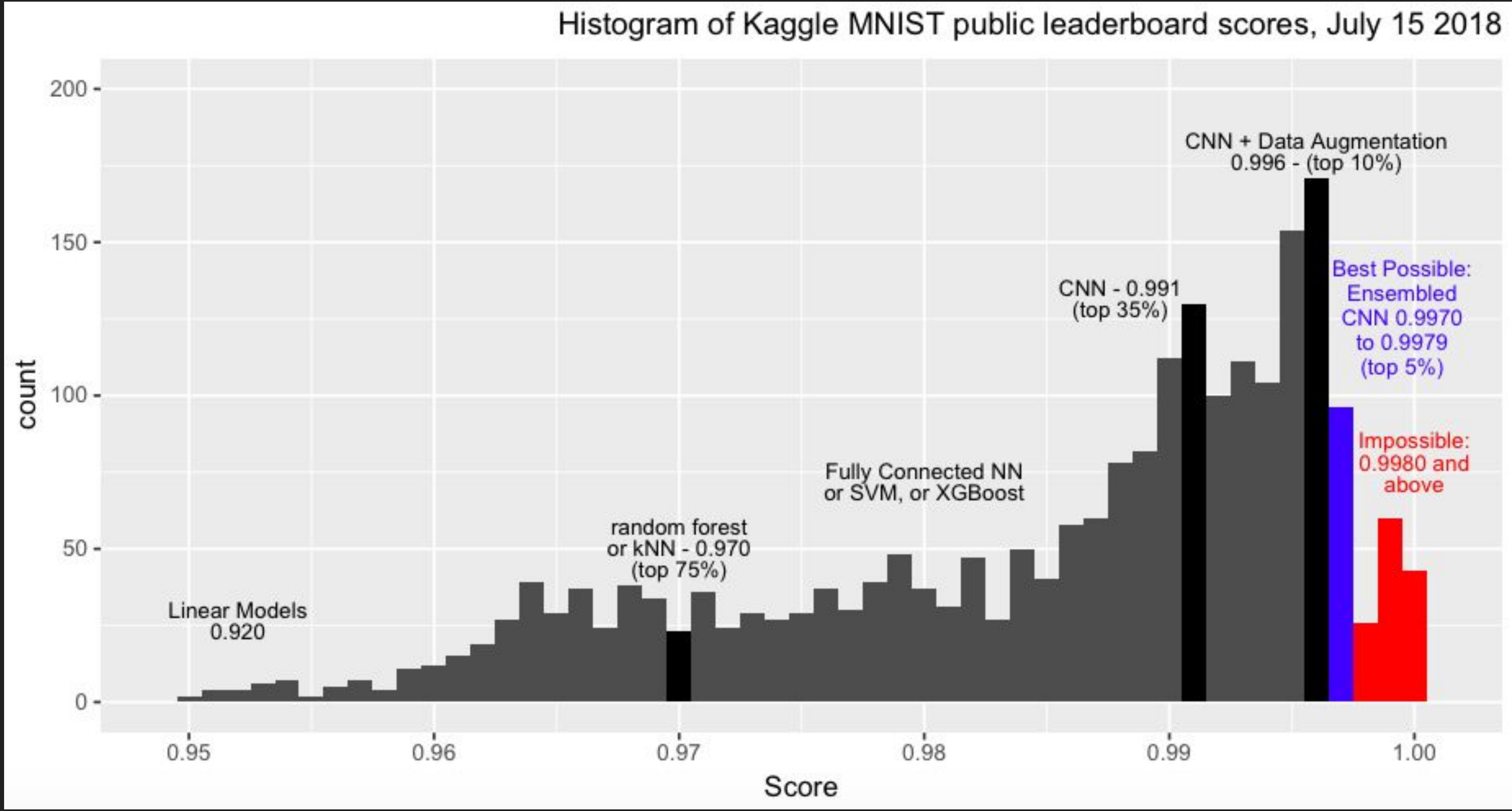
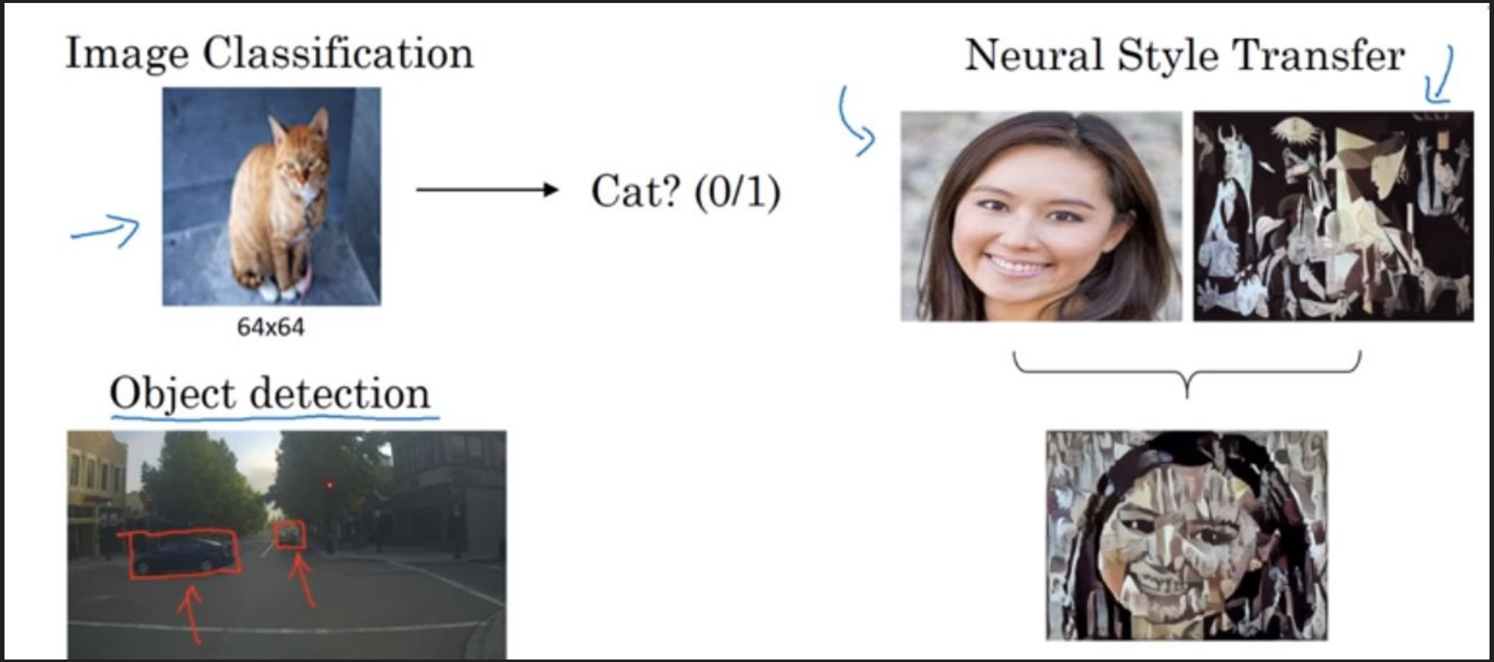
Convolutional Neural Networks

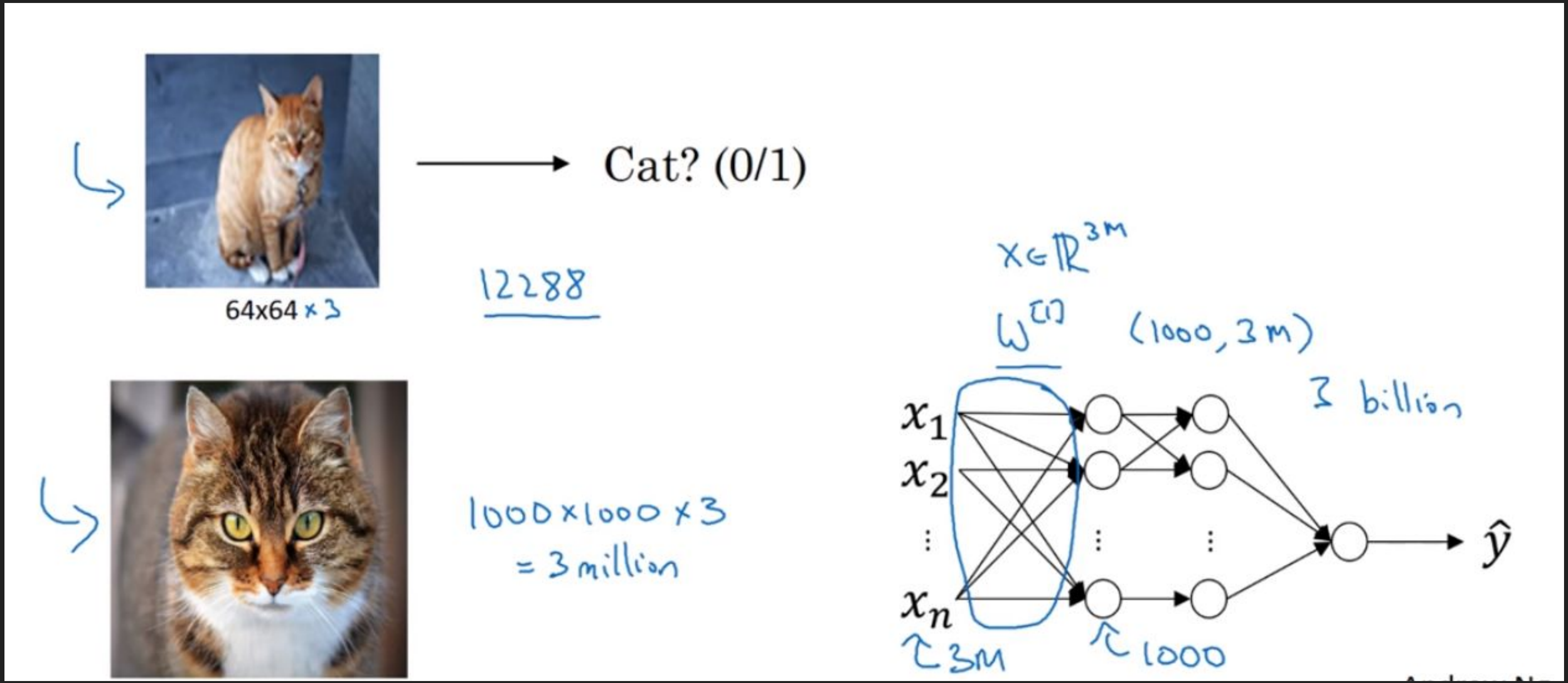
Sources: Elements of Stat. Learning, Andrew Ng, A. Mueller


Histogram of Kaggle MNIST public leaderboard scores, July 15 2018
count
200
150
100
50
0
0.95
0.96
0.97
0.98
0.99
1.00
Score
Linear Models
0.920
random forest
or kNN - 0.970
(top 75%)
Fully Connected NN
or SVM, or XGBoost
CNN - 0.991
(top 35%)
CNN + Data Augmentation
0.996 - (top 10%)
Best Possible:
Ensembled
CNN 0.9970
to 0.9979
(top 5%)
Impossible:
0.9980 and
above

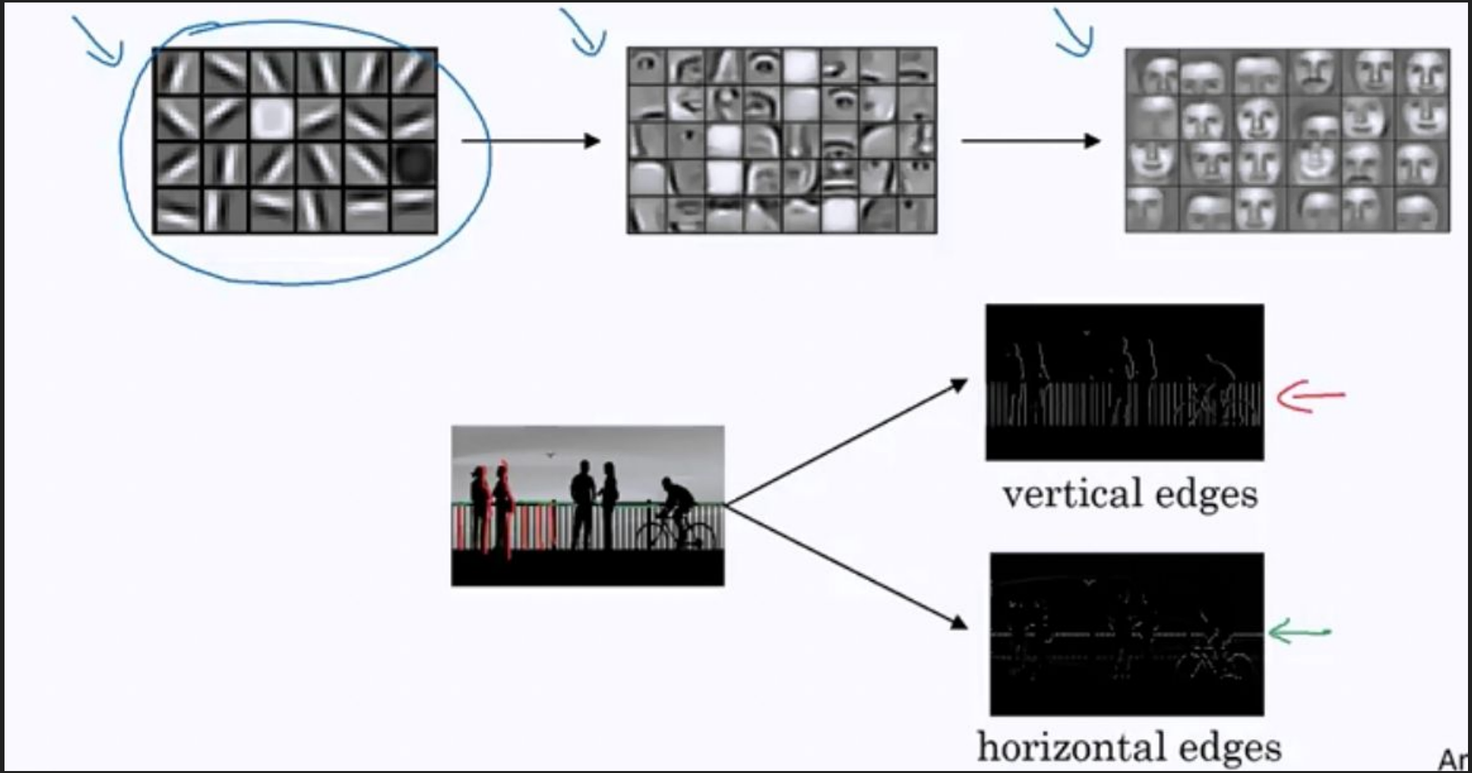
Computer Vision Problems:



Deep Learning on Larger Images:



Edge Detection:



Vertical edge detection

$$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$$

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} \underset{\text{"convolution"}}{*} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} \boxed{-5} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

6×6

3×3
<u>filter</u>

4×4

Vertical edge detection

$$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$$

$$\underbrace{\begin{bmatrix} 3 & 0^{1} & 1^{0} & 2^{-1} & 7 & 4 \\ 1 & 5^{1} & 8^{0} & 9^{-1} & 3 & 1 \\ 2 & 7^{1} & 2^{0} & 5^{-1} & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix}}_{6\times6} \overset{\text{“convolution”}}{*} \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{\substack{3\times3 \\ \text{filter}}} = \underbrace{\begin{bmatrix} \boxed{-5} & \boxed{-4} & & \\ & & & \\ & & & \\ & & & \end{bmatrix}}_{4\times4}$$

Vertical edge detection

$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$

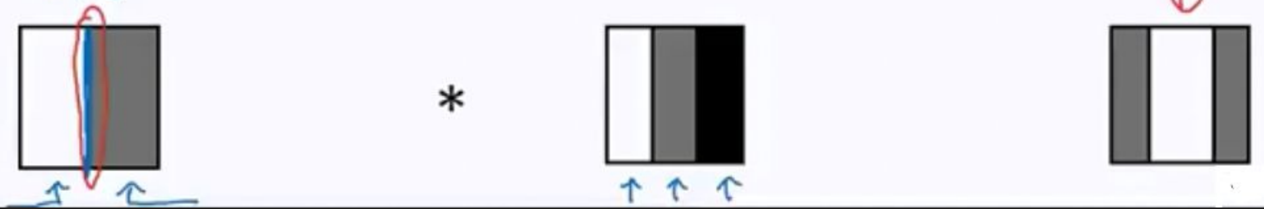
"convolution"

$$\underbrace{\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix}}_{\boxed{6\times6}} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{\substack{3\times3 \\ \text{filter}}} = \underbrace{\begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}}_{4\times4}$$

python: conv-forward
tensorflow: tf.nn.conv2d
keras: Conv2D

Vertical edge detection

$$\underbrace{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}}_{6\times 6} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{3\times 3} = \underbrace{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}_{4\times 4}$$



More on vertical edge detection

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \boxed{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}$$

$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \boxed{\begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}}

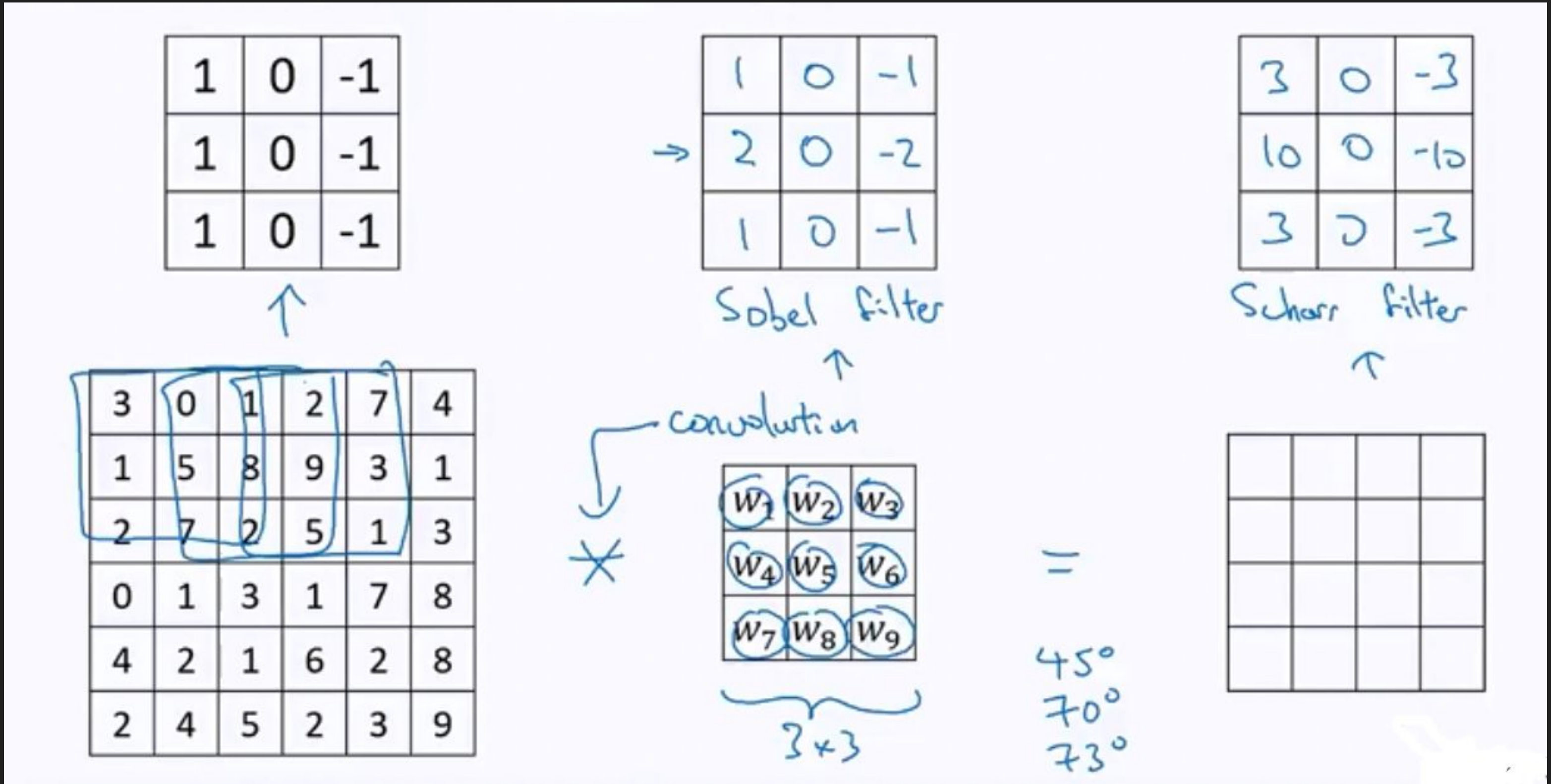
Vertical and Horizontal edge detection

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

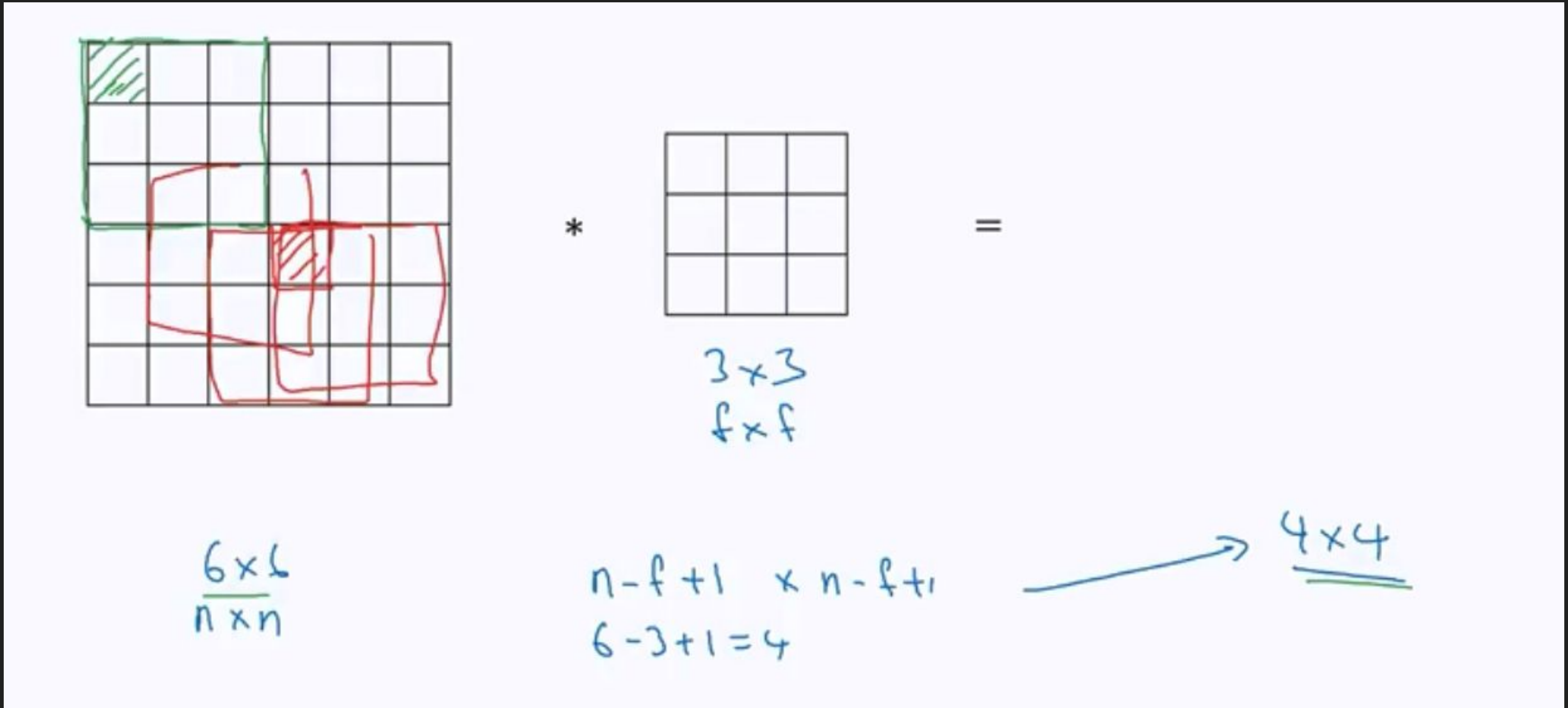
Vertical — Horizontal

$$\underbrace{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix}}_{6 \times 6} * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 30 & 10 & -10 & -30 \\ 30 & 10 & -10 & -30 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

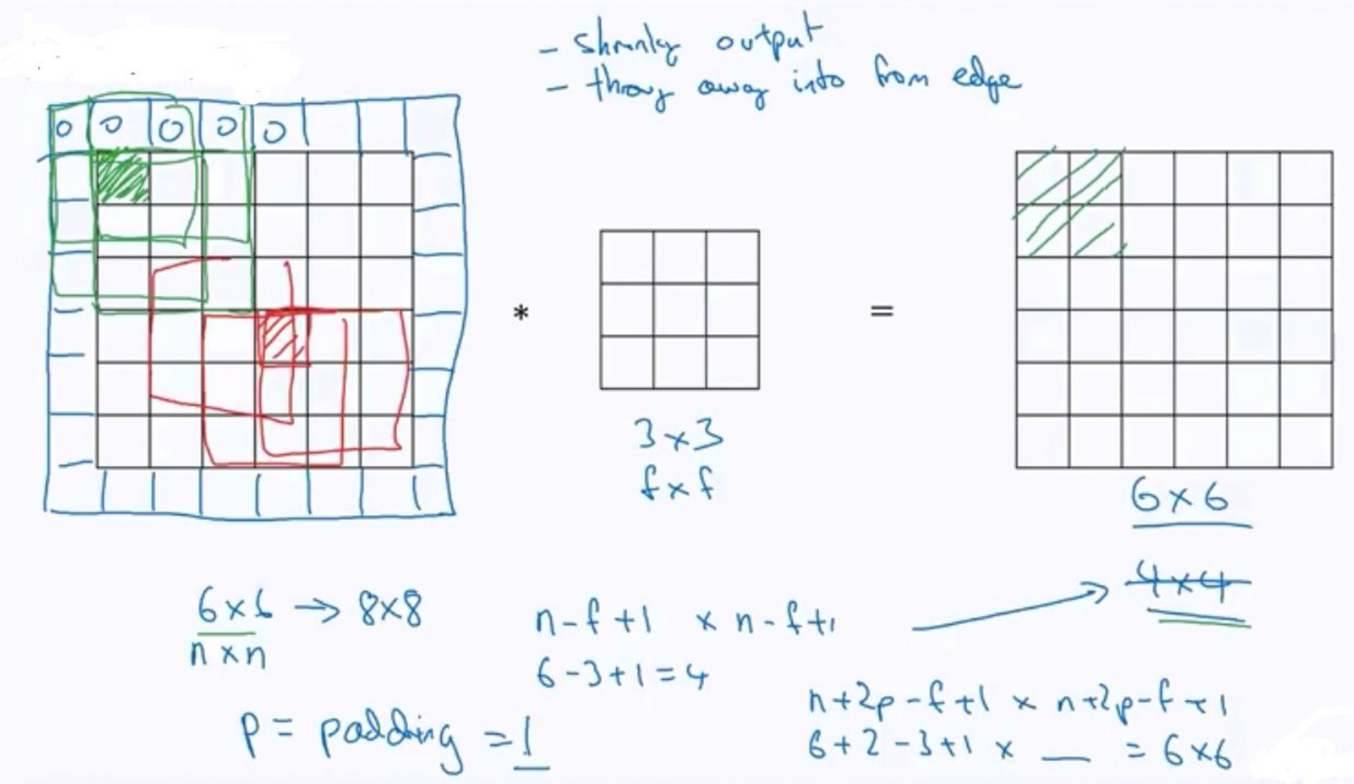
Learning to detect edges:



Padding:

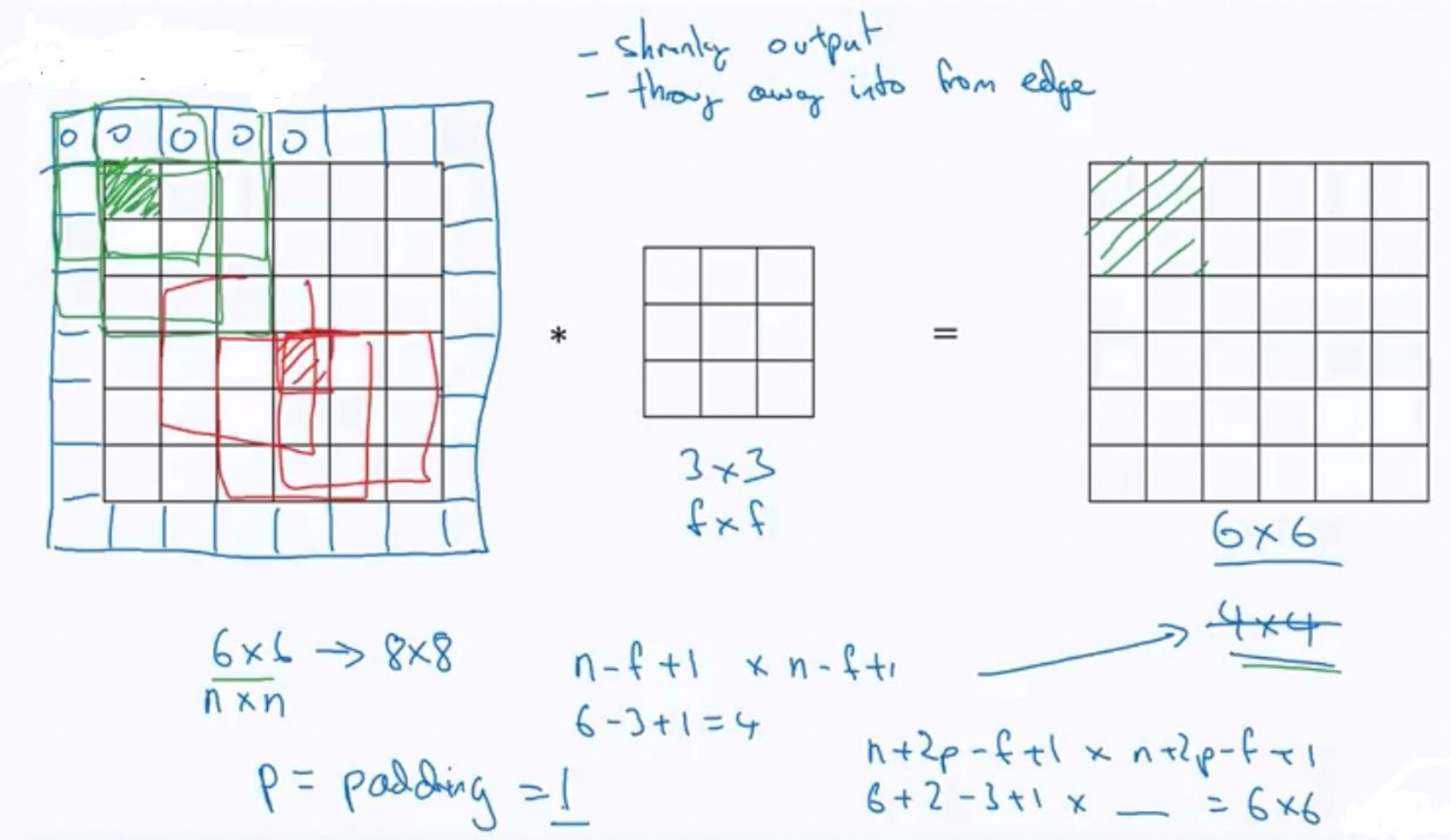


Padding:



- shrinky output
- throw away info from edge
*
=
3x3
fxf
6x6
~~4x4~~
6x6 → 8x8
n x n
n-f+1 x n-f+1
6-3+1=4
p = padding = 1
n+2p-f+1 x n+2p-f+1
6+2-3+1 x ___ = 6x6

Padding:


- shrinking output
- throw away info from edge
0 0 0 0 0
*
=
3×3
f×f
6×6
4×4
6×6 → 8×8
n×n
n-f+1 × n-f+1
6-3+1=4
p = padding = 1
n+2p-f+1 × n+2p-f+1
6+2-3+1 × ___ = 6×6

Valid versus same padding:

no padding

"Valid": $n \times n \quad \ast \quad f \times f \quad \longrightarrow \quad n-f+1 \times n-f+1$

$6 \times 6 \quad \ast \quad 3 \times 3 \quad \longrightarrow \quad 4 \times 4$

"Same": Pad so that output size is the same as the input size.

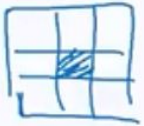
$n+2p-f+1 \times n+2p-f+1$

$\cancel{n}+2p-f+1 = \cancel{n} \Rightarrow \boxed{p = \frac{f-1}{2}}$

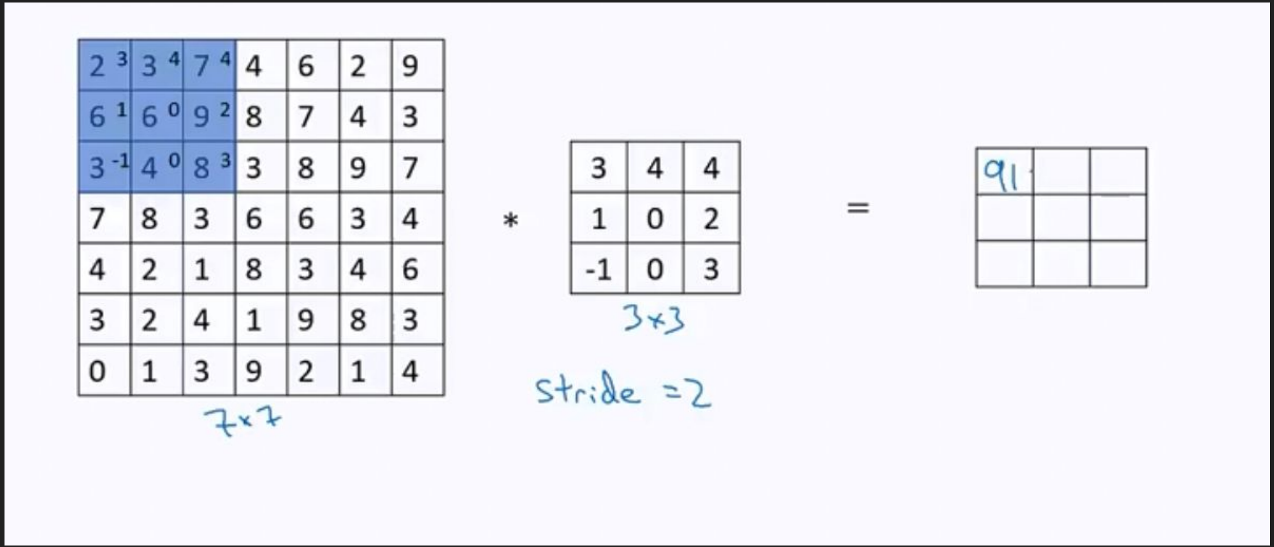
$3 \times 3 \quad p = \frac{3-1}{2} = 1$ | 5×5 $f=5$ $p=2$

f is usually odd

1×1
3×3
5×5
7×7



Strided convolution:

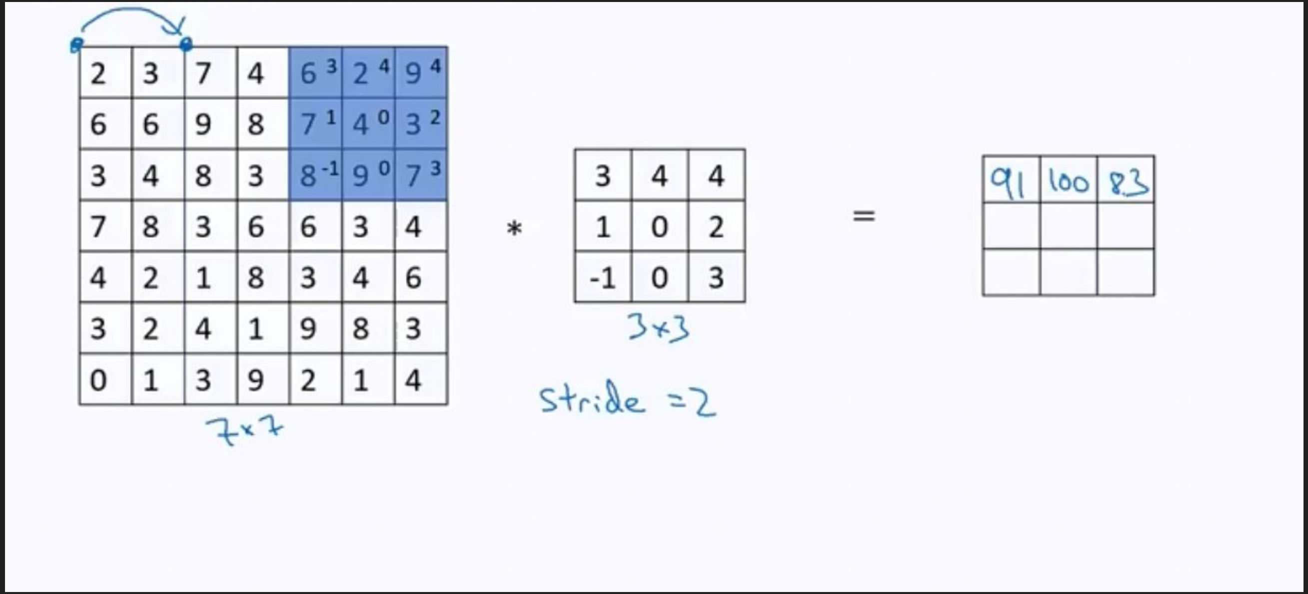


Strided convolution:

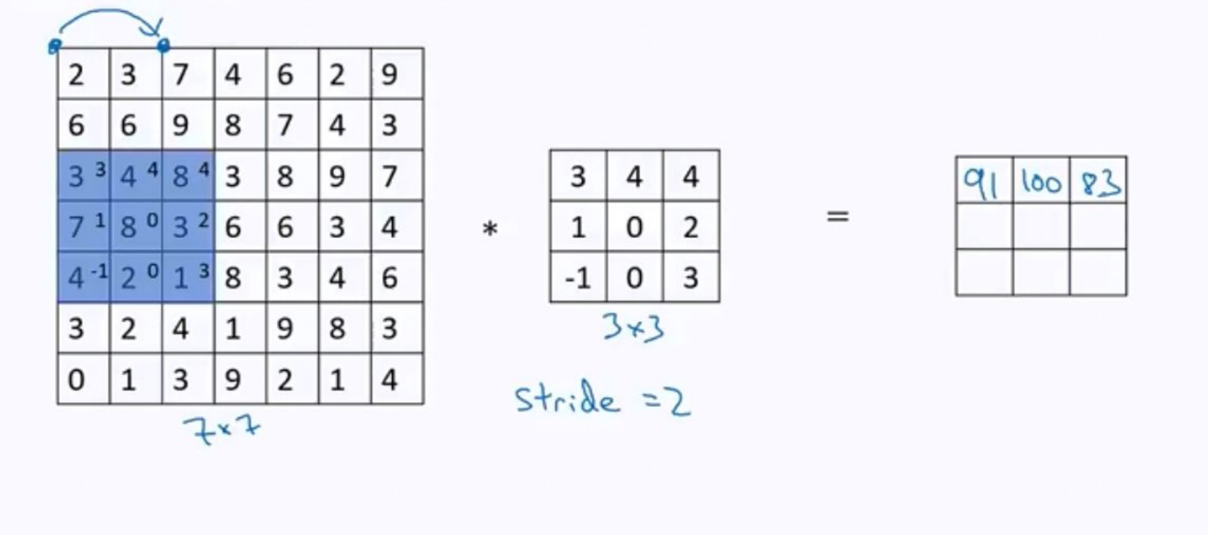
$$\underbrace{\begin{bmatrix} 2 & 3 & 7^{3} & 4^{4} & 6^{4} & 2 & 9 \\ 6 & 6 & 9^{1} & 8^{0} & 7^{2} & 4 & 3 \\ 3 & 4 & 8^{-1} & 3^{0} & 8^{3} & 9 & 7 \\ 7 & 8 & 3 & 6 & 6 & 3 & 4 \\ 4 & 2 & 1 & 8 & 3 & 4 & 6 \\ 3 & 2 & 4 & 1 & 9 & 8 & 3 \\ 0 & 1 & 3 & 9 & 2 & 1 & 4 \end{bmatrix}}_{7 \times 7} * \underbrace{\begin{bmatrix} 3 & 4 & 4 \\ 1 & 0 & 2 \\ -1 & 0 & 3 \end{bmatrix}}_{3 \times 3} = \begin{bmatrix} 91 & 100 & \\ & & \\ & & \end{bmatrix}$$

stride = 2

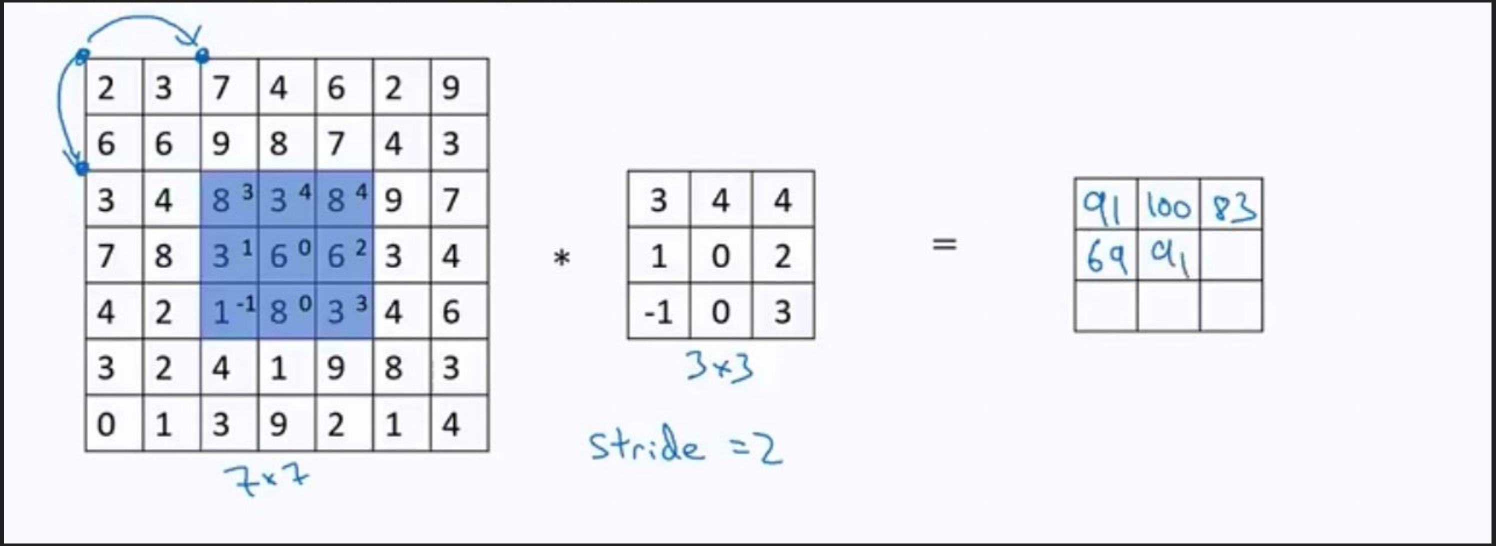
Strided convolution:



Strided convolution:



Strided convolution:



Strided convolution:

$$\underbrace{\begin{bmatrix} 2&3&7&4&6&2&9\\ 6&6&9&8&7&4&3\\ 3&4&8&3&8^{3}&9^{4}&7^{4}\\ 7&8&3&6&6^{1}&3^{0}&4^{2}\\ 4&2&1&8&3^{-1}&4^{0}&6^{3}\\ 3&2&4&1&9&8&3\\ 0&1&3&9&2&1&4 \end{bmatrix}}_{7\times7} * \underbrace{\begin{bmatrix} 3&4&4\\ 1&0&2\\ -1&0&3 \end{bmatrix}}_{3\times3} = \begin{bmatrix} 91&100&83\\ 69&91&12.\\ && \end{bmatrix}$$

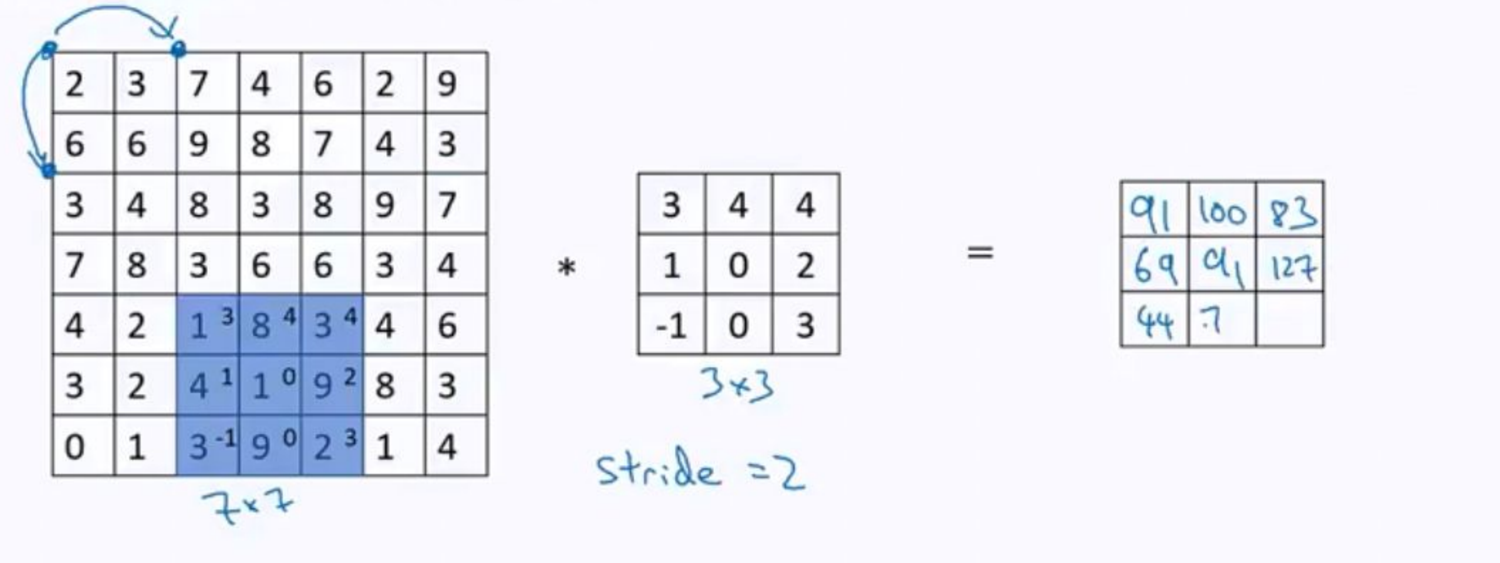
stride = 2

Strided convolution:

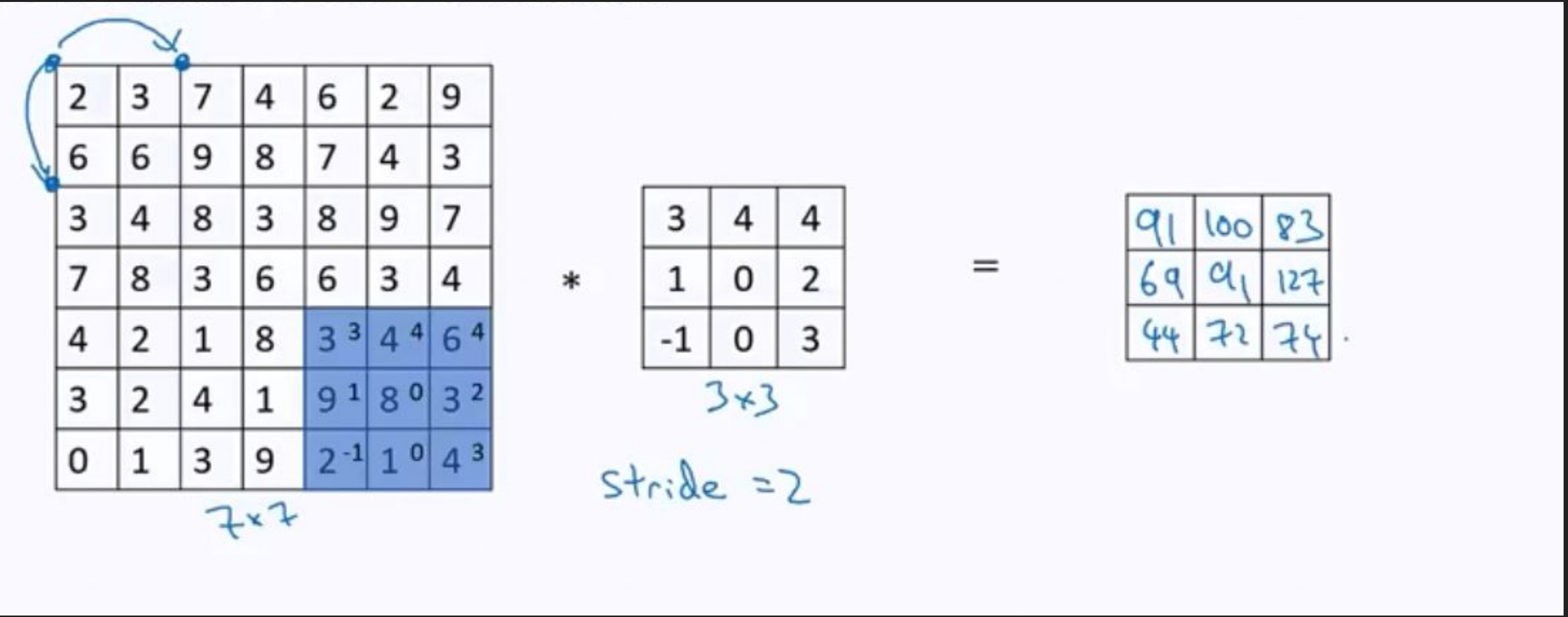
$$\underbrace{\begin{bmatrix} 2 & 3 & 7 & 4 & 6 & 2 & 9 \\ 6 & 6 & 9 & 8 & 7 & 4 & 3 \\ 3 & 4 & 8 & 3 & 8 & 9 & 7 \\ 7 & 8 & 3 & 6 & 6 & 3 & 4 \\ 4^{3} & 2^{4} & 1^{4} & 8 & 3 & 4 & 6 \\ 3^{1} & 2^{0} & 4^{2} & 1 & 9 & 8 & 3 \\ 0^{-1} & 1^{0} & 3^{3} & 9 & 2 & 1 & 4 \end{bmatrix}}_{7\times 7} * \underbrace{\begin{bmatrix} 3 & 4 & 4 \\ 1 & 0 & 2 \\ -1 & 0 & 3 \end{bmatrix}}_{3\times 3} = \begin{bmatrix} 91 & 100 & 83 \\ 69 & 91 & 127 \\ & & \end{bmatrix}$$

stride = 2

Strided convolution:



Strided convolution:



Strided convolution:

$$
\begin{bmatrix}
2 & 3 & 7 & 4 & 6 & 2 & 9 \\
6 & 6 & 9 & 8 & 7 & 4 & 3 \\
3 & 4 & 8 & 3 & 8 & 9 & 7 \\
7 & 8 & 3 & 6 & 6 & 3 & 4 \\
4 & 2 & 1 & 8 & 3^{3} & 4^{4} & 6^{4} \\
3 & 2 & 4 & 1 & 9^{1} & 8^{0} & 3^{2} \\
0 & 1 & 3 & 9 & 2^{-1} & 1^{0} & 4^{3}
\end{bmatrix}
*
\begin{bmatrix}
3 & 4 & 4 \\
1 & 0 & 2 \\
-1 & 0 & 3
\end{bmatrix}
=
\begin{bmatrix}
91 & 100 & 83 \\
69 & 91 & 127 \\
44 & 72 & 74
\end{bmatrix}
$$

7×7 $\qquad$ 3×3 $\qquad$ 3×3

stride $= 2$

$\lfloor z \rfloor = \text{floor}(z)$

$n \times n \quad * \quad f \times f$

padding p $\qquad$ stride s

$s = 2$

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

$$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 = 3$$

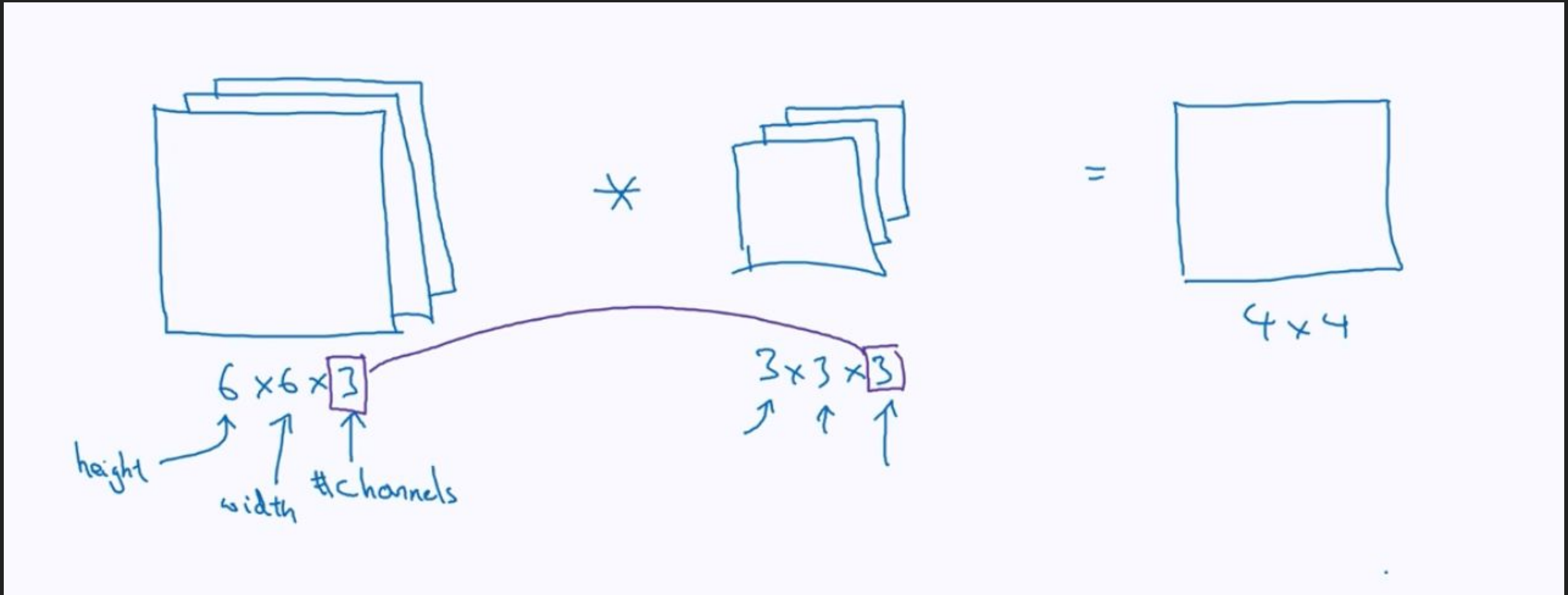
Summary of convolutions:

$n \times n$ image $\quad f \times f$ filter

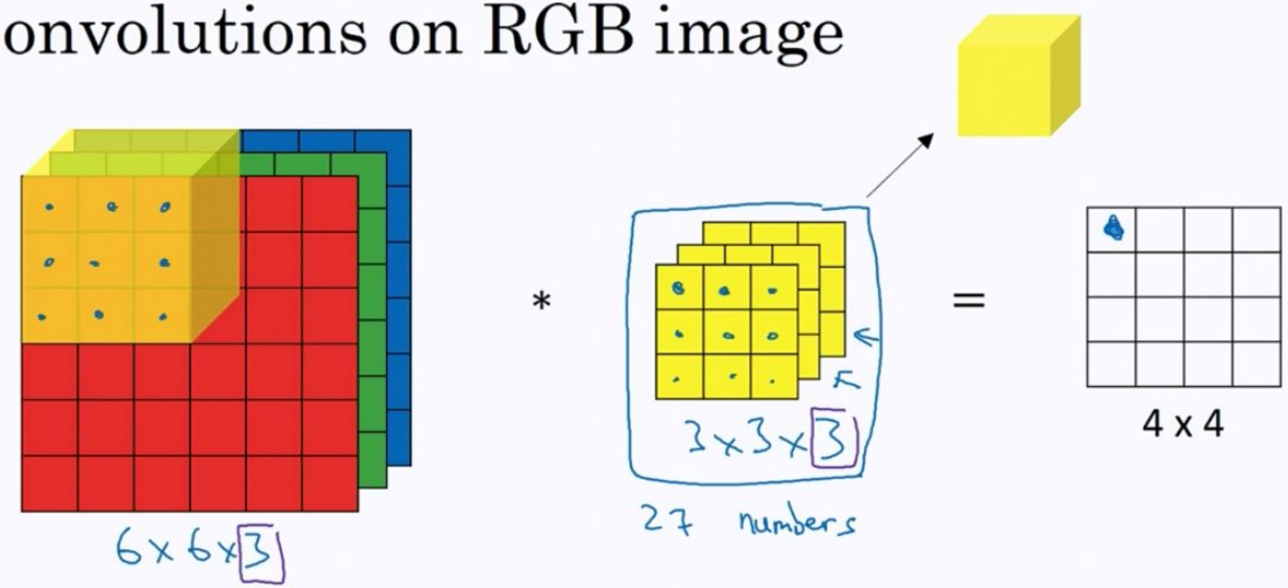
padding $p \quad$ stride s

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

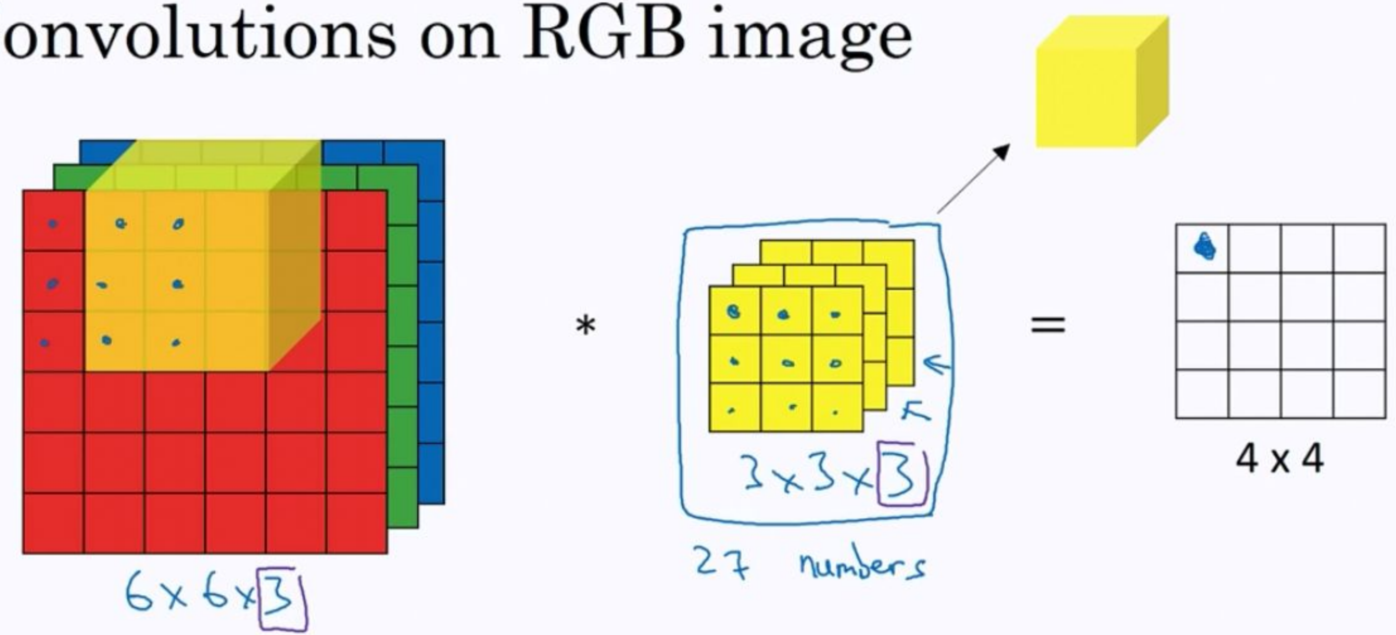
Convolutions on RGB images



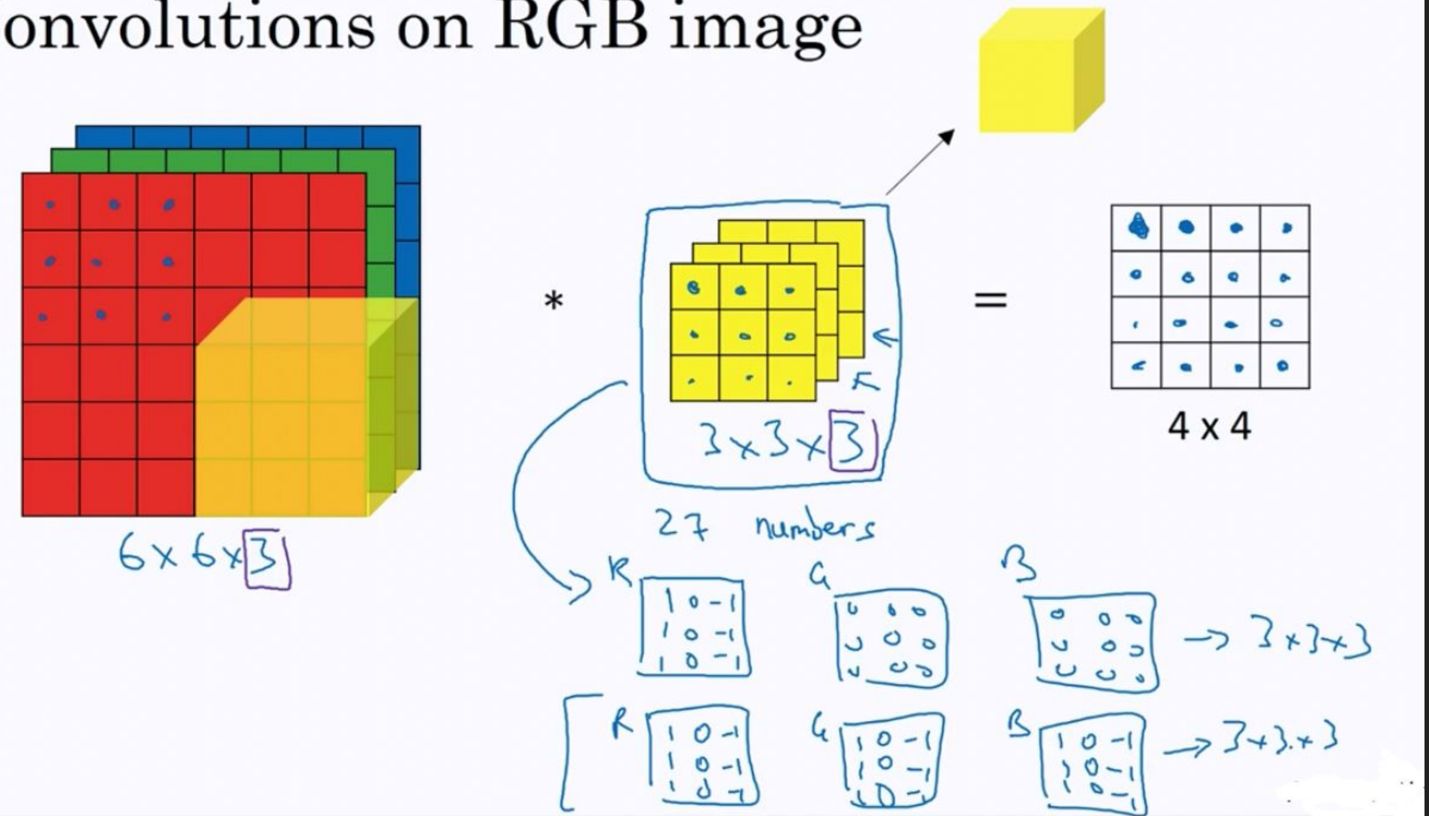
Convolutions on RGB image



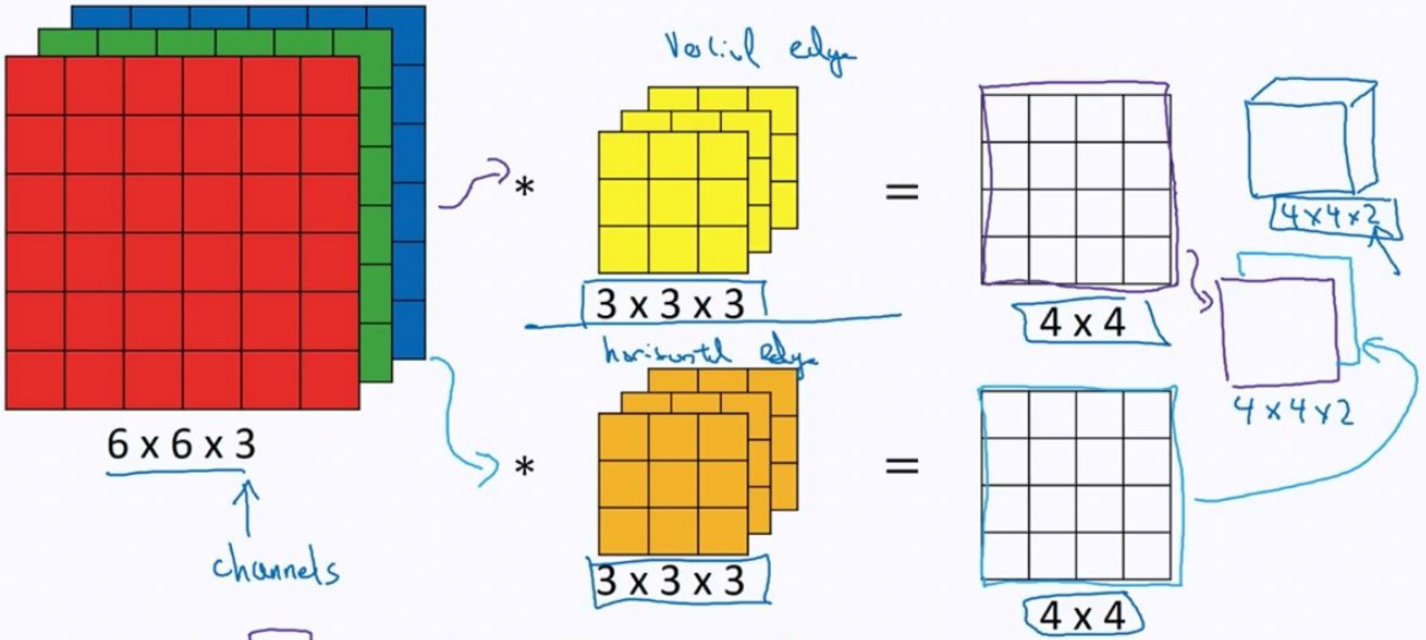
Convolutions on RGB image



Convolutions on RGB image



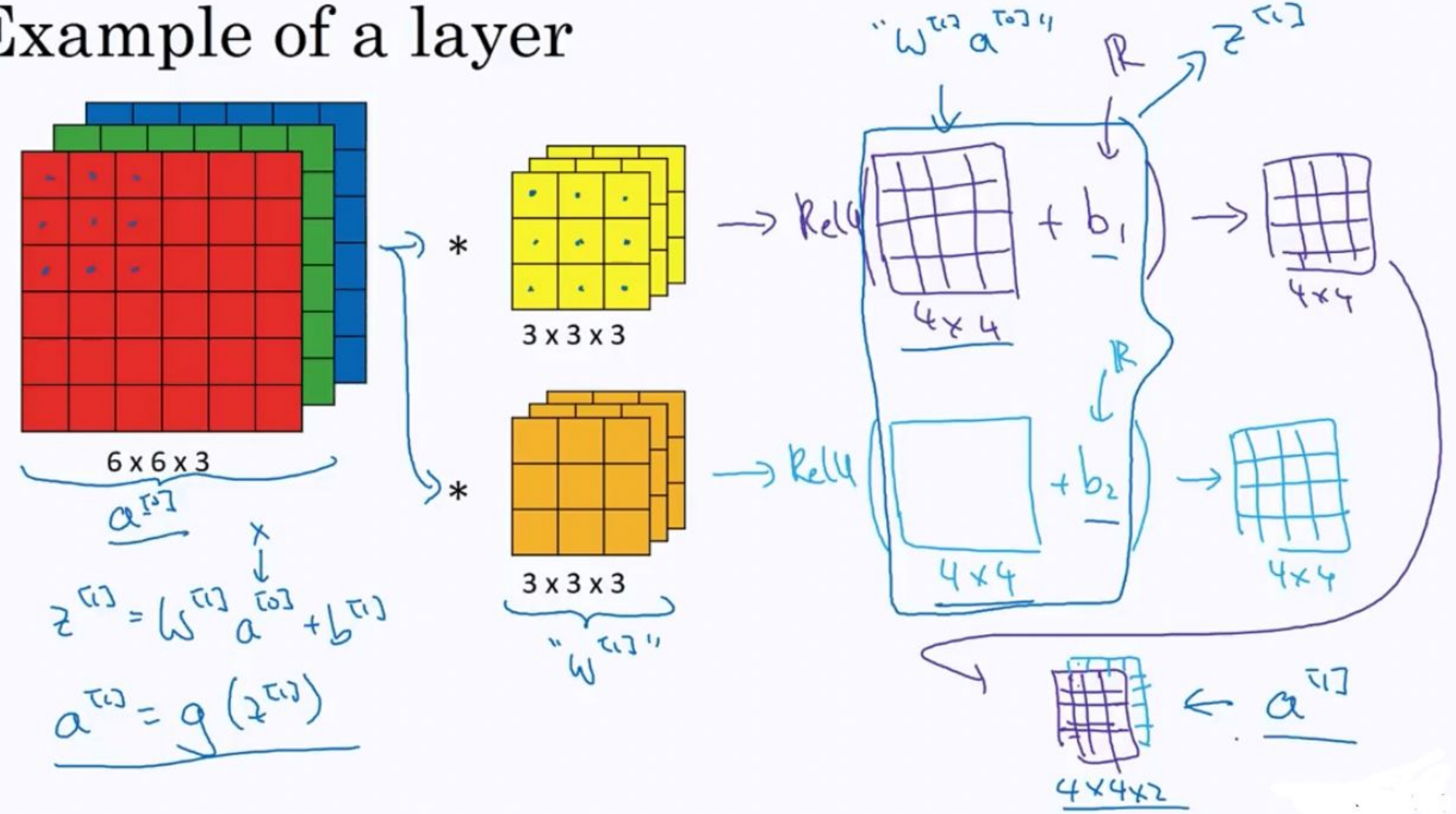
Multiple filters



Summary: $n \times n \times \boxed{n_c} \; * \; f \times f \times \boxed{n_c} \;\longrightarrow\; n-f+1 \times n-f+1 \times n_c'$

$6 \times 6 \times 3 \qquad 3 \times 3 \times 3 \qquad\qquad 4 \times 4 \times 2$ # filters

Example of a layer



Number of parameters in one layer

If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?

$3 \times 3 \times 3$

27 parameters.

+ bias

→ 28 parameters.

280 parameters.

Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

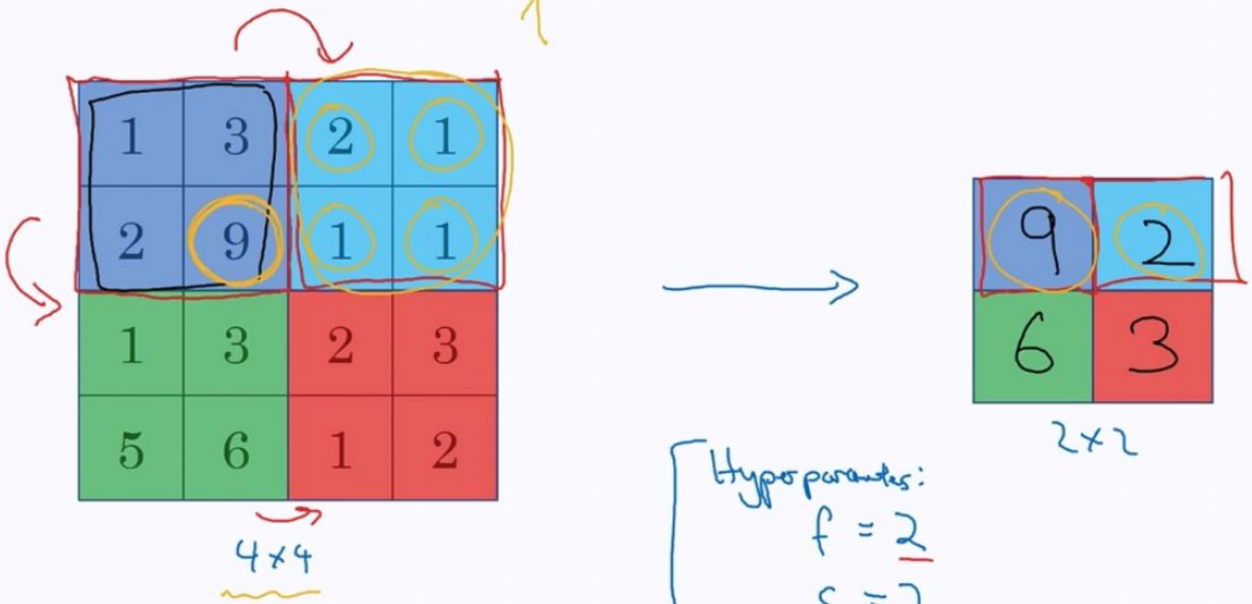
Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_{H/W}^{[l]} = \left\lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

Types of layer in a convolutional network:

- Convolution (CONV) ←
- Pooling (POOL) ←
- Fully connected (FC) ←

Pooling layer: Max pooling



Hyperparameters:

$f = 2$

$s = 2$

No parameters!

Pooling layer: Max pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

$5 \times 5 \times 2$

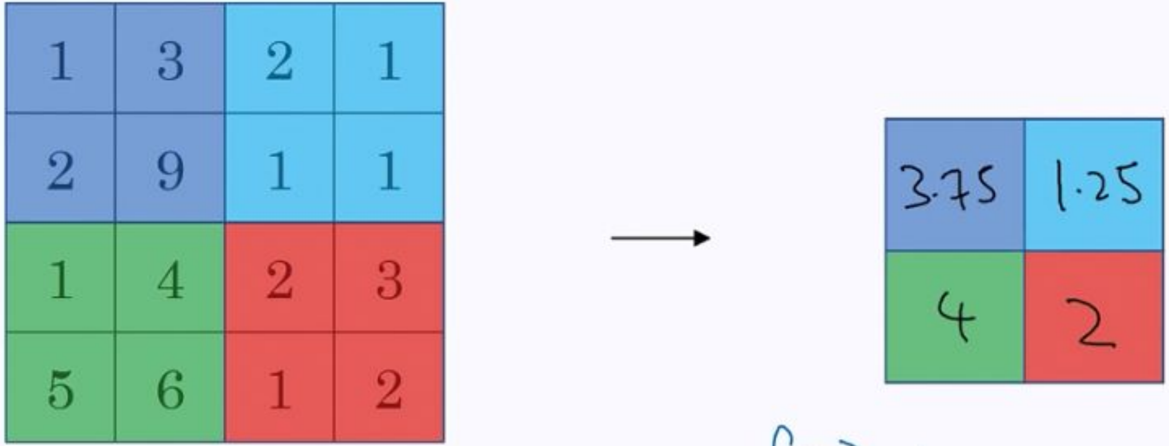
9	9	5
9	9	5
8	6	9

$3 \times 3 \times 2$

$\boxed{\begin{array}{c} f=3 \\ \hline s=1 \end{array}}$

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

Pooling layer: Average pooling



$f=2$

$s=2$

Summary of pooling

Hyperparameters:

f : filter size

s : stride

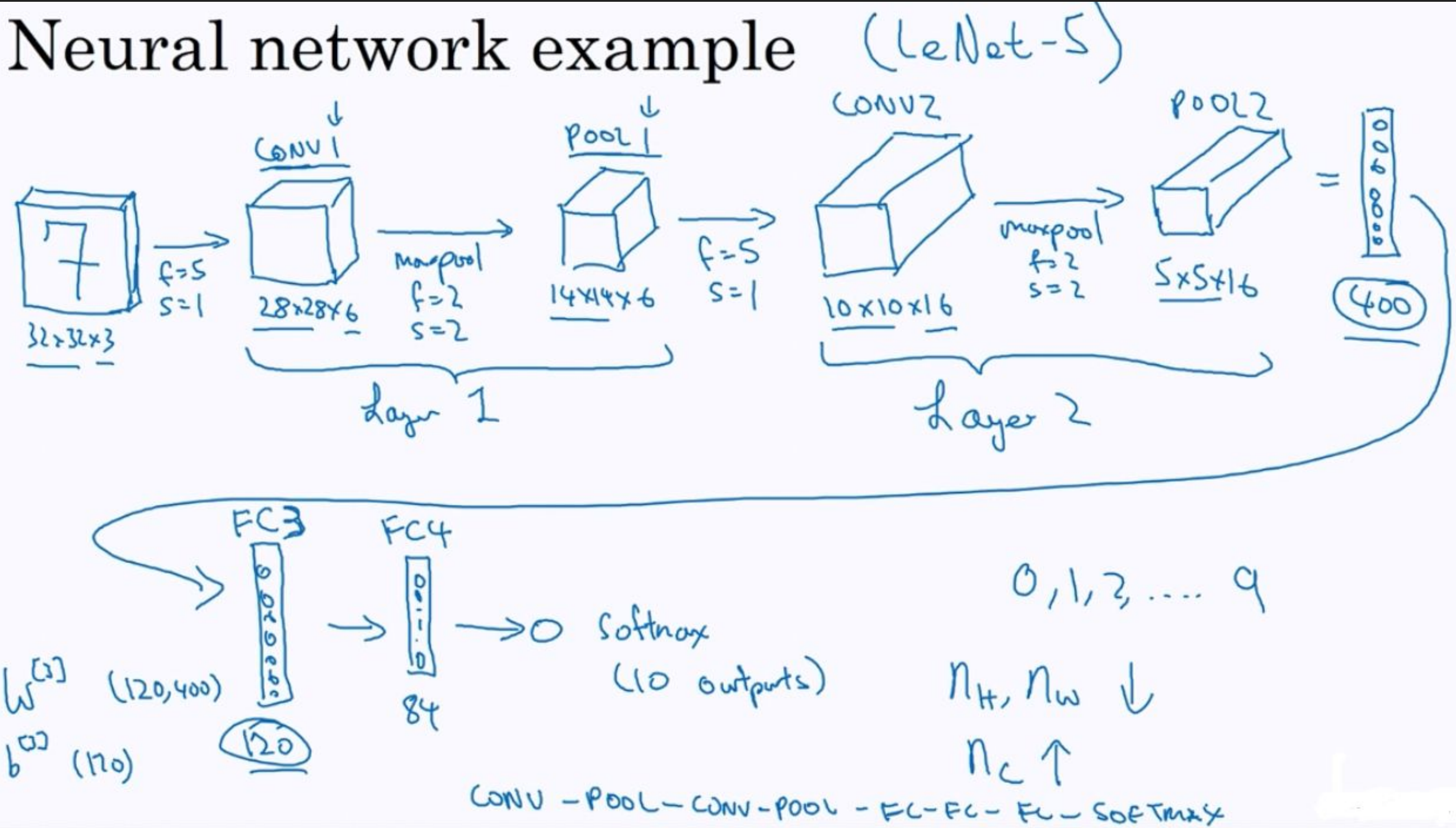
Max or average pooling

$f=2, s=2$

$f=3, s=2$

→ p: padding.

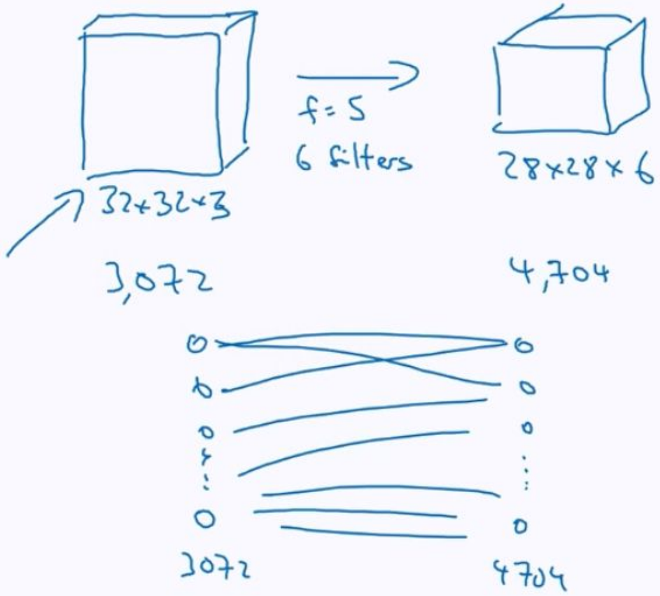
Neural network example



Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

Why convolutions



$5 \times 5 - 25$

26

$6 \times 26 = 156$ parameters

$3{,}072 \times 4{,}704 \approx \underline{14M}$

Why convolutions

translation invariance

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{3 \times 3} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

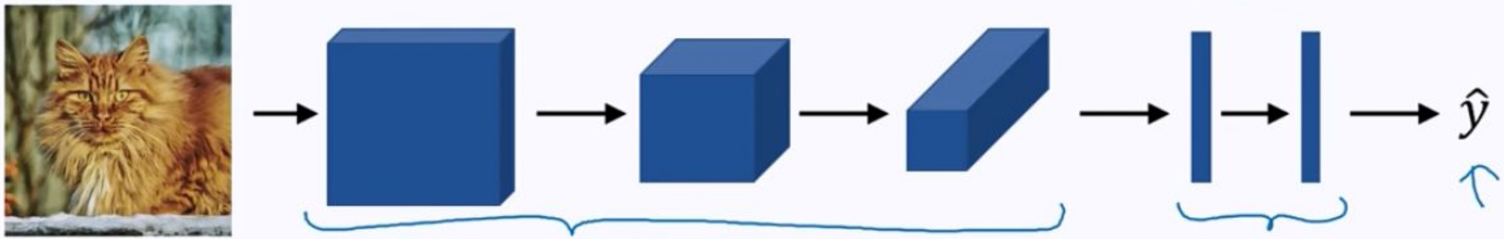
Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Putting it together

Training set $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$.

$\boxed{w, b}$



Cost $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Use gradient descent to optimize parameters to reduce J