Deep Residual Learning for Image Recognition

Part 0

논문의 개요

- 본 논문에서는 잔여학습을 적용한 CNN으로 깊은 네트워크의 성능이 떨어지는 문제를 해결할 수 있다.
- 논문의 contribution : 일반적인 생각(깊이가 깊을수록 성능이 좋을 것)을 유지

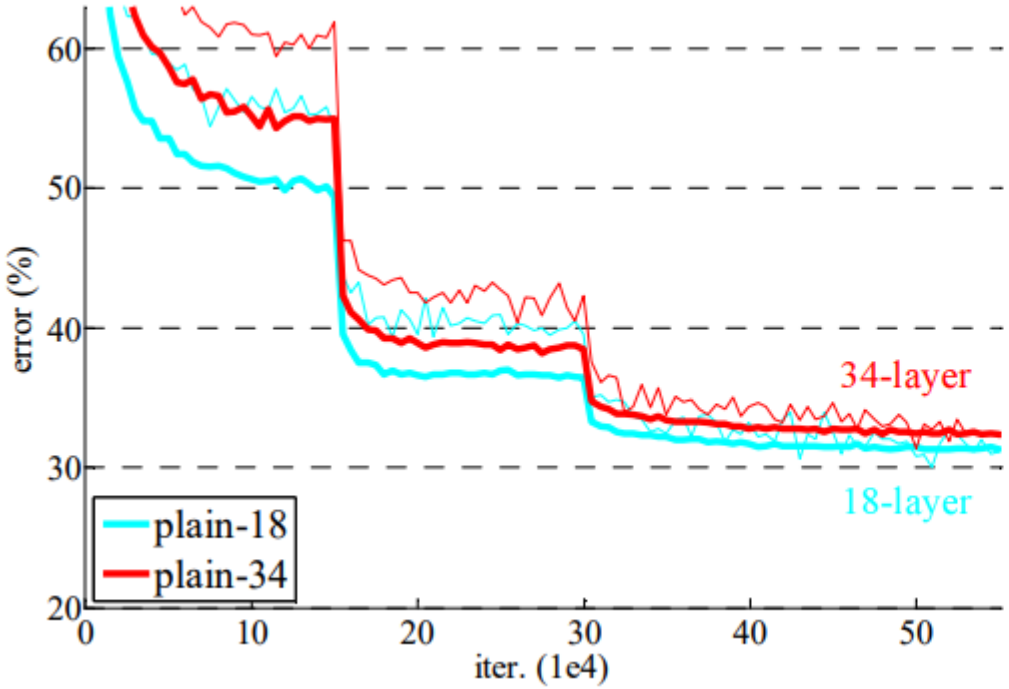
- VGG network (Very Deep Convolutional Networks for Large-Scale Image Recognition)

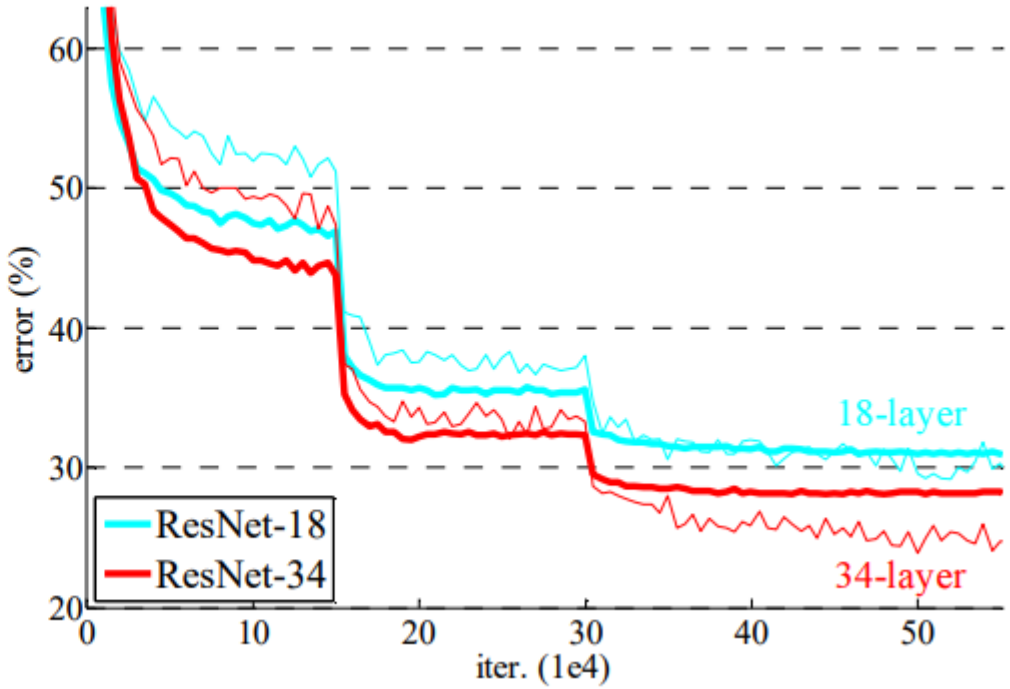
⇒ 작은 크기의 3x3 convolution filter를 이용해 레이어의 깊이를 늘려 우수한 성능을 보인다.

⇒ VGG-16, VGG-19

논문의 개요

- 일반적으로 네트워크가 깊을수록 다양한 특징을 추출할 수 있어서 좋은 성능을 보일 것이라는 생각을 할 수 있으나 너무 깊어지면 오히려 성능이 떨어진다.





Abstract

- 더 깊은 네트워크를 정확하게 학습하기 위해 “residual learning framework”를 제안

⇒ VGG net 보다 8배 깊은 152 layer의 residual network가 VGG보다 더 정확하고 복잡도도 낮았다.

⇒ 특히, residual network로 앙상블 모델을 만든 결과 3.57%의 매우 낮은 에러로 성능이 좋았다.

- 네트워크의 깊이는 visual recognition task에서 중요하다.

⇒ deep residual net은 “object detection” 또는 “segmentation” 등의 분야에 응용될 수 있다.

Introduction

- 네트워크의 깊이가 깊어질수록 잘 학습할까?

=> 논문이 발표될 당시 deep model일 수록 성능이 향상된다는 것이 일반적인 견해

⇒ vanishing/exploding gradients 문제점 발생 : 학습 초기에 수렴을 방해

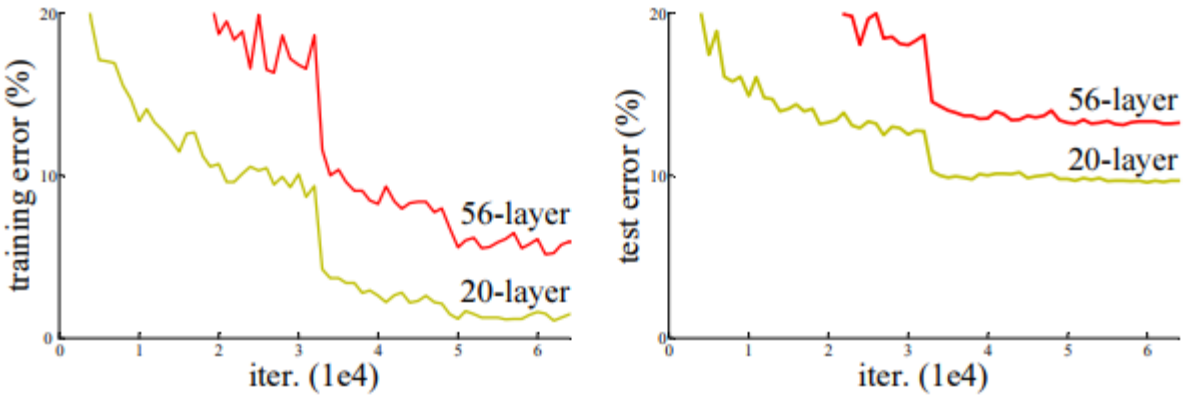
⇒ 그러나 이 문제점은 네트워크의 가중치 값을 초기에 적절히 초기화 함으로써 해결할 수 있다.

- 좀 더 깊은 deeper network가 수렴을 시작할 때 “degradation problem”이 발생

⇒ degradation problem : 깊이가 어느정도 깊어지면 정확도가 떨어지는 것

⇒ overfitting 때문이 아님

⇒ high training error를 야기한다.



Introduction

- degradation problem 해결 : identity mapping을 이용한 deeper model construction

⇒ 핵심 idea : layer를 깊게 쌓되 많은 layer들이 identity mapping으로 이루어지도록 만든다면 deeper model이 shallow model에 비해 높은 training error를 가지면 안된다

⇒ deep residual learning framework(ResNet)을 제안

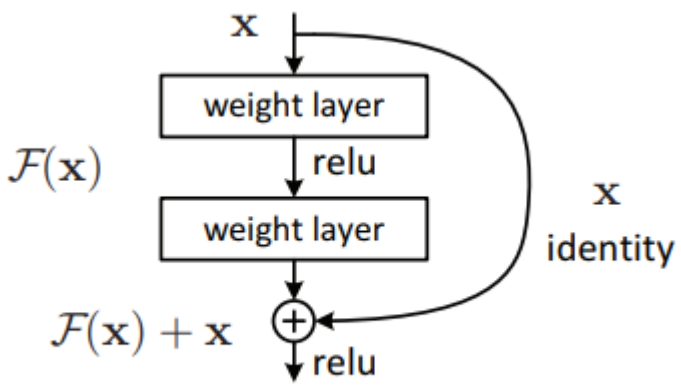


Figure 2. Residual learning: a building block.

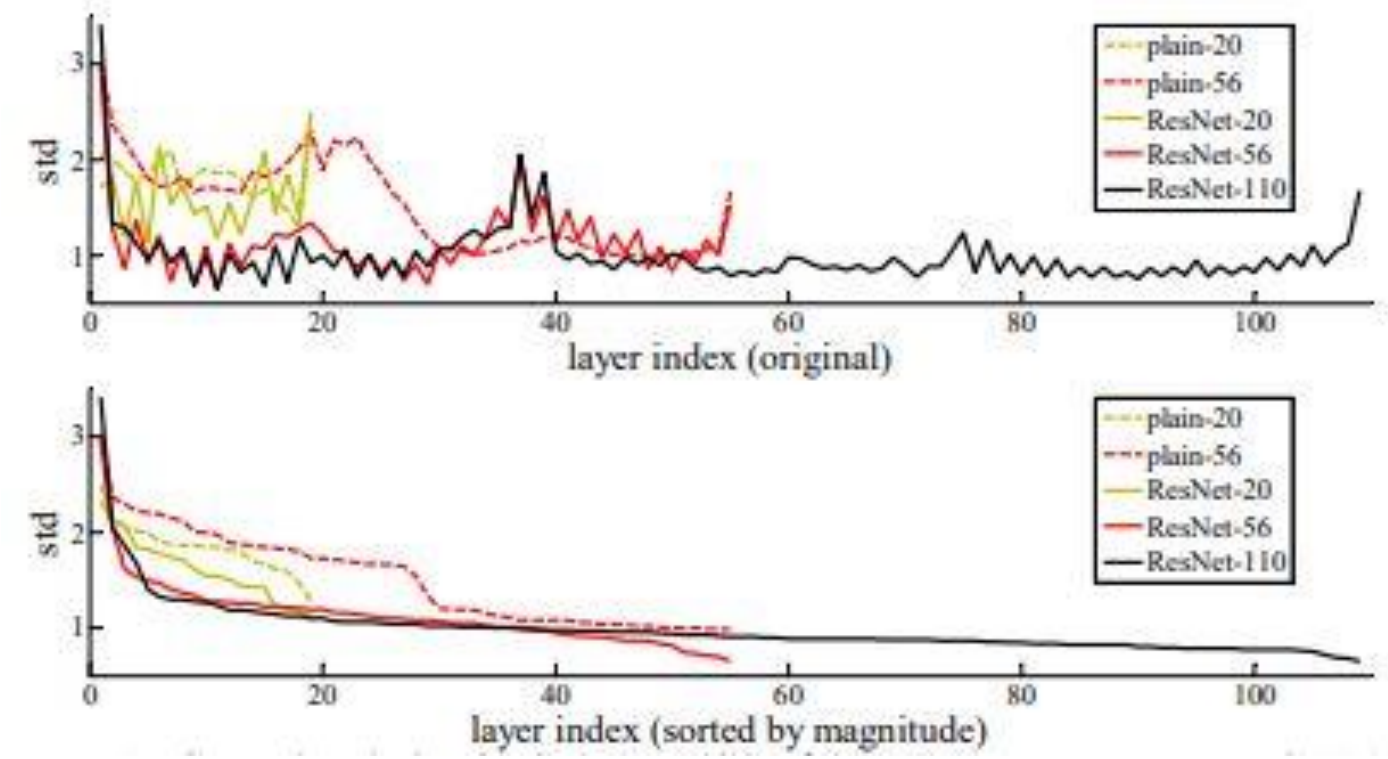
- H(x) : 기존 neural net은 H(x)를 최소화하는 것이 목표
- F(x) := H(x) – x (reformulation)
- 이 논문에서는 H를 학습하는 대신, F를 학습하고자 한다.

⇒ 가정) unreferenced mapping인 H(x)를 최적화하는 것보다 identity mapping시킨 residual mapping F(x)가 더 최적화가 쉽다

Part 2

Introduction

residual network가 작은 response값을 가지고 있고 이는 학습이 쉬울 것이라는 것을 대변



Introduction

- shortcut connection : x를 더해주는 것

⇒ 장점 1: identity mapping이랑 같다.

⇒ 장점 2: 추가적인 parameter, computational complexity를 요구하지 않는다.

⇒ 장점 3: common library를 이용해서 구현할 수 있다.

- 이렇게 구현한 deep residual net은 깊이가 깊어질 수록 정확도도 높았다.

⇒ ImageNet dataset, CIFAR-10에서도 (특정 dataset에 국한되지 않았다.)

⇒ 특히, 앙상블 모델은 3.57%의 top-5 error를 보였다.

Deep Residual Learning

1. Residual Learning

- **자명한 사실** : 얕은 모델에서 단순히 아무것도 하지 않는 layer인(convolution을 통과하지 않고 값을 전달하는) identity mapping을 쌓으면(덧셈 연산으로) 얕은 모델 그대로의 성능을 나타낸다.
- **가정** : 쌓여 있는 레이어가 underlying mapping을 fit하는 것보다 residual mapping을 fit하는 것이 쉽다. 그리고 shortcut connection이 이 역할을 정확하게 할 수 있다.

=> 즉, 이전에 학습된 모델(레이어들)의 출력과 추가된 레이어의 출력의 차이 값인 나머지(residual)만 학습하면 되기에 연산이 간단해 진다.

Deep Residual Learning

2. Identity Mapping by Shortcuts

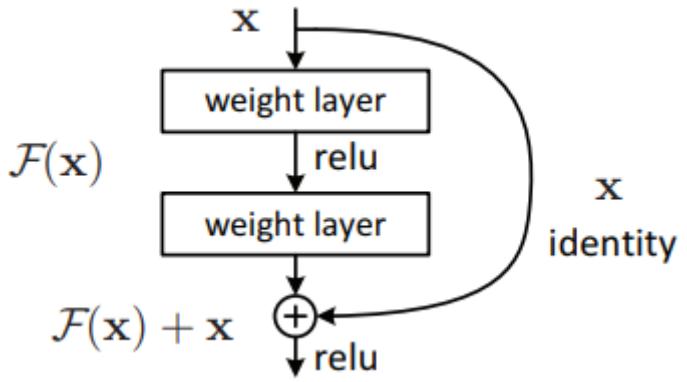
- x와 F의 dimension이 같을 때

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

- x와 F의 dimension이 다를 때 : shortcut connection에 linear projection(Ws)를 해서 dimension을 맞춰 줌

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$



Part 4

Deep Residual Learning

3. Network Architectures

- VGG-19 : 일반적으로 사용되는 CNN
- 34-layer plain : 논문에서 사용되는 baseline, VGG 기반

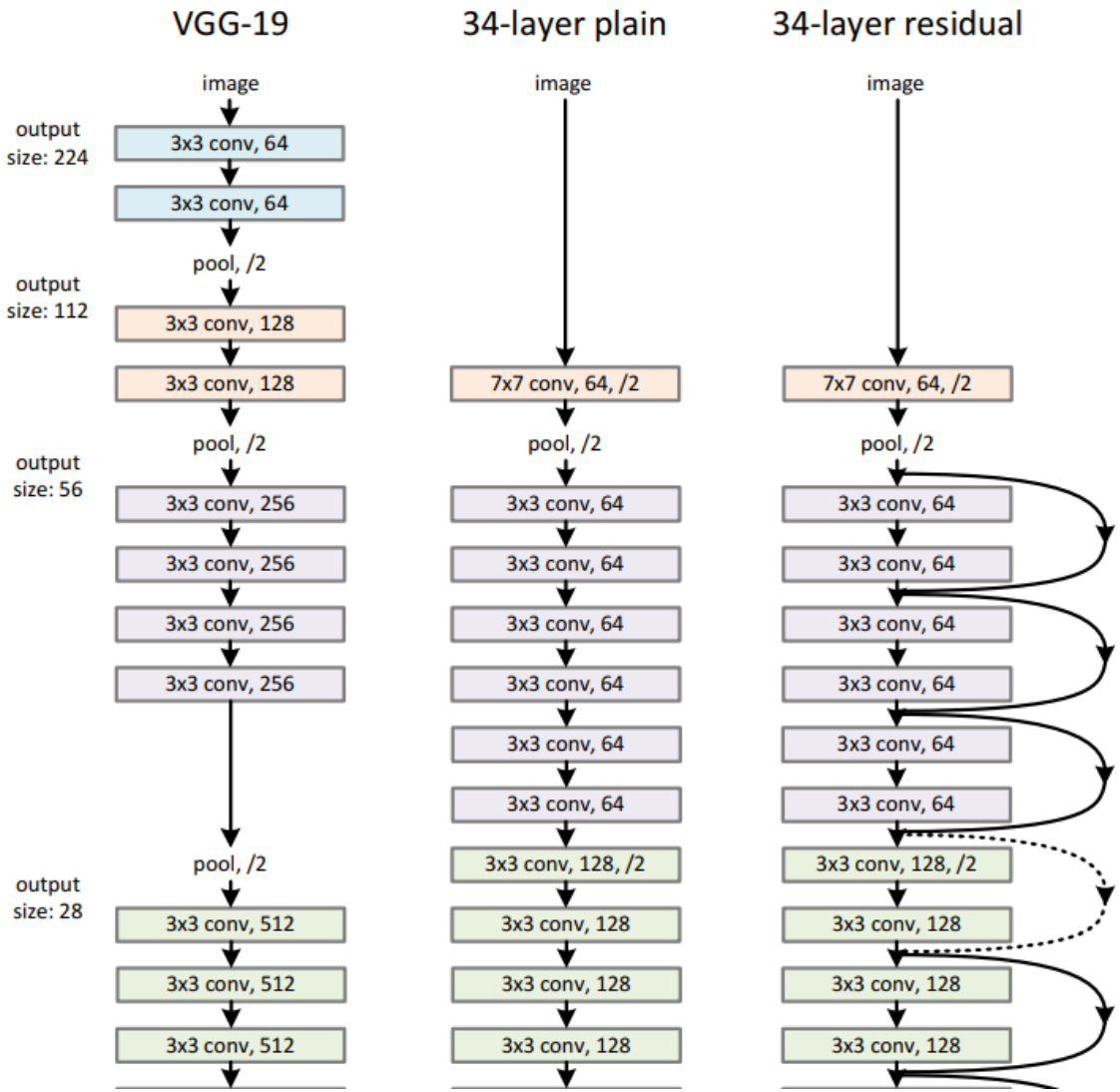
⇒ 모든 filter의 크기가 같음 (3X3)

⇒ stride 2로 down-sampling할 때에는 channel수를 두 배로 늘려 time complexity를 유지

- 34-layer residual

⇒ 34-layer plain 기반

• 이러한 구조로 VGG에 비해 필터의 개수를 낮추어 parameter 수를 줄였고 복잡도를 낮춰 연산도 줄였다.



Experiments

1. Plain Networks

- deeper network가 shallower network 보다 에러 ↑
- optimization difficulty가 발생하는 이유

⇒ vanishing gradient 때문이 아니라 low convergence rate 때문이다.

⇒ convergence rate : 수렴을 위해 필요한 epoch, 난이도 등을 나타내기 위한 척도로 training error을 낮추는 데 기여

2. Residual Networks

- Resnet은 초기 수렴 속도를 빠르게 해줘서 optimization 용이

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Experiments

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

A : Zero padding으로 dimension을 늘려서 identity mapping
B : dimension이 증가할 때만 projection
C : 모든 shortcut에 대해서 projection

⇒ 별 차이 없음 : degradation problem을 해결하는 데 projection shortcut이 주요한 요소는 아니다.

⇒ 근데 Deeper Bottleneck Architecture에서는 중요

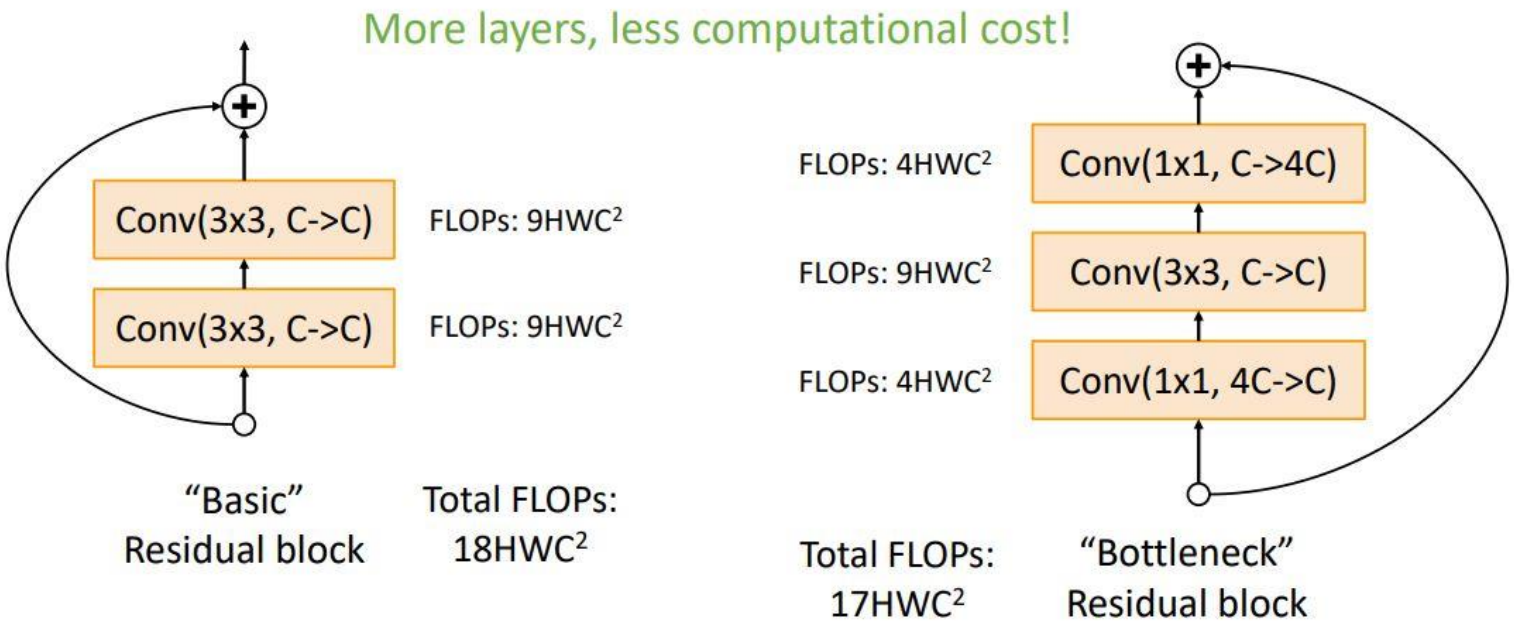
Deep Residual Learning

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				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	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

50-layer 이상부터 구조가 조금 달라진 것을 볼 수 있는데 이는 layer가 많아질 수록 parameter가 많아지고 FLOP 수가 늘어나 연산이 많아져 complexity가 높아진다.

=> 이를 해결하기위해 bottleneck architecture를 사용

Experiments



identity mapping이 deeper bottleneck에서 더 중요한 이유?

더 작은 커널은 파라미터의 수가 적은데 identity mapping은 파라미터가 존재하지 않아서 bottleneck에서 파라미터를 효과적으로 줄일 수 있다.

Experiments

CIFAR-10 test set에서 classification을 했을 때에도 기존 방법보다 parameter의 수는 적지만 성능은 좋다.

method			error (%)
Maxout [10]			9.38
NIN [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	0.46M	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

Experiments

1. 그럼 무조건 layer가 깊어질수록 성능이 좋을까?

⇒ 불필요하게 layer가 깊어지면 (1000 layer 이상) 오버 피팅으로 인해 성능이 떨어진다.

2. ResNet은 Classification 외에도 object detection이나 segmentation에도 뛰어난 성능으로 활용될 수 있다.