컴퓨터비전특론 7th week summary

[Lec08: Training Neural Networks Part 2] 이어서

**Regularization**

- Overfitting 문제를 해결하기 위해 loss function에 regularization을 적용하는 방법(W 안의 원소의 크기를 줄여 overfitting을 완화시킴)

🡪 L2 regularization

🡪 L1 regularization

🡪 Elastic net (L1 + L2)

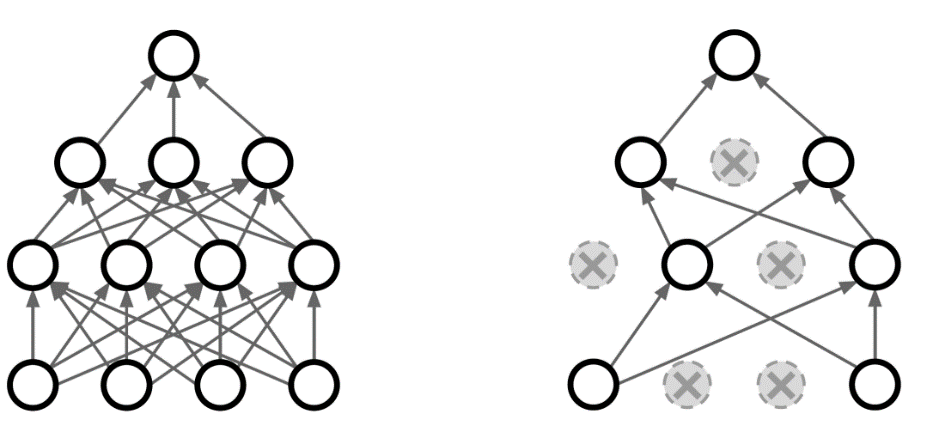
🡪 Max norm regularization:

- Input 또는 network에 randomness를 추가하는 방법

Data augmentation, dropout, batch normalization 등이 있다.

🡪 Training: randomness를 추가

🡪 Testing: Average out randomness: randomness로 인해 발생할 수 있는 여러가지 결과를 averaging

Dropout

Overfitting 방지를 위해 특정 비율만큼의 노드를 random하게 drop시켜서 feed forward 과정에서 사용하지 않음.

🡪 네트워크가 중복되는 정보를 갖는 것을 줄여준다.

Dropout의 average out in test: model ensemble

그러나 실제로 integral을 계산하는 과정이 복잡하므로 이에 대한 approximation으로 original output에 dropout probability를 곱하여 average를 계산한다.

ex. 이고 dropout probability 일 때

: .가 선택되는 경우의 수는 (0,0), (0,1), (1,0), (1,1)로 총 4가지, 모두 0.5\*0.5의 확률

->

\*nonlinear layer를 적용할 경우 실제 integral 값과 근사값이 정확히 일치하지는 않는다.

Data Augmentation

Image에 randomness를 적용하는 방법으로, 여러 방법으로 random하게 이미지를 변형시킨다.

- Geometric transform: horizontal flip, translation(이미지가 x축 또는 y축을 따라 이동), rotation, stretching, shearing 등

- Lens distortion: 물체가 굽어지는 효과

- Random crops and scales

- Color jitter: RGB 픽셀에 PCA를 적용하여 RGB가 분포할 수 있는 영역 중 가장 대표적인 방향을 따라 RGB를 변경해준다.

DropConnect

네트워크의 뉴런 수는 동일하게 유지하면서 weight parameter의 일부를 0으로 만든다. (주로 DropConnect 보다는 Dropout이 많이 사용된다.)

Fractional Max Pooling

Input data size가 , output은 이라고 할 때 의 크기로 max pooling을 적용한다. 일반적인 max pooling과 다르게 할 때마다 결과가 달라지는 stochastic한 방법이다.

Stochastic Depth

네트워크 내부의 layer들을 사용할지 말지 stochastic하게 결정하여 layer의 일부를 건너뛰는 방법.

Transfer learning

Training data가 부족할 때, 큰 데이터로 먼저 모델을 pretrain 한 이후 그 모델을 이용하여 하고자 하는 task를 진행하는 방법. 데이터의 크기와 pretrain data와의 유사성에 따라 얼마나 finetuning을 얼마나 할지가 달라질 수 있다.

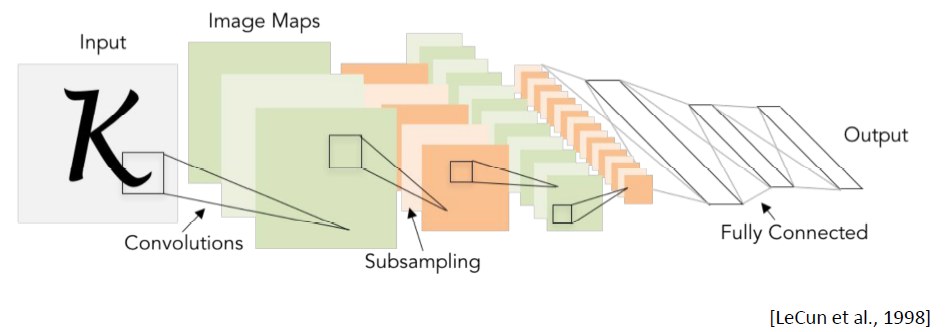
- very little data & very similar dataset: 가장 마지막의 linear classifier만 finetune

- lots of data & very similar dataset: 몇 개의 레이어를 finetune

- lots of data & very different dataset: 많은 레이어를 finetune

[Lec09: CNN Architectures]

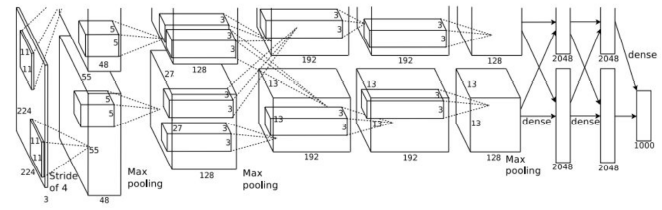
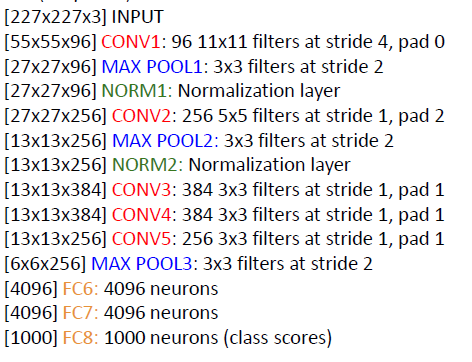
LeNet-5



5 x 5, stride=1인 filter를 사용하여 convolution & 2 x 2, stride=2인 filter로 pooling

Convolution layer가 처음 나온 모델로 현재까지 만들어지는 CNN 모델들도 이와 유사한 구조를 갖는다.

AlexNet



- 첫번째 layer(CONV1)

: 11 x 11, stride=4인 96개의 필터 사용  
 🡪 output size: (227-11)/4+1 = 55, output volume [55 x 55 x 96]

🡪 Total number of parameters: (11\*11\*3)\*96 = 35K, Bias: 96

- 두번째 layer(POOL1): 3x3, stride=2인 필터 사용

🡪 output size: (55-3)/+2+1=27, output volume [27\*27\*96]

🡪 parameter 개수: 0개 (weight x)

- conv1, conv2, conv4, conv5: 같은 GPU 상에 있는 feature map과의 connection만 있음

- conv3, fc6, fc7, fc8: 이전 layer의 모든 feature map과 연결

- ReLU 함수를 처음으로 사용한 모델

- norm layers(여기서 사용한 방법은 요즘에는 쓰이지 않음!)

- heavy data augmentation

- p=0.5의 dropout 사용, batch size=0.8, SGD momentum=0.9, learning rate=0.01, L2 weight decay: 5e4

ZFNet

Alexnet에서 filter size를 변경하여 더 낮은 error를 얻음.

VGGNet

- 3x3, stride=1, padding=1 size의 convolutional layer와 2x2, stride=2의 max pooling layer만 사용: 작은 필터를 여러 개 반복

🡪 smaller filters, deeper networks

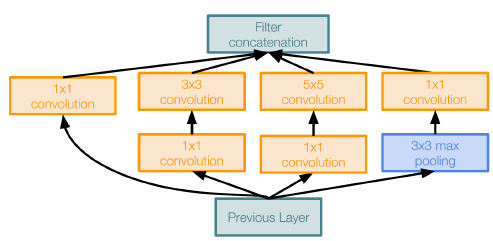
🡪 Alexnet보다 파라미터 수가 훨씬 적으면서도, 비선형함수를 여러 번 통과하기 때문에 더 많은 nonlinearity를 가질 수 있다.

- 총 파라미터 수에서는 Fully connected layer가 차지하는 비중이 크다.

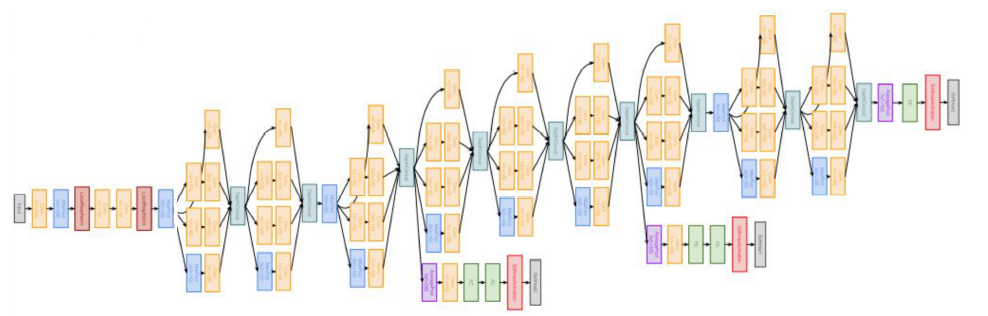
- 소요되는 총 메모리는(VGG16) 96MB로, 초반의 conv layer에서 activation 저장을 위해 메모리가 많이 소요된다.

- local response normalization 사용 x

- FC7는 주로 다른 task로 일반화되어 사용된다.

GoogLeNet

- Inception module의 사용: network in network 구조. 이전 layer에서 들어온 값들이 여러 layer를 병렬적으로 통과한 뒤 마지막에 concatenate된다. 이 때 병렬 계산을 하게 되면 channel size가 크게 증가하고 계산량이 너무 커지므로, 1x1 conv layer인 bottleneck layer를 이용하여 feature depth를 줄임으로써 계산량도 줄인다.

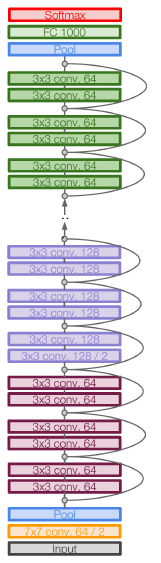
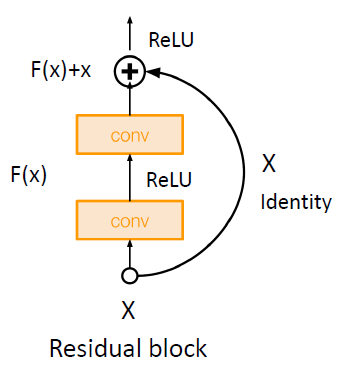


- 최종적으로 나오는 output 이외에도 layer 중간에 2번의 auxiliary classification output이 출력된다. 이 output들을 이용하여 lower layers에서 gradient를 얻을 수 있고, 이를 통해 초반부 layer들의 gradient를 더 잘 전달할 수 있다.

- 총 22개의 layer로 구성되어 있으며, 하나의 inception module에는 2개의 layer가 있는 것으로 본다. Auxiliary output layer는 layer로 count하지 않는다.

- 파라미터 수는 적지만 모델이 복잡하므로 계산량이 많다.

ResNet



- Residual connection을 이용한 매우 깊은 network

- Layer의 개수가 152개로 이전의 CNN 모델들에 비해 매우 많다.

- deep network의 구조에서의 optimization: identity mapping을 통해 앞의 간단한 구조에서 나온 값들을 그대로 전파함으로써 gradient기 소실되지 않은 채로 전달된다.

- Residual mapping: H(x) = F(x)+x의 H(x) 대신 residual F(x)=H(x)-x를 fitting한다.