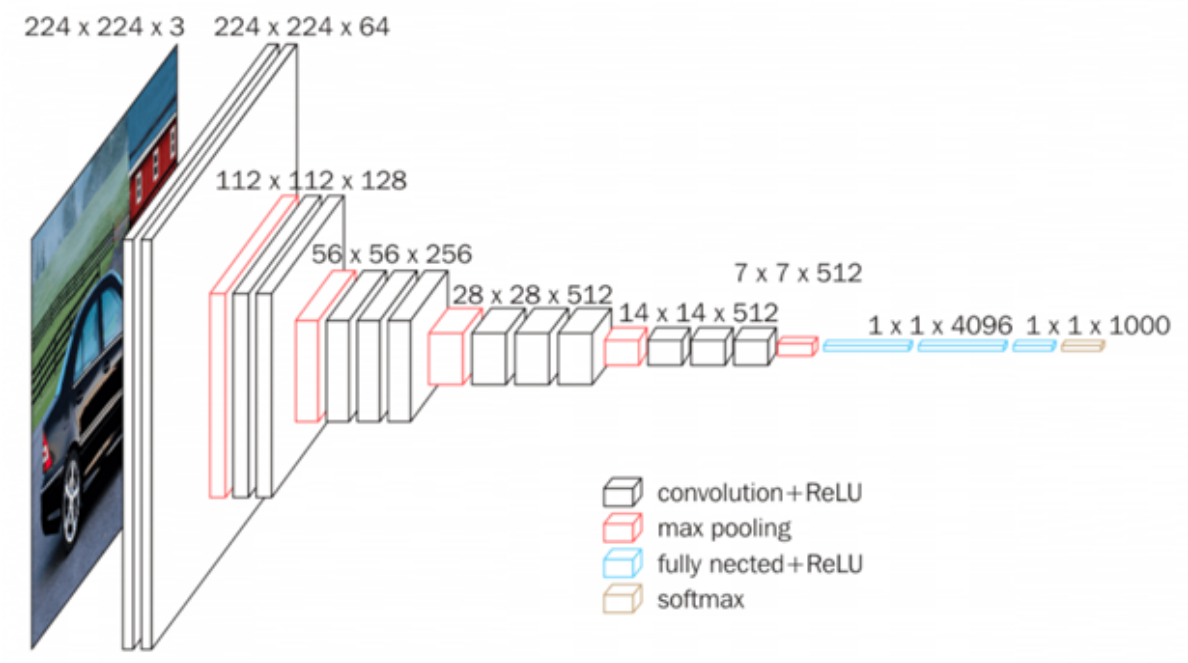
Very Deep Convolutional Networks For Large-Scale Image Recognition (ICLR 2015)

- VGGNet 이라는 모델로 유명한 논문 (paper)

VGG Structure



출처 : https://medium.com/@ommore524/vgg-16-convolution-neural-network-bae747a7494a

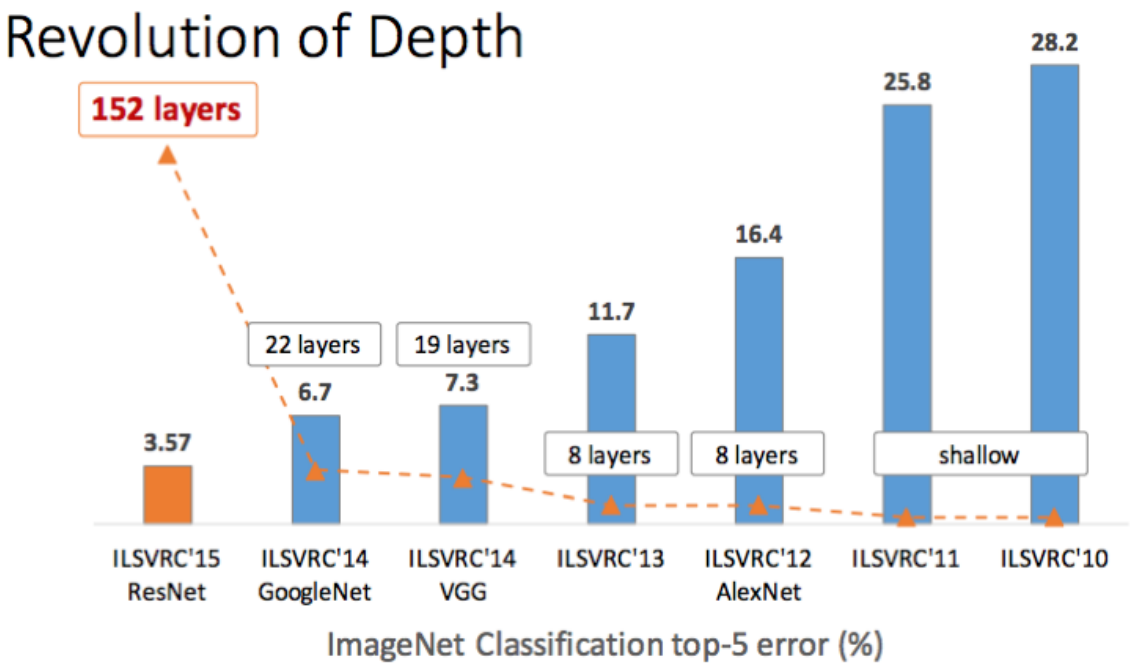
Introduction

- 해당 논문에서는, ConvNet architecture 설계에서 네트워크의 깊이 (depth)를 깊게 만드는 것이 성능에 어떠한 영향을 끼치는 지에 대한 실험을 진행하였으며,
  어떠한 방식으로 depth 를 깊게 만들 수 있었는 지에 대하여 설명함

- 옥스퍼드 대학의 연구팀 VGG (Visual Geometry Group)에 의해 개발되어
  VGGNet 이라는 이름을 갖게 되었으며, layers 의 수에 따라 VGG16 (16 layers), VGG19 (19 layers) 로 불린다
- ILSVRC 2014에서 2등을 차지하며 좋은 성능을 입증 (GoogLeNet 이 해당 대회 1등)
  - 사용하기 쉬운 구조와 좋은 성능 덕분에,
    그 대회에서 우승을 거둔 조금 더 복잡한 형태의 GoogLeNet보다 더 인기를 얻음

- 기존의 AlexNet (2012) 의 8-layers 모델보다 깊이가 2배 이상 깊은 네트워크 학습에 성공하였으며, ImageNet Challenge 에서 오차율을 절반으로 줄임

- 역사적으로 VGGNet 부터 네트워크 depth 가 크게 깊어지게되는 의의를 가짐
(ILSVRC' 2015 ResNet 은 152 layers)



출처 : *https://medium.com/@Lidinwise/the-revolution-of-depth-facf174924f5*

- cf) Top-5 error (rank-5 error)
  - Classification 논문에서 자주 사용하는 지표 중 하나 (top-1 error, top-5 error 등)
  - confidence score 상위 5개 class 안에 실제 class 가 존재하지 않는 오류
  - 참고 자료

VGGNet 의 핵심 구조

3 X 3 Conv filter

- Model 의 depth 의 영향만을 확인하기 위해 가장 작은 filter size 인 3x3 conv filter 를 사용

- 이 논문에서는 ConvNet 의 depth 를 증가시키기위해 더 많은 convolutional layer 를 추가하였고, 이것은 모든 layers에 매우 작은 3x3 conv filters 를 사용하였기에 가능하다
  - filter 사이즈가 크면 layer 를 거칠 때마다 image 의 사이즈가 많이 축소되기에, 네트워크의 depth 를 깊게 만들기 어렵다

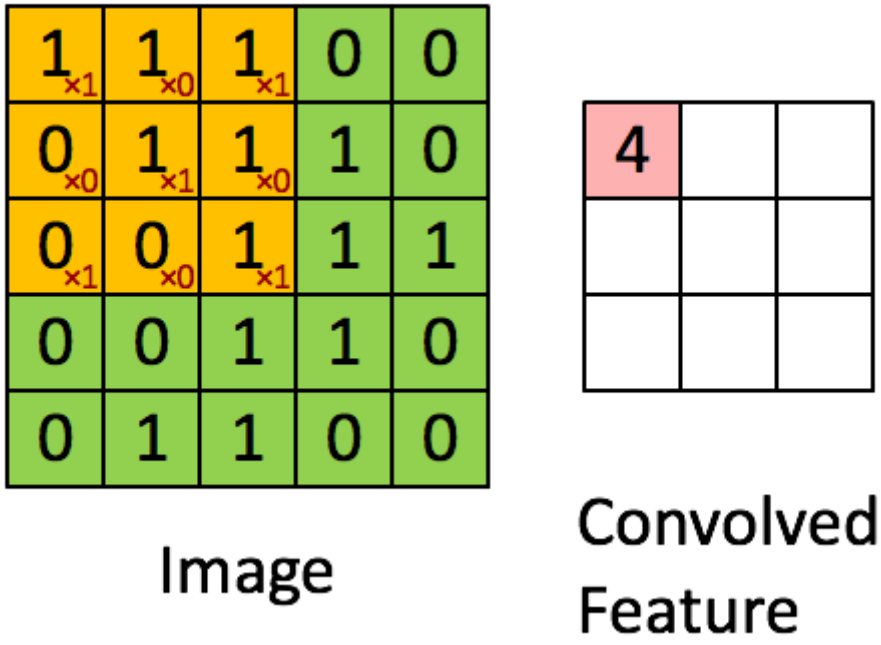
- 3x3 filter 를 이용해 3 번의 Convolution 필터링을 반복한 특징맵은
  7x7 fileter 의 **Receptive field** 와 동일한 효과를 볼 수 있다

3 x3 conv filter 사용의 장점

- (1) 3x3 filter 를 사용하게되면 레이어가 증가하고
  비선형성이 증가하게되며 **모델 특징 식별성 증가**로 이어진다는 장점이 있다
- (2) **파라미터 수를 감소**시킬 수 있다는 장점이 있다
  (7x7 필터 1개는 49개, 3x3 필터 3개는 27개)
  - 파라미터 수가 감소되면 학습의 속도가 빨라진다

- VGG 이전의 좋은 성과를 보이던 Model 들은
  비교적 큰 Receptive Field 인 7x7, 11x11 필터를 포함하였던 것과 차이가 있음

- Convolution filter 연산 적용 예시

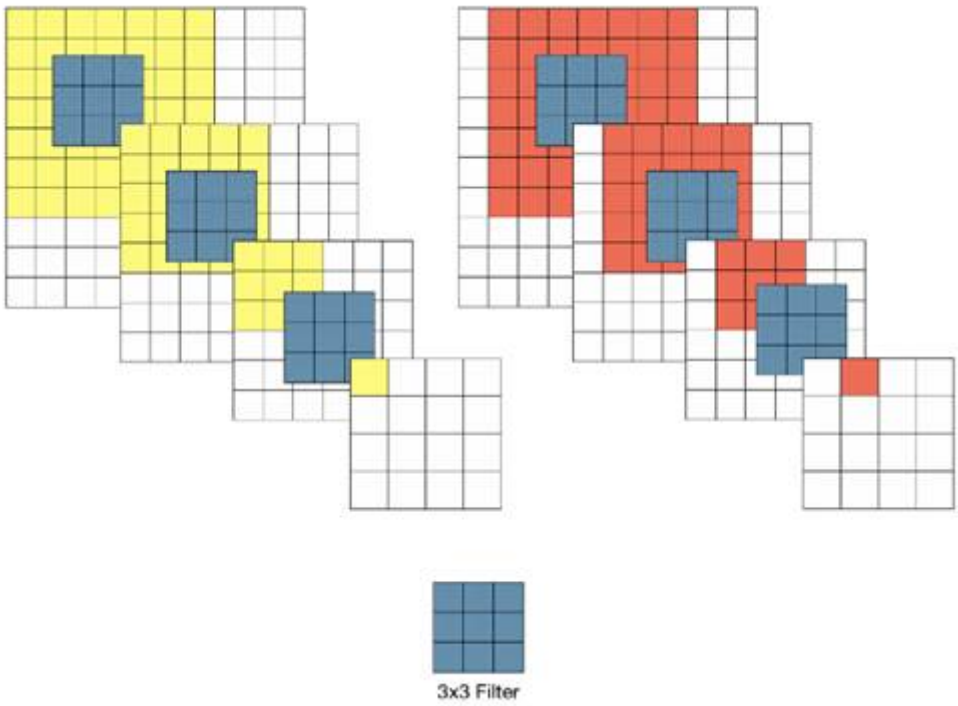


출처 : https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

3x3 filter Vs. 7x7 filter

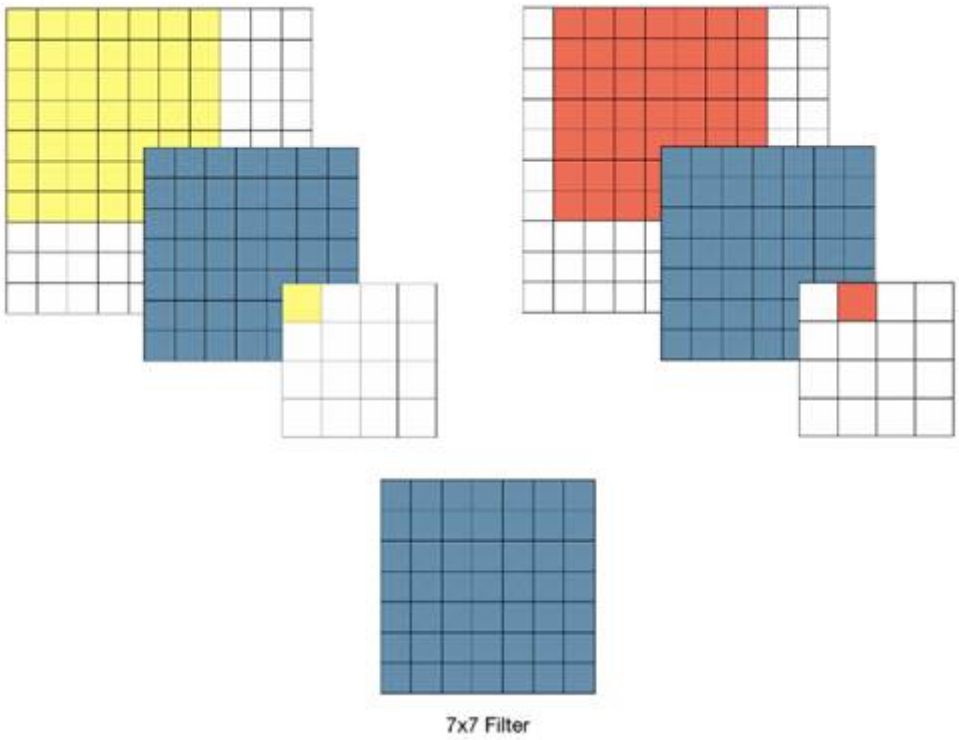
- 3 x 3 Filter 3번 적용
  - 1st 적용 : 10 x 10 -> 8 x 8
    - 10 - 3 + 1 = 8
  - 2nd 적용 : 8 x 8 -> 6 x 6
    - 8 - 3 + 1 = 6
  - 3rd 적용 : 6 x 6 -> **4 x 4**

    - 6 - 3 + 1 = 4



출처 : https://medium.com/@msmapark2/vgg16-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a

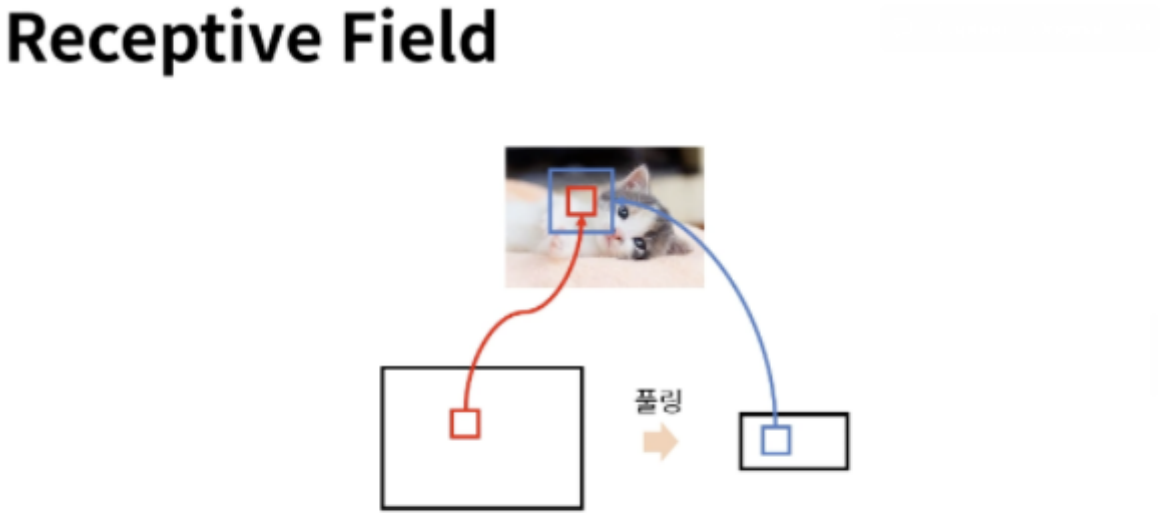
- 7 x 7 Filter 1번 적용
    - 10 x 10 -> **4 x 4**
        - 10 - 7 + 1 = 4



출처 : https://medium.com/@msmapark2/vgg16-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a

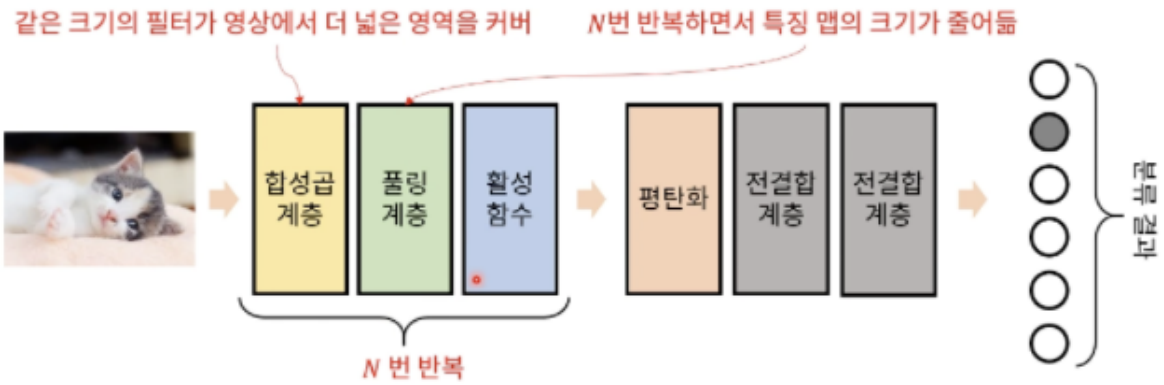
cf) Receptive Field

- 출력에 영향을 미치는 입력 뉴런의 공간 크기를 **Receptive Field** 라고 한다

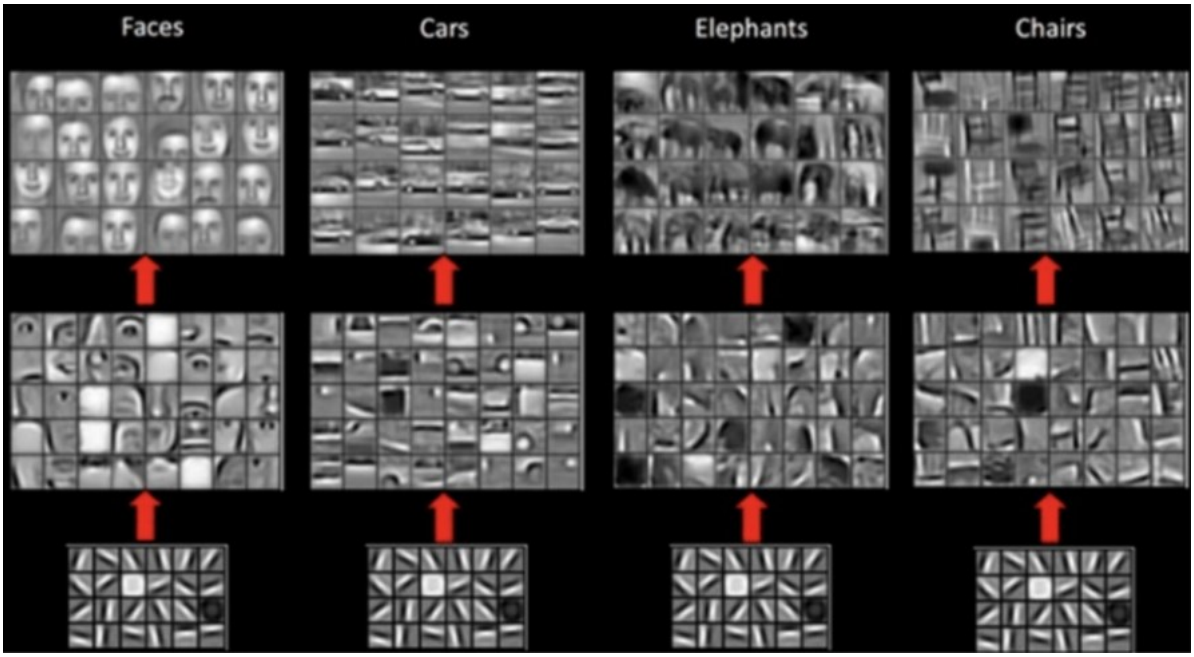


같은 크기의 필터여도, 풀링에 의해 작아진 특징 맵에 적용되면 원본 영상에서 차지하는 범위가 넓다.
이 범위를 Receptive Field라고 한다.

- 풀링(pooling) 계층을 통해 크기가 작아진 특징 맵 (Feature Map)에,
  합성곱 계층의 필터를 적용시키게 되면 더 넓은 영역을 커버할 수 있다

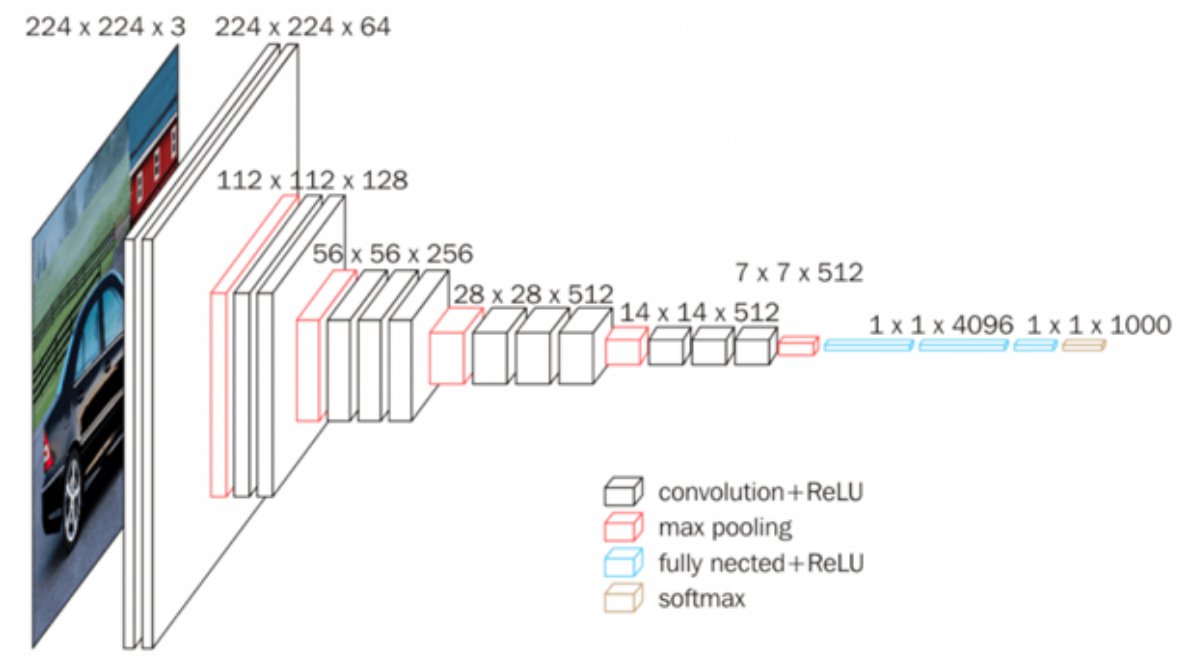


- 계층적으로 학습이 진행될 수록 (초반 계층 -> 후반 계층),
  좁은 부분의 Feature 에서 전체적인 형상을 만들어가는 모습을 보여준다



출처 : https://www.topbots.com/advanced-topics-deep-convolutional-neural-networks/

Model 아키텍쳐



출처 : https://bskyvision.com/504

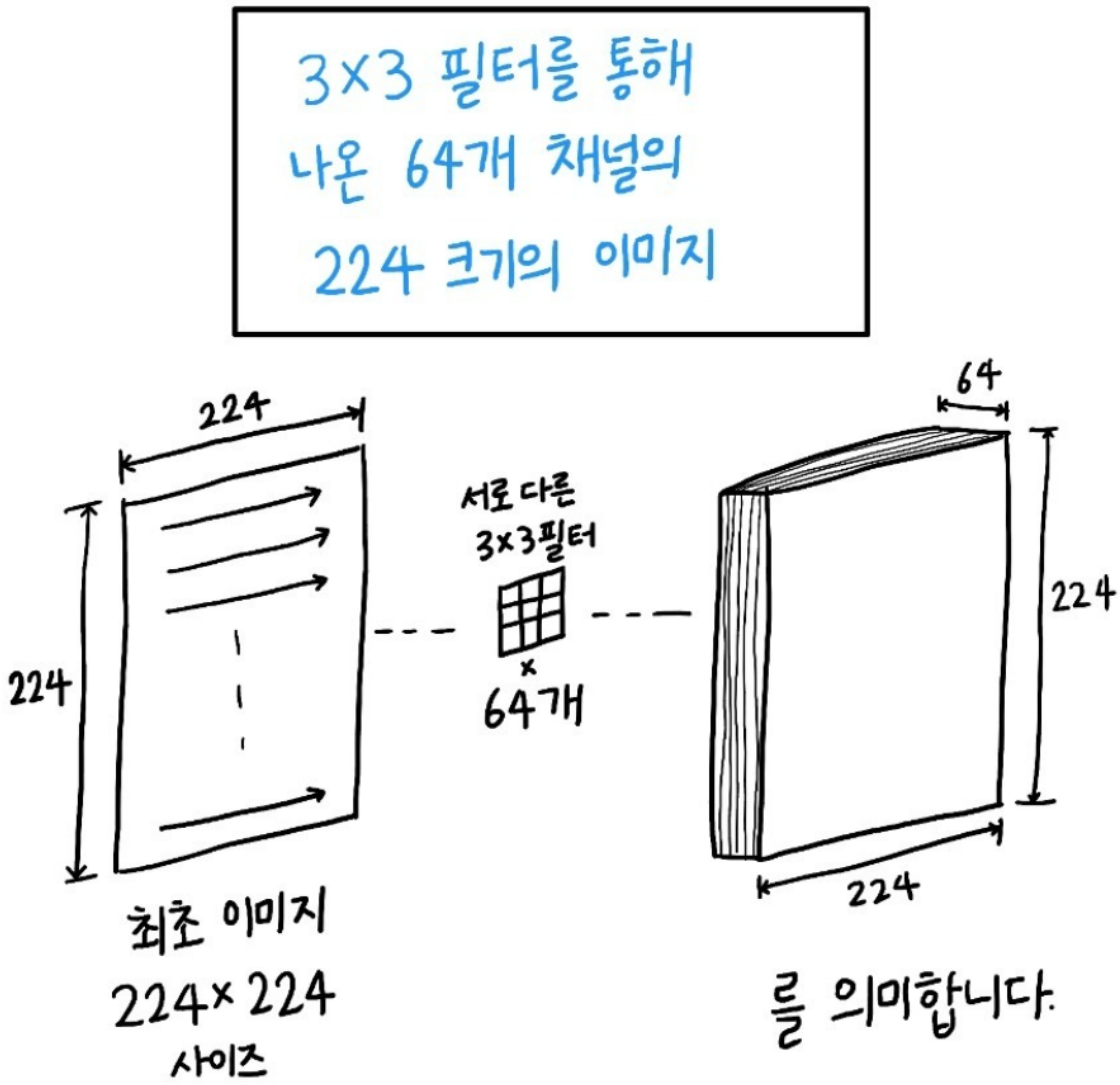
- 단순화 version

출처 : https://brunch.co.kr/@hvnpoet/126

- **0) Input (입력)**
  - 224 x 224 x 3 이미지 (224 x 224 RGB 이미지)를 입력

- **1) 1층 (conv 1_1)**
  - 64개의 3 x 3 x 3 필터커널로 입력 이미지를 convolution 연산
  - zero padding은 1 로 설정, 컨볼루션 보폭(stride)는 1로 설정
    - zero padding과 컨볼루션 stride에 대한 설정은 모든 컨볼루션 층에서 모두 동일
    - 이에, convolution 연산 이후에도 output shape 가 유지된다

  - output 으로 64장의 224 x 224 특성맵(224 x 224 x 64) 생성
  - 활성화 함수로 ReLU 함수 적용
    - ReLU 함수는 마지막 16층을 제외하고는 항상 적용



출처 : https://brunch.co.kr/@hvnpoet/126

- 3 x 3 필터의 갯수가 더 많아짐을 의미
  - 채널의 깊이가 깊어짐을 의미

- **2) 2층 (conv 1_2)**
  - 64개의 3 x 3 x 64 필터커널로 특성맵을 convolution 연산
  - 결과적으로 64장의 224 x 224 특성맵들(224 x 224 x 64)이 생성
  - 다음으로, 2 x 2 최대 풀링 (max pooling layer) 을 stride 2로 적용하여, 특성맵 (feature map) 의 사이즈를 112 x 112 x 64로 줄인다

- **3) 3층 (conv 2_1)**
  - 128개의 3 x 3 x 64 필터 커널로 특성맵을 convolution 연산
  - 결과적으로 128장의 112 x 112 특성맵들 (112 x 112 x 128) 이 산출

- **4) 4층 (conv 2_2)**
  - 128개의 3 x 3 x 128 필터커널로 특성맵을 convolution 연산
  - 결과적으로 128장의 112 x 112 특성맵들 (112 x 112 x 128)이 산출

- 다음으로, 2 x 2 최대 풀링 (max pooling layer) 을 stride 2로 적용
- 특성맵의 사이즈가 56 x 56 x 128로 줄어든다

- **5) 5층 (conv 3_1)**
  - 256개의 3 x 3 x 128 필터커널로 특성맵을 convolution 연산
  - 결과적으로 256장의 56 x 56 특성맵들(56 x 56 x 256)이 생성

- **6) 6층(conv 3_2)**
  - 256개의 3 x 3 x 256 필터커널로 특성맵을 convolution 연산
  - 결과적으로 256장의 56 x 56 특성맵들(56 x 56 x 256)이 생성

- **7) 7층(conv 3_3)**
  - 256개의 3 x 3 x 256 필터커널로 특성맵을 convolut ion 연산
  - 결과적으로 256장의 56 x 56 특성맵들(56 x 56 x 256)이 생성
  - 그 다음에 2 x 2 최대 풀링을 stride 2로 적용
  - 특성맵의 사이즈가 28 x 28 x 256으로 줄어든다

- **8) 8층 (conv 4_1)**
  - 512개의 3 x 3 x 256 필터커널로 특성맵을 convolution 연산
  - 결과적으로 512장의 28 x 28 특성맵들(28 x 28 x 512)이 생성

- **9) 9층 (conv 4_2)**
  - 512개의 3 x 3 x 512 필터커널로 특성맵을 convolution 연산
  - 결과적으로 512장의 28 x 28 특성맵들(28 x 28 x 512)이 생성

- **10) 10층 (conv 4_3)**
  - 512개의 3 x 3 x 512 필터커널로 특성맵을 convolution 연산
  - 결과적으로 512장의 28 x 28 특성맵들(28 x 28 x 512)이 생성
  - 다음으로, 2 x 2 최대 풀링을 stride 2로 적용
  - 특성맵 의 사이즈가 14 x 14 x 512로 줄어든다

- **11) 11층(conv 5_1)**
  - 512개의 3 x 3 x 512 필터커널로 특성맵을 convolution 연산
  - 결과적으로 512장의 14 x 14 특성맵들(14 x 14 x 512)이 생성

- **12) 12층 (conv 5_2)**
  - 512개의 3 x 3 x 512 필터커널로 특성맵을 convolution 연산
  - 결과적으로 512장의 14 x 14 특성맵들(14 x 14 x 512)이 생성

- **13) 13층 (conv 5_3)**

- 512개의 3 x 3 x 512 필터커널로 특성맵을 convolution 연산
- 결과적으로 512장의 14 x 14 특성맵들(14 x 14 x 512)이 생성
- 다음으로, 2 x 2 최대 풀링을 stride 2로 적용
- 특성맵의 사이즈가 7 x 7 x 512로 줄어든다

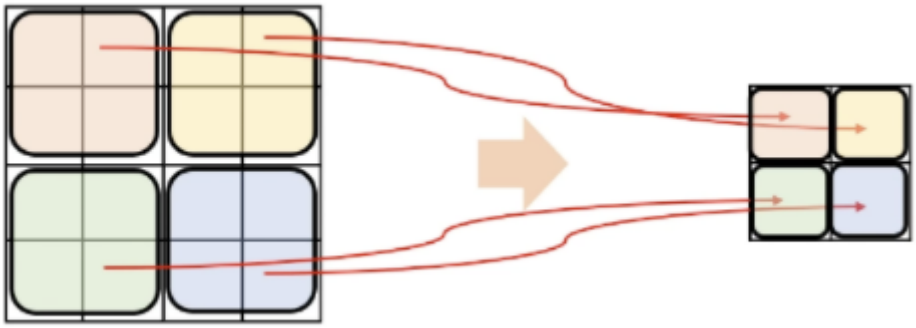
- **14) 14층 (fc1)**
  - 7 x 7 x 512의 특성맵을 flatten 해주는 layer
  - flatten은 전 층의 출력을 받아서 단순히 1차원의 벡터로 펼쳐주는 것을 의미
  - 결과적으로 7 x 7 x 512 = 25088개의 뉴런이 되고,
    fc1층의 4096개의 뉴런과 fully connected 된다

- **15) 15층 (fc2)**
  - 4096개의 뉴런으로 구성
  - fc1층의 4096개의 뉴런과 fully connected 된다

- **16) 16층 (fc3)**
  - 1000개의 뉴런으로 구성
  - fc2층의 4096개의 뉴런과 fully connected된다
  - 출력값들은 softmax 함수로 활성화
  - 1000개의 뉴런으로 구성되었다는 것은
    1000개의 클래스로 분류하는 목적으로 만들어진 네트워크라는 것을 의미한다
    - ImageNet Class 가 1000개이기 때문

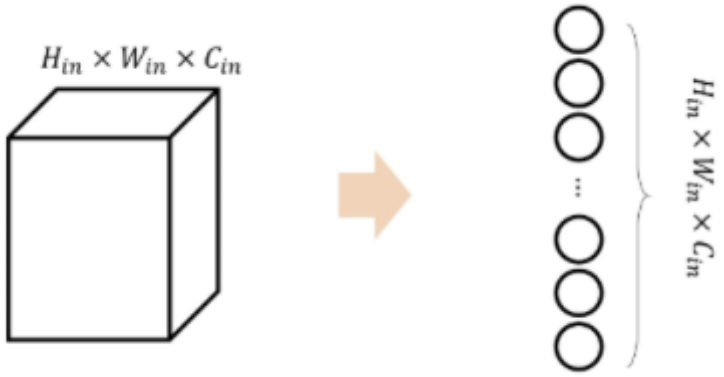
풀링 계층 (Pooling Layer)

- 정보의 종합을 위해, 영상의 크기를 줄이는 기능을 하는 Layer
- 여러 화소를 종합하여 하나의 화소로 변환하는 계층



평탄화 (Flatten)

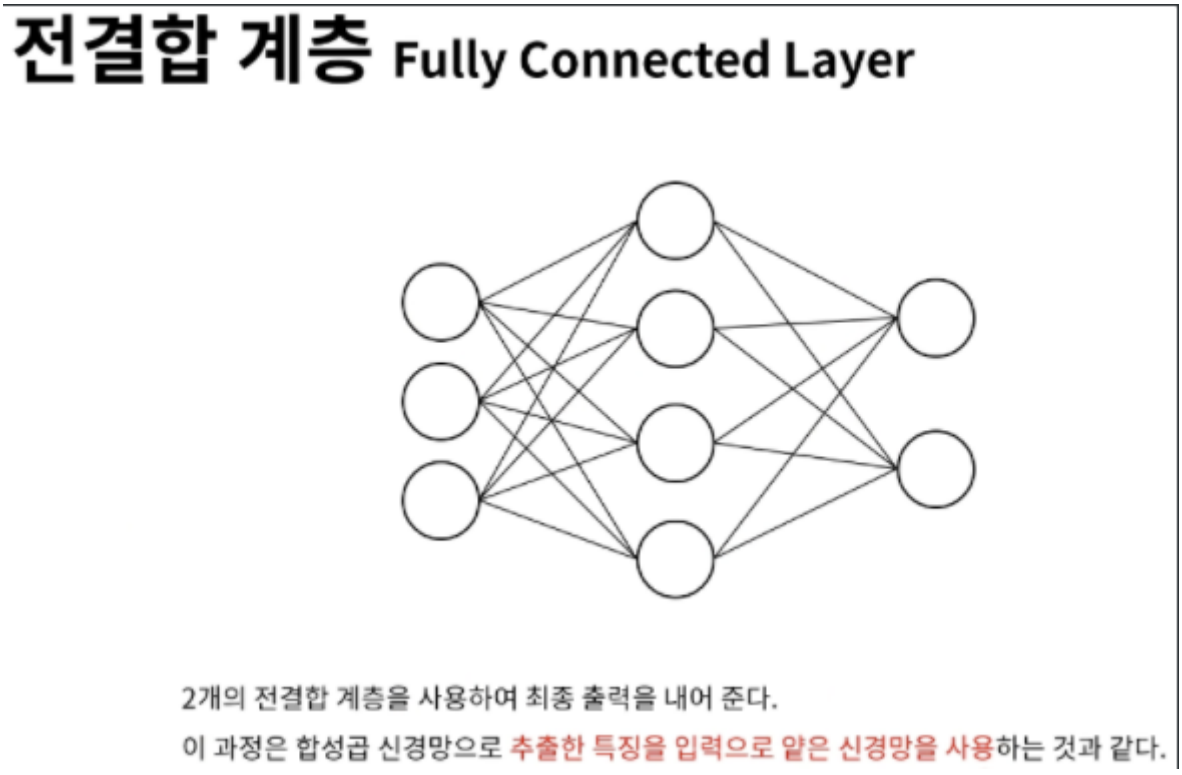
- 입력된 특징 맵 (Feature Map) 의 모든 화소를 나열하여 하나의 벡터로 만드는 것
- 합성곱 계층(Convolutional Layer)의 Output인 Feature Map 형태를 하나의 벡터로 펼쳐, 전결합 계층(Fully Connected Layer)으로 연결해주는 역할을 한다
    - 아무 연산도 일어나지 않으며, 합성곱 계층과 전결합 계층을 연결하는 역할을 수행



전결합 계층 (Fully Connected Layer)

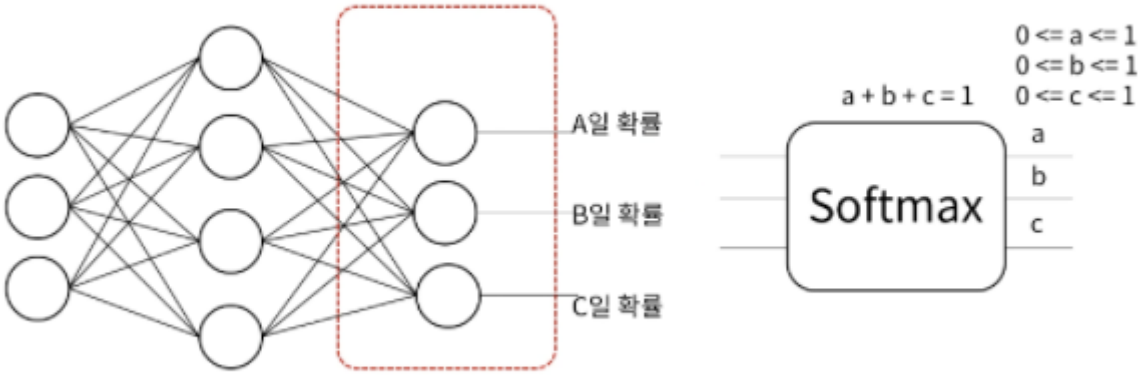
- 일반적으로, 2개의 전결합 계층을 사용하여 최종 출력을 내어준다

- 합성곱 신경망으로 추출한 특징을 입력으로 얕은 신경망을 사용하는 것과 같다 (기본 MLP 구조)



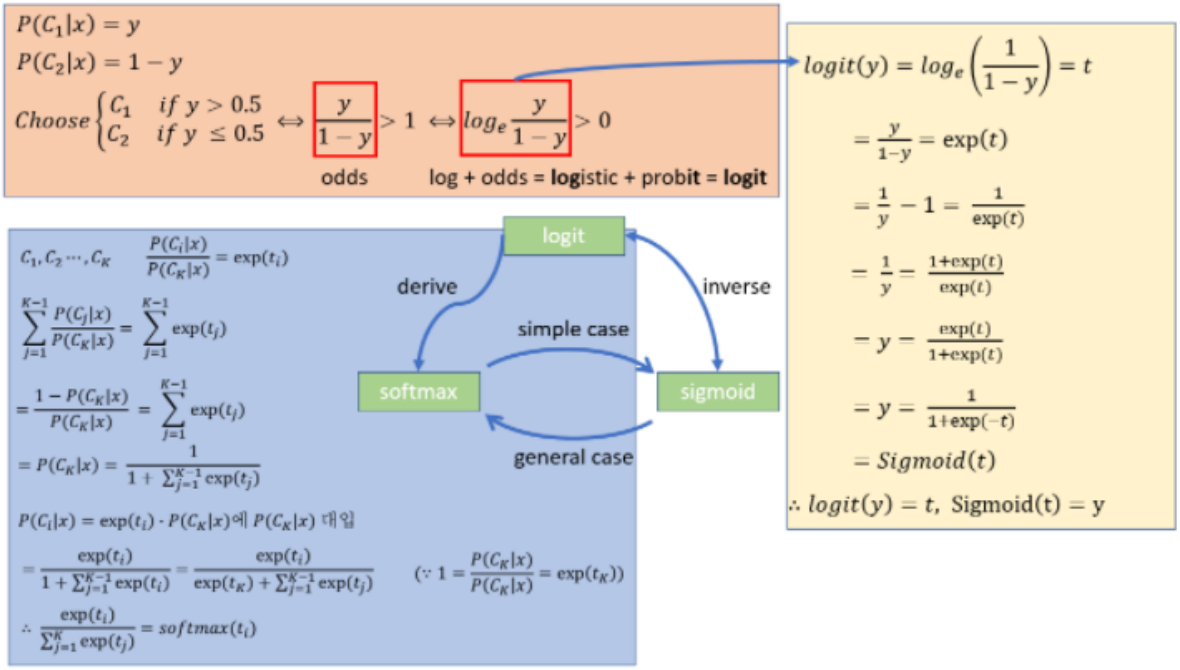
- 마지막 출력 Layer 의 Activation 에 따라서 역할 (Task)이 달라지게 되며,
  CNN 을 통해 Classification 을 수행하는 경우, Softmax 함수를 활성화 함수로 사용한다

Softmax 함수



다중 클래스 분류 문제를 해결하기 위해서 마지막 계층에는 Softmax 활성 함수를 사용한다.

- cf) Sigmoid Vs. Softmax



Experiments

ConvNet configurations

- 총 6개의 구조 (A, A-LRN, B, C, D, E) 를 만들어 성능을 비교
  - 깊이에 따른 성능 변화를 비교하기 위함
  - layer 가 16 개인 D 구조가 VGG16, layer 가 19 개인 E 구조가 VGG19

- DropOut
  - 최초 두 개의 FCL에 0.5 비율로 적용
- Learning Rate
  - 최초 0.01에서 시작해 정확도 향상이 멈춤에 따라 감소시킴
- Initialization
  - VGG-11을 적당히 학습시킨 값을 가지고
    최초 4장의 Conv Layer와 3장의 FCL에 초기화 목적으로 가중치를 적용,
    나머지 레이어는 Weight는 평균 0과 분산 0.01의 랜덤값으로, Bias는 0으로 설정

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					

출처 : *https://arxiv.org/pdf/1409.1556.pdf*

1x1 conv filter

- 1x1 CNN layer의 경우, “Network in Network (NiN)”에 영향을 받아 사용한 것으로,
  비선형성을 늘리면서도 input으로 들어오는 정보는 최대한 건드리지 않기 위해 사용한 것

- Paper 설명

> The incorporation 1x1 conv layers is a way to increase the nonlinearity of the decision function without affecting the receptive fields of the conv layers.

Training

- 네트워크의 Input이 고정값을 받기 때문에 이미지의 크기를 조정한 뒤
  랜덤한 범위를 224x224로 잘라서 학습시키는 방법을 선택
    - 이미지 크기를 가로, 세로 비율을 유지한 채로 조정할 때, S는 조정된 이미지의 짧은 변의 값
- 네트워크는 Single-scale과 Multi-scale 의 두 가지 방식으로 학습됨

- Single-scale
    - S=256으로 모든 이미지를 조정하고 224x224의 고정 크기로 일부를 잘라 0.01의 학습률로 학습시키고,
      추가로 조금 더 유연성을 부여하기 위해 S=384 환경에 0.001의 학습률을 적용해 학습시켰다
- Multi-scale
    - 다양한 크기의 이미지를 학습하는 모델을 만들기 위해 고려한 방법으로,
      이미지마다 범위 256~512의 S값으로 크기를 조정하고 고정 크기로 잘라 학습시키는 방법을 적용
    - 이 방식은 Scale Jittering이라고 부를 수 있으며,
      학습을 반복함에 따라 하나의 이미지를 다양한 스케일로 학습하게 되므로
      일종의 **Data Augmentation** 역할을 한다

Test

- 두 가지 방식으로 Evaluation 을 진행

- Dense Evaluation
    - 첫번째 Fully-Connected Layer를 7x7 Conv Layer로 변환시킨 뒤 원본 이미지를 투입하는 방식
    - AlexNet에서처럼 테스팅 시에 데이터를 Multiple Crop하는 과정을 제거할 수 있기 때문에
      효율적인 연산이 가능하다는 장점이 존재

- Multi-crop Evaluation
    - GoogLeNet에서 사용된 **Multi-crop Evaluation**도 적용해
      Dense Evaluation과 상호보완적인 시너지를 내게 만들었다

Result

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**

출처 : *https://arxiv.org/pdf/1409.1556.pdf*

- 깊이가 11층, 13층, 16층, 19층 으로 깊어지면서 분류 에러가 감소하는 것을 관찰함
  - 깊이가 깊어질수록 성능이 향상됨

- A와 A-LRN 구조의 성능을 비교했을 때, AlexNet 에서 사용되던 LRN (Local Response Normalization) 은
  성능 향상에 효과가 없는 것으로 확인되어 B, C, D, E 에서는 적용하지 않음
  - LRN 참고 자료

Further study

Experiments

- Training 과 Test (Evaluation) 의 실험 방법론에 대한 이해 부족
  - Training
    - Single-scale 과 Multi-scale 로 나누어서 학습을 시킨 이유가 무엇인지
  - Test
    - Dense Evaluation 과 Multi-crop Evaluation 이 정확히 무엇인지

Initialisation

- using the random initialisation procedure of Glorot & Bengio 부분
  - Xavier Initialization 을 의미하는 것 같다
    - Xavier *Glorot* (2010)
    - Understanding the difficulty of training deep feedforward neural networks

- In paper

> It is worth noting that after the paper submission we found that it is possible to initialise the weights without pre-training by **using the random initialisation procedure of Glorot & Bengio** (2010).

- Initialisation 추가 참고 자료
  - weight initializer 종류
  - Initialization - Xavier / He

Reference

- https://arxiv.org/pdf/1409.1556.pdf
  - Original Paper
- https://medium.com/@msmapark2/vgg16-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a
- https://www.youtube.com/watch?v=bweAXJgZGyE
- https://zl-zon.tistory.com/6
- https://rladuddms.tistory.com/75
- https://ysbstudy.tistory.com/3
- https://ctkim.tistory.com/114