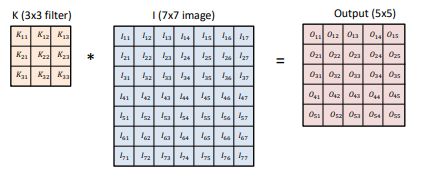
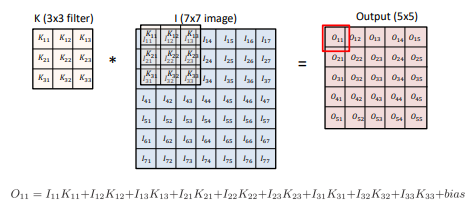


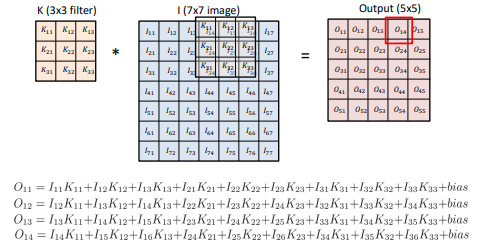
Convolution이라는 것은 Convolutional neural network에서 처음 나온 말은 아니다. Signal Processing에서 두 개의 함수가 있을 위에 나온 것처럼 두 개의 함수를 잘 섞어주는 어떤 방법으로 정의한다.



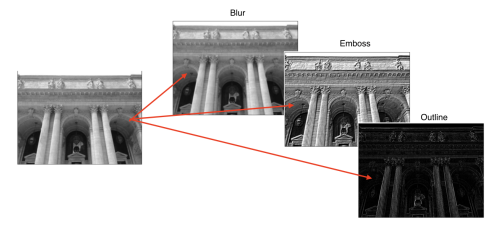
가장 기본적인 Convolution을 시도해보자.



하나의 output값은 convolution filter를 내가 적용하고자 하는 이미지에 찍는다고 보면 된다. 보통 '도장을 찍는다'는 표현을 많이 사용한다. 위는 3\*3 convolution filter를 7\*7 image의 적절한 위치에 찍게 되면 해당 위치의 convolution filter의 값과 image의 pixel값을 곱해서 모두 더한다.

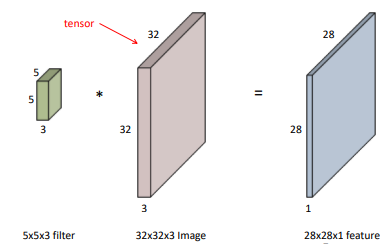


위 과정을 계속 반복하면 된다.

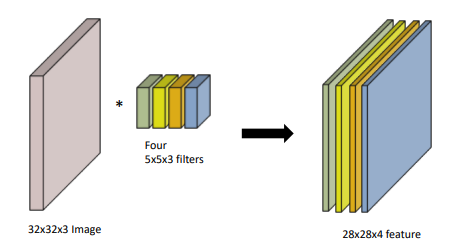


위 과정을 사용하면 Convolution Operation, 2D Image Convolution을 할 수 있다.

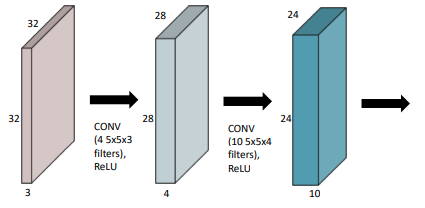
그러면 우리가 Convolution Operation을 한다는 것은 어떤 의미가 있을까? 2D Convolution을 한다는 것은 해당 convolution filter의 모양을 해당 image에다 찍는 것이라고 했었다. 그래서 우리가 적용하고자 하는 filter의 모양에 따라 같은 이미지에 대해 convolution output이 Blur, Embossm Outline이 될 수 있다.



우리가 일반적으로 다루는 것은 RGB image이다. RGB image를 수학적으로 표현할 때 tensor로 표현하기 때문에 32\*32 image가 있고, depth 방향으로 3\*channel이 있으므로 RGB가 들어가 있게 된다. 마찬가지로 우리가 32\*32\*3 image를 convolution한다고 하면 우리가 5\*5 convolution을 한다고 해도 filter의 크기는 항상 같아야 하므로 5\*5\*3크기의 convolution filter을 사용해야 한다.



우리가 일반적으로 convolution을 생각하면 이미지가 들어가서 여러 개의 channel을 가지는 convolutional feature map이 나오게 된다. 이때 feature맵의 channel숫자는 convolution filter의 개수와 동일하다. 그래서 우리가 input channel과 output convolution featurn map의 channel을 알면 여기에 적용되는 convolution filter의 크기 역시 알 수 있다.



하나의 convolution만이 아니라 여러 번 수행할 수도 있다. 그리고 Multiple Layer Perceptron에서 배운 것처럼 한 번 convolution을 거치고 나면 그 다음에는 nonlinear activation이 들어간다. 위에서는 RELU 함수가 사용되었다.

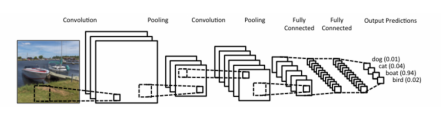
우리가 항상 집중해서 봐야할 것은 이 연산에 필요한 parameter의 숫자를 잘 생각해야 한다. 위에서 32\*32\*3 image에서 28\*28\*4 convolution feature map으로 만들기 위해서는 5\*5\*3\*4개의 parameter가 필요하다. 계산하는 법은 간단하다. convolution filter(혹은 kurnel)의 사이즈(5\*5)에서 input channel 숫자(3), output channel 숫자(4)만큼이 parameter 수가 된다.

마찬가지로 다음번에는 5\*5\*4\*10 개의 parameter가 필요하다.

CNN consists of convolution layer, pooling layer, and fully connected layer.

- Convolution and pooling layers: feature extraction

- Fully connected layer: decision making (e.g., classification)

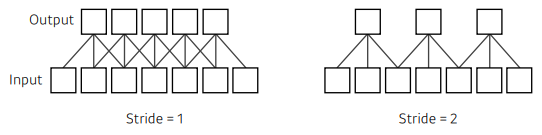


일반적인 Convolution Neural Network는 어떤 convolution layer가 존재한다. 여기에 Pooling layer가 존재한다. 이를 마지막에 다 합쳐서 최종적으로 원하는 어떤 결과값을 만들어 주는 fully connected layer가 된다. 그래서 convolution과 pooling layer가 해주는 것은 이미지에서 유용한 정보를 뽑아주는 feature extraction이 된다.

위 방식으로 만들어진 CNN이 가장 기본적이고 고전적인 CNN 구조이다. 최근 fully connected layer를 없애거나 최소화시키는 추세이다. 최소화시키는 이유는 파라미터 숫자에 dependant하게 되기 때문이다. 머신러닝에서는 내가 학습하고자 하는 어떤 모델의 parameter의 숫자가 늘어나면 늘어날수록 학습이 어렵고 generalization performance가 떨어진다고 알려져 있다. 그래서 CNN이 발전방향은 같은 모델을 만들고 최대한 모델을 deep하게 가져가지만 동시에 파라미터 숫자를 줄이는데 집중한다. 이를 위해 여러 테크닉이 들어가게 된다.

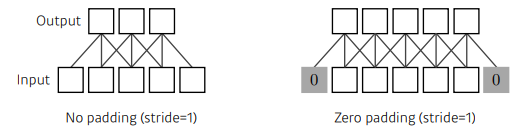
그래서 꼭 기억해야할 것은 어떤 neural network를 봤을 때 이 network의 layer별로 몇 개의 parameter가 필요한지, 전체 parameter는 몇 개가 필요한지 어느정도 감을 가지는 것이 중요하다.

\* Stride



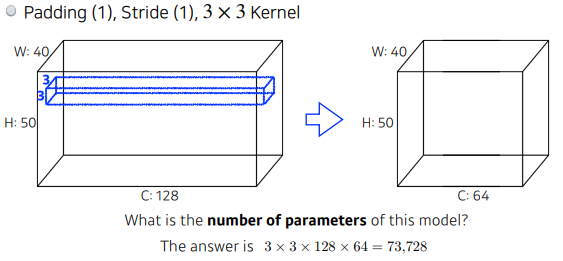
Stride는 넓게 걷는다는 의미이다. Stride가 1인 경우, 내가 가지고 있는 convolution filter를 매 픽셀마다 찍는 것을 말한다. Stride가 2인 경우, filter를 한 번 찍고 바로 옆으로 옮기는 것이 아니라 두 칸을 옮기는 것이다. 위 그림은 1차원이지만 2차원으로 가면 Stride의 파라미터는 2개가 된다.

\* Padding



앞에서 32\*32 이미지가 있을 때 convolution operation을 하게 되면 output이 32\*32가 나오지 않았다. 이는 boundary 정보가 벌어지기 때문이다. 3\*3 filter을 찍기 위해서는 이미지의 가장자리를 찍을 수가 없다. 왜냐하면 filter가 image밖으로 삐져나오기 때문이다. 그래서 이 부분에 어떤 값을 채워줘야 가장자리에 convolution operation을 수행할 수 있다. Padding이 바로 이 역할을 수행해준다.

\* Convolution Arithmetic

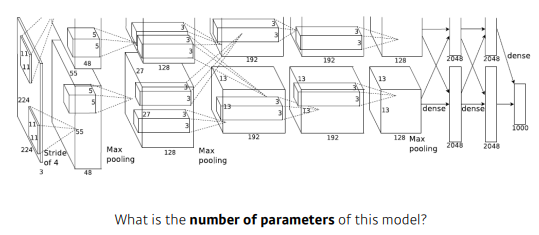


parameter를 계산해보자. 40\*50 입력이 있고 channel이 128이다. 그리고 출력으로 나오는 것은 40\*50이고 channel은 64이다.

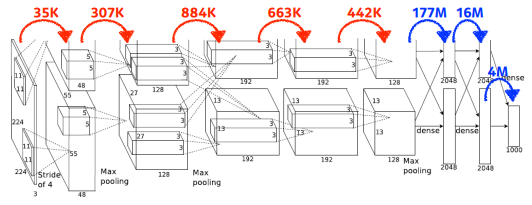
kurnel의 channel은 input data의 channel과 동일하다. 그런데 kurnel은 1개당 convolution feature map 1개를 출력한다. output channel이 64이므로 kurnel의 개수는 64개가 된다.

그래 parameter의 개수는 3\*3\*128\*64가 된다. Padding과 Stride는 parameter의 개수와 연관이 없다.

\* Exercise



위 model에서 각 layer당 parameter의 개수가 몇 개일지 계산해보자.



(35,000)

마지막에 2가 곱해진 이유는 옛날에는 GPU 메모리가 크지 않아서 96 channel의 feature map을 만들어야 하는데 메모리가 부족으로 48channel 두 개로 나누어 계산했기 때문이다.

(307,000)

(884,000)

2개의 Input에서 Interchange가 발생하므로 128에 2를 곱해야 한다.

(663,000)

(442,000)

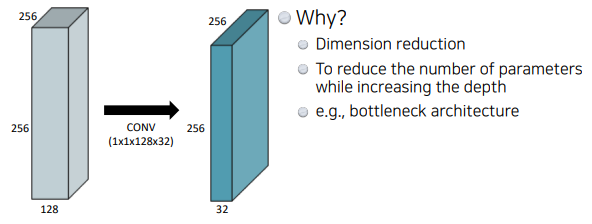


해당 층부터는 dense layer에 속한다. 파라미터의 숫자가 엄청나게 커졌다.





\* 1\*1 Convolution



convolution layer를 더 deep하게 쌓으면서 parameter의 개수는 줄이는 효과를 낸다.