ResNet 정복기

Deep residual learning for image recognition

목표: 나중에 다시 꺼내봐도 resnet을 생생히 떠올릴 수 있도록 정리하기

문제가 무엇인가

- 네트워크 depth는 딥러닝에서 아~주 중요한 화두
- deep한 모델들이 ImageNet에서 성능을 잘 낸다

- 그렇다면,, 논문이 던지는 중요한 질문!

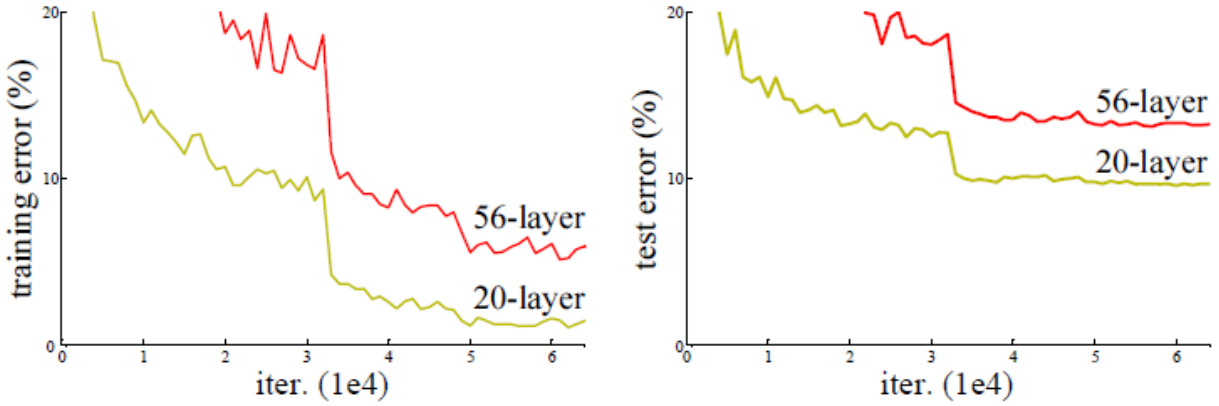
- “Is learning better networks as easy as stacking more layers?”
- 층을 쌓을수록, 모델의 깊이가 깊어질수록, 더 좋은 모델이 되는가?

대답은 “아니다!”

- 알다시피 뉴럴넷의 back propagation 시 발생하는 vanishing/exploding gradient 문제는
- 가중치 초기화(weight initialization)와 batch normalization으로 어느 정도 해결이 되었다

- 그런데! 수렴 문제를 해결하니 “degradation” 문제가 발생하더라!

degradation이 뭔데?



- CIFAR-10 데이터셋으로 학습한 두 종류의 plain network(20층, 56층)
- 이 그래프에서 이상한 점을 찾아보자
  - 깊어질수록 test error만 높아지면 그건 오버피팅(과적합) 문제라고 볼 수 있찌
  - 그런데 이 경우는 다르다! train error도 56층이 20층보다 높다!
- 논문 왈, "깊어질수록 accuracy는 포화상태에 이르고 이후 급격히 성능 저하하는 문제가 발생한다"
- 모델들이 전부 최적화하기 쉬운게 아니다!

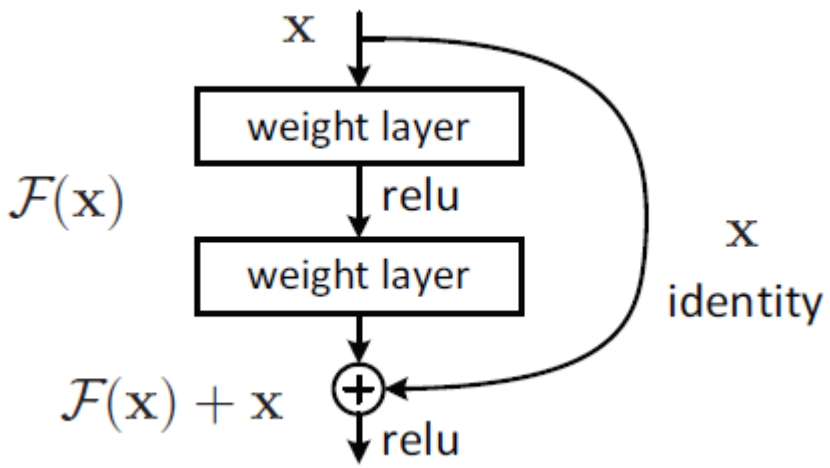
Degradation을 어떻게 해결할 것인가

- 얕은 모델에서 identity mapping(x -> x)인 층들을 추가해서 깊은 모델을 만들었다 치자(construction solution)
- 이렇게 되면 깊은 모델의 성능은 적어도 얕은 모델만큼 나오거나 그 이상이어야 한다
- 그런데! 실험을 해보니 결과가 그렇지 않다더라
- 즉, 현재의 알고리즘들은 깊은 모델(deeper model)로부터 얕은 모델(shallower model)만큼의 성능도 못 끌어내고 있다는거다!

- 논문 왈, “우리가 Residual Learning Framework로 성능저하문제를 해결했다!”

Residual Learning Framework가 뭔데?

- residual은 잔차다. 그렇다면 residual learning은 잔차를 학습하는거다.
- 여기서 정의하는 잔차가 뭘까?
- 아래 그림을 이해해보자



- original mapping을 H(x)라 할 때
- 입력 x와의 잔차 F(x) = H(x) - x
- residual mapping은 F(x)를 학습하는 것!
- 이렇게 되면 original mapping은 F(x)+x로 재구성된다(reformulation)

- 이 논문의 가설 : unreferenced mapping인 original mapping보다, referenced mapping인 residual mapping을 optimize하는 문제가 더 쉽다.

뭔 말??

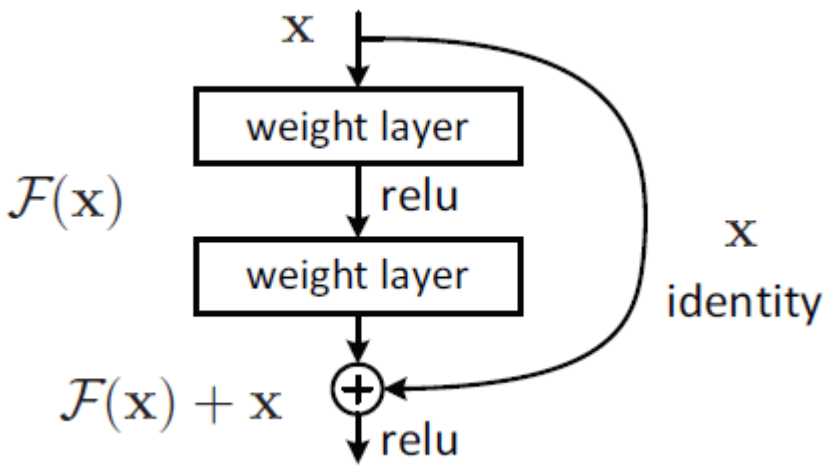
- 이 논문의 가설 : unreferenced mapping인 original mapping보다, referenced mapping인 residual mapping을 optimize하는 문제가 더 쉽다

- 극단적으로 H(x)에 대한 optimal solution이 identity mapping이라는 가정을 한다면, H(x)의 결과를 x가 되도록 학습하는 것보다 F(x)가 0이 되도록 학습하는 것이 더 쉬울 것이란 말이다!
- H(x)=F(x)+x이니 H(x)가 x가 된다는건 F(x)가 0이 된다는 말이겠지
- 이 가설에서 말하는 optimal solution은 이러한 identity mapping인 것이다

아~

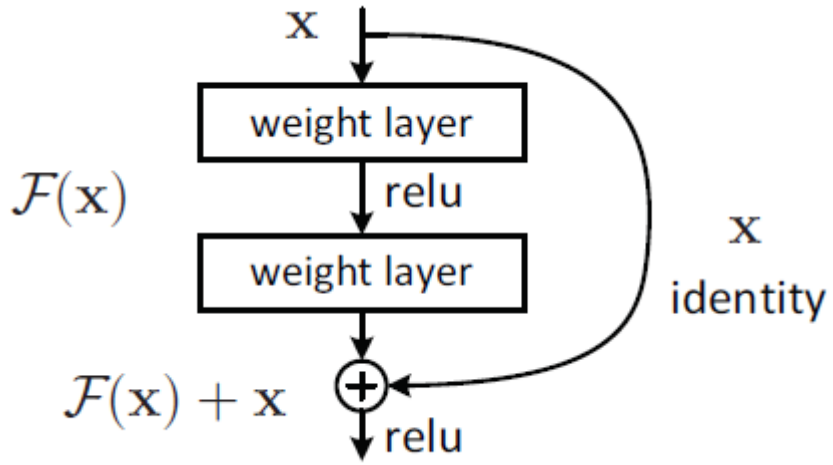
- 그럼 이 F(x)+x를 어떻게 구현할까?
- “shortcut connection”으로 구현할 수 있다!

shortcut connection이 뭔데?



- 이 그림을 말하는거다!
- 그림처럼 입력 x가 2개의 stacked layer를 거쳐서 identity인 입력 x를 더한 후에 nonlinearity layer(ReLU)를 통과하는거다.
- 별도의 파라미터나 computational complexity가 추가되지 않는다고 한다
- 좋다는 말이다
  - 논문 예시를 참고하자면 152층 resnet이 19층 VCG보다 낮은 computational complexity를 가진다고 하더라

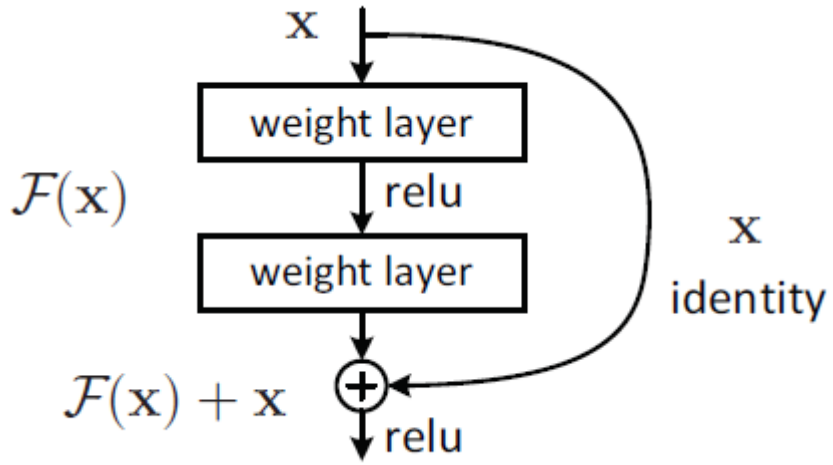
이 과정을 다시 좀 더 자세히 살펴보자
Identity Mapping by Shortcuts



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

- x, y는 각각 input과 output
- $F(x, \{W_i\})$는 residual mapping (앞에서 계속 언급한 F(x)다)
- 위 그림처럼 layer가 2개라면 $F = W_2\sigma(W_1x)$로 표현할 수 있다.(σ는 렐루!)
- weight layer를 거치고 렐루를 적용한 후 다시 weight layer를 거치고
- input이었던 x를 더하고(shortcut connection) 두번째 렐루를 거친다!

이 과정을 다시 좀 더 자세히 살펴보자
Identity Mapping by Shortcuts

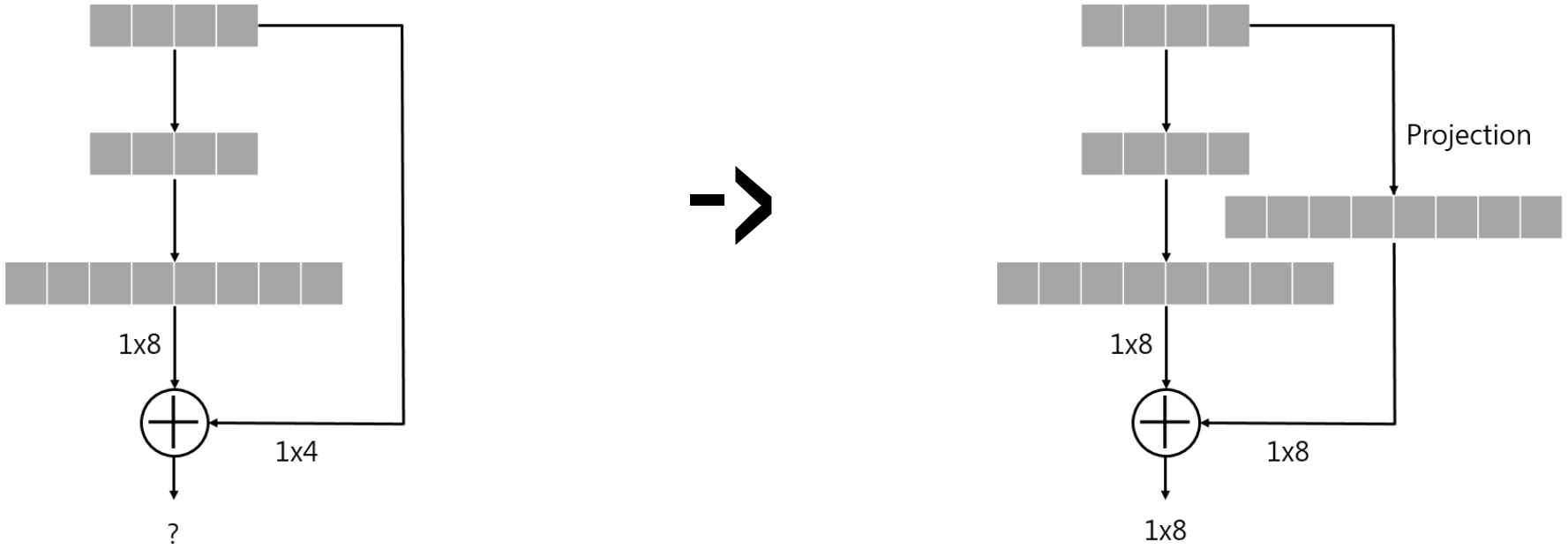


1. $y = F(x, \{W_i\}) + x$
2. $y = F(x, \{W_i\}) + W_s x$

- 그럼 1번과 2번의 차이는 무엇인가?
- F + x 시에 둘의 dimension이 같아야 하는데 같지 않을 경우 linear projection W_s로 dimension matching을 해주는 거다.
- 잠깐! projection은 또 무엇인가?

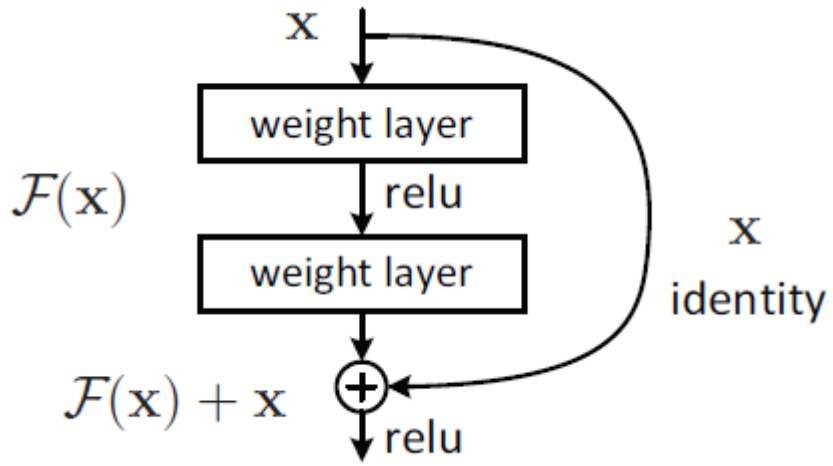
projection이 뭔데?

- 현재 데이터를 다른 차원에서 보기 위해 취해주는 과정이라고 생각하면 된다.



- 왼쪽 그림에서 입력 차원과 출력 차원이 달라 addition이 불가능하다
- 이 때 projection 과정을 추가해서 차원을 맞춘 후에 더한다
- FC layer에선 노드 개수만 맞추면 되지만, conv layer에서는 feature size와 channel 개수까지 맞춰야 한다.

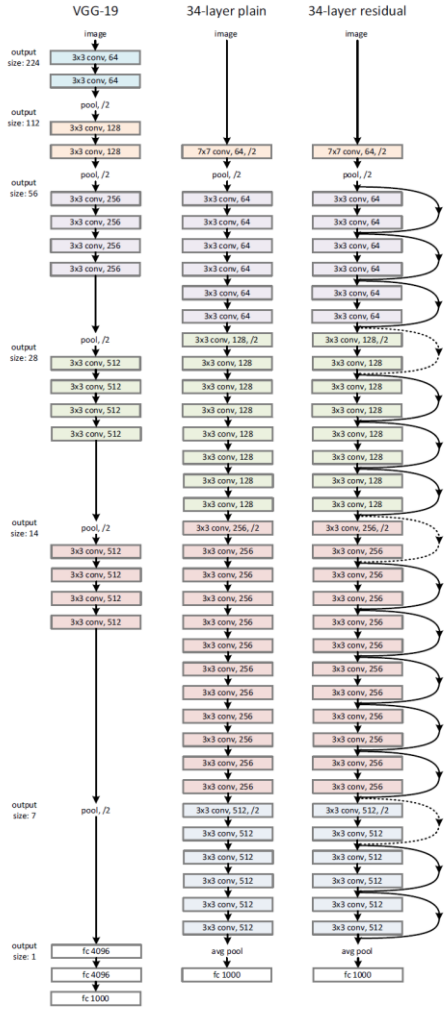
이 과정을 다시 좀 더 자세히 살펴보자
Identity Mapping by Shortcuts

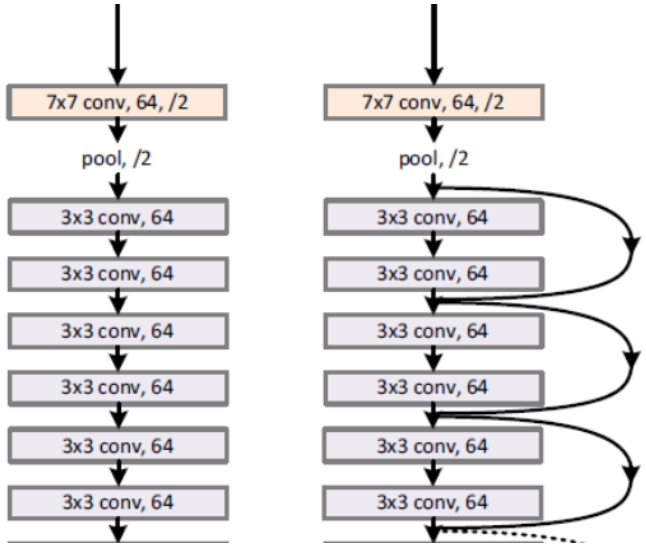


1. $y = F(x, \{W_i\}) + x$
2. $y = F(x, \{W_i\}) + W_s x$

- layer는 2개만 가능한건감?
- 아니다. F의 형태는 유연하게 결정 가능하다.
- 더 많은 layer를 추가해도 된다는 말!

그럼 이제 network architectures를 살펴보자

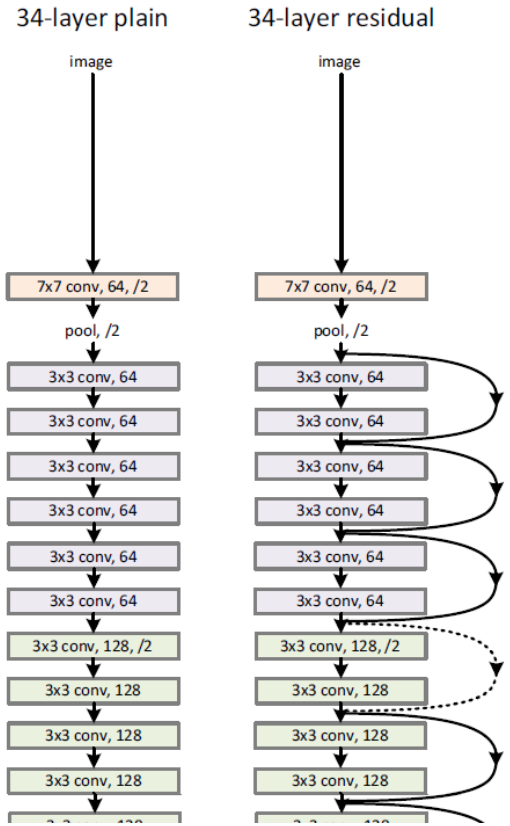




- 논문에서는 두가지 네트워크를 비교한다
- 왼쪽이 34층 plain network
- 오른쪽이 34층 residual network다
- 맨 왼쪽은 모델 구성에 영감이 된 19층 VCG

- 논문에 있는 네트워크 설명을 옮겨보자면
- plain network
  - 동일한 output feature map size에 대해, layer는 동일한 수의 filter를 갖는다.
  - feature map size가 절반인 경우, layer 당의 time complexity를 보전하기 위해 filter의 수를 2배로 한다.
  - downsampling 시 stride = 2, 네트워크 마지막에는 average pooling과 softmax activation을 사용한 1000-way FC layer로 구성된다.

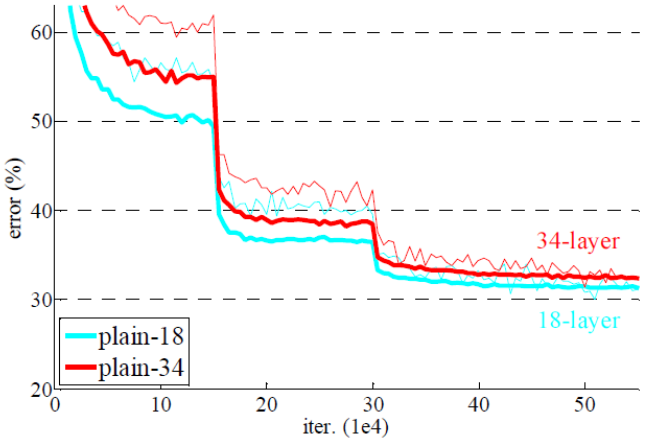
그럼 이제 network architectures를 살펴보자

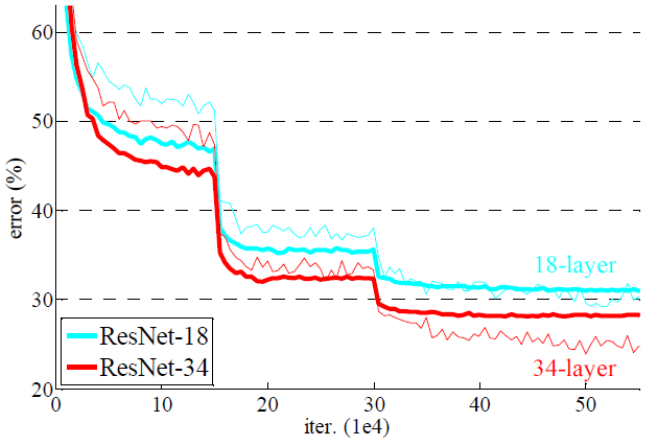


- residual network
  - plain network에서 short connection 추가
  - identity shortcut은 input과 output이 동일한 dimension인 경우에는 직접 사용될 수 있으며, dimension이 증가하는 경우(왼쪽 그림에서 점선)에는 아래의 두 옵션을 고려한다.
    - zero entry를 추가로 padding하여 dimension matching 후 identity mapping을 수행한다. (별도의 parameter가 추가되지 않음)
    - projection shortcut을 dimension matching에 사용한다.
  - shortcut connection이 다른 크기의 feature map 간에 mapping될 경우, 두 옵션 모두 strides를 2로 수행한다.

그래서 결과는?

- 데이터셋 분할과 구체적 실험 과정을 보고싶다면
- https://sike6054.github.io/blog/paper/first-post/
- 여기서 4.Experiments를 참고하자
- 본인은 과정 이해를 중심으로 했기 때문에 결과해석부분은 자세하게 정리하지 않았다.....





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

- ImageNet dataset을 학습하는 동안의 training/validation error 추이를 나타낸 그래프
- 초반에 문제가 되었던 깊은층의 높은 에러율 문제가 resnet으로 인해 해결!!된 것을 볼 수 있다.
- 이후 identity shortcut vs. projection shortcut, deeper bottleneck architectures, 50-layer resnet, 101-layer 152-layer resnets 등 여러 실험 결과들이 논문 결과정리에 나와있다.
- 이에 대해 자세히 보고 싶다면 깃헙 블로그로 가보자.....

다시 논문의 첫 질문!

"Is learning better networks as easy as stacking more layers?"
층을 쌓을수록, 모델의 깊이가 깊어질수록, 더 좋은 모델이 되는가?

"Residual Network라면 가능하다!"