1. VGGNet 아키텍처
   ① Input 이미지
      (1) 채널 개수: RGB (3채널)
      (2) 이미지 사이즈: 224 x 224
      (3) 전처리: 평균 제거
   ② Convolution Layers
      (1) 커널 사이즈: 3 x 3
      (2) 스트라이드: 1
      (3) 패딩: 1
      (4) 활성화 함수: ReLU
   ③ Pooling Layers
      (1) 방법: Max Pooling
      (2) 커널 사이즈: 2 x 2
      (3) 스트라이드: 2
   ④ Fully Connected Layers
      (1) 입력층과 은닉층의 출력 크기: 4096
      (2) 입력층과 은닉층의 활성화 함수: ReLU
      (3) 출력층의 출력 크기: 클래스 개수와 동일
      (4) 출력층의 활성화 함수: Softmax
      (5) 입력층과 은닉층에 Dropout 적용 (p = 0.5)

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					

   ⑤ 하이퍼파라미터 및 모델 구성 요소
      (1) 벤치마크 데이터셋: ILSVRC-2012 dataset
      (2) 배치 사이즈: ~~256~~ 128
      (3) 학습률: ~~(최초) 0.01~~ 0.001
      (4) 에폭 수: 조기 멈춤 (Validation Dataset 기준)
      (5) 손실 함수: Cross Entropy Loss (= Multinomial Logistic Regression Objective)
      (6) 옵티마이저: Momentum (0.9) + Scheduler (0.1)
      (7) 가중치 초기화: ~~정규분포(평균 0 분산 0.001) + 전이학습~~ Pytorch 디폴트 초기화
      (8) 바이어스 초기화: 0
      (9) 오버피팅 방지: L2 정칙화 (λ = 0.0005), 조기 멈춤

⑥ 예외

(1) VGGNet(A-LRN): 첫 번째 Convolution Layers에 Local Response Normalization 적용
AlexNet 논문의 영향을 받은 부분
한계: 본 논문에서는 좋지 않은 성능, 모델의 복잡도 증가

(2) VGGNet(C): 3, 4, 5번째 Convolution Layers의 마지막에서 1 x 1 커널 사이즈 적용 (패딩 X)
NIN(Network In Network) 논문의 영향을 받은 부분
NIN Topic: 1x1 컨볼루션을 사용하면 공간 정보를 줄이지 않고도 모델의 표현력을 높일 수 있지 않을까?
※ 표현력: 이미지에서 다양한 표현을 추출할 수 있는 능력
ⓐ 학습해야 할 가중치 개수 감소
ⓑ 비선형성 강화
ⓒ 채널 간 관계를 학습하는 데 강력
ⓓ 한계: 본 논문에서는 좋지 않은 성능, 픽셀 간 관계를 학습하는 데 어려움

2. VGGNet 아키텍처 성질

① Convolution에서 1 패딩과 3x3 커널의 조합 ==> 인풋과 아웃풋의 사이즈가 동일

② Pooling에서 2x2 커널과 2 스트라이드의 조합 ==> Convolution Layers의 아웃풋을 절반으로 다운스케일
채널의 개수가 2배씩 증가 (64 -> 128 -> 256 -> 512)

③ 모든 모델의 FC층이 서로 동일함 (512*7*7 -> 4096 -> 4096 -> C)

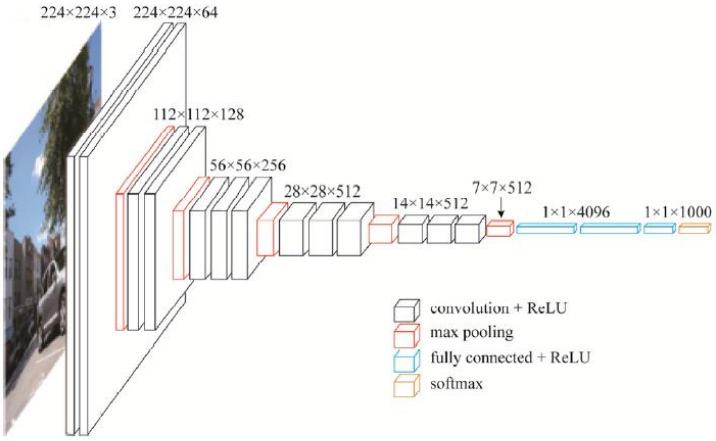


그림 4-22 VGGNet 구조[Simonyan2015]

3. 너비 vs 깊이

① 수용장

(1) 정의: 출력 레이어의 뉴런 하나에 영향을 미치는 입력 레이어의 공간 크기

(2) 의의: 아키텍처 설계시 분류기(FC층)가 넓은 수용장을 가지게끔 설계

ⓐ 좁은 수용장: 저급 특징 추출

ⓑ 넓은 수용장: 고급 특징 추출

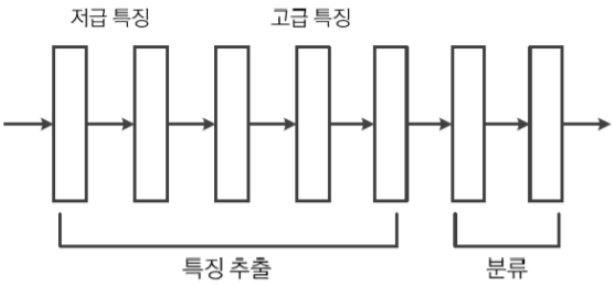
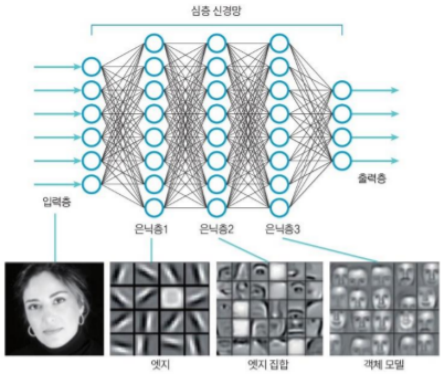


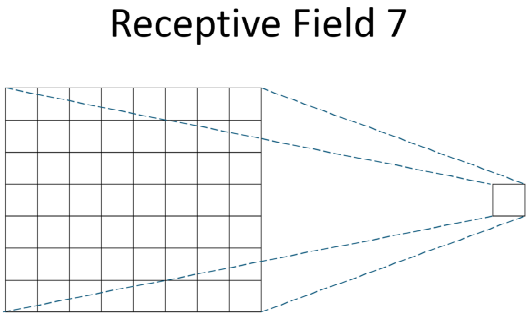
그림 4-2 깊은 신경망의 처리 절차

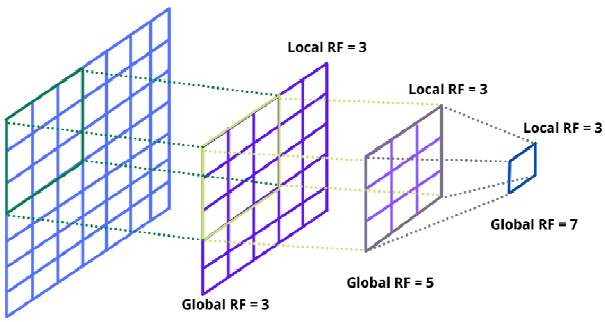


② 수용장을 늘이는 방법

(1) 더 넓게 설계 (큰 커널 사이즈 적용)

(2) 더 깊게 설계 (더 많은 레이어 적용)





③ 깊게 설계하는 것이 더 낫다

(1) 가중치 개수 감소

ⓐ 넓게 설계한 모델: 가중치 개수 $(7C)^2$ 개

ⓑ 깊게 설계한 모델: 가중치 개수 $3(3C)^2$ 개

(2) 더 많은 활성화 함수를 통과함으로써 모델의 표현력 증가

ⓐ 넓게 설계한 모델: ReLU 한 번 통과

ⓑ 깊게 설계한 모델: ReLU 세 번 통과

4. 실험 결과

① 층을 깊게 쌓을수록 성능이 높아짐

② Multi Scaling 사용이 바람직

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**

5. VGGNet 논문 의의
   ① 논문이 발표된 시점 ImageNet 최고 성능 모델 (SOTA)
   ② 간단한 구조를 이용하여 깊은 아키텍처의 중요성을 입증
   ③ 이후 모델들이 3x3 Kernel을 사용하는데 일조

6. VGGNet 및 논문 한계
   ① 아키텍처가 깊어질수록 기울기 소실 문제 발생
   ② 무거운 용량의 모델 (이미지 하나를 처리하는데 최소 500MB 용량의 메모리 필요)
   ③ 가중치 초기화 방식을 구체적으로 언급하지 않음

7. 후속 연구: ResNet(2015)
   ① VGGNet-19 기반
   ② 기울기 소실 문제 해결: Skip Connection
   ③ 무거운 용량 문제 해결: 분류기의 기반을 Fully Connected Layers에서 Global Average Pooling으로 변경