Very Deep Convolutional Networks For Large-Scale Image Recognition

다른 ConvNet 모델들

1. smaller receptive window size and smaller stride of the first convolutional layer

   → 데이터 손실이 적다

2. training and testing the networks densely over the whole image and over multiple scales

VGG : 네트워크의 깊이에 대해서 연구

매우 작은 3*3 conv. filter를 여러 겹 쌓아서 사용해서 훨씬 더 좋은 성능을 보이는 Conv Net을 제시.

1. ILSVRC의 classification과 localization 분야에서 state-of-the-art 정확성을 보임
2. 다른 image recognition dataset에도 잘 적용됨

2.1. Architecture

training input : 고정된 사이즈 (224 x 224 RGB image)

preprocessing : train set 이미지에서 RGB value 평균을 구하고 이를 각 픽셀에서 뺌

convolution filter : 3 x 3 또는 1 x 1 , stride 1, padding same → 3 x 3 필터의 경우 stride 1, padding 1

activation : ReLU

pooling : 5개의 Max Pooling. conv filter 뒤에 옴 ; 2 x 2, stride 2

FC layers : 3 FC layers 사용 ; 처음 2 layers는 4096 channels, 마지막 layer는 1000 channels

2.2. Configurations

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

parameter

conv3 - N : #input channel * 3 * 3 * N

1st FC – 4096 : 512 * 7 * 7 * 4096 = 102 M

2nd FC – 4096 : 4096 * 4096 = 16 M

3rd FC – 1000 : 4096 * 1000 = 4 M

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

2.3. Discussion

model	ILSVRC-2012	ILSVRC-2013	VGG
Filter size	11 x 11	7 x 7	3 x 3
stride	4	2	1

3 x 3 filter 2개가 5 x 5 filter 한 개와 receptive field가 같고, 3 x 3 filter가 3개가 7 x 7 filter 한 개와 같음

1. non-linearity (ReLU)를 여러 개 포함할 수 있다. ← regularization
2. parameter 수가 적다.

   7 x 7 = 49, 3 x 3 x 3 = 27

2.3. Discussion

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

모델 C에서 conv1 사용하는 이유?

non-linearity 증가

2.3. Discussion

다른 모델과 비교

- Cireasan et al. 도 작은 사이즈의 필터를 사용했지만 얕고 큰 데이터셋에서 evaluate하지 않았음
- Goodfellow et al. 은 11 layer 사용 → 깊은 네트워크가 더 나은 성능을 보인다는 것을 보여줌
- GoogLeNet은 22 layer 사용 ; 3 x 3 뿐 아니라 1 x 1, 5 x 5 filter도 사용.

  → VGG가 구조는 더 간단하고, single-network classification 에서는 GoogLeNet보다 더 좋은 성능을 보인다

Optimization

- 다중 로지스틱 회귀 방식
- mini-batch GD with momentum (0.9)
- learning rate : e-2
- lr decay : 1/10 decay when validation set accuracy stopped improving (총 3번 decay)
- regularization : weight decay ; L2 penalty multiplier 5e-4
- dropout : first two FC layers. ratio : 0.5

→ 74 epoch 만에 수렴

타 모델들보다 parameter 수는 많은데 빠르게 수렴한 이유?

1. 더 깊은 네트워크, 더 작은 conv. filter를 사용함이 regularization으로 작용
2. 특정 레이어의 pre-initialization

Initialization

1. 얕은 모델 A를 먼저 random initialize 한 후에 훈련
2. 깊은 모델들의 첫 conv. layer랑 마지막 3개 FC layer를 A에서 훈련시킨 layer를 사용
   이 때 A를 사용한 레이어들에서도 lr을 감소시키지 않아서 training 가능하도록 함
3. 나머지 layer는 정규분포 N(0, e-2)에서 sampling
4. bias 는 0으로 초기화

3.1. Training

Input

batch size : 256

고정된 사이즈 224 x 224 RGB image 받기 위해서 rescale 된 training 이미지들을 crop해서 사용
augment : crop을 random horizontal flipping, random RGB color shift

Training Image Rescale

Fixed S (256 or 384)

1. S = 256으로 train
2. S = 384 이용 시 training 속도를 위해
   1. 에서 학습된 weight로 초기화하고
   lr을 e-3으로 줄여서 train시킴

Multi-scale Training

1. [Smin, Smax] 구간 내에서 S를 샘플링
2. train 속도를 위해 각 모델들과 같은 configuration에서 fixed S=384로 pretrain 후 layer들 fine-tuning

3.2. Testing

1. test scale Q를 이용하여 image rescale : Q != S → 성능 향상에 도움
2. FC layer를 Conv layer로 변환
3. 변환된 FCN (Fully convolutional Network)을 전체 이미지에 적용
4. class score map, variable spatial resolution이 return
5. class score map은 sum-pooled ← fixed size class score vector를 얻기 위함
6. augmentation : horizontal flipping

   → 원본과 flip된 이미지의 softmax class 평균으로 final score를 구함.

evaluation	Multi-crop	Dense
Accuracy	높음	낮음
Computational cost	비쌈	비교적 쌈
padding	Zero padding	주변 픽셀

dense evaluation이 context를 더 잘 잡아냄

3.3. Implementation Details

- C++ Caffe를 이용하여 구현
- 4 - GPU system에 data parallelism을 이용함.
- 각 batch가 4개의 GPU로 나눠 들어가서 처리됨. batch gradient는 GPU gradient의 평균으로 구함.
- single GPU 사용 시보다 3.7배 더 빠른 성능을 보임.

4.1. Single Scale Evaluation

fixed S → Q = S

jitter S → Q = (Smin + Smax) / 2

1. LRN 사용하는 것이 효과가 없음

   → 더 deep한 network에서 LRN 사용하지 않았음.

2. layer가 19개 쌓일 쯤에 accuracy saturate. 그래도 더 큰 데이터셋에서는 더 쌓으면 더 좋은 성능을 보일지도
3. train 시 scale jittering 모델이 더 나은 성능을 보여줌

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

dataset : ILSVRC-2012

validation set을 test set으로 사용

< 모델 비교 >

- non-linearity가 성능 향상 도움
- non-trivial receptive field를 가지는 필터를 사용하는 것이 context를 잡는 데 도움
- 작은 필터를 사용하는 깊은 모델이 큰 필터를 사용하는 얕은 모델보다 좋은 성능을 보여줌

4.2. Multi-scale Evaluation

fixed S → Q = { S-32, S, S+32}

jitter S → Q = { Smin, (Smin + Smax) / 2, Smax }

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

Table 4: **ConvNet performance at multiple test scales.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

여러 rescale 이미지에서 나온 class posteriors를 평균을 내어 사용

test time에 jittering하는 것이 fixed S를 사용하는 것보다 더 좋은 성능을 냄

4.3. Multi-Crop Evaluation

Table 5: **ConvNet evaluation techniques comparison.** In all experiments the training scale S was sampled from $[256; 512]$, and three test scales Q were considered: $\{256, 384, 512\}$.

ConvNet config. (Table 1)	Evaluation method	top-1 val. error (%)	top-5 val. error (%)
D	dense	24.8	7.5
	multi-crop	24.6	7.5
	multi-crop & dense	**24.4**	**7.2**
E	dense	24.8	7.5
	multi-crop	24.6	7.4
	multi-crop & dense	**24.4**	**7.1**

multi-crop이 dense보다 아주 살짝 더 나은 성능을 보임. 그리고 둘을 상보적으로 사용한 경우 outperform

4.4. ConvNet Fusion

앙상블 방법 : averaging softmax class posteriors

Table 6: **Multiple ConvNet fusion results.**

Combined ConvNet models	Error		
	top-1 val	top-5 val	top-5 test
ILSVRC submission			
(D/256/224,256,288), (D/384/352,384,416), (D/[256;512]/256,384,512) (C/256/224,256,288), (C/384/352,384,416) (E/256/224,256,288), (E/384/352,384,416)	24.7	7.5	7.3
post-submission			
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), dense eval.	24.0	7.1	7.0
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop	23.9	7.2	-
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop & dense eval.	**23.7**	**6.8**	**6.8**

4.5. Comparison with The State Of the Art

Table 7: **Comparison with the state of the art in ILSVRC classification**. Our method is denoted as "VGG". Only the results obtained without outside training data are reported.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

single net의 경우 VGG가 GoogLeNet보다 error rate가 0.9% 낮음
2개의 net만 이용했음에도 불구하고 Clarifai처럼 여러 net 이용한 것보다 훨씬 낮은 error rate