Deep Residual Learning for Image Recognition

1. Abstract

훨씬 깊어진 network 학습을 용이하게 하기 위한 **“Residual Learning Framework”**를 제안한다.

- How?

  unreferenced function을 학습하는 대신 layer의 input을 참조하는 residual function을 학습하도록 layer를 명시적으로 **reformulation**

- Positive?
  - opimize 하기 쉽다.
  - 상당히 깊은 network에서도 합리적인 정확도

2. Introduction

Depth가 깊어질 수록 Vanishing/Exploring 문제점이 발생한다. 하지만 이는 Normalized Initialization과 Intermediated Normalization으로 Deep Neural이 수렴 가능해진다.

> ⚙️ Normalized Initialization은 weight initialization이며, Intermediated Normalization은 batch normalization과 같다.

하지만, 또다른 문제점 **Degradation**이 발생한다.

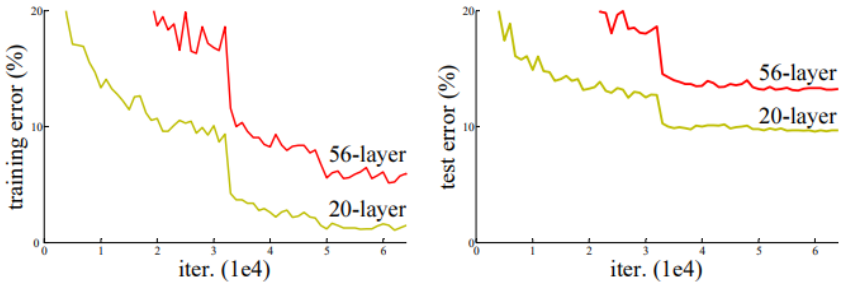


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Degradation은 network의 depth가 증가하면서 정확도가 포화상태에 도달하게 되면 성능이 급격하게 떨어지는 현상이다.

Figure 1 을 확인해보면 depth가 적절히 설계된 모델에 layer를 추가하면서 training error도 증가하게 된 것으로 보아 overffiting이 문제가 아니라는 사실을 알 수 있다.

일반적으로 shallower architecture보다 deeper architecture에서 좋은 성능을 기대한다. 따라서, shallower architecture에다가 identity mapping인 layer만 추가하여 deeper architecture를 만드는 것을 해결책으로 한다.

따라서 이에 대한 해결책으로 **“Residual Learning Framework”**를 도입한다. few stacked layer를 underlying maaping으로 직접하지 않고, redisual mapping에 맞추어 학습한다.’

⚙️ **underlying mapping**
기존 네트워크는 x를 입력받고 layer를 거쳐 $H(x)$를 출력한다. 이 때, 입력값 x를 타겟값 y로 mapping하는 $H(x)$를 얻는 것이 목적이다.

⚙️ **residual mapping**
출력과 입력값의 차인 $H(x)-x$를 얻도록 수정한다. $F(x)=H(x)-x$를 최소화 해야하며, $F(x)=0$이 되는 것이 최적의 해이다.

- Positive?

  unreferenced mapping인 original mapping보다 residual mapping을 optimize하는 문제가 더 쉽다.

- Why?

  H(x)에 대한 optimal solution이 identity mapping이라는 가정을 한다면, $H(x)$의 결과가 x가 되도록 하는 것보다 $F(x)$가 0이 되도록 학습하는 것이 더 쉬울 것이다.

- How?

  $F(x)+x$는 **short connection**으로 구현 가능하다.

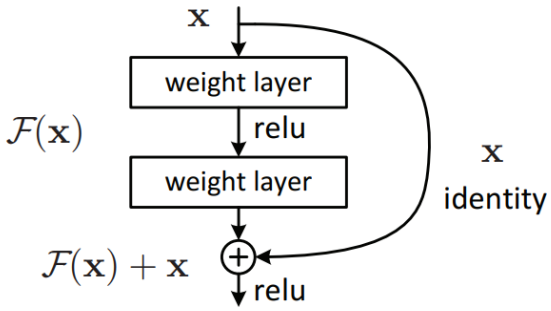


Figure 2. Residual learning: a building block.

> **short connection**
> identity mapping을 수행하고, 그 출력을 stacked 출력에 더한다.

- 별도의 parameter나 computational complexity가 추가되지 않는다.
- 이를 이용한 네트워크는 SGD에 따른 역전파로 end-to-end 학습이 가능하며, solver 없이도 commno library를 사용하여 쉽게 구현 가능하다.

3. Deep Residual Learning

- **Residual Learning**

  stacked layer를 $H(x) - x$에 mapping하기 위해서 original mapping을 $F(x) + x$로 reformulation한 것은 성능 저하 문제를 해결하기 위해서이다.

  - 실제 상황에서 $H(x)$의 optimal이 identity mapping이 아닐지라도, 이 formulation은 문제에 pre-condition을 제공하는 효과를 준다. 만약 optimal function이 zero padding보다 identity mapping에 더 가깝다면, solver가 identity mapping을 참조하여 작은 변화를 학습하는 것이 새로운 funciton을 학습하는 것보다 쉬울 것이라고 마이크로소프트팀은 주장한다.

- **Identity Mapping by Shorcut**

  $F(x) = W_2\sigma(W_1x)$라고 하자. $F + x$는 F와 x의 차원이 같아야한다. 이를 위해서 linear projection을 수행할 수 있다.

  $$F(x) = W_2\sigma(W_1x) + W_sx$$

  여기서 W_s는 차원 matching에서만 사용된다.

- Network Architecture

- Plain Network
  - 동일한 output feature map size에 대해, layer는 동일한 수의 filter를 갖는다.
  - feature map size가 절반인 경우, layer 당 time complexity를 보전하기 위해 filter 수를 2배로 한다.

- Residual Network

  Plain Network + skip connection

  identity shortcut은 input과 output이 동일한 dimension일 경우 직접 사용한다.

  dimension이 증가할 경우에는 아래 옵션을 고려한다.

  - zero entry를 추가로 padding하여 dimension matching 후 identity mapping을 수행한다.
  - projection shortcut을 dimension matching에 사용한다.

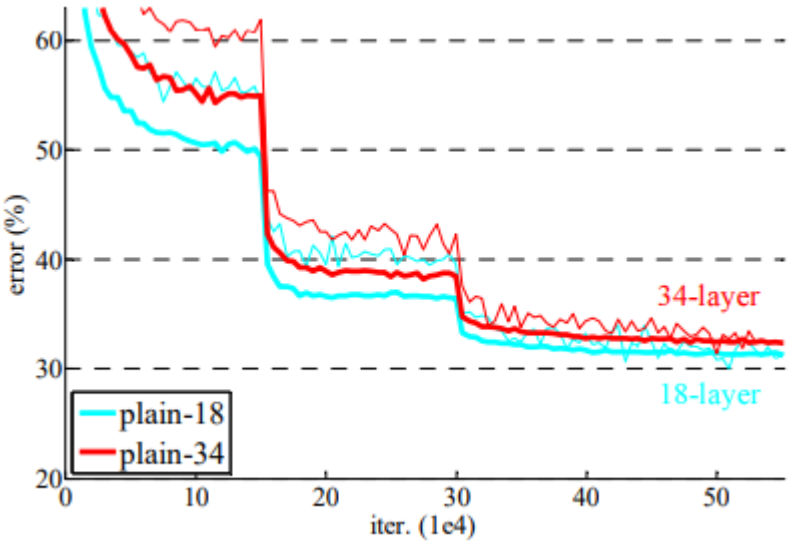
4. 실험

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

ImageNet을 대상으로 한 모델 구조

- Plain Network

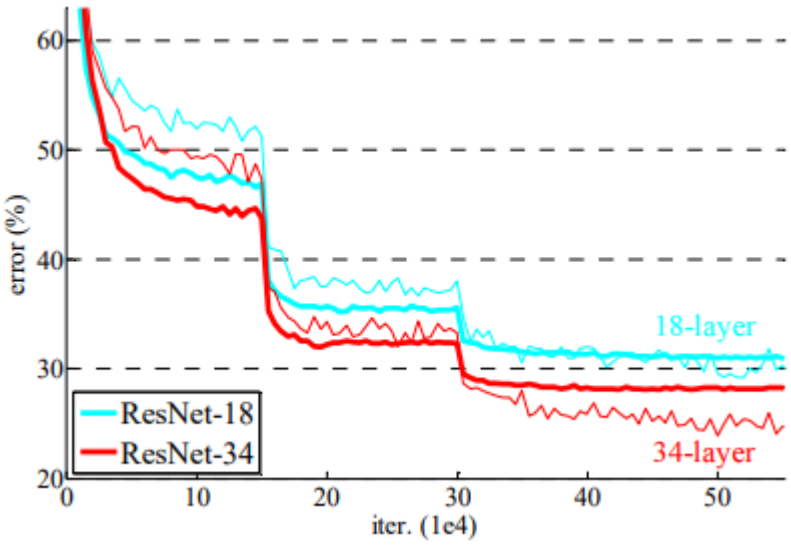
  18 layer의 얕은 plain model에 비해 34 layer의 더 깊은 plain 모델에서 높은 Validation error가 발생하였다. training/validation error 모두를 비교한 결과, 34 layer model에서 training error도 높았기 때문에 **degradation 문제**가 있다고 판단한다.



이러한 최적화 문제는 Vanishing gradient 때문에 발생한 것이 아니라 판단하였는데, plain model은 Batch Normalization이 적용되어 순전파 과정에서 모든 신호의 variance는 0이 아니며, 역전파 과정에서의 기울기 또한 healthy norm을 보였기 때문이다. 따라서 순전파, 역전파 신호 모두 사라지지 않았으며, 실제로 34 layer의 정확도ㅇ는 경쟁력 있게 높은 수준이었다. 이에 마이크로소프트 팀은 deep plain model은 **exponentially low convergence rate**를 가지기 때문에 training error의 감소에 좋지 못한 영향을 끼쳤을 것이라 추측하였다.

- Residual Network

  18 layer와 34 layer의 ResNet과 plain 모델을 비교한다. 이때, 모든 Shortct은 identity mapping을 사용하고, 차원을 키우기 위해 zero padding을 사용하였기에 파라미터 수는 증가하지 않았다.



- residual learning으로 인해 34 layer ResNet이 18 layer ResNet보다 2.8%가량 우수한 성능을 보였다. 특히, 34 layer ResNet에서는 상당히 낮은 training error를 보였고, 이에 따라 validation 성능 또한 높아졌다. 이는 **degradation 문제가 잘 해결되었으며, depth가 증가하더라도 좋은 정확도를 얻을 수 있음을 의미**한다.

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

plain model과 ResNet model의 깊이에 따른 Top-1 error

- 34 layer ResNet의 Top-1 error는 3.5%가량 줄었고, 이는 **residual learning이 extremely deep system에서 매우 효과적**임을 알 수 있다.
- 18 layer plain/residual network 간에는 유사한 성능을 보였지만, 18 layer ResNet이 더욱 빠르게 수렴한다. 이는 network가 “not overly deep”한 경우, 현재의 SGD solver는 여전히 plain net에서도 좋은 solution을 찾을 수 있다는 것으로 볼 수 있다. 하지만, 이런 경우에도 ResNet에서는 빠른 수렴속도를 기대할 수 있다.

- Identity vs. Projection Shortcuts
  - A) zero-padding shortcust을 사용한 경우, 이때, 모든 shrotcut은 parameter-free하다.
  - B) projection shortcut을 사용한 경우, 다른 모든 shortcut은 identity하다.
  - C) 모든 shortcut으로 projection shortcut을 사용한 경우

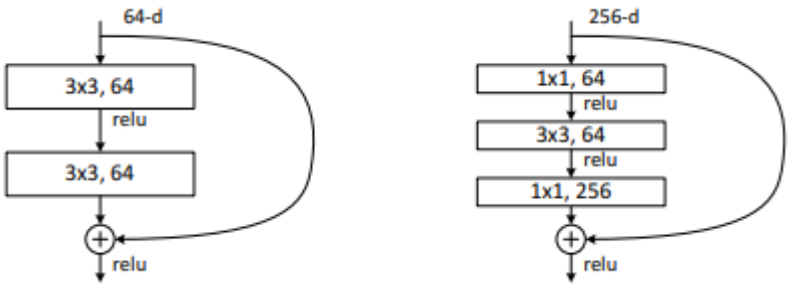
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

세 가지 모두 plain model보다 좋은 성능을 보였고, 그 순위는 A>B>C순이었다. 먼저 A>B는 zero-padding 차원이 residual learning을 수행하지 않았기 때문이고, B>C는 projection shrortcut에 의해 파라미터가 추가되었기 때문이다.

세가지 옵션의 성능차가 미미했기 때문에 projection shorcut이 degradation 문제를 해결하는데 필수적이지 않다는 것을 확인할 수 있다. 따라서 **memory/time complexity와 model size를 줄이기 위해 이 논문에서는 C 옵션을 사용하지 않는다**. 특히 **identiy shortcut은 bottleneck 구조의 복잡성을 높이지 않는 데에 매우 중요**하기 때문이다.

- Deeper Bottleneck Architecture

  ImageNet에 대하여 학습을 진행할 때, training time이 매우 길어질 것 같아 building block을 bottleneck design으로 수정하였다고 한다. 따라서 각각의 residual function인 F는 3-layer stack 구조로 바뀌었는데, 이는 1x1, 3x3, 1x1 Conv로 구성된다.



기존 ResNet building block과 bottleneck design이 적용된 building block

여기서 parameter-free인 identity shortcut은 이 architecture에서 특히 중요하다. 만약 오른쪽 다이어그램에서 identity shortcut이 projection으로 대체된다면, shortcut이 두 개의 high-dimensional 출력과 연결되므로 time complexity와 model size가 두 배로 늘어난다. 따라서 identity shortcut은 이 bottleneck design을 보다 효율적인 모델로 만들어준다.