Today-I-Learned]/Deep Learning

[논문 리뷰] Deep Residual Learning for Image Recognition - ResNet(1)

2020. 3. 31. 04:46

목차

ResNet

ResNet 이라는 이름으로 더 유명한 논문을 리뷰해보겠습니다. 최고의 빅데이터 분석 동아리 '투빅스' 과제 겸사겸사 하는 리뷰입니다.

(사실 이게 아니라 구현 과제를 해야되는데 어렵네요....)

논문 흐름대로 리뷰되었습니다.

- https://arxiv.org/pdf/1512.03385.pdf
- https://arxiv.org/pdf/1603.05027.pdf

Abstract

딥러닝에서 neural networks가 깊어질수록 성능은 더 좋지만 train이 어렵다는 것은 알려진 사실입니다. 그래서 이 논문에서는 잔차를 이용한 잔차학습 (residual learning framework)를 이용해서 깊은 신경망에서도 training이 쉽게 이뤄질 수 있다는 것을 보이고 방법론을 제시했습니다.

We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. : 함수를 새로 만드는 방법 대신에 residual function, 잔차 함수를 learing에 사용하는 것으로 layer를 재구성합니다.

이 논문은 empirical evidence showing 방법으로 residual을 이용한 optimize를 더 쉽게 하는 법, accuracy를 증가시키고 더 깊게 쌓는 법에 초점을 둡니다. (empirical 적이기 때문에 경험적으로 보여주는, 즉 데이터를 이용한 실험을 통해 증명하는 방식입니다.)

결과적으로 152개의 layer를 쌓아서 기존의 VGG net보다 좋은 성능을 내면서 복잡성은 줄였습니다. 3.57%의 error를 달성해서 ILSVRC 2015에서 1등을 차지했으며, CIFAR-10 데이터를 100개에서 1000개의 레이어로 분석한 것을 논문에 담았습니다.

(COCO, ILSVRC , ... 자랑 생략)

1. Introduction

이 논문이 reference한 많은 논문들로 인해 deep model일수록 성능이 좋다는 것은 알려져 있습니다.(그리고 CNN에 대한 많은 연구 결과가 있음.)

깊어지는 layer와 함께 떠오르는 의문은 ***"과연 더 많은 레이어를 쌓는 것만큼 network의 성능은 좋아질까?"*** 입니다. 왜냐하면 이와 관련해서 악명높은 많은 문제들이 발생하는데

표적으로 ***problem of vanishing/exploding gradients*** 문제 입니다. 그래도 이 문제는 다양한 방법들로 개선되어왔습니다. (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 backpropagation [22].)
이 논문에서 깊게 다룰 문제는 ***Degradation Problem***입니다. network가 깊어질수록 accuracy가 떨어지는 (성능이 저하되는) 문제입니다. 이 문제는 overfitting의 문제가 아니기 때문에 주목받습니다. (오버피팅이면 깊은 layer의 train accuracy는 높고 test accuracy는 낮아야 하는데 이건 둘 다 낮습니다.)
Fig1을 참고하면 56-layer가 20-layer보다 train error, test error 둘 다 높음을 알 수 있습니다.

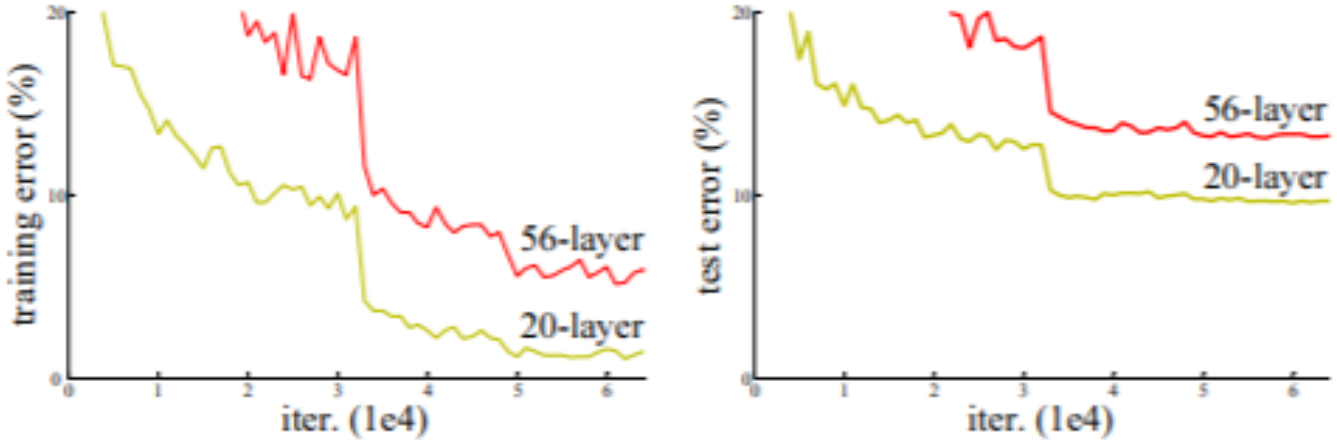

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

이 degradation 문제는 모든 시스템이 optimize하기 쉬운 것이 아니라는 뜻이기 때문에 얕은 구조와 더 깊은 아키텍처를 비교해보려고 합니다. (즉 이 논문에서 degradation의 문제는 더 깊은 레이어가 쌓일 수록 optimize가 복잡해지기 때문에 일어나는 부작용으로 보고 해결하려고 노력합니다.)
먼저 간단하게 ***Identity mapping layer***를 추가해봤지만 실험을 통해 좋은 solution이 아니라는 결과를 얻었습니다. 그래서 이 논문에서는 ***Deep residual learing framework***

는 개념을 도입합니다.

Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. : 쌓여진 레이어가 그 다음 레이어에 바로 적합되는 것이 아니라 잔차의 mapping에 적합하도록 만들었습니다.

기존의 바로 mapping 하는 것이 **H(x)** 이면 이 논문에서는 비선형적인 layer 적합인 **F(x) = H(x) - x**를 제시합니다. 이를 전개하면 H(x) = F(x) + x 의 형태가 됩니다. 여기서는 ***residual mapping (잔차 매핑) 이 기존의 mapping보다 optimize(최적화) 하기 쉽다는 것을 가정합니다.*** (극단적으로 identity mapping이 최적이라고 할 때, 비선형 레이어를 쌓아서 identity mapping을 맞추는 것 보다 잔차를 0으로 만드는 것이 더 쉽습니다.) -> Function에 대해서 맞추는 것보다 0이라는 숫자 개념으로 잔차를 수렴하게 하는 것이 더 쉽기 때문에 쓰입니다.

F(x) + x 는 ***Shortcut Connection***과 동일한데 이는 하나 또는 이상의 레이어를 skip하게 만들어줍니다. (이 모델의 아키텍처를 보면 건너뛴다는 것을 볼 수 있음.) 즉 여기서는 identity mapping으로 shortcut connection이 되게 하면서 skip을 만듭니다.

이 Identity Short Connection은 추가적인 파라미터도 필요하지 않고 복잡한 곱셈 연산도 필요하지 않는 것이 장점입니다.

위의 내용을 그림으로 이해해보면 다음과 같습니다.

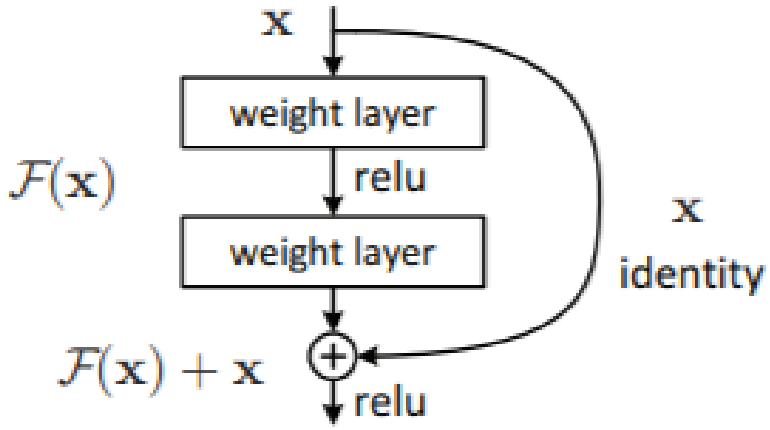

Figure 2. Residual learning: a building block.

x 는 input인데 Model 인 F(x)라는 일련의 과정을 거치면서 자신(identity)인 x가 더해져서 output으로 F(x) + x 가 나오는 구조입니다.

(추가 : 개념적으로 보면 F(x)는 Model, relu는 function 이라고 합니다.)

그래서 이제부터 실험적인 방법을 통해 degradation 문제를 보이고 이 논문의 방법을 평가하는 것이 나옵니다.

이 논문의 목표 두 가지는 다음과 같습니다. (쉬운 최적화와 accuracy 높이기)

1) Our extremely deep residual nets are easy to optimize, but the counterpart "plain" nets (that simply stack layers) exhibit higher training error when the depth increases;

-> plain net과 다르게 residual net이 더 쉽게 최적화 되는 것 보이기

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

-> residual net이 더 쉽게 accuracy높이는 것 보이기

(그 후로 성공했다는 자랑 생략)

2. Related Work : Residual Representation, Shortcut Connections

이 논문에서 사용 할 Residual Representation 과 Shortcut Connections 개념에 대한 논문들과 이에 비해 ResNet의 장점

3. Deep Residual Learning (깊은 잔차 학습, Deep Learning + Residual Learning)

아래와 같은 순서로 설명

3.1. Residual Learning

3.2. Identity Mapping by Shortcuts

3.3. Network Architectures (Plain Network , Residual Network)

3.4. Implementation (구현)

3.1. Residual Learning

앞에서 계속 했던 말과 똑같습니다. 다만 설명이 덧붙여졌습니다. H (x)를 기본 매핑으로 간주하고 x 가 input 일때 다수의 비선형 레이어가 복잡한 함수를 점근적으로 근사 할 수 있다고 가정하면 잔차함수, 즉 H (x)-x를 무의식적으로 근사 할 수 있다는 가설과 같다고 합니다.
(즉 복잡한 함수를 여러개의 비선형 레이어가 근사시킬 수 있으면 잔차함수도 근사할 수 있음.)
(사실 잔차란 쉽게 생각해서 예측값 - 실제값이기 때문에 당연한 소리입니다. 잔차함수 F(x)는 model에 input 넣어서 예측한 H(x) - 실제값 x 이니깐요.)
이 식은 수학적으로 이항만 하고 있는 것이기 때문에 동일한 하나의 식이지만 형태에 따라서 학습의 용이함이 달라진다고 합니다.
즉 H(x) ~ 형태보다 F(x) 잔차함수가 학습에 용이해서 사용합니다.
이에 대한 증명은 실험을 통해 이 논문에서 밝혔습니다.

3.2. Identity Mapping by Shortcuts

(1)식과 (2)식을 설명합니다.

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

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

식 1은 사실 F = W2σ(W1x)를 간소화한 모양입니다. 단순 덧셈으로 인해 복잡한 구조와 연산이 필요없다는 것이 이 방법의 핵심이기 때문에 이 로테이션을 통해 직관적으로 보여주는데, x는 relu함수 σ를 한 번 통과했고 bias는 생략해서 나타냅니다. 이 때 x 와 F의 차원은 동일하게 맞춰줘야 합니다. 따라서 W1, W2 등의 linear projection을 사용해서 같은 차원으로 만들어줍니다. (Ws 는 단순히 차원을 맞춰주는 방법)

3.3. Network Architectures (Plain Network , Residual Network)

그림과 같이 살펴보겠습니다. 그림에서는 VGG net, Plain net, Residual net 을 비교하고 있습니다.


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Plain Network

VGG net을 참고해서 만들었습니다. 모두 동일한 feature map 사이즈를 갖게 하기 위해서 레이어 들은 동일한 수의 filter를 가지게 합니다. 또한 feature map 사이즈가 절반이 되면 filter 수는 2배가 되도록 만들었습니다. Conv layer 는 3 * 3의 filter, stride 2로 downsampling, global average pooling layer 를 사용했고, 마지막에는 softmax로 1000-way fully-connected layer를 통과시켰습니다. 최종 레이어는 34개 입니다. 이것도 VGG net에 비하면 적은 filter와 복잡도를 가지고 VGG net 의 18%의 연산 밖에 안합니다. 이와 비교해서

Residual Network

위에서 만든 ***Plain net***을 기본 바탕으로 하지만 여기에 ***shortcut connection*** 개념을 도입합니다.

Identity Shortcut은 input과 output을 같은 차원 (same dimension)으로 맞춰줘야 합니다. 그 후에 차원이 증가할 때 두 가지 방법을 사용합니다.

(A) The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions.

This option introduces no extra parameter;

-> 차원을 늘리기 위해 0을 넣어서 padding 하기(같은 차원으로 만들어 주기 위해서), 추가 파라미터가 없음

(B) The projection shortcut in Eqn.(2) is used to match dimensions (done by 1×1 convolutions).

-> 식2에서 쓴 방법 사용 (linear projection,)

For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

즉 input 과 output, x와 F 를 모두 같은 차원으로 만들어주는 것이 추가된 과정입니다.

Shortcut Connection이란 같은 차원으로 나오는 것

3.4. Implementation (구현)

단순한 구현 방법을 설명합니다.

- image resized 224 * 224
- Batch normalization BN 사용
- Initialize Weights
- SGD, mini batch 256
- Learning rate 0.1

(참고 : learning rate 설정에 대한 이론을 알고 싶으면 armijo rule 참고할 것, 딥러닝은 0.1,0.01,0.001 의 정해진 수를 많이 씀)

Iteration 60 * 10^4

- weight decay 0.0001, momentum 0.9
- No dropout

4. Experiments

아래와 같은 순서로 설명

4.1. ImageNet Classification 이미지 분류

4.2. CIFAR-10 and Analysis

4.3. Object Detection on PASCAL and MS COCO

4.1. ImageNet Classification 이미지 분류

plain net, residual net, identity vs Projection Shortcuts, deeper bottleneck architectures, 50-layer resnet, 101-layer and 152 later resnets comparisons with state-of-the-art methods 순서로 설명합니다.

Plain Network

pain net 으로 쌓은 18-layer와 34-layer를 비교하면 34-layer에서 train error, test eroor 모두 높은 gradation problem 볼 수 있습니다.

-> plain net 의 경우 34-layer의 성능이 더 낮다.

plain net 이 BN으로 학습되었기 때문에 non-zero varience를 가지고 optimiziation 이 어려운 이유가 vanishing gradient 때문이 아닌 것을 알 수 있습니다.

-> 기하급수적으로 수렴이 어려워지는 것을 training error 의 reducing에서 어려운 점으로 본다.

Residual Network

18-layer,34-layer residual nets(Res Net) 을 평가하면 다음과 같습니다.

먼저 모든 shortcuts을 위해서는 identity mapping을 사용했고, 차원증가를 위해서는 zero-padding을 사용합니다. (option A)

(그래서 plain과 비교시 추가 파라미터 없음)

논문에 있는 Figure4와 Table2에서도 알 수 있 듯이 3가지 시사점이 있다.

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

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

1) 18-layer ResNet 보다 34-layer ResNet이 성능이 좋다. 더 낮은 training error를 가진다. layer의 깊이가 증가할 때도 degradation problem 을 잘 조절했음을 보인다.

2) plain과 비교했을 때 성공적으로 training error 를 줄였다.

3) 18-layer들 끼리 비교했을 때 accruracy는 비슷했지만 18-layer ResNet이 더 수렴이 빨랐다. (converges faster)

-> ResNet 이 SGD를 사용한 optimization이 더 쉬움을 알 수 있다.

Identity vs Projection Shortcuts

training에서 identity shortcuts을 돕기 위해서 parameter-free 방법을 사용하는데 projection shortcut과 비교해보겠습니다.

(위의 3.3 참고)

A. 차원 증가를 위해 zero-padding을 사용하는 경우 (추가 파라미터 없음)

B. 차원 증가를 위해 projection shortcuts을 사용, (Ws) (다른 shortcuts은 identity)

C. 모든 shortcut들이 projection인 경우

-> 논문의 Table3로 비교해본 결과는 다음과 같습니다.

-> C > B > A

-> A는 residual learning 이 이뤄지지 않기 때문에 B가 성능이 미세하게 더 좋고, C가 B보다 좋은 이유는 projection shortcut에 사용된 extra parameter때문입니다.

-> 하지만 이러한 A,B,C 들 사이의 작은 차이는 주요 문제가 projection shortcut 때문에 생기는 것이 아니라는 것을 뜻 합니다.

-> 그래서 C가 성능이 제일 좋았지만 model 이 복잡해지는 것을 위해서 이 논문에서는 사용하지 않겠습니다.

(identitiy shortcut은 병목구조의 복잡성을 막기위해 중요한데 다음 병목 구조에서 설명)

Deeper Bottleneck Architectures

(깊은 병목구조, Building block -> Bottleneck)

train 시간을 생각해서 아키텍처의 구조를 ***Bottleneck design***을 사용합니다.

각각 Residual function F를 사용하는데 3층을 쌓을 때, (1 x 1) -> (3 x 3) -> (1 x 1) 순서로 병목모양으로 쌓습니다.

-> 이는 기존 구조와 비슷한 복잡성을 가지면서 input과 output의 dimension을 줄이기 때문에 사용합니다.

50-layer

기존에 만들었던 34-layer에다가 ㅏ 3-layer bottleneck block을 추가해서 50-layer ResNet을 만듭니다.

-> option B 사용, 연산은 3.8billion FLOPs

101-layer and 152-layer ResNets

더 많은 3-layer를 추가해서 101-layer와 152-layer를 만듭니다. 이 중 152-layer ResNet이 VGG보다 작은 복잡성과 적은 연산을 가져서 유의미합니다. 또한 50/101/151 layer는 34-layer보다 확실히 더 정확합니다. (degradation problem X, 깊이의 이점을 살림)

Comparisons with State-of-the-art Methods

결과적으로는 위에서 만들었던 6가지의 모델을 앙상블을 해서 error fmf 3.57%까지 줄였고 이 결과가 ILSVRC 2015에서 우승한 모델입니다.

4.2. CIFAR-10 and Analysis

CIFAR-10 data에도 적용해본 실험 결과를 보여줍니다. 결과적으로 1000개 이상의 레이어도 쌓아봤는데 1000개 이상의 레이어에서 성능이 낮아지는 것은 overfitting 현상이고 이 논문에서 dropout, maxout 등의 regularization이 쓰이지 않기 때문에 추후 regularization으로 이 문제를 해결해야합니다.

4.3. Object Detection on PASCAL and MS COCO

자랑 생략

Reference - 논문 참고

덧붙임

이 논문은 다음 논문인 [Identity Mappings in Deep Residual Networks] 과 이어집니다. 이 후 논문에는 개선된 형태와 부족했던 수식적 설명, 추가 실험 등이 실려있습니다.

Identity Mappings in Deep Residual Networks 리뷰는 다음 글에서 포스팅 하겠습니다.

이 밖에 Pytorch 의 ResNet 구현 https://pytorch.org/hub/pytorch_vision_resnet/

추가 의견 및 잘못된 점의 지적은 댓글로 부탁드립니다~!