7강 cnn

Filter를 이미지 위에서 슬라이드 하면서 dot product 연산.

Ex) 32x32x3 을 5x5x3 필터로 (channe=depth 수는 같다) (한번 slide당 75번의 연산이 있다.) => 1 slide당 1개의 숫자로 결과 나옴. => 다음 activation map(conv, relu 거친 후)은 28\*28\*(필터의 수)이 된다. => 필터는 파라미터가 된다. (공유하는, 초기화는 랜덤하게)

처음 이미지 -> low-level feature(엣지나 컬러 등등, 조각의 blob) -> mid-level feature(조각 하나하나를 통합해서 볼 수 있다.) -> high-level feature(상당히 좀 더 상위 레벨의 이미지) <필터를 visualization 했을 때> => 좋은 웨이트(좋은 피쳐를 얻기 위해 conv 하는 것이다.)

Activation map에서 하얀 부분이 activation이 높은 지점.

Filter 거친 map들의 뭉치 = volume이라 함

Convolution 공식 = (n – f) / stride+1 = output size

7\*7에 3x3 필터 stride 3이면 적용 불가 => 적용 안함. = 정수로 떨어지지 않으면 사용 x = 인위적인 예외처리는 가능.

Zero-padding도 가능 7x7에 0으로 padding하면 9x9의 이미지고 원래 크기가 유지됨.

Filter 3 => pad = 1 / 5->2 / 7->3 (인풋과 크기 같게 하려면)

왜 보존이 중요한가? 수많은 layer를 거치다 보면 volume이 shrink => 더 이상 convolution 불가 => 레이어 많으면 convolution 불가 => padding 필요 => 하지만 사이즈 줄이는거도 중요 (downsampling도 중요) = 풀링에서 해결

(n+2p-f)/s+1

5\*5\*3 필터가 10개 + 바이어스까지 합치면 => 760개의 파라미터

즉, w1\*h1\*d의 볼륨을 받아 새로운 w2\*h2\*d2의 볼륨을 생성

필터의 개수는 일반적으로 2의 자승 => computational 차원에서

- 근데 1\*1필터를 왜하냐? 1\*1\*64로 56\*56\*64모든 뉴런을 거쳐 convolution 3차원 공간의 64depth를 거쳐 컨볼루션하기 때문에 의미가 충분이 있다 (2차원 1\*1 => 56\*56 이라면 의미 없을 것임)

좀 더 뉴런의 의미에서 보면 activation map들의 각 뉴런(픽셀)은 dnn에서 처럼 wx summation에 +b한것과 같은 논리

=> 동일한 filter로 부터온 동일한 depth의 뉴런들은 파라미터 공유!

Pooling은 channel 수는 같지만 downsample 하는 것(파라미터, 웨이트, 패딩 모두 없다.) => maxpool vs avgpool(2\*2 filter stride 2로 생각)

줄이면 근데 정보 손실 아님? 의도적으로 이렇게하는게 오히려 좋다. 왜?? 약간의 정보 손실로 invariance한 능력을 얻게 된다.

주요 conv net

Lenet-5

Alexnet => input = 227\*227\*3 => 위에 stream과 아래 stream 두개로 구성 = gpu의 성능이 많이 딸려서 두 개의 gpu를 사용하기 위해 => 지금은 하나로 묶어도 됨. First layer => 96개의 11\*11\*3 filter with stride 4 => 55\*55\*96이 된다.

레이어의 첫번째 전체 파라미터는 ? 11\*11\*3\*9 = 약 35K …. 쭉쭉

Normalization layer는 지금은 효용이 없다하여 사용 x

필터의 수는 점차 많아지지만 사이즈는 역시 줄어든다.

FC7 layer는 통칭인데. Classifier 직전 레이어라고 하면됨

최초로 relu 사용, norm layer 사용, data augmentation 많이 사용, dropout 0.5(fc layer에서만), batch size는 128, 7 cnn ensemble (7가지 모델을 앙상블) => 18.2에서 15.4로 에러 줄임 (앙상블 꼼수 아니냐)

ZFNet =>alexnet과 거의 유사 하지만 필터나 그런거 좀 바꿈 (2014년)

Vggnet => 2014년 오직 3\*3 stride 1 pad1 만 그리고 2\*2 maxpool stride 2만 적용!(레이어 개수는 여러 개 옵션. 깊을수록 좋다)

메모리 => 하나 이미지당 93MB(포워드) / 백워드까지 고려하면 2배 // 파라미터는 138M개

근데 fc에서 파라미터 매우많음 무려 7\*7\*512\*4096 => 비효율적이라 생각 => 7\*7을에버리지 풀링해준다. => 512의 단일 컬럼 가지도록 한다. 파라미터 줄이며 성능 비슷 => 구글넷에 활용

Googlenet => 인셉션 모듈 활용

7\*7\*1024를 에버리지 풀링해서 1\*1\*1024 하게 된다. = 5백만개로 줄임 1/12 수준의 파라미터 연산은 두배이상 빠름

Resnet 2015년

레이어가 구글넷은 22개 => 이번에는 152개..? error rate는 급 감소

하지만 layer 많아지면 문제… 기존의 network는 layer가 많아지면 오히려 error rate가 커지네?? 아직 뭔가 최적화에 실패 resnet은 반대 layer 많아져도 error rate 작아짐

어떻게 했나? 첫 레이어의 Conv 다음 바로 pooling(초기에 사이즈 줄여 효율적인 연산) => skip connection -> 효율적으로 뒷레이어에서 제일 앞으로?? 좀더 보기

일단 개오래걸림. 8개 gpu 2~3주 / test 시에는 vgg 보다 빠르다 근데

Batch normalization 사용 => lr 높게해도 괜찮, dropout도 사용 안해도된다.

Alphago: 19\*19\*48이 입력 => 최종 아웃풋 => 1\*1필터 해서 아웃풋이 1000개 클래스가 아니라 19\*19로 나온다.