Deep Residual Learning for Image Recognition 리뷰

13신민정

Abstract

본 논문은 이전의 연구에서 다룬 네트워크보다 훨씬 더 깊어진 네트워크의 용이한 학습을 위해 residual learning framework를 제안한다. 이 논문에서 말하는 residual function 이란 layer의 입력을 참고하는 형태이다. 이러한 Residual network는 optimize가 쉽고, 깊은 네트워크에서도 정확도를 얻을 수 있다. VGG보다 8배 깊어진 152 layer를 사용한 residual network를 ImageNet으로 평가하였다. VGG보다 좋은 성능을 내고 복잡도는 감소하였고. 3.57%의 error를 달성해서 ILSVRC 2015에서 1등을 차지했다.

1.Introduction

Network들의 깊이가 깊어질수록 좋은 성능을 보여줬다. 이에 따라'model의 layer를 단순히 쌓으며 depth을 늘리는 것이 학습에 유리한가?'라는 궁금증이 생기게 되었다. 하지만 Deep network의 깊이가 어느 정도 깊어지면, 수렴을 방해하는 vanishing/exploding gradient 문제가 있다는 것을 알 수 있다. 이는 normalized initialization(->weight initialization)과 intermediate normalization(->batch normalization)을 통해 해결되었다. 그러나 또 다른 문제가 발생한다.

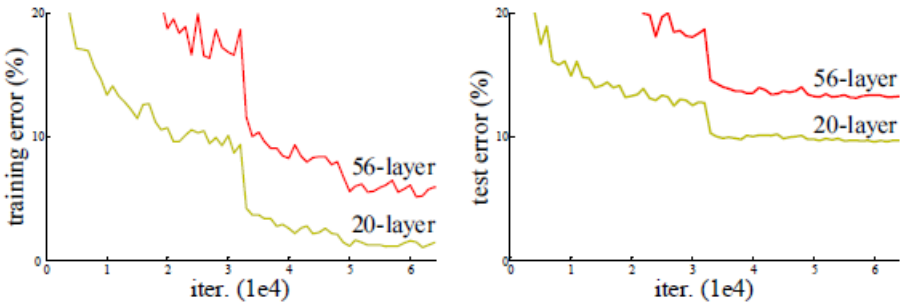


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Figure1에서 볼 수 있듯, layer가 깊은 network의 training error와 test error가 layer가 얕은 network보다 높게 나타난다(Degradation Problem). 이것은 overfitting의 문제가 아니라 최적화가 쉽지 않기 때문에 나타나는 현상이며, 모든 시스템의 최적화가 유사하지 않음을 나타낸다.

본 논문에서는 이 문제를 Residual Learning Framework를 도입해서 해결한다.

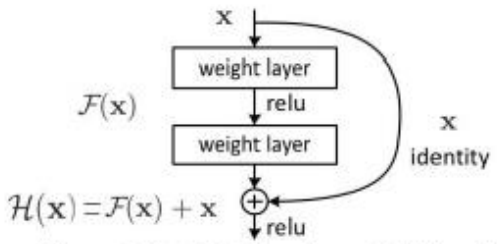


Figure 2. Residual learning: a building block. + 손글씨

이는 기존의 방식으로 H(x)를 optimization 하는 것이 아닌

$F(x) = 0$이 되도록 optimize를 하는 것이다. Residual Learning의 Residual(잔차)은 H(x)와 x의 잔차인 F(x)를 의미하는 것이다.

$H(x) = F(x) + x$ 는 shortcut connection(skip connection)으로 구현할 수 있다. Shortcut connection 이란 하나 이상의 layer를 skip(건너뜀) 하는 것이다. 본 논문에서는 Figure2와 같이 shortcut connection이 identity mapping을 수행하고, 그 출력을 stacked layer의 출력에 더해진다. Input인 x가 output에 더해지는 것이기 때문에 추가적인 파라미터가 필요하지도 않고 연산 복잡도 또한 증가하지 않는다. 또한 backpropagation의 gradient 측면에서도 유리하다.

2.Related Work

Residual Representations

Shortcut Connection

3. Deep Residual Learning

3.1 Residual Learning

1에서 언급한 것 처럼, 학습 시 H(x)이 아닌 F(x)(residual function)을 사용하는 것이 더 효과적이다. 수학적으로 같은 의미이지만(F(x)=H(x)-x), 식의 형식에 따라 학습의 효율이 달라진다.

3.2. Identity Mapping by Shortcuts

본 논문에서는 몇 개의 연속된 layer마다 residual learning을 적용한다.

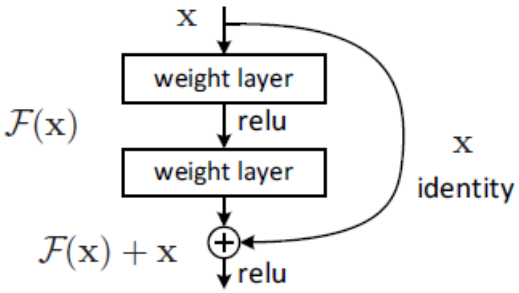


Figure 2. Residual learning: a building block.

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

(x : input vector / y : output vector)

$\mathcal{F}(\mathbf{x}, \{W_i\})$은 residual mapping을 의미한다. Figure2같은 경우에는 $\mathcal{F} = W_2\sigma(W_1\mathbf{x})$로 표기할 수 있고, 여기서 σ는 Relu(렐루~) function이다. (bias생략)

F+x는 shortcut connection(skip connection)과 element-wise addition(+)으로 수행된다. **1**에서 언급한 바와 같이 추가되는 parameter나 연산량 증가는 없다.

F와 x의 차원은 같아야 하며, 같지 않은 경우 x에 projection을 하여 차원을 맞추어 준다.

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

(W_s : linear projection)

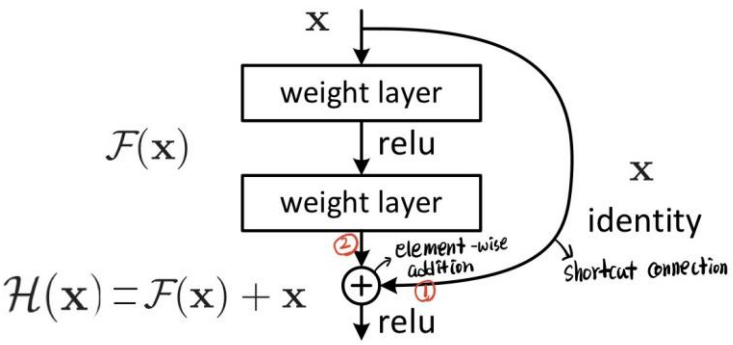


Figure 2. Residual learning: a building block. + 손그림

위 그림에서 ①(x)과 ②의 차원이 다르다면 ①에 projection을 적용하여 ②의 차원과 맞추어주는 것이다.

본 논문에서는 2~3개의 layer가 포함된 F를 사용하지만, F로 묶을 수 있는 layer수는 유연하게 결정 가능하다. (1의 layer를 F로 묶는 것은 residual mapping의 의미 없쩡!)

3.3 Network Architectures

*4.Experiments*에서 사용 할 Plain Network와 Residual Network 설명

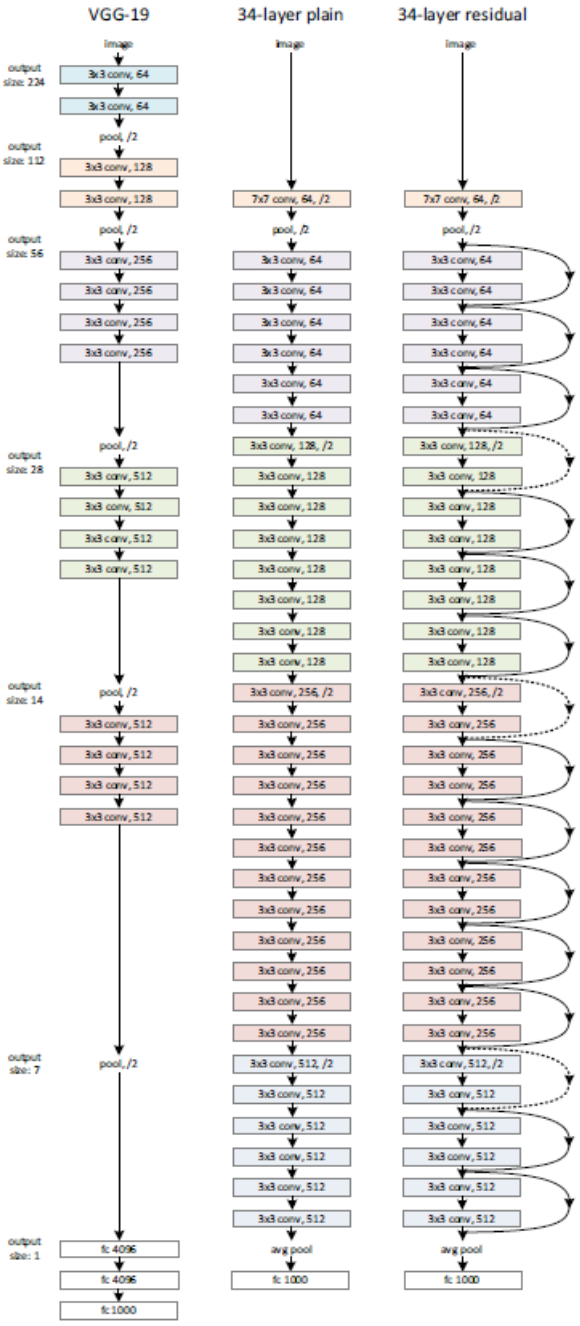


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [40] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Plain Network

VGGNet의 구조에 기본을 둔다. Convolutional layer는 거의 3x3 filter이며, 아래 두가지 규칙을 따른다.

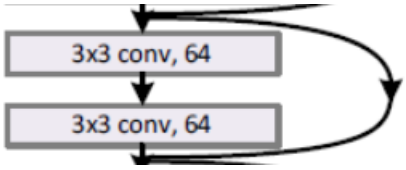
(i) 같은 output feature map의 크기를 갖는 layer는 filter 개수가 같다.

(ii) feature map의 크기가 절반(halve : Half의 verb....?) 필터의 개수는 두배이다.

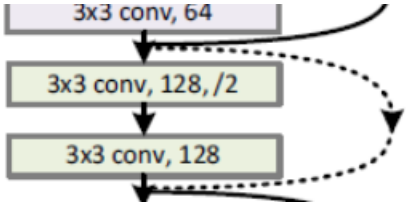
(down sampling -> stride2사용. 마지막 global average plling,1000-way FC 다음 softmax)

Residual Network

Plain Network에 short connection을 추가하여 만든다.



(input과 output의 차원이 같을 경우, 3.2에서 설명한 identity shortcut (Eqn1)사용)



(input과 output의 차원이 다를 경우(3.2에 숫자로 설명한 부분))

차원을 늘려야 할 때는, Zero padding으로 차원을 증가시킨 후 identity mapping을 수행하거나, projection shortcut(3.2의 Eqn2)으로 차원을 늘려준다.

3.4 Implementation

Data augmentation (horizontal flip, per-pixel mean subtract, standard color augmentation)

Batch Normalization

Stochastic Gradient Descent (minibatch size = 256)

Learning rate = 0.1(error plateau마다 나누기 10)

# of iteration = 60×10^4

Weight decay = 0.0001

Momentum = 0.9

Drop out X

Weight Initialization -> Xavier (??잘모르겠어여..)[12]에서 사용한 weight initialization사용

2.2. Initialization of Filter Weights for Rectifiers

Rectifier networks are easier to train [8, 16, 34] compared with traditional sigmoid-like activation networks. But a bad initialization can still hamper the learning of a highly non-linear system. In this subsection, we propose a robust initialization method that removes an obstacle of training extremely deep rectifier networks.

Recent deep CNNs are mostly initialized by random weights drawn from Gaussian distributions [16]. With fixed standard deviations (*e.g.*, 0.01 in [16]), very deep models (*e.g.*, >8 conv layers) have difficulties to converge, as reported by the VGG team [25] and also observed in our experiments. To address this issue, in [25] they pre-train a model with 8 conv layers to initialize deeper models. But this strategy requires more training time, and may also lead to a poorer local optimum. In [29, 18], auxiliary classifiers are added to intermediate layers to help with convergence.

Glorot and Bengio [7] proposed to adopt a properly scaled uniform distribution for initialization. This is called "*Xavier*" initialization in [14]. Its derivation is based on the assumption that the activations are linear. This assumption is invalid for ReLU and PReLU.

In the following, we derive a theoretically more sound initialization by taking ReLU/PReLU into account. In our experiments, our initialization method allows for extremely deep models (*e.g.*, 30 conv/fc layers) to converge, while the "*Xavier*" method [7] cannot.

Reference [12]의 한부분

(Keras로 구현한 ResNet : https://eremo2002.tistory.com/76)

4 Experiments

4.1 ImageNet Classification

ImageNet 2012 classification data set은 1000개의 class로 구성되어 있으며, trainset은 1.28million, validation set은 50k, test set은 100k 개의 이미지 data가 있다. 테스트 결과는 top-1 error와 top-5 error를 모두 평가한다.

Plain Networks

(Figure3의 가운데 구조, Residual mapping이 없는 구조)

plain network의 형태로 쌓은 18-layernetwork와 34-layer network를 비교해보자.

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

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

위의 Table2에서 보면 18-layer network에 비해, 34-layer network의 validation error가 높다.

아래 Figure4를 통해 training error와 validation error를 확인해보자.

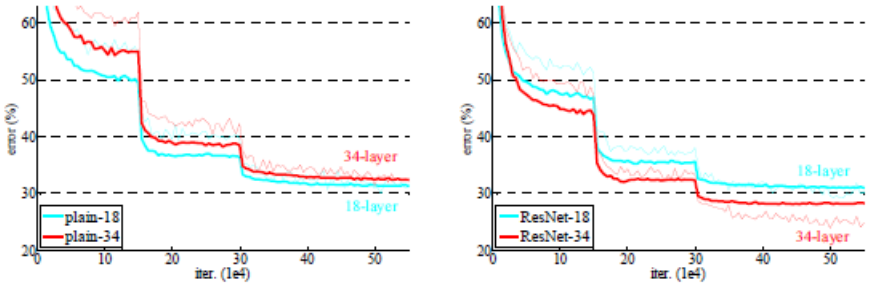


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

(왼쪽-plain net 오른쪽-residual net / 굵은 선-validation error 얇은 선-training error)

Plain network의 경우 깊은 network가(34-layer net) 오히려 error가 높게 나타난다.

본 논문에서는 이 문제가 vanishing gradients에 의한 것은 아니라고 말한다. plain network가 batch normalization을 사용하였기 때문에 forward propagated signal의 분산이 0이 되고, backward propagated gradients가 healthy norm을 보이기 때문이다.

Residual Networks

18-layer 및 34-layer residual network(이하 ResNet)를 평가한다.

Figure3의 34-layerresidualnetwork(Figure3의가운데)와 동일하게 각각의 3x3 filter pair에 shortcut connection을 추가했다. 모든 shortcut connection은 identity mapping을 사용하며, 차원을 맞추기 위하여 zero-padding을 사용한다.

(추가되는 parameter 없다고 3번째 말하는 중.....ㅡ,.ㅡ)

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

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

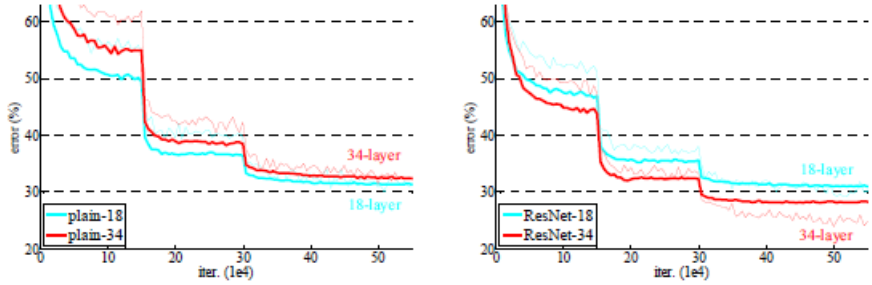


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

(왼쪽-plain net 오른쪽-residual net / 굵은 선-validation error 얇은 선-training error)

위의 Table2와 Figure4로 알 수 있는 세 가지 observation이 있다.

첫째, 34-layer ResNet이 18-layer ResNet보다 우수한 성능을 보인다. 이는 위의 Plainnetwork에서 직면한 성능 저하 문제를 잘 해결하고 깊은 network가 높은 정확성을 보장하는 것을 의미한다.

둘째, 34-layer ResNet과 34-layerplain network와 비교할 때, validation data에 대한 top-1 error를 3.5% 줄였다. 이는 extremely deep systems에서도 residual learning이 효과적임을 나타낸다.

셋째, Table2에서 18-layerResNet과 18-layer plain network를 비교하면 accuracy는 비슷하지만, Figure4에서 보면 ResNet의 경우가 더 빨리 수렴하는 것을 알 수 있다. 이는 SGD가 ResNet에서 더 빠르게 수렴한 기 때문에 optimization이 더 쉽다는 것을 의미한다.

Identity vs Projection Shortcuts

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**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

위에서 parameter-free 한 identity shortcut이 학습에 용이하다는 것을 알아봤다.

$y = \mathcal{F}(x, \{W_i\}) + W_s x.$ projection shortcuts에 대해 조사해보겠다.

(in 3.2, projection shortcut: Wi는 x의 dimension을 맞추기 위해서 사용)

Table3의 3가지의 옵션->(A, B, C).

A. zero-padding shortcut은 차원을 높이기 위해 사용, 모든 shortcut은 parameter-free.

B. projection shortcut은 차원을 높이기 위해 사용, 다른 shortcut은 모두 identity

C. 모든 shortcut은 projection.

성능은 C>B>A이지만, 차이가 미미하다. Projection shortcut은 성능 저하를 해결하는 데 크게 효과를 내지 못한다는 것을 알 수 있다.

(본 논문에서는 옵션 C를 사용하지 않는다. Memory/time complex와 model size를 줄이기 위해,)

Deeper Bottleneck Architecture

ImageNet data set 사용을 위한 deep network 구조를 알아보자. 이 경우 Training 시간을 고려하여 building block architecture 구조에 Bottleneck 구조를 사용한다

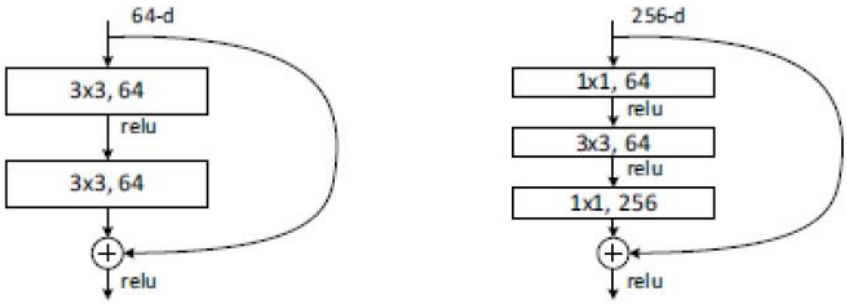


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

(왼쪽-> ResNet34, 오른쪽-> bottleneck 구조 for ResNet50/101/152)

Figure5오른쪽 bottleneck 구조에서 1x1 convolution layer는 차원을 조절하기 위해 사용되며, 3x3 convolution layer는 input/output의 차원을 줄이는 용도로 사용된다.

50-layer ResNet

34-layer ResNet의 2-layer block들을 3-layer bottleneck block으로 대체하여(Figure5의 왼쪽 구조를 오른쪽 구조로 대체) 50-layer ResNet을 구성했다. 옵션 B.(projection shortcut은 차원을 높이기 위해 사용, 다른 shortcut은 모두 identity)를 사용했다.

101-layer and 152-layer ResNets

3-layer bottleneck block을 더 추가하여 101-layer/152-layer ResNet을 구성했다. 34-layer보다 50/101/152의 정확도가 더 높다. 특히 152-layer ResNet은 VGG와 비교하였을 때, 복잡도와 연산량이 적다

Comparisons with State-of-the-art Methods

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**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

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

Table.4에서는 previous best single-model의 성능과 비교한다. 본 논문의 baseline인 34-layer ResNet은 previous best에 비준하는 정확도를 달성했으며, 152-layer ResNet의 single-model top-5 error는 4.49%를 달성했다. 또한, 서로 다른 depth의 ResNet을 6개 ensemble 하여 top-5 test error를 3.57%까지 달성했고. 이는 ILSVRC 2015 classification task에서 1위를 차지했다.

4.2 CIFAR-10 and Analysis

CIFAR-10은 50K의 training image, 10K의 test image가 있다.

model구조

Input image size = 32x32 with per-pixel mean subtracted

First layer = 3x3 conv

6n layer(with 3x3 conv on the feature map size of {32,16,8 -> 2n layer로 구성}

# of Filter {16,32,64}

마지막 global average plling,10-way FC 다음 softmax

(total 6n+2 stacked weighted layer)

output map size	32×32	16×16	8×8
# layers	1+2n	2n	2n
# filters	16	32	64

Shortcut connection은 모두 identity shortcut, residual mapping을 제외하고 residual network와 plain network의 조건이 같음

Training condition

Weight initialization – Xavier [3.4와 같은 initializer]

Weight decay = 0.001

Momentum = 0.9

Batch Normalization

DropOut X

Minibatch = 128 on Two GPU

Learning rate = 0.1(32k,48k번째 iteration에서 나누기 10)

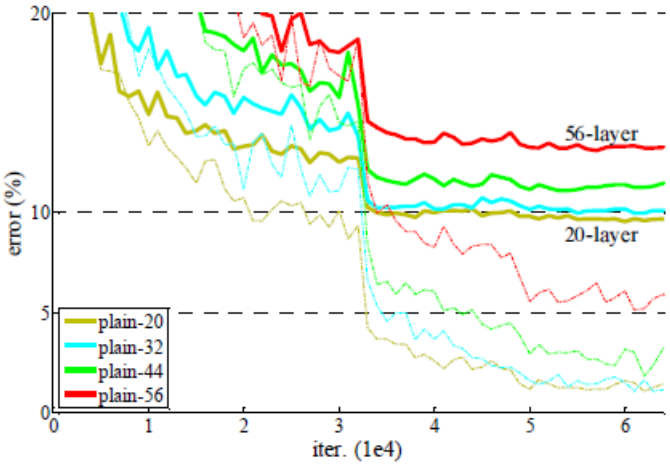
Testing Condition

3x3 original image의 sing view

결과~

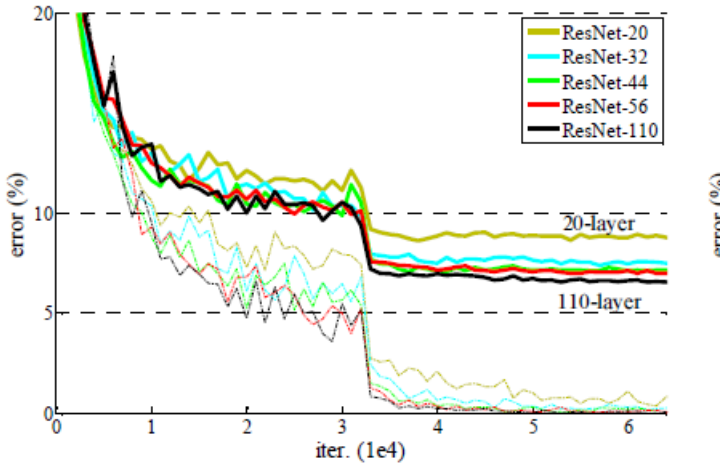
(얇은선 – training / 굵은선 – testing)

Plain Networks



ImageNet과 동일하게 깊이가 깊어질수록 error가 크다. Degradation problem이겠징

ResNet



ImageNet과 마찬가지로, Plain Network의 문제가 해결된 것이 보인다. 깊이가 깊을수록 error가 낮다.

method			error (%)
Maxout [9]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [34]	19	2.5M	8.39
Highway [41, 42]	19	2.3M	7.54 (7.72±0.16)
Highway [41, 42]	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	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Table 6. Classification error on the **CIFAR-10** test set. All methods are with data augmentation. For ResNet-110, we run it 5 times and show "best (mean±std)" as in [42].

110-ResNet이 가장 성능이 좋더라~

Analysis of Layer Response

..?무슨소리징..?

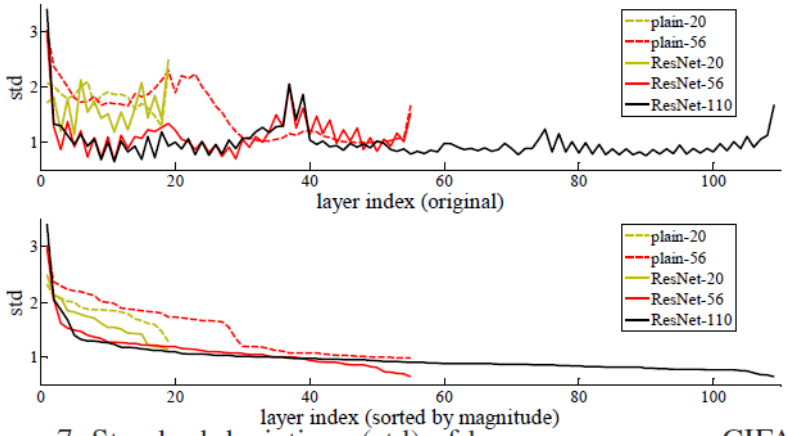
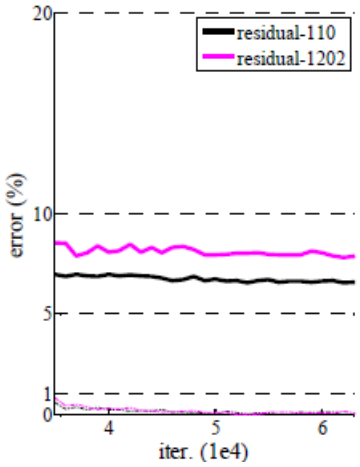


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

각 3x3 conv(BN 후, ReLU전)의 표준편차이다. ResNet이 Plain Network에 비해 small response를 보인것으로 보아, 조금 더 안정적으로 학습을 진행하였다는 것을 알 수 있다.

출처: https://curaai00.tistory.com/1

Exploring Over 1000 layers



1202-ResNet과 110-ResNet을 비교해보았다. 1202-ResNet도 error가 낮게 나오기는 했지만, 110-ResNet보단느 성능이 떨어진다. 1202-layer정도의 아주 깊은 Network는 CIFAR-10 data set(small dataset)에는 적합하지 않은 것 같다.

4.3. Object Detection on PASCAL and MS COCO

training data	07+12	07++12
test data	VOC 07 test	VOC 12 test
VGG-16	73.2	70.4
ResNet-101	**76.4**	**73.8**

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

metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	**48.4**	**27.2**

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

PASCAL VOC&COCO dataset에서 objecti detection에, Fast RCNN+ResNet을 적용하였더니 기존 VGG보다 성능이 좋았다.

참고: https://datascienceschool.net/view-notebook/958022040c544257aa7ba88643d6c032/

결론:Deep으로 가면 Plain보다 ResNet 짱

팀원한테 설명하는 것 처럼 리뷰를 했는데, 논문을 리뷰 한 건지 번역한 건지 모를 정도로 양이 많아졌네욤 8ㅁ8 (9쪽 논문에 14쪽 리뷰라니...8ㅁ8)