Residual Learning: ResNet

2021203098
소프트웨어학부
연재혁

About ResNet

(He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

ResNet 등장 배경

딥러닝에서 네트워크 깊이는 이미지 인식 성능 향상에 핵심적인 역할을 한다. VGG, GoogleNet 등의 모델이 ImageNet 대회에서 성공하면서, 더 깊은 네트워크가 더 높은 정확도를 낼 수 있다는 사실이 강조되었다. 깊이는 저,중,고수준의 특징을 풍부하게 표현할 수 있도록 해준다. 그러나 단순히 레이어를 더 쌓는다고 해서 항상 성능 향상이 되는 것은 아니다.

문제 제기

네트워크를 더 깊게 만들면 초기에는 학습이 잘 되지만, 일정 깊이를 넘으면 degradation problem(성능 저하 문제)이 발생한다.

- 깊이가 깊어질수록 학습 정확도(Training Accuracy)와 검증 정확도가 떨어짐
- 단순히 Overfitting 때문이 아니라 Train Error 자체가 증가함

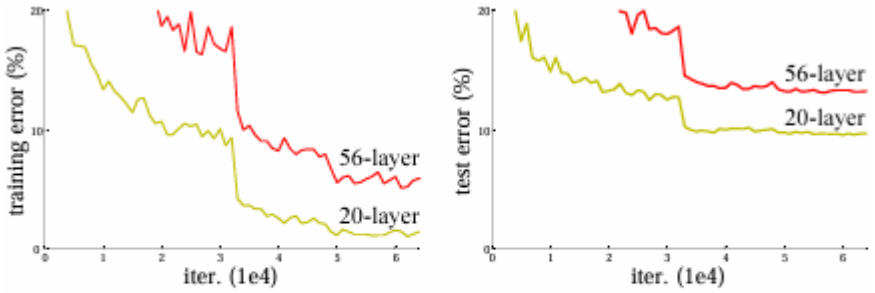


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Fig.1(CIFAR-10 실험): 20-layer 모델보다 56-layer 모델이 학습 및 테스트 에러가 더 높음
- 이는 단순히 "더 많은 레이어가 더 나은 성능을 보장하지 않는다"는 점을 보여줌

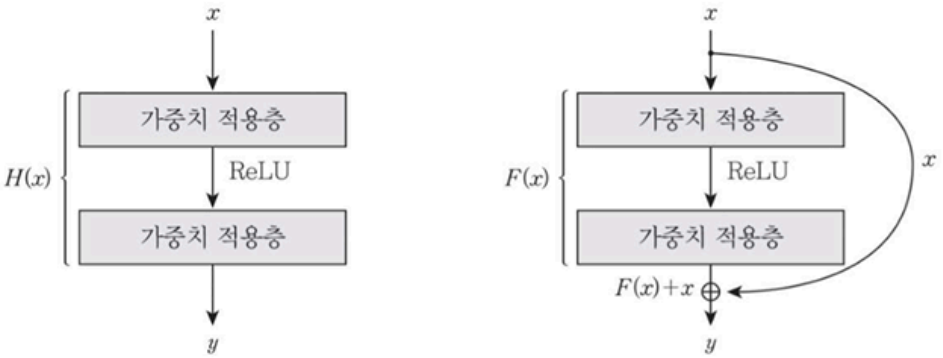
원인 분석

이론적으로, 깊은 네트워크는 얕은 네트워크의 해를 포함할 수 있다. 예를 들어, 추가된 레이어들이 **Identity Mapping**이면 얕은 네트워크와 동일한 출력이 가능하다.
(Identity Mapping : 입력을 그대로 출력으로 전달하는 함수를 의미 H(x) = x)

하지만 현실적으로 기존의 최적화 기법**(SGD + Backprop)**은 이런 해를 찾는 데 어려움이 존재(최적화 난이도 증가)

핵심 아이디어: Residual Learning

기존 방식: 각 스택이 직접 원래의 목표 함수 **H(x)**를 학습
제안 방식: 각 스택이 잔차 함수 **F(x) = H(x) - x**를 학습하고, 최종 출력은 **y = F(x) + x**



$$y = H(x) \qquad F(x) = y - x; \quad y = F(x) + x$$

즉, 네트워크가 학습해야할 목표를 “입력과의 차이(Residual)”로 재정의

구현 방식 : Shortcut Connection

Residual Block : F(x) + x를 구현하기 위해 입력을 출력에 더하는 Skip Connection을 사용한다.
Skip Connection은 추가 파라미터나 계산량을 거의 증가시키지 않는다, 그러므로 기존 구조에 비해 매우 간단히 적용이 가능하다.

과거에도 여러 형태의 Skip Connection이 존재하였다. 예를 들면, MLP에서 입력 -> 출력 직선 연결 시도, Auxiliary classifiers(중간 레이어에서 보조 분류기), Highway Networks 등이 있지만, **ResNet**의 Shortcut은 항상 열려 있다.(파라미터가 없는 단순한 연결)

Deep Residual Learning

Residual Learning

기존 네트워크는 **H(x)**를 직접 근사하려고 한다. ResNet은 **F(x) = H(x) - x**를 학습하고, 최종 출력은 y = F(x) + x 이 된다.

만약 H(x)가 x와 가깝다면, F(x)는 작은 값(H(x)와 x의 차이)만 학습하게 된다.

H(x) ≈ x —> F(x) = H(x) - x ≈ 0

즉, 학습해야 할 값이 0에 가까운 작은 값으로 파라미터 조정 폭이 작아져, 최적화 경로가 짧고 더 매끄러워진다.(최적화 난이도 감소)

Identity Mapping by Shortcuts

ResNet은 여러 개의 층을 묶은 블록마다 Residual Learning 개념을 적용하며, 이 묶음을 Residual Block이라고 한다.
기본 Residual Block:

$$y = F(x, \{W_i\}) + x$$

- x: 블록의 입력 벡터
- y: 블록의 출력 벡터
- $F(x, \{W_i\})$: 학습해야 할 Residual Function(즉, 잔차 함수)
- x는 Shortcut Connection을 통해 그대로 더해짐(Identity Mapping)

Shortcut Connection은 추가 파라미터가 없으며, 연산량 증가 또한 없다. 단순히 입력 x를 출력에 더하는 element-wise addition을 수행한다. 이것이 중요한 이유는 plain Network와 비교 실험 시, 파라미터 수와 계산량을 동일하게 유지할 수 있다. 즉, 성능 향상이 shortcut 덕분임을 명확히 보여줄 수 있다.

입력과 출력 크기가 같으면 Identity Shortcut, 다르면 Projection Shortcut(1 x 1 Conv) 사용

기본적으로 x와 F(x)의 차원이 같아야 더할 수 있다.
만약 차원이 다르다면,

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

- W_s : 1x1 Convolution 같은 Projection Matrix를 사용
- 해당 논문 실험에서는 가능하면 Identity Mapping를 사용, W_s는 차원 맞출 때만 사용

1x1 Convolution Projection Matrix

Projection Matrix는 shortcut connection에서 입력과 출력의 차원이 다를 때 쓰이는 핵심 요소이다.

Residual Block의 기본 수식 : $y = F(x) + x$

여기서 F(x)와 x , 두 텐서를 더하려면 Shape이 동일해야 한다. 하지만 네트워크는 깊어지면서 채널 수가 변화하고(64 -> 128), Feature map의 크기가 변화한다. (ex. 56x56 > 28x28, stride=2) 이런 경우 F(x)와 x의 크기가 달라서 단순 더하기가 불가능해진다.

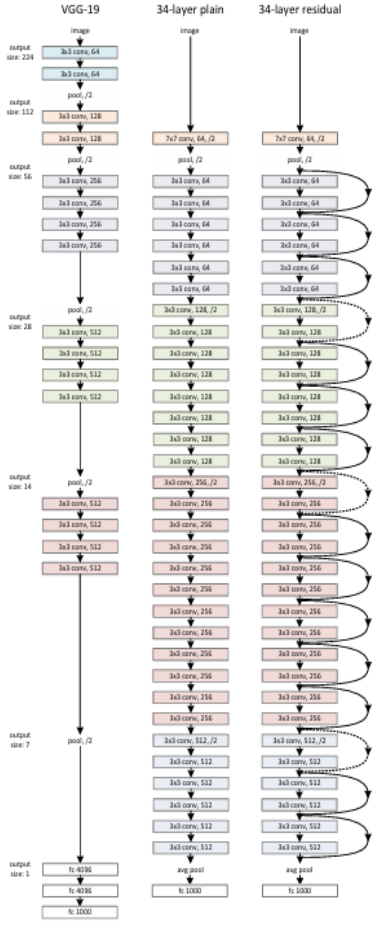
이 문제의 해결 방법이 Projection Matrix이다.

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

- W_s : Projection Matrix(차원 맞춤 역할)
- 구현 방식 : 1x1 Conv을 이용한 Linear Projection

1x1 Conv 특징

- 커널 크기 : 1x1 각 위치의 채널 간 조합만 수행한다.
- 역할 및 장점
    1. 채널 수 조정 : in_channels => out_channels 변환
    2. stride > 1일 때 공간 크기 줄이기
    3. 연산량 적음



ResNet Model Architecture

Plain Network

VGG 철학을 따름, 대부분 3x3 Conv

두가지 규칙 존재

1. 출력 feature map 크기가 같으면 동일한 필터 개수 유지
2. feature map 크기를 절반으로 줄일 대, 채널 수를 2배로 늘려 연산량 균형 유지

- Downsampling : stried=2 Conv 사용
- Global Average Pooling -> FC Layer(1000 way softmax)
- 연산량(FLOPs) : 3.6B
- VGG-19(19.6B)의 18% 수준 (훨씬 효율적)

Residual Network

plain 구조에 shortcut connection 추가)(3.6B FLOPs)

- 같은 크기의 feature map일 때 Identity mapping
- 크기가 다를 때
    - 옵션 A : Zero padding(파라미터 없음)
    - 옵션 B : Projection Shortcut(1x1 conv) > 차원 맞추기
    - 차원 변경 시 stride = 2 적용
    - 결과 : 연산량 증가x, 더 깊은 학습 o

Experiments Result in ImageNet Classification

성능 요약

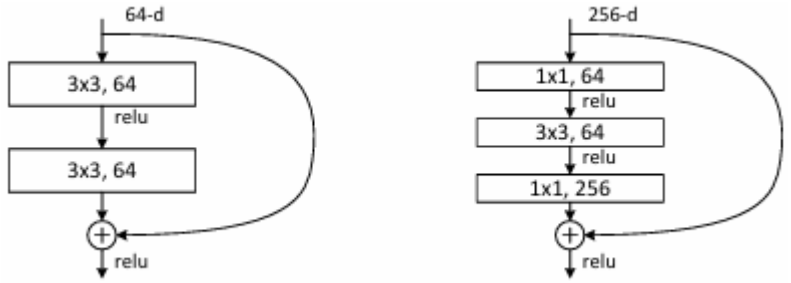
모델	**Top-1 Error**	**Top-5 Error**
VGG-16	28.07%	9.33%
plain-34	28.54%	10.02%
ResNet-34	24.19%	7.40%

ResNet-50	22.85%	6.71%
ResNet-101	21.75%	6.05%
ResNet-152	**21.43%**	**5.71%**

- ResNet은 깊어질수록 정확도가 향상됨을 확인할 수 있다.(50, 101, 152-layer)
- Degradation Problem이 사라져, 매우 깊은 네트워크도 학습이 가능함을 알 수 있다.
- 연산량, ResNet-152: 11.3B FLOPs (VGG-19의 19.6B보다 적음)
- 이유: FC layer 제거 + Bottleneck 설

Identity vs Projection Shortcut
실험 결과
- Identity Shortcut + Zero-padding(Option A) : 충분히 성능 좋음
- Projection Shortcut(1x1 Conv, Option B) : 약간 더 좋음
- 모든 Shortcut을 projection으로 (Option C) : 약간 추가 향상, 하지만 파라미터 증가 큼
- 결론 : Projection은 필수는 아님, 차원 변경 시만 사용

Bottleneck 구조



- 깊은 ResNet(50, 101, 152)에서 연산 효율을 위해 사용
- 1x1 Conv -> 3x3 Conv -> 1x1 Conv
- 채널 축소 -> 3x3 Conv -> 채널 복원
- 장점 : 연산량 줄이고, 파라미터 효율성 증가
- Identity Shortcut 그대로 사용 가능 > 시간/메모리 절약

ResNet에 대한 오해

(x)기존에는 Residual Module 방식이 이미 학습된 특징은 건너뛰고, 새로운 특징만 학습하는 구조라고 생각했음. 그래서 이미 학습된 특징과 새로운 특징을 구별하는 조건문이 필요하다고 생각했음

(O)Residual의 Shortcut 연산은 단순한 덧셈이다. 즉, 출력 값에 입력 x를 그대로 더해주기 때문에, 이전 레이어의 특성이 그대로 보존되는 상태에서 새롭게 학습한 변화만 누적하는 방식이다.