Foundations of Convolutional Neural Networks

Edge detection

image * filter = another image

filter = kernel

1	0	−1
1	0	−1
1	0	−1

Vertical edge detector

1	1	1
0	0	0
−1	−1	−1

Horizontal edge detector

Vertical edge detection

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

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

*

more examples

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

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

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

*

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

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Horizontal edge detection

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

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Learning to detect edges

1	0	−1
2	0	−2
1	0	−1

Sobel filter

3	0	−3
10	0	−10
3	0	−3

Scharr filter

핸드 코딩된 필터 말고 파라미터를 학습시키면? 데이터로부터 자동으로 학습하면서 신경망이 low-level features를 배우고 edge와 같은 특징을 배우게 할 수 있을 것이다.

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

$*$

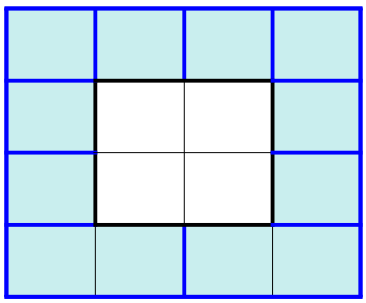
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$=$

Padding

1. Shrinking output
   픽셀의 수가 적어지므로 conv 연산은 얼마 못 할 것이다.
2. throwing away into from edge
   conv 연산 특성상 중앙에 있는 픽셀은 많이 사용하지만 코너나 모서리에 있는 픽셀들은 훨씬 적게 사용힌다. 가장자리 부근 많은 정보를 버리는 것과 같다.

해결 방법: 입력 이미지 가장자리에 패드를 덧붙여 이미지를 크게 만든다.



p = padding = 1,

padding 안의 값이 0 이면 Zero-Padding

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

Valid Convolution: no padding

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

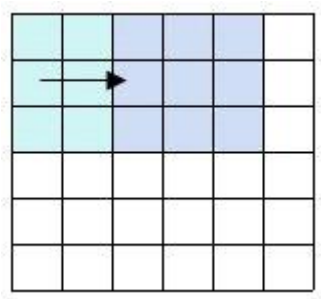
$$n + 2p - f + 1 = n \Rightarrow p = \frac{f-1}{2}$$

f is usually odd. 대부분 3x3, 5x5, 7x7 필터

이유:

1. f가 짝수면 몇 가지 비대칭적인 패딩을 해야 한다.
   그 중에 하나가 same convolution(input size = ouput size)일 때다
2. f가 홀수면 중앙점을 가지고 있어 CV에 있어서 특징점을 가지고 있다는 장점이 있다.
   중앙 픽셀을 정할 수 있고 필터의 포지션에 대해 이야기할 수 있다.

Strided convolutions



만약 필터가 이미지를 넘어가면 계산하지 마라

s = stride = 2, $\lfloor z \rfloor = floor(z)$: 정수 아니면 내림

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

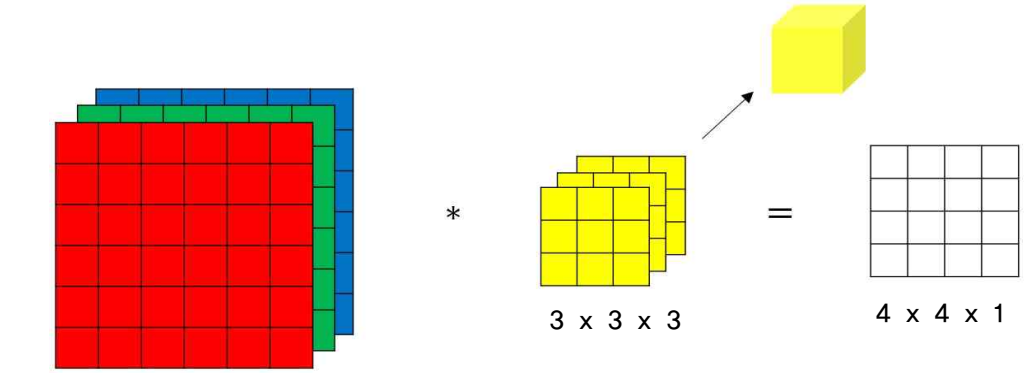
Summary

n x n image	f x f filter
padding p	stride s
Output size: $\lfloor \frac{n+2p-f}{s}+1 \rfloor \times \lfloor \frac{n+2p-f}{s}+1 \rfloor$	

Technical note on cross-correlation vs. convolution

수학책에서 나오는 convolution은 filter를 x축, y축 대칭(flip)하고 연산을 한다. 그래서 딥러닝에서 말하는 convolution은 실제 수학에서는 cross-correlation이라고 한다. 기계학습 관례상 flipping operation을 무시하고 convolution이라 부를 뿐이다.

Convolutions over volumes



height x width x # Channels

6 x 6 x 3

※ output의 channel이 더 이상 3개가 아니다.(3D → 2D)

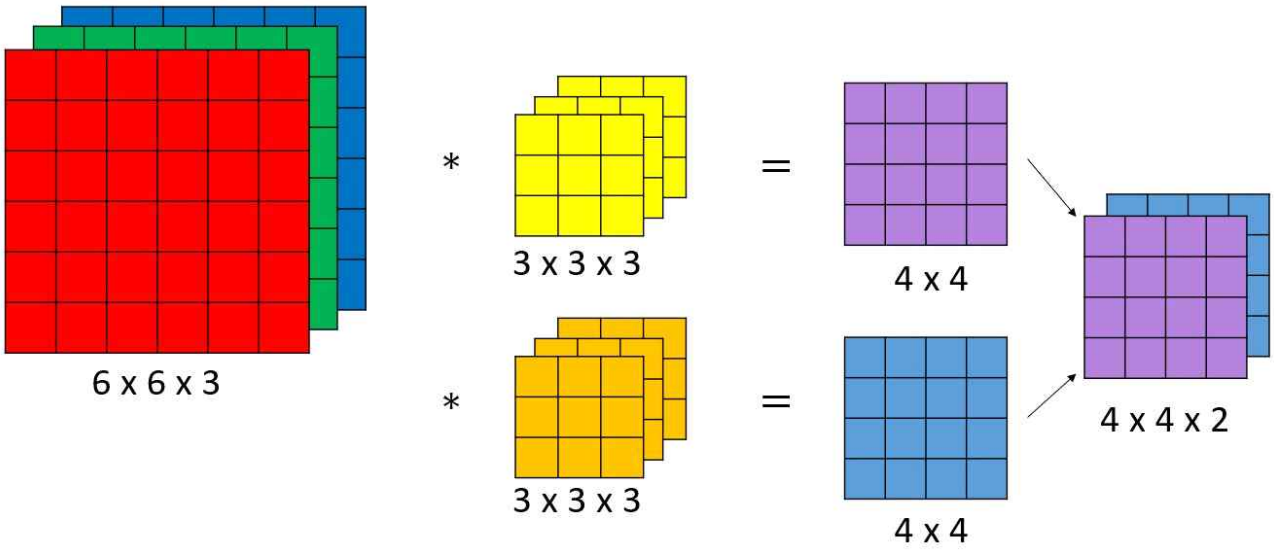
ex) R 채널의 vertical edge detector

: 필터 1만 vertical edge detector로 하고 나머지 두 필터는 0

ex) 채널 상관 없이 vertical edge detector

: 모든 세 필터가 vertical edge detector

Multiple filters



차원을 나타내기 위해 'number of channels' 라고 하지만 종종 'depth of 3D volume' 이라고 하기도 한다. 하지만 depth라고 하면 신경망의 depth와 혼동을 줘서 주의해야 한다.

Summary: if no pad and stride = 1,

$$(n_W \times n_H \times n_C) * (f \times f \times n_C) \rightarrow (n_H - f + 1) \times (n_W - f + 1) \times n_C'$$

One layer of a Convolution network

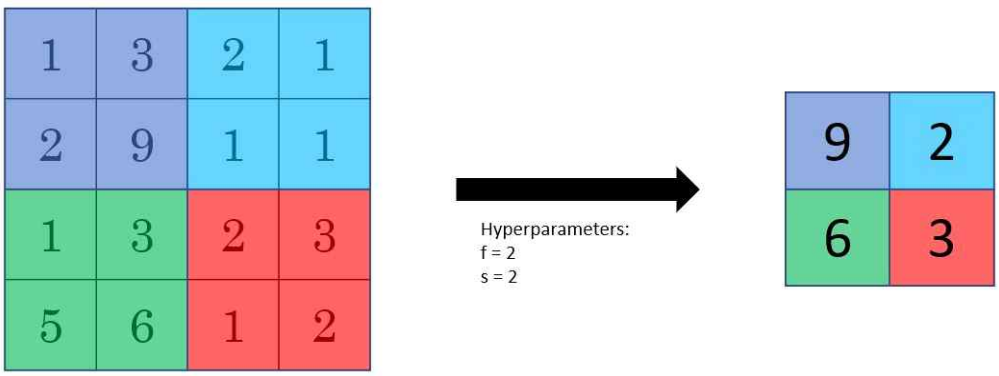
If layer l is a convolution layer:	
$f^{[l]}$ = filter size	Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$
$p^{[l]}$ = padding	Output: $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$
$s^{[l]}$ = stride	$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$
$n_C^{[l]}$ = number of filters	$n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$
Each filter is: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$	
Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$	$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$
Weights: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$	
bias: $n_C^{[l]}$ or $(1,1,1,n_C^{[l]})$	
일반적으로 layer를 지날수록 이미지의 크기는 줄어들고 채널의 숫자는 커진다.	

Types of layer in a convolution network:

1. Convolution(CONV)
2. Pooling(POOL)
3. Fully connected(FC)

Pooling layers

Max Pooling

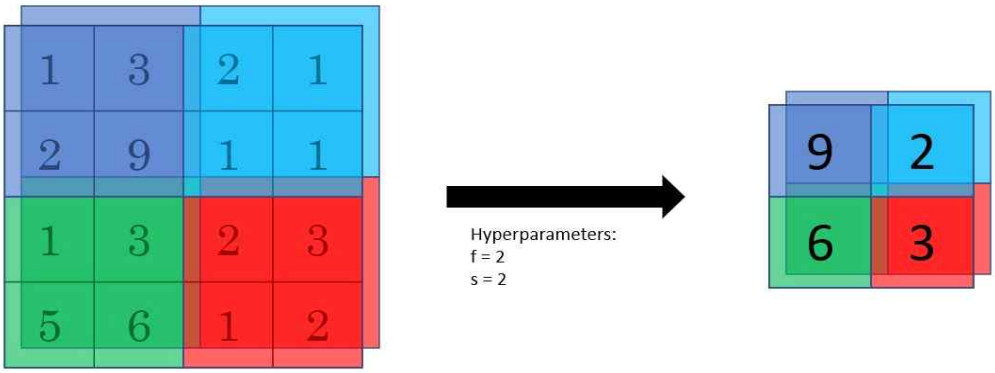


어디에서나 feature가 많이 감지되도록 하여 필터 안에서 feature가 감지되면 최대값을 잡아둔다. feature가 감지되지 않으면 오른쪽 사분면에는 존재하지 않는다.

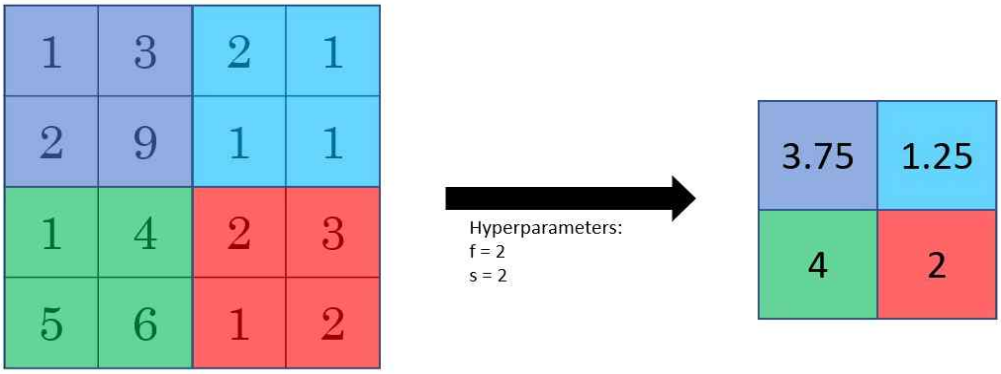
사람들이 많이 사용하는 이유는 많은 실험에서 잘 작동하기 때문이지만 확실한 이유를 아는 사람은 없다.

max pooling의 흥미로운 속성 중 하나는 hyperparameter는 있지만 parameter는 없다는 것이다.

Multiple Max Pooling



Average Pooling



Summary of Pooling

Hyperparameters:	Input: $n_H \times n_W \times n_C$
f = filter size	Output: (if no padding) $\left\lfloor \frac{n_H - f + 1}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f + 1}{s} + 1 \right\rfloor \times n_C$
s = stride	
Max or Average Pooling	
예외가 있기는 하지만 pooling 할 때 padding은 대부분 사용하지 않는다.	
No parameters to learn	

Although pooling layers do not have parameters, they affect the backpropagation (derivatives) calculation, because pooling layers modify the input.

hyperparameter가 많다. hyperparameter를 고르는 몇 가지 구체적인 방법이 있다.
일반적인 가이드라인은 자기만의 hyperparameter를 설계하려 하지 말고 다른 사람들은 어떻게 적용했는지 찾아보는 것이다.

보통 파리미터가 있는 층을 레이어라고 부른다. Max Pooling 같은 경우에는 하이퍼파라미터만 있고 파라미터는 없기 때문에 Max Pooling layer라고 부르긴 하지만 CONV와 함께, 즉 CONV + POOL 을 합쳐서 layer 라고 한다. ex) conv1 + pool1 = layer 1

layer를 지날수록 높이(n_H)와 너비(n_W)가 줄어들고 채널의 개수(n_C)는 늘어난다.
너무 빨리 줄어들면 수행 능력이 좋지 않다.

Neural network example

	Activation shape	Activation size	# parameters
Input:	(32, 32, 3)	3,072	0
CONV1 (f=5, s=1)	(28, 28, 8)	6,272	608
POOL1	(14, 14, 8)	1,568	0
CONV2 (f=5, s=1)	(10, 10, 16)	1,600	3,216
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48,120
FC4	(84, 1)	84	10,164
Softmax	(10, 1)	10	850

Why Convolution?

장점

1. 파라미터 공유(Parameter sharing)
   : A feature detector (such as a vertical edge detector) that is useful in one part of the image is probably useful in another part of the image.
   – It reduces the total number of parameters, thus reducing overfitting.
   – It allows a feature detector to be used in multiple locations throughout the whole input image/input volume.
2. 연결의 희소성(Sparsity of connections)
   : In each layer, each ouput value depends only on a small number of inputs.
   – Each activation in the next layer depends on only a small number of activations from the previous layer.