VGG

0. Abstract

1. 3x3 (같이 작은 filter) ConvNet + 깊이↑ = 정확도↑
   (11,13,16,19 실험
   → 그 중 16,19가 가장 성능↑)
2. 결과: ImageNet challenge 2014

1. Introduction

1. AlexNet 개선 트릭 (2012)
   - Zeiler & Fergus 2013, OverFeat 2014
     → ILSVRC-2013
     : AlexNet의 첫번째 Conv의 receptive field 크기와 stride 가 커서 이미지를 한번에 너무 축소시키는 문제 → 줄이기
   - Sermanet 2014, Howard 2014
     : 전체 이미지에 dense 학습, 테스트를 수행하고 multi-scale 적용 → 여러 이미지 scale과 이미지 전역을 dense하게 보기위함
     = 여러 해상도 (HxW)

↳

		resize	multi-scale	crop
Sermanet	2014			
	train	O	X	O (위치 랜덤, 크기는 고정)
	test	O	O	X (대신 슬라이딩 윈도우)
Howard	2014			
	train	O	X	O (위치 랜덤, 크기는 고정)
	test	O	O	O (크기, 위치 고정 크롭)

⇒ 다른 기법: "깊이 + 3x3 (작은 필터)" (VGG)

ex) 600 x 900
↓
Resize(480) (rescaling : 224, 384 ...)
↳ 작은변 (600)을 480으로 하되, 종횡비 유지
↓
Crop

2. ConvNet Configuration

1. AlexNet 계통 구조, 가중치 동일

2.1 Architecture

1. img
   - size: 224 × 224
   - channel: 3(RGB)
   - 전처리: 각 픽셀에서 평균 RGB 값 빼기

2. conv 계통
   - receptive field: 3×3 (↳ 최소 단위 (1×1은 고려 X))
   - stride: 1 → 모든 픽셀에 conv 적용 가능
   - 3×3은 padding = 1 (입력 해상도 = 출력 해상도): $\frac{(224-3+2\times1)}{1}+1=224$

   or

   1×1 (↳ 채널 선형변환 (H×W는 그대로)) → 후에 activation func (비선형)

3. max-pooling 계통: 일부 Conv 뒤에 (총 5개)
   kernel_size: 2, stride: 2

4. 3개 fc
   - → 4096
   - 4096 → 4096
   - 4096 → 1000 (num_classes)

   + softmax

5. activation func: ReLU

6. LRN (Local Response Normalization): AlexNet 설명 참고

⇒ "모두 공통: Net 깊이만 다름"

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** (filter out-ch)	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					

2.2 configurations

Table 2: **Number of parameters** (in millions).

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

→ 얕고 넓은 (receptive field = 필터 크기 ↑) net 보다 파라미터 ↓

2.3 Discussion

1. 3×3 3개 결과 크기 $\left(\frac{224-3}{1}+1=222\rightarrow220\rightarrow218\right)$

   = 1×1 1개 // $\left(\frac{224-7}{1}+1=218\right)$

   ↳ 3×3 3개 장점
   1) 비선형성 증가: 각 3개 계층 후 ReLU
   2) 파라미터 수(가중치) 감소: $3\text{개}\times(3^2\times C^2)=27C^2 \ll 1\text{개}\times(7^2\times C^2)=49C^2$

↳ 공식: $C_{out} \times C_{in} \times \underbrace{K_H \times K_W}_{\text{kernel size}}$

3) 추가적 제약 = 정규화

→ fm 결과가 3x3 크기를 거쳐야 만들어짐

→ "정규화" ; 과적합 감소 장치

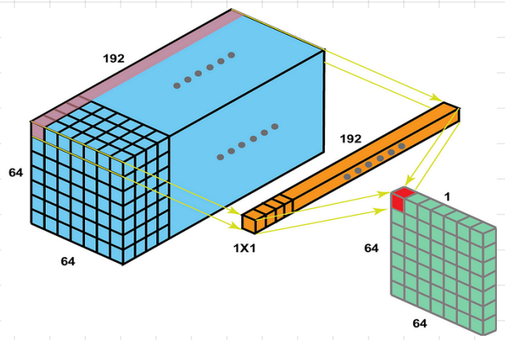
2. 1x1 장점

→ $y_{i,j} = \underline{W} x_{i,j} + b$ (선형변환)

= 필터

→ 1x1 이라도 Conv 계산은 함, 이때 HxW는 그대로이고, 채널만 섞임

→ + 비선형성 (1x1 자체는 선형이지만 ReLU)



3. GoogLeNet은 보다 복잡, 계산량 늘리기 위해 초기레이어에서 HxW 급격 감소

3. Classification Framework

3.1 Training

1. Loss : multinomial Logistic Regression

= logits에 softmax(activation func)를 거쳐 나온 확률의 Cross-Entropy를 최소화

↳ 샘플 $i=1 \dots N$ / 클래스 $k=1 \dots K$

logit : $z_{ik} = w_k^T x_i + b_k$

softmax : $P_\theta(y=k|x_i) = \frac{e^{z_{ik}}}{\sum_{j=1}^{K} e^{z_{ij}}} = q_{ik}$

Loss : $L(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} z_{ik} + \sum_{i=1}^{N} \log \sum_{j=1}^{K} e^{z_{ij}}$ = 데이터 샘플 (x_i, y_i)의 NLL (CE)

↳ Cross-entropy loss (NLL) : $CE = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log q_{ik}$ (정답 분포, 예측 분포)

정답 / 예측

$$CE = -\sum_{i=1}^{N}\sum_{k=1}^{K}\left[y_{ik}\left\{z_{ik} - \log\sum_{j=1}^{K} e^{z_{ij}}\right\}\right]$$

$$= -\sum_{i,k} y_{ik} z_{ik} + \sum_{i,k}\left[y_{ik} \cdot \log\sum_{j} e^{z_{ij}}\right]$$

$$= \sum_{i}\left[\sum_{k} y_{ik} \cdot \log\sum_{j} e^{z_{ij}}\right]$$

$\sum_k$ 먼저 계산

$$= \sum_{i}\left[1 \cdot \log\sum_{j} e^{z_{ij}}\right]$$

$$= \sum_{i=1}^{N} \log\sum_{j=1}^{K} e^{z_{ij}}$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} z_{ik} + \sum_{i=1}^{N} \log\sum_{j=1}^{K} e^{z_{ij}}$$

2. optimizer: mini-batch gradient descent

batch for문 내부 ↲ 매번 갱신 (epoch 내부)

\+ momentum (0.9)

\+ weight decay (= L2 penalty, 5×10^{-4})

⇒ optim.SGD(lr=0.01, momentum=0.9, weight_decay=5e-4)

3. batch size: 256
4. dropout(0.5): FC 초기 2개
5. lr: 10^{-2} → 10배씩 3번 감소
6. epochs: 74 (AlexNet 보다 파라미터 · 깊이↑ but, 필터↓(정규화) + 가중치 사전 초기화)
7. 가중치 초기화: 잘못 설정하면 gradient 불안정
   - → Net A에서 무작위 초기화 후 안정 수렴 가중치 확보
   - → 첫 4개 conv, 3 fc 가중치만 더 깊은 Net에서 초기 가중치로 설정 (계속 미세조정 / 나머지는 무작위)
   - $\mu=0, \sigma^2=10^{-2}$ 인 정규분포
     - 작은 분산 = 가우시안 초기화 방향
     - 정규분포 = 가우시안
     - ↳ μ기준 좌우대칭, 평균=중앙값=최빈값
   - → "Glorot & Bengio (2010): Xavier"
8. img size 224 x 224

↓

training Rescaling: 등방(짧은 변=S로 하고 종횡비 유지, S≥224)

↳ S=224이면 크롭이 전체 이미지 포함

S>224이면 객체 일부 포함

→ S 설정

1) single scale
- → S고정 (256, 384 이용)
- → S=256 학습 후 S=384는 학습 속도를 위해 S=256의 가중치를 초기값으로 하고 lr=1e-3 로 해서 미세조정

2) multi-scale
- → [256,512] random rescaling
- → 384 기반 가중치 초기값으로 하고 학습하면서 계속 미세조정

↓

random crop (224 고정)

(SGD iteration, 즉 loader for문 마다 img = transform(img))

\+

crop된 img에 [RandomHorizontalFlip(p), ColorJitter()]

3.2 Testing

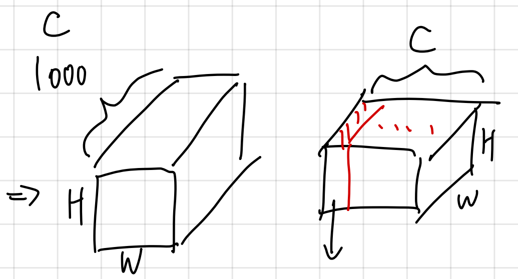
0. 원본 + 수평 뒤집기 (평균)
1. 짧은 변을 Q로 하는 등방성 리스케일링 (종횡비 유지)
2. densely net 적용 (학습의 crop 대신)

→ 전체 이미지에 fully-convolutional net

fc1 → 7x7 conv

fc2,3 → 1x1 conv

```
nn.Linear(512*7*7, 4096)
nn.Linear(4096,4096)
nn.Linear(4096, 1000)
        ⇓
nn.Conv2d(512, 4096, kernel_size=7)
nn.Conv2d(4096, 4096, 1)
nn.Conv2d(4096, 1000, 1)
```



⇓

최종 클래스 점수 map (채널 수 = 클래스 수, 즉 그냥 fc처럼 logits)

→ 클래스 점수 "벡터" → softmax

각 클래스에 대해 "spatially averaged (sum-pooled)" 공간 (HxW)

※ weight initialization

$fan\_in = C_{in} \times k_{in}^2 = n_{in}$ = 입력 뉴런 수

$fan\_out = C_{out} \times k_{out}^2 = n_{out}$ = 출력 뉴런 수

(FC: $k_{in}=k_{out}=1$)

1) Gaussian
- → 스케일 조정X 작은 분산 + 단순 무작위
- → 소실/폭주 위험

2) Xavier
- → sigmoid, tanh 적합
- → 균등분포: $W \sim U\left[-\sqrt{\frac{6}{n\_in+n\_out}}, \sqrt{\frac{6}{n\_in+n\_out}}\right]$
- → 정규분포: $W \sim N\left(0, \frac{2}{n\_in+n\_out}\right)$ (μ, σ^2)

3) He/kaiming
- → ReLU
- → 균등분포: $W \sim U\left[-\sqrt{\frac{6}{n\_in}}, \sqrt{\frac{6}{n\_in}}\right]$
- → 정규분포: $W \sim N\left(0, \frac{2}{n\_in}\right)$ (μ, σ^2)

→ AlexNet 처럼 test 마다 여러 크롭 샘플링해서 매번 네트워크 실행 X

↳ AlexNet : multi-crop testing vs vgg : dense evaluation

AlexNet (multi-crop testing) ↓
crop 한다면 전체 이미지에서
특정 부분을 자르고 0 패딩
→ 문맥 단절

vgg (dense evaluation) ↓
기존 이미지에서 conv로
슬라이딩 윈도우
→ 인접 픽셀을 사용 O → 문맥 연결 (receptive field ↑)

4. Classification Experiments

1. ILSVRC-2012 (class 1000, train 1.3M, val 50k, test 100k)

→ Top-1 error : 잘못 분류된 이미지 비율
Top-5 error : 예측된 상위 5개 클래스 내 정답 클래스가 포함X 비율

4.1 Single scale Evaluation

고정 (S=Q) / scale Jittering : $S \in [S_{min}, S_{max}]$인 경우,
$Q = 0.5(S_{min} + S_{max})$
↳ 사후 확률은 평균으로

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 (1x1 3개)	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D (3x3 1개)	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**

1. 정규화 효과 (LRN) : X (A > A-LRN) → B~E에서 LRN X
2. A→E : 깊어질수록 오류↓
3. C < D : 동일한 깊이 → 비선형이 추가되어도 유의미한 수용영역 필요
   (유의미한 수용영역: 공간적 문맥 파악)
4. Scale Jittering : 성능↑ (∵ 학습 증강)

⊕ B : 3x3 2개 → 5x5 1개 대체 시 error↑

4.2 Multi scale Evaluation

1.

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

ConvNet config. (Table 1)	smallest image side: train (S)	smallest image side: test (Q)	top-1 val. error (%)	top-5 val. error (%)
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**

1. 학습 size 고정시 test에서 scale Jittering하면 성능↓

→ $Q = S-32, S, S+32$

2. 학습 시 Jittering : $[S_{min}, S_{max}]$ → test: $(S_{min}, 0.5(S_{min}+S_{max}), S_{max})$

⇒ 4.1 Table 3 보다 성능↑ (Test scale Jittering 효과)

4.3 Multi-CROP Evaluation

1.

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-crop eval vs dense eval

성능: > ← dense가 crop(0패딩)보다 자연스러운 주변 패딩을 갖지만 잡음

⇓

결합 ← 서로 보완

4.4 ConvNet Fusion

1.

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

Combined ConvNet models	Error: top-1 val	Error: top-5 val	Error: 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**

↳ 여러 모델 앙상블

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	-

5. Conclusion

→ 깊이 + 작은 filter