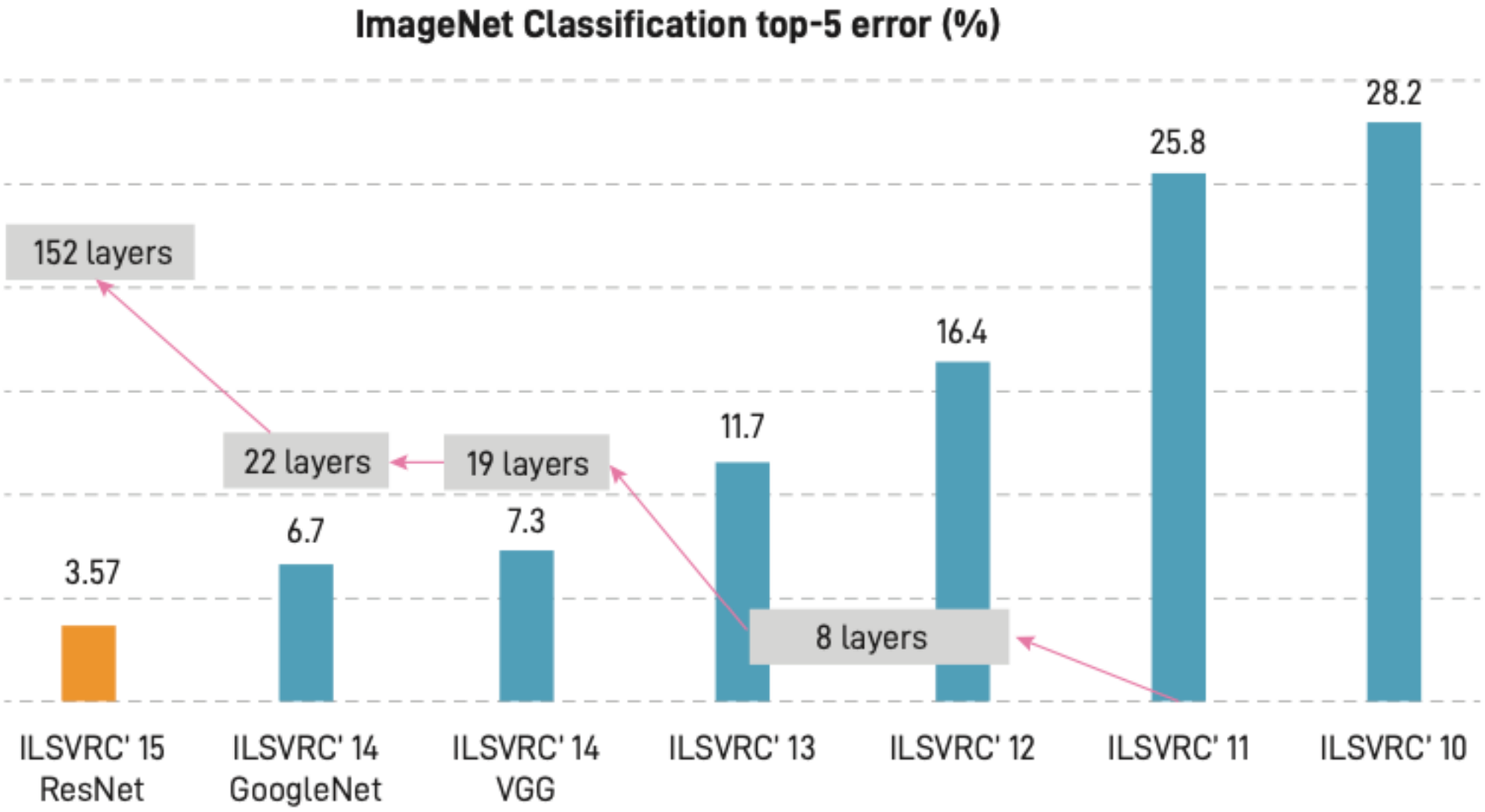
인공지능 심화 과정

ILSVRC 대표적인 분류 모델 II

ResNet



Residual Network (ResNet) 은 마이크로소프트의 Kaiming He et al에 의해 개발되었고, 2015년 ISVRC 에서 우승하였다.

https://arxiv.org/pdf/1512.03385.pdf

ResNet

MNIST와 같은 데이터셋에서는 얕은 층의 CNN으로도 높은 분류 성능을 얻을 수 있다. 하지만 CIFAR나 ImageNet과 같이 좀 더 복잡하고 도전적인 데이터셋에서 얕은 네트워크로는 한계가 있어 연구자들은 점점 더 깊은 층을 가진 CNN을 만들려고 했다.

Abstract

딥러닝에서 neural networks가 깊어질수록 성능은 더 좋지만 train이 어렵다는 것은 알려진 사실이다. 그래서 이 논문에서는 잔차를 이용한 잔차 학습 (residual learning framework)를 이용해서 깊은 신경망에서도 training이 쉽게 이뤄질 수 있다는 것을 보이고 방법론을 제시한다.

결과적으로 152개의 layer를 쌓아서 기존의 VGGNet보다 좋은 성능을 내면서 복잡성은 줄였다. 3.57%의 error를 달성해서 ILSVRC 2015에서 1등을 차지했으며, CIFAR-10 데이터를 사용하여 100개에서 1000개의 레이어를 구성하여 테스트하고 분석한 것을 논문에 담았다.

ResNet – 기존에 레이어를 쌓기 어려운 이유

이 논문이 참조한 많은 논문들로 인해 deep model일수록 성능이 좋다는 것은 알려져 있다. 깊어지는 layer와 함께 떠오르는 의문은 **"과연 더 많은 레이어를 쌓는 것만큼 신경망을 학습시키는 것이 쉬운가"** 이다. 이 질문에 답하는 데 있어 장애물은 처음부터 수렴을 방해하는 **그레이디언트 소실/발산**이라는 문제였다. 그러나 이 문제는 **정규화된 초기화와 중간 정규화 계층으로 대부분 해결되었으며**, 이는 수십 개의 계층을 가진 네트워크가 역전파와 함께 확률적 경사하강(SGD)을 위한 수렴을 시작할 수 있게 한다.

[추가 설명]

층을 쌓을 때 마다 각 레이어마다 weight와 bias라는 파라메터를 갖는데, 신경망을 학습시킨다는 것은 이 파라메터를 SGD(stochastic gradient descent)로 업데이트 한다는 것을 의미 한다. 여기서 기울기를 통해 weight를 업데이트 할 때, 기울기 소실/발산으로 인해 안정적이지 않은 문제가 있다. 기울기가 안정적이지 않은 이유는 신경망에서 사용하는 activation function과 연관성이 있다. 작은 값을 가지는 난수로 네트워크의 가중치를 초기화하면 모든 activation이 0이 되는 현상이 발생한다.

ResNet - 기존에 레이어를 쌓기 어려운 이유

이렇게 모든 activation이 0이나 -1, 1로 포화되는 문제를 해결하기 위해 **Xavier initialization**과 같은 초기화 방법이나 **batch normalization** 등의 기법이 나왔다. 초기화와 정규화는 이와 같은 기울기 문제 점을 해결해 더 깊은 네트워크의 학습이 가능하게 했지만 이에 비례하여 train loss가 증가하는 문제가 생겼다.

Xavier Initialization

각 layer의 활성화 값을 더 광범위하게 분포시킬 목적으로 weight의 적절한 분포를 찾으려 했다. 또한, ***tanh* 또는 *sigmoid*로 활성화 되는 초기 값을 위해 이 방법을 주로 사용**한다. 이전 layer의 neuron의 개수가 n이라면 표준편차가 $\frac{1}{\sqrt{n}}$인 분포를 사용하는 개념이다. 너무 크지도 않고 작지도 않은 weight을 사용하여 gradient가 소실되거나 발산하는 문제를 막는다. *Xavier Initialization*을 사용하게 된다면, 뉴런의 개수가 많을수록 초깃값으로 설정하는 weight이 더 좁게 퍼짐을 알 수 있다.

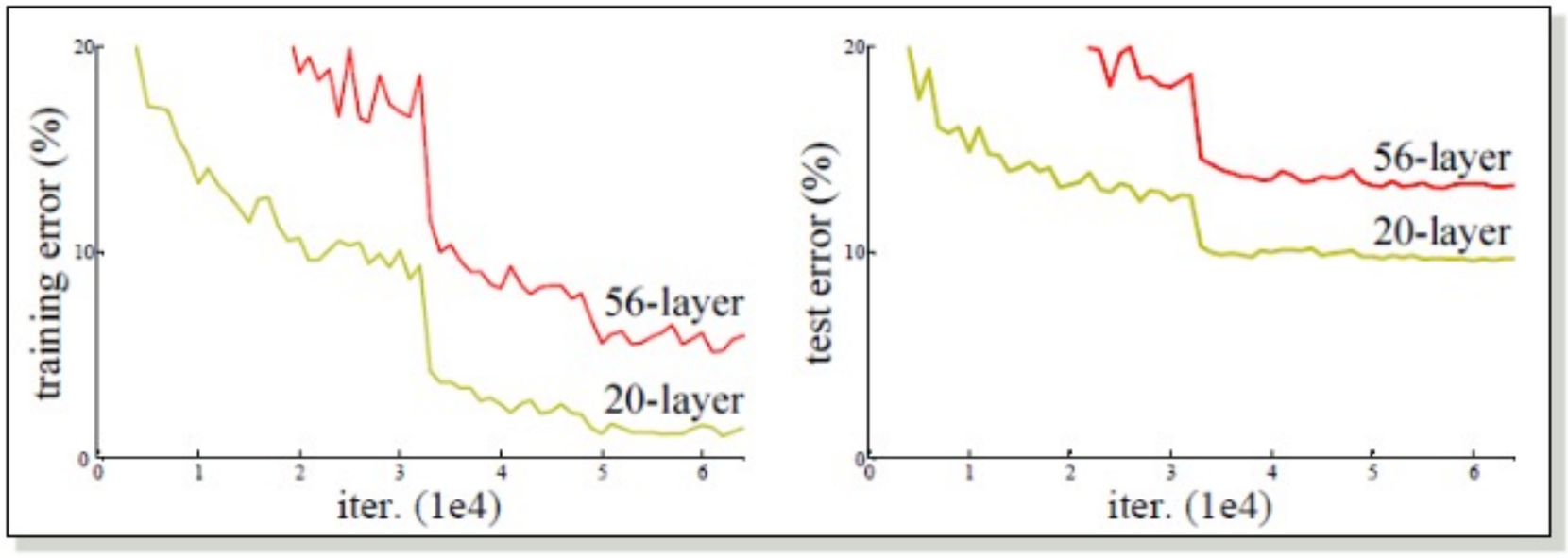
- **주의 : 활성화 함수가 ReLU인 경우 비선형성이 발생해서 멈추게 된다.**
  ReLU는 He Initialization 사용

https://velog.io/@cha-suyeon/DL-가중치-초기화Weight-Initialization-

ResNet

이 논문이 참조한 많은 논문들로 인해 deep model일수록 성능이 좋다는 것은 알려져 있다.(그리고 CNN에 대한 많은 연구 결과가 있음.)

위 그림을 보면 네트워크가 깊어지면서 top-5 error가 낮아진 것을 확인할 수 있다. 한마디로 성능이 좋아진 것이다. 그렇다면 "망을 깊게 하면 무조건 성능이 좋아질까?" 이것을 확인하기 위해 ResNet의 저자들은 컨볼루션 층들과 fully-connected 층들로 20층의 네트워크와 56층의 네트워크를 각각 만든 다음에 성능을 테스트 해보았다.



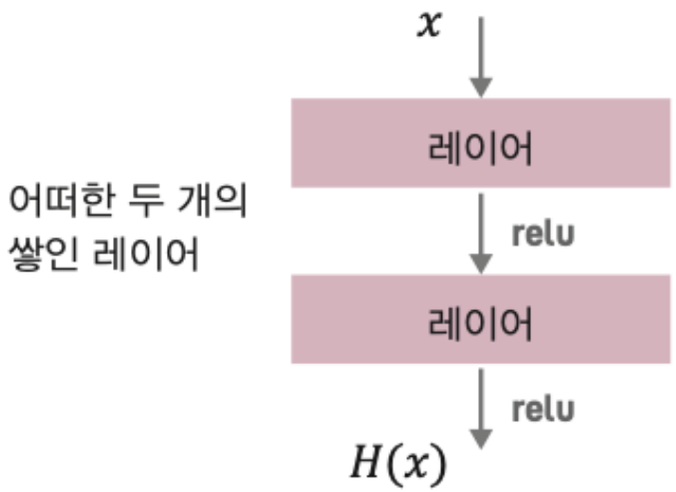
ResNet

앞의 그래프들을 보면 오히려 더 깊은 구조를 갖는 56층의 네트워크가 20층의 네트워크보다 더 나쁜 성능을 보임을 알 수 있다. **기존의 방식으로는 망을 무조건 깊게 한다고 능사가 아니라는 것을 확인한 것이다.** 뭔가 새로운 방법이 있어야 망을 깊게 만드는 효과를 볼 수 있다는 것을 알게 되었다.

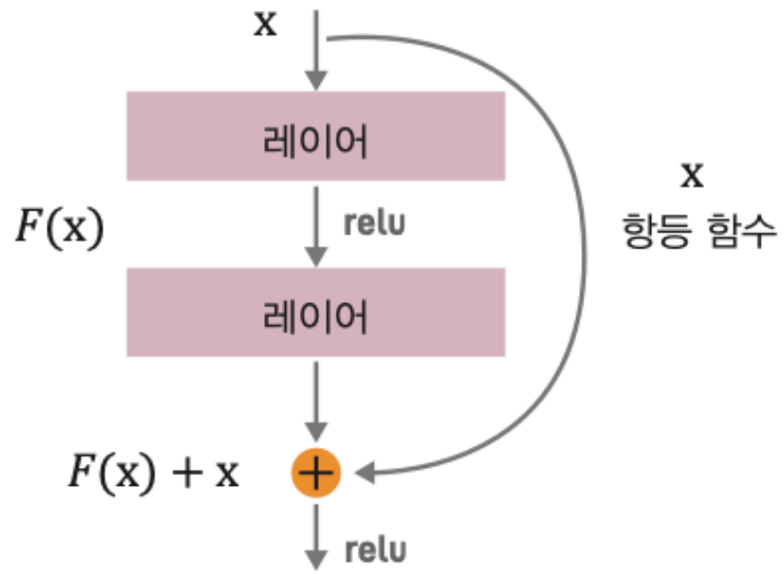
ResNet

Residual Block

그것이 바로 ResNet의 핵심인 Residual Block의 출현을 가능케 했다. 아래 그림에서 오른쪽이 Residual Block을 나타낸다. 기존의 망과 차이가 있다면 입력 값을 출력 값에 더해줄 수 있도록 **지름길(shortcut)**을 하나 만들어준 것 뿐이다.



[그림3] 기존 Plain Net



[그림4] Residual Block 구조

ResNet

기존의 신경망은 입력값 x를 타겟값 y로 매핑하는 함수 H(x)를 얻는 것이 목적이었다. 그러나 **ResNet은 F(x) + x를 최소화하는 것을 학습 목표로 한다.** x는 현시점에서 변할 수 없는 값이므로 F(x)를 0에 가깝게 만드는 것이 목적이 된다. F(x)가 0이 되면 출력과 입력이 모두 x로 같아지게 된다. F(x) = H(x) - x이므로 F(x)를 최소로 해준다는 것은 H(x) - x를 최소로 해주는 것과 동일한 의미를 지닌다. 여기서 H(x) - x를 **잔차(residual)**라고 한다. 즉, 잔차를 최소로 해주는 것이므로 ResNet이란 이름이 붙게 된다.

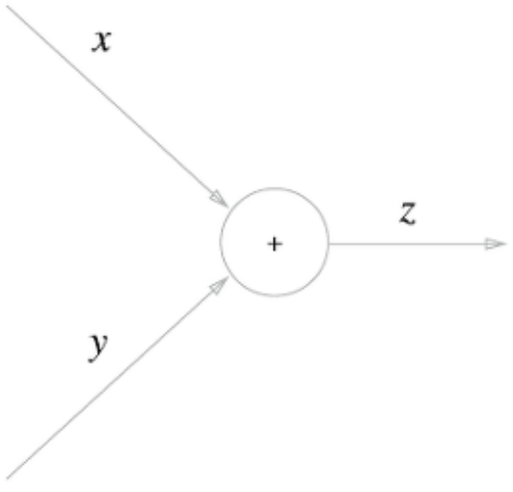
기존의 신경망이 direct mapping을 학습했다면 Residual learning은 입력부터 얼마만큼 달라져야 하는지를 학습한다. H(x)가 학습해야 하는 mapping이라면 이를 H(x)=F(x)+x로 새롭게 정의한다. 이때, 레이어의 입력을 레이어의 출력에 바로 연결하는 “Skip Connection, Shortcut”을 사용하였다. 앞에 그림에서 이는 항등 함수 (identity) x로 표현되어 있다.

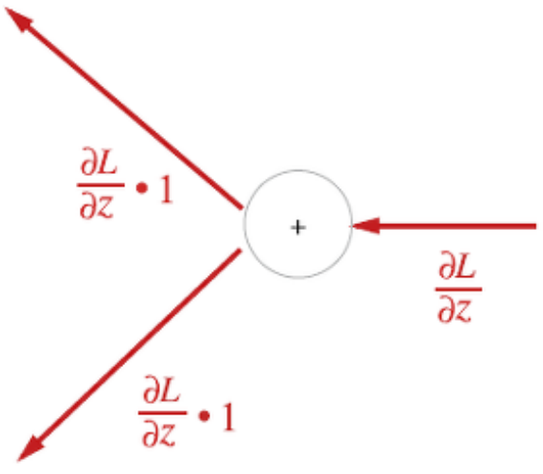
ResNet

이 방법은 최적화가 비교적 쉽고, 단순한 지름길을 통해 x를 식에 추가하기만 하면 되기 때문에 기존 네 트워크의 구조를 크게 변경하지 않고 학습을 진행할 수 있다. 또한, 매개변수의 수나 연산량의 증가도 없으며, 복잡도가 증가하지도 않는다. 그리고 신경망을 깊게 구성하더라도 기울기 소실 문제가 발생하 지 않는다는 가장 큰 장점이 있다. 이는 전체 신경망에 걸쳐 가중치들을 최적화하는 것이 아니라, 짧은 레이어들로 이루어진 Residual Block마다 부분적인 학습이 진행되기 때문이다.

덧셈 노드의 역전파

https://excelsior-cjh.tistory.com/171

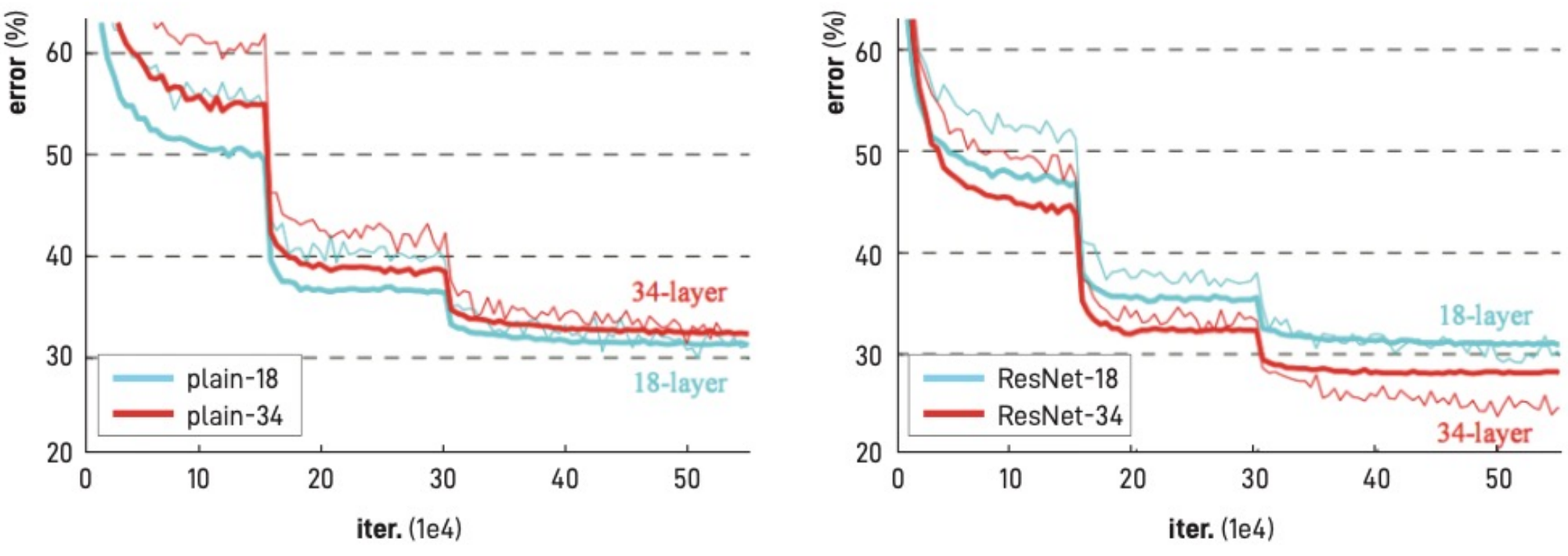




ResNet

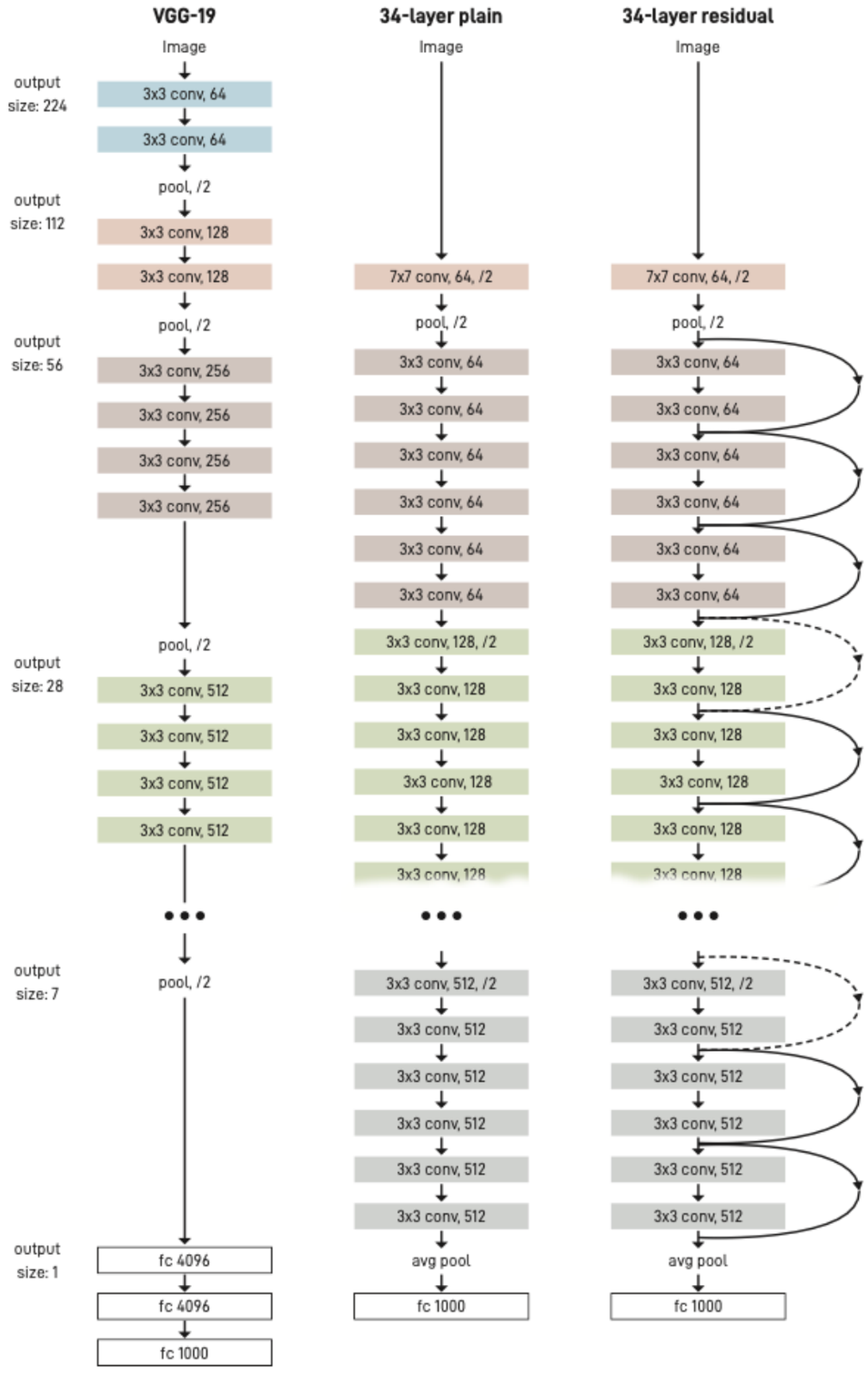
ResNet에서 Skip connection은 2개의 레이어를 건너뛰는 형태로 제작되었다. 레이어에 들어오는 입력이 Skip connection을 통해서 건너뛰어 레이어를 거친 출력과 Element-wise addition을 한다. 즉, 기울기를 residual block과 함께 더 깊은 레이어로 전송한다.

Plain network의 경우 18층에서 34층으로 깊이를 늘리면 학습 오류가 늘어나는 것을 볼 수 있다. ResNet의 경우 degradation 문제가 어느 정도 해결됨을 알 수 있다.



Plain Net과 ResNet의 비교 실험

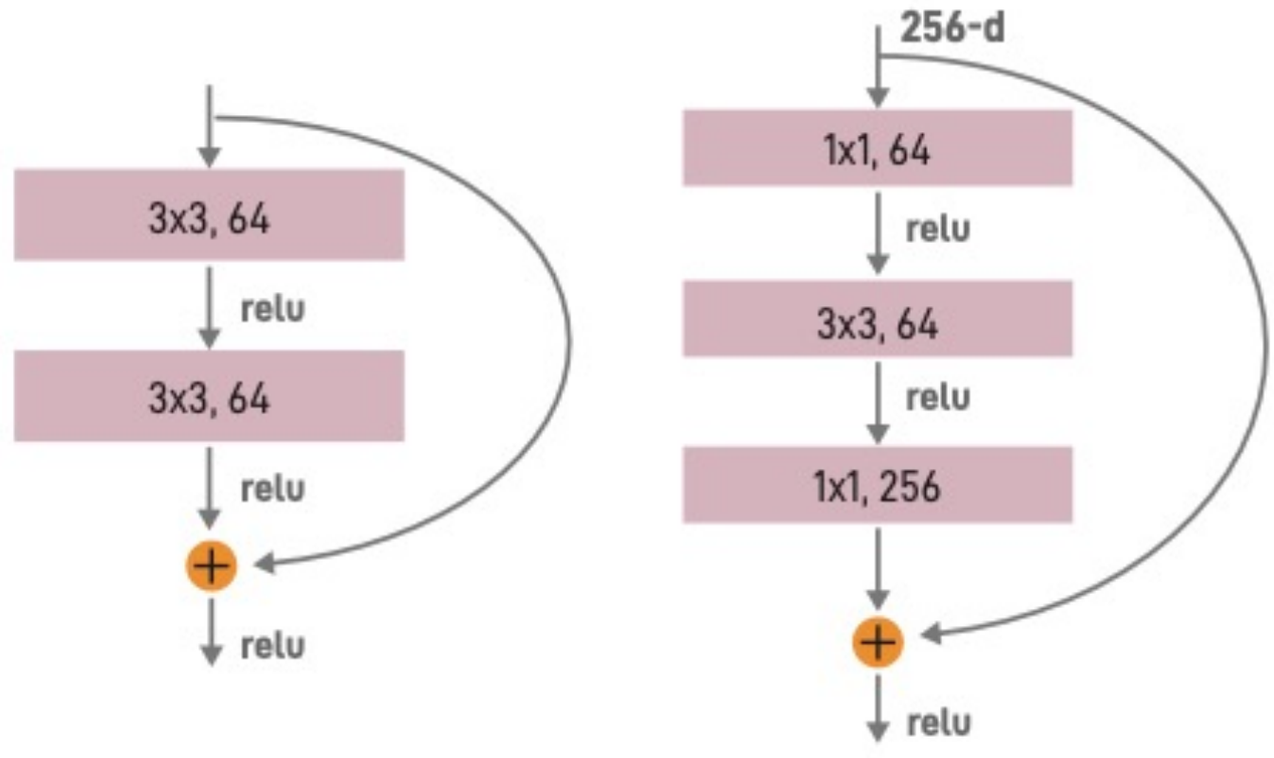
ResNet – 모델 depth 비교



주교재 21페이지 참조

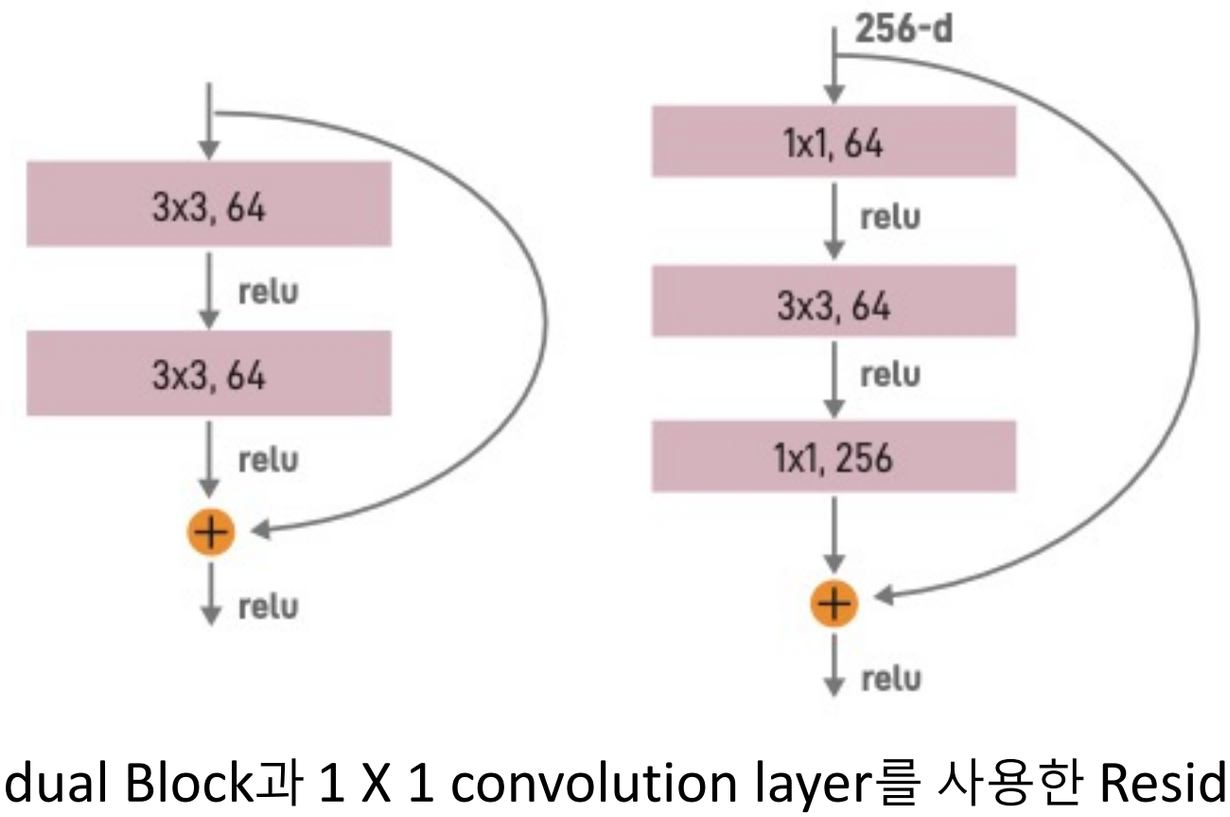
ResNet

ReResNet도 신경망의 깊이가 점점 깊어지면 parameter의 수가 많아지기 때문에 residual block을 다른 구조로 사용하는 방식 또한 고안되었다.



기본 Residual Block과 1 X 1 convolution layer를 사용한 Residual Block

ResNet



기본 Residual Block과 1 X 1 convolution layer를 사용한 Residual Block

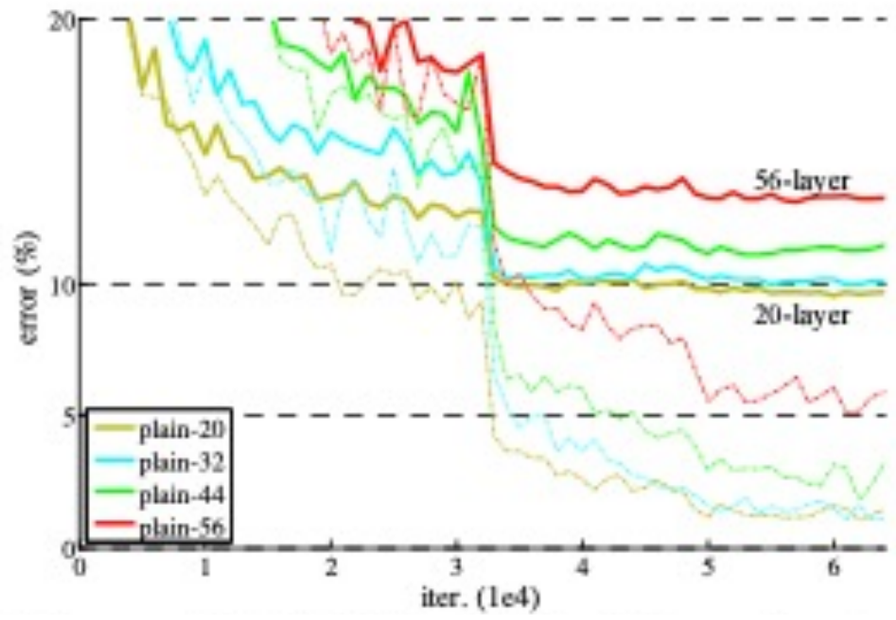
Residual Block의 원래 구조는 왼쪽과 같지만 50층 이상의 ResNet에서는 오른쪽 그림과 같이 사용한다. 1 × 1 convolution을 통해 필터 크기를 줄인 이후 3 × 3 convolution을 하면 parameter 수를 조금 줄일 수 있다. 이러한 구조의 residual block을 bottleneck block이라고 부른다. Bottleneck 구조를 사 용하여 ResNet의 내부 구조를 바꾸면 152 레이어까지 쌓아도 VGG보다 모델의 크기가 작다.

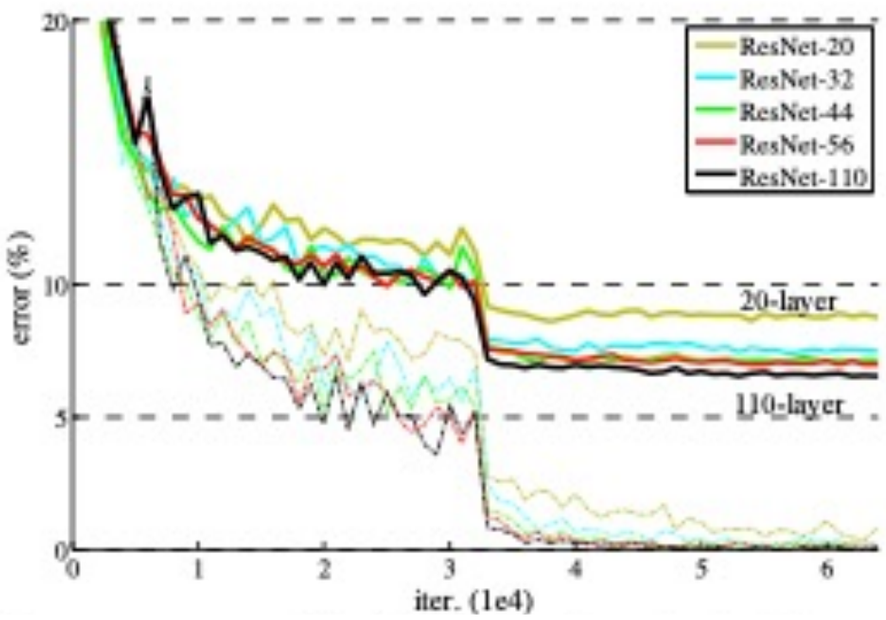
ResNet

다음은 CIFAR-10에서의 성능이다.

method			error(%)
Maxout [10]			9.38
MIN [25]			8.81
DSN [24]			8.22
	# Layers	# Params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.47M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	6.43 (6.61±0.16)
ResNet	1202	19.4M	7.93

ResNet





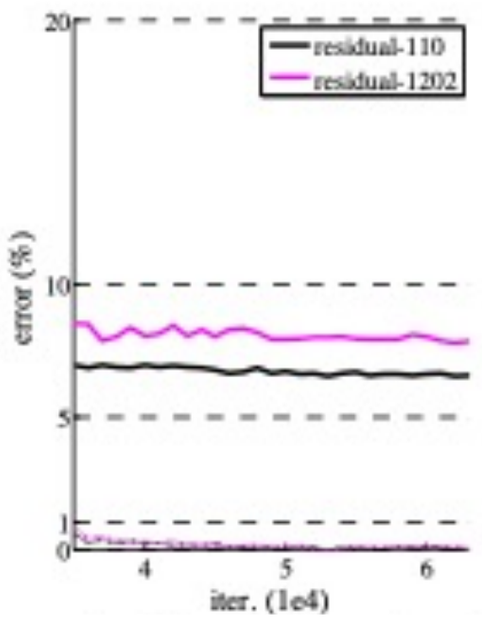


Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

ResNet

아래 표는 18층, 34층, 50층, 101층, 152층의 ResNet이 어떻게 구성되어 있는가를 잘 나타내준다. 깊은 구조일수록 성능도 좋다. 즉, 152층의 ResNet이 가장 성능이 뛰어나다.

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

ResNet

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).

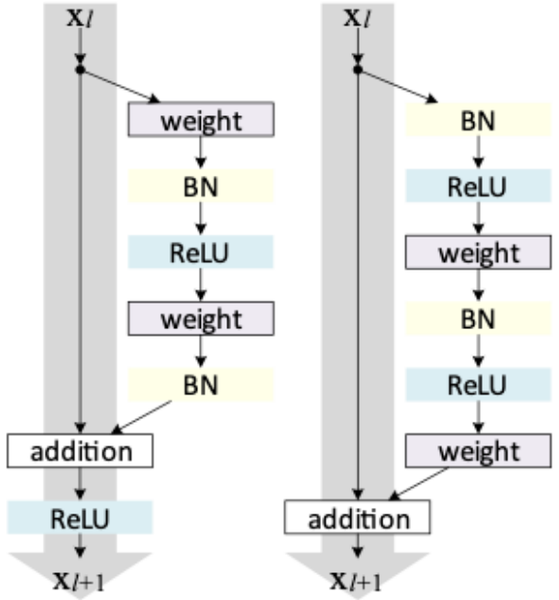
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

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

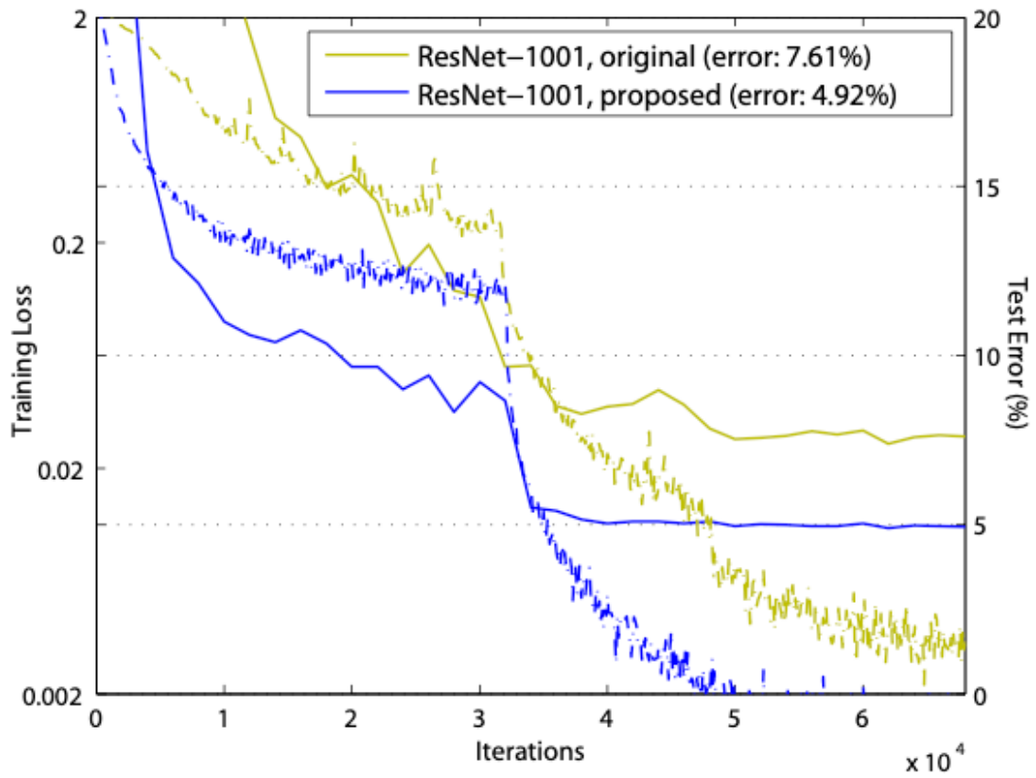
ResNet v2

Bottleneck 구조를 사용해도 1000층 이상의 깊이를 가지게 되면 다시 degradation 문제가 생겼다. ResNet-110이 6.43% 오차율을 가지는 반면 ResNet-1202는 7.93% 오차율을 가진다. 저자는 이후 1000개의 층을 쌓아도 degradation 문제가 생기지 않게 하는 방법을 고안하였다.

ResNet v2 : https://arxiv.org/pdf/1603.05027.pdf



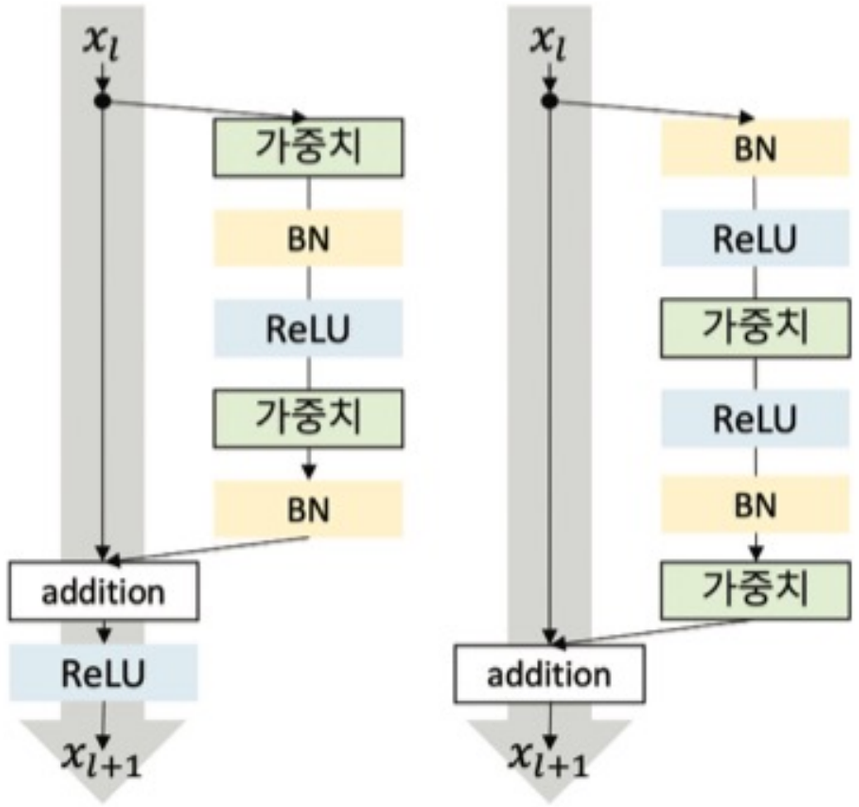
(a) original (b) proposed



배치 정규화 : https://eehoeskrap.tistory.com/430

https://arxiv.org/pdf/1502.03167.pdf

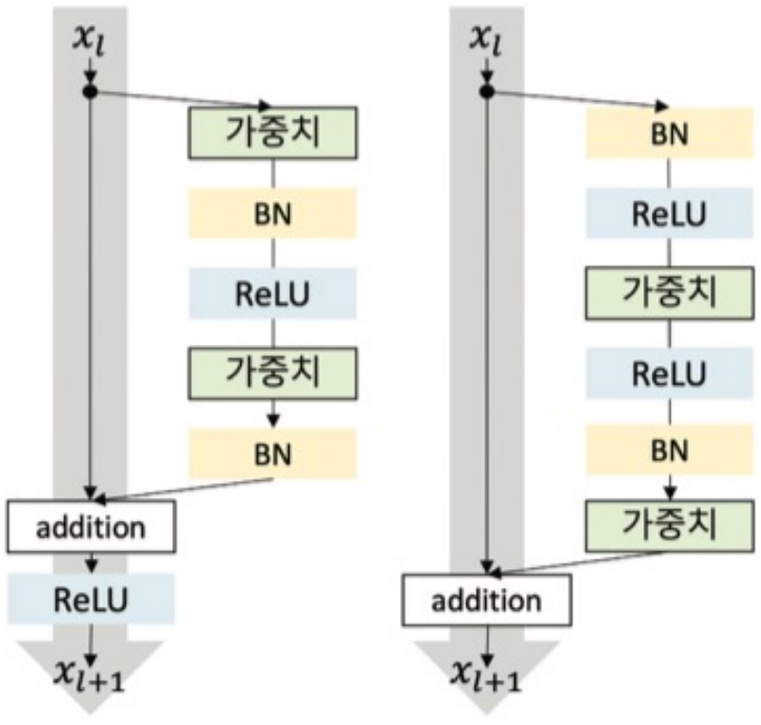
ResNet



1) 최적화를 더 쉽게 해주었다.

기존 residual block에서는 ReLU activation 이후에 다음 residual block으로 넘어가기 때문에 back- propagation 할 때 ReLU에 의해 정보가 잘려 나갈 수 있기 때문에 학습 손실이 느리게 줄어들었다. 하지 만 새로운 residual unit에서는 back-propagation할 때 identity에 대한 부분은 출력에서 입력까지 유지 될 수 있었다. 따라서 1000층 이상의 매우 깊은 구조에서 최적화가 더 쉬워질 수 있게 하였다.

ResNet



2) Batch Normalization으로 인한 regularization 효과 때문이다.

기존 Residual Block의 경우 BN을 한 이후에 identity mapping과 addition을 해주고 그 이후에 ReLU를 붙이는 형태이다. 따라서 Batch normalization의 regularization 효과를 충분히 보지 못하였다. 새로운 residual block에서는 Batch normalization 이후에 바로 ReLU를 붙여 성능을 개선할 수 있었다. 여러 번의 실험과 변형을 통해 저자는 더 깊은 레이어를 잘 쌓을 수 있게 하는 ResNet을 제작하였고 이후 많은 연구자들이 구조를 차용하여 많은 네트워크 구조를 생성할 수 있게 되었다.

MobileNet V1

Abstract

모바일, embedded vision 앱에서 사용되는 것을 목적으로 한 MobileNet이라는 효율적인 모델을 제시한다. **Depth-wise separable convolutions**라는 구조에 기반하며 2개의 단순한 hyper-parameter를 가진다. 이 2가지는 사용되는 환경에 따라 적절히 선택하여 적당한 크기의 모델을 선택할 수 있게 한다. 수많은 실험을 통해 가장 좋은 성능을 보이는 설정을 찾았으며 타 모델에 비해 성능이 거의 떨어지지 않으면서 모델 크기는 몇 배까지 줄인 모델을 소개 한다.

논문 링크 : https://arxiv.org/pdf/1704.04861.pdf

MobileNet V1

1. Introduction

ConvNet은 computer vision 분야라면 어디서든 사용되었지만(2017년 논문이므로 이 당시에는 vision에 Transformer를 쓰지 않았다.) 모델의 크기가 너무 커지고 가성비가 좋지 않다. 그래서 이 논문에서는 모델의 크기와 성능(low latency)을 적절히 선택할 수 있도록 하는 2개의 hyper-parameter를 갖는 효율적인 모델을 제시한다.

섹션 2에서는 지금까지의 작은 모델을,

섹션 3에서는 **MobileNet**을

섹션 4에서는 실험을

섹션 5에서는 요약을 서술한다.

MobileNet V1

Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

MobileNet V1

2. Prior Work

모델 크기를 줄이는 연구도 많았다. 크게 2가지로 나뉘는데,

- 사전 학습된 네트워크를 압축하거나
- 작은 네트워크를 직접 학습하는 방식이다.

이외에도 모델을 잘 압축하거나, 양자화, hashing, pruning 등이 사용되었다.

MobileNet은 기본적으로 작은 모델이지만 latency를 최적화하는 데 초점을 맞춘다.

MobileNets은 Depthwise separable convolutions을 사용한다.

MobileNet V1

3. MobileNet Architecture

3.1. Depthwise Separable Convolution

Factorized convolutions의 한 형태로, 표준 convolution을

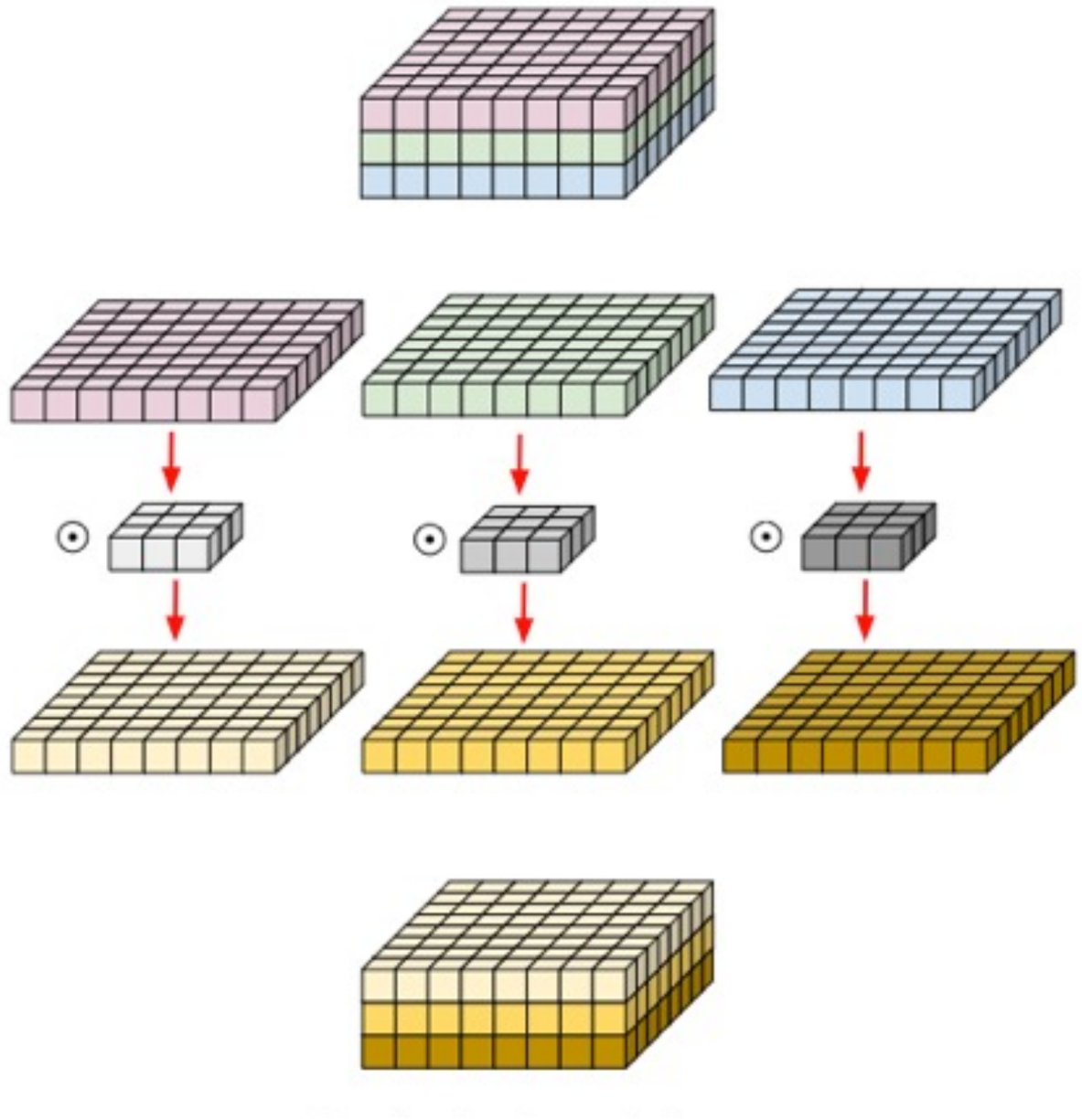
Depthwise convolution(dwConv)과

Pointwise convolution(pwConv, 1×1 convolution) 으로 쪼갠 것이다.

dwConv는 각 입력 채널당 1개의 filter를 적용하고, pwConv는 dwConv의 결과를 합치기 위해 1×1 conv를 적용한다. 이와 비교해서 표준 conv는 이 2가지 step이 하나로 합쳐져 있는 것이라 보면 된다.

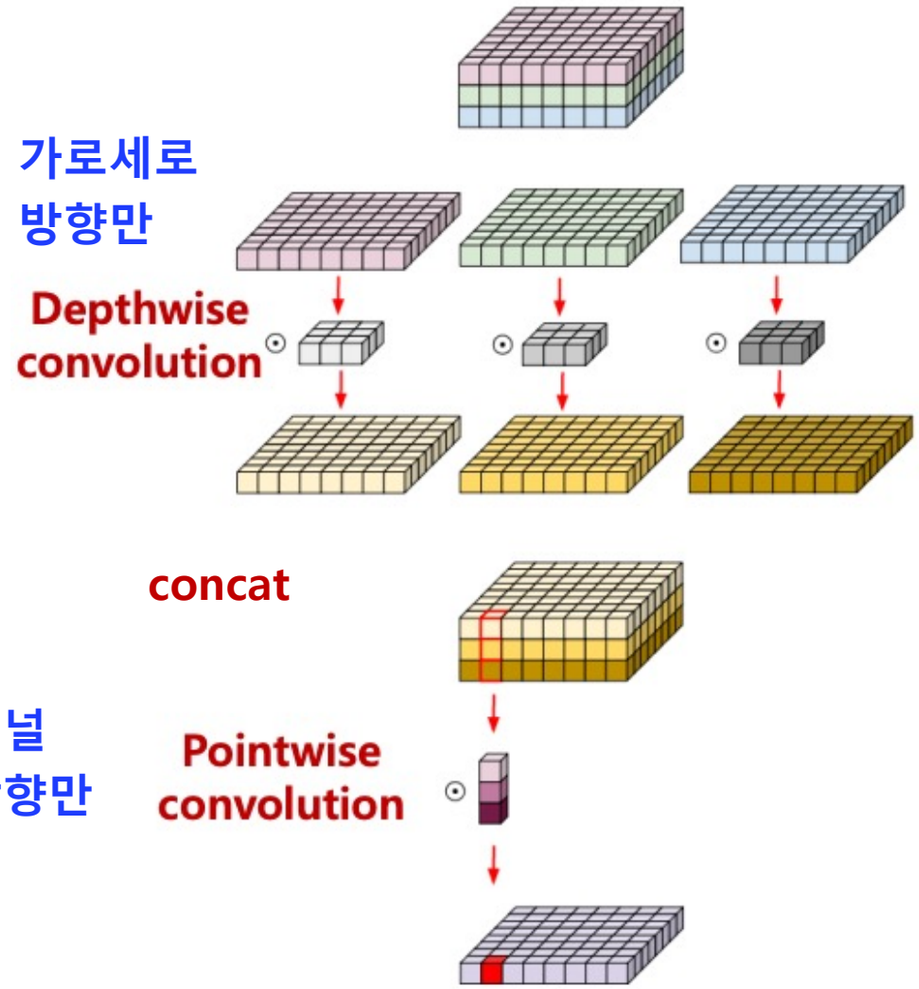
DSConv(depthwise separable convolution)은 이를 2개의 layer로 나누어, filtering을 위한 separate layer, combining을 위핸 separate layer로 구성된다. 이는 모델 크기를 많이 줄일 수 있게 해 준다.

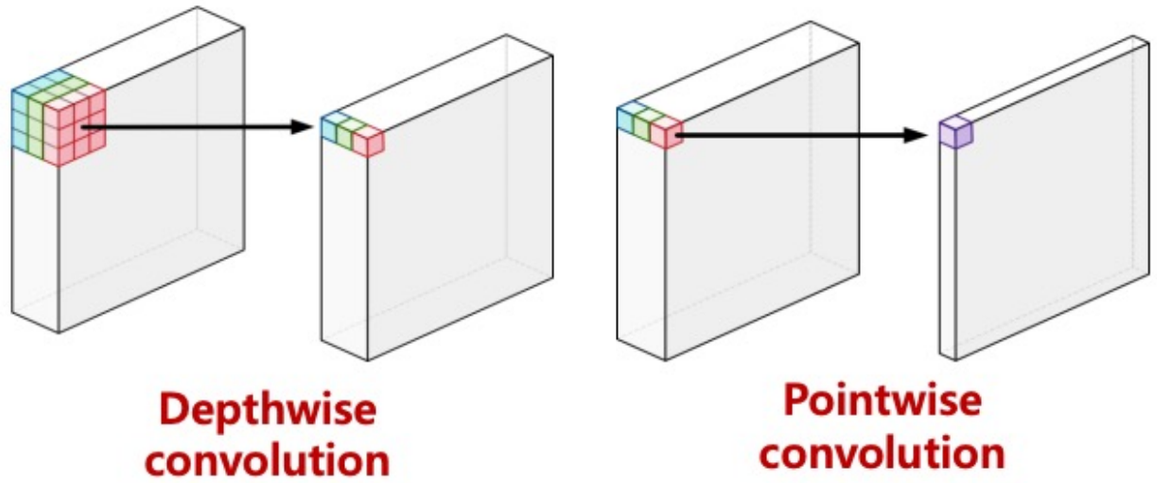
MobileNet V1 - Depthwise Convolution



Depthwise Convolution

MobileNet V1 - Depthwise Separable Convolution





MobileNet V1 - Depthwise Separable Convolution

- Standard convolution takes an $h_i \times w_i \times d_i$ input tensor L_i, and applies convolutional kernel $K \in R^{k \times k \times d_i \times d_j}$ to produce an $h_i \times w_i \times d_j$ output tensor L_j .
- Standard convolutions have the computational cost of
  - $h_i \times w_i \times d_j \times k \times k \times d_i$
- Depthwise separable convolutions cost
  - $h_i \times w_i \times d_i \times (k^2 + d_j)$
- Reduction in computations
  - $^1/_{d_j} + {}^1/_{k^2}$
  - If we use 3x3 depthwise separable convolutions, we get between 8 to 9 times less computations

MobileNet V1

3.2 Network Structure and Training

표준 Conv와 DSConv layer의 구조를 비교하면 아래 그림과 같다. 1×1 conv가 pwConv이다.

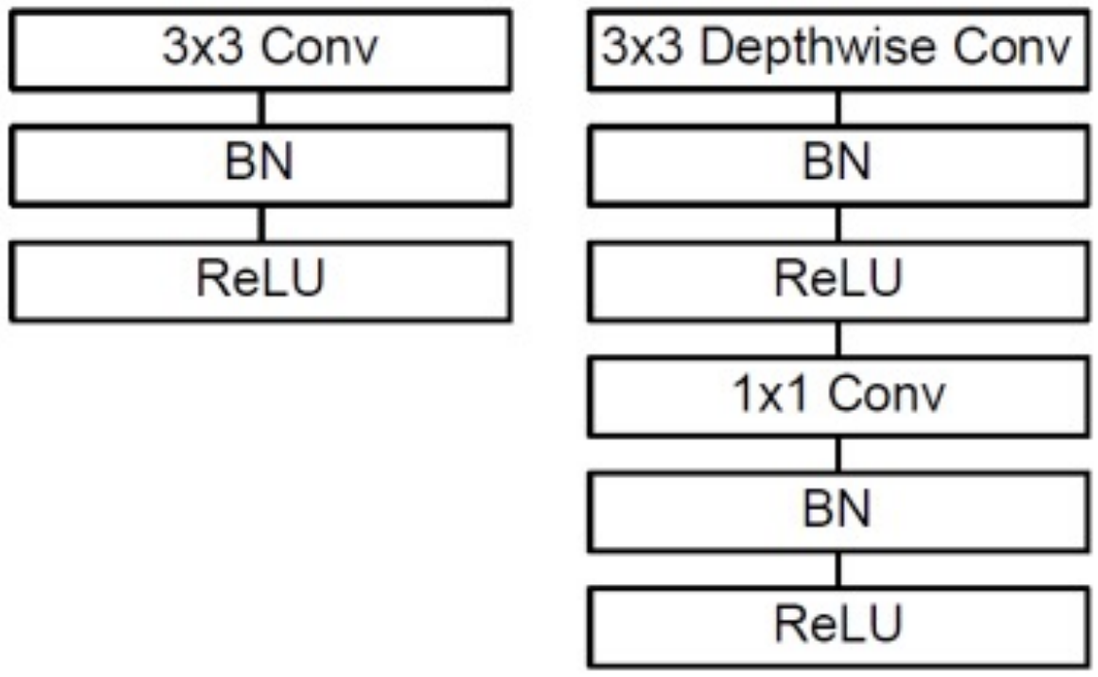


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

MobileNet V1

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet은 맨 처음 layer를 full conv로 쓰는 것을 제외하면 전부 DSConv로 사용한다.
dwConv는 표에는 Conv dw라고 표시되어 있다.
1×1 conv를 수행하는 부분이 pwConv이다.

Tensorflow, 그리고 Inception V3와 비슷하게 asynchronous gradient descent를 사용하는 RMSProp로 학습을 진행하였다.
DSConv에는 parameter가 별로 없어서 weight decay는 거의 또는 전혀 사용하지 않는 것이 좋다.

MobileNet V1

3.3 Width Multiplier : Thinner Models

이미 충분히 작지만 더 작고 빠르고 만들어야 하는 경우가 있다. Width multiplier라 부르는 α라는 hyper-parameter는 각 layer마다 얼마나 (전체적으로 다) 얇게 만드는지를 결정한다. 입출력 채널 수는 M, N에서 $\alpha M, \alpha N$이 된다. $\alpha \in (0, 1]$이고 0.75, 0.5, 0.25등의 값을 쓸 수 있다. $\alpha = 1$은 기본 MobileNet이다. 이 α를 통해 얼마든지 작은 시스템에도 모델을 집어넣어 사용할 수 있다. 물론 정확도, latency, 크기 사이에는 trade-off가 존재한다.

3.4. Resolution Multiplier: Reduced Represent

두 번째 hyper-parameter는 해상도에 관한 multiplier ρ이다. 입력 이미지와 각 레이어의 내부 표현 전부를 이 multiplier를 곱해 줄인다. 이 역시 α와 비슷하게 $\rho \in (0, 1]$이고 보통 이미지 해상도를 224, 192, 160, 128 정도로 만들게 한다. 계산량은 ρ^2에 비례하여 줄어든다.

Table 3. Resource usage for modifications to standard convolution. Note that each row is a cumulative effect adding on top of the previous row. This example is for an internal MobileNet layer with $D_K = 3$, $M = 512$, $N = 512$, $D_F = 14$.

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

MobileNet V1

4. Experiments

MobileNet을 여러 multiplier 등 여러 세팅을 바꿔가면서 실험한 결과인데, 주로 성능 하락은 크지 않으면서도 모델 크기나 계산량이 줄었음을 보여준다. 혹은 정확도는 낮아도 크기가 많이 작기 때문에 여러 embedded 환경에서 쓸 만하다는 주장을 한다.

MobileNet V1

4.1. Model Choices

Full conv와 DSConv의 차이는 명확하다. 정확도는 1%p 낮지만, 모델 크기는 7배 이상 작다.

또 Narrow와 Shallow MobileNet을 비교하면 아래와 같다. (깊고 얇은 모델 vs 얕고 두꺼운 모델)

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Table 5. Narrow vs Shallow MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.75 MobileNet	68.4%	325	2.6
Shallow MobileNet	65.3%	307	2.9

MobileNet V1

4.2. Model Shrinking Hyperparameters

모델이 작을수록 성능도 떨어지긴 한다.

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

MobileNet V1

계산량과 성능 상의 trade-off는 아래처럼 나타난다. 계산량이 지수적으로 늘어나면, 정확도는 거의 선형적으로 늘어난다.

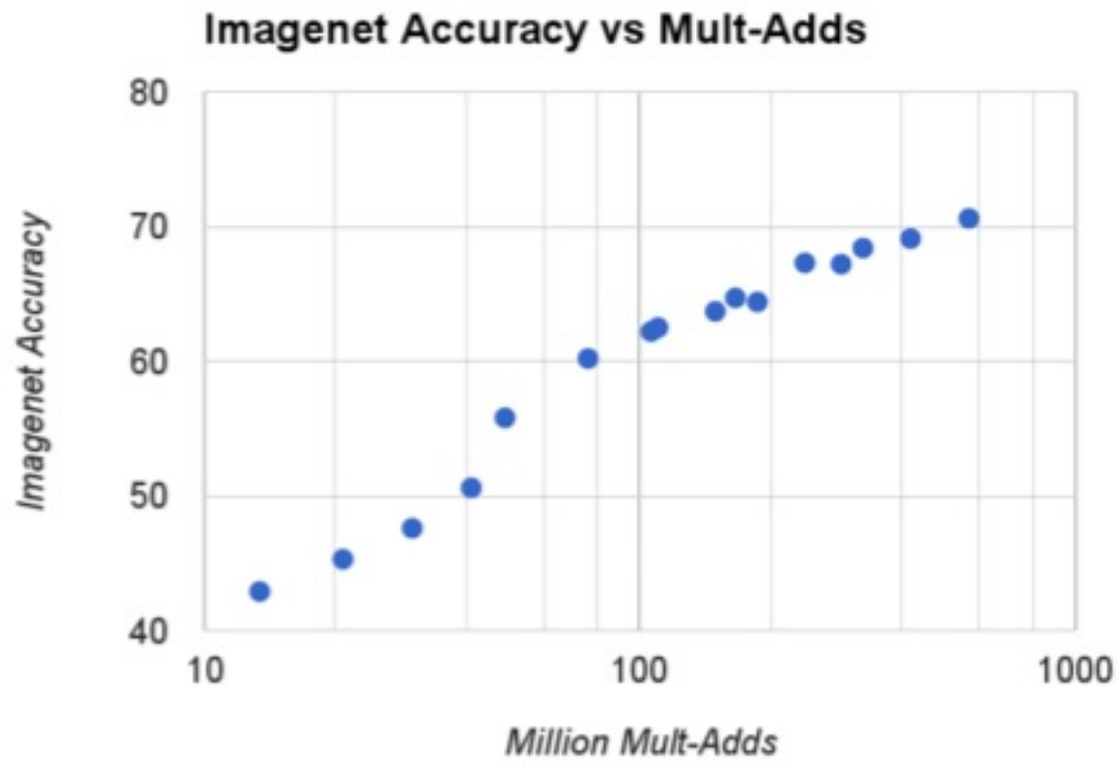


Figure 4. This figure shows the trade off between computation (Mult-Adds) and accuracy on the ImageNet benchmark. Note the log linear dependence between accuracy and computation.

MobileNet V1

정확도, 계산량, 모델 크기를 종합적으로 비교해보자.

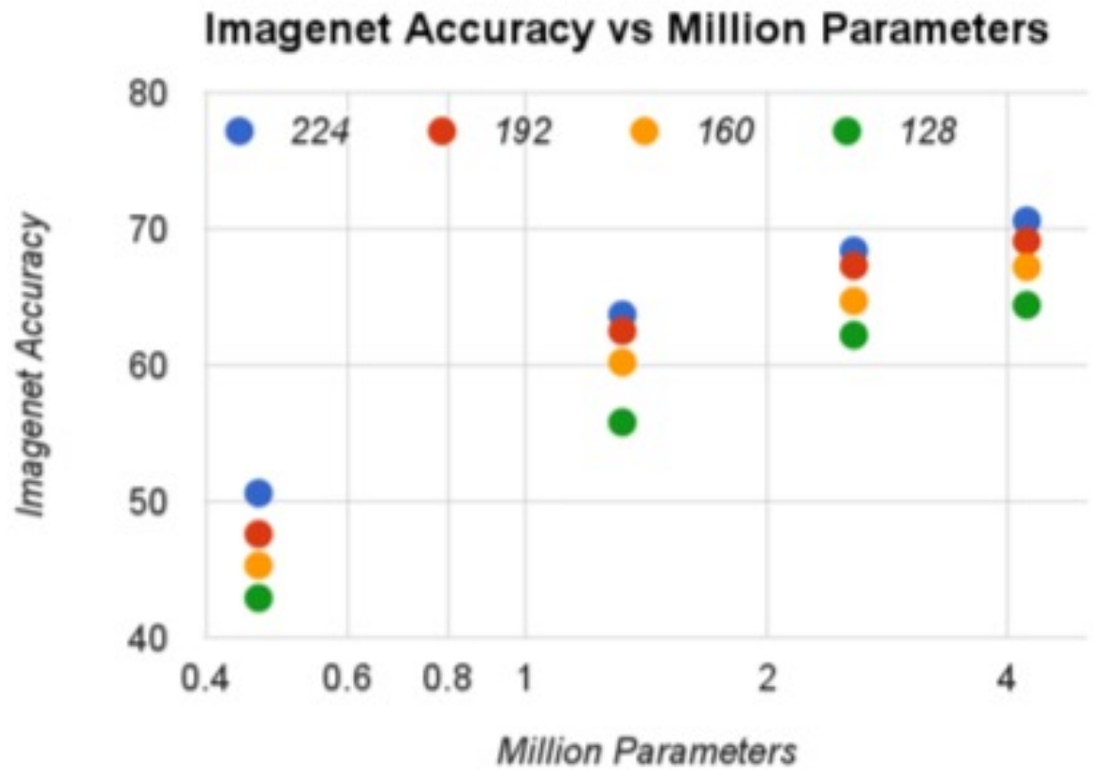


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

MobileNet V1

다른 모델(GoogleNet, VGG16 등)과 비교했을 때 MobileNet은 성능은 비슷하면서 계산량과 모델 크기에서 확실한 우위를 점한다.

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

MobileNet V1

4.3. SFine Grained Recognition

웹에서 얻은 대량이지만 noisy한 데이터를 사용하여 학습한 다음 Stanford Dogs dataset에서 테스트해보았다.

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9

Table 11. Performance of PlaNet using the MobileNet architecture. Percentages are the fraction of the Im2GPS test dataset that were localized within a certain distance from the ground truth. The numbers for the original PlaNet model are based on an updated version that has an improved architecture and training dataset.

Scale	Im2GPS [7]	PlaNet [35]	PlaNet MobileNet
Continent (2500 km)	51.9%	77.6%	79.3%
Country (750 km)	35.4%	64.0%	60.3%
Region (200 km)	32.1%	51.1%	45.2%
City (25 km)	21.9%	31.7%	31.7%
Street (1 km)	2.5%	11.0%	11.4%

MobileNet V1

5. Conclusion

Depthwise Separable Convolutions을 사용한 경량화된 모델 MobileNet을 제안하였다. 모델 크기나 연산량에 비해 성능은 크게 떨어지지 않고, 시스템의 환경에 따라 적절한 크기의 모델을 선택할 수 있도록 하는 여러 옵션(multiplier)를 제공하였다.

MobileNet V2

21. Mar 2019

Introduction & Related work

Modern state of the art networks require high computational resources beyond the capabilities of many mobile and embedded Applications.

Recently opened up a new direction of bringing optimization methods including genetic algorithms and reinforcement learning to architectural search. However one drawback is that the resulting networks end up very complex.

강화학습으로 구조를 검토하면 성능은 좋아지지만 구조는 복잡해 지더라...

MobileNet V2

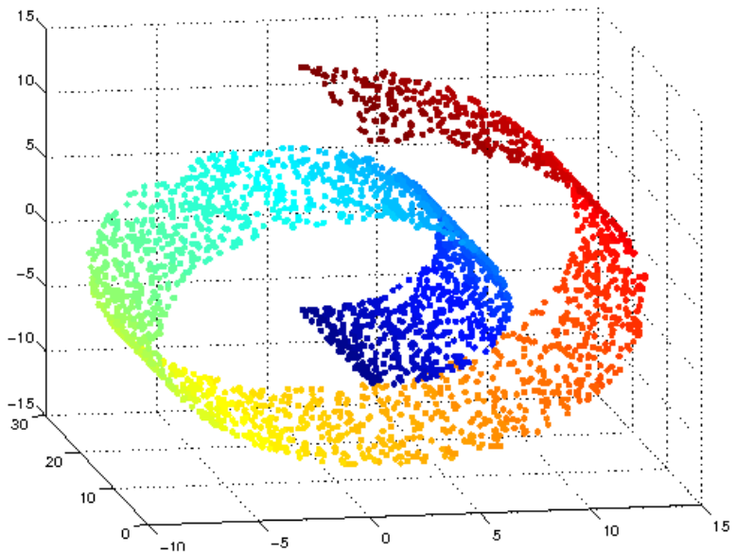
Key Features

- Depthwise Seperable Convolutions
- Linear Bottlenecks
- Inverted Residuals

MobileNet V2 – linear Bottlenecks

• Informally, for an input set of real images, we say that the set of la yer activations forms a “manifold of interest”

• It has been long assumed that manifolds of interest in neural netw orks could be embedded in low-dimensional subspaces.



원은 모든 점에 대해서 국소적으로 직선과 같은 구조를 가지고 있다. 따라서, 원은 1차원 다양체이다.

MobileNet V2 – linear Bottlenecks

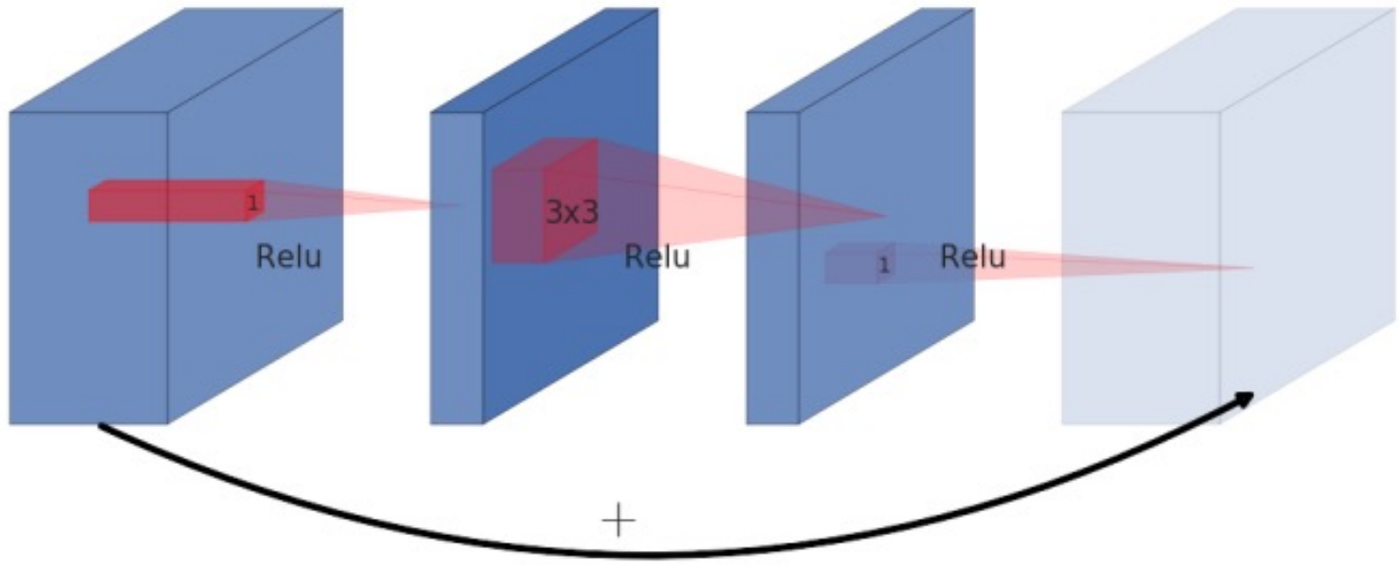
• The authors have highlighted two properties that are indicative of the requi rement that the manifold of interest should lie in a lowdimensional subspace of the higher-dimensional activation space

1. If the manifold of interest remains non-zero volume after ReLU transform ation, it corresponds to a linear transformation.
2. ReLU is capable of preserving complete information about the input mani fold, but only if the input manifold lies in a low-dimensional subspace of the input space.

• Assuming the manifold of interest is low-dimensional we can capture this b y inserting linear bottleneck layers into the convolutional blocks

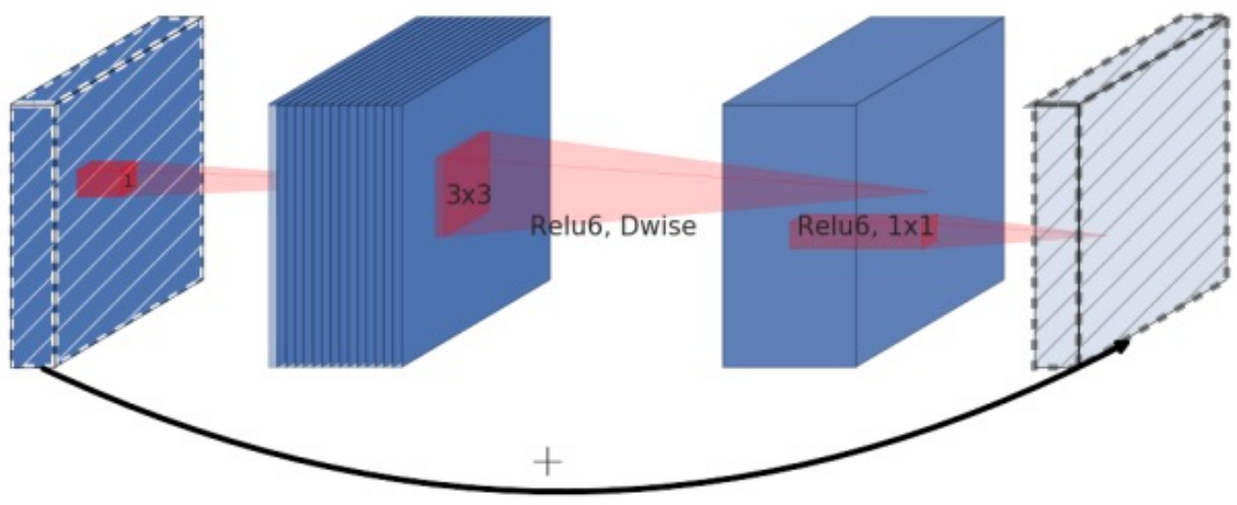
MobileNet V2 – Residual Blocks

- Residual blocks connect the beginning and end of a convolutional block with a shortcut connection. By adding these two states the network has the opportunity of accessing earlier activations that weren't modified in the convolutional block.
- wide → narrow(bottleneck) → wide approach



MobileNet V2

- Inspired by the intuition that the bottlenecks actually contain all the necessary information, while an expansion layer acts merely as an implementation detail that accompanies a non-linear transformation of the tensor, the authors use shortcuts directly between the bottlenecks
- narrow → wide → narrow approach

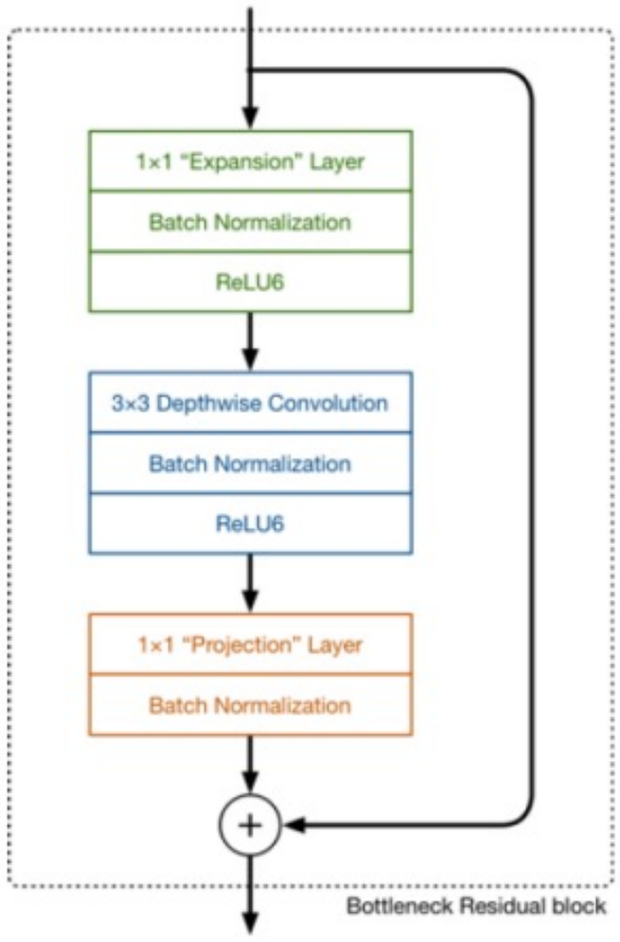


MobileNet V2

- The basic building block is a bottleneck depth-separable convolution with residuals

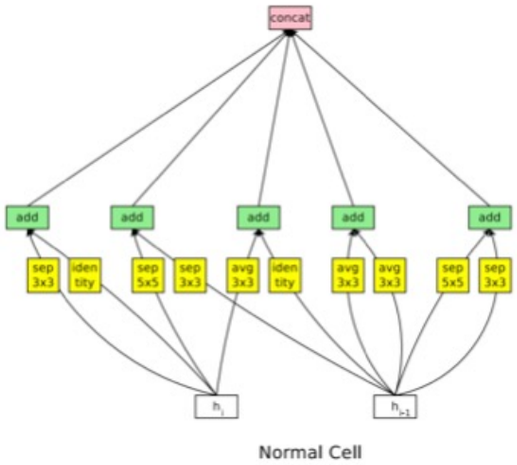
Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Table 1: *Bottleneck residual block* transforming from k to k' channels, with stride s, and expansion factor t.

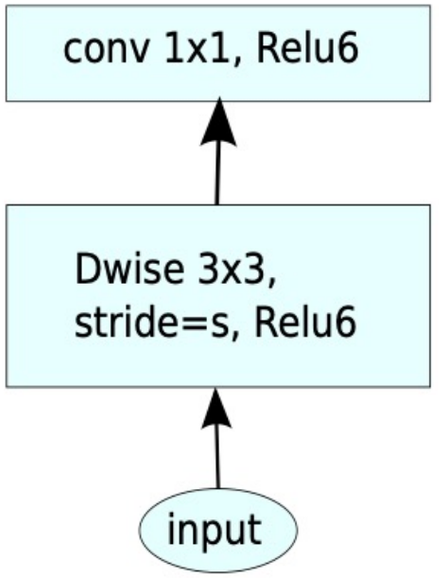


MobileNet V2

2018년 4월



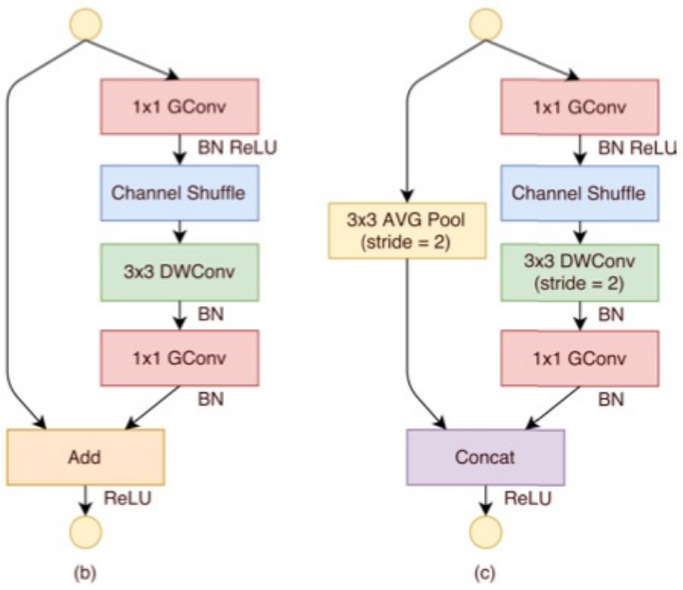
(a) NasNet[23]



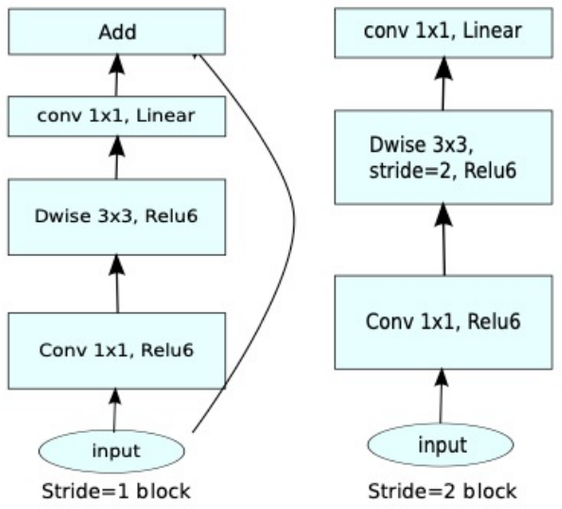
(b) MobileNet[27]

2017년 4월

2017년 12월



(c) ShuffleNet [20]



(d) Mobilenet V2

2019년 3월

MobileNet V1 VS MobileNet V2 VS ShuffleNet

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	1/O(1)	1/O(1)	1/O(1)
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	800K	**200K**	600K

Top-1 Accuracy

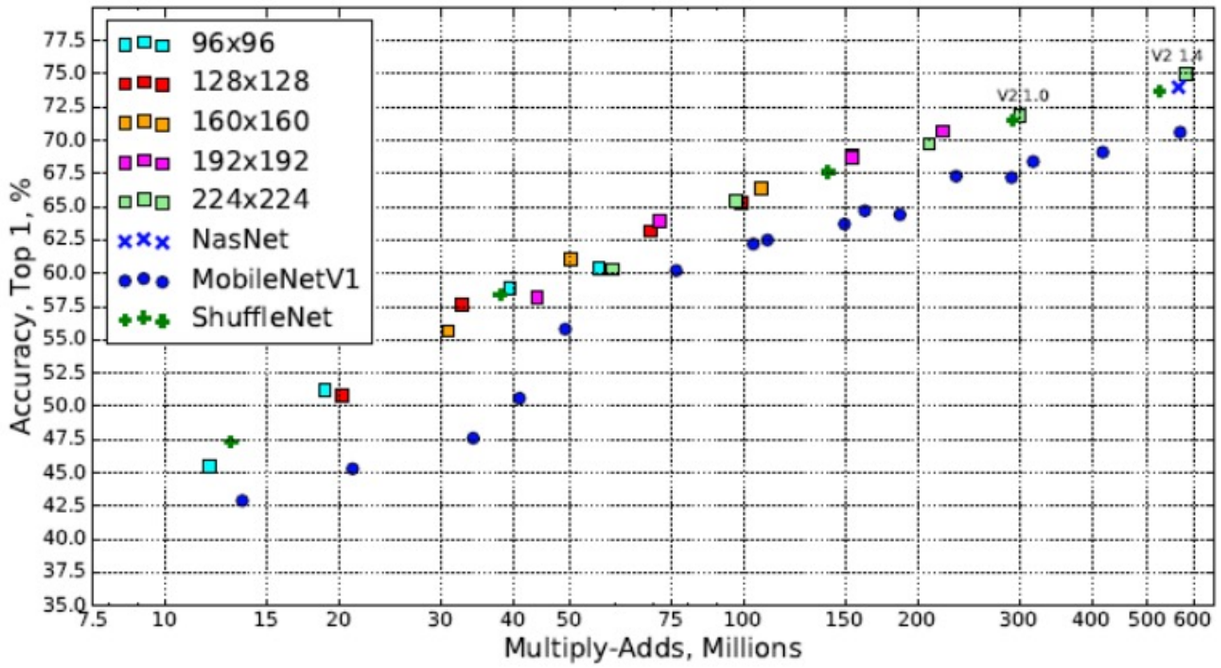


Figure 5: Performance curve of MobileNetV2 vs MobileNetV1, ShuffleNet, NAS. For our networks we use multipliers 0.35, 0.5, 0.75, 1.0 for all resolutions, and additional 1.4 for for 224. Best viewed in color.

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table 4: Performance on ImageNet, comparison for different networks. As is common practice for ops, we count the total number of Multiply-Adds. In the last column we report running time in milliseconds (ms) for a single large core of the Google Pixel 1 phone (using TF-Lite). We do not report ShuffleNet numbers as efficient group convolutions and shuffling are not yet supported.

MobileNet V1 VS MobileNet V2 Apple

Version	MACs (millions)	Parameters (millions)
MobileNet V1	569	4.24
MobileNet V2	300	3.47

Version	iPhone 7	iPhone X	iPad Pro 10.5
MobileNet V1	118	162	204
MobileNet V2	145	233	220

Version	Top-1 Accuracy	Top-5 Accuracy
MobileNet V1	70.9	89.9
MobileNet V2	71.8	91.0

MobileNet V2