



STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET

이상용 / 2020-03-27



Computational Data Science LAB



STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET

Computational Data Science LAB

목차

1. INTRODUCTION
2. MODEL DESCRIPTION
3. EXPERIMENTS



논의사항 및
결정사항

관련문서

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. Accepted as a workshop contribution at ICLR 2015



CONTENTS

1. INTRODUCTION
 2. MODEL DESCRIPTION
 3. EXPERIMENTS
- 
- 

01 | INTRODUCTION

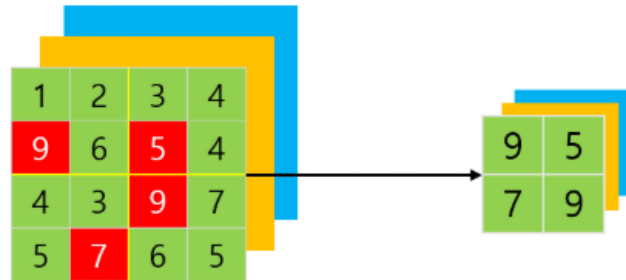
- 현대 대부분의 CNN은 object recognition 문제를 풀 때 동일한 원칙을 가지고 있음
- 주로 convolution과 max-pooling layer를 교대로 사용하거나 fully connected layer를 사용
- 본 논문은 정확도의 손실없이 max-pooling을 간단한 convolution layer로 바꾸는 새로운 아키텍처를 제안

02 | MODEL DESCRIPTION

- CNN에서 pooling layer가 사용되는 이유
 1. $p - norm$ 을 사용하면 feature들을 좀 더 invariant하게 만들 수 있음
 2. Spatial dimensionality reduction을 수행하기 때문에, 큰 사이즈의 이미지 입력을 허용
 3. Convolution을 사용하여 얻은 mixed feature보다 optimization이 쉬움
- Pooling layer의 단점
 - ✓ pooling을 통해 dimensionality reduction을 수행하면, 일부 feature information이 손실될 수 있음
- Convolution layer로의 대체
 - ✓ CNN으로 우수한 성능을 달성하는 이유 중 하나를 pooling layer가 사용되는 이유의 두 번째라고 가정한다면, convolution을 사용하여 feature information을 잃지 않으면서 spatial dimensionality reduction을 수행 가능

02 | MODEL DESCRIPTION

p-norm subsampling(pooling)



$$s_{i,j,u}(f) = \left(\sum_{h=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k}{2} \rfloor} \sum_{w=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k}{2} \rfloor} |f_g(h,w,i,j,u)|^p \right)^{1/p}$$

f : output of convolutional layer

$s_{i,j,u}$: output of p – norm subsampling with 3 – dimensional array of size $i \times j \times u$

h, w : height, width of feature map

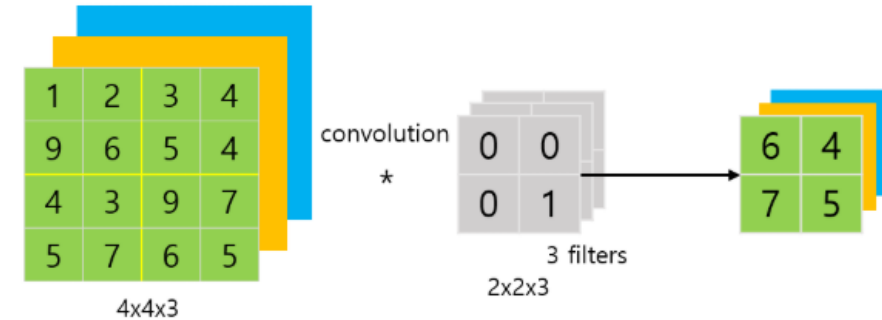
k : pooling size

p : order of the p – norm (for $p \rightarrow \infty$, it becomes the commonly used max pooling)

$g(h, w, i, j, u) = (r \cdot i + h, r \cdot j + w, u)$

r : stride

convolution with stride 2



$$c_{i,j,o}(f) = \sigma \left(\sum_{h=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k}{2} \rfloor} \sum_{w=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k}{2} \rfloor} \sum_{u=1}^N \theta_{h,w,u,o} \cdot f_g(h,w,i,j,u) \right)$$

N : number of channels in output of convolutional layer

$\sigma(\cdot)$: activation function

θ : convolutional weights(kernel weights, filters)

$o: o \in [1, M]$ number of output feature of the convolutional layer

02 | MODEL DESCRIPTION

- CNN에서 pooling layer를 1보다 큰 stride를 갖는 convolution layer로 교체
 - ✓ Filter와 stride로 dimensionality reduction을 수행하여 pooling과 동일한 shape의 출력을 생성하는 convolution layer로 교체
- Pooling layer를 convolution layer로 바꾸는 것은 θ 에 어느 제한이 있지 않는 한 feature들 간의 관계성 고려 가능
- Convolution layer를 사용하는 것은 pooling layer의 고정된 값을 '사용' 하는 것이 아닌 값을 '학습' 하는 것

03 | EXPERIMENTS

• 사용된 모델

Model		
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
Input 32×32 RGB image		
3×3 conv. 96 ReLU	3×3 conv. 96 ReLU	3×3 conv. 96 ReLU
3×3 conv. 96 ReLU	3×3 conv. 96 ReLU	3×3 conv. 96 ReLU
with stride $r = 2$	3×3 conv. 96 ReLU	
	3×3 max-pooling stride 2	3×3 conv. 96 ReLU
		with stride $r = 2$
3×3 conv. 192 ReLU	3×3 conv. 192 ReLU	3×3 conv. 192 ReLU
3×3 conv. 192 ReLU	3×3 conv. 192 ReLU	3×3 conv. 192 ReLU
with stride $r = 2$	3×3 conv. 192 ReLU	
	3×3 max-pooling stride 2	3×3 conv. 192 ReLU
		with stride $r = 2$
	Pooling	Convolution

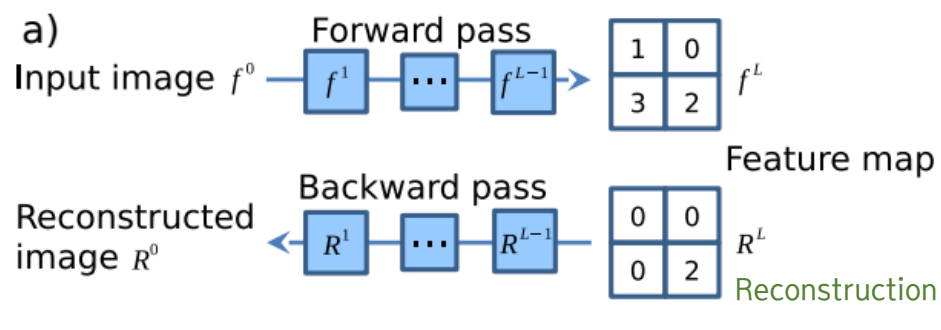
• 실험결과

CIFAR-10 classification error		
Model	Error (%)	# parameters
without data augmentation		
Model A	12.47%	≈ 0.9 M
Strided-CNN-A	13.46%	≈ 0.9 M
ConvPool-CNN-A	10.21%	≈ 1.28 M
ALL-CNN-A	10.30%	≈ 1.28 M
Model B	10.20%	≈ 1 M
Strided-CNN-B	10.98%	≈ 1 M
ConvPool-CNN-B	9.33%	≈ 1.35 M
ALL-CNN-B	9.10%	≈ 1.35 M
Model C	9.74%	≈ 1.3 M
Strided-CNN-C	10.19%	≈ 1.3 M
ConvPool-CNN-C	9.31%	≈ 1.4 M
ALL-CNN-C	9.08%	≈ 1.4 M

03 | EXPERIMENTS

Guided-backpropagation

- 네트워크를 분석하기 위해 ‘deconvolution’ approach 수행
 - ✓ 본 논문은 pooling을 쓰지 않기 때문에, pooling 없이도 시각화가 가능한 새로운 방법 제안



c)

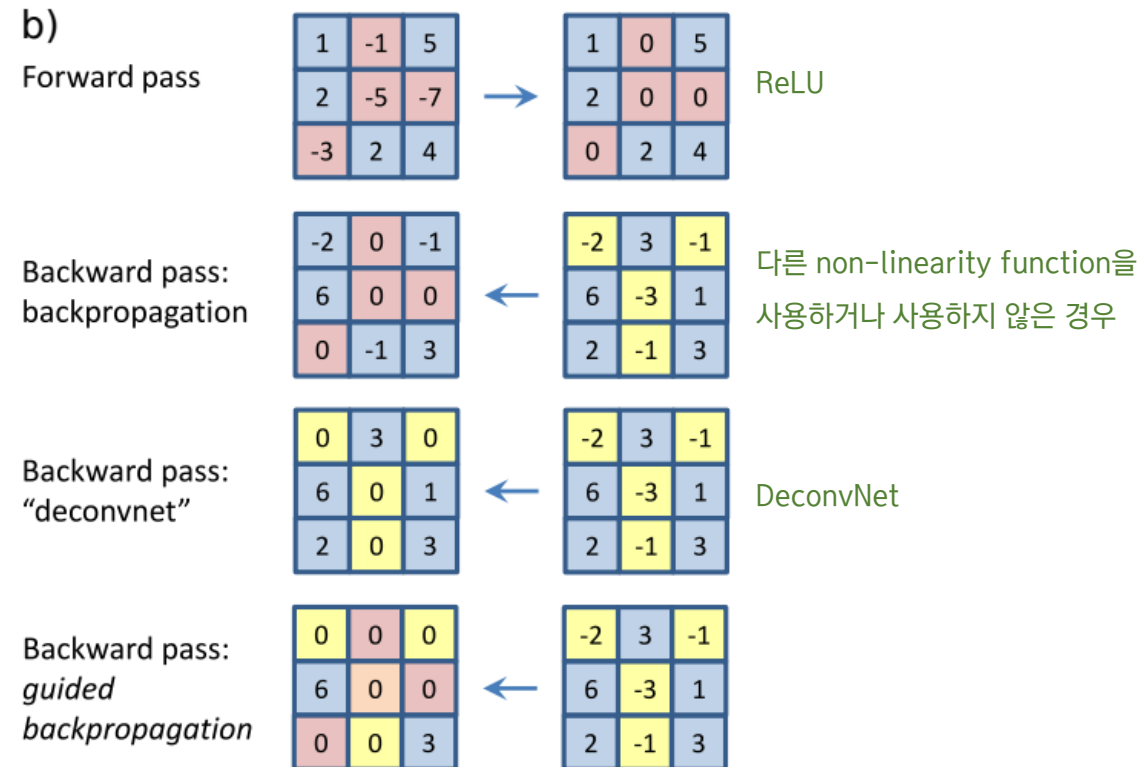
activation: $f_i^{l+1} = \text{relu}(f_i^l) = \max(f_i^l, 0)$

backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f_{out}}{\partial f_i^{l+1}}$

backward 'deconvnet': $R_i^l = (R_i^{l+1} > 0) \cdot R_i^{l+1}$

guided backpropagation: $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$

ReLU n DeconvNet



03 | EXPERIMENTS

deconv



guided backpropagation



corresponding image crops



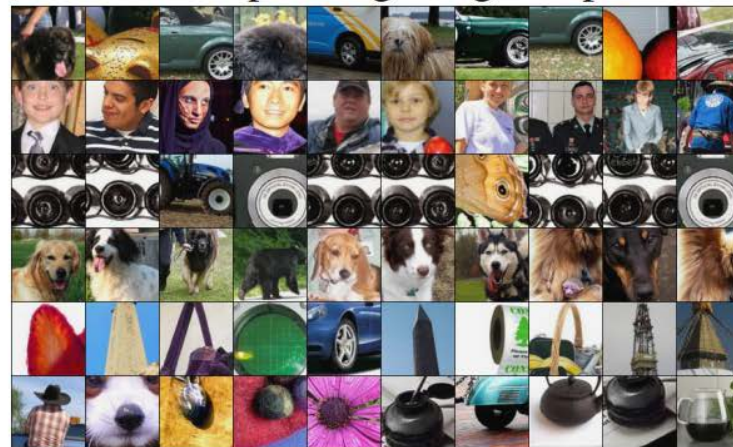
deconv



guided backpropagation



corresponding image crops



Q&A

감사합니다.