philBaek (백광록)
philBaek (백광록) 개발 일기장

CATEGORY

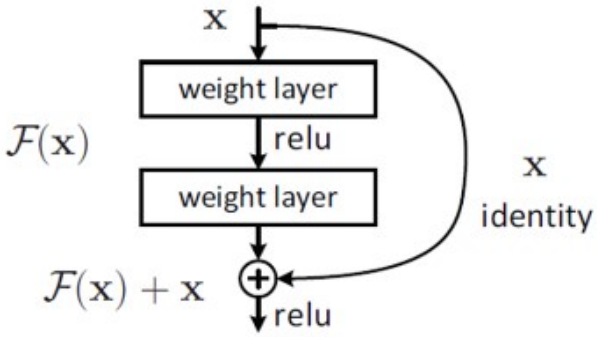
방구석 논문 리뷰

ResNet (Deep Residual Learning for Image Recognition) 논문 리뷰

philBaek (백광록) 2021. 1. 10. 21:01

논문 제목 : Deep Residual Learning for Image Recognition

오늘은 Deep Residual Learning for Image Recognition에서 마이크로소프트팀이 소개한 ResNet에 대해 다뤄보려 한다. ResNet은 수학적으로 어려운 개념이 적용되었다기보다는 **방법론적으로 신박한 개념**이 도입되었는데, 바로 **잔차 (Residual)라는** 개념이다.

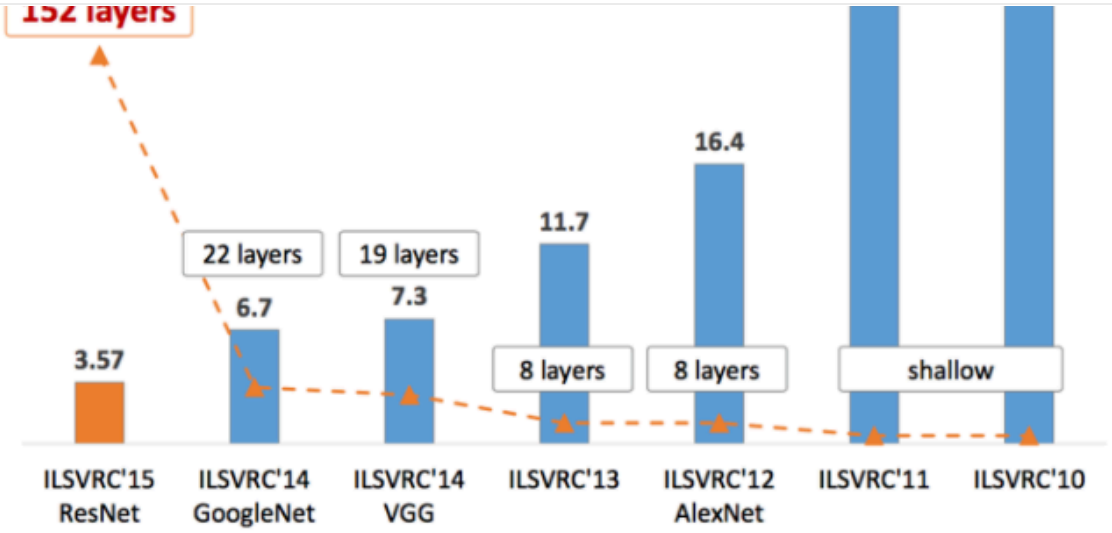


residual 개념

Residual이란 쉽게 말해서 결과의 오류 정도로 생각하면 되는데 **Y에서 X를 뺀 나머지**라고 생각하면 된다. 그렇기에 이전에는 Residual을 평가의 기준으로만 삼았지, 이를 이용해 학습을 진행한다는 생각은 없었는데 마이크로소프트에서는 이 Residual을 학습하는데 이용하였고, 그 결과 **ILSVRC 2015에서 1위**를 차지할 수 있었다! 그럼 무려 **152 layer**를 달성할 수 있었던 ResNet의 비결에 대해 알아보자!

philBaek (백광록) ... 구독하기

philBaek (백광록) 개발 일기장



ILSVRC 대회에 제출된 각 모델의 성능 및 layer 수

1. Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

초록에서는 **depth의 중요성과 residual learning**에 대해 간략히 설명하고 있다.
역대 ILSVRC 대회 결과를 보면, **dept의 깊이**가 모델의 성능에 큰 영향을 준다는 것을 알 수 있다. 즉, visual recognition task에서 depth는 매우 중요한 요소이다. 하지만, depth가 올라감에 따라 필연적으로 발생하는 문제도 있는데, **오버 피팅, gradient의 소멸, 연산량의 증가** 등이 있다. 따라서 심층 신경은 학습시키기가 상당히 까다로운데, 이에 마이크로소프트 팀은 **residual learning framework**를 이용하여 이전보다 훨씬 깊은 네트워크를 더 쉽게 학습시킬 수 있었다고 한다.

오차를 **3.75%**까지 줄일 수 있었다.

2. Introduction

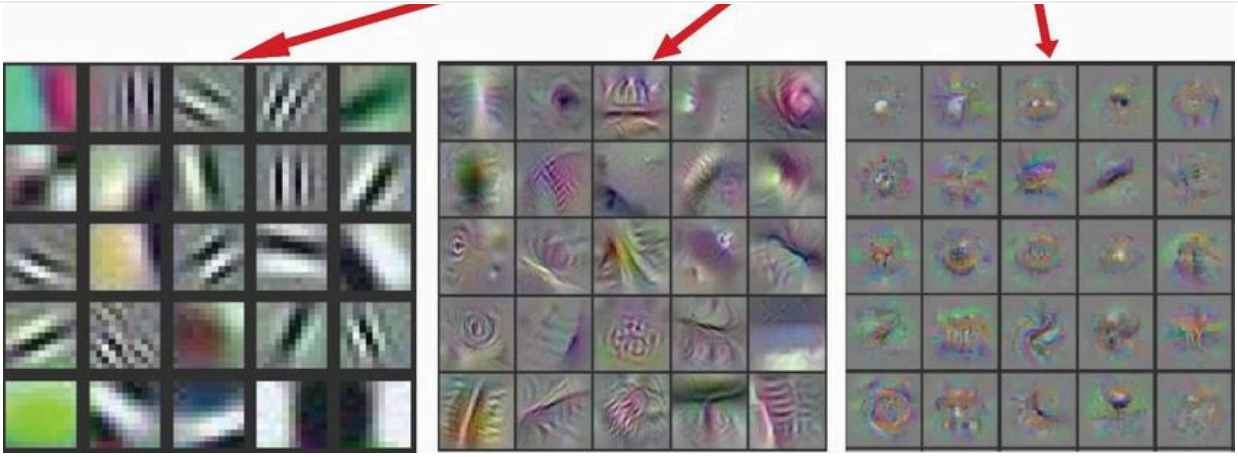
소개에서는 residual learning에 대해 자세하게 설명하고 있어 내용이 조금 길다. 따라서 섹션 2. 소개에서 이에 대한 내용을 설명하고 섹션 3에서는 해당 내용을 스킵하도록 하겠다!

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 50, 40]. Deep networks naturally integrate low/mid/high-level features [50] and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Recent evidence [41, 44] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the challenging ImageNet dataset [36] all exploit "very deep" [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-trivial visual recognition tasks [8, 12, 7, 32, 27] have also greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [1, 9], which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization [23, 9, 37, 13] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with back-propagation [22].

심층 신경망은 추상화에 대한 low / mid / high level의 특징을 classifier와 함께 multi-layer 방식으로 통합한다. 여기서 각 추상화 level은 쌓인 layer의 수에 따라 더욱 높아질 수 있다. 즉, 높은 추상화 특징은 high layer에서 파악할 수 있다는 것이다.

philBaek (백광록) 개발 일기장



layer의 level에 따라 다르게 추출되는 특징

최근 연구결과에 따르면 네트워크의 깊이는 매우 중요한데, 실제로 많은 모델들이 깊은 네트워크를 통해 좋은 성능을 보였다고 한다. 이렇게 depth의 깊이가 중요해지면서, layer를 쌓는 만큼 더 쉽게 네트워크를 학습시킬 수 있는지에 대한 의문이 생기기 시작했고, 특히 그중에서도 **Vanishing / Exploding gradient 현상**이 큰 방해 요소였다. 다행히도 SGD를 적용한 10개의 layer까지는 normalization 기법과 Batch normalization과 같은 intermediate normalization layer를 사용했을 경우, 문제가 없었다고 한다.

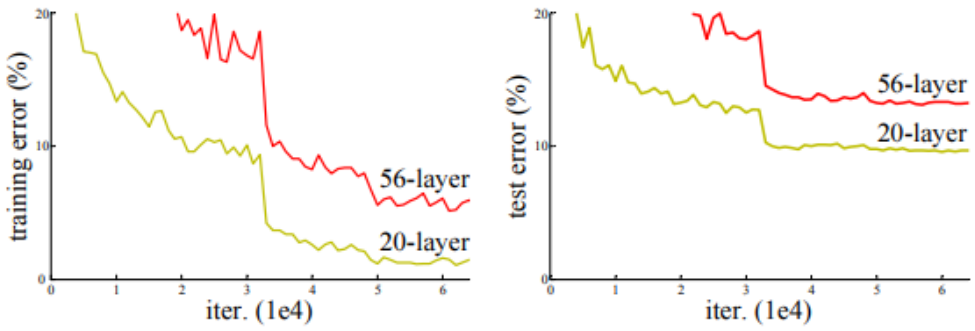


그림 1. layer 수에 따른 training error와 test error

When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [11, 42] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

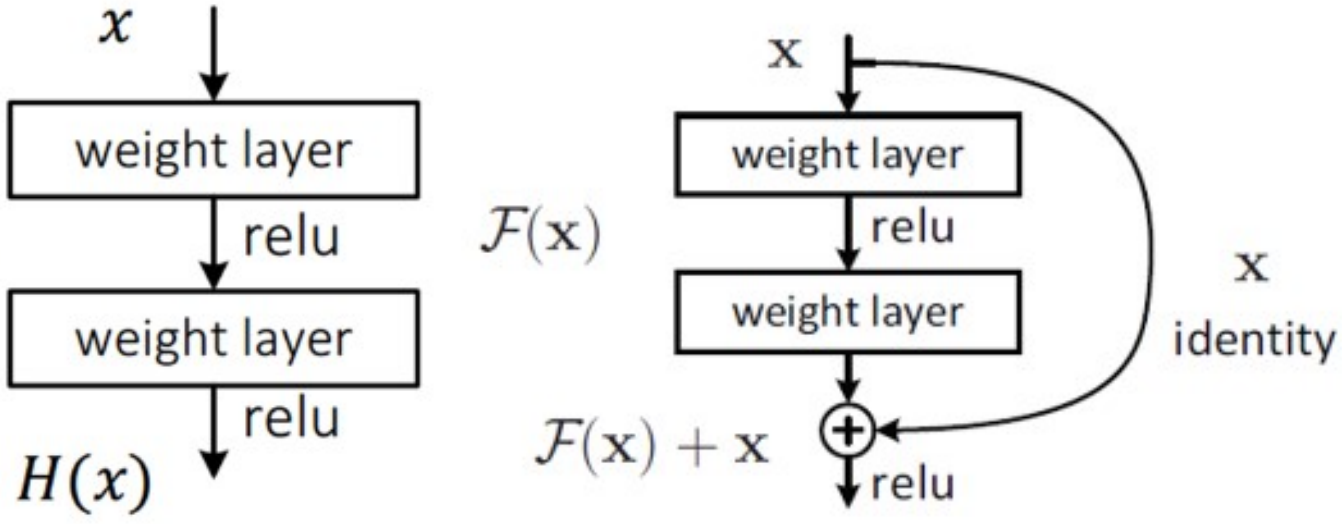
하지만 심층 신경망의 경우, **성능이 최고 수준에 도달할 때 degradation 문제가 발생**하였고, 이는 네트워크의 깊이가 깊어 짐에 따라 정확도가 포화하고, 급속하게 감소하는 것을 의미한다. 이러한 degradation 문제의 원인

philBaek (백광록) 개발 일기장

> The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution *by construction* to the deeper model: the added layers are *identity* mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates

하지만, deeper 모델에서도 제한된 상황에서는 최적화될 수 있는 방법이 있다. 이 방법은 **추가된 레이어가 identity mapping이고, 추가되지 않은 다른 레이어들은 더 얕은 모델에서 학습된 layer를 사용하는 것**이다. 이 같이 제한된 상황에서의 deeper 모델은 shallower 모델보다 더 낮은 training error를 가져야 하고, 마이크로소프트 팀은 이 개념을 모델에 적용하였다.

기존 네트워크는 입력 x를 받고 layer를 거쳐 H(x)를 출력하는데, 이는 입력값 x를 타겟값 y로 mapping 하는 함수 H(x)를 얻는 것이 목적이다. 여기서 ResNet의 Residual Learning은 H(x)가 아닌 **출력과 입력의 차인 H(x) - x를 얻도록 목표를 수정**한다. 따라서 Residual Function인 **F(x) = H(x) - x를 최소화**시켜야 하고 이는 **즉, 출력과 입력의 차을 줄인다는 의미**가 된다. 여기서 **x의 값은 도중에 바꾸지 못하는 입력 값이므로 F(x)가 0이 되는 것이 최적의 해**이고, 결국 0 = H(x) - x로 H(x) = x가 된다. 즉, **H(x)를 x로 mapping 하는 것이 학습의 목표**가 된다.



기존 네트워크와 ResNet의 구조

이전에는 **Unreferenced mapping인 H(x)를 학습시켜야 한다는 점 때문에** 어려움이 있었는데 (알지 못하는 최적의 값으로 H(x)를 근사 시켜야 하므로), 이제는 **H(x) = x라는 최적의 목표값이 사전에 pre-conditioning으로 제공되기에 Identity mapping이 Identity mapping인 F(x)가 학습이 더 쉬워지는 것**이다.

philBaek (백광록) 개발 일기장

무엇보다도 **곱셈 연산에서 덧셈 연산으로 변형**되어 몇 개의 layer를 건너뛰는 효과가 있었는데, 이 덕에 **forward와 backward path가 단순해지는 효과가 있었으며, gradient의 소멸 문제를 해결**할 수 있었다고 한다.

$$x_{l+1} = x_l + F(x_l) \quad \Rightarrow \quad x_{l+2} = x_{l+1} + F(x_{l+1})$$
$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

수식을 보면 기존의 곱센 연산에서 덧셈 연산으로 바뀌는 것을 확인할 수 있다.

덧셈 연산으로 바뀌는 과정은 이 링크를 참고하면 된다.

ResNet은 depth가 매우 깊어도 최적화가 쉬웠지만, 일반적인 CNN model인 plain net은 depth가 증가하면 training error도 함께 증가하였다. 반면에, ResNet은 매우 깊은 구조 덕에 높은 정확도를 쉽게 얻을 수 있었고, 그 결과는 이전의 다른 네트워크보다 월등했다. 또한 ImageNet, CIFAR-10 모두에서 훌륭한 성능을 보였고, 여러 ResNet 모델을 ensemble 했을 때, 에러는 3.75%에 불과하였다.

3. Related Work

이 섹션에서는 ResNet 모델의 대표적인 2가지 개념인 Residual Representations와 Shortcut Connections의 다른 모델에 적용된 사례에 대해 간략히 설명하고 있다.

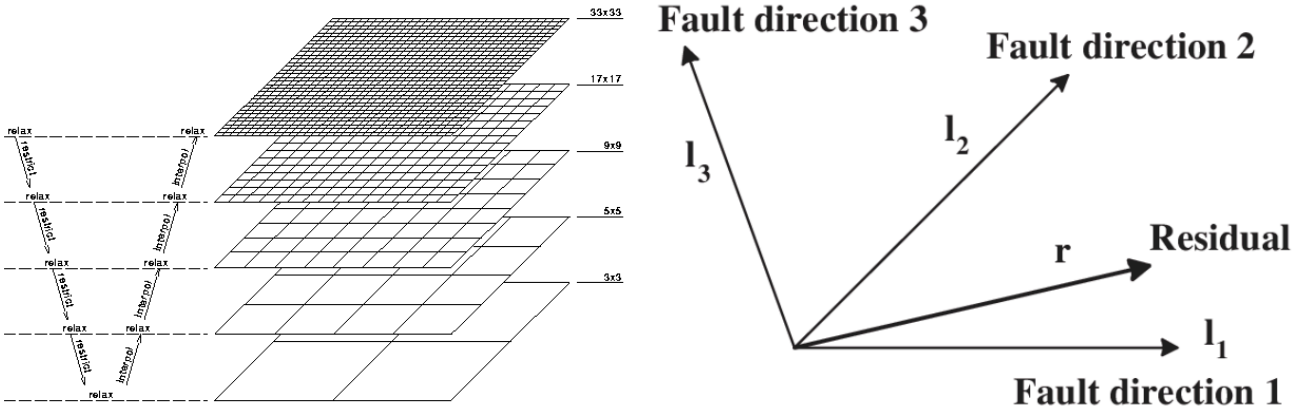
Residual Representations

philBaek (백광록) 개발 일기장

formulated as a probabilistic version [18] of VLAD. Both of them are powerful shallow representations for image retrieval and classification [4, 48]. For vector quantization, encoding residual vectors [17] is shown to be more effective than encoding original vectors.

In low-level vision and computer graphics, for solving Partial Differential Equations (PDEs), the widely used Multigrid method [3] reformulates the system as subproblems at multiple scales, where each subproblem is responsible for the residual solution between a coarser and a finer scale. An alternative to Multigrid is hierarchical basis preconditioning [45, 46], which relies on variables that represent residual vectors between two scales. It has been shown [3, 45, 46] that these solvers converge much faster than standard solvers that are unaware of the residual nature of the solutions. These methods suggest that a good reformulation or preconditioning can simplify the optimization.

뭐 VLAD와 Fisher Vector 등 다른 사례에 대해 설명하고 있는데, 결국 핵심 내용은 **벡터 양자화에 있어 residual vector를 인코딩하는 것이 original vector보다 훨씬 효과적**이다는 것이다. 여기서 벡터 양자화란 특징 벡터 X를 클래스 벡터 Y로 mapping 하는 것을 의미한다.



multi grid 방식과 residual vector 방식

low-level 비전 및 컴퓨터 그래픽 문제에서 편미분 방정식을 풀기 위해 멀티 그리드 방식을 많이 사용해왔는데, 이 방식은 시스템을 여러 scale의 하위 문제로 재구성하는 것이다. 이때, 각 하위 문제는 더 큰 scale과 더 작은 scale 간의 residual을 담당한다.

여기서 멀티 그리드 방식 대신에 두 scale 간의 residual 벡터를 가리키는 변수에 의존하는 방식이 있는데, 이를 계층 기반 pre-conditioning이라고 한다. 이 방식은 해의 residual 특성에 대해 다루지 않는 기존 방식보다 훨씬

philBaek (백광록) 개발 일기장

Shortcut Connections

> Concurrent with our work, "highway networks" [42, 43] present shortcut connections with gating functions [15]. These gates are data-dependent and have parameters, in contrast to our identity shortcuts that are parameter-free. When a gated shortcut is "closed" (approaching zero), the layers in highway networks represent *non-residual* functions. On the contrary, our formulation always learns residual functions; our identity shortcuts are never closed, and all information is always passed through, with additional residual functions to be learned. In addition, high-

여기서도 Shortcut connection을 구성하기 위해 시도되었던 다양한 사례들에 대해 소개하고 있다. 그중에서 ResNet의 Shortcut connection은 다른 방식들과는 달리, **parameter가 전혀 추가되지 않으며, 0으로 수렴하지 않기에 절대 닫힐 일이 없어 항상 모든 정보가 통과된다고 한다. 따라서 지속적으로 residual function을 학습하는 것이 가능**하다고 한다.

4. Deep Residual Learning

이제 본격적으로 Residual Learning과 Shortcut에 대한 내용을 다루는데, 이에 대한 내용은 앞서 섹션 2. 소개에서 다루었으므로 간략하게만 언급하고 넘어가도록 하겠다.

4.1 Residual Learning

> In real cases, it is unlikely that identity mappings are optimal, but our reformulation may help to precondition the problem. If the optimal function is closer to an identity mapping than to a zero mapping, it should be easier for the solver to find the perturbations with reference to an identity mapping, than to learn the function as a new one. We show by experiments (Fig. 7) that the learned residual functions in general have small responses, suggesting that identity mappings provide reasonable preconditioning.

philBaek (백광록) 개발 일기장

mapping보다 identity mapping에 더 가깝다면, solver가 identity mapping을 참조하여 작은 변화를 학습하는 것이 새로운 function을 학습하는 것보다 더 쉬울 것이라고 마이크로소프트팀은 주장한다.

4.2 Identity Mapping by Shortcuts

The shortcut connections in Eqn.(1) introduce neither extra parameter nor computation complexity. This is not only attractive in practice but also important in our comparisons between plain and residual networks. We can fairly compare plain/residual networks that simultaneously have the same number of parameters, depth, width, and computational cost (except for the negligible element-wise addition).

The dimensions of $\mathbf{x}$ and $\mathcal{F}$ must be equal in Eqn.(1). If this is not the case (*e.g.*, when changing the input/output channels), we can perform a linear projection W_s by the shortcut connections to match the dimensions:

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

We can also use a square matrix W_s in Eqn.(1). But we will show by experiments that the identity mapping is sufficient for addressing the degradation problem and is economical, and thus W_s is only used when matching dimensions.

Shortcut connection은 파라미터나 연산 복잡성을 추가하지 않는다. 이때, F + x 연산을 위해 x와 F의 차원이 같아야 하는데, 이들이 서로 다를 경우 linear projection인 W_s 를 곱하여 차원을 같게 만들 수 있다. 여기서 W_s 는 차원을 매칭 시켜줄 때에만 사용한다.

4.3 Network Architectures

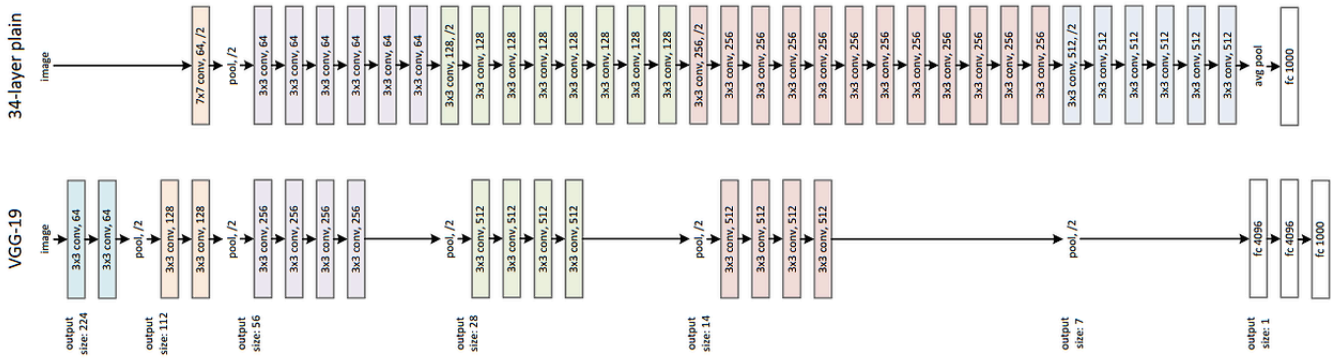
Plain Network

follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax. The total number of weighted layers is 34 in Fig. 3 (middle).

baseline 모델로 사용한 plain net은 VGGNet에서 영감을 받았다. 즉, conv filter의 사이즈가 3 x 3이고, 다음 2가지 규칙에 기반하여 설계하였다.

1. Output feature map의 size가 같은 layer들은 **모두 같은 수의 conv filter를 사용**한다.
2. Output feature map의 size가 반으로 줄어들면 time complexity를 동일하게 유지하기 위해 **필터 수를 2배로 늘려준다.**

추가로, 만약 downsampling을 수행한다면 pooling을 사용하는 것이 아니라 **stride가 2인 conv filter를 사용**한다. 마지막으로, 모델 끝단에 **GAP를 사용**하고, **사이즈가 1,000인 FC layer와 Softmax를 사용**한다. 결과적으로 전체 layer 수는 34인데 이는 VGGNet보다 적은 필터와 복잡성을 가진다. (VGGNet은 4,096 사이즈 FC layer가 2개 추가된다)



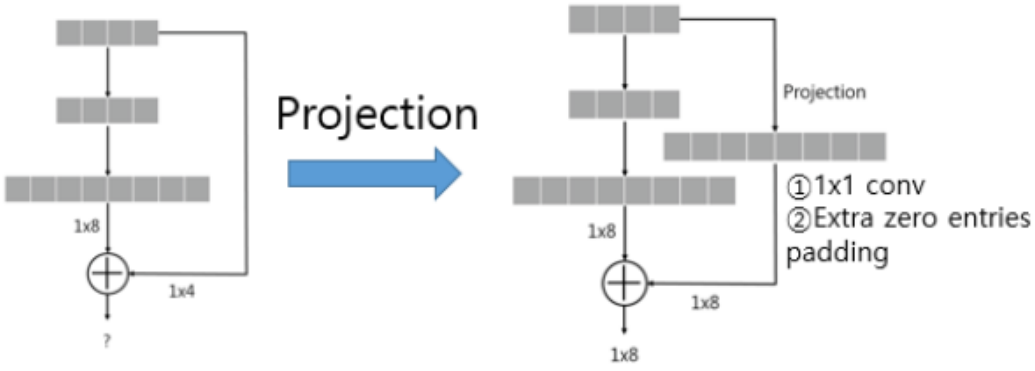
VGGNet과 34-layer plain net

Residual Network

shortcuts (Eqn.(1)) can be directly used when the input and output are of the same dimensions (solid line shortcuts in Fig. 3). When the dimensions increase (dotted line shortcuts in Fig. 3), we consider two options: (A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter; (B) The projection shortcut in Eqn.(2) is used to match dimensions (done by 1×1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

Residual Network는 Plain 모델에 기반하여 Shortcut connection을 추가하여 구성한다. 이때 input과 output의 차원이 같다면, identity shortcut을 바로 사용하면 되지만, dimension이 증가했을 경우 두 가지 선택권이 있다.

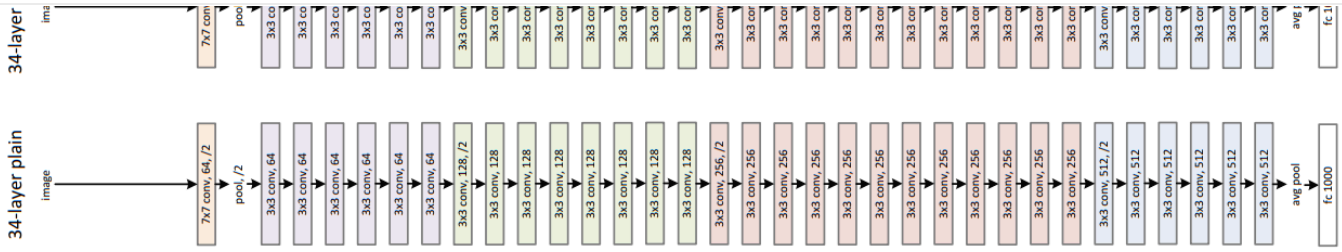
1. **zero padding**을 적용하여 차원을 키워준다.
2. 앞서 다뤘던 **projection shortcut**을 사용한다. **(1 x 1 convolution)**



dimension matching 방법 2가지

이때, shortcut이 feature map을 2 size씩 건너뛰므로 stride를 2로 설정한다.

philBaek (백광록) 개발 일기장



34-layer plain net과 34-layer ResNet

3.4 Implementation

Our implementation for ImageNet follows the practice in [21, 41]. The image is resized with its shorter side randomly sampled in [256, 480] for scale augmentation [41]. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [21]. The standard color augmentation in [21] is used. We adopt batch normalization (BN) [16] right after each convolution and before activation, following [16]. We initialize the weights as in [13] and train all plain/residual nets from scratch. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. We do not use dropout [14], following the practice in [16].

In testing, for comparison studies we adopt the standard 10-crop testing [21]. For best results, we adopt the fully-convolutional form as in [41, 13], and average the scores at multiple scales (images are resized such that the shorter side is in $\{224, 256, 384, 480, 640\}$).

모델 구현은 다음과 같이 진행한다.

1. 짧은 쪽이 [256, 480] 사이가 되도록 random 하게 resize 수행
2. horizontal flip 부분적으로 적용 및 per-pixel mean을 빼준다
3. 224 x 224 사이즈로 random 하게 crop 수행
4. standard color augmentation 적용
5. z에 Batch Normalization 적용

philBaek (백광록) 개발 일기장

9. Weight decay : 0.0001
10. Momentum : 0.9
11. 60 X 10^4 반복 수행
12. dropout 미사용

이후, 테스트 단계에서는 10-cross validation 방식을 적용하고, multiple scale을 적용해 짧은 쪽이 {224, 256, 384, 480, 640} 중 하나가 되도록 resize 한 후, 평균 score을 산출한다.

4. Experiments

4.1 ImageNet Classification

이제 plain net과 ResNet을 대상으로 ImageNet을 이용해 수행한 실험의 결과와 그 특징에 대해 알아본다. 각 모델 구조의 세부적인 내용은 표 1을 확인하면 알 수 있다.

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

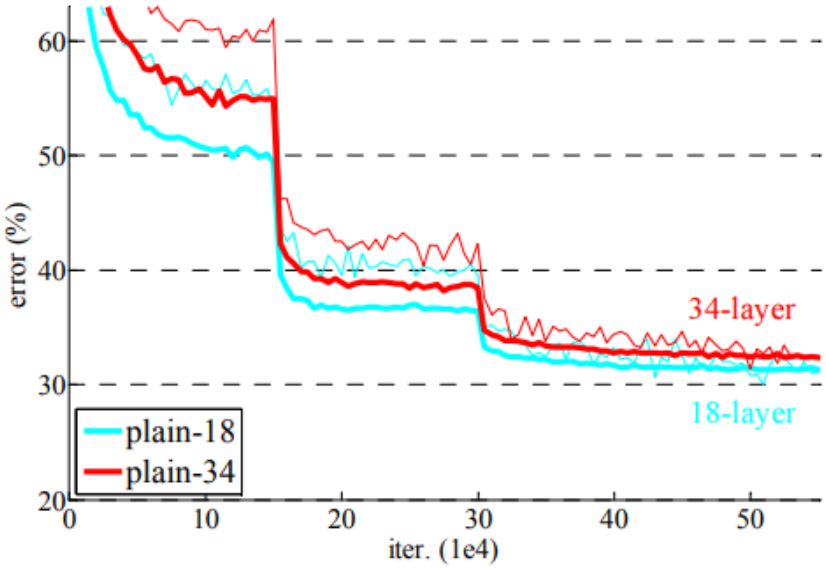
표 1. ImageNet을 대상으로 한 모델 구조

Plain Networks

pare their training/validation errors during the training procedure. We have observed the degradation problem - the 34-layer plain net has higher *training* error throughout the whole training procedure, even though the solution space of the 18-layer plain network is a subspace of that of the 34-layer one.

We argue that this optimization difficulty is *unlikely* to be caused by vanishing gradients. These plain networks are trained with BN [16], which ensures forward propagated signals to have non-zero variances. We also verify that the backward propagated gradients exhibit healthy norms with BN. So neither forward nor backward signals vanish. In fact, the 34-layer plain net is still able to achieve competitive accuracy (Table 3), suggesting that the solver works to some extent. We conjecture that the deep plain nets may have exponentially low convergence rates, which impact the reducing of the training error[3]. The reason for such optimization difficulties will be studied in the future.

먼저, plain 모델에 대해 실험을 수행하였는데, 18 layer의 얕은 plain 모델에 비해 34 layer의 **더 깊은 plain 모델에서 높은 Validation error가 나타났다고 한다.** training / validation error 모두를 비교한 결과, 34 layer plain 모델에서 training error도 높았기 때문에 **degradation 문제**가 있다고 판단하였다.



plain 모델에서는 depth가 깊을수록 train error도 높아진다.

정확도는 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 Networks

Residual Networks. Next we evaluate 18-layer and 34-layer residual nets (*ResNets*). The baseline architectures are the same as the above plain nets, expect that a shortcut connection is added to each pair of 3×3 filters as in Fig. 3 (right). In the first comparison (Table 2 and Fig. 4 right), we use identity mapping for all shortcuts and zero-padding for increasing dimensions (option A). So they have *no extra parameter* compared to the plain counterparts.

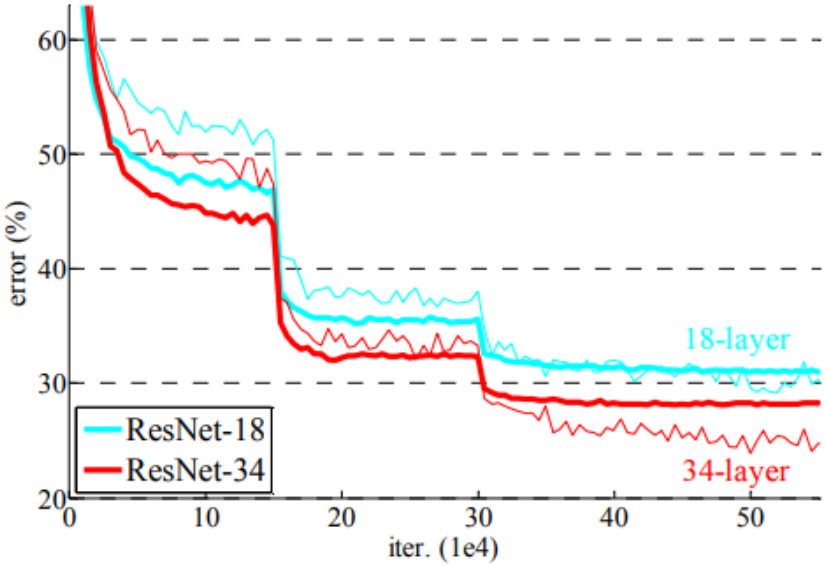
We have three major observations from Table 2 and Fig. 4. First, the situation is reversed with residual learning – the 34-layer ResNet is better than the 18-layer ResNet (by 2.8%). More importantly, the 34-layer ResNet exhibits considerably lower training error and is generalizable to the validation data. This indicates that the degradation problem is well addressed in this setting and we manage to obtain accuracy gains from increased depth.

Second, compared to its plain counterpart, the 34-layer ResNet reduces the top-1 error by 3.5% (Table 2), resulting from the successfully reduced training error (Fig. 4 right *vs.* left). This comparison verifies the effectiveness of residual learning on extremely deep systems.

Last, we also note that the 18-layer plain/residual nets are comparably accurate (Table 2), but the 18-layer ResNet converges faster (Fig. 4 right *vs.* left). When the net is "not overly deep" (18 layers here), the current SGD solver is still able to find good solutions to the plain net. In this case, the ResNet eases the optimization by providing faster convergence at the early stage.

다음으로, 18 layer 및 34 layer ResNet을 plain 모델과 비교해보았다. 이때, 모든 Shortcut은 identity mapping을 사용하고, 차원을 키우기 위해 zero padding을 사용하였기에 파라미터 수는 증가하지 않았다.
실험 결과, 다음 3가지를 확인할 수 있었다.

다.



ResNet에서는 depth가 깊어지면 오히려 train error가 더 줄어들고 있다.

2. 34-layer ResNet의 top-1 error는 3.5%가량 줄었고, 이는 **residual learning이 extremely deep system에서 매우 효과적**임을 알 수 있다.

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

plain 모델과 ResNet 모델의 깊이에 따른 Top-1 error

3. 18-layer ResNet과 plain net을 비교했을 때 성능이 거의 유사했지만, 18-layer의 ResNet이 더 빨리 수렴하였다. 즉, 모델이 과도하게 깊지 않은 경우 (18-layer), 현재의 SGD Solver는 여전히 plain net에서도 좋은 solution을 찾을 수 있지만, **ResNet은 같은 상황에서 더 빨리 수렴**할 수 있다.

Identidy vs. Projection Shortcuts

Identity *vs.* Projection Shortcuts. We have shown that

for increasing dimensions, and all shortcuts are parameter-free (the same as Table 2 and Fig. 4 right); (B) projection shortcuts are used for increasing dimensions, and other shortcuts are identity; and (C) all shortcuts are projections.

Table 3 shows that all three options are considerably better than the plain counterpart. B is slightly better than A. We argue that this is because the zero-padded dimensions in A indeed have no residual learning. C is marginally better than B, and we attribute this to the extra parameters introduced by many (thirteen) projection shortcuts. But the small differences among A/B/C indicate that projection shortcuts are not essential for addressing the degradation problem. So we do not use option C in the rest of this paper, to reduce memory/time complexity and model sizes. Identity shortcuts are particularly important for not increasing the complexity of the bottleneck architectures that are introduced below.

앞서 parameter-free 한 identity shortcut이 학습에 도움된다는 것을 알 수 있었는데, 이번에는 projection shortcut에 대해 알아본다. 다음 3가지 옵션에 대해 비교하였다.

A) **zero-padding shortcut**을 사용한 경우. (dimension matching시에 사용) 이때, 모든 shortcut은 parameter-free 하다.
B) **projection shortcut**을 사용한 경우. (dimension을 키워줄 때에만 사용) 다른 모든 shortcut은 identity 하다.
C) **모든 shortcut으로 projection shortcut**을 사용한 경우.

이때, 3가지 옵션 모두 plain model보다 좋은 성능을 보였고, 그 순위는 A < B < C 순이였다. 먼저 A < B는 zero-padded 차원이 residual learning을 수행하지 않기 때문이고, B < C는 projection shortcut에 의해 파라미터가 추가되었기 때문이다.

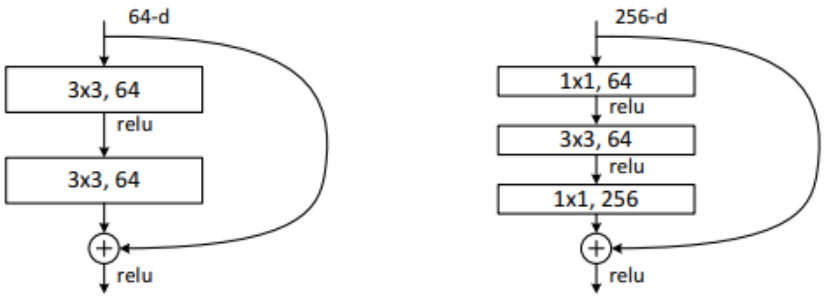
3가지 옵션의 성능차가 미미했기에 projection shortcut이 degradation 문제를 해결하는데 **필수적이지는 않다**는 것을 확인할 수 있었다. 따라서 **memory / time complexity와 model size를 줄이기 위해 이 논문에서는 C 옵션을 사용하지 않는다. 특히 Identity shortcut은 bottleneck 구조의 복잡성을 높이지 않는 데에 매우 중요하기 때문**이다.

Deeper Bottleneck Architectures

as a *bottleneck* design[4]. For each residual function $\mathcal{F}$, we use a stack of 3 layers instead of 2 (Fig. 5). The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions. Fig. 5 shows an example, where both designs have similar time complexity.

The parameter-free identity shortcuts are particularly important for the bottleneck architectures. If the identity shortcut in Fig. 5 (right) is replaced with projection, one can show that the time complexity and model size are doubled, as the shortcut is connected to the two high-dimensional ends. So identity shortcuts lead to more efficient models for the bottleneck designs.

ImageNet에 대하여 학습을 진행할 때 training time이 매우 길어질 것 같아 **building block을 bottleneck design으로 수정**하였다고 한다. 따라서 각각의 residual function인 F는 3-layer stack 구조로 바뀌었는데, 이는 **1 x 1, 3 x 3, 1 x 1 conv로 구성**된다. 이때 1 x 1은 dimension을 줄이거나 늘리는 데 사용되어, 3 x 3 layer의 input / output 차원을 줄인 bottleneck 구조를 만들어준다.



기존 ResNet building block과 bottleneck design이 적용된 building block

여기서 parameter-free 한 identity shortcut은 bottleneck 구조에서 특히 중요하다. 만약 identity shortcut이 projection shortcut으로 대체되면, shortcut이 2개의 high-dimensional 출력과 연결되어 time complexity와 model size가 2배로 늘어난다. (위 그림에서 64-d가 256-d로 늘어난 건 identity shortcut을 유지하기 위해 zero-padding을 통해 차원을 늘려준 것으로 생각된다) 따라서 **identity shortcut은 bottleneck design을 더 효율적인 모델로 만들어준다.**

50-layer ResNet

philBaek (백광록) 개발 일기장

a 50-layer ResNet (Table 1). We use option B for increasing dimensions. This model has 3.8 billion FLOPs.

34-layer ResNet의 2-layer block을 3-layer bottleneck block으로 대체하여 50-layer ResNet을 구성하였다. 이 때, dimension matching을 위해 B 옵션을 사용하였다.

101-layer and 152-layer ResNets

101-layer and 152-layer ResNets: We construct 101-layer and 152-layer ResNets by using more 3-layer blocks (Table 1). Remarkably, although the depth is significantly increased, the 152-layer ResNet (11.3 billion FLOPs) still has *lower complexity* than VGG-16/19 nets (15.3/19.6 billion FLOPs).

The 50/101/152-layer ResNets are more accurate than the 34-layer ones by considerable margins (Table 3 and 4). We do not observe the degradation problem and thus enjoy significant accuracy gains from considerably increased depth. The benefits of depth are witnessed for all evaluation metrics (Table 3 and 4).

더 많은 3-layer block을 사용하여 101-layer 및 152-layer ResNet을 구성하였다. depth가 상당히 증가하였음에도 VGG-16 / 19 모델보다 더 낮은 복잡성을 가졌으며, degradation 문제없이 상당히 높은 정확도를 보였다.

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**

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	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**

10-crop testing을 적용했을 때의 Error rate와 validation set에 대한 single-model Error rate

Comparisons with State-of-the-art Methods

we compare with the previous best single-model results. Our baseline 34-layer ResNets have achieved very competitive accuracy. Our 152-layer ResNet has a single-model top-5 validation error of 4.49%. This single-model result outperforms all previous ensemble results (Table 5). We combine six models of different depth to form an ensemble (only with two 152-layer ones at the time of submitting). This leads to **3.57%** top-5 error on the test set (Table 5). *This entry won the 1st place in ILSVRC 2015.*

ResNet의 single 모델 (앙상블 적용 안 한 모델) 은 앙상블이 적용된 이전의 다른 모델을 능가하였고, 앙상블을 적용했을 경우, 무려 top-5 error 3.57%를 달성할 수 있었다.

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

앙상블 기법이 적용되었을 때의 Error rate

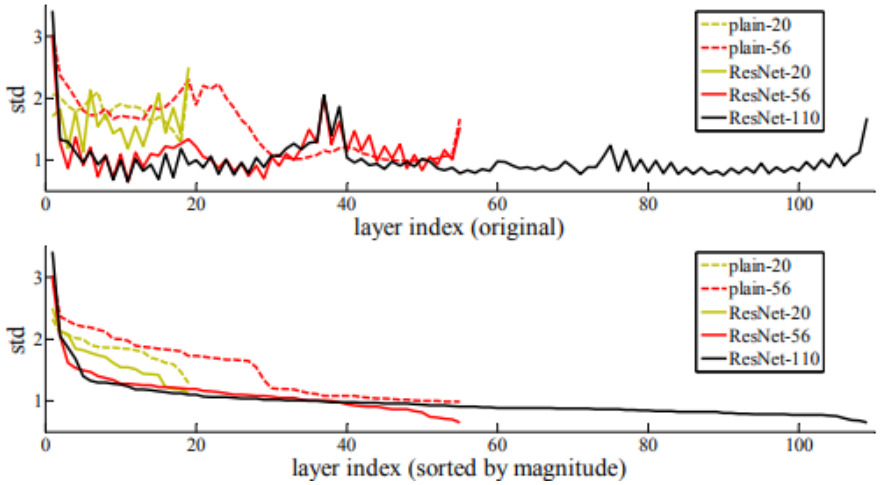
4.2 CIFAR-10 and Analysis

여기서는 ImageNet 말고도 CIFAR-10을 대상으로 모델을 학습 및 검증하였을 때의 결과에 대해서 다루고 있다. 이때까지 위에서 다룬 내용과 거의 비슷하기에 간략하게만 짚고 넘어가겠다.

Analysis of Layer Responses

nonlinearity (ReLU/addition). For ResNets, this analysis reveals the response strength of the residual functions. Fig. 7 shows that ResNets have generally smaller responses than their plain counterparts. These results support our basic motivation (Sec.3.1) that the residual functions might be generally closer to zero than the non-residual functions. We also notice that the deeper ResNet has smaller magnitudes of responses, as evidenced by the comparisons among ResNet-20, 56, and 110 in Fig. 7. When there are more layers, an individual layer of ResNets tends to modify the signal less.

ResNet의 response가 plain net보다 상대적으로 많이 낮게 나왔는데, 이는 **residual function이 non-residual function보다 일반적으로 0에 가까울 것이라는 주장을 뒷받침**해준다. (depth가 깊어짐에 따라 response가 작아지는 것은 각 layer가 학습 시 signal이 변화하는 정도가 작다는 것을 의미한다)

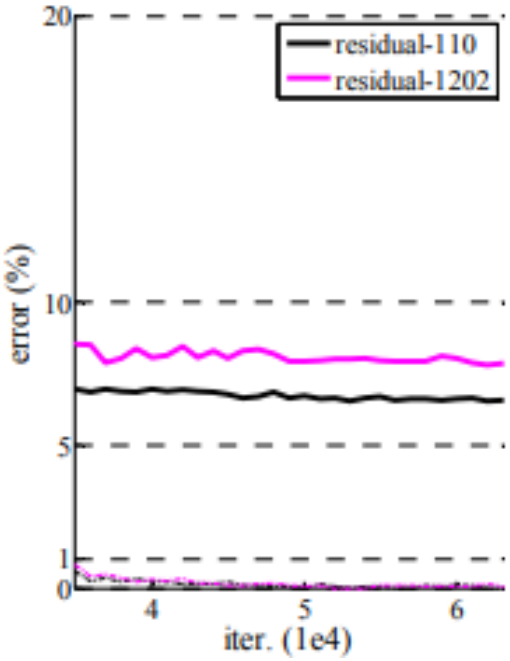


CIFAR-10을 대상으로 한 layer response의 표준편차

Exploring Over 1000 layers

But there are still open problems on such aggressively deep models. The testing result of this 1202-layer network is worse than that of our 110-layer network, although both have similar training error. We argue that this is because of overfitting. The 1202-layer network may be unnecessarily

philBaek (백광록) 개발 일기장



110-layer ResNet과 1000-layer ResNet의 error rate

참고 문헌

1. arxiv.org/abs/1512.03385
2. lv99.tistory.com/25
3. 89douner.tistory.com/64?category=873854
4. leechamin.tistory.com/184
5. sike6054.github.io/blog/paper/first-post/
6. m.blog.naver.com/siniphia/221387516366