15주차

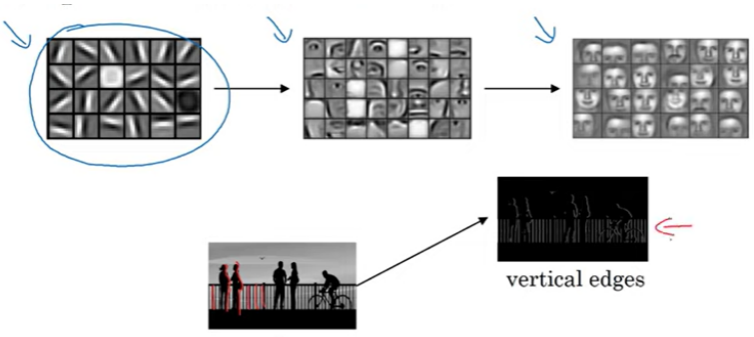
태그	완료
텍스트	`Kaggle 필사`

컴퓨터 비전

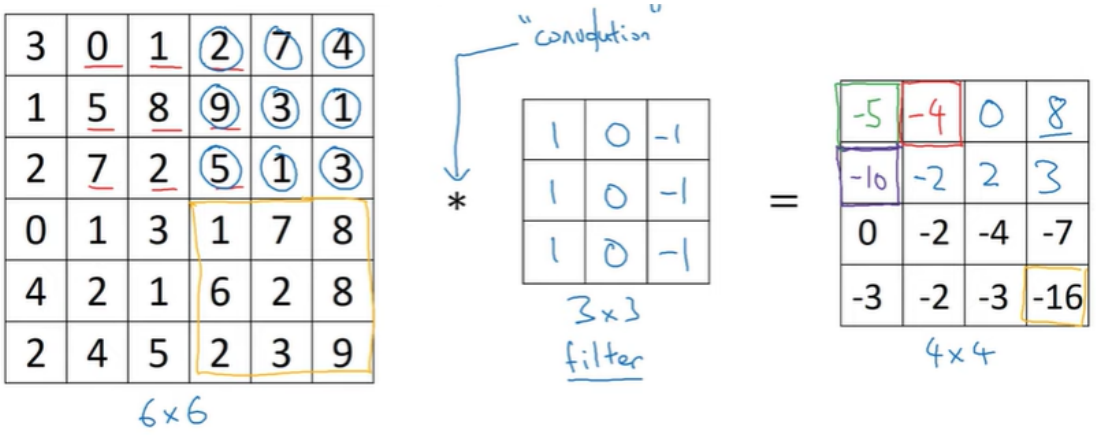
- 컴퓨터 비전과 딥러닝
  - CV의 발전 → 많은 새로운 application이 만들어질 수 있음
- CV Problems
  - Image classification: 이미지 분류
  - Object detection: 단순히 분류하는 것이 아닌 물체의 위치나 종류를 탐지
  - Neural Style Transfer: 신경망을 이용해 이미지의 스타일을 재구성
- Challenges of CV
  - 입력데이터(이미지)가 아주 크다는 것(과적합 + 높은 계산 비용) ⇒ 합성 신경망을 이용

모서리 감지 예시

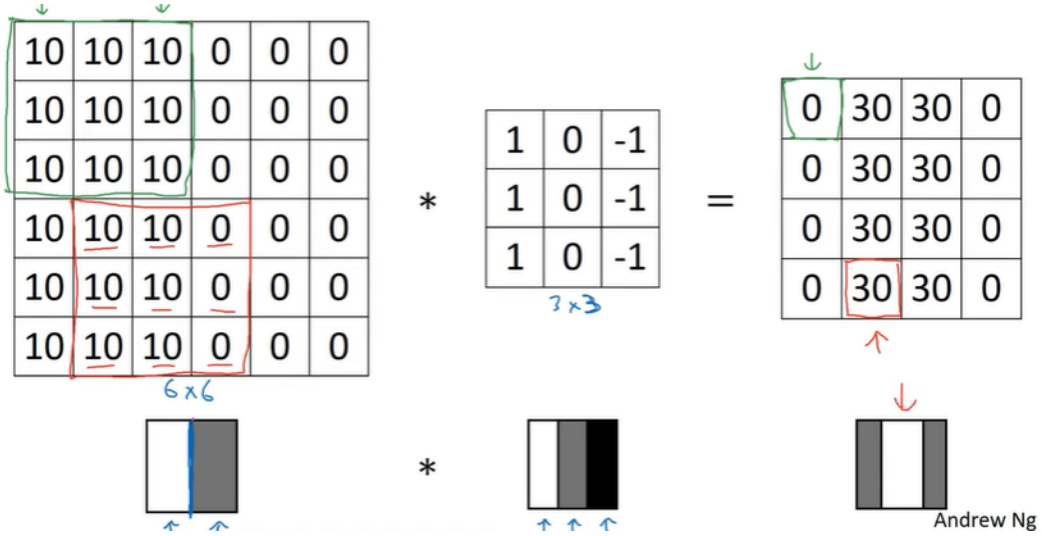
- 첫 번째로 하는 일은, 이미지에서 세로 선(vertical edges)들을 찾는다.



⇒ how?



원본 이미지(왼쪽)와 3x3 filter을 합성곱을 해 새로운 이미지(우측)을 만듦.
ex) 마지막 노란 박스: 1x1 + 7×0 +8x(-1) + 6×1 + 2×0 + 8x(-1) + 2×1 + 3×0 +9x(-1) = -16

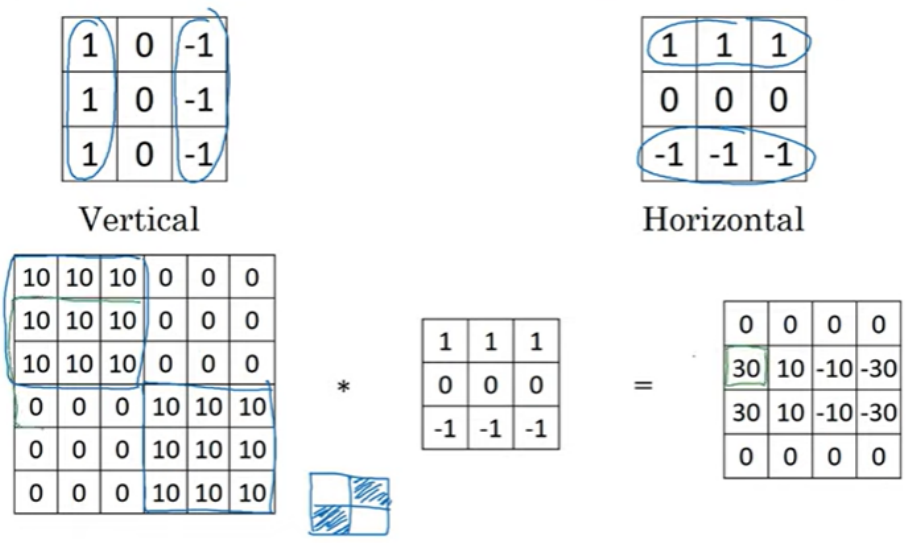


원본 이미지(왼쪽)에 흰색(10)과 회색(0) 사이에 세로 경계선이 있음. 세로 경계선 검출 필터를 사용해 합성곱 신경망을 거친 이미지(오른쪽)에 그 경계선이 흰색(30)으로 표현됨.

⇒ 비록 크기가 안 맞지만 이는 굉장히 작은 이미지를 예시로 들었기 때문임. 몇 천 x 몇 천 이미지를 사용하면 정교하게 수직 경계선을 찾아낼 수 있음.

더 많은 모서리 감지 예시

- Vertical and Horizontal Edge Detection



Vertical filter의 왼쪽은 밝고 오른쪽은 어둡다. Horizontal filter의 위쪽은 밝고 아래쪽은 어둡다.

    - 10과 같은 어중간한 숫자가 나온 이유는, 원본 이미지의 왼쪽은 양의 윤곽선이고 오른쪽은 음의 윤곽선이기 때문이다. → 마찬가지로 이미지의 크기가 커지면 무시된다.
- 역사적으로 CV 분야에서 어떤 조합의 숫자 필터를 사용하느냐에 관한 논쟁이 있었다.
    - sobel: 장점은 (가로로/세로로)중간 부분의 픽셀에 더 중점을 둬 더 선명해 보인다는 것이다.

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

vertical

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

sobel

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

scharr

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

horizontal

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

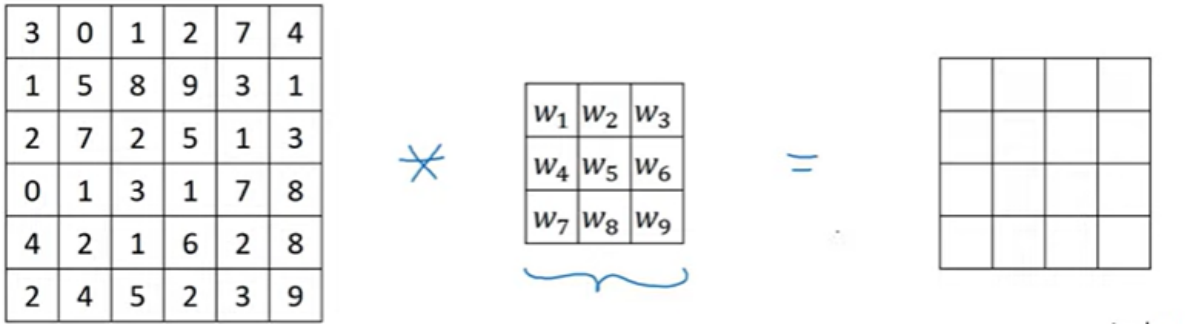
sobel

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

scharr

⇒ 딥러닝의 발전으로 필터의 숫자를 수동으로 고를 필요가 없어졌다. 9개의 숫자를 변수로 두고 backpropagation을 통해 스스로 필터의 숫자를 학습해 문제에 적합한 필터를 만들도

록 한다. 이는 세로, 가로의 경계선뿐 아니라 기울어진 경계선을 학습할 수도 있다.



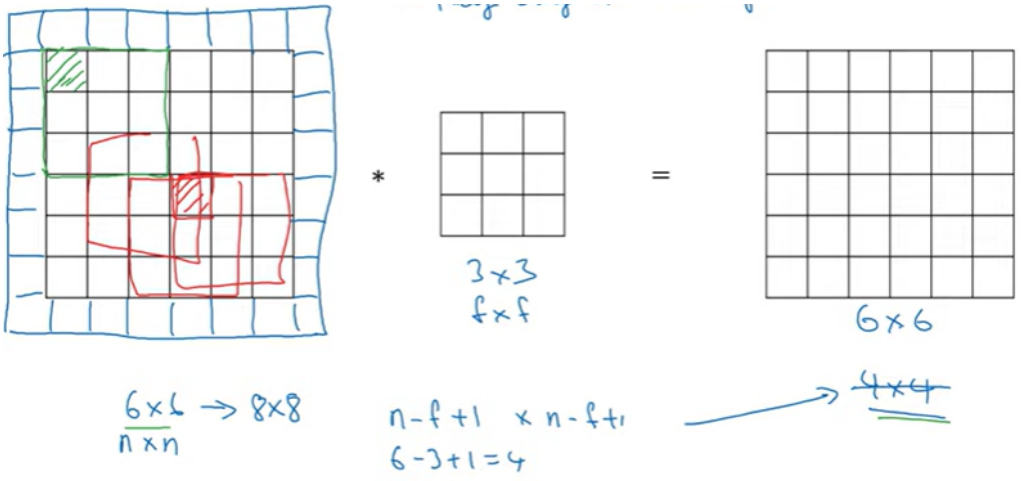
Padding

- 그냥 합성곱만 실행할 때의 단점
  1. 계속 합성곱을 실행하면, 이미지도 계속 축소된다. → 수백 개의 층을 거친 후에는 매우 작은 크기의 이미지만 남게 된다.
  2. 이미지의 가장자리 픽셀은 단 한 번만 사용된다. → 정보가 날아간다.

💡 **Solution:** 이미지의 가장자리에 픽셀을 덧대준다. (padding)

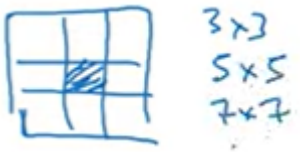
- p = 1
- 보통 숫자 0으로 패딩한다.
- 합성곱 후에도 이미지의 크기가 6x6으로 유지된다.
- 최종 이미지 크기 (n: 이미지 크기, p: 패딩 크기, f: 필터 크기)

$$(n+2p-f+1) * (n+2p-f+1)$$



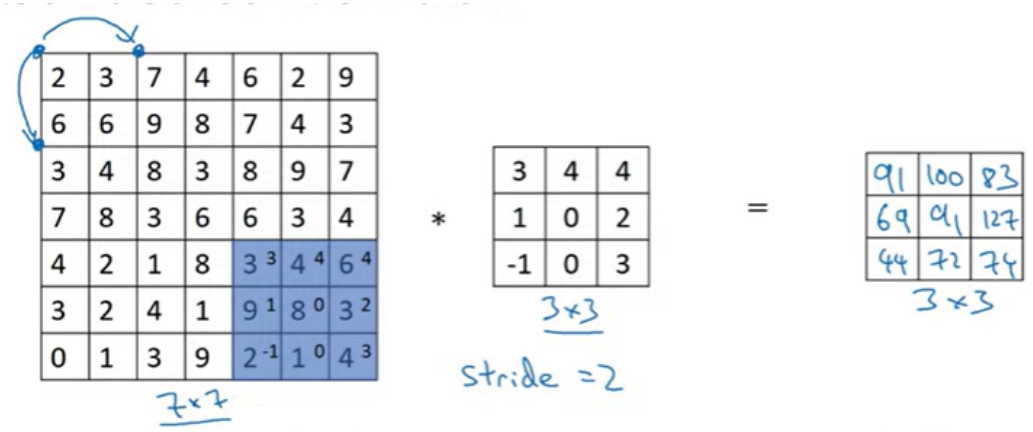
⇒ 패딩을 사용하면 이미지의 크기를 유지할 수 있고, 가장자리의 정보를 덜 쓰는 일을 막을 수 있다.

- Valid and Same convolutions
  - valid(유 합성곱): 패딩 없는 합성곱 → (n-f+1)*(n-f+1)
  - same(동일 합성곱): 패딩을 적용해 입출력의 크기가 똑같은 합성곱 → (n+2p-f+1)*(n+2p-f+1)
    - 이때, n+2p-f+1 = n이므로 **p = (f-1)/2**이다.
    - 필터의 크기가 3이면 패딩의 크기는 1, 필터의 크기가 5면 패딩의 크기는 2가 된다.
    - 일반적으로 CV에서 필터의 크기 f는 홀수다.
      1. 필터의 크기가 짝수라면 패딩이 비대칭이 된다. f가 홀수일 때만 왼쪽과 오른쪽을 동일한 크기로 패딩을 더해줄 수 있다.
      2. 홀수 크기의 필터에서는 필터에 중심이 존재한다.



Stride

- stride: 필터의 이동 보폭



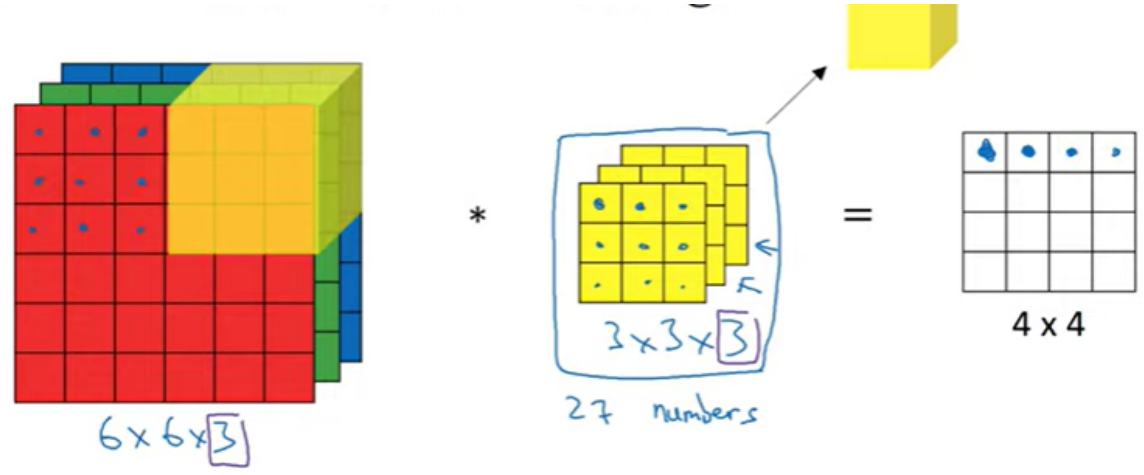
- 최종 이미지 크기: 소수점이라면 내림을 한다. → 일반적으로 필터의 크기에 패딩과 스트라이드의 크기를 조절해 값이 정수로 나오도록 한다.

$$(\frac{n+2p-f}{s}+1)*(\frac{n+2p-f}{s}+1)$$

- 신호처리에서의 교차상관과 합성곱
  - 일반적으로 수학에서 정의하는 합성곱은, 합성곱을 하기 전에 필터를 가로축과 세로축으로 뒤집는 연산(미러링 과정)을 해야 한다.
  - 지금까지 배운 합성곱은 사실은 교차상관이지만 딥러닝에서는 관습적으로 합성곱이라고 한다.
  - 딥러닝에서는 필터를 뒤집는 연산을 생략한다. 이 연산은 신호처리에서는 유용하지만 딥러닝에서는 아무런 영향이 없기 때문에 생략한다.

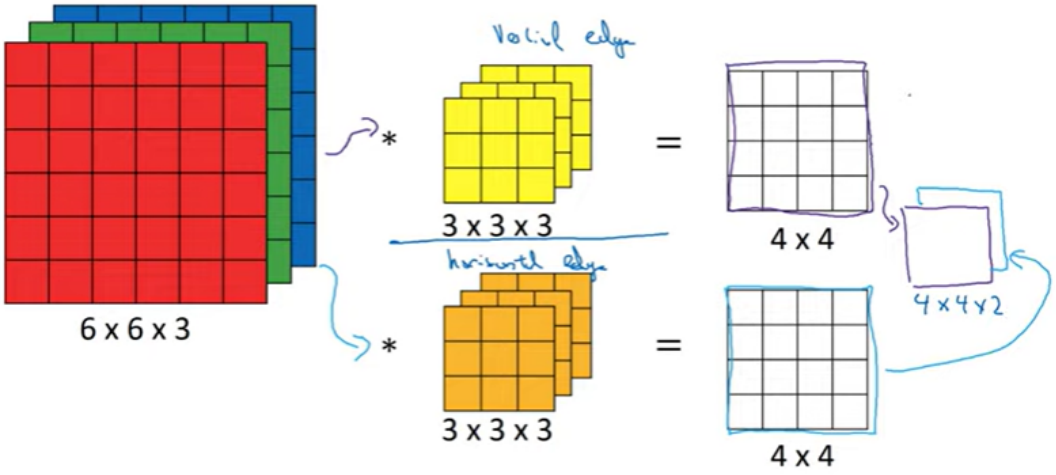
입체형 이미지에서의 합성곱

- 이미지에 색상(RGB)가 들어가게 되면 입체형으로 변하게 되며, 차원이 하나 증가한다. → height x width x channels
- 이에 따라 필터도 입체형으로 변하며 차원이 하나 증가한다. 이미지의 채널과 필터의 채널 수는 같아야 한다.
- 그러나 결과는 하나의 이미지로, 2차원이다.



각 채널별로 합을 구하고, 그 합을 합쳐서 결과 이미지의 한 칸의 값을 구함.

- Multiple Filters



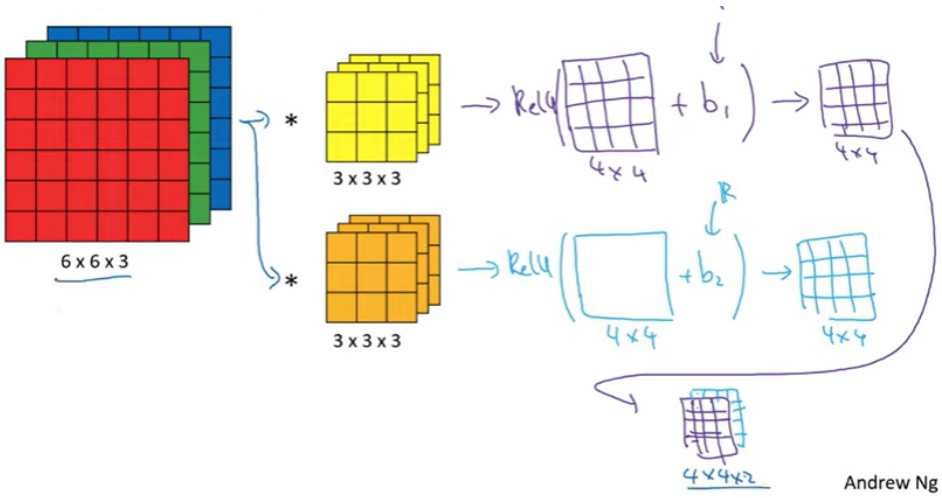
패딩=0 & 스트라이드=1이라고 가정했을 때, 최종 이미지의 크기는 다음과 같다.

$$(n * n * n_c) * (f * f * n_c) = (n - f + 1) * (n - f + 1) * n_c'$$

이때 n_c는 채널의 수, n_c'는 사용한 필터의 개수이다.

합성곱 네트워크의 한 계층 구성하기

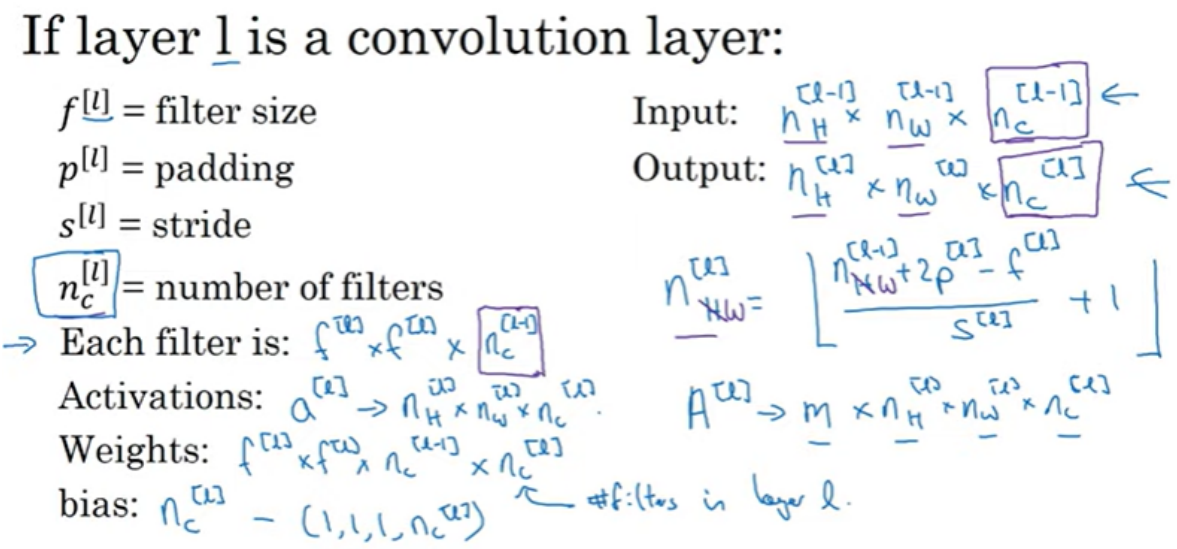
- 각 필터를 사용해 나온 결과에 실수의 편향을 더해준 뒤 ReLU로 비선형성을 적용해준다. ⇒ 결과를 쌓아주면 합성곱 신경망의 한 층이 된다.



입력 이미지 = a^[0], 필터들 = w[1], 1차 결과 이미지 + b = z^[1], ReLU(z^[1]) = a^[1]

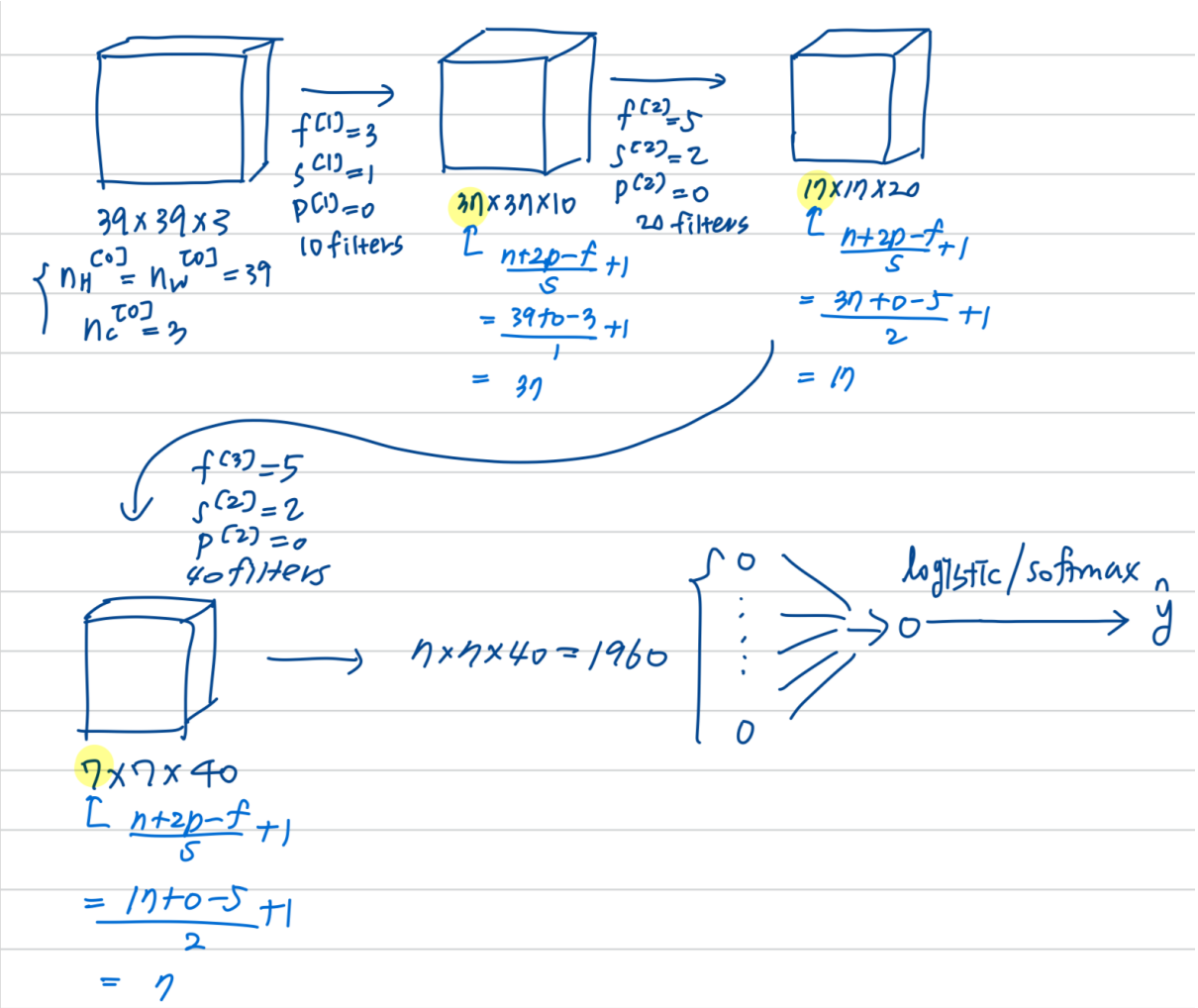
한 계층의 합성곱 신경망 생성

- 문제: 만약 하나의 층에 3x3×3 짜리 필터 10개가 있다면 몇 개의 파라미터가 있는가?
  → (27 + 1(bias)) x 10 = 280개
- Summary of notation
  - 단순 신경망을 사용하는 것보다 합성곱 신경망을 사용하면 필요한 파라미터의 수를 줄일 수 있다.



$n_c^{[l-1]}$: l-1층에서의 채널 수, $n_c^{[l]}$: l층에서의 채널 수(l-1층에서의 사용된 서로 다른 필터 수)

간단한 합성곱 네트워크 예시



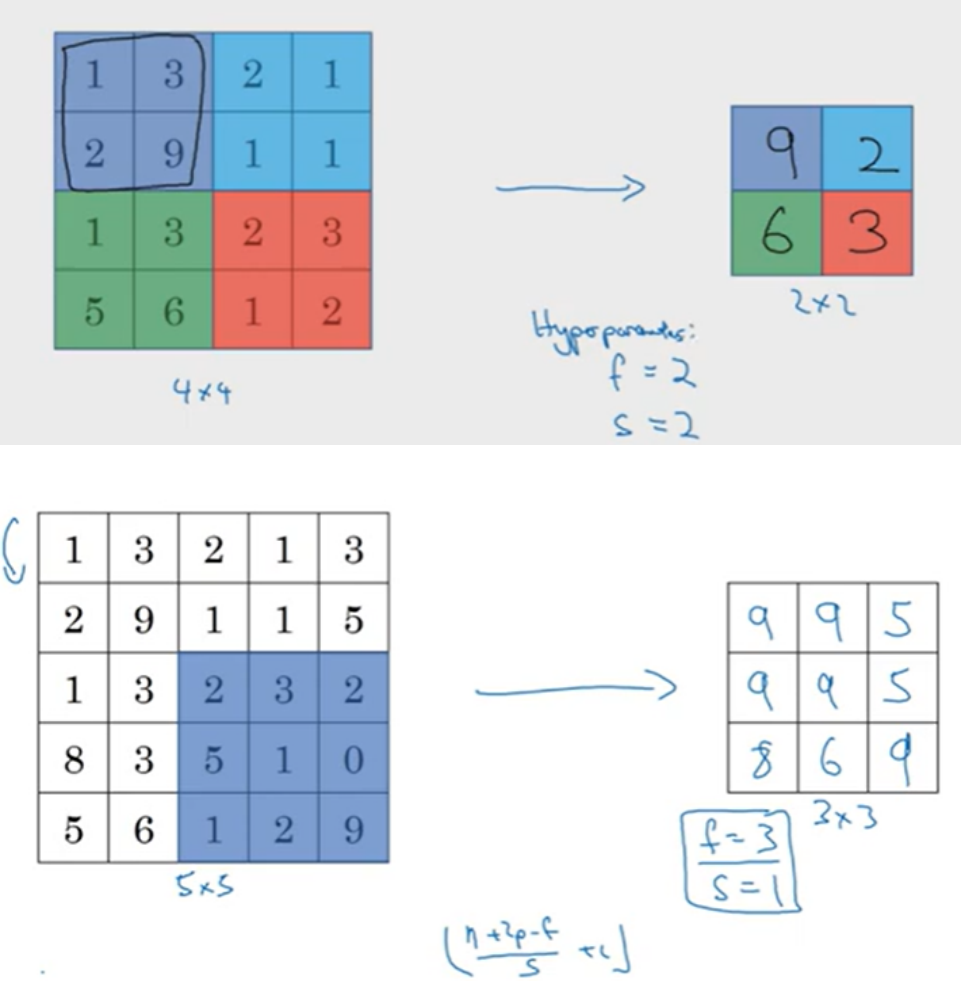
이때 f(필터 크기), s(스트라이드), p(패딩 크기)는 모두 하이퍼파라미터이다.

- 이미지의 크기는 합성곱 신경망을 거침에 따라 줄어들지만 채널의 수는 계속 늘어난다.
- Types of layer in a convolutional network
  - convolution
  - pooling
  - fully connected

풀링(Pooling)층

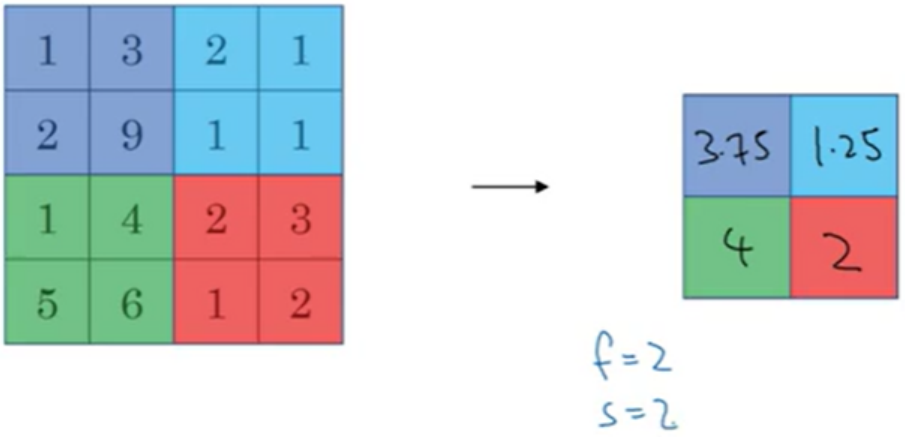
- 풀링 층을 사용해 표현 크기를 줄임으로써 계산 속도를 줄이고 특징을 더 잘 추출할 수 있다.
- Max Pooling
  - 일반적으로 잘 작동하고 가장 많이 사용된다.

- 필터 크기 = 2, 스트라이드 = 2, 패딩 크기 = 0이고 출력 계산 과정에서 (n+2p-f/2)+1 공식을 사용할 수 있다.
- 학습하는 파라미터가 없다.



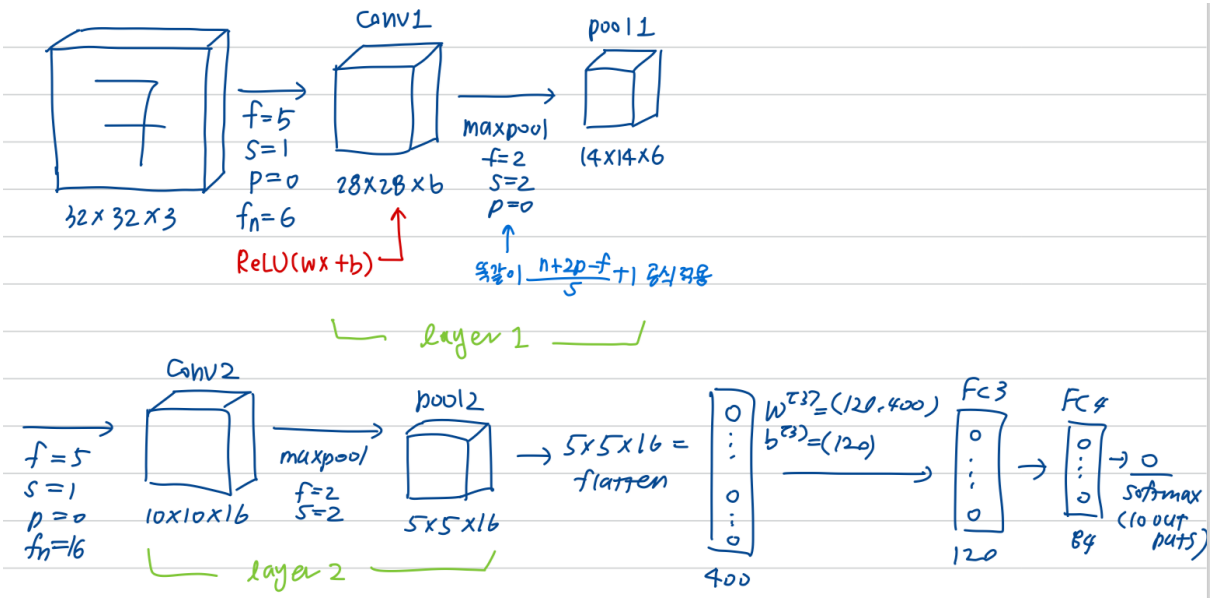
채널의 수가 늘어나도 각 채널에 개별적으로 똑같이 적용하면 된다.

- Average Pooling



CNN(Convolutional Neural Network)

- Neural Network Example (LeNet-5)

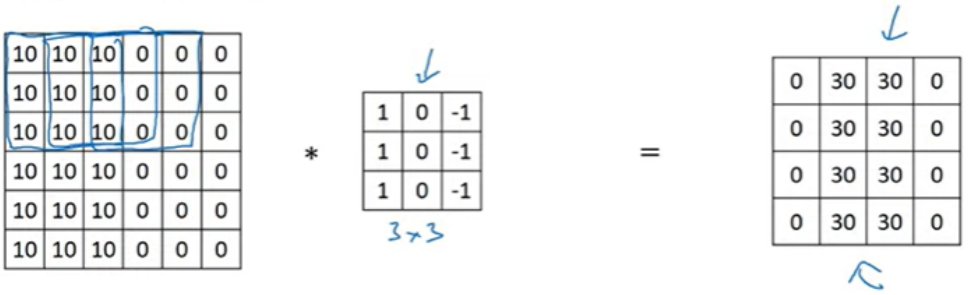


pooling layer에는 파라미터(변수)가 없다.

왜 합성곱을 사용할까요?

- 일반 신경망 대신 합성곱을 사용하는 것의 장점
  1. 파라미터(변수)의 개수를 확연히 줄일 수 있다.
     a. 예를 들어, 32x32×3 이미지를 5x5 필터 6개를 통해 28x28×6의 이미지로 합성곱 연산을 했을 경우, 필요한 변수의 개수는 5x5×3×6+6 = 456개이다. 그러나 만약 일반 신경망을 사용했다면 (32x32×3)x(28×28×6) = 약 14만 개의 변수가 필요할 것이다.

b. Parameter sharing(변수 공유): 이미지의 한 부분에 유용한 필터가 다른 부분에도 유용한 경우



9개의 변수를 이용해 16개의 값을 계산한다.

c. Sparsity of connection(희소 연결): 각 출력의 각 값은 몇 개의 입력 값에만 의존한다. 예를 들어 첫 번째 0은 36개의 입력 값 중 6개의 10과만 연결되어 있다. 다른 픽셀들은 0에 아무런 영향을 주지 않는다. 이는 과적합을 방지할 수 있다.

2. 이동 불변성을 포착한다.
    a. 이미지에 약간의 변형이 있어도 이를 포착할 수 있다.

- Putting it together
    - m: 훈련 세트 데이터 개수