💚

Deep Residual Learning for Image Recognition

기간	@03/05/2025 → 03/11/2025
주차	1주차
논문	https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
상태	완료
예습/복습	예습과제
참고자료	참고 블로그1 참고 블로그2

0. Abstract

- 심층 신경망의 경우 학습이 어렵다는 문제 존재 → 본 논문에서는 이러한 문제 해결을 위해 **잔차 학습(Residual Learning)** 프레임워크를 제안함.
- 기존 층이 직접 매핑을 학습하는 것이 아니라, 입력에 대한 **잔차 함수(residual function)를 학습**하도록 유도함. → 이를 통해 매우 깊은 네트워크도 쉽게 최적화할 수 있으며, 깊이가 증가함에 따라 성능이 향상됨을 입증함.

- ImageNet 데이터셋 : 최대 152층의 잔차 네트워크 학습하여 기존 VGG 네트워크보다 **8배 깊으면서도 연산 복잡도는 낮은** 모델 개발, 앙상블 기법 적용해 오차를 3.75%까지 줄임.
- ILSVRC & COCO 2015 대회에서 ImageNet detection, localization, COCO detection, segmentation에서 1등 달성함.

1. Introduction

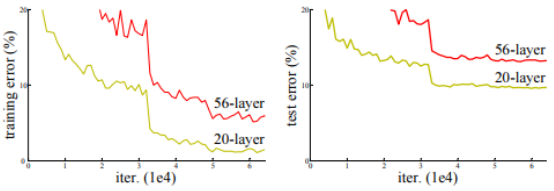
논문이 다루는 분야

- **딥러닝을 이용한 이미지 인식**을 다룸.
- CNN은 저차원, 중간 수준, 고차원 특징을 다층 구조에서 통합하고, 이 "특징 수준(feature level)"은 층을 쌓아 깊이를 증가시킴으로써 더욱 풍부해짐.
- 최근 연구들에서도, 네트워크의 **깊이(depth)**가 중요함이 밝혀짐.
  - ImageNet과 같은 복잡한 데이터셋에서 좋은 성능 내는 최신 모델들 대부분 "매우 깊은(very deep)" 구조를 채택함.

해당 task에서 기존 연구 한계점

- 단순히 층을 더 쌓는 것만으로 더 나은 네트워크 만드는 것이 가능한가? → No.
- 신경망의 깊이가 증가하면 **기울기 소실(Gradient Vanishing) 또는 폭발(Exploding)** 문제가 발생하여 학습이 어려움.
  - 역전파 과정(가중치 업데이트 과정)에서 기울기 값이 매우 작아지거나 커지면서 학습이 원활하게 진행되지 않음.
- 제안된 해결 방법
  1. 정규화된 초기화(Normalized Initialization) 기법
  2. 배치 정규화(Batch Normalization)

  ⇒ 이러한 기술들이 도입되면서 깊은 네트워크도 학습 가능해졌지만, 네트워크의 깊이가 너무 깊어질 경우 정확도가 오히려 빠르게 감소하는 **"퇴화 문제(Degradation Problem)"** 발생함.
- 퇴화 문제(Degradation Problem)



layer 추가할수록 높은 training error, test error 가짐을 볼 수 있음.

  - 네트워크 깊이가 증가함에 따라 Training error와 Test error 모두 증가하는 현상. 일반적으로 네트워크 깊어질수록 표현력 증가하므로, 높은 오류 보일 이유 없어야 함.
  - 그러나 네트워크 깊이가 너무 깊어지면 오히려 학습 어려워지고, 최적의 가중치 찾지 못하는 경우 발생함.
  - 이는 Overfitting으로 인한 것이 아니라, **layer의 수가 추가**되었기 때문.

논문의 contributions

- **잔차 학습(Residual Learning)** ⇒ 깊이가 증가해도 학습이 원활히 이루어지는 네트워크 구조를 제안함.
- 핵심 아이디어
  - 기존 네트워크는 입력 x를 받고 layer를 거쳐 $H(x)$ 출력 → 입력값 x를 출력값 y로 매핑하는 함수 $H(x)$ 얻는 것이 목적.
  - 그러나 우리가 기대하는 출력값을 **입력값과의 차이(잔차, residual)**를 학습하도록 설계하면, 학습이 더욱 쉬워짐. ($H(x)$가 아닌 $H(x)-x$ 얻도록 목표 수정. $F(x)=H(x)-x$)
  - $F(x)$ 최소화시켜야 함. = 출력과 입력의 차를 줄임. = $F(x)$가 0이 되는 것이 최적.

    ⇒ $0=H(x)-x$, $H(x)=x$ ⇒ 목표 값이 사전에 제공되므로 학습 더욱 쉬워짐.

- 즉, 기존 방식에서는 Unreferenced mapping인 H(x)를 학습시켜야 했지만, 잔차 학습에서는 잔차를 0에 가깝게 만드는 방식으로 최적화가 가능함.

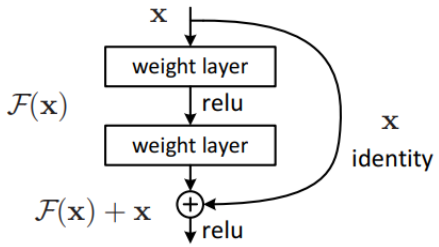


Figure 2. Residual learning: a building block.

- $H(x) = F(x) + x$, 입력에서 출력으로 바로 연결되는 shortcut 만 추가하면 됨.
- 입력과 같은 x가 그대로 출력으로 연결되므로 파라미터 수에 영향이 없고, 덧셈 연산 늘어나는 것 제외하면 shortcut 연결을 통한 연산량 증가 없음.

2. Related Work

1. **잔차 표현(Residual Representations)**
   - 잔차를 활용한 특징 표현 방식은 이미지 인식에서 이미 사용된 바 있음.
     - **VLAD(Vector of Locally Aggregated Descriptors)** 기법 : 특징 벡터를 특정 사전(Dictionary)에 대해 잔차 벡터로 변환하여 인코딩하는 방식.
     - **Fisher Vector** : VLAD의 확률적 버전으로, 이미지 검색과 분류에서 강력한 성능을 보임.
     - **벡터 양자화(Vector Quantization, VQ)** : 원본 벡터(original vector)가 아닌 잔차 벡터(residual vector)를 인코딩하는 것이 더욱 효과적인 것으로 밝혀짐.
       - 벡터 양자화란? 특징 벡터 X를 클래스 벡터 Y로 매핑하는 것을 의미함.
   - 수치해석 및 그래픽스 분야에서도 잔차 표현 활용한 방법 존재함.
     - **Multigrid Method(다중 격자 방법)** : 부분 미분 방정식(PDE) 해결할 때 다중 스케일 문제를 잔차 표현을 통해 해결하는 방법.(시스템을 여러 scale의 하위 문제로 재구성, 각 하위 문제는 더 큰 scale과 더 작은 scale 간의 residual 담당함.)
     - **계층적 기저 함수(Hierarchial Basis Functions)** : 이를 이용한 사전 조건화(Pre-conditioning) 기법도 잔차 벡터 활용하여 최적화 속도 향상시킴.
2. **Shortcut Connections (지름길 연결, 단축 경로)**
   - Shortcut Connection은 오래전부터 신경망 연구에서 다루어진 개념.
     - 초기 **다층 퍼셉트론(MLP)** 모델 : 네트워크 입력을 출력과 직접 연결하는 방식 시도된 바 있음.
     - **Deep Supervision Networks(DSN)** : 중간 계층을 보조 분류기(Auxiliary Classifier)와 직접 연결항여 학습 돕는 방식을 사용함.
     - **Batch Normalization(BN)** : 층의 응답값을 정규화하는 과정에서 Shortcut Connection 활용하는 경우 있었음.
   - **Highway Networks**에서도 Shortcut Connection 개념 사용됨.
     - Highway Networks : 게이트(gating) 기능 추가하여 정보 흐름을 조절할 수 있도록 설계됨. 하지만 Shortcut Connection이 완전히 닫힐 수도 있고, 따라서 모든 정보가 전달되는 것이 보장되지 않음.
     - 반면, **ResNet에서는 항상 정보가 흐를 수 있도록 Identity Mapping 사용하여 학습이 더욱 안정적**임. (100층 이상의 네트워크에서도 성능 향상을 실험적으로 입증함.
       - parameter 전혀 추가되지 않으며, 0으로 수렴하지 않기에 절대 닫힐 일이 없어 항상 모든 정보가 통과됨. → 지속적으로 residual function 학습하는 것이 가능.

3. 제안 방법론 - Deep Residual Learning

3.1. 잔차 학습(Residual Learning)

- $H(x)$를 few stacked layers이 학습해야 하는 목표 함수(매핑 함수)라고 하자. (여기서 x는 해당 층들의 입력을 의미함.)
- 다중 비선형 층이 복잡한 함수를 점진적으로 근사할 수 있다는 가정을 토대로, 원래의 매핑 함수 $H(x)$를 직접 학습하는 것보다, **잔차 함수(Residual Function)** $F(x)$를 학습하는 것이 더 쉽지 않을까? → 이를 기반으로 재구성한 네트워크 학습 방식.

$$F(x) = H(x) - x$$

  - 기존처럼 $H(x)$ 직접 학습하는 것이 아닌, 입력 x 대비 출력 값이 얼마나 달라져야 하는지(잔차 $F(x)$) 학습하는 방식. ⇒ $H(x) = F(x) + x$

✅ 기존 방식보다 쉬울 것이라고 가정하는 이유 (Introduction에서 언급.)

1. Identity Mapping 학습하는 것이 쉬움.
   - 기존에는 여러 층이 항등 함수 직접 학습해야 했다면, 잔차 학습에서는 단순히 $F(x) = 0$ 되도록 하면 매핑 자동으로 학습 가능.
2. 최적화가 쉬워짐.
   - 기존에는 네트워크가 완전히 새로운 함수 $H(x)$ 학습해야 하지만, 잔차 학습의 경우 이미 존재하는 x를 기반으로 변화량(잔차) $F(x)$만 학습하면 됨.

     ⇒ 기존 입력 기반으로 작은 변화만 조정하는 방식이므로 학습 과정이 더 안정적, 최적화 쉬워짐.

- **잔차 학습을 네트워크에 적용하는 방법**
  - Feedforward Neural Network에 적용하기 위해 Shortcut Connection 개념 도입.

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

    - 출력에 입력 값 x 그대로 더하는 구조. (Shortcut Connection)
    - 기존 네트워크에 추가적인 연산 요구하지 않고, SGD(확률적 경사 하강법) 및 역전파(Backpropagation) 알고리즘 그대로 적용 가능.

3.2. Identity Mapping by Shortcuts

- 잔차 학습은 여러 층을 쌓을 때마다 적용할 수 있음. ⇒ **잔차 블록(Residual Block)**
- 기본 구조

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

  - x : 입력 벡터
  - y : 출력 벡터
  - $F(x, \{W_i\})$: 학습할 잔차 함수(즉, 뉴런이 학습하는 함수)
  - W_i : 가중치 파라미터

✅ **여기서 Shortcut Connection 포함** ⇒ 추가적인 파라미터 필요하지 않음.(연산량 거의 증가 x), Identity Mapping 가능함.(불필요한 층 거치지 않고 정보 원활하게 흐를 수 있음.)

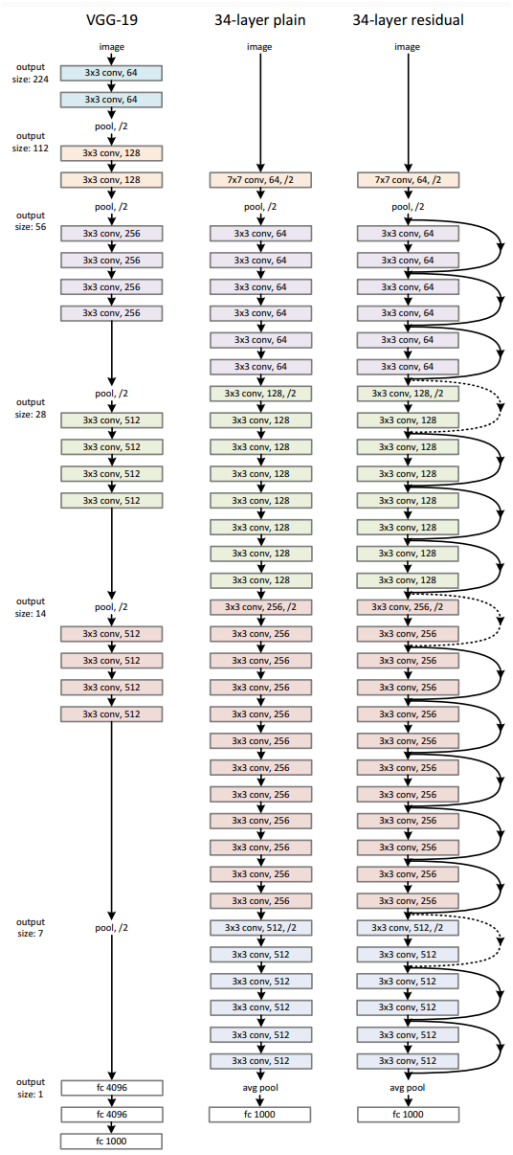
- **입출력 차원이 다를 때의 문제 해결**
  - **프로젝션 매핑(Projection Mapping)**
    - square matrix 사용하여 차원을 변환하는 방식.
    - 수식

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

- W_s : a linear projection ⇒ 입력 차원을 출력 차원에 맞게 조정하는 역할.

3.3. Network Architectures

- **Plain Networks**
  - VGGNet과 유사한 구조를 기반으로 함.
  - 주로 **3 x 3 conv filter**를 사용하여 층을 쌓고, 다음 2가지 규칙에 기반하여 설계함.
    1. Output feature map의 size가 같은 layer들은 모두 같은 수의 conv filter 사용함.
    2. Output feature map의 size가 절반으로 줄어들면 time complexity를 동일하게 유지하기 위해 필터 수를 2배로 늘림.
       - downsampling 수행 → pooling 사용하는 것이 아니라 stride가 2인 conv filter 사용함.
       - 모델 끝부분에 GAP(Global Average Pooling) 사용하고, 사이즈가 1,000인 FC layer와 Softmax 사용함.
  - 결과적으로, 전체 layer 수는 34, 이는 VGGNet 보다 적은 필터와 복잡성을 가짐.
- **Residual Networks, ResNet**
  - Plain 모델에 기반하여 Shortcut Connection 추가하여 구성함.
  - 이때 input과 output 차원이 같다면, Identity shortcut 바로 사용하면 됨.
  - 만약, 차원이 증가했다면 2가지 선택권이 존재함.
    1. **zero padding** 적용
    2. **projection shortcut** 사용(1 x 1 convolution)
       - 이때, shortcut이 feature map을 2 size씩 건너뛰므로 stride를 **2**로 설정함.



실선 : 차원이 동일할 때, 점선 : 차원이 증가할 때

3.4. Implementation (모델 구현)

- 모델 구현 진행
  1. Image는 더 짧은 쪽의 길이로 [256, 480] 사이가 되도록 랜덤하게 resize 됨.
  2. horizontal filp 부분적으로 적용, per-pixel mean을 제거함.
  3. 224 × 224 사이즈로 랜덤하게 crop 수행함.
  4. Standard color augmentation 적용함.
  5. 각 convolution 이후와 activation 전에 Batch Normalization 적용함.
  6. Optimizer : SGD (mini-batch size : 256)
  7. Learning rate : 0.1에서 시작. (학습이 정체될 때 10씩 나눠줌.)
  8. Weight decay : 0.0001
  9. 10. Momentum : 0.9
  10. $60 \text{ x } 10^4$ 반복 수행
  11. dropout 미사용
- Test 단계

- 10-cross validation 방식 적용, multiple scale 적용하여 짧은 쪽이 {224, 256, 384, 480, 640} 중 하나가 되도록 resize → 평균 score 산출함.

4. 실험 및 결과

Dataset

- 사용한 데이터셋
  - **ImageNet 2012**
    - 1000개 클래스
    - **학습 데이터**: 128만 개
    - **검증 데이터**: 50,000개
    - **테스트 데이터**: 100,000개
  - **CIFAR-10**
    - 10개 클래스
    - **학습 데이터 :** 50,000개
    - **테스트 데이터 :** 10,000개
  - **PASCAL VOC 2007 & 2012** 데이터셋 & **MS COCO** 데이터셋
    - 객체 탐지용 데이터

Baseline

- 비교 기준
  - 모델은 학습 데이터로 훈련되고 검증 세트를 이용하여 성능 평가함.
  - 최종적으로 test 서버를 통해 test 세트의 성능도 측정.
  - 모델의 **Top-1 및 Top-5 오류율**을 평가 기준으로 사용함.

✅ Baseline 모델 : 기존 **VGG-16, GoogLeNet, Highway Networks**

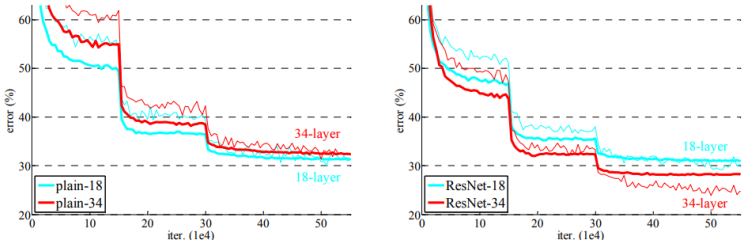
✅ 잔차 네트워크(**ResNet**) : 18층, 34층, 50층, 101층, 152층 실험

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

ImageNet 대상으로 한 모델 구조

결과

- **ImageNet Classification**



- **Plain Networks (18층, 34층)**
  - 34층 모델 구조 - VGGNet과 유사한 형태, 3 x 3 conv filter 사용
  - 결과 : 34층 네트워크가 오히려 18층 네트워크보다 검증 오류 높았음. ⇒ **Degradation 문제** 있다고 판단.

    💡 **시사점**
    - 단순히 네트워크를 깊게 하는 것만으로는 성능 향상되지 않음.
    - 네트워크 깊어질수록 최적화 어려워지는 **Degradation Problem** 발생.

    ⇒ 훈련 과정에서의 오류 변화 그래프 : 단순 Overfitting 문제가 아니라, 최적화 자체가 제대로 이루어지지 않음을 의미.
- **Residual Networks, ResNet (18층, 34층)**
  - 동일한 네트워크에서 잔차 학습(Residual Networks) 프레임 워크 적용.
  - 모든 블록에 Shortcut Connection 추가한 버전(ResNet-18, ResNet-34)
  - 결과 : **34층 ResNet이 18층 ResNet보다 더 낮은 오류율**을 기록.

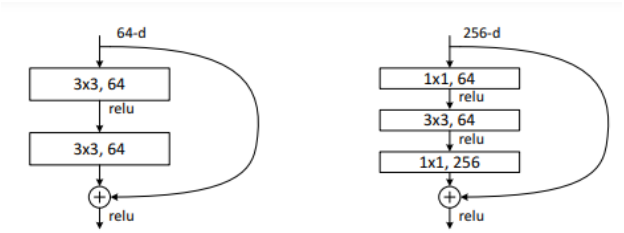
    💡 **시사점**
    - 일반 네트워크에서는 깊이 증가하면 성능 감소하지만, ResNet에서는 **깊이 증가할수록 성능이 향상**됨.
    - parameter-free한 **identity shortcut**이 학습에 도움이 됨을 알 수 있음.
- **Identity vs Projection Shortcuts**
  - 3가지 옵션

    (A) **zero-padding shortcut** 사용한 경우 (dimension matching할 때 사용)

    (B) **projection shortcut** 사용한 경우 (dimension 증가시킬 때 사용)

    (C) **모든 shortcut**으로 **projection shortcut** 사용한 경우
  - 결과 : 3가지 옵션 모두 plain model 보다 좋은 성능을 보임.
    - **A < B < C**
      - A < B : zero-padded 차원이 residual learning 수행하지 않기 때문.
      - B < C : projection shortcut에 의해 파라미터가 추가되었기 때문.
    - 3가지 옵션이 성능 차가 크지 않았기 때문에 projection shortcut이 degradation 문제 해결에 **필수적인 것은 아니다**는 결론.
    - C는 다루지 않고, complexity와 model size에 중요한 identity shortcut만 사용할 것.
      - **Why?** bottleneck 구조의 복잡성을 높이지 않는 데에 매우 중요함.

      ⇒ **Deeper Bottleneck Architectures**



순서대로 기존 ResNet building block과 bottleneck design 적용된 building block(64→256 늘어난 이유 : indentity shortcut 유지하기 위해 zero-padding 통해 차원 늘려준 것.)

- ImageNet 학습 진행 시 학습 시간이 매우 길어질 것 같아 **bottleneck design**으로 수정함.
- 각각의 residual function F는 **3-layer stack** 구조로 바뀜. (1 x 1, 3 × 3, 1 × 1 conv로 구성.)
  - 1 × 1의 경우 **차원을 줄이거나 늘리는 데 사용**되어, 3 x 3 layer의 input/output 차원을 줄인 bottleneck 구조 만들어줌.
- 만약, identity shortcut이 projection shortcut으로 대체되면, shortcut이 2개의 고차원 출력과 연결되어 time complexity와 model size 2배로 늘어남.

- 성능 비교

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**

Error rates(%, 10-crop testing)

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**

Error rates (%) of single-model results

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**

Error rates (%) of ensembles.

✅ **Plain-34**는 **VGG-16보다도 성능이 낮았음.**

✅ **ResNet-34**는 VGG-16보다 **훨씬 좋은 성능을 보였으며, ResNet-50 이상**에서는 **성능이 더욱 향상.**

✅ 특히 **152층 ResNet**이 **Top-5 오류율 5.71%**를 기록하며 **최상의 성능을 달성**.

✅ **앙상블**을 적용했을 경우, 무려 **top-5 error 3.57% 달성**.

- **CIFAR-10 and Analysis**
  - 목표
    - 기존 신경망과 ResNet을 비교하여 **잔차 학습이 CIFAR-10에서도 효과적인지** 확인.
    - **더 깊은 네트워크(110층, 1202층)**를 훈련하여 **깊이가 증가할수록 어떤 영향을 미치는지** 분석.
  - 결과 분석
    - **층이 깊어질수록 성능이 좋아짐. (20층 → 110층까지 오류율 감소)**
    - 하지만 **1202층 모델에서는 오히려 성능이 감소 → 과적합** 가능성이 있음.
    - 즉, 무조건 층을 깊게 하는 것이 아니라, **적절한 깊이에서 최적의 성능을 얻는 것이 중요.**

- **Object Detection on PASCAL and MS COCO**
  - **Faster R-CNN** 모델을 사용하여 객체 탐지(Object Detection) 성능을 평가.
  - 결과
    - VGG-16을 ResNet-101로 교체하자 성능이 **6% 향상됨.**
    - COCO 데이터셋에서 28% 성능 향상을 기록.

5. 결론 (배운점)

5.1. 의의

- 신경망의 깊이 한계를 극복한 최초의 모델
  - 기존 신경망에서 발생하던 **기울기 소실(Gradient Vanishing) 및 퇴화 문제(Degradation Problem)를 해결**, 신경망의 깊이를 혁신적으로 확장한 연구.
- Shortcut Connection 도입 → 최적화 개선
  - ResNet의 **Shortcut Connection** 개념은 이후 다양한 딥러닝 모델에서 활용됨.
  - 기울기 전파를 안정적으로 만들어 깊은 네트워크에서도 학습이 원활하게 진행되도록 함.
- 다양한 딥러닝 분야로 확장 가능
  - 이미지 분류뿐만 아니라, 다양한 컴퓨터 비전 및 딥러닝 분야(객체 탐지, 영상 분할(segmentation), NLP, RL) 등에 큰 영향 미침.

💡
잔차 학습의 개념이 다양한 딥러닝 응용 분야에서 사용되며, "딥러닝의 기본 구조"로 자리 잡음.

5.2. 한계점

- 네트워크 깊이가 지나치게 증가하면 **성능 정체**(Diminishing Returns of Depth)
  - 1000층 이상의 실험에서는 오히려 성능이 저하되는 현상.
  - 너무 깊은 네트워크가 불필요한 매개변수를 많이 가지면서 **과적합(Overfitting)**이 발생할 가능성이 높아짐.
- 연산량이 많고, 실시간 애플리케이션에 적합하지 않음.
  - VGG보다 연산량이 적지만, GoogLeNet에 비해서는 더 많은 연산량을 필요로 함.
  - **MobileNet, ShuffleNet, EfficientNet** 같은 경량 모델들이 등장하면서 실용적인 대안이 개발.
- Self-Attention 기반 모델(ViT, Transformer) 등장 → CNN의 한계가 부각.
  - ResNet과 같은 CNN 모델들은 국소적(local) 특징을 학습하는 데 최적화되어 있지만, **전역적인(global) 특징을 학습하는 능력이 부족**.