



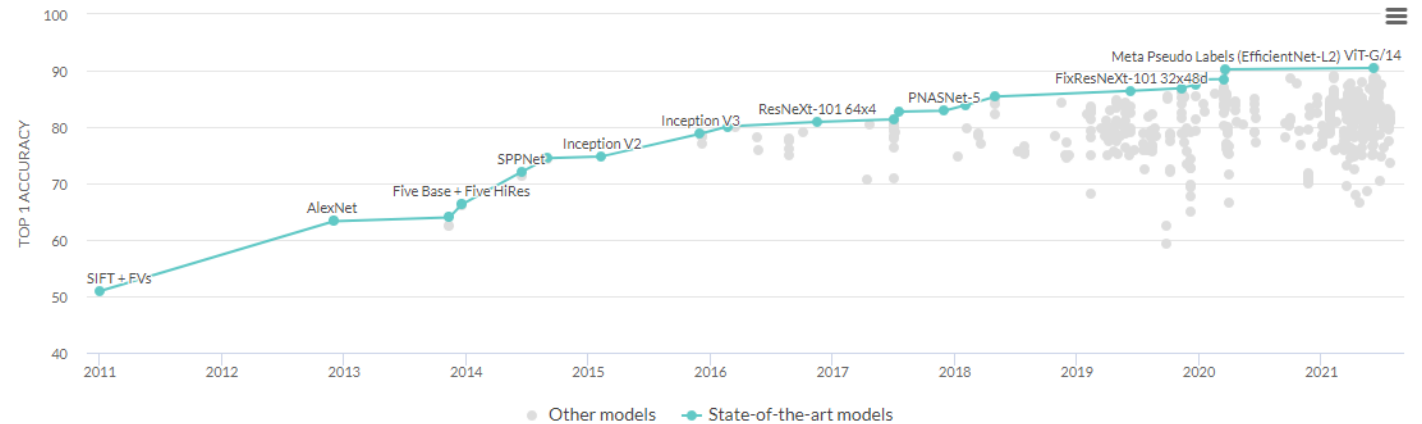
1. ImageNet Competition

Dataset



- 1000만장 이상의 이미지
- 1000개의 class
- 227 x 227 사이즈의 RGB 이미지

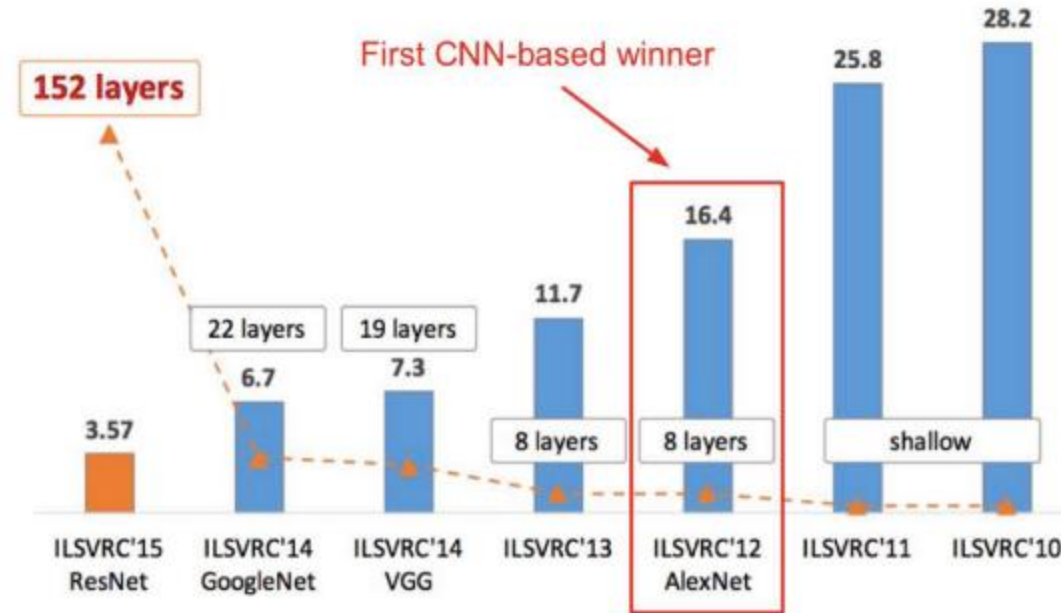
SOTA



2021 SOTA model = One of the ViT

1. ImageNet Competition

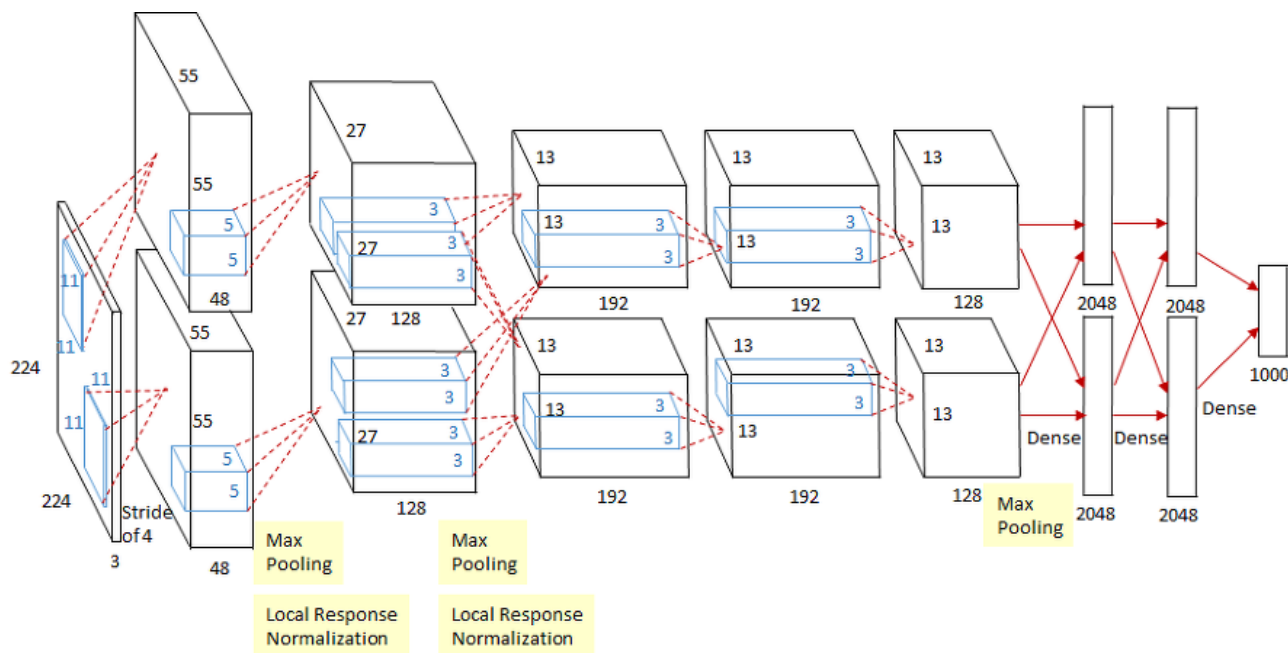
~2015



AlexNet → VGG → GoogleNet → ResNet

2. AlexNet

Architecture



Activation Function = ReLU

Normalization Layer 사용

Optimizer = SGD + Momentum(0.9)

Contribution

최초로 CNN을 사용하여 우승

최초로 ReLU 함수 사용

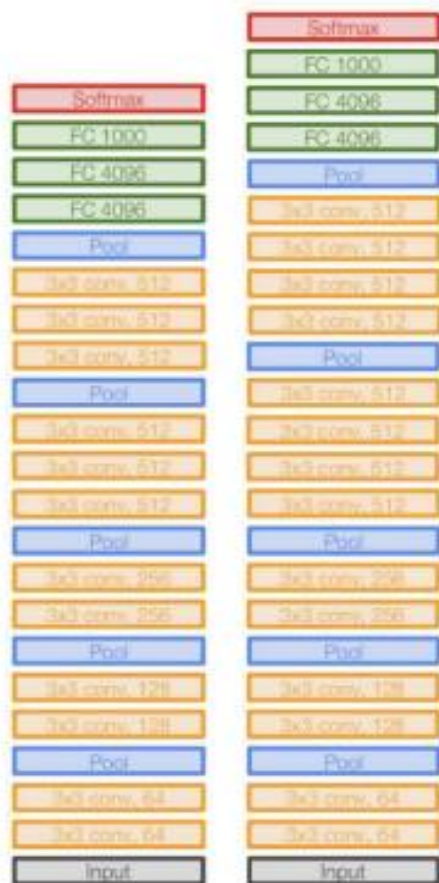
앙상블을 이용하여 성능 향상

3. VGGNet

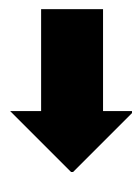
Architecture

Contribution

Future Work



작은 필터를 사용하여 층을 더 깊게 만듦
(AlexNet의 약 2배 Layer 사용)



비선형성

Overfitting

Number of Paramter

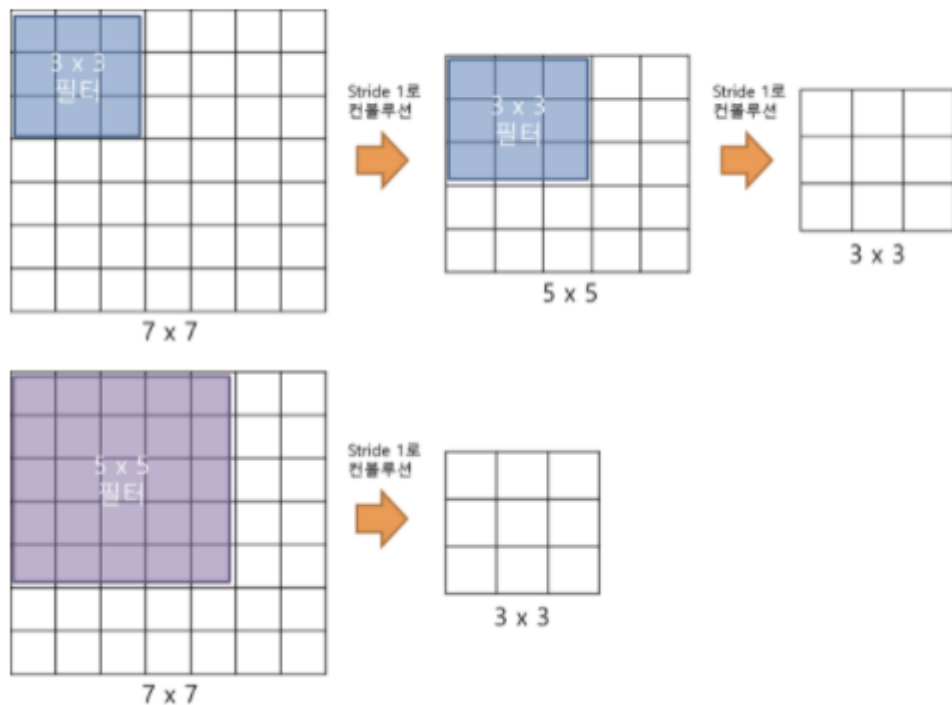


VGG를 기반으로 한
많은 응용 연구 증가

3. VGGNet

Small Filter

3x3 filter 사용



7x7 Image -> 3x3 filter 2번 사용

파라미터 수 : $3 \times 3 + 3 \times 3 = 18$



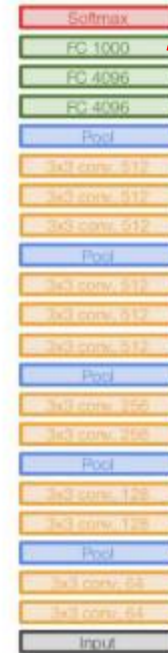
7x7 Image -> 5x5 filter 1번 사용

파라미터 수 : $5 \times 5 = 25$

3. VGGNet

VGG parameter

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
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
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [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
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [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
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [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
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

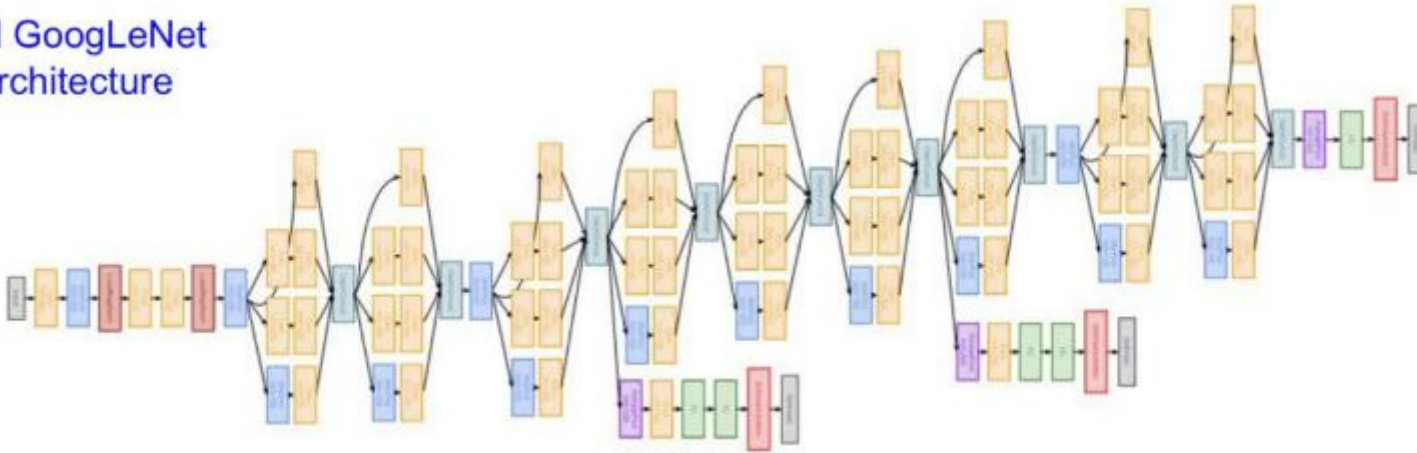
Fully Connected Layer

Parameter 수 급격하게 증가

4. GoogleNet

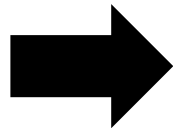
GoogleNet Architecture

Full GoogLeNet architecture



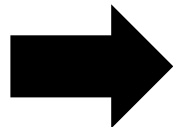
Contribution

More Deeper than VGG19



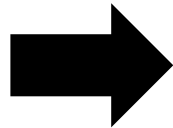
22 Layer > 19Layer(VGG)

Use Inception Module



1x1 filter 사용

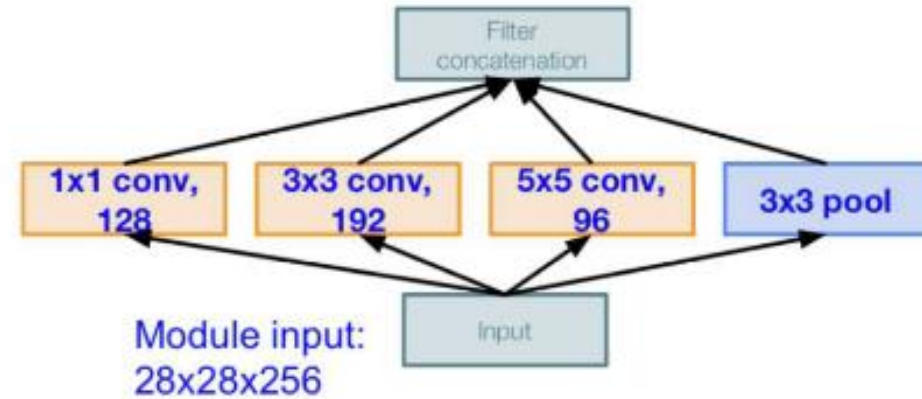
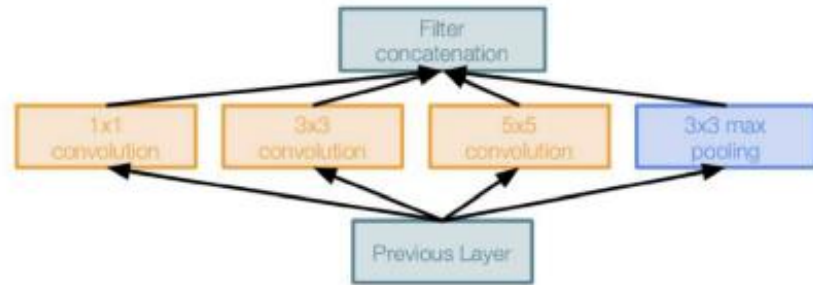
Fewer Paramter



Under 5M parameter

4. GoogleNet

Inception Module



$$1 \times 1 \text{ filter } 128 \text{개} = 28 \times 28 \times 128 \times 1 \times 1 \times 256 = 25,690,112$$

$$3 \times 3 \text{ filter } 192 \text{개} = 28 \times 28 \times 192 \times 3 \times 3 \times 256 = 346,816,512$$

$$5 \times 5 \text{ filter } 96 \text{개} = 28 \times 28 \times 96 \times 5 \times 5 \times 256 = 481,689,600$$

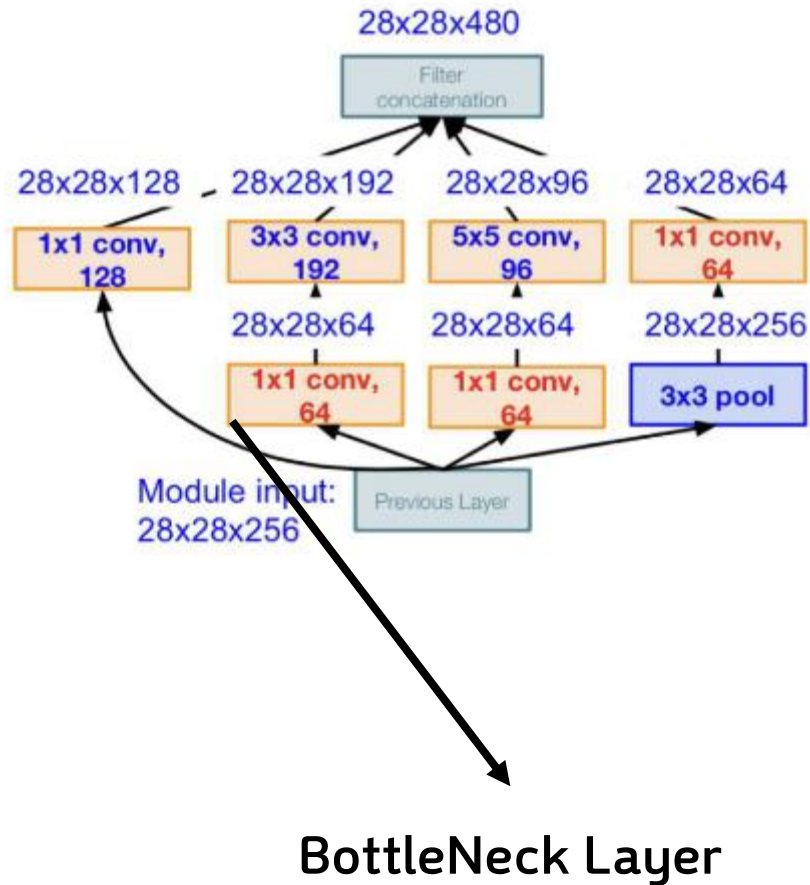
$$\text{MaxPooling} = 28 \times 28 \times 256 = 200,704$$

Total

854,396,939 parameter

4. GoogleNet

Efficient Inception Module



$$1 \times 1 \text{ filter } 128개 = 28 \times 28 \times 128 \times 1 \times 1 \times 256 = 25,690,112$$

$$1 \times 1 \text{ filter } 64개 = 28 \times 28 \times 64 \times 1 \times 1 \times 256 = 12,845,056$$

$$1 \times 1 \text{ filter } 64개 = 28 \times 28 \times 64 \times 1 \times 1 \times 256 = 12,845,056$$

$$3 \times 3 \text{ filter } 192개 = 28 \times 28 \times 192 \times 3 \times 3 \times 64 = 86,704,128$$

$$5 \times 5 \text{ filter } 96개 = 28 \times 28 \times 96 \times 5 \times 5 \times 64 = 120,422,400$$

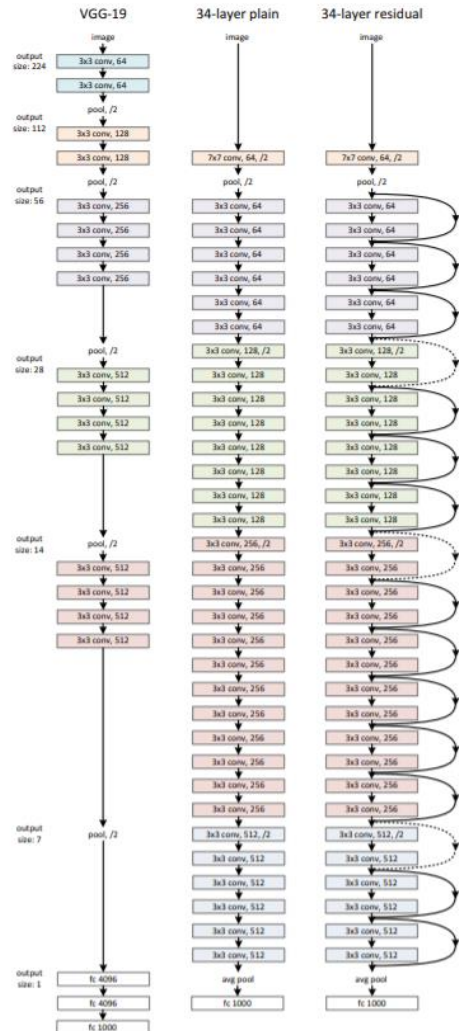
$$1 \times 1 \text{ filter } 64개 = 28 \times 28 \times 64 \times 1 \times 1 \times 256 = 12,845,056$$

Total

271,351,808 paramter

5. ResNet

Architecture

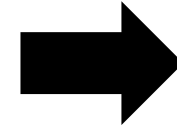


Contribution

More Deeper Layer

Skip Connection

Solution of Degradation Problem

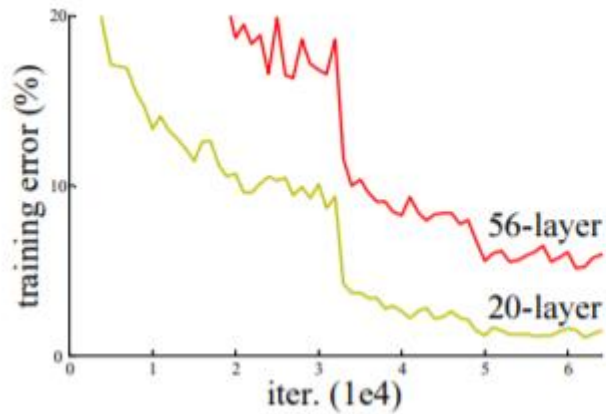


152 Layer

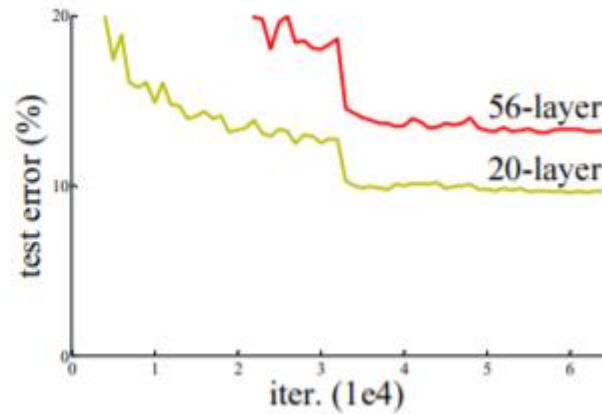
VGG Based

5. ResNet

Degradation



Train error rate



Test error rate

Overfitting

Train 성능



Test 성능



Degradation

Train 성능



Test 성능

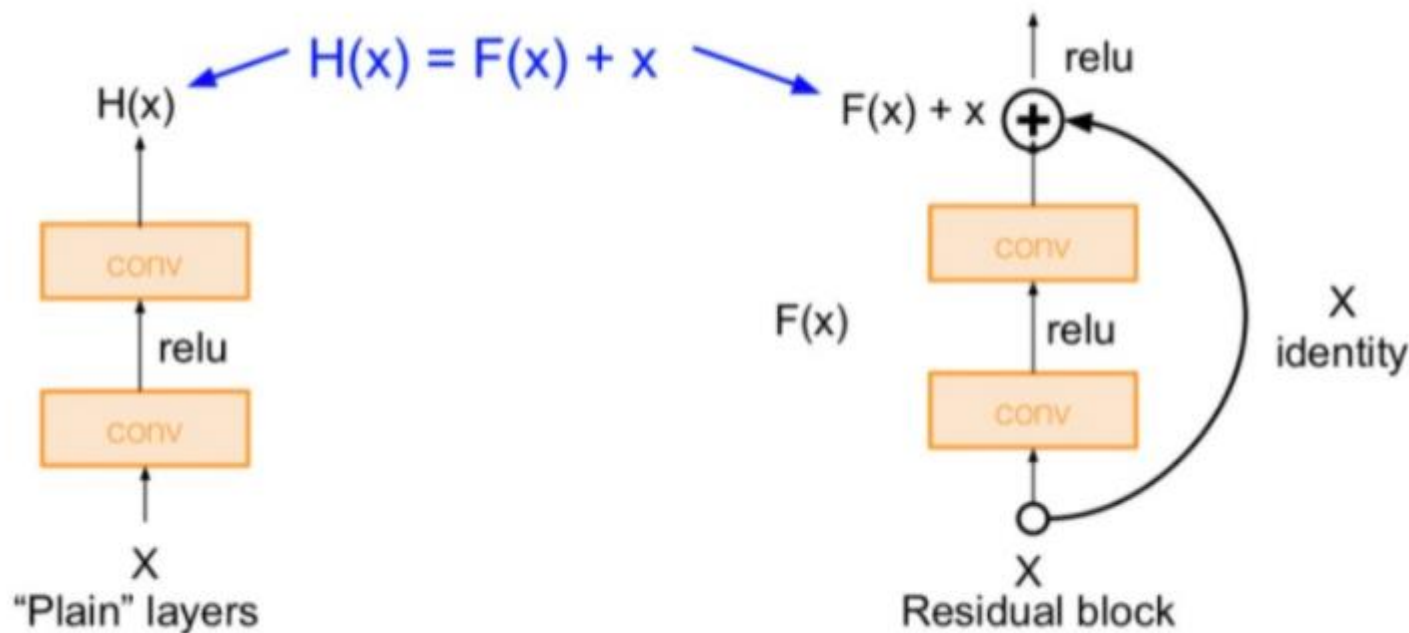


What is Solution? → Residual Learning

5. ResNet

Skip Connection = ShortCut Connection

한 개 이상의 layer를 skip 하는 것



여러 비선형 Layer들이 복잡한 함수이고
Identity mapping이 최적이라면

$H(x)$ 를 mapping 시키는 것 보다
잔차인 $F(x)=0$ 으로 만드는 것이 더 쉽다

Plain Layers

$$H(x) = F(x) + x$$

Residual Layers

$$F(x) = H(x) - x$$

$H(x)$: Original mapping

$F(x)$: Residual

x : Input