Deep Residual Learning for Image Classification

0. Abstract

Neural Network가 더 깊어지면서 학습시키기는 더 어려워졌다.

따라서 우리는 깊어진 network의 학습을 좀 더 쉽게 만들 수 있는 residual learning에 대해 소개한다!

우리는 layer를 learning unreferenced function에서 layer의 input을 reference하는 residual function으로 reformulate한다.

우리는 이러한 residual function이 더 최적화하기 쉽고, depth가 증가하더라도 accuracy의 향상을 가져온다는 것을 보일 것이다.

VGG보다 8배나 깊은 152layer의 residual net을 평가해본 결과, VGG보다 더 낮은 complexity를 갖고 있었다. 이러한 residual net의 ensemble은 ImageNet dataset에서 3.57%의 오류율을 달성하였고, ILSVRC 2015 classification에서 1등을 수상할 수 있었다.

(여러가지 성과, 수상 ...) 우리의 모델은 깊게 쌓아 accuracy를 높이는 것이 가능하다. (자랑 끝)

---

1. Introduction

deep convolutional neural network는 image classification에서 놀라운 성과를 보였다!

깊은 네트워크는 low/mid/high 레벨의 feature를 잘 결합해주고,

이 다양한 level의 feature는 층이 깊어지면서 더욱 다양해진다.

최근에는 네트워크의 깊이가 매우 중요하다고 밝혀졌고, 따라서 여러 깊은 모델이 제안되었다.

-> 그래! layer를 깊게 쌓으면 좋다는 거 이제 알겠어 😊

근데, 여기서 떠오른 생각

Q : 레이어를 깊게 쌓는 게 쉬운 만큼 학습도 잘 되는가?

A : 그 유명한 gradient vanishing/exploding이 문제가 된다 ☹

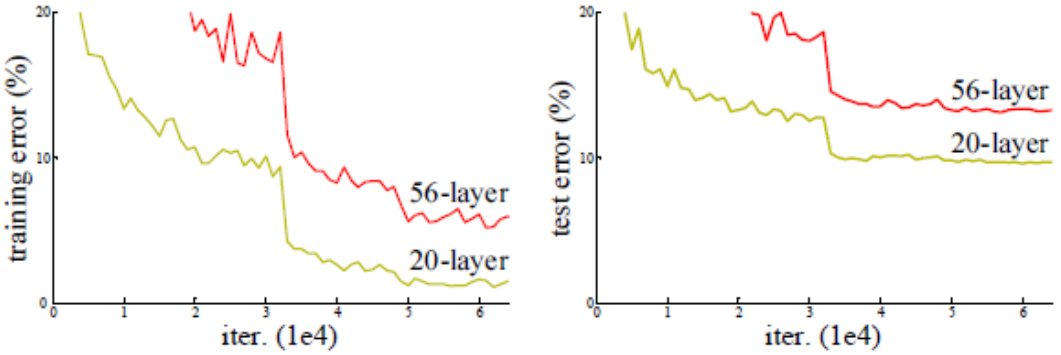
network의 깊이가 증가함에 따라 degradation 문제가 발생한다.

즉, 깊이가 증가할수록 **accuracy는 더 이상 증가하지 않고, 오히려 감소하게 된다.**

이러한 문제는 자칫 overfitting처럼 보일 수 있으나, 실제로는 이미 충분히 깊은 모델에 layer를 더 추

가함으로써 training error가 높아지는 것이다.

overfitting은 training error는 낮으나, test error가 높은 현상을 말하지만 degradation은 layer가 높아졌음에도 불구하고 training error와 test error 모두 낮은 현상을 의미한다.



training accuracy의 감소문제는 model의 parameter를 optimize하는 것이 쉽지 않다는 것을 보여준다.

여기서 shallower architecture와 이것에 layer를 좀 더 추가한 deeper counterpart가 있다고 가정하자.

여기서 추가된 layer는 identity mapping이고, 다른 layer는 learned shallower model로부터 복제되었다. (=identity mapping외에는 shallow와 deeper가 동일하다는 의미로 판단된다.)

이러한 방법은 shallower counterpart 대비 deeper model이 반드시 높은 training error를 갖지 않는다는 것을 보여준다. 하지만, 실험에서 현재의 solver는 constructed solution이나 그 이상의 성능을 보이는 solution을 찾을 수 없다는 것을 보여주었다. ☹

// identity mapping의 한계를 드러냄

// deeper model의 training error의 하한이 shallower model의 error

// shallower model의 error보다 더 낮아지지 못한다는 것 ☹

따라서 논문에서는 deep residual learning을 도입하여 degradation 문제를 해결하고자 한다.★

즉, desired underlying mapping (H(x))을 학습하지 말고, residual mapping(F(x))을 학습하는 것!

원래의 original mapping을 H(x)로 나타낸다면, stacked nonlinear layer의 mapping인 F(x)는 H(x)-x를 학습하는 것이다. 따라서 original mapping(H(x))은 F(x)+x로 재구성될 수 있다.

// F(x) = H(x)-x의 관계가 성립하므로! (F(x)가 residual)

// 예측값 = 잔차 + 입력(여기서는 정답으로 취급하고 있음 by identity mapping)

우리는 original mapping(unreferenced mapping)보다 residual mapping(referenced mapping)이 학습하기가 더 쉬울 것이라 가정한다. 극단적으로, H(x)에 대한 optimal solution이 identity mapping이라고 가정한다면, H(x)의 결과를 x가 되도록 학습하는 것보다 F(x)가 0이 되도록 학습하는 것이 쉬울 것이라 생각한다.

// H(x)에 대한 optimal solution이 identity mapping이라고 가정한다면 -> H(x)의 optimal이 x !

// 즉, H(x)가 x가 되도록 학습하는 것은 H(x)-x=F(x)=0이 되도록 학습하는 것과 동일한 문제

// H(x)의 결과가 x와 유사하게 되도록 학습하는 것보다, 그냥 H(x)가 0이 되도록 학습하는 게 더 쉬움!

// 따라서 H(x)가 x가 되도록 학습하기 위해 F(x)가 0이 되도록 학습! (residual이 0이 되도록 학습)

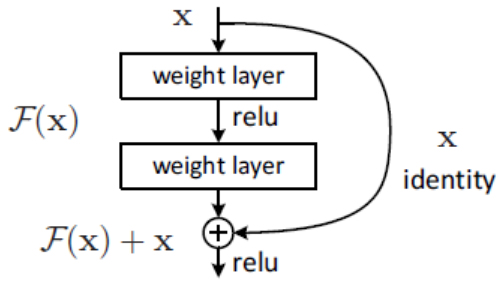


Figure 2. Residual learning: a building block.

F(x)+x는 shortcut connections를 가진 neural network의 feedforward에 의해 실현될 수 있다.

shortcut connections은 identity mapping을 하는 건데, input이 layer의 output에 더해지는 것이다.

identity shortcut connection은 parameter가 더 생기는 것도 아니고, 계산이 복잡한 것도 아니다. (😊)

이 네트워크는 여전히 backpropagation, SGD에 의해 학습될 수 있다.

우리는 degradation 문제를 보여주고, 우리의 방법에 대해 평가하기 위해서 다양한 실험을 했다.

그리고 우리는 다음 2가지를 보였다.

1) 우리의 residual net은 극도로 깊더라도 optimize하기 쉽다. (학습 잘 된다.)

단순히 layer만 쌓아 올린 plain net은 깊이가 깊어짐에 따라 training error가 높아졌다.

하지만 우리의 모델은 아니다 😊 layer가 깊어져도 학습 잘한다 (error가 낮아진다)

2) 우리의 deep residual net은 깊이가 깊어짐에 따라 accuracy도 증가한다. (degradation문제 해결!)

이러한 현상은 CIFAR-10 데이터셋에서도, ImageNet 데이터셋에서도 보였다.

// 즉 특정 데이터셋에만 적용되는 것이 아니라는 것!

우리의 모델은 ImageNet에서 제안된 모델 중에 가장 깊은 모델이었지만, VGG보다 더 낮은 complexity를 보였다.

// 우리의 모델 깊어, 근데 complexity는 낮다고 자랑하는 것😊

그리고 ImageNet test set에서 top-5 error를 기록했고, ILSVRC 2015에서 1등을 거두었다!

또! 훌륭한 generalization performance를 보였고, 다른 대회에서도 수상했다.

즉, residual learning이 generic하고, vision, non-vision 분야에서도 활용도가 높을 것이라는 증거이다.

결론 > 우리가 제안한 residual learning 매우 좋고, 몹시 좋고, 엄청 훌륭하고 ...

---

2. Related Work

1) Residual Representation

vector quantization에서 encoding residual vectors는 encoding original vector보다 더 효과적이라고 나타났다.

2) Shortcut connections

highway network와 달리 우리의 identity shortcut은 절대 닫히지 않으며, 모든 정보는 항상 흐른다.

---

3. Deep Residual Learning

3.1. Residual Learning

x가 stacked layer의 input으로 들어갔을 때, H(x)가 underlying mapping이라고 가정하자.

여기서 approximate complicated functions과 approximate residual function(H(x)-x)는 동일한 문제이다.

// 복잡한 함수를 여러 개의 비선형 layer로 근사시키는 것이 가능하다면

// 잔차 함수 (H(x)-x)도 여러 개의 비선형 layer로 근사시키는 것이 가능하다!

따라서 H(x)를 approximate하는 것보다는, residual function (H(x)-x=F(x))를 approximate하기로 결정한다.

// 왜냐하면 위에서 가정한 것과 같이 residual function을 approximate하는 것이 더 쉬운 문제이므로!

따라서 original function H(x)는 F(x)+x로 재정의 될 수 있다.

complicated function(H(x)), residual function(F(x)) 모두 desired functions을 approximate하는 것은 동일하지만, learning의 쉬움은 다르다.

// 여기서는 residual function(F(x))의 learning이 더 쉽다고 가정한 것 같다!

이것은 reformulation degradation 문제에 대한 **counterintuitive phenomena**에 의해 motivated되었다.

introduction에서 언급한 것과 동일하게, 추가된 layer가 identity mapping이라면 deeper model은 얕은 모델 대비 더 큰 error는 갖지 않을 것이다. (이것이 counterintuitive라는 단어를 선택한 이유!)

counterintuitive phenomena? 무슨 의미? H(x)의 optimal이 identity mapping이라고 가정한다면 얕은 모델에 layer를 추가한 깊은 모델에서 얻을 수 있는 성능의 상한이 얕은 모델에서의 성능이라는 것! (즉 optimal이 identity mapping이라면 깊게 layer를 쌓아도 얕은 모델의 성능을 넘어설 수 없다는 의미) 이것은 depth가 깊어짐에 따라 성능이 좋아진다는 intuitive와 반대이다 (**counterintuitive**)

degradation 문제는 solver가 identity mapping을 approximating하는 것에 어려움을 겪고 있기에 발생한다고 추정한다.

1) identity mapping이 optimal이라면

solver는 residual이 0가 되도록 weights를 업데이트 할 것이다.

2) identity mapping이 optimal이 아니라면

그래도, 우리의 reformulation은 문제에 recondition을 제공하는 효과를 준다.

만약에 optimal function이 zero mapping보다 identity mapping에 더 가깝다면, solver가 identity mapping을 참조하여 작은 변화를(perturbation) 학습하는 것이, 처음의 그냥 function으로 학습하는 것

보다 쉬울 것이다.

// original function을 재정의함으로써 얻을 수 있는 이점에 대해서 설명
// 실제 identity mapping이 optimal이 아닌 경우 F(x)+x라는 식에서 입력 x가 학습 시에 일종의
// 가이드로서 역할을 수행하기 때문에 이러한 reformulation이 긍정적인 효과를 얻는다는 의미!
 	
-> identity mapping이 optimal이든 아니든,
identity mapping이 optimal이라고 가정하고 residual function이 0이 되도록 학습하는 건
어찌되었든 도움은 된다!!!

실제 실험에서 학습된 residual function에서 일반적으로 작은 반응이 있다는 결과를 보여준다.

이 결과는 identity mapping이 합리적인 preconditioning을 제공한다는 것을 보여준다.

* reformulation? H(x)를 F(x)+x로 재정의한 것

요약 > optimal이 identity mapping이든 identity mapping이 아니든,

reformulation(H(x)=F(x)+x)은 학습에 도움을 준다!

3.2 Identity Mapping by Shortcuts

우리는 stacked layer에 residual learning을 적용할 것이다.

수식적으로 나타낸다면, building block은 $y = F(x, \{Wi\}) + x$ 이다.

여기서 x와 y는 각각 layer에서 input와 output이고, 함수 F(x, W)는 학습될 residual mapping을 의미한다.

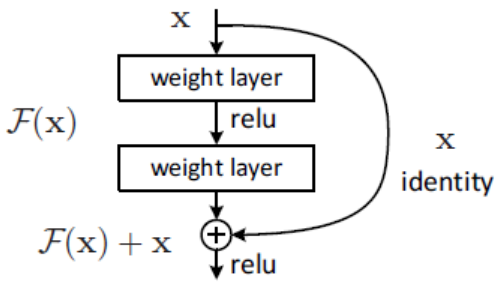


Figure 2. Residual learning: a building block.

2개의 layer를 가진 Figure2의 구조에서 F 함수(residual function)에 대해서 설명하겠다.

Fig2에서 F는 $W_2\sigma(W_1x)$를 의미한다. // 그냥 weight-layer -> relu -> weight layer 의미!

(여기서 시그마는 ReLU를 의미하고, 수식상의 간편함을 위하여 bias는 생략하였다.)

F+x의 연산은 shortcut connection에 의해 수행되었고, x를 더해주는 건 element-wise addition이다.

우리는 addition 다음에 **second nonlinearity**를 적용한다 // 위의 그림에서 보면 두 번째 relu를 의미

shortcut connection은 parameter가 추가되는 것도, 계산상의 복잡성을 가져오는 것도 아니다.

$y = F(x, \{Wi\}) + x$ 의 식 (shortcut connection)에서 x와 F(x, W)는 반드시 동일한 차원을 가져야 한다.

만약에 그렇지 않을 경우, shortcut connection에 linear projection(W_s)을 수행해 차원을 동일하게 해준다.

$y = F(x, \{Wi\}) + W_s x$ (여기서 x에 W_s를 곱해주어 F와의 dimension을 동일하게 해주었다.)

물론 F와 x의 차원이 동일함에도 불구하고 square matrix W_s를 x에 곱해줄 수도 있다.

하지만, 우리는 identity mapping으로 degradation 문제를 충분히 해결할 수 있다는 것을 뒤에서 보일 것이고, 계산상으로도 이것이 훨씬 효율적이다.

따라서 W_s는 오로지 차원을 맞출 경우에만 사용하도록 할 것이다. (즉, F와 x의 차원이 동일하면 사용 X)

residual function F의 형태는 다양하게 설계될 수 있다.

이 논문에서는 F를 2개, 혹은 3개의 layer로 구성할 것이다. (layer를 더 쌓은 것도 가능하다!)

하지만 layer를 1개만 쌓게 되면 shortcut connection의 효과가 없다. $y = W_1 x + x = (W_1 + I)x$

또한 현재의 F는 FC layer로 구성되어 있지만, convolutional layer에서도 적용이 가능하다!

만약에 F에 여러 개의 convolutional layer를 사용하는 경우, element-wise는 두 개의 feature map을 element-wise addition을 수행하는 것이 된다. (channel by channel)

요약 >

1) residual function은 FC를 여러 개 쌓는 것, conv layer를 여러 개 쌓은 것 등 여러가지 형태 가능!

2) 하지만 1-layer의 FC를 residual function(F)으로 사용하게 되면 shortcut connection의 효과 X

(그냥 하나의 FC 통과한 것과 다를 바가 없다.)

3) residual function를 통과하기 전의 dimension과 통과한 후의 dimension은 동일해야 한다.

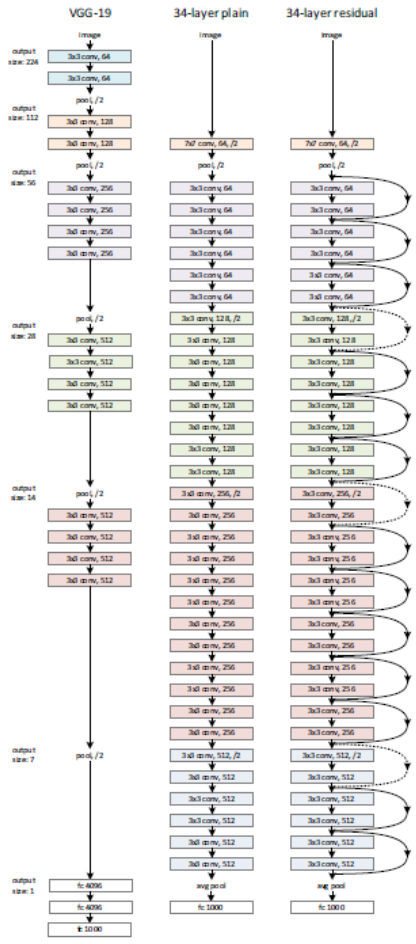
만약, 동일하지 않을 경우 x(input)에 W를 곱해주어 dimension을 맞추어 준다.

(x와 W가 동일한 경우 x에 굳이 square matrix W를 곱해주지 X : 그러지 않아도 성능 좋고, 계산 효율적)

3.3 Network Architectures

우리는 다양한 paint net와 residual net을 실험해보았고, 일관적인 현상을 발견했다.

먼저 실험에서 설계한 model은 다음과 같다.



<Plain Network>

우리의 plain baseline model은 VGG net과 유사하다.

(34-layer plain model)

conv layer는 대부분 3*3 filter를 사용하였고, 두 개의 단순한 rule을 따른다.

1) 동일한 feature map size를 위해서 동일한 개수의 filter를 적용한다.

(input feature map의 channel의 수와 filter의 개수를 동일하게 하면 output feature map의 channel이 이전과 동일!)

2) feature map의 size가 반으로 줄어들면, filter의 개수를 2배로 늘려서 layer 당 time complexity가 유지되도록 한다.

우리는 down sampling을 stride를 2로 주어 conv layer에서 수행되도록 했다.

// pooling 대신 conv에서 stride를 이용해 size down

network의 마지막에는 global average pooling layer(GAP), 1000-way FC with softmax를 통과시킨다.

<Residual Network>

위의 plain network 를 기반으로, 우리는 **shortcut connection 을 집어넣었다.**

(위의 오른쪽 model 을 보면, turn the network into its counterpart residual version)

1) input, output 이 동일한 dimension 갖는 경우

identity shortcut 은 바로 사용될 수 있다.

2) input, output 이 다른 dimension 갖는 경우

- option A :

shortcut 은 identity mapping 을 수행하고,

increasing dimension 을 위해 **zero entries padding 추가**

- option B :

projection shortcut 사용하기 (위에서 봤던 $y = F(x, \{Wi\}) + W_s x$ 이거!)

shortcut connection 이 다른 크기의 feature map 사이에 수행될 경우, stride 를 모두 2 로 설정!

3.4 Implementation

ImageNet dataset 에서 실행하기 위해 다음과 같은 절차 따른다.

1. image 는 shorter size 로 resize 된다.

(shorter size 는 [256, 480]에서 랜덤으로 추출되며 scale augmentation 을 위해 수행되는 것!)

2. 224*224 crop 은 horizontal flip with per-pixel mean subtracted 이미지로부터 랜덤하게 추출

3. standard color augmentation 사용

// RGB 값을 바꾸는 건데, random noise 를 더해주기도 한다

// 좀 더 잘하는 방법은 이미지에 PCA 를 적용하여 찾은 주요 성분을 random 하게 더해주는 것)

4. conv 후, activation 전에 BN(batch normalization)이 적용되었다.

5. weight 초기화

6. learning rate 는 0.1 부터 시작하며, error plateaus(정체기)에서는 learning rate 가 1/10 으로 줄어든다.

// 좀 더 세밀하게 탐색하기 위함

7. 60*10^4 iteration 으로 학습됨

8. weight decay 0.001, momentum 0.9 사용

9. dropout 사용하지 X

10. testing 에서 비교를 위해 standard 10-crop testing 사용

---

4. Experiments

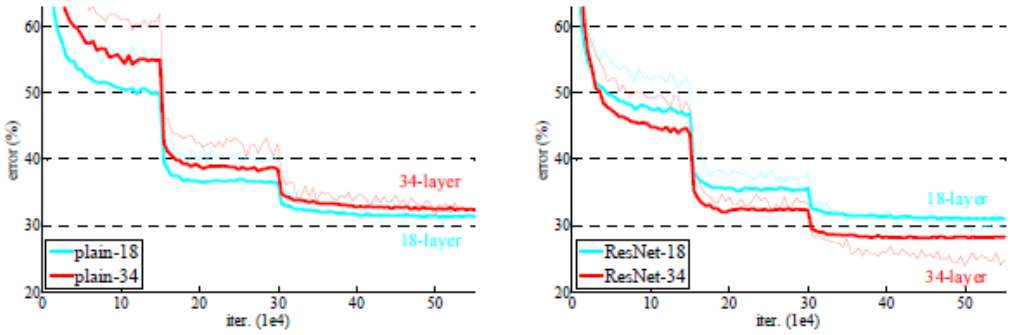
4.1 ImageNet Classification

먼저 1000개의 class로 구성된 ImageNet dataset으로 우리의 모델을 평가해보겠다.

plain network와 residual network에서의 결과를 한 번 구경해보자.

먼저 plain network의 결과부터!

<Plain Network>



우리는 먼저 18-layer와 34-layer의 plain nets을 평가해보았다.

table표를 보면 34-layer의 plain net은 18-layer의 plain net보다 높은 validation error를 갖고 있음을 확인할 수 있다.

// layer가 더 많은데, validation error가 높아서 overfitting 의심

// 하지만 training error 역시 layer가 더 많은 경우에 높다는 것을 확인! -> overfitting 문제가 아니다

우리는 training/validation error를 비교하여 degradation 문제가 발생하고 있음을 알아냈다.

34-layer의 plain net은 전체 training동안 높은 에러를 보이는 반면, 18-layer는 이보다 낮다.

// degradation : layer가 더 깊음에도 불구하고 training error, validation error모두 높은 현상

우리는 이러한 optimization difficulty가 **vanishing gradient문제로부터 오는 것이 아니라고 생각**한다.

이유1. 이 네트워크는 BN와 함께 train되었는데, 이것은 forward propagated signals이 non-zero variance를 갖는다.

이유2. backward propagated gradients가 healthy norm이라는 것을 확인했다.

=> 따라서 forward와 backward signal 모두 **vanish되지 않았다고 판단하였다**.

Table.3의 결과를 따르면 34-layer plain network가 여전히 경쟁력 있는 정확도를 달성했으며, 이는 solver의 작동이 이루어지긴 한다는 것을 의미한다. 저자들은 또한, deep plain network는 exponentially low convergence rate를 가지며, 이것이 training error의 감소에 영향 끼쳤을 것이라 추측하고 있다.

저자들은 훨씬 많은 iteration 을 수행해봤지만 여전히 성능 저하 문제가 관찰됐다고 한다. 이는 이 문제가 단순한 반복 학습으로 해결가능한 것이 아님을 시사한다.

<Residual Network>

그 다음에는 18-layer 와 34-layer 의 residual net(ResNet)을 평가해보겠다.

baseline architecture 는 위의 plain net 과 동일하며, shortcut connection 이 3*3 filter 에 추가되었다.

우리는 모든 shortcut 에서 identity mapping 을 사용하였고, increasing dimension 에 대해 zero-padding 을 적용하였다. 따라서 plain counter parts 대비 parameter 가 증가하지는 않았다.

우리는 3 개의 주된 현상을 발견하였다.

1) 18-layer 의 ResNet 보다 34-layer 의 ResNet 의 결과가 훨씬 좋았다.

이 말은 곧 degradation 문제가 해결되었고,
depth 을 증가함에 따라 높은 accuracy 를 얻을 수 있게 되었다는 것!

2) plain net 에 비해서 error 가 낮았다.

3) 18-layer plain net 과 18-layer ResNet 의 accuracy 는 유사하나, ResNet 이 더 빠르게 수렴하였다.

Identity vs Projection Shortcut

앞에서 parameter-free 한 identity shortcut 이 training 에 도움이 된다는 것을 밝혔다.

// residual network 에서 identity mapping 사용하였고, plain net 보다 결과 좋았다. 바로 위!

다음으로 우리는 projection shortcut 에 대해서 알아보고자 한다.

우리는 3 개의 옵션을 비교해볼 것이다.

(A) 증가한 차원에 대해서 **zero-padding shortcut**

(B) 증가한 차원에 대해서 **projection shortcut, 동일한 차원인 경우 identity**

(C) **모든 차원에 대해 projection shortcut** (즉, 동일한 차원에서도 projection shortcut)

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	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**

1. 우선 plain net 보다는 A, B, C 옵션을 적용한 ResNet 이 error 가 낮다.

2. A 보다는 B 의 error 가 낮다 -> **zero-padded dimension 은 no residual learning 이라서!**

3. B 보다는 C 의 error 가 낮다 -> **projection shortcut 에서 추가된 parameter 가 영향을 준 것!**

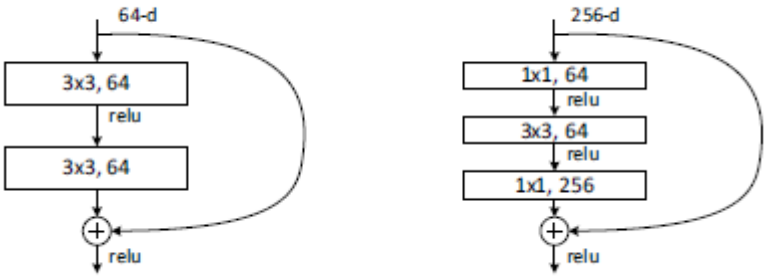
결론적으로 C 가 error 가 가장 낮았지만 이 논문에서는 C 를 사용하지 않는다.

이유 1. error 의 차이를 보았을 때 projection shortcut 이 degradation 문제를 다루는 데 그다지 필수적이지 않아 보인다.

이유 2. memory/time complexity 를 낮추기 위함이다.

Deeper Bottleneck architecture

training time 을 줄이기 위하여 building block 을 bottleneck design 으로 수정한다.



각각의 residual function F 에 3-layer 를 사용한다.

이 3-layer 는 1*1, 3*3, 1*1 convolutions 로 구성된다.

1*1 layer 는 dimension 을 줄이고, 다시 증가시키는데 사용하고, 3*3 는 더 작아진 input/output dimension 을 위해 bottleneck 으로 남겨둔다.

Q. 왜 이러한 bottleneck 구조를 취하는가? 우선 1*1 을 통과하여 feature map 의 depth 를 줄인 다음에 3*3 filter 를 통과시킨다. 그리고 다시 1*1 을 통과시켜 이전의 depth(256)로 복구해준다. 이렇게 bottleneck 구조를 취하는 이유는 3*3 에서 parameter 의 수와 학습 시간을 줄이기 위함이다

bottleneck architecture 에서 parameter-free **identity shortcut** 은 매우 중요하다.

identity shortcut 이 **projection 으로 대체된다면**, time complexity 와 model size 가 2 배로 증가할 것이다.

따라서 identity shortcut 은 bottleneck 구조를 더 효율적으로 만들어준다.

50-layer ResNet

34-layer 에서 2-layer block 을 3-layer bottleneck block 으로 교체한 것이다.

우리는 증가한 차원에 대해 option B(증가한 차원에 대해서 projection shortcut, 동일한 차원인 경우 identity)를 취하였다.

101-layer & 152-layer ResNet

1. 깊이를 엄청 늘렸음에도 불구하고 VGG-16/19 보다도 complexity 가 낮다!

2. 50/101/152-layer 의 ResNet 은 34-layer 의 ResNet 보다 훨씬 더 정확도가 높다.

=> degradation problem 이 나타나지 않았으며, depth 를 증가시킴에 따라 accuracy 가 증가하고 있다.

method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	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**

4.2 CIFAR-10 and Analysis

1. CIFAR-10 은 10 개의 클래스가 존재하고, training image 5 만장, test image 1 만장으로 이루어져 있다.

2. network 의 input 은 32*32 이미지이고, per-pixel mean subtracted 되었다.

* per-pixel mean 은 training data image 들의 평균

* per-pixel mean-subtracted 는 즉, 그 위치의 pixel 의 평균을 빼는 것

3. 첫 번째 layer 는 3*3 convolution 이다.

4. 그리고 3*3 convolution 의 6n 개의 layer stack 을 사용하고, 각 size 마다 2n 개의 layer 로 구성된다.

5. subsampling 은 convolution 에서 stride=2 를 통해 수행되었다.

6. network 의 마지막에는 global average pooling(GAP), FC layer, softmax 가 있다.

7. 총 6n+2 개의 stacked layers 가 있다.

8. shortcut connection 은 모두 identity shortcut 을 사용하였다. (option A)

9. weight decay 0.0001, momentum 0.9, weight initialization, BN 사용, dropout 사용 X

10. 0.1 의 learning rate 로 시작하여, 32k, 48k iteration 에서 1/10 으로 감소

n={3, 5, 7, 9}에 대해 20, 32, 44, 56-layer network 를 실험해보았다.

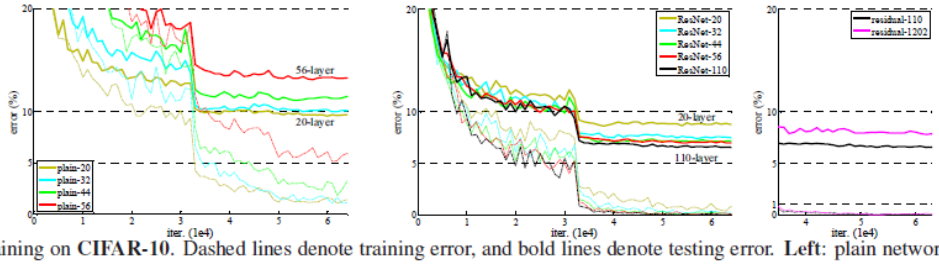


Figure 6. Training on **CIFAR-10**. Dashed lines denote training error, and bold lines denote testing error. **Left**: plain networks. The error of plain-110 is higher than 60% and not displayed. **Middle**: ResNets. **Right**: ResNets with 110 and 1202 layers.

왼쪽의 경우는 plain net 이다.

plain net 의 경우 depth 가 깊어지면서 training error 도 함께 증가하고 있음을 확인할 수 있다. ☹

이러한 현상은 ImageNet 과 MNIST 에서도 동일하게 나타났으며, optimization 의 어려움이 이 현상의 주된 원인이다.

하지만 가운데의 ResNet 의 경우,

degradation 문제를 극복한 것으로 보이며, depth 가 깊어질수록 error 가 낮아지고 있다. 😊

Analysis of Layer Response

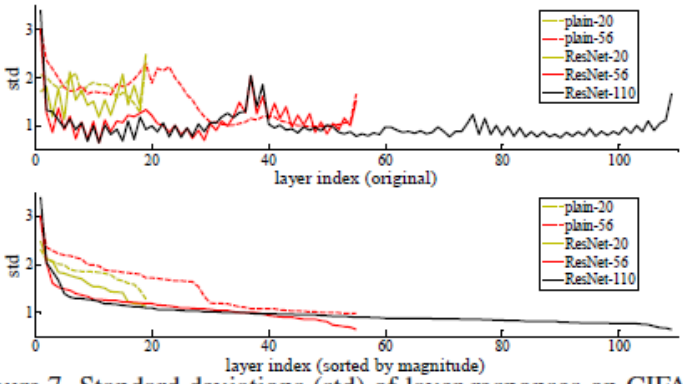


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

ResNet 의 경우, 이 분석에서 residual function 의 response 강도가 드러난다. Fig.7 에서는 ResNet 이 이에 대응하는 plain network 보다 일반적으로 작은 response 를 보여준다. 이는 residual function 이 non-residual function 보다 일반적으로 0 에 가까울 것이라는 저자들의 basic motivation 을 받쳐주는 결과이다. 또한, Fig.7 을 통해 ResNet 의 depth 가 깊을 수록 더 작은 response 를 보이는 것을 알 수 있다.

// "response 가 더 작다"의 의미?

optimal 이 0 보다 identity 에 가깝다면 F(x, W)+x 에서 x 를 기준으로 optimal 을 찾게 될 것이므로, residual function 인 F(x, W)는 x 로부터 optimal 이 되기 위한 작은 변화량을 학습하면 된다.

depth 가 깊어짐에 따라 response 가 작아지는 현상은 각 layer 가 학습 시에 signal 을 변화하는 정도가 작아지는 경향이 있음을 나타낸다.

exploring over 1000 layers

우리는 n=200 으로 설정한 1202-layer network 를 만들어 실험해보았다.

여기에서도 optimization difficulty 문제는 발견되지 않았고, 0.1% 미만의 training error 를 달성하였다. test error 역시 상당히 좋다.

하지만 1202-layer network 가 110-layer network 와 training error 는 유사했음에도 불구하고, 1202-layer 의 testing result 가 더 나빴다. 우리는 이것의 원인이 overfitting 이라고 생각한다.

1202-layer 의 네트워크는 이 작은 데이터셋에 대해서는 불필요하게 크기 때문이다.

따라서 maxout 이나 dropout 과 같은 regularization 이 적용된다면 결과는 향상될 것이라 예상한다.

4.3 Object Detection on PASCAL and MS COCO

우리의 방법은 다른 recognition task 에서도 좋은 성능을 보인다. 😊

이해하지 못한 부분에 대해서는 아래의 주소를 참고하였습니다.

https://sike6054.github.io/blog/paper/first-post/