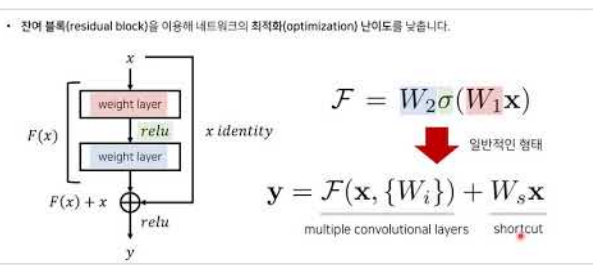
DNN_02_5. CNN_ResNet

생성일	@2022년 6월 22일 오후 1:48
유형	머신러닝/딥러닝
작성자	동훈 오

ResNet: Deep Residual Learning for Image Recognition (꼼꼼한 딥러닝 논문 리뷰와 코드 실습)

오늘은 ResNet으로 알려진 논문에 대한 리뷰 영상을 준비했습니다. ResNet은 나온 지 5년이 되었으나 최근까지도 많은 논문에서 베이스라인 아키텍처로 비교되고 있을 정도의 좋은 논문입니다. 2020년 기준 인용 수는 약 60,000회입니다.논문 핵심 요약: 00:00:00논문...

https://www.youtube.com/watch?v=671BsKl8d0E&list=PLRx0vPvlEmdADpce8aoBhNnDaaHQN1Typ&index=5



Deep Residual Learning for Image Recognition

(Microsoft Research)

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.

1. Introduction

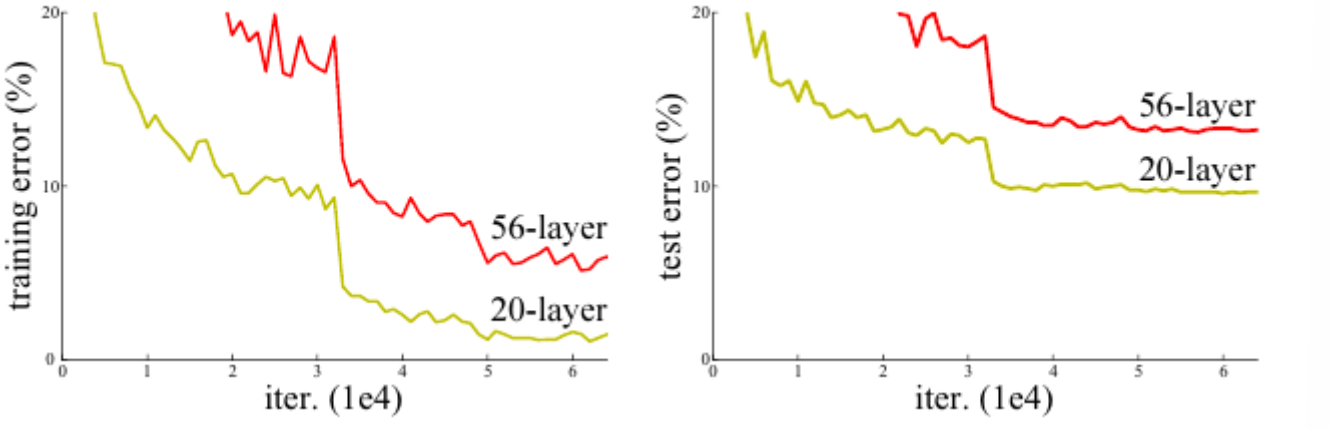
Convolutional Neural Network 발전 과정에 있어서 맞닥뜨린 중요한 질문은, layer를 많이 쌓을 수록 네트워크가 학습이 더 잘되고, 성능이 좋아지는가? 이다.

→ 그렇지 않다, Degradation 문제가 항상 있었다. layer 를 많이 쌓아서 train 점수도 좋고, 오버피팅도 일어나지 않았는데 성능이 잘 나오지 않는 현상이 지속적으로 나타나고 있었다.

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].

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.

[숫자] 는 논문에서 reference 로 넣은 다른 자료(논문)들 이다.



논문의 저자들은 한 가지 가설을 세웠다. H(x) 라는 최적화된 네트워크가 있다면, H(x) 를 직접 구해서 optimize 하는 것은 어려울 것이다. 만약 비선형 층이 쌓여져서 fit 되어진 어떠한 mapping F(x) := H(x) - x 가 설정된다면, H(x) = F(x) + x 로 recast 되고 이것을 최적화 시키는 것이 더 쉽지 않을까 판단했다.

In this paper, we address the degradation problem by introducing a *deep residual learning* framework. Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. Formally, denoting the desired underlying mapping as $\mathcal{H}(\mathbf{x})$, we let the stacked nonlinear layers fit another mapping of $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original mapping is recast into $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.

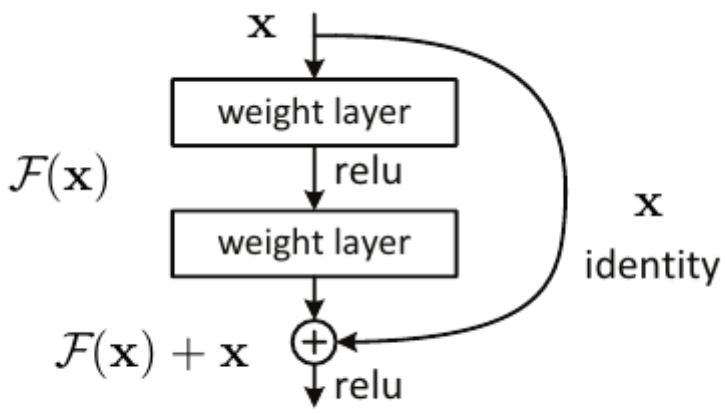


Figure 2. Residual learning: a building block.

- **x : 해당 레이어들의 input**
- H(x) : (전체가 아닌) 소규모의 다층 레이어(a few stacked layers) 의 output
- id(x) : identity mapping 은 단순히 id(x) = x 으로서, x 값을 받으면 동일한 x 를 리턴시킨다.

수식 “F(x) + x” 는 “shortcut connection” 을 가진 feedfoward neural network로서 이해할 수 있다. 논문 저자들의 경우, shortcut connection 은 단순히 identity mapping 을 수행하는 것이고, 그 결과(=x) 를 다층 레이어의 출력에 더하는 것이다.

논문의 저자들은 degration 문제를 시연하고, ResNet 을 평가하기 위해 ImageNet 을 이용한 종합적인 실험을 시도했다. 그 결과로, 다음의 두 가지를 논문에서 보인다.

1. **Our extremely deep residual nets are easy to optimize, but the counterpart “plain” nets (that simply stacked layers) exhibit higher training error when the depth increases.**

2. **Our deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than previous networks.**

그리고 저자들은 ImageNet 데이터셋 뿐만 아니라 CIFAR-10 데이터셋에서도 여러 실험을 수행했다. ImageNet 과 다른 데이터셋에서 residual net 이 최적화하는데 어려움이 있을 것이라 추측했지만, 100개가 조금 넘는 레이어를 가진 모델이 성공적으로 훈련되었고, 1000개가 넘는 레이어를 가진 일부 모델들도 탐구했다.

ImageNet 데이터셋 실험 중 152개의 레이어를 가진 residual net 은 논문이 나온 당시까지 ImageNet을 사용한 모델 중 가장 깊은 모델을 이었고, VGG net 보다 복잡성이 덜 했다.

---

2. Related Work

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-

3. Deep Residual Learning

3.1. Residual Learning

Let us consider $\mathcal{H}(\mathbf{x})$ as an underlying mapping to be fit by a few stacked layers (not necessarily the entire net), with $\mathbf{x}$ denoting the inputs to the first of these layers. If one hypothesizes that multiple nonlinear layers can asymptotically approximate complicated functions[2], then it is equivalent to hypothesize that they can asymptotically approximate the residual functions, *i.e.*, $\mathcal{H}(\mathbf{x}) - \mathbf{x}$ (assuming that the input and output are of the same dimensions). So rather than expect stacked layers to approximate $\mathcal{H}(\mathbf{x})$, we explicitly let these layers approximate a residual function $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original function thus becomes $\mathcal{F}(\mathbf{x})+\mathbf{x}$. Although both forms should be able to asymptotically approximate the desired functions (as hypothesized), the ease of learning might be different.

This reformulation is motivated by the counterintuitive phenomena about the degradation problem (Fig. 1, left). As we discussed in the introduction, if the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counterpart. The degradation problem suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. With the residual learning reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings.

3.2. Identity Mapping by Shortcuts

논문에서는 모델의 기본 구조 - building block 을 다음과 같이 정립했다.

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

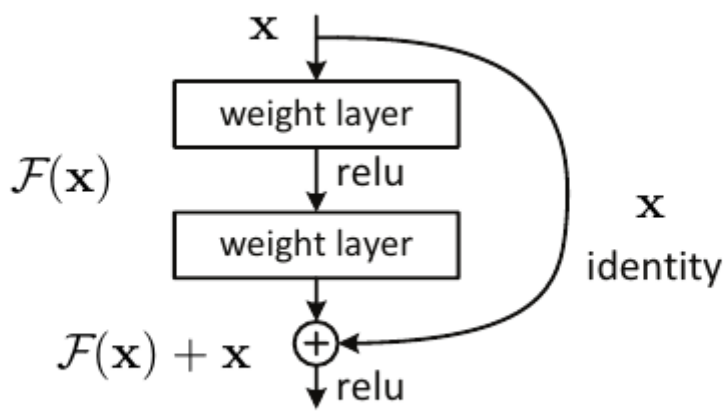


Figure 2. Residual learning: a building block.

논문에서 제시된 모델은 2개의 레이어로 구성되어 있다. 따라서 F function 은 다음과 같이 정리된다.

$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$

W1, W2 는 각 레이어에서의 weights 이고, σ 는 논문에선 ReLU 활성화 함수이다.

Eqn (1) 에서 보여지는 building block 에 대한 수식를 살펴보자. 우선 x 라는 shortcut connections 이 단순히 더해진 형태인 것을 확인할 수 있다. 이러한 방식은 파라미터가 추가되지 않아서 계산 복잡도가 늘어나지 않는 매우 큰 장점을 가지고 있다. 물론, x 와 레이어의 출력물 F 값의 차원의 크기는 같아야 한다. 만약 덧셈을 위해 두 값의 크기가 같지 않다면 x 에 projection matrix 를 곱해서 F 와 더하는 형태로 연산을 수행할 수 있다. 식으로 나타내면 다음과 같다.

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

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.

The form of the residual function $\mathcal{F}$ is flexible. Experiments in this paper involve a function $\mathcal{F}$ that has two or three layers (Fig. 5), while more layers are possible. But if $\mathcal{F}$ has only a single layer, Eqn.(1) is similar to a linear layer: $\mathbf{y} = W_1\mathbf{x} + \mathbf{x}$, for which we have not observed advantages.

We also note that although the above notations are about fully-connected layers for simplicity, they are applicable to convolutional layers. The function $\mathcal{F}(\mathbf{x}, \{W_i\})$ can represent multiple convolutional layers. The element-wise addition is performed on two feature maps, channel by channel.

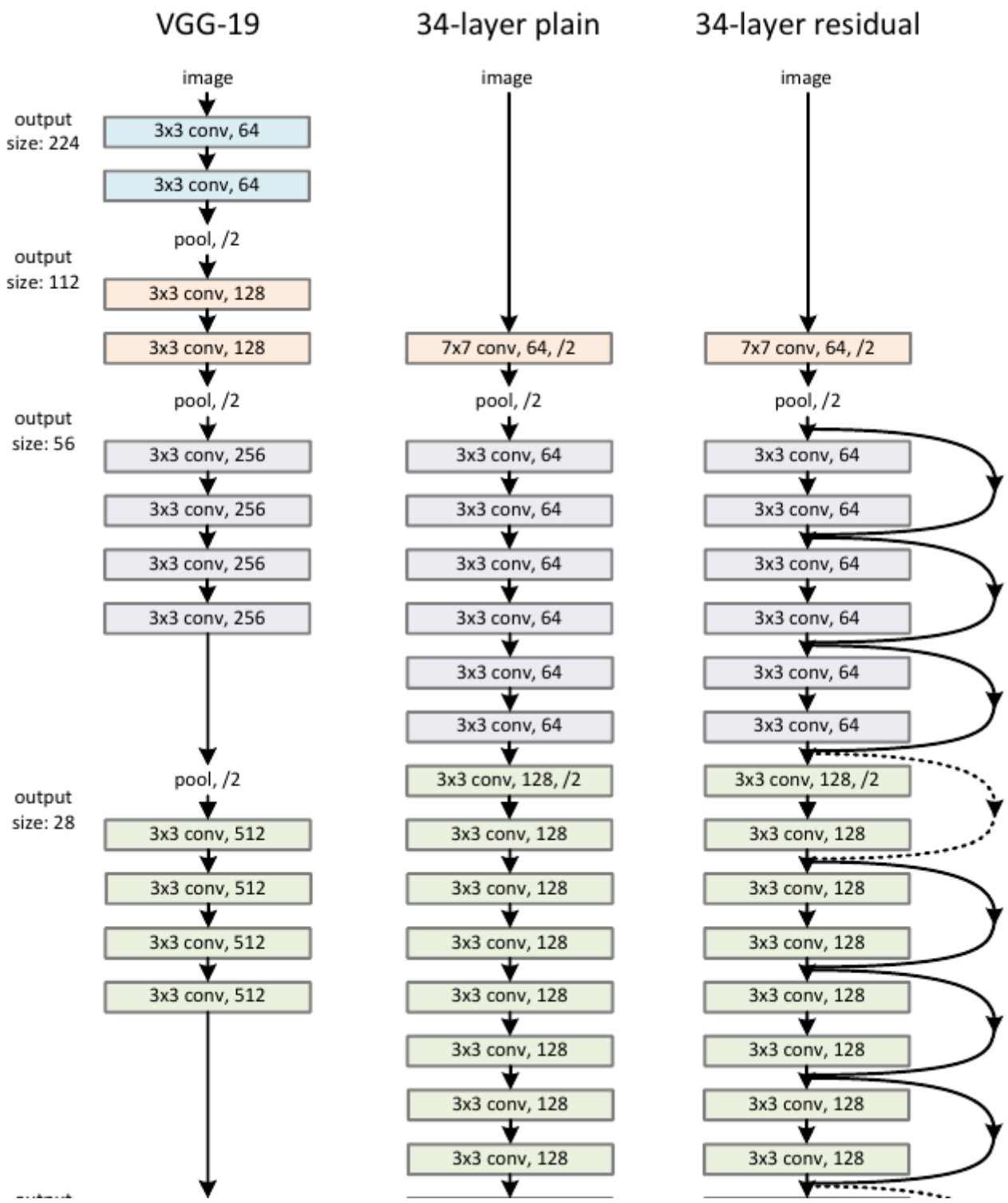
3.3. Network Architectures

Plain Network. Our plain baselines (Fig. 3, middle) are mainly inspired by the philosophy of VGG nets [41] (Fig. 3, left). The convolutional layers mostly have 3×3 filters and 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).

It is worth noticing that our model has *fewer* filters and *lower* complexity than VGG nets [41] (Fig. 3, left). Our 34-layer baseline has 3.6 billion FLOPs (multiply-adds), which is only 18% of VGG-19 (19.6 billion FLOPs).

Residual Network. Based on the above plain network, we insert shortcut connections (Fig. 3, right) which turn the network into its counterpart residual version. The identity 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.

다음 그림을 통해서 아키텍처의 일부를 나타낼 수 있다.



3.4. Implementation

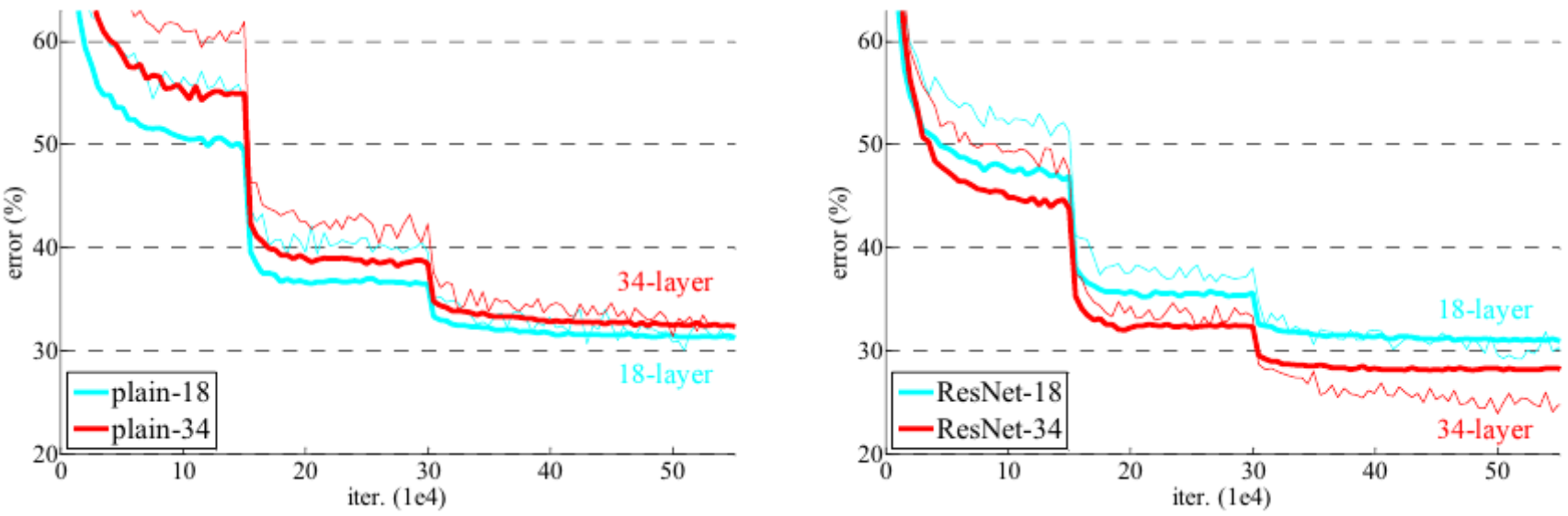
- Data augmentation 기법을 사용해서 ImageNet 데이터셋의 샘플들을 변형 시켰다.
- 배치 정규화 기법을 사용했다.
- 256의 미니배치 사이즈로 SGD를 optimizer 로 사용했다.
- learning rate 는 0.1부터 시작했고 error 가 고착화되는 시점에 lr 을 10으로 나눈 값을 사용했다.
- 0.0001 만큼의 weight decay 를 사용했으며 모멘텀은 0.9를 주었다.

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 $\{224, 256, 384, 480, 640\}$).

4. Experiments

4.1. ImageNet Classification



plain model 에 비해서 residual model 은 degradation 문제를 어느정도 해결한 결과를 보인다.

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.

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**

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

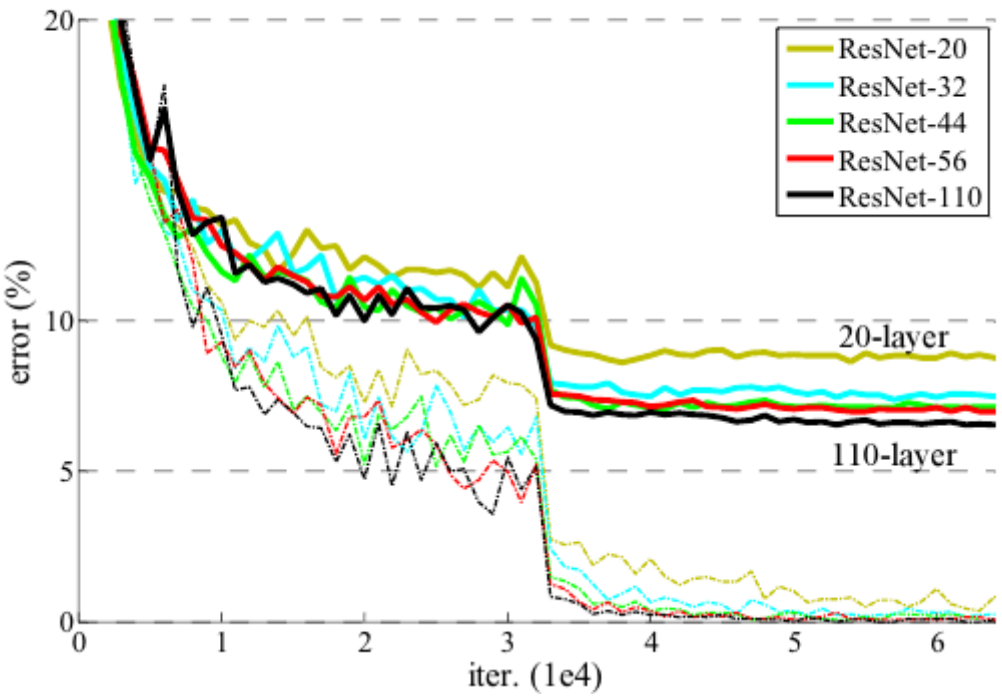
method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

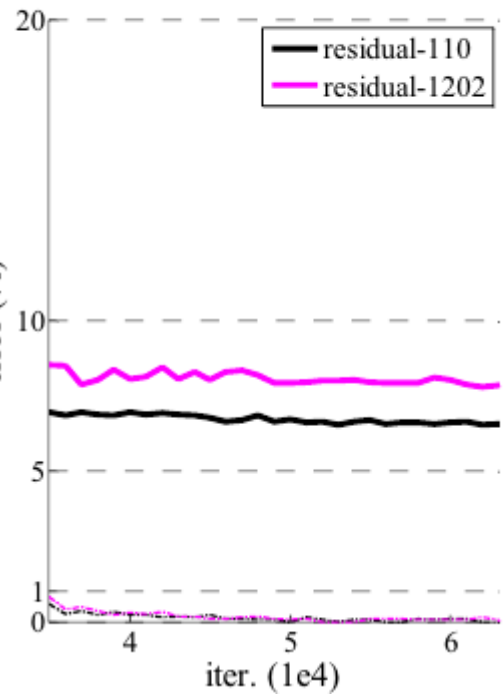
Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

Comparisons with State-of-the-art Methods. In Table 4 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.*

4.2. CIFAR-10 and Analysis

We conducted more studies on the CIFAR-10 dataset [20], which consists of 50k training images and 10k testing images in 10 classes. We present experiments trained on the training set and evaluated on the test set. Our focus is on the behaviors of extremely deep networks, but not on pushing the state-of-the-art results, so we intentionally use simple architectures as follows.





Exploring Over 1000 layers. We explore an aggressively deep model of over 1000 layers. We set $n = 200$ that leads to a 1202-layer network, which is trained as described above. Our method shows *no optimization difficulty*, and this 10^3-layer network is able to achieve *training error* <0.1% (Fig. 6, right). Its test error is still fairly good (7.93%, Table 6).

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 large (19.4M) for this small dataset. Strong regularization such as maxout [10] or dropout [14] is applied to obtain the best results ([10, 25, 24, 35]) on this dataset. In this paper, we use no maxout/dropout and just simply impose regularization via deep and thin architectures by design, without distracting from the focus on the difficulties of optimization. But combining with stronger regularization may improve results, which we will study in the future.