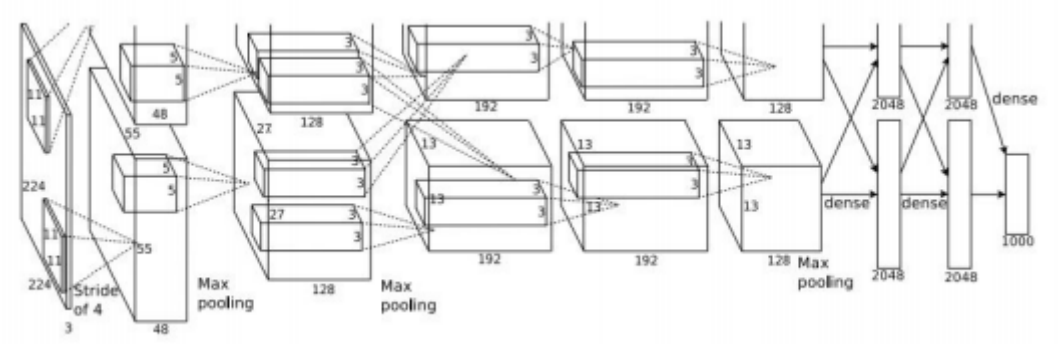
Lec 9. CNN Architectures

LeNet-5: 성공적인 최초의 ConvNet

- stride = 1 인 5×5 filter
- Conv와 Pooling Layer 거친 후 FC Layer가 붙음
- 간단한 모델이나, 숫자 인식에서 좋은 결과를 얻음

AlexNet

- 최초의 large scale CNN
- ImageNet Classification task에서 좋은 성능을 보임
- Conv-Pool-Norm 구조 두번 반복 후 Conv Pool FC layer가 붙음
- LeNet과 유사, layer가 더 많아짐(5개 conv, 2개 FC)
    - 참고) Pooling layer에는 parameter가 없음
- ReLU 사용
- Norm layer 있음(지금은 사용 x)
- data augmentation을 많이 함 (flipping, jittering, color norm)
- dropout 0.5, batch size 128, SGD momentum 0.9, initial lr 1e-2, final lr 12-10, weight decay, model ensemble
- 모델이 두개로 나누어져 교차하는 것을 볼 수 있음(GPU의 용량 한계 때문)
    - Conv 1,2,4,5에서는 같은 GPU 내의 feature map만 이용
    - Conv 3, FC 6,7,8에서는 이전 input layer의 전체 depth를 전부 가져옴
- transfer learning에 많이 사용되었음



ZFNet

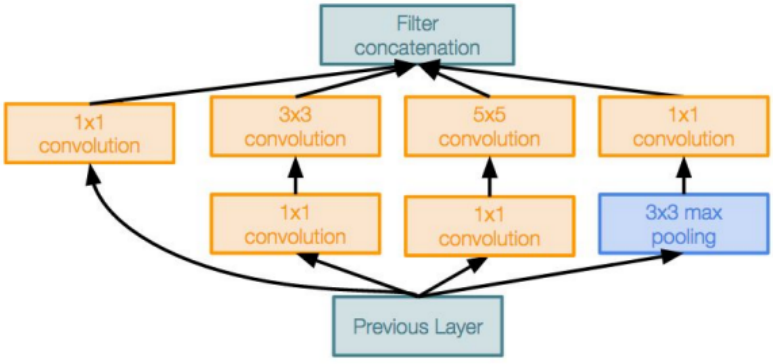
- AlexNet의 hyperparameter를 개선한 모델

VGGNet(이 때부터 네트워크가 훨씬 깊어짐)

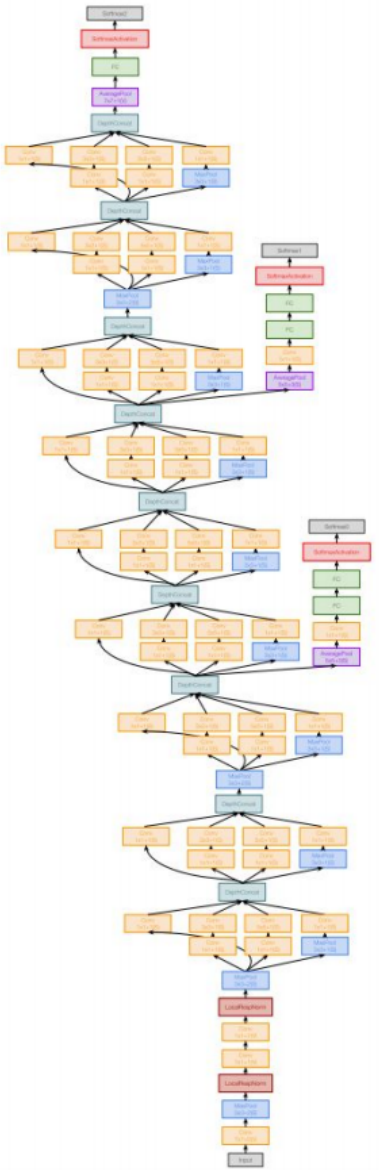
- 더 깊은 네트워크(16-19 layer)
- 더 작은 filter(3x3)
  - depth를 더 키울 수 있음
  - 7×7 filter와 실질적으로 동일한 receptive field를 가지면서도 더 깊은 layer를 쌓을 수 있음
  - non linearity 추가 가능
  - parameter 줄어듦
- AlexNet과 비슷한 패턴 반복: Conv와 Pooling layer가 반복적으로 진행. 단, local response normalization은 사용하지 않음
- VGG16과 VGG19가 있는데, 19는 성능이 아주 조금 올라가지만 메모리도 더 사용하므로 주로 VGG16 사용
- 마지막 FC7은 4096 사이즈의 레이어로, 아주 좋은 featrue representation을 가짐

GoogLeNet

- 22 layer
- FC layer는 없음 → 파라미터 감소
- Inception Module 쌓아 사용
  - network within a network
  - 동일한 input을 받는 서로 다른 다양한 필터들이 병렬로 존재
  - 각 계산을 거친 후, concatenate
  - 계산 비용 문제 해결을 위해 inception module 내의 병렬적인 layer들에 각각 1×1 conv적용
- auxiliary classifier 존재
  - 작은 부가 네트워크
  - ImageNet trainset loss 계산함 → 네트워크가 깊기 때문 → 추가 시 중간 레이어의 학습을 도울 수 있음



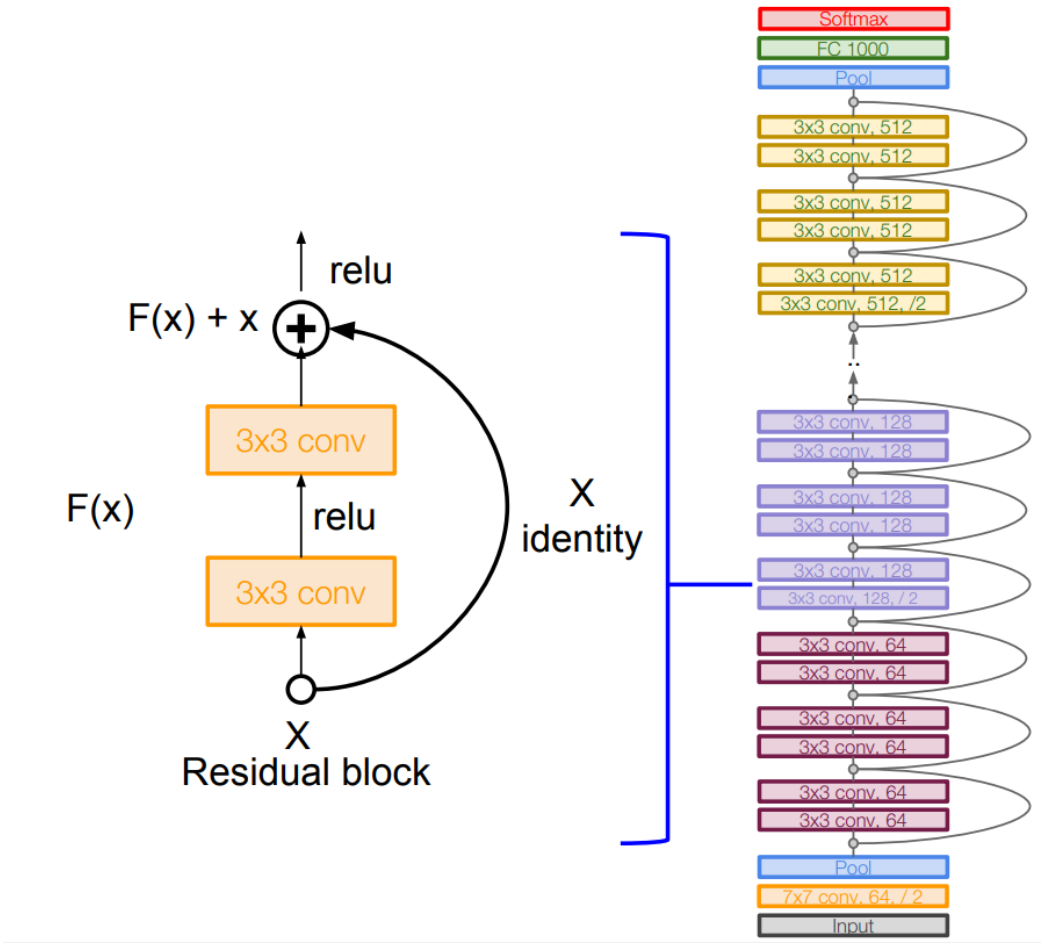
Inception module



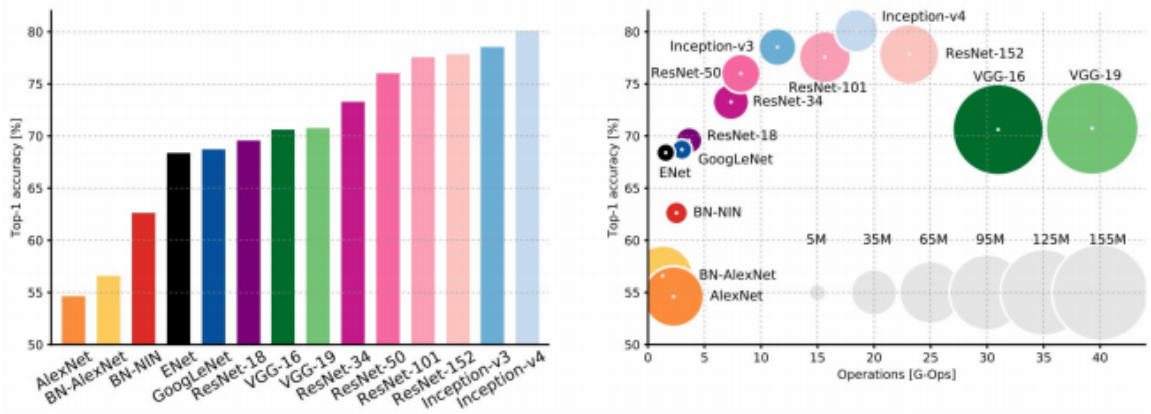
ResNet

- 아주 깊은 네트워크 (152)
- 네트워크가 깊어질수록 더 좋은 성능을 내는가 질문 → 성능이 더 안 좋을 수 있음 (optimization 문제) → network가 깊어질수록 optimization이 어려울 수 있음
- Residual block 쌓아 올림
  - H(x) - x라는 변화량을 학습 (잔차/residual)
  - Skip connection 도입
  - 두 개의 3×3 conv layer

- 주기적으로 filter를 두배 씩 늘리고 stride 2를 이용해 downsampling
- 초반에는 Conv layer가 붙고 FC layer가 없음. 대신 Global Average Pooling 진행
- 마지막에는 1000개의 클래스 분류를 위한 노드가 붙음
- model의 depth가 50이상일 때 bottleneck layer를 도입(depth 조절)
- Conv layer 다음에 batch norm 사용
- 초기화 xavier
- minibatch size 256, weight decay, no dropout, lr scheduling, 2로 나누는 scaling factor 추가(SGD + Momentum에서 초기화 성능)
- 네트워크가 깊어질수록 train error는 줄어듦



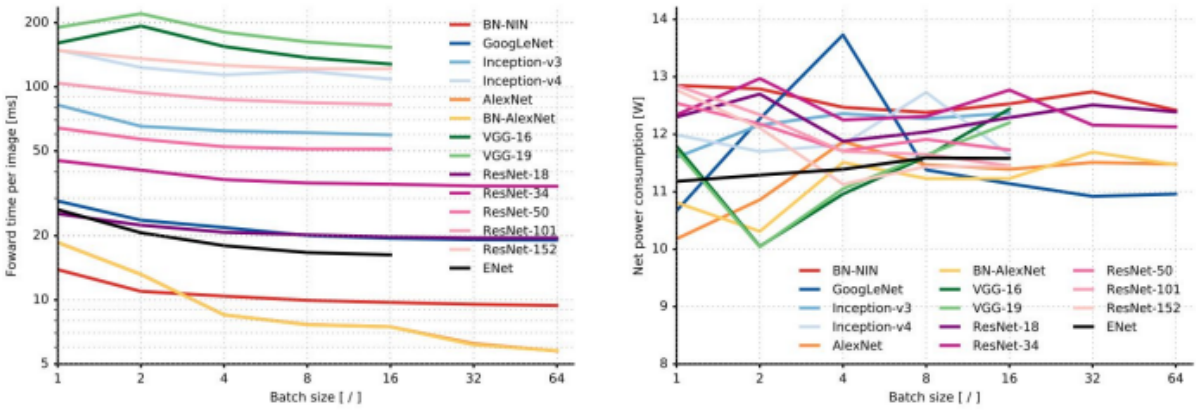
Comparing complexity...



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

앞서 살펴본 모델들의 model complexity 비교

Forward pass time and power consumption



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

forward pass time과 power consumption 비교

다른 architecture

- Network in Network (NiN)
  - MLP conv layer를 기본 아이디어로
  - 각 conv 안에 MLP 쌓음
  - ResNet, GoogLeNet보다 먼저 bottleneck 개념 정립
- Identity Mappings in Deep Residual Networks

    - ResNet 파생
    - ResNet block path 조절
- Wide Residual Networks
    - depth보다 residual이 중요하다고 주장
    - Residual Connection이 있다면 네트워크가 더 깊어질 필요가 없다고 주장
    - residual block을 더 넓게 만듦 → filter 크게
    - 병렬화되어 계산 효율 증가
- ResNeXt
    - ResNet의 저자
    - width 관련 연구
    - residual block 내에 다중 병렬 경로를 추가함
    - thinner block을 병렬로 여러개 묶음
    - wWide ResNet, Inception Module과 연관
- Stochastic Depth
    - depth에 관란 연구
    - train time에 layer의 일부를 제거 → vanishing gradient problem에 대한 sol
    - dropout과 유사
- FractalNet
    - non-ResNet
    - shallow/deep network의 정보 모두를 잘 전달하는 것이 중요하다 생각
    - shallow/deep path를 모두 output에 연결
    - train에는 일부, test에는 full network 사용
- DenseNet
    - Dense Block
    - 한 layer가 그 하위의 모든 layer와 연결됨
    - vanishing gradient problem 완화
- SqueezeNet

- 효율성 중시
- fire module
- squeeze layer, expand layer