[Abstract]

신경망의 깊이가 깊어질수록 학습시키기에 어려움이 발생했다.

따라서, 이 어려움을 해소시키고자 본 논문에서는 **Residual learning(잔차 학습)** 방식을 도입하였다.

본 논문에서 ImageNet 실험에서 152개의 층을 최대로 하는 ResNet과 VGG를 비교하였을 때,

complexity 즉, 연산량이 ResNet이 훨씬 낮다는 것을 확인했다.

또, COCO 데이터셋에 대해서도 실험한 결과

28% 정도의 상대적인 object detection(객체 탐지) 성능이 향상되었음을 확인하였다.

[Introduction]

최근 CNN(Convolutional Neural Network)이 이미지 분류와 객체 인식에서 뛰어난 성능을 보여왔으며,

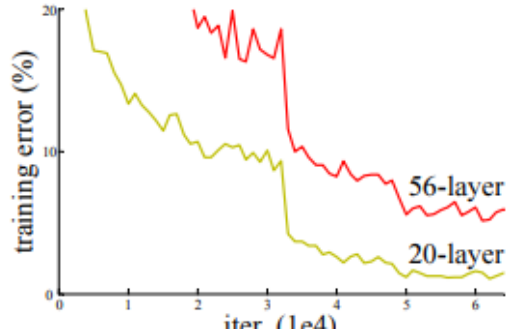
네트워크 깊이가 깊어질수록 성능이 향상된다는 연구 결과가 많았다.

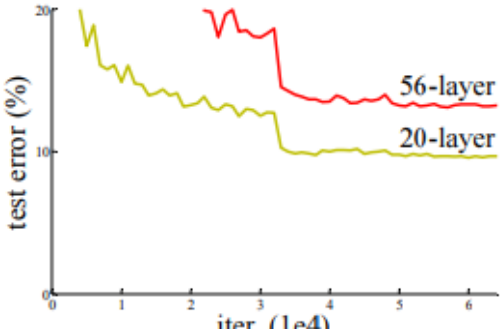
하지만 단순히 층을 많이 쌓는 것만으로는 성능이 좋아지지 않는다는 것을 확인할 수 있었다.

아래의 두 그래프에서 확인할 수 있듯이, 20층 짜리의 네트워크와 56층 짜리의 네트워크를 비교하였을 때,

training 과정에서 발생하는 error가 더 깊이가 깊은 56층 짜리의 네트워크가 더 크다는 것을 알 수 있었다.

즉, 56층 네트워크에서 성능 저하 문제가 발생한 것이다.





좌측: Training error / 우측: Test error

신경망의 깊이를 늘리면 일반적으로 성능이 향상될 것으로 기대되지만,

실제로는 더 깊은 네트워크가 오히려 성능이 저하되는 문제가 발생한다.

그러나 여전히 네트워크의 깊이가 깊어질수록 **degradation problem(성능 저하 문제)**가 발생하는데,

이는 단순한 **과적합(overfitting)** 문제가 아니라, 깊은 네트워크 자체가 학습 자체를 제대로 수행하지 못하는 문제이다.

즉, 네트워크가 깊어질수록 최적화 과정이 어려워지고, 학습 데이터에서조차 높은 훈련 오류(training error)를 보이게 된다.

이를 해결하기 위해 논문에서 **Residual Learning(잔차 학습)**을 제안하였다.

< **Residual learning(잔차 학습)** >

일반적인 네트워크에서는 각 층이 직접 원하는 함수를 학습하려 한다.

이때, **원하는 출력과 입력 간의 직접적인 관계를 학습하는 것이 어렵다.**

이를 해결하기 위해, 논문에서는 **잔차(residual)** 개념을 학습하는 방식을 제안한다.

- 원하는 매핑 함수: H(x)
- 대신 학습할 함수(잔차 함수): F(x)=H(x)−x
- 따라서, 네트워크는 원래 함수 H(x)H(x)H(x) 를 직접 학습하는 대신, H(x)=F(x)+x 형태로 변환하여 **잔차 함수 F(x)만 학습하면 된다.**

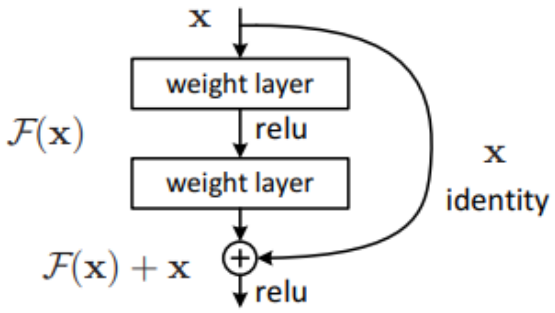


Figure 2. Residual learning: a building block.

신경망이 직접 함수를 학습하는 방법 대신에, 기존 입력과의 차이(잔차)를 학습하도록 유도하였다.

새로운 잔차 학습을 통해 만약 **최적의 매핑 함수가 단순한 항등 함수(Identity Mapping)** 라면, 일반 네트워크에서는 이를 여러 개의 비선형 층이 복잡한 방식으로 학습해야 한다. 반면, ResNet에서는 최적의 항등 함수를 **잔차를 0으로 만드는 것** 만으로 쉽게 구현할 수 있다. 즉, 네트워크는 불필요한 계산을 줄이고, **잔차만 조정하여 최적화 과정 자체가 훨씬 쉬워진다.**

[Related Work]

ResNet의 핵심 아이디어인 **Residual Learning(잔차 학습)**과 **Shortcut Connection(단축 연결)**은 이전에 비슷한 아이디어가 다양한 연구에서 사용되었으며, 본 논문에서는 이러한 관련 연구들을 정리하고 차이점을 설명하고 있다.

1. **Residual Learning(잔차 학습)**

이미지 인식에서 다양하게 활용되었다.

VLAD (Vector of Locally Aggregated Descriptors) 의 경우, 이미지에서 로컬 피처를 추출한 후, 사전 정의된 사전(dictionary)과의 차이를 벡터 형태로 표현하는 방법이다.

ResNet과 개념적으로 유사하지만, ResNet은 신경망 학습 과정에서 직접 잔차를 학습한다는 차이가 있다.

Fisher Vector 의 경우, VLAD의 확장 버전으로서, 확률 모델을 이용해 이미지 피처를 잔차 형태로 변환하는 방법이다.

Residual Vector Quantization (RVQ) 의 경우, 벡터 양자화(quantization) 과정에서 원본 벡터와 기준 벡터(dictionary) 간의 차이를 잔차 벡터로 표현하여 보다 효과적인 데이터 압축을 수행하는 방법이다.

이러한 기법들은 이미지 피처 표현에서 잔차를 활용하는 방법이지만, ResNet은 **신경망이 학습하는 과정에서 직접 잔차를 최적화하는 점에서 차별화**된다.

---

2. **Shortcut Connection(단축 연결)**

역시나 기존에 여러 연구에서 시도되었다.

MLP에서 입력을 출력으로 바로 연결하는 방법의 경우, 초창기 다층 퍼셉트론(MLP) 연구에서는 네트워크 학습을 돕기 위 해 입력을 출력층과 직접 연결하는 기법이 사용되었다. 하지만 당시 연구는 신경망 전체를 shortcut 하는 것이었으며,

ResNet처럼 층(layer) 단위의 잔차를 학습하는 구조는 아니었다.

Deeply Supervised Networks (DSN)의 경우, 네트워크의 중간 층에 보조 분류기 (auxiliary classifier)를 추가하여 기울기 소실(gradient vanishing)을 완화하는 방법이다. ResNet의 shortcut과 유사한 개념이지만, ResNet은 보조 분류기 없이 모 든 층을 직접 연결하는 방식으로 최적화한다는 차이가 있다.

Inception 네트워크의 Branch 구조의 경우, 구글이 발표한 GoogLeNet(Inception)에서 여러 크기의 필터를 병렬로 사 용 하고, 일부 경로를 shortcut처럼 구성하는 방법을 사용한다. 하지만 ResNet처럼 명확한 잔차 학습을 기반으로 한 구조는 아니었다.

---

3. Highway Networks

ResNet과 가장 유사한 개념이다.

Highway Networks의 경우, Gating Mechanism(게이트 메커니즘) 을 사용하여 정보의 흐름을 조절하는 방식이다. ResNet은 게이트 없이 항등 단축 연결(identity shortcut) 을 활용하여 구조를 단순화하고 학습을 쉽게 만든다.

Highway Networks는 100개 이상의 층에서는 성능 개선 정도에 별 차이가 없었지만, ResNet은 1000개 이상의 층에서도 정상적으로 학습된다는 것을 확인하였다.

결론적으로, ResNet은 기존의 연구들의 개념을 발전시켜 학습이 더 쉬운 형태로 네트워크 구조를 제안한 것이 가장 중요한 점이다.

[Deep Residual Learning]

.1. Residual Learning

.2. Identity Mapping by Shortcuts

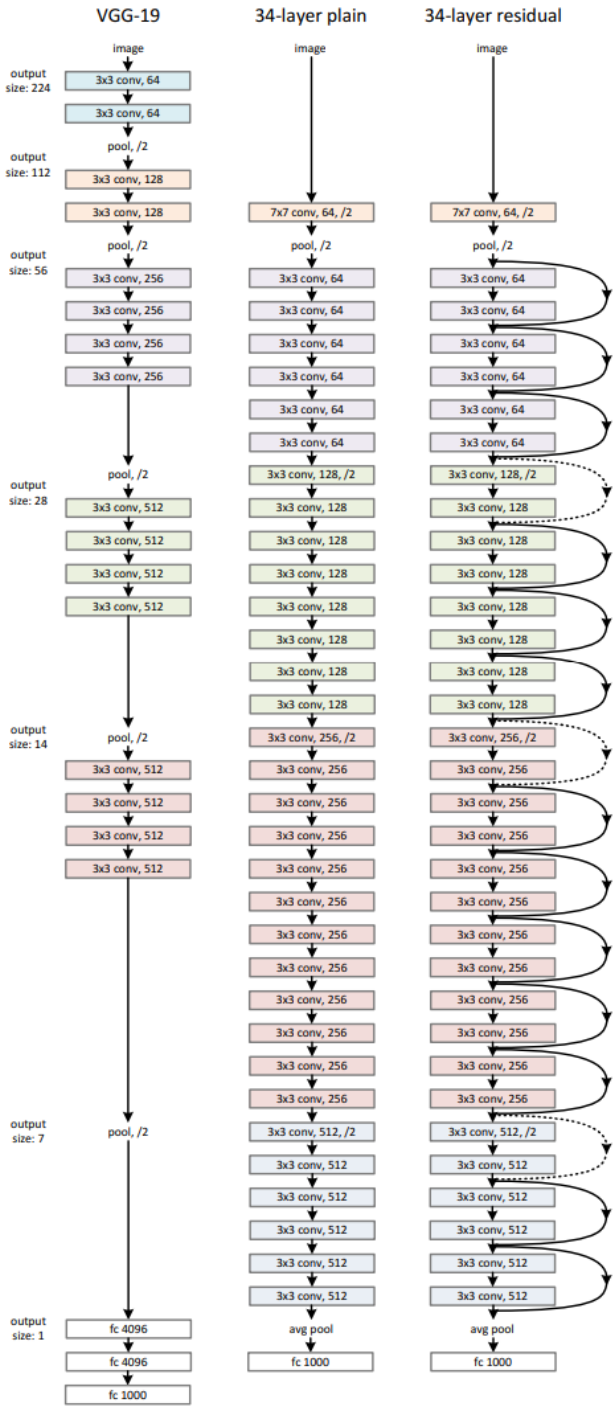
잔차 학습을 위해 **shortcut connection(단축 연결)** 을 도입했다. 층을 건너뛰는 연결을 사용한다.

기본적으로는 항등 함수(identity mapping)를 수행한다. 즉, 입력값을 그대로 다음 층에 전달하며, 여기에 학습된 잔차 함수 $F(x)$가 추가된다.

$$y = F(x) + x$$

Shortcut Connection 방법은 추가적인 매개변수를 필요로 하지 않으며, 네트워크의 깊이가 깊어져도 성능 저하가 발생하지 않게 한다.

.3. Network Architectires



1. Plain Network

VGG 네트워크의 철학을 기반으로 설계되었다.

< 네트워크 설계 규칙>

- 동일한 출력 feature map 크기를 가지면, 레이어는 동일한 수의 필터를 사용한다.

- feature map의 크기가 절반으로 줄어들면, 필터 수는 두 배로 증가하여 레이어별 시간 복잡도를 유지한다.

Plain Network의 경우 VGG 네트워크보다 필터 수가 적고 복잡도가 낮다.

예를 들어, 34-layer 모델은 3.6억 개의 FLOPs를 가진다. 이는 VGG-19의 19.6억 개의 FLOPs의 18%에 불과한다.

2. Residual Network

Plain Network에 **Shortcut Connection**을 추가한 모델이다.

차원이 동일한 경우 그대로 사용하고, 차원이 달라질 때는 다음 두 가지 방식 중에서 하나를 선택한다.

- 방법1 : **Shortcut Connection**이 동일한 방식으로 작동한다

  차원 증가에 따른 Zero Padding 과정을 추가한다.

  추가적인 파라미터를 도입은 없다.

- 방법2 : 1×1 convolution을 사용하여 차원을 맞추는 Projection Shortcut을 사용한다.

Residual Network의 경우 **Shortcut Connection** 과정을 통해 더 깊은 네트워크를 가능하게 하고, 모델 성능을 향상시킨다.

.4. Implementation

해당 부분은 ImageNet 데이터셋에 대한 모델 구현 방법을 설명한다.

가장 먼저, 이미지 전처리 및 증강 과정에서는

이미지의 짧은 쪽을 [256, 480] 범위 내에서 무작위로 샘플링하여 scale augmentation을 진행한다.

224×224 크기의 이미지를 무작위로 크롭하거나 수평 반전을 수행하고, 각 픽셀의 평균값을 뺀 후에 표준 색상 증강 방법 을 사용한다.

두 번째로, 배치 정규화 및 정규화 과정에서는

각 합성곱 레이어 후, 활성화 함수 전에 배치 정규화BN를 적용한다. 그 후, 가중치는 기존 방법에 따라 초기화하고, SGD 즉, 확률적 경사 하강법를 이용하여 학습을 진행한다.

학습률은 0.1로 시작하고, 에러가 정체되면 학습률을 10배씩 나눈다. 가중치 감소(weight decay) 값은 0.0001, 모멘텀은 0.9로 설정하고, dropout은 사용하지 않는다.

마지막으로 테스트 과정의 경우, 표준 10-crop 방법을 사용하고, 다중 스케일에서 이미지 크기 조정을 통해 결과를 평균화하여 최적의 성능을 이끌어내는 과정을 진행한다.

[Experiments]

.1. ImageNet Classification

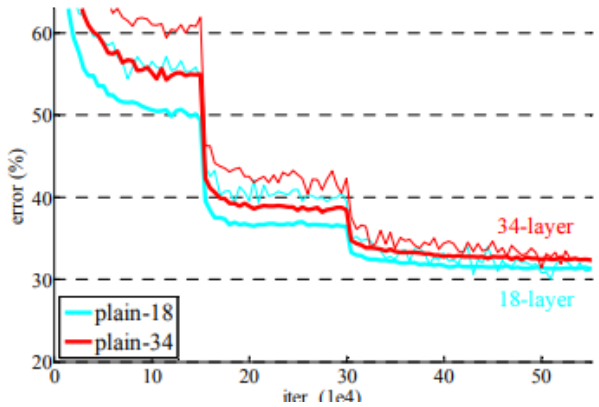
ResNet의 성능을 검증하기 위해 ImageNet 2012 분류 데이터셋 을 사용했다.

<데이터셋 특징>

- 총 1000개의 클래스(class) 를 포함하는 대규모 이미지 데이터셋
- Training set : 128만 개
- Validation set : 5만 개
- Test set : 10만 개
- 평가 기준: top-1 : 모델이 가장 확률이 높은 클래스로 예측한 값이 정답과 일치하지 않는 비율 top-5 : 모델이 확률이 높은 상위 5개 클래스 중 하나라도 정답을 포함하지 않을 경우 오류로 간주하는 비율

본 논문은 ImageNet 데이터셋에 ResNet을 적용하여 성능을 검증했다.

최대 152개 층을 쌓아도 성능이 계속 향상되었으며, 다음과 같은 결과를 얻었다.



34층 Plain Network가 18층보다 Training Error도 높고, Validation Error도 높은 성능 저하 문제가 발생하였다.

이 문제의 원인을 알아보고자 한다.

원인1) Vanishing Gradient (기울기 소실)

네트워크의 깊이가 깊어지면 기울기가 점점 작아지는 Vanishing Gradient가 발생할 수 있다.

하지만 본 실험에서는 Batch Normalization(BN) 을 사용하여, 신호가 네트워크를 통과할 때 항상 적절한 분산값을 유지하도록 설계하였다.

또한, 역전파(Backpropagation) 과정에서 기울기 값(Gradient)이 충분히 유지되는 것을 확인하였다.

따라서, Vanishing Gradient가 원인이 아님을 알 수가 있었다.

원인2) Convergence rate (수렴 문제)

깊이가 깊은 네트워크일수록 최적의 가중치값을 찾기가 어려워지고, 학습이 매우 느려지는 현상이 발생한다.

이 때문에, 34층 Plain Network는 학습이 제대로 진행되지 않고 성능이 떨어지는 문제가 발생한다.

이를 통해, 원인이 네트워크의 깊이가 깊어지면서 최적화 난이도가 증가되기 때문으로 보인다.

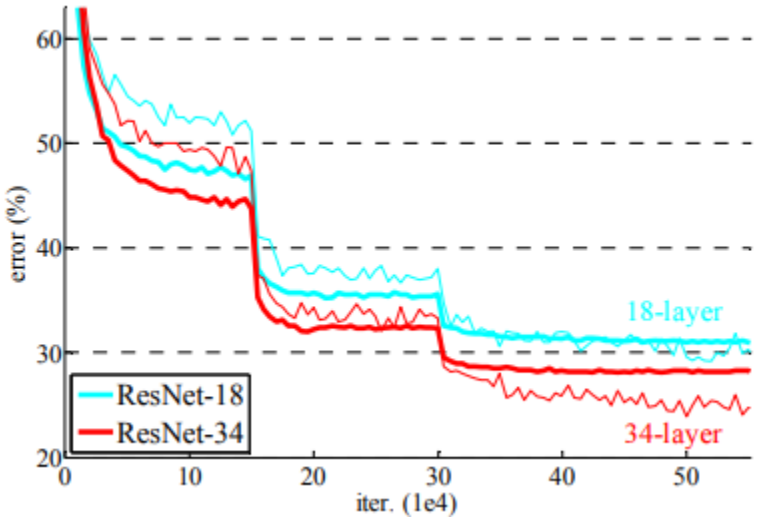
이 문제를 해결하기 위해, Residual Learning(잔차 학습)과 Shortcut Connection(단축 연결) 을 적용한 ResNet 을 도입하여 성능을 비교 실험한다.

---

2. Residual Networks

18-layer와 34-layer ResNet을 설계하여 기존의 Plain Network와 비교하였다.

Baseline 아키텍처는 기존 Plain Network와 동일 하지만, 각 3×3 필터 블록마다 **Shortcut Connection(단축 연결)**을 추가 하여 ResNet을 구성하였다.



1번 이미지

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

2번 이미지

모든 단축 연결에는 항등 매핑(identity mapping) 을 사용하며, 입력 차원이 증가할 때는 Zero Padding(0 채우기) 방법 을 적용하였다. 따로 추가적인 파라미터 없이 기존의 네트워크 구조와 동일한 조건에서 비교를 수행하였다.

위의 1번 이미지와 2번 이미지의 실험 결과를 비교해보았다.

먼저, 깊이가 깊어질수록 34-layer ResNet이 성능이 더 좋다는 것을 확인할 수 있었다.

34-layer ResNet의 성능이 18-layer ResNet보다 오류율(top-1 Error)이 2.8% 더 낮다는

것을 확인하였다.

즉, Residual Learning을 사용하면 네트워크가 깊어질수록 성능이 향상된다는 사실을 알 수 있었다.

또한, 34-layer ResNet은 Training Error가 훨씬 낮고, 검증 데이터에서도 Generalization가 잘되었다.

이를 통해 성능 저하 문제를 해결할 수 있었다.

뿐만 아니라, 기존의 34-layer Plain Network은 학습이 제대로 되지 않아 성능이 낮았었는데, ResNet을 사용하면 성능이 크게 향상된다는 사실도 확인할 수 있었다.

2번 이미지를 참고해보면 top-1 Error가 3.5% 감소하였고, 1번 이미지에서 확인할 수 있듯이, Training Error가 크게 감소하여 최적화가 훨씬 잘 이루어졌음을 확인할 수 있었다.

즉, ResNet의 잔차 학습(Residual Learning)이 깊은 네트워크에서도 효과적으로 학습을 도와준다는 사실을 알 수 있었다.

3. Identity vs Projection Shortcuts

ResNet에서 Shortcut Connection을 구현하는 방법을 비교하였다.

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**

이미지 3

이미지 3에서

A는 Zero-padding을 사용하여 차원이 증가할 때 0을 추가하고, 나머지는 Identity Shortcut를 유지한 모델로 추가 파라미터가 없는 경우

B는 Projection Shortcut 사용하여 필요할 때만 1×1 컨볼루션 적용하고, 추가 파라미터가 조금씩 있는 경우

C는 모든 Shortcut을 1×1 컨볼루션으로 변환한 경우를 의미한다.

비교 결과 세 가지 방법 모두 기존의 Plain Network보다 성능이 우수하다는 것을 확인할 수 있었다.

B의 경우 A보다 약간 더 나은 성능을 보인다는 것을 확인할 수 있었다. 그 이유는 A에서는 차원이 증가할 때 Zero-padding을 추가하는데, 이때 Zero-padding 부분에는 학습

되는 정보가 없으므로 정보의 일부에 손실이 발생하기 때문이 다.

C의 경우 B보다 성능이 약간 더 좋지만, 추가적인 1×1 컨볼루션 연산이 많아지면서 모델 크기와 연산량이 크게 증가하였다.

종합적으로 보았을 때, Projection Shortcut을 일부 사용하는 B 방식이 가장 효율적이며, 모든 Shortcut을 Projection으 로 변환하는 C 방식은 성능은 B보다 좋으나, 메모리와 연산량의 증가로 인해 비효율이라는 것을 알 수 있었다.

---

4. Deeper Bottleneck Architectures

이 부분에서는 더 깊은 ResNet 모델을 설계하는 과정을 다룬다.

특히, 신경망이 깊어질수록 연산량이 급격히 증가하고, 학습 시간이 길어진다는 문제점이 생긴다.

기존의 34-layer ResNet 구조를 그대로 50-layer, 101-layer, 152-layer로 확장하면 학습 시간이 너무 길어져 비효율적인 구조가 만들어지게 된다.

이를 해결하기 위해, Residual Block을 "Bottleneck Block"으로 변경 하여 연산량을 최적화하고자 한다.

<Bottleneck 구조>

Residual Block을 3개의 층으로 변경 하여 연산량을 줄이는 방법이다.

다음과 같은 과정으로 연산량을 줄인다.

1×1 컨볼루션의 경우, 64에서 16으로 차원을 감소시킨다.

3×3 컨볼루션에서 연산을 수행하여 핵심 정보를 학습한다.

다시 1×1 컨볼루션으로 16에서 64로 차원을 복원시킨다.

이를 통해, 연산량을 줄이면서도 성능을 유지 할 수 있다.

50- layer ResNet은 기존의 34-layer 모델을 위에서 제시한 BottleNeck 구조로 변경하여 나오는 구조이다.

입력 차원이 증가할 때, projection 과정을 일부 사용하는 B를 사용한다.

더 깊은 네트워크를 만들기 위해 Bottleneck Block의 개수를 증가시켜, ResNet-101과 ResNet-152를 만들었다.

기존의 VGG-16의 연산량과 비교해보았을 때, 기존의 VGG-16의 경우 15.3billion, ResNet-152 경우 11.2 billion으로 연산 량이 훨씬 적다는 것을 확인할 수 있었.

즉, VGG보다 더 깊지만, 연산량은 오히려 적은 효율적인 모델이라는 것이다.

---

5. Comparisons with State-of-the-art Methods

기존의 최신 모델들과 ResNet의 성능을 비교해보았다.

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**

Single-model의 error rates

34-layer 모델의 경우 기존의 모델들과 비교했을 때, 가장 경쟁력 있는 정확도를 기록했다.

152-layer 모델의 경우 top-5 error가 4.49%로, 기존의 모든 모델들과 비교해봤을 때, 가장 우수한 성능을 나타낸다는 것을 확인하였다.

뿐만 아니라, 여러 개의 ResNet 모델들을 ensemble하여 새로운 형태의 모델을 만들어내기도 하였다.

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**

Ensemble-model의 error rates

6개의 서로 다른 깊이를 가진 ResNet 모델을 조합하여 앙상블 모델을 만들었다. test data의 error rate을 계산한 결과,

top-5 오류율을 3.57%를 기록하였다. 이 결과는 기존의 모든 앙상블 모델을 능가하는 수치이고, 단일 모델의 성능을 크게 향상시켰다.

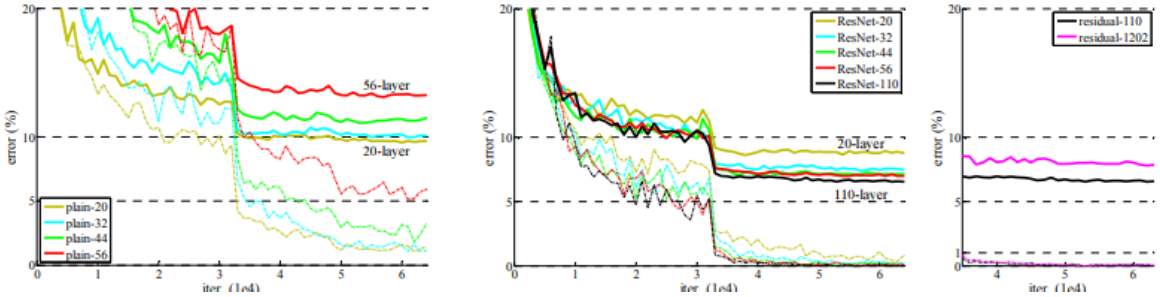
.2. CIFAR-10 and Analysis

CIFAR-10 데이터셋에서도 동일한 실험을 진행했다.

<데이터셋 특징>

- 10개의 클래스(class) 를 포함하는 작은 이미지 데이터셋.
- Training Set : 50,000개 이미지
- Test Set : 10,000개 이미지
- 이미지 크기: 32×32 픽셀

기존의 ImageNet 실험과 비슷하게, 깊이가 깊어질수록 학습이 어려워지는 최적화 문제를 다루고, Residual Learning의 효과를 검증하고자 한다.



왼쪽: 기존의 CNN 모델 / 중간: ResNet 모델 / 오른쪽: 깊이가 깊은 ResNet 모델

네트워크 깊이에 따라 성능의 변화를 분석하였다.

위 이미지의 왼쪽 이미지, 기존의 CNN 모델의 경우 깊어질수록 Training Error가 증가하는 문제가 발생한다.

즉, 네트워크가 깊어질수록 학습이 더 어려워진다는 것을 확인할 수 있다.

이러한 현상은 ImageNet과 MNIST에서도 동일하게 나타나는 최적화 문제이며, 이를 통해 깊은 네트워크는 기본적으로 학습 최적화가 어려운 근본적인 문제를 가지고 있다는 것을 알 수가 있었다.

ResNet을 적용하면, 기존 CNN의 학습 오류 증가 문제를 해결할 수 있다. 깊이가 증가할수록 오히려 성능이 향상된다.

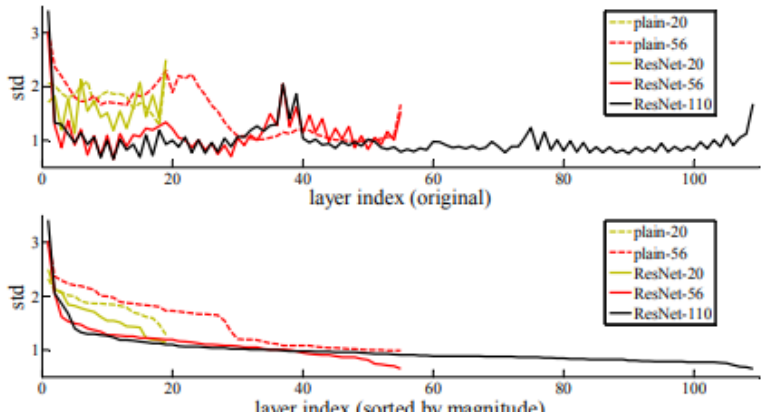
더 깊은 110-layer ResNet도 학습할 수 있는지 확인하기 위해 실험을 추가적으로 수행하였다.

기존과 동일한 학습률 0.1 을 사용했을 때, 학습 초기에 수렴이 잘 되지 않는 문제 발생하여 초기 학습률을 0.01로 낮추어 적용시켜 훈련을 진행한 결과 110-layer ResNet 모델이 안정적으로 학습했다는 사실을 확인할 수 있었다.

1. Analysis of Layer Responses

본 논문에서 각 층의 출력 반응을 분석하여 ResNet과 기존 CNN(Plain Network)의 차이점을 알아보았다.

각 층의 출력값에서 BN 과정을 적용하여 후의 출력 분포를 분석하고 신호가 어떻게 변화하는지 알아보았다.



층의 출력 분포의 표준편차값 비교

위 그래프 이미지에서 ResNet의 표준편차값이 작다는 것을 통해, 각 층에서 출력되는 값들의 변화 폭이 작다는 사실을 알 수 있었다.

또, 깊이가 깊어질수록 각 층별로 반응의 크기가 작아지는 것을 확인할 수 있었다.

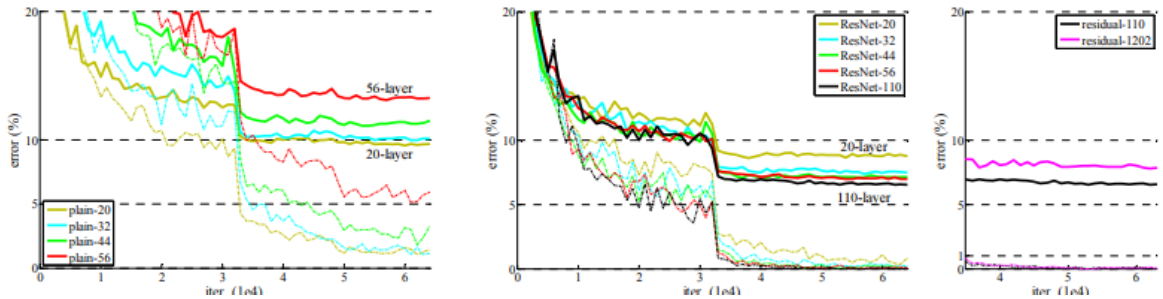
이는 깊이가 깊은 모델일수록 입력값을 크게 변형시키지 않고 작은 변화값들만 조금씩 추가한다는 것을 알 수 있었다. 이러한 과정으로 인해 깊이가 깊어져도 최적화 과정이 크게 어려워지지 않고 안정적인 학습이 가능해진다.

2. Exploring over 1000 layers

ResNet 모델이 극심하게 깊은 네트워크에서도 학습이 가능한지를 알아보고자 하였다.

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

CIFAR-10 test의 분류 error



왼쪽: 기존의 CNN 모델 / 중간: ResNet 모델 / 오른쪽: 깊이가 깊은 ResNet 모델

위의 오른쪽 그래프에서 확인할 수 있듯이, 1202 layer ResNet의 경우 error가 1%보다 작은 값으로, 최적화가 잘 이루어졌다는 것을 확인할 수 있었다.

하지만, 위 표에서 확인할 수 있듯이, ResNet-110 과 ResNEt-1202를 비교했을 때, test error값의 경우 ResNet-110이 더 낮다는 것을 알 수가 있다. 이는 네트워크의 깊이가 지나치게 깊어지면 test data의 경우 error값이 커진다는 것이다.

이를 해결하기 위해서는 좀 더 강력한 정규화 과정을 적용해야 한다.

3. Object Detection on PASCAL and MS COCO

ResNet 모델은 단순히 이미지 분류 뿐만이 아니라 Object Detection과 분할 과정에서도 좋은 성능을 보인다.

먼저, Object Detection의 경우 Faster R-CNN 모델을 PASCAL VOC 2007/2012과 COCO 데이터에 적용하였다.

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**

PASCAL VOC 2007/2012에서의 object detection mAP

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

COCO에서의 object detection mAP

* mAP의 경우 Mean Average Precision, 즉 평균 정밀도를 의미한다.

먼저, COCO 데이터셋의 경우 VGG-16에서 ResNet-101로 모델을 변경하였을 때, 6.0%의 성능 향상을 확인할 수 있었다. 상대적으로는 28%의 성능이 향상되었음을 확인할 수 있다.

이를 통해, object detection 부분에서도 뛰어난 성능을 보임을 확인할 수 있었다.