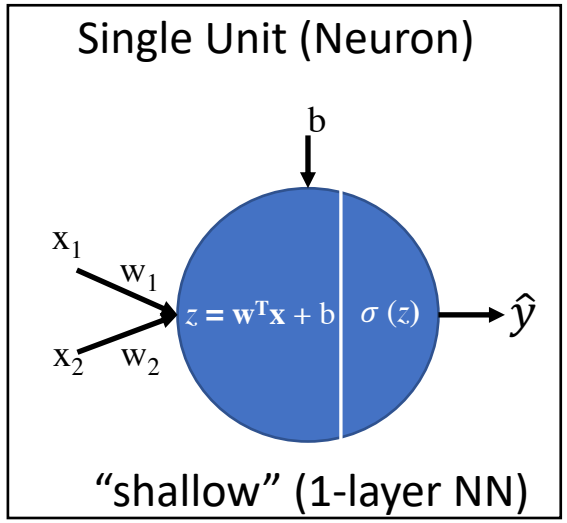
CAP 5415: Computer Vision

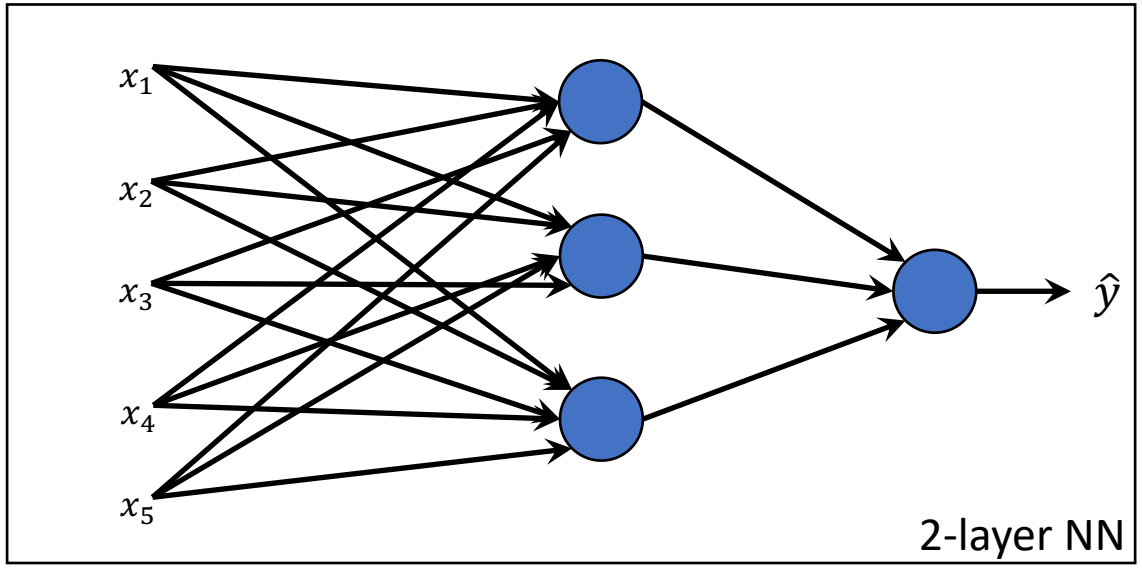
Intro to Convolutional Neural Networks

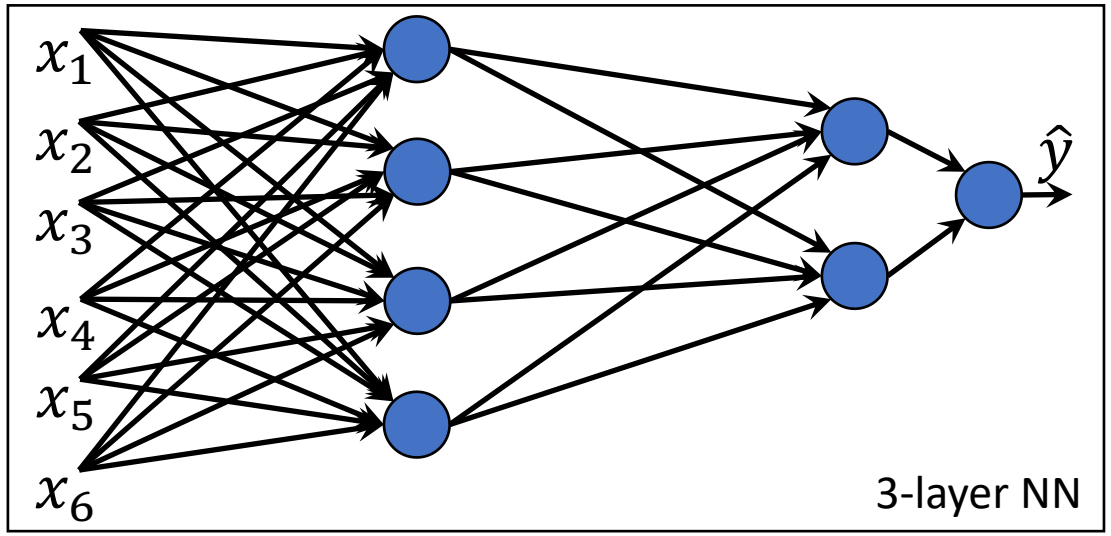
Dr. Sedat Ozer

Deep vs. Shallow

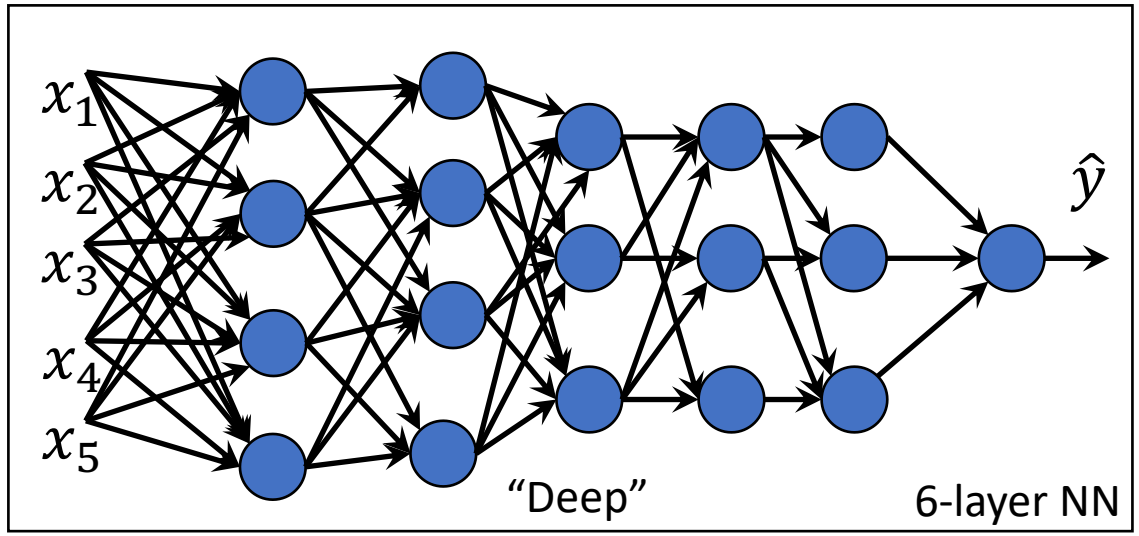


1-hidden layer





2-hidden layers



5-hidden layers

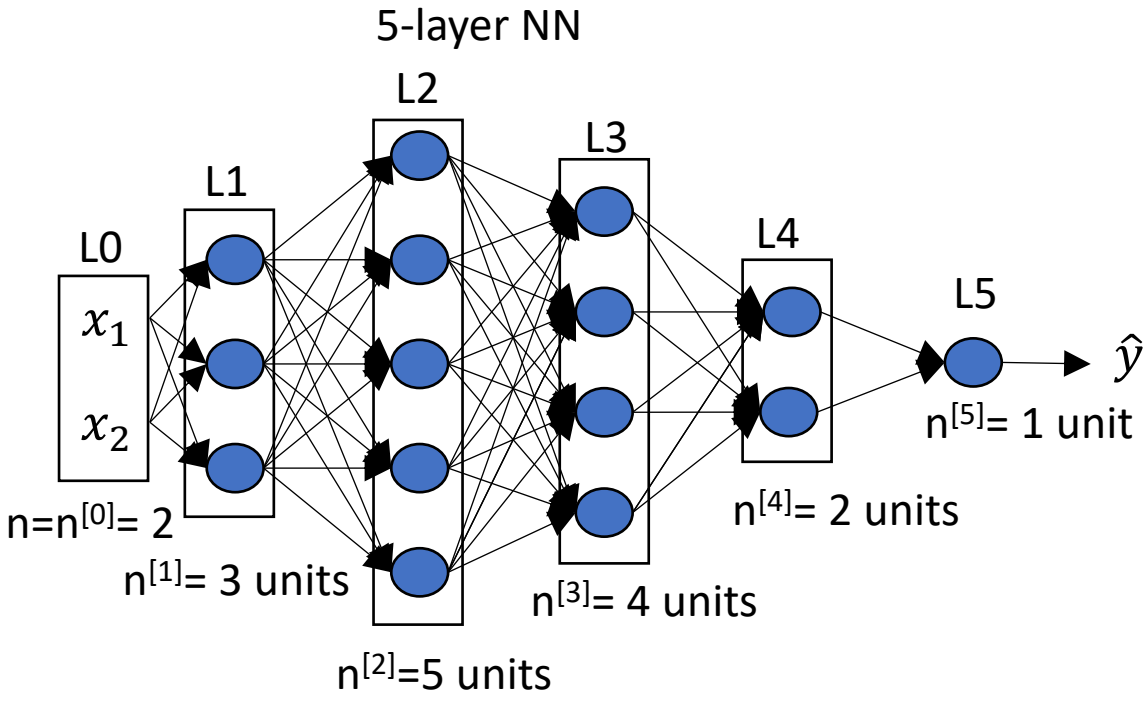
Which one is deeper?

A deep NN: Notation

$$A^{[0]} = X = \begin{bmatrix} | & | & & | \\ \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(m)} \\ | & | & & | \end{bmatrix}_{(n^{[0]} \text{ x } m)}$$

$$A^{[l]} = \begin{bmatrix} | & | & & | \\ \mathbf{a}^{[l](1)} & \mathbf{a}^{[l](2)} & \dots & \mathbf{a}^{[l](m)} \\ | & | & & | \end{bmatrix}_{(n^{[l]} \text{ x } m)}$$

($Z^{[l]}$ has the same shape as $A^{[l]}$)



$$Z^{[l]} = W^{[l]} A^{[l-1]} + \mathbf{b}^{[l]}$$
$$A^{[l]} = g(Z^{[l]})$$

Dims

$\mathbf{z}^{[l]}$: $(n^{[l]} \text{ x } 1)$

$\mathbf{a}^{[l]}$: $(n^{[l]} \text{ x } 1)$

$\mathbf{W}^{[l]}$: $(n^{[l]} \text{ x } n^{[l-1]})$

Hyperparameters

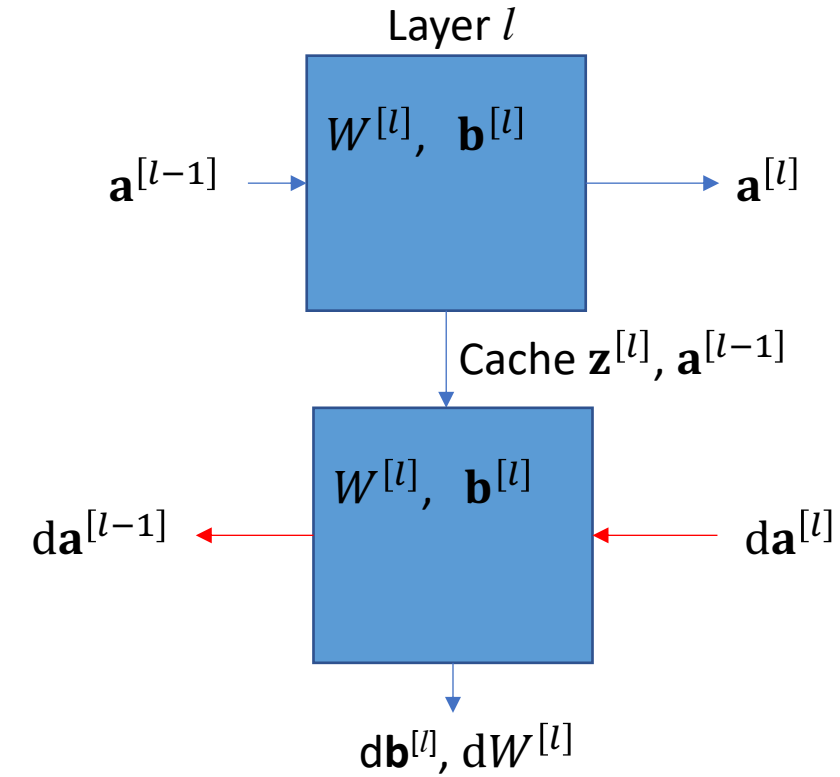
- We have parameters: $W^{[l]}$, $\mathbf{b}^{[l]}$ for each layer (actual parameters)
- We also have additional parameters affecting the performance of our algorithm: Hyperparameters. I.e., the hyperparameters are the parameters that affect our actual parameters. Examples:
  - Learning rate,
  - number of iterations,
  - total number of hidden layers,
  - hidden unit numbers in each layer,
  - Choice of activation function,
  - Momentum,
  - mini batch size,
  - regularization parameters,
  - choice of cost function, choice of the optimization technique, …

Forward vs. Backward computation

- Forward direction computes the output and the loss.
- Backward direction computes the derivatives to find the update directions for the parameters: weights & biases.

Forward direction:

Backward direction:



Element-wise product

$$\mathrm{d}\mathbf{z}^{[l]} = \mathrm{d}\mathbf{a}^{[l]} * g^{[l]\prime}(\mathbf{z}^{[l]})$$

$$\mathrm{d}W^{[l]} = \mathrm{d}\mathbf{z}^{[l]}\ \mathbf{a}^{[l-1]}$$

$$\mathrm{d}\mathbf{b}^{[l]} = \mathrm{d}\mathbf{z}^{[l]}$$

$$\mathrm{d}\mathbf{a}^{[l-1]} = W^{[l]T}\ \mathrm{d}\mathbf{z}^{[l]}$$

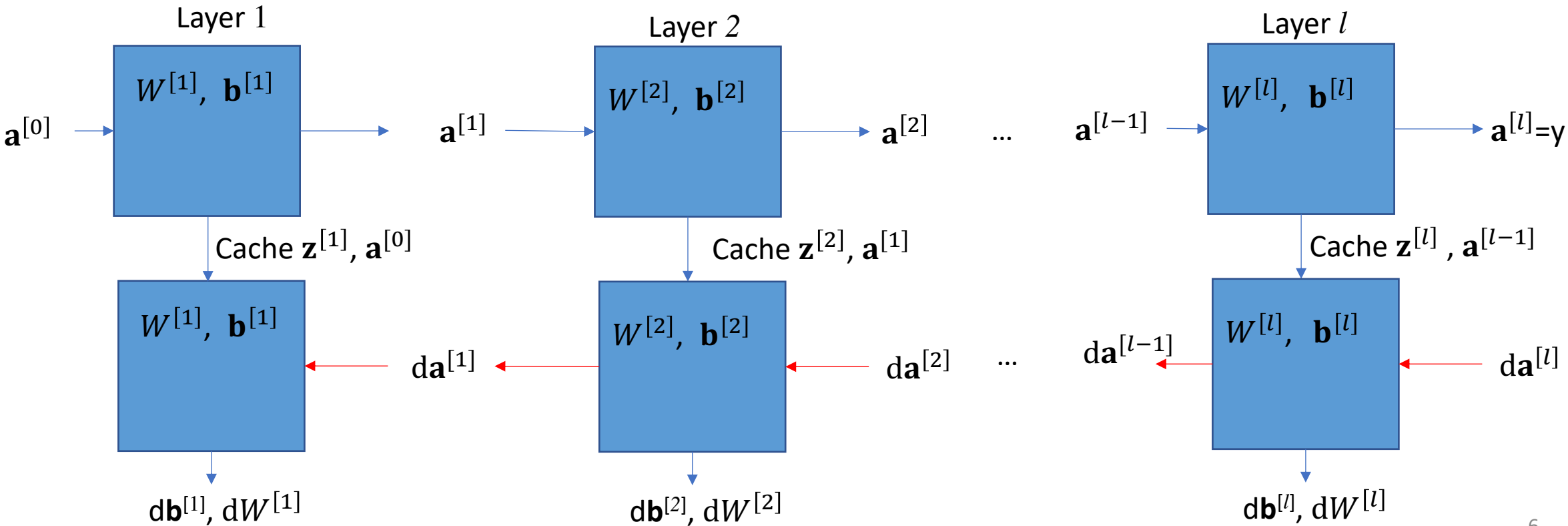
Forward vs. Backward computation

$$d\mathbf{z}^{[l]} = d\mathbf{a}^{[l]} * g^{[l]'}(\mathbf{z}^{[l]})$$

$$dW^{[l]} = d\mathbf{z}^{[l]}\ \mathbf{a}^{[l-1]}$$

$$d\mathbf{b}^{[l]} = d\mathbf{z}^{[l]}$$

$$d\mathbf{a}^{[l-1]} = W^{[l]T}\ d\mathbf{z}^{[l]}$$



Backward Propagation

Element-wise multiplication

$$\mathrm{d}\mathbf{z}^{[l]} = \mathrm{d}\mathbf{a}^{[l]} * g^{[l]\prime}(\mathbf{z}^{[l]})$$

$$\mathrm{d}W^{[l]} = \mathrm{d}\mathbf{z}^{[l]}\ \mathbf{a}^{[l-1]}$$

$$\mathrm{d}\mathbf{b}^{[l]} = \mathrm{d}\mathbf{z}^{[l]}$$

$$\mathrm{d}\mathbf{a}^{[l-1]} = W^{[l]T}\ \mathrm{d}\mathbf{z}^{[l]}$$

For a single sample

Element-wise multiplication

$$\mathrm{d}\boldsymbol{Z}^{[l]} = \mathrm{d}A^{[l]} * g^{[l]\prime}(\boldsymbol{Z}^{[l]})$$

$$\mathrm{d}W^{[l]} = \frac{1}{m}\mathrm{d}\boldsymbol{Z}^{[l]}\ \mathbf{A}^{[l-1]T}$$

$$\mathrm{d}\mathbf{b}^{[l]} = \frac{1}{m}\mathrm{np.sum}(\mathrm{d}\boldsymbol{Z}^{[l]}, axis = 1, keepdims = True)$$

$$\mathrm{d}\mathbf{A}^{[l-1]} = W^{[l]T}\ \mathrm{d}\mathbf{Z}^{[l]}$$

For m data samples ("Vectorized" versions)

Convolutional Neural Networks

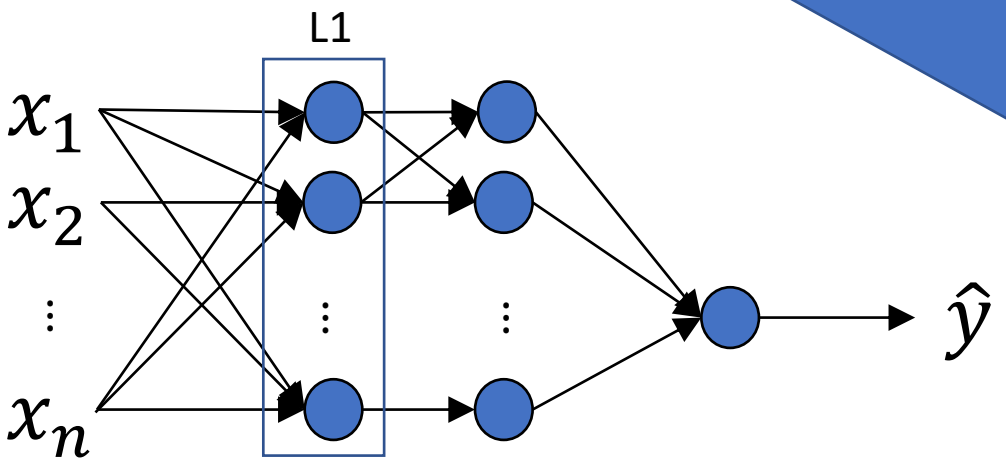
Story so far…

- Up until now, we learned how to
- Lets consider the case where the

As an alternative approach:
Use a convolutional NN !!

A high-resolution image:1000x1000x3
= (3Million x 1)

Problem: Is there a person in this picture?
(answer: yes/no)



n =3Million

$W^{[1]}$
(1000 x 3Million) = **3 Billion** parameters!

Convolutional Neural Networks (CNN)

Includes many additional components - operations. Most common ones are:

- Filtering,
- Padding,
- Stride,
- Pooling.

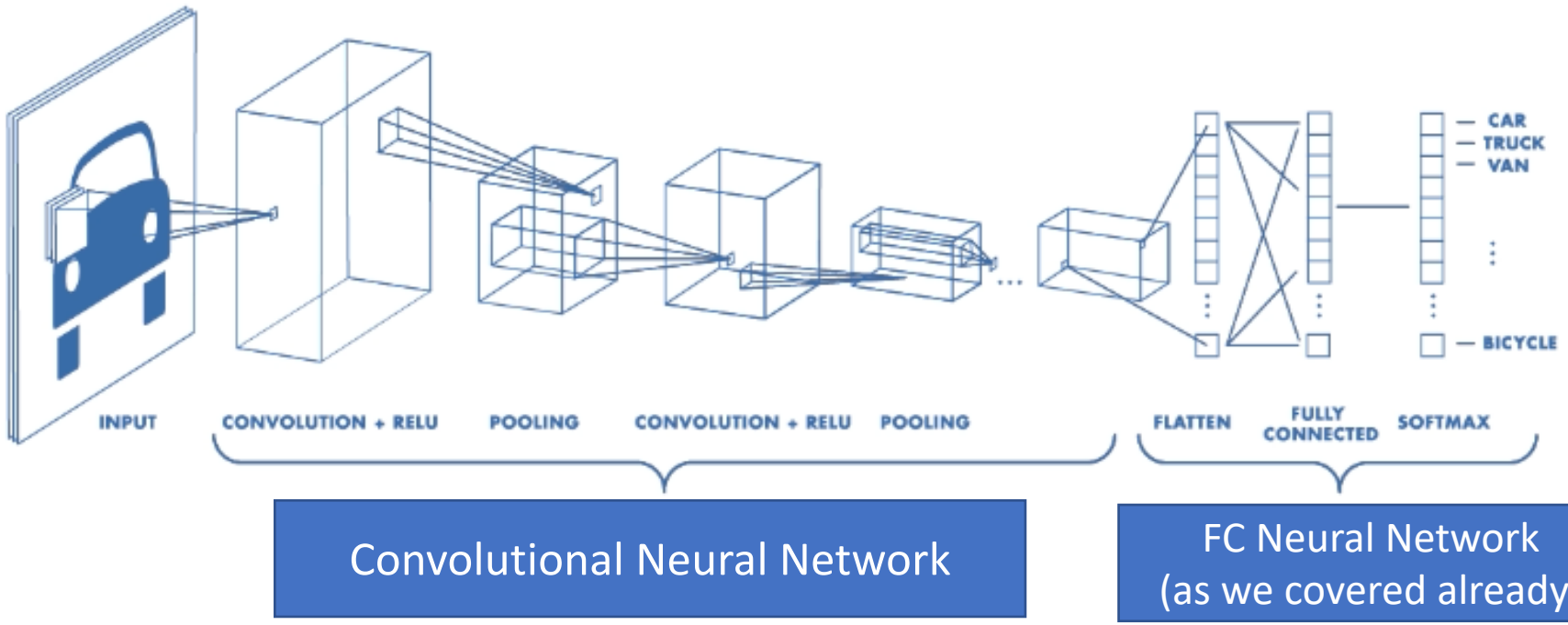


Image source: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Features ?

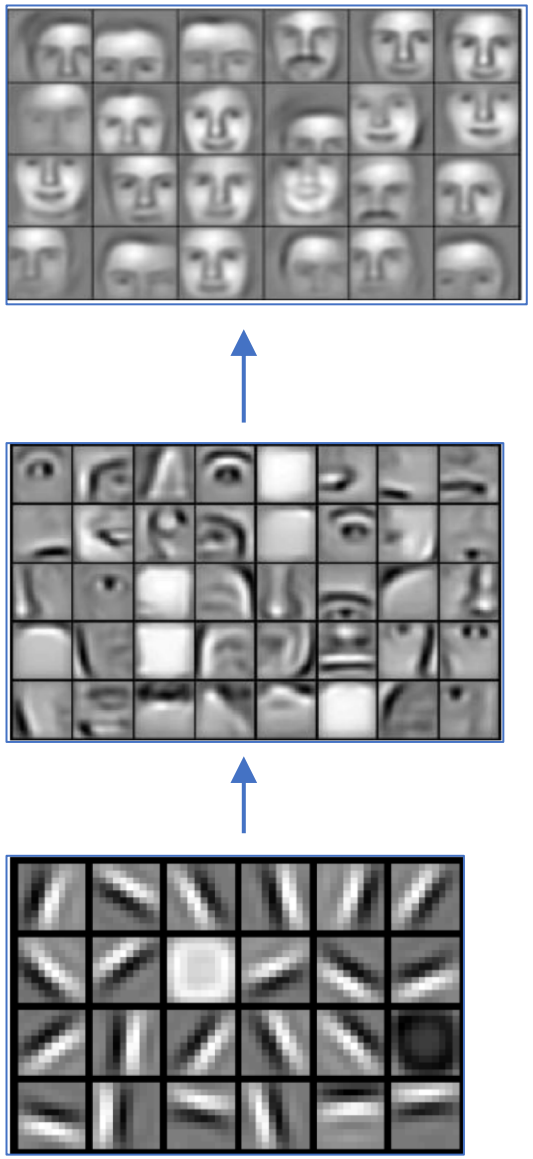


Image credit: Andrew Ng

Edge Detection

- To explain the Convolutional NNs, we will look at the edge detection example first.

Edges

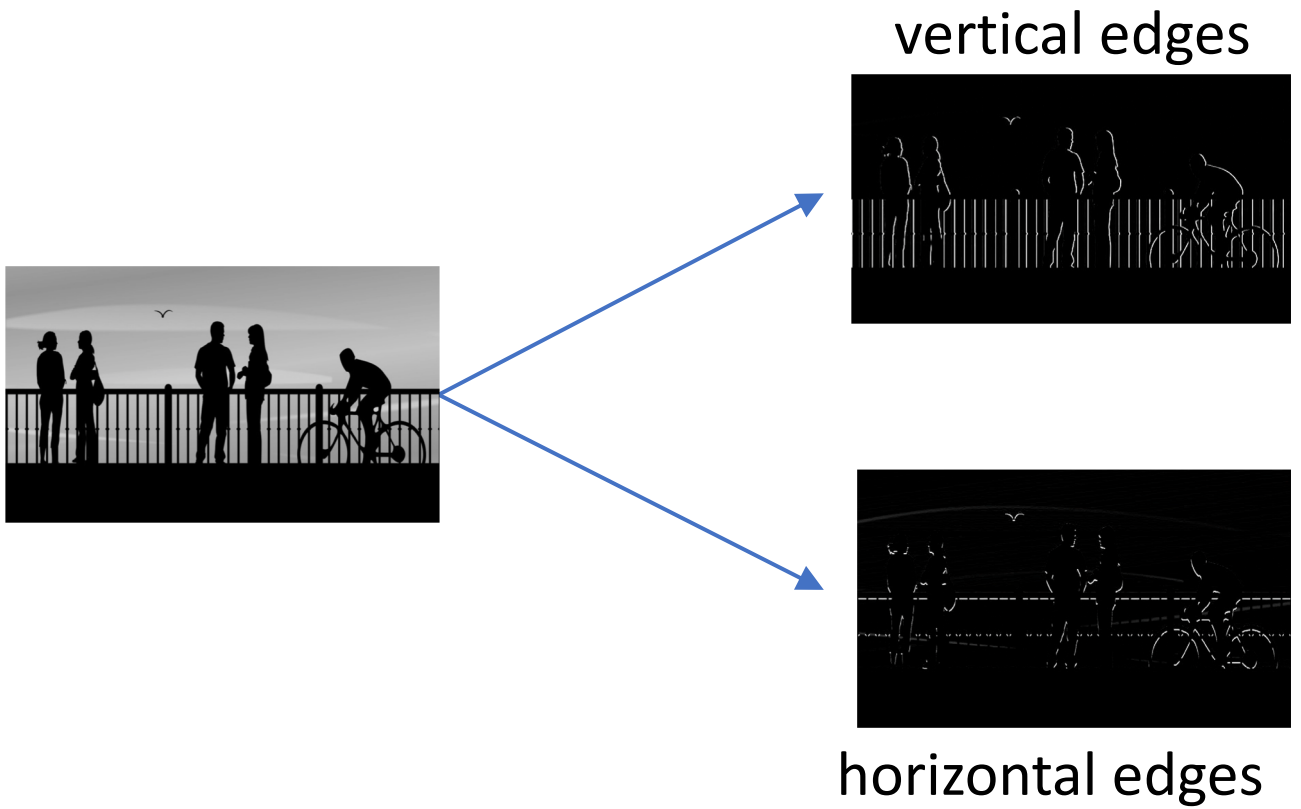
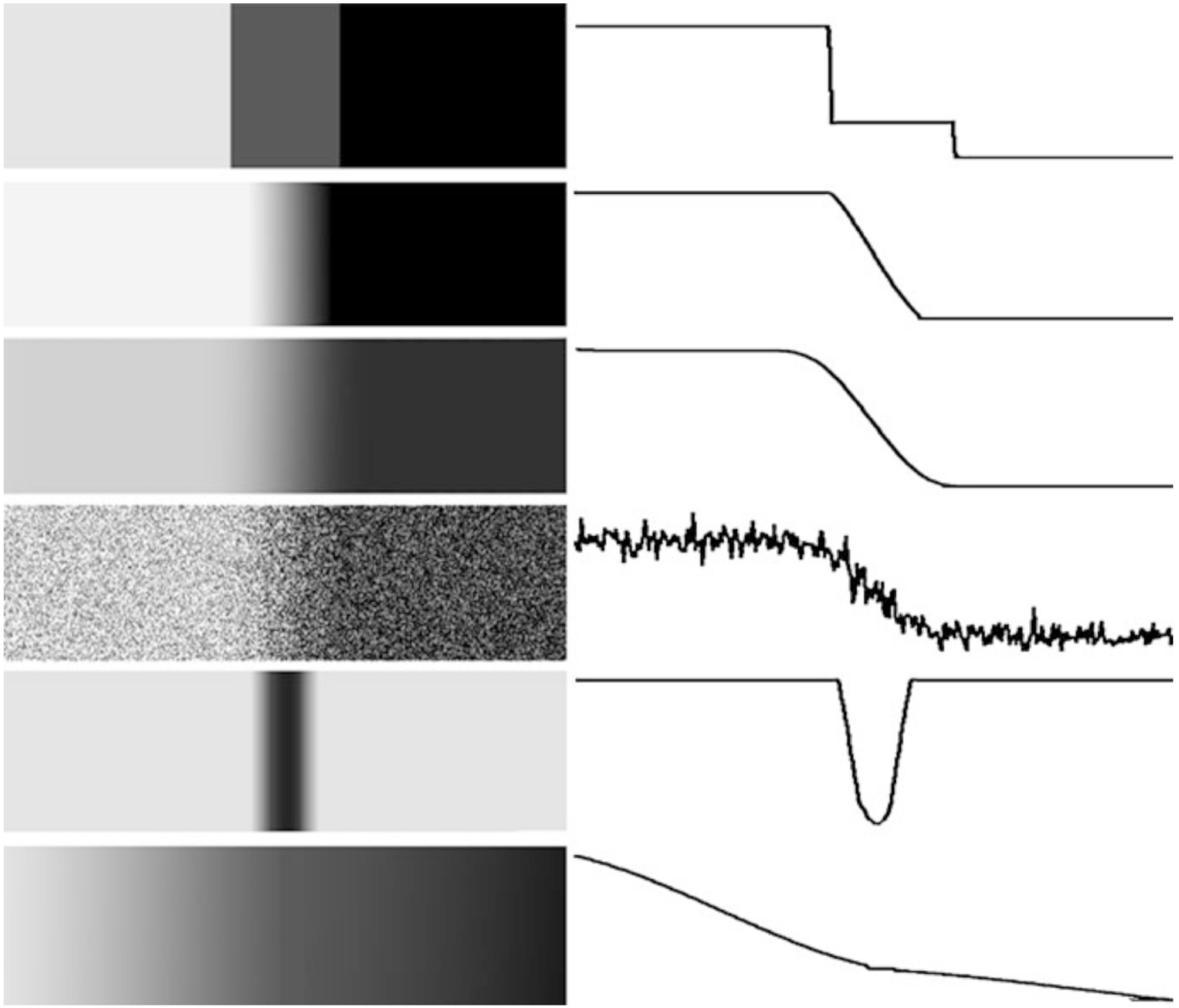


Image credit: Andrew Ng

EDGES

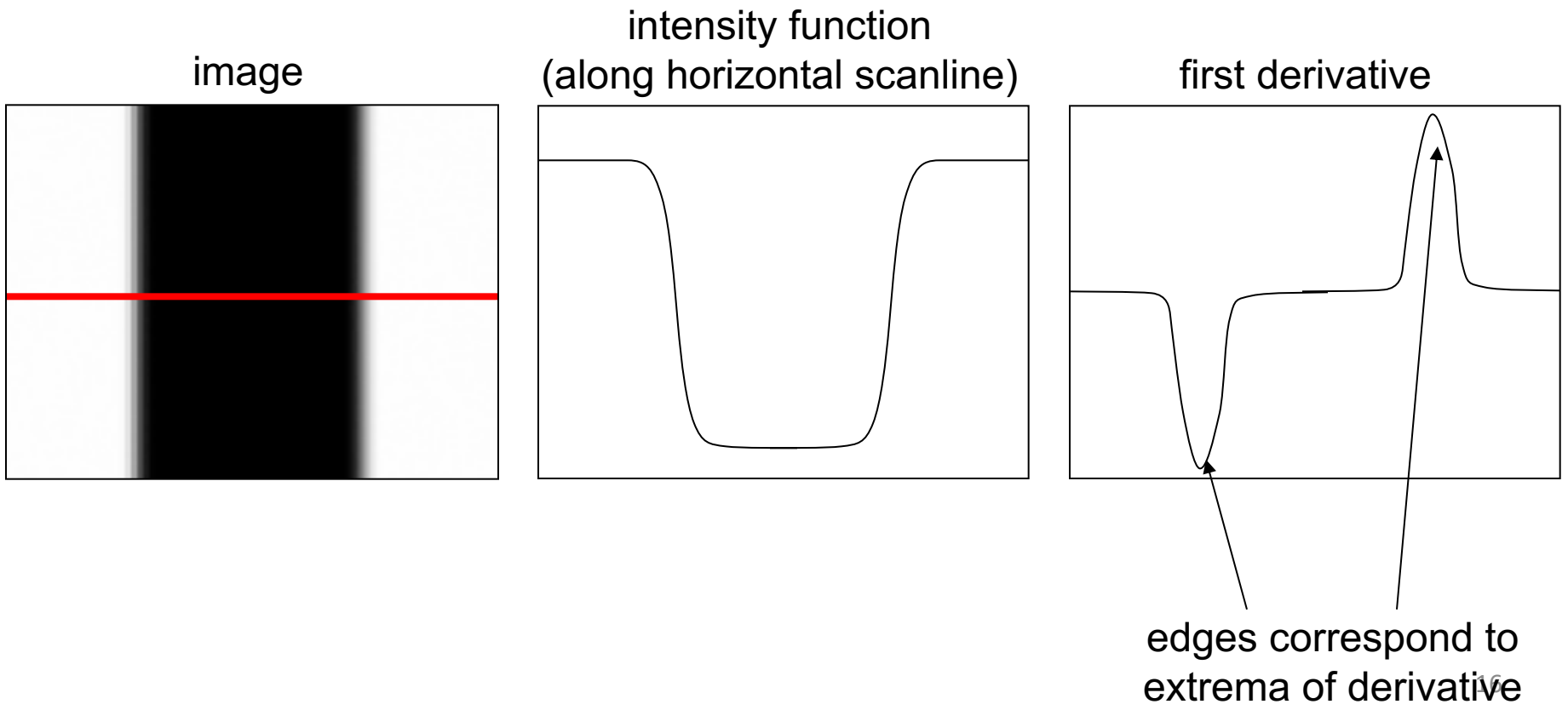
- Discontinuities in images are features that are often useful for initializing an image analysis procedure.
- Edges are important information for understanding an image; by moving “non-edge” data we also simplify the data.

Edge Models



Characterizing Edges

- An edge is a place of rapid change in the image intensity function



Derivatives and Average

- Derivative: rate of change
- Examples:
  - Speed is a rate of change of a distance, X=V.t
  - Acceleration is a rate of change of speed, V=a.t

Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

Example: $y = x^2 + x^4$

$$\frac{dy}{dx} = 2x + 4x^3$$

Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Discrete Derivative / Finite Difference

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$ Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$ Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$ Central difference

Example: Finite Difference

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

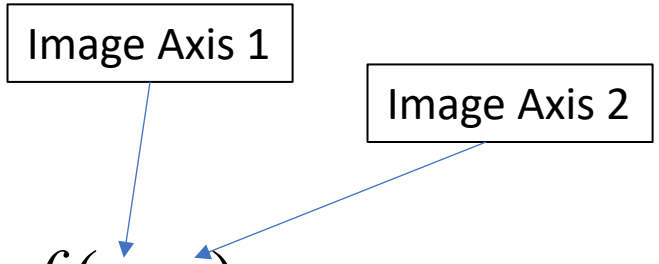
[-1 1] [-1 1] [-1 1] [-1 1] [-1 1] [-1 1] [-1 1]

$$f'(x) = \quad 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

Derivative Masks

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivative in 2-D



Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

Derivative of Images

Derivative masks (filters):

$$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(Vertical) Edge Detection

4	1	4	0	1	2
2	7	4	3	0	1
0	6	5	4	2	2
5	2	1	0	4	2
1	5	3	5	6	6
3	2	2	1	7	5

Convolution

*

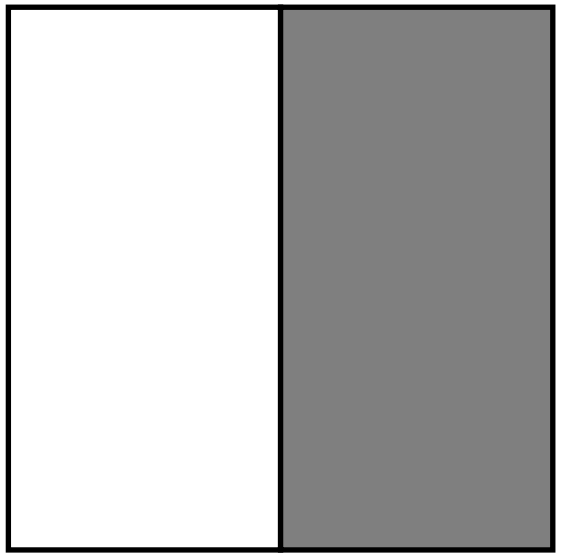
1	0	-1
1	0	-1
1	0	-1

=

-7	7	10	2
-3	8	4	2
-3	-4	-3	-1
3	3	-11	-7

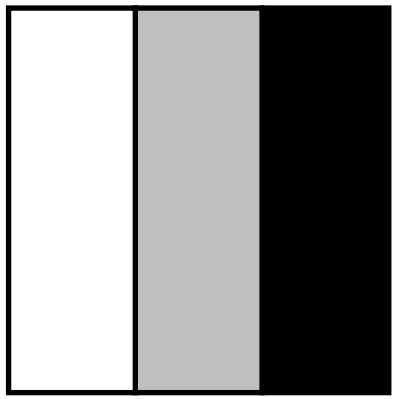
Edge Detection (1)

20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0



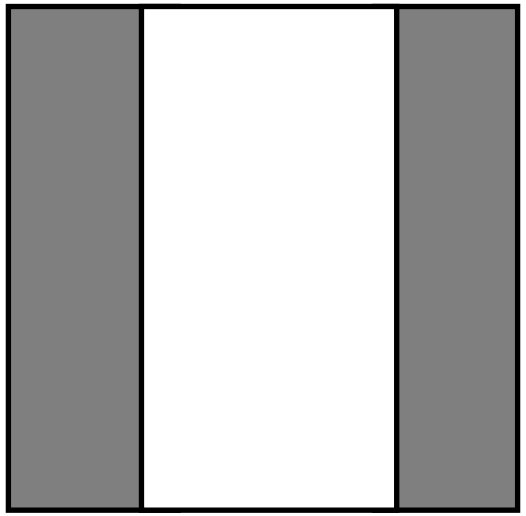
*

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



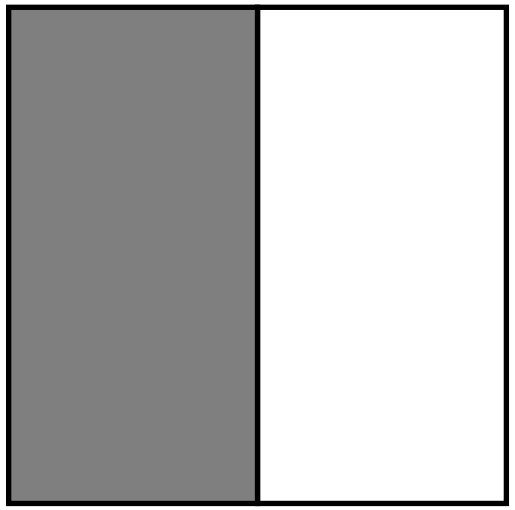
=

0	**60**	**60**	0
0	**60**	**60**	0
0	**60**	**60**	0
0	**60**	**60**	0



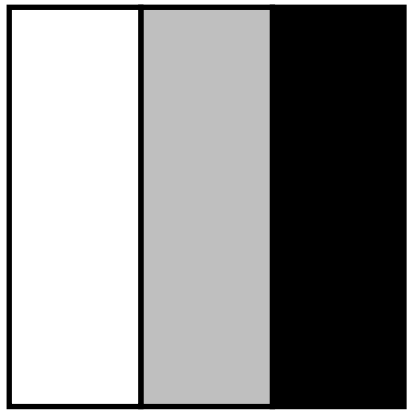
Edge Detection (2)

0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**



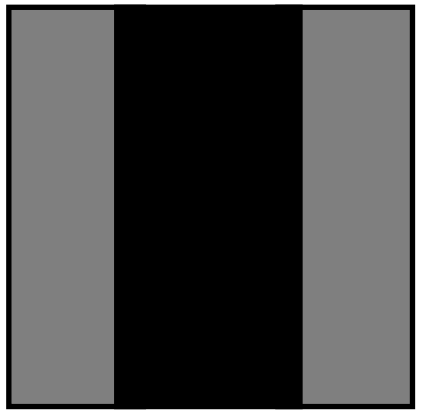
*

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

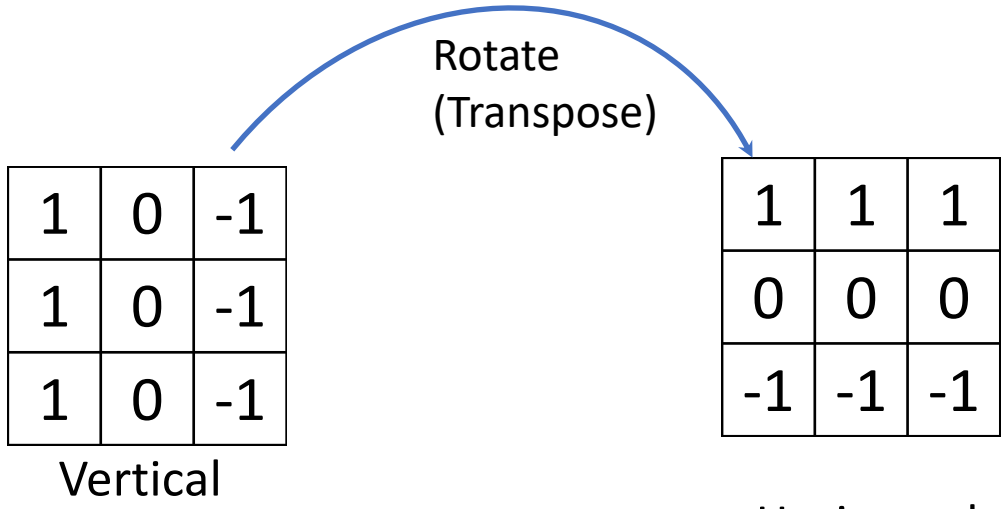


=

0	**-60**	**-60**	0
0	**-60**	**-60**	0
0	**-60**	**-60**	0
0	**-60**	**-60**	0



Vertical vs. Horizontal Edge Detection



20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0
20	**20**	**20**	0	0	0
0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**
0	0	0	**20**	**20**	**20**

*

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

=

0	0	0	0
60	**20**	**-20**	**-60**
60	**20**	**-20**	**-60**
0	0	0	0

More Filters

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

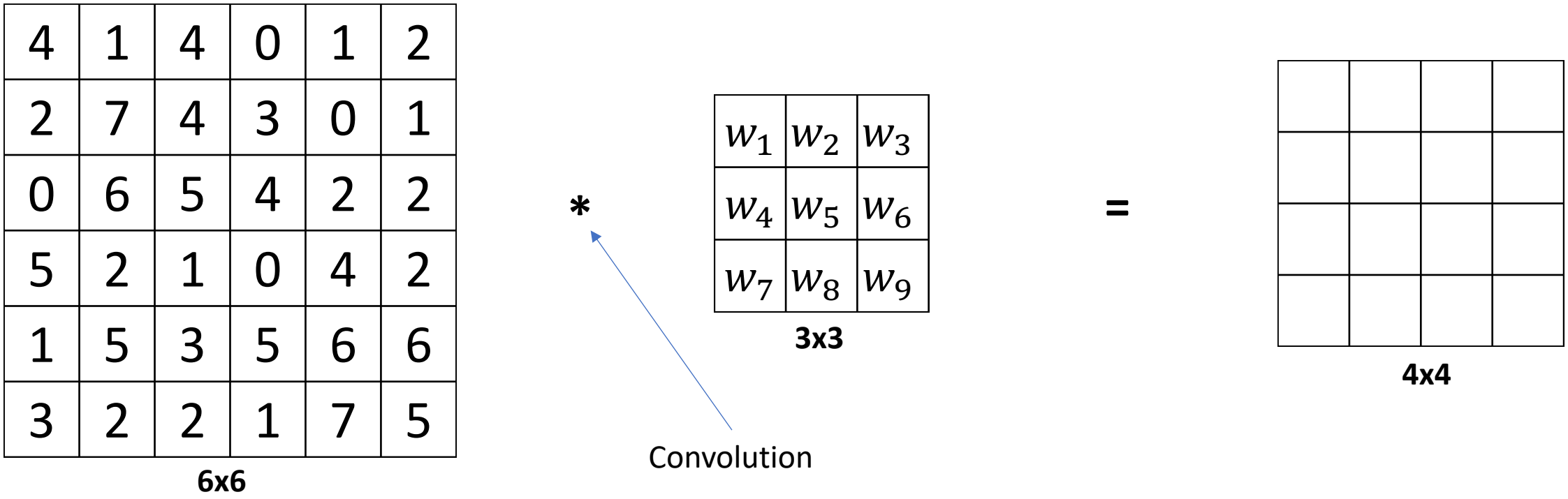
1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-2

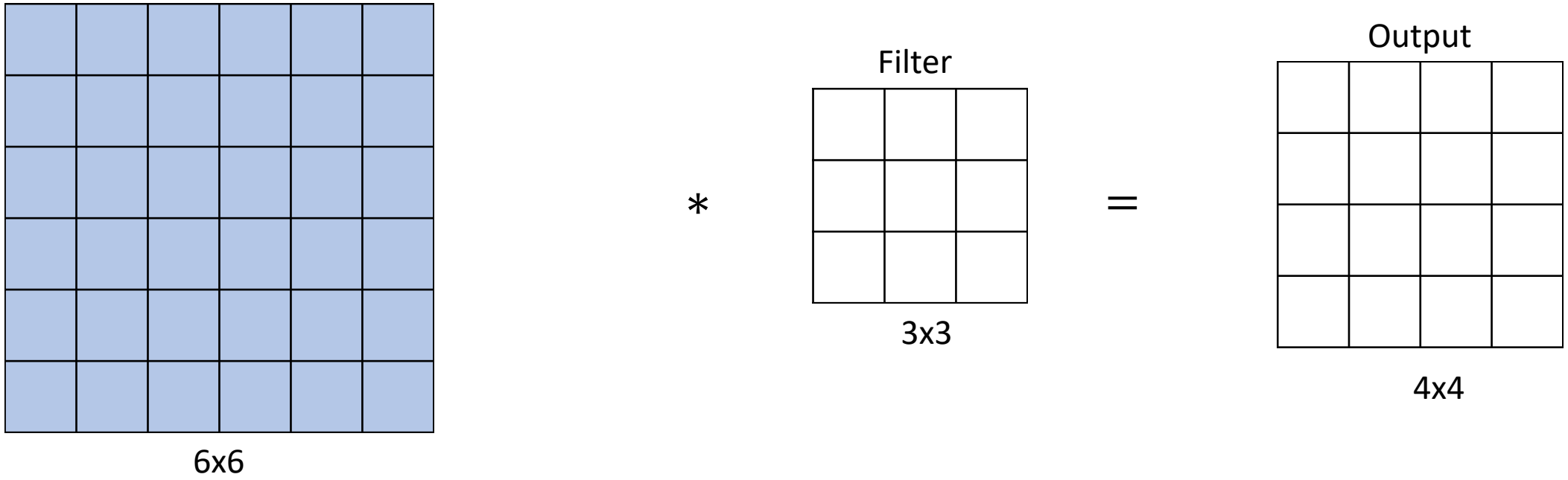
Scharr filter

Which filter to pick? (Why to pick! Learn it)



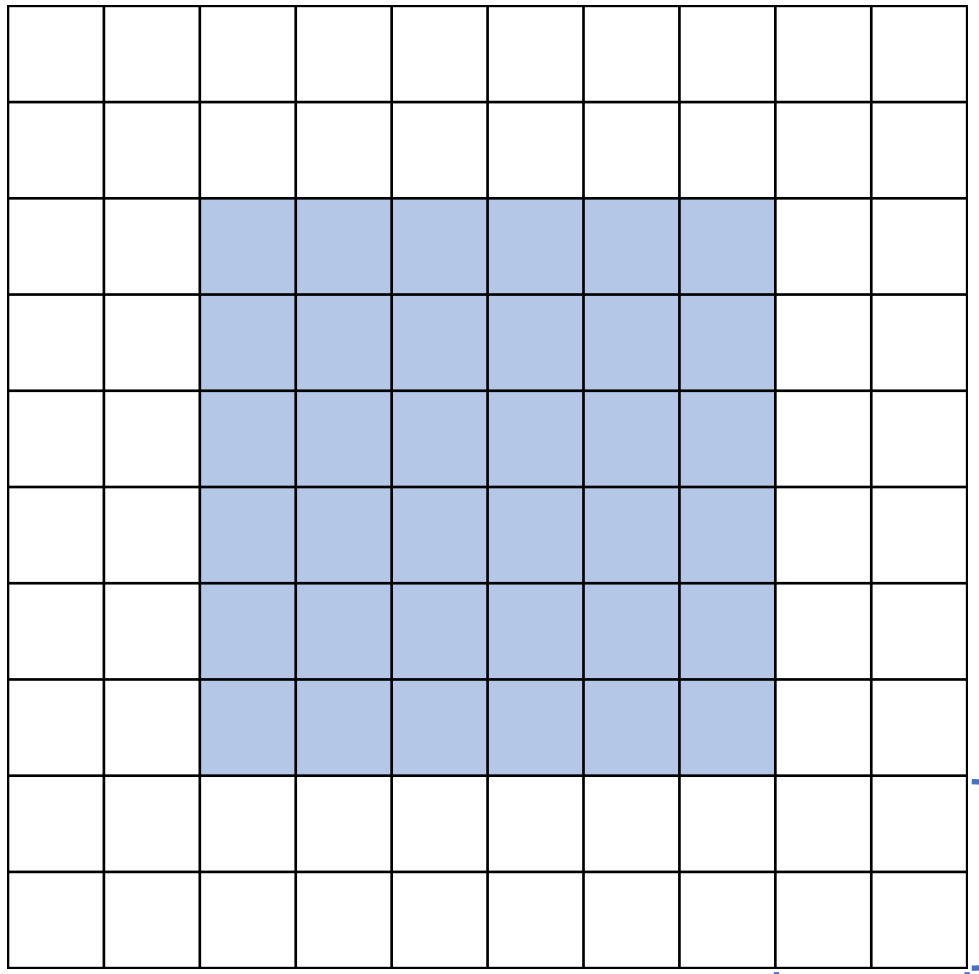
Padding

Remember the original example:



Padding

Original image: 6 x 6 = n x n



p = 2

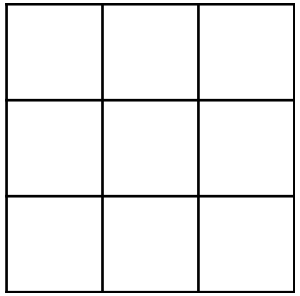
Padded image: 10 x 10

p = 2

p = padding

*

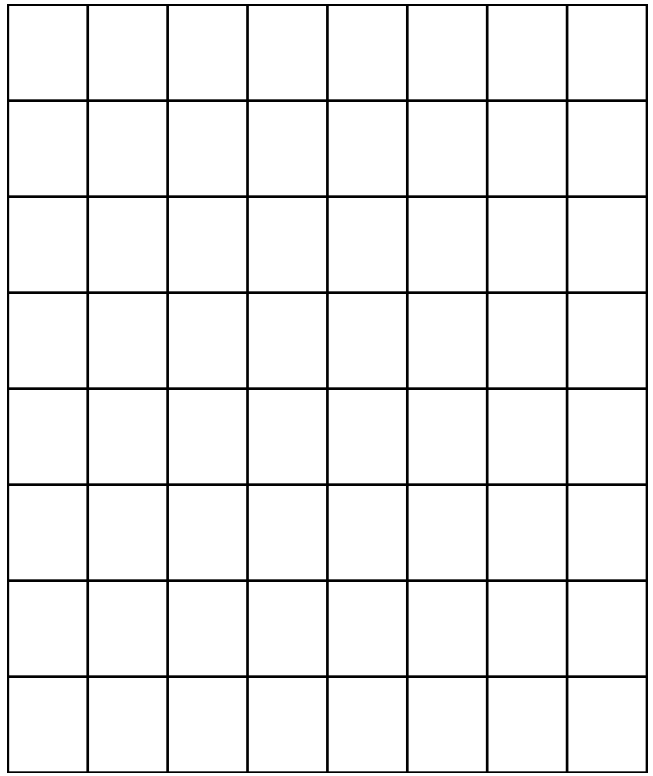
Filter



3 x 3

f x f

=



Output: 8 x 8

Final output dims = (n-f+2p+1) x (n-f+2p+1)

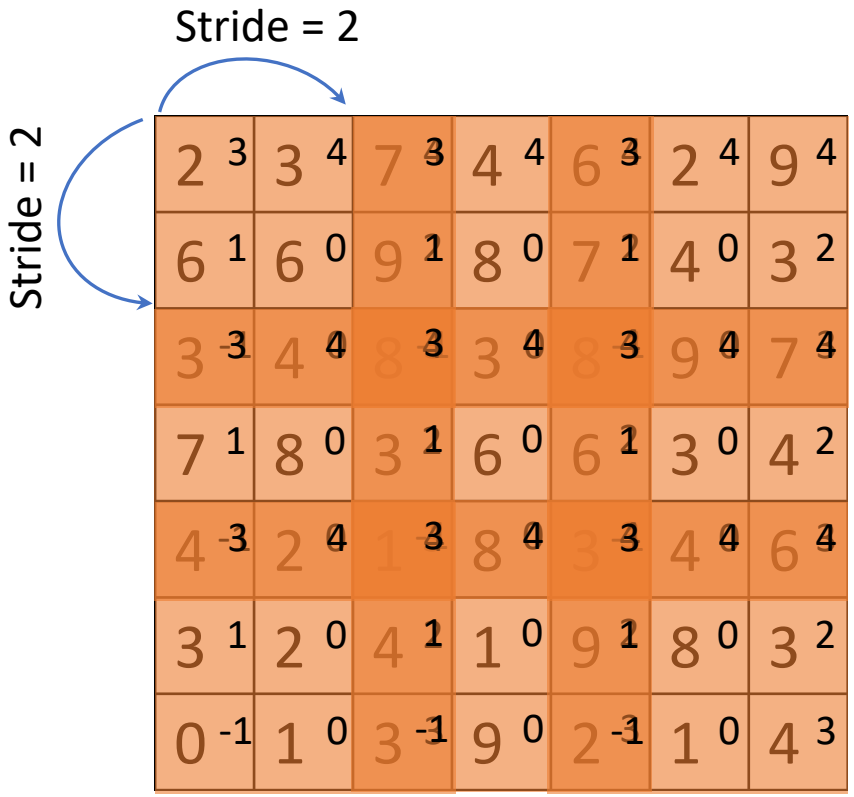
“Valid” & “Same” Convolutions

“Valid”: if no padding!

“Same”: Pad such that the output size matches the input size.

$$P = (f-1)/2$$

Strided convolution



*

3	4	4
1	0	2
-1	0	3

=

91	**100**	**83**
69	**91**	**127**
44	**72**	**74**

Output Size (Output "Shape")

Image dims: $n \times n$

Filter dims: $f \times f$

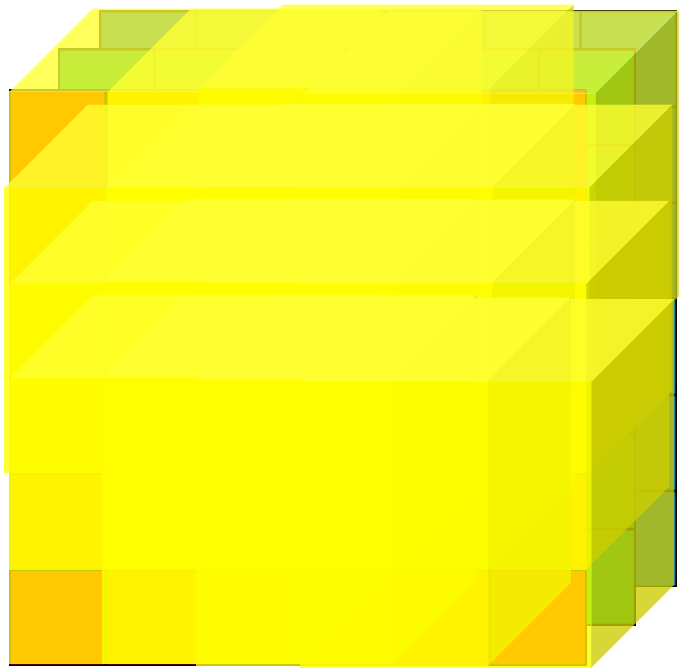
Padding: p

Stride: s

Floor (round to the lowest integer)

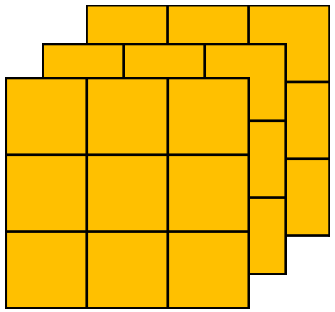
Output dims: $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Convolutions on RGB image



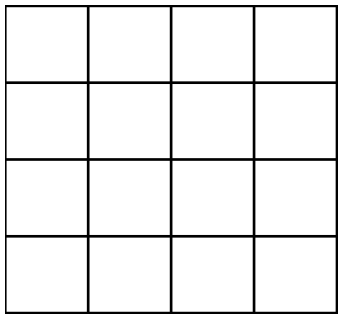
6x6x3

*



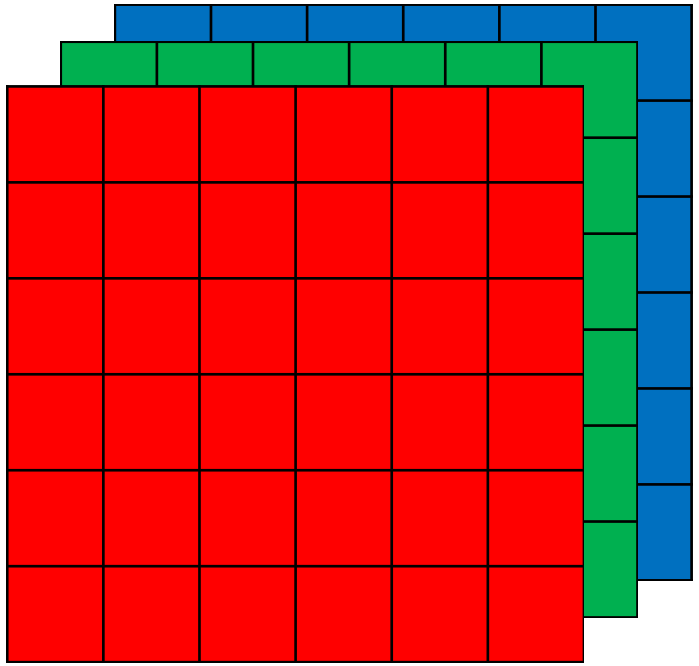
3x3x3

=



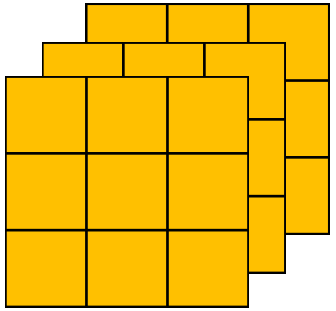
4 x 4

Multiple filters



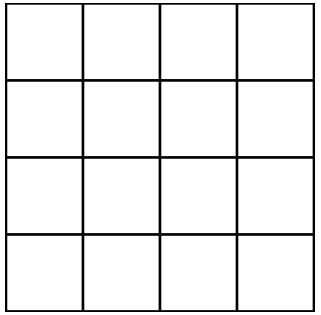
6 x 6 x 3

*



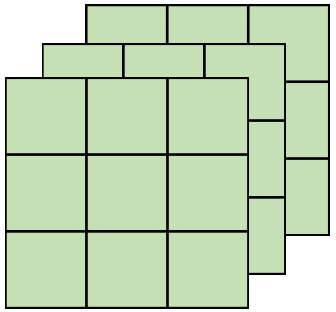
3 x 3 x 3

=



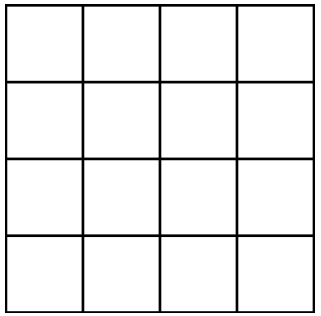
4 x 4

*



3 x 3 x 3

=



4 x 4

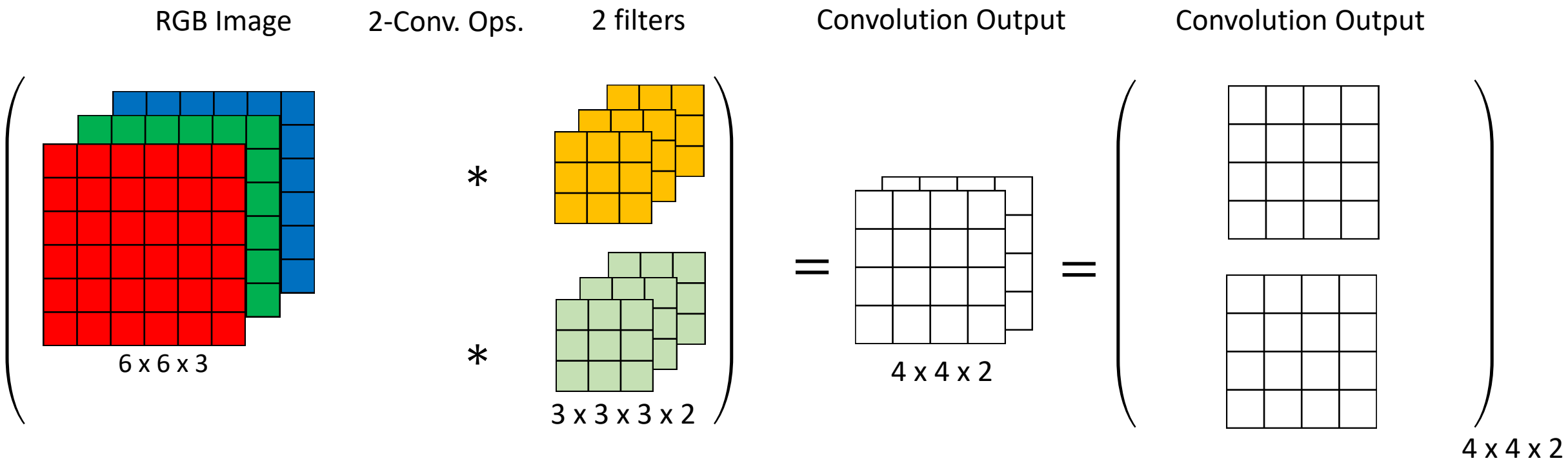
Number of channels

Number of filters

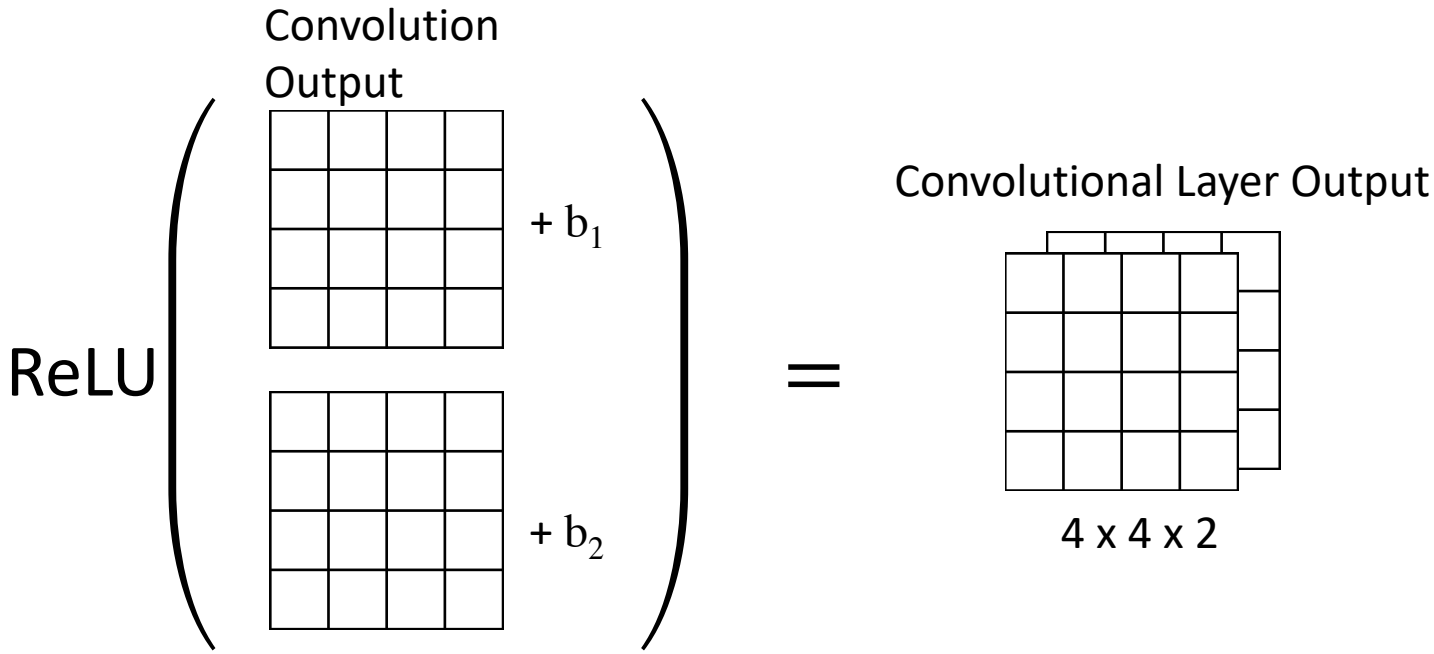
Number of filters

$n \times n \times n_c$ * $f \times f \times n_c \times 2$ = $(n-f+1) \times (n-f+1) \times 2$

Multiple filters



A Convolutional Layer of CNN (ConvNet)



Convolutional Layer Output = ReLU ($(RGBImage * filter) + b$)

Output of a neuron in a FC NN: a = ReLU ($\mathbf{w}^T\mathbf{x} + b$) in FC NN

Similar notation!
(but not equal)

Input – Output Dimensions for l^{th} Convolutional Layer

Input (One image): $n_H^{[l-1]}$ x $n_W^{[l-1]}$ x $n_C^{[l-1]}$ ⟶ Output: $n_H^{[l]}$ x $n_W^{[l]}$ x $n_C^{[l]}$

Padding size: $p^{[l]}$ x $p^{[l]}$

Stride size: $s^{[l]}$ x $s^{[l]}$

Filter size: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]} = One\_Filter_{Dims} \times n_C^{[l]}$

(Number of Weights = Filter size), Bias size: $n_C^{[l]}$

Output dims (for one image) : $\left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \times \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \times n_C^{[l]}$

$n_H^{[l]}$ x $n_W^{[l]}$ x $n_C^{[l]}$

The output dimensions for m input samples $A^{[l]}$: m x $n_H^{[l]}$ x $n_W^{[l]}$ x $n_C^{[l]}$

This lecture contains material from Andrew Ng and Ulas Bagci