Lecture 5) Convolutional Neural Networks

Lecture 4 review) Neural Networks

- 선형레이어 사이에 비선형성을 추가한 형태

CNN history

- 문자 인식 → LeNet(Gradient-based learning. 1998)
- ImageNet Classification → AlexNet(CNN, 2012)
- 최근 적용되는 분야: Image Recognition, Detection, Segmentation, 자율주행, video 분류, pose recognition, 의료영상분석, 은하, 표지판인식, Kaggle Challenge, Image Captioning ...

CNN 구성요소, 작동방식

*CNN = ConvNet

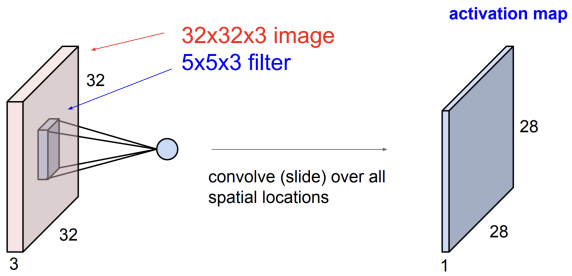
*이미지가 WxHxD라고 할 때, spatial location(WxH), depth(D)라고 부른다.

Fully Connected Layer

- Input(WxHxD)을 Nx1 벡터로 만든 후(stretch), 다음 레이어의 모든 값과 연산하는 레이어
- FC Layer 이후에 activation 역할을 수행
- Input image에 대해 각 클래스 가 어느 정도의 score를 가지는지 계산할 때 사용된다.
- 계속해서 spatial map을 유지하다가, 마지막 FC layer에서 모든 값을 aggregate하는 역할을 한다.

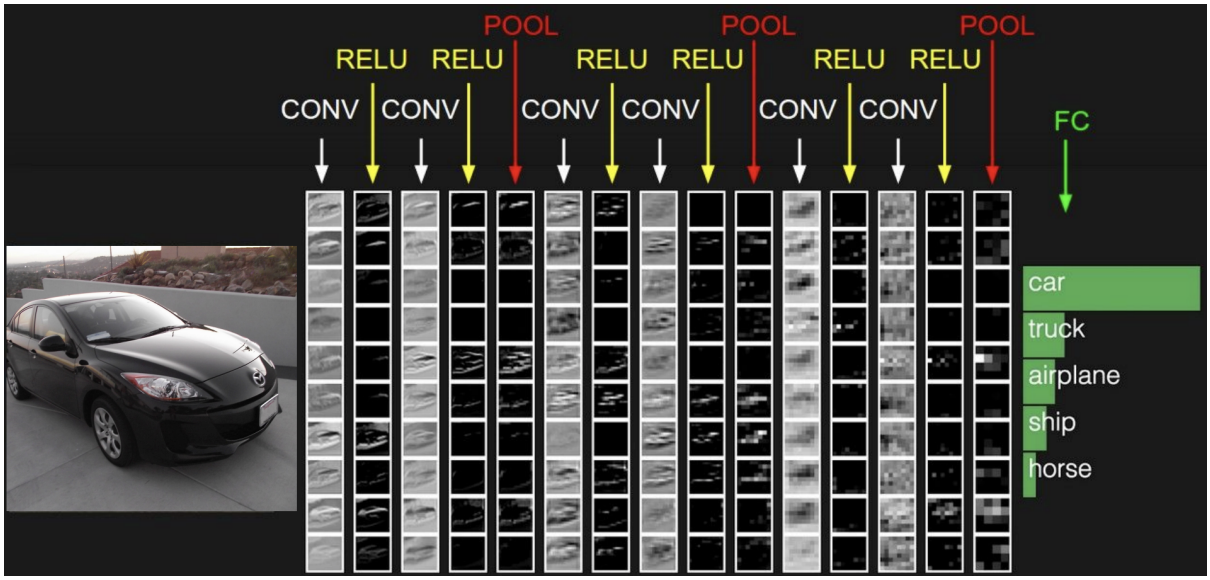
Convolution Layer

- 그림



- 행렬을 벡터로 stretch하지 않고, 기존 input의 형태를 유지한다.
- Filter의 depth는 input image의 depth를 동일하게 사용한다.
- (Filter)와 (이미지의 chunk)를 dot product 연산 → 하나의 chunk를 연산하게 되면, 하나의 값이 나온다. 이를 반복적으로 모든을 slide하며 연산 → activation map들을 모아서 다음 layer의 input으로 사용
- Conv Layer가 깊어질수록, 각 레이어는 간단한 feature에서 점점 복잡한 feature를 학습하게 된다.
  - 초기 filter가 학습하는 feature는 low-level feature(ex. edge)
  - 뒤로 갈수록 high-level feature를 학습하게 된다.(ex. corner, blob)
- Hyper-parameter: filter size, stride, filter 수...
- 하나의 filter는 하나의 activation map을 만든다.
- Filter는 이미지의 region의 템플릿을 얻는다고 해석할 수 있다.

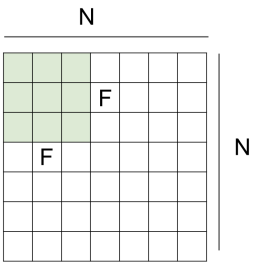
구조



- Conv → ReLU → Conv → ReLU → Pooling Layer → ...
- ConvNet은 convolutional layer들의 연속이다.
- 간단한 선형레이어 사이에 activation function(ex. ReLU)을 끼워놓은 상태이다.(비선형성 제공)
- Filter(conv)를 거치며 activation map을 생성하게 된다.
  - Activation map: 연산의 결과가 되는 output image

Spatial dimension 계산방법

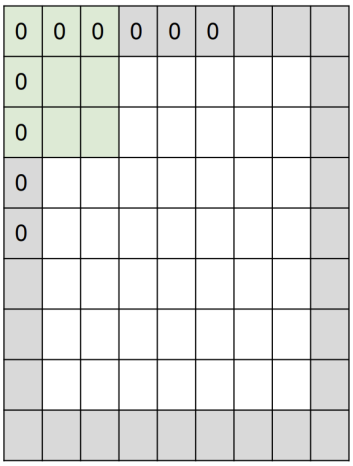
- Stride: 몇 개의 픽셀을 뛰어넘어서 연산을 수행할 것인지
  - larger stride → downsampled image



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

- Padding
  - 테두리를 어떠한 값으로 채우는 것
  - 장점
    - 모서리나 테두리의 픽셀을 필터의 중앙에 위치시킬 수 있다.
    - 이전 input image의 크기를 유지할 수 있다.
  - Zero padding



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

  - 필터의 크기가 커질수록, 이미지의 크기를 유지하기 위해 패딩의 수가 커져야한다.

- Output volume 계산 예시

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

- 레이어 당 Parameter 수 계산 예시

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

- 요약

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

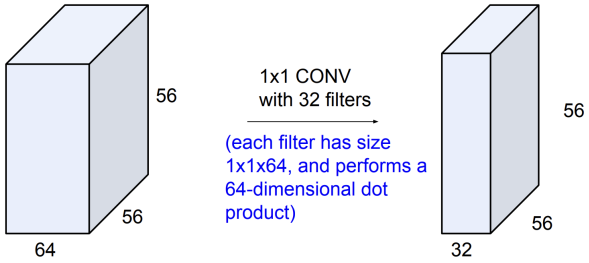
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

  - 일반적으로 filter size는 3x3을 많이 사용하고, stride는 1, padding은 다른 값에 따라 적절한 값을 사용한다고 한다. K값은 2의 제곱수(32, 64, 128 ...)를 많이 사용한다.

1x1 convolution

- volume을 줄이는 데에 사용된다.

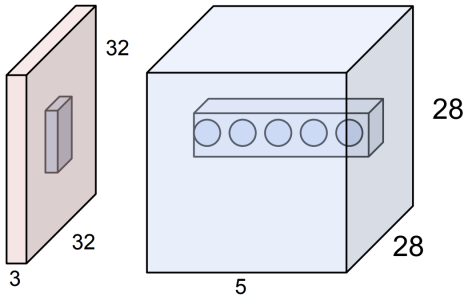
(btw, 1x1 convolution layers make perfect sense)



- Down sampling의 이점
  - Activation map의 사이즈를 줄여주고, 총 파라미터 수를 줄일 수 있다.(연산량 감소)
    따라서, 최종적으로 FC layer의 연산 수가 줄어드는 효과를 얻을 수 있다.

Conv Layer의 brain/neuron view

- Local region을 보는 뉴런이 있다고 생각한다.
- 5x5 filter = 각 뉴런마다 5x5 receptive field가 있다.
  - Receptive field: input field, 한번에 볼 수 있는 영역
- 필터의 크기 = spatial location을 슬라이딩하는 단위(= weight, parameter)
- 하나의 레이어에 5개의 필터가 있다고 했을 때, input image의 동일한 구역을 관찰하는 5개의 서로 다른 뉴런이 있다고 해석할 수 있다.
- 예시



  - 우측의 activation map을 만들기 위해, filter 5개가 필요
- Q. 하나의 레이어에 여러 필터를 썼는데, 필터가 모두 동일하게 나오면? → random init으로 인해 괜찮을듯

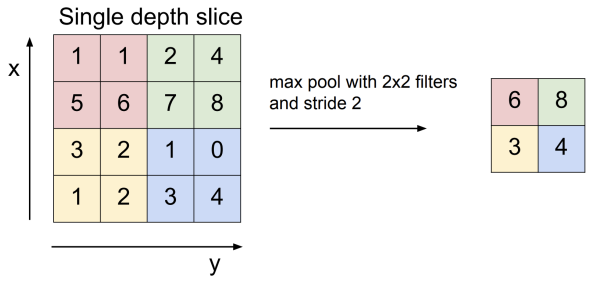
Pooling Layer

= spatial down sampling

- 왜 더 작게 보려고 하는가?
  - 더 적은 파라미터를 위해
  - 각 지역에 invariance(불변성)를 주기 위해
- 특징
  - Pooling 연산은 depth에는 영향을 주지 않는다.
  - Pooling layer에도 filter size, stride를 정할 수 있다. 이들은 pooling 할 지역의 크기를 결정한다.
  - 보통 pooling할 구역이 중복되지 않게 한다.
    → 목적이 down sampling이므로, 한 구역이 하나의 값으로 나오게 하기 위해

max pooling

- 필터 내의 값들 중에 가장 큰 값을 내보낸다.



- Max pooling이 average pooling보다 많이 사용되는 이유?
  - pooling은 activation of neuron을 위한 작업이므로, filter가 각 region에서 얼마나 작동하는지에 대한 값이므로. Max pooling은 이 filter가 해당 구역에서 얼마나 활성화 되었는지 나타내기 좋다고 한다?
- 최근에는 down sampling 시, pooling보다 stride를 더 많이 쓴다.
- pooling 요약

Common settings:

F = 2, S = 2
F = 3, S = 2

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

  - Hyper-parameter: filter size(pooling 대상), stride(output volume과 연관)
  - 보통 zero padding은 pooling에서 사용되지 않는다. Pooling 자체가 down sampling을 위해 사용되기 때문이다.

최근 트렌드

- 작은 filter 크기, deeper 아키텍처
- Pool/FC layer를 제(conv만 사용하는 추세)
- 전형적인 아키텍처
  - $[(CONV - RELU) * N - POOL?] * M - (FC - RELU) * K, SOFTMAX$
    where N is usually up to ~5, M is large, 0 <= K <= 2.

  → ResNet, GoogLeNet이 패러다임을 바꾸는 중