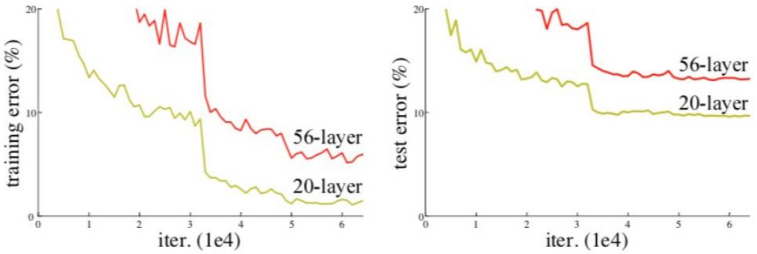
Deep Residual Learning for Image Recognition

1. [문제 제기] Degradation Problem

기존의 image classification 이 크게 발전할 수 있었던 계기가 DCNN(Deep Convolution Neural Network) 였던 만큼, 성능에 있어서 모델의 깊이는 중요하다. 대회에서 우수한 성적을 거둔 모델들이 다 매우 깊은 네트워크를 가지고 있었으며, 이는 classification 뿐만 아니라 다른 분야에서도 적용되는 이야기이다.

그렇다면 좋은 성능을 내려면 layer 를 더 깊게 쌓으면 되는 것일까?



위의 그래프를 보면 아니란 것을 알 수 있다. 왼쪽은 training error, 오른쪽은 test error 를 나타내는데, 두 그래프 다 더 깊게 쌓은 모델(56-layer)에서 높은 error 가 나타났다. 왜 그런 것일까?

깊게 쌓인 네트워크의 학습을 방해하는 두 가지 고질적인 요인이 있다. 첫 번째 문제는 vanishing/exploding gradients 문제로 학습 중간에 gradient 가 사라지거나 폭발하는 현상이다. 이를 해결하기 위한 방법으로 normalized initialization(가중치 초기화), intermediate normalization layers 가 있다.

두 번째 문제는 Degradation problem 으로 모델이 학습할 때 깊은 네트워크로 가면서 accuracy 가 포화되었다가 바로 빠르게 감소(Degradation)하는 현상이다. 이는 더 많은 레이어를 더하는 것이 더 높은 training error 를 만들어 내기 때문에 발생한다. 이를 해결할 방법은 없을까?

2. [해결 방법] : Residual Mapping

이 Degradation Problem 을 해결하기 위해 Residual Mapping 이 있다. 핵심 아이디어는 깊은 모델에 identity mapping 을 한 layers 를 더하는 것이다. (이때 다른 layer 들은 얕은 모델에서 학습한 것을 복제하여 둔다.) 이렇게 만들어진 모델은 더 깊은 네트워크에서도 높아지지 않는 training error 를 보여주었다.

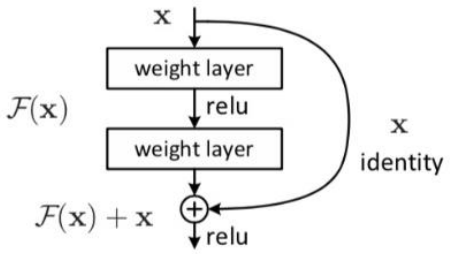


Figure 2. Residual learning: a building block.

① Deep Residual Learning

X 가 들어갔다가 다시 x 로 나오되, 레이어를 쌓는 이점은 계속 누린다면 training error 을 줄일 수 있지 않을까? 라는 생각에서 출발하였다. 기존 출력이 H(x)라고 할 때 (이때 x 는 input 이다) F(x)는 입력값과 출력값의 차이로 F(x) = H(x)-x 로 둘 수 있다. 이에 따라 출력값 H(x)는 F(x)+x 로 정의할 수 있게 되는데, 이 상태에서 F(x)를 0 으로 보내면 결국 x→x 형태로 입력과 출력이 같아진다. (identity mapping) 즉, 기존의 Loss 를 0 으로 보내는 방식 말고, 이제는 F(x)를 학습하여 0 으로 보내 보자는 아이디어가 Residual Learning 인 것이다.

이를 이용하면 다음과 같은 장점을 얻을 수 있다.

① 최적화하기 더 쉽다. (Degradation problem을 해결할 수 있다.)

② 이를 이용하면 상당히 깊은 layer에서도 좋은 Accuracy를 얻을 수 있다.

③ 기존의 VGG nets보다 8배 깊게 쌓았음에도 불구하고 복잡하지 않다.

[정리] Residual mapping

X가 들어갔다가 레이어를 거쳐 X로 다시 나오는 건 어떨까? → 즉 H(x) = x!

➔ 그러면 stacked nonlinear layer의 F(x)는 H(x)-x를 학습하는 것이고

➔ 출력결과는 H(x)대신 F(x)+x가 되겠고! 그러면 x가 들어가서 x가 나오려면 F(x)를 0으로 보내면 되겠네?

➔ 아하 ~ 같은 말이어도 H(x)가 x가 되게 만드는 것보다 F(x)를 0으로 보내는 게 더 쉽구나~!!

② Identity Mapping by Shortcuts

Feedforward Network에서 하나 이상의 레이어를 스킵하는 것을 shortcut connections라고 한다. 그리고 앞에서 본 identity mapping이 바로 shortcut connections를 한 것과 같은 형태라고 볼 수 있다. 이 identity shortcut connection은 추가적인 파라미터를 더하지 않으며 계산적 복잡성도 낮다는 장점이 있다.

[feedforward network] 순환하지 않고 순방향으로만 진행되는 네트워크

수식적으로 설명을 해보자면, 다음과 같이 적을 수 있다.

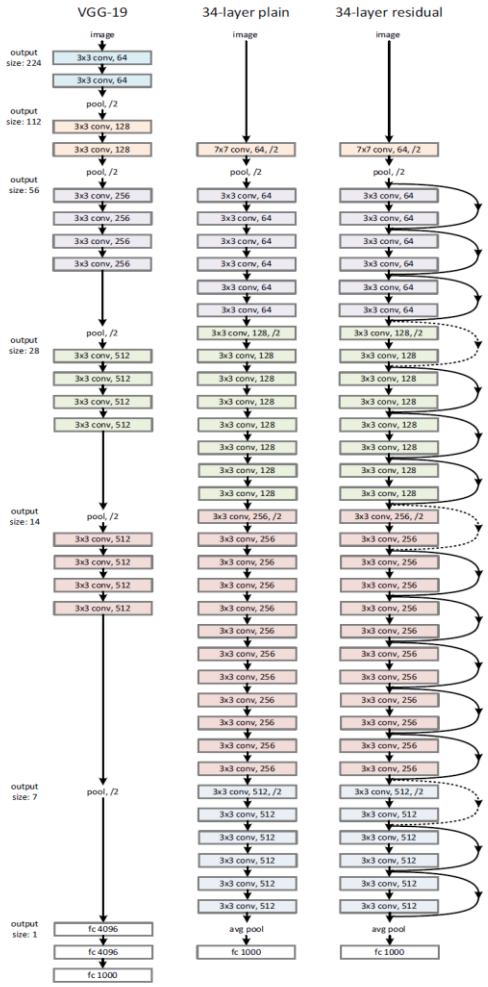
y = F(x, {W}) + x : 이때의 x와 y는 각각 input, output이며 function F(x, {W})는 residual mapping을 나타낸 것이다. Activation function은 Relu를 사용했으며 편의상 bias는 뺐다. 이때 F+x는 shortcut connection과 element-wise addition에 의해 수행된다.

[element-wise additon] 백터/행렬의 덧셈, 뺄셈을 의미함.

이때 x와 F를 더하기 때문에 차원은 같아야 하는데, 채널이 바뀌면서 안 맞게된 경우에는 Ws와의 투영을 통해 맞춰줄 수 있다. (투영된 케이스 : y = F(x, {W}) + Wsx)

우리는 정방행렬 Ws를 사용하는 것을 고려해볼 수도 있었는데, 우리 실험에서는 identity mapping으로도 충분했기 때문에 Ws의 경우 차원을 맞출 때만 이용하도록 한다. 또 예시에서의 레이어는 2, 3개이지만 더 많이도 충분히 가능하다. 하지만 layer가 1개인 경우 linear와 유사해지면서 이점이 없어지니 주의할 것! 또 fully-connected layer 뿐만 아니라 convolution layer에서도 적용이 가능하다.

③ Network Architectures



- Plain network

: VGG nets 을 바탕으로 제작, convolutional layers 는 대부분 3 X 3 필터를 가지며 동일한 output feature map size 에 대해 layer 들은 동일한 수의 filter 를 갖는다. 또 feature map size 가 절반인 경우에는 각 레이어의 시간 복잡성을 보전하기 위해 filter 의 수를 2 배로 한다. 이렇게 만든 우리 모델은 VGG net 보다 필터도 적고 복잡하지도 않고 최종 파라미터도 18% 밖에 되지 않는다.

- Residual network

: plain 기반의 모델에 shortcut connections를 투입하여 residual version을 만든 것이 residual network! Input과 output은 같은 차원이어야 하지만 만약 dimension이 증가하는 경우 두 가지 옵션을 고려해볼 수 있다. 첫 번째, zero entry를 추가로 padding한 후 identity mapping한다. 이 경우 파라미터가 추가되지 않는다. 두 번째, projection shortcut을 dimensions를 맞추는 데 사용한다. (projection shortcut은 뒤에 나온다.)

- Implementation

: input 에 사용될 이미지는 데이터 증식을 거친다. 각각의 convolution 과 activation 전에 Batch normalization 을 사용했고 weight initialize 를 했다. mini-batch size 를 256 으로 지정한 SGD 를 사용했으며 learning-rate 는 0.1 로 두고 시작했다. 또 weight decay 는 0.0001 로, momentum 은 0.9 로 두었으며 dropout 은 사용하지 않았다.

3. ImageNet Classification 으로 알아보는 ResNet

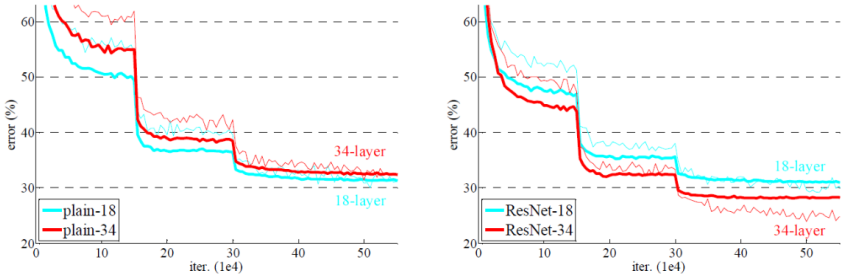
① plain vs residual network

ImageNet 2012 에서 새 방식을 적용해보았다.

Plain network 로는 18-layer 와 34-layer 을 평가했는데 여기서도 더 깊은 34-layer 에서 더 높은 error 가 발생했다. 다시 말하지만 이건 Vanishing gradients 의 문제는 아니다. Batch Normalization 해줬는데도 그랬으니까. 그럼 뭐 때문일까? 깊은 plain network 에서의 exponentially low convergence rate 가 training error 에 영향을 끼쳤기 때문이라고 추측한다.

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

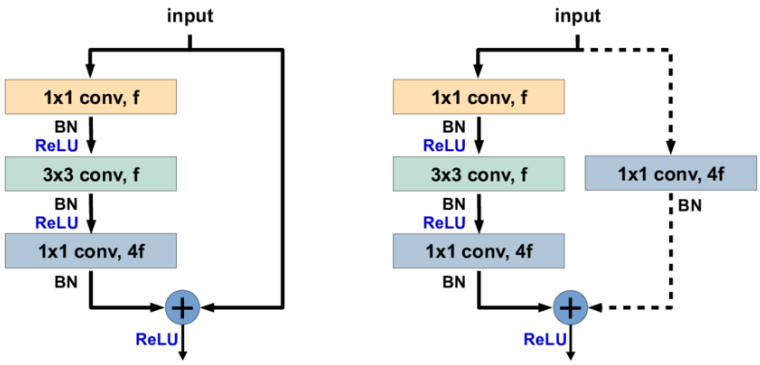
다음은 Residual Net 에서 실행한 결과를 보자. 이는 plain 에서 오직 shortcut connection 만 추가된 형태이다. 이 또한 마찬가지로 18-layer 와 34-layer 로 구성하였는데, 반전된 결과를 얻을 수 있었다. 바로 드디어 더 깊은 34-layer 에서 더 낮은 training error 을 기록한 것이다. 이는 degradation problem 을 이 방식으로 해결할 수 있다는 것을 다시 한 번 보여준다.



Plain 의 34-layer 와 residual 의 32-layer 를 비교해보았을 때도 residual 의 결과가 월등히 좋았는데, 18layer 비교에서는 두 모델 다 유사한 수치를 보였다. 그러나 위의 그래프에서 알 수 있듯이, ResNet-18 의 수렴속도가 더 빠르다.

② Identity shortcuts vs projection shortcuts

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**



(오른쪽 그래프의 왼쪽은 ResNet-50 에서의 identity shortcut, 오른쪽은 projection shortcut 을 나타냄.)

지금까지 identity shorcuts 를 알아보았다면 이제는 projection shortcuts 를 보자. 우리는 세 가지 옵션을 구성하였다.

(a) increasing dimension 을 위해 zero-padding shortcut 를 사용, 나머지 shortcuts 는 parameter-free 함.

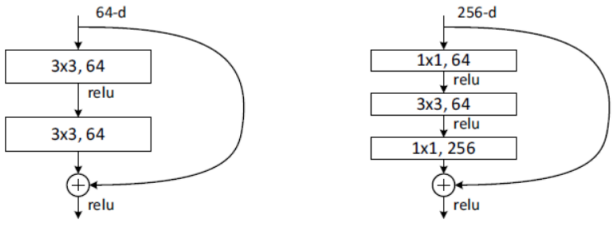
(b) increasing dimension 을 위해 projection shortcut 를 사용, 나머지 shortcuts 는 identity

(c) 모든 shortcuts 를 다 projection 으로 둠.

이때, a 가 지금까지 소개해왔던 ResNet 에 사용했던 기법이다.

세 가지 옵션으로 실험을 수행한 결과가 왼쪽 표인데, 중간에 있는 ResNet-34 A, B, C 가 각 옵션을 적용한 모델이다. 결과를 보면 알 수 있듯이, 뭘 쓰든 우수하긴 했다. A 가 B 보다 낮고, B 보다 C 가 낮긴 했지만 그 차이가 미묘하여 projection shortcuts 가 degradation problem 에 필수적인 것은 아니라고 판단했다. 그래서 메모리/시간 복잡성을 감소시키기 위해 이 논문의 남은 부분에서는 c 옵션은 사용하지 않는다.

③ Deeper Bottleneck Architectures



우리가 감당할 수 없는 training 시간을 위해 기존의 building block 대신 bottleneck design 을 제안한다. 왼쪽의 2 layer stack 대신 오른쪽의 3layer stack 을 사용하며, 1X1, 3X3 다시 1X1 Convolution 레이어로 구성되어 있다.

1X1 Convolution layer 는 dimension 을 줄이거나 늘이기 위해 사용하며 3X3 레이어에서 input/output 의 dimension 을 줄인 bottleneck 형태로 둔다. 위의 그림에서 두 모델은 유사한 time complexity 를 갖는다.

Bottleneck building block 은 ResNet-50/101/152 같은 경우에 사용되었다.

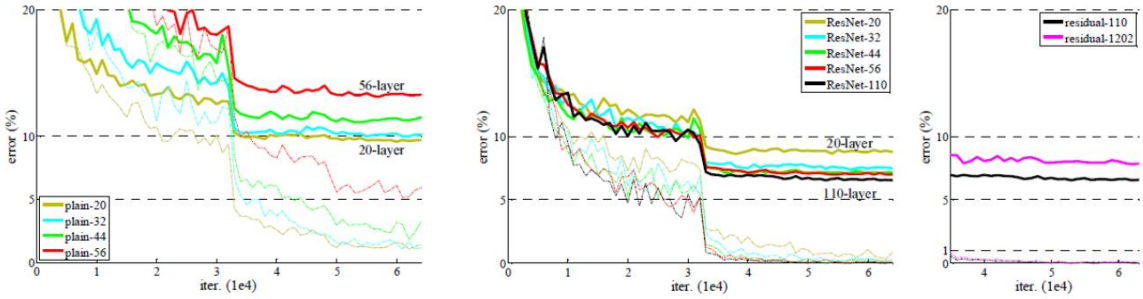
파라미터에서 자유로운 identity shortcuts 는 특히 bottleneck 구조에서 중요하다. 만약 identity shortcut 이 projection 으로 대체된다면 shortcut 이 두 개의 high-dimensional 한 출력에 연결되기 때문에 시간 복잡도와 모델 사이즈가 두 배가 되는 것을 볼 수 있을 것이다. 따라서 identity shortcuts 는 bottleneck 구조의 모델을 더 효율적으로 만들어준다.

④ 101-layer and 152-layer ResNets

우리는 3-layer blocks 를 이용해 101, 152 layer 를 구성했다. 역시나 깊이가 상당히 늘어났음에도 불구하고 152-layer 는 VGG-16/19 와 비교했을 때 여전히 복잡하지 않았다. 이들은 이전의 34-layer 보다 더 정확했으며 degradation problem 문제가 발생하지도 않았고 enjoy 한 accuracy 도 얻을 수 있었다. 다른 평가 metrics 에서도 이와 같은 결과를 얻을 수 있었다.

5. CIFAR-10 set 에서 알아보는 ResNet

더 깊은 네트워크에서의 behavior 를 알아보기 위해 CIFAR-10 set 을 가지고 실험을 진행해보았다. 이것은 가장 최근의 것을 뛰어넘기 위한 것은 아니기 때문에 모델은 단순하게 구성하였다. 사용된 Residual model 은 기존의 plain 모델과 identity mapping 말고 다른 점이 없다.



① Plain vs Residual

CIFAR-10 set 에서도 예상과 같이 plain model 의 층이 깊어질수록 training error 가 높아지는 것을 볼 수 있었다. 왼쪽의 그래프가 plain model 의 결과인데, 가장 깊게 쌓은 plain-56 의 error 가 가장 높은 것을 볼 수 있다. 하지만 우리의 ResNet 은 이 최적화 문제를 극복해냈다. 오른쪽을 보면 가장 깊게 쌓은 ResNet-110 이 가장 낮은 error 를 가진다. ResNet 을 이용하면 깊은 층에서도 안정적인 accuracy 를 얻을 수 있다는 것을 입증해낸 것이다.

** 이건 그냥 궁금한 점 인데요. 이 부분 뒤에 n=18 인 110layer ResNet 으로 확장시켜 보았는데 처음에는 initial learning rate 를 0.1 로 주었는데 수렴을 시작할 때 조금 큰 것 같아서 이후에는 트레이닝 에러가 80% 이하로 떨어질 때 까지는 0.01 로 주었다가, 다시 0.1 로 바꾸는 방식을 선택했다고 하는데, 반대 아닌가요? 처음에는 쭉쭉 떨어져야 하니까 0.1 로 주고 나중에 0.01 로 하는 방식이 더 수렴에 유리하지 않나요? **

② Over 1000 layers

Layer 를 1202 까지 깊게 쌓아도 최적화 문제는 발생하지 않았다. Train, test error 도 여전히 괜찮았다. 다만 110 layer 보다 test error 가 높다는 게 문제였다. 위 그래프의 맨 오른쪽 그림에서 굵은 선은 test error 를 나타낸다. Layer-1202 에서 더 높은 수치를 보이고 있는 것을 볼 수 있다.

이 문제는 Overfitting 때문 같았다. 작은 데이터셋에 비해 너무 큰 레이어를 쌓았기 때문일까? 이를 해결하기 위해 maxout 이나 dropout 같은 정규화 방법을 고려해볼 수 있으나 하지 않았다. 우리의 목적은 residual function 이 optimization difficulty 를 해결해줄 수 있는가에 대한 것이기 때문이다. 그래도 아마 정규화해보면 해결될 수 있을 것 같은데 후속 논문에서 다루겠다.

6. ResNet 으로 이뤄낸 것들

우리 건 classification 뿐만 아니라 다른 recognition tasks 에서도 좋은 성능을 거뒀다. 우리의 deep residual nets 를 이용해 우리는 ILSVRC & COCO 2015 에서의 ImageNet detection, ImageNet localization, COCO detection, COCO segmentation 에서 1 등을 했다. 짱이지? 또 우리 거 generic 해서 vision 문제 뿐만 아니라 non-vision 문제도 풀 수 있다!!

<과제 2 번 스크린샷>