Convolutional Neural Network
-Advanced-

최건호

INDEX

ImageNet Challenge

성능을 측정하려면 공통된 데이터와
성능 측정 메트릭이 필요

ImageNet Challenge

성능을 측정하려면 공통된 데이터와
성능 측정 메트릭이 필요

필요한 건 알지만 수많은 데이터를
모으고 정제하는 건 쉬운 일이 아님

ImageNet Challenge

성능을 측정하려면 공통된 데이터와
성능 측정 메트릭이 필요

필요한 건 알지만 수많은 데이터를
모으고 정제하는 건 쉬운 일이 아님

Prof. Fei-Fei Li
B.S. in Physics from Princeton University.
Ph.D. degree from California Institute of Technology
Associate Professor, Stanford University
Google Cloud Chief Scientist AI/ML

ImageNet Challenge

https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures

ImageNet Challenge

IMAGENET

1.2 Million Images(1,200,000), 1000 Categories

ImageNet Challenge

현재 14,197,122 Images

IMAGENET

1.2 Million Images(1,200,000), 1000 Categories

ImageNet Challenge

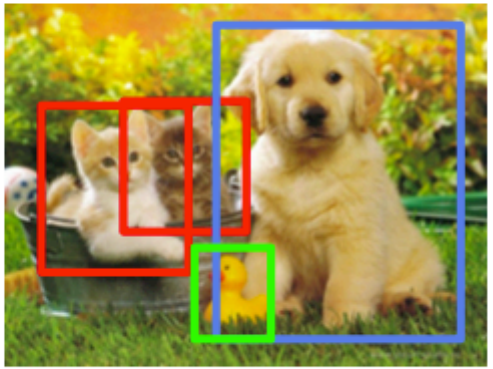
Classification

CAT

Classification + Localization

CAT

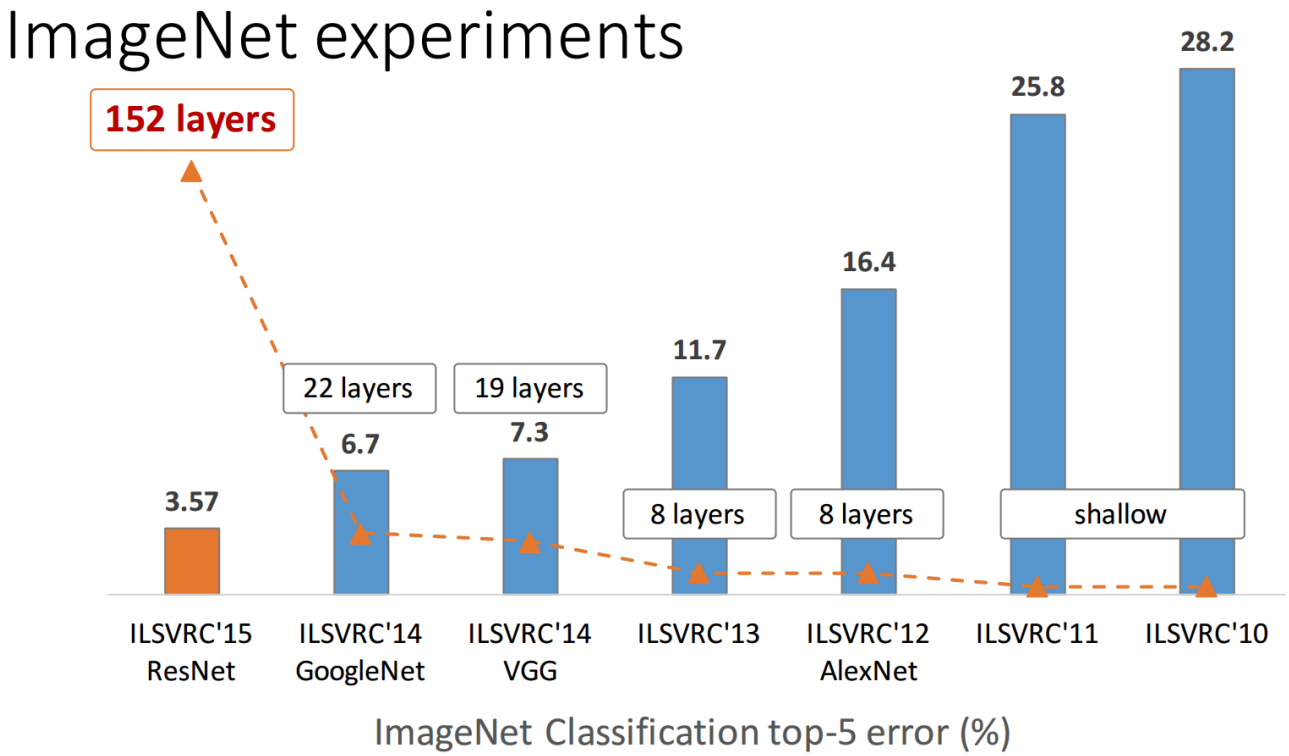
Object Detection



CAT, DOG, DUCK

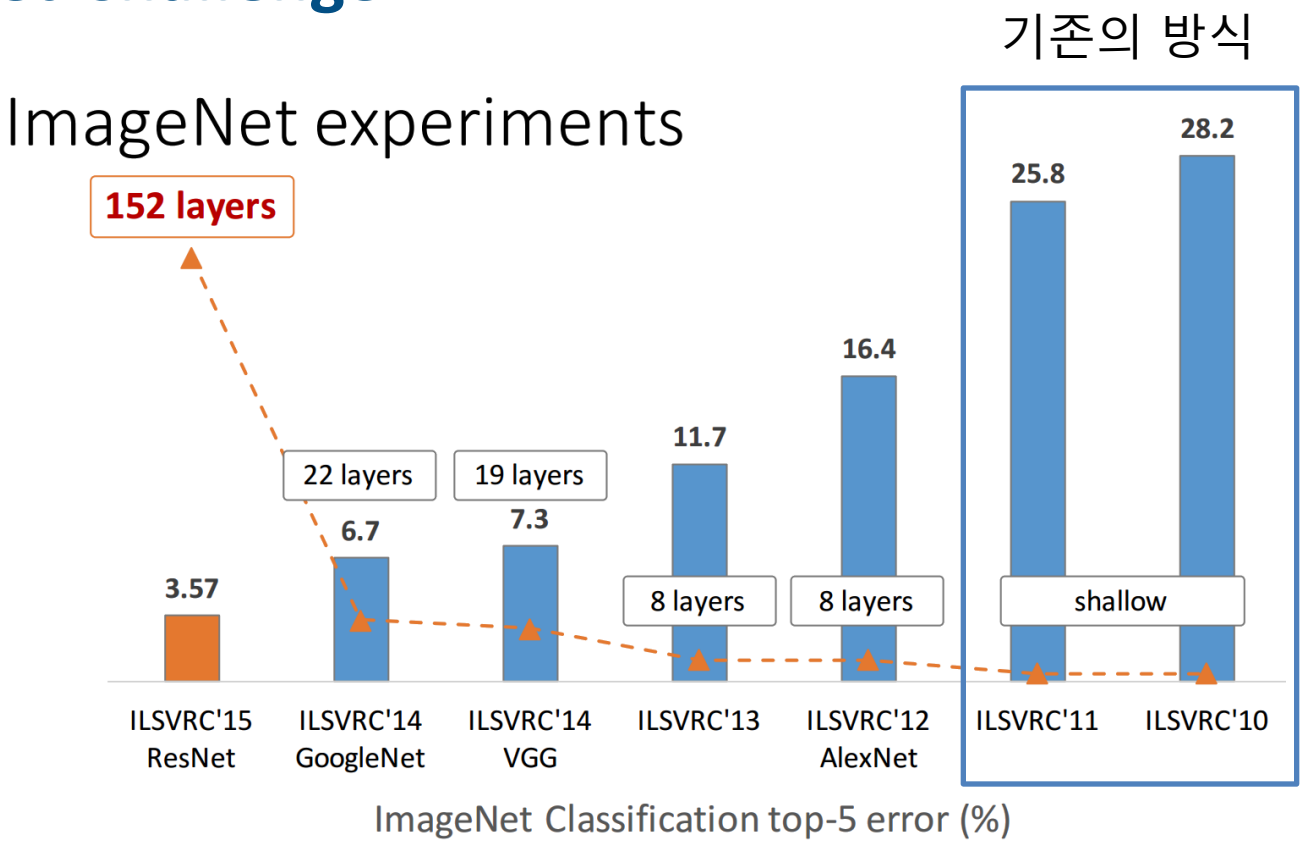
(출처:CS231n_lecture_slide)

ImageNet Challenge



(출처: Kaiming He "Deep residual learning for image recognition." ICML. 2016.)

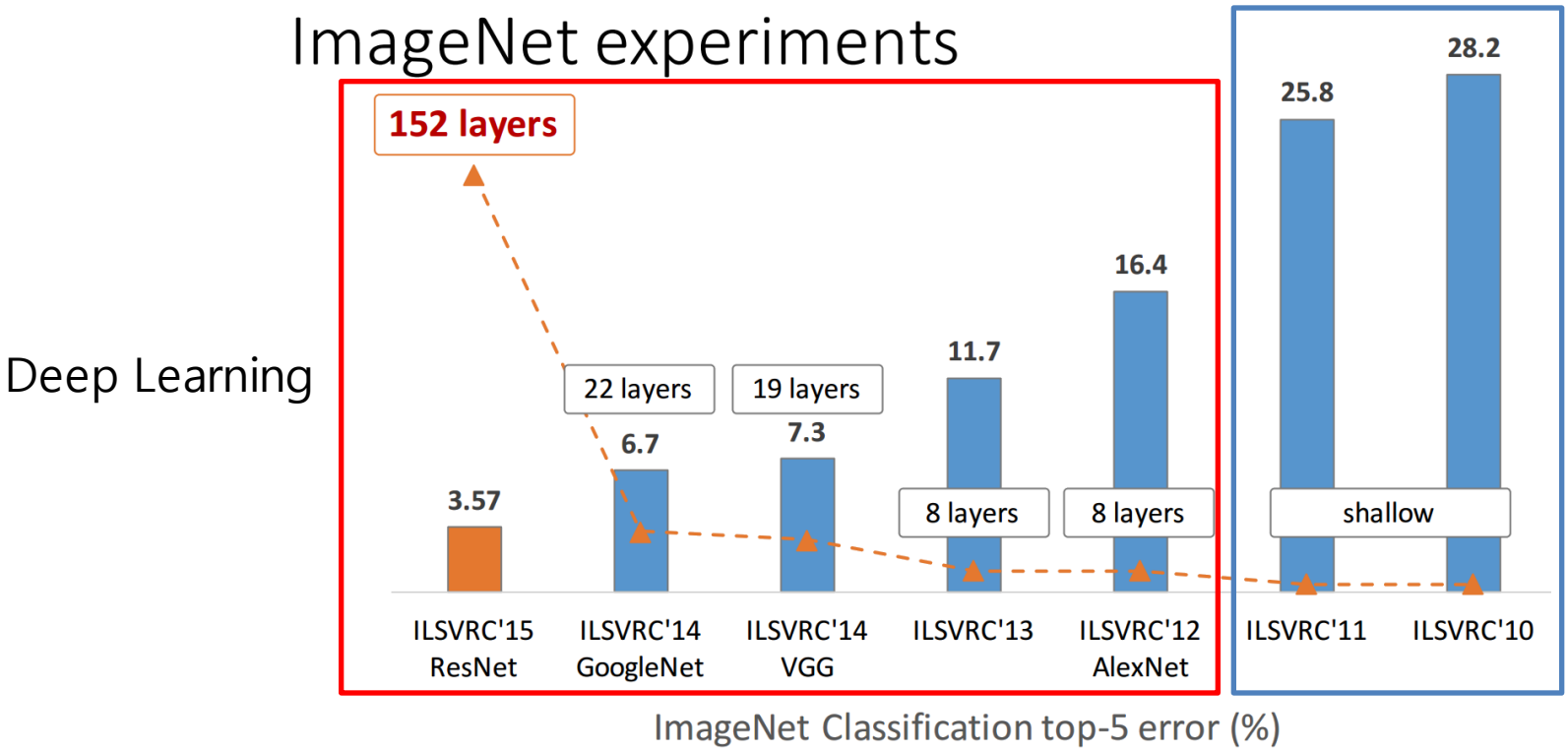
ImageNet Challenge



(출처: Kaiming He "Deep residual learning for image recognition." ICML. 2016.)

ImageNet Challenge

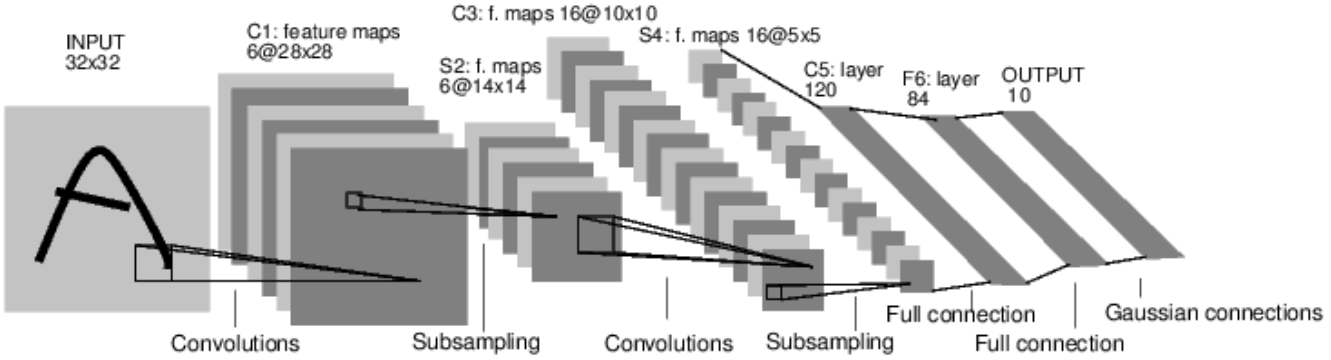
기존의 방식



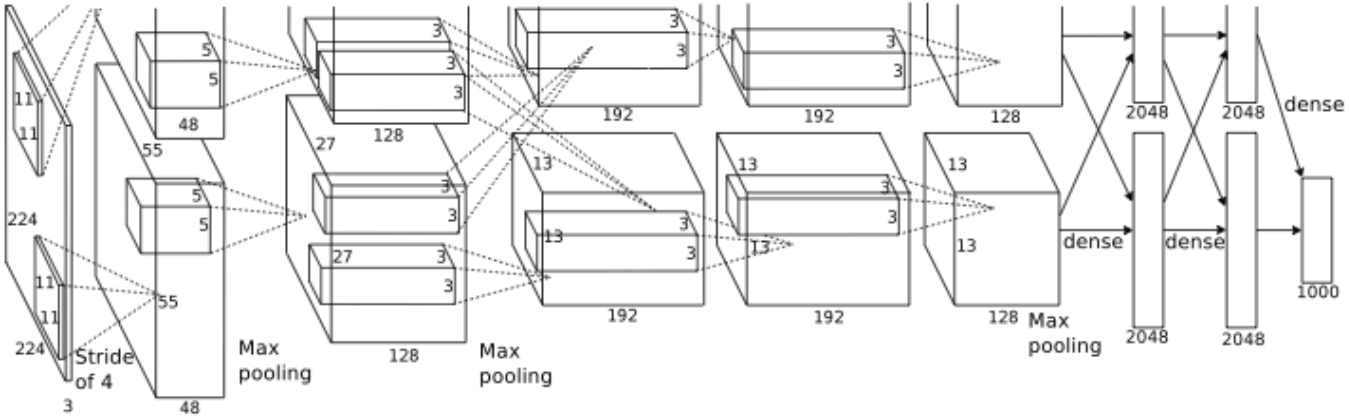
(출처: Kaiming He "Deep residual learning for image recognition." ICML. 2016.)

ImageNet Challenge

LeNet
(1998)

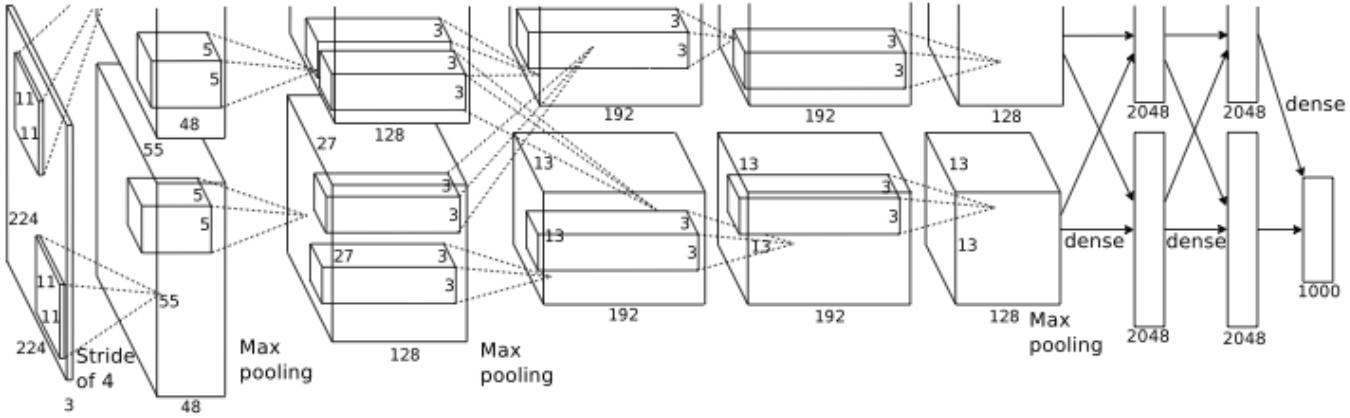


AlexNet
(2012)



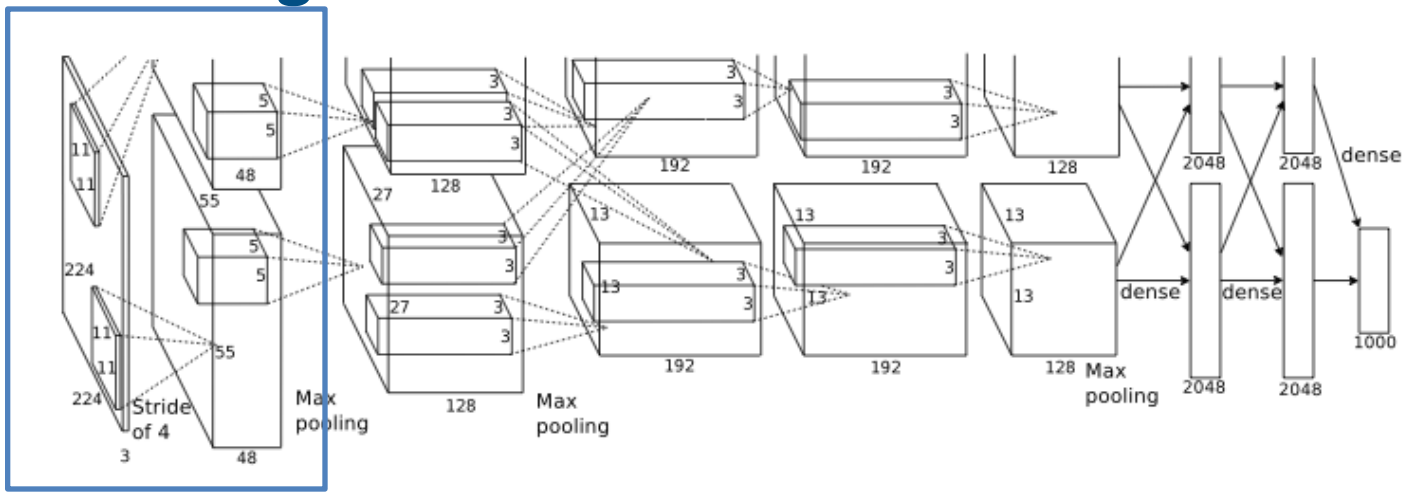
ImageNet Challenge

AlexNet
(2012)



ImageNet Challenge

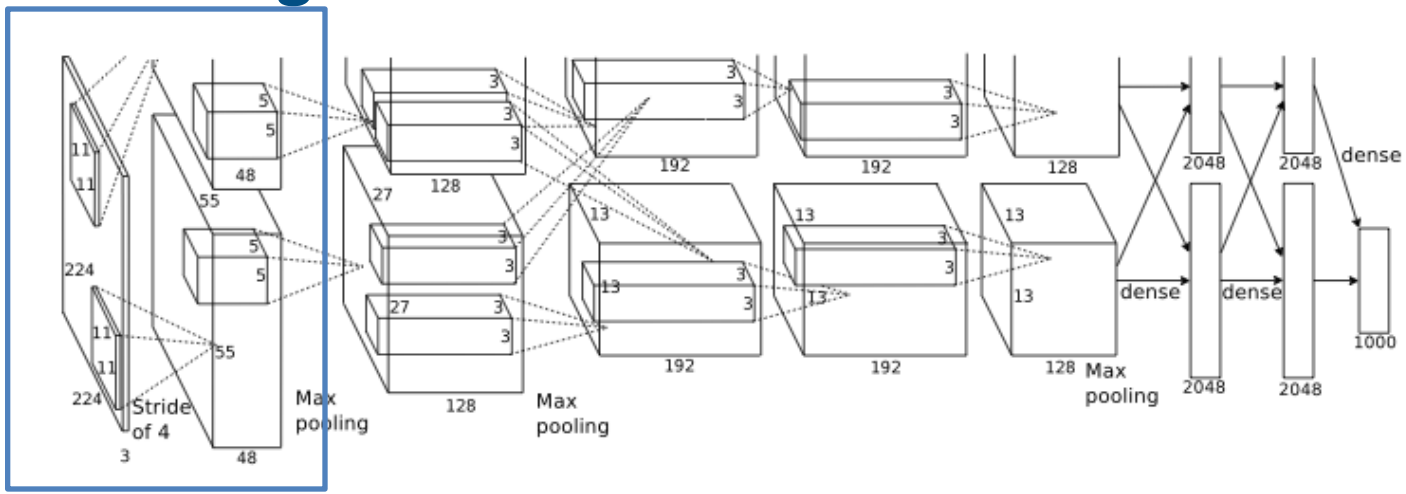
AlexNet
(2012)



이 부분에 얼만큼의 파라미터가 생성될까?

ImageNet Challenge

AlexNet
(2012)

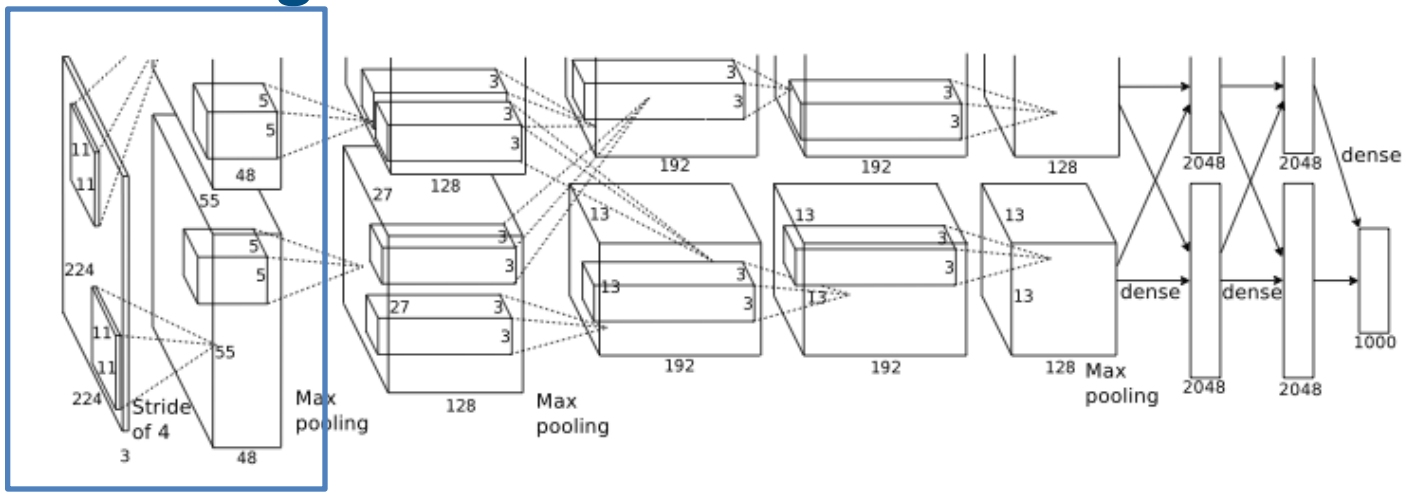


이 부분에 얼만큼의 파라미터가 생성될까?

Input: 224x224x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{224-11}{4}+1]$

ImageNet Challenge

AlexNet
(2012)

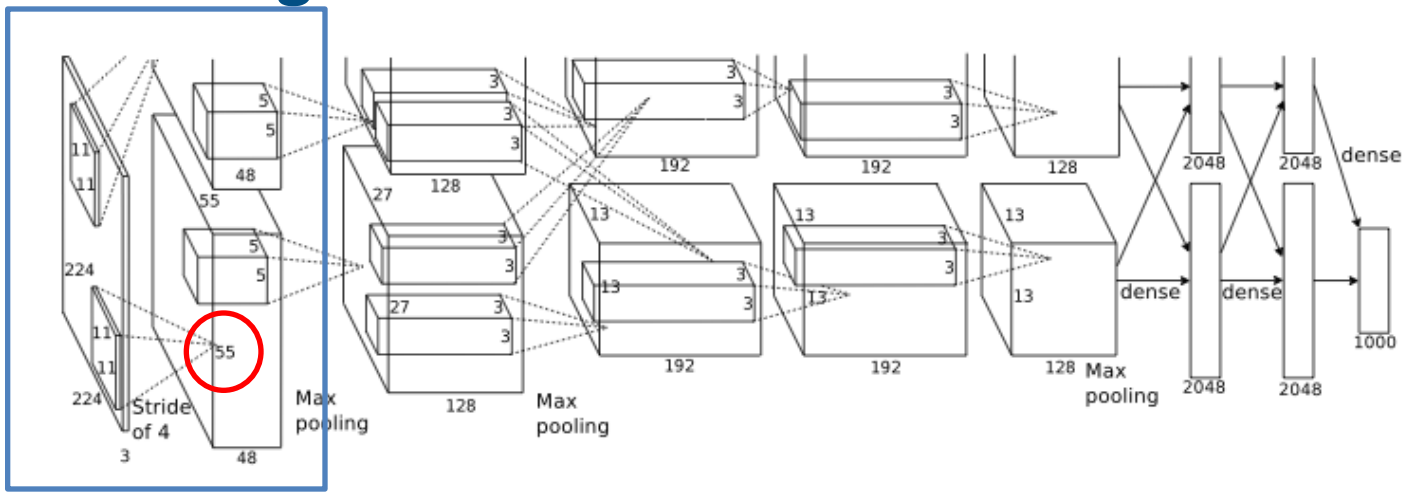


이 부분에 얼만큼의 파라미터가 생성될까?

Input: 224x224x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{224-11}{4}+1]=54$

ImageNet Challenge

AlexNet
(2012)

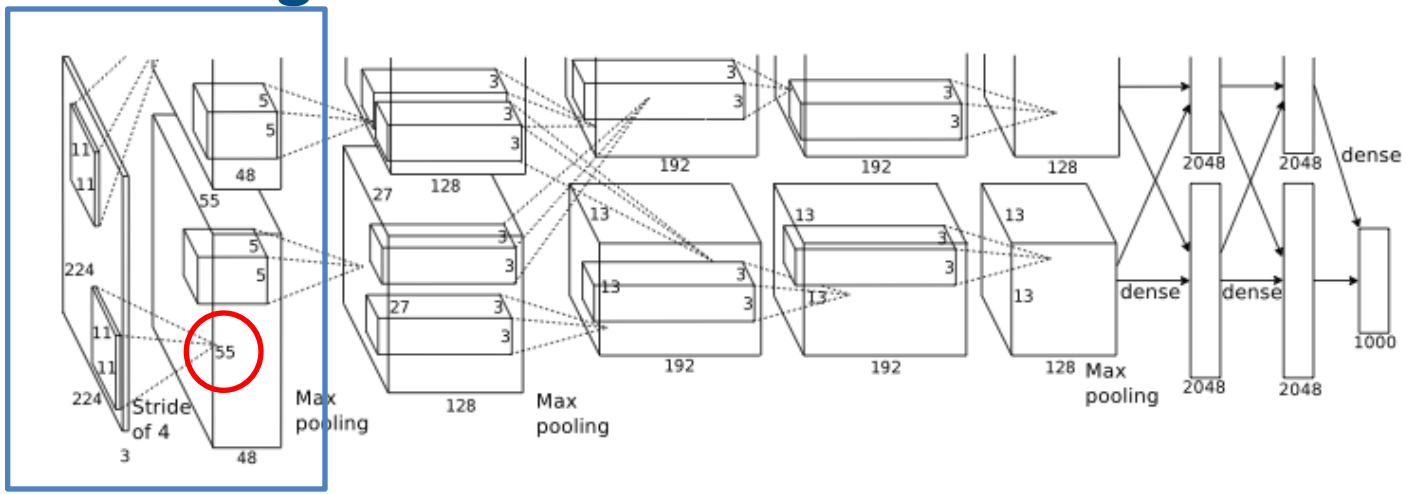


이 부분에 얼만큼의 파라미터가 생성될까?

Input: 224x224x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{224-11}{4}+1]$=54(???)

ImageNet Challenge

AlexNet
(2012)



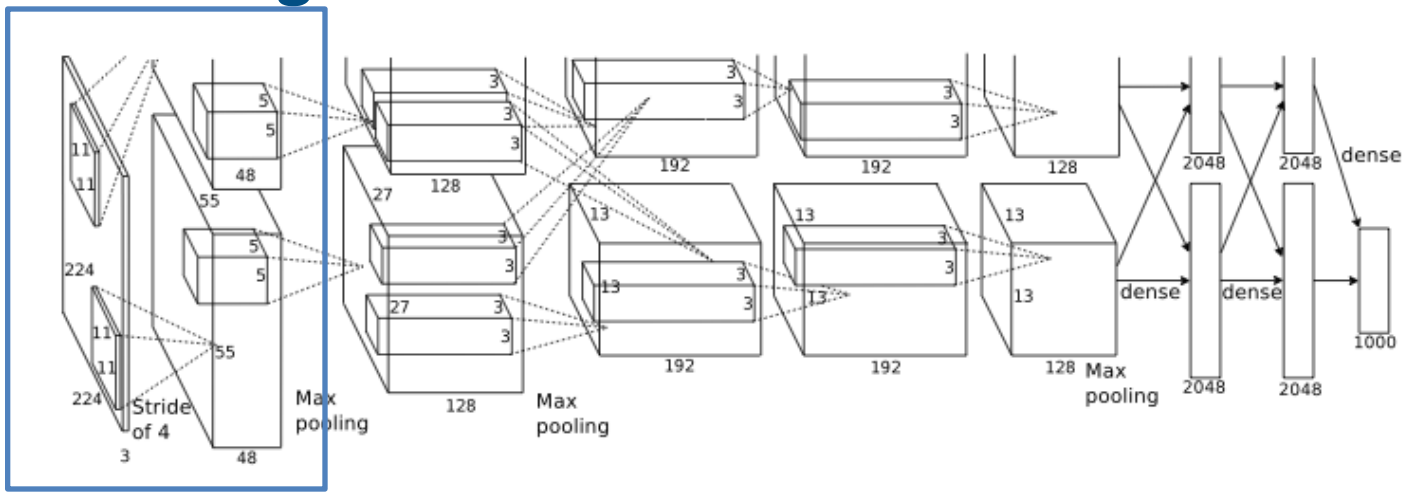
이 부분에 얼만큼의 파라미터가 생성될까?

Input: 224x224x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{224-11}{4}+1]$=54(???)

논문이 잘못했네..

ImageNet Challenge

AlexNet
(2012)

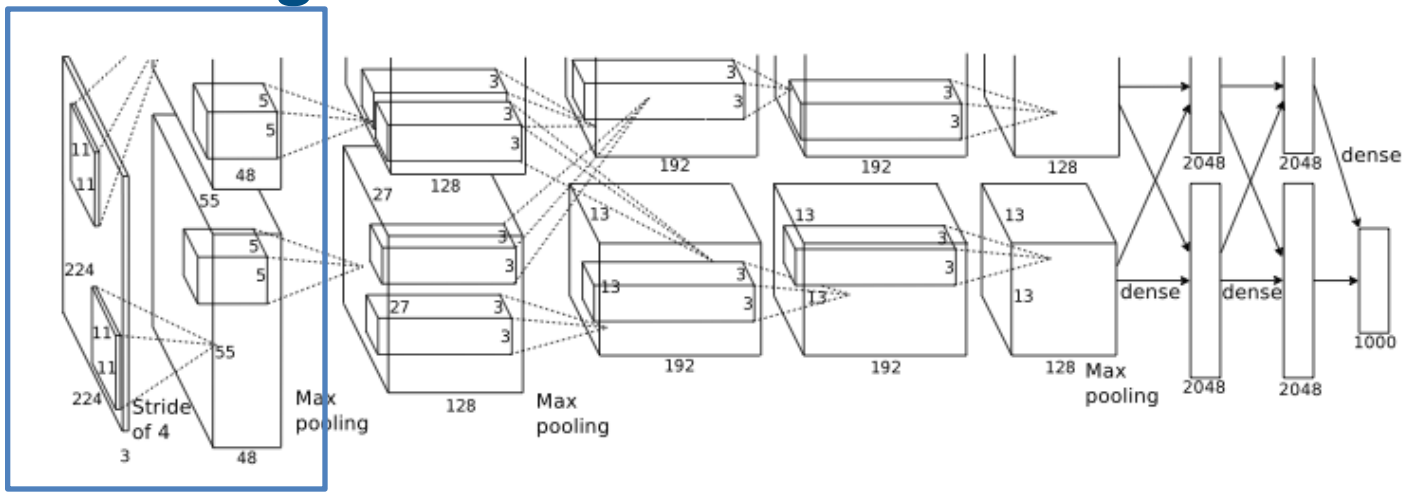


이 부분에 얼만큼의 파라미터가 생성될까?

Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

ImageNet Challenge

AlexNet
(2012)



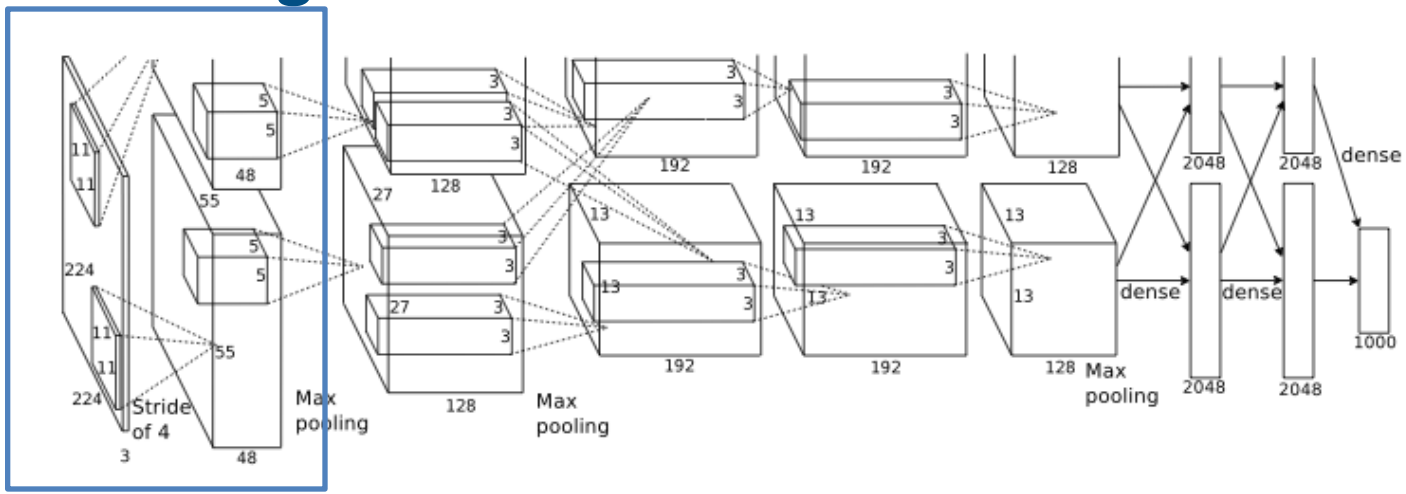
이 부분에 얼만큼의 파라미터가 생성될까?

필터 하나당 11x11x3+1=364

Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

ImageNet Challenge

AlexNet
(2012)



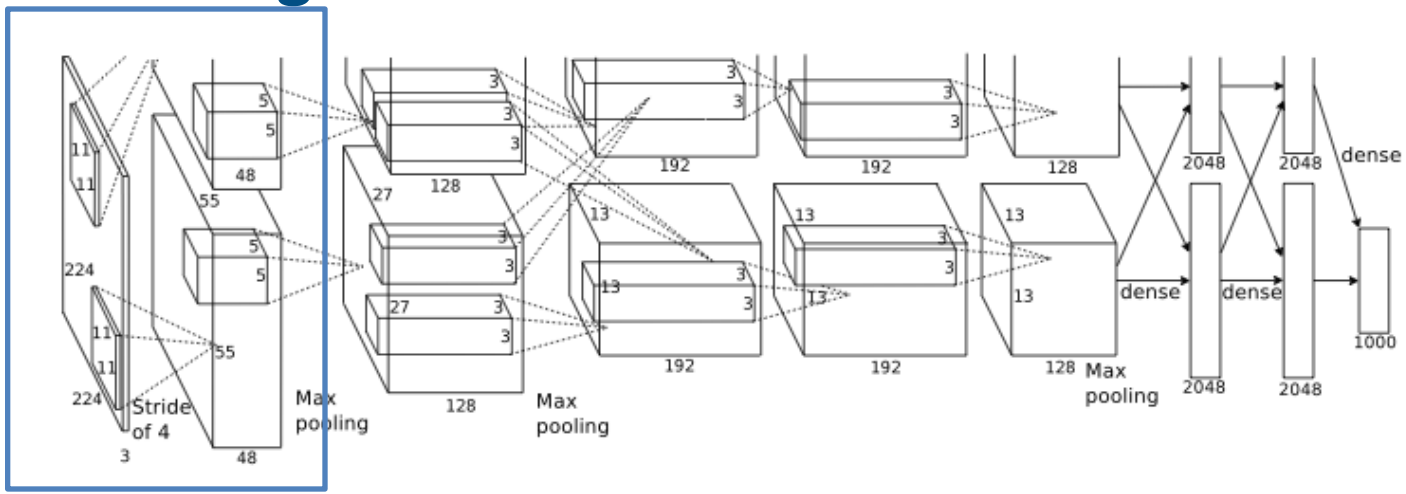
이 부분에 얼만큼의 파라미터가 생성될까?

Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

필터 하나당 11x11x3+1=364
필터가 96개
결과적으로 364x96 = 34,944

ImageNet Challenge

AlexNet
(2012)



이 부분에 얼만큼의 파라미터가 생성될까?

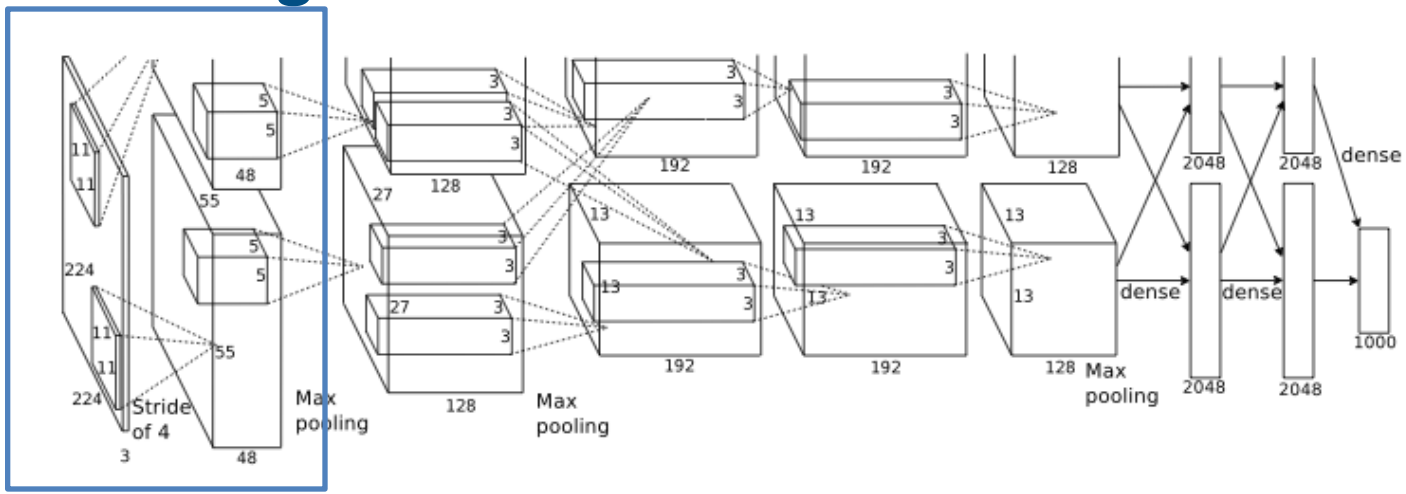
Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

필터 하나당 11x11x3+1=364
필터가 96개
결과적으로 364x96 = 34,944

output은 55x55x96 = 290,400

ImageNet Challenge

AlexNet
(2012)

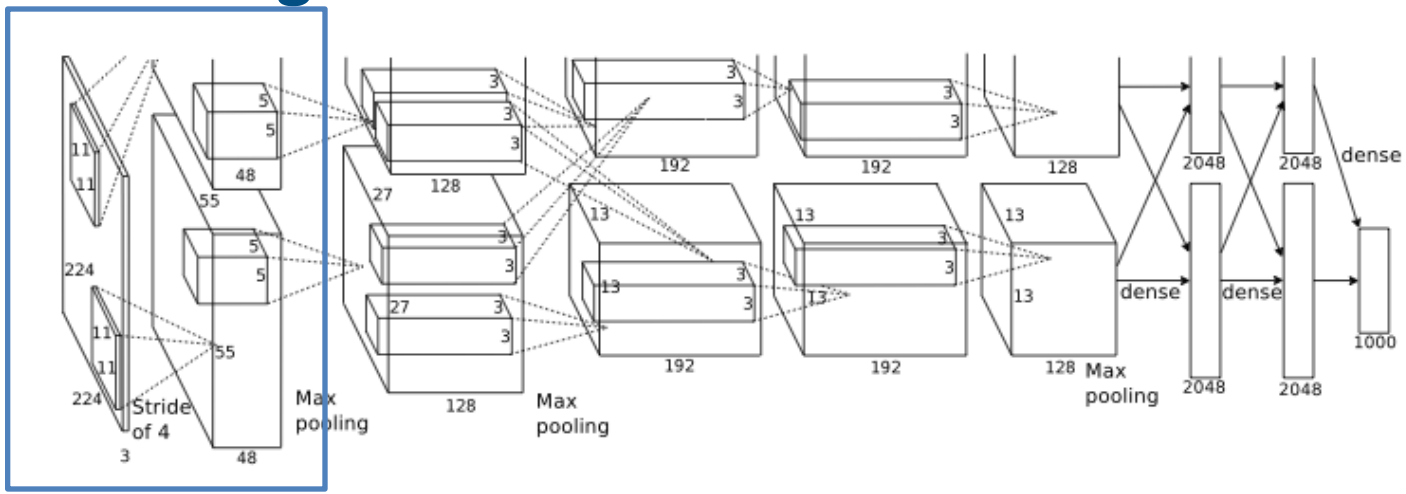


이 부분에 얼만큼의 파라미터가 생성될까? 34,944 + 290,400 = 325,344

Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

ImageNet Challenge

AlexNet
(2012)



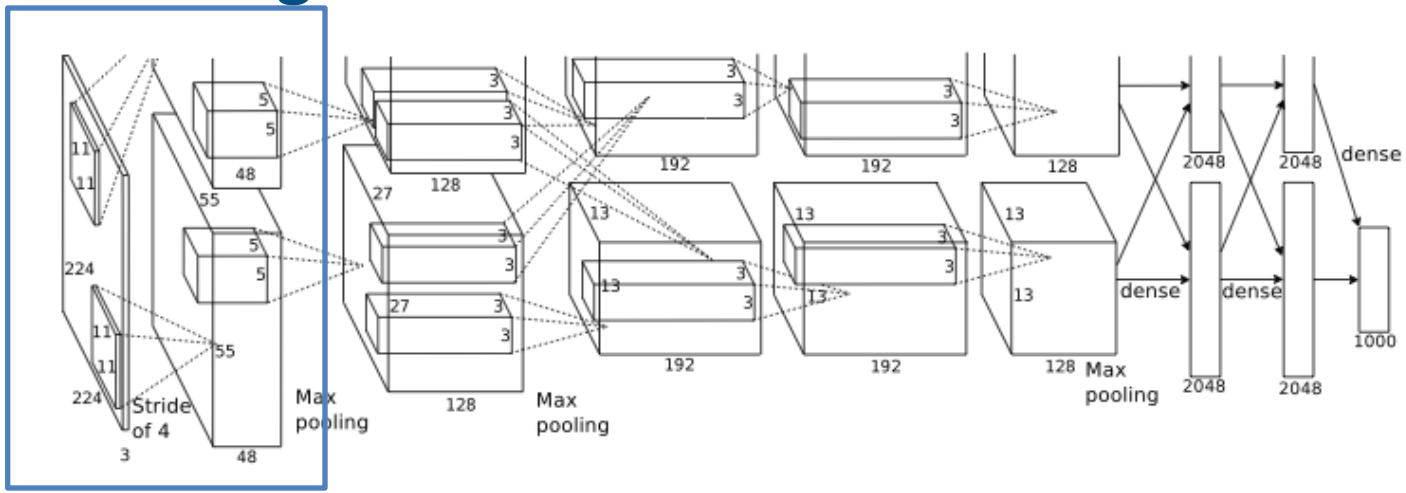
이 부분에 얼만큼의 파라미터가 생성될까?

34,944 + 290,400 = 325,344
기본형이 float32 = 4 byte

Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

ImageNet Challenge

AlexNet
(2012)



이 부분에 얼만큼의 파라미터가 생성될까?

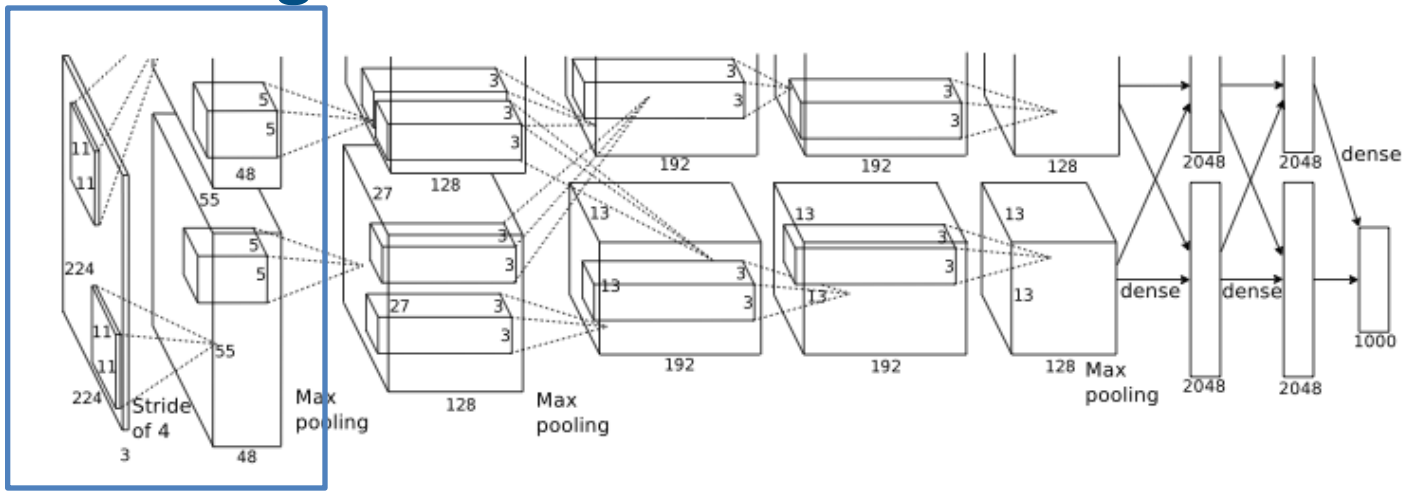
Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

34,944 + 290,400 = 325,344
기본형이 float32 = 4 byte

즉 325,344 x 4=1,301,376 byte

ImageNet Challenge

AlexNet
(2012)



이 부분에 얼만큼의 파라미터가 생성될까?

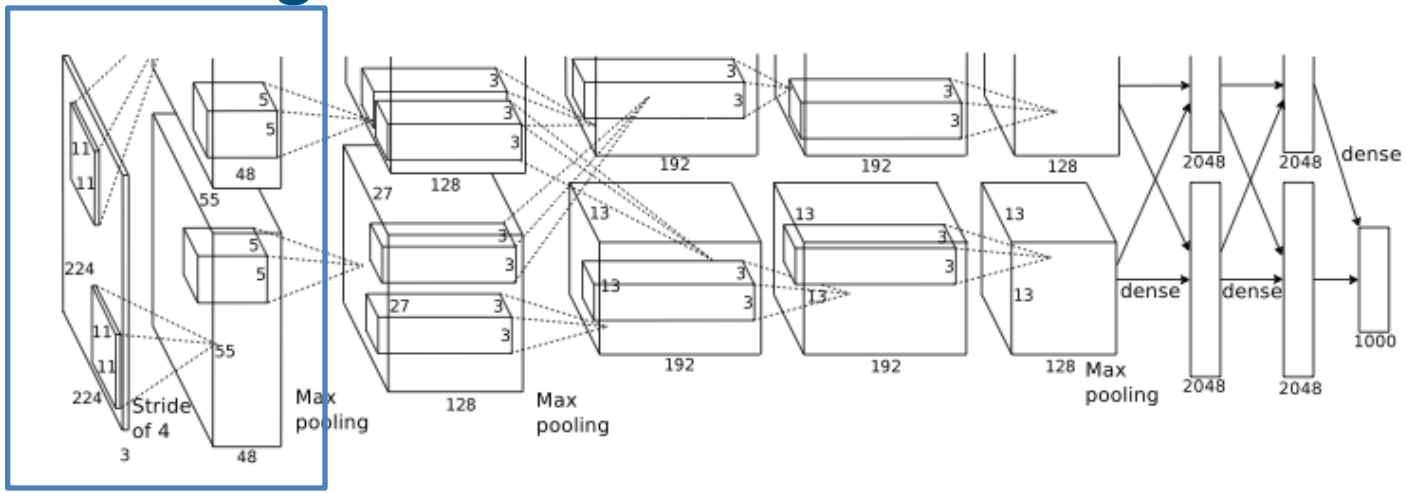
Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

34,944 + 290,400 = 325,344
기본형이 float32 = 4 byte

즉 325,344 x 4=1,301,376 byte
=1,301.376 KB

ImageNet Challenge

AlexNet
(2012)



이 부분에 얼만큼의 파라미터가 생성될까?

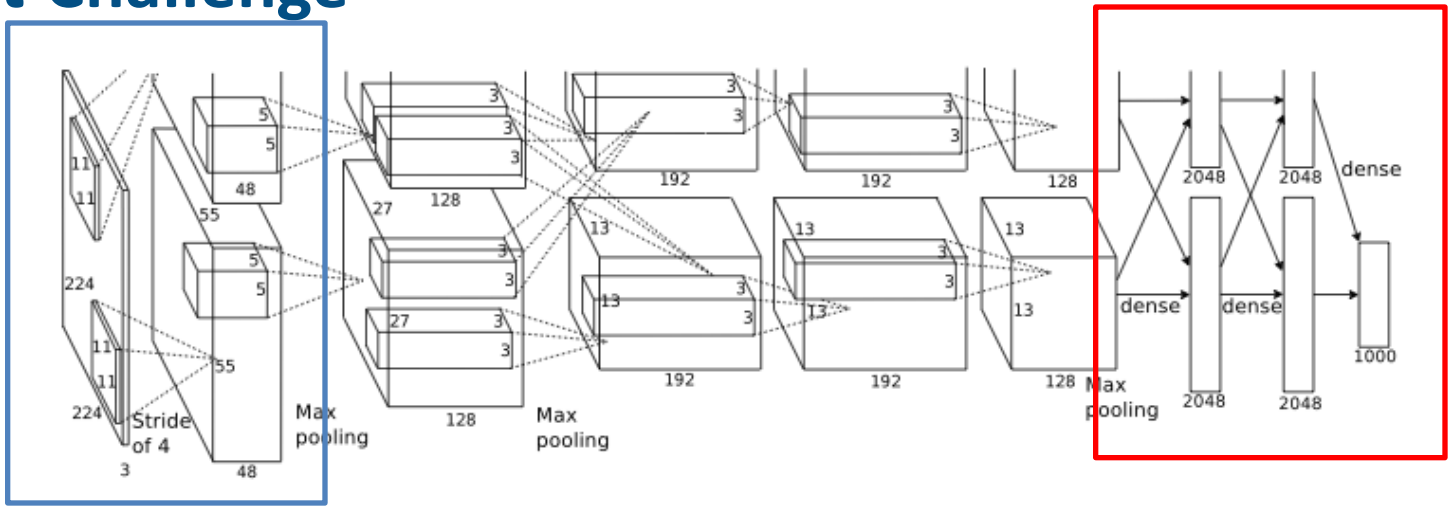
Input: 227x227x3
Filter: 11x11
#Filter: 3 -> 96
Output: $floor[\frac{227-11}{4}+1]=55$

34,944 + 290,400 = 325,344
기본형이 float32 = 4 byte

즉 325,344 x 4=1,301,376 byte
=1,301.376 KB
=1.3 MB

ImageNet Challenge

AlexNet
(2012)



이 부분에 얼만큼의 파라미터가 생성될까?

Input: 13x13x2048
Output: 2048 ->2048 ->1000

13 x13 x 256= 43,264
43,264 x4096 +4096 =177,213,440
=177.2 MB

4096 x4096+4096 = 16,781,312
= 16.8 MB

ImageNet Challenge

Memory Per Model

	Size (MB)	Error % (top-5)
SqueezeNet Compressed	0.6	19.7%
SqueezeNet	4.8	19.7%
AlexNet	240	19.7%
Inception v3	84	5.6%
VGG-19	574	7.5%
ResNet-50	102	7.8%
ResNet-200	519	4.8%

(출처:AWS re:Invent 2016: Bringing Deep Learning to the Cloud with Amazon EC2 (CMP314)

ImageNet Challenge

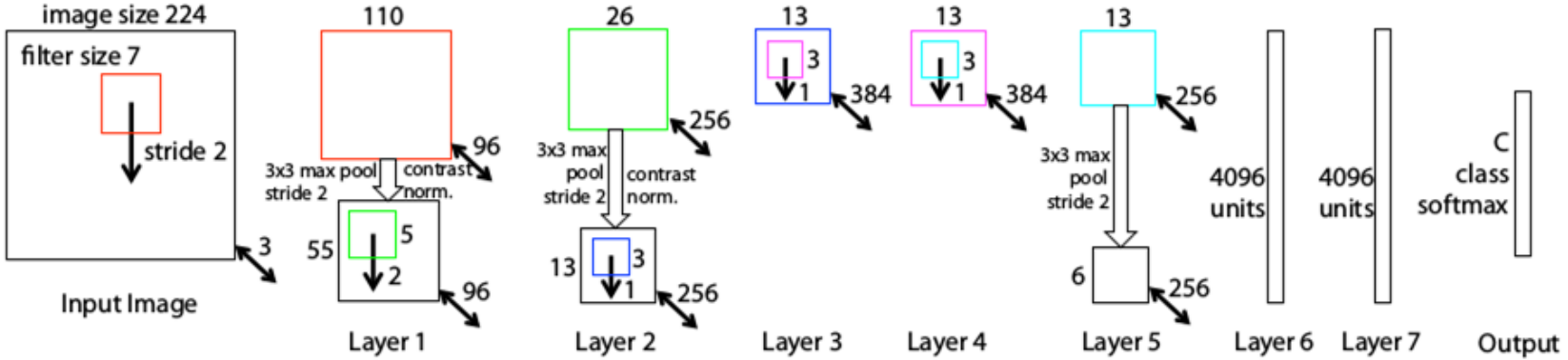
Memory Per Model

	Size (MB)	Error % (top-5)
SqueezeNet Compressed	0.6	19.7%
SqueezeNet	4.8	19.7%
AlexNet	240	19.7%
Inception v3	84	5.6%
VGG-19	574	7.5%
ResNet-50	102	7.8%
ResNet-200	519	4.8%

(출처:AWS re:Invent 2016: Bringing Deep Learning to the Cloud with Amazon EC2 (CMP314)

ImageNet Challenge

ZF Network(2013)



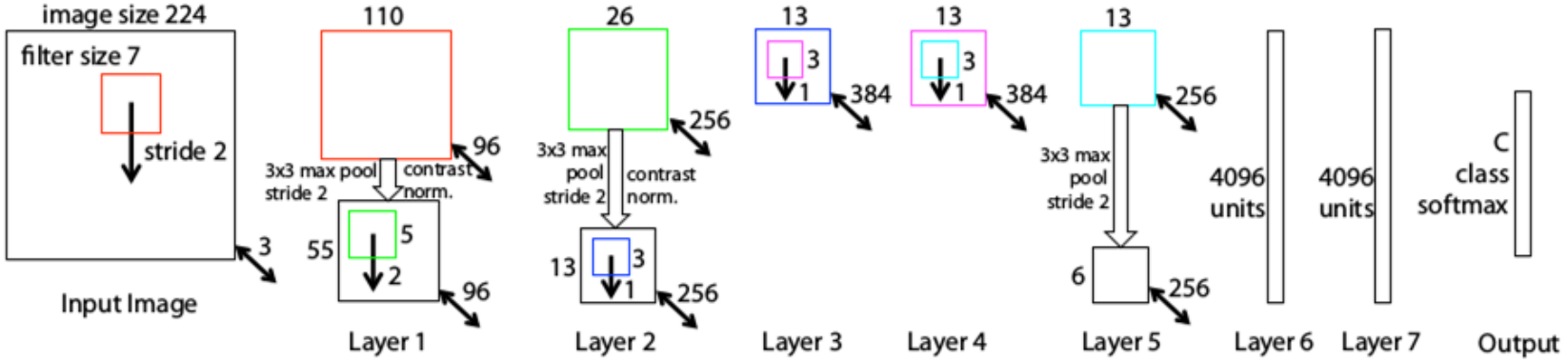
ZF Net Architecture

전체적 구조는 같음

(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

ZF Network(2013)



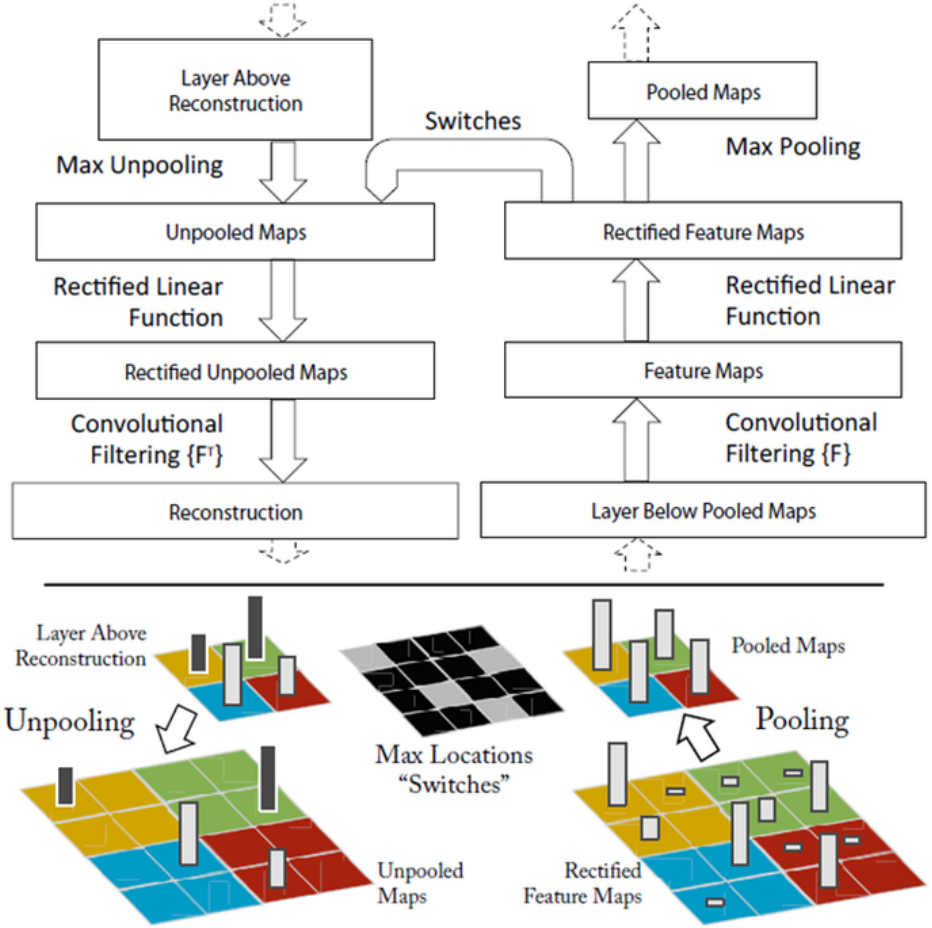
ZF Net Architecture

전체적 구조는 같음
ZF Net에서 배울 점은
CNN의 시각화

(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

Convolution 연산을 역으로 진행하여 필터가 어떤 모양에 반응하는지 시각화

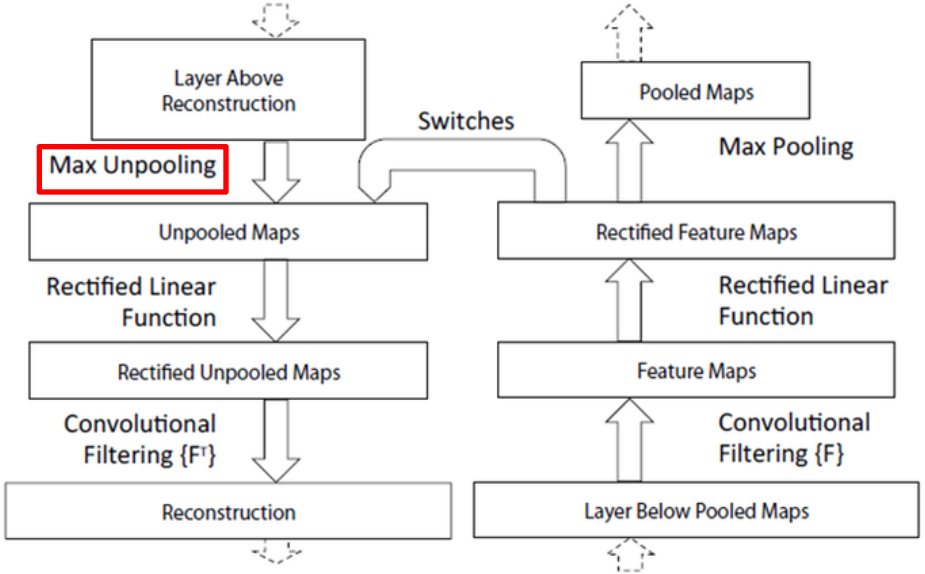


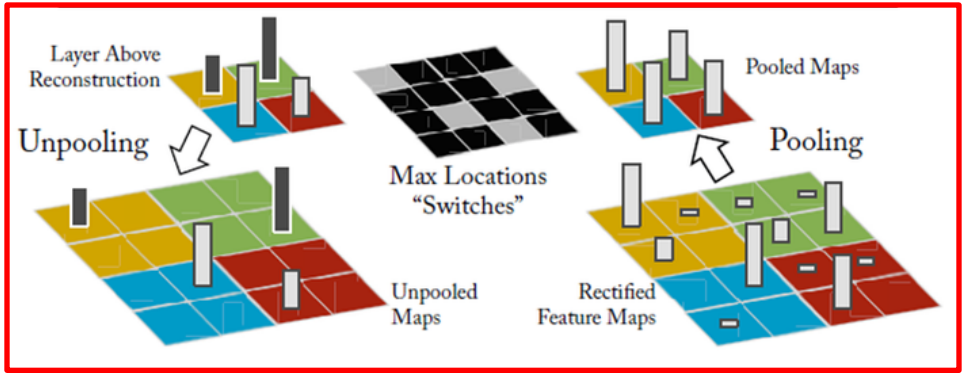
(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

Convolution 연산을 역으로 진행하여 필터가 어떤 모양에 반응하는지 시각화

Max pooling은 Max였던 위치를 기억하여 그대로 확대





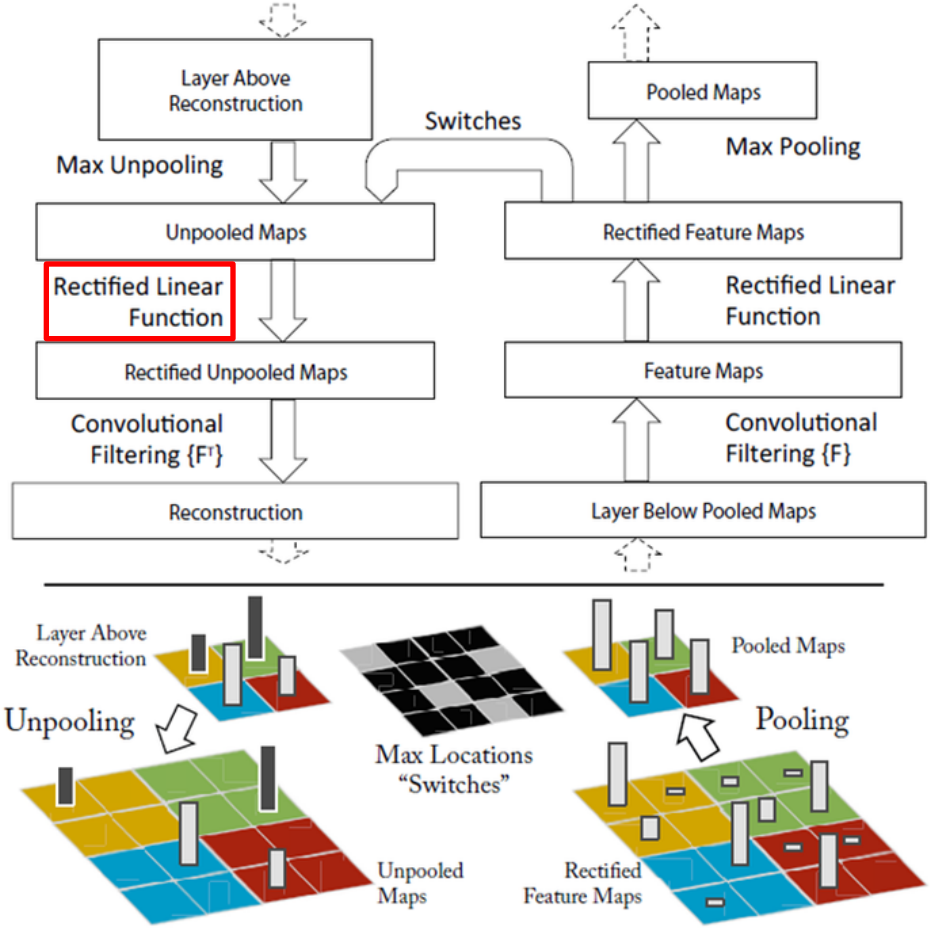
(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

Convolution 연산을 역으로 진행하여 필터가 어떤 모양에 반응하는지 시각화

Max pooling은 Max였던 위치를 기억하여 그대로 확대

Rectified Linear Unit는 0이하의 숫자들이 문제가 되는데 연구에 의하면 시각화에 큰 영향이 없었다고 함



(출처:https://arxiv.org/pdf/1311.2901.pdf)

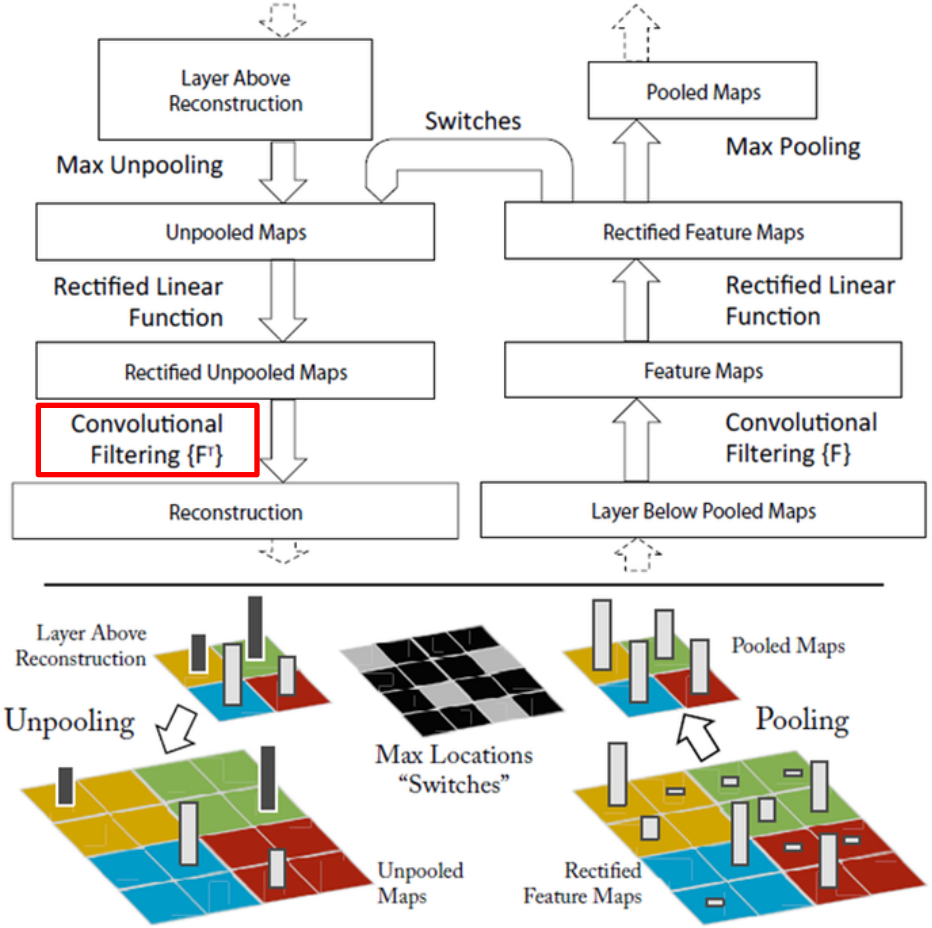
ImageNet Challenge

Convolution 연산을 역으로 진행하여 필터가 어떤 모양에 반응하는지 시각화

Max pooling은 Max였던 위치를 기억하여 그대로 확대

Rectified Linear Unit는 0이하의 숫자들이 문제가 되는데 연구에 의하면 시각화에 큰 영향이 없었다고 함

Convolution 연산은 weight 값을 알고 있으므로 그냥 역으로 연산



(출처:https://arxiv.org/pdf/1311.2901.pdf)

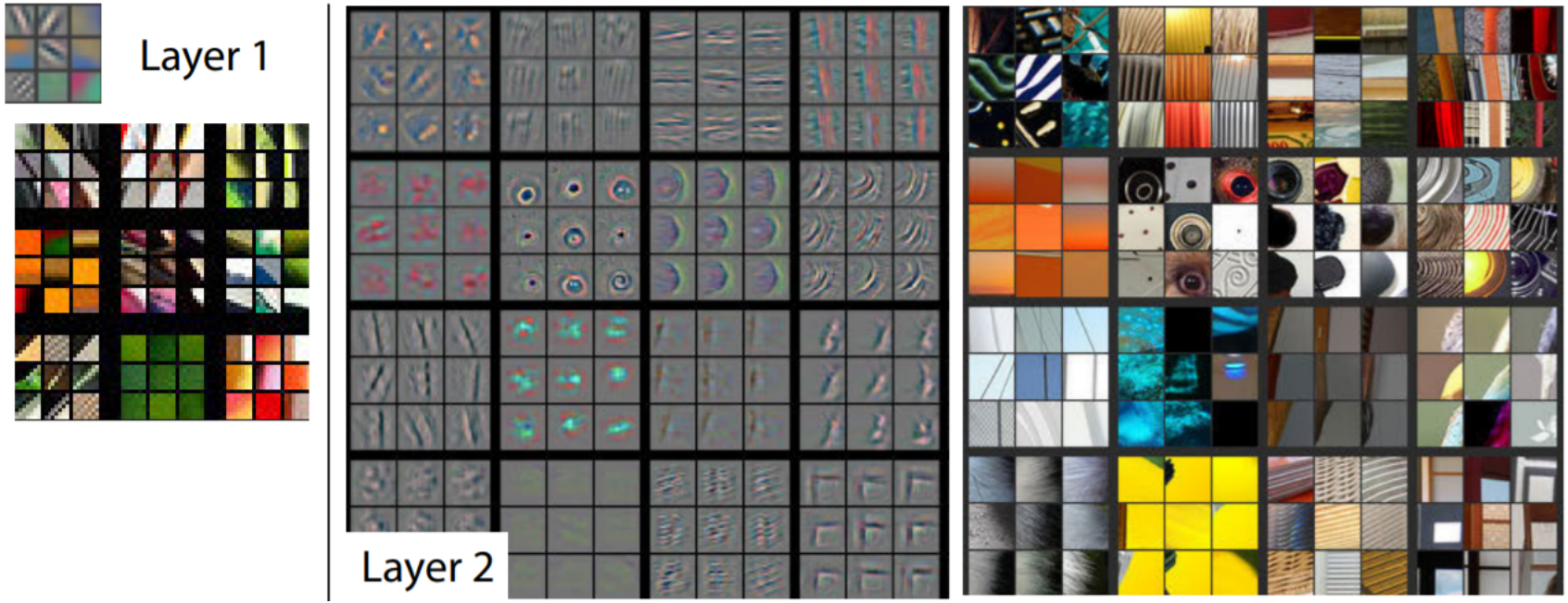
ImageNet Challenge

Layer 별로 필터 별로
강한 자극을 시각화 한 결과

(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

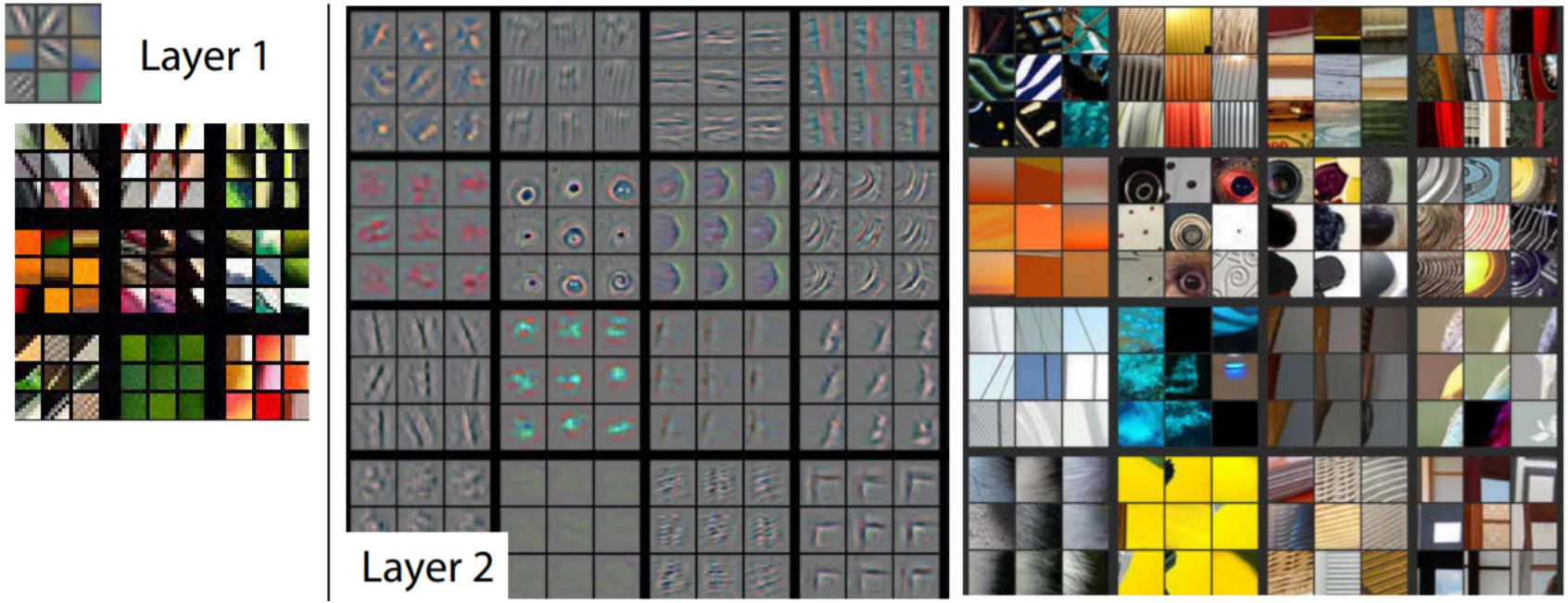
Layer 별로 필터 별로
강한 자극을 시각화 한 결과



(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

Layer 별로 필터 별로
강한 자극을 시각화 한 결과

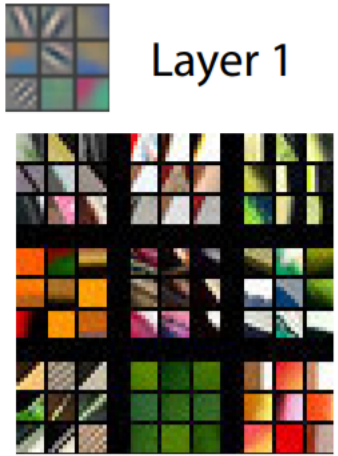


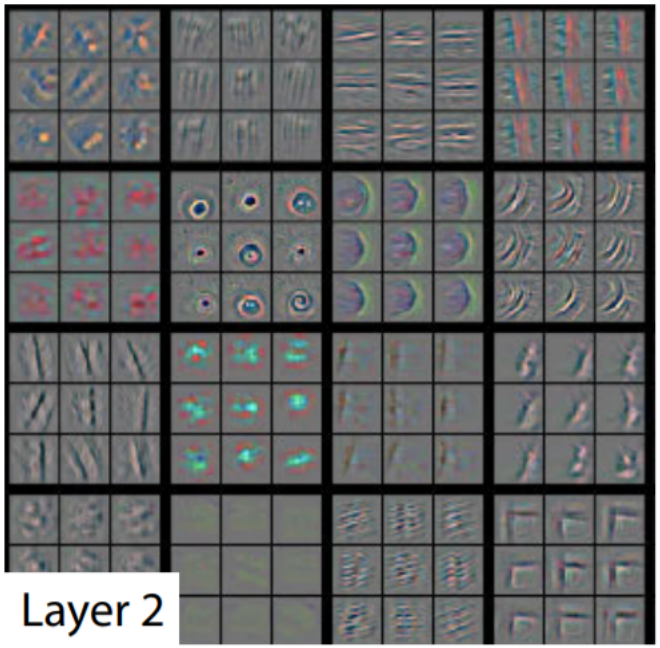
필터 별 자극

(출처:https://arxiv.org/pdf/1311.2901.pdf)

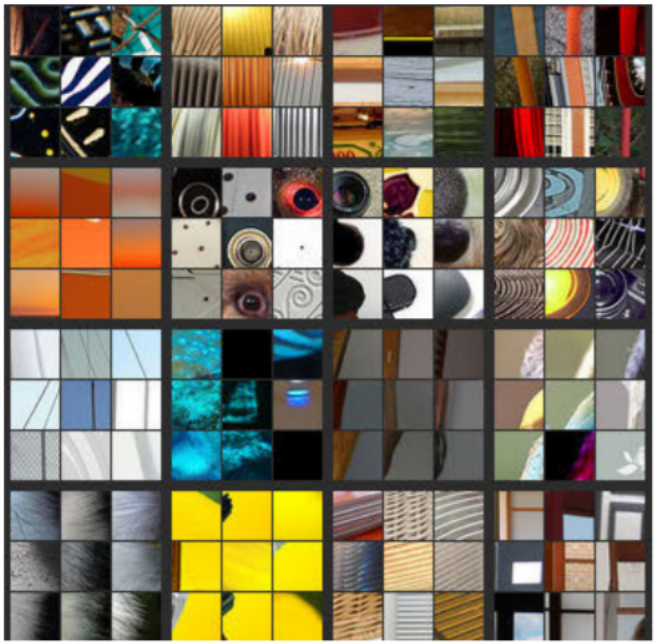
ImageNet Challenge

Layer 별로 필터 별로
강한 자극을 시각화 한 결과





필터 별 자극



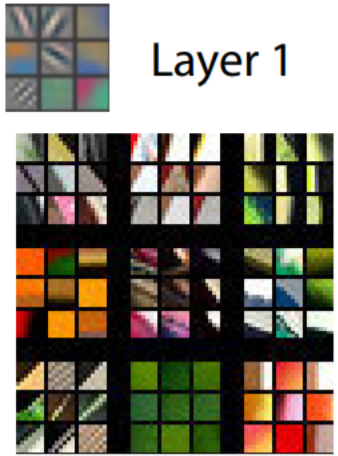
원본 이미지

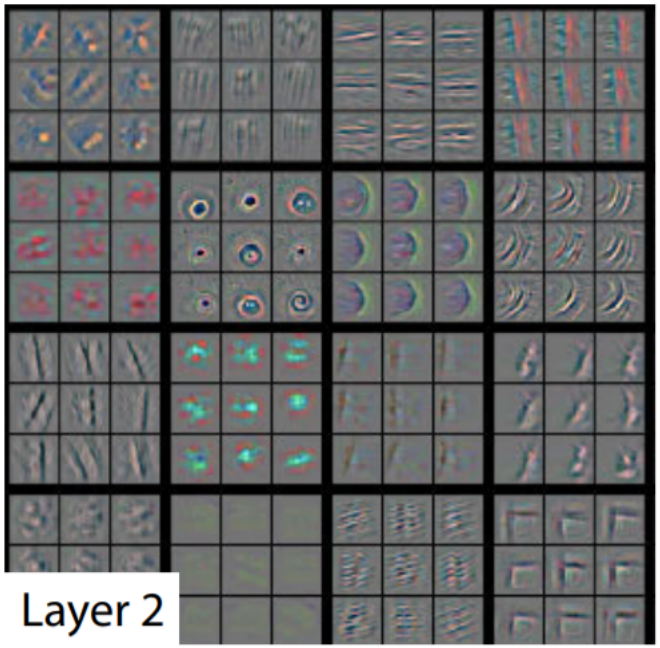
(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

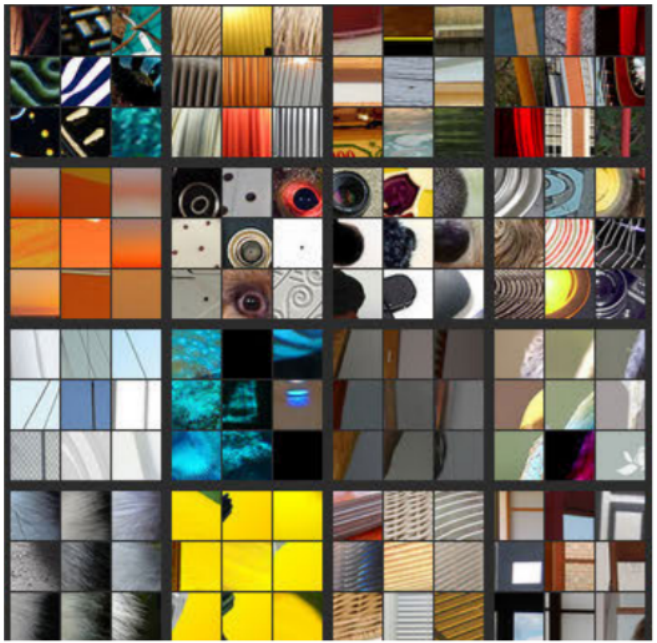
Layer 별로 필터 별로
강한 자극을 시각화 한 결과

단순한 선들





필터 별 자극



원본 이미지

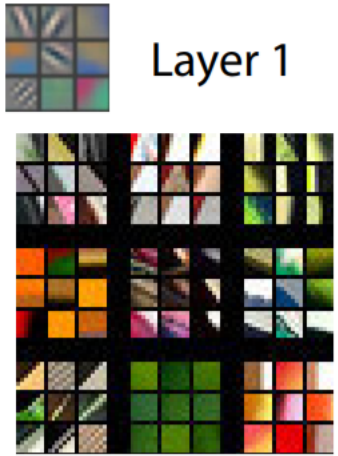
(출처:https://arxiv.org/pdf/1311.2901.pdf)

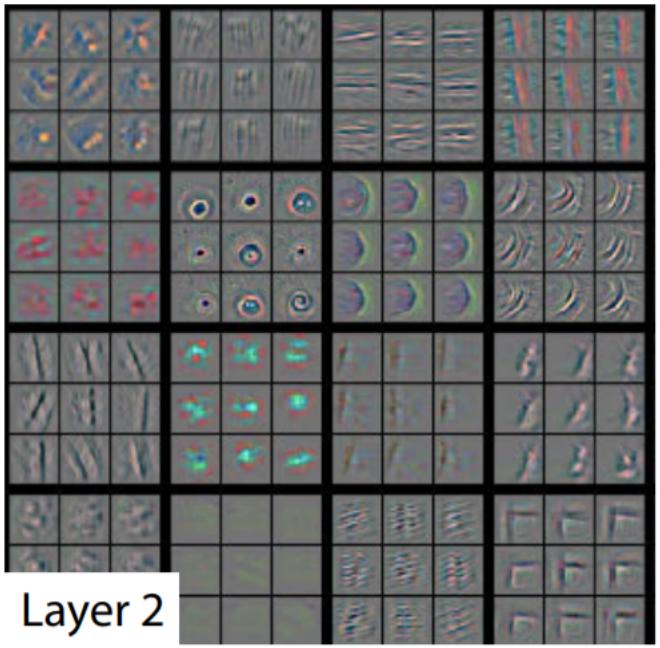
ImageNet Challenge

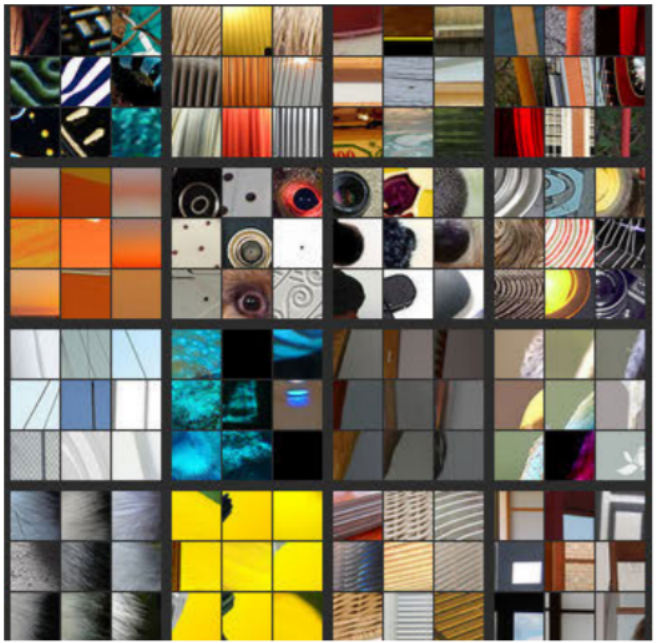
Layer 별로 필터 별로
강한 자극을 시각화 한 결과

단순한 선들

비교적 복잡







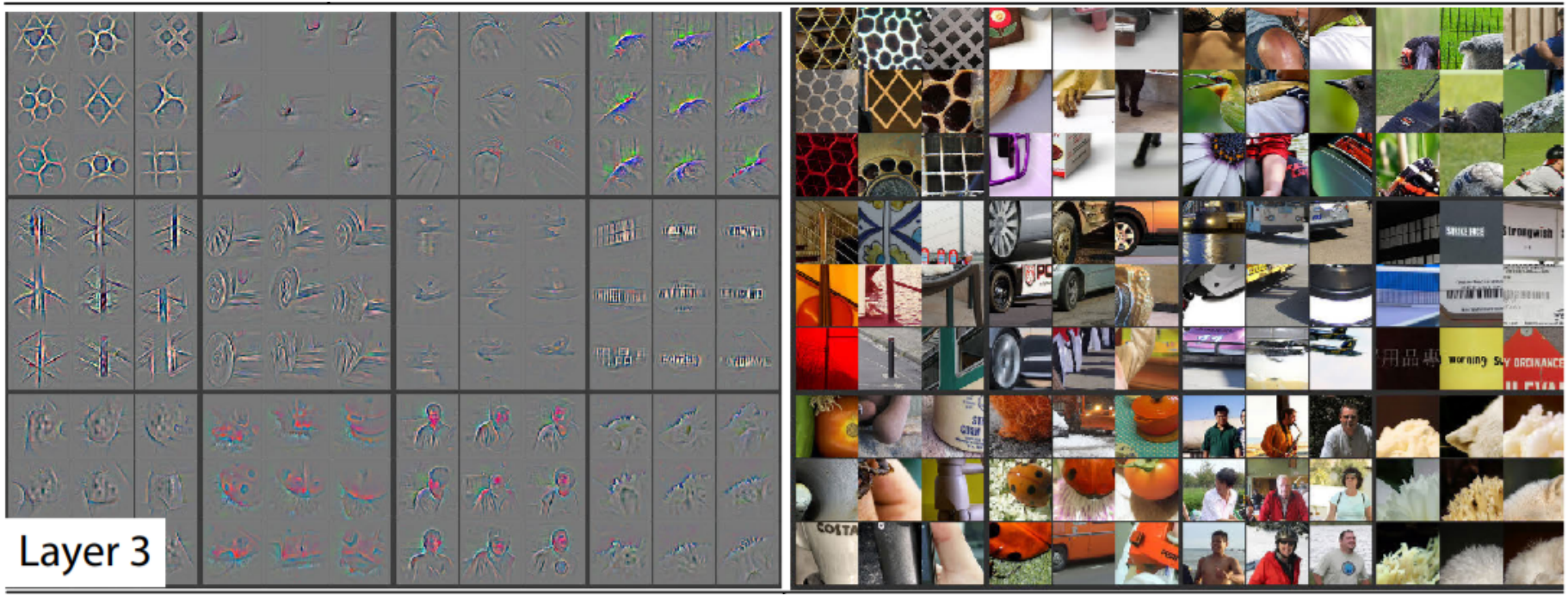
필터 별 자극

원본 이미지

(출처:https://arxiv.org/pdf/1311.2901.pdf)

ImageNet Challenge

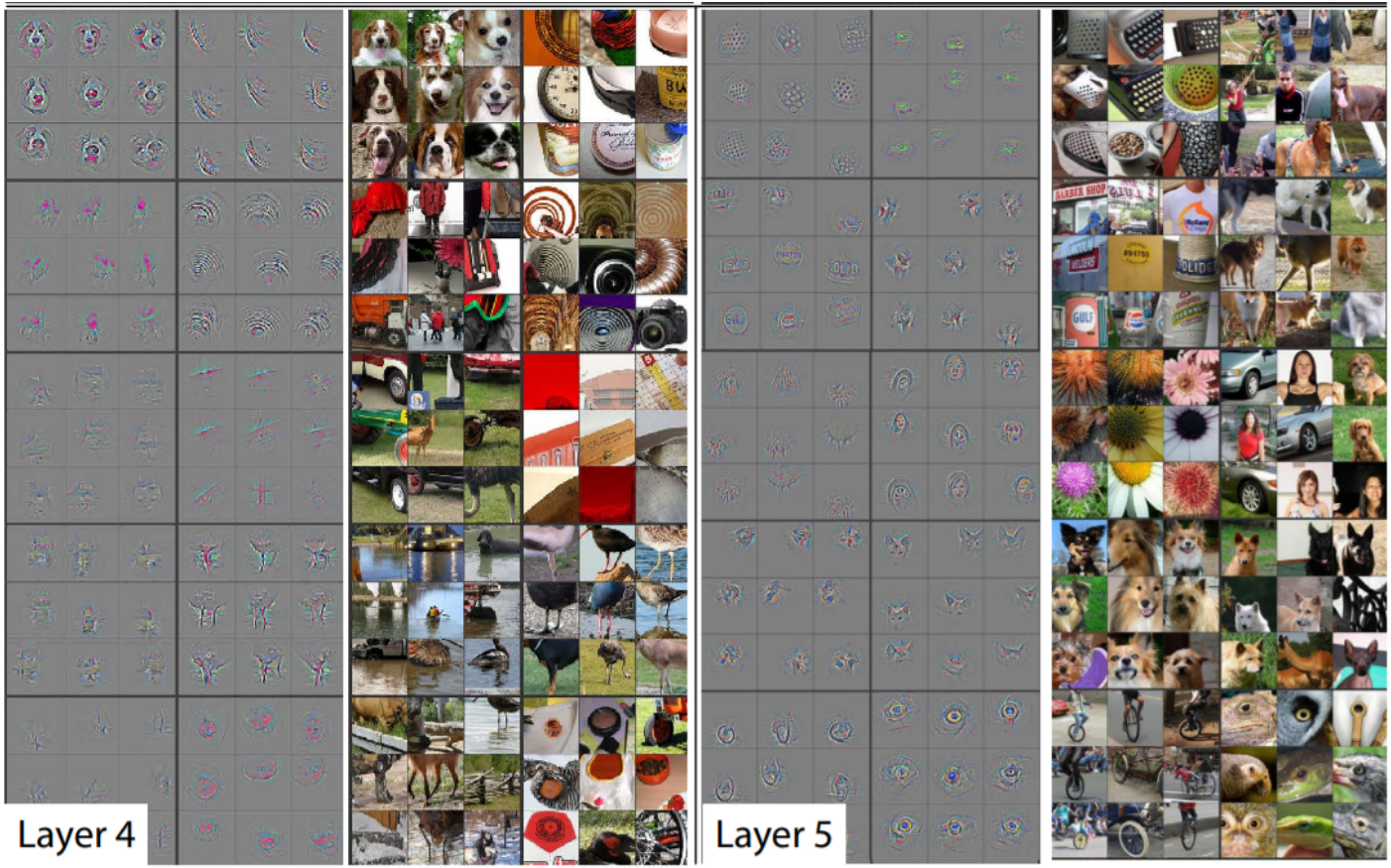
더 복잡



(출처:https://arxiv.org/pdf/1311.2901.pdf)

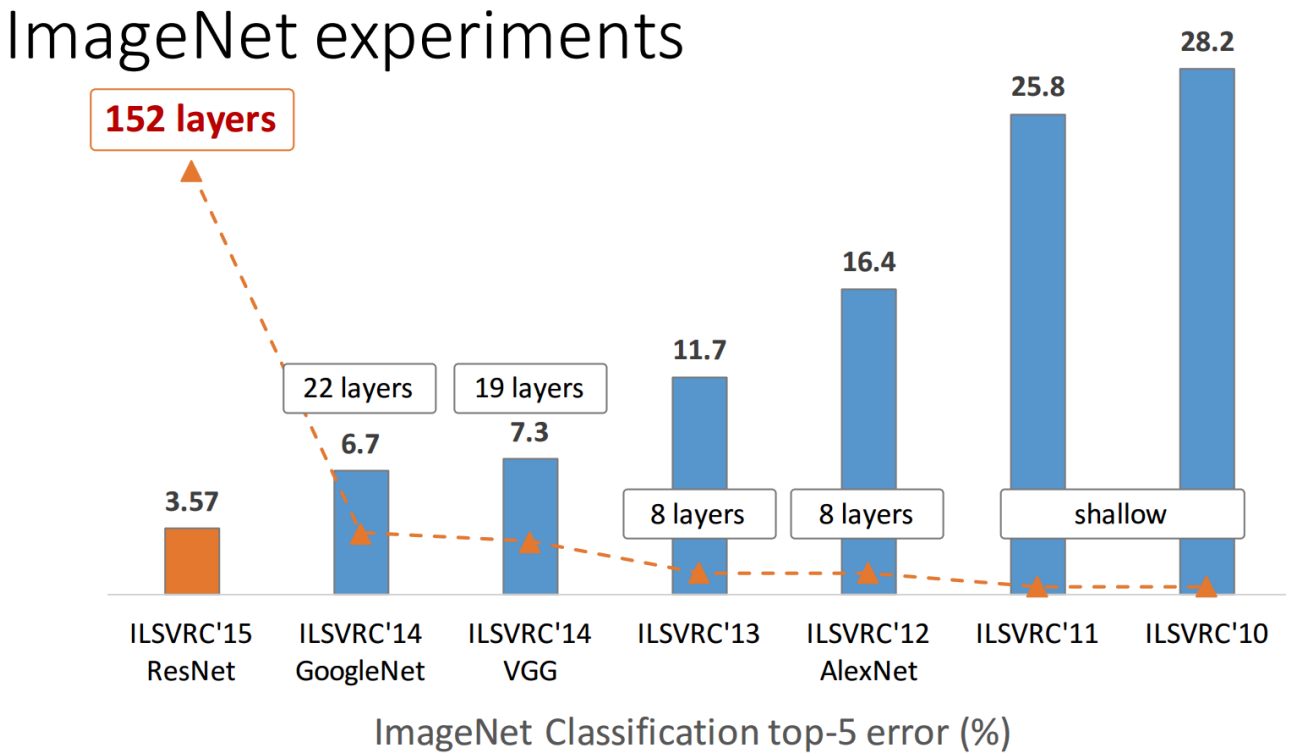
ImageNet Challenge

더욱 더 복잡

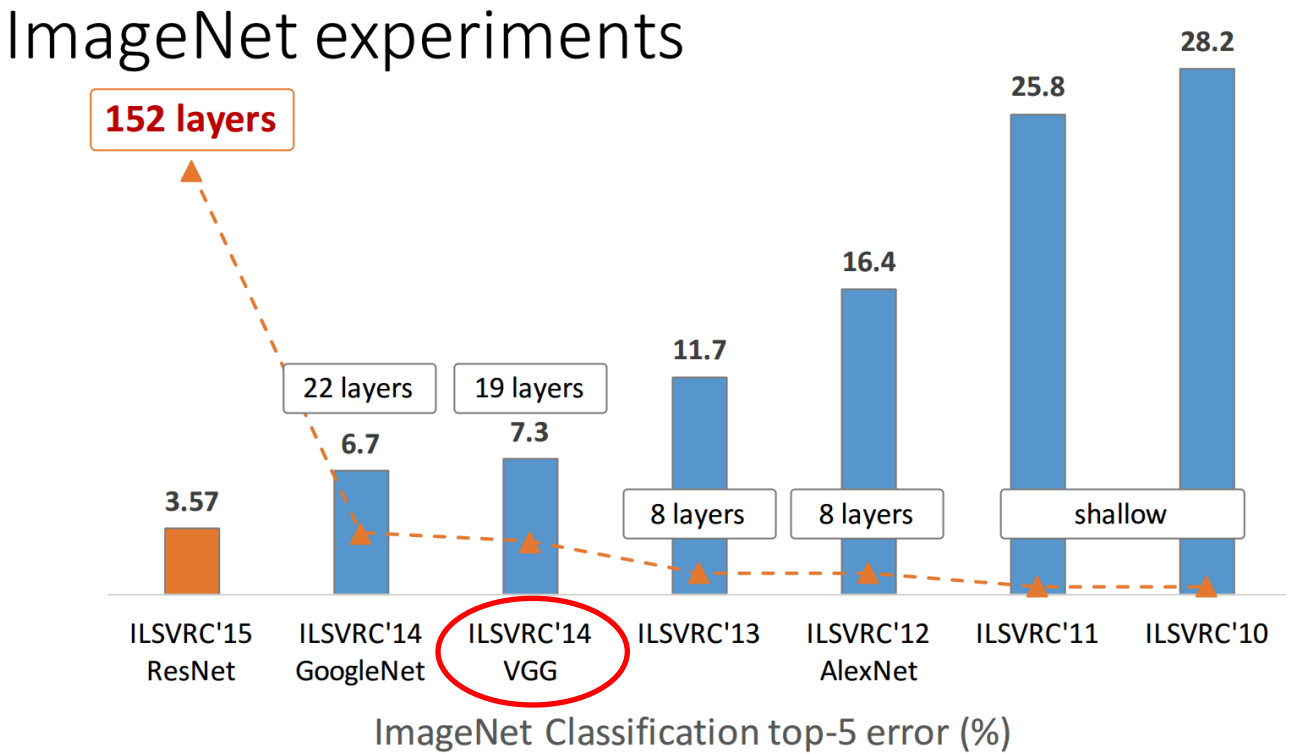


(출처:https://arxiv.org/pdf/1311.2901.pdf)

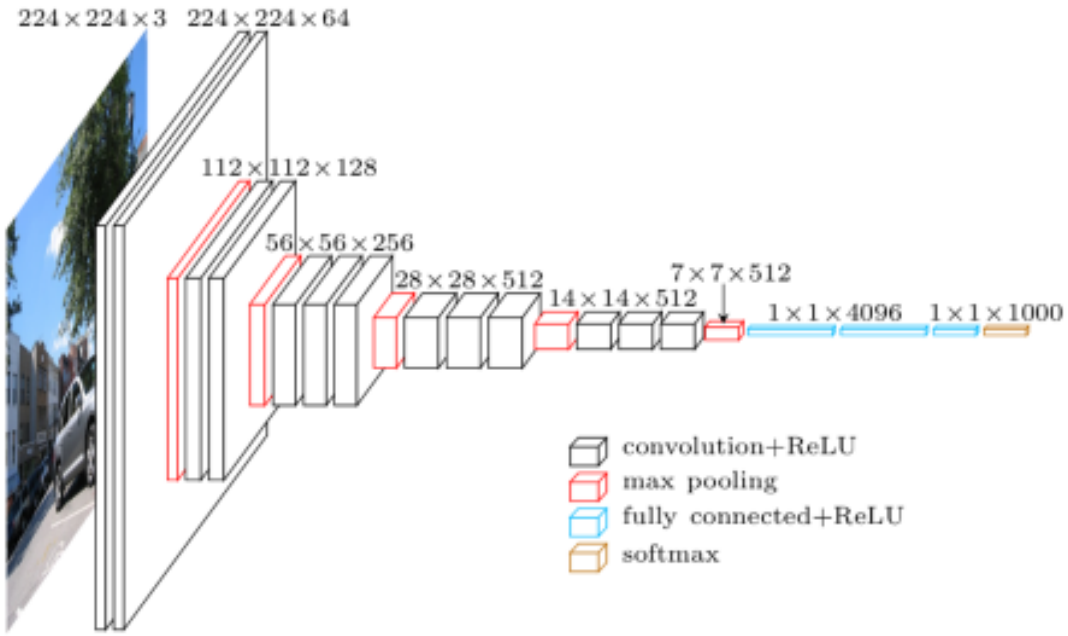
VGG Network


ImageNet experiments
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

VGG Network

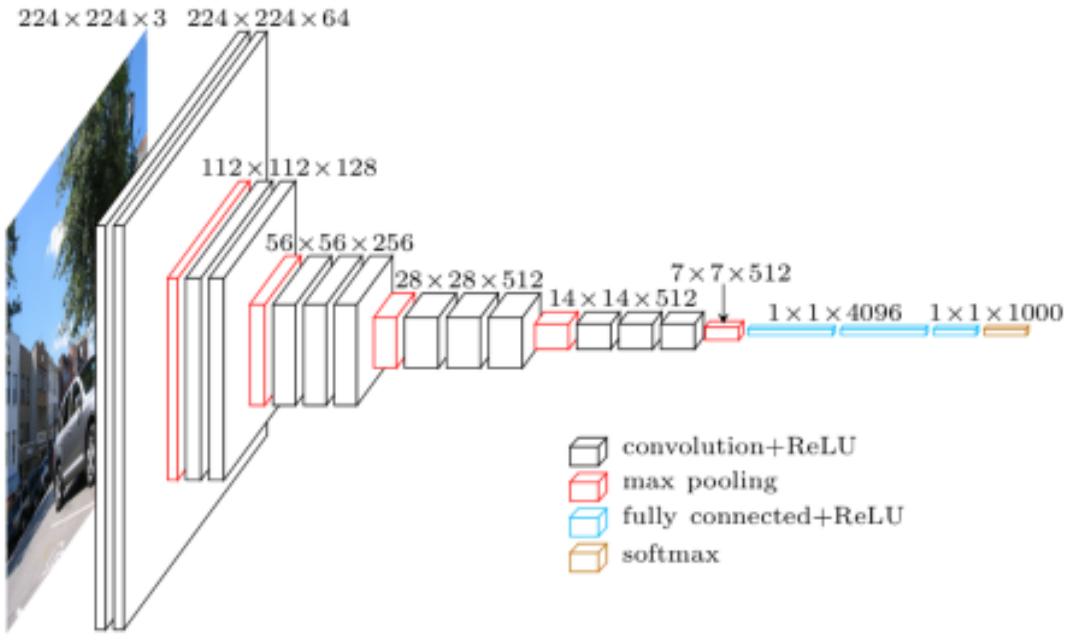

ImageNet experiments
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

VGG Network



(출처:https://www.cs.toronto.edu/~frossard/post/vgg16/)

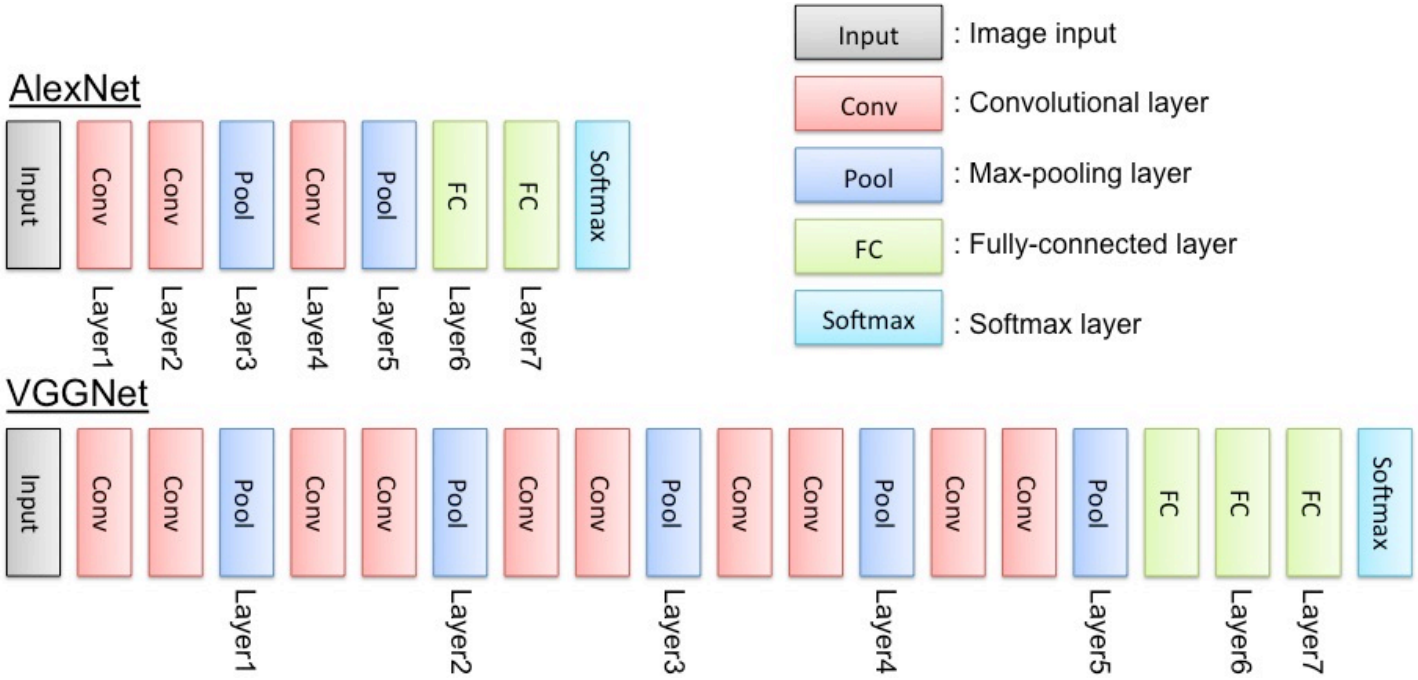
VGG Network



2014년 2위이지만 간단한 구조 때문에 오히려 더 많이 사용됨.

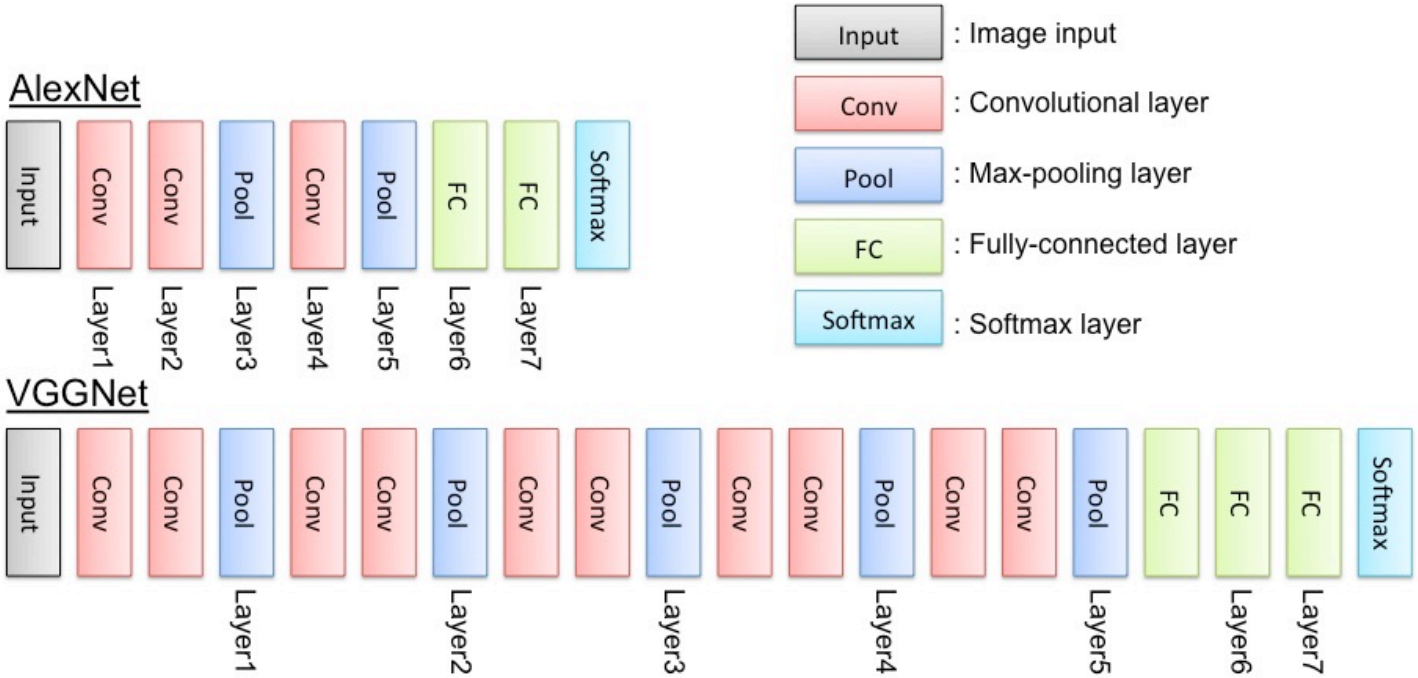
(출처:https://www.cs.toronto.edu/~frossard/post/vgg16/)

VGG Network



(출처: http://www.hirokatsukataoka.net/research/cnnfeatureevaluation/cnnfeatureevaluation.html)

VGG Network



그냥 단순히 망의 깊이를 늘린 것

(출처: http://www.hirokatsukataoka.net/research/cnnfeatureevaluation/cnnfeatureevaluation.html)

VGG Network

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

망의 깊이가 모델의 성능에 어떤 영향을 끼치는가?

(출처:https://arxiv.org/pdf/1409.1556.pdf)

VGG Network

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

망의 깊이가 모델의 성능에 어떤 영향을 끼치는가?

3가지만의 조합으로 모델 구성

(출처:https://arxiv.org/pdf/1409.1556.pdf)

VGG Network

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

망의 깊이가 모델의 성능에 어떤 영향을 끼치는가?

3가지만의 조합으로 모델 구성

- 3x3 convolution stride 1
- Max Pooling
- Fully Connected Layer

(출처:https://arxiv.org/pdf/1409.1556.pdf)

VGG Network

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

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters
```

VGG Network

Memory Per Model

	Size (MB)	Error % (top-5)
SqueezeNet Compressed	0.6	19.7%
SqueezeNet	4.8	19.7%
AlexNet	240	19.7%
Inception v3	84	5.6%
VGG-19	574	7.5%
ResNet-50	102	7.8%
ResNet-200	519	4.8%

(출처:AWS re:Invent 2016: Bringing Deep Learning to the Cloud with Amazon EC2 (CMP314)

VGG Network

망이 깊을수록 성능이 좋다는 것은
알겠지만 효율이 그렇게 좋지는 않음

Memory Per Model

	Size (MB)	Error % (top-5)
SqueezeNet Compressed	0.6	19.7%
SqueezeNet	4.8	19.7%
AlexNet	240	19.7%
Inception v3	84	5.6%
VGG-19	574	7.5%
ResNet-50	102	7.8%
ResNet-200	519	4.8%

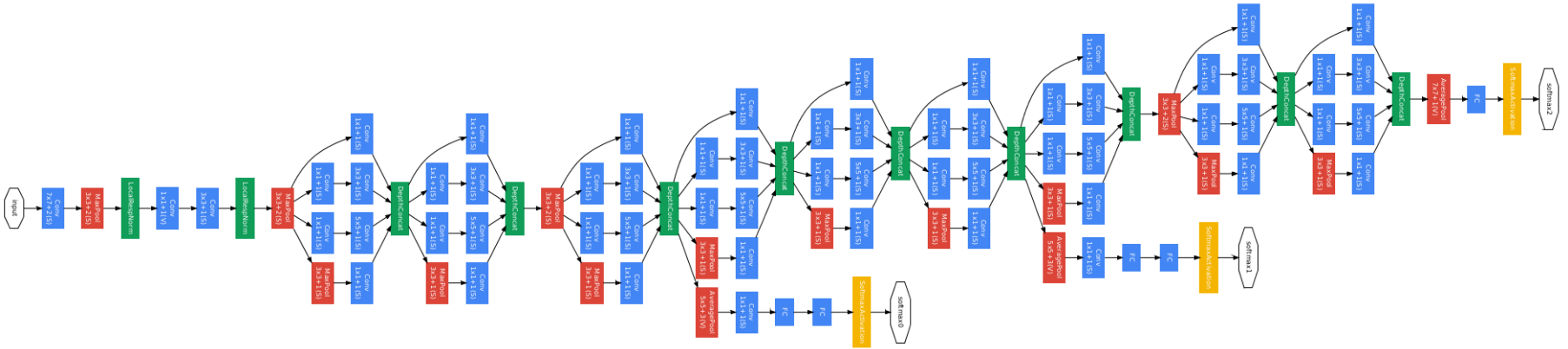
(출처:AWS re:Invent 2016: Bringing Deep Learning to the Cloud with Amazon EC2 (CMP314)

Google Network

Google Network

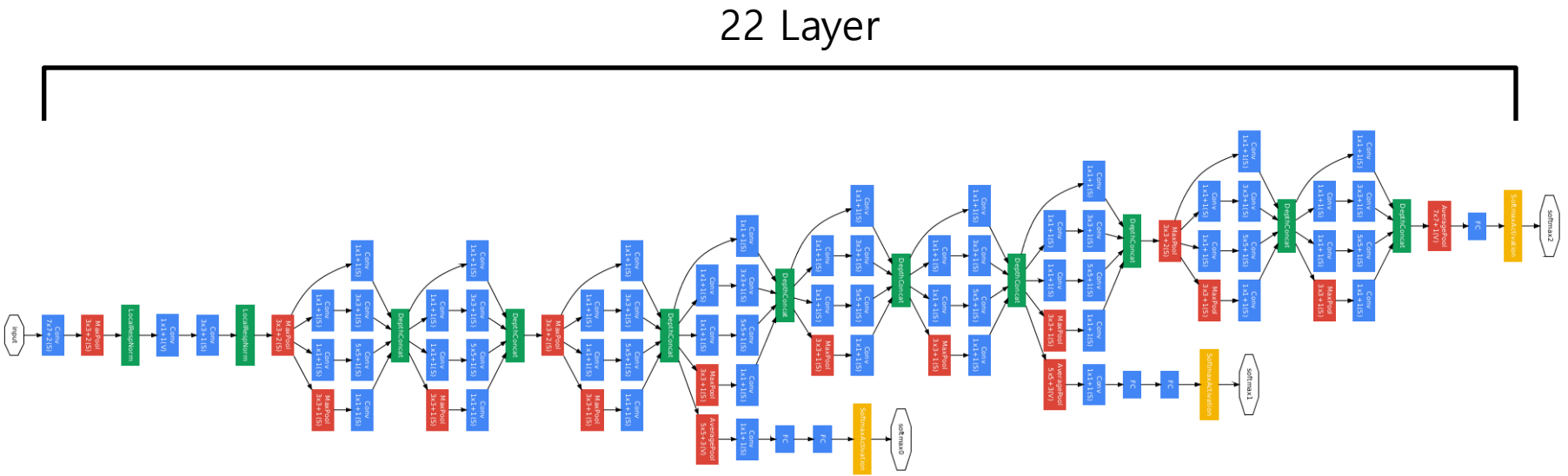
Inception Network

Google Network



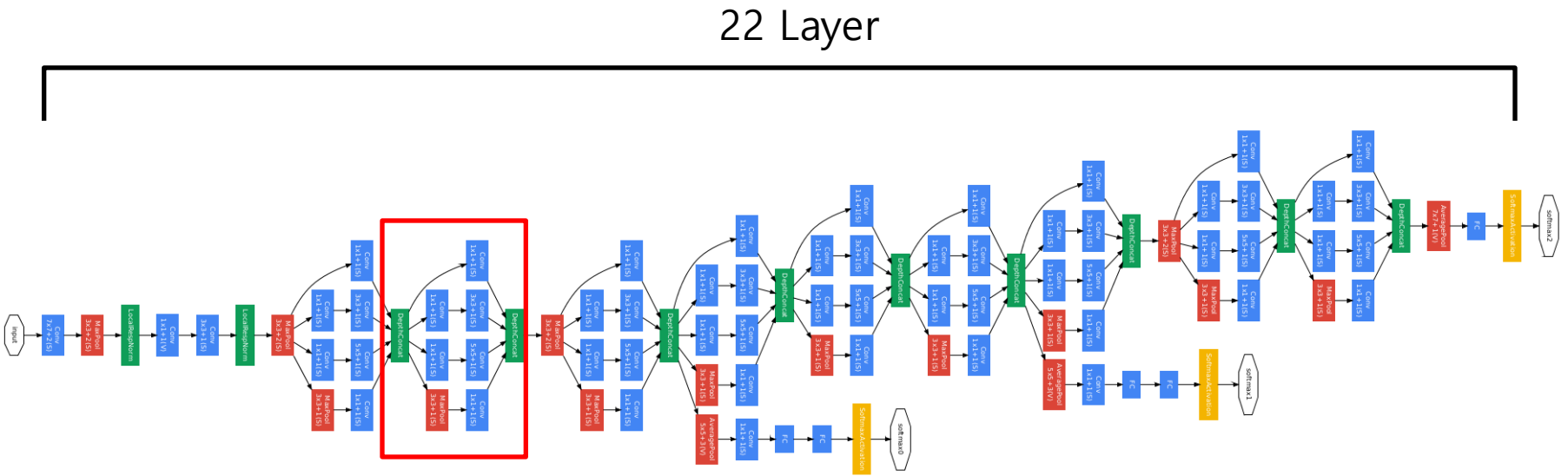
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(출처:https://arxiv.org/pdf/1409.4842.pdf)

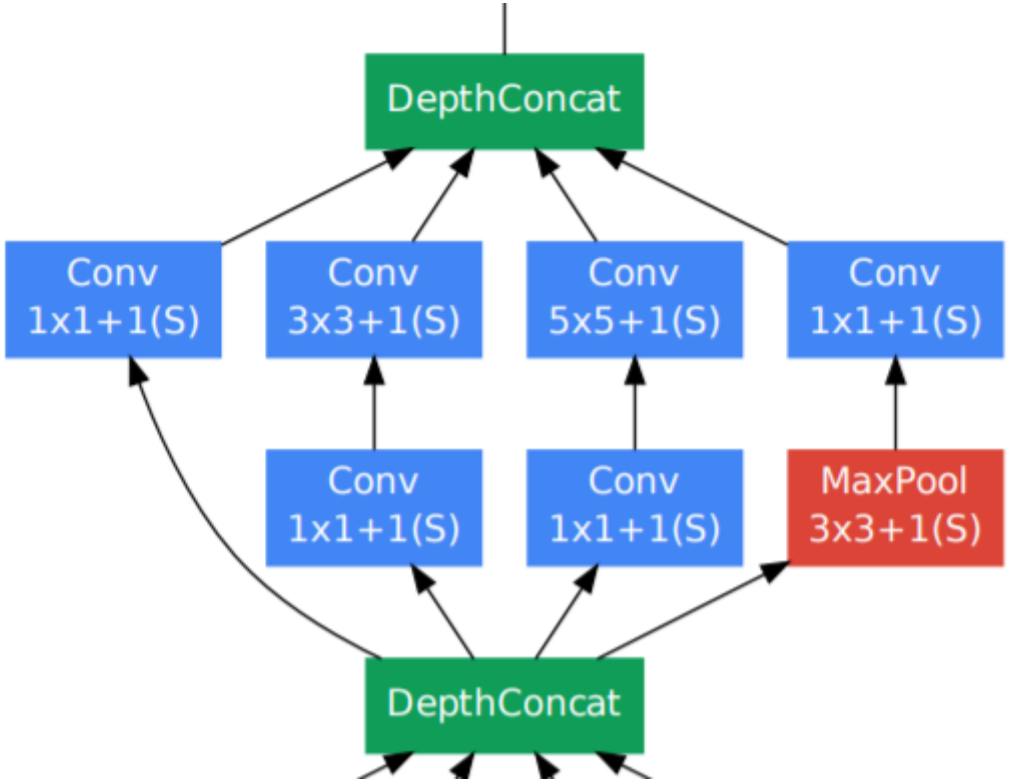
Google Network



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

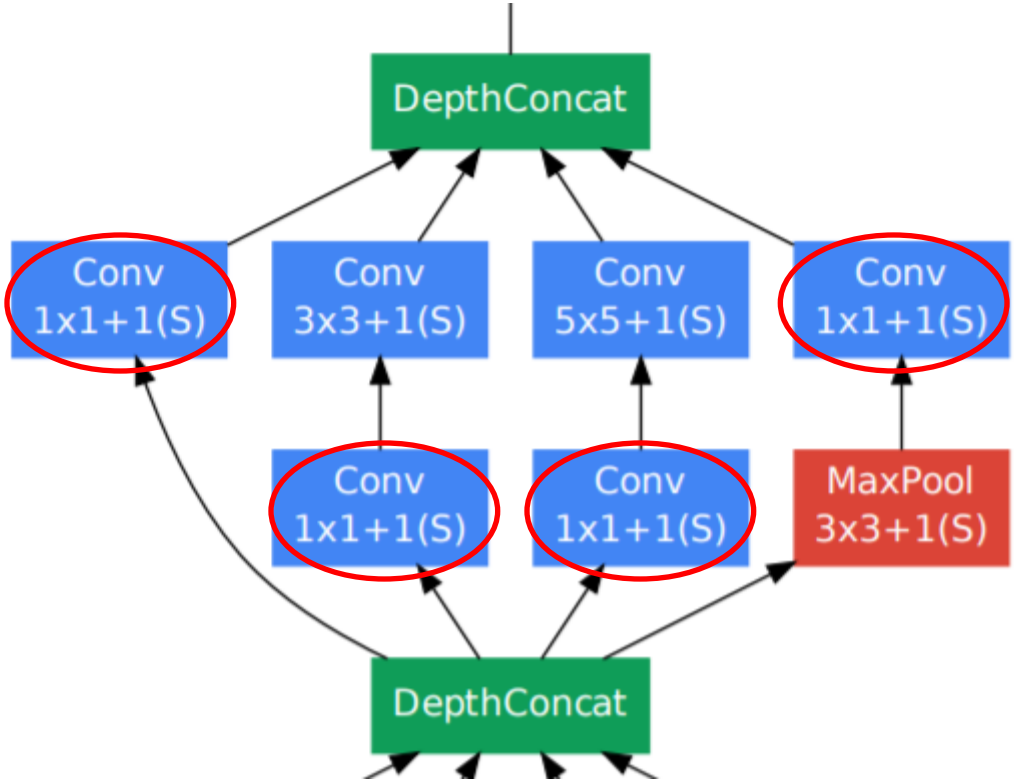
s는 stride



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

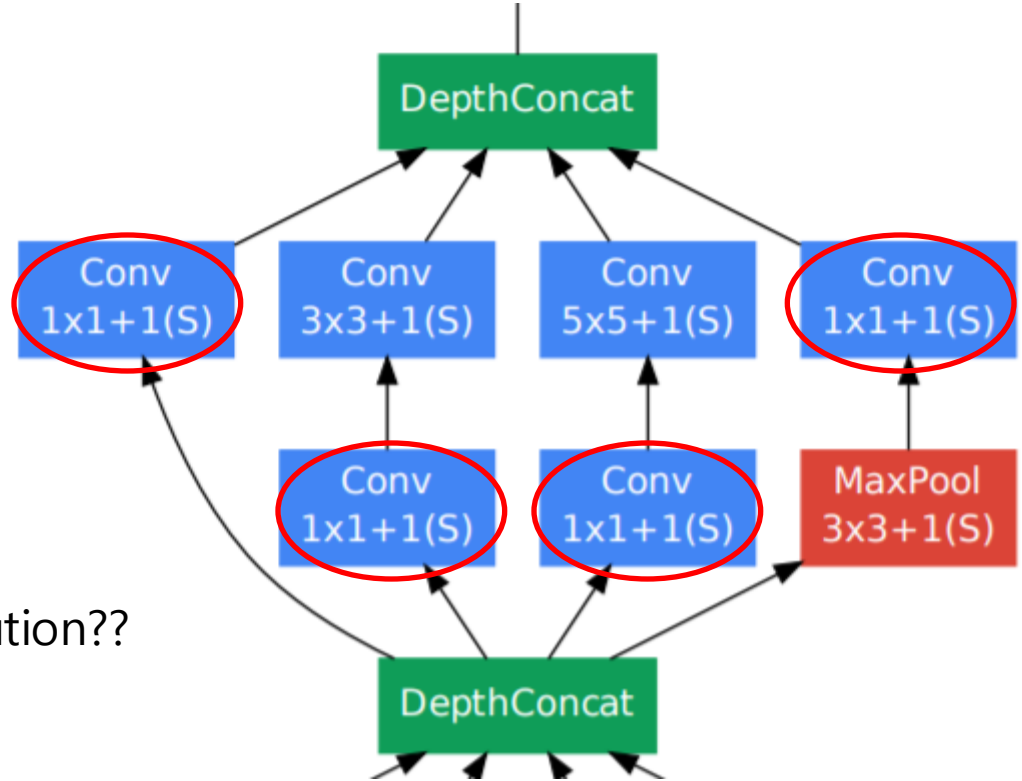
s는 stride



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

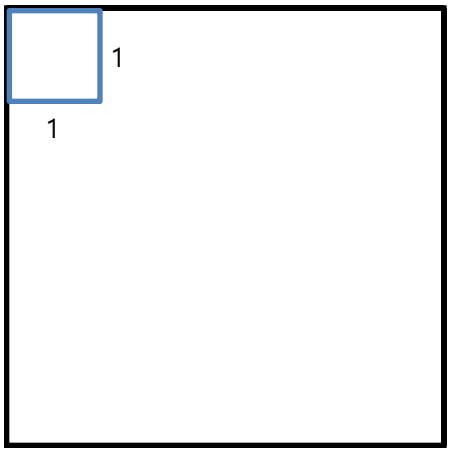
s는 stride



1x1 convolution??

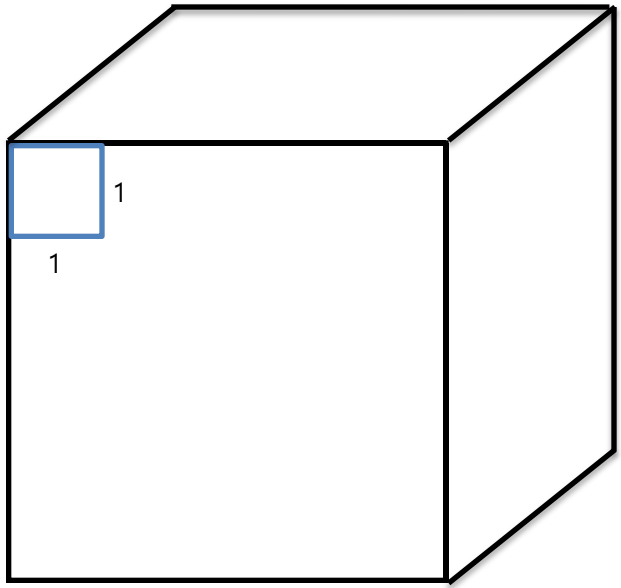
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



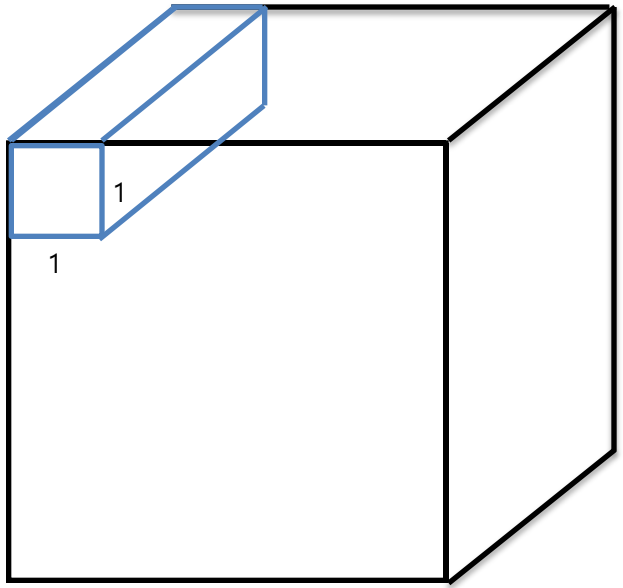
1x1 필터가 무슨 의미가 있지?

Google Network



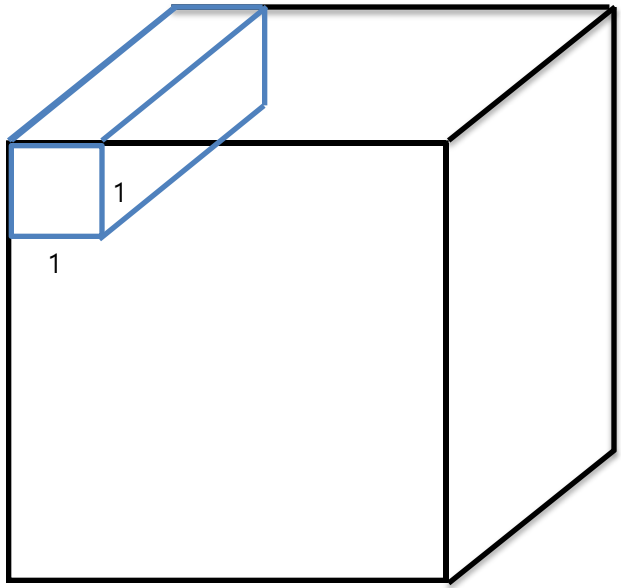
1x1 필터가 무슨 의미가 있지?

Google Network



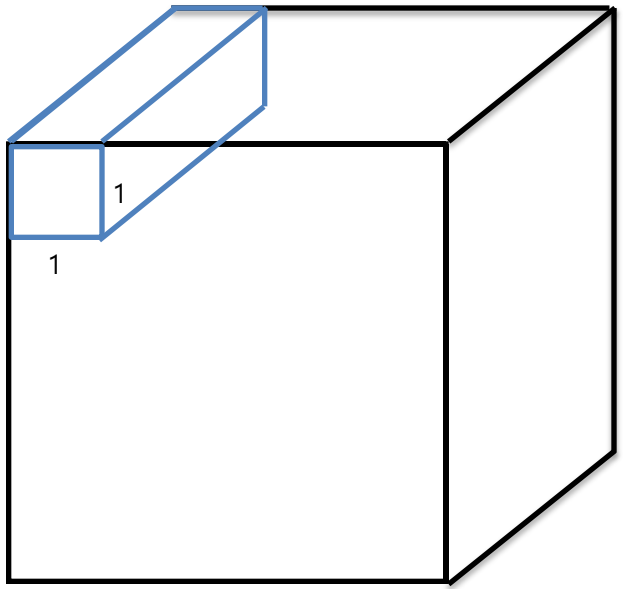
1x1 필터가 무슨 의미가 있지?

Google Network



1x1 convolution은 사실 필터를
node로 하는 Fully connected Network

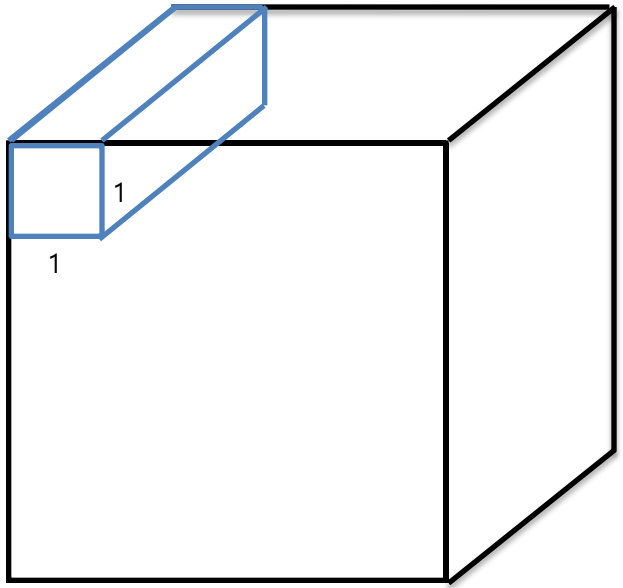
Google Network



1x1 convolution은 사실 필터를 node로 하는 Fully connected Network

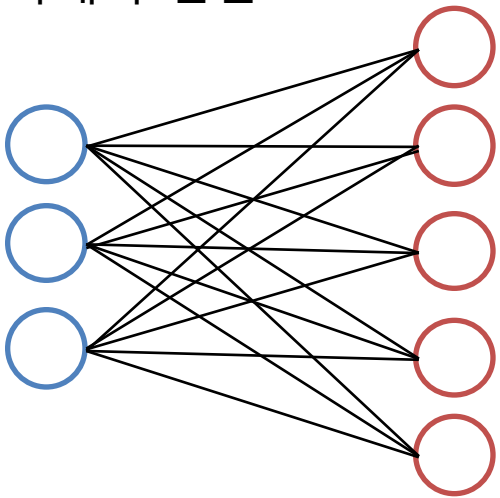
Input의 filter수가 3 output이 5이면 결국 아래와 같음

Google Network

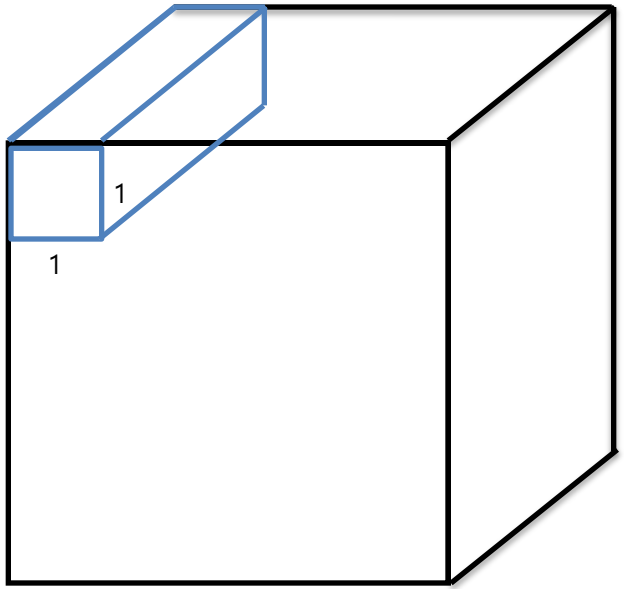


1x1 convolution은 사실 필터를 node로 하는 Fully connected Network

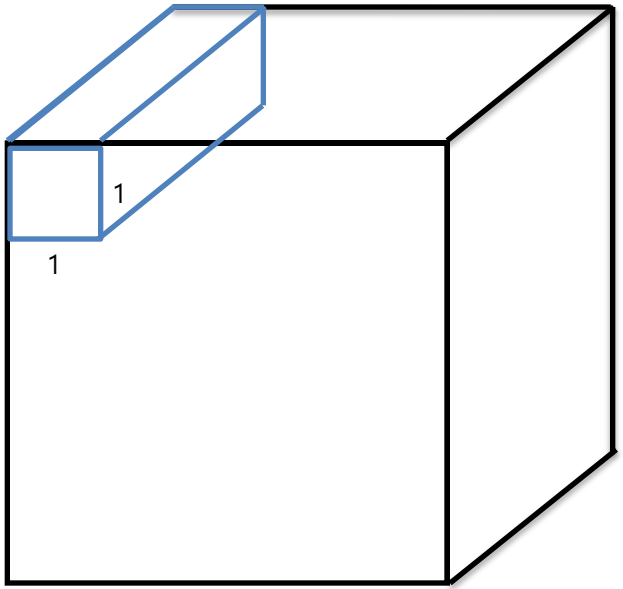
Input의 filter수가 3 output이 5이면 결국 아래와 같음



Google Network

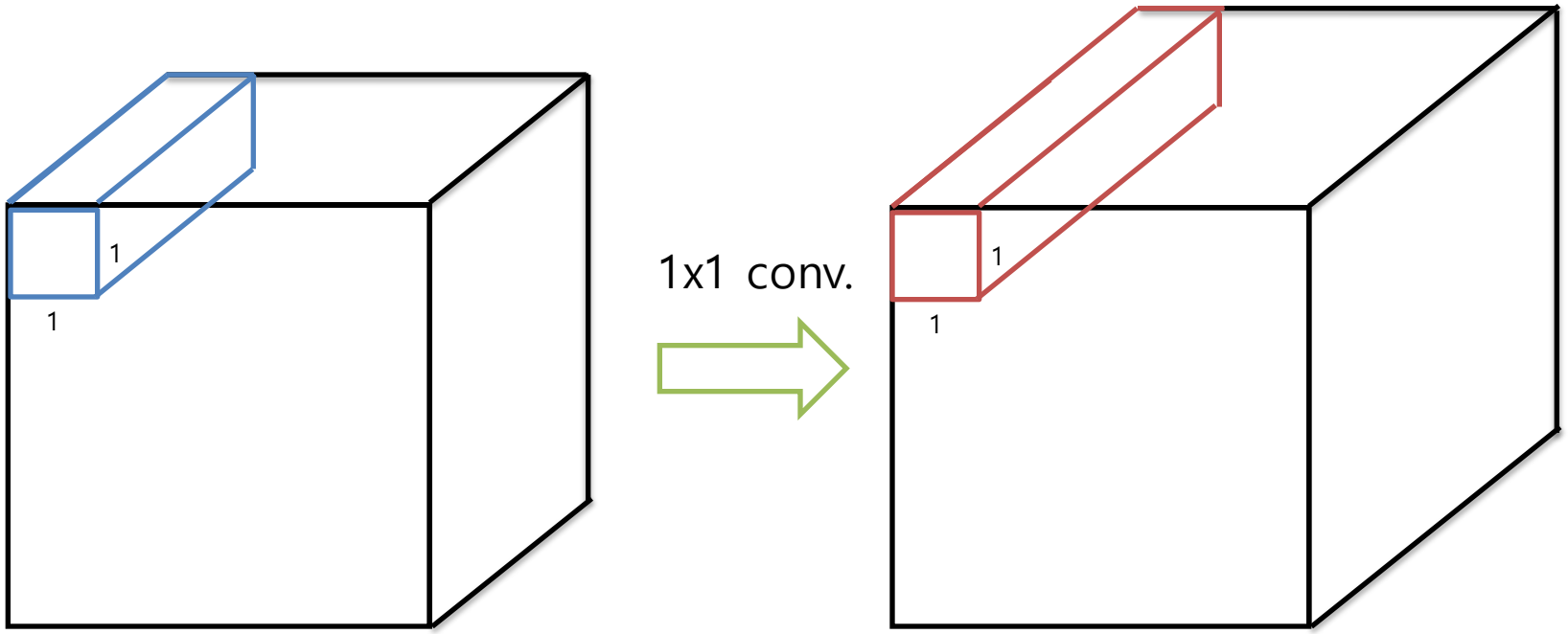


Google Network

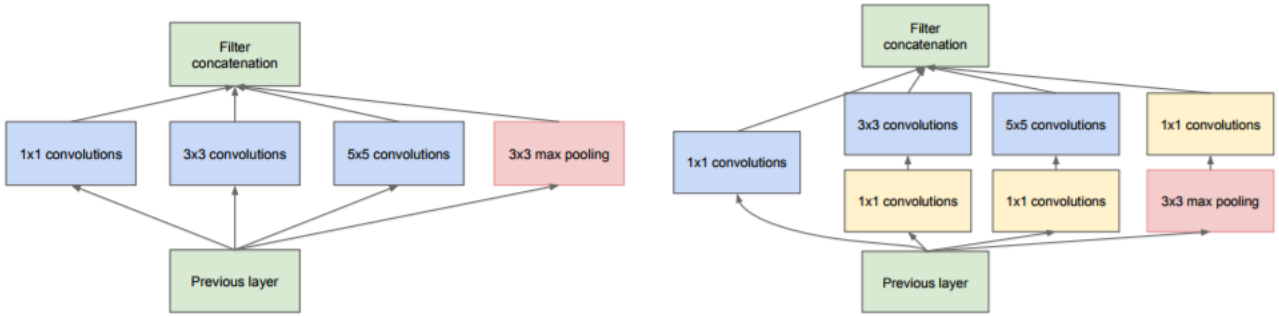


1x1 conv.

Google Network



Google Network



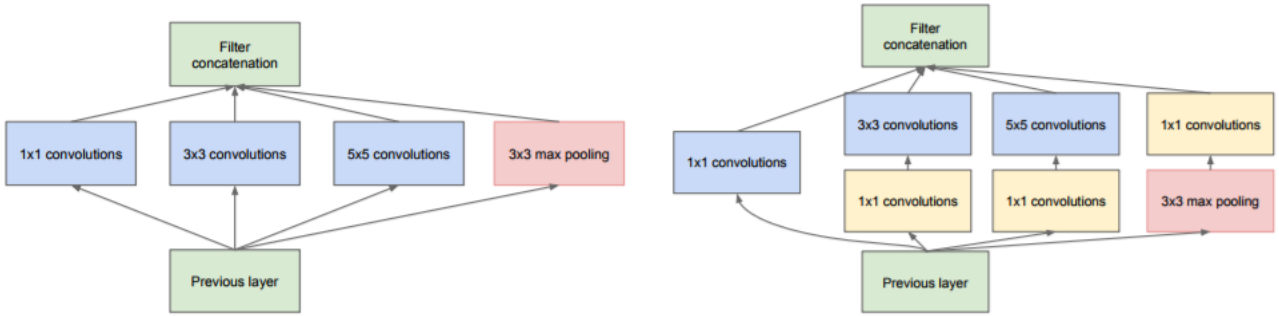
(a) Inception module, naïve version

(b) Inception module with dimension reductions

Figure 2: Inception module

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(a) Inception module, naïve version (b) Inception module with dimension reductions

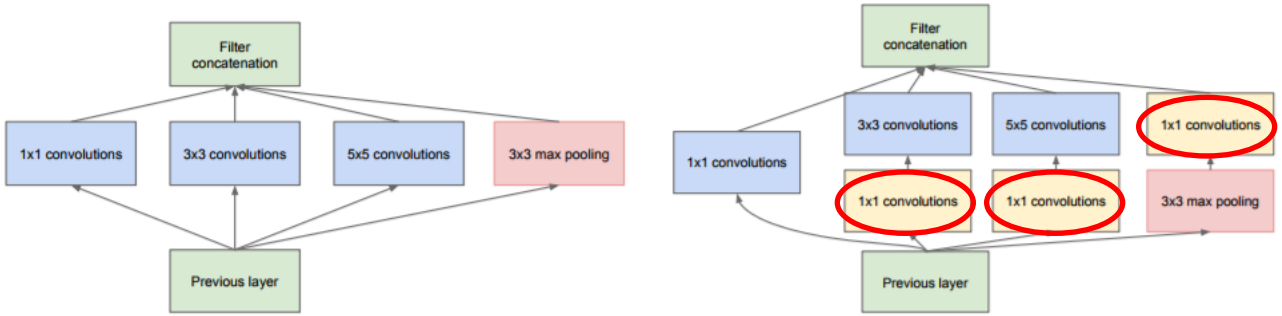
Figure 2: Inception module

하나의 input을 1x1, 3x3, 5x5, max pooling의 서로 다른 관점으로 보는 것

가까이, 중간, 멀리서 보는 거라고 생각하면 됨

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



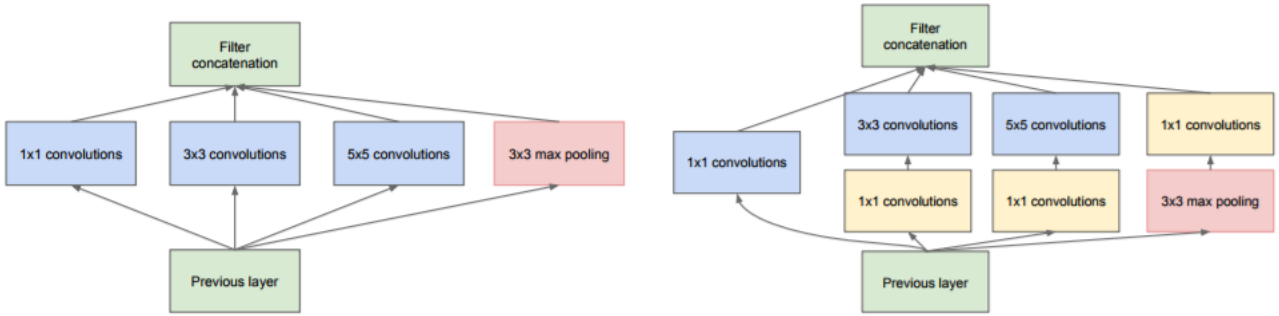
(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

하지만 무언가 개선할 필요가 생겼고 그렇기 때문에 1x1 convolution을 사용하게 됨

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(a) Inception module, naïve version (b) Inception module with dimension reductions

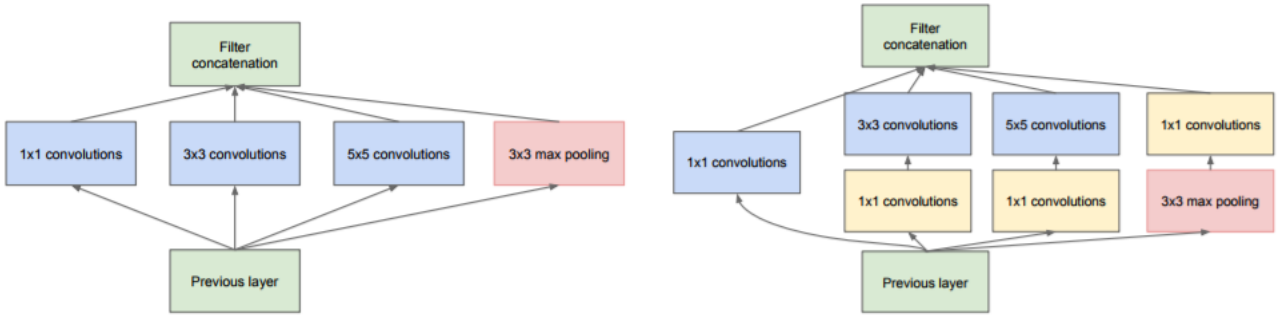
Figure 2: Inception module

하지만 무언가 개선할 필요가 생겼고 그렇기 때문에 1x1 convolution을 사용하게 됨

메모리!

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

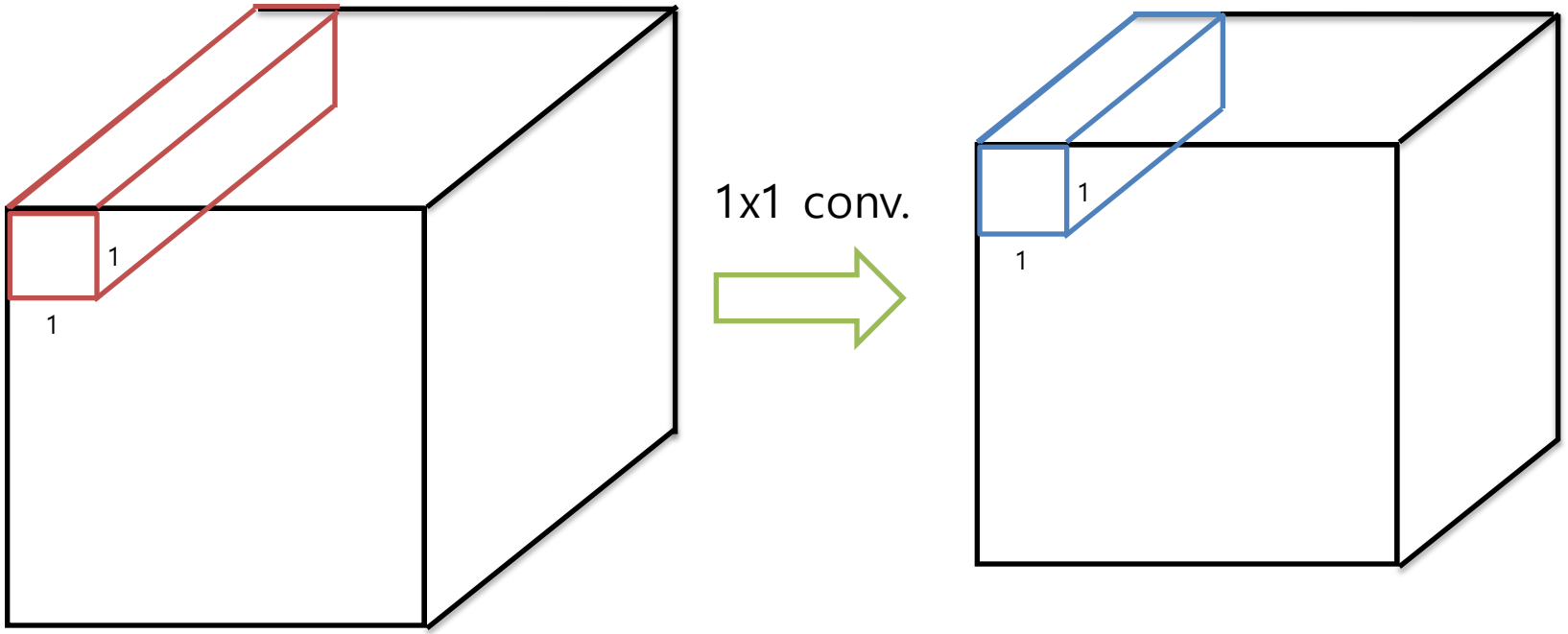
그렇다면 1x1 convolution은 어떻게 메모리 효율을 높이는가?

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

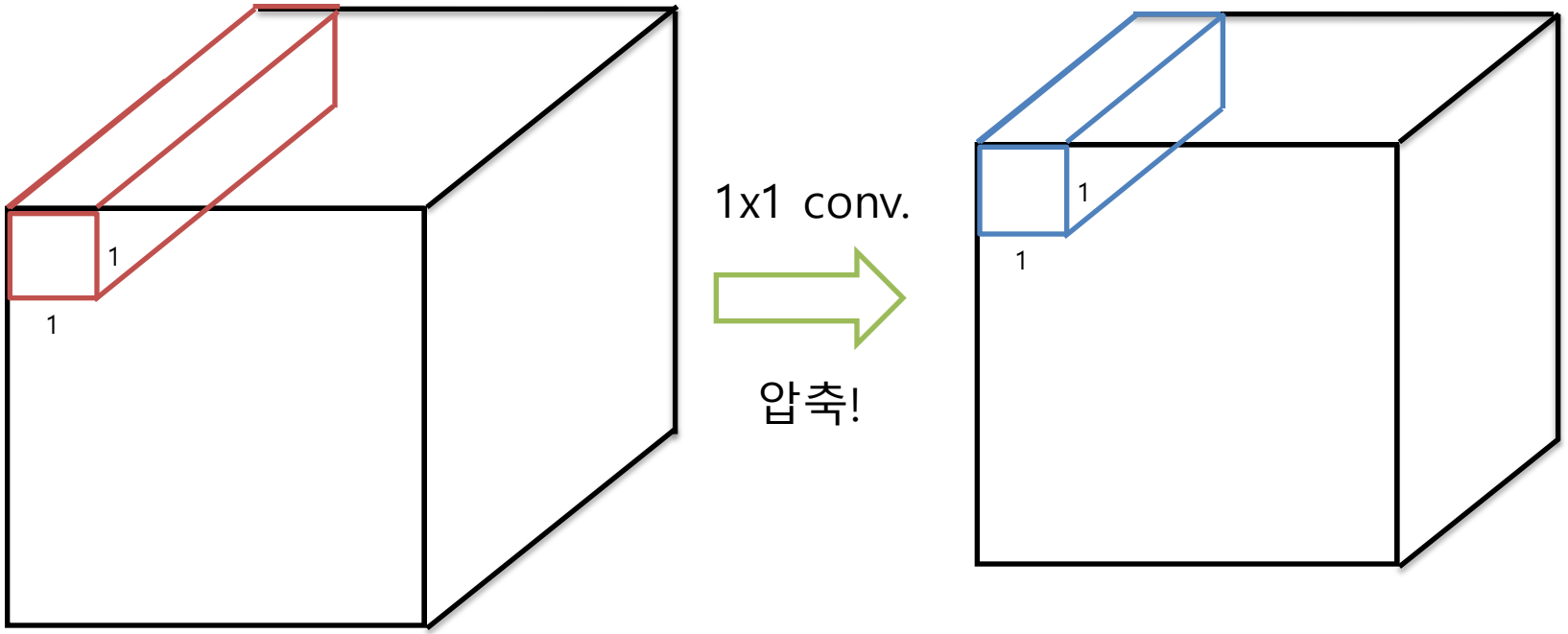


(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

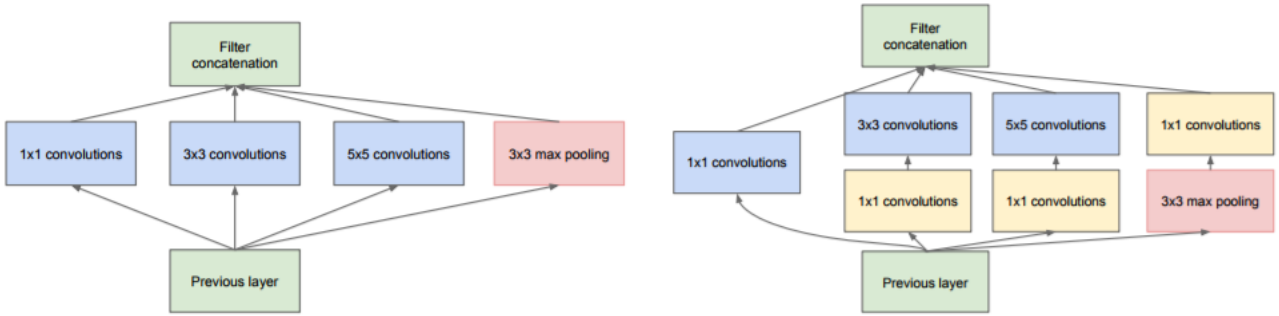
(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

1x1 convolution으로 필터의 수를 줄이고 거기에다 3x3, 5x5 conv. 연산을 함으로써 결과적으로는 (a), (b)가 같은 결과를 내더라도 필요한 연산량과 메모리는 더 적다.

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

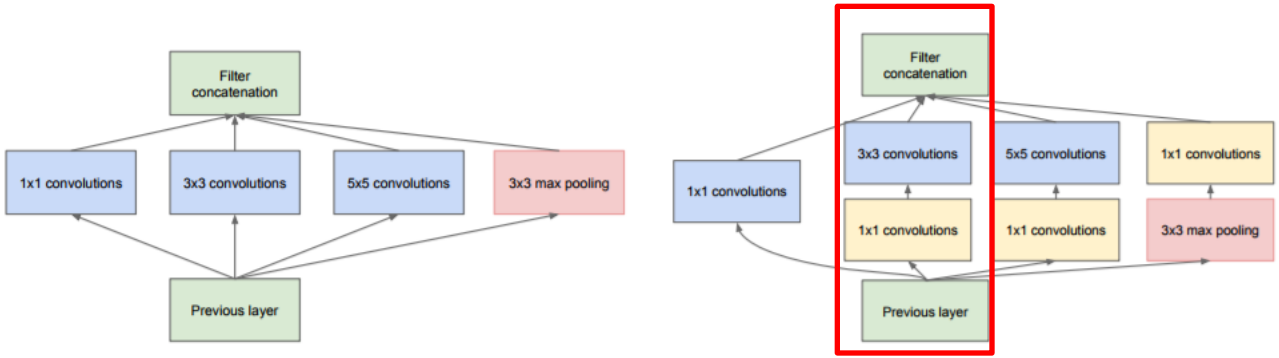
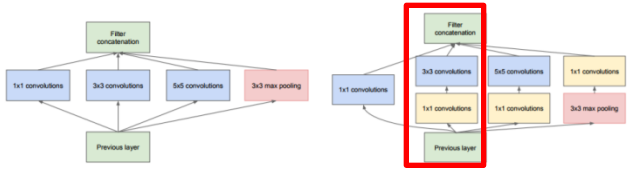


Figure 2: Inception module

1x1 convolution으로 필터의 수를 줄이고 거기에다 3x3, 5x5 conv. 연산을 함으로써 결과적으로는 (a), (b)가 같은 결과를 내더라도 필요한 연산량과 메모리는 더 적다.

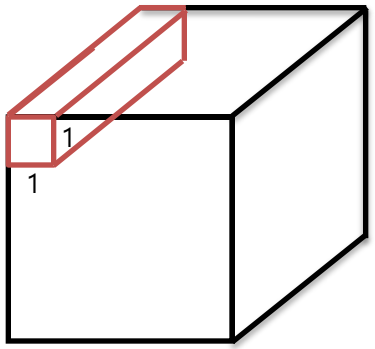
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



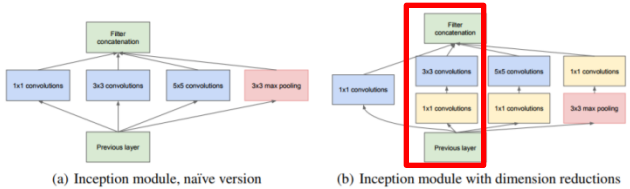
(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module



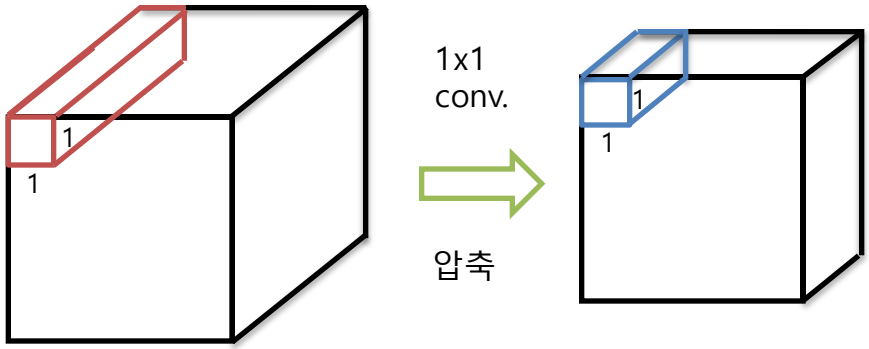
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



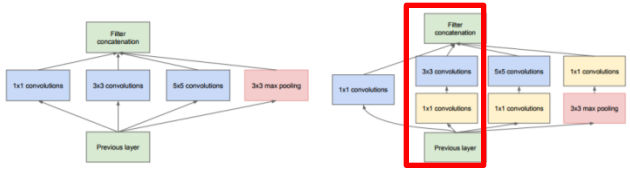
(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module



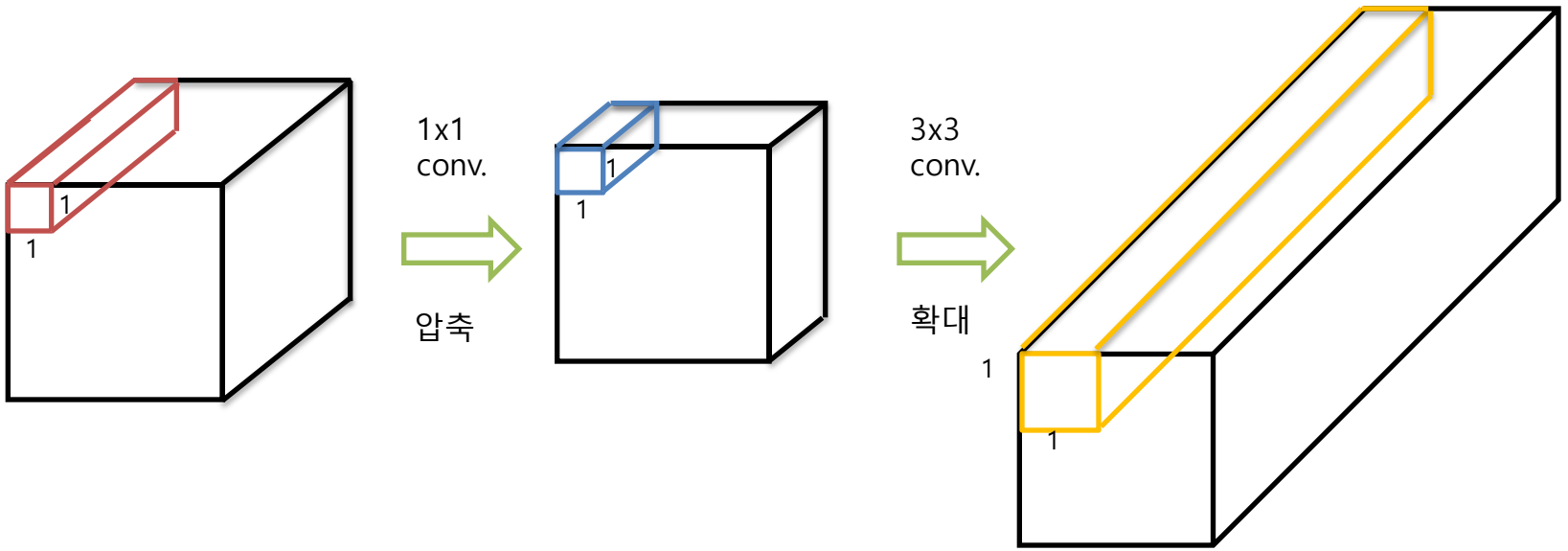
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

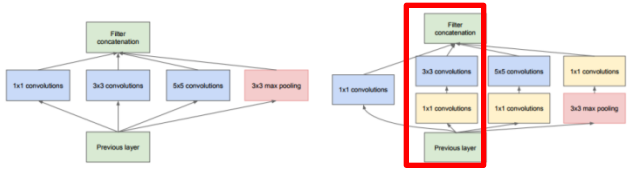
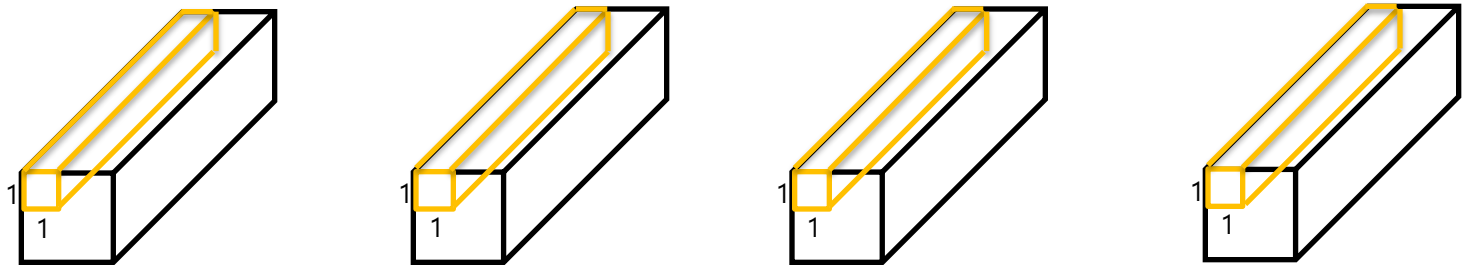


Figure 2: Inception module



1x1
conv.

5x5
conv.

1x1
conv.

3x3
conv.

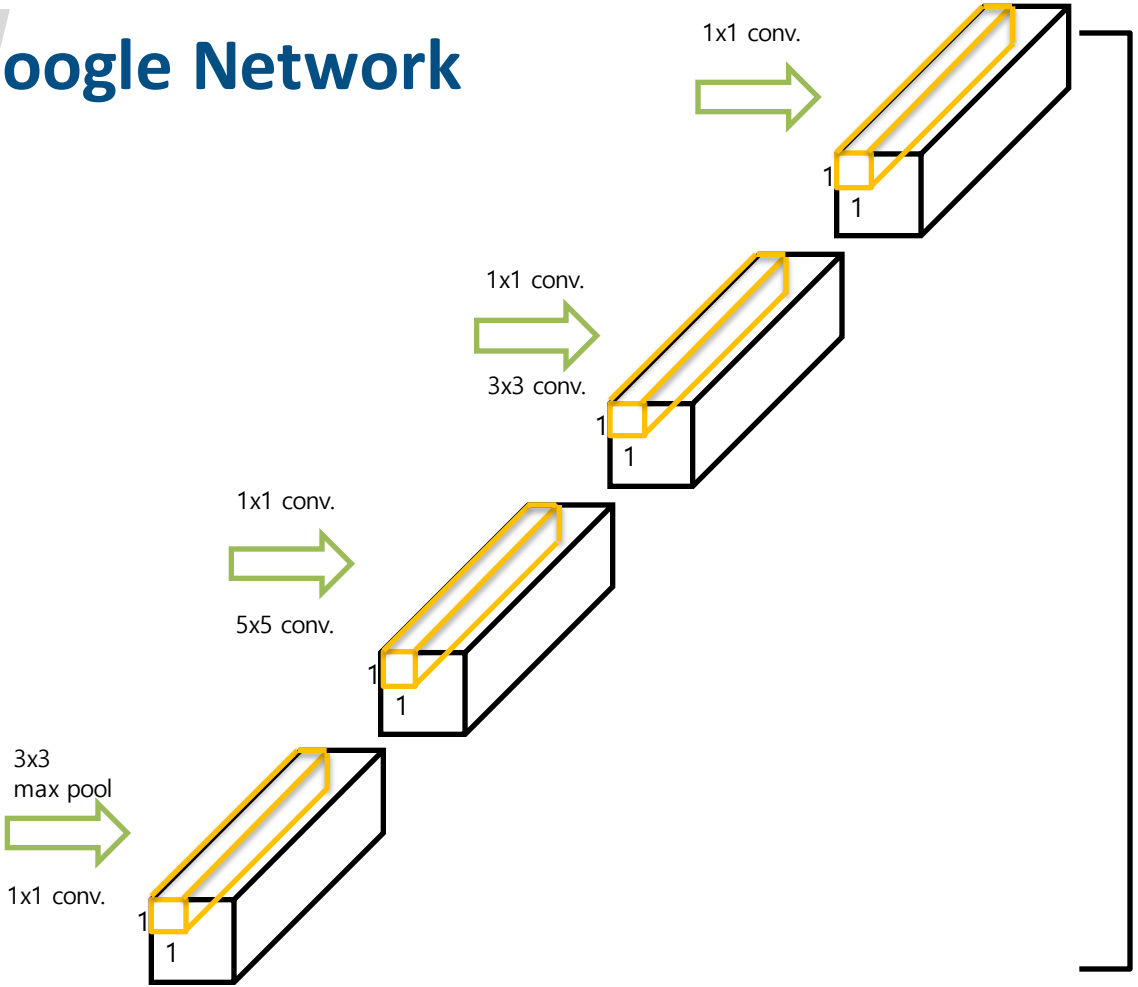
1x1
conv.

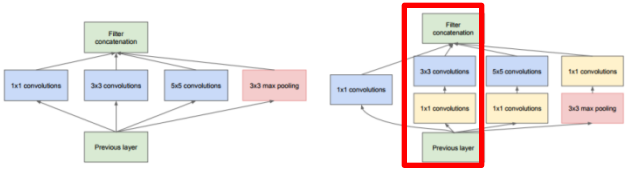
1x1
conv.

3x3
max pool

(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network





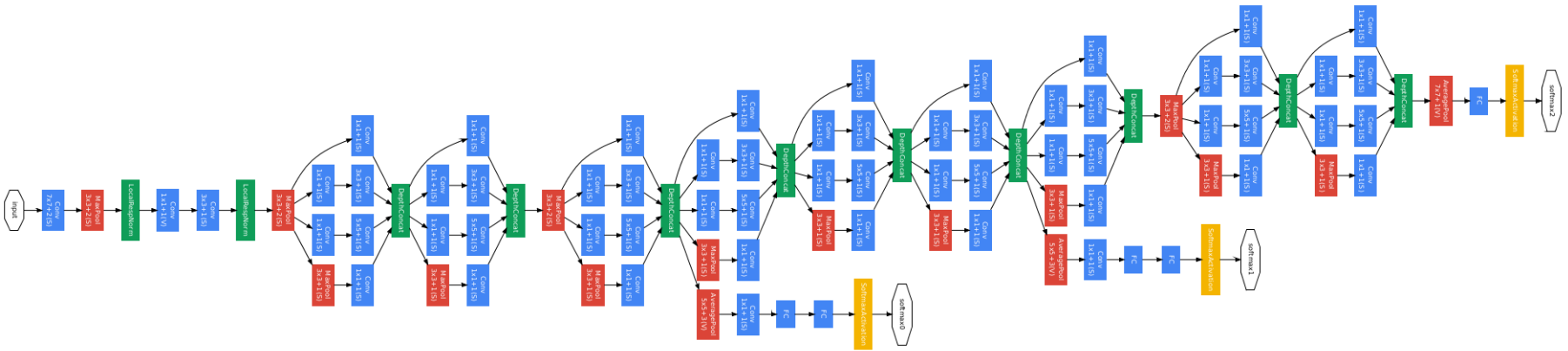
(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

concatenate!

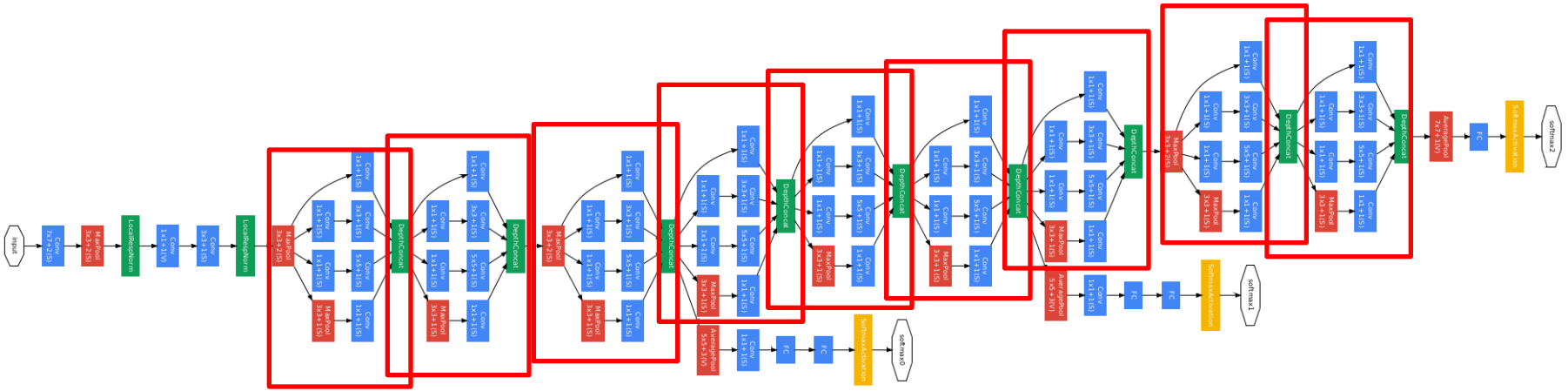
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

(출처:https://arxiv.org/pdf/1409.4842.pdf)

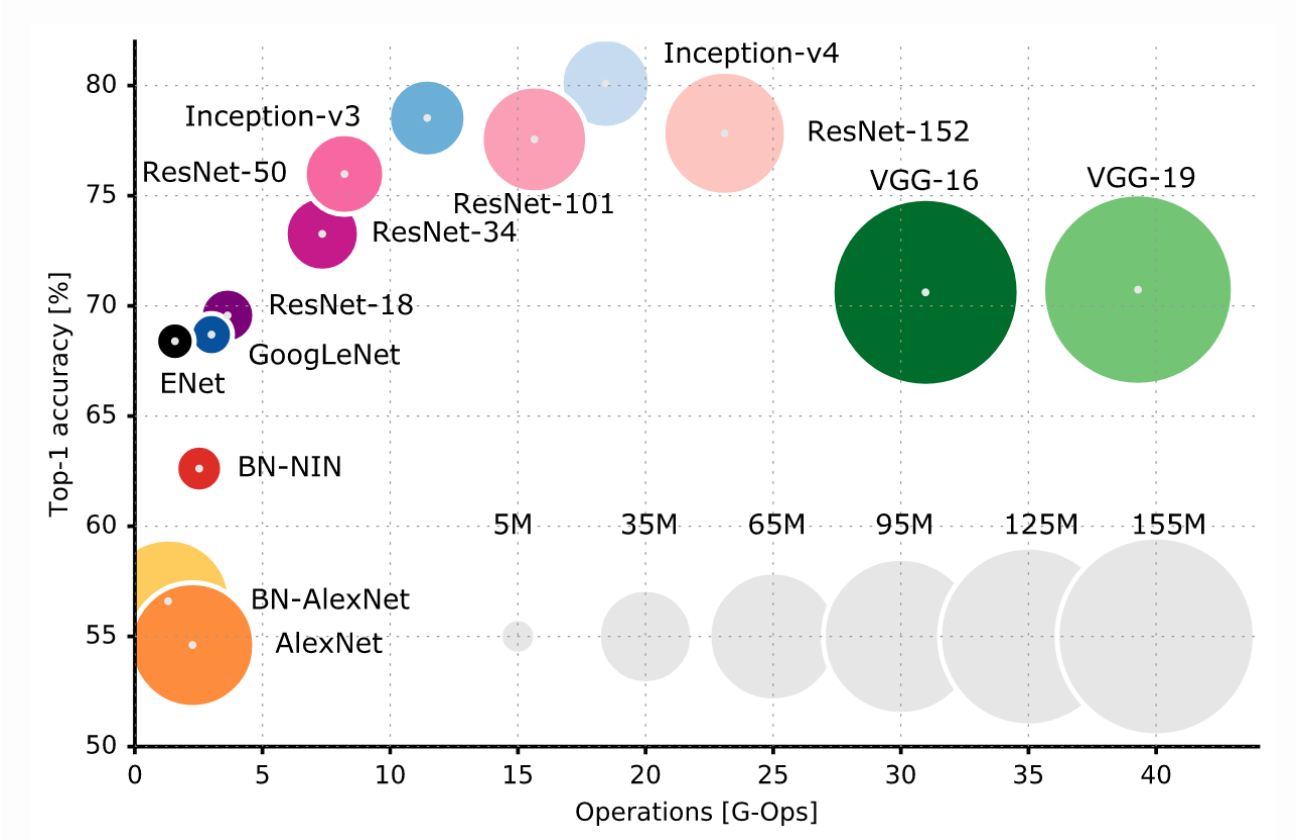
Google Network

Parameter만 보면 27MB

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

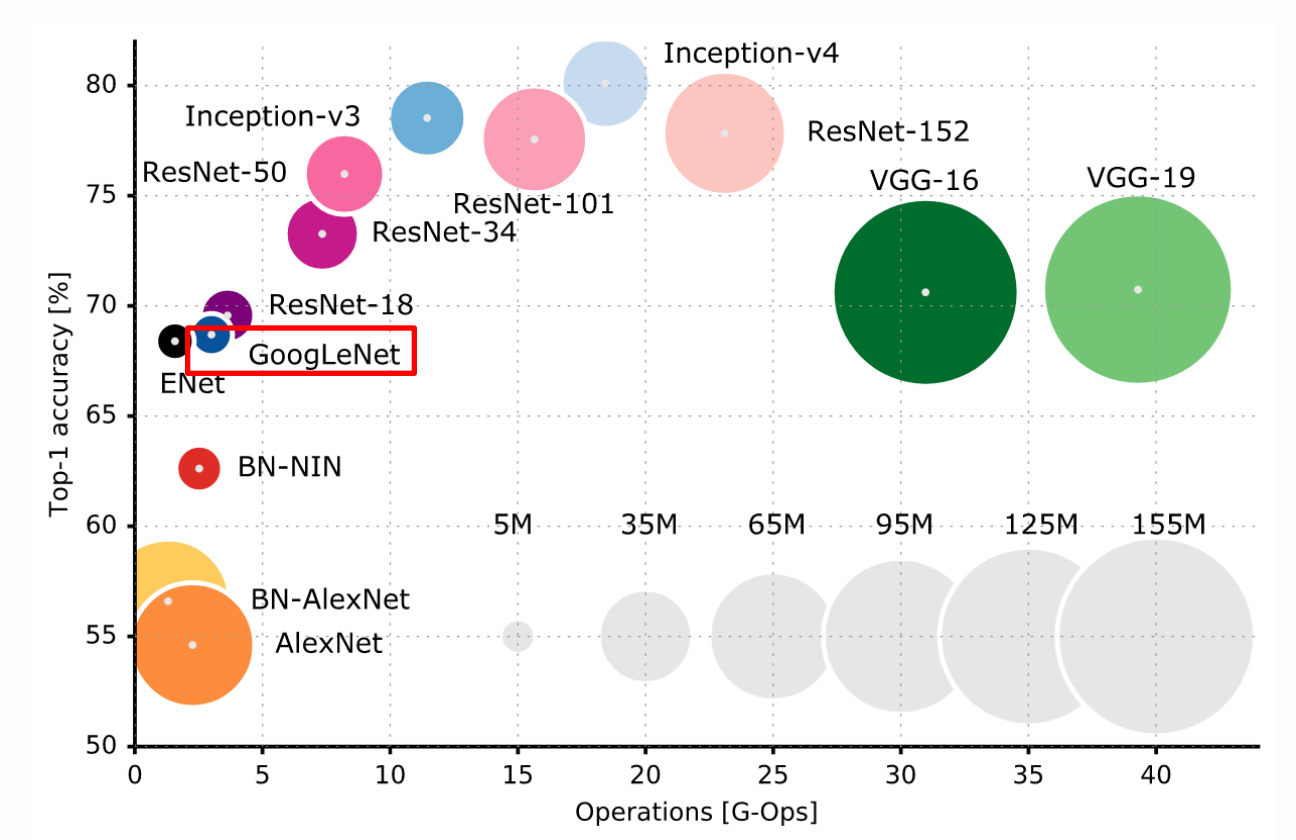
(출처:https://arxiv.org/pdf/1409.4842.pdf)

Google Network



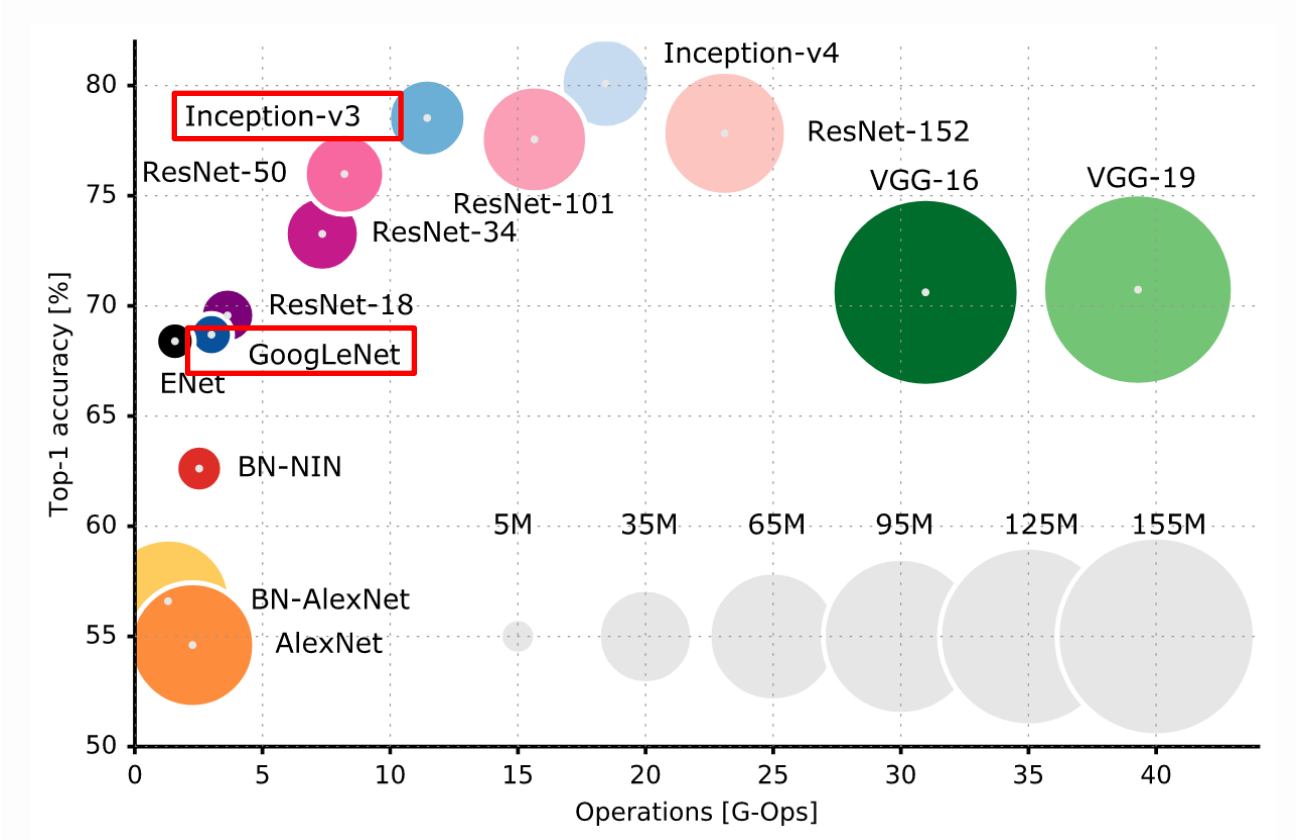
(출처: https://culurciello.github.io/tech/2016/06/20/training-enet.html)

Google Network



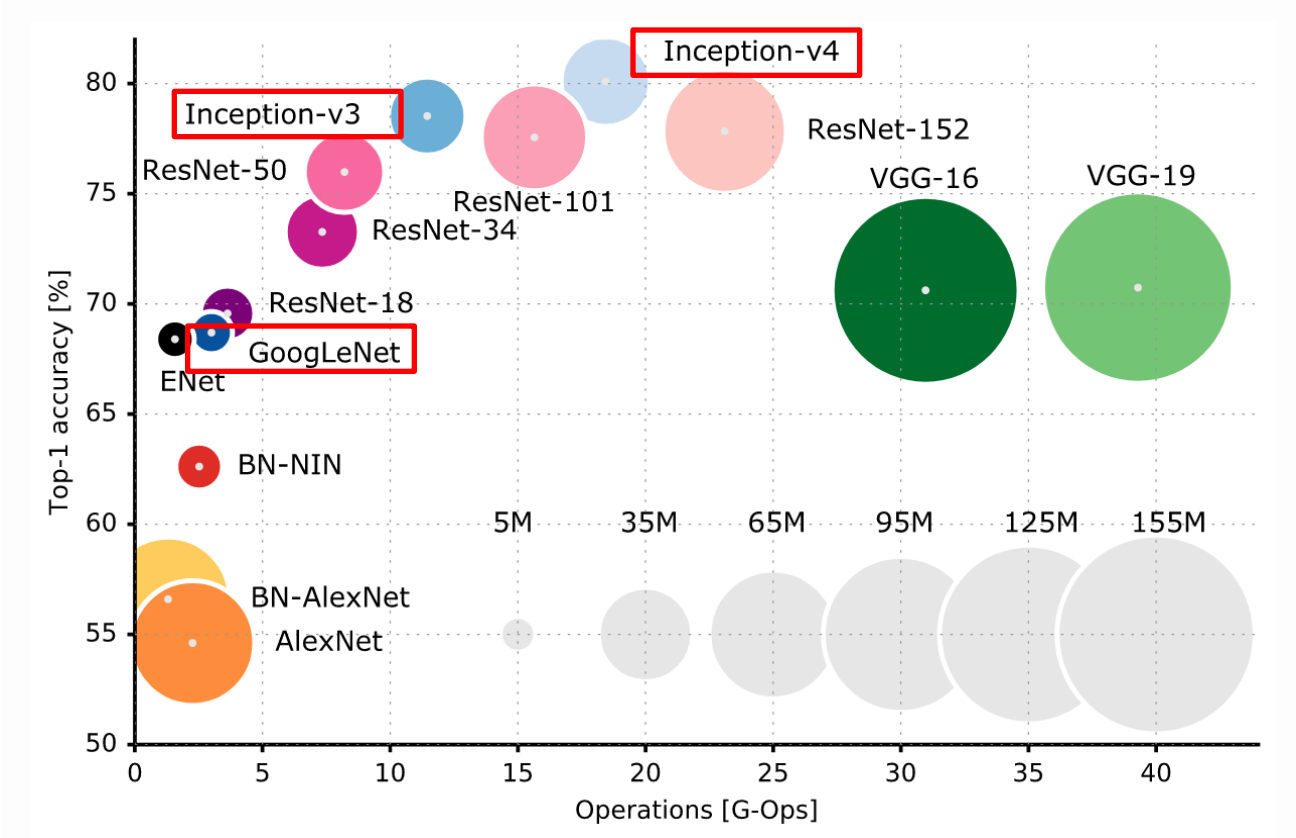
(출처: https://culurciello.github.io/tech/2016/06/20/training-enet.html)

Google Network



(출처: https://culurciello.github.io/tech/2016/06/20/training-enet.html)

Google Network



(출처: https://culurciello.github.io/tech/2016/06/20/training-enet.html)

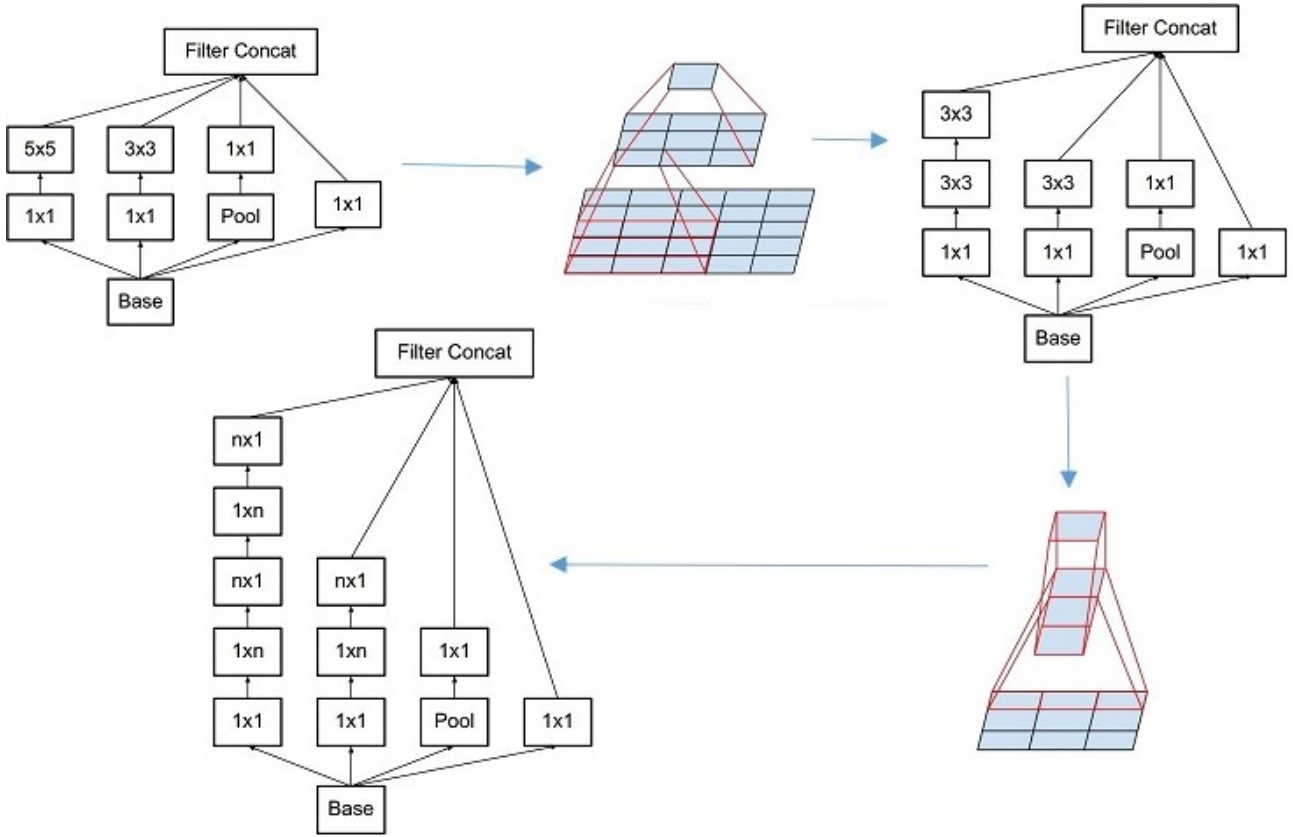
Google Network

Inception v1 -> v2
Convolution Factorization

(출처: https://arxiv.org/pdf/1512.00567v3.pdf)

Google Network

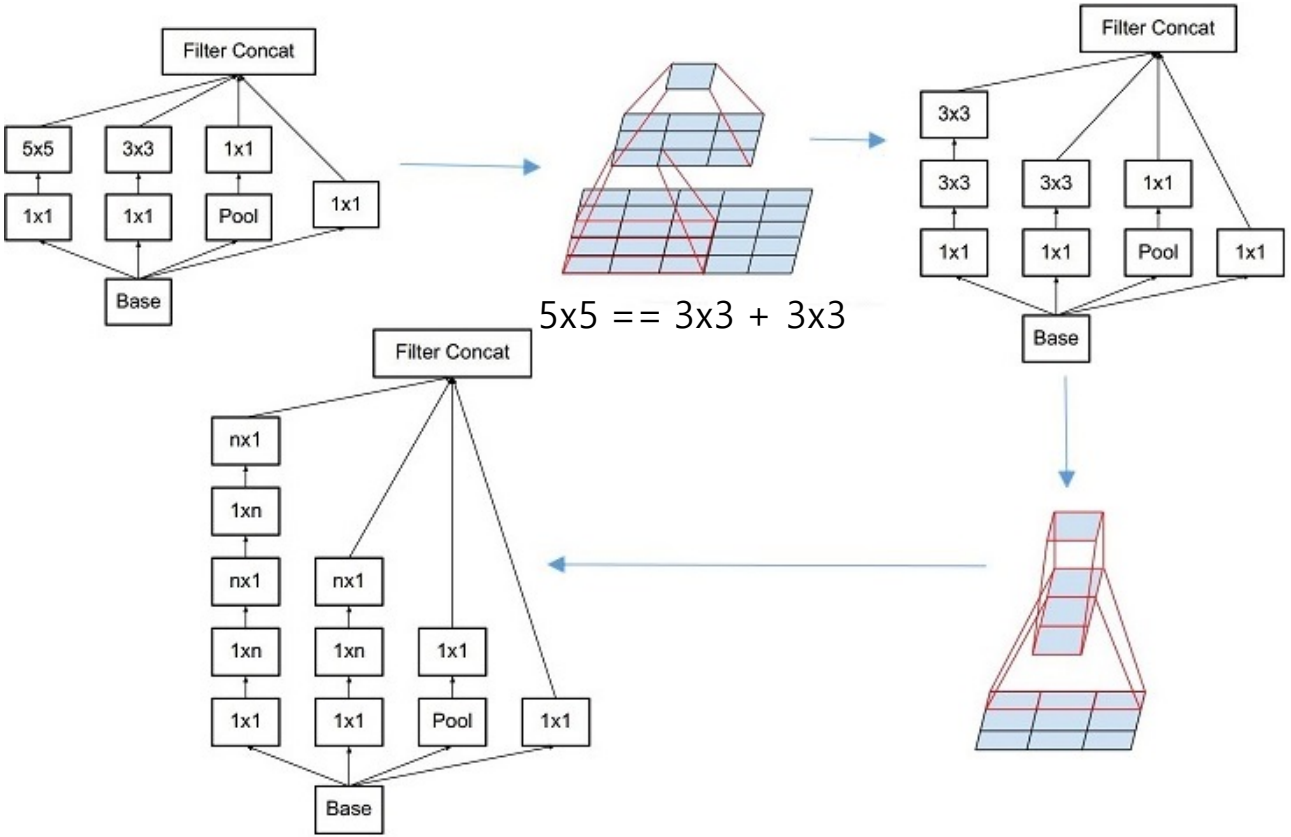
Inception v1 -> v2
Convolution Factorization



(출처: https://arxiv.org/pdf/1512.00567v3.pdf)

Google Network

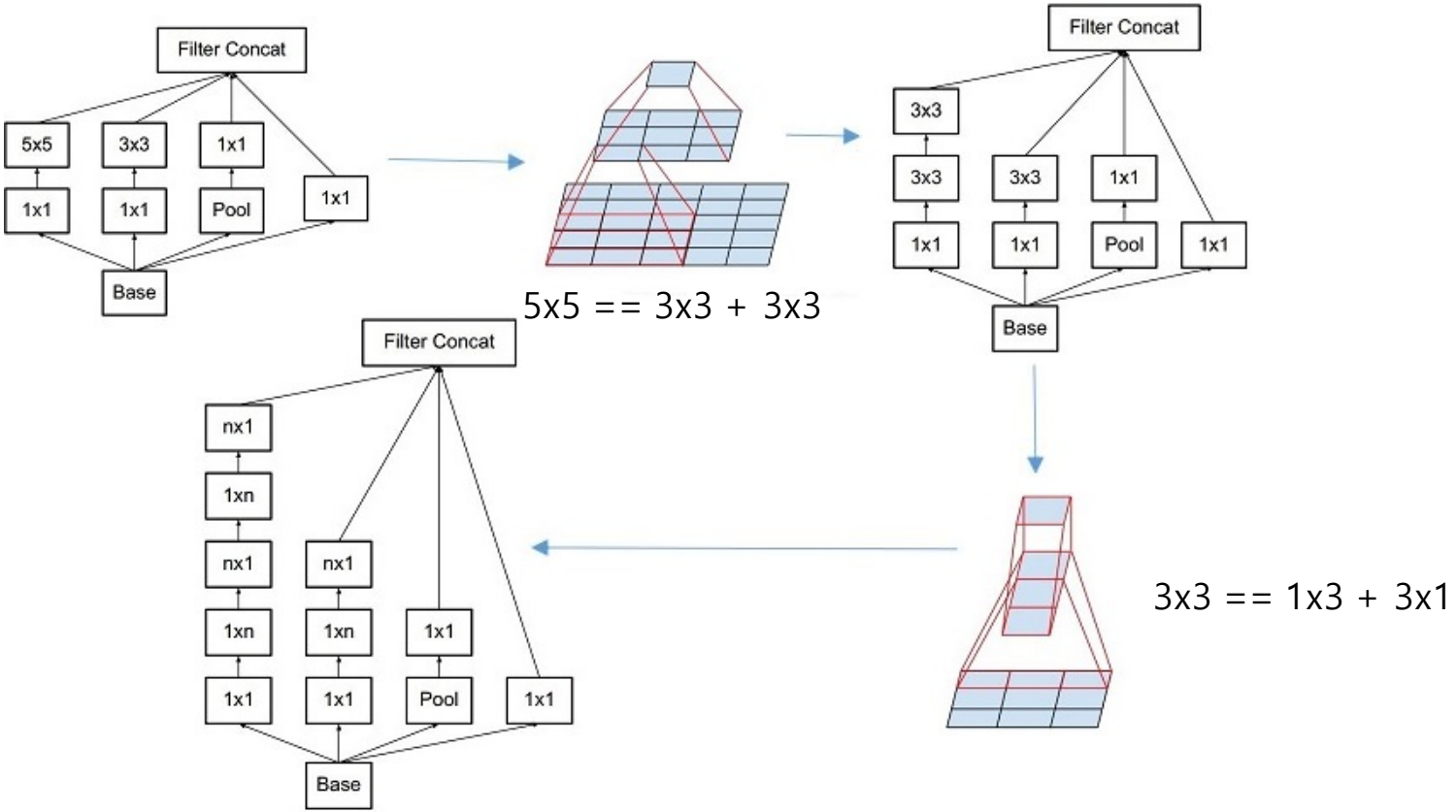
Inception v1 -> v2
Convolution Factorization



(출처: https://arxiv.org/pdf/1512.00567v3.pdf)

Google Network

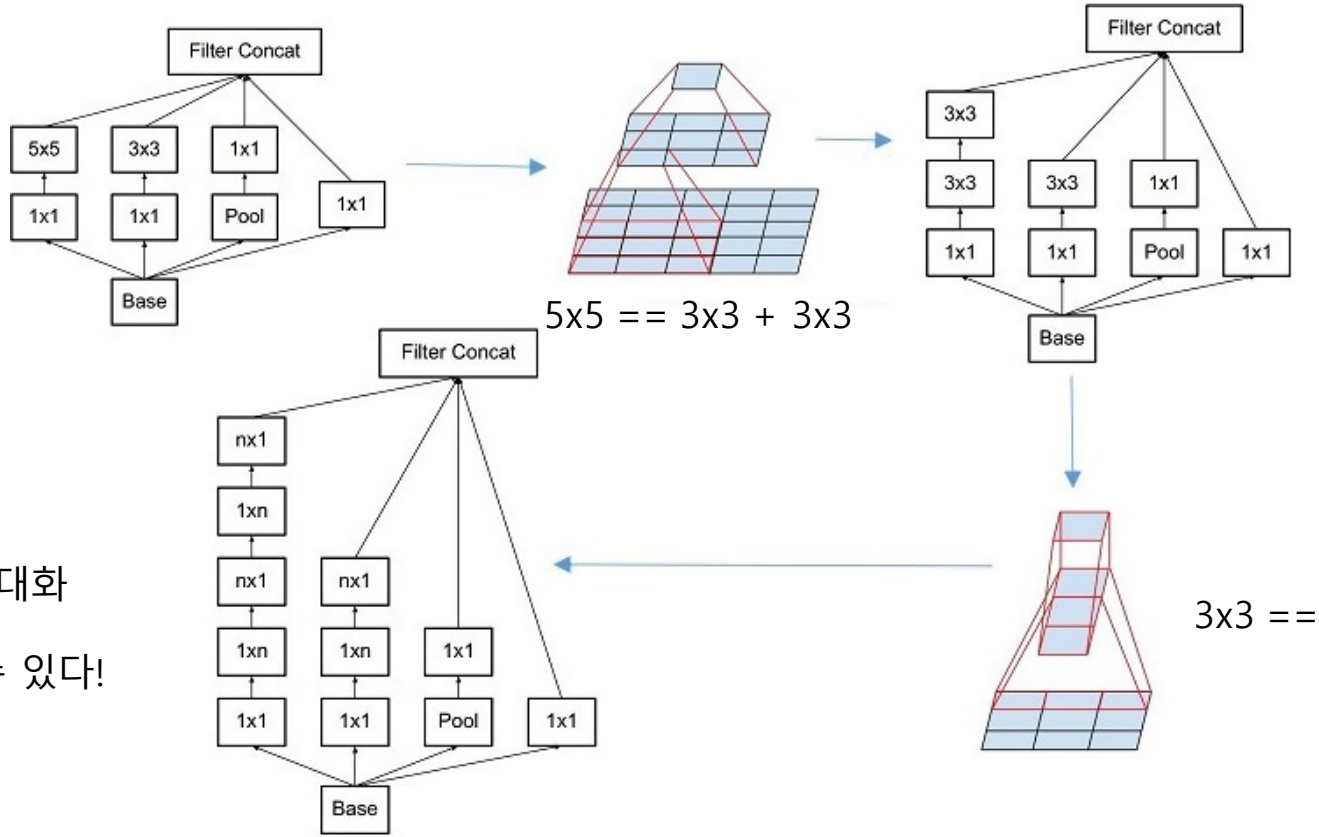
Inception v1 -> v2
Convolution Factorization



(출처: https://arxiv.org/pdf/1512.00567v3.pdf)

Google Network

Inception v1 -> v2
Convolution Factorization



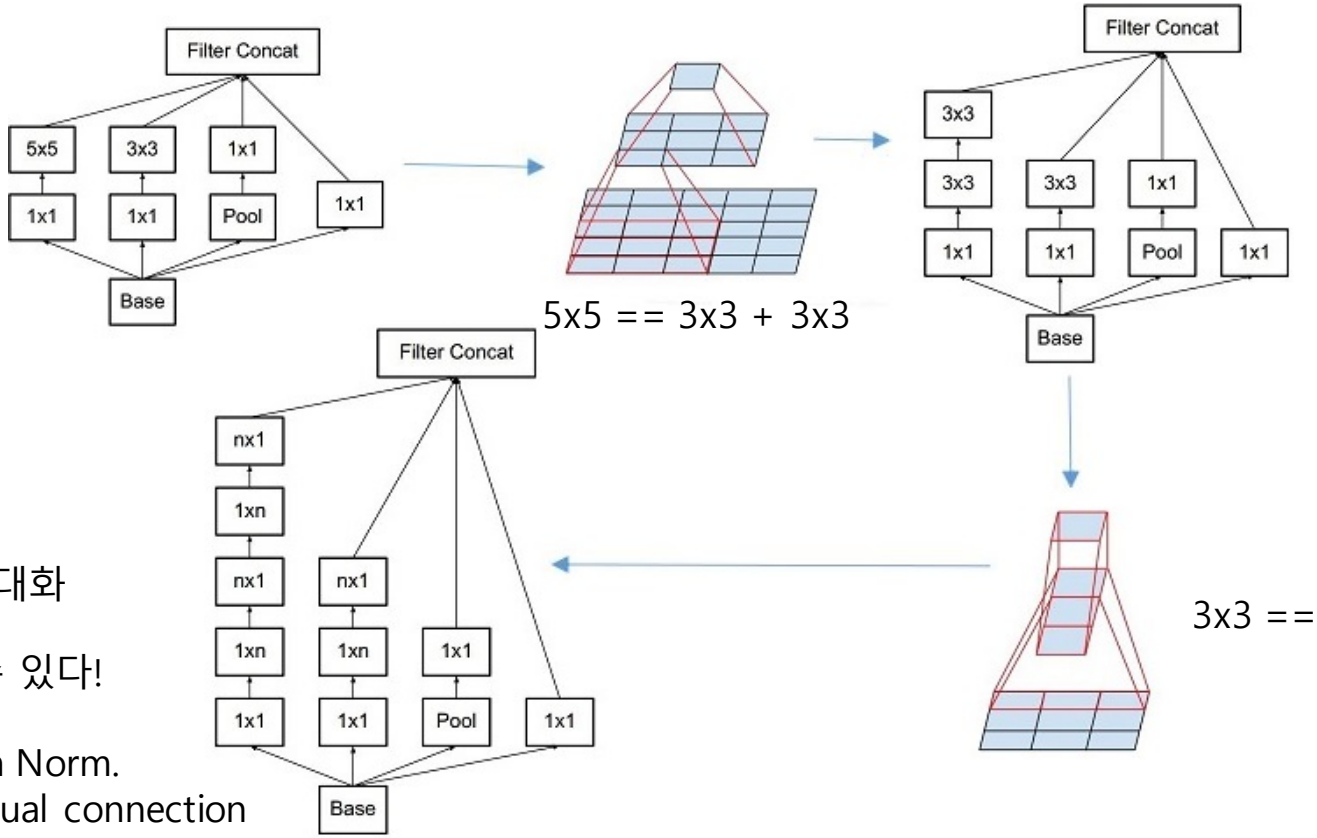
메모리 효율 극대화

더 깊이 쌓을 수 있다!

(출처: https://arxiv.org/pdf/1512.00567v3.pdf)

Google Network

Inception v1 -> v2
Convolution Factorization



메모리 효율 극대화

더 깊이 쌓을 수 있다!

v2->v3는 Batch Norm.
v3->v4는 Residual connection

(출처: https://arxiv.org/pdf/1512.00567v3.pdf)

Residual Network

2014년 모델들을 보고 든 생각

Residual Network

2014년 모델들을 보고 든 생각

보아하니 깊어지면 깊어질수록 성능이 올라가는데 지금보다 더 쌓으면 어떻게 될까?

Residual Network

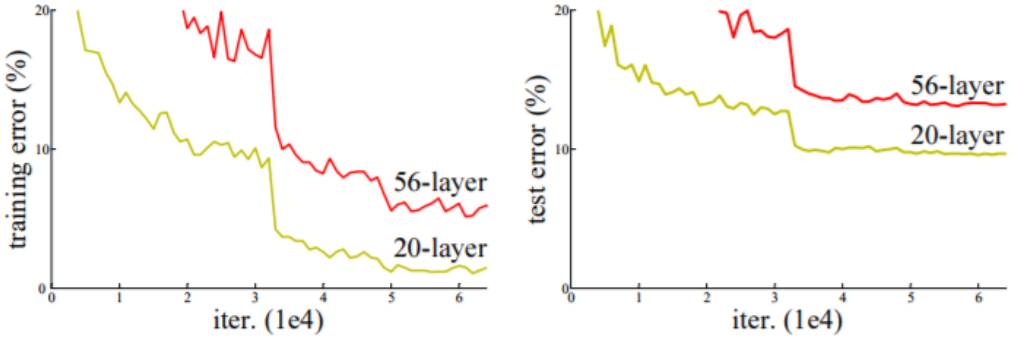


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

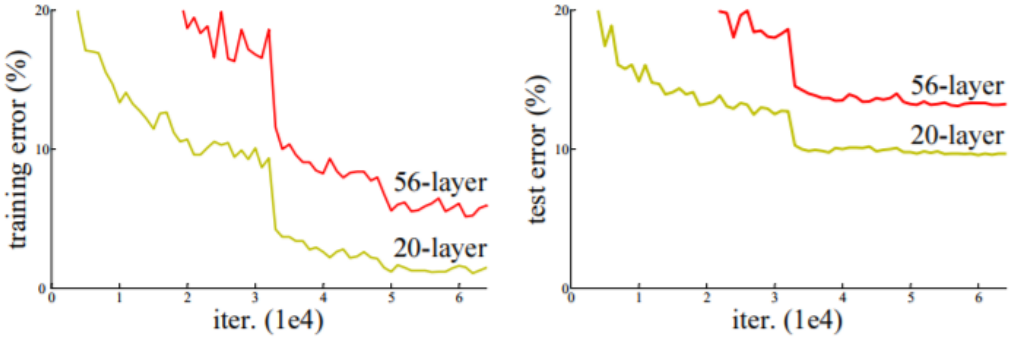


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

무작정 깊다고 더 좋아지는 건 아니다. 또한 training error와 test error 둘 다 높은 것은 아예 학습이 잘 안 되었음을 의미함

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

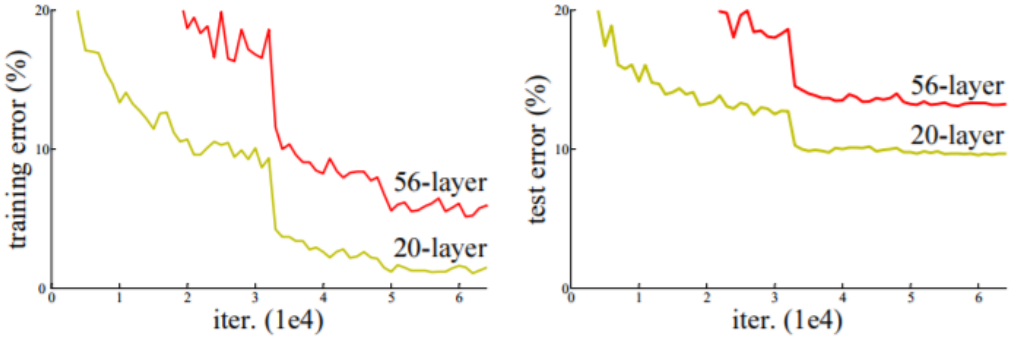


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

무작정 깊다고 더 좋아지는 건 아니다. 또한 training error와 test error 둘 다 높은 것은 아예 학습이 잘 안 되었음을 의미함 -> 무슨 방법이 없을까?

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

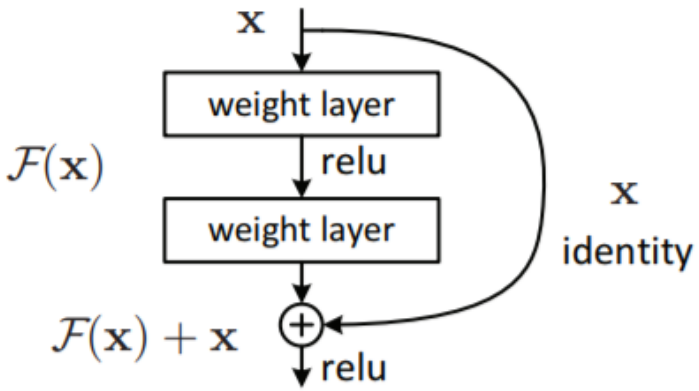


Figure 2. Residual learning: a building block.

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

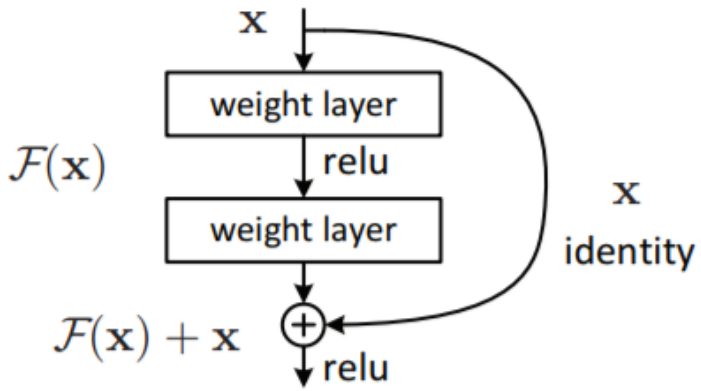


Figure 2. Residual learning: a building block.

ResNet 저자들이 생각해 낸 것이 Residual Learning
하지만 이전에 없던 완전 새로운 것은 아니고 identity mapping, shortcut connection 등의 비슷한 아이디어가 있었음.

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

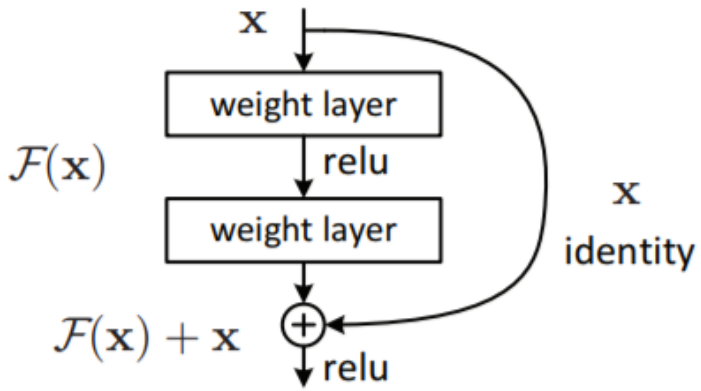


Figure 2. Residual learning: a building block.

ResNet 저자들이 생각해 낸 것이 Residual Learning
하지만 이전에 없던 완전 새로운 것은 아니고 identity mapping, shortcut connection 등의 비슷한 아이디어가 있었음.
-> 과연 좋을까?

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

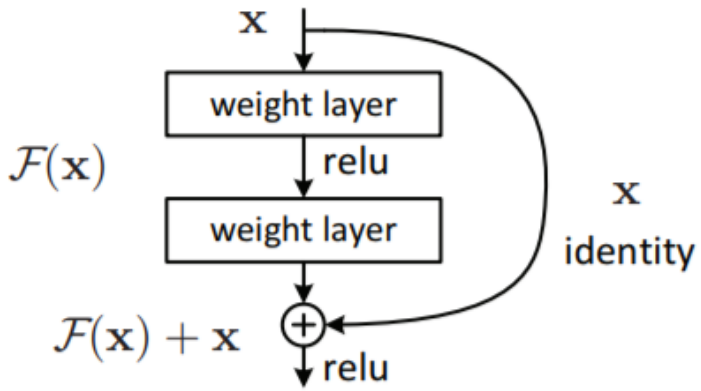
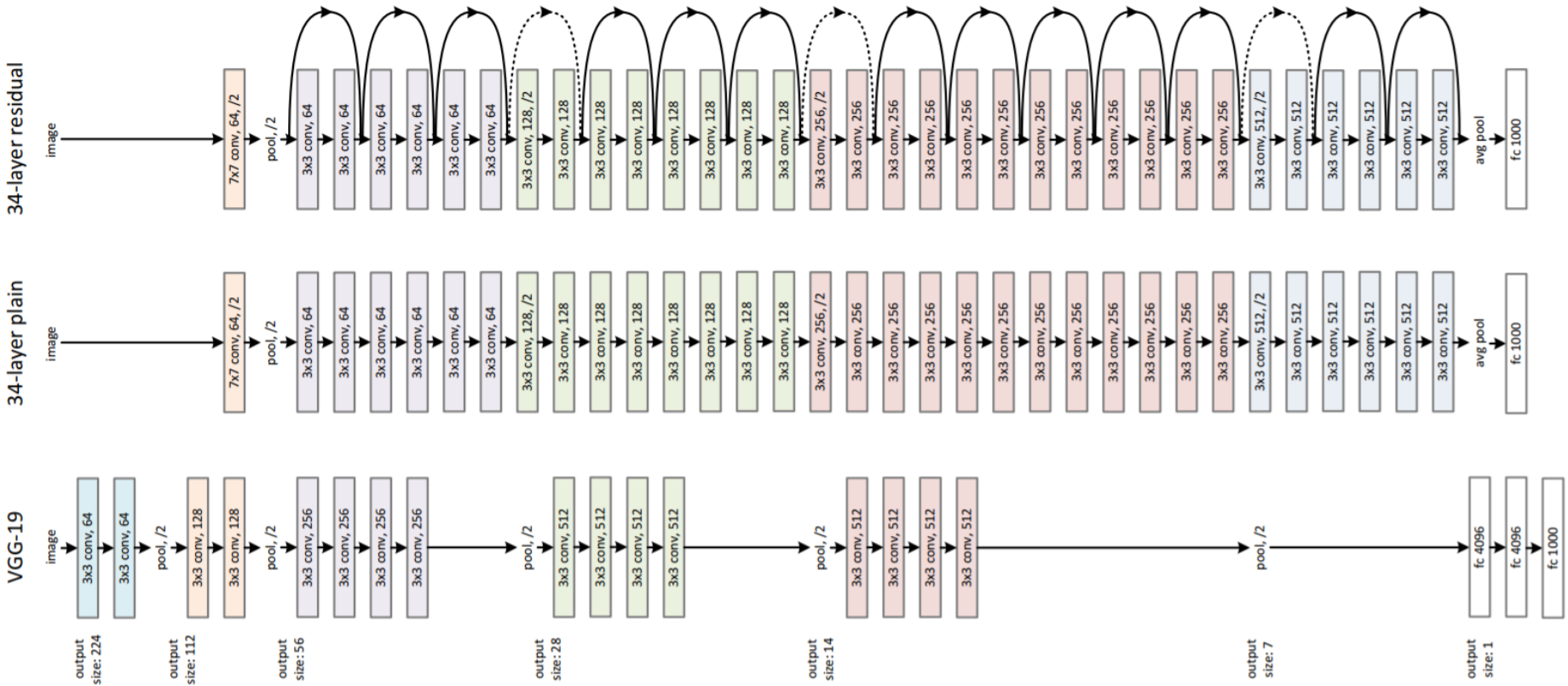


Figure 2. Residual learning: a building block.

직관적으로 생각해보면 초반에 뽑힌 단순한 형태들이 residual connection 없이 넘어가면 마지막 단에서는 복잡한 feature만을 가지고 판단해야 했다. 하지만 사실 단순하게 생각할수록 쉽게 풀리는 문제도 있듯이 앞 단의 비교적 간단한 feature 정보도 추가로 넘겨줌으로써 모델이 깊어지더라도 잘 판단할 수 있을 것

(출처: https://arxiv.org/abs/1512.03385)

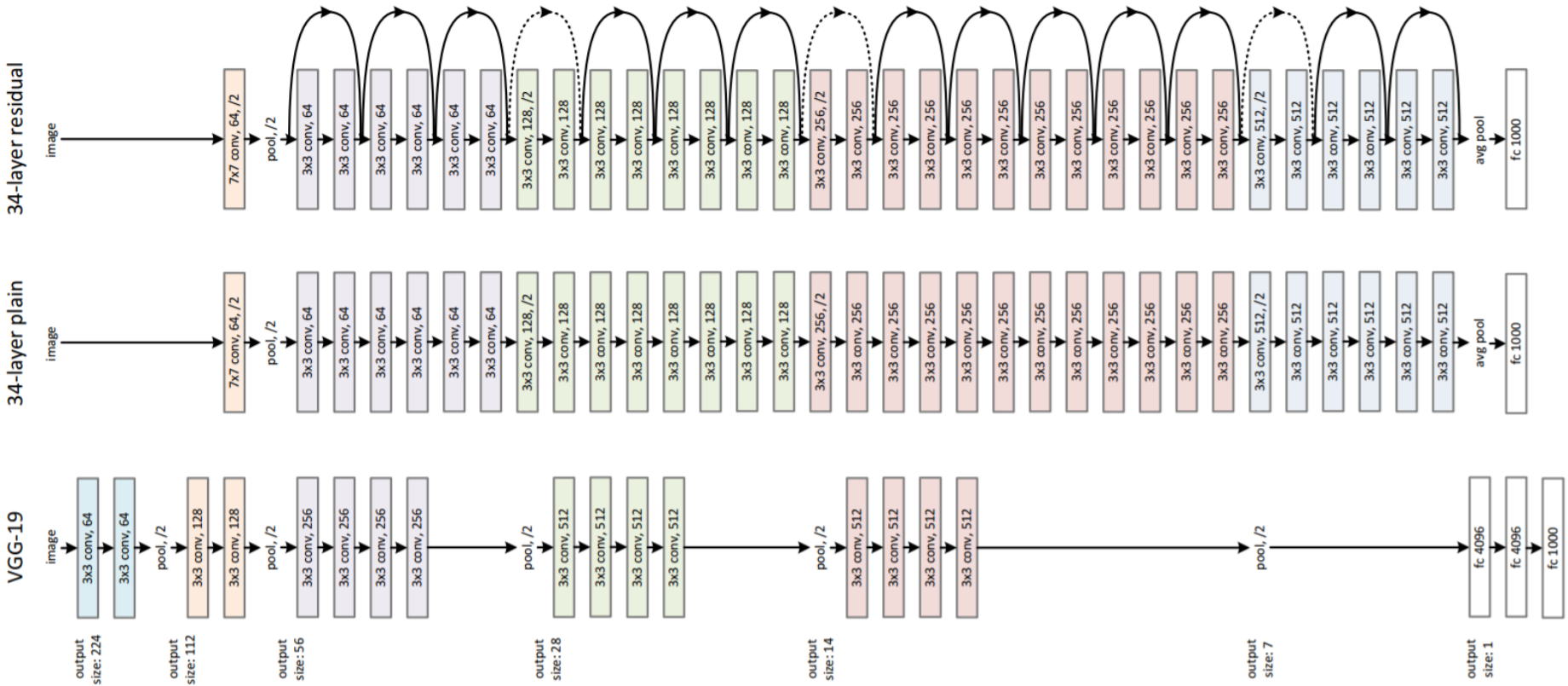
Residual Network



(출처: https://arxiv.org/abs/1512.03385)

Residual Network

점선은 convolution 연산 stride 2로 이미지가 작아졌을 때 맞춰주는 residual connection



(출처: https://arxiv.org/abs/1512.03385)

Residual Network

Plain Network

Residual Network

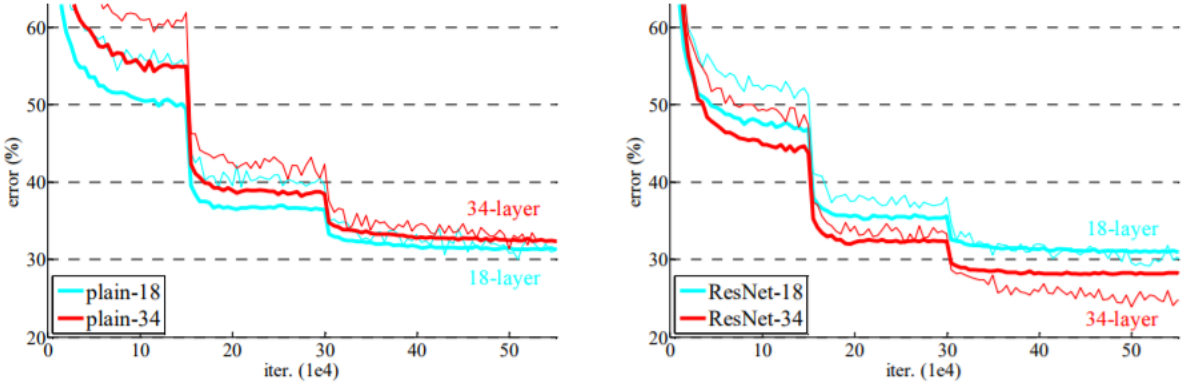


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

Plain Network

Residual Network

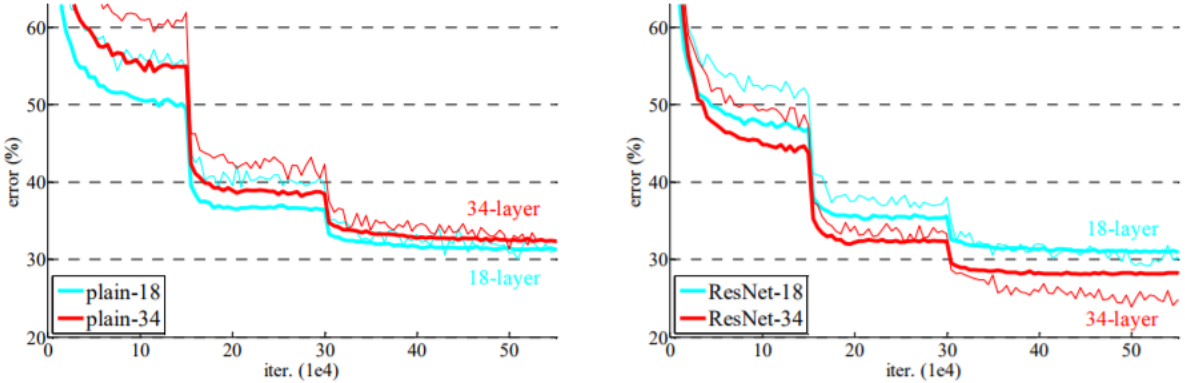


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

파라미터 수는 같음
18 layer는 별 차이가 없었지만
34 layer는 눈에 띄게 성능이 더 좋다

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

Plain Network

Residual Network

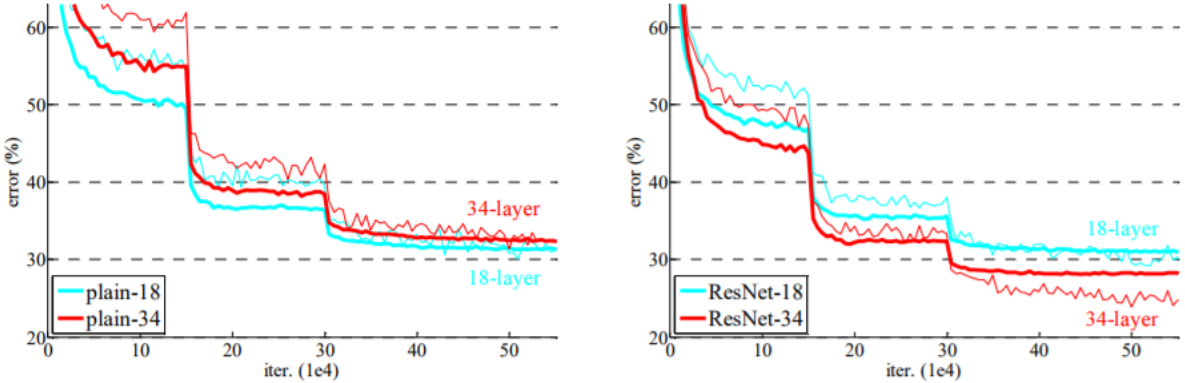


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

파라미터 수는 같음
18 layer는 별 차이가 없었지만
34 layer는 눈에 띄게 성능이 더 좋다
-> 그렇다면 이번엔 어디까지 깊어질 수 있을까?

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

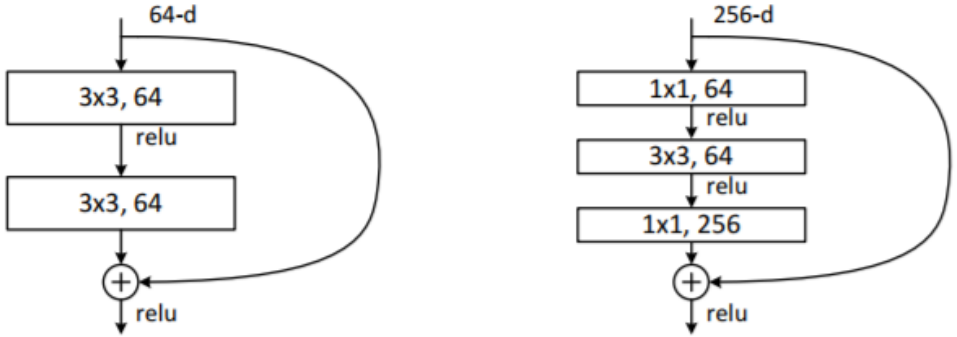


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

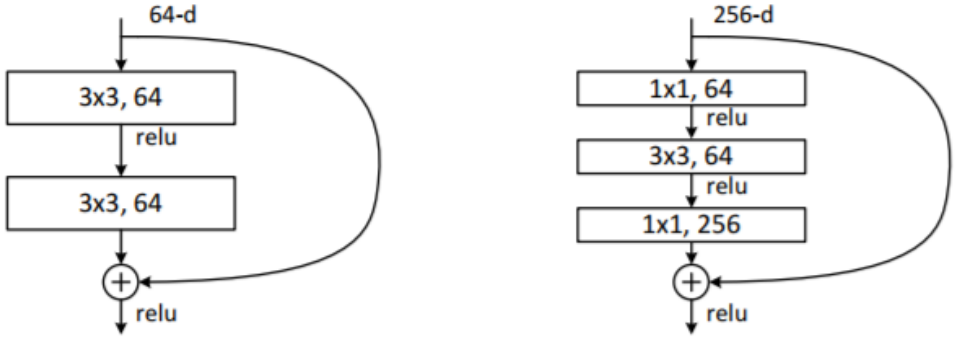


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

ResNet 역시 깊어질수록 연산량과 메모리 문제 때문에 1x1 convolution을 도입하여 압축한 상태로 feature를 뽑고 다시 확장하는 "Bottleneck"을 도입함

(출처: https://arxiv.org/abs/1512.03385)

Residual Network

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

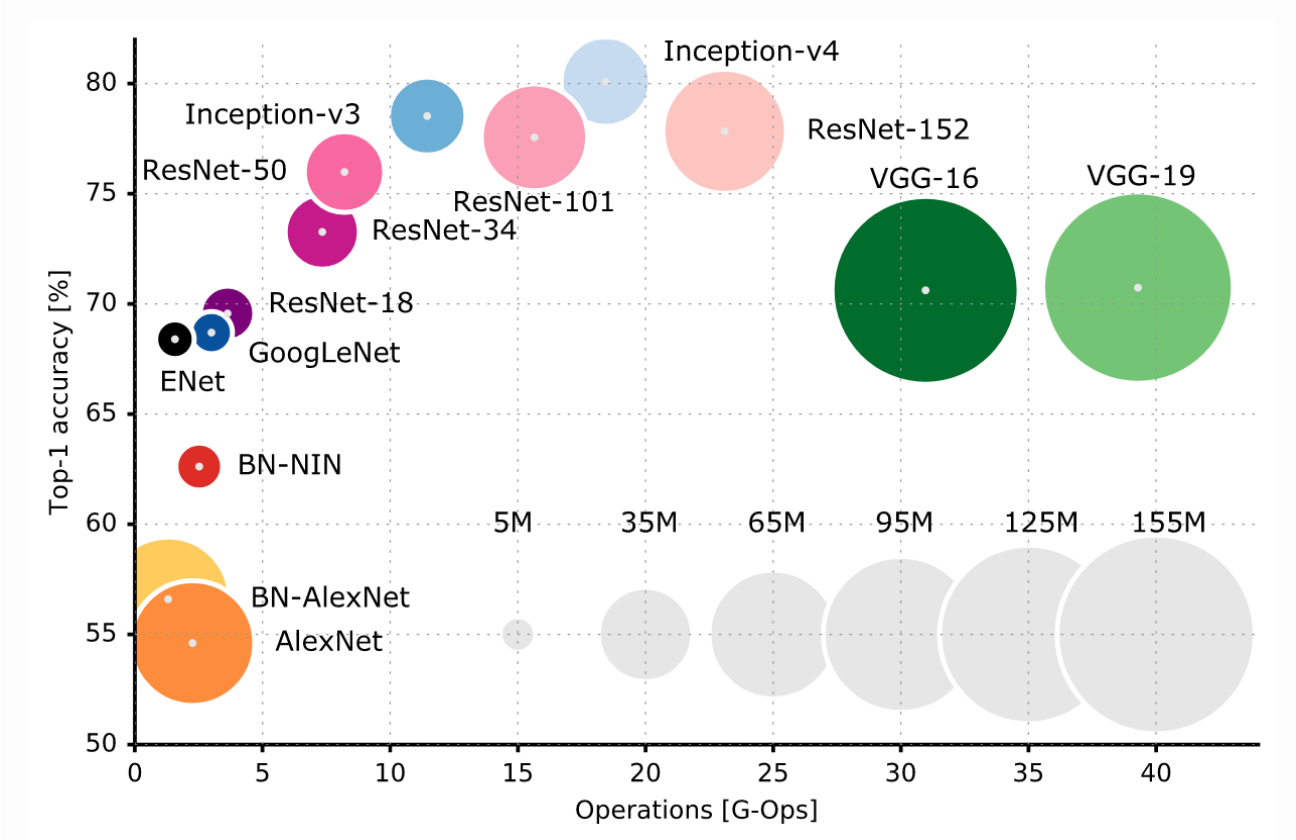
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

다른 모델들을 제치고 ImageNet Challenge 1위

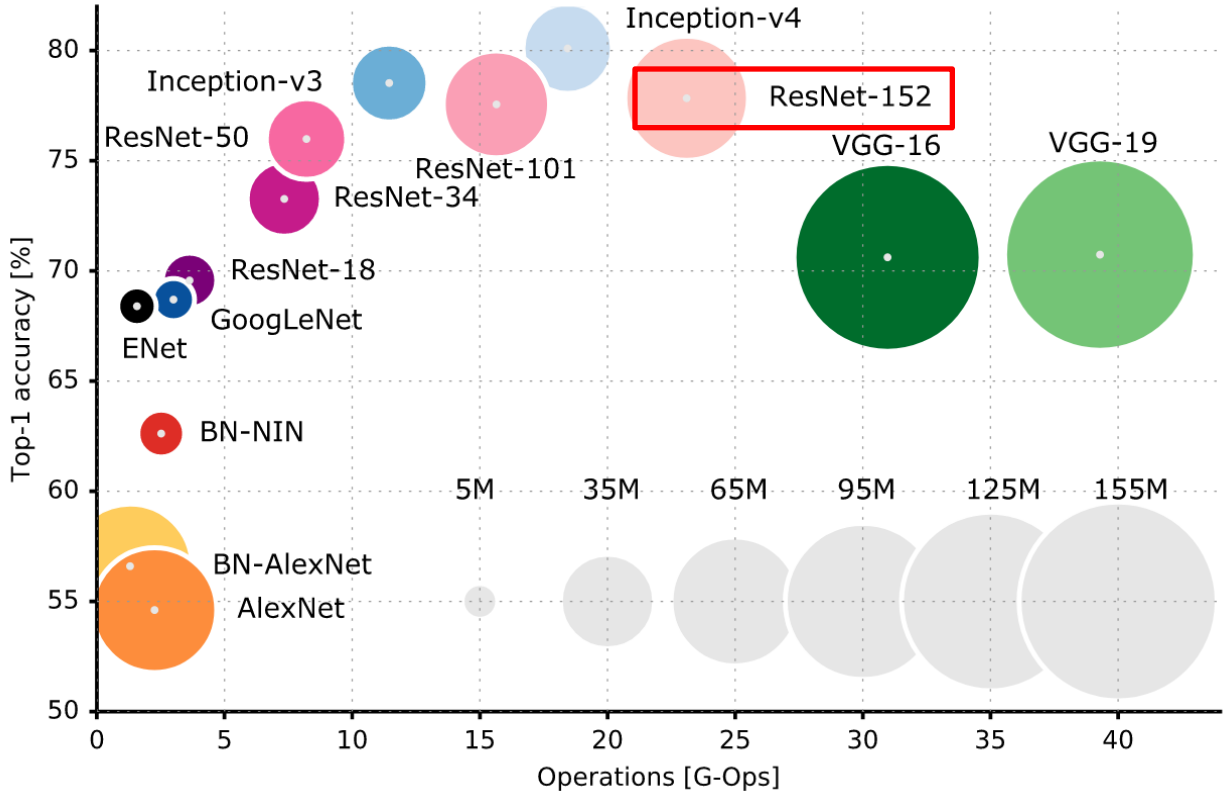
(출처: https://arxiv.org/abs/1512.03385)

Residual Network



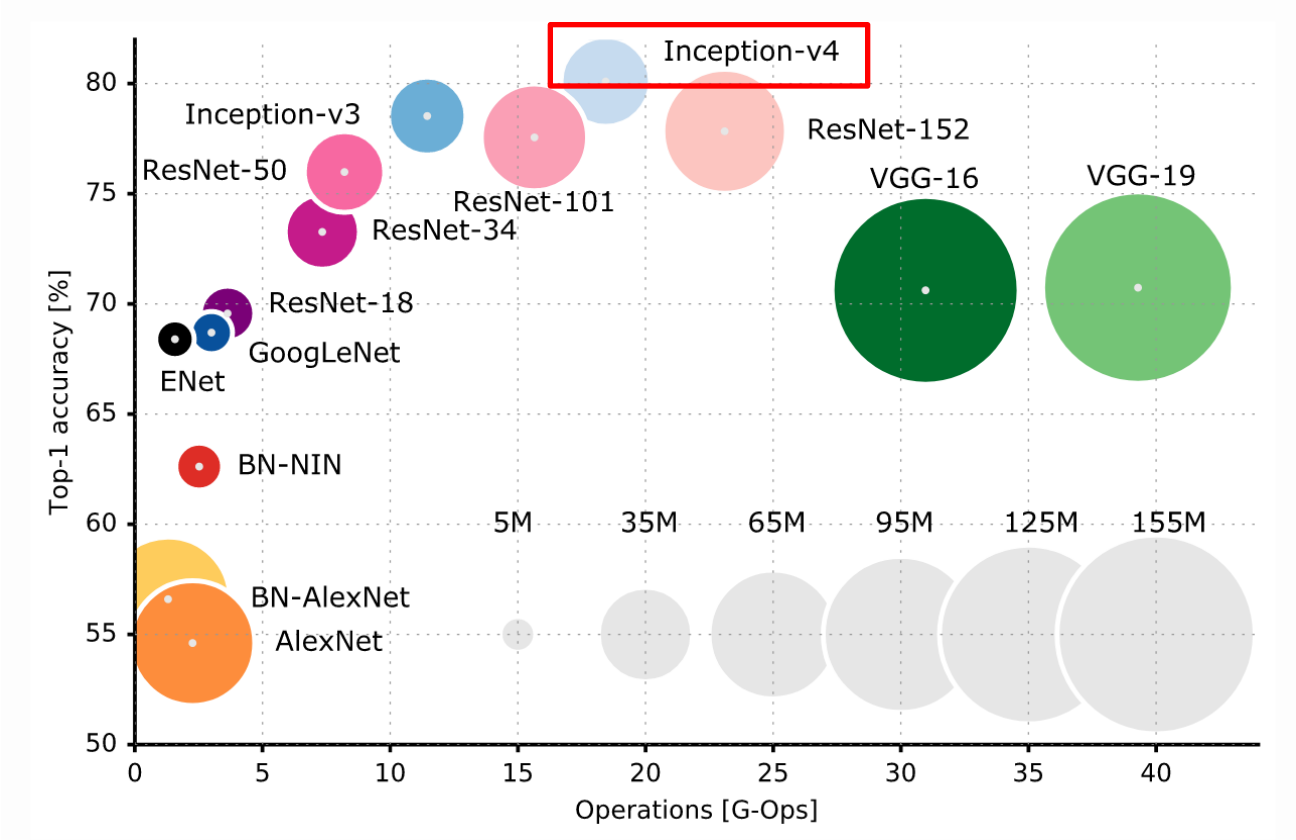
(출처: https://arxiv.org/abs/1512.03385)

Residual Network



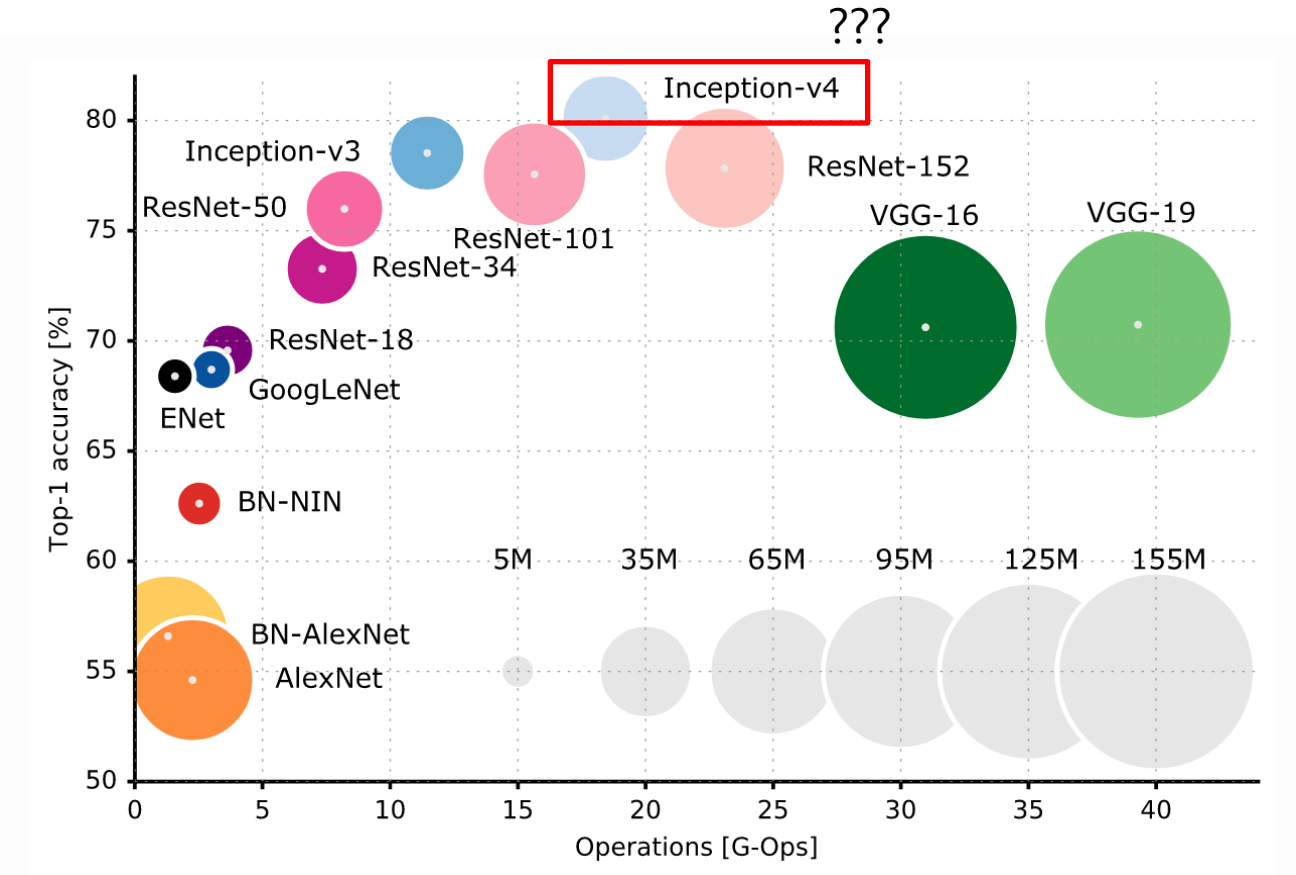
(출처: https://arxiv.org/abs/1512.03385)

Residual Network



(출처: https://arxiv.org/abs/1512.03385)

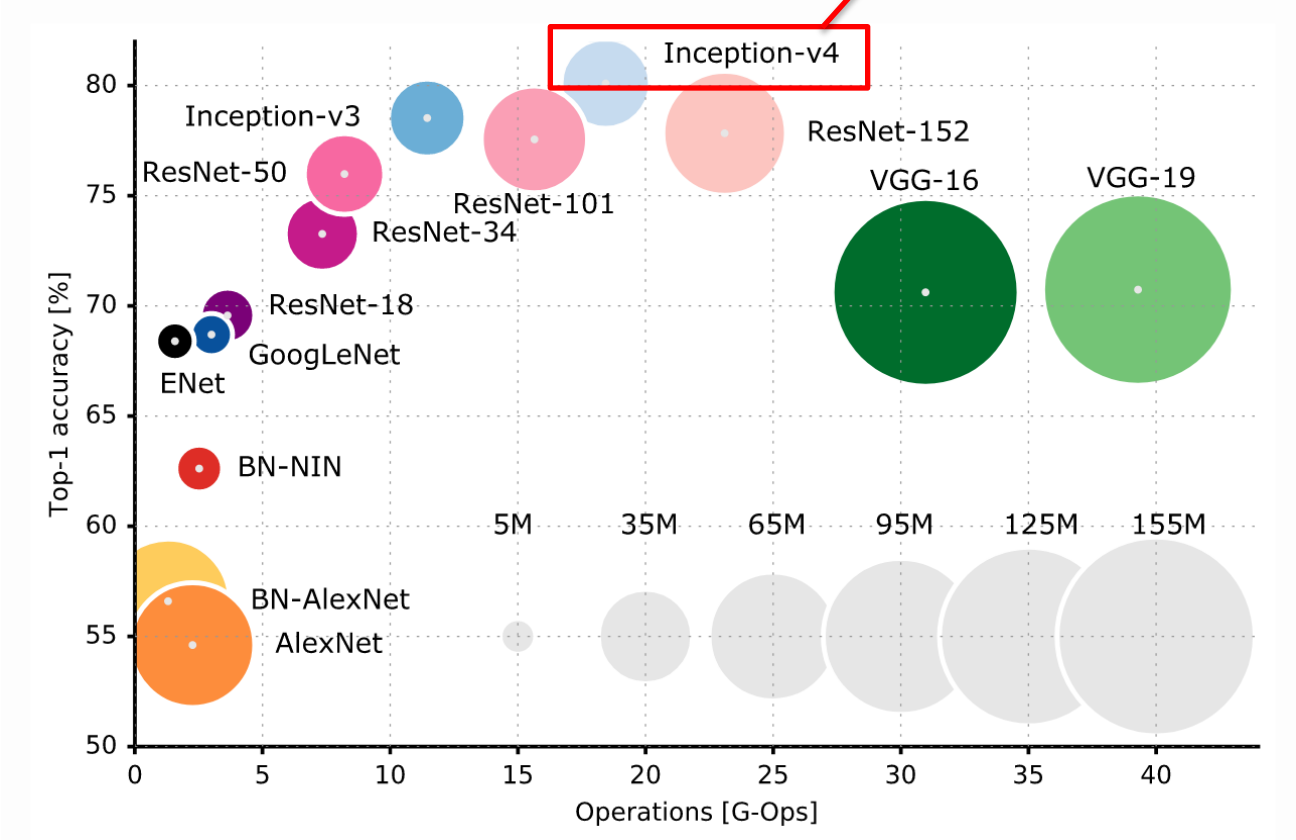
Residual Network



(출처: https://arxiv.org/abs/1512.03385)

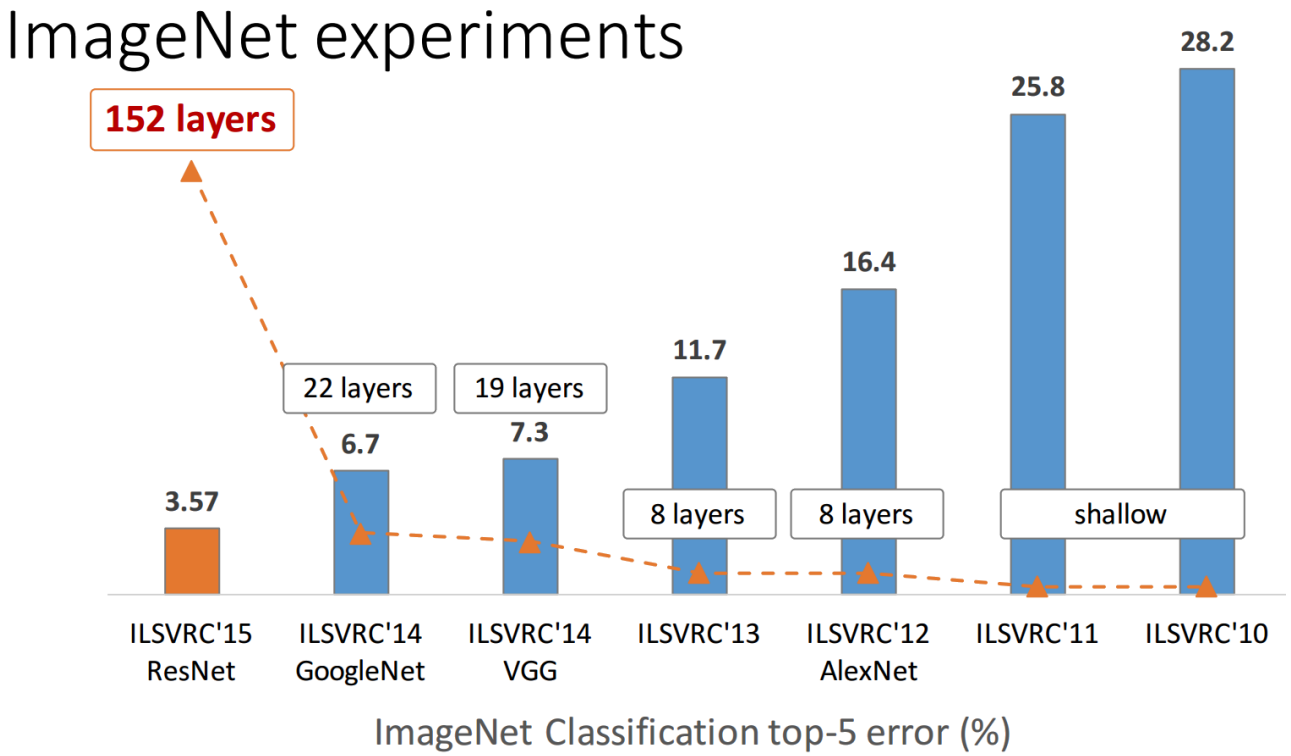
Residual Network

Inception + Residual + BN



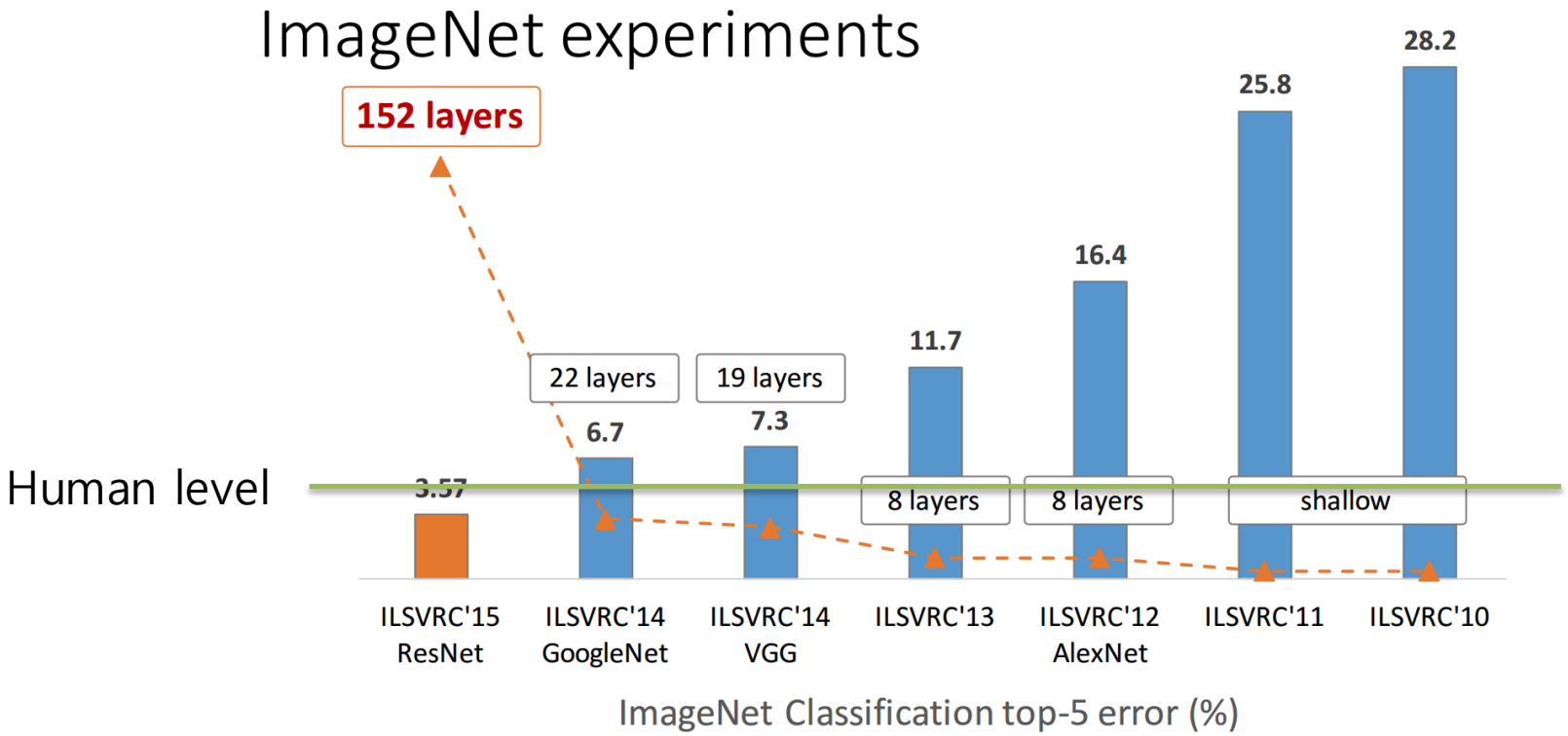
(출처: https://arxiv.org/abs/1512.03385)

Residual Network



(출처: Kaiming He "Deep residual learning for image recognition." ICML. 2016.)

Residual Network



(출처: Kaiming He "Deep residual learning for image recognition." ICML. 2016.)

Q&A