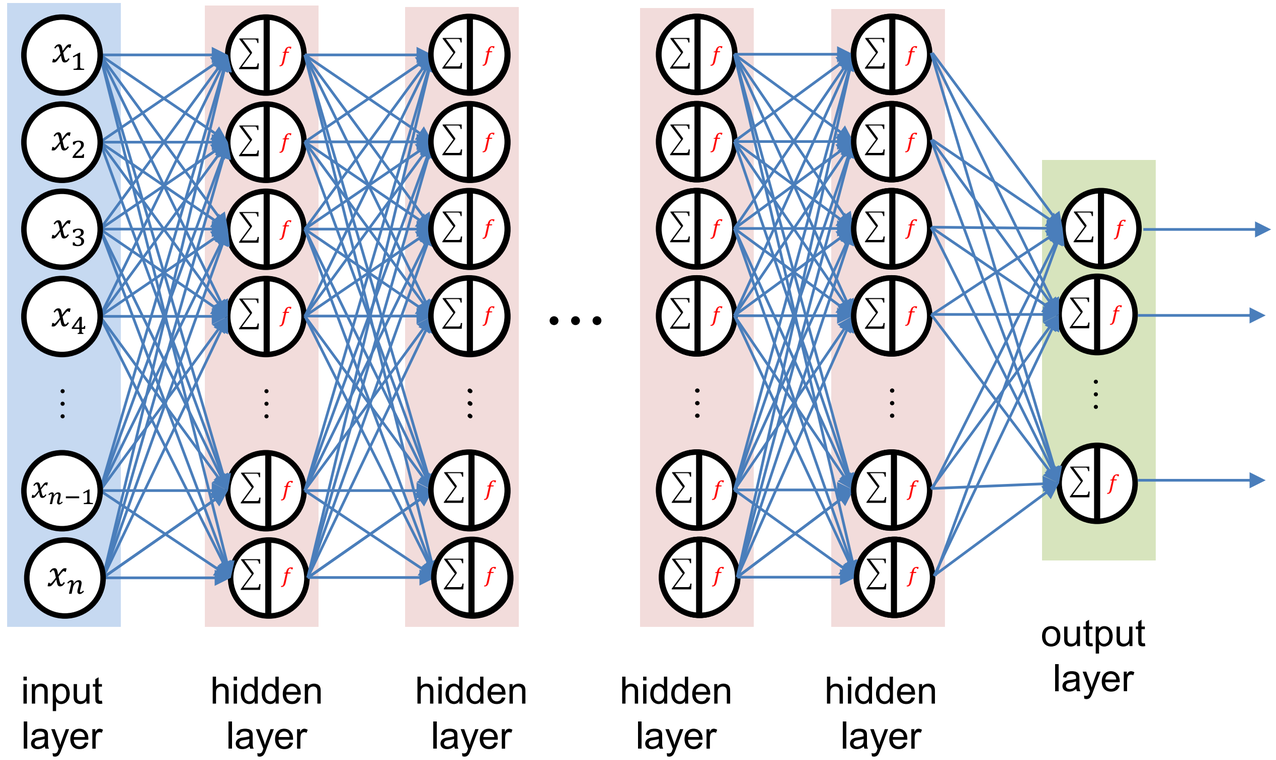
Aprendizado Profundo (Deep Learning)

Convolutional Neural Networks

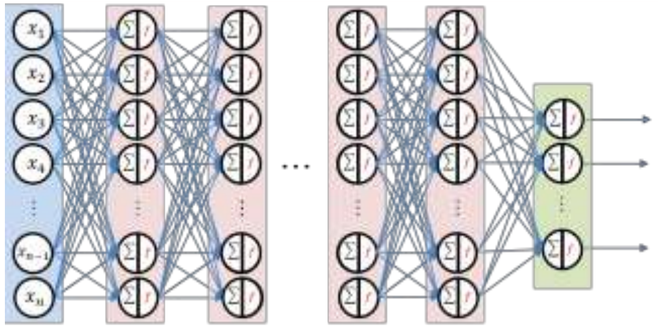
Dario Oliveira (dario.oliveira@fgv.br)

Multilayer Perceptron

$$f(\boldsymbol{W}^T\,\boldsymbol{x}) = f_d\left(\boldsymbol{W}_d^T f_{d-1}\left(\boldsymbol{W}_{d-1}^T f_{d-2}\left(\ldots \boldsymbol{W}_2^T f_1(\boldsymbol{W}_1^T \boldsymbol{x})\right)\right)\right)$$



Problem with the MLP



- Each neuron is **fully connected** to all neurons in the previous layer.
- Neurons in a single layer are independent.
- Neurons in a single layer do not share any connections.
- Consider a MLP for classifying a 200×200 RGB image.
- The number of parameters in a single fully connected layer will be:

$$14.4 \times 10^9$$

Content

- Convolution Operation
- Convolution Layers
- Pooling Layers
- Visualization of Feature Maps

Digital Image

Digital Image

Red

Digital Image

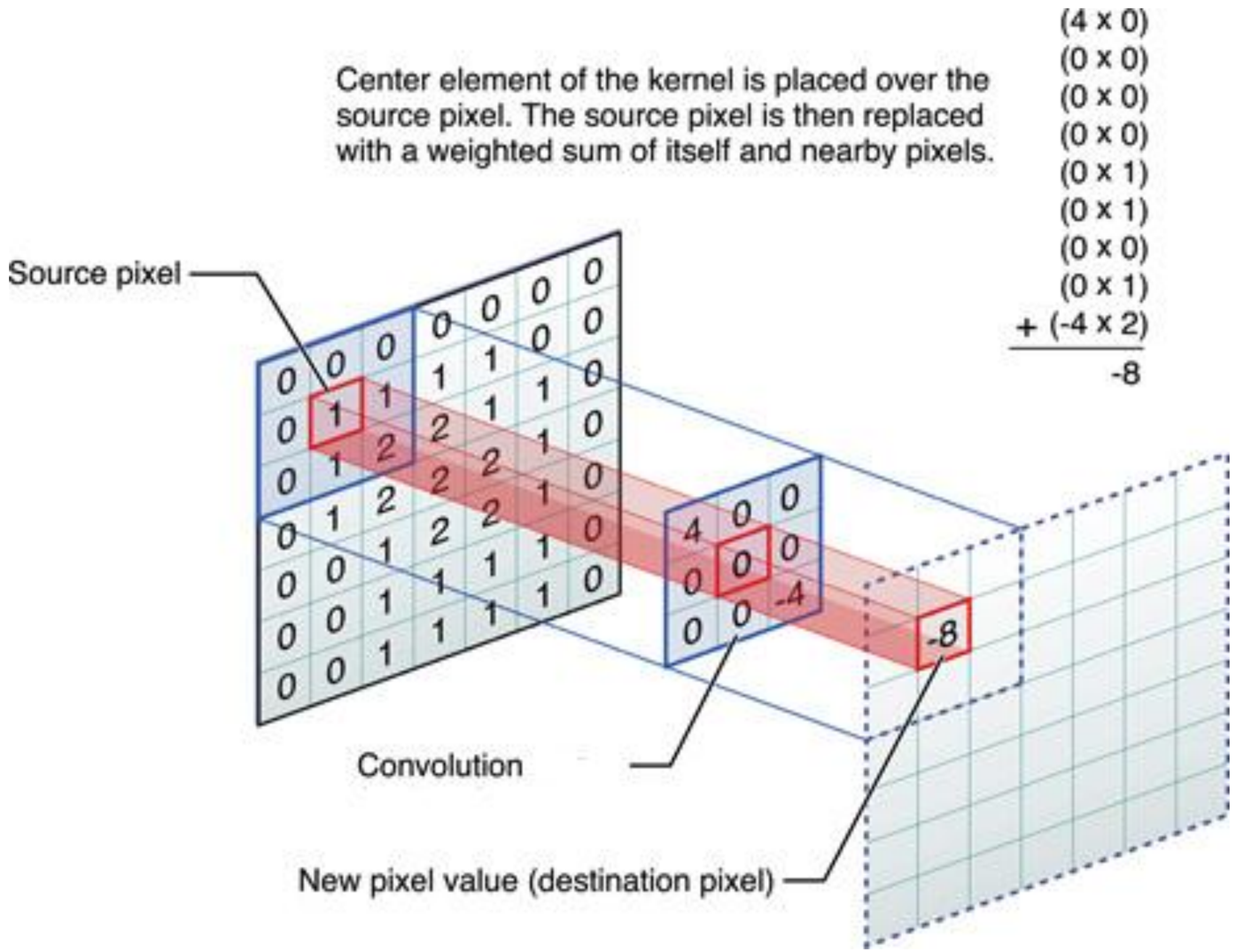
Green

Digital Image

Blue

Convolution Operation

Convoluting an image and a **kernel**: finding edges example



$$(f * g)(t) = \int_0^t f(\tau)\, g(t-\tau)\, d\tau$$

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Convolution Operation

Sobel X

Original

Sobel Y

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

kernel X

*

(convolution)

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

kernel Y

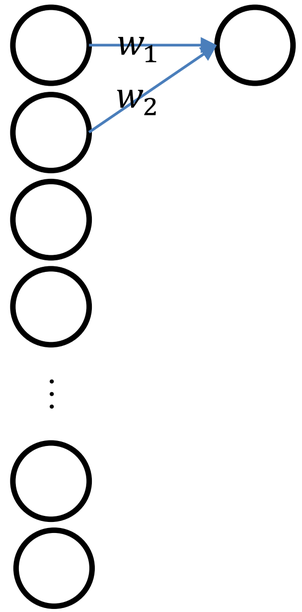
$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Content

- Convolution Operation
- Convolution Layers
- Pooling Layers
- Visualization of Feature Maps

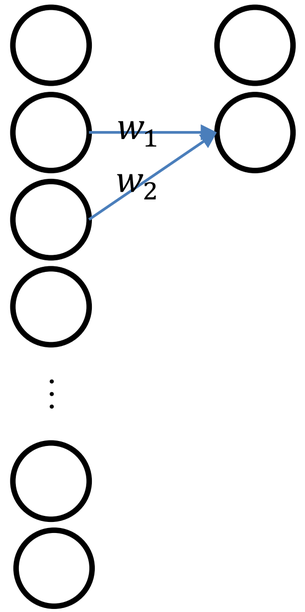
Convolutional Neural Networks

- Each neuron is **connected to part of the neurons** in the previous layer.
- Neurons in a single layer function **share all connections** (weights).
- Such connections/weights slide over the neurons of the previous layer, …



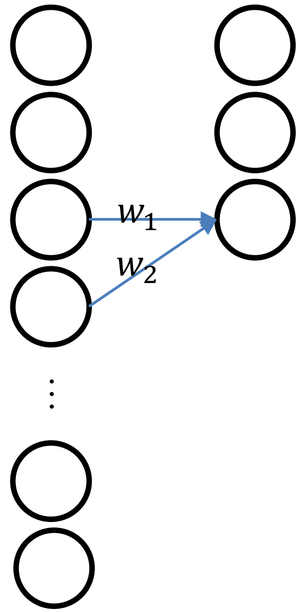
Convolutional Neural Networks

- Each neuron is **connected to part of the neurons** in the previous layer.
- Neurons in a single layer function **share all connections** (weights).
- Such connections/weights slide over the neurons of the previous layer, …



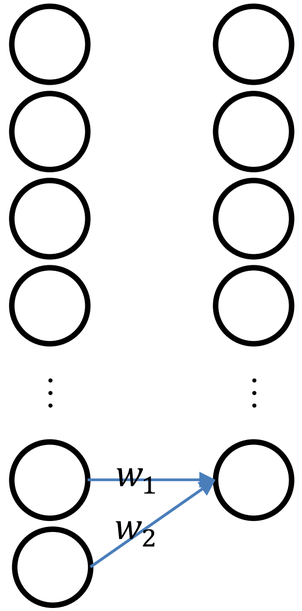
Convolutional Neural Networks

- Each neuron is **connected to part of the neurons** in the previous layer.
- Neurons in a single layer function **share all connections** (weights).
- Such connections/weights slide over the neurons of the previous layer, …



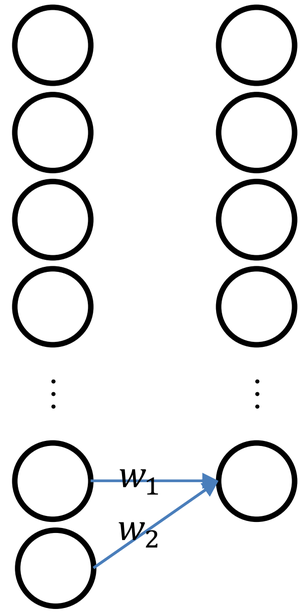
Convolutional Neural Networks

- Each neuron is **connected to part of the neurons** in the previous layer.
- Neurons in a single layer function **share all connections** (weights).
- Such connections/weights slide over the neurons of the previous layer, …



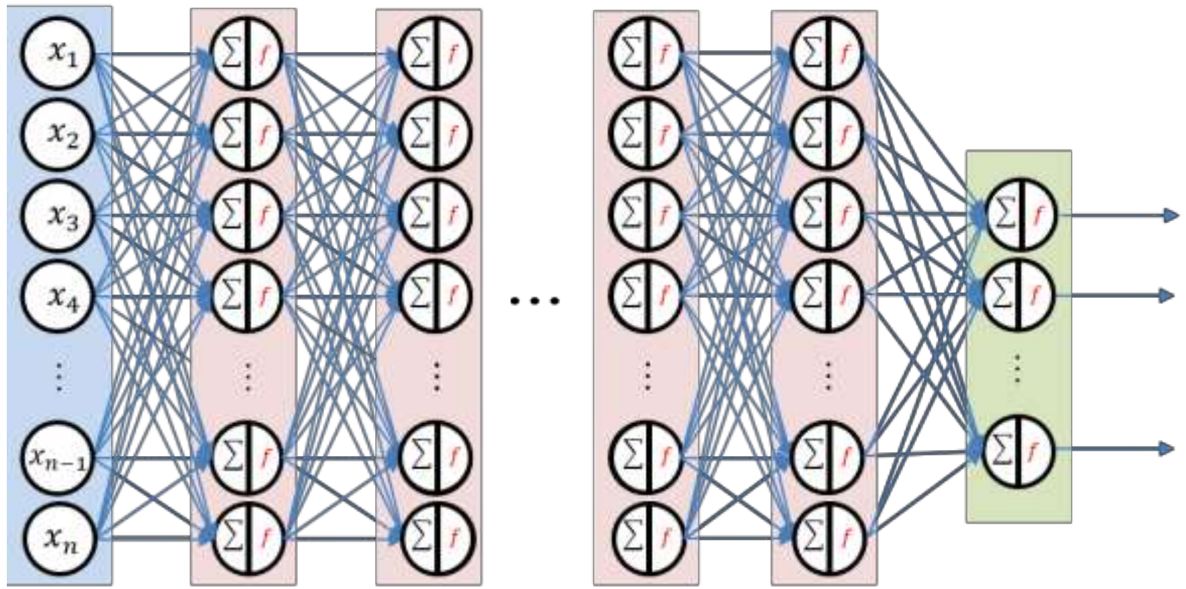
Convolutional Neural Networks

- Each neuron is **connected to part of the neurons** in the previous layer.
- Neurons in a single layer function **share all connections** (weights).
- Such connections/weights slide over the neurons of the previous layer, …
- … like a **convolution**.
- The number of weights per layer drops from m^2 to k (for some constant k).



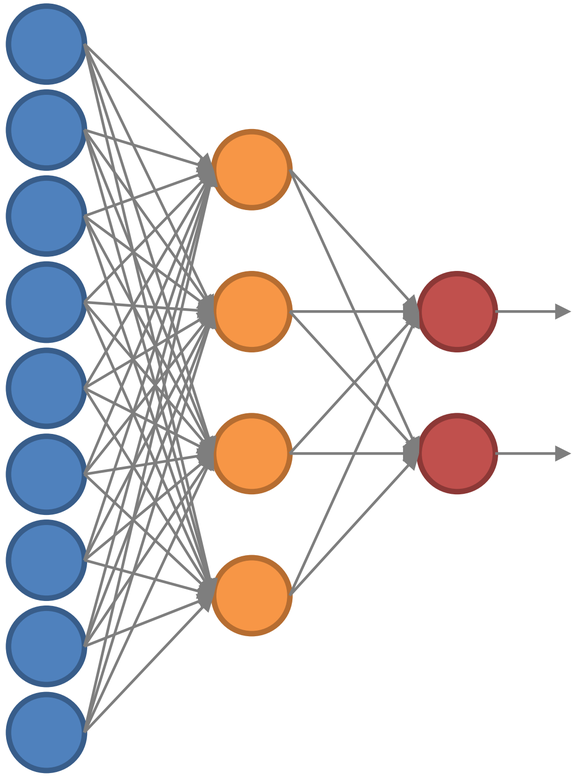
Convolutional Neural Networks

- A CNN comprises **multiple convolutional layers** as in a standard MLP followed by one or more fully connected layer.
- Still many parameters to estimate and a high demand for labeled samples.

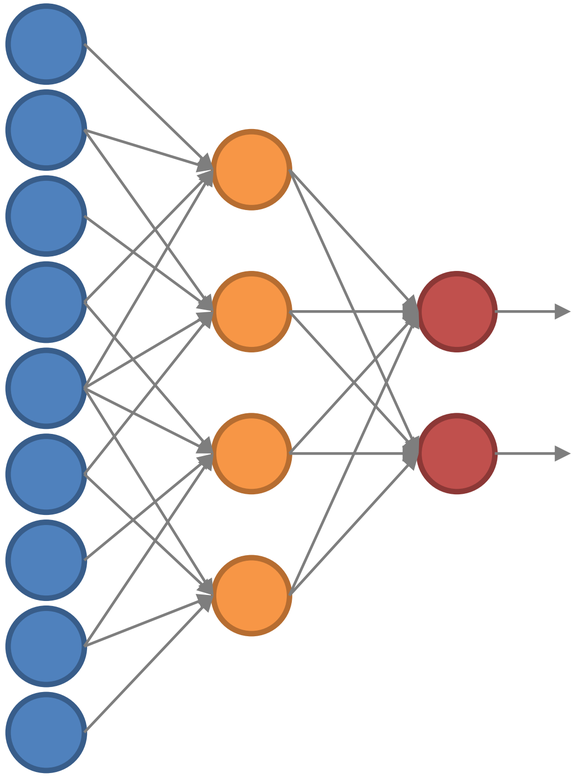


MLP to CNN

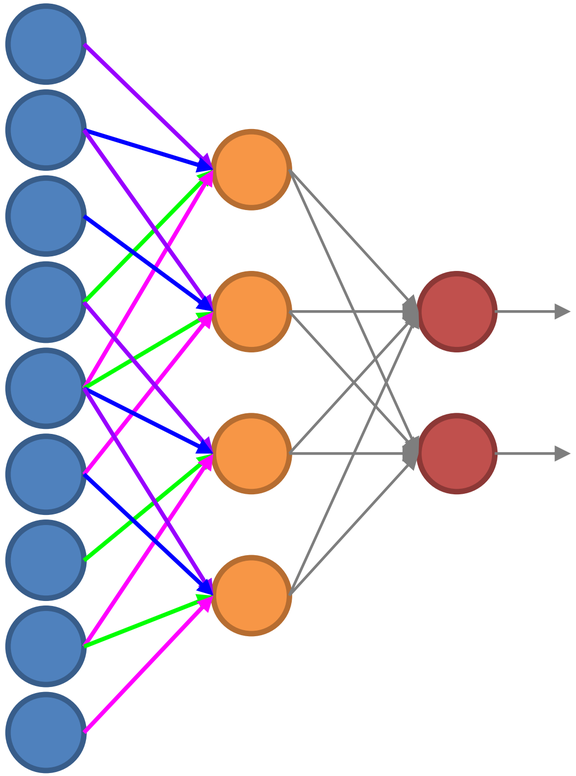
Standard MLP



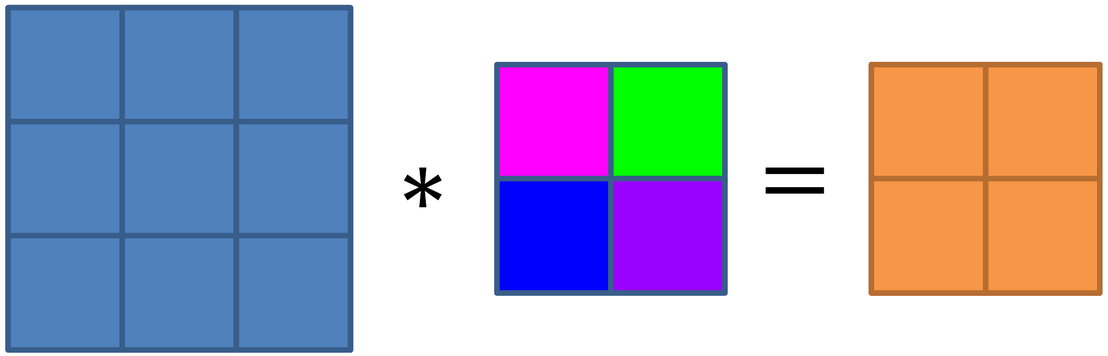
Cutting connections

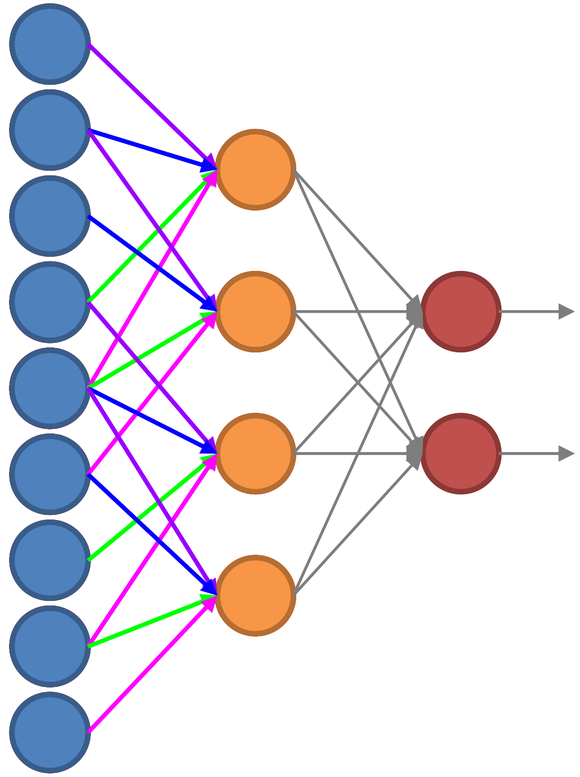


Sharing weights

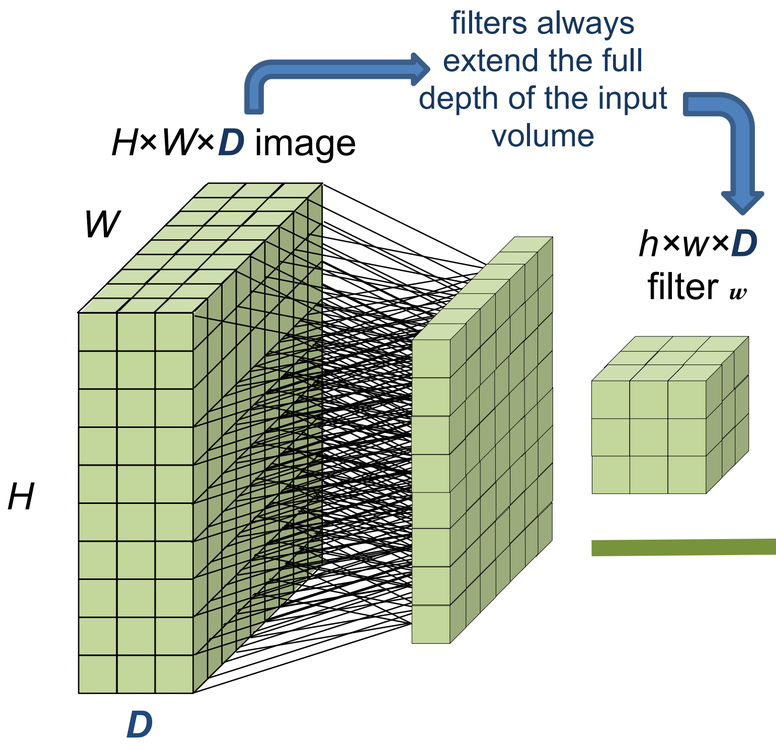


MLP to CNN

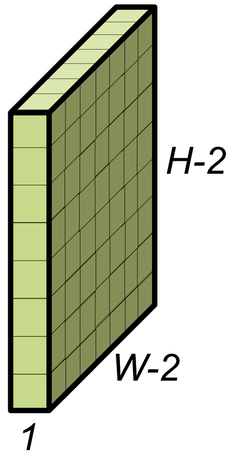




Convolution Layer



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”



convolve (slide) over all spatial locations

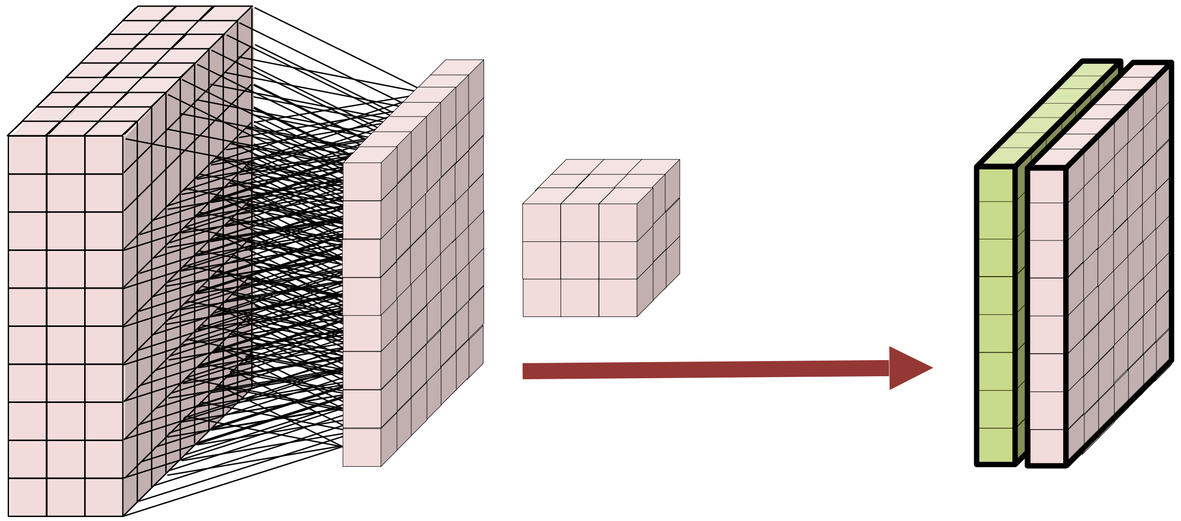
activation map

the dot product between the filter and a small $h \times w \times D$ chunk of the image (a $h \times w \times D$ dimensional dot product + bias)

$$w^T x + b$$

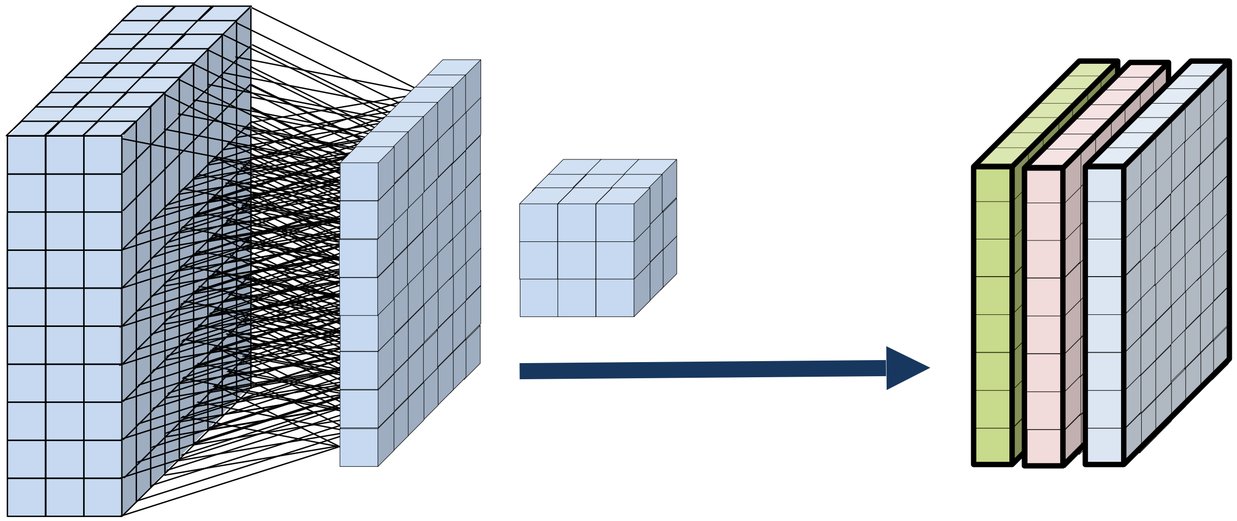
Convolution Layer

Consider a second filter,



Convolution Layer

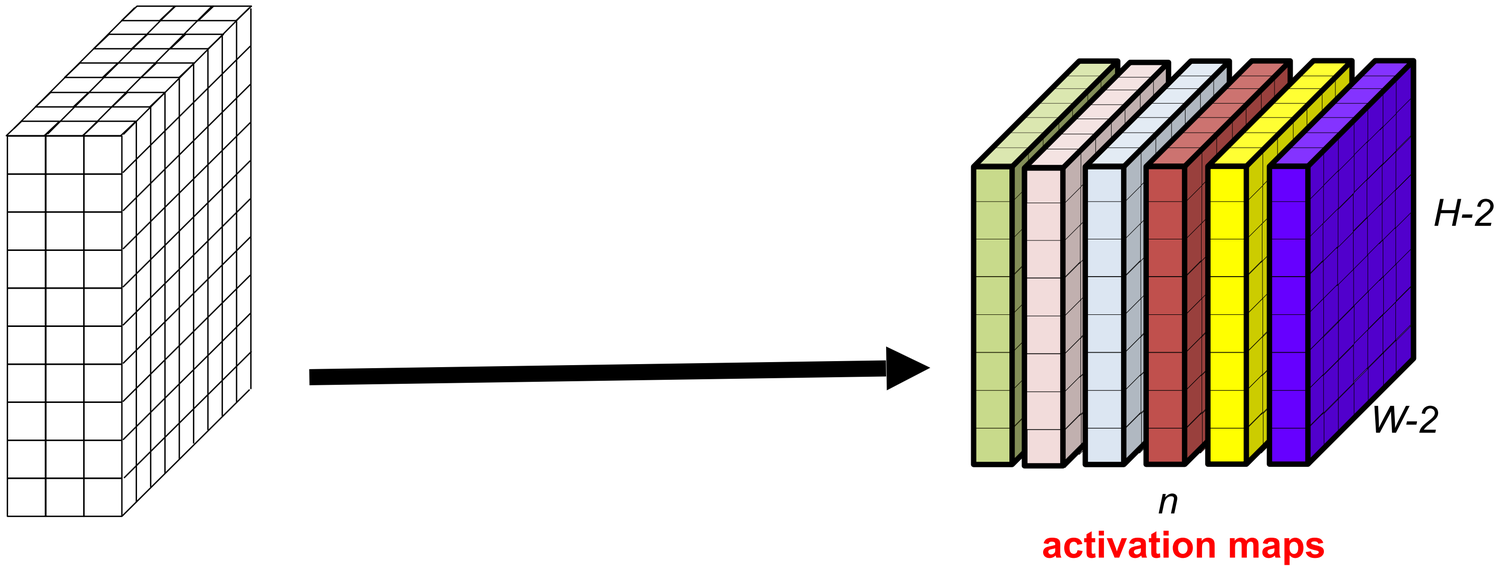
… and a third filter,



The *n* activation maps are stacked forming a (W-2) x (H-2) x n “new image”!

Convolution Layer

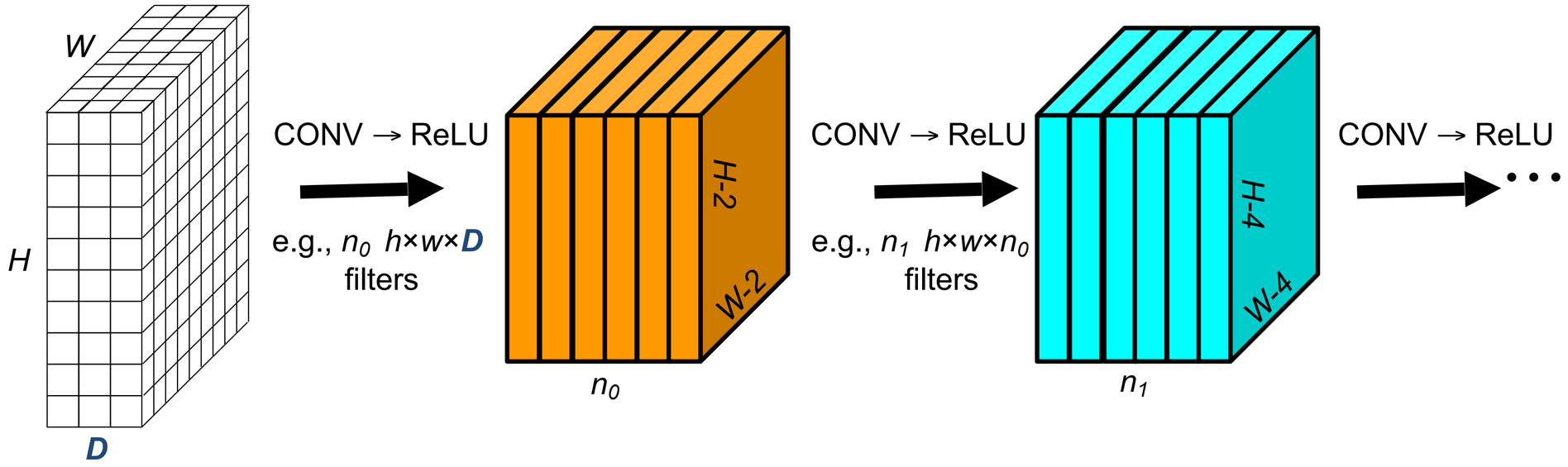
… up to *n*-th filters:



The *n* activation maps are stacked forming a (W-2) × (H-2) × n “new image”!

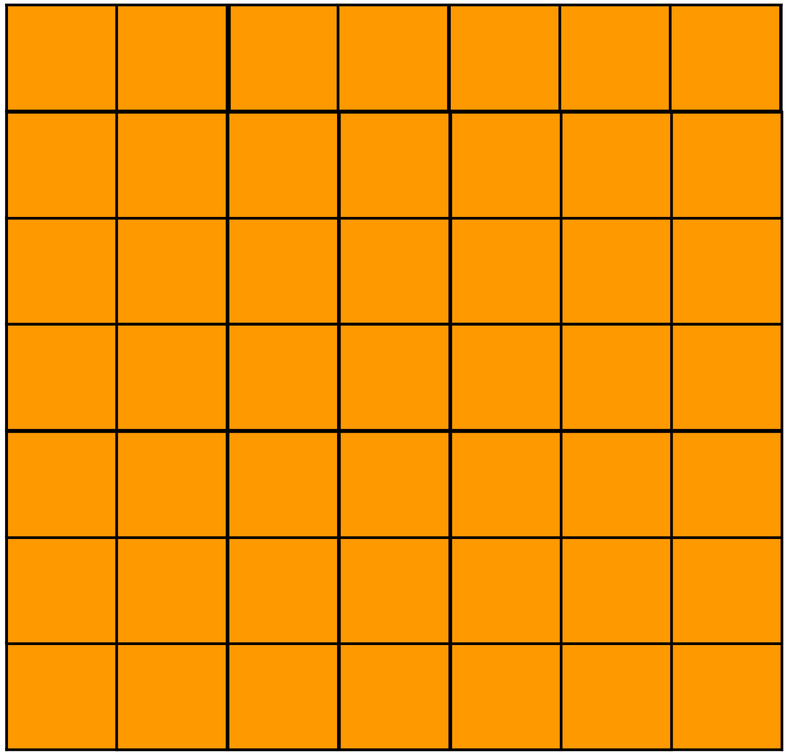
Basic Convolutional Network

A ConvNet is a sequence of Convolutional Layers, interspersed with activation functions.

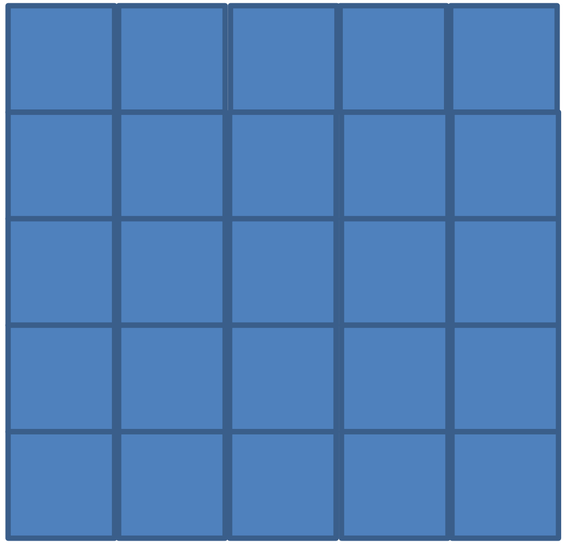


Spatial Dimension

Stride = 1



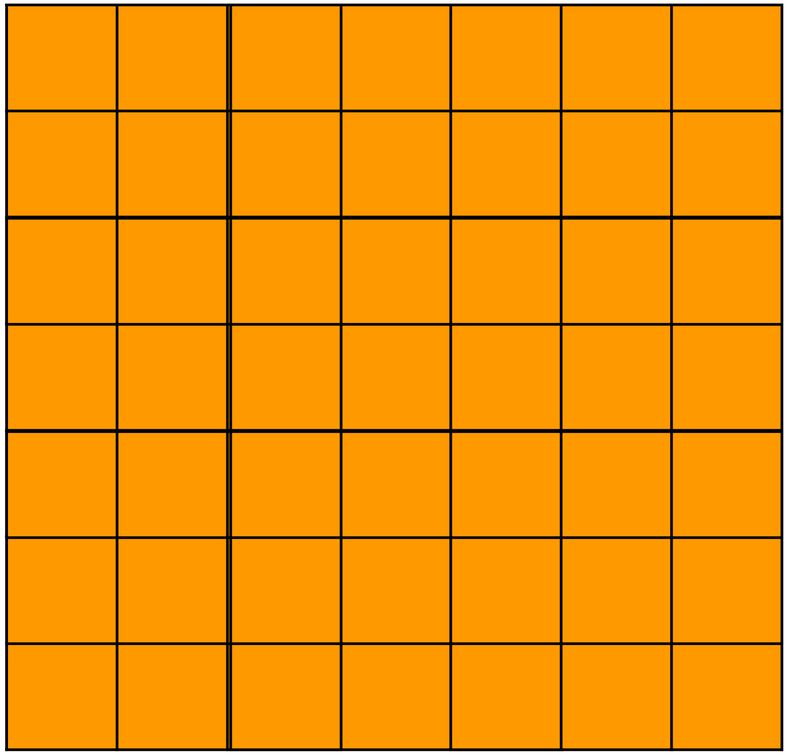
7x7 input (spatially) and 3×3 filter



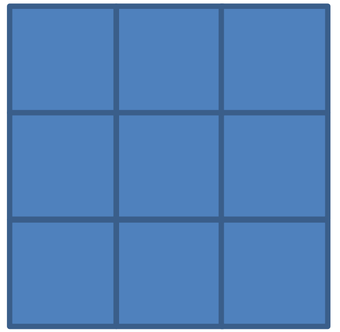
→ 5×5 output

Spatial Dimension

Stride = 2



7x7 input (spatially) and 3×3 filter



→ 3×3 output

and for stride=3?

Zero Padding

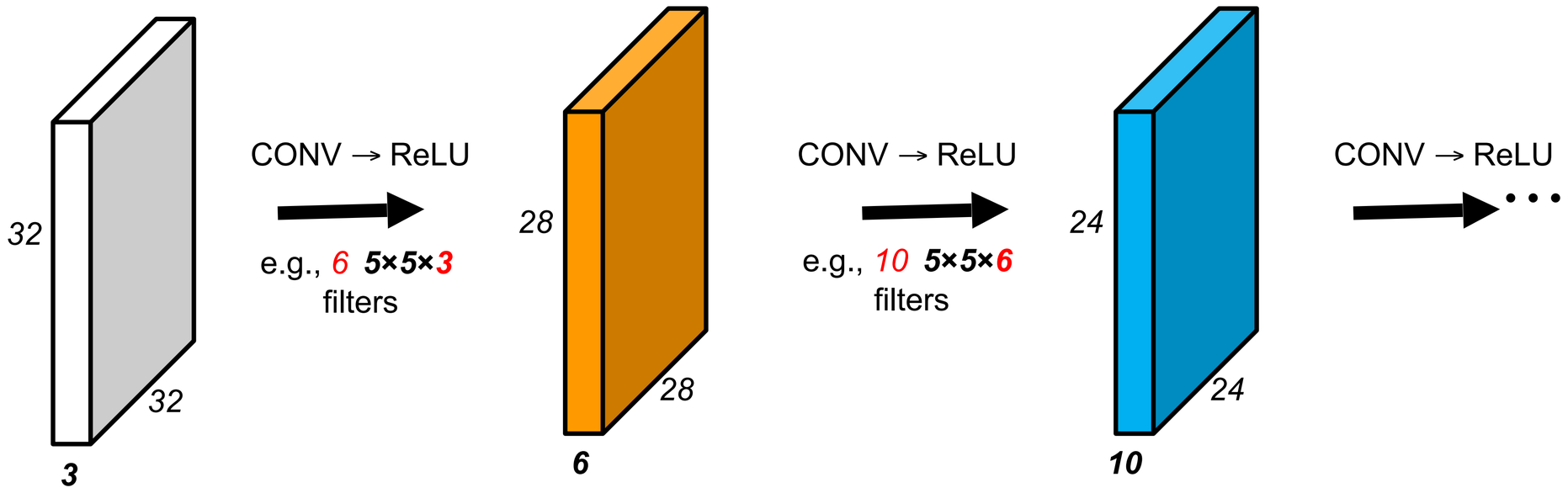
0	0	0	0	0	0	0	0	0
0								0
0								0
0								0
0								0
0								0
0								0
0								0
0	0	0	0	0	0	0	0	0

e.g. input 7×7 and a 3×3 filter, s=1
pad with 1 pixel border
→ what is the output?
7×7 output!

In general, CONV layers with s=1, zero-padding with (h-1)/2 (to preserve size spatially)
e.g. h = 3 → zero pad with 1
h = 5 → zero pad with 2
h = 7 → zero pad with 3

Spatial Dimension Along the Layers

E.g., a *32×32×3 input* convolved with *5×5* filters with no padding shrinks the volume spatially.



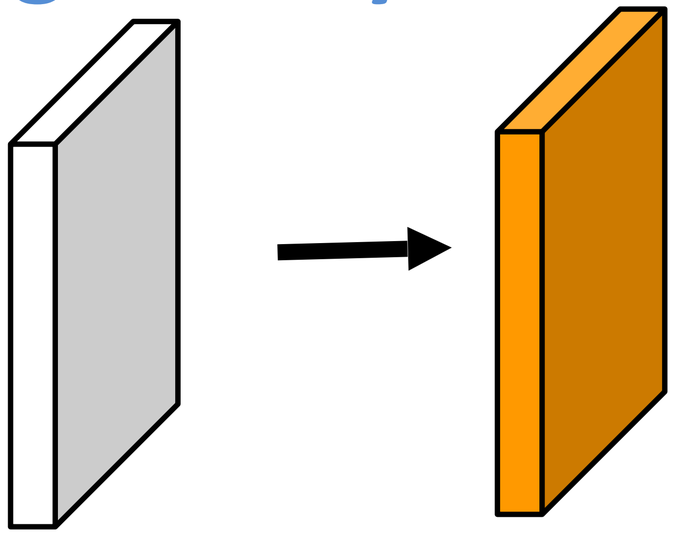
Spatial Dimension Along the Layers

E.g., a *32×32×3 input*

10 5×5 filters

stride= 1,

pad 2



Output volume?

→ *10240*

Number of parameters?

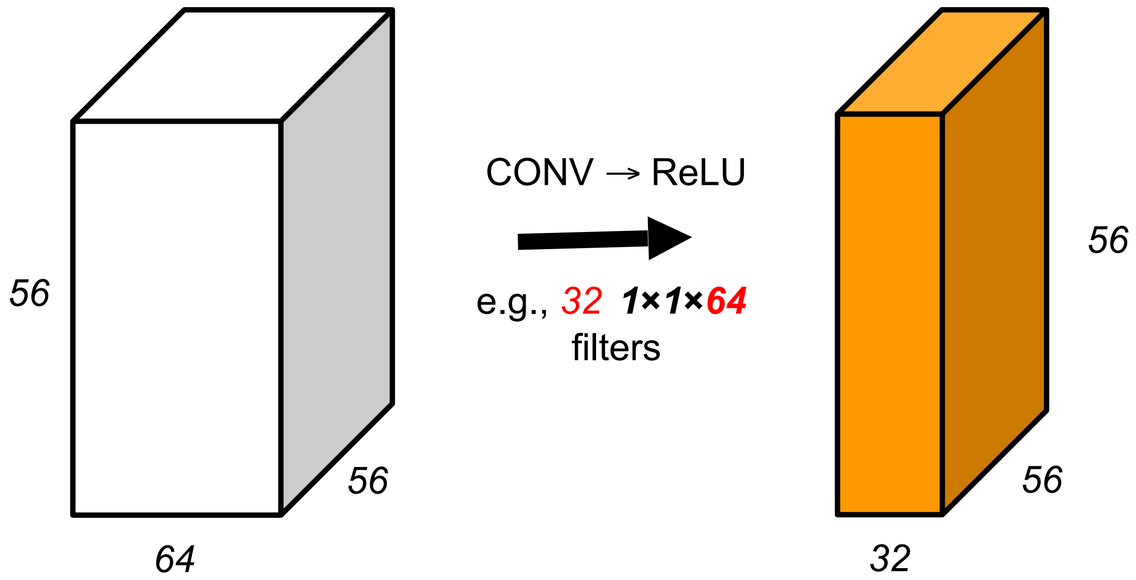
→ *10×(5×5×3+1)= 760*

Number of parameters for a fully connected layer?

→ *(32×32×3+1)×(32×32×3)×10≅ 10^8*

Remark: 1x1 Convolution

Quite common …

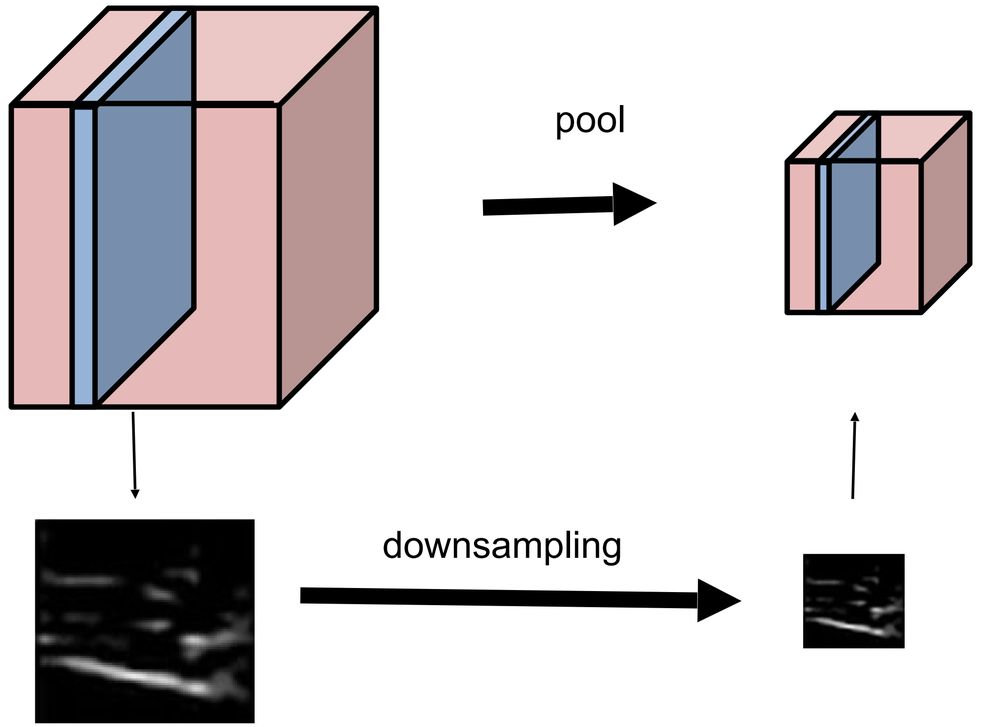


Content

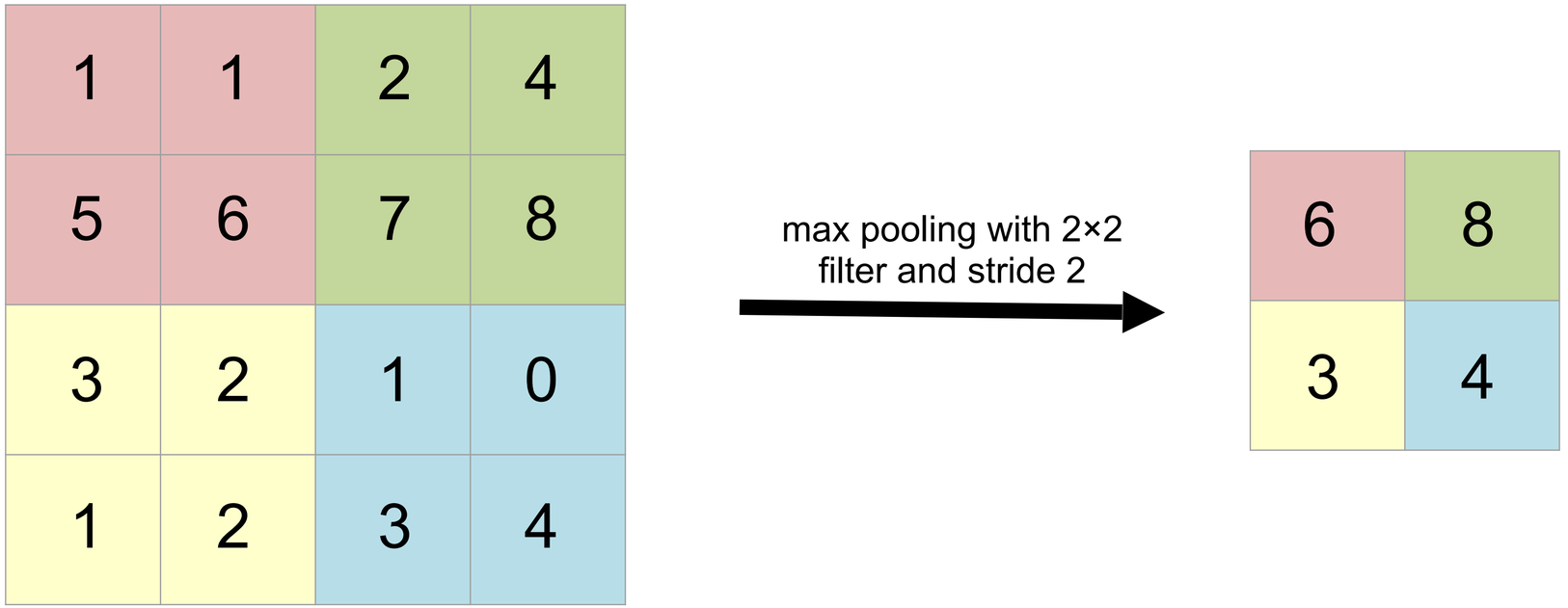
- Convolution Operation
- Convolution Layers
- Pooling Layers
- Visualization of Feature Maps

Pooling

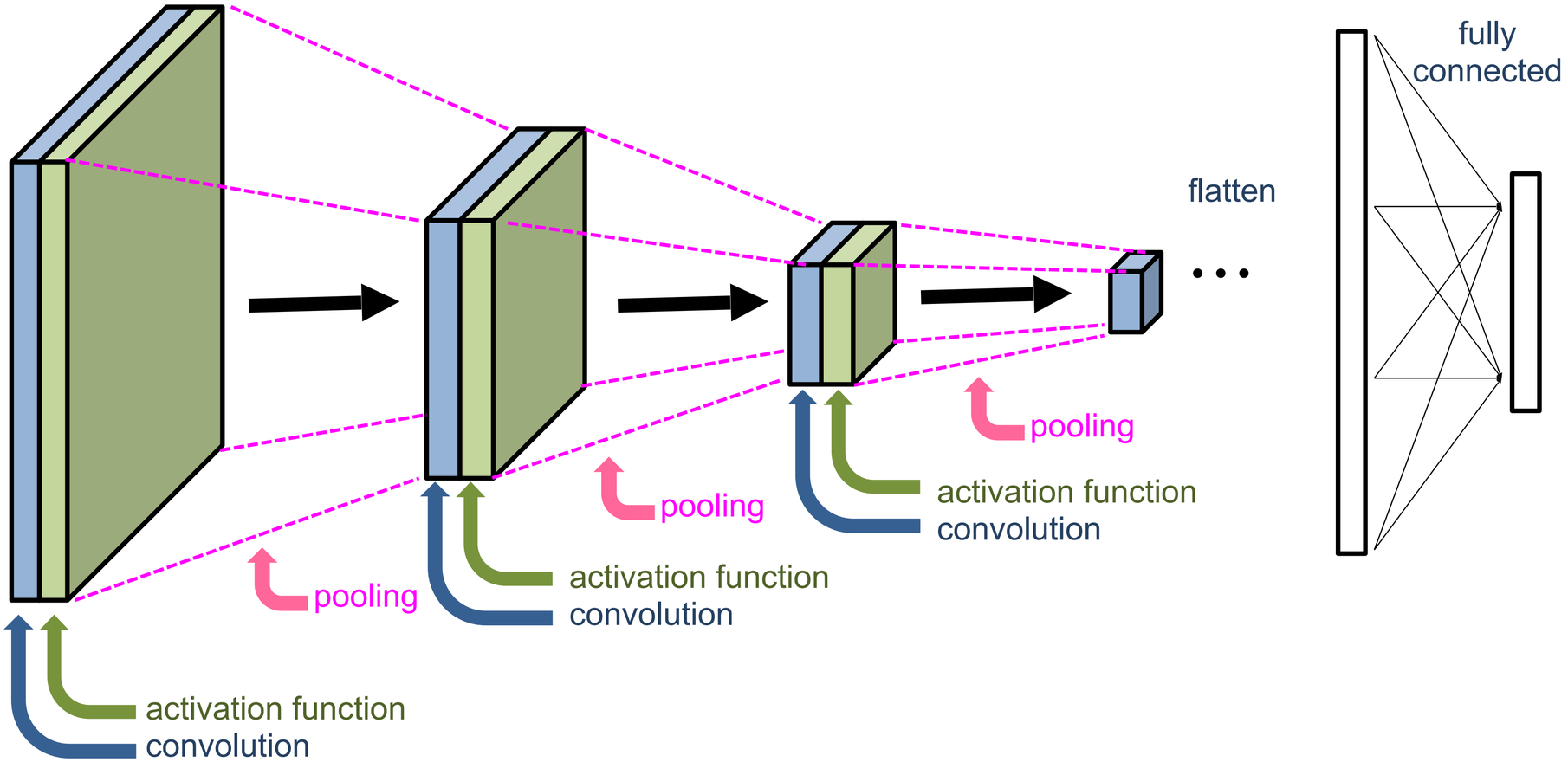
- replaces the output of the net at a certain location with a summary statistic of the nearby outputs.
- In all cases, pooling helps to make the representation become approximately *invariant* to small translations of the input.



Max Pooling


1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pooling with 2×2
filter and stride 2
6 8
3 4

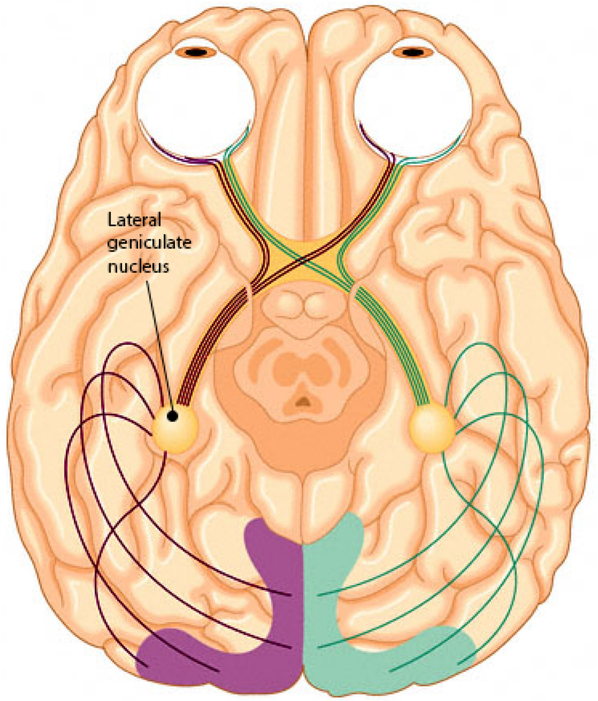
Typical ConvNet for Image Classification

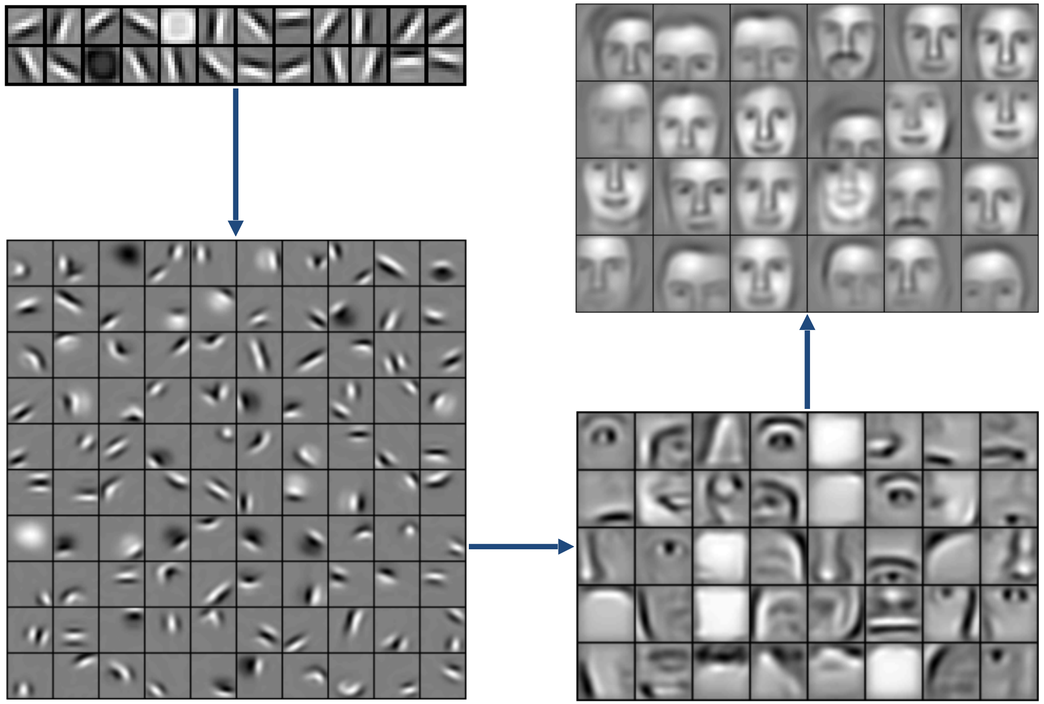


Content

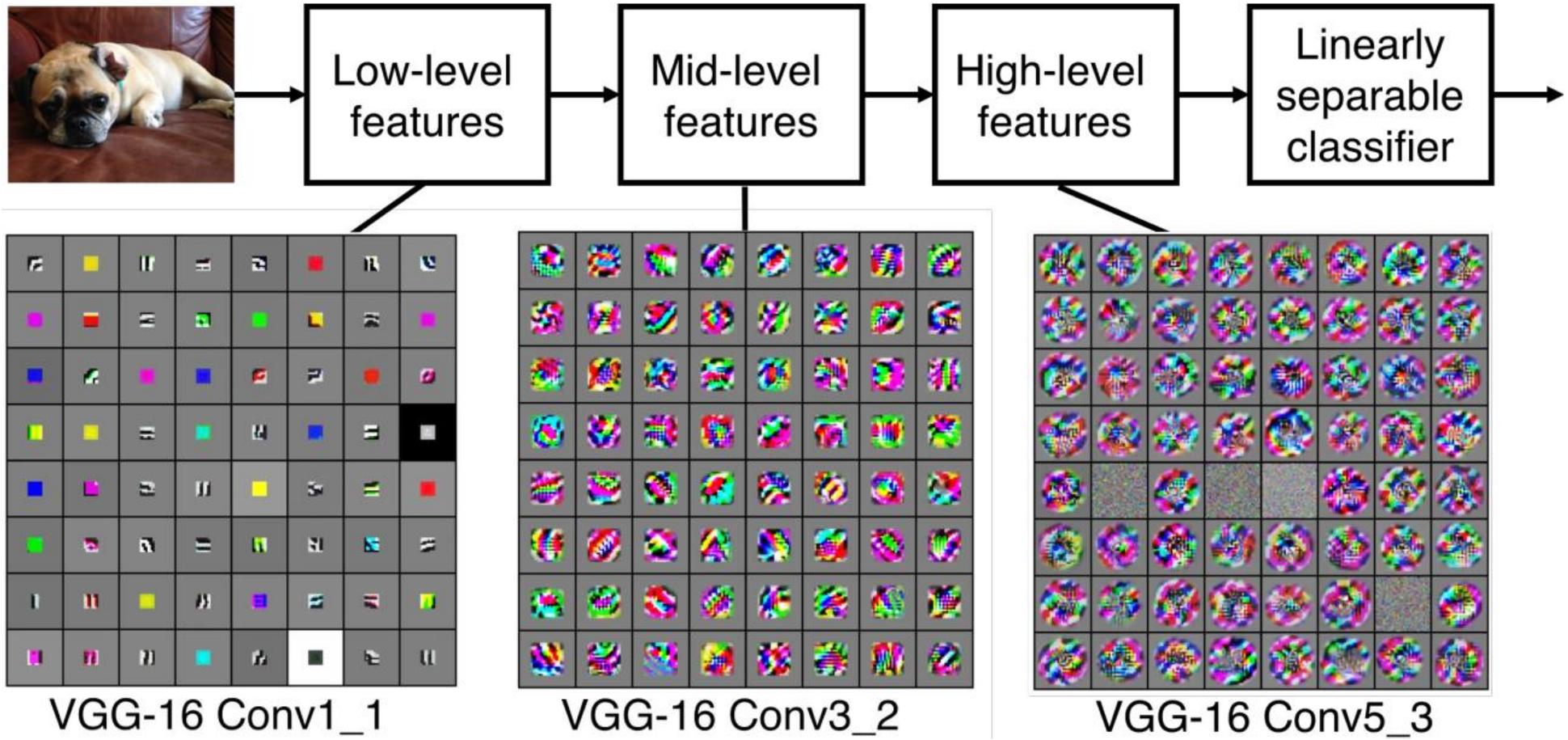
- Convolution Operation
- Convolution Layers
- Pooling Layers
- Visualization of Feature Maps

Analogy to the brain



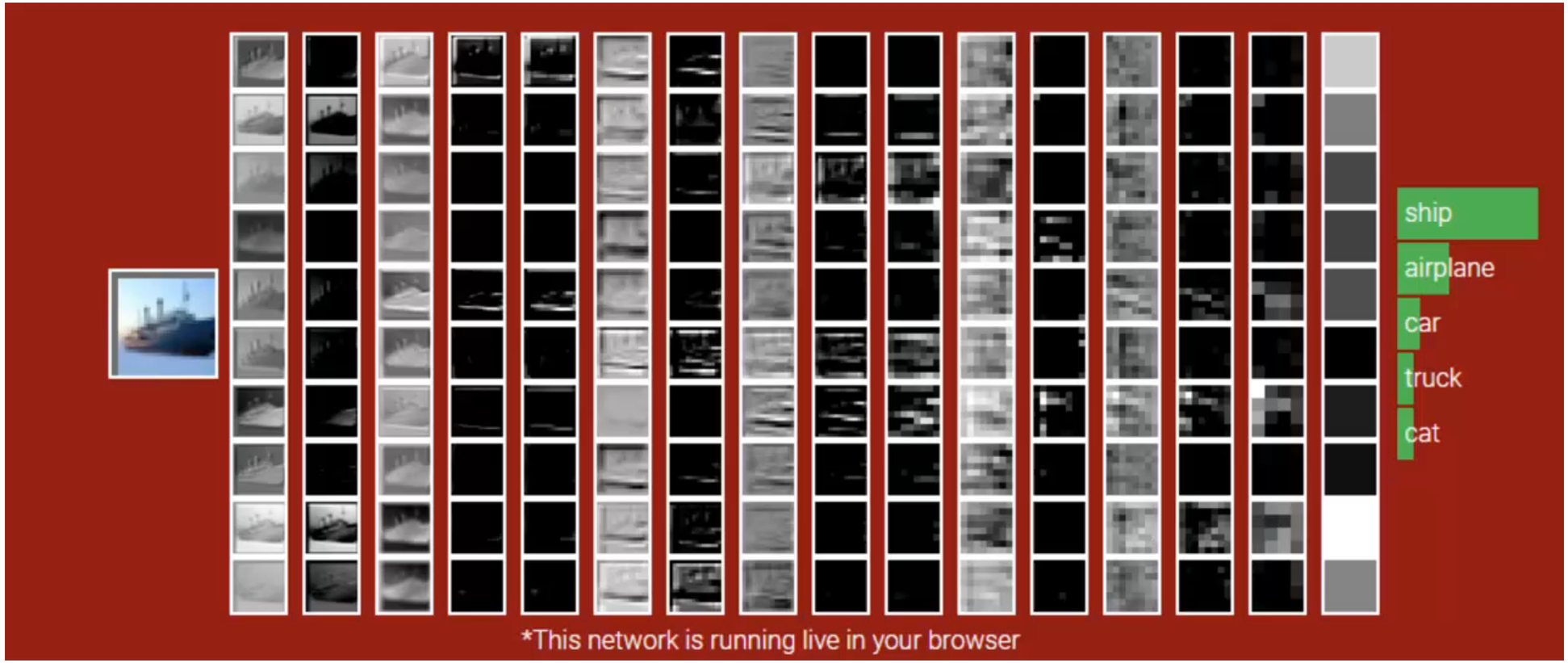


Filters' Visualization



Picture from https://stackoverflow.com/questions/48220598/convolution-layer-in-cnn?noredirect=1&lq=1

Visualization of Layers' Output



From http://cs231n.stanford.edu/

Next Lecture

Lecture 3

Training Deep Networks

See you next class!