












Deep Learning Study

2020/09/08

Byungchan Lee

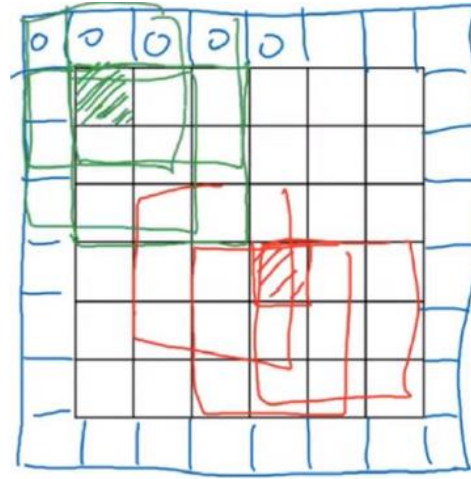


딥러닝 3단계 : 머신러닝 프로젝트 구조화하기

합성곱 신경망 (Convolutional Neural Networks)	
	컴퓨터비전 업데이트 : 2020.07.11 ♥ 9
	모서리 감지 예시 업데이트 : 2020.07.11 ♥ 11
	더 많은 모서리 감지 예시 업데이트 : 2020.07.12 ♥ 8
	패딩 (Padding) 업데이트 : 2020.07.12 ♥ 10
	스트라이드 (Stride) 업데이트 : 2020.07.12 ♥ 8
	입체형 이미지에서의 합성곱 업데이트 : 2020.07.12 ♥ 9
	합성곱 네트워크의 한 계층 구성하기 업데이트 : 2020.07.12 ♥ 9
	간단한 합성곱 네트워크 예시 업데이트 : 2020.08.15 ♥ 8
	풀링(Pooling)층 업데이트 : 2020.07.12 ♥ 6
	CNN 예시 업데이트 : 2020.09.07 ♥ 8
	왜 합성곱을 사용할까요? 업데이트 : 2020.09.07 ♥ 9

CNN

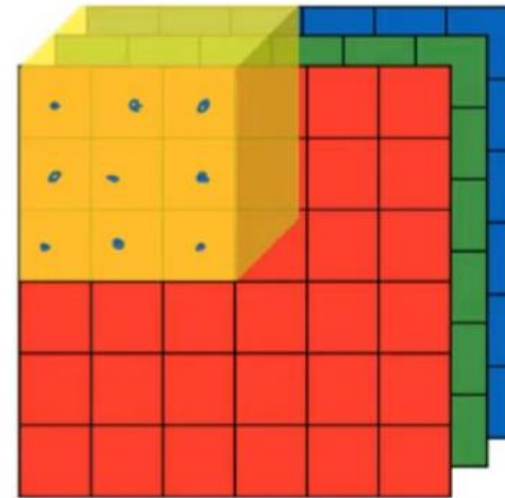
- Filter
- Padding
- Stride
- Channel



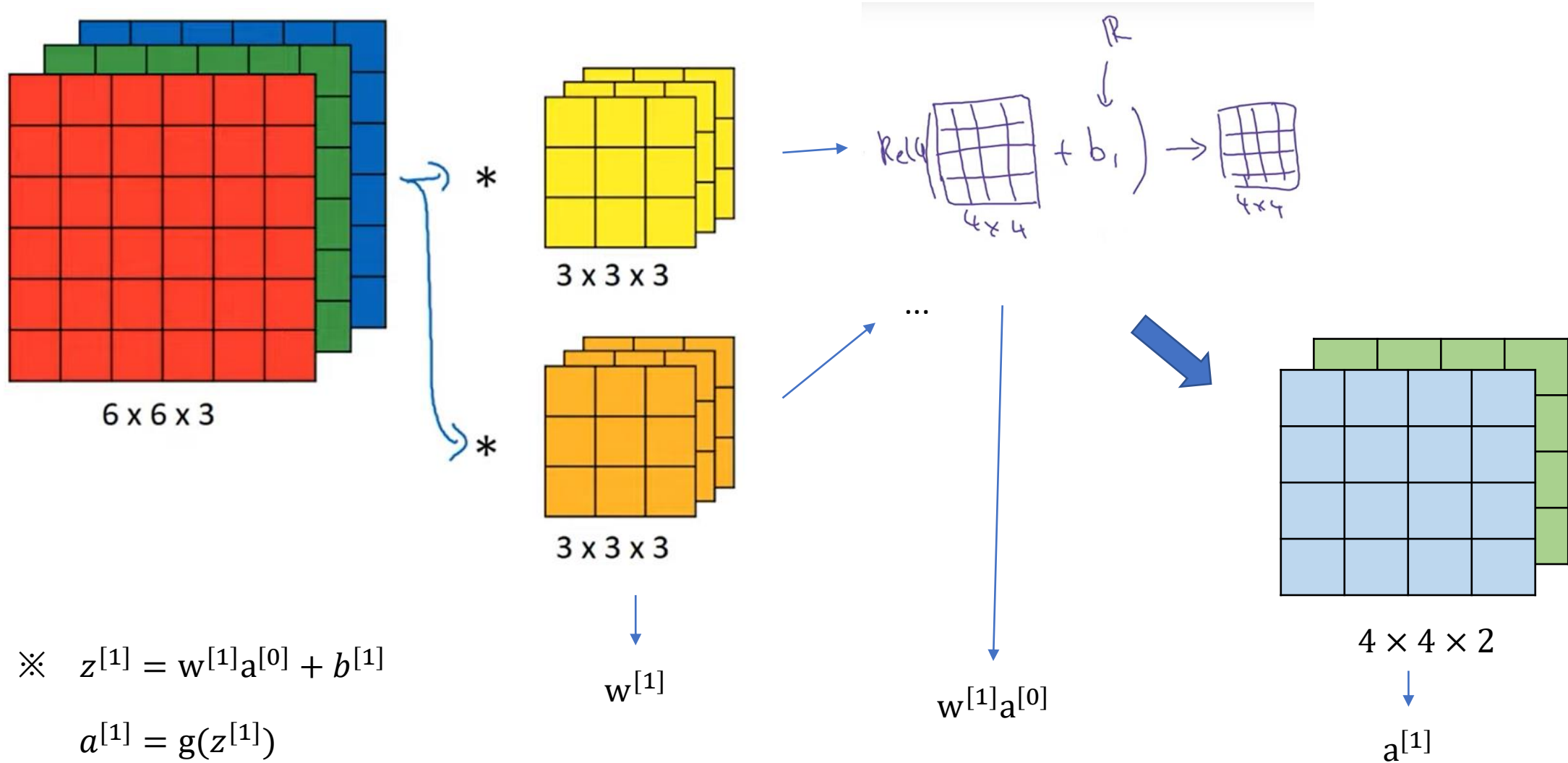
2	3	7	4	6	2	9
6	6	9	8	7	4	3
3 ³	4 ⁴	8 ⁴	3	8	9	7
7 ¹	8 ⁰	3 ²	6	6	3	4
4 ⁻¹	2 ⁰	1 ³	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

7x7

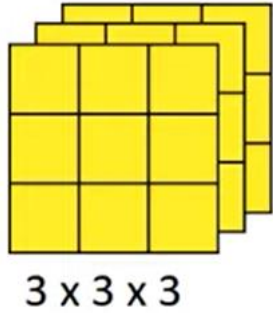
- 입력 $n \times n \times n_c$
- 필터 $f \times f \times n_c \times n_c'$
- 출력 $\left(\frac{n+2p-f}{s} + 1\right) \times \left(\frac{n+2p-f}{s} + 1\right) \times n_c'$



One Layer of a Convolutional Network

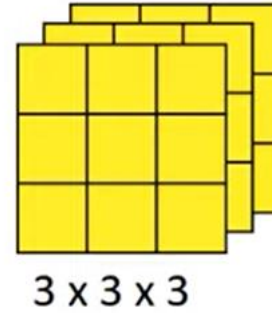


Number of Parameters in one Layer



1

...



10

- 10 filter, $3 \times 3 \times 3$
- How many parameters does that layer have?

Summary of notation

If layer l is a convolution layer:

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $A^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$ ← #filters in layer l.

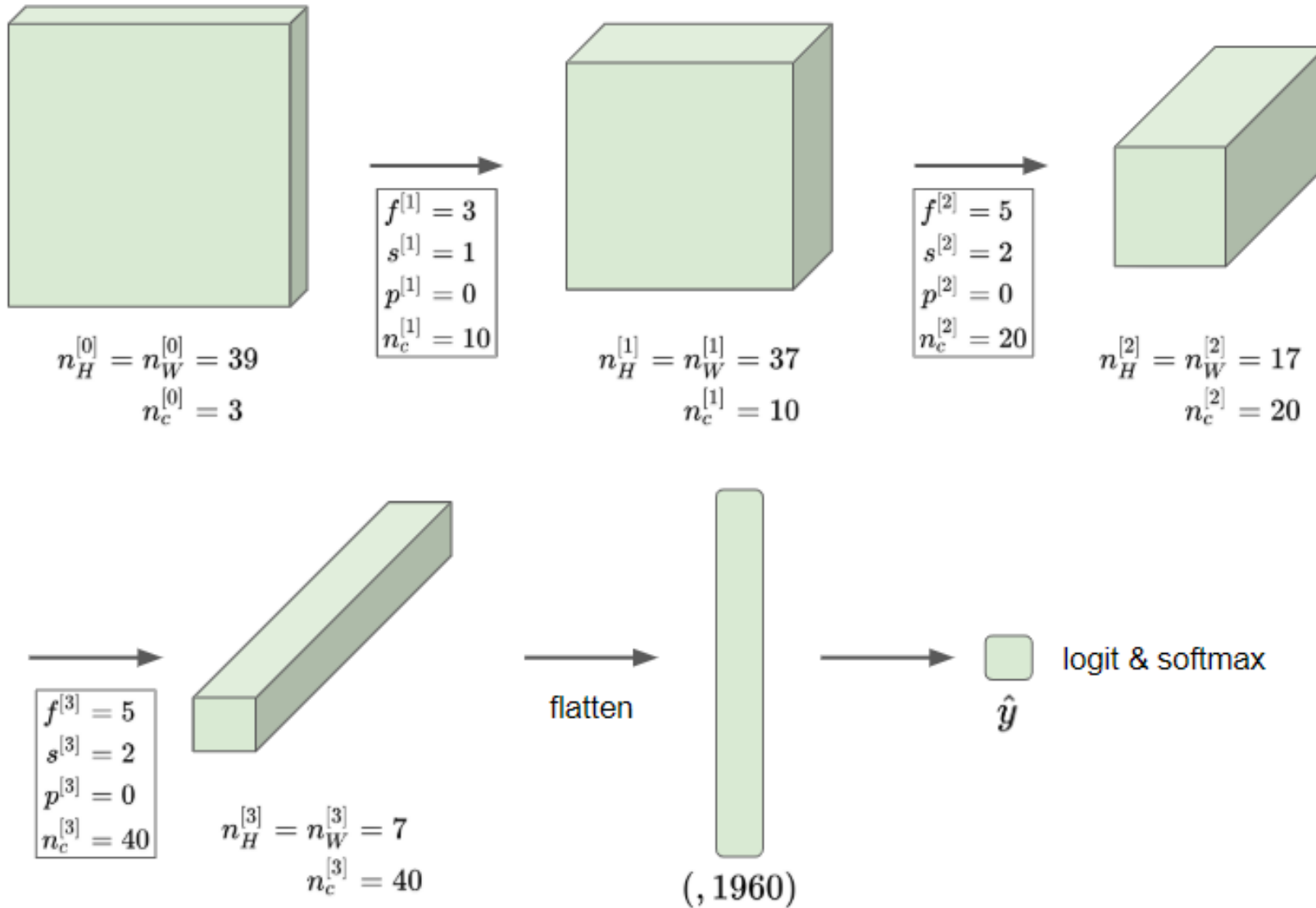
Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

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

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

$$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$$

Simple CNN Example



Pooling Layers

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

→
Max Pooling

9	2
6	3

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

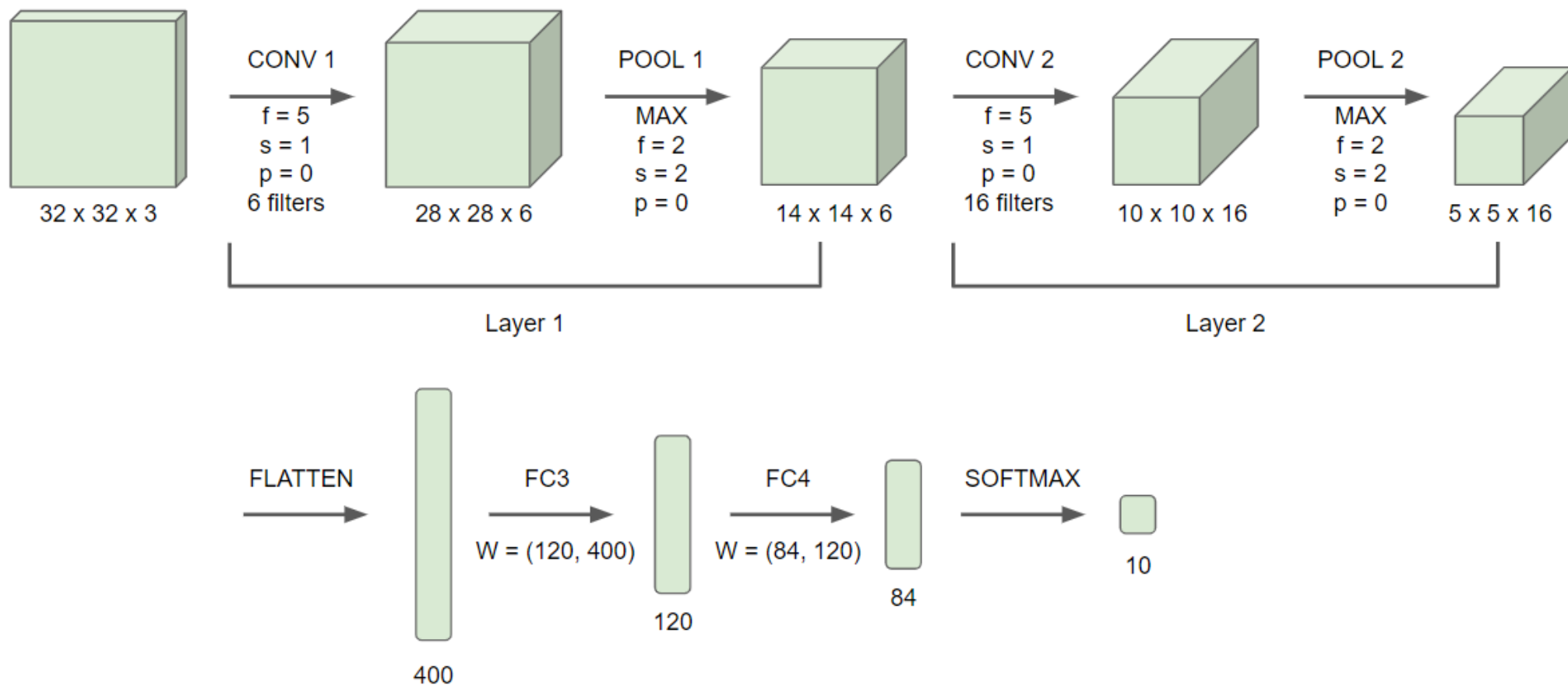
→
Average Pooling

3.75	1.25
3.75	2.0

- $f = 2, s = 2, p = 0$
- No parameters to learn!

CNN Example

LeNet - 5



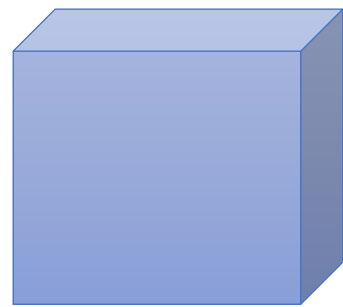
<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

CNN Example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	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

Andrew Ng

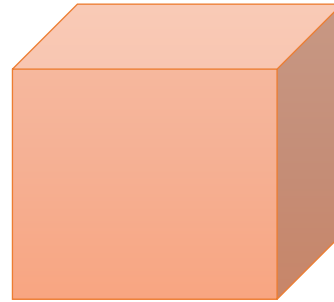
Why Convolutions?



$32 \times 32 \times 3$



$f = 5$
6 filters



$28 \times 28 \times 6$

of parameters

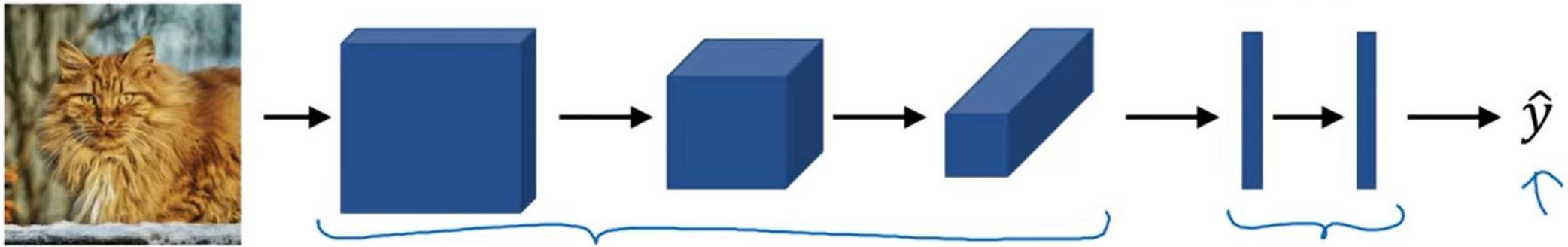
- Fully Connected : $3072 * 4704 \sim 14\text{M}$
- Convolution : $(5*5 + 1) * 6 = 156$

- 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
→ Translation Invariance

Why Convolutions?

Putting it together

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



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

Use gradient descent to optimize parameters to reduce J