

# Deep Residual Learning for Image Recognition

21 신혜빈  
21 우진주



# CONTENTS



Deep Residual Learning for  
Image Recognition

**01** Introduction

**02** Related Work

**03** Deep Residual Learning

**04** Experiment

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 이전 연구 논문

<2015, VGG Network>, <CVPR, 2015 GoogleNet>

: 두 논문 모두 네트워크가 깊어짐에 따라 얻을 수 있는 장점들을 적절히 취했다.

-> 그렇다면, 단순히 레이어를 깊게 쌓으면 더 좋은 네트워크를 학습할 수 있는 것인가에 대한 의문

## 아이디어 발전

단순히 레이어를 깊게 쌓는 것

(1) 문제 : vanishing/exploding gradient

(2) 제안된 해결책(논문)

: "초기에 네트워크 가중치값들을 적절히 초기화 해두어 학습이 잘 이루어질 수 있도록 하자"

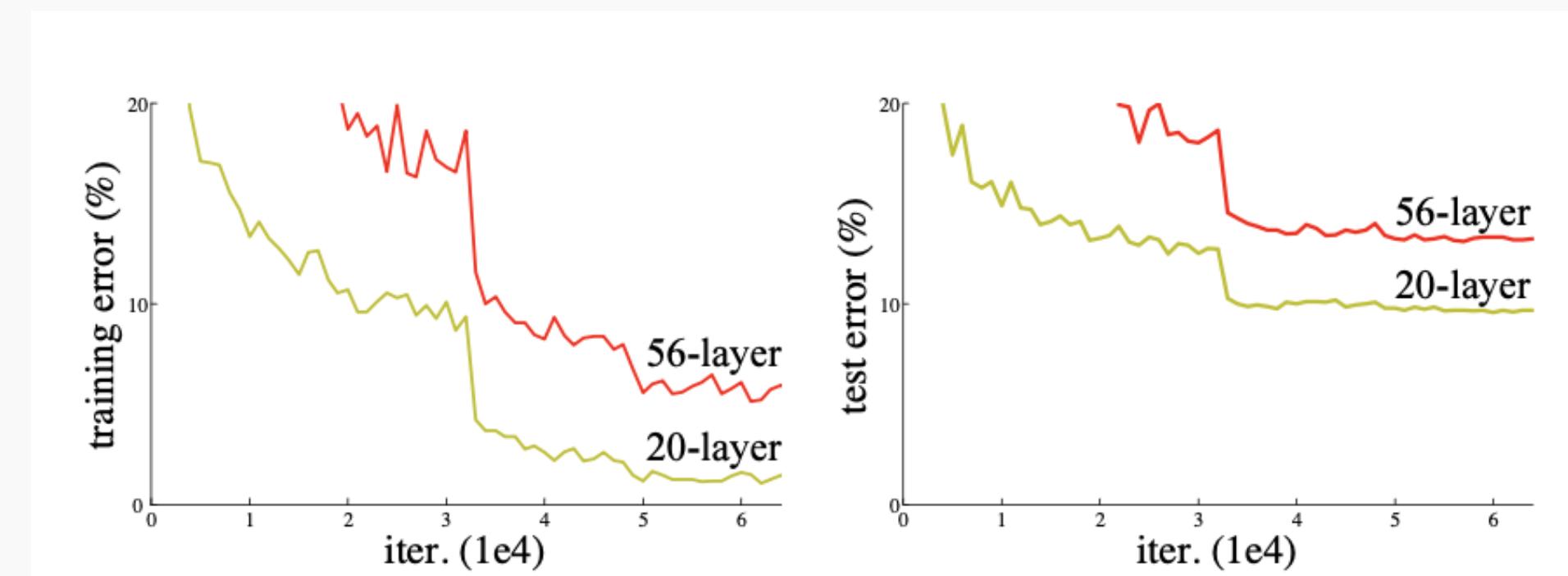
1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 본 논문의 주장



"레이어가 깊어짐에 따라 degradation problem이 발생할 수 있다"

"즉, accuracy는 레이어가 어느 정도 깊어지면 오히려 감소할 수 있다"

(단순히 overfitting 때문에 발생하는 것은 아니며, 레이어를 깊게 쌓는 것은 training error 를 증가시킨다.)

Figure 1 : training error, test error 모두 레이어가 깊어도 error가 높다.

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## Degradation problem

(1) 정의 : 네트워크가 깊어질수록 정확도가 감소하는 현상

(2) 간단히 떠올릴 수 있는 해결책

: 레이어를 깊게 쌓기 위해 단순히 "**identity mapping**" 만 증가

\* identity mapping : 입력값과 출력값이 같은 함수, 보통 네트워크 깊이 증가에 쓰인다.

1 Introduction

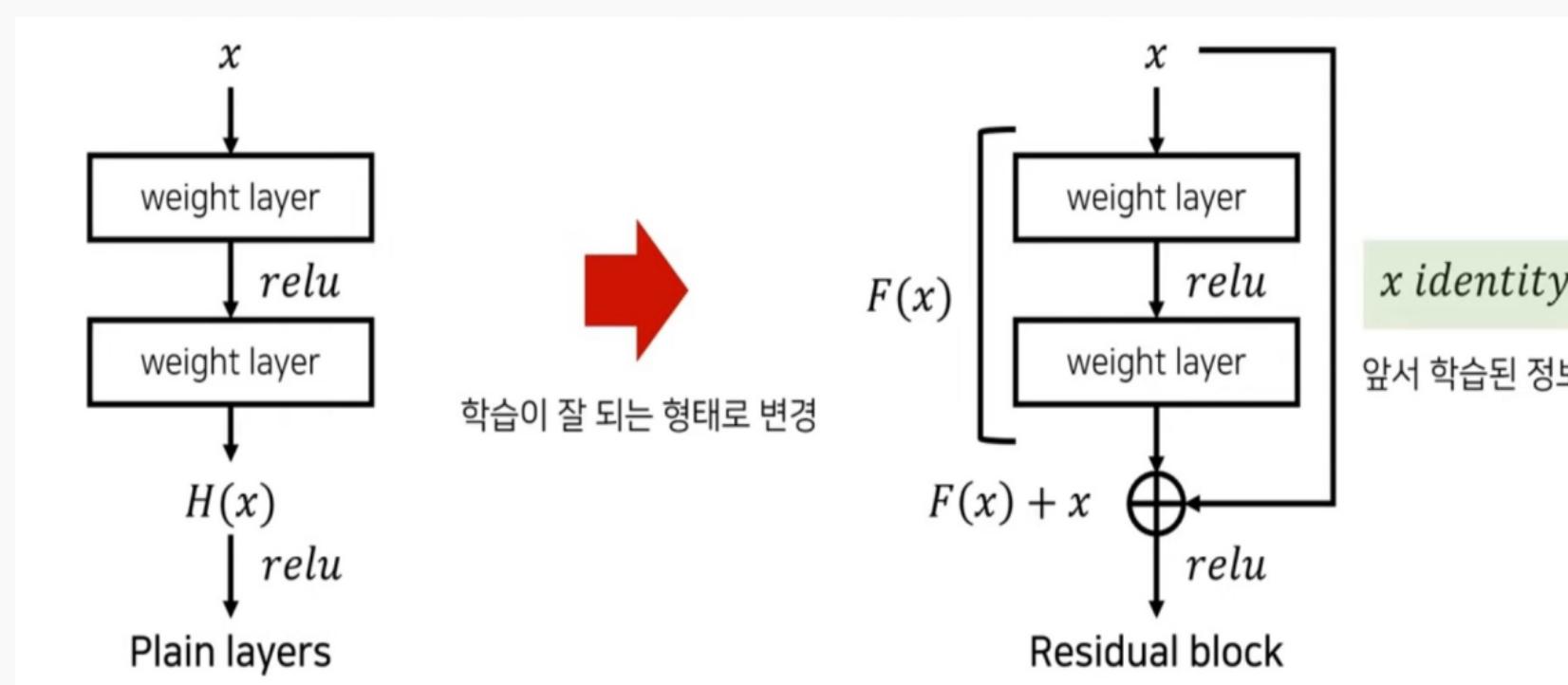
2 Related Work

3 Deep Residual Learning

4 Experiment

## Residual block

### A deep learning framework



" $H(x)$ 를 학습하는 것보다  
 $F(x)$ 를 학습하는 것이 낫다"

- (1) 의도했던 mapping  $H(x)$ 를 direct하게 학습하기보다는,  
명시적으로 더 학습하기 쉬운 residual mapping을 따로 정의하여 대신 학습
- (2)  $H(x)$ 가 아닌  $F(x)$ 를 대신 학습 ( $F(x) = H(x)-x$ )
- (3)  $F(x)+x$  : 실제로 내보내는 output값
- (4) 극단적인 경우( $H(x)$ 가  $x$ 인 경우) : 함수  $F(x)$ 가 0이 될 수 있도록 하는 게 학습 난이도가 쉬울 것이다.

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## Shortcut connection

(1) 정의 : 결과에  $x$ 를 더하는 것

(2) 본 논문에서는 shortcut connection을 단순히 **identity mapping**으로 사용 가능하다.

(3) 장점

1) 별도의 추가적인 parameter 불필요하다.

2) 복잡도가 증가하지 않는다 (출력값에  $x$ 를 더해주는 것으로 끝나기 때문)

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## Residual Network

(1) 장점

- 1) 학습 난이도 감소
- 2) (네트워크) 깊이가 깊어질수록 높은 accuracy를 가진다.

(2) 특정 데이터셋에서만 국한된 방법이 X

: CIFAR-10 과 ImageNet에서도 공통적으로 ResNet을 이용했을 때 성능이 개선되었고,  
양상을 기법 적용 시 top-5 error로 3.57%의 좋은 accuracy를 보였다.

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## Related Work

### (1) Residual Representation

: residual한 technique들이 본 논문 이전부터 꾸준히 사용된 방법임을 언급한다.

### (2) Shortcut Connections

: 최근 딥러닝 연구에서도 많이 사용되고 있다.

### (3) 이전 논문과의 차이점

① 본 논문은 항상 residual function을 학습하는 형태로 네트워크를 구성한다.

② identity mapping은 매번 사용되어 더해진다.

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.1. Residual Learning

(1)  $H(x)$ 를 원래 의도한 optimal한 mapping이라 보았을 때, 뉴런 네트워크의 역할은 여러 개의 nonlinear layer를 이용해 어떤 복잡한 함수를 점진적으로 학습하는 것이다.

(2) "residual function을 이용한다" ->  $H(x) - x$ , 즉 "F(x)를 학습한다"

$H(x)$ 를 학습하는 것과의 차이점은  $F(x)$ 를 학습할 때 난이도 ↓

추가적으로 더해지는 레이어가 identity mapping이라 하면 최소한 얇은 모델보단  
깊은 모델의 training error가 더 높아지지는 않을 것

(3) 항상 이전 입력값을 출력값으로 하여 기본적인 identity mapping을 수행할 수 있게 한다.

-> 학습 난이도 ↓

(4) 결과적으로 identity mapping을 이용해 residual function이

더 추가적인 잔여 정보만 학습하도록 만드는 것이 더 좋은 성능을 낸다.

다시 말해, 이전 layer에서의 값인  $x$ 를 보존하고, 추가적으로 필요한 정보를 학습하는 방식으로 동작한다.

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 3.2. Identity Mapping by Shortcut

(1) 하나의 block 정의

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \quad (1)$$

- 1)  $\mathcal{F}$  : residual mapping
- 2)  $x$  : identity mapping ( 즉, shortcut connection)
- 3) Fig 2에서  $\mathcal{F} = W_2\sigma(W_1x)$ , 즉 weight 값을 2번 중첩하여 사용한 함수

①  $W_1$ : first weight

②  $W_2$ : second weight

③  $\sigma$ : activation function

참고로, bias 값은 단순히 고려하지 않도록 정의됨

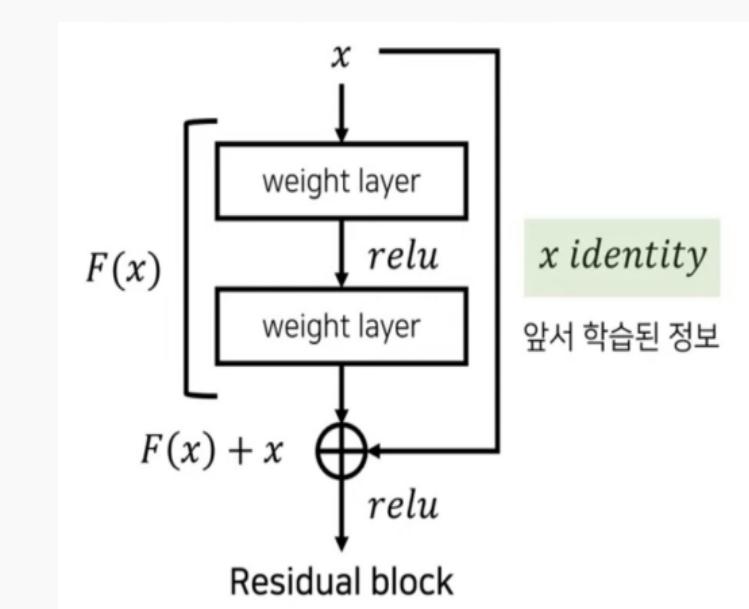


Fig 2.

1 Introduction

2 Related Work

3 Deep Residual Learning

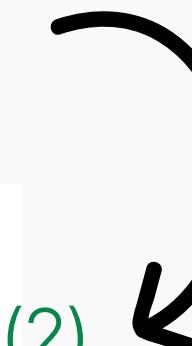
4 Experiment

### 3.2. Identity Mapping by Shortcut

(2)  $x$ 를 추가적으로 shortcut connection을 통해 mapping

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}. \quad (1)$$

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s \mathbf{x}. \quad (2)$$



input dimension( $x$ )과 output dimension( $\mathcal{F}(x, \{W_i\})$ )이 일치하지 않는다고 하면,  
 $W_s$ 를 곱해줌으로써 linear projection을 통해 dimension 값을 match시킬 수 있다.

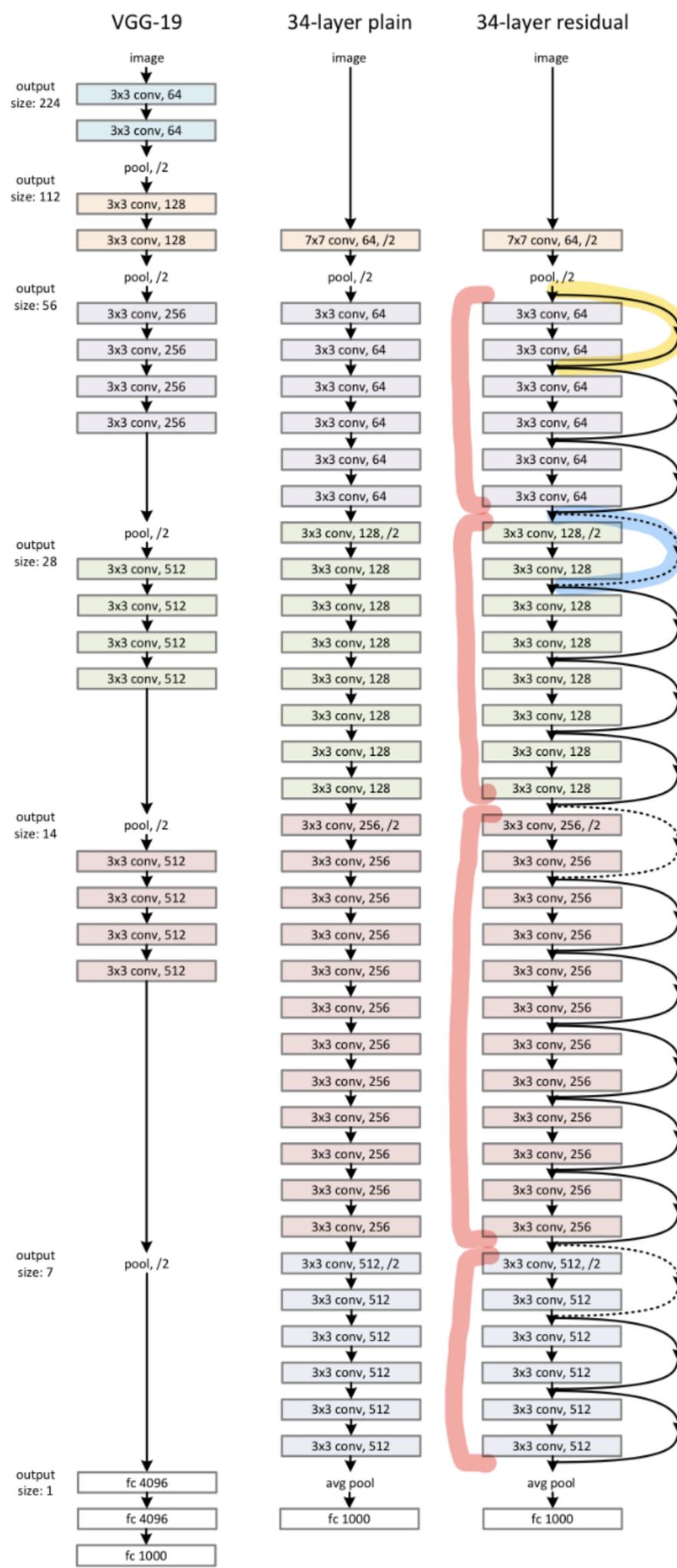
그러나, identity mapping을 이용했을 때에도(1) 충분하게 높은 성능을 보일 수 있다.  
여러 개의 weight 값이 중첩된 형태로 사용될 때 실제로 유의미한 성능을 보인다.

## 1 Introduction

## 2 Related Work

## 3 Deep Residual Learning

## 4 Experiment



## 3.3. Network Architecture

**Figure 3**

### (1) VGG-19

: 이전 연구에서 제안되었던 19 layer VGG Network

### (2) 34-layer plain

: 일반적인 VGG Network과 비슷하게 3 x 3 convolution filter를 사용

### (3) 34-layer residual

: 34-layer plain에서 residual block만 사용하는 형태로 바꾼 모델

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture

#### Plain Network

- (1) 비교 목적으로, 기본적인 CNN 네트워크를 가져와서 실험 진행
- (2) 기본적인 네트워크들은 VGG 네트워크에서 제안된 기법들을 적절히 따름
  - ① 3 by 3짜리 작은 필터를 이용
  - ② output feature map size를 동일하게 하기 위해 같은 개수의 필터 사용
  - ③ feature map size가 절반으로 줄어들 때 (즉, 그 다음 레이어에서 채널 값을 2배로 늘림)  
∴ 이런 방법들로 레이어마다 time complexity를 보존할 수 있는 형태로 네트워크 구성
- (3) 별도 pooling layer를 사용하지 않고, convolution layer의 stride값 2로 설정 downsampling 진행
- (4) 레이어 마지막 부분에서 average pooling 이용하여 결국 1,000개의 클래스로 분리할 수 있도록 만듦

1 Introduction

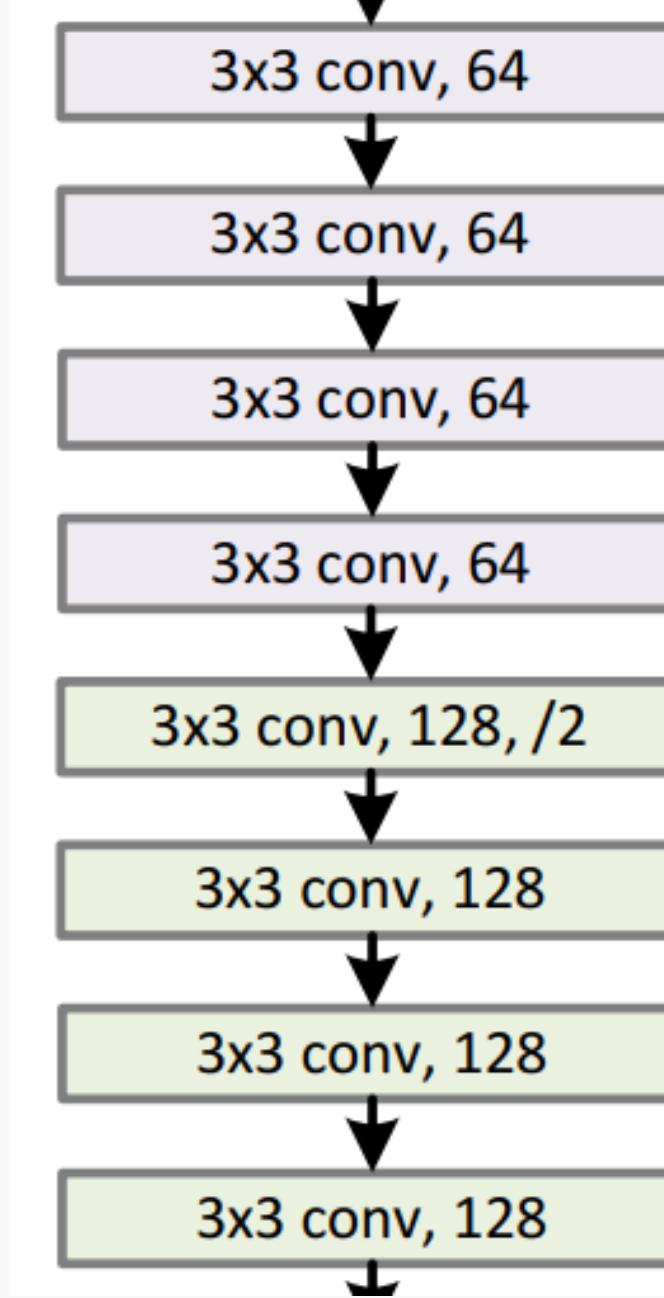
2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture

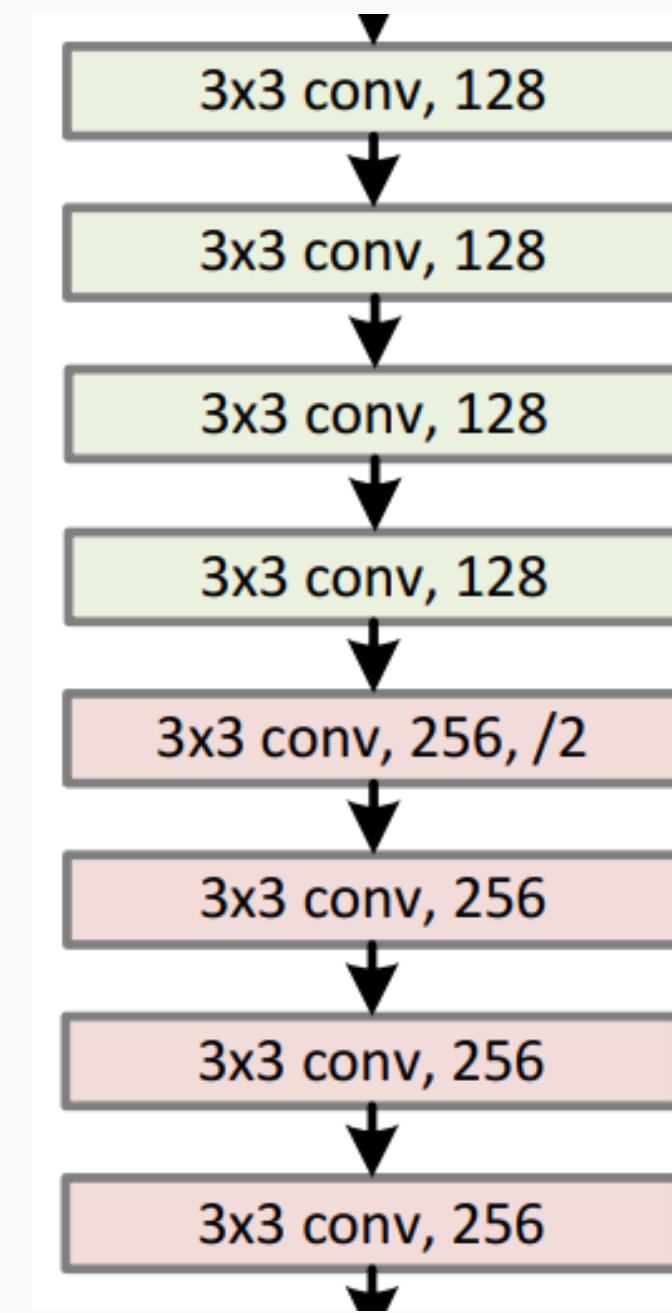
output size: 56



output size: 28

output size: 28

output size: 14



1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture

#### Plain Network

- (1) 비교 목적으로, 기본적인 CNN 네트워크를 가져와서 실험 진행
- (2) 기본적인 네트워크들은 VGG 네트워크에서 제안된 기법들을 적절히 따름
  - ① 3 by 3짜리 작은 필터를 이용
  - ② output feature map size를 동일하게 하기 위해 같은 개수의 필터 사용
  - ③ feature map size가 절반으로 줄어들 때 (즉, 그 다음 레이어에서 채널 값을 2배로 늘림)  
∴ 이런 방법들로 레이어마다 time complexity를 보존할 수 있는 형태로 네트워크 구성
- (3) 별도 pooling layer를 사용하지 않고, convolution layer의 stride값 2로 설정 downsampling 진행
- (4) 레이어 마지막 부분에서 average pooling 이용하여 결국 1,000개의 클래스로 분리할 수 있도록 만듦

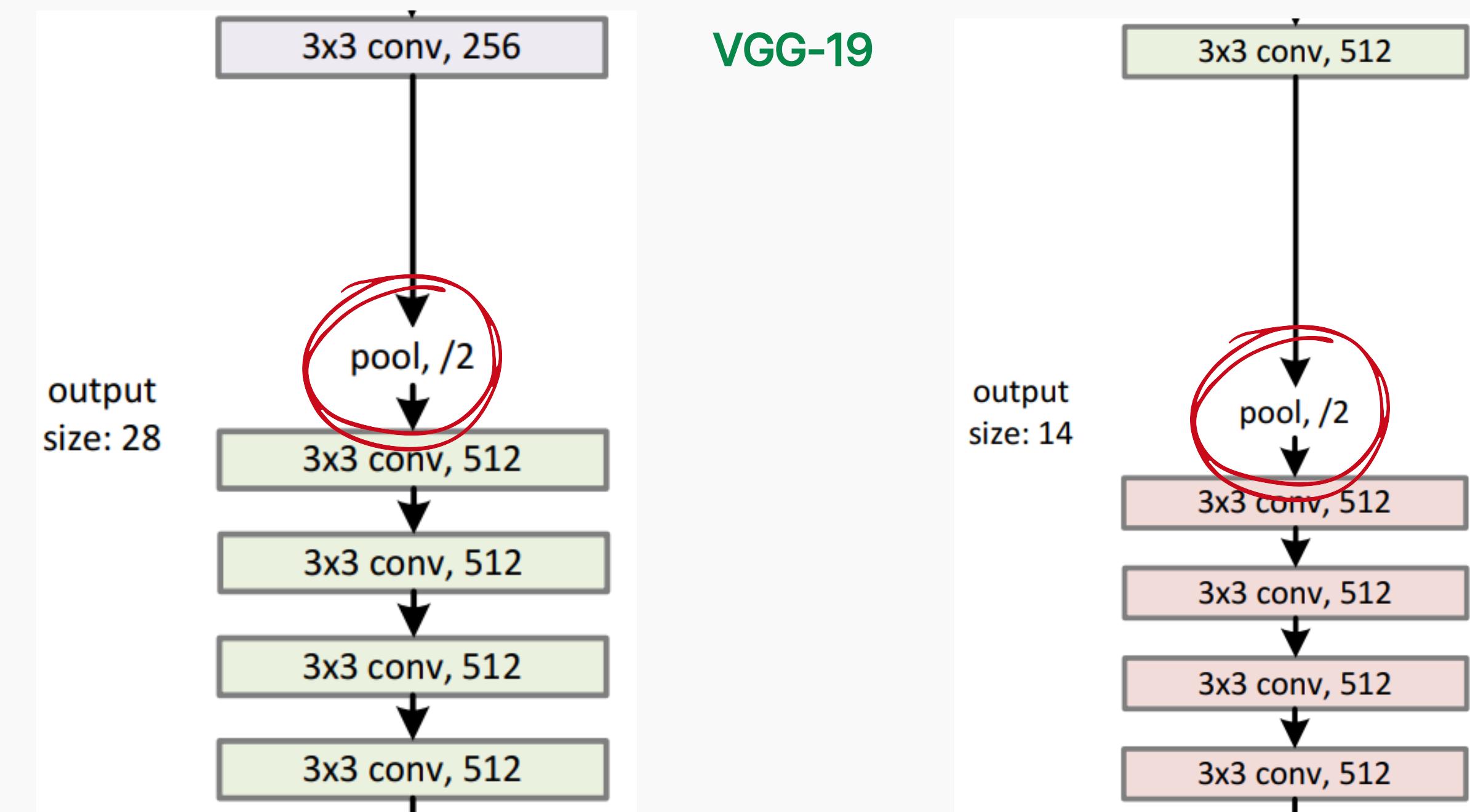
1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture



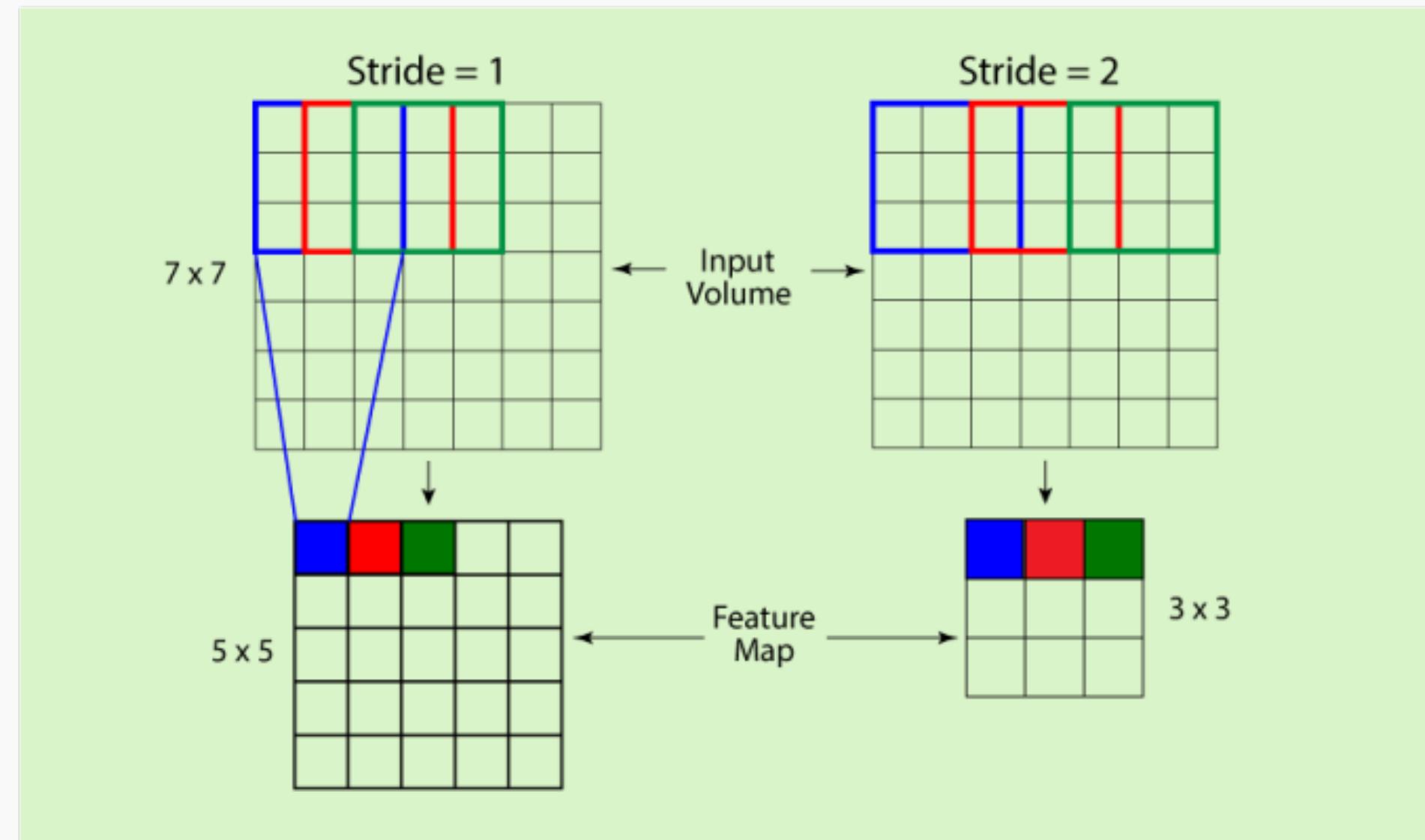
1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture



1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture

#### Plain Network

- (1) 비교 목적으로, 기본적인 CNN 네트워크를 가져와서 실험 진행
- (2) 기본적인 네트워크들은 VGG 네트워크에서 제안된 기법들을 적절히 따름
  - ① 3 by 3짜리 작은 필터를 이용
  - ② output feature map size를 동일하게 하기 위해 같은 개수의 필터 사용
  - ③ feature map size가 절반으로 줄어들 때 (즉, 그 다음 레이어에서 채널 값을 2배로 늘림)  
∴ 이런 방법들로 레이어마다 time complexity를 보존할 수 있는 형태로 네트워크 구성
- (3) 별도 pooling layer를 사용하지 않고, convolution layer의 stride값 2로 설정 downsampling 진행
- (4) 레이어 마지막 부분에서 average pooling 이용하여 결국 1,000개의 클래스로 분리할 수 있도록 만듦

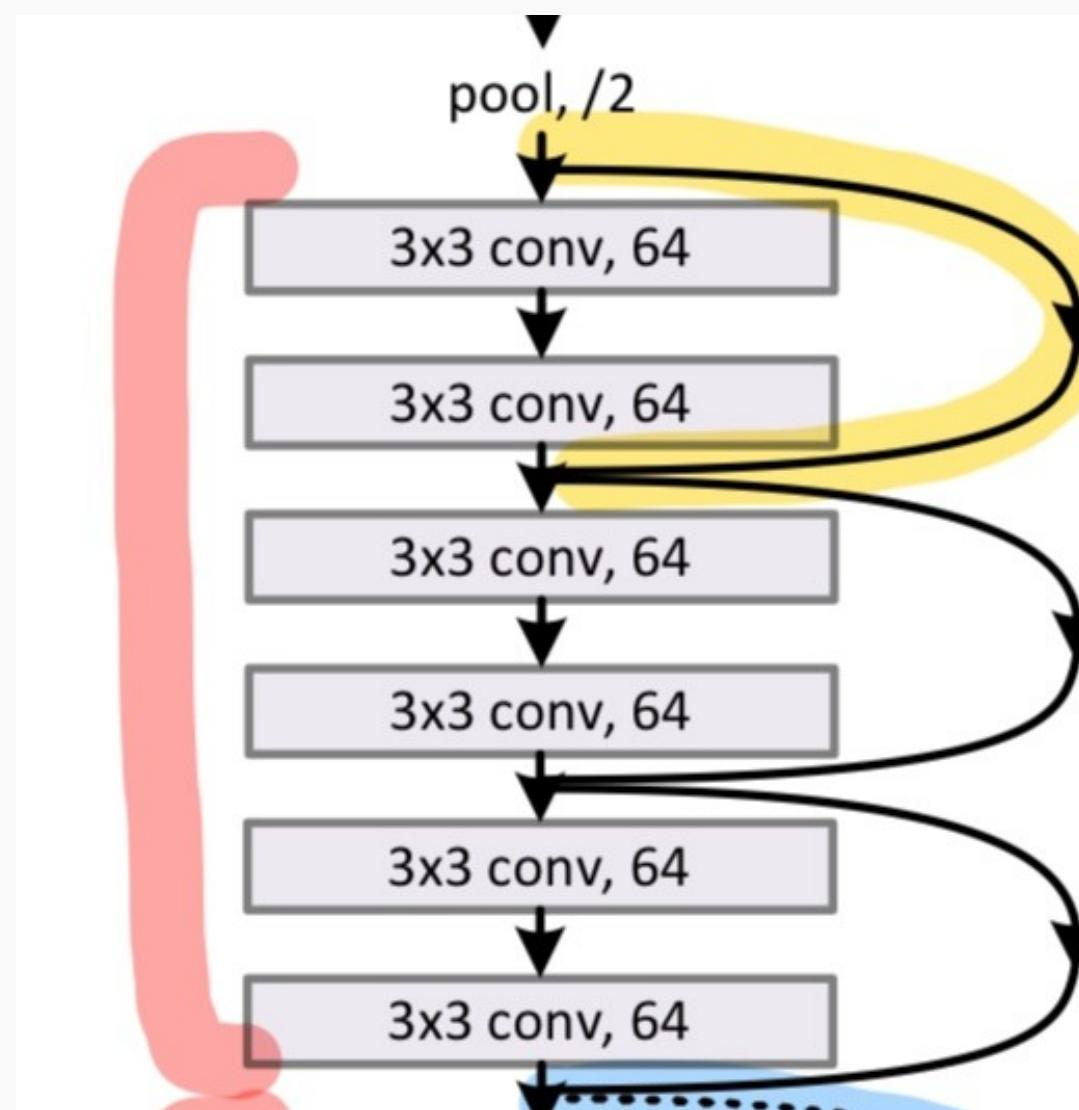
1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture



#### ResNet - 34 layer residual

① filter를 2개씩 묶어(실선)

: convolution filter를 2개씩 묶어 매번 residual function

형태로 학습 진행하도록 만듦

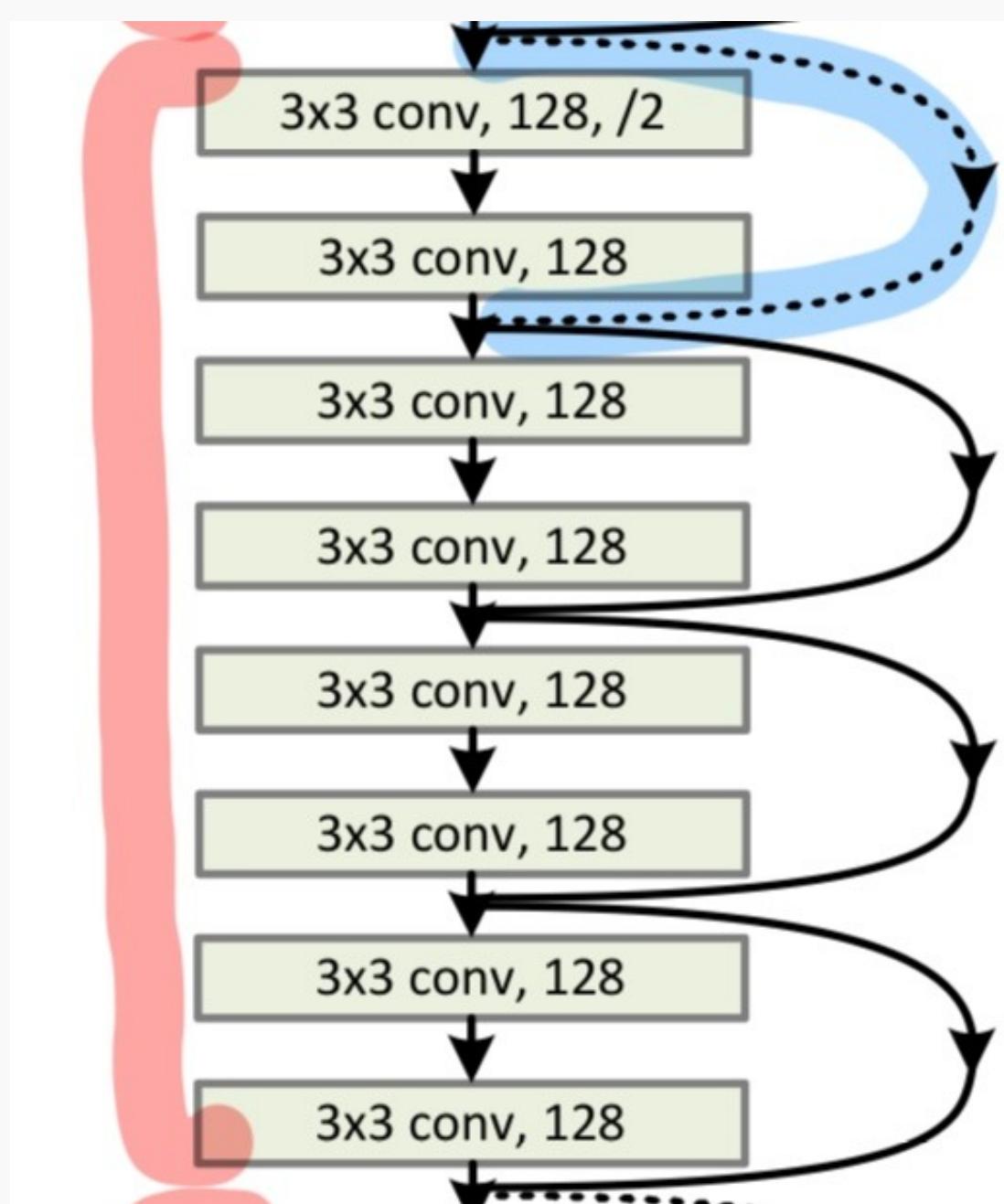
1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture



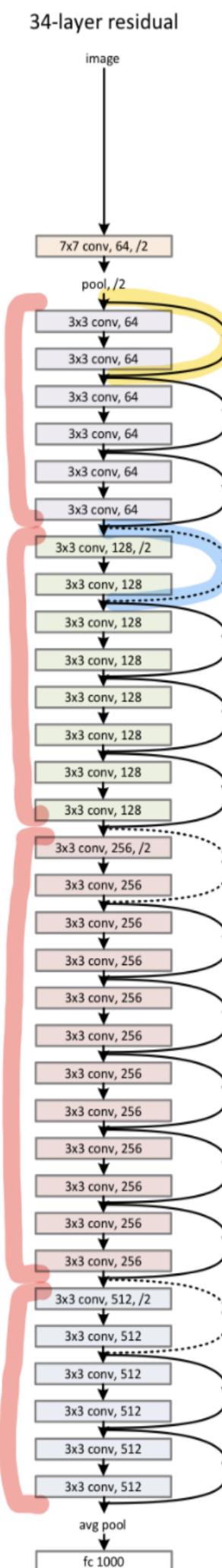
② 점선을 통해,  
shortcut connection이 가미됨을 알 수 있음.  
dimension을 낮춰줄 수 있도록 하는 역할을 하는데,  
입력단과 출력단의 dimension이 불일치하여 사용

## 1 Introduction

## 2 Related Work

## 3 Deep Residual Lear

## 4 Experiment



# Network Architecture

layer name	output size	18-layer	34-layer	layer	152-layer
conv1	112×112				
conv2_x	56×56	$[3 \times 3, 64] \times 2$	$[3 \times 3, 64] \times 3$	$[3 \times 3, 128] \times 4$	$[3 \times 3, 64] \times 3$
conv3_x	28×28	$[3 \times 3, 128] \times 2$	$[3 \times 3, 128] \times 4$	$[3 \times 3, 256] \times 6$	$[1 \times 1, 64] \times 3$
conv4_x	14×14	$[3 \times 3, 256] \times 2$	$[3 \times 3, 256] \times 6$	$[3 \times 3, 512] \times 3$	$[3 \times 3, 64] \times 4$
conv5_x	7×7	$[3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 3$	$[3 \times 3, 512] \times 23$	$[1 \times 1, 128] \times 8$
	1×1				$[1 \times 1, 256] \times 36$
					$[3 \times 3, 256] \times 36$
					$[1 \times 1, 1024]$
					$[1 \times 1, 512] \times 3$
					$[3 \times 3, 512] \times 3$
					$[1 \times 1, 2048]$
average pool, 1000-d fc, softmax					
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$
					$11.3 \times 10^9$

- ③ 컨볼루션 레이어를 2개씩 묶는 작업을 총 3회 반복( $3 \times 3$  conv, 64),  
크기를 바꾼 후에 4회 반복( $3 \times 3$  conv, 128),  
크기를 또 바꾸어 6회 반복( $3 \times 3$  conv, 256),  
크기를 또 바꾸어 3회 반복( $3 \times 3$  conv, 512)

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

### 3.3. Network Architecture

#### Residual Network

(1) input dimension과 output dimension이 동일할 때

: identity mapping 사용 가능

(2) input dimension과 output dimension이 동일하지 않을 때

1) 옵션 A.

: 사이드에 패딩을 붙인 다음, identity mapping 수행

(패딩 : 패딩은 0으로 채워진 엔트리를 입력값에 더함으로써 차원을 동일하게 함)

2) 옵션 B.

: projection 연산을 하려면 shortcut connection을 이용해 구현

---

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 3.4. Implementation

(1) ImageNet의 경우

: 224 x 224로 random하게 추출할 수 있게 하고 horizontal flip과 같은 테크닉이 사용될 수 있음  
( 랜덤하게 추출한다: 이미지의 다양한 부분을 커버한다)

(2) ResNet의 경우

: 매 컨볼루션 레이어를 거칠 때마다 배치 정규화를 이용했다는 점이 특징  
(배치 정규화: 네트워크 내부의 각 레이어에서 입력 데이터를 정규화하는 과정을 포함)

(3) learning rate의 경우

: 학습을 진행하는 과정에서 점진적으로 줄여나가도록 학습을 구성한 것을 확인할 수 있음

(4) weight decay와 momentum의 하이퍼파라미터 값을 설정하여 training을 진행했음을 알 수 있음

1 Introduction

2 Related Work

3 Deep Residual Learning

▣ 4 Experiment

## 4.1. ImageNet Classification

### 데이터셋

(1) 출처 : ImageNet 2012 데이터셋

(2) 구성

① 총 1,000개의 클래스로 분류

② Training image : 1,000,000개 이상

③ Validation image : 50,000개 이상

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.1. ImageNet Classification

### Plain Network

(1) 더 깊은 네트워크를 쌓는 것은 오히려 더 얕은 네트워크보다 error율이 높아진 것을 확인할 수 있음

(2) plain과 ResNet의 다른 점

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	<b>25.03</b>

: shortcut connection이 더해진 것밖에 없음에도 불구하고, 성능이 훨씬 좋다.

(3) 18-layer plain network와 비교하여 34-layer plain network의 error가 높은 문제가 발생하는 원인

1) by vanishing gradient(x), 수렴률이 기하급수적으로 낮아짐(o)

2) vanishing gradient 확인

: forward path, backward path를 확인했을 때 signal 값들이 점진적으로 사라지는 문제가 발생하지 않음

1 Introduction

2 Related Work

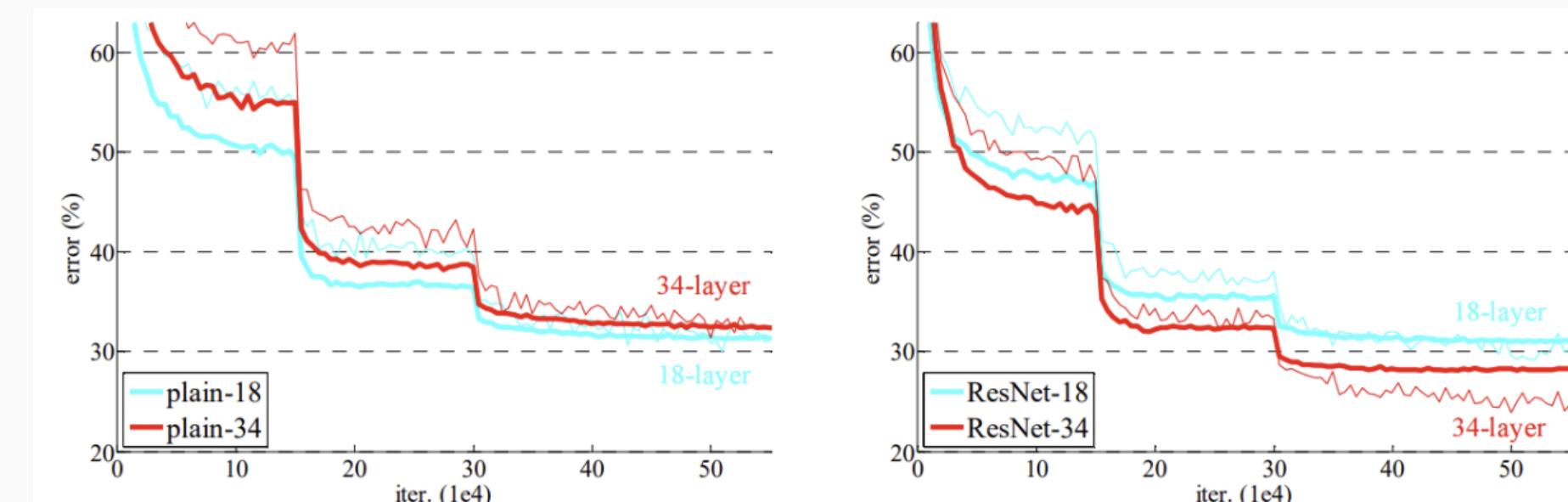
3 Deep Residual Learning

4 Experiment

## 4.1. ImageNet Classification

### Residual Network

(1) Fig 4



- 1) plain network와 비교했을 때 residual network의 경우, 더 깊은 레이어가 얕은 레이어에 비해 잘 동작
- 2) training error 또한 확실히 줄어듦
- 3) 일반화 성능 또한 높았음
- 4) optimization 자체를 더욱더 쉽게 만들어 주는 장점이 있음

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.1. ImageNet Classification

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43 <sup>†</sup>
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	<b>19.38</b>	<b>4.49</b>

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except <sup>†</sup> 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
<b>ResNet (ILSVRC'15)</b>	<b>3.57</b>

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

top-1 err.	top-5 err.
-	8.43 <sup>†</sup>
-	7.89
24.4	7.1
21.59	5.71
21.99	5.81
21.84	5.71
21.53	5.60
20.74	5.25
19.87	4.60
<b>19.38</b>	<b>4.49</b>

top-5 err. (test)
7.32
6.66
6.8
4.94
4.82
<b>3.57</b>

왼: table 4(single model) - 양상블 기법 미적용

오른: table 5 - 양상블 기법 적용

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

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	<b>21.43</b>	<b>5.71</b>

ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40

### shortcut connection을 위한 옵션

- (A) zero-padding으로 dimension 늘리고 identity mapping 사용
  - (B) dimension이 증가할 때만 projection 연산 수행
  - (C) 모든 shortcut에 대해 항상 projection 사용
- 성능 자체는 ResNet-34 C가 가장 높지만,  
projection shortcut이 필수라 할 정도로 높은 개선을 보인 것은 아님  
∴ identity shortcut을 이용해 성능을 많이 개선시킬 수 있음

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.1. ImageNet Classification

### Deeper Bottleneck Architectures

(1) Fig 5

1) 초반에  $1 \times 1$  filter 64개 사용,

중간에 일반 residual block과 마찬가지로  $3 \times 3$ ,

마지막에 다시  $1 \times 1$  filter 256개 사용

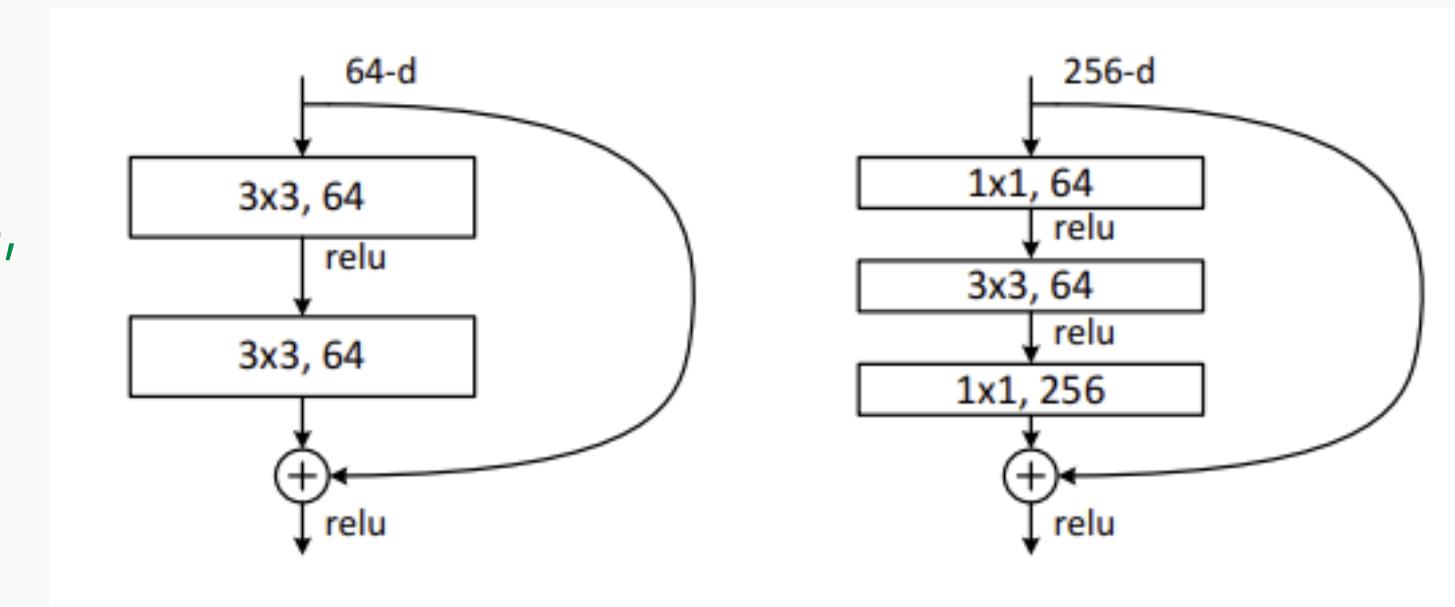
2) bottleneck architecture 구조

:  $1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$

3) identity shortcut은 parameter 자체가 존재 X

-> 이러한 bottleneck architecture에 대해서 더욱 효과적으로 parameter 수를 줄이는 데 기여

다시 말해, 이런 bottleneck architecture에 대해서는 identity shortcut이 더욱더 효과적일 수 있음



1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.1. ImageNet Classification

### Deeper Bottleneck Architectures

#### (2) 50-layer ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			$7 \times 7, 64, \text{stride } 2$		
				$3 \times 3 \text{ max pool, stride } 2$		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	
	1×1			average pool, 1000-d fc, softmax		
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

34-layer	50-layer
$7 \times 7, 64, \text{stride } 2$	$7 \times 7, 64, \text{stride } 2$
$3 \times 3 \text{ max pool, stride } 2$	$3 \times 3 \text{ max pool, stride } 2$
$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
average pool, 1000-d fc, softmax	
$3.6 \times 10^9$	$3.8 \times 10^9$

34-layer net와 50-layer ResNet을 비교했을 때 parameter나 계산 속도는 거의 동일.

#### (3) 101-layer and 152-layer ResNets

레이어를 더 깊게 쌓으면 성능도 좋아지고, 기존의 VGG와 비교했을 때 복잡도 또한 낮아졌다라는 장점

(4) 레이어를 깊게 쌓고, 앙상블까지 더했을 때 매우 좋은 성능을 보임. (by 152-layer, 3.57%)

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.2. CIFAR-10 and Analysis

### 데이터 구성

- (1) training image - 50,000개, testing image - 10,000개
- (2) 입력 이미지 크기: 32x32(ImageNet에 비해 입력 이미지의 크기가 훨씬 작음)
- (3) first layer로 3x3 convolution layer를 사용하도록 만듦
- (4) 보폭을 2로 하여 각각의 feature map size (32, 16, 8)를 가지는 layer들을 2n개씩 스택하여 총 스택이 6n개 반복. 이때 필터 개수는 각각 (16, 32, 64)
- (5) 마지막에 fully-connected layer, softmax가 사용되어서 총  $6n+2$  만큼의 레이어가 사용됨

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.2. CIFAR-10 and Analysis

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	<b>6.43</b> (6.61±0.16)
ResNet	1202	19.4M	7.93

Table 6

(1) 공통 적용

- identity shortcut
- data augmentation technique들은 그대로 사용
- 깊이, 너비, 파라미터 수 유지
- BN, 가중치 초기화 기법 사용
- weight decay: 0.0001
- momentum: 0.9
- 드롭아웃 적용 X
- 학습을 진행함에 따라 learning rate을 줄이는 방법 사용

1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.2. CIFAR-10 and Analysis

method			error (%)
# layers	# params		
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 ( $7.72 \pm 0.16$ )
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	<b>6.43</b> ( $6.61 \pm 0.16$ )
ResNet	1202	19.4M	7.93

Table 6

### (2) 110-layer

: 처음에는 learning rate을 간단히 0.01로 설정했다가,  
다시 0.1부터 차근히 줄여나가는 방법으로 학습 진행.

### (3) 결과

: 레이어를 늘여가며 확인했더니(20, 32, 44, 56)  
ImageNet, MNIST와 마찬가지로 레이어가 깊어질수록  
더 좋은 성능을 보이는 것을 확인할 수 있었음

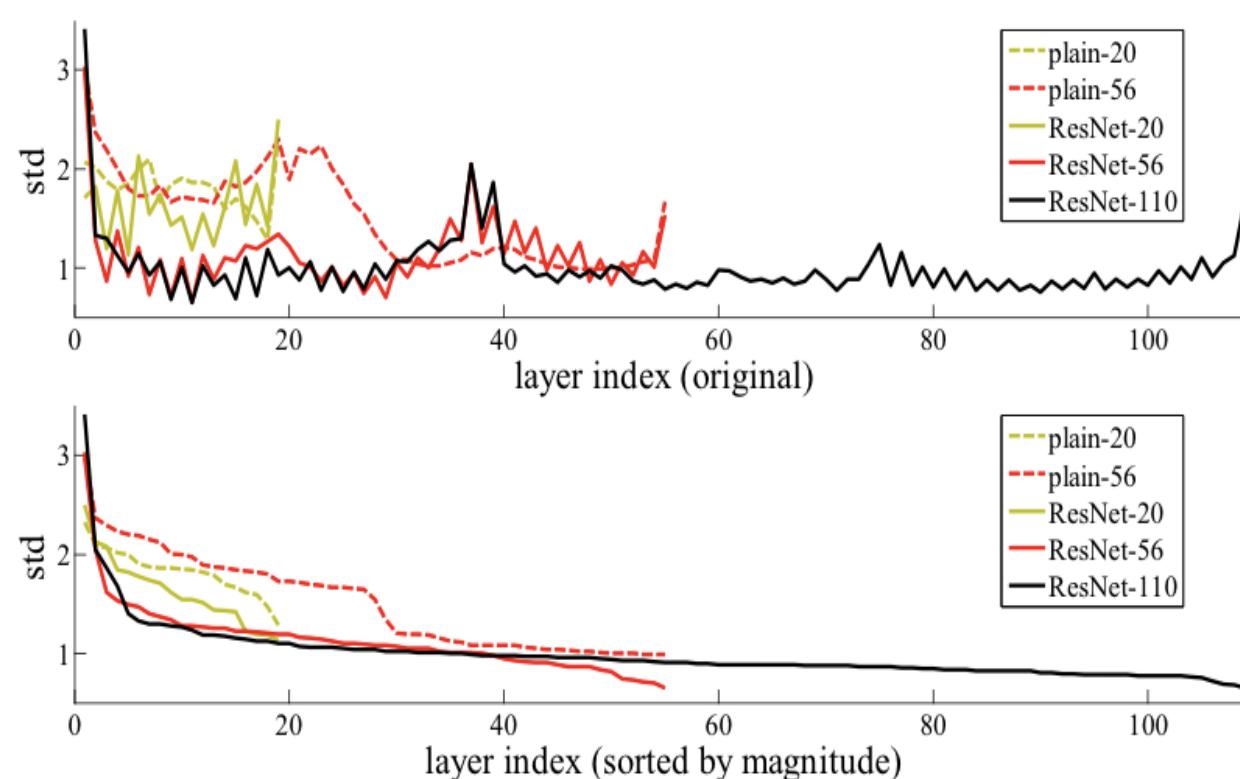
1 Introduction

2 Related Work

3 Deep Residual Learning

4 Experiment

## 4.2. CIFAR-10 and Analysis



**Fig 7**

- (1) 각 레이어에 대한 response를 관찰
- (2) residual function과 non-residual function을 비교했을 때 더욱 0에 가까운 형태로 optimization이 이루어졌다고 판단할 수 있음

### Exploring Over 1000 layers

: 불필요하게 너무 깊은 레이어를 쌓게 되면, overfitting 문제 가 발생할 수 있어 성능이 떨어질 수 있음

1 Introduction

2 Related Work

3 Deep Residual Learning

## 4 Experiment

training data	07+12	07++12
test data	VOC 07 test	VOC 12 test
VGG-16	73.2	70.4
ResNet-101	<b>76.4</b>	<b>73.8</b>

Table 7. Object detection mAP (%) on the PASCAL VOC 2007/2012 test sets using **baseline** Faster R-CNN. See also Table 10 and 11 for better results.

metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	<b>48.4</b>	<b>27.2</b>

Table 8. Object detection mAP (%) on the COCO validation set using **baseline** Faster R-CNN. See also Table 9 for better results.

### 4.3. Object Detection on PASCAL and MS COCO

#### Table 7, 8

- VGG 기반, ResNet 기반으로 각각 Faster R-CNN이라는 객체 탐지를 위한 모델을 사용해 실험
- ResNet이 더 우수



감사합니다