2021 Spring

Artificial Intelligence & Deep Learning

Prof. Minsuk Koo

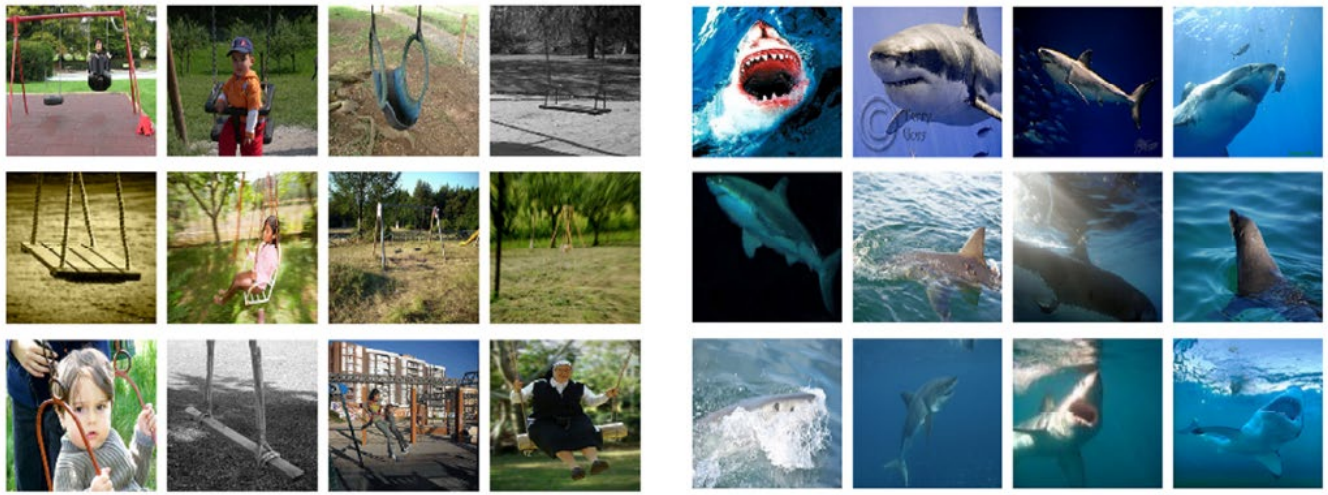
Department of Computer Science & Engineering
Incheon National University

4.4 컨볼루션 신경망 사례연구

- 4.4.1 AlexNet
- 4.4.2 VGGNet
- 4.4.3 GoogLeNet
- 4.4.4 ResNet

컨볼루션 신경망 사례연구

- 자연영상 분류라는 도전적 문제
  - ImageNet 데이터베이스
    - 2만 부류에 대해 부류별로 500~1000장의 영상을 인터넷에서 수집하여 구축하고 공개
  - ILSVRC 대회 (CVPR 학술대회에서 개최)
    - 1000부류에 대해 분류, 검출, 위치 지정 문제: 1순위와 5순위 오류율로 대결
    - 120만 장의 훈련집합, 5만 장의 검증집합, 15만 장의 테스트집합
    - 우승: AlexNet(2012) → Clarifi팀(2013) → GoogLeNet&VGGNet(2014) → ResNet(2015)
  - 우승한 CNN은 프로그램과 가중치를 공개함으로써 널리 사용되는 표준 신경망이 됨



(a) 'swing' 부류 (b) 'Great white shark' 부류

그림 4-20 ImageNet의 예제 영상

AlexNet -1

- 구조
  - 컨볼루션층 5개와 완전연결(FC)층 3개
    - 8개 층에 290400-186624-64896-43264-4096-4096-1000개의 노드 배치
  - 컨볼루션층은 200만개, FC층은 6500만개 가량의 매개변수
  - FC층에 30배 많은 매개변수 → 향후 CNN은 FC층의 매개변수를 줄이는 방향으로 발전

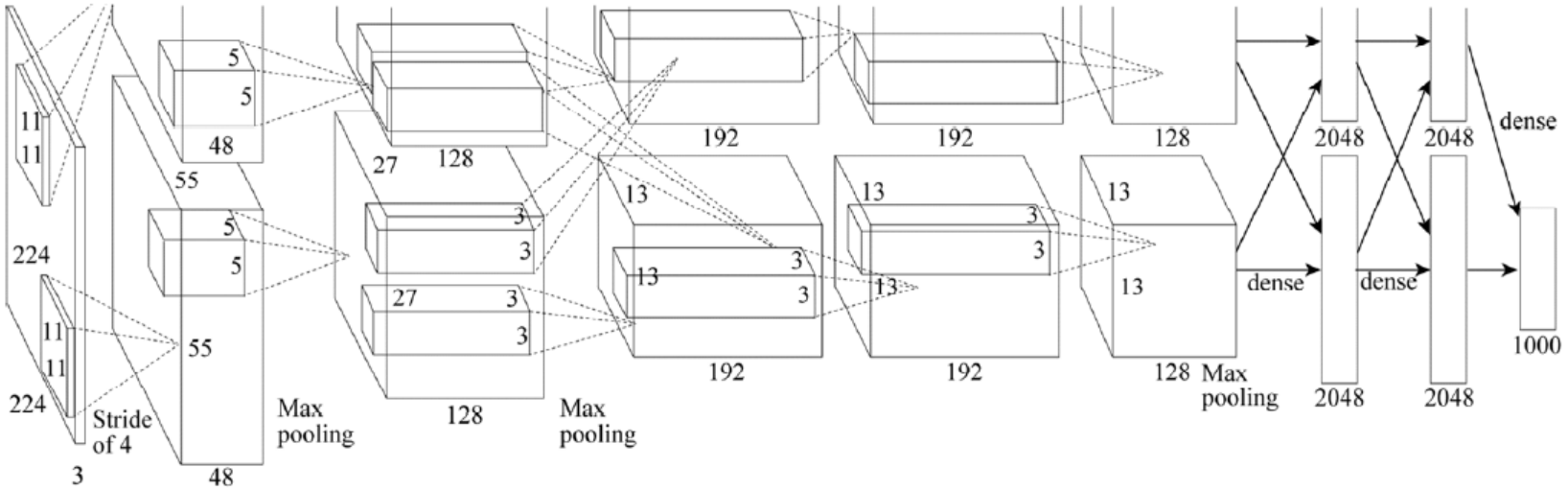
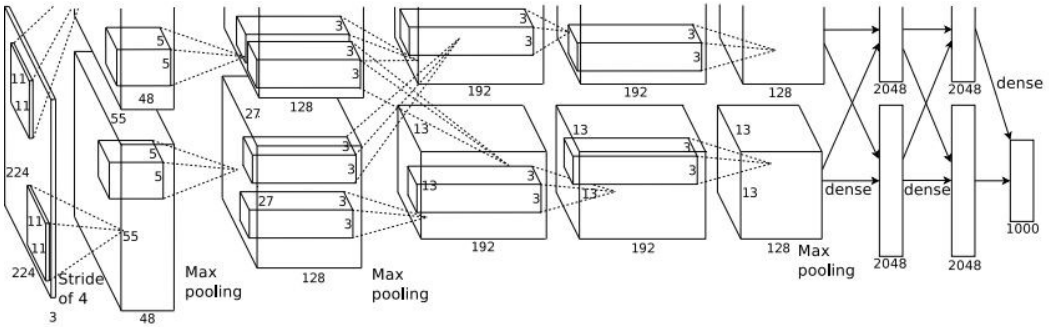


그림 4-21 AlexNet 구조[Krizhevsky2012]

AlexNet -2

Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>

Q: what is the output volume size? Hint: (227-11)/4+1 = 55

Q: What is the total number of parameters in this layer?

Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
=>

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

Q: What is the total number of parameters in this layer?

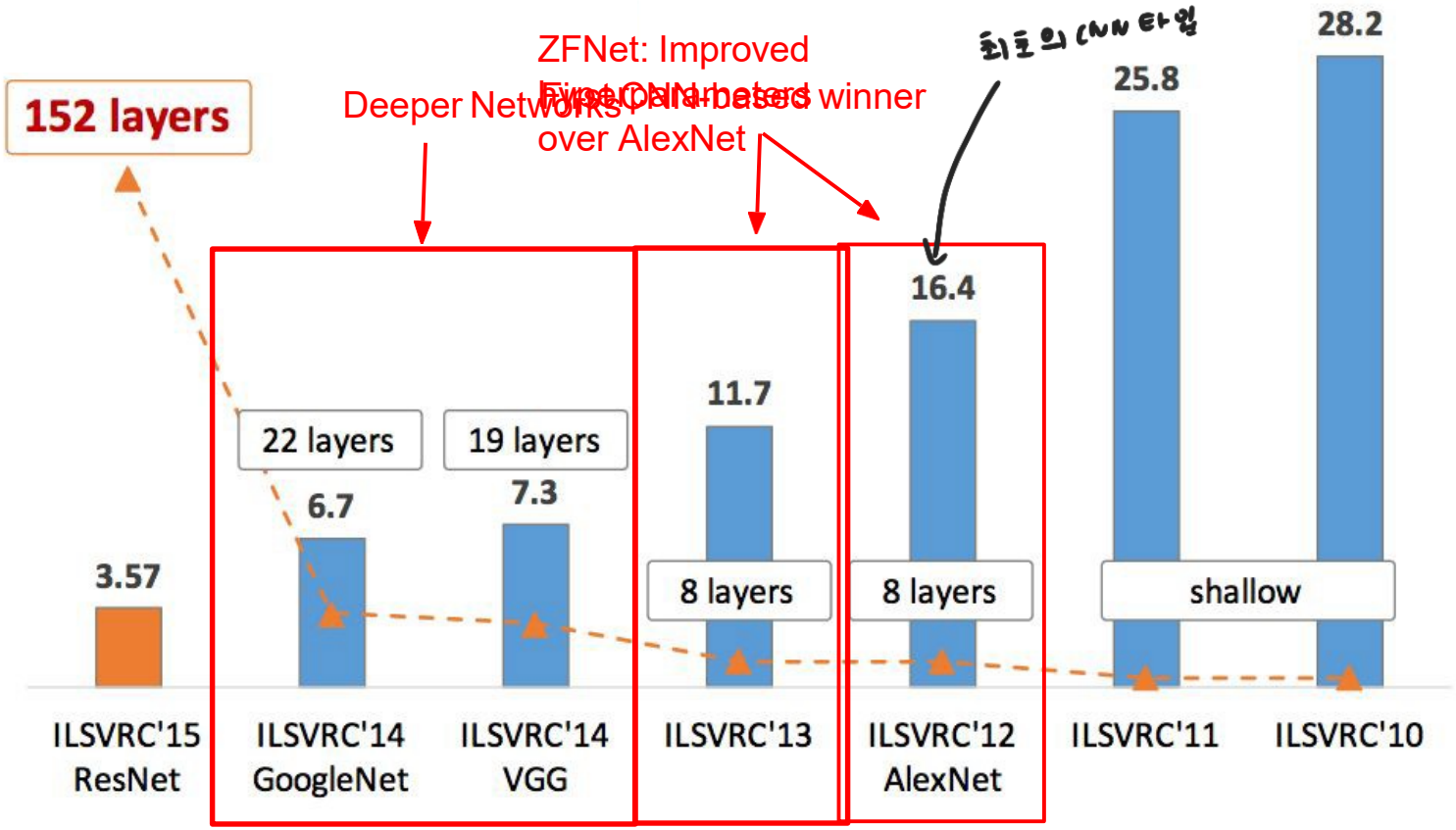
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

AlexNet -3

- AlexNet이 학습에 성공한 요인
  - 외부 요인
    - ImageNet이라는 대용량 데이터베이스
    - GPU를 사용한 병렬처리
  - 내부 요인
    - 활성함수로 ReLU 최초로 사용
    - 지역 반응 정규화 기법 적용 (더 이상 사용하지 않음)
    - Batch size 128, SGD Momentum 사용
    - 학습률 1e-2 -> 1e-3
    - 과잉적합 방지하는 여러 규제 기법 적용
      - 데이터 확대(잘라내기와 반전으로 2048배로 확대)
      - 드롭아웃 등
- 테스트 단계에서 앙상블 적용
  - [그림 5-26]과 [그림 12-5]
  - 2~3%만큼 오류율 감소 효과

Note:
Trained on GTX 580 GPU with only 3 GB of memory.
Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

ILSVRC Winners



VGGNet -1

- VGGNet의 핵심 아이디어
  - 3*3의 작은 커널을 사용하여 신경망을 더욱 깊게 만듦
  - 컨볼루션층 8~16개를 두어 AlexNet의 5개에 비해 2~3배 깊어짐
- 16층짜리 VGG-16(컨볼루션 13층+FC 3층) [그림 4-22]

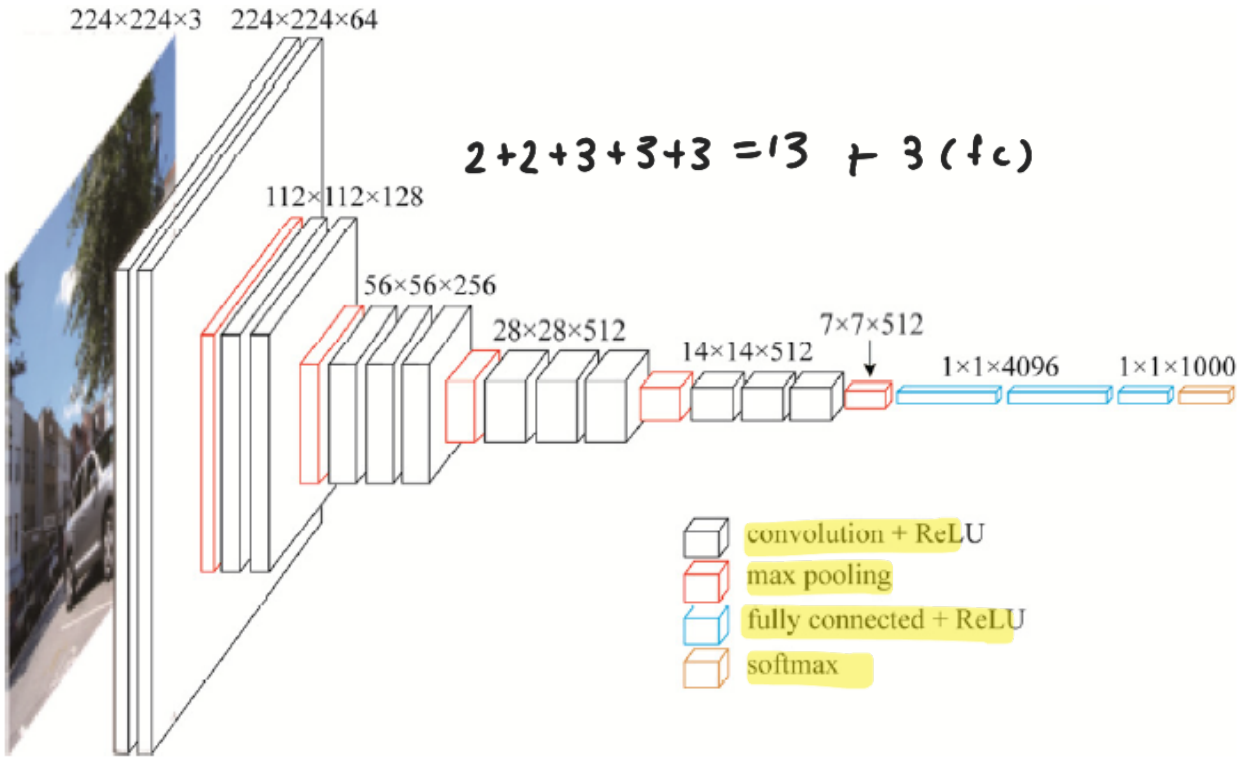


그림 4-22 VGGNet 구조[Simonyan2015]

VGGNet -2

- 1*1 커널 ↪ Bottle neck Network
  - 차원 축소 효과
  - [그림 4-23]의 예)
    - $m*n$의 특징 맵 8개에 1*1 커널을 4개 적용 → $m*n$의 특징 맵 4개가 됨
    - 다시 말하면, $8*m*n$ 텐서에 $8*1*1$ 커널을 4개 적용하여 $4*m*n$ 텐서를 출력하는 셈
  - ReLU와 같은 비선형 활성함수를 적용하면 특징 맵의 분별력 증가
  - 『네트워크 속의 네트워크(NIN)』에서 유래 [Lin2014]
  - VGGNet은 적용 실험을 하였지만 최종 선택하지는 않음 (GoogLeNet이 많이 사용함)

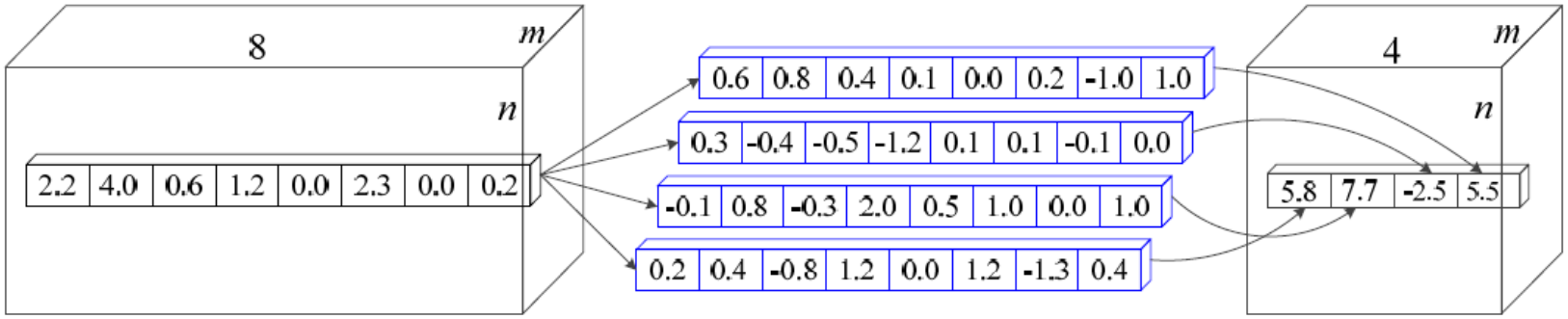
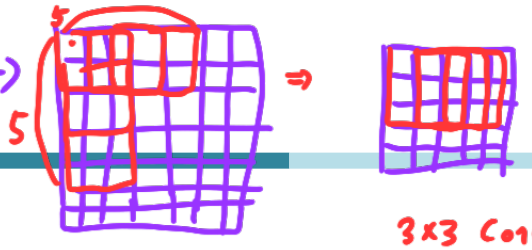


그림 4-23 1*1 컨볼루션 예제

VGGNet -3



3 x (3x3) vs 1 x (7x7)

Small filters, Deeper networks

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

= 7x7 하나랑 3x3 3개랑 동일한 효과

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

[7x7]

But deeper, more non-linearities

And fewer parameters: 3 * (3^2C^2) vs. 7^2C^2 for C channels per layer

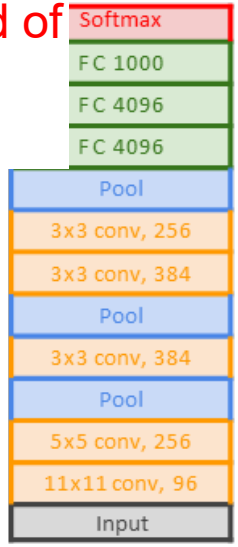
① ②

if C = 3 : ① 27 x 9 = 243

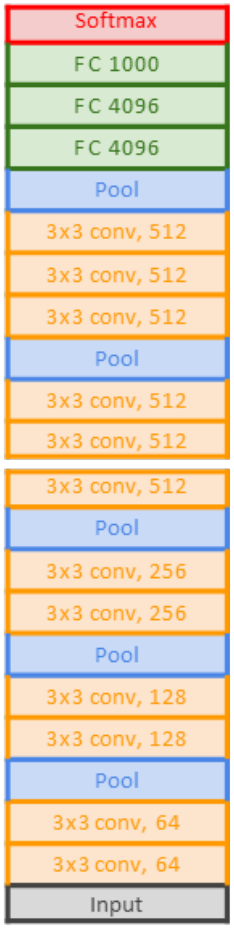
② 49 x 9 = 441

3x3 Conv, stride = 1
Pad = 1

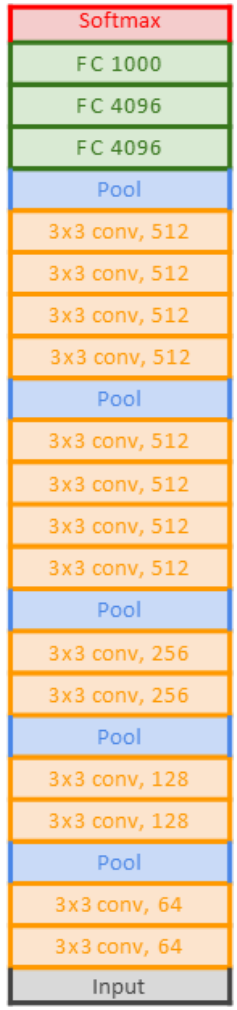
2x2 Max Pool
Stride = 2



AlexNet



VGG16



VGG19

VGGNet -4

메모리 용량 학습 파라미터

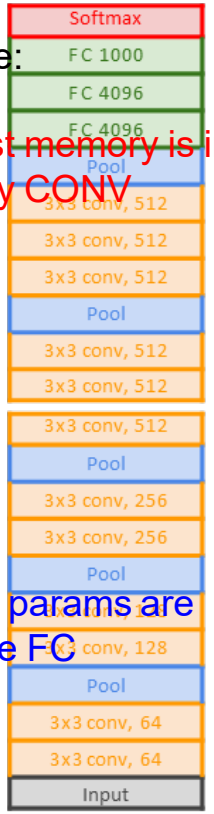
(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912 C
ONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824 CO
NV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648 C
ONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296 CO
NV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296 C
ONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296 CO
NV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448 F
C: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Note:

Most memory is in early CONV

Most params are in late FC



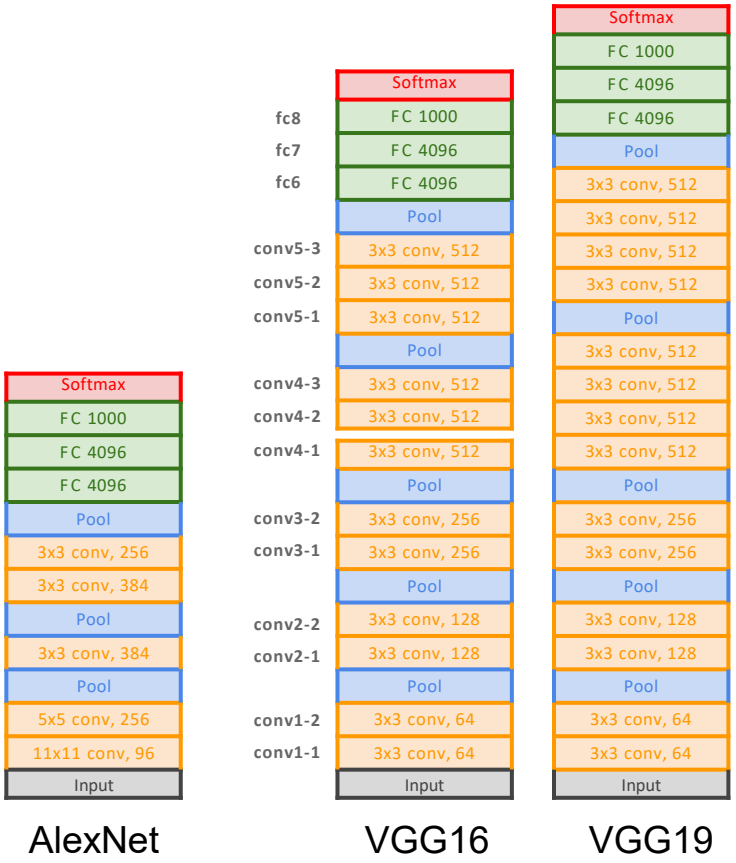
VGG16

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

VGGNet -5

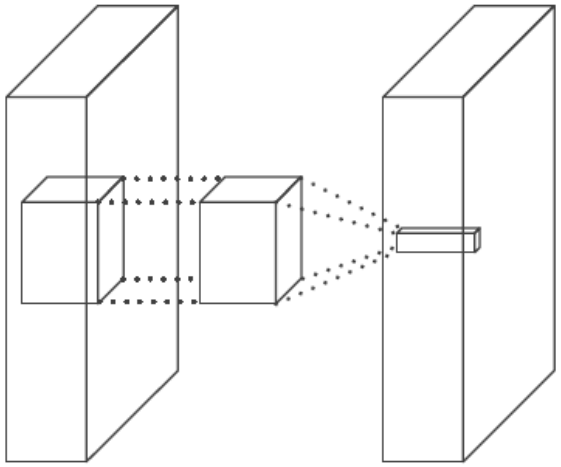
Details:

- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks

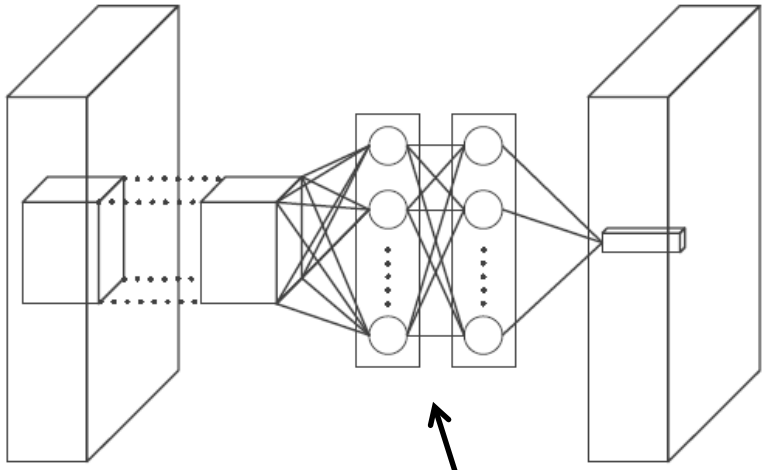


GoogLeNet -1

- GoogLeNet의 핵심 아이디어인 인셉션 모듈
  - NIN의 구조를 수정한 것
- NIN 구조
  - MLPconv층이 컨볼루션 연산을 대신함
  - MLPconv는 커널을 옮겨가면서 MLP의 전방 계산을 수행함



(a) 기존 컨볼루션층



(b) NIN의 MLPconv층

그림 4-24 기존 컨볼루션 신경망과 NIN의 비교

MLPconv층 (마이크로 네트워크)

GoogLeNet -2

- NIN이 사용하는 전역 평균 풀링
  - [그림 4-25(a)]의 VGGNet의 완전연결층
    - 1억2천2백만 개의 매개변수를 가짐 (VGGNet의 전체 매개변수의 85%) → 과잉적합 원인
  - [그림 4-25(b)]의 전역 평균 풀링
    - MLPconv가 부류 수만큼 특징 맵을 생성하면, 특징 맵 각각을 평균하여 출력 노드에 입력 → 이 방식으로 매개변수를 없앰

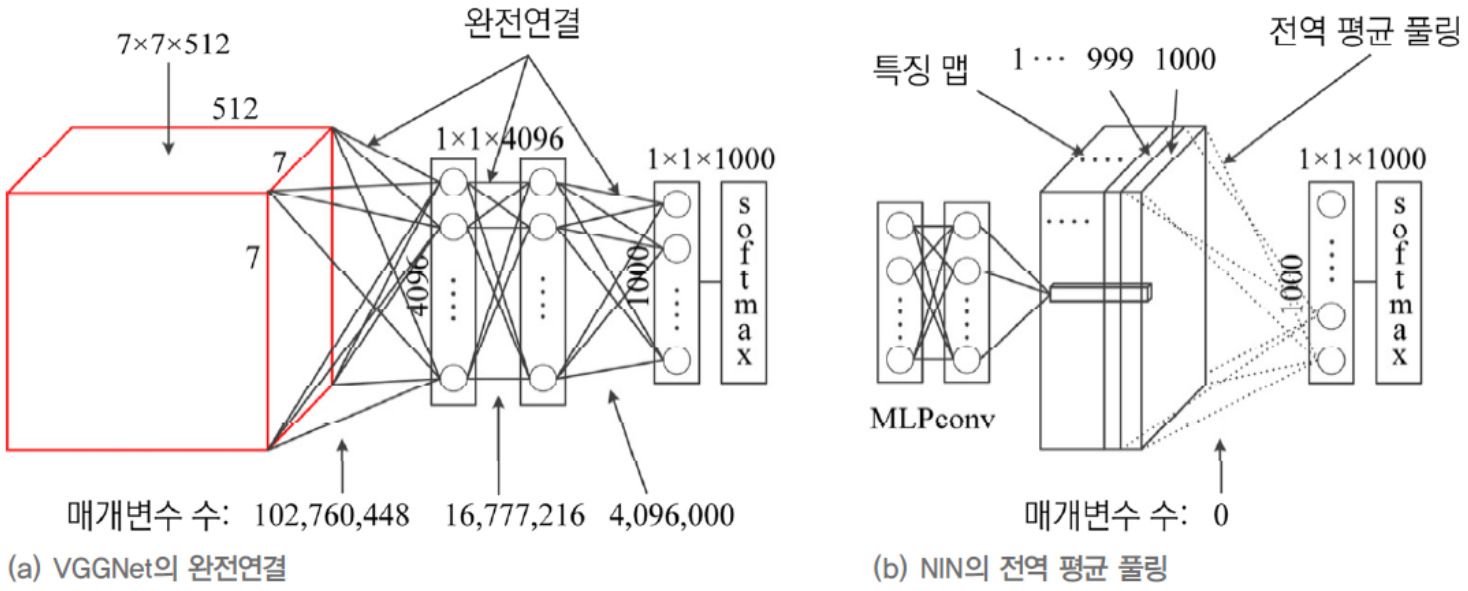
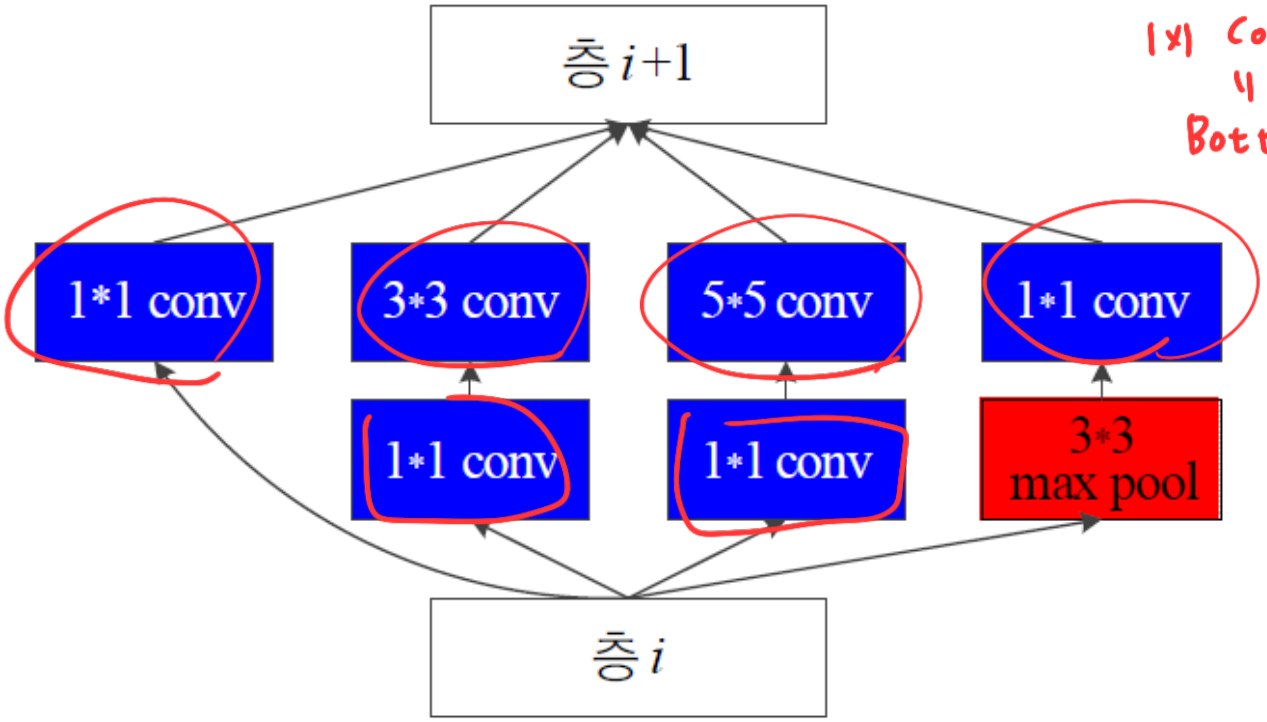


그림 4-25 완전연결과 NIN의 전역 평균 풀링의 비교

GoogLeNet -3

- GoogLeNet은 NIN 아이디어를 확장한 신경망
  - 인셉션 모듈
    - 마이크로 네트워크로 MLPconv 대신 네 종류의 컨볼루션 연산 사용
    - 1*1 컨볼루션을 사용하여 차원 축소

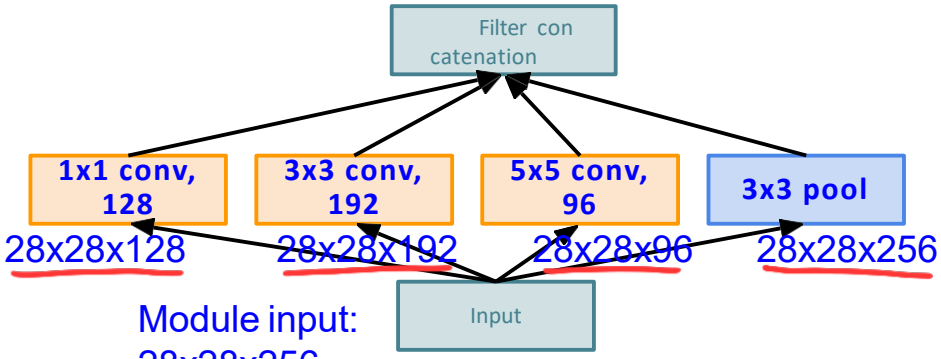


1x1 Conv
↓
Bottle Neck Layer
↓
차원축소

그림 4-26 GoogLeNet의 인셉션 모듈

GoogLeNet -4

28x28x(128+192+96+256) = 28x28x672



Module input: 28x28x256

channel 개수

Naive Inception m

계산복잡도↑

Q: What is the problem w complexity]

Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5 x5)
- Pooling operation (3x3)

Concatenate all filter outputs epth-wise

Q1: What is the output size of the 1x1 conv, with 128 filters?

Q2: What are the output sizes of all different filter operations?

Q3:What is output size after filter concatenation?

Pooling layer also preserves feature dept h, which means total depth after concate nation can only grow at every layer!

Conv Ops:

Input = filter개수 = filter 크기

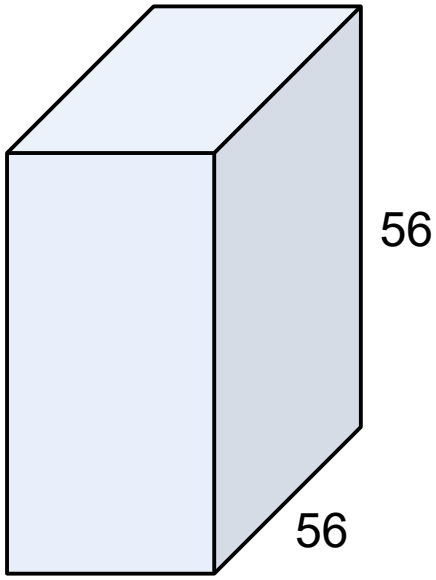
[1x1 conv, 128] 28x28x128x1x1x256 → Input channel

[3x3 conv, 192] 28x28x192x3x3x256

[5x5 conv, 96] 28x28x96x5x5x256

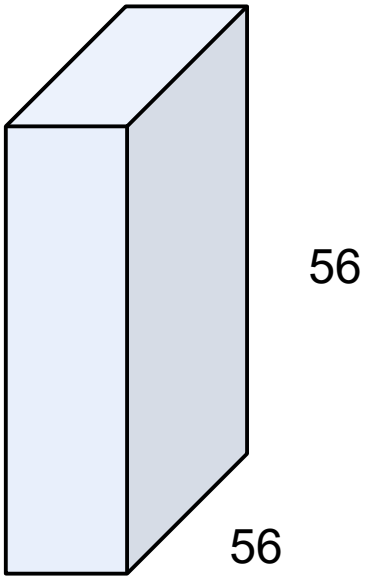
Total: 854M ops

GoogLeNet -5



1x1 CONV
with 32 filters

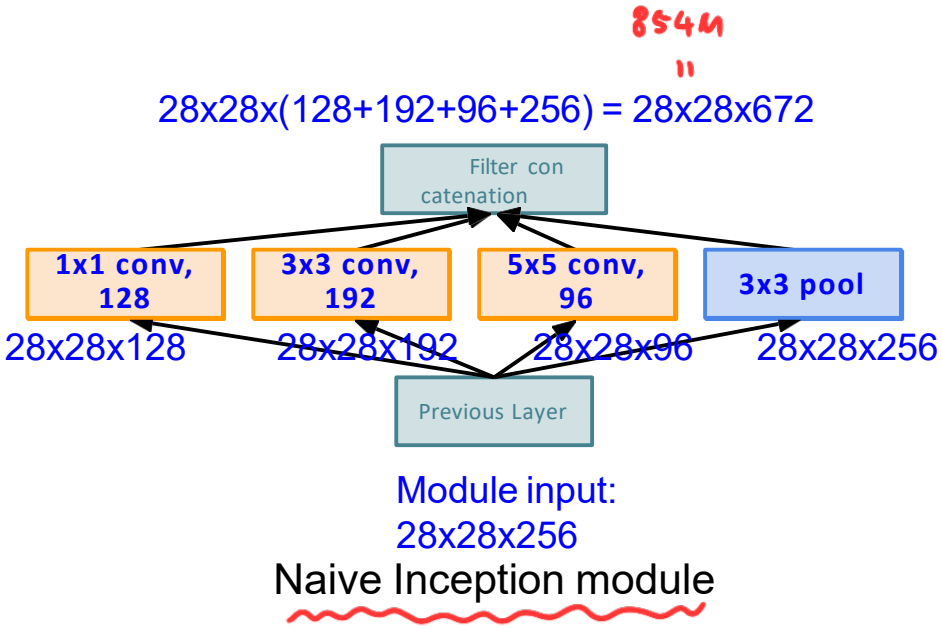
(each filter has size 1x 1x64, and performs a 6 4-dimensional dot prod uct)

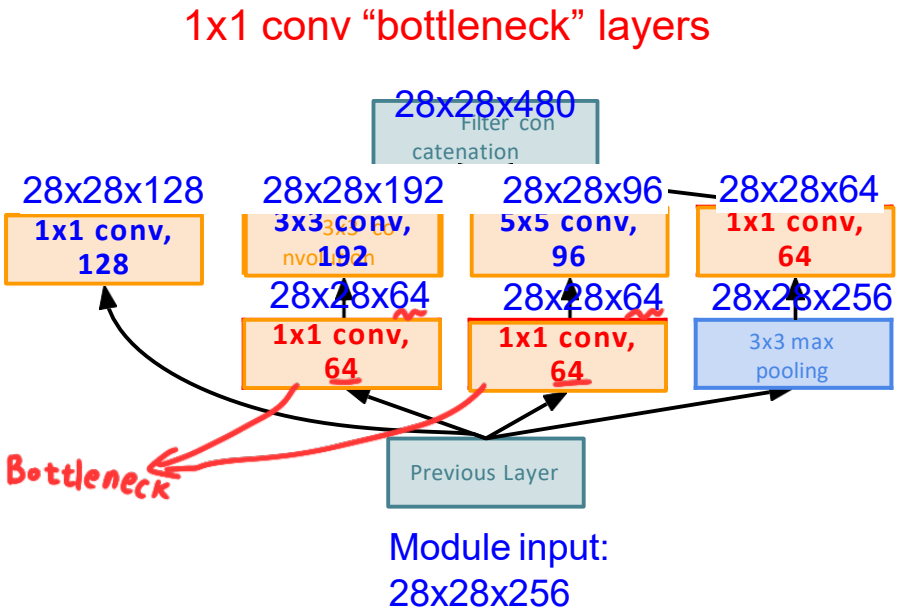


preserves spatial dimensions, reduces depth!

Projects depth to lower dimension (combination of feature maps)

GoogLeNet -6





Conv Ops:

[1x1 conv, 128] 28x28x128x1x1x256

[3x3 conv, 192] 28x28x192x3x3x256

[5x5 conv, 96] 28x28x96x5x5x256

Total: 854M ops

input channel

Conv Ops:

[1x1 conv, 128] 28x28x128x1x1x256

[1x1 conv, 64] 28x28x64x1x1x256

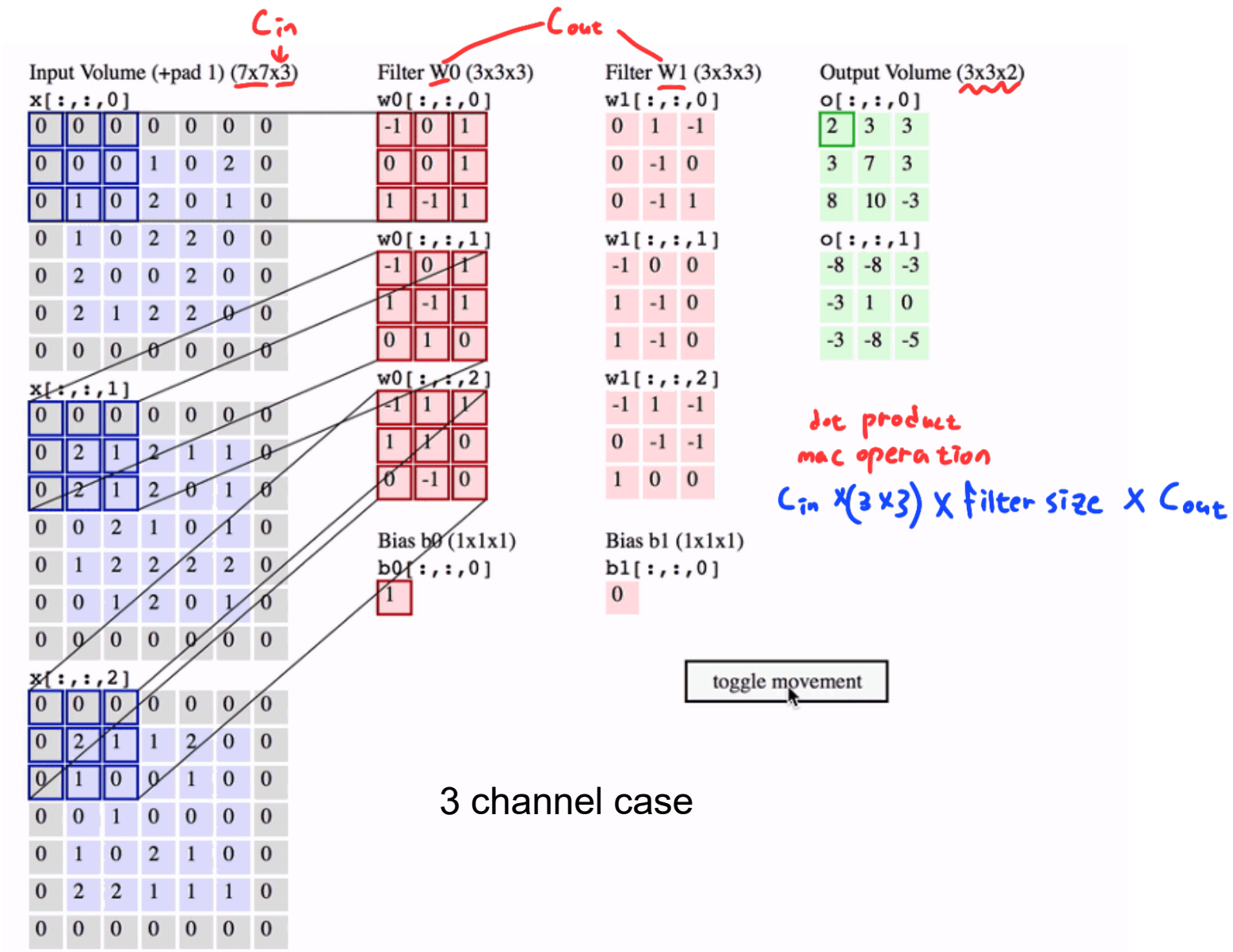
[3x3 conv, 192] 28x28x192x3x3x64

[1x1 conv, 64] 28x28x64x1x1x256

[5x5 conv, 96] 28x28x96x5x5x64

[1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops



3 channel case

GoogLeNet -7

- 인셉션 모듈을 9개 결합한 GoogLeNet ([그림 4-27])
  - 매개변수가 있는 층 22개, 없는 층 5개로 총 27개 층
  - 완전열결층은 1개에 불과
    - 1백만 개의 매개변수를 가지며, VGGNet의 완전연결에 비하면 1%에 불과

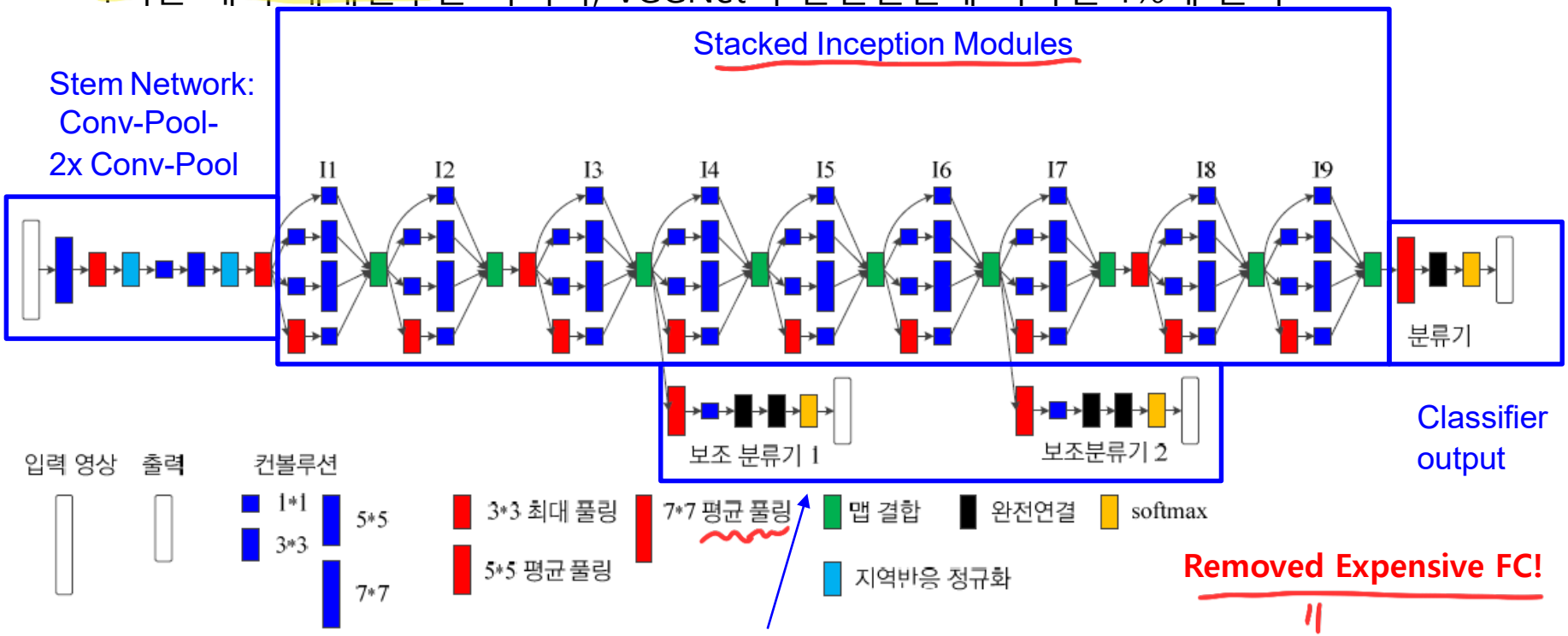
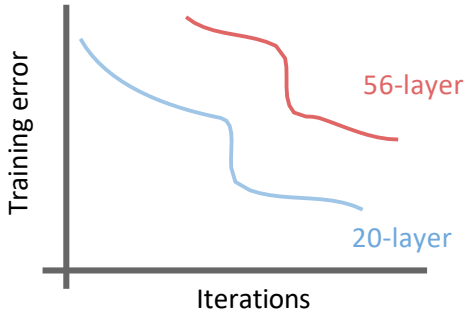
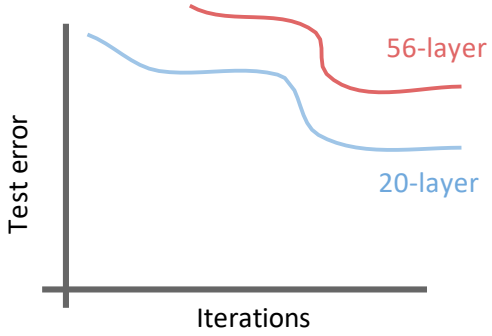


그림 4-27 GoogLeNet의 구조

ResNet -1

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?





⇒ degradation

56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it’s not caused by overfitting!

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

The deeper model should be able to perform at least as well as the shallower model.
A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

ResNet -2

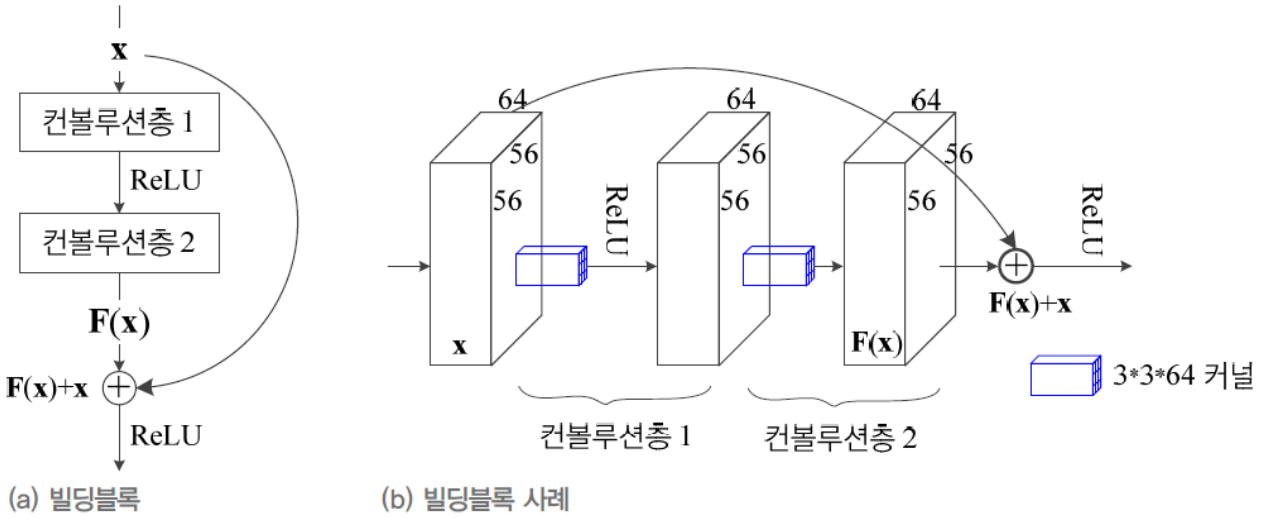
- ResNet
  - 잔류 학습이라는 아이디어를 이용하여 성능 저하를 피하면서 층 수를 대폭 늘림(최대 1202층까지)
  - 원래 컨볼루션 신경망

$$\mathbf{F(x) = \tau(x \circledast w_1) \circledast w_2}$$

$$\mathbf{y = \tau(F(x))}$$

  - 잔류 학습은 지름길 연결된 $\mathbf{x}$를 더한 $\mathbf{F(x) + x}$에 τ를 적용. $\mathbf{F(x)}$는 잔류

$$\mathbf{y = \tau(F(x) + x)}$$



(a) 빌딩블록 (b) 빌딩블록 사례

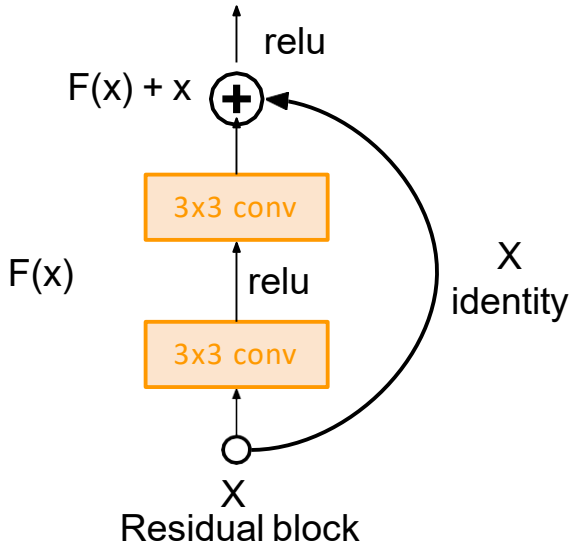
그림 4-28 잔류 학습의 구조와 동작

ResNet -3

- 지름길 연결을 두는 이유는?
  - 그레이디언트 소멸 문제 해결 ⇒ gradient=0 이 되는 문제
  - 식 (4.14)의 그레이디언트 식에서 $\frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathbf{F}(\mathbf{x}_i)$ 이 -1이 될 가능성이 거의 없음

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathbf{F}(\mathbf{x}_i)\right) \tag{4.14}$$

- Residual 블록 구조



Every residual block has two 3x3 conv layers

Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)

↳ 주기적으로 filter의 개수를 2배씩 증가시켜줌

⇒ 2배로 증가할 때마다 Stride =2 로 down sampling

⇒ Dimension $\frac{1}{2}$배

ResNet -4

- 34층짜리 ResNet 예시

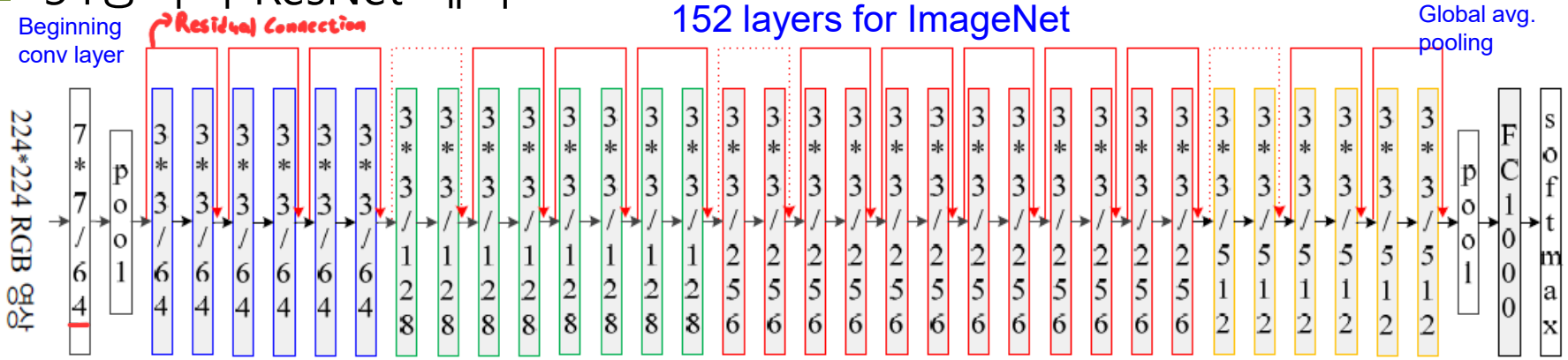


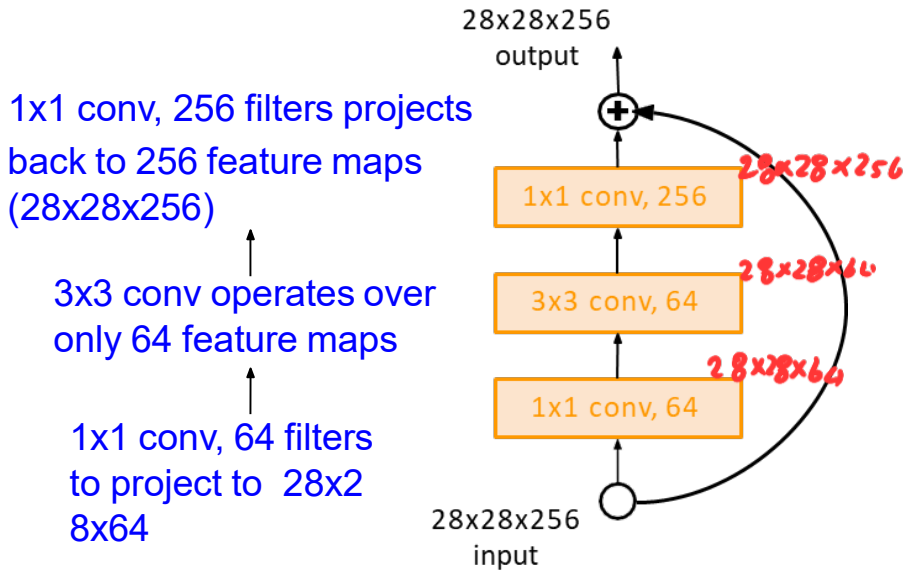
그림 4-29 ResNet 예제(34층)

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

No FC layers but FC 1000

- VGGNet과 같은 점
  - 3*3 커널 사용
- VGGNet과 다른 점
  - 잔류 학습 사용
  - 전역 평균 풀링 사용(FC 층 제거)
  - 배치 정규화 적용(드롭아웃 적용 불필요)

  → 배치 정규화는 5.2.6절



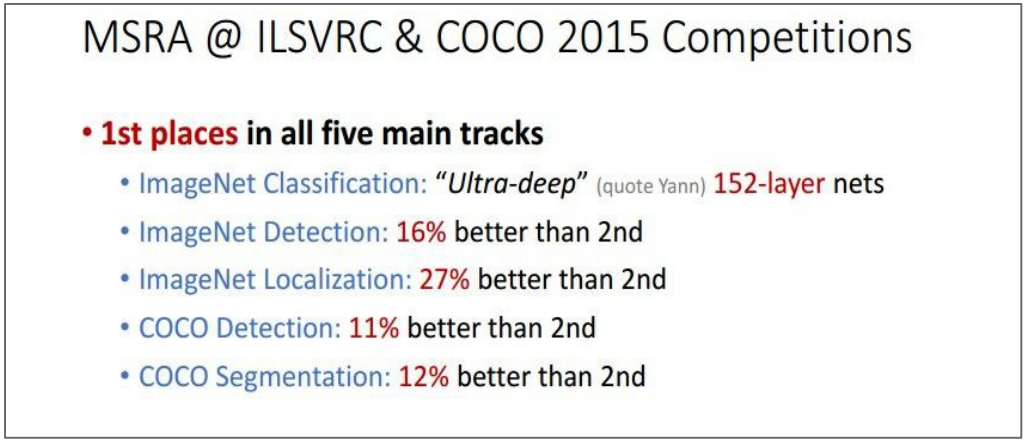
ResNet -5

Training ResNet in practice:

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

Experimental Results

- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lo wing training error as expected
- Swept 1st place in all ILSVRC an d COCO 2015 competitions



ILSVRC 2015 classification winner (3.6% top 5 error) -- b etter than "human performance"! (Russakovsky 2014)

ResNet -6

- ILSVRC 대회
  - 분류 문제는 성능 포화 (사람 성능에 필적함)
  - 물체 검출 문제에 집중함

그림 4-31 ILSVRC 물체 검출 문제

ResNet -7

- ILSVRC 대회 성적
  - 2012년 AlexNet의 15.3% 오류율은 당시로서 경이로운 성능
  - 2015년에 ResNet은 3.5% 오류율 달성

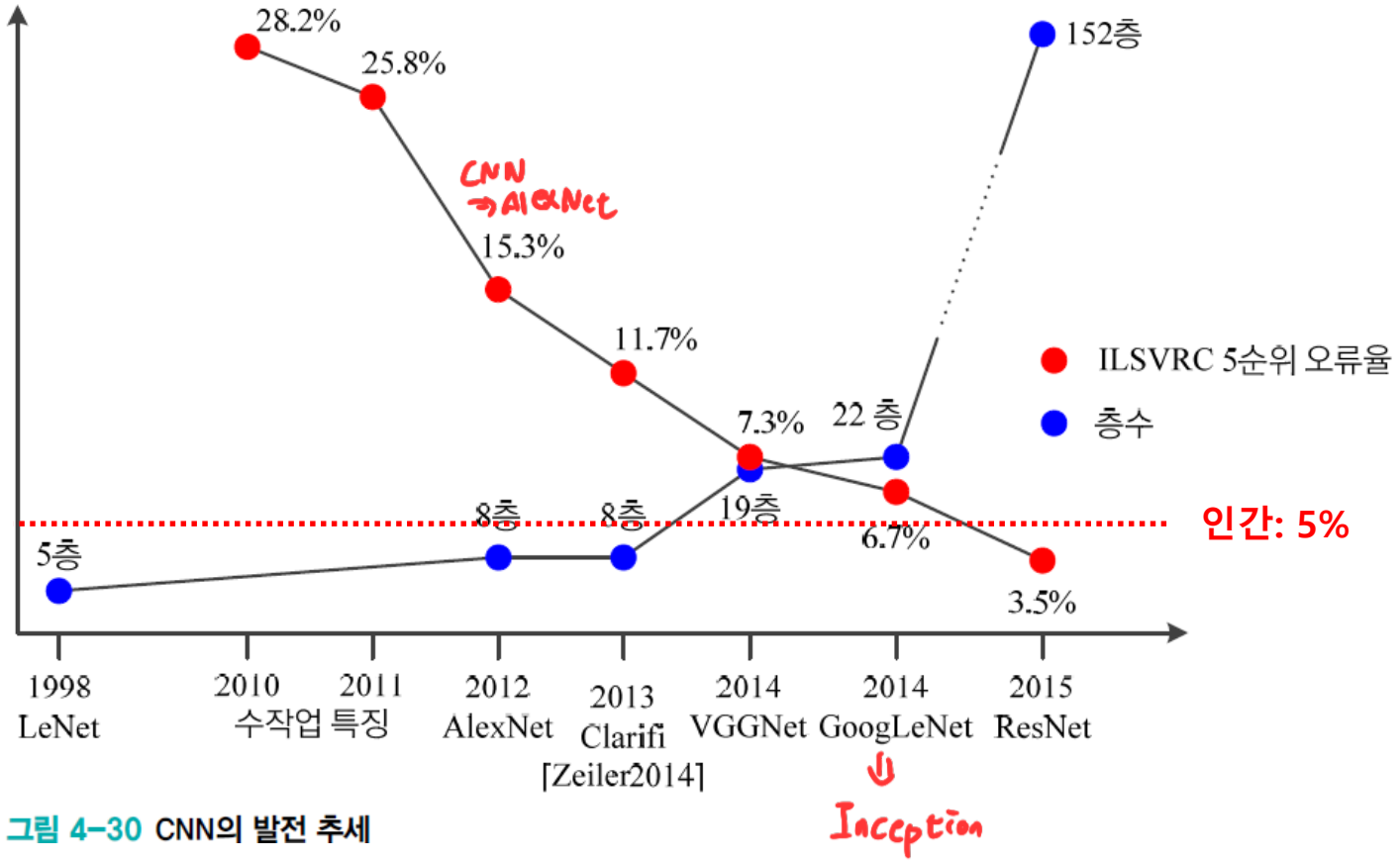
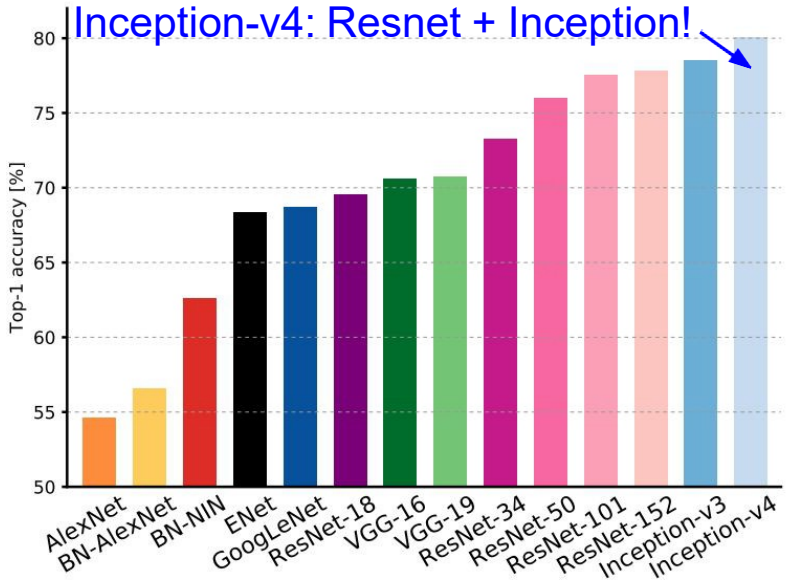
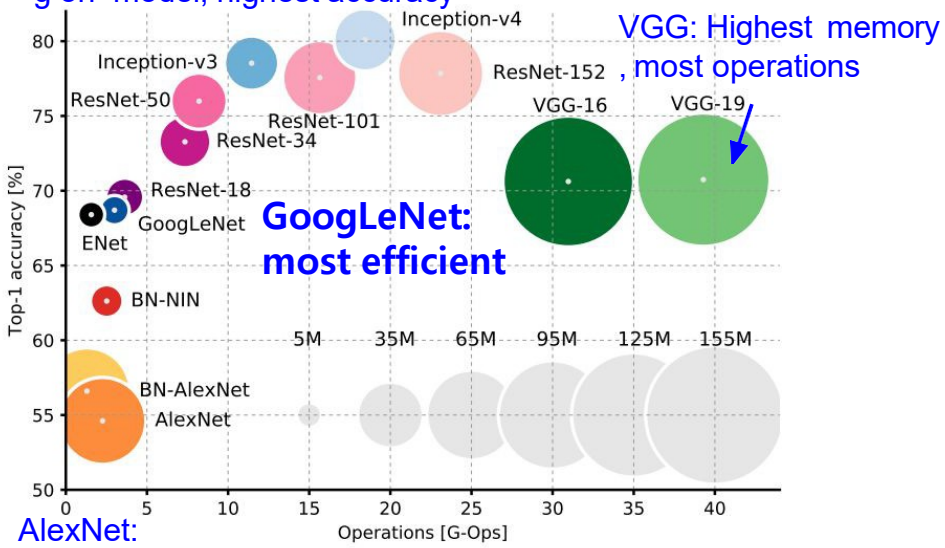


그림 4-30 CNN의 발전 추세

CNN 비교



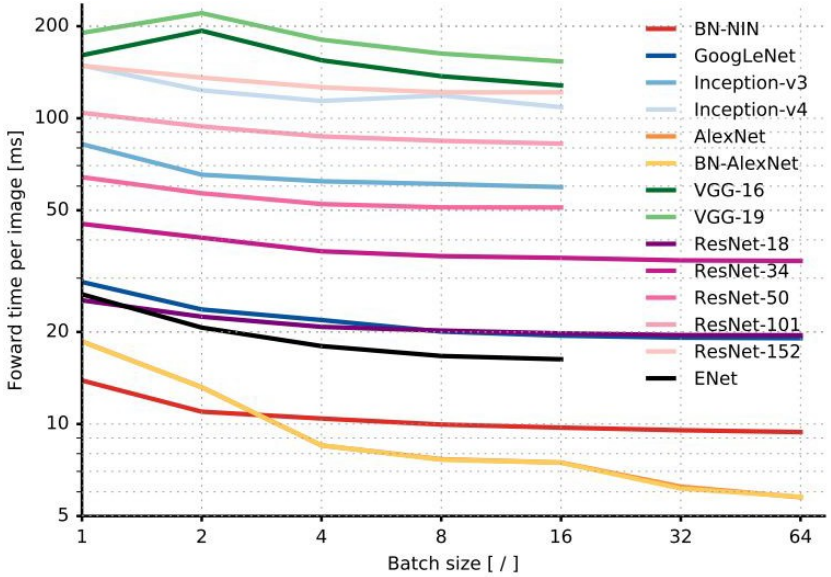
ResNet:
Moderate efficiency dependin
g on model, highest accuracy

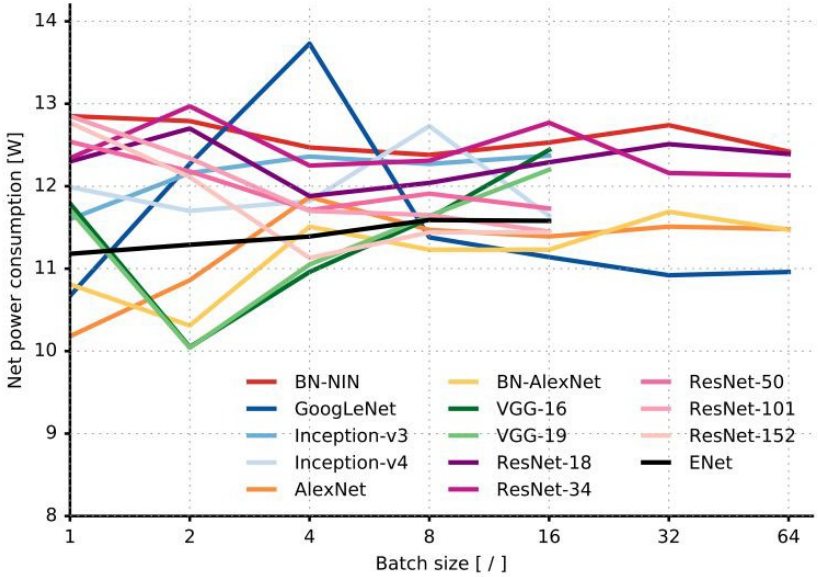


VGG: Highest memory
, most operations

**GoogLeNet:
most efficient**

AlexNet:
Smaller compute, still memory heavy, lower accuracy





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