

AlexNet

0. Abstract, 2. The Dataset

1. 이미지 종류: ImageNet LSVRC-2010 의 (20만장 / 1000 labels)

2012
2. 결과: top-1 error 37.5% (모델이 가장 확률이 높은 예측한 클래스가 정답이 아닌 비율)

top5 15.9%
3. 파라미터: 6천만개

뉴런: 65만개

conv + maxpool: 577

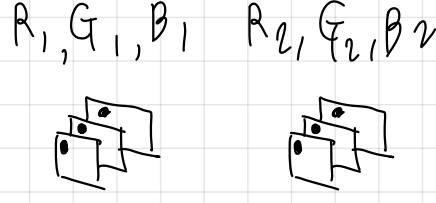
fc: 777 (마지막 1000개의 softmax)

activation: ReLU func

Dropout

오버피팅: 256x256 \Rightarrow 400
- (모델이 상위 5개에 예측한 클래스 중 정답이 없는 비율)

$$\mu_R = (R_1 + R_2) / 2 \Rightarrow R'_1 = R_1 - \mu_R, R'_2 = R_2 - \mu_R$$



\Rightarrow "224 x 224"

(참고 4.1 Data Augmentation)

1. Introduction

1. 문제: 데이터가 적 \downarrow \rightarrow 과적합 방지
- 해결: 데이터가 많 \uparrow
- label-preserving 변환: 증강기법

\rightarrow 미신능: 높은 learning capacity X

\downarrow

CNN: HxW를 용량 제어 통계적 정상성

큰바른 가정 (stationarity or statistics: 각 픽셀은 위치가 다른 특성이 담겨지지 않음)

& locality or pixel dependency: 근처에 있는 픽셀끼리 밀접 연관

픽셀 의존성 최소화

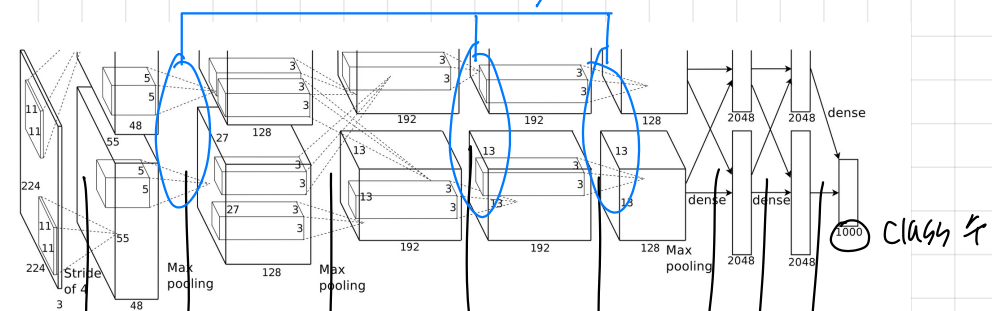
\downarrow

반응 fci 비례 변환수 parameter 수 \downarrow , 성능 \uparrow

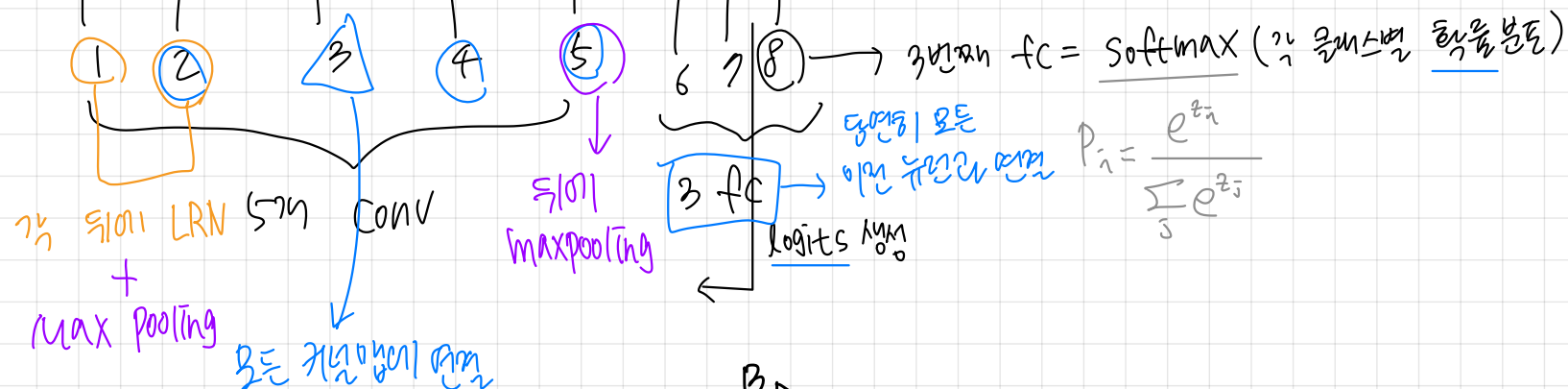
3. The Architecture

1. 전체적인 Overall Architecture

같은 GPU 연결된 커널맵에만 연결시킴



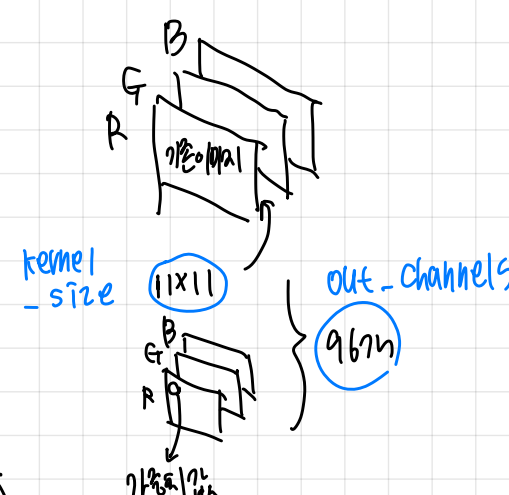
← 모든 Conv, fc 뒤에는 RELU



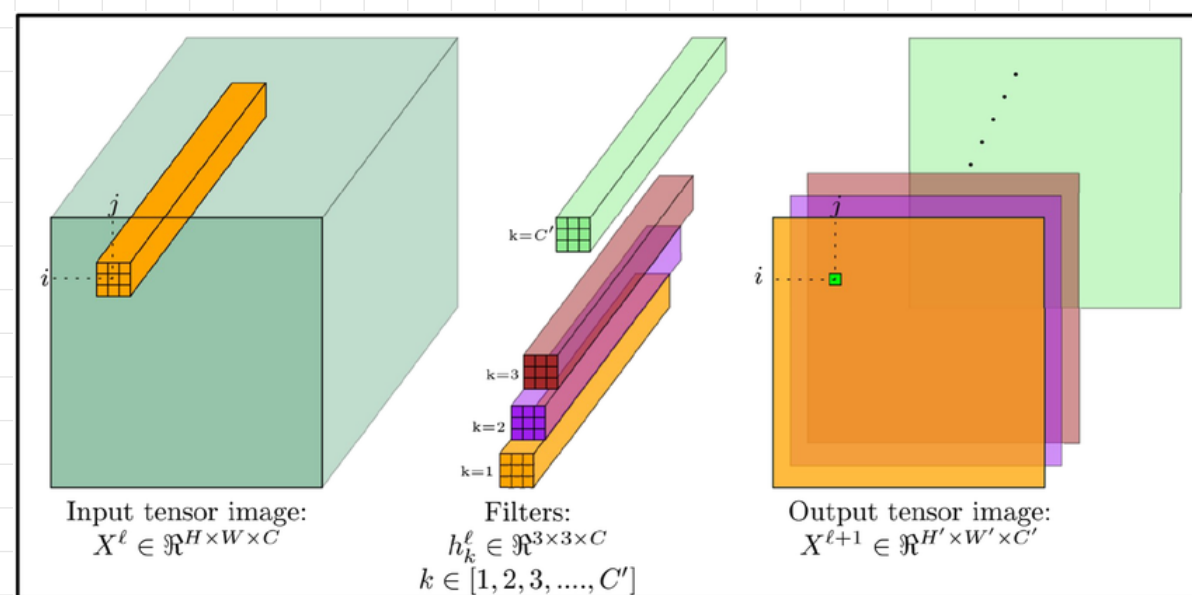
→ 87M 용량 가능 계층

(다음과 같은 다음과 같이 계산)

ex) 224x224x3
 → 96x11x11x3
 ⇒ 96x(11x11x3) + 96(bias) ≈ 34k
 fc) 4096 x 4096 + 4096 bias

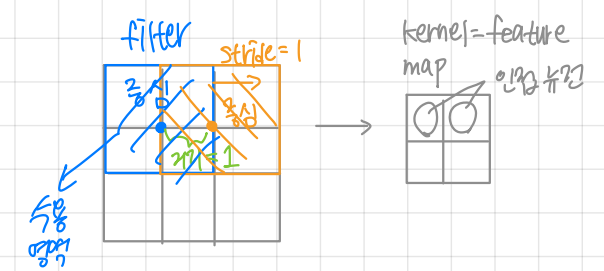


→ 7차원 텐서 이미지에
 → 7차원 평면들 조합해서
 featuremap 생성
 → 이어져서 96개라는 96개
 평면으로 96개 f.m 생성



→

커널맵에서 인접 두칸 간
각 수용 영역의 중심간 거리
= 커널 매수
||
커널맵 각 원소가 뉴런이긴



출력 차원

150528
253440

186624

64896

64896

43264

Layer	img/ f.m size	kernel-size	out-channels	stride	padding	
Input	224x224x3					
conv1	55x55x96	11x11x3	96	4	2	+LRN +RELU
max pooling	27x27x96	3x3x96		2		+RELU
conv2	27x27x256	5x5x96	256	1	2	+LRN +RELU
max pooling	13x13x256	3x3x256		2		+RELU
conv3	13x13x384	3x3x256	384	1	1	+RELU
conv4	13x13x384	3x3x384	384	1	1	+RELU
conv5	13x13x256	3x3x384	256	1	1	+RELU
max pooling	6x6x256	3x3x256		2		+RELU

$$\frac{224 + 2 \times 2 - 11}{2} + 1 = [55.25] = 55$$

$$\frac{55 - 3}{2} + 1 = 27$$

$$\frac{27 + 2 \times 2 - 5}{2} + 1 = 27 \quad (\text{원래는 GPU 27만 있어서 } 128/48)$$

$$\frac{27 - 3}{2} + 1 = 13$$

$$\frac{13 + 2 \times 1 - 3}{2} + 1 = 13$$

$$\frac{13 + 2 \times 1 - 3}{2} + 1 = 13 \quad 192/192$$

$$\frac{13 + 2 \times 1 - 3}{2} + 1 = 13 \quad 128/192$$

$$\frac{13 - 3}{2} + 1 = 6$$

flatten 6x6x256=9216

4096 fc1 9216 → 4096 + dropout

4096 fc2 4096 → 4096 + dropout

num_classes fc3 4096 → num_classes

→ 입력 차원 : 224x224x3=150528

2. RELU (반전경량) : $\max(0, x)$

→ 경사하강법 시 $\tanh, \text{sigmoid}$ 등 포화 비선형은 속로 느낌 // \oplus $|\tanh(x)|$ but, 2차원 방식 해결은 불
가능. 3차원 방식은 X

(GPU)

2. Local Response Normalization (국소응답 정규화) : LRN // ⊕ LCN (3차원 정규화)

→ RELU는 포화방식을 위한 입력정규화 필요X (positive 입력이면 학습됨)

→ ReLU \bar{x}_i 에 local normalization 적용하면 일반화 성능 ↑

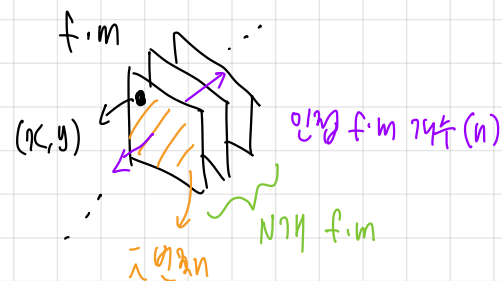
$$b_{x,y}^i = a_{x,y}^i \left(\left(\frac{1}{K} + \Delta \right) \sum_{j=\max(0, i-n/2)}^{\min(\sqrt{N}-1, i+n/2)} (a_{x,y}^j)^2 \right)^{\Delta}$$

Δ : hyper-parameter

$K=2$
 $n=5$ non

ReLU 후의 N 개 f.m 중 λ 번째
f.m i 의 위치 (x, y)

\Rightarrow top 1 1.4% ↓
top 5 1.2% ↓



4. Reducing overfitting (과적합 감소)

① label-preserving transformation 27/21

1. translation + horizontal reflection

$$256 \times 256$$

↓

224 x 224 112 | 224 | 224

(transforms, RandomCrop(224))

→ test 시기에 사료: 각 원본당 224×224 픽셀 5개 + 각 라우본전 \Rightarrow 각 층 10개 픽셀

$$\begin{aligned} & \Sigma 172 | 474 \\ & + \frac{2}{6} \text{ ov } 174 \end{aligned}$$

각 패치에 대한 softmax 후 평균

4. overlapping pooling (중첩 풀링)

→ Pooling : 중간 피쳐맵 (커널 맵) 의 모든 뉴런들이 같은 인접 뉴런들의 출력을 받아,
= 같은 필터에 만 피쳐맵

보통 풀림이 발생하는 영역이 적지(지) 않음
stride \ll $\frac{K(\text{원본 크기})}{\text{한 번에 보는 크기}}$ \ll $\frac{K}{K \times B}$

→ $z_m: S < z$
 $\quad \quad \quad \parallel \quad \parallel$
 $\quad \quad \quad 2 \quad 3$ $\begin{matrix} \text{MM} \\ 2 \ 0 \end{matrix}$

↳ $s=2, z=2$ (비중심) 일 때와 동일하기 동일하되,

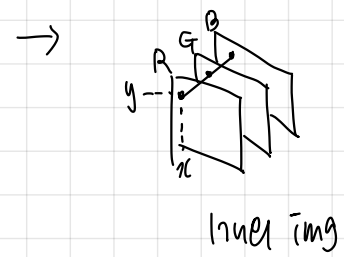
top 1 0.4% ↓

top 5 0.3% ↓

과거 20년 ↓

2. RGB 채널 강도 조정 \rightarrow top 1 1% 감소

\hookrightarrow 문제 정체성 (identity) 이 조정 강도에 색상 변화에 불변



$$\Rightarrow I_{xy} = \begin{bmatrix} I_{xy}^R \\ I_{xy}^G \\ I_{xy}^B \end{bmatrix}$$

가 224×224 개 픽셀 (1개 img당) $\rightarrow N$ 개 훈련 이미지

$$\Rightarrow (50/16 \times N) \times 3 \quad (2차원 행렬)$$



N 개 ; (계속 50/16 두리) 50/16 개씩 이미 붙임)

\downarrow 공분산 행렬

- 1) 각 채널의 평균 μ_R, μ_G, μ_B 구하고 각 픽셀 값들에서 빼주기 $\Rightarrow X$ 행렬
- 2) $\frac{1}{N} X^T \cdot X = [3 \times (50/16 \times N)] \times [(50/16 \times N) \times 3] = 3 \times 3$ 공분산 행렬 $= \Sigma$
- 3) PCA 수행: P (고유벡터), λ (고유값)

$\hookrightarrow \Sigma P = \lambda P$ 수행 ($\det(\Sigma - \lambda I) = 0$ 구하면 $\lambda_1, \lambda_2, \lambda_3$ 나옴 $\Sigma P = \lambda P$ P 구하기)

\downarrow

$$[P_1 \ P_2 \ P_3] \begin{bmatrix} \alpha_1 \lambda_1 \\ \alpha_2 \lambda_2 \\ \alpha_3 \lambda_3 \end{bmatrix} \text{의 값을 모든 각 픽셀에 대입}$$

\downarrow

$$\alpha = \frac{\text{가우시안 분포에서 뽑은 변수}}{\mu=0, \sigma=0.1}$$

② dropout: 훈련할 시 일부 뉴런 끄기 (역전파도 당연히 사용 x)

\rightarrow 서로 뉴런끼리 지닌 의존 방지, 매번 다른 구조의 (서브) 네트워크 = 앙상블 (즉, 가중치는 같은 연결이지만 다른 네트워크여도 공유) / 즉, 수렴까지 반복 횟수 2배 \uparrow

\rightarrow test: 모든 뉴런을 켜고, 출력값에 $\times 0.5 \rightarrow$ 앙상블의 (기하) 평균 효과 내기 위해서

5. Details of learning

1. Optimizer: SGD

→ `optim.SGD(model.parameters(), lr, momentum, weight_decay)`

→ momentum: 0.9 / weight_decay: 0.0005

↳ "관성": GD의 가파른

경사에서 진동하는 문제

해결하기 위해 이전 기울기의

이동 방향 기억해서 현재

기울기에 대해서 업데이트

(이전 방향으로 계속해서)

↳ 과적합 막기 위한 Regularization (weight 커지는 것 막기)

: $\text{loss}(y, \hat{y}) + \lambda \sum_{\tilde{n}} w_{\tilde{n}}^2$ (loss 줄이는데 목적이냐)

(training errors 줄여줌)

⇒ 최종 weight update 규칙: $w_{\tilde{n}+1} = w_{\tilde{n}} + u_{\tilde{n}+1}$

$$= 0.9 \cdot u_{\tilde{n}} - 0.0005 \cdot \epsilon \cdot w_{\tilde{n}} - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} \middle|_{w_{\tilde{n}}} \right\rangle_{D_{\tilde{n}}}$$

↳ \tilde{n} : iteration

$u_{\tilde{n}}$: momentum 변수

ϵ : lr

$D_{\tilde{n}}$: \tilde{n} 번째 미니배치

$E(x, y) \in \left\langle \frac{\partial L}{\partial w} \middle|_{w_{\tilde{n}}} \right\rangle_{D_{\tilde{n}}}$: 해당 배치에 대한 손실함수 기울기

↳ 가중치 초기화: $\mu=0, \sigma=0.01$ (가우시안)

→ bias: conv 2, 4, 5 + fc 1, 2, 3 모두 1로 초기화해서 $y = w^T x + b$ 의

값이 음수가 되거나 $\text{ReLU}=0$ 이 되는 문제 (ReLU 미분 $x > 0 \rightarrow 1$
 $x < 0 \rightarrow 0$)를 막음

→ 학습 초기에 수렴 가속

(배치 평균: 0)

2. batch size: 128 / epochs = 90 / lr = 0.01 ($\frac{1}{10}$ 씩 감소, 3번 감소)

6. Results

7. Discussion

→ depth 가 성능 탐색에 핵심