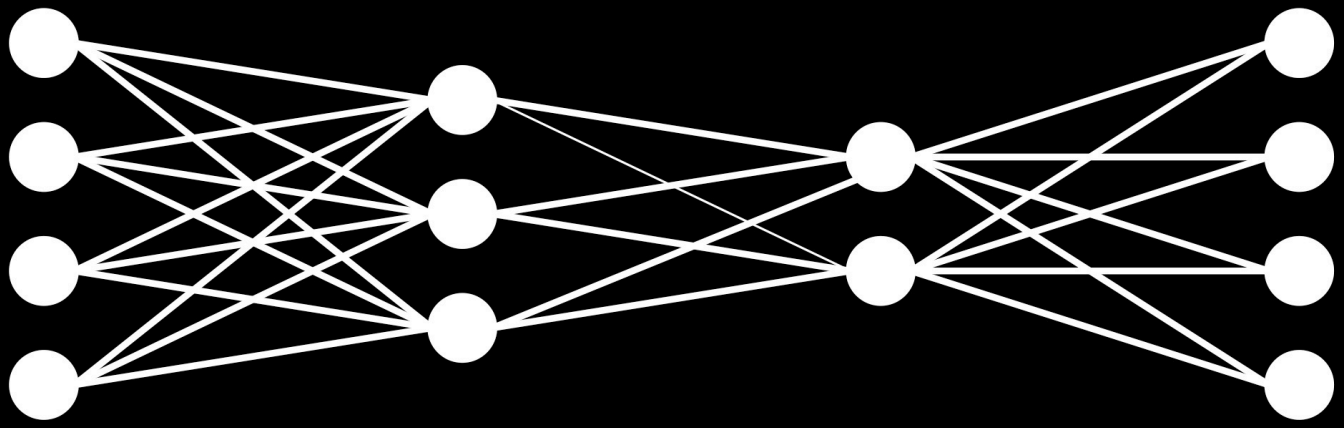
Neural Networks

Lecture

#3 CNNs



Cyrill Stachniss

Summer term 2024 – Cyrill Stachniss

5 Minute Preparation for Today

https://www.youtube.com/watch?v=0aE_J0fwye0

Photogrammetry & Robotics Lab

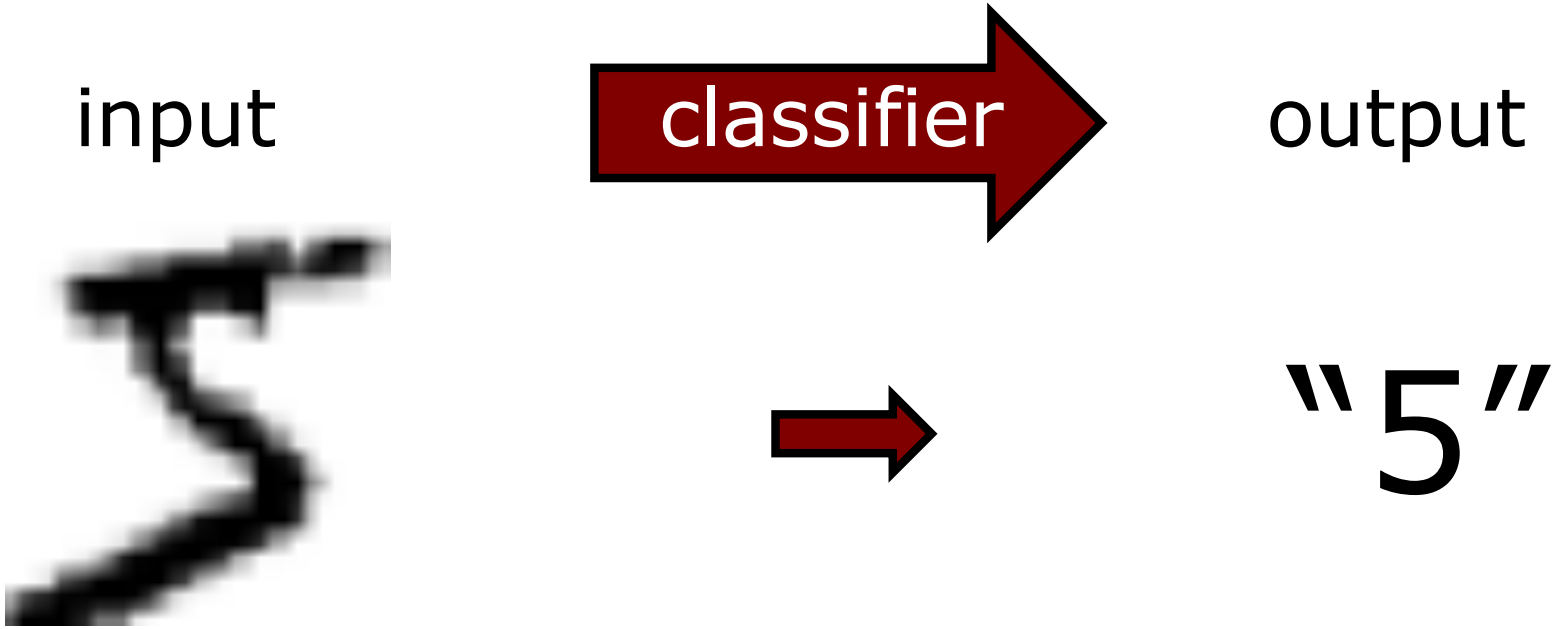
Convolutional Neural Networks

Cyrill Stachniss

The slides have been created by Cyrill Stachniss.
I took a lot of inspiration from lectures given by Justin Johnson and Fei-Fei Li.

NN Part 1 on MLPs

- What are neurons and neural networks
- Activations, weights, biases
- Multi-layer perceptron (MLP)
- MLP for simple image classification

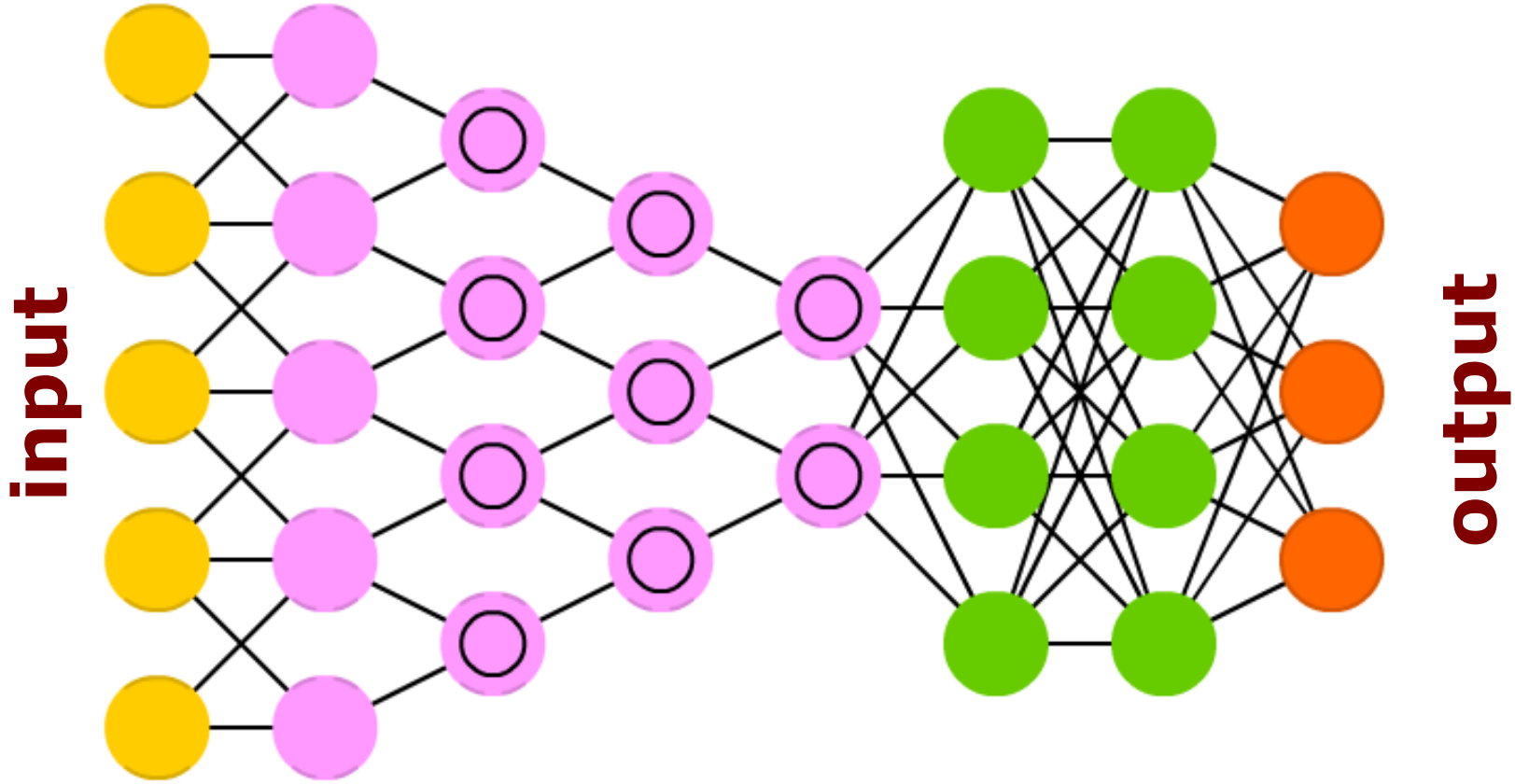


NN Part 2 on Training NNs

- Parameters define the NN's task
- Parameters are the weights and biases
- Learning/training NNs means estimating these paramaters
- Done by minimizing of a loss function using stochastic gradient descent
- Backpropagation to compute gradients
- End-to-end: no manually designed features, raw inputs to outputs

Convolutional Neural Networks

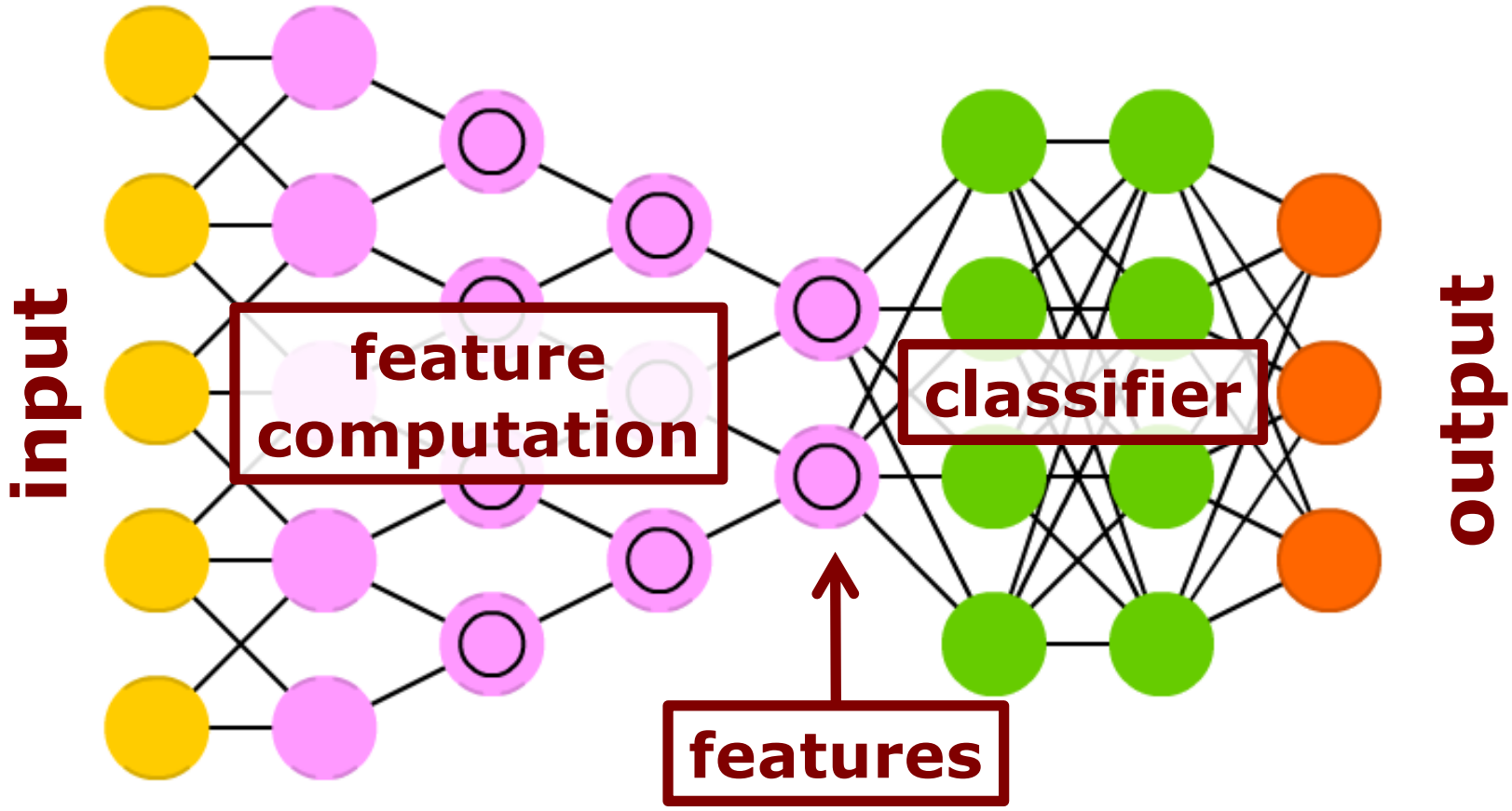
In image-related learning tasks, CNNs play an important role



[Image courtesy: van Veen]

Convolutional Neural Networks

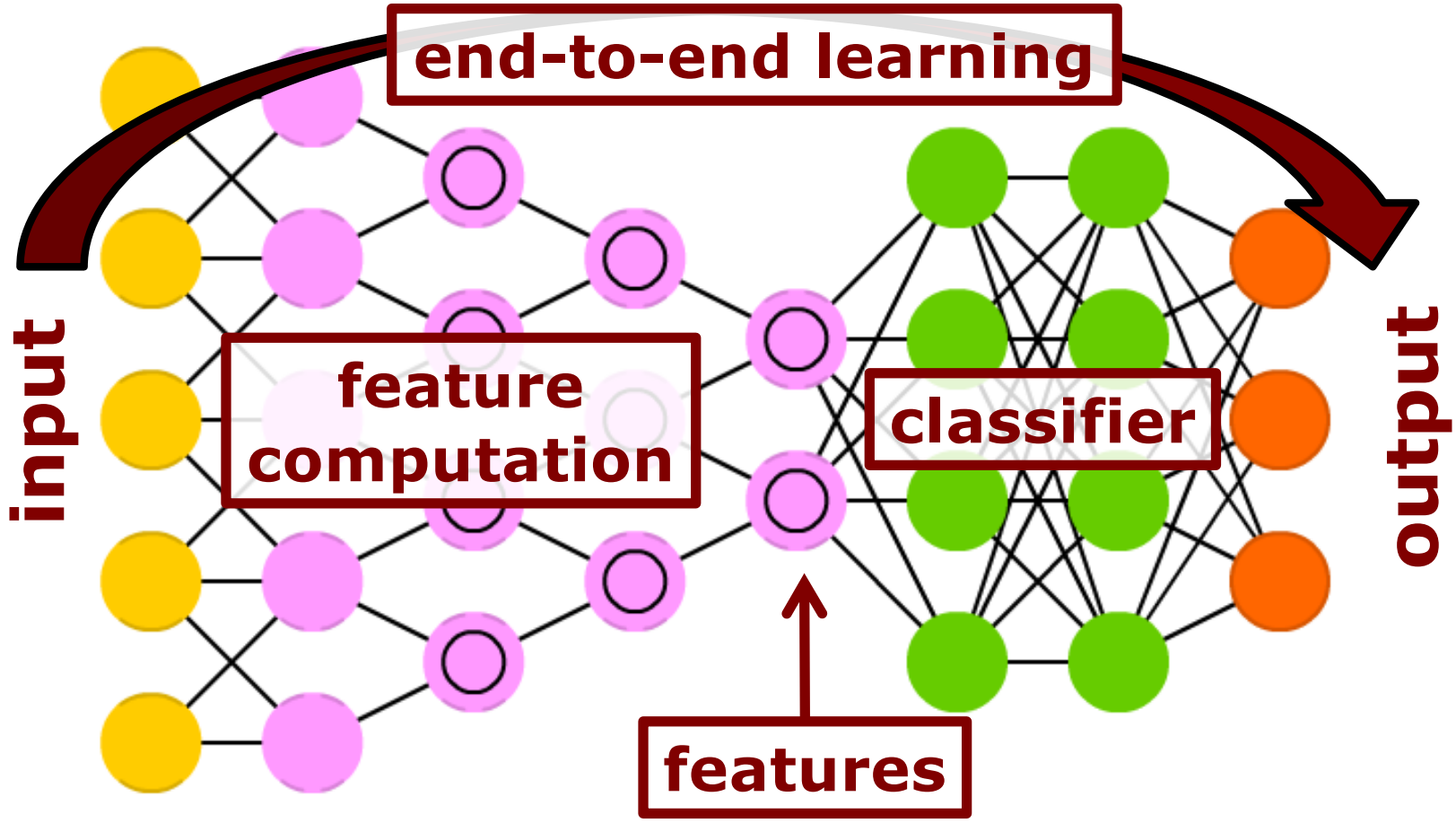
In image-related learning tasks, CNNs play an important role



[Image courtesy: van Veen]

Convolutional Neural Networks

In image-related learning tasks, CNNs play an important role



[Image courtesy: van Veen]

The Good Old MLP's Input...

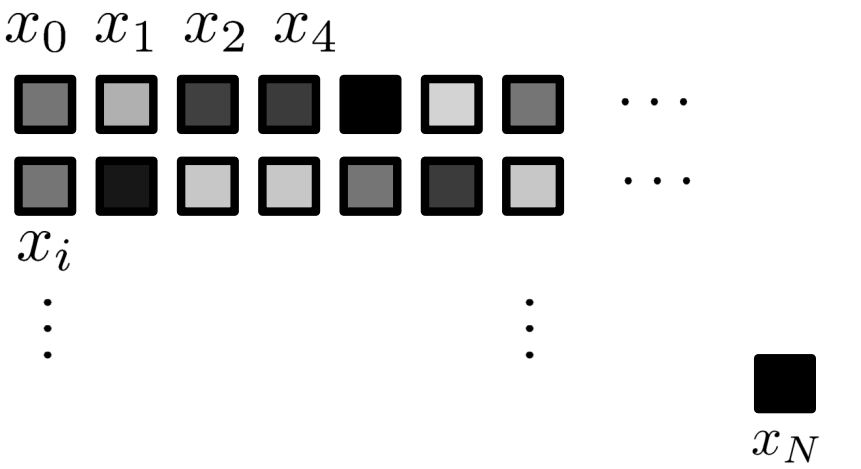
pixel intensities

image

An image consists of individual pixels.

Each pixel is an intensity value.

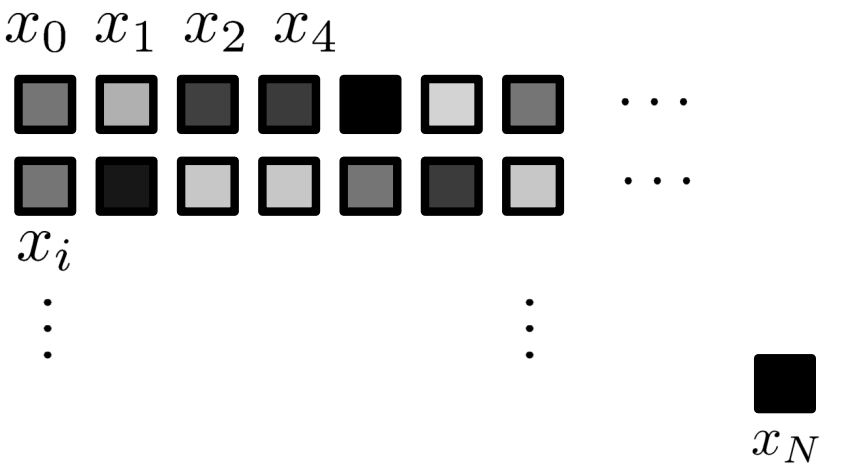
The Good Old MLP's Input...



An image consists of individual pixels.

We have N+1 such intensity values.

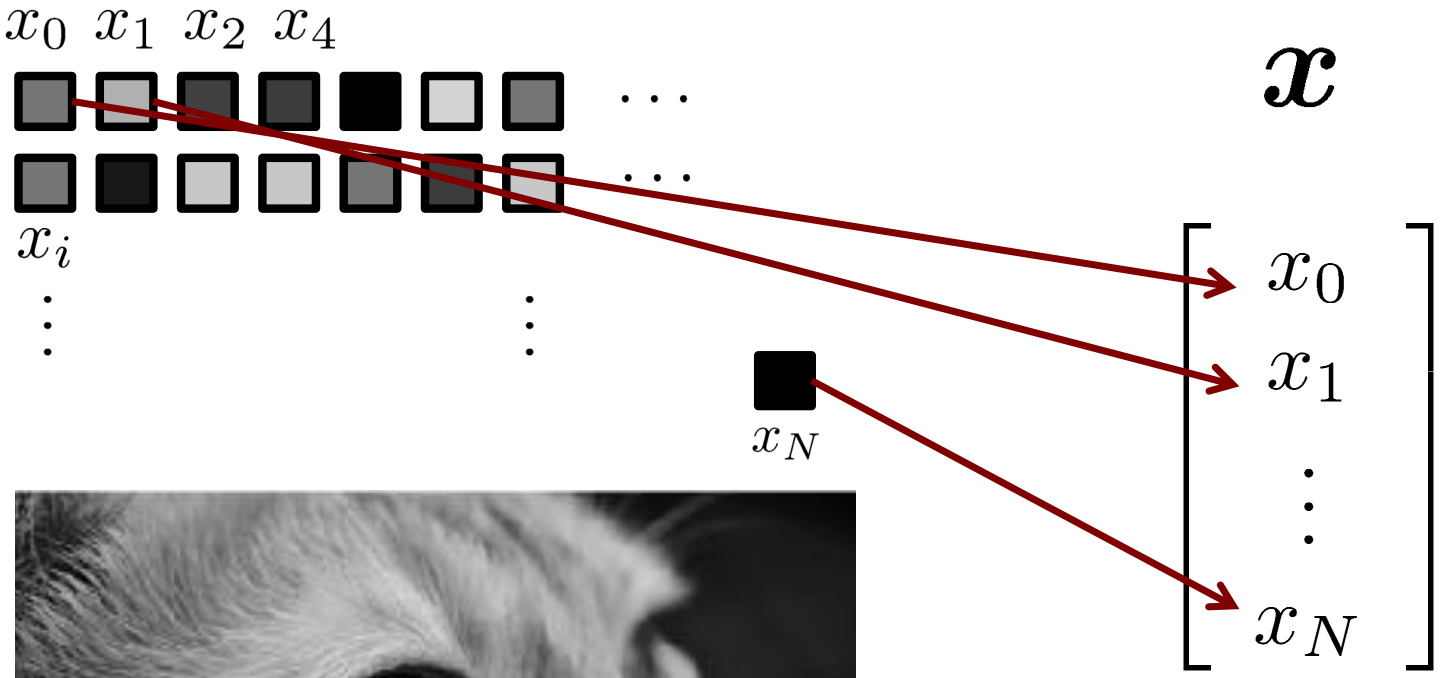
The Good Old MLP's Input...



$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix}$$

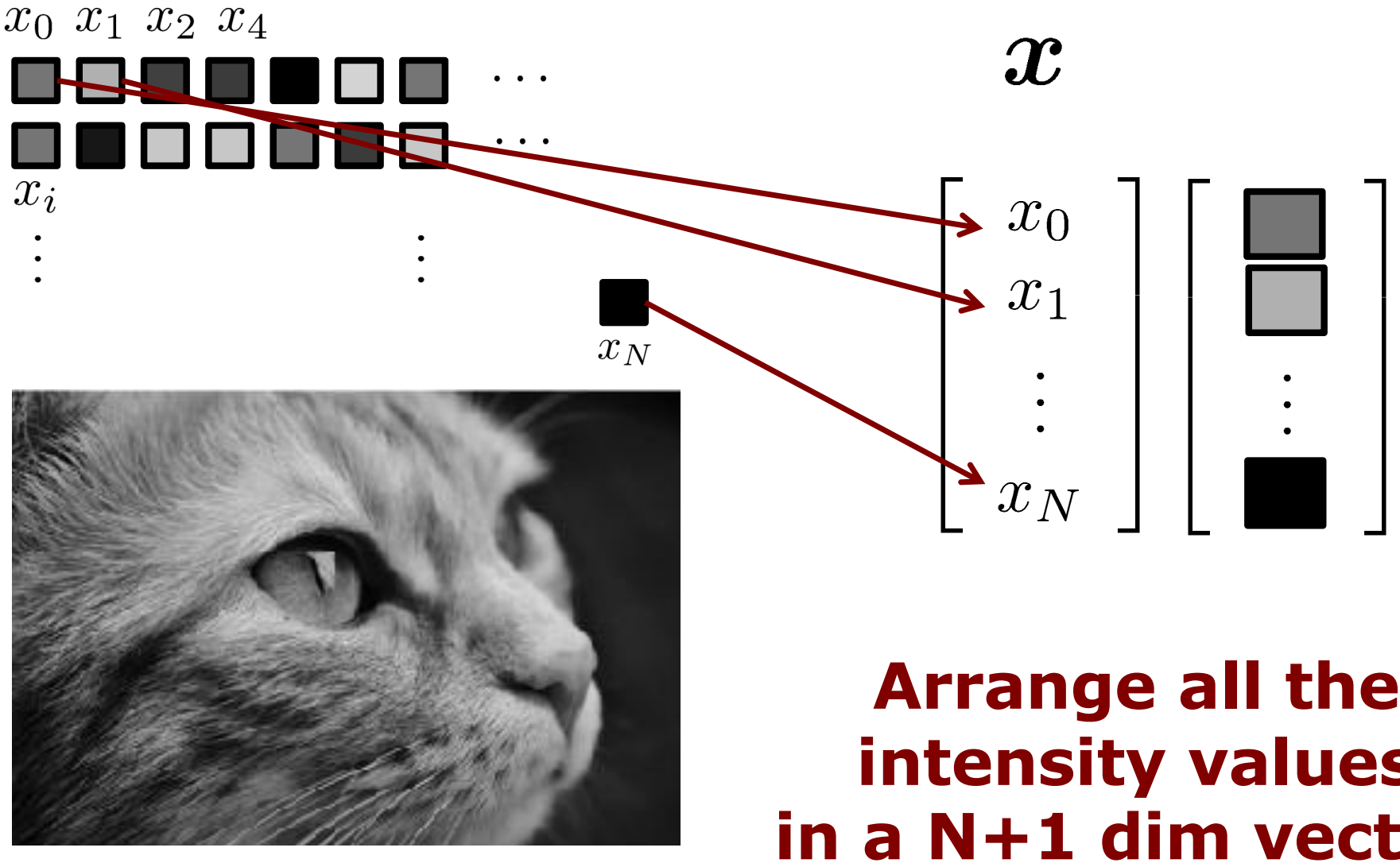
Arrange all the intensity values in a N+1 dim vector.

The Good Old MLP's Input...



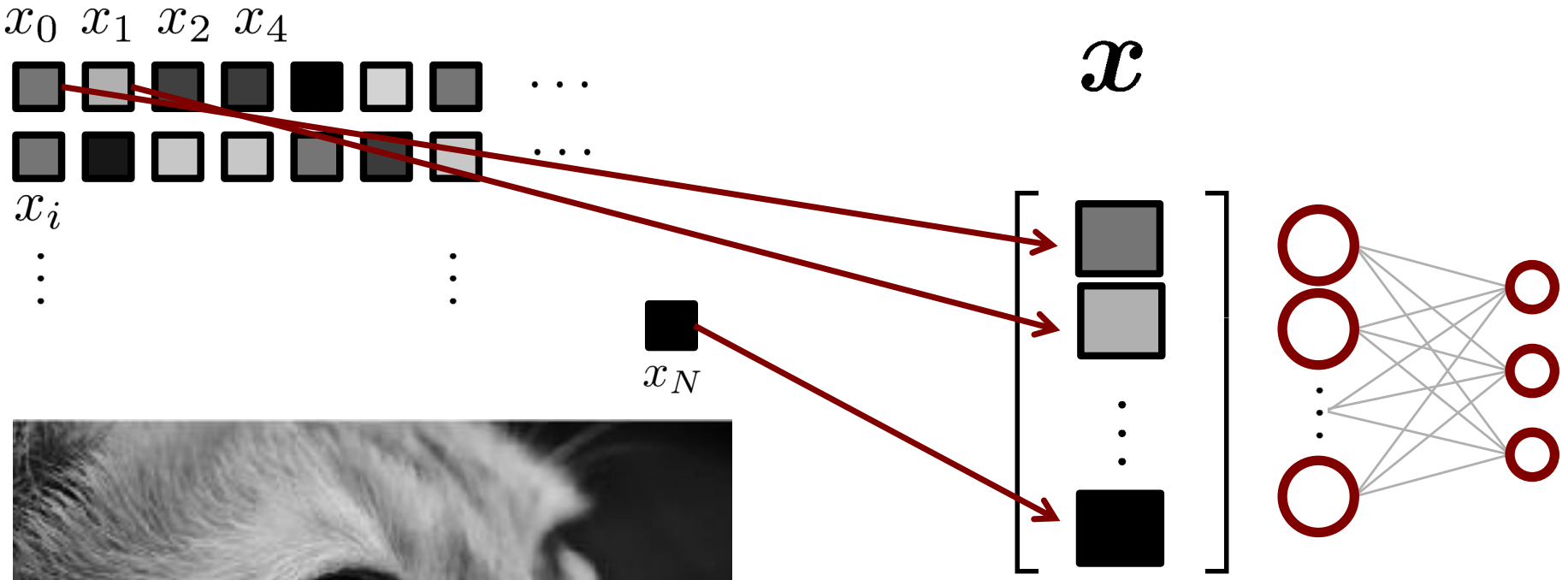
Arrange all the intensity values in a N+1 dim vector.

The Good Old MLP's Input...



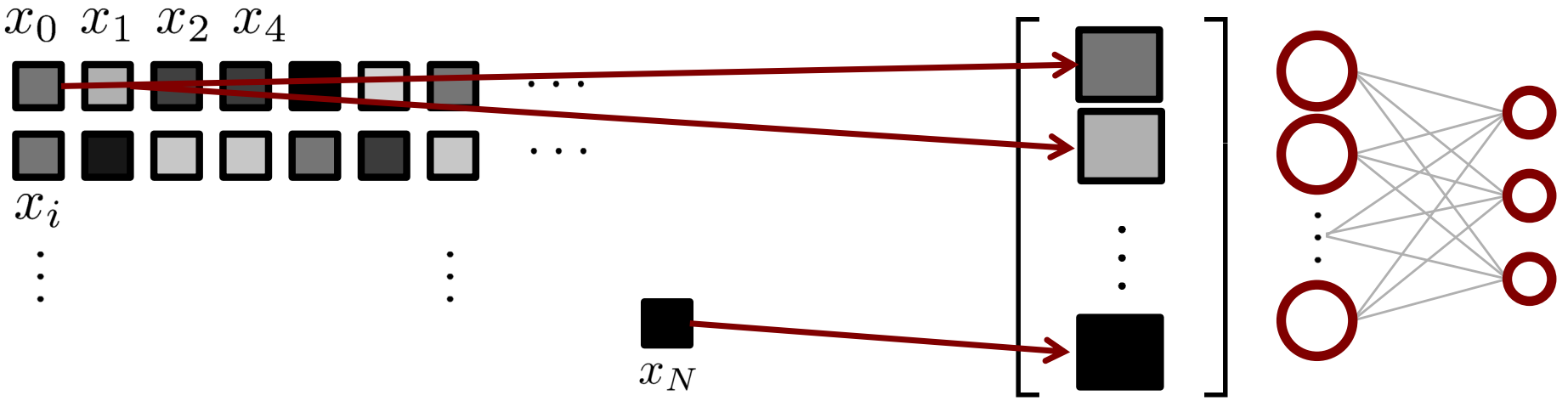
Arrange all the intensity values in a N+1 dim vector.

The Good Old MLP's Input…



This vector is the input layer of our network!

The Good Old MLP's Input...



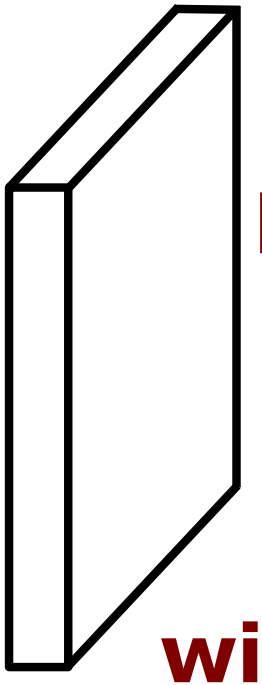
Problem: the approach destroys the spatial information as it ignores the local pixel neighborhoods in the image!

CNNs Overcome this Problem

- CNNs maintain the 2D image structure
- **Neighborhoods** are maintained
- Network layers can learn features encoding local spatial information
- Convolutions are **local** operators
- CNNs use convolutions & subsampling (called pooling)
- Local vs. global operators

Convolutions

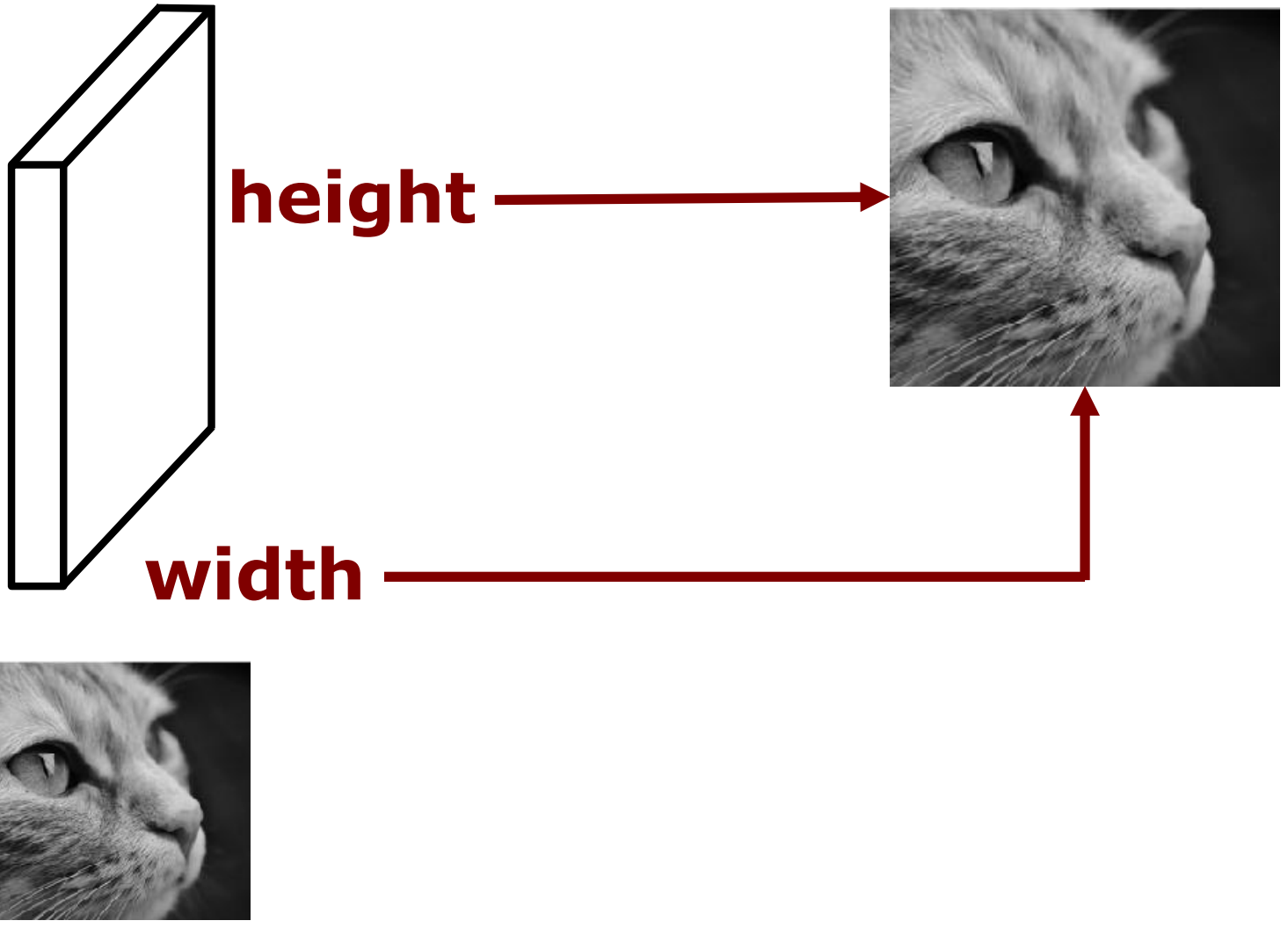
Let's Start With the Input



height

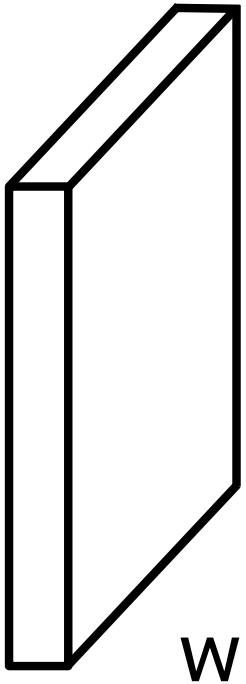
width

Let's Start With the Input



Let's Start With the Input

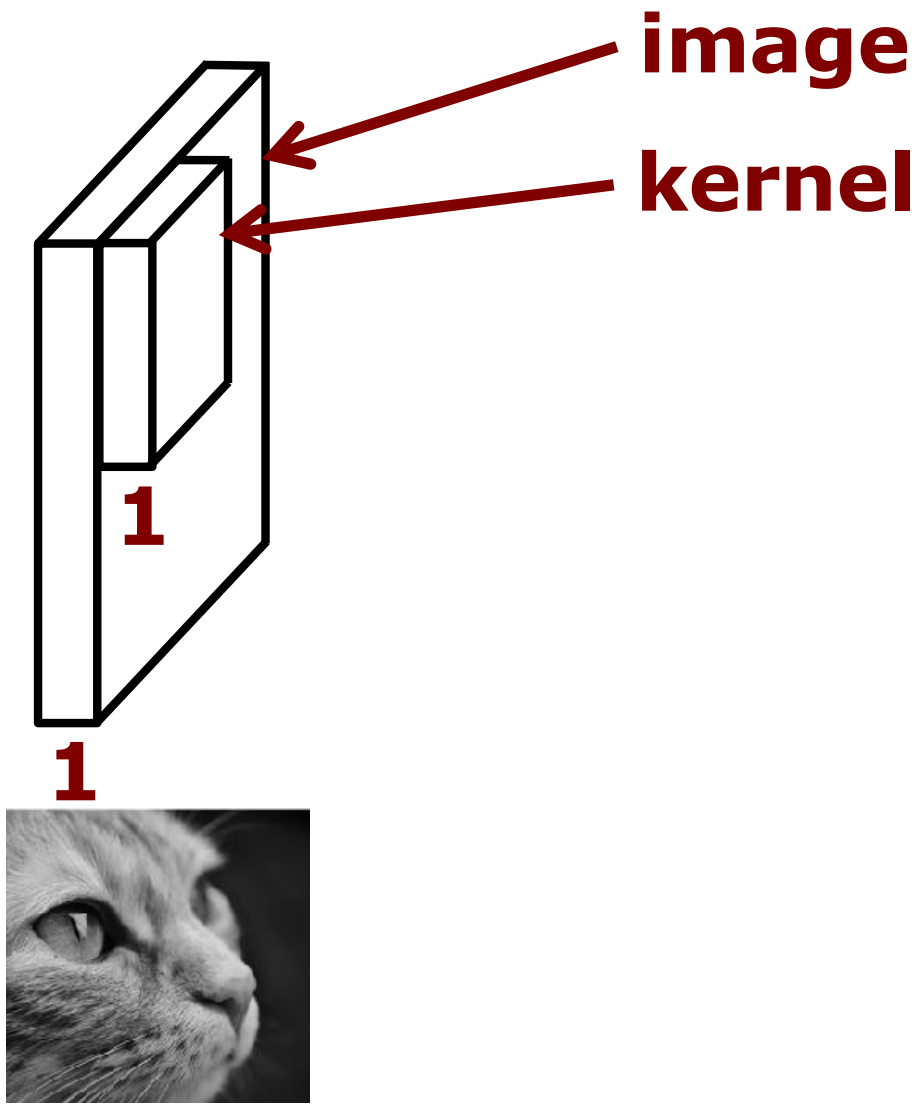
channels/depth



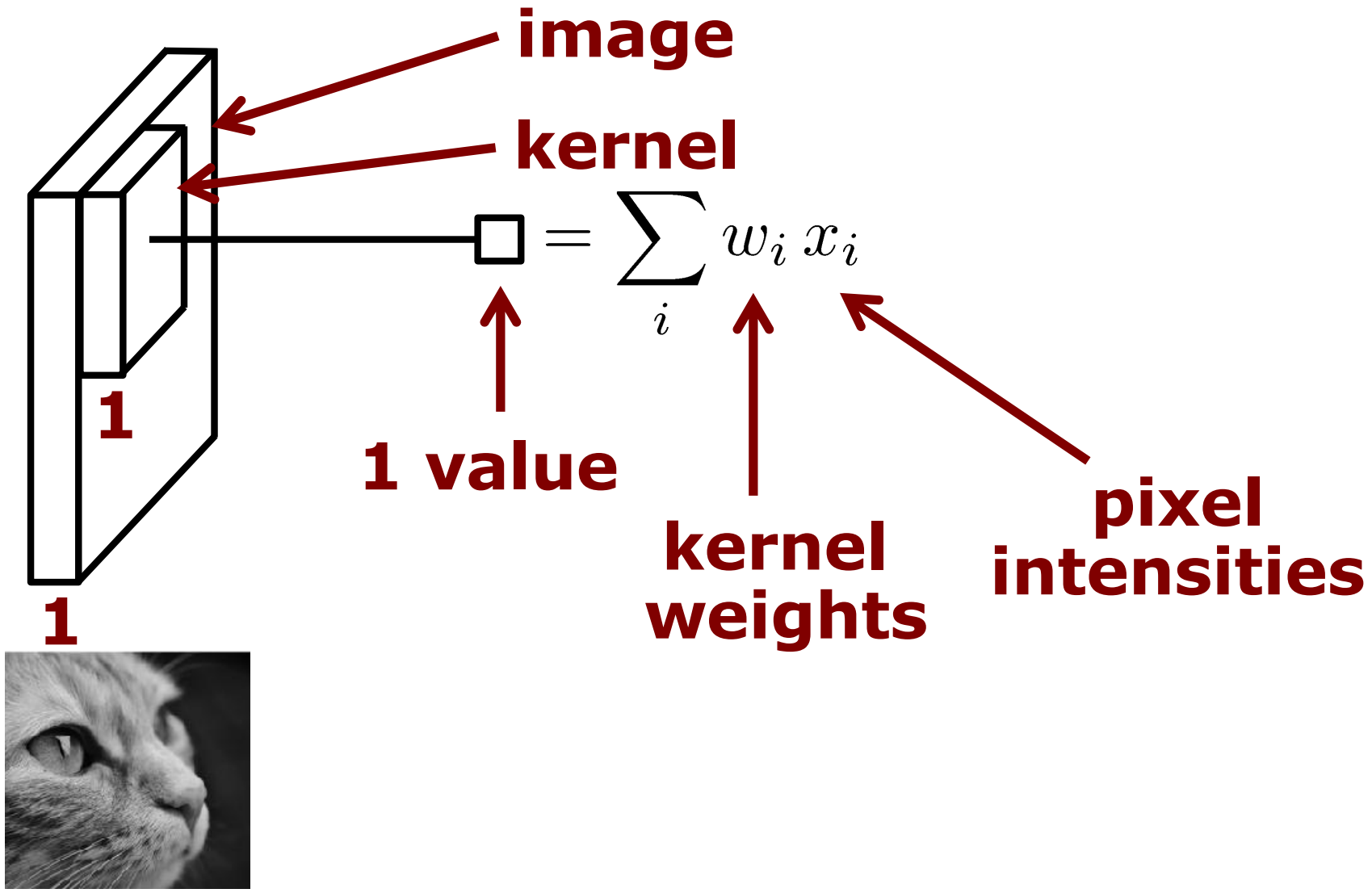
depth=1

depth=3

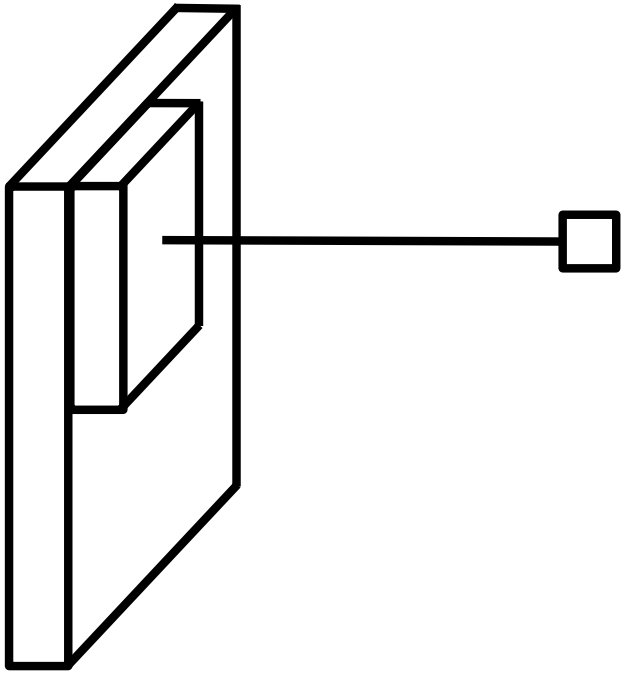
Convolution Using a Kernel



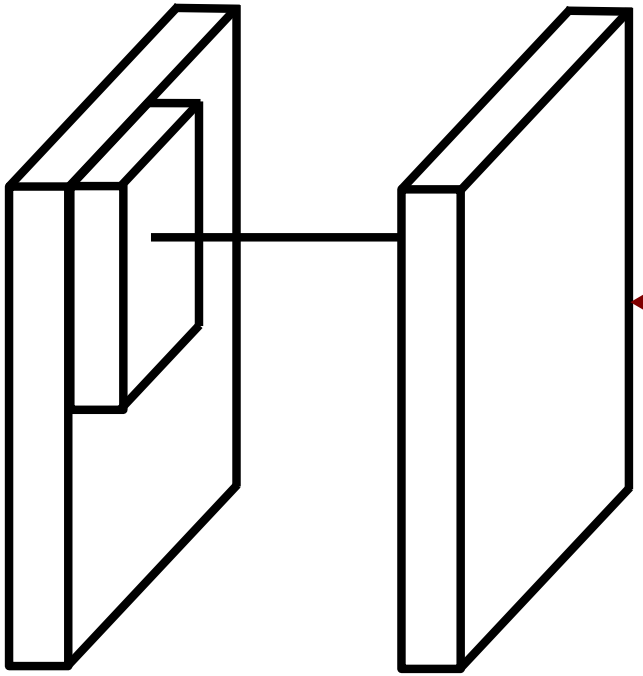
Convolution Using a Kernel



Convolution Using a Kernel



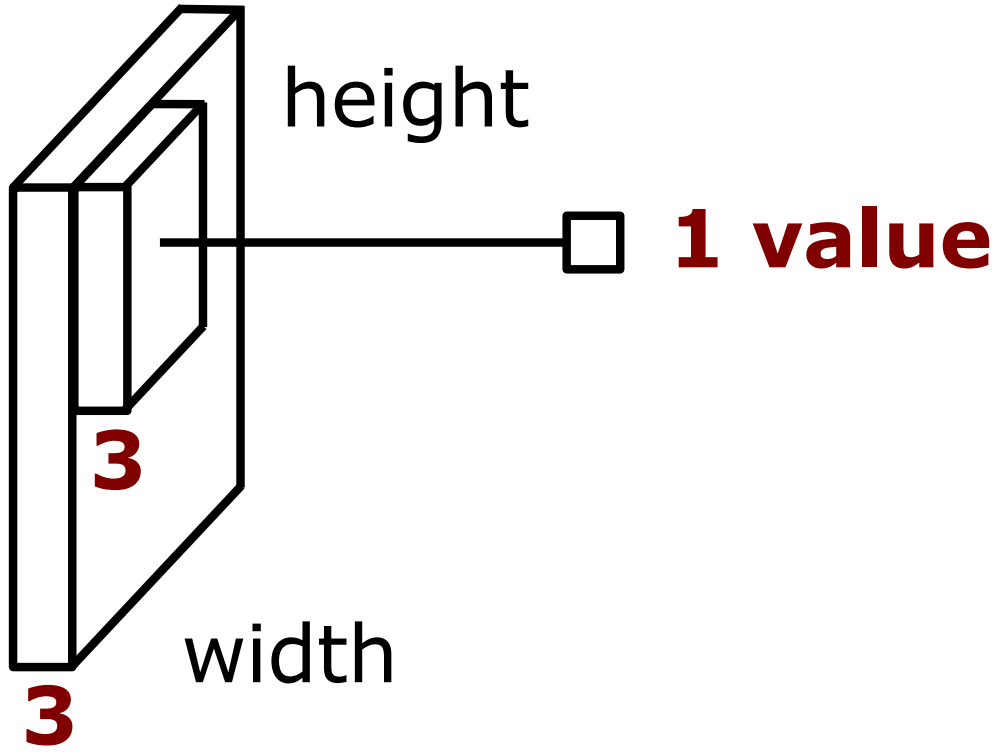
Convolution Using a Kernel



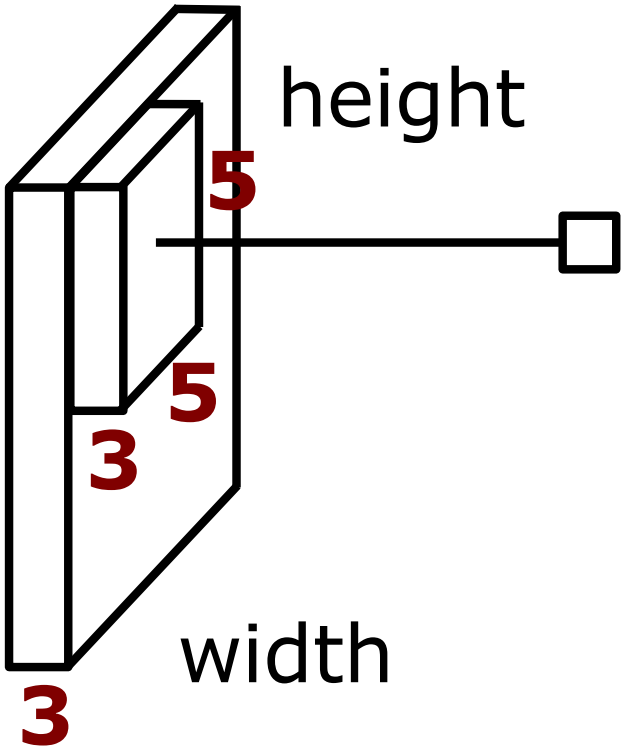
This is the output (image) of a convolution!

example for blurring through convolution

Sizes



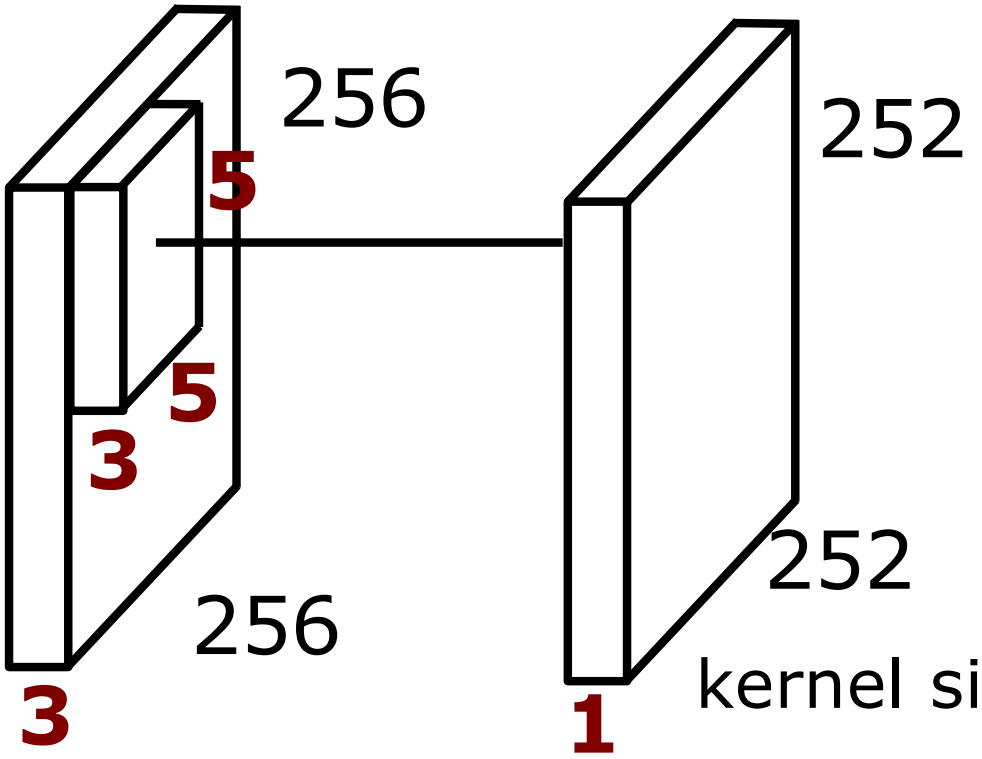
Sizes



kernel size: $3 \times 5 \times 5 = 75$

→ dot product of 75 dim. vectors

Sizes



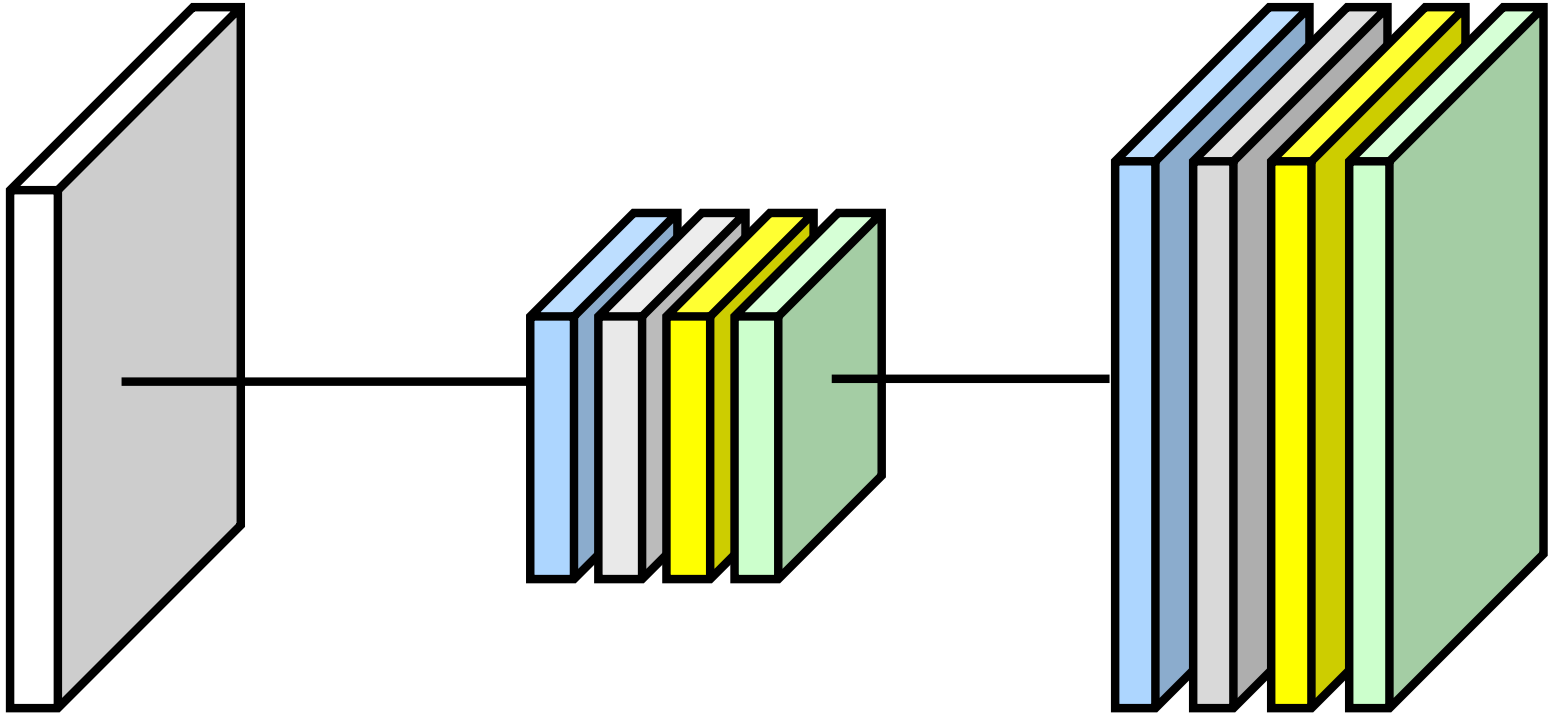
kernel size: $3 \times 5 \times 5 = 75$

→ dot product of 75 dim. vectors

number of such dot products:

$(256 - 4) \times (256 - 4) = 63.5k$

We Can Use Multiple Kernels



1 input	4 kernels	4 outputs activation maps
3xWxH	4x 3x5x5	4x 1x(W-4)x(H-4)

Sizes

Size of the activation map depends on

- Size of the input (W, H)
- Kernel size (K)

$$W' \times H' = (W - K + 1) \times (H - K + 1)$$

output size input size and kernel size

Padding

- Convolutions slightly shrink the image
- We can solve this by creating a border around the input image

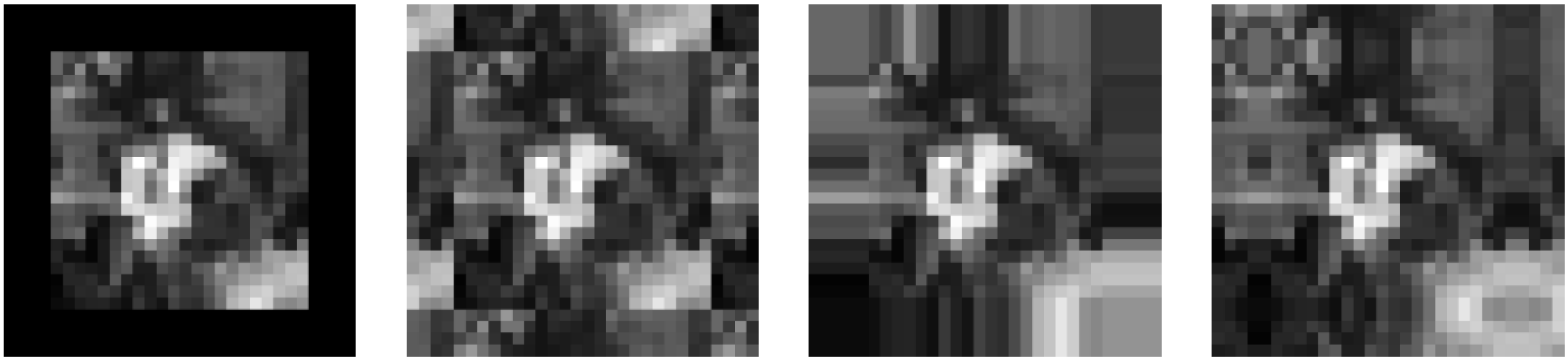


Image courtesy: Szelinsky

CNNs often use zero-padding

Sizes

Size of the activation map depends on

- Size of the input (W, H)
- Kernel size (K)
- Padding (P)

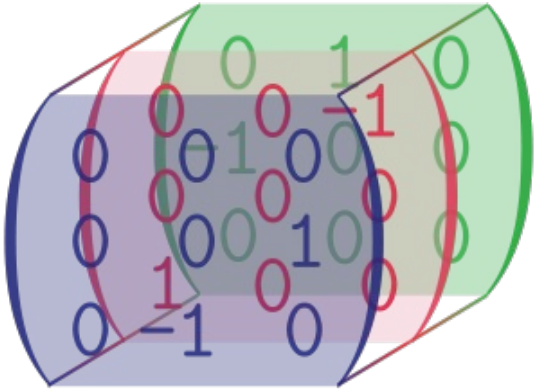
$$W' \times H' = (W - K + 1 + 2P) \times (H - K + 1 + 2P)$$

Tensor

- Vector is a 1 dimensional array
- Matrix is a 2-dimensional array
- Voxelgrid is a 3-dimensional array

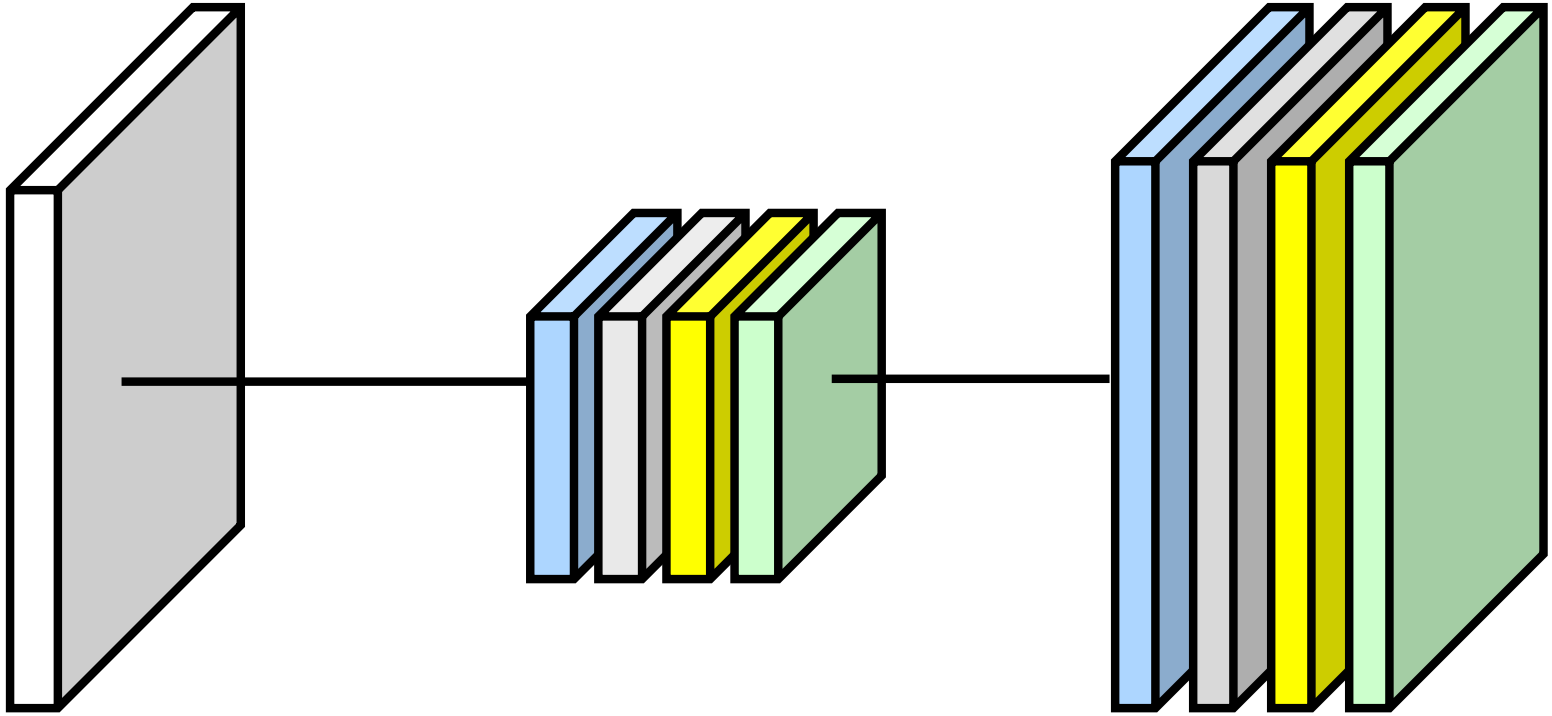
Tensor = generalization of the "array" (matrix) concept with a flexible number of dimensions

$$\epsilon_{ijk} =$$



[Image courtesy: A. Kriesch]

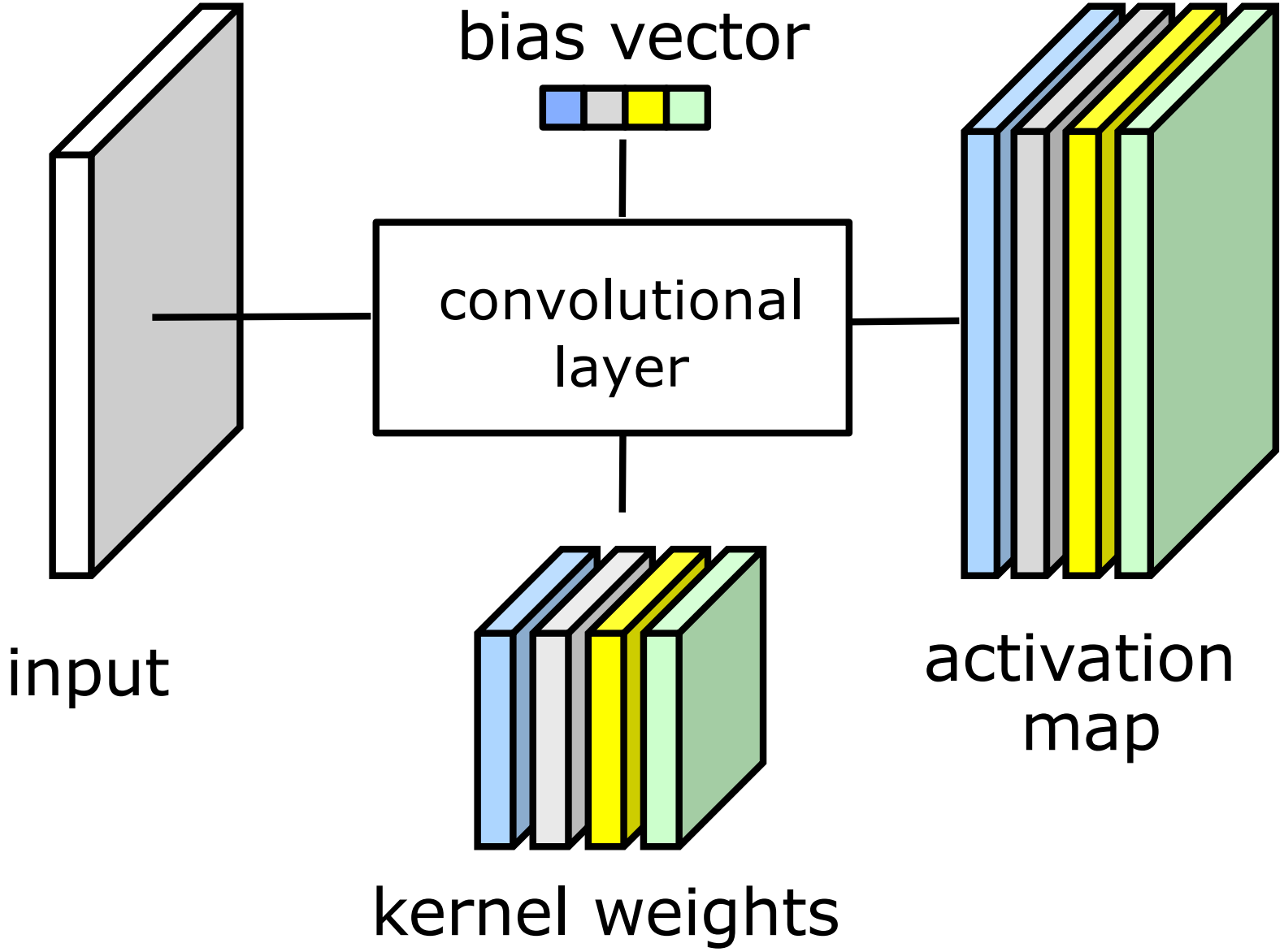
Each Layer is a Tensor



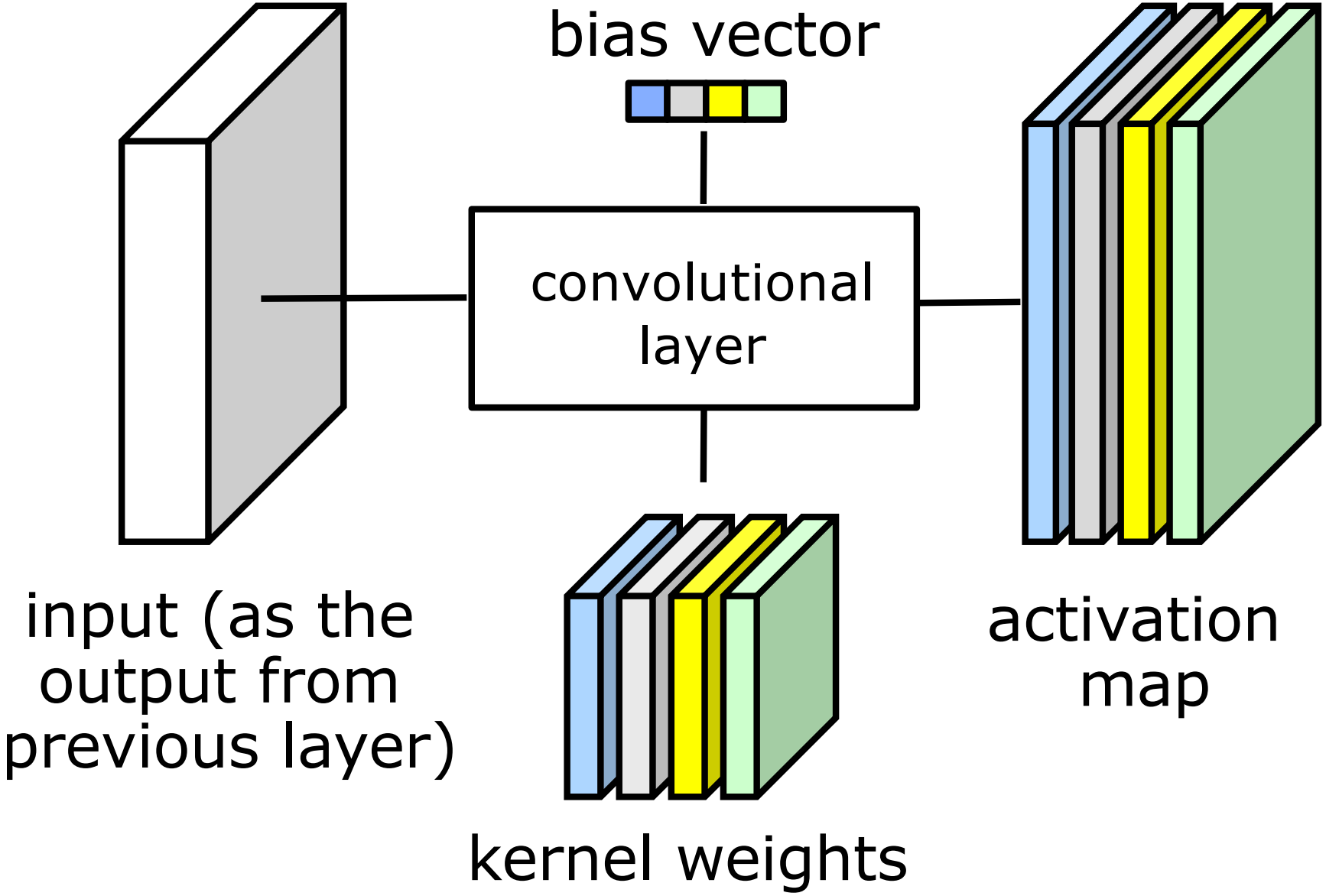
3D tensor | 4D tensor | 4D tensor

3xWxH | Nx3xKxK | Nx1xW'xH'

Convolutional Layer Parameters



Stacking Convolutional Layers



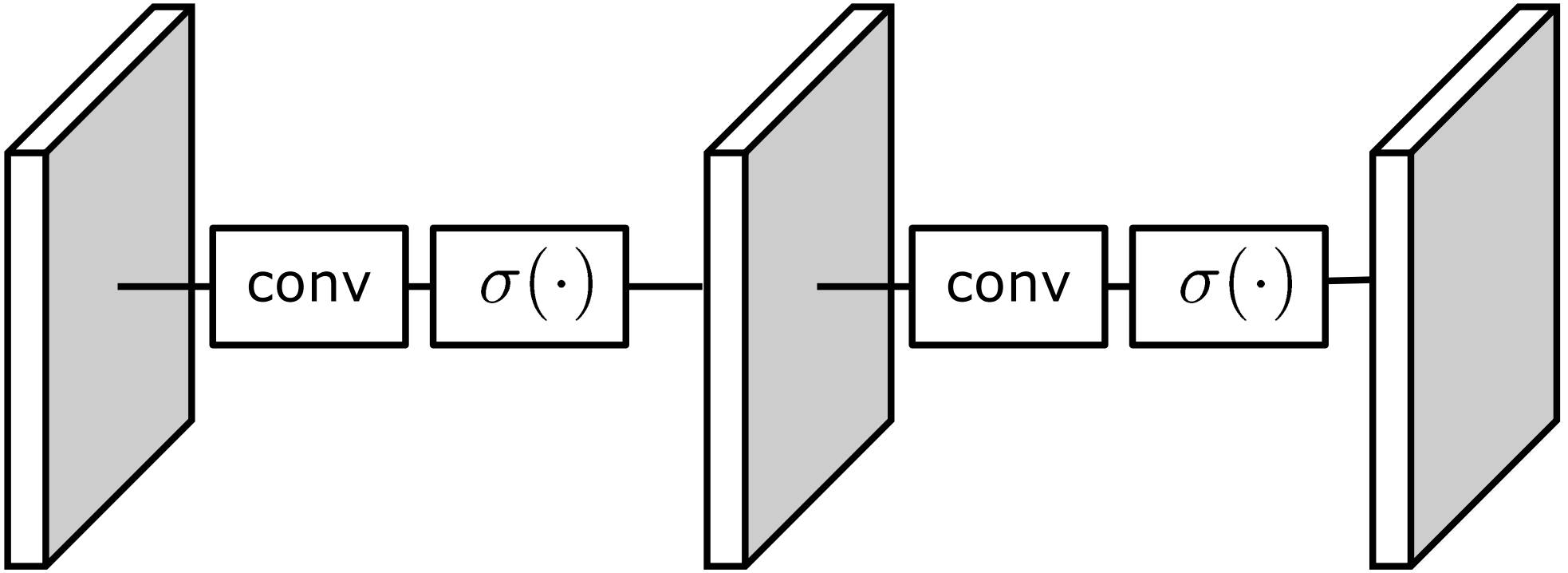
Stacking Convolutions

- For linear shift-invariant kernels, we know that concatenations of convolutions are again a convolution

$$f * (g_1 * \ldots * g_n) = f * g$$

- Multiple layers can be combined into a single convolution
- Property breaks when introducing a non-linear activation function

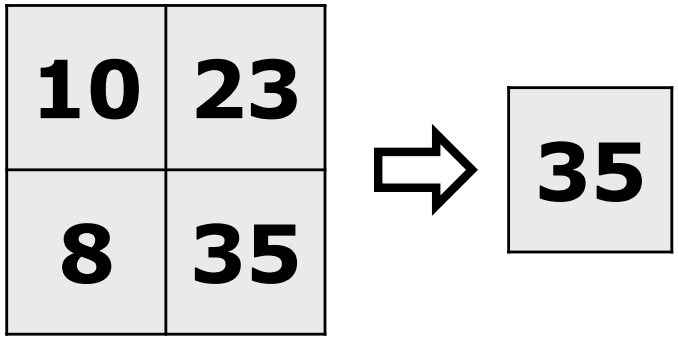
Stacking Convolutions Layers With Activation Functions



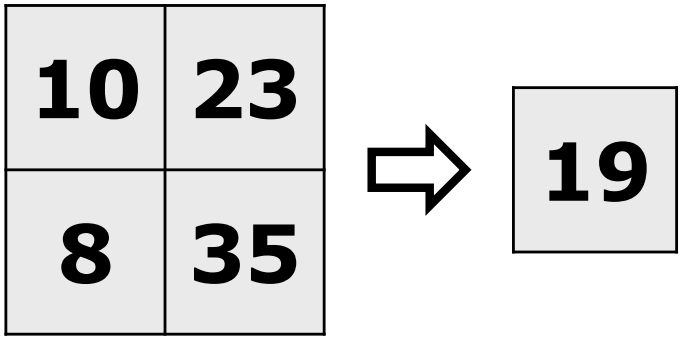
Pooling

Pooling

- Besides convolutions, CNNs also use pooling layers
- Pooling combines multiple values into a single value to reduce the tensor sizes and combine information
- Prominent examples are:



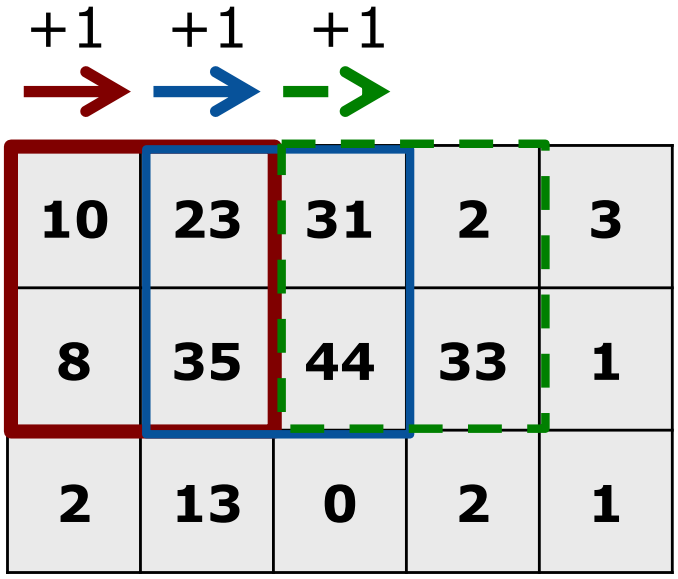
max-pooling



avg-pooling

Stride

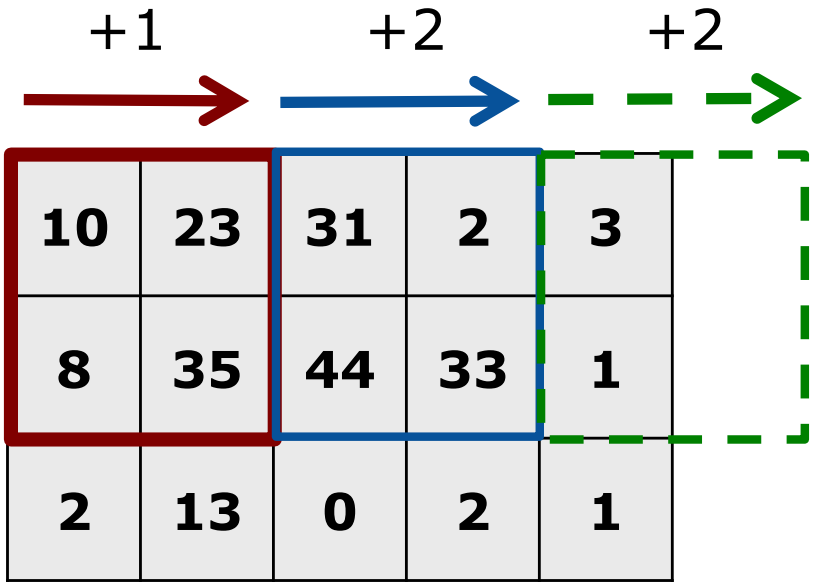
- Stride defines by how many pixels we shift the filter forward each step
- Larger stride reduces overlaps and makes the resulting image smaller



stride=1

Stride

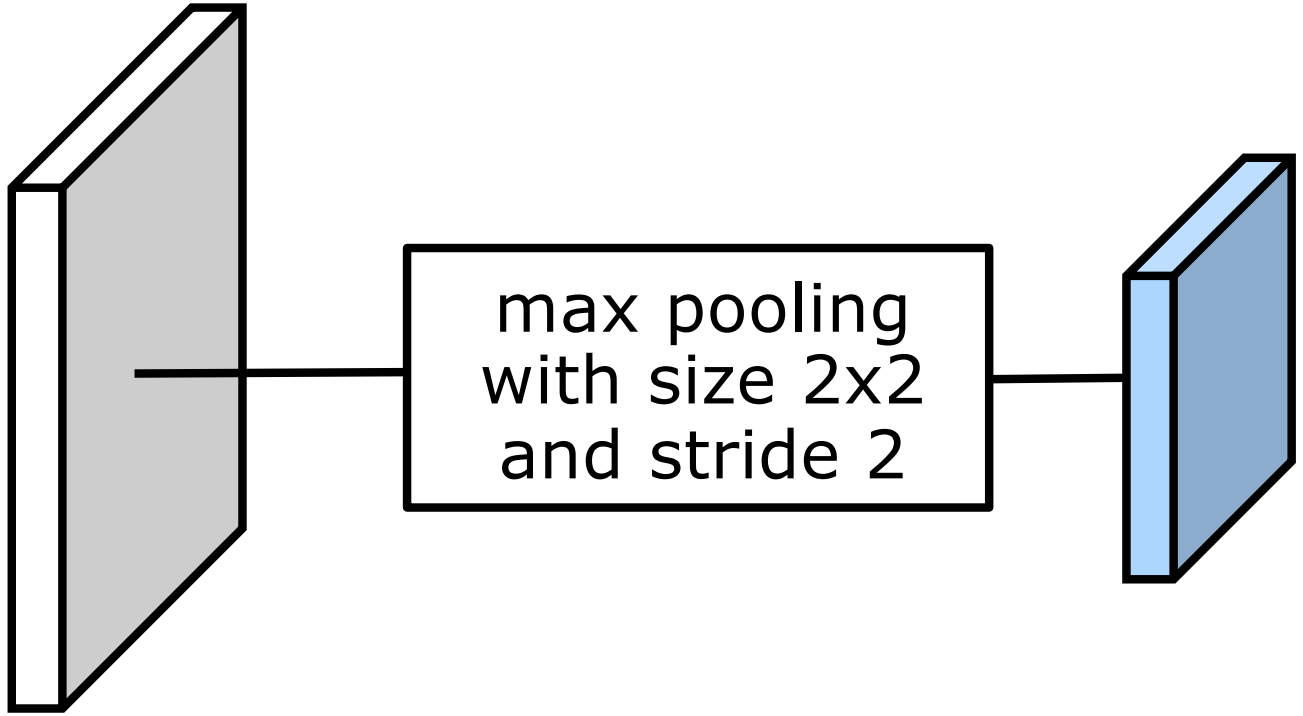
- Stride defines by how many pixels we shift the filter forward each step
- Larger stride reduces overlaps and makes the resulting image smaller



stride=2

Max Pooling Example

Size tells us how many values to combine and stride define by how much to shift the mask



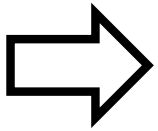
$W \times H$ 2×2 $W/2 \times H/2$

Max Pooling Example

2×2

10	23	31	2	3	34
8	35	44	33	1	45
2	13	0	2	1	7
12	3	8	22	9	88
22	88	3	0	2	0
1	9	33	3	4	4

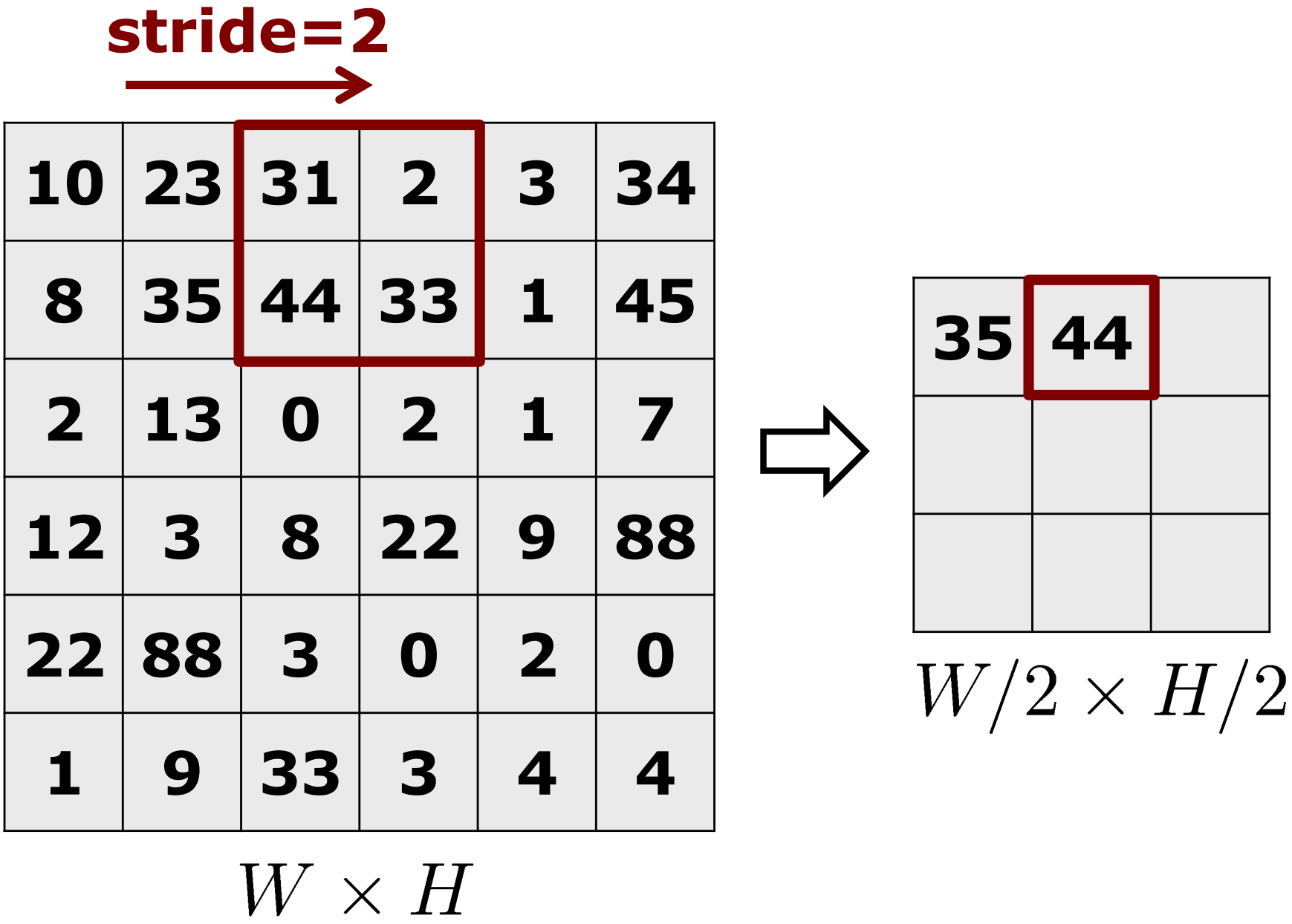
$W \times H$



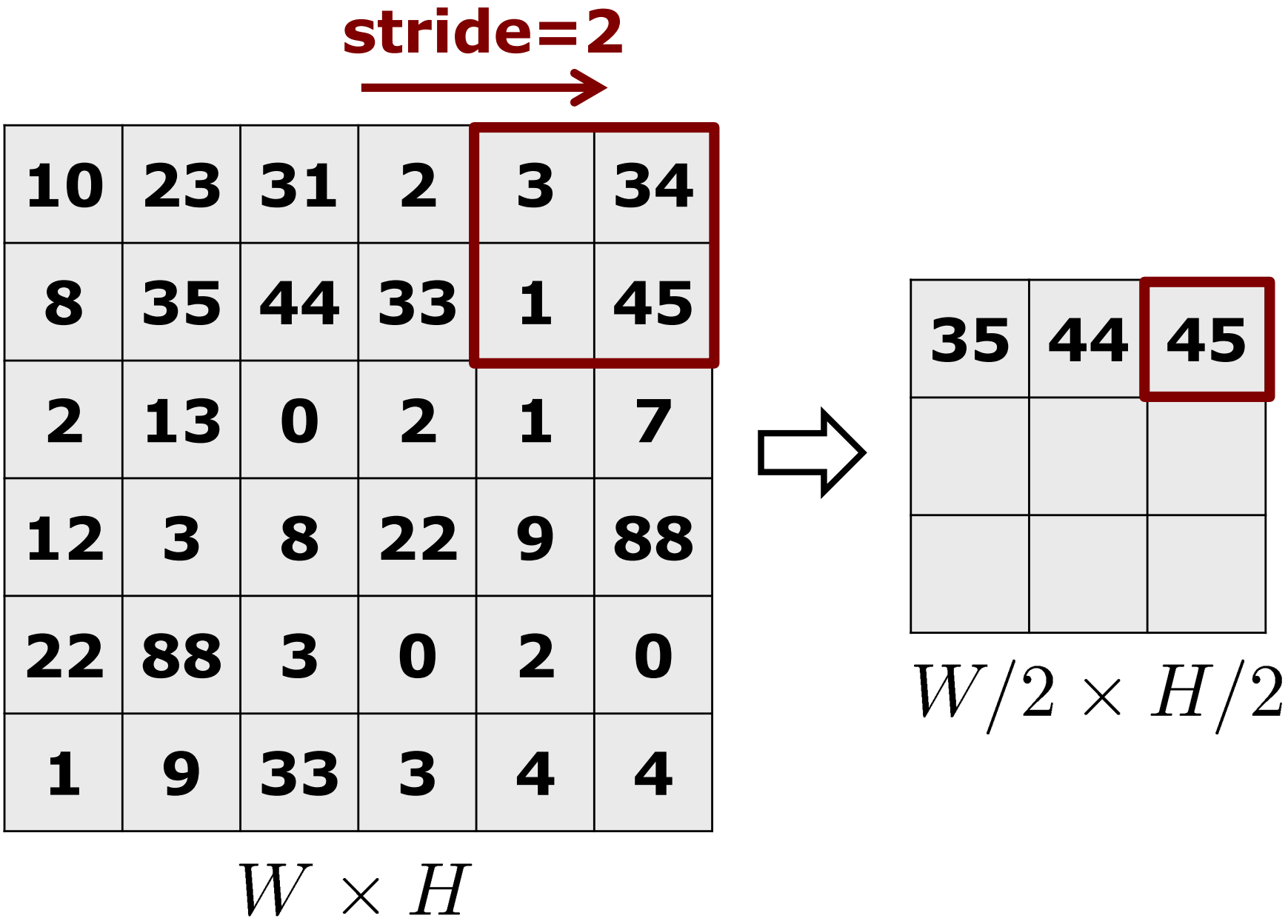
35		

$W/2 \times H/2$

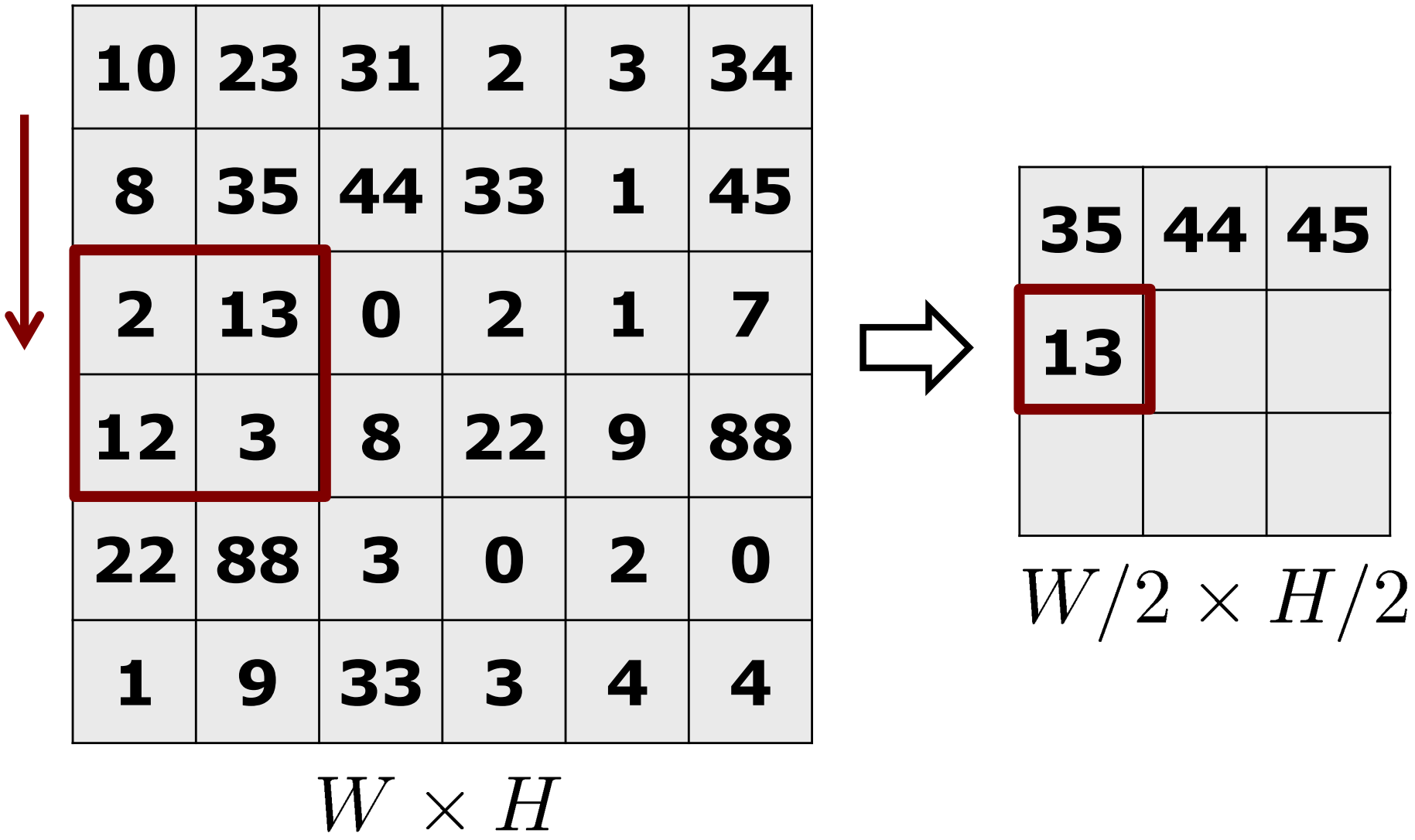
Max Pooling Example



Max Pooling Example



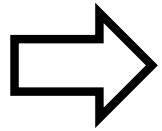
Max Pooling Example



Max Pooling Example

10	23	31	2	3	34
8	35	44	33	1	45
2	13	0	2	1	7
12	3	8	22	9	88
22	88	3	0	2	0
1	9	33	3	4	4

$W \times H$

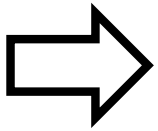


35	44	45
13	22	88
88	33	4

$W/2 \times H/2$

Max Pooling Example

10	23	31	2	3	34
8	35	44	33	1	45
2	13	0	2	1	7
12	3	8	22	9	88
22	88	3	0	2	0
1	9	33	3	4	4

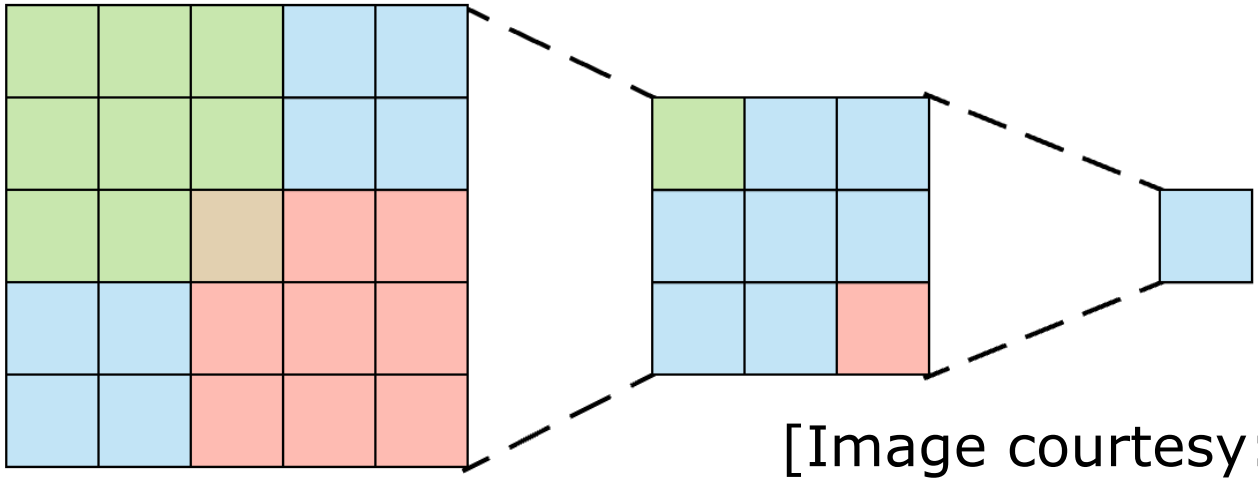


35	44	45
13	22	88
88	33	4

stride and size determine the output size

Receptive Field

- Stacking multiple pooling operations reduces W, H of the activation maps
- Elements in deeper layer are impacted by larger areas of the inputs



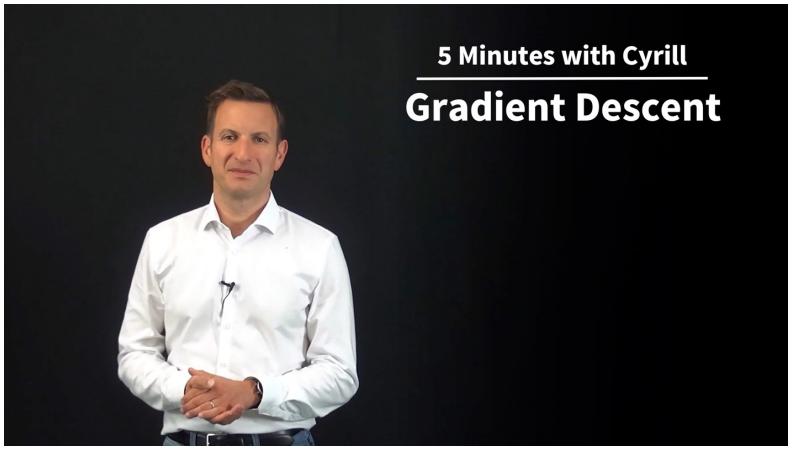
[Image courtesy: A. Dertat]

Normalization

Training CNNs

- Similar to MLPs, CNNs are trained using SDG and backpropagation
- Large number of parameters need to be determined
- Fairly large training sets are needed (end-to-end vs. given features)

SGD & Backpropagation for Training Neural Networks



https://youtu.be/4F0_V_0OO2Q

https://youtu.be/eAIAZIv2m0s

Normalization

- The first CNNs used convolutional and pooling layers in the first part
- Hard to train
- Normalizing the layers makes SGD converge faster
- Normalization of means and variance
- Somewhat unclear why it helps
- Different normalization approaches (batch, layer, instance, ...)

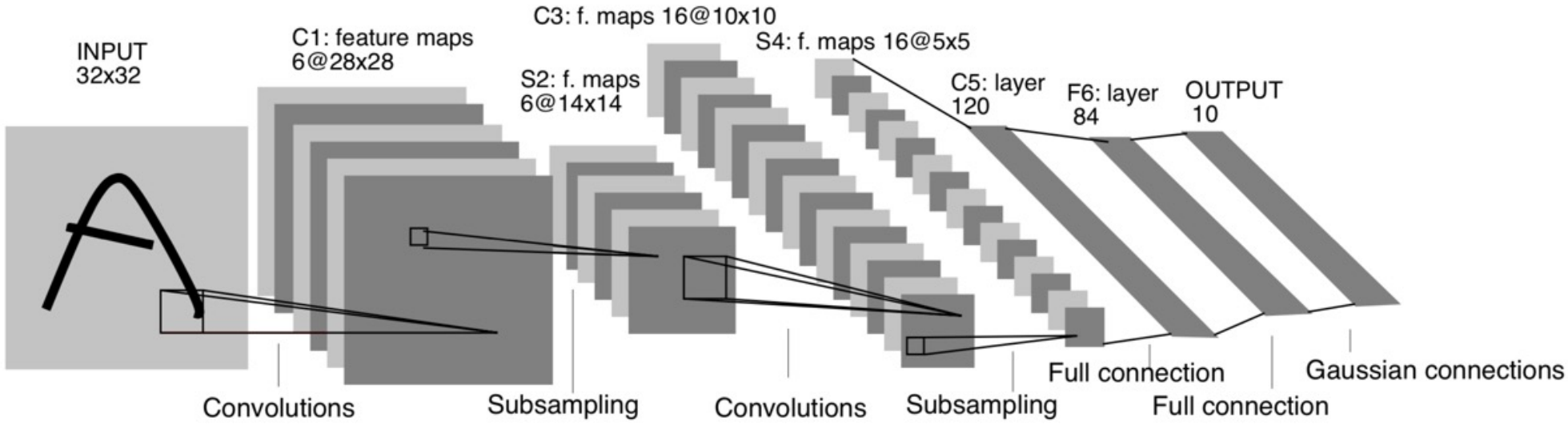
The First CNN
[LeCun et al. 1989]

LeNet-5 by LeCun et al., 1989

- First convolutional network
- Proposed by Yann LeCun et al.
- Recognition of handwritten digits
- Outperformed all other networks at that time
- 5 layers: 2 convolutional and 3 fully connected ones

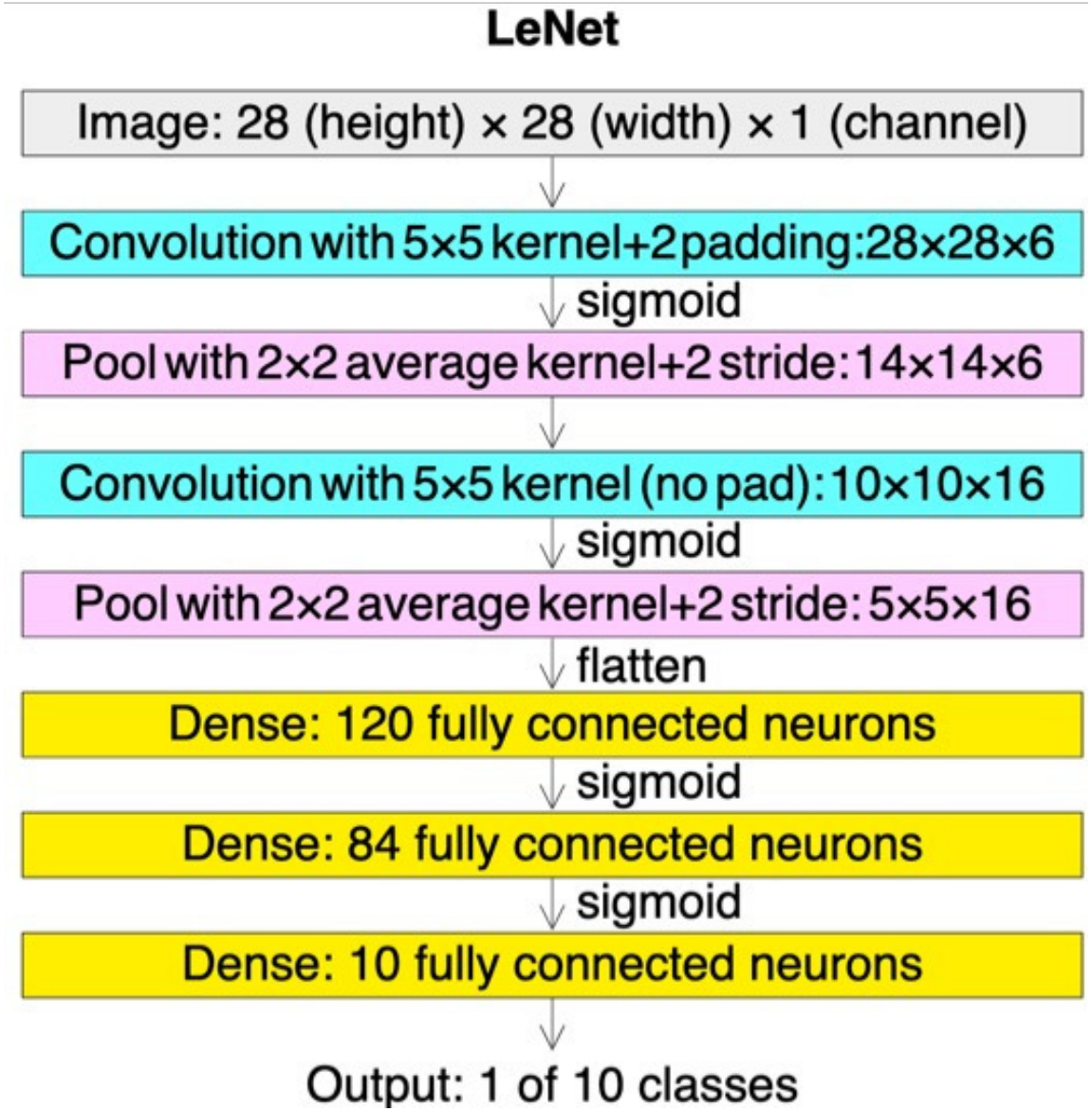
LeNet-5 by LeCun et al.

- 5 layers: 2 convolutional and 3 fully connected ones
- Each convolutional layer combines: convolutions, pooling, activation fct.



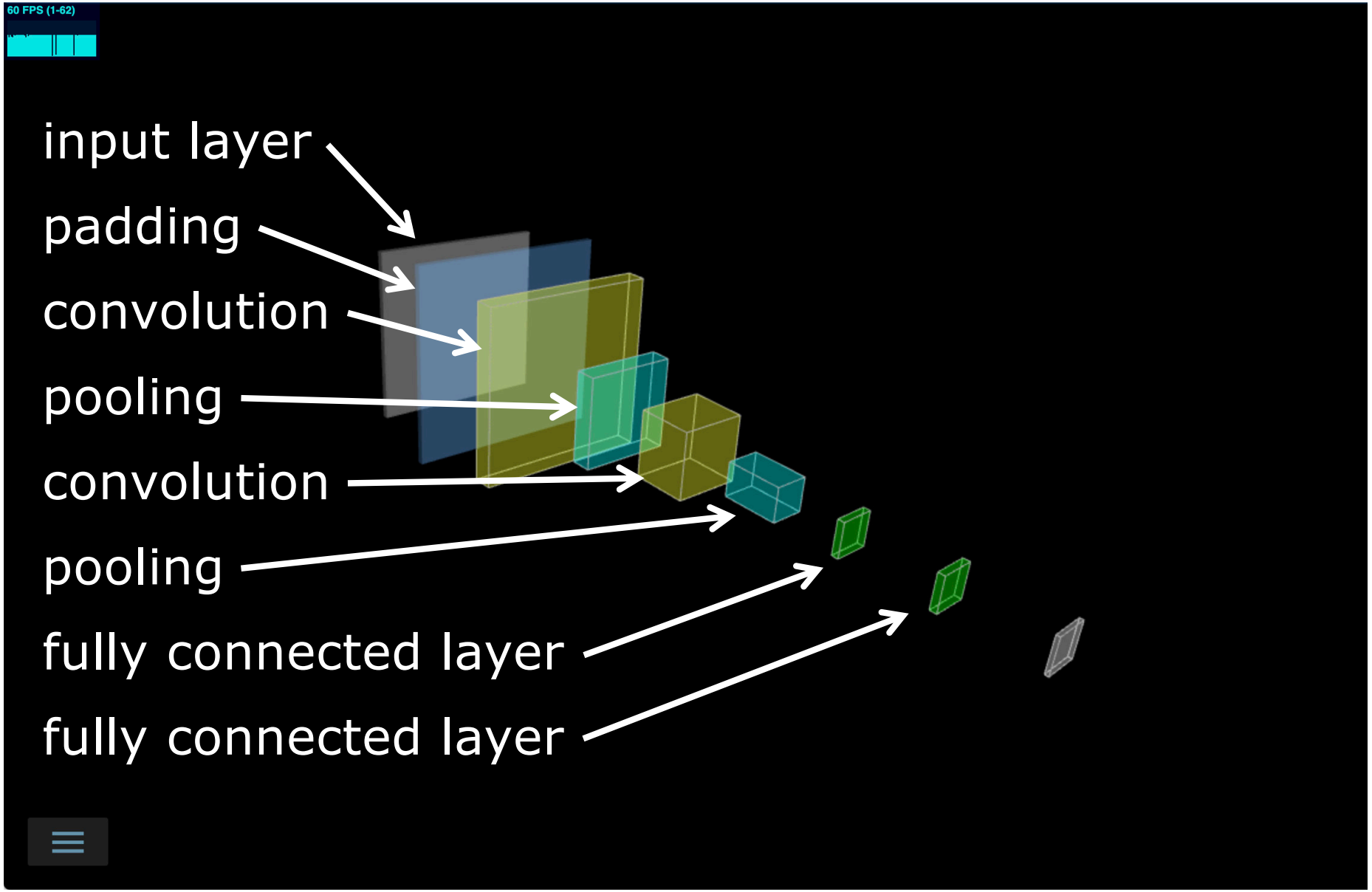
[Image courtesy: LeCun et al.]

LeNet-5 by LeCun et al.



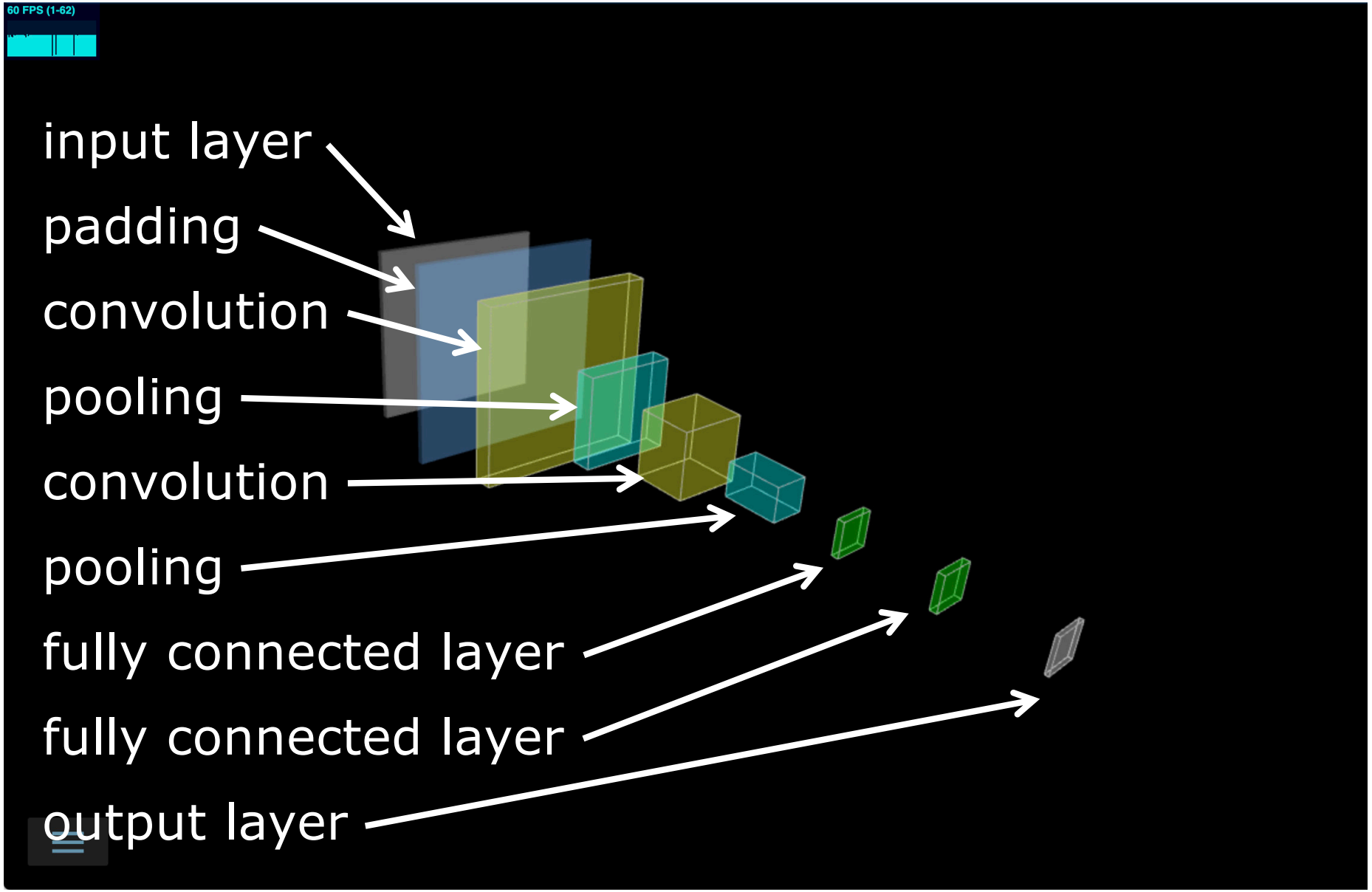
[Image courtesy: Wikipedia]

LeNet-5 by LeCun et al.



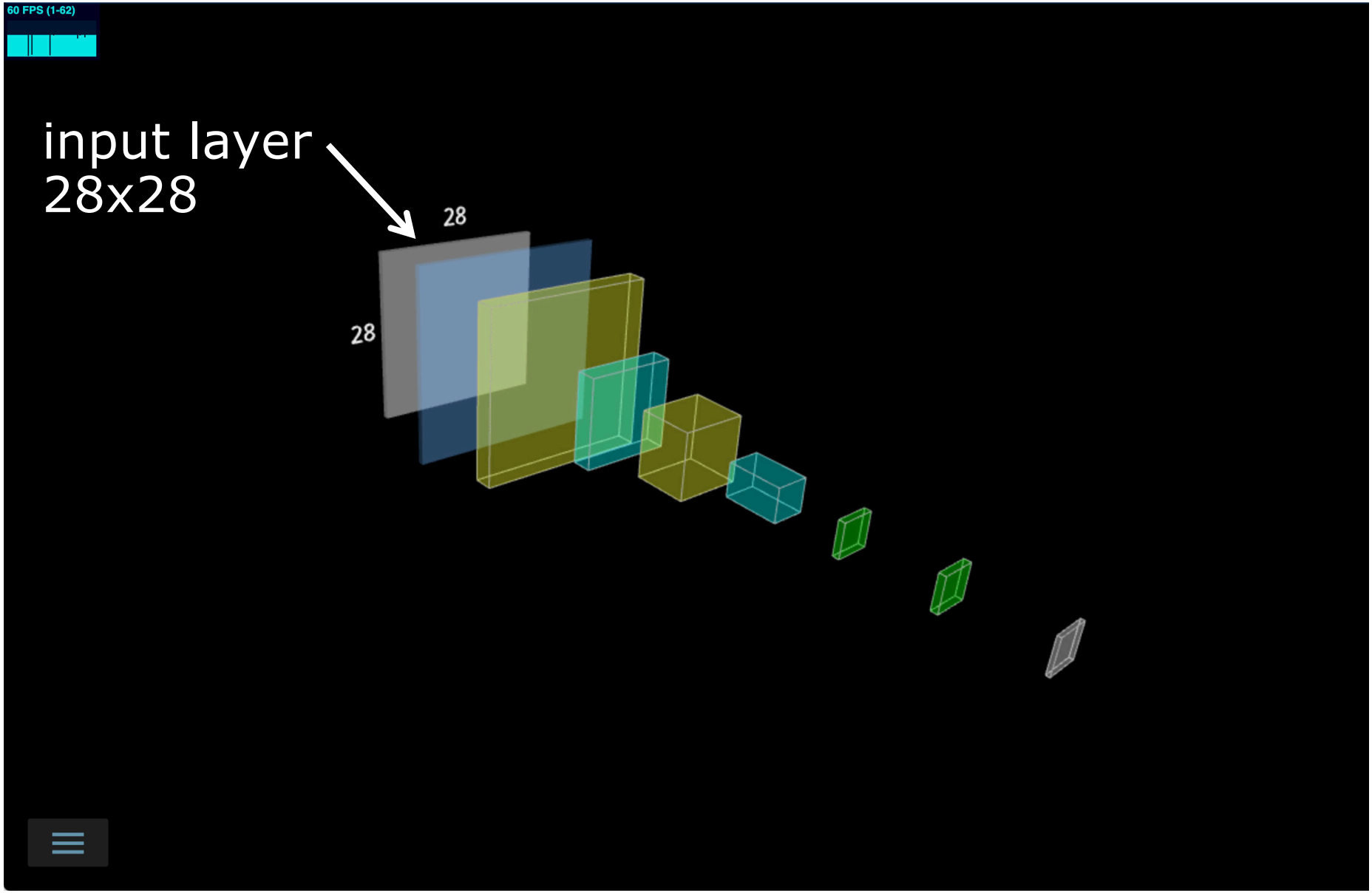
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



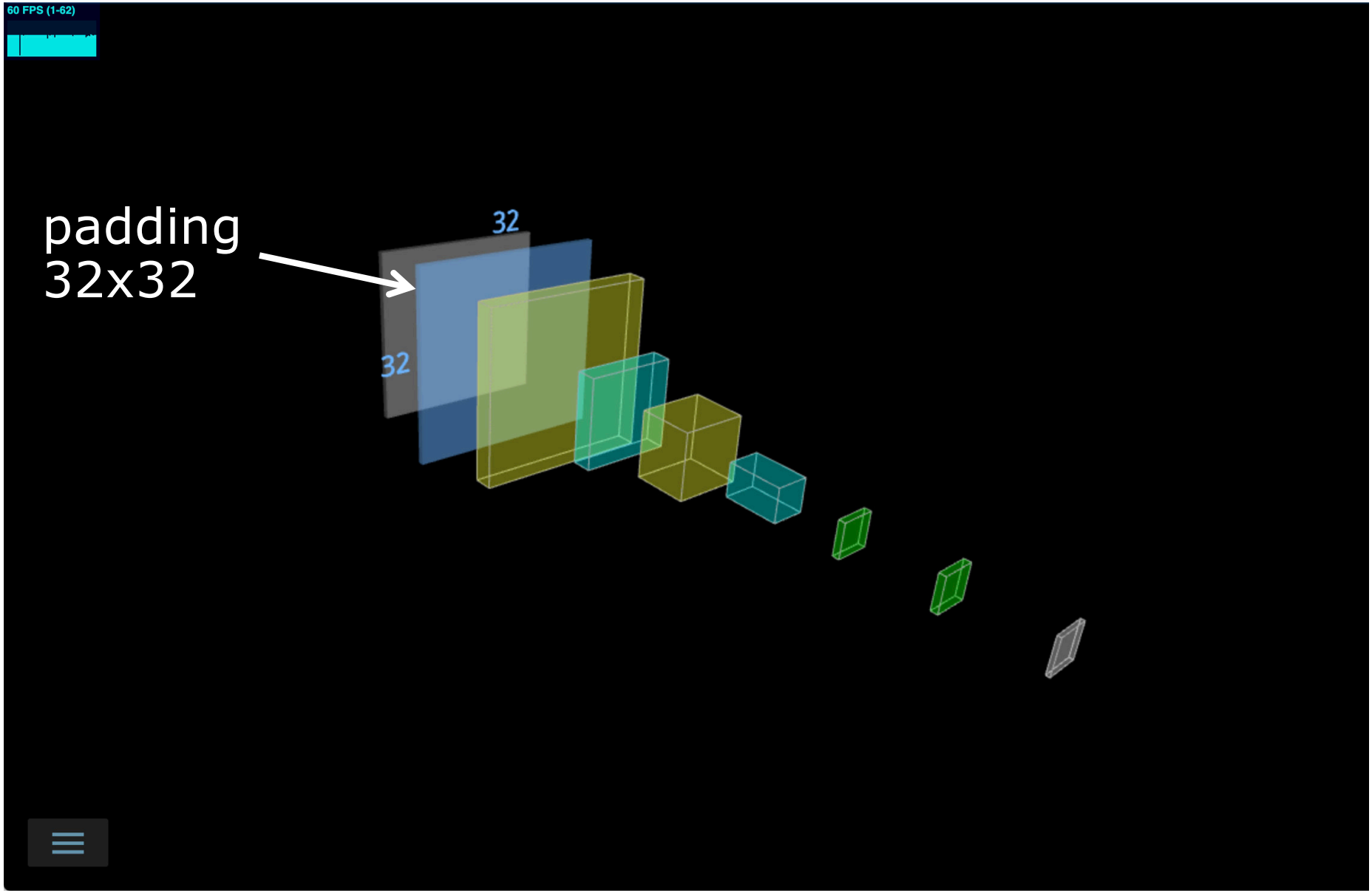
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



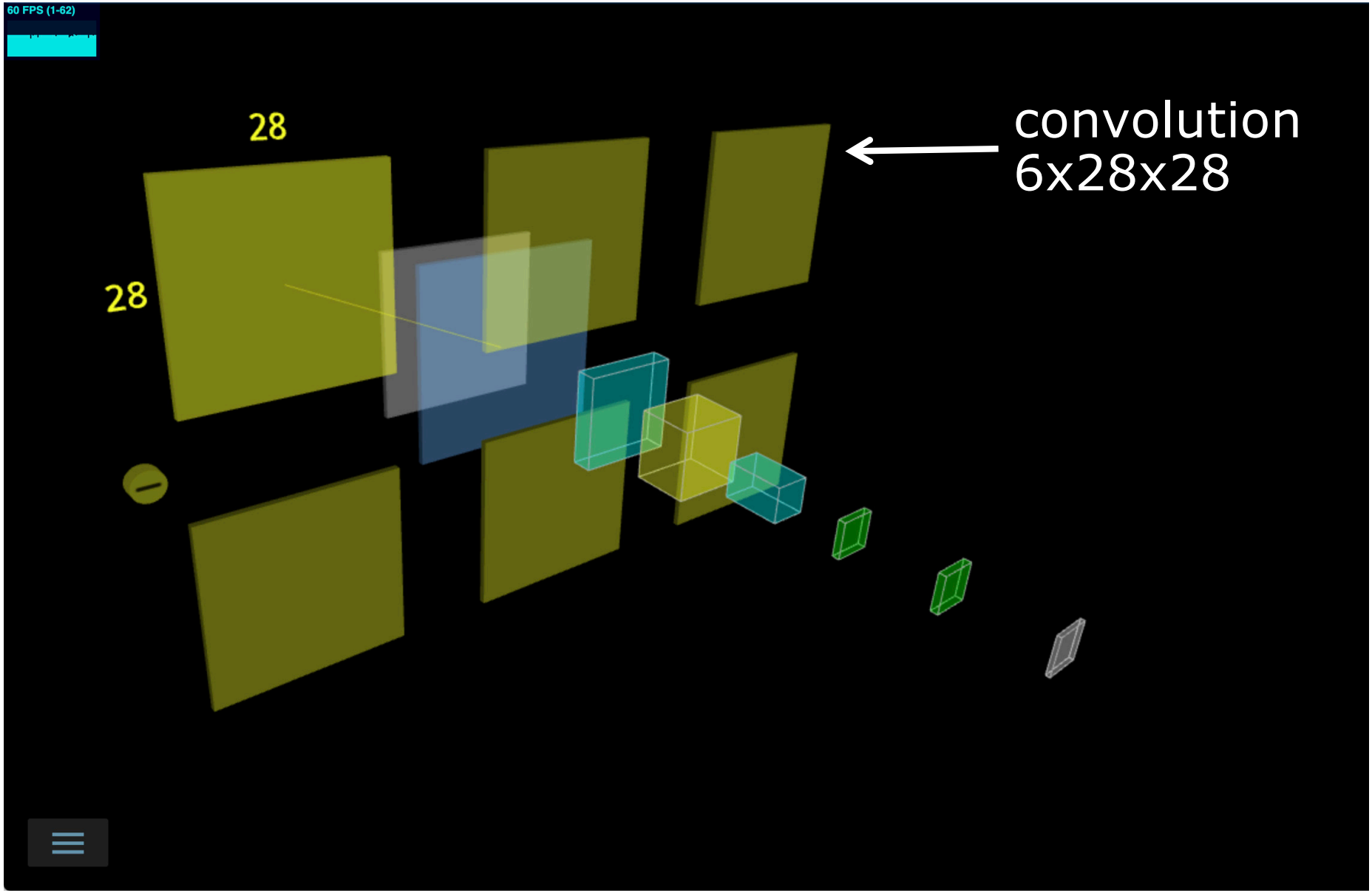
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



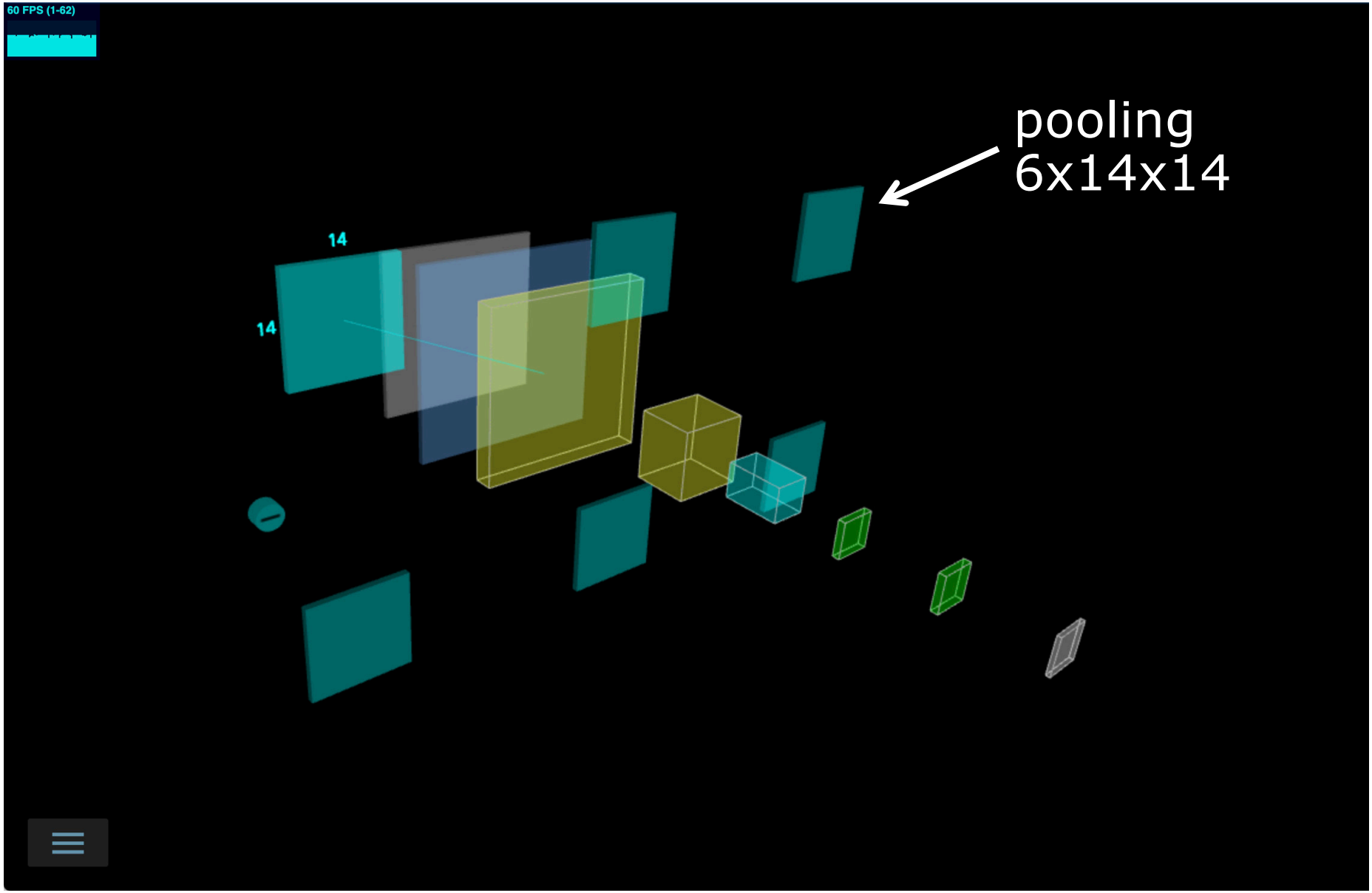
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



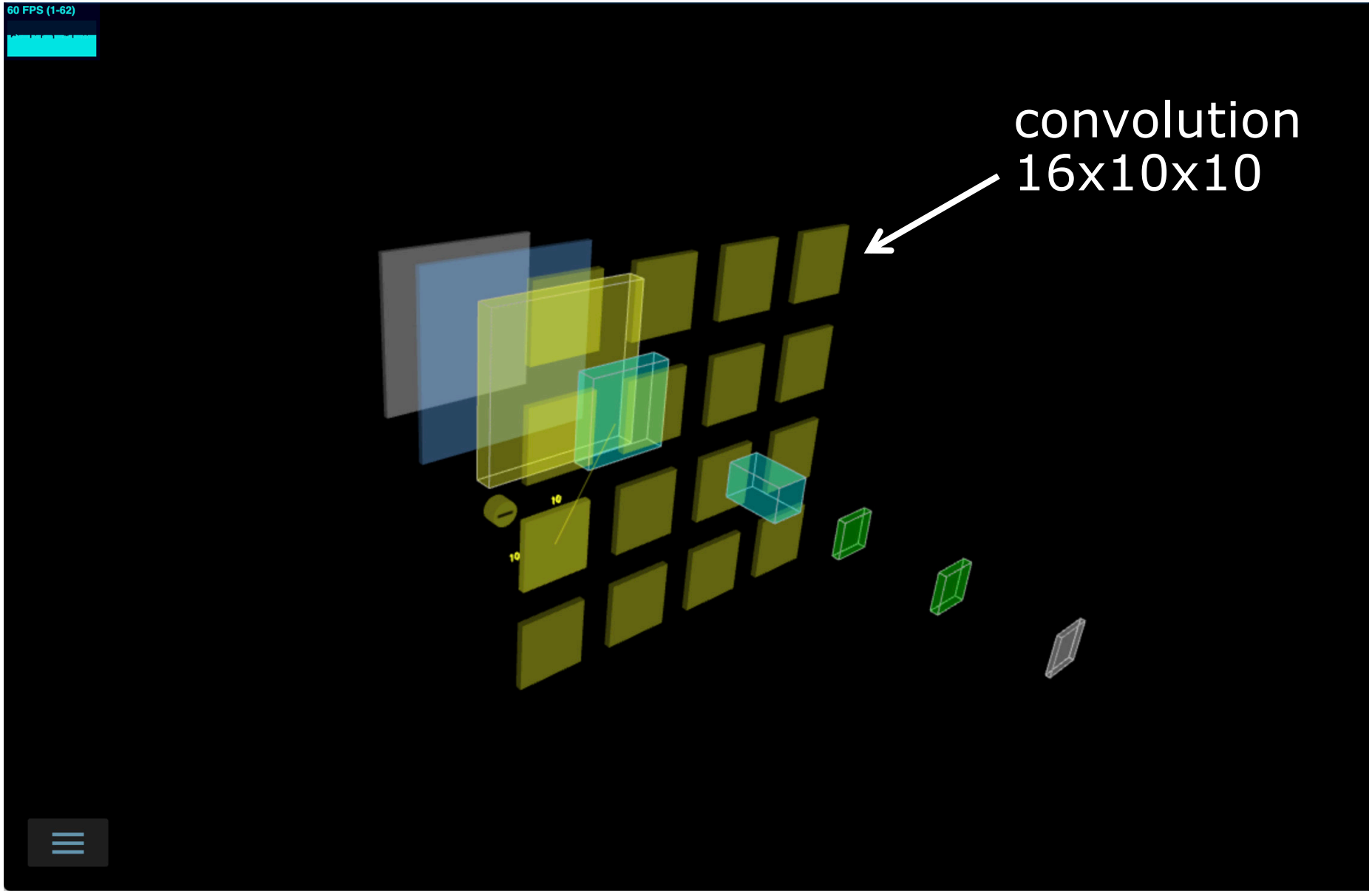
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



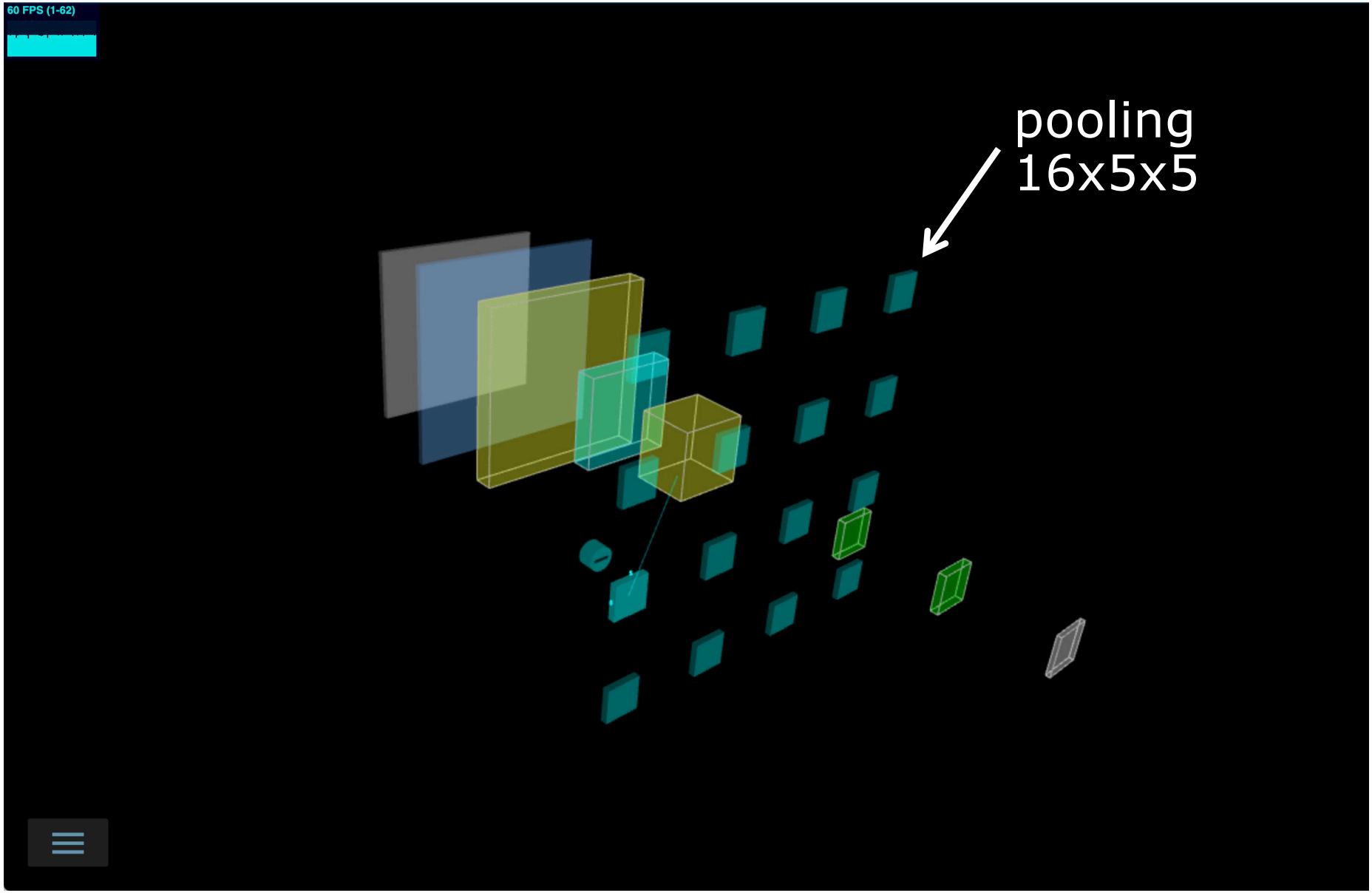
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



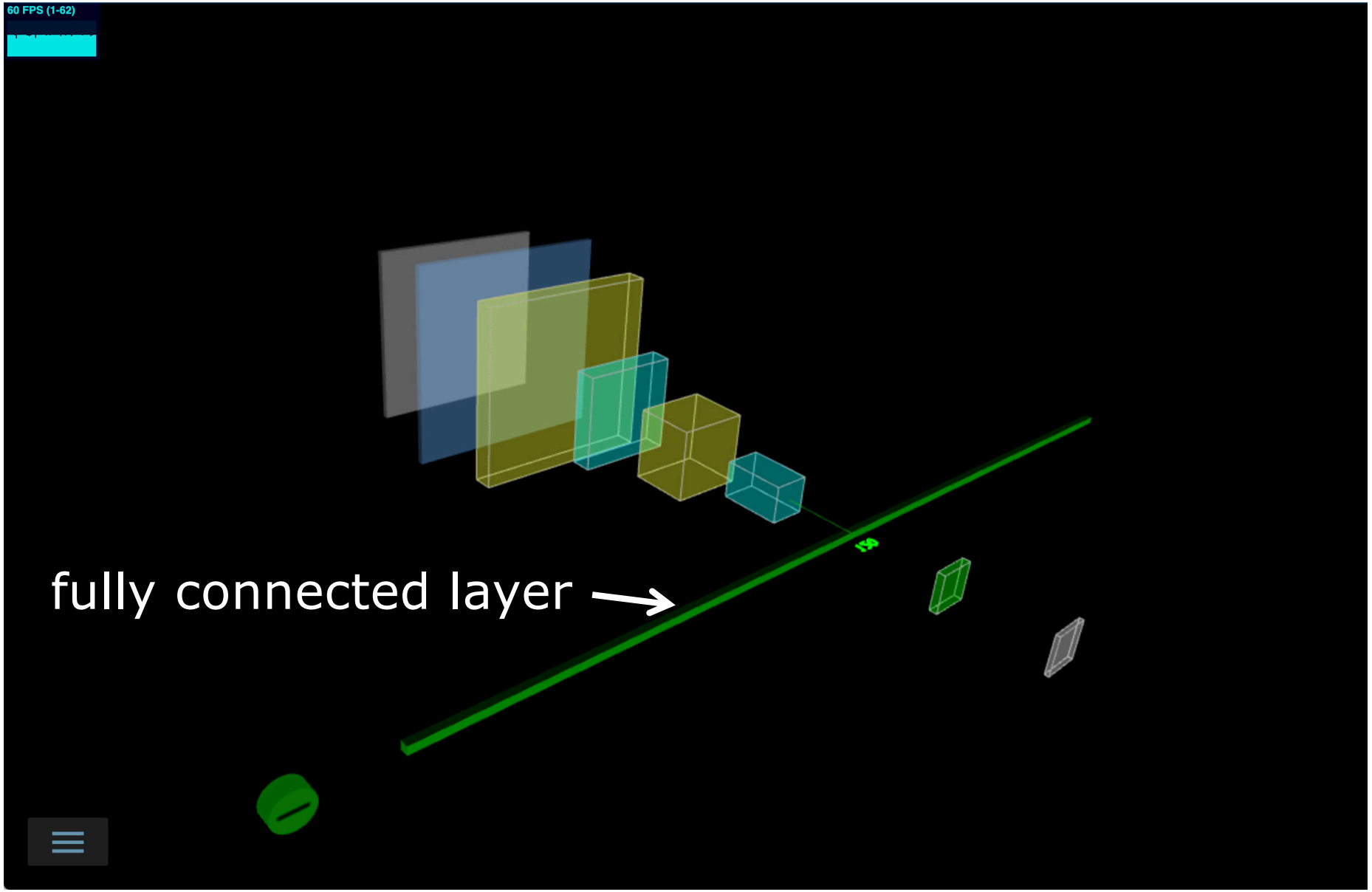
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



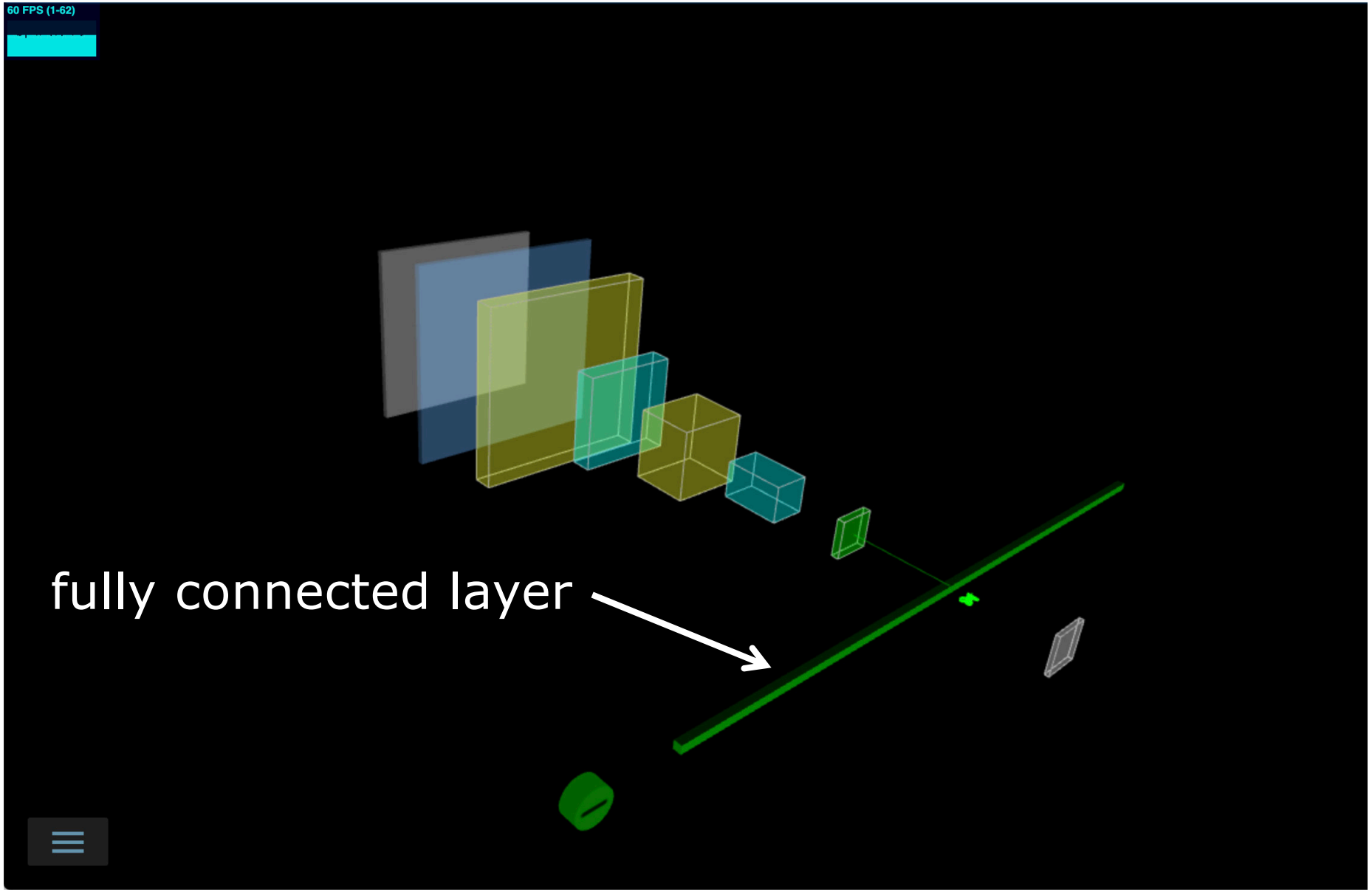
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



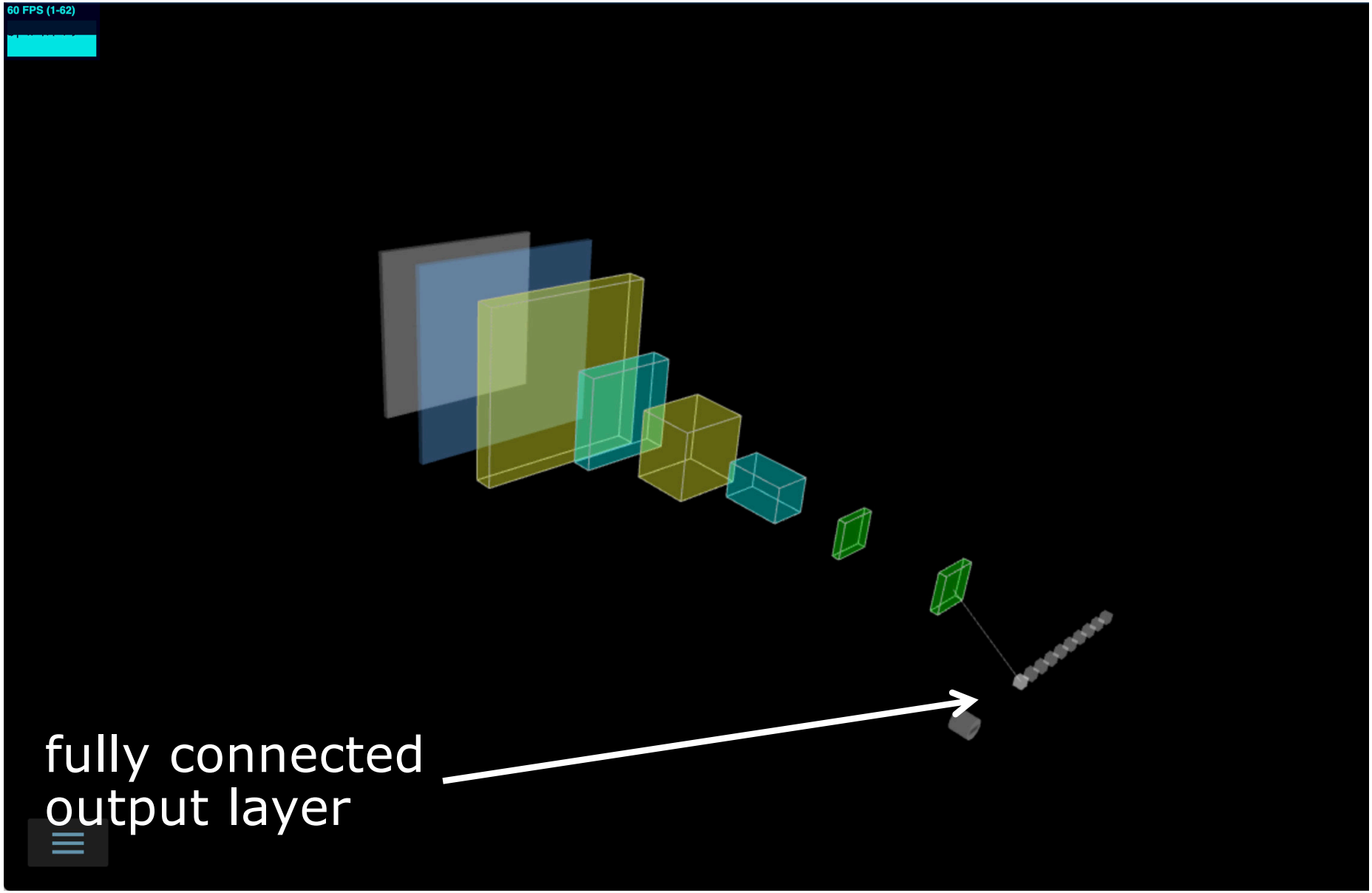
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



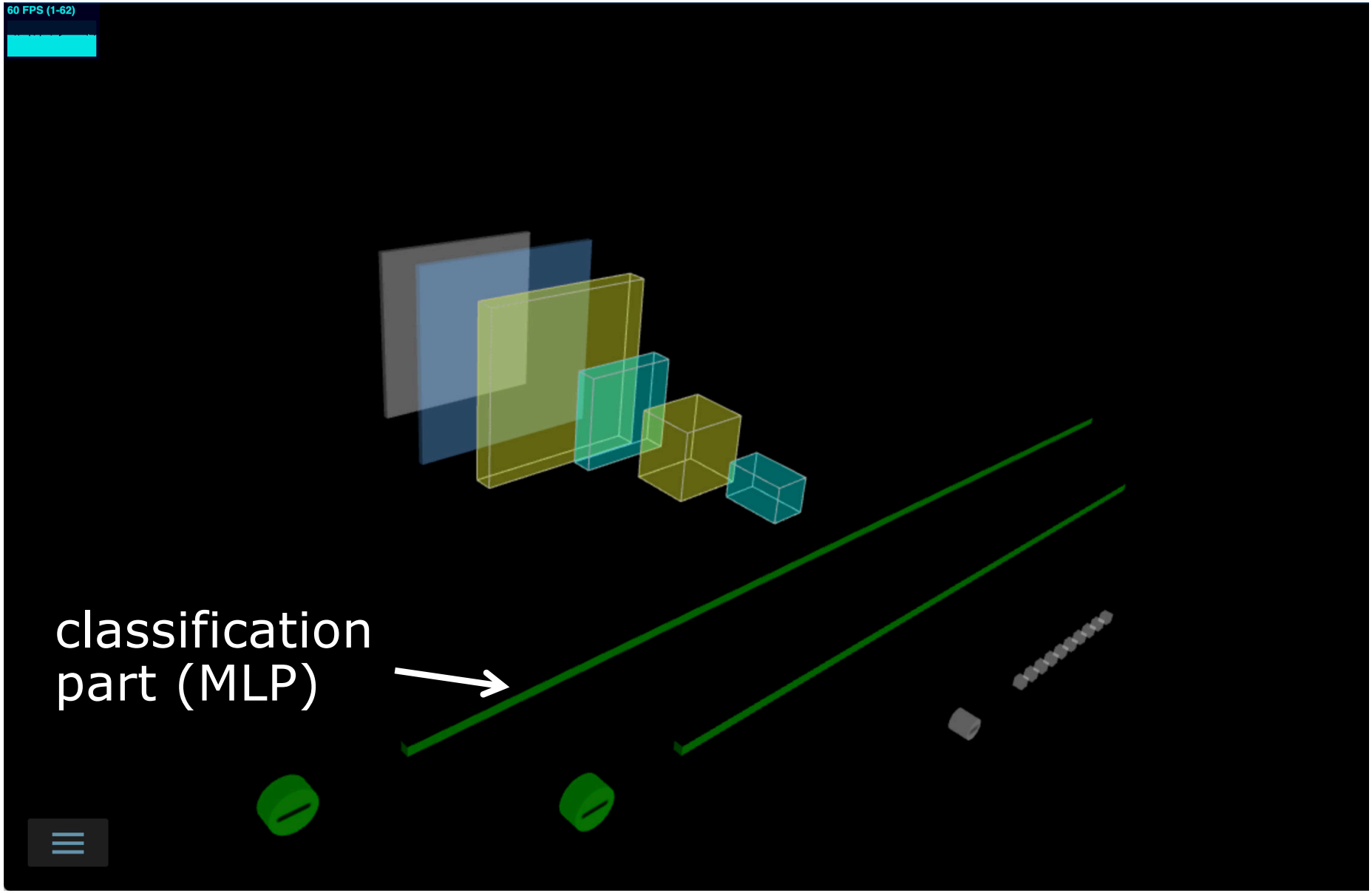
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



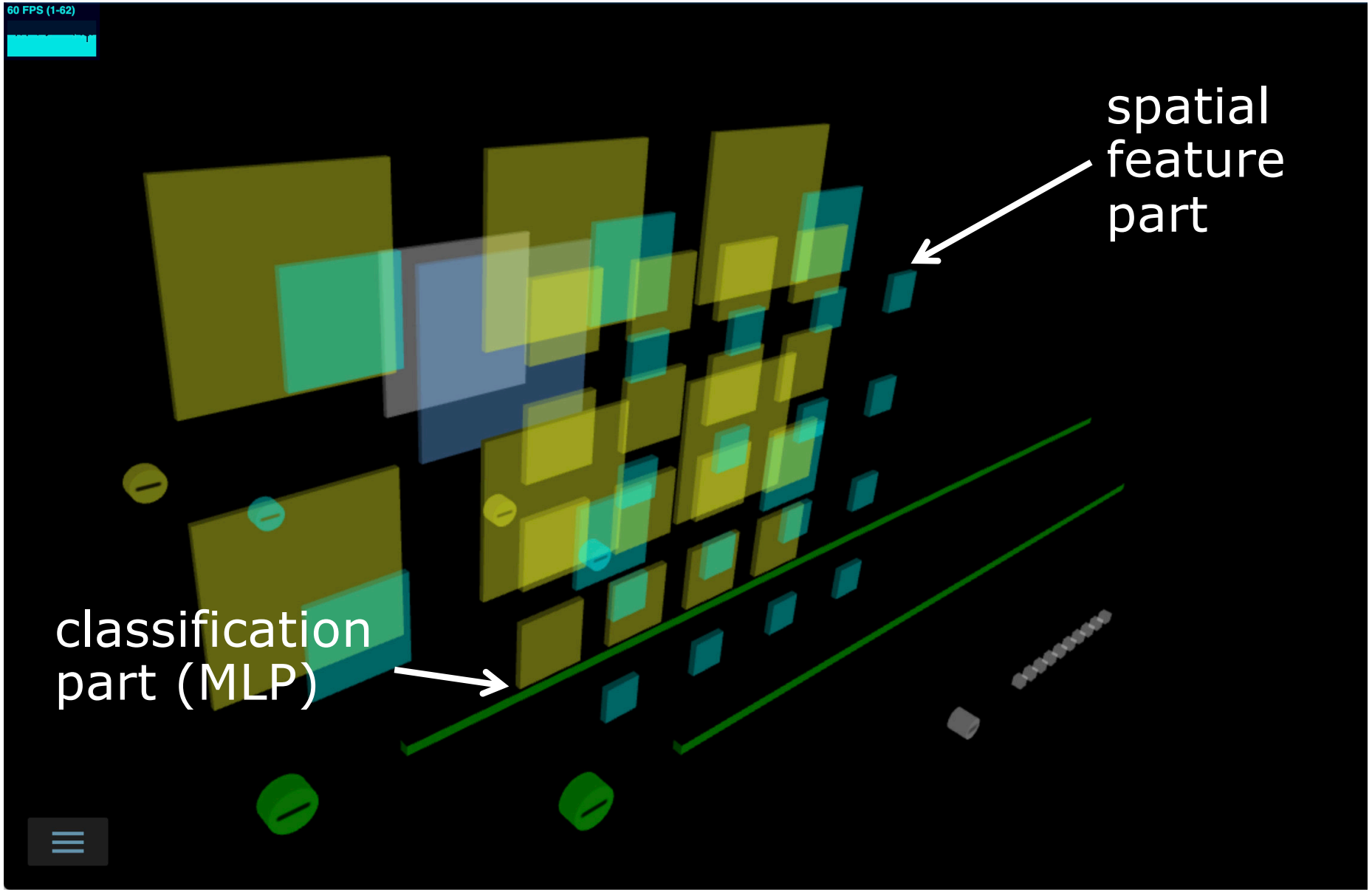
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



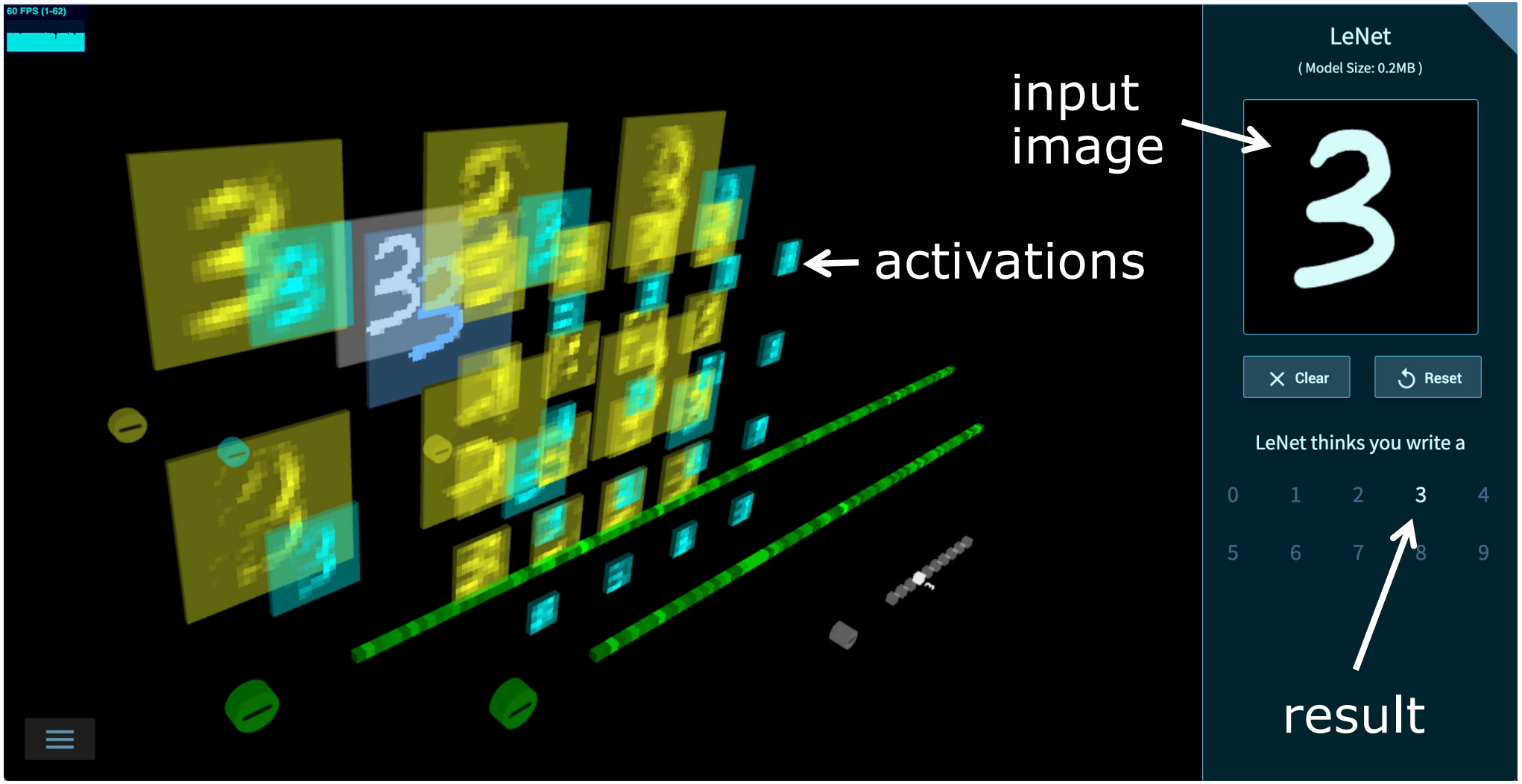
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



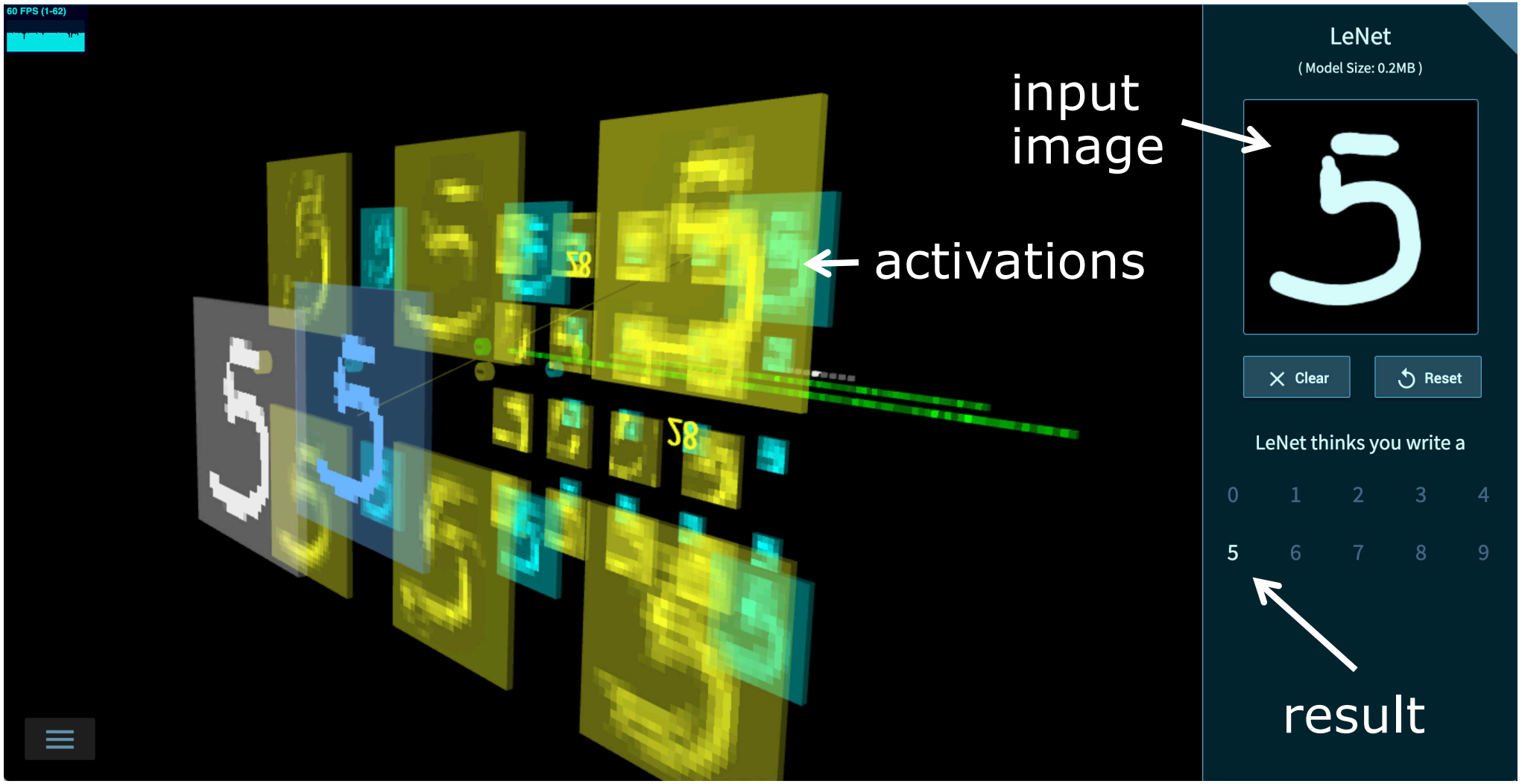
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



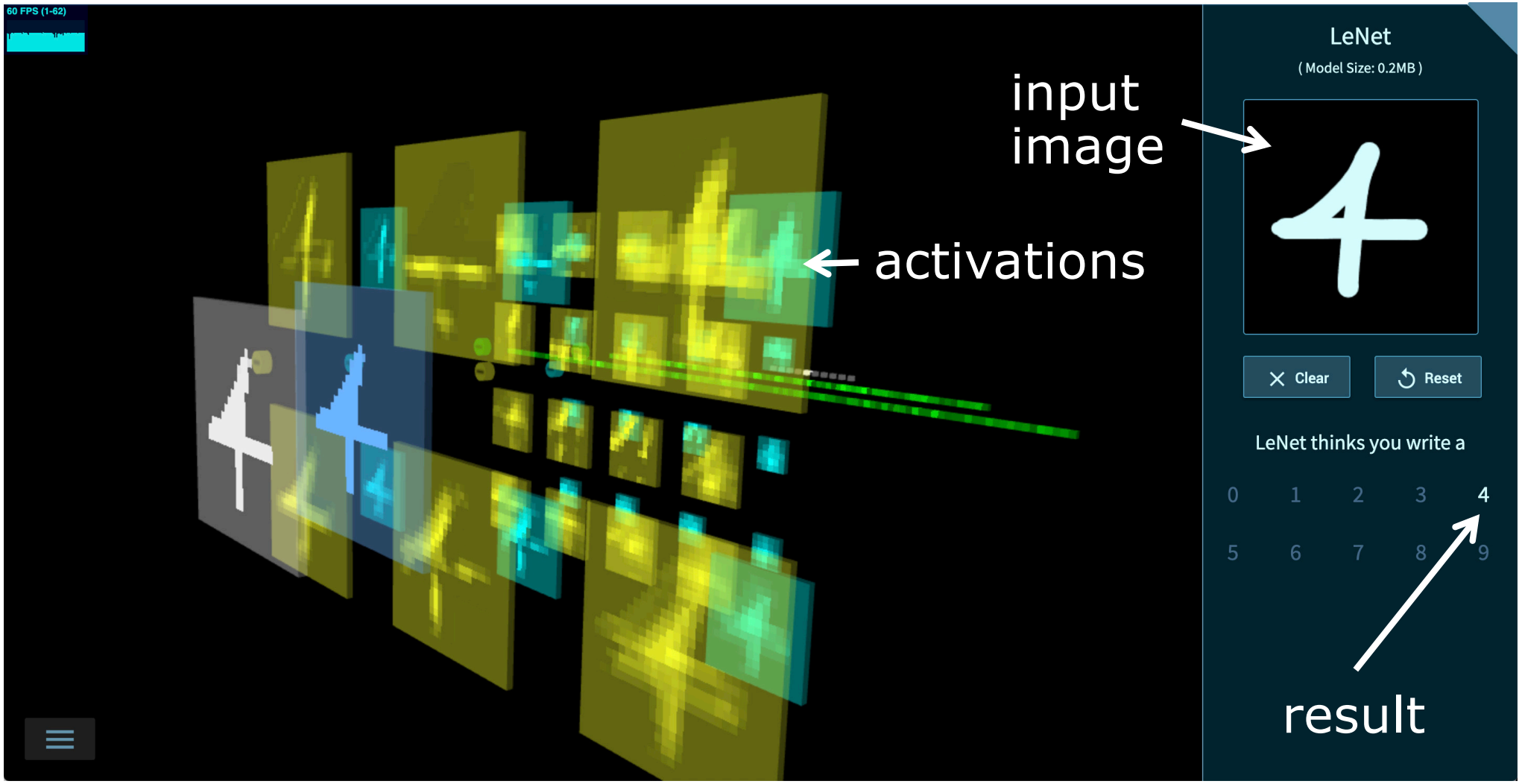
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



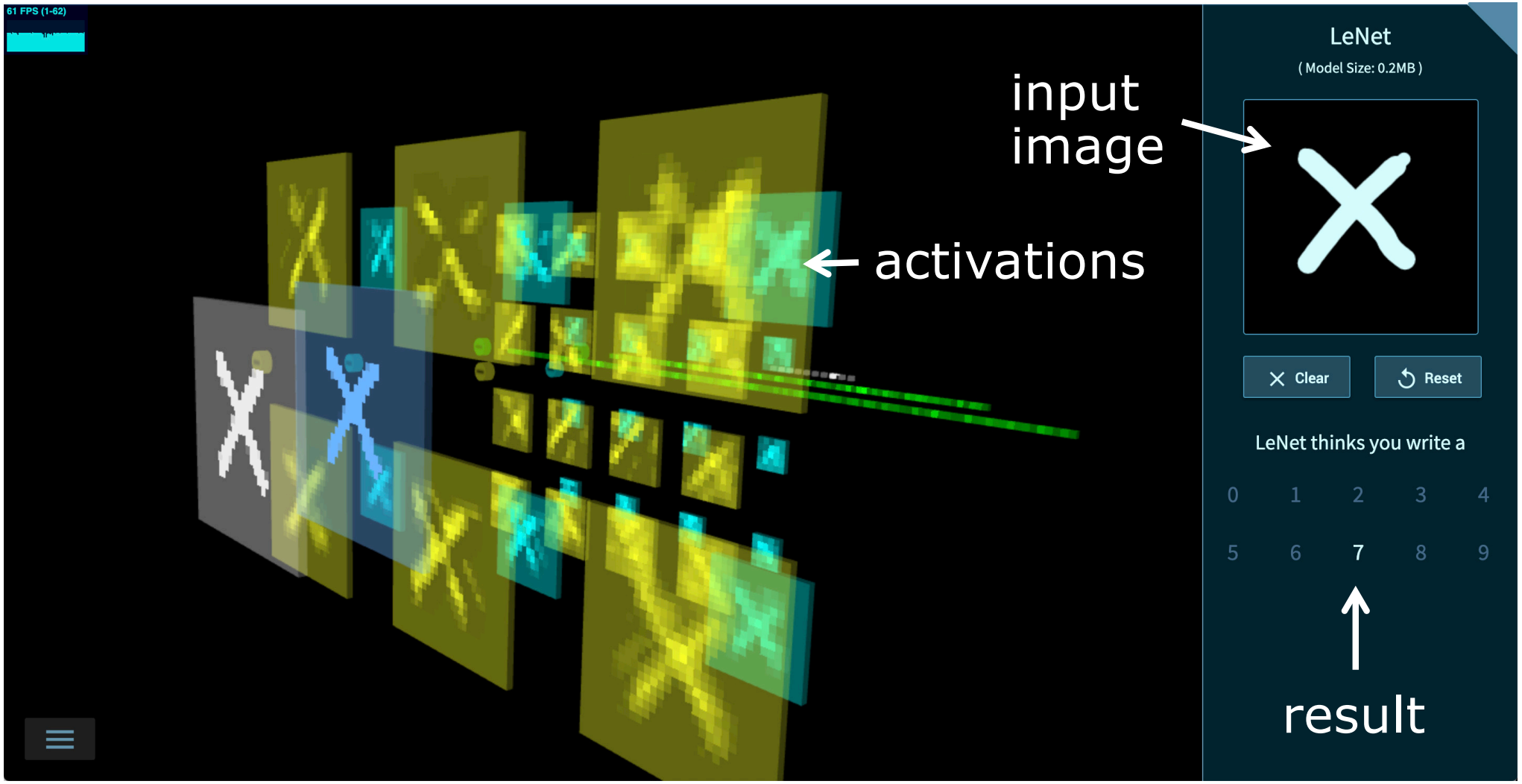
[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.



[Image generated with TensorSpace.js]

LeNet-5 by LeCun et al.

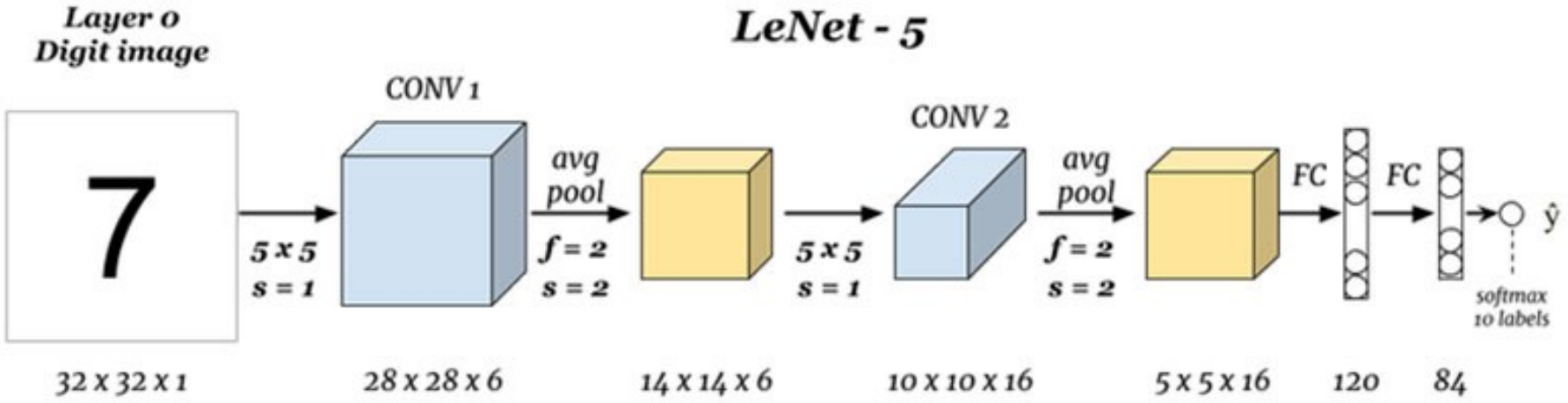


[Image generated with TensorSpace.js]

CNNs Today

- CNNs are the standard approach for image-based tasks
- Networks also also used for other inputs (e.g., point clouds)
- A large number of architectures have been proposed

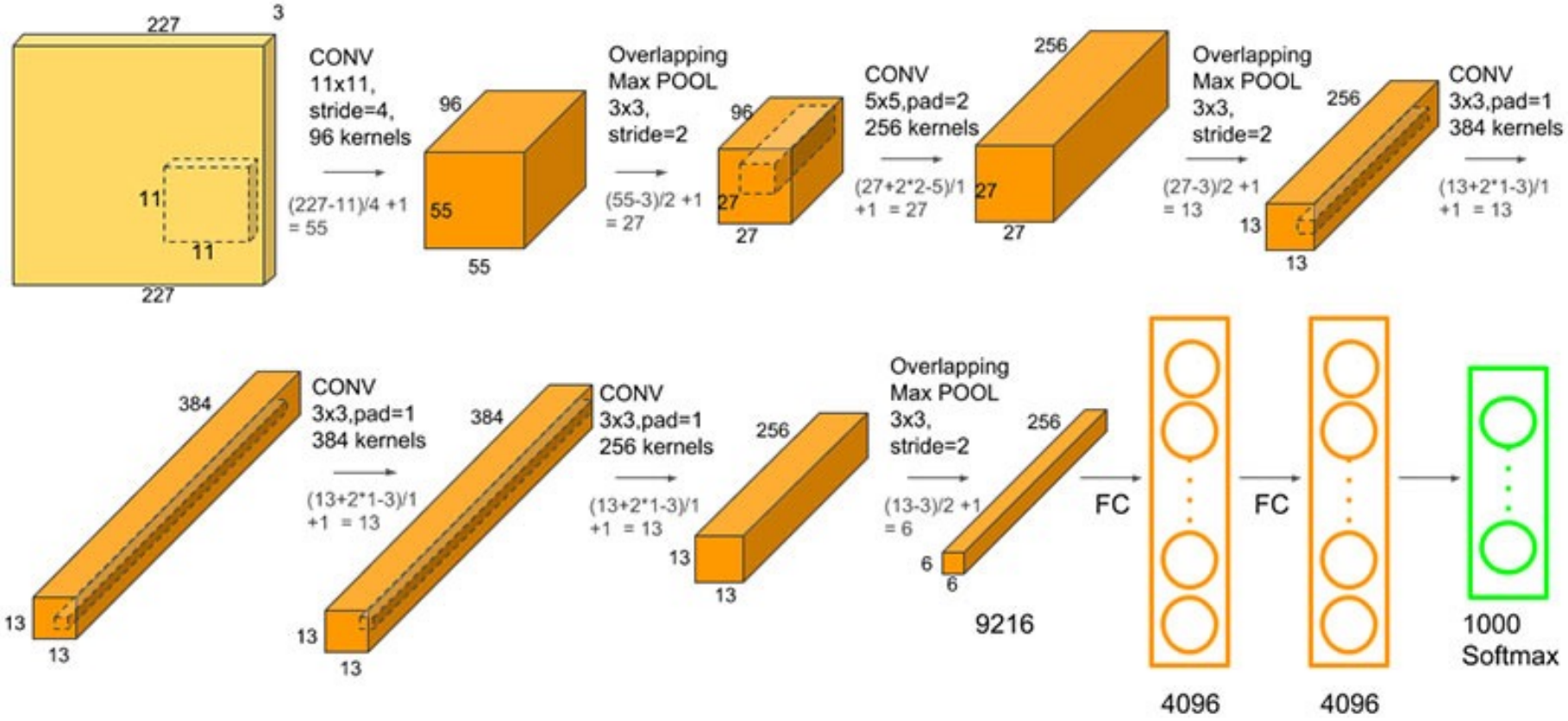
LeNet-5 (1989)



- The first CNN out there
- Used for character recognition

[Image courtesy: A. Dertat]

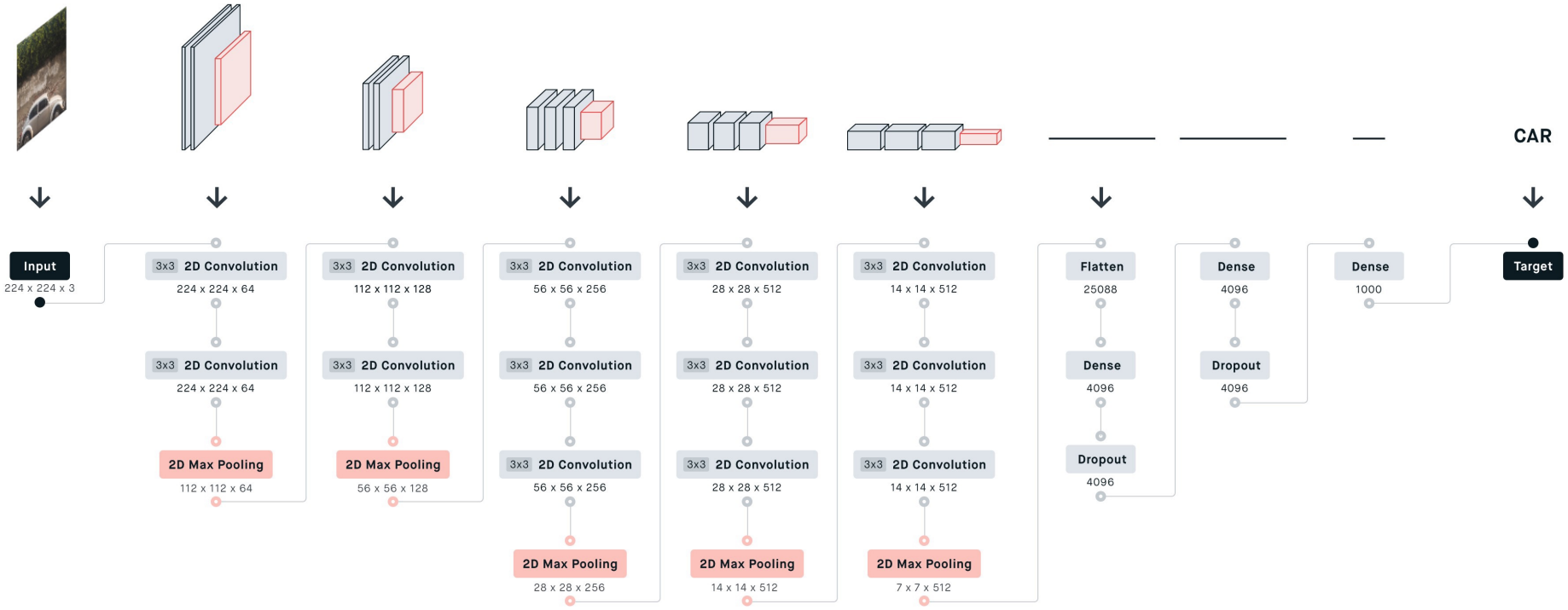
AlexNet (2012)



- Conceptually similar to LeNet-5
- Larger and deeper architecture

[Image courtesy: A. Dertat]

VGG-16/19 (2014)

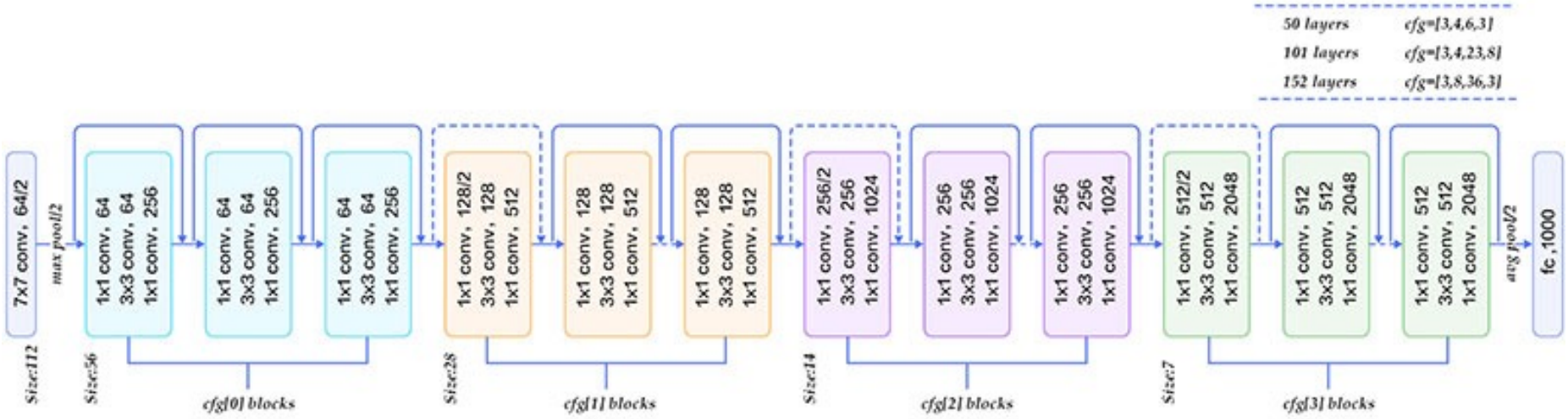


- Stacks elements from AlexNet using smaller filters
- 16/19 layers
- 138M parameters (16 layer version)

[Image courtesy: A. Dertat]

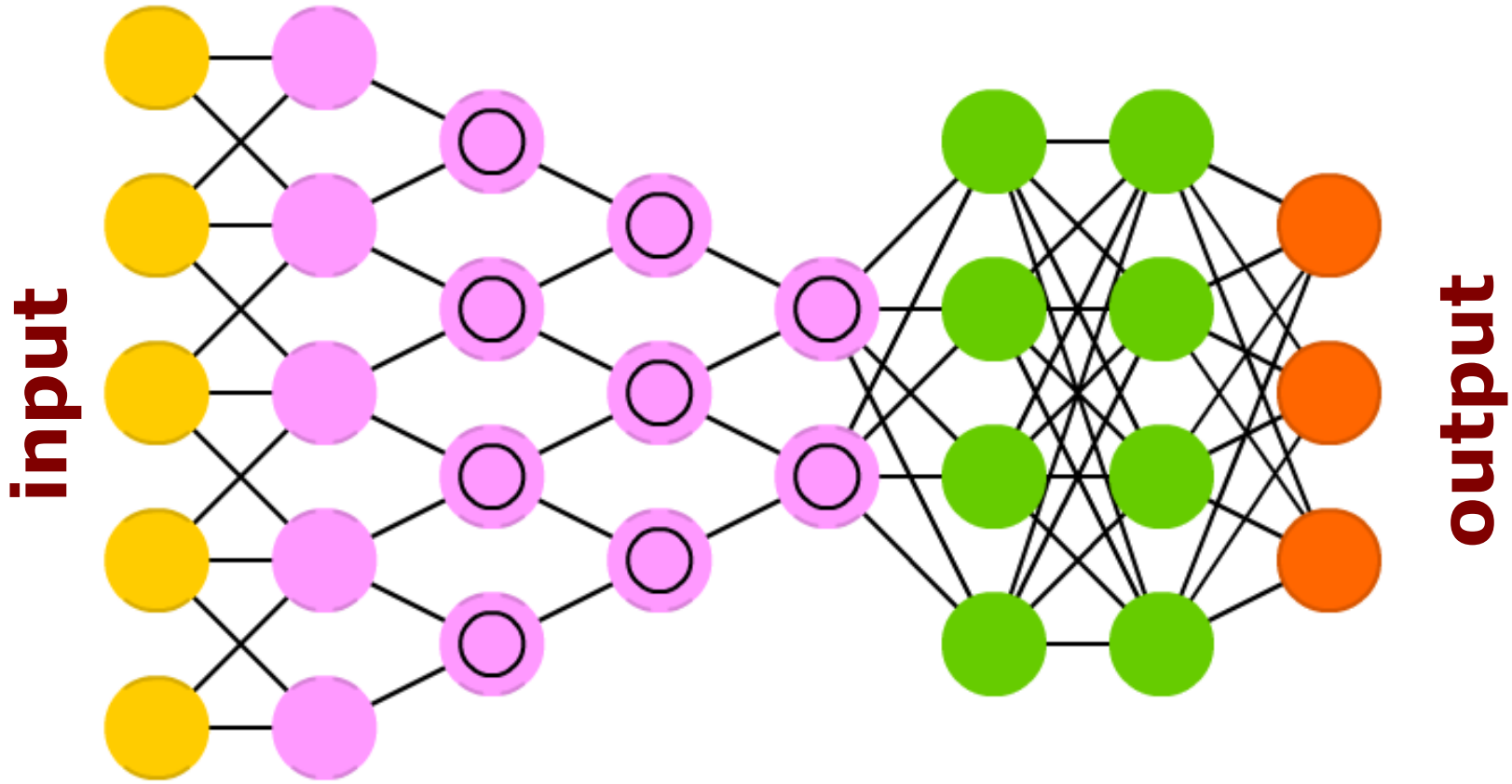
ResNet (2015)

- Very deep network: Up to 152 layers
- Consists of residual blocks



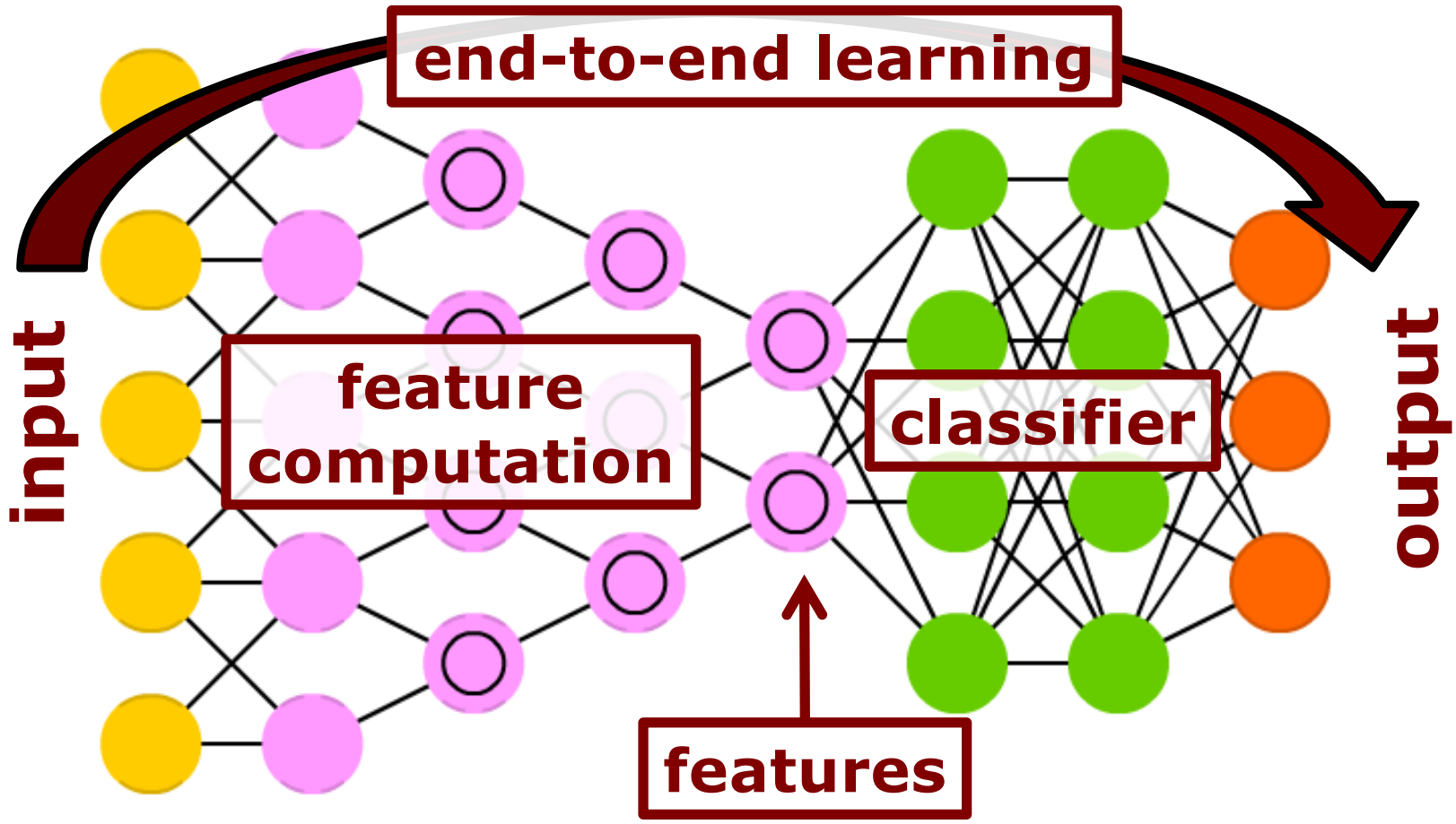
[Image courtesy: A. Dertat]

Convolutional Neural Networks



[Image courtesy: van Veen]

Convolutional Neural Networks



[Image courtesy: van Veen]

Recent Trends in CNNs

- Attention & self-attention mechanisms
- Capsule networks
- Self-supervised learning
- CNNs for 3D spatial data
- Pre-trained models
- Few-shot learning
- Transfer learning & fine-tuning

CNN Summary

- This lecture was a brief overview on convolutional neural networks (CNNs)
- Standard tool today for vision tasks
- Layers combining convolutional blocks, pooling, and normalization
- Classification layers at the end
- End-to-end trainable networks (input image to output)

Literature & Resources

- Goodfellow, Bengio, Courville: “Deep Learning”
  https://www.deeplearningbook.org/
- Online Book by Michael Nielsen, Chapter 1:
  http://neuralnetworksanddeeplearning.com/chap1.html
- Online book by Deisenroth, Faisal, Ong:
  Mathematics for Machine Learning
  https://mml-book.github.io/
- Alpaydin, Introduction to Machine Learning
- UMich NN Lecture by Johnson
- Standford AI Lectures by Li et al.

Slide Information

- The slides have been created by Cyrill Stachniss as part of the photogrammetry and robotics courses.
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**
- I took a lot of inspiration from lectures given by Justin Johnson (UMich, Stanford) and Fei-Fei Li (Stanford).
- If you are a university lecturer, feel free to use the course material. If you adapt the course material, please make sure that you keep the acknowledgements to others and please acknowledge me as well. To satisfy my own curiosity, please send me email notice if you use my slides.

Cyrill Stachniss, cyrill.stachniss@igg.uni-bonn.de