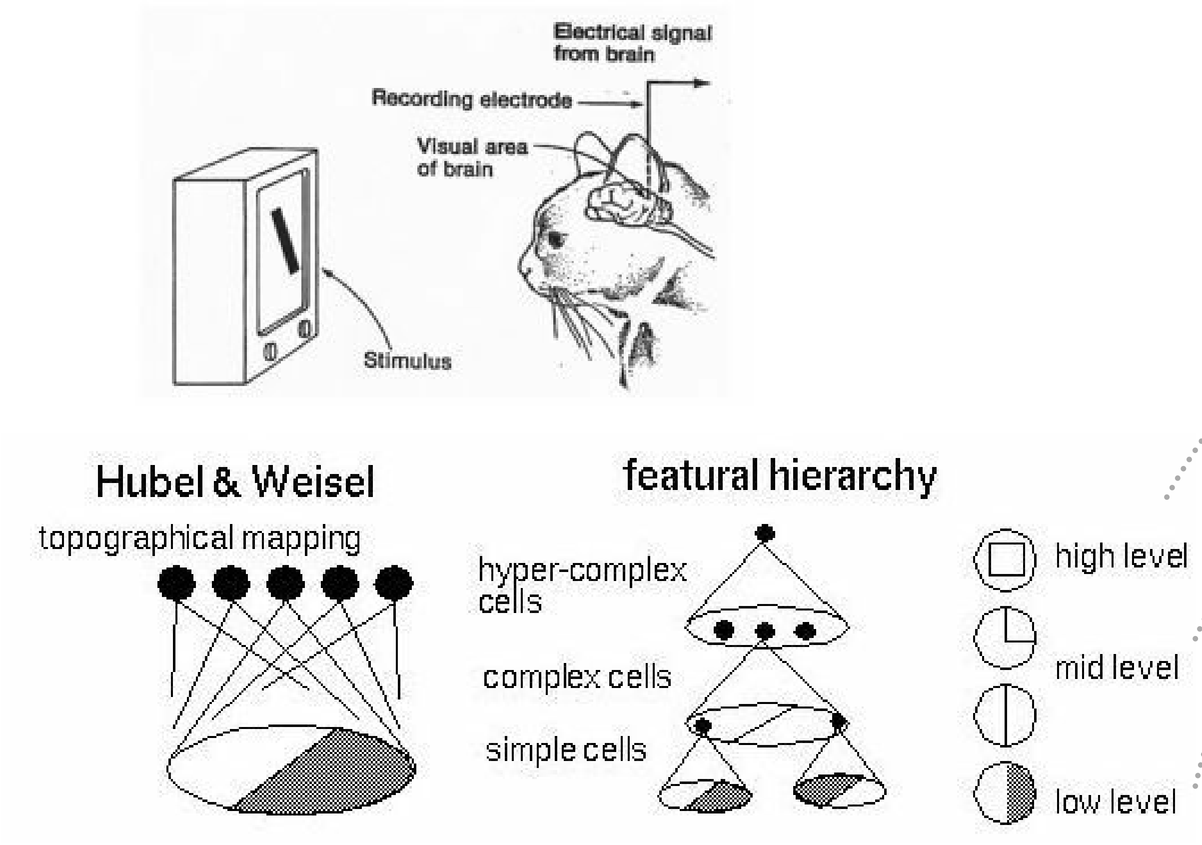
[video edited from 3b1b]

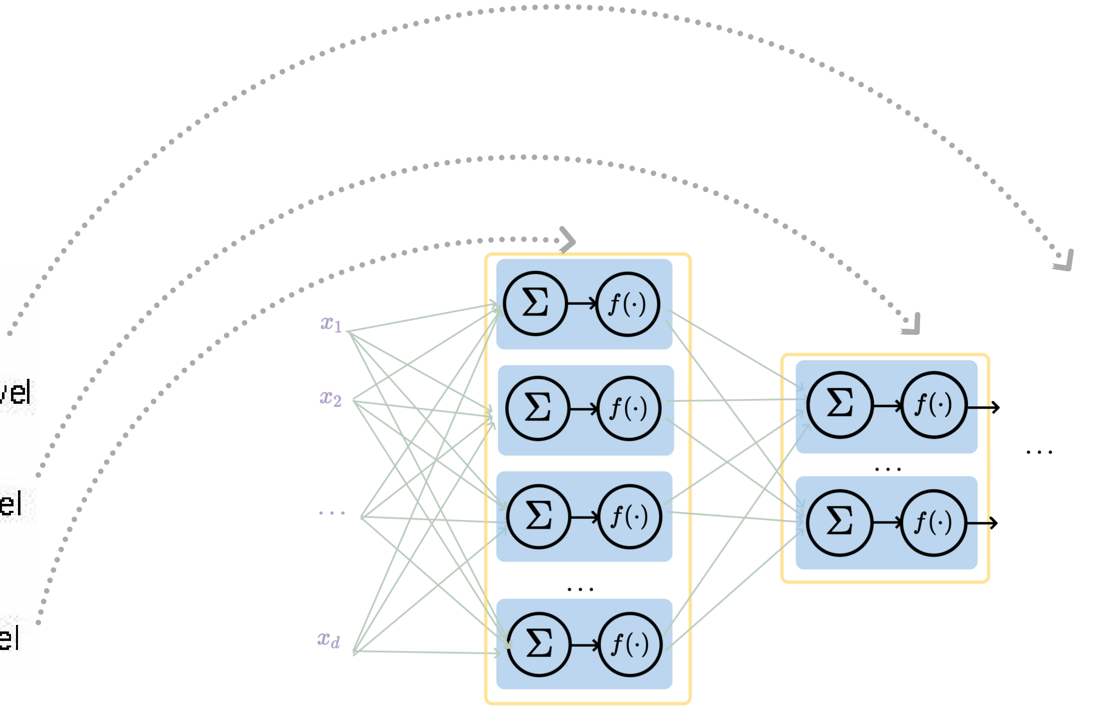

[video edited from 3b1b]



[video edited from 3b1b]

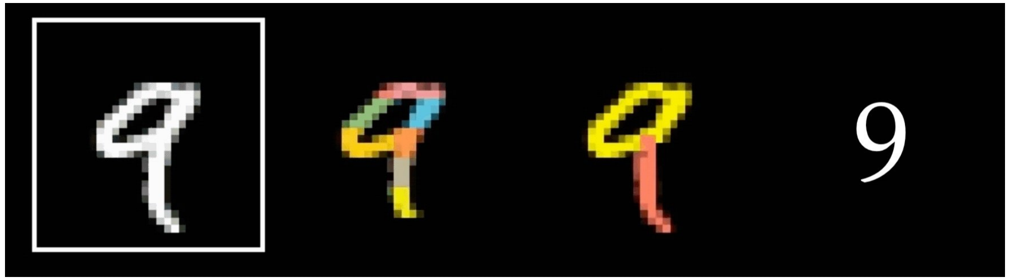
- Visual hierarchy



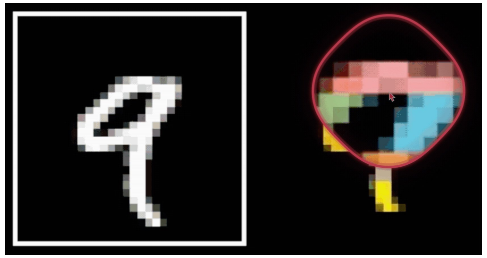


Layered structure are well-suited to model this hierarchical processing.

- Visual hierarchy



- Spatial locality



- Translational invariance

CNN cleverly exploits

- Visual hierarchy
- Spatial locality
- Translational invariance

via

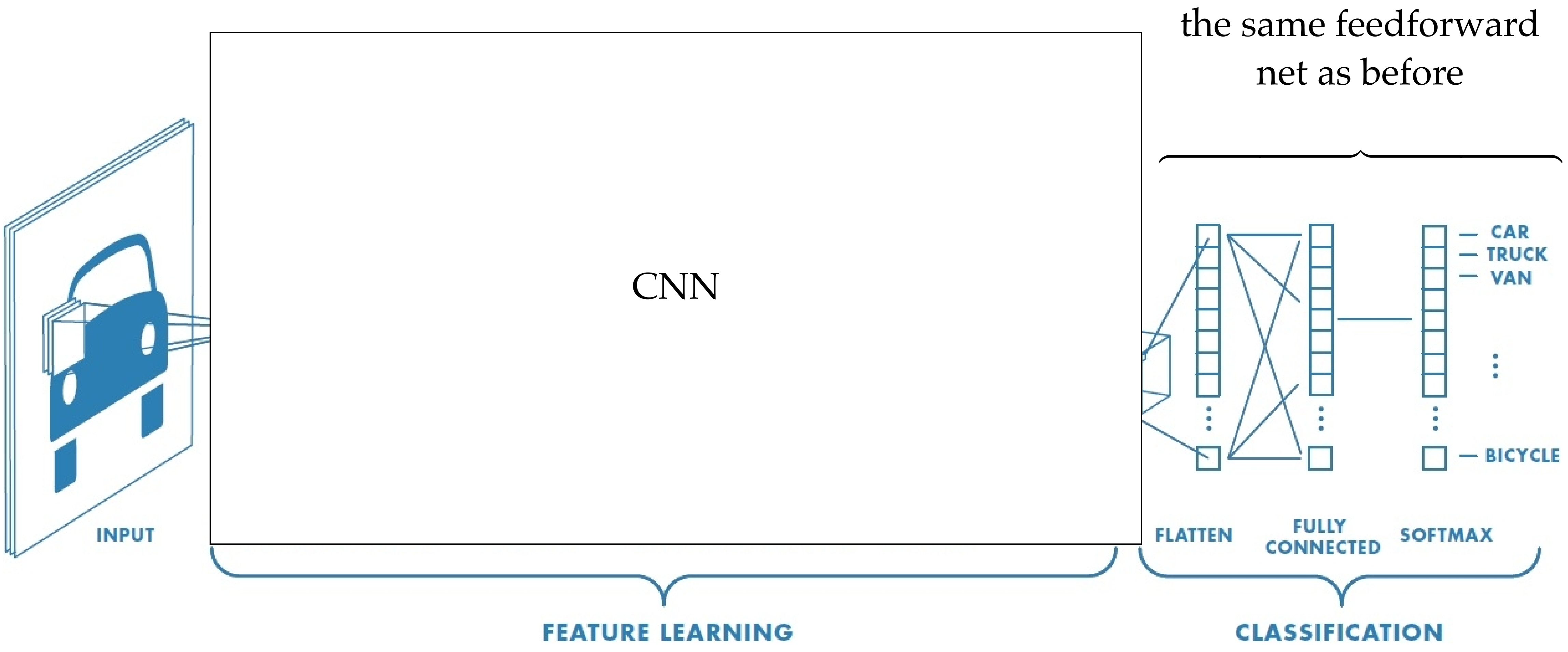
- Layered structure
- convolution
- pooling

to handle images efficiently and sensibly.

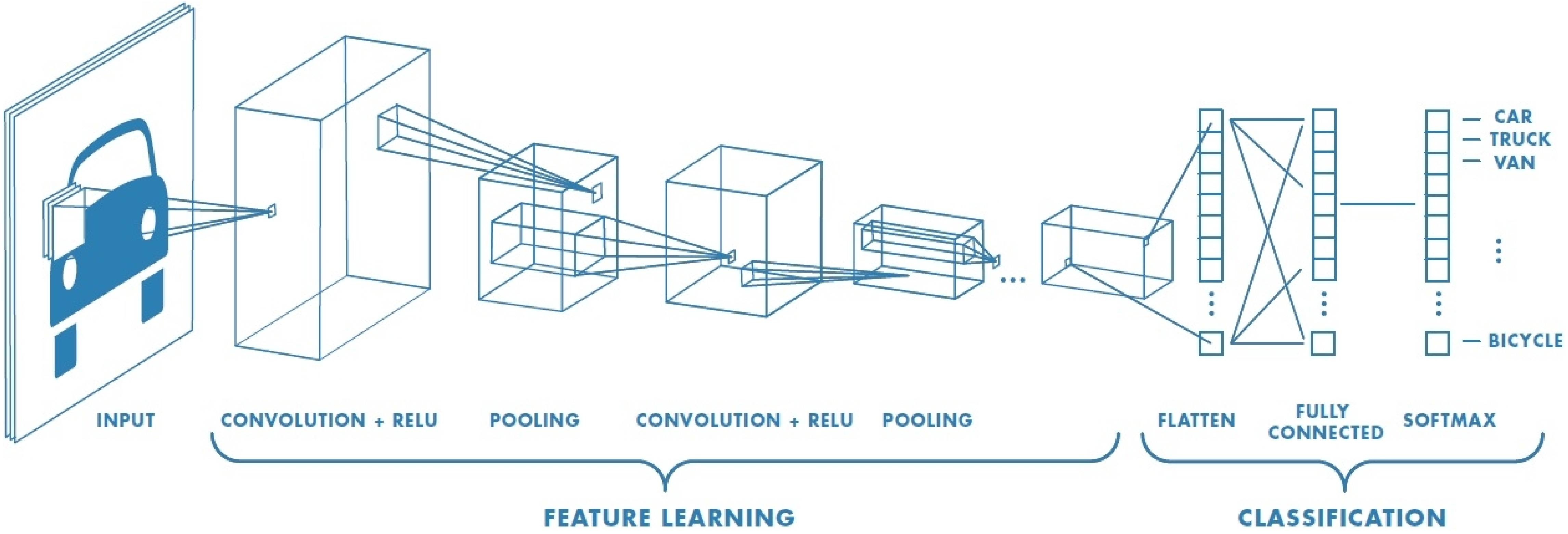
Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
  - 1-dimensional and 2-dimensional *convolution*
  - 3-dimensional *tensors*
- Max pooling
- (Case studies)

typical CNN structure for image classification



typical CNN structure for image classification



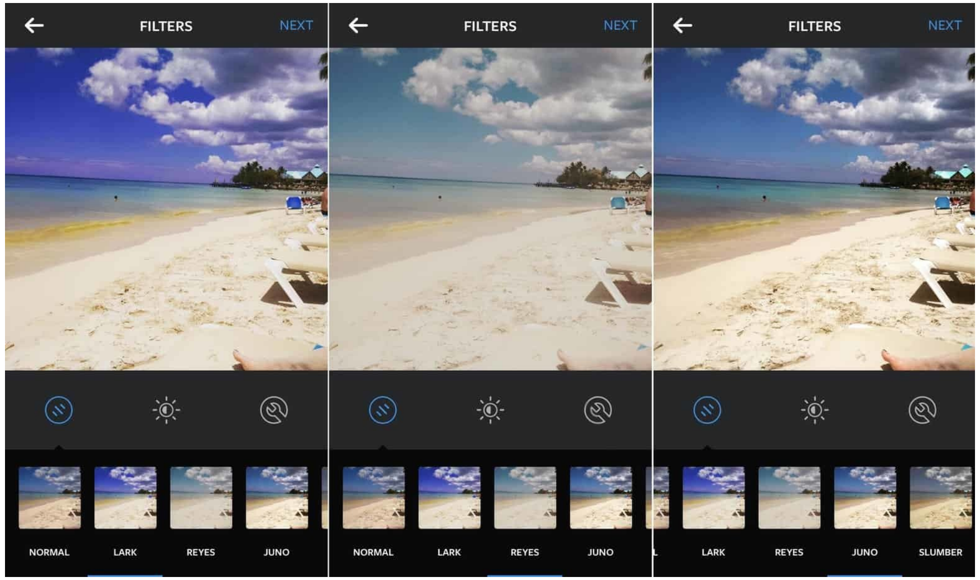
Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network 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 fully connected layer

Layer	Forward pass, *do*	Backward pass, *learn*
fully-connected	dot-product, activation	neuron weights
convolutional	convolution, activation	filter (kernel) weights

convolution with filters:



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: template matching

input	0	1	-1	1	1

filter	-1	1

convolved output	1	-2	2	0

convolution interpretation: "look" locally

input

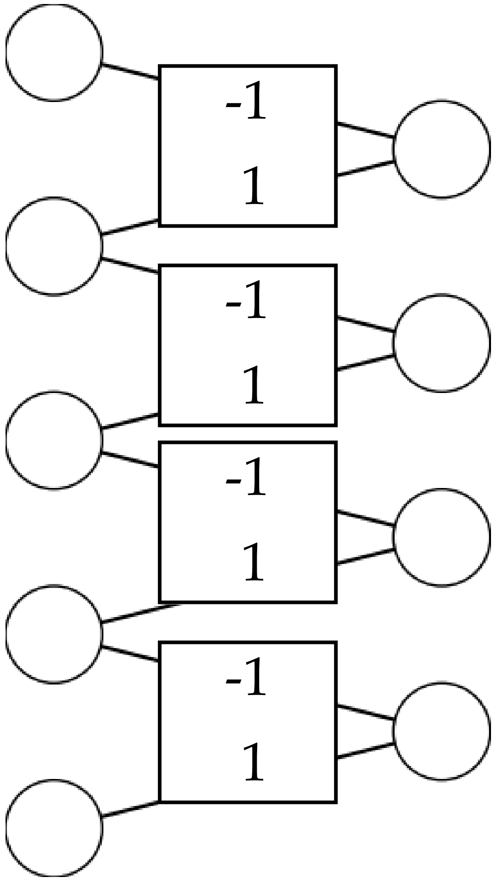
0	1	-1	1	1

filter

-1	1

convolved output

1	-2	2	0



convolution interpretation: parameter sharing

0	1	-1	1	1

convolve with

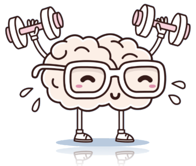
-1	1

dot product with

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

=

1	-2	2	0



0	1	0	1	1

↙ ↘

convolve with

dot product with

1

$$I_{5\times 5}$$

↘ ↙

0	1	0	1	1

convolution interpretation: translational equivariance

input	0	1	0	1	0

filter	0	1

convolved output	1	0	1	0

hyperparameters

- Zero-padding

0	0	1	0	1	1	0

- Stride (e.g. stride of 2)

0	0	1	0	1	1	0

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

-1	1

1

these weights are what CNN learn eventually

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

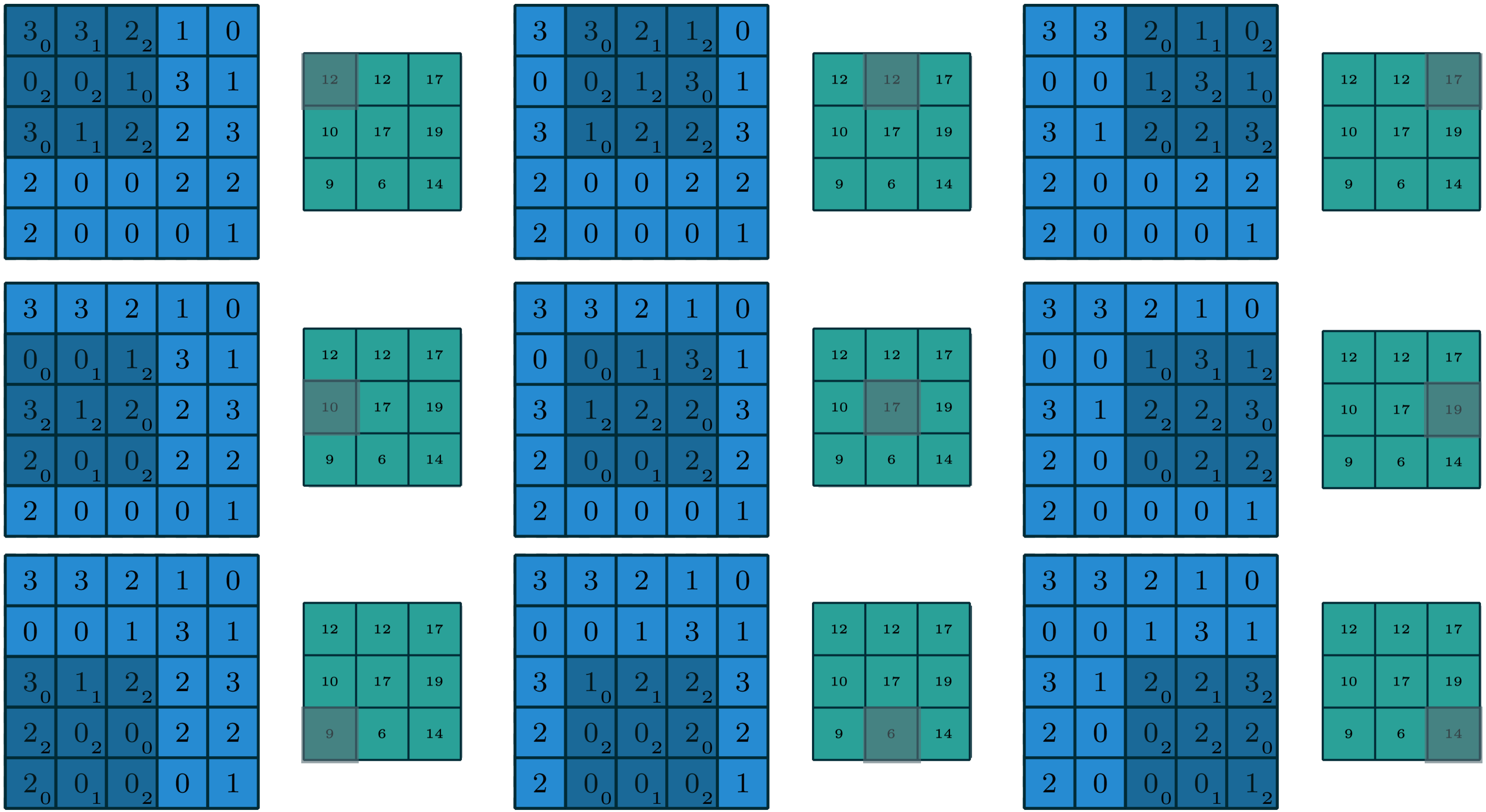
filter

0	1	2
2	2	0
0	1	2

output

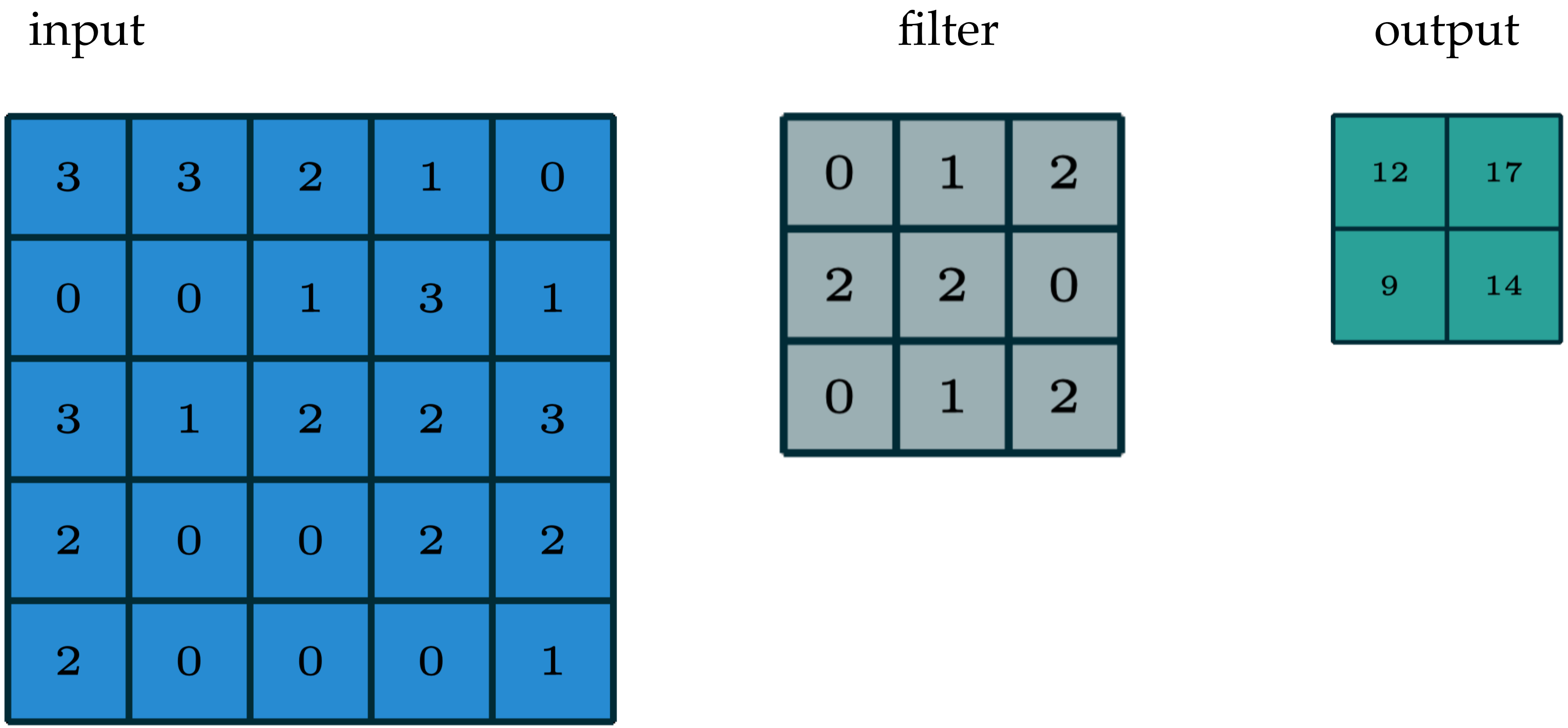
12	12	17
10	17	19
9	6	14

[image edited from vdumoulin]



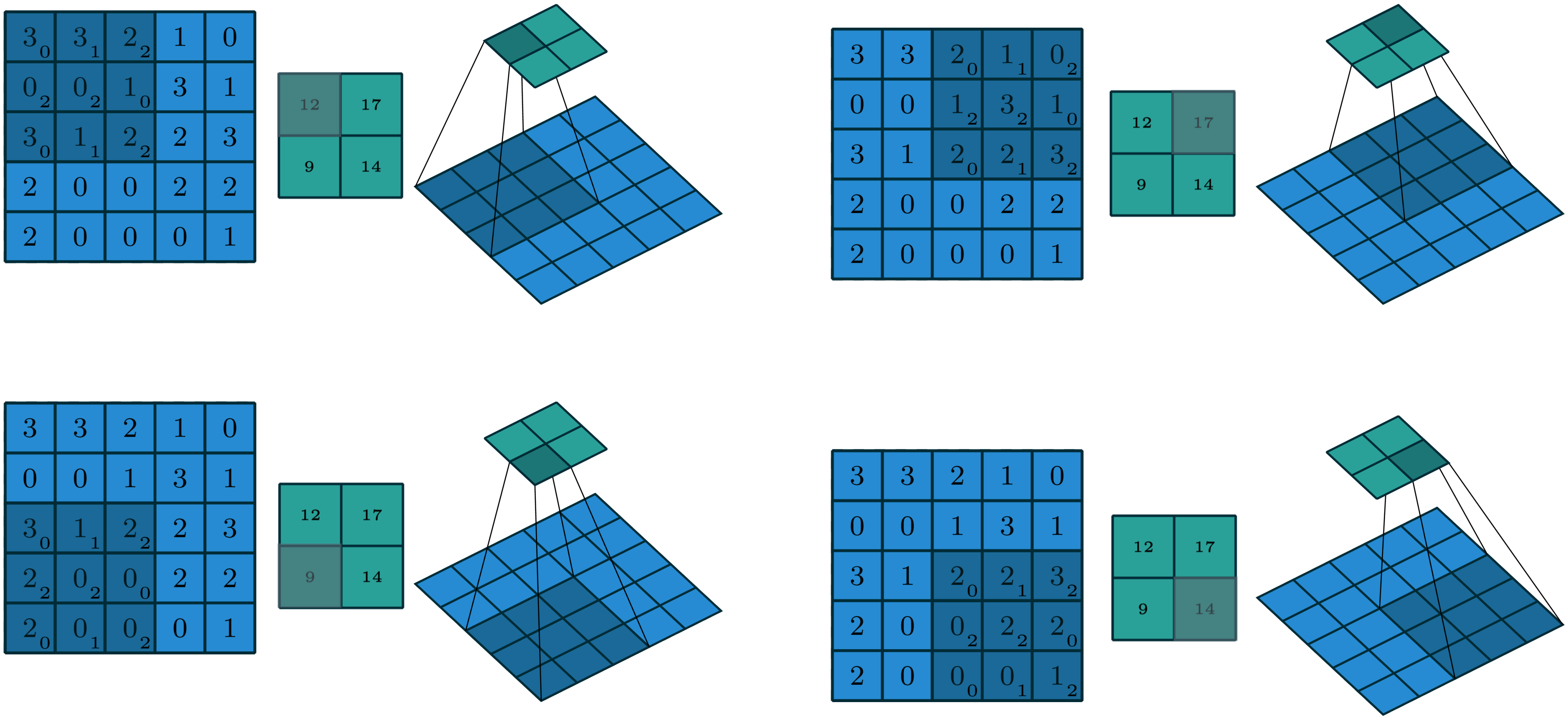
[image edited from vdumoulin]

stride of 2



[image edited from vdumoulin]

stride of 2



[image edited from vdumoulin]

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

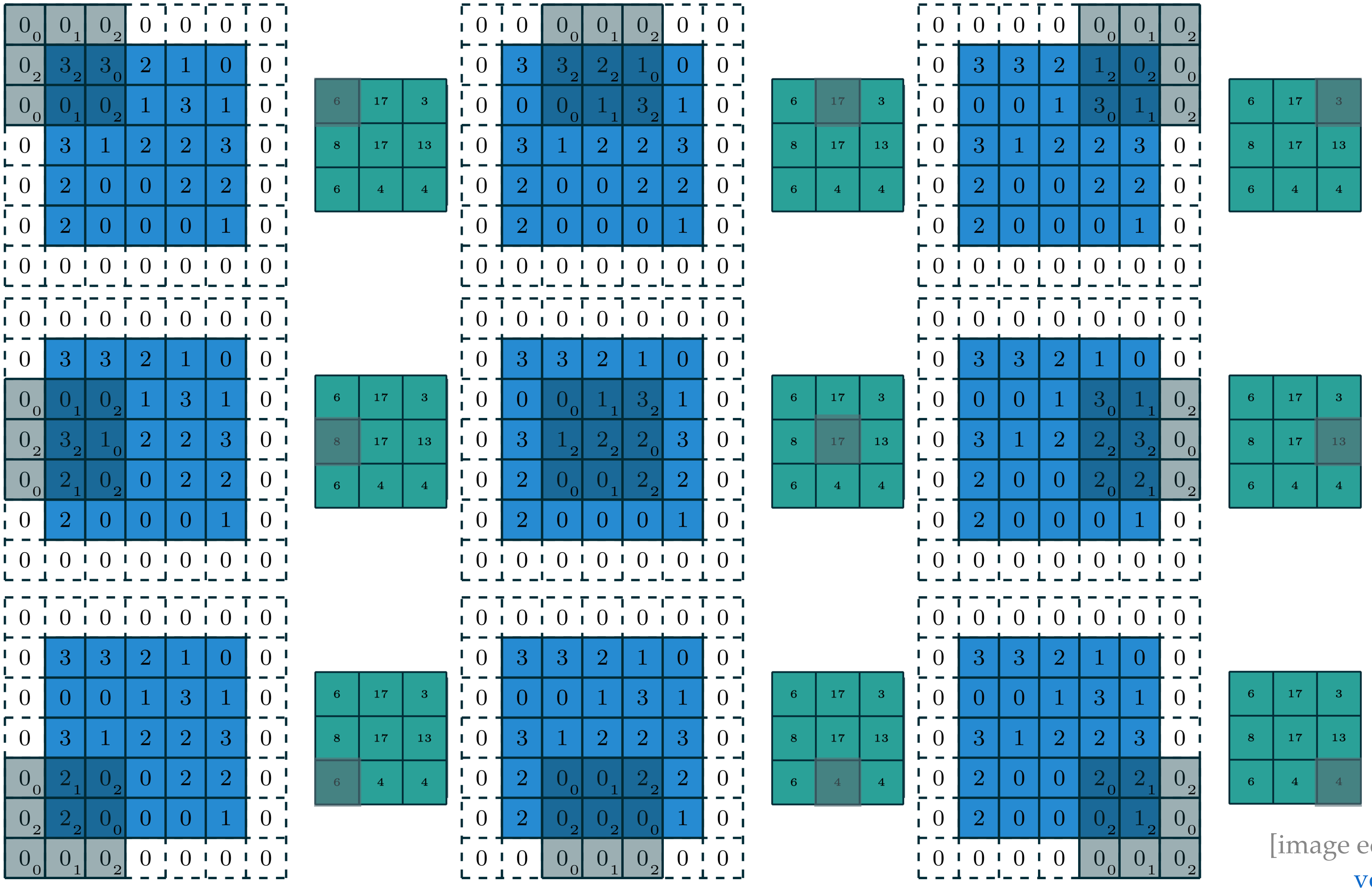
filter

0	1	2
2	2	0
0	1	2

output

6	17	3
8	17	13
6	4	4

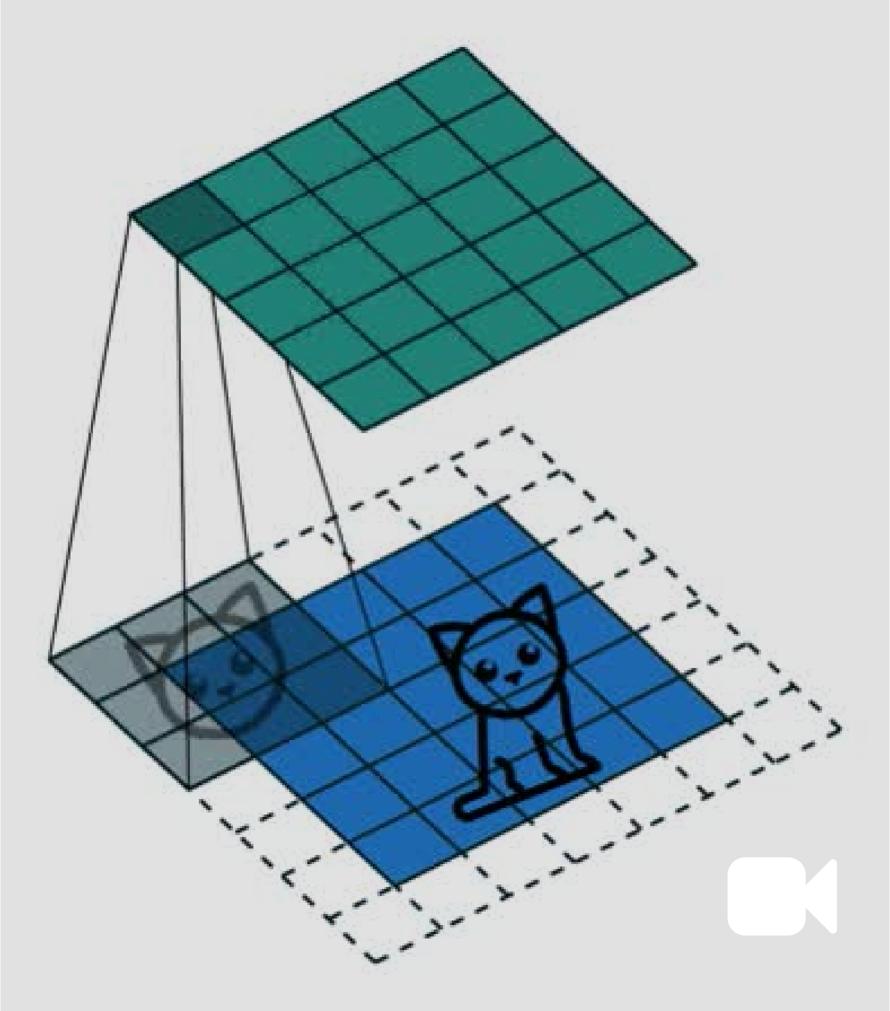
[image edited from vdumoulin]



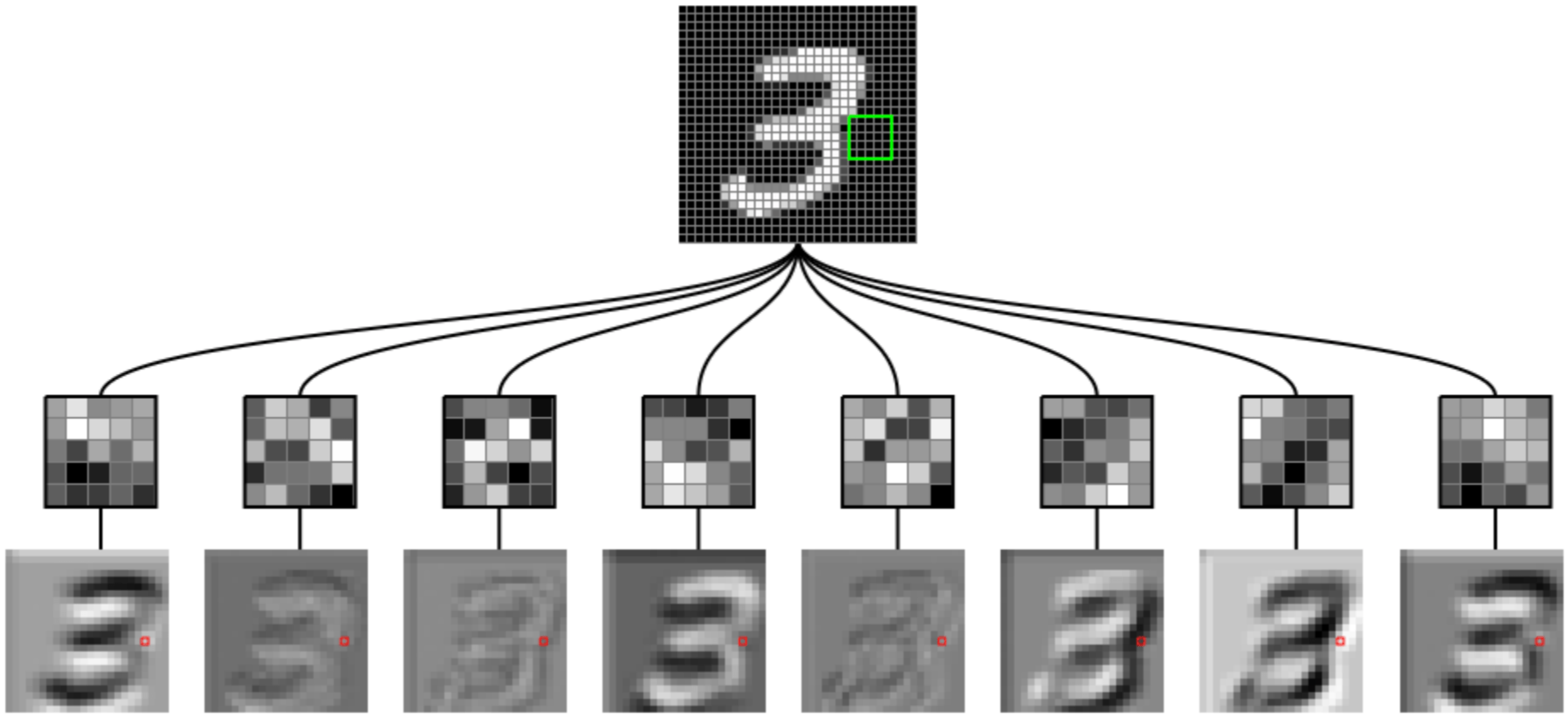
[image edited from vdumoulin]

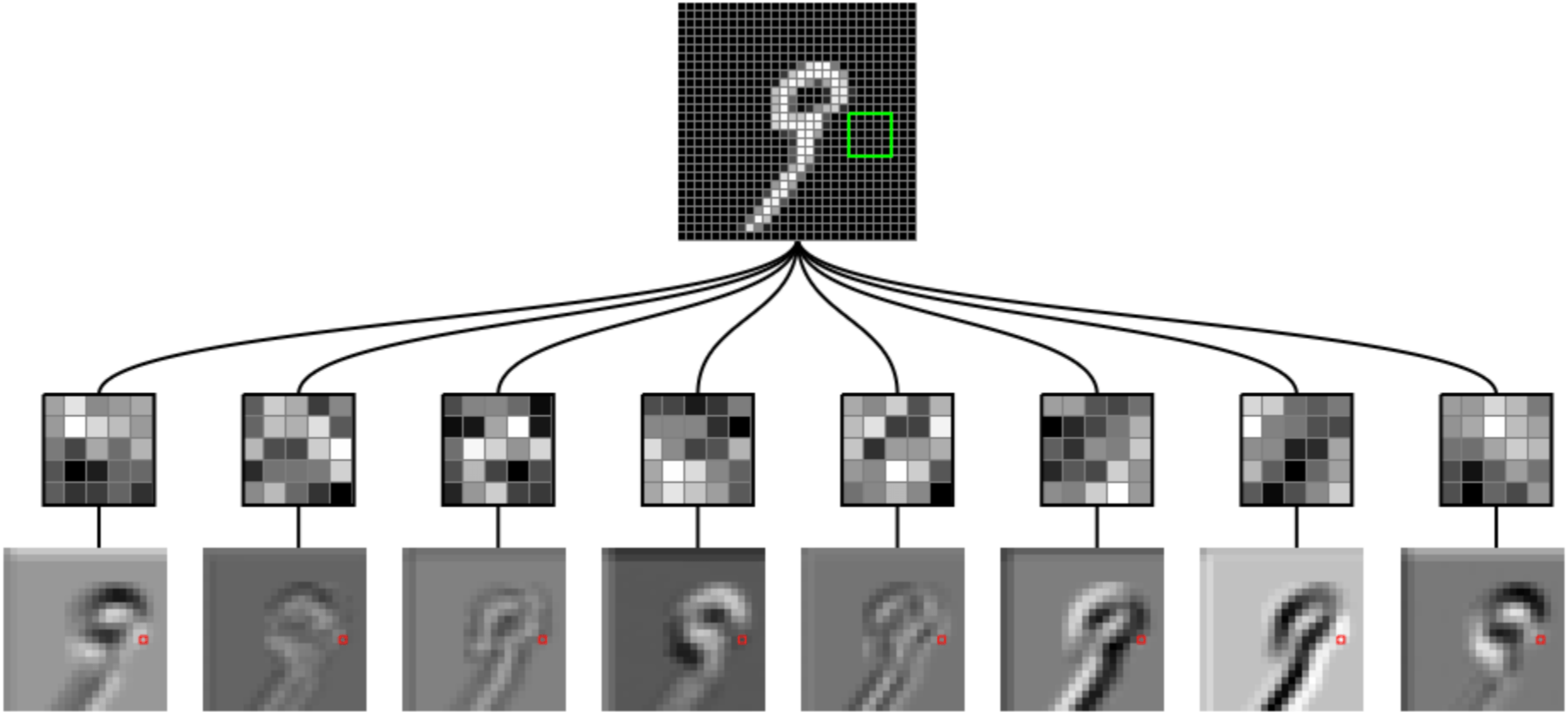
convolution interpretation:

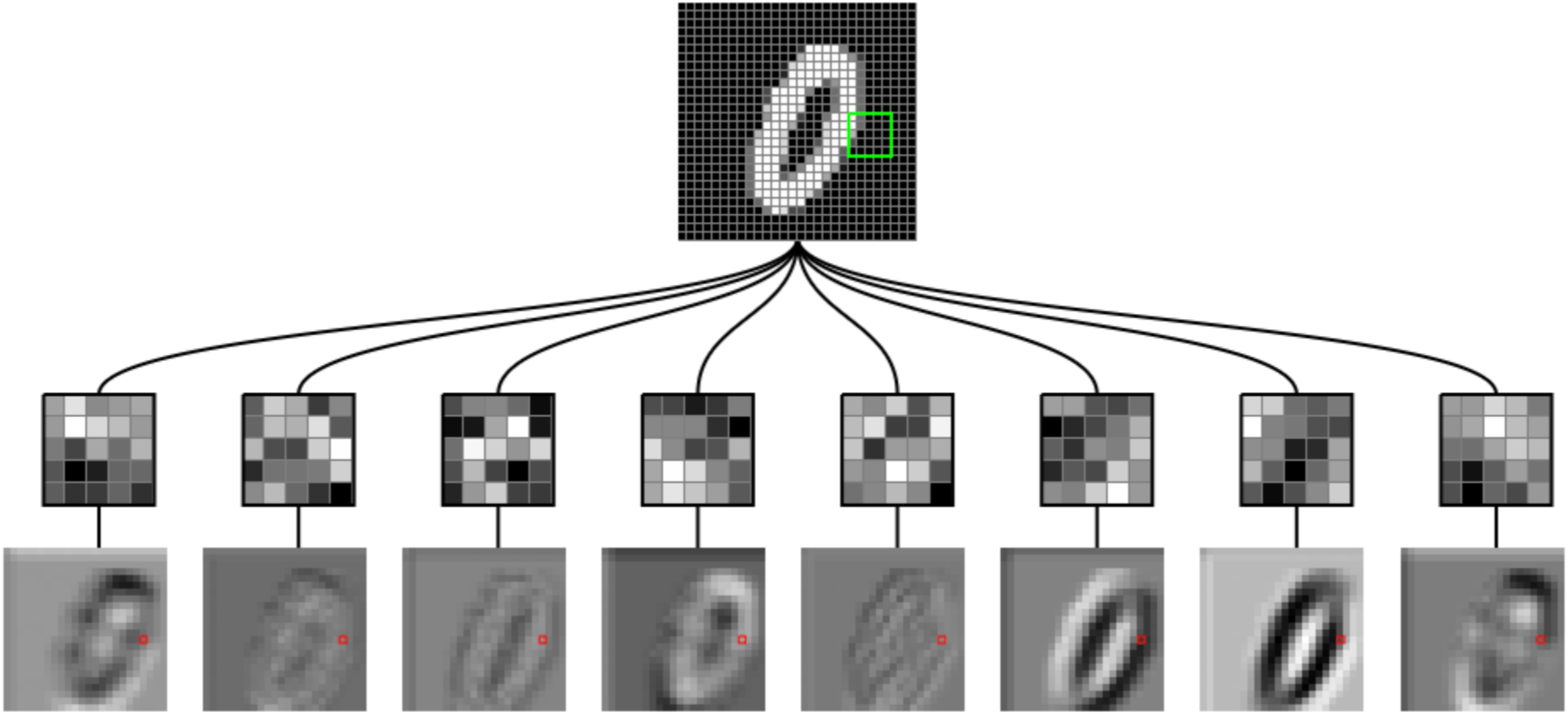
- Looking locally
- Parameter sharing
- Template matching
- Translational equivariance



[video credit Lena Voita]



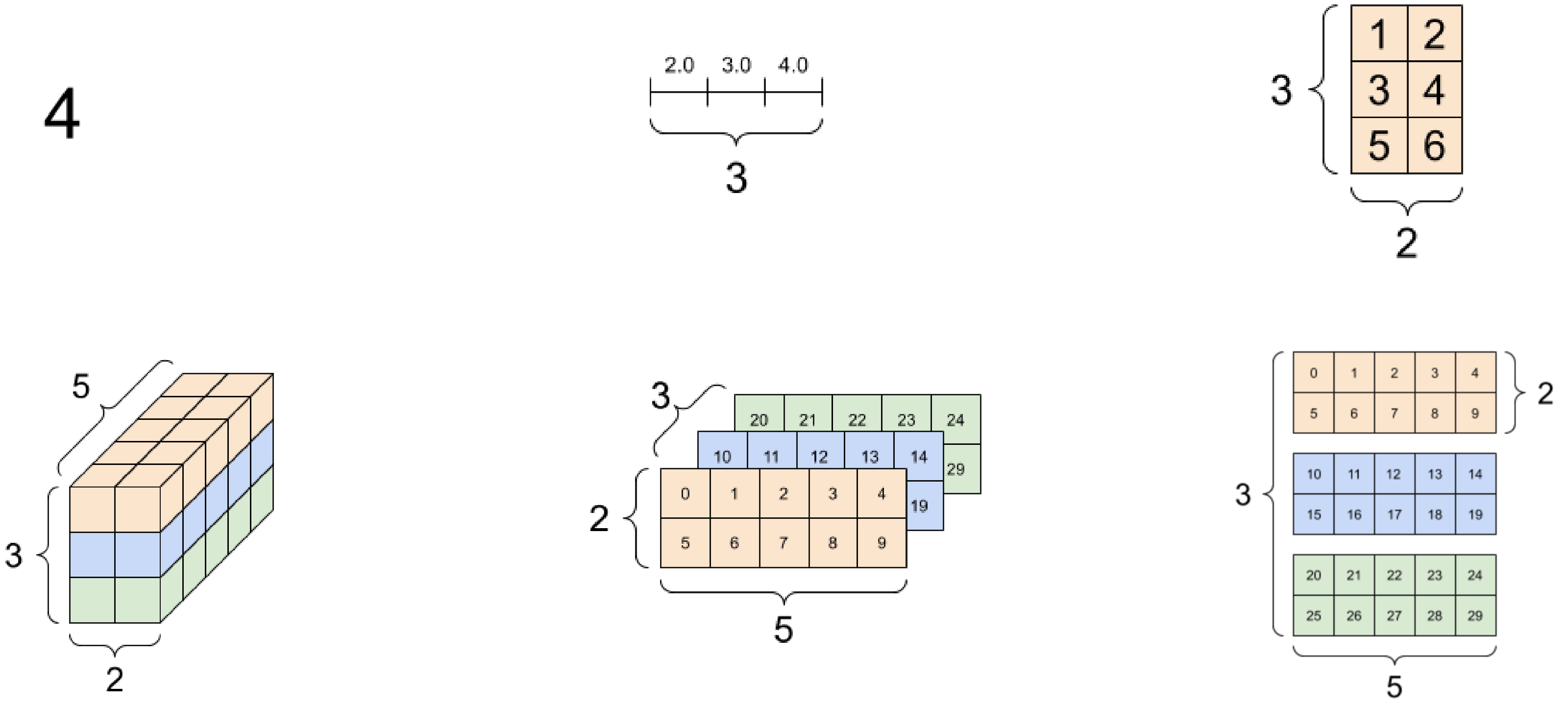




Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
  - 1-dimensional and 2-dimensional *convolution*
  - 3-dimensional *tensors*
- Max pooling
- (Case studies)

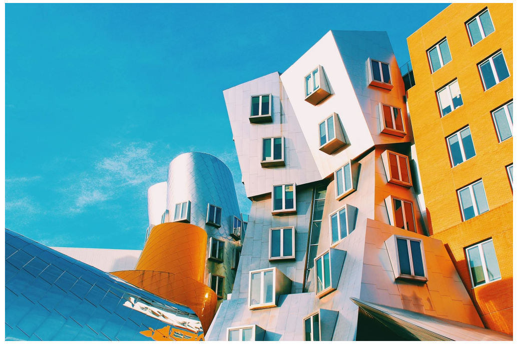
A tender intro to tensor:



[image credit: tensorflow]

We'd encounter 3d tensor due to:

1. color input

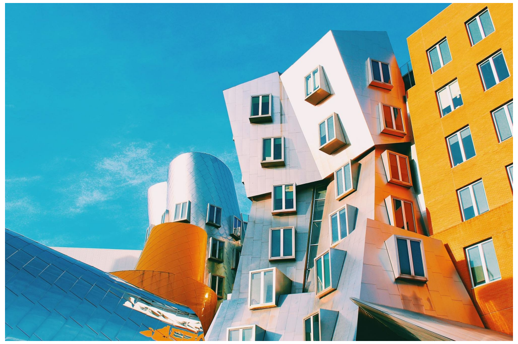


blue

green

red

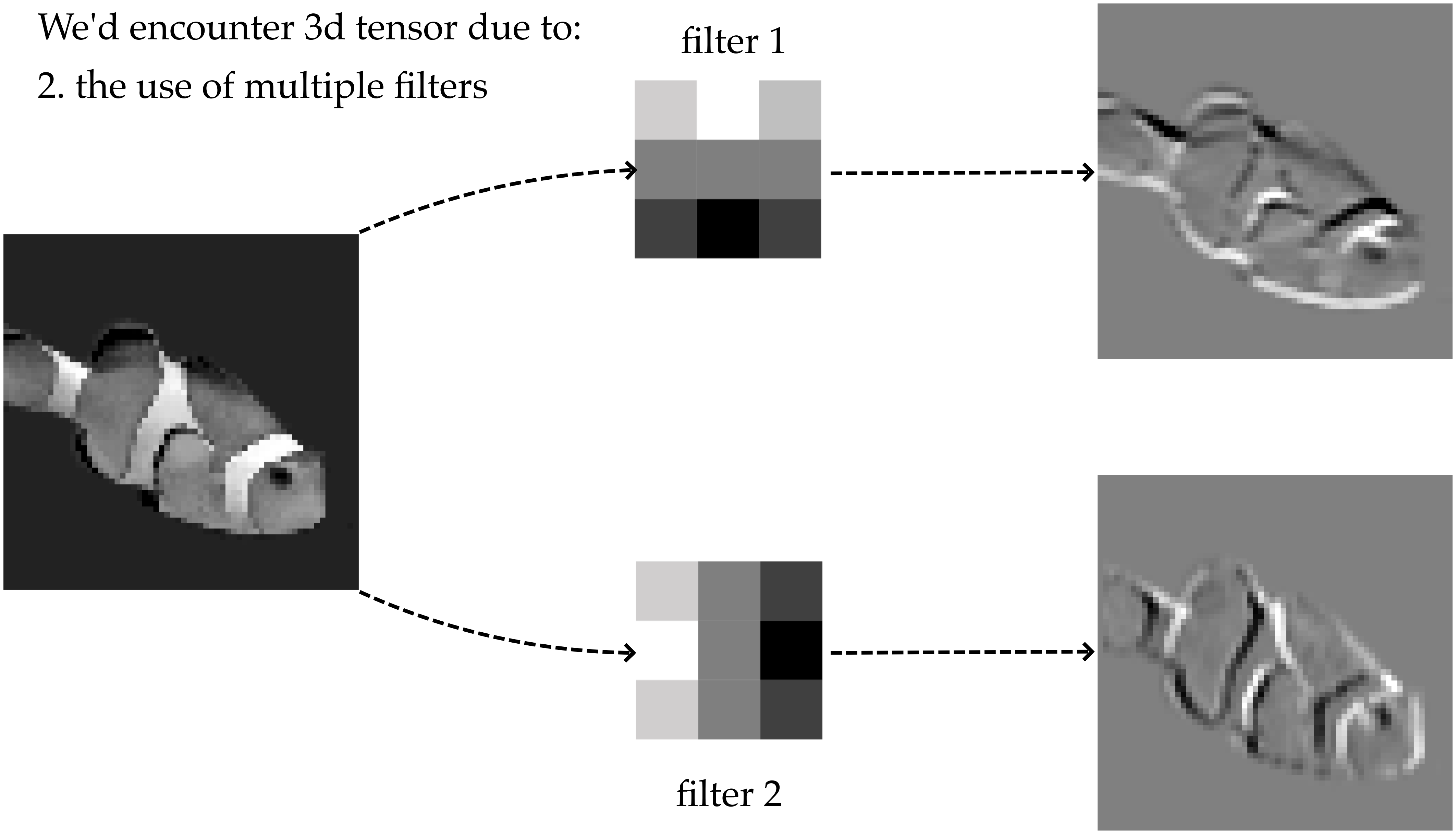
[Photo by Zayn Shah, Unsplash]



[Photo by Zayn Shah, Unsplash]

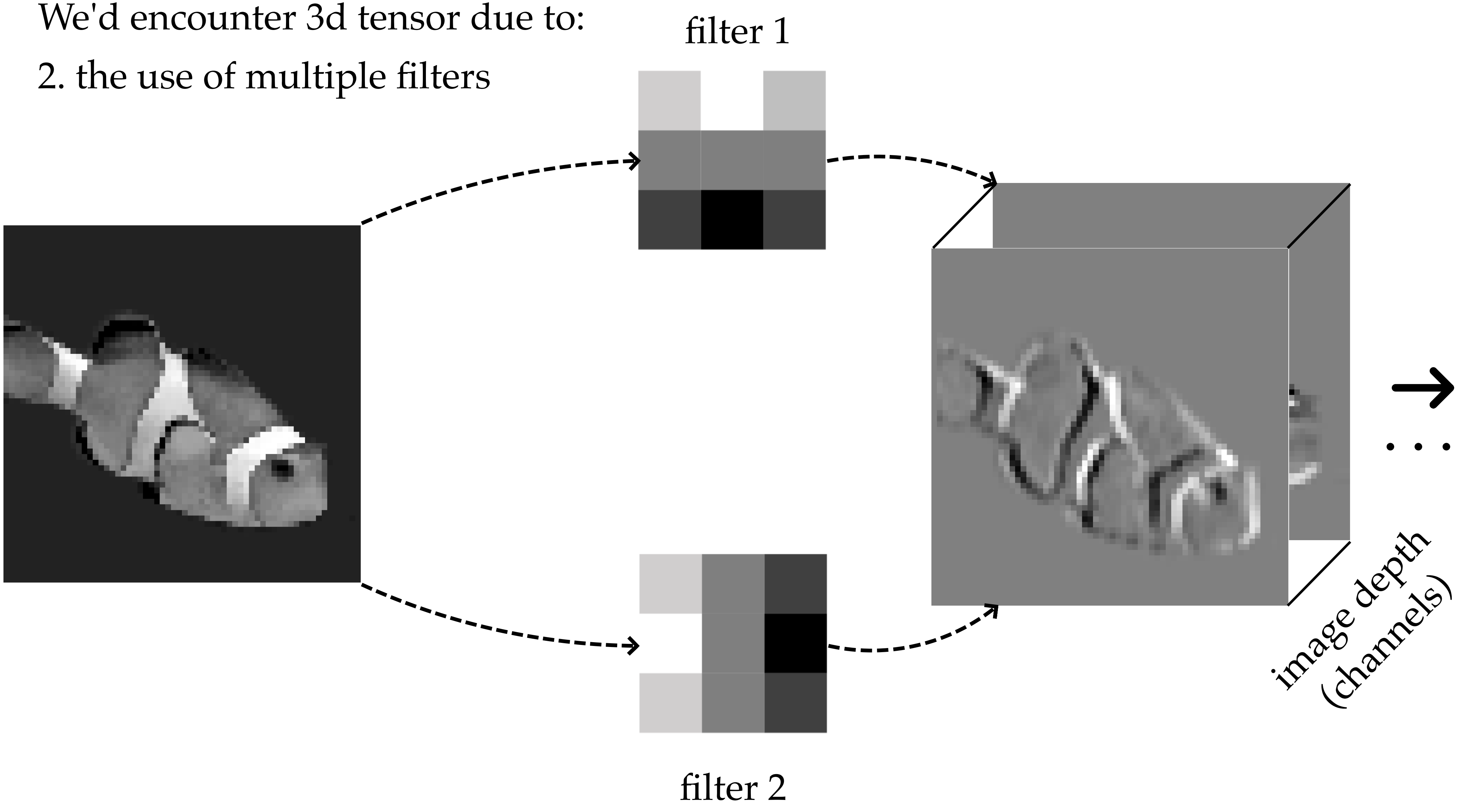
We'd encounter 3d tensor due to:

2. the use of multiple filters



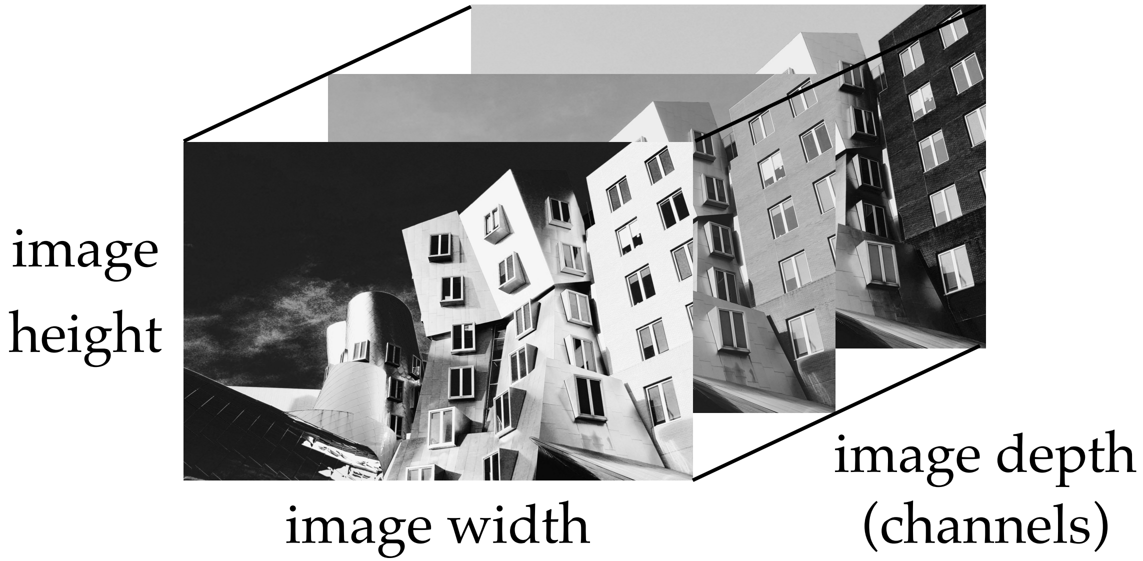
We'd encounter 3d tensor due to:

2. the use of multiple filters

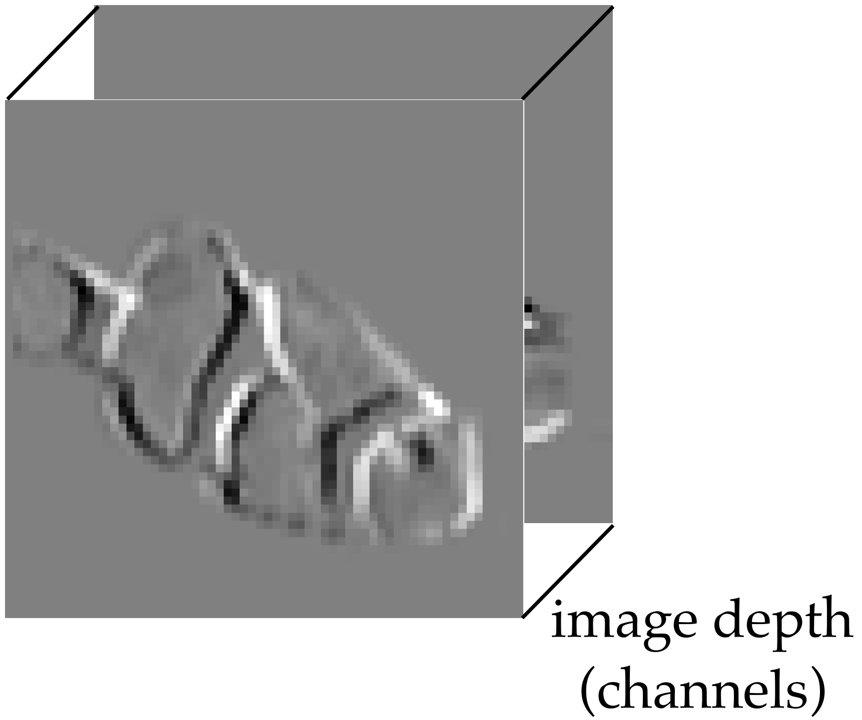


We'd encounter 3d tensor due to

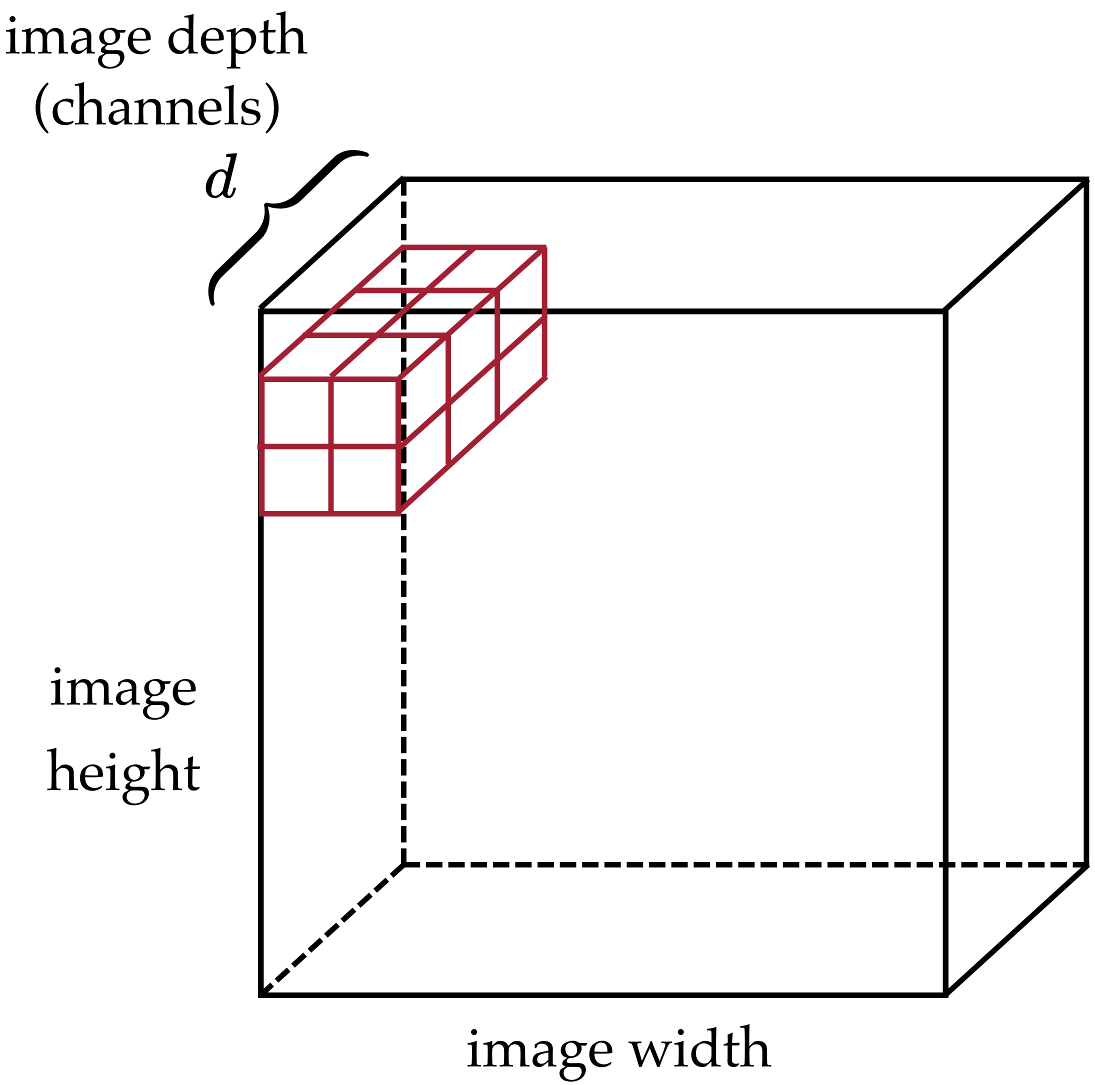
1. color input



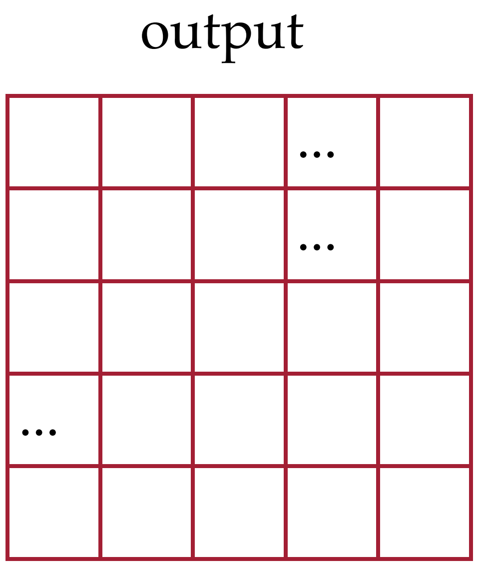
2. the use of multiple filters



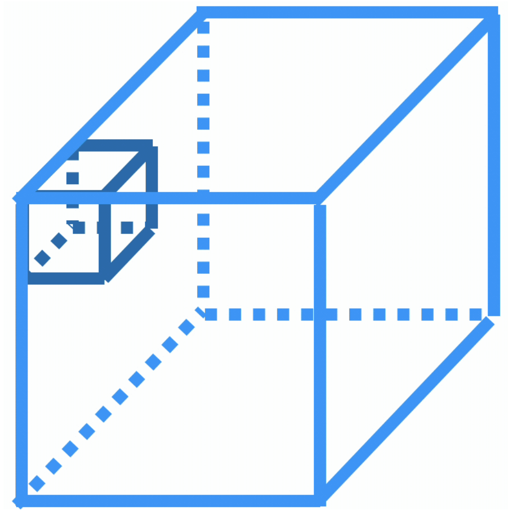
But, we *don't* typically do 3-dimensional convolution (in 6.390). Instead:

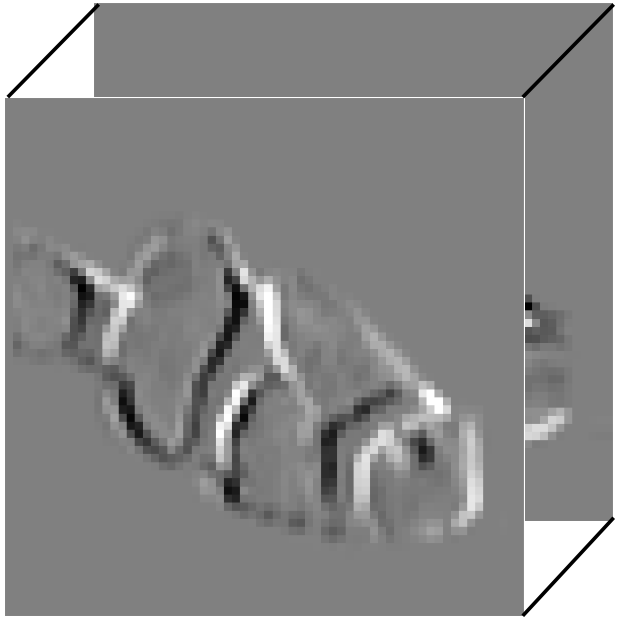


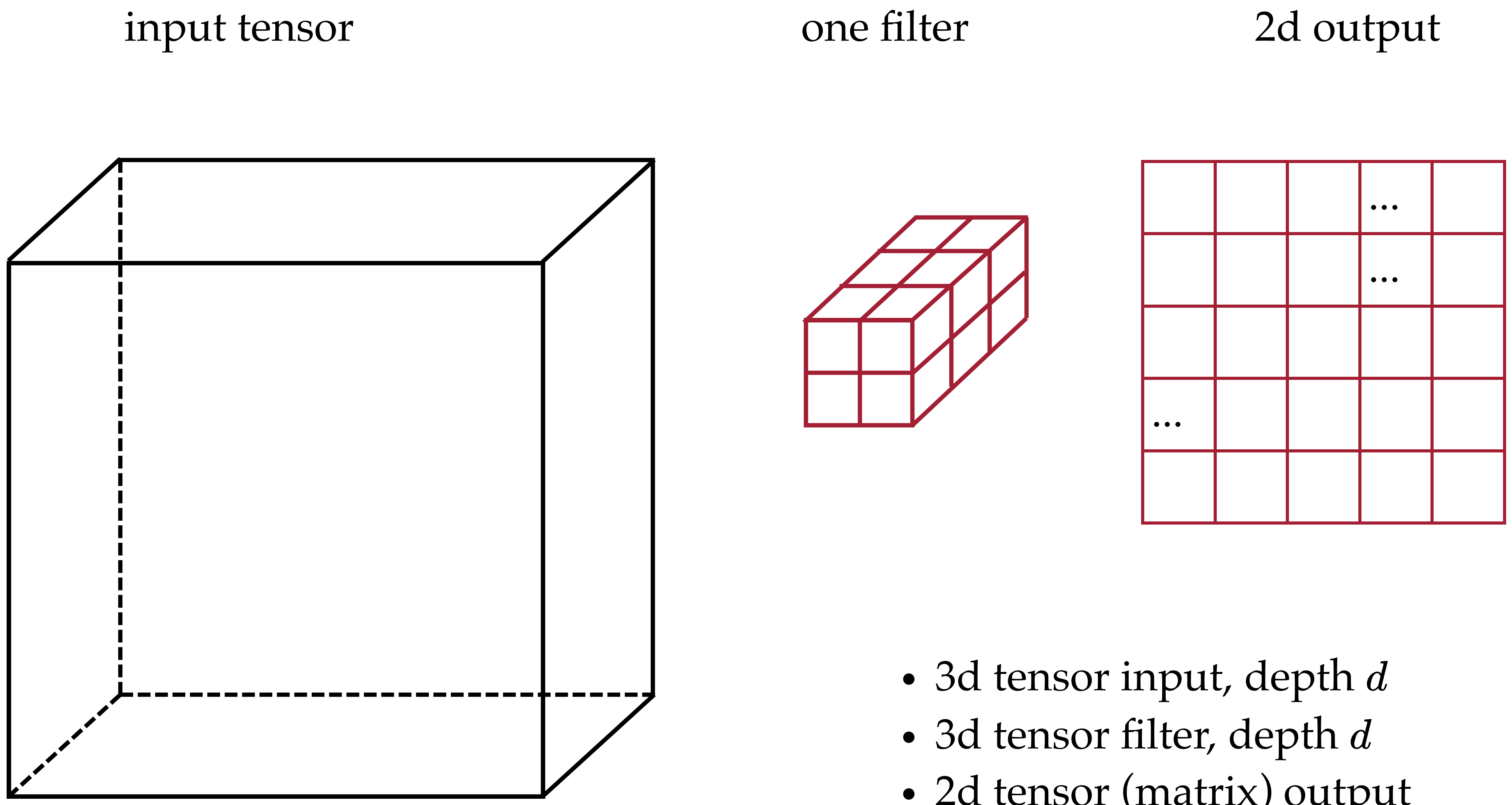
- 3d tensor input, depth d
- 3d tensor filter, depth d
- 2d convolution, 2d output



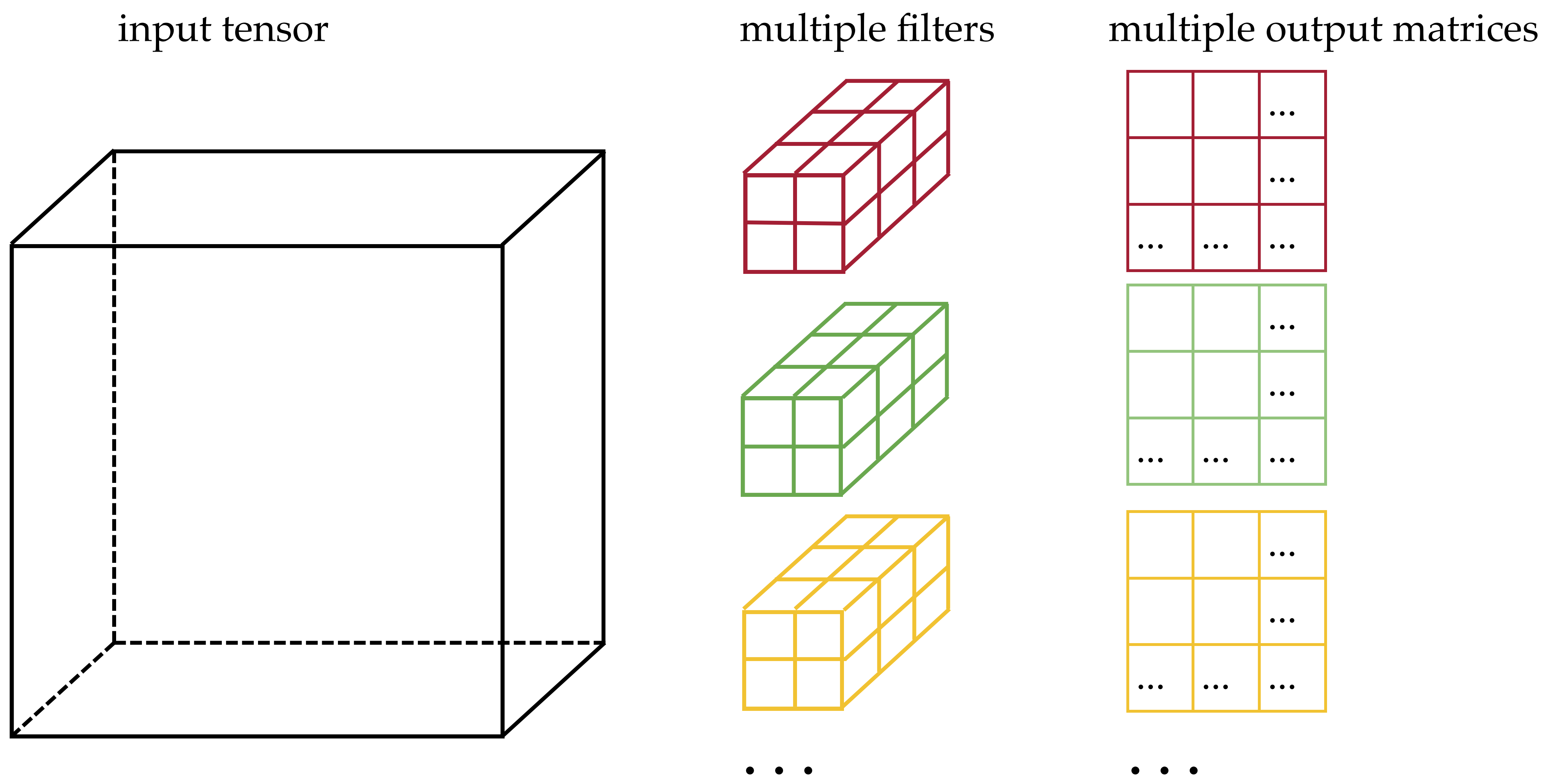
We *don't* typically do 3-dimensional convolution (in 6.390). Because:

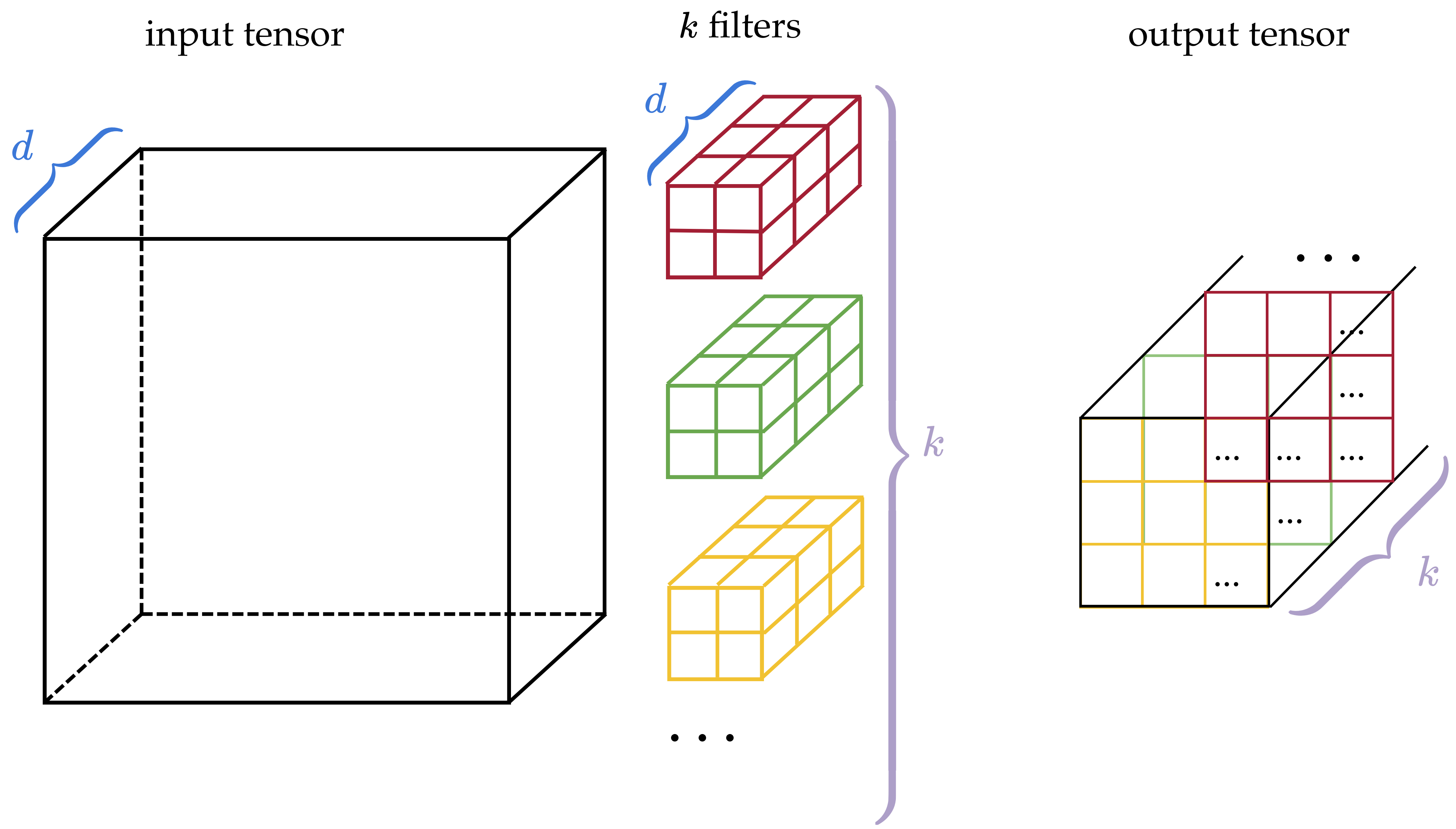






- 3d tensor input, depth d
- 3d tensor filter, depth d
- 2d tensor (matrix) output

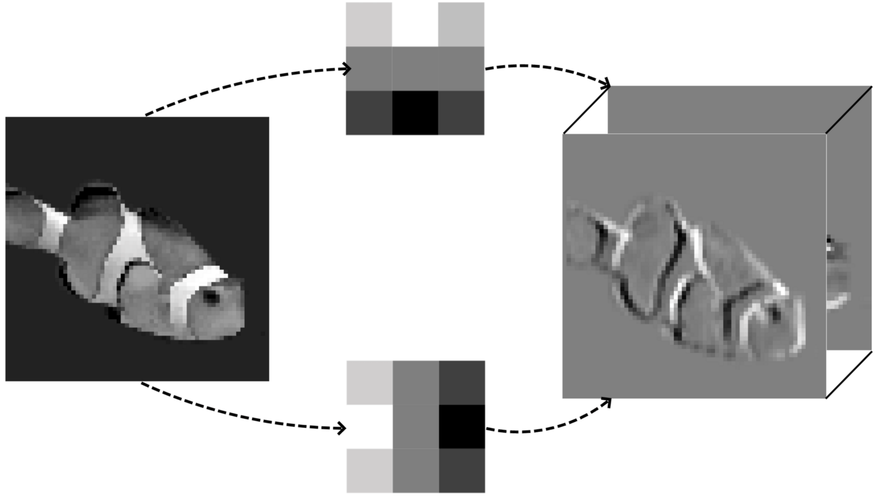

input tensor
multiple filters
multiple output matrices
...
...
... ...
...
...
... ...
...
...
... ...
. . .
. . .

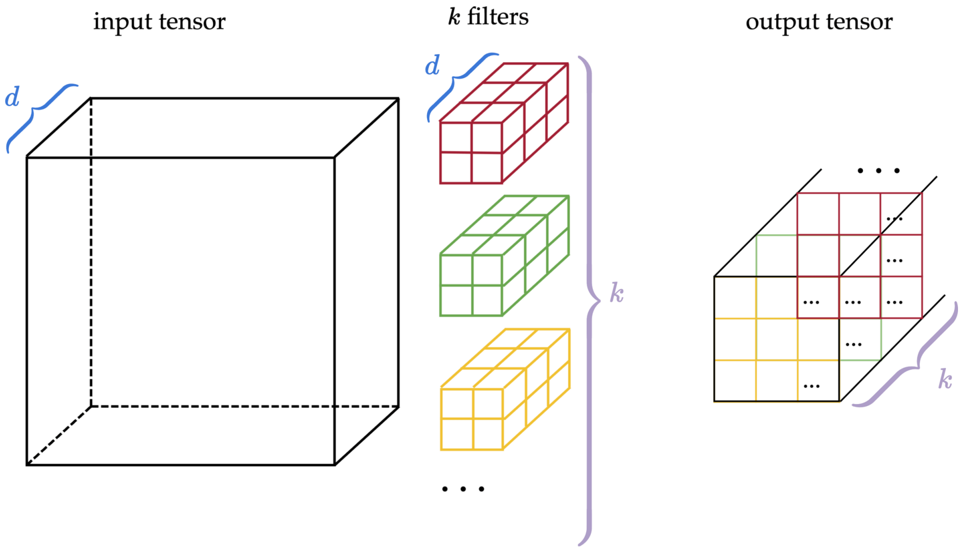

input tensor
k filters
output tensor
d
d
k
k

We'd encounter 3d tensor due to:

1. color input

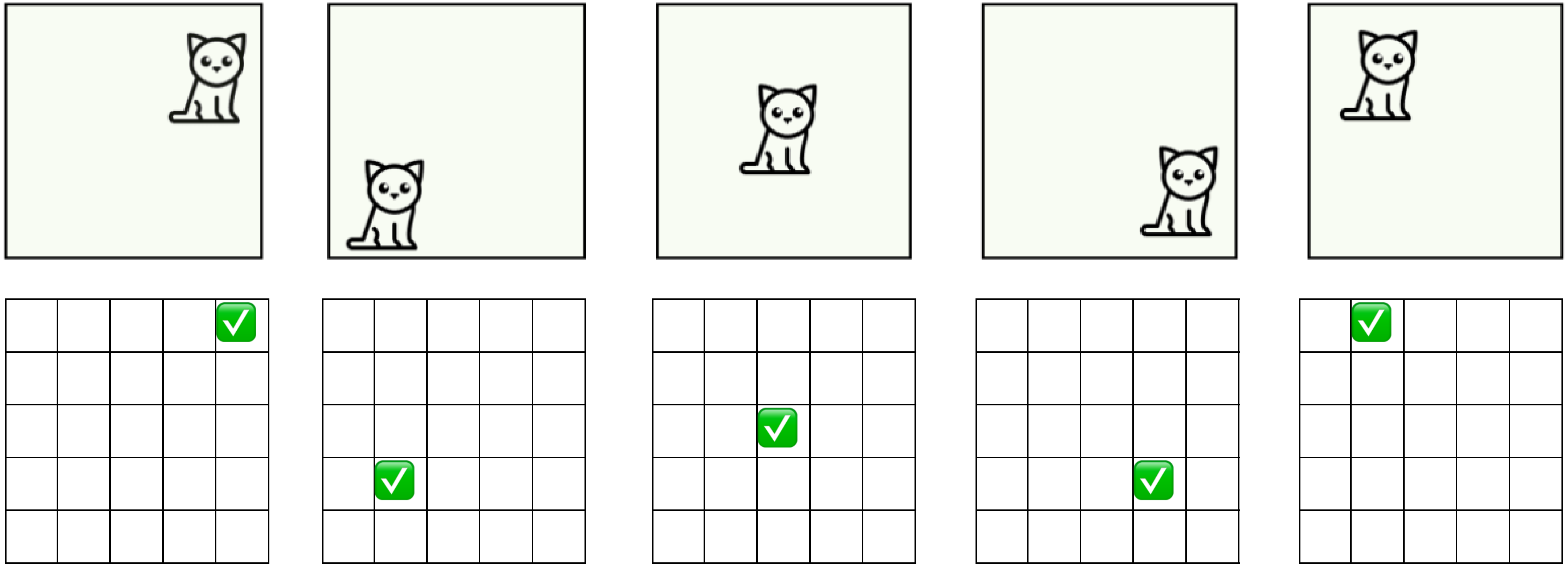
2. the use of multiple filters -- in doing 2-dimensional convolution



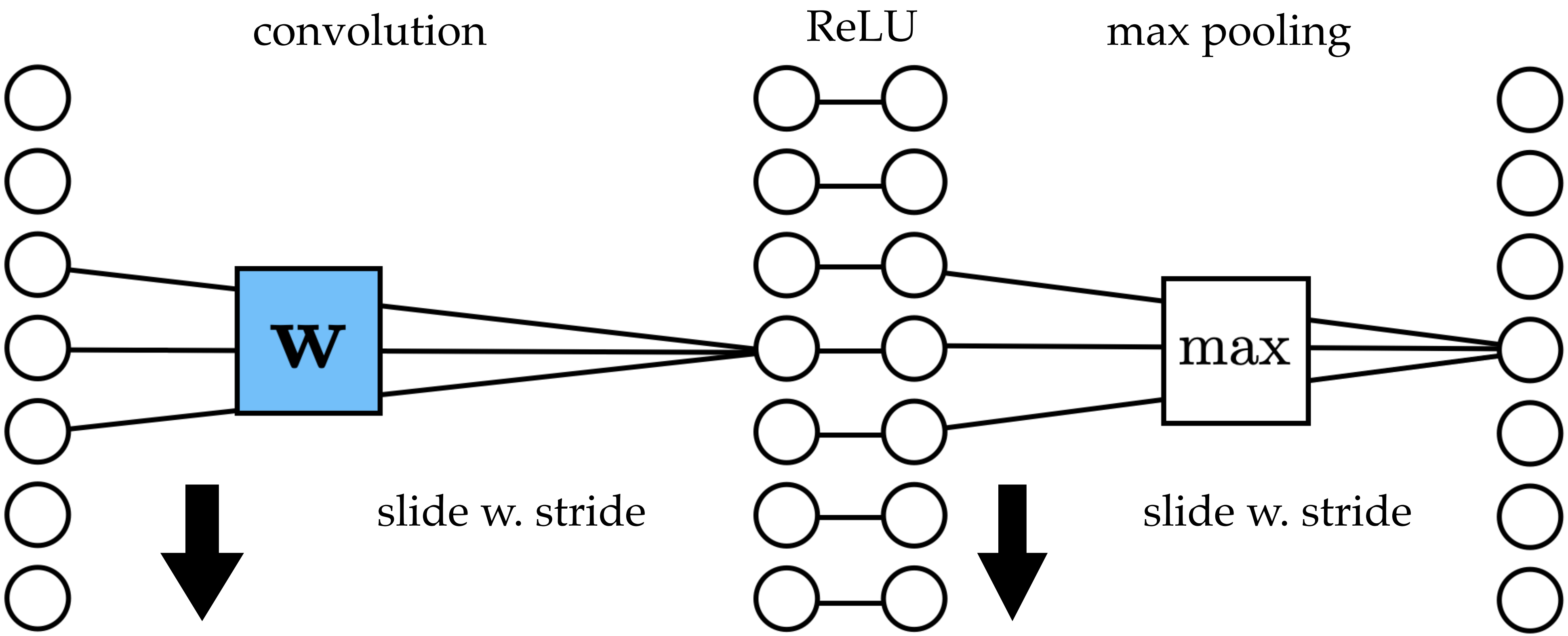


Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
  - 1-dimensional and 2-dimensional *convolution*
  - 3-dimensional *tensors*
- Max pooling
- (Case studies)



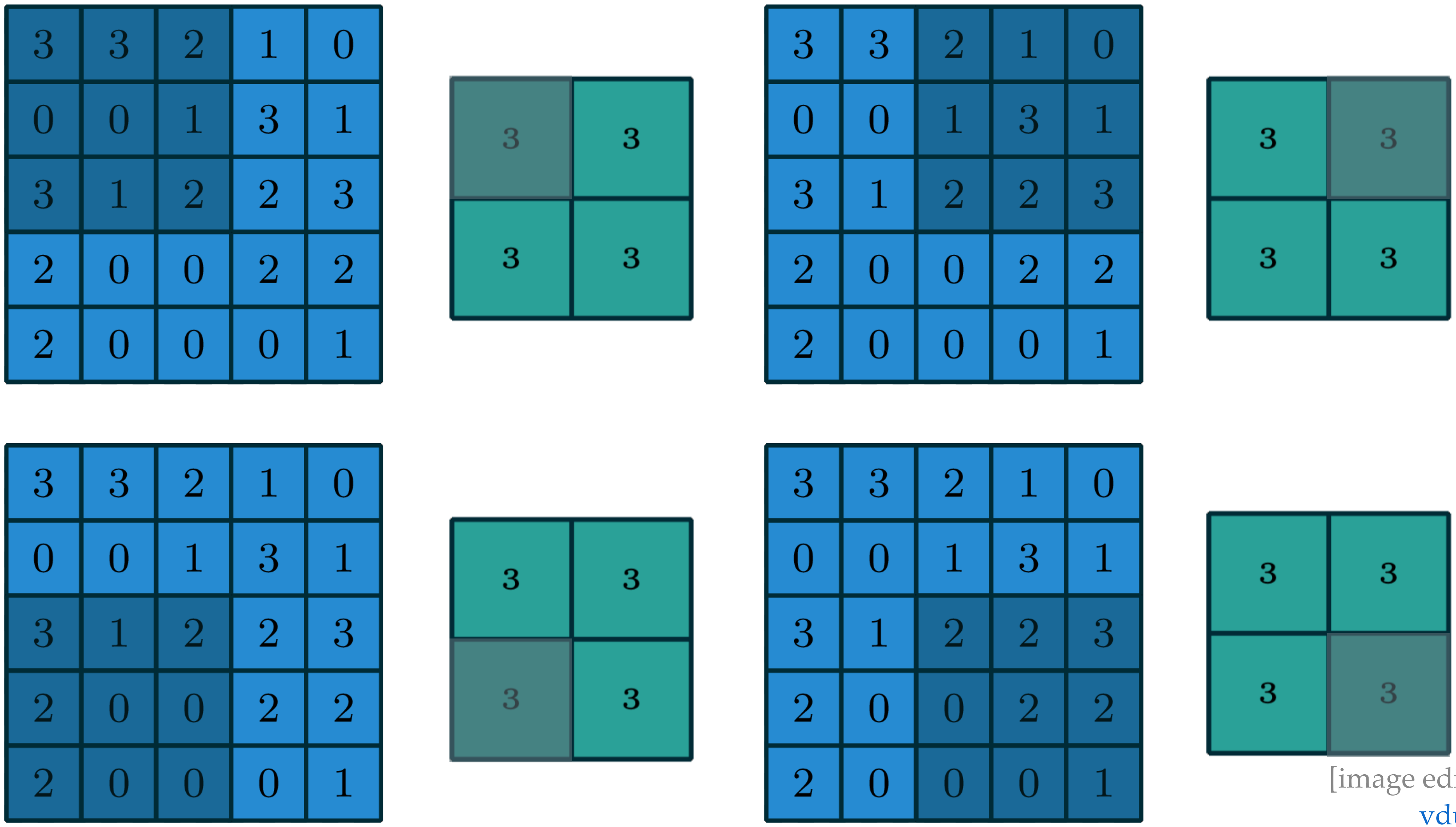
1-dimensional pooling



filter weights are the learnable parameter

no learnable parameter

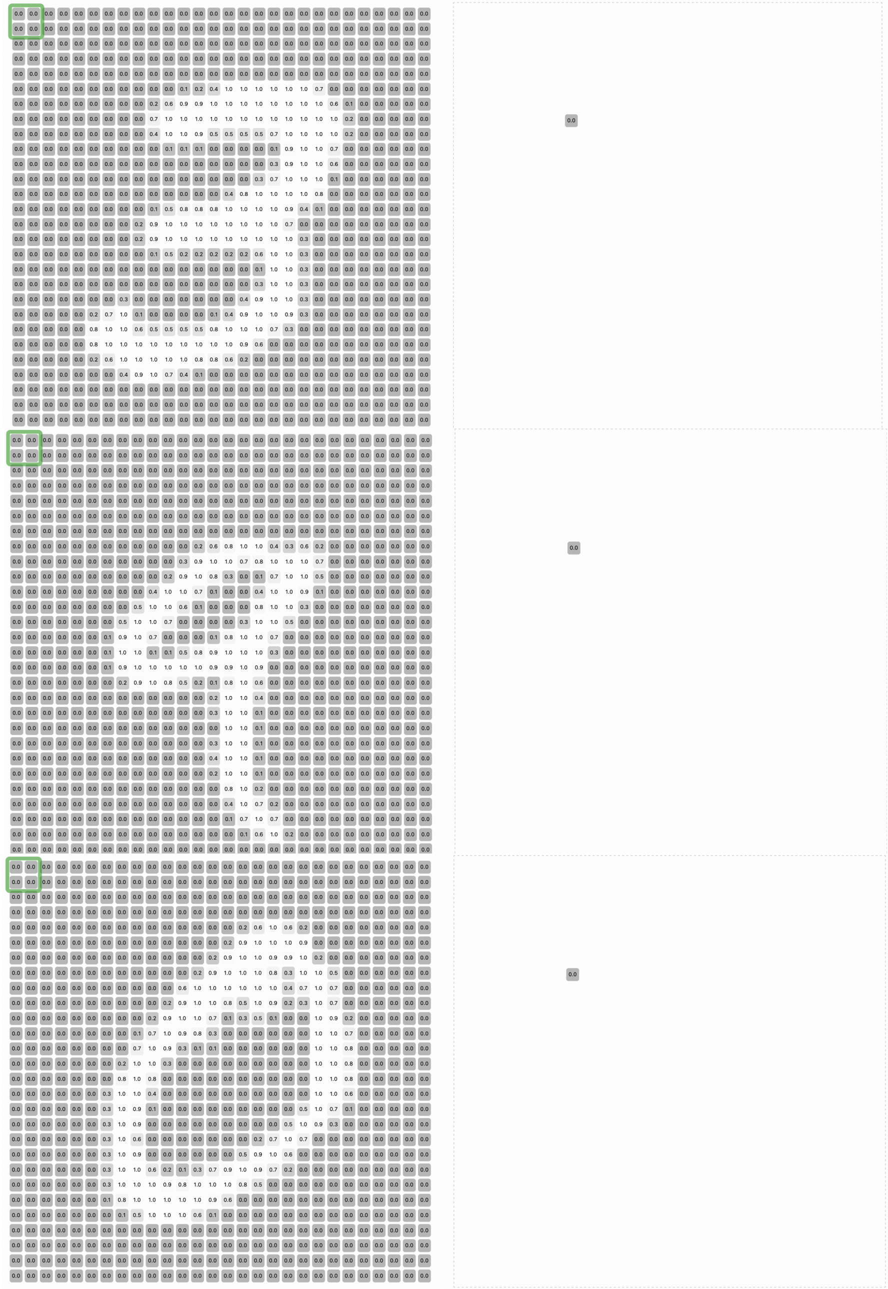
2-dimensional max pooling (example)



[image edited from vdumoulin]

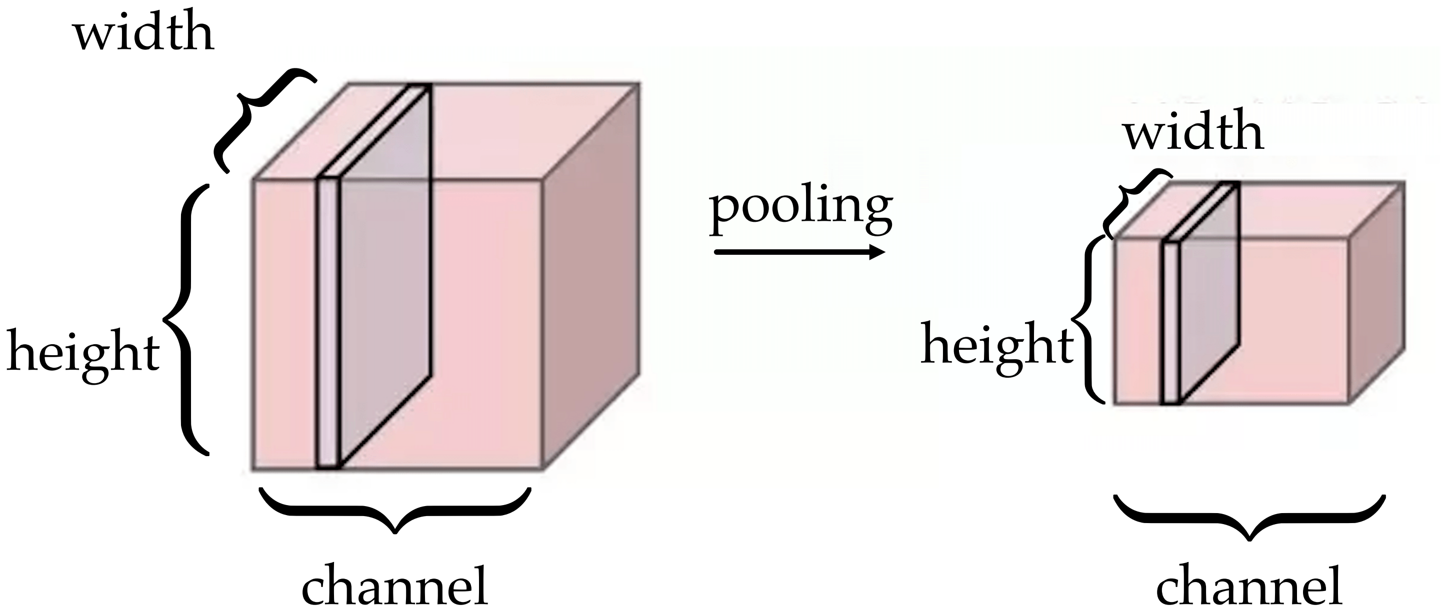
2-dimensional max pooling (example)

- can choose filter size
- typically choose to have no padding
- typically a stride >1
- reduces spatial dimension



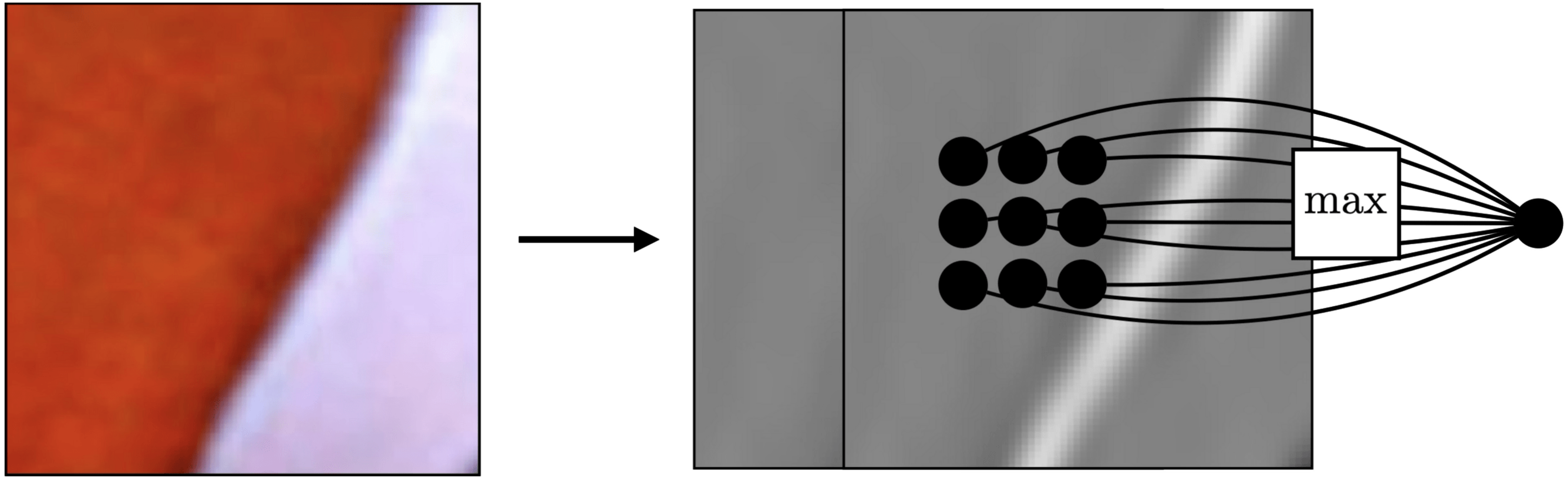
[gif adapted from demo source]

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



so the *channel* dimension remains *unchanged* after pooling.

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

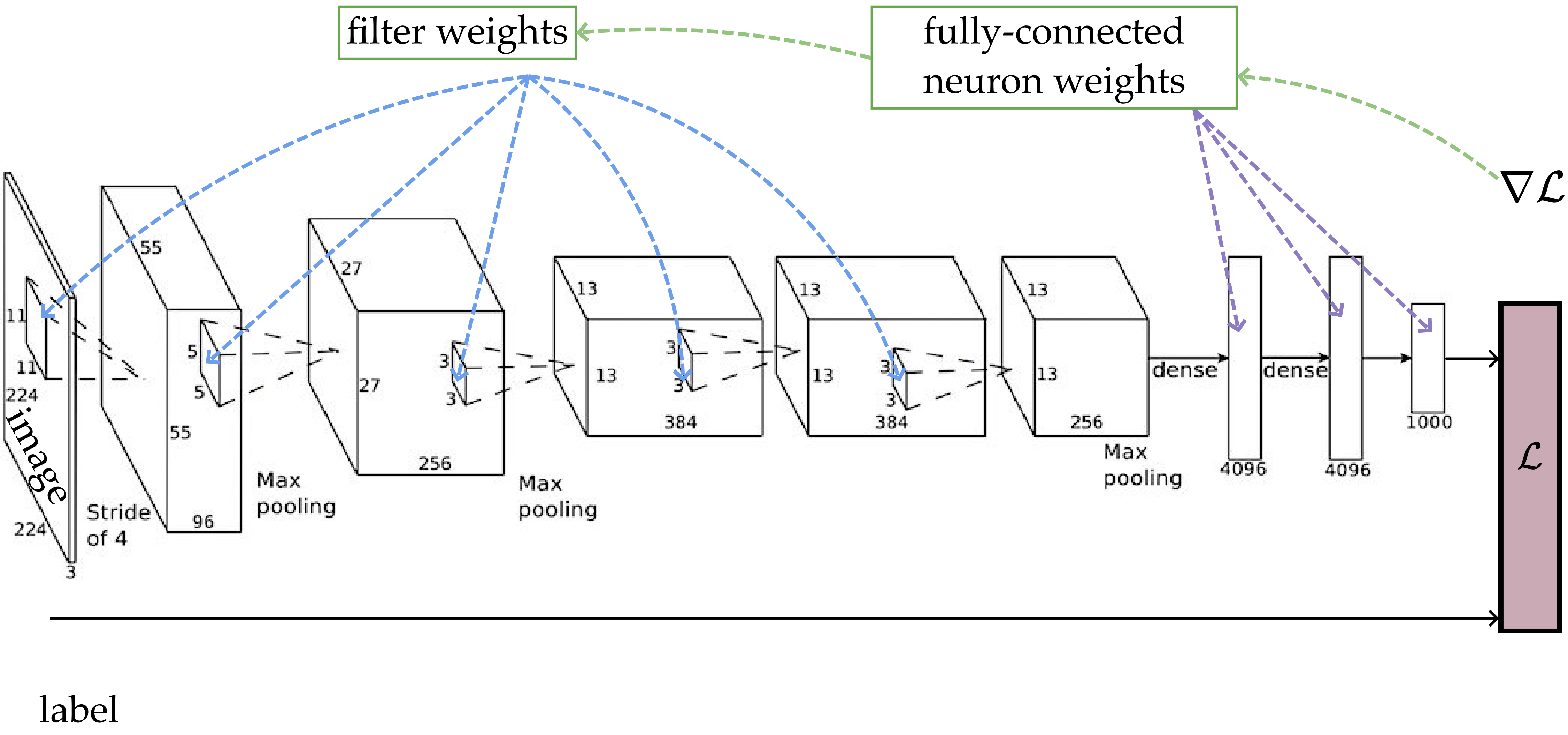


large response regardless of exact position of edge

[image credit Philip Isola]

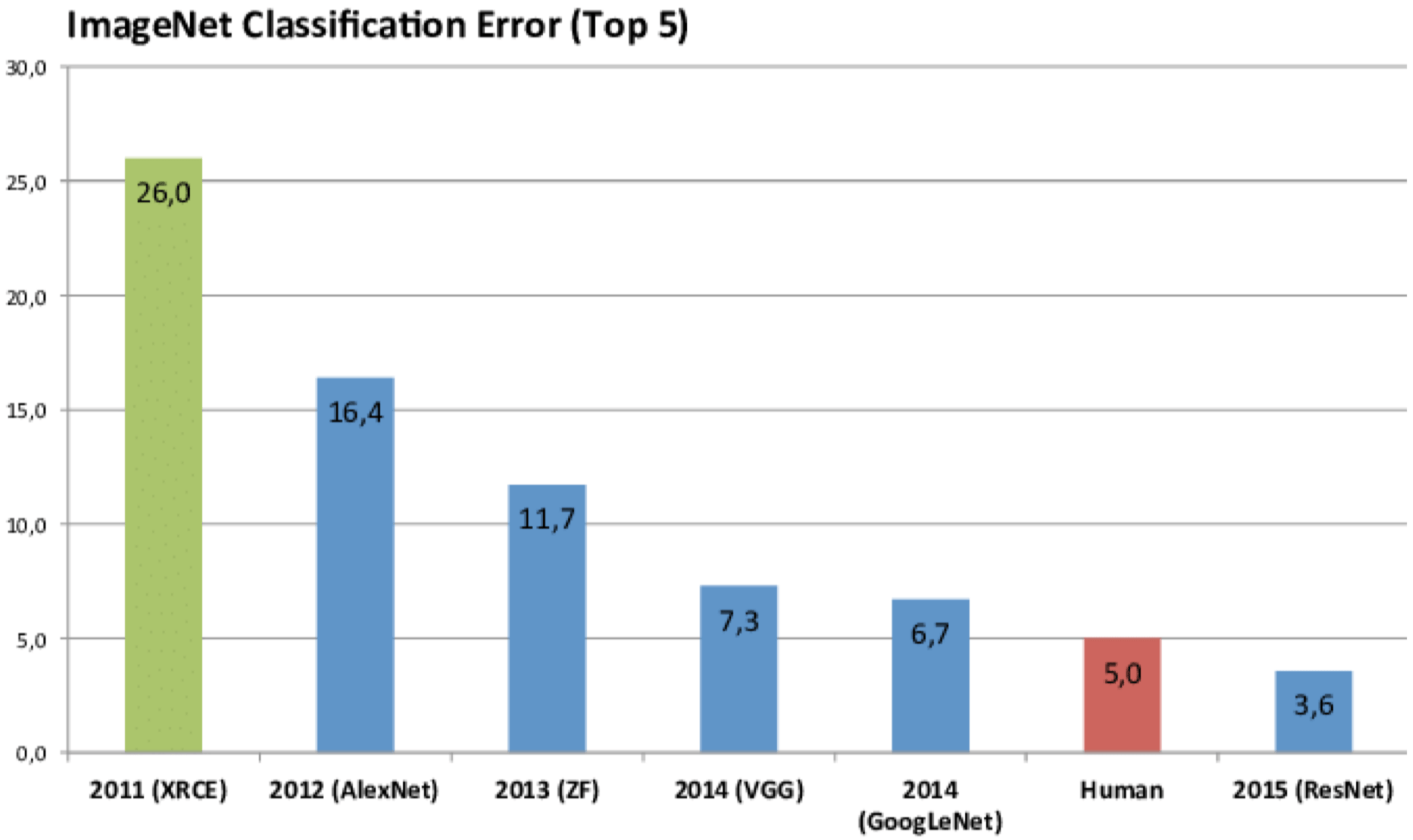
Outline

- Recap, fully-connected net
- Vision problem structure
- Convolutional network structure
- Convolution
  - 1-dimensional and 2-dimensional *convolution*
  - 3-dimensional *tensors*
- Max pooling
- (Case studies)

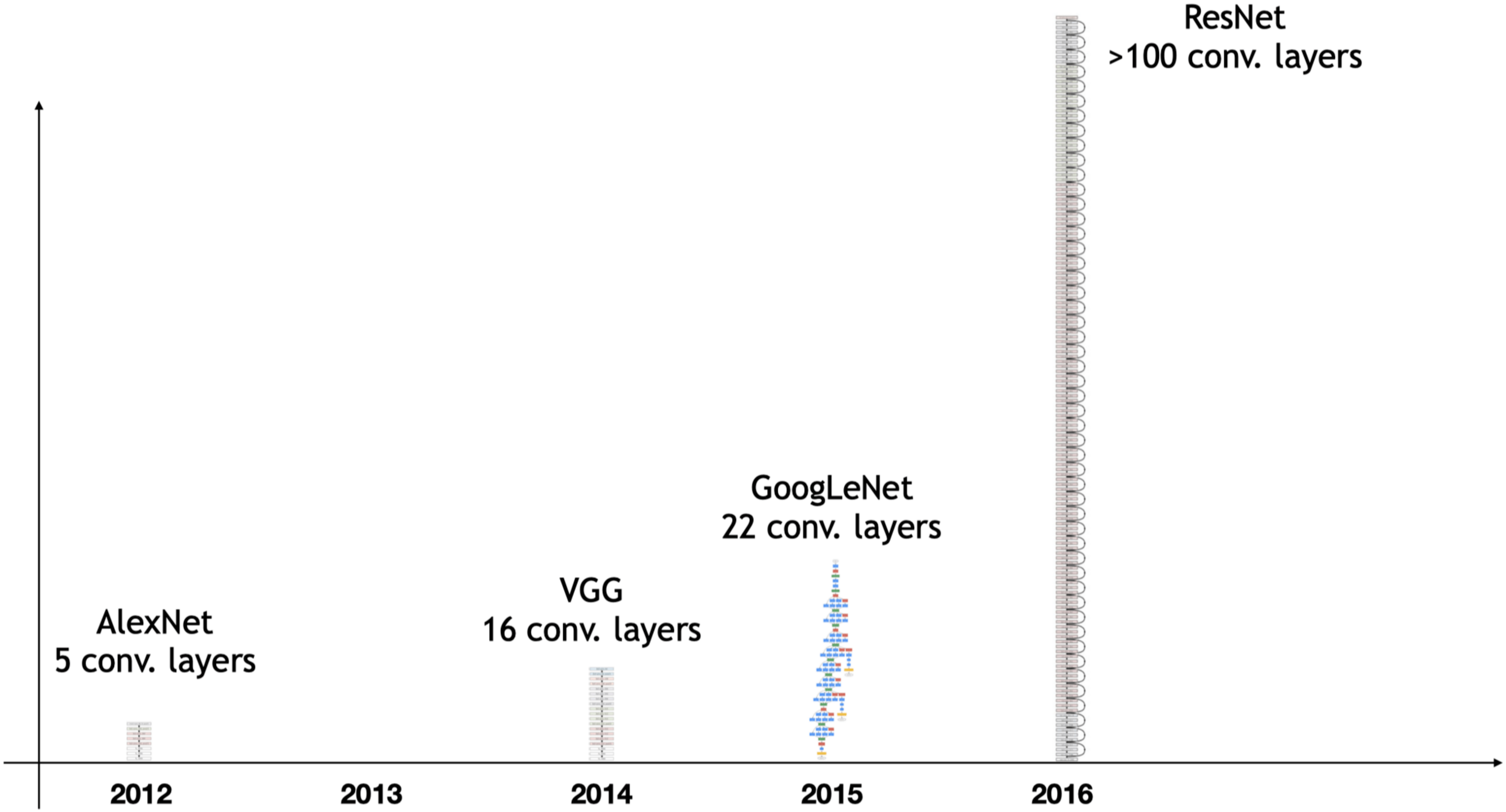


[AlexNet paper]

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



[image credit Philip Isola]

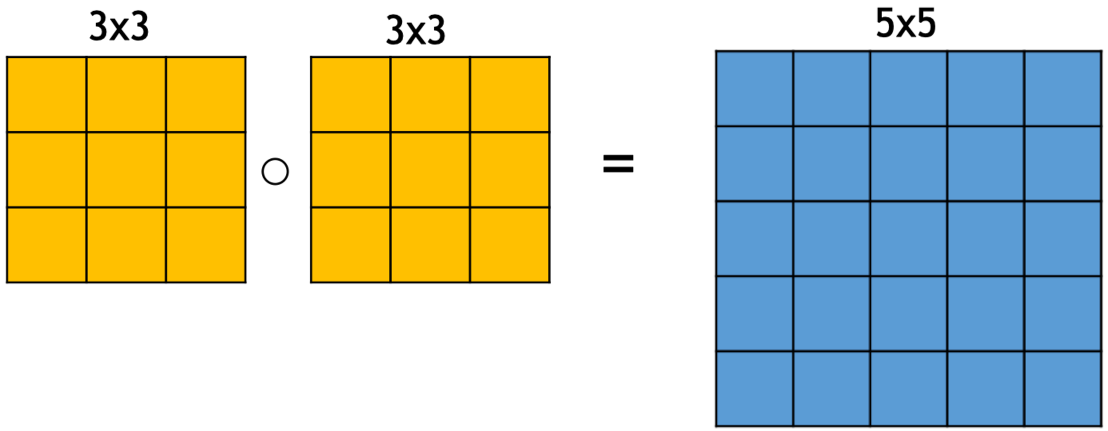


[image credit Philip Isola]

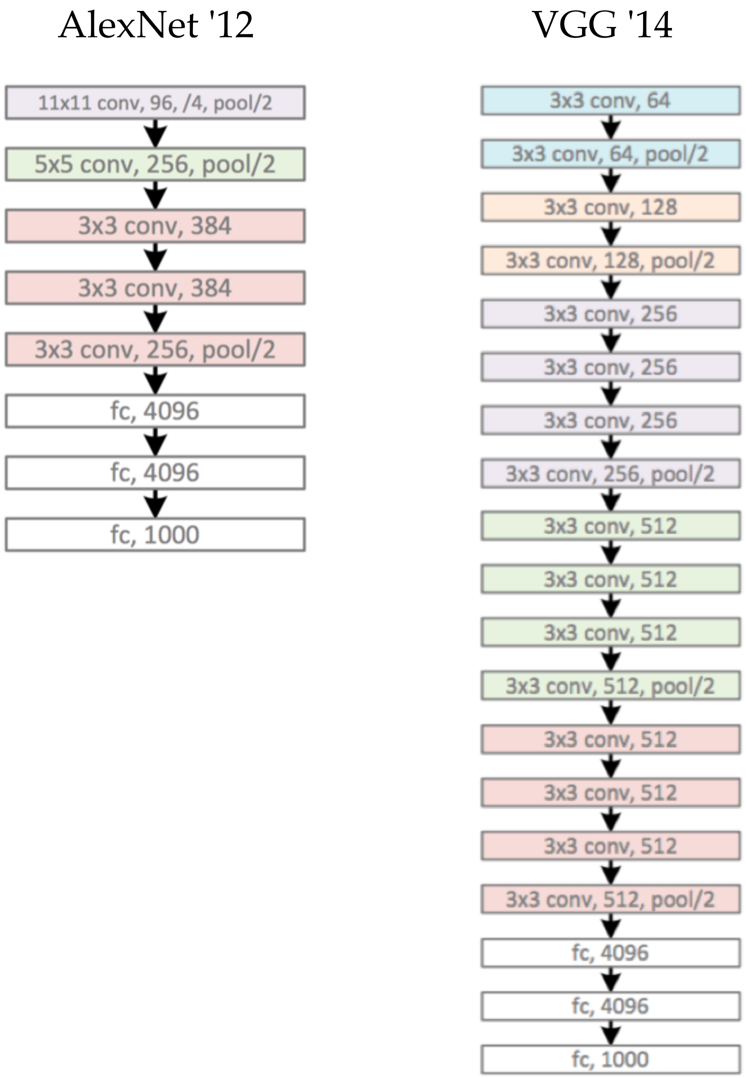
VGG '14

Main developments:

- small convolutional kernels: only 3x3



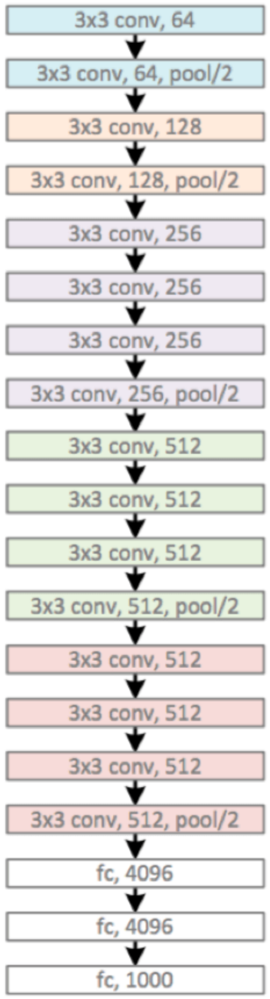
- increased depth: about 16 or 19 layers
- stack the same modules



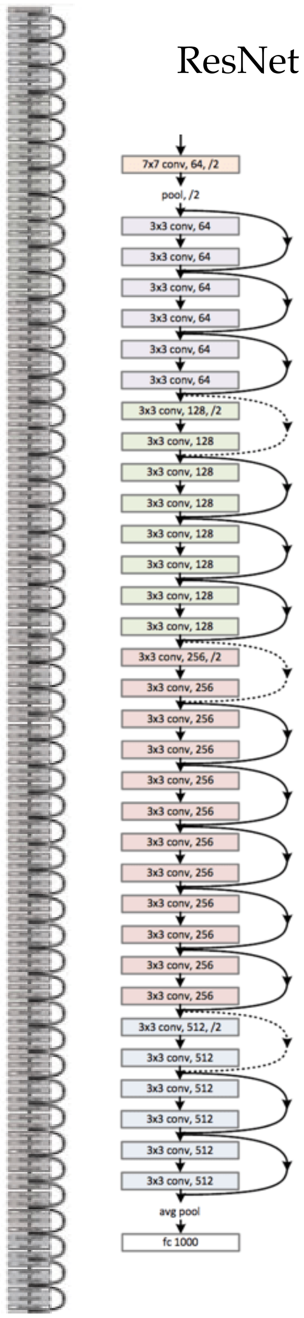
"Very Deep Convolutional Networks for Large-Scale Image Recognition", Simonyan & Zisserman. ICLR 2015

[image credit Philip Isola and Kaiming He]

VGG '14

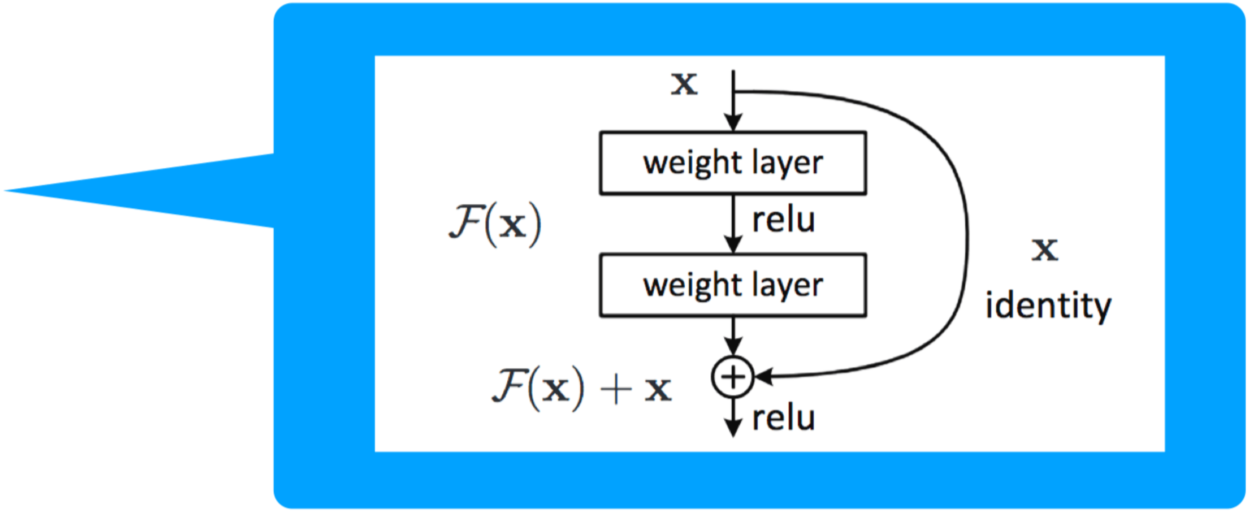


ResNet '16



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

- 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!