7. 합성곱 신경망(2)

세종대학교
인공지능데이터사이언스학과
김 정 현

7. 합성곱 신경망(2)

7.1 이미지 분류를 위한 신경망

7.2 객체 인식을 위한 신경망

7.3 이미지 분할을 위한 신경망

7.1 이미지 분류를 위한 신경망

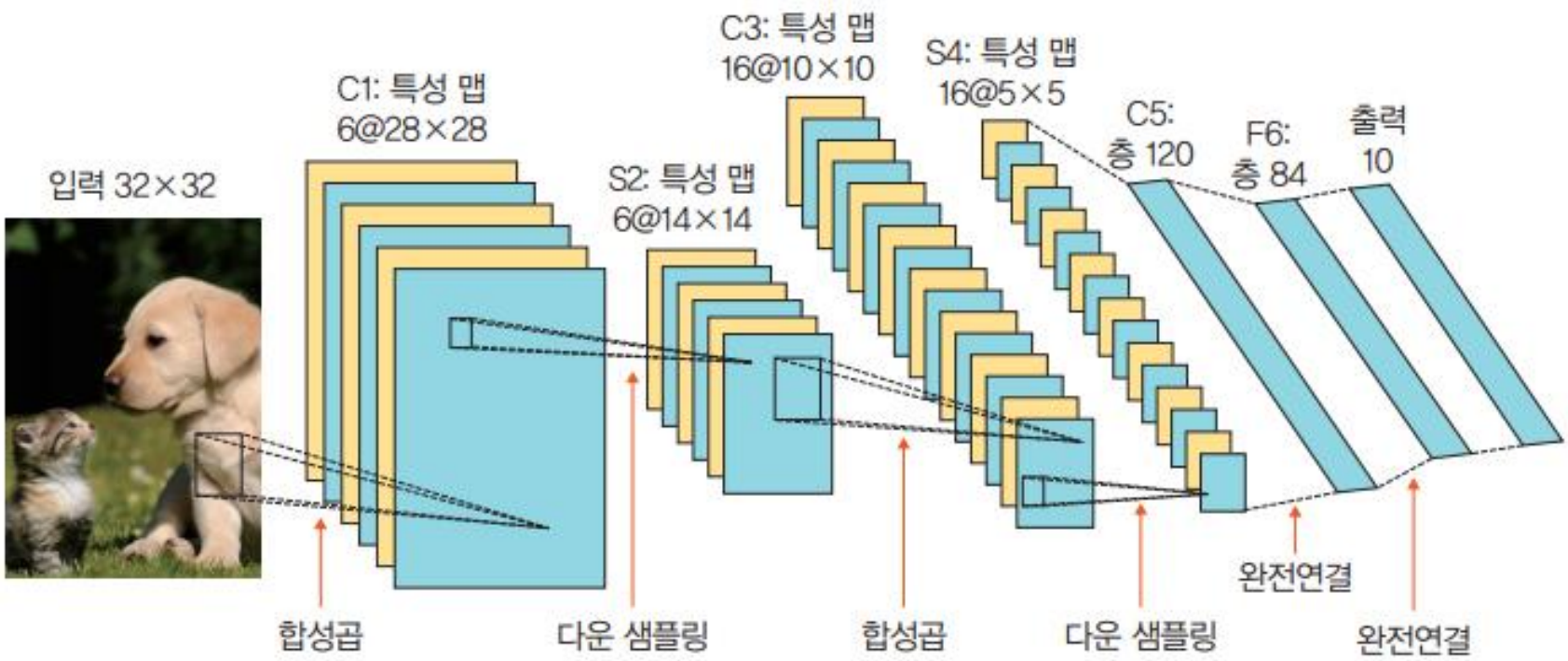
- 이미지 분류를 위한 신경망이란?
  - 입력 데이터로 이미지를 사용한 분류(classification)는 특정 대상이 영상 내에 존재하는지 여부를 판단하는 것
  - 이미지 분류(image classification)에서 주로 사용되는 합성곱 신경망으로 LeNet-5, AlexNet, VGGNet, GoogLeNet, ResNet 등이 있음

7.1 이미지 분류를 위한 신경망

- LeNet-5
  - LeNet-5는 합성곱 신경망이라는 개념을 최초로 얀 르쿤(Yann LeCun)이 개발한 구조
  - 1995년 얀 르쿤, 레옹 보토(Leon Bottu), 요슈아 벤지오(Yosua Bengio), 패트릭 하프너(Patrick Haffner)가 수표에 쓴 손글씨 숫자를 인식하는 딥러닝 구조 LeNet-5를 발표했는데, 그것이 현재 CNN의 초석이 됨
  - LeNet-5는 합성곱(convolutional)과 다운 샘플링(sub-sampling)(혹은 풀링)을 반복적으로 거치면서 마지막에 완전연결층에서 분류를 수행

7.1 이미지 분류를 위한 신경망

▼ 그림 6-1 LeNet-5



7.1 이미지 분류를 위한 신경망

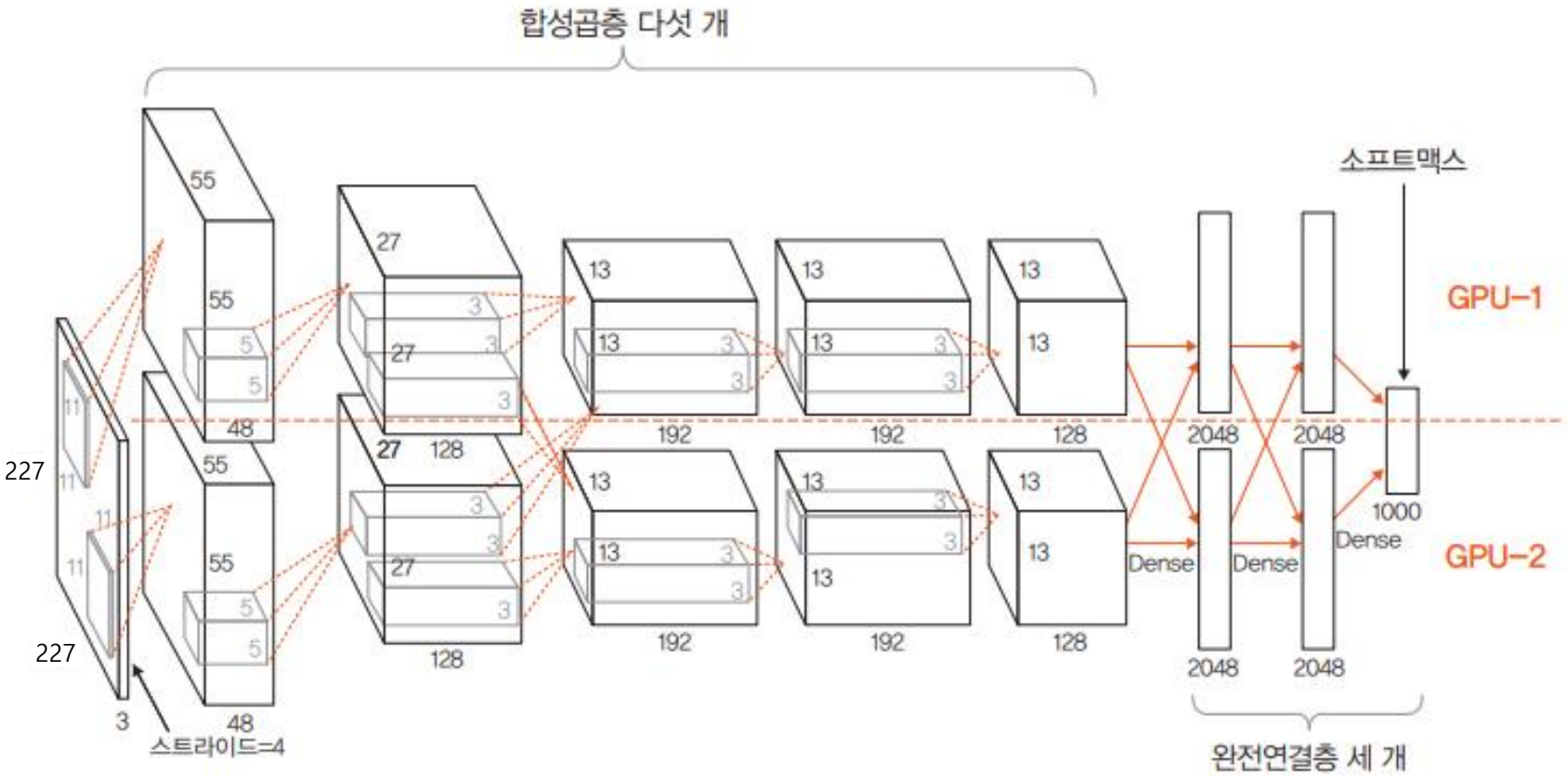
- LeNet-5
  - 다음 그림을 이용하여 구체적으로 살펴보면 C1에서 5×5 합성곱 연산 후 28×28 크기의 특성 맵(feature map) 여섯 개를 생성
  - S2에서 다운 샘플링하여 특성 맵 크기를 14×14로 줄임
  - 다시 C3에서 5×5 합성곱 연산하여 10×10 크기의 특성 맵 16개를 생성하고, S4에서 다운 샘플링하여 특성 맵 크기를 5×5로 줄임
  - C5에서 5×5 합성곱 연산하여 1×1 크기의 특성 맵 120개를 생성하고, 마지막으로 F6에서 완전연결층으로 C5의 결과를 유닛(unit)(또는 노드) 84개에 연결
  - 이때, C로 시작하는 것은 합성곱층을 의미하고, S로 시작하는 것은 풀링층을 의미하며 F로 시작하는 것은 완전연결층을 의미

7.1 이미지 분류를 위한 신경망

- AlexNet
  - AlexNet은 ImageNet 영상 데이터베이스를 기반으로 한 화상 인식 대회인 'ILSVRC 2012'에서 우승한 CNN 구조
  - 이미지 크기를 나타내는 너비(width)와 높이(height)뿐만 아니라 깊이(depth)를 갖음
  - 보통 색상이 많은 이미지는 R/G/B 성분 세 개를 갖기 때문에 시작이 3이지만, 합성곱을 거치면서 특성 맵이 만들어지고 이것에 따라 중간 영상의 깊이가 달라짐
  - AlexNet은 합성곱층 총 다섯 개와 완전연결층 세 개로 구성되어 있으며, 맨 마지막 완전연결층은 카테고리 1000개를 분류하기 위해 소프트맥스 활성화 함수를 사용하고 있음
  - 전체적으로 보면 GPU 두 개를 기반으로 한 병렬 구조인 점을 제외하면 LeNet-5와 크게 다르지 않음

7.1 이미지 분류를 위한 신경망

▼ 그림 6-9 AlexNet 구조



7.1 이미지 분류를 위한 신경망

▼ 표 6-2 AlexNet 구조 상세

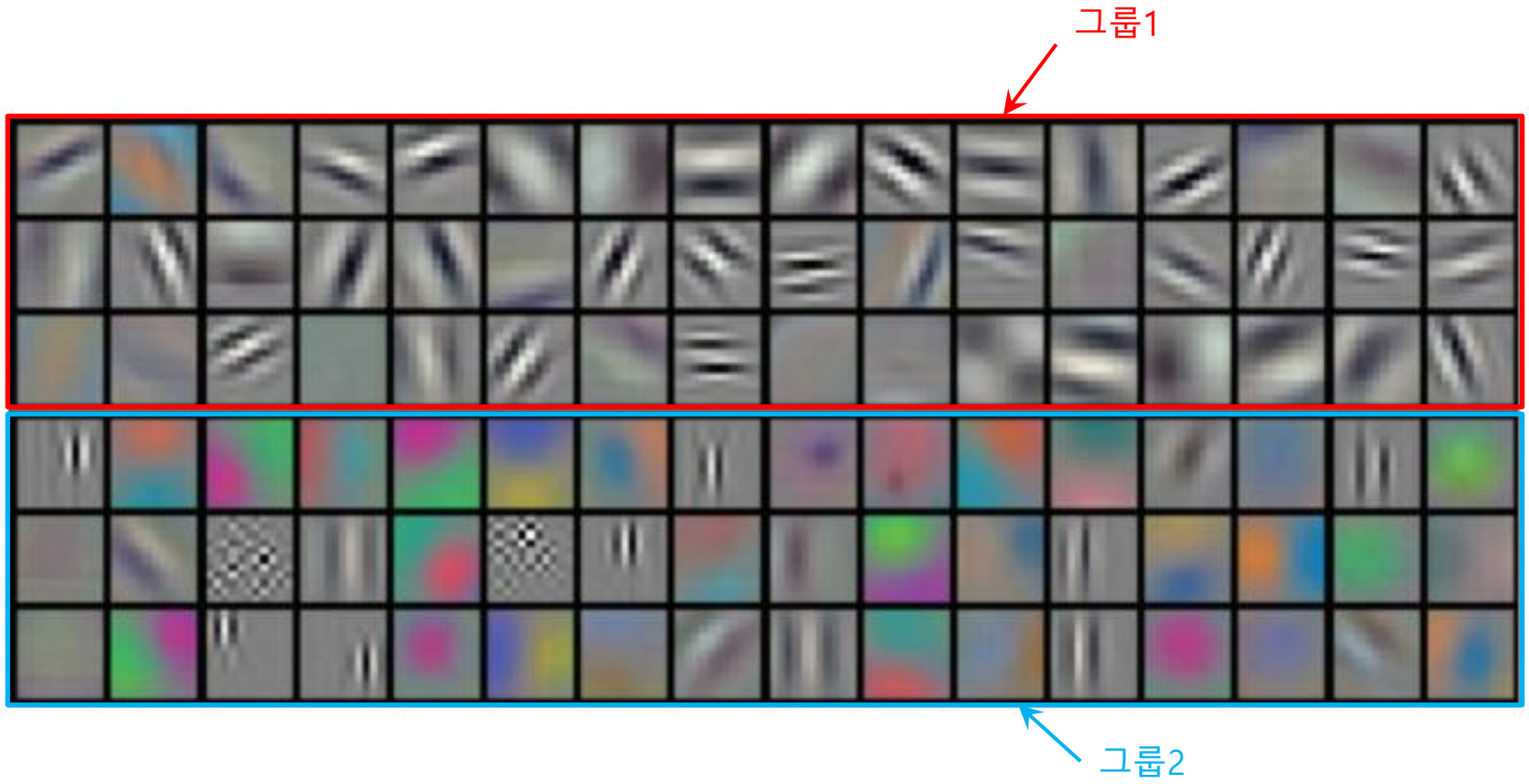
계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
이미지	1	227×227	–	–	–
합성곱층	96	55×55	11×11	4	렐루(ReLU)
최대 풀링층	96	27×27	3×3	2	–
합성곱층	256	27×27	5×5	1	렐루(ReLU)
최대 풀링층	256	13×13	3×3	2	–
합성곱층	384	13×13	3×3	1	렐루(ReLU)
합성곱층	384	13×13	3×3	1	렐루(ReLU)
합성곱층	256	13×13	3×3	1	렐루(ReLU)
최대 풀링층	256	6×6	3×3	2	–
완전연결층	–	4096	–	–	렐루(ReLU)
완전연결층	–	4096	–	–	렐루(ReLU)
완전연결층	–	1000	–	–	소프트맥스(softmax)

7.1 이미지 분류를 위한 신경망

- AlexNet
  - 네트워크에는 학습 가능한 변수가 총 6600만 개 있음
  - 네트워크에 대한 입력은 227×227×3 크기의 RGB 이미지이며, 각 클래스(혹은 카테고리)에 해당하는 1000×1 확률 벡터를 출력
  - 단, 전처리 과정에서 227 크기로 자름
  - AlexNet의 첫 번째 합성곱층 커널의 크기는 11×11×3이며, 스트라이드를 4로 적용하여 특성 맵을 96개 생성하기 때문에 55×55×96의 출력을 갖음
  - 첫 번째 계층을 거치면서 GPU-1에서는 주로 컬러와 상관없는 정보를 추출하기 위한 커널이 학습되고, GPU-2에서는 주로 컬러와 관련된 정보를 추출하기 위한 커널이 학습

7.1 이미지 분류를 위한 신경망

▼ 그림 6-10 AlexNet GPU-1·2 적용 결과



7.1 이미지 분류를 위한 신경망

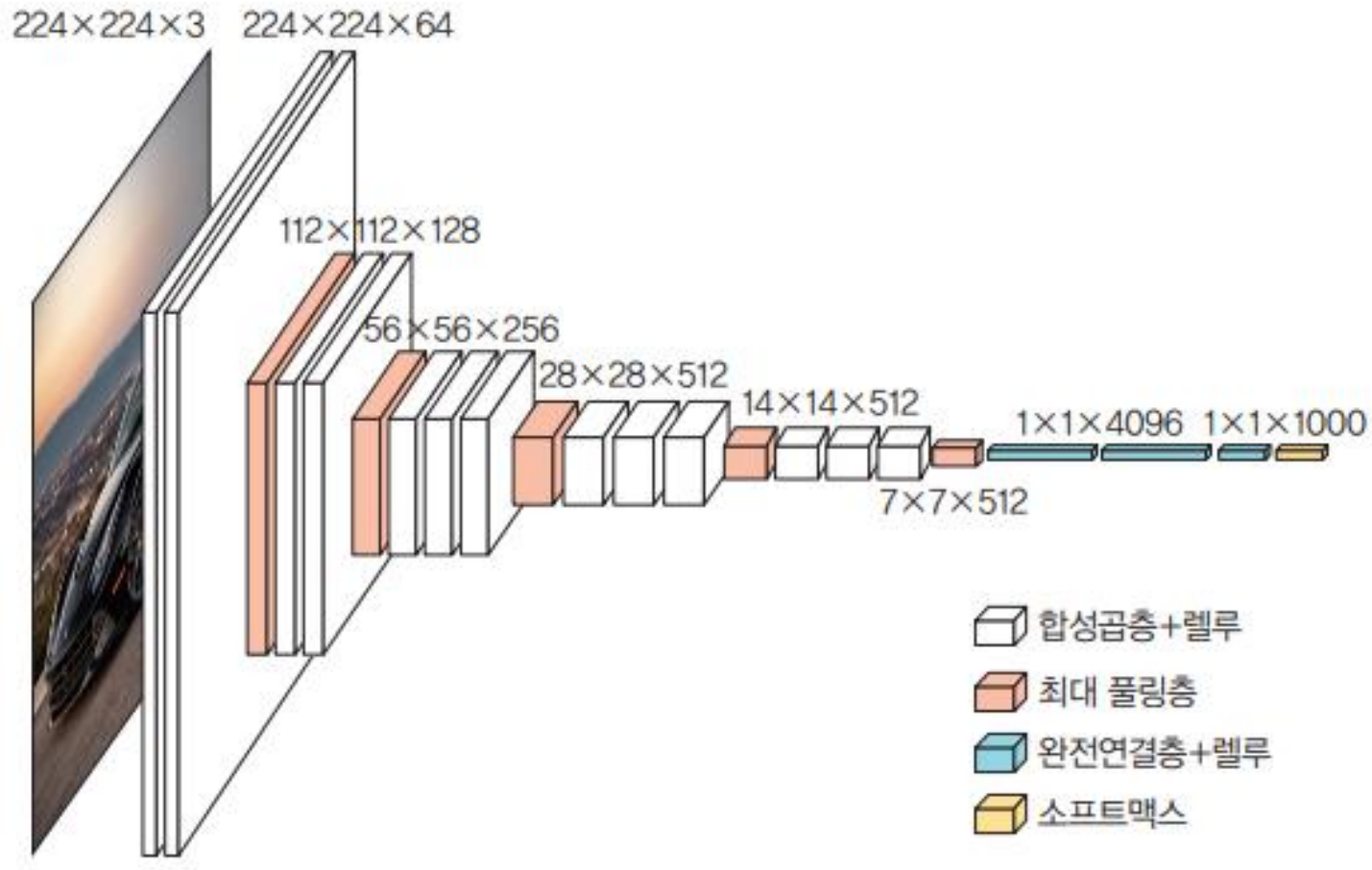
- VGGNet
  - VGGNet은 카렌 시모니안(Karen Simonyan)과 앤드류 지서만(Andrew Zisserman)이 2015 ICLR에 게재한 “Very deep convolutional networks for large-scale image recognition” 논문에서 처음 발표
  - VGGNet은 합성곱층의 파라미터 수를 줄이고 훈련 시간을 개선하려고 탄생
  - 즉, 네트워크를 깊게 만드는 것이 성능에 어떤 영향을 미치는지 확인하고자 나온 것이 VGG
  - VGG 연구 팀은 깊이의 영향만 최대한 확인하고자 합성곱층에서 사용하는 필터/커널의 크기를 가장 작은 3×3으로 고정

7.1 이미지 분류를 위한 신경망

- VGGNet
  - 네트워크 계층의 총 개수에 따라 여러 유형의 VGGNet(VGG16, VGG19 등)이 있으며, 이 중 VGG16 네트워크의 구조적 세부 사항은 다음 그림과 같음
  - VGG16에는 파라미터가 총 1억 3300만 개 있음
  - 여기에서 주목할 점은 모든 합성곱 커널의 크기는 3×3, 최대 풀링 커널의 크기는 2×2이며, 스트라이드는 2라는 것
  - 결과적으로 64개의 224×224 특성 맵(224×224×64)들이 생성
  - 또한, 마지막 16번째 계층을 제외하고는 모두 ReLU 활성화 함수가 적용

7.1 이미지 분류를 위한 신경망

▼ 그림 6-13 VGG16 구조



7.1 이미지 분류를 위한 신경망

▼ 표 6-3 VGG16 구조 상세

계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
이미지	1	224×224	–	–	–
합성곱층	64	224×224	3×3	1	렐루(ReLU)
합성곱층	64	224×224	3×3	1	렐루(ReLU)
최대 풀링층	64	112×112	2×2	2	–
합성곱층	128	112×112	3×3	1	렐루(ReLU)
합성곱층	128	112×112	3×3	1	렐루(ReLU)
최대 풀링층	128	56×56	2×2	2	–
합성곱층	256	56×56	3×3	1	렐루(ReLU)
합성곱층	256	56×56	3×3	1	렐루(ReLU)

7.1 이미지 분류를 위한 신경망

▼ 표 6-3 VGG16 구조 상세

계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
합성곱층	256	56×56	3×3	1	렐루(ReLU)
합성곱층	256	56×56	3×3	1	렐루(ReLU)
최대 풀링층	256	28×28	2×2	2	–
합성곱층	512	28×28	3×3	1	렐루(ReLU)
합성곱층	512	28×28	3×3	1	렐루(ReLU)
합성곱층	512	28×28	3×3	1	렐루(ReLU)
합성곱층	512	28×28	3×3	1	렐루(ReLU)
최대 풀링층	512	14×14	2×2	2	–

7.1 이미지 분류를 위한 신경망

▼ 표 6-3 VGG16 구조 상세

계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
합성곱층	512	14×14	3×3	1	렐루(ReLU)
합성곱층	512	14×14	3×3	1	렐루(ReLU)
합성곱층	512	14×14	3×3	1	렐루(ReLU)
합성곱층	512	14×14	3×3	1	렐루(ReLU)
최대 풀링층	512	7×7	2×2	2	–
완전연결층	–	4096	–	–	렐루(ReLU)
완전연결층	–	4096	–	–	렐루(ReLU)
완전연결층	–	1000	–	–	소프트맥스(softmax)

7.1 이미지 분류를 위한 신경망

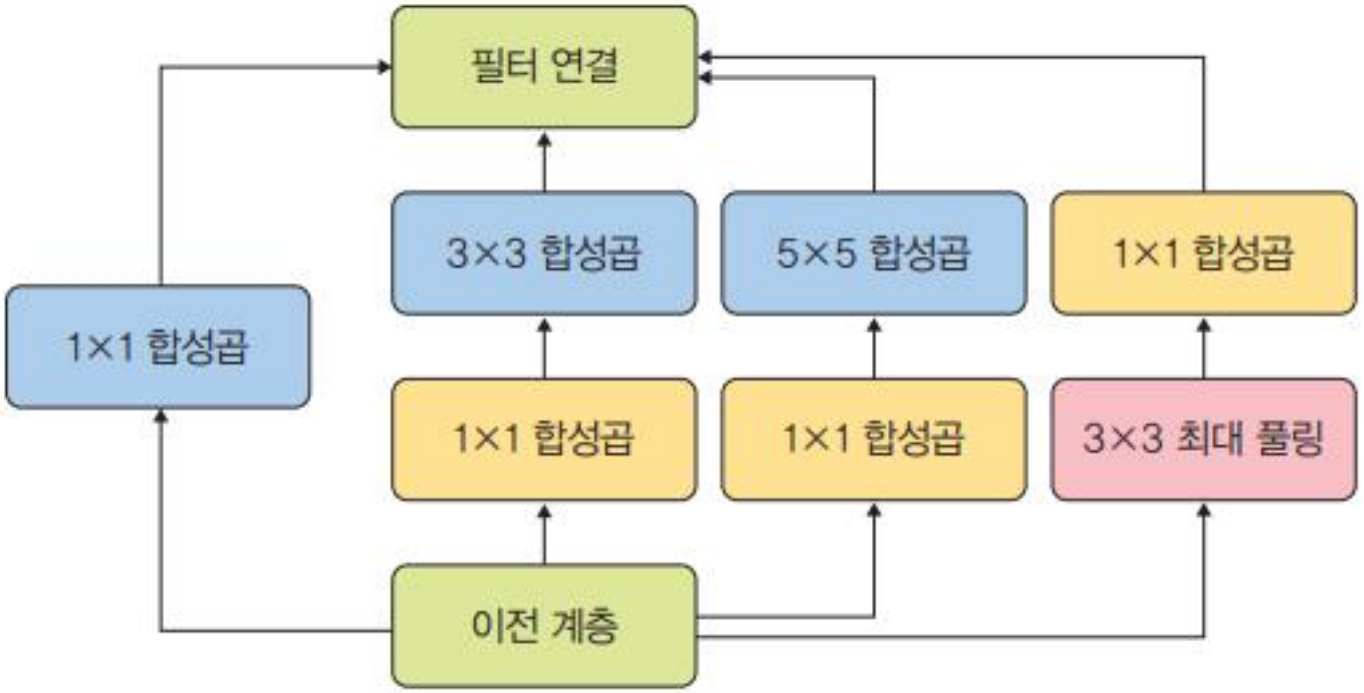
- GoogLeNet
  - GoogLeNet은 주어진 하드웨어 자원을 최대한 효율적으로 이용하면서 학습 능력은 극대화할 수 있는 깊고 넓은 신경망
  - 깊고 넓은 신경망을 위해 GoogLeNet은 인셉션(inception) 모듈을 추가
  - 인셉션 모듈에서는 특징을 효율적으로 추출하기 위해 1×1, 3×3, 5×5의 합성곱 연산을 각각 수행
  - 3×3 최대 풀링은 입력과 출력의 높이와 너비가 같아야 하므로 풀링 연산에서는 드물게 패딩을 추가해야 함
  - 결과적으로 GoogLeNet에 적용된 해결 방법은 희소 연결(sparse connectivity)

7.1 이미지 분류를 위한 신경망

- GoogLeNet
  - CNN은 합성곱, 풀링, 완전연결층들이 서로 밀집(dense)하게 연결되어 있음
  - 빽빽하게 연결된 신경망 대신 관련성(correlation)이 높은 노드끼리만 연결하는 방법을 희소 연결이라고 함
  - 이것으로 연산량이 적어지며 과적합도 해결할 수 있음
  - 인셉션 모듈은 여러 스케일의 합성곱 층을 병렬로 사용

7.1 이미지 분류를 위한 신경망

▼ 그림 6-23 GoogLeNet의 인셉션 모듈



7.1 이미지 분류를 위한 신경망

- GoogLeNet
  - 인셉션 모듈의 네 가지 연산은 다음과 같음
    - 1×1 합성곱 : 채널 수 조절
    - 1×1 합성곱 + 3×3 합성곱 : 채널 수 조절 후 좁은 지역 정보 추출
    - 1×1 합성곱 + 5×5 합성곱 : 채널 수 조절 후 넓은 지역 정보 추출
    - 3×3 최대 풀링 + 1×1 합성곱 : 다운샘플링 후 채널 수 조절

7.1 이미지 분류를 위한 신경망

- GoogLeNet
  - 딥러닝을 이용하여 ImageNet과 같은 대회에 참여하거나 서비스를 제공하려면 대용량 데이터를 학습해야 함
  - 심층 신경망의 아키텍처에서 계층이 넓고(뉴런이 많고) 깊으면(계층이 많으면) 인식률은 좋아지지만, 과적합이나 기울기 소멸 문제(vanishing gradient problem)를 비롯한 학습 시간 지연과 연산 속도 등의 문제가 있음
  - 특히 합성곱 신경망에서 이러한 문제들이 자주 나타나는데, GoogLeNet(혹은 인셉션이라고도 불림)으로 이러한 문제를 해결할 수 있다고 생각하면 됨

7.1 이미지 분류를 위한 신경망

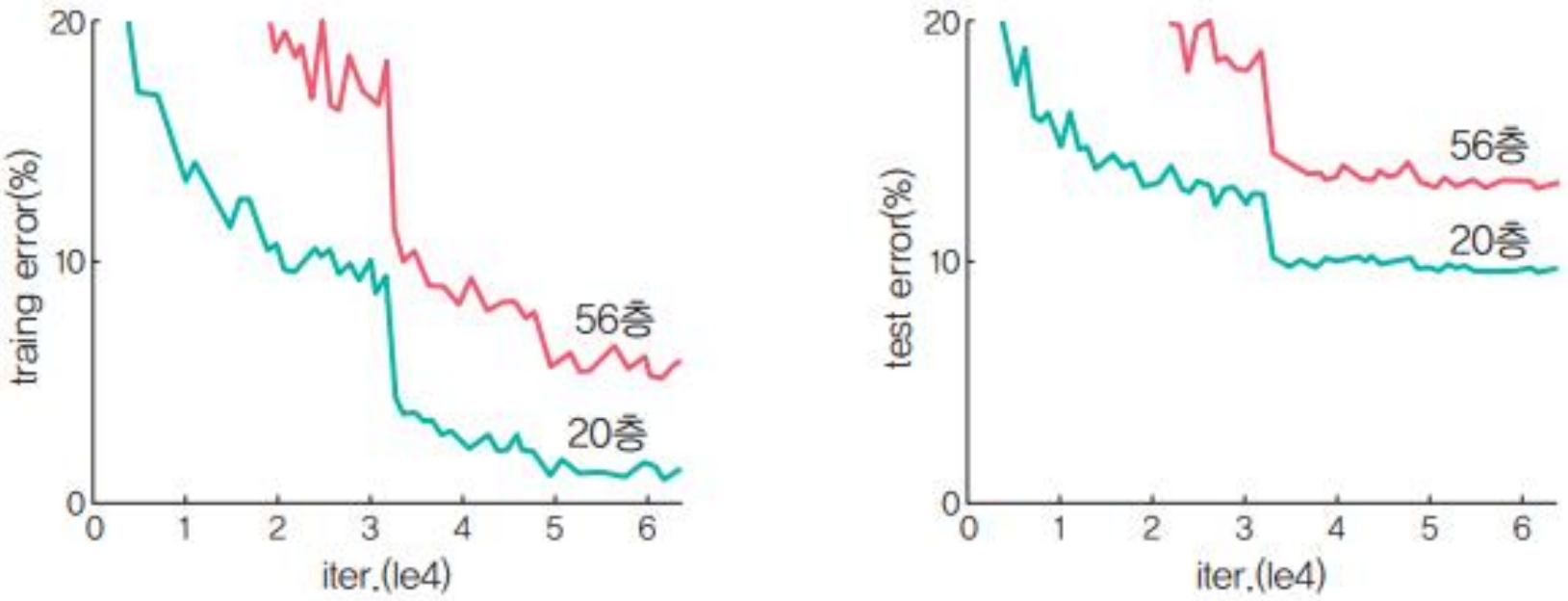
- ResNet
  - ResNet은 마이크로소프트에서 개발한 알고리즘으로 “Deep Residual Learning for Image Recognition”이라는 논문에서 발표
  - ResNet 핵심은 깊어진 신경망을 효과적으로 학습하기 위한 방법으로 레지듀얼(residual) 개념을 고안한 것

7.1 이미지 분류를 위한 신경망

- ResNet
  - 일반적으로 신경망 깊이가 깊어질수록 딥러닝 성능은 좋아질 것 같지만, 실상은 그렇지 않음
  - “Deep Residual Learning for Image Recognition” 논문에 따르면, 신경망은 깊이가 깊어질수록 성능이 좋아지다가 일정한 단계에 다다르면 오히려 성능이 나빠진다고 함
  - 다음 그림과 같이 네트워크 56층이 20층보다 더 나쁜 성능을 보임을 알 수 있음
  - 즉, 네트워크 깊이가 깊다고 해서 무조건 성능이 좋아지지는 않는다는 것을 보여 주고 있음
  - ResNet은 바로 이러한 문제를 해결하기 위해 레지듀얼 블록(residual block)을 도입
  - 레지듀얼 블록은 기울기가 잘 전파될 수 있도록 일종의 숏컷(shortcut, skip connection)을 만들어 줌

7.1 이미지 분류를 위한 신경망

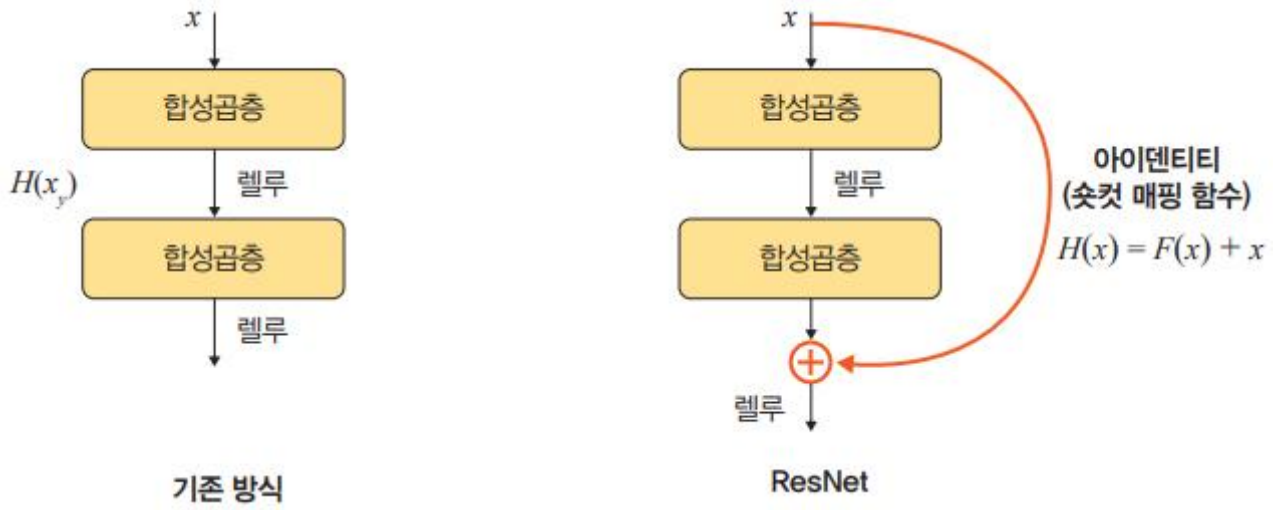
▼ 그림 6-24 네트워크 56층이 20층보다 더 나쁜 성능을 보임



7.1 이미지 분류를 위한 신경망

- ResNet
  - 이러한 개념이 필요한 이유는 2014년에 공개된 GoogLeNet은 층이 총 22개로 구성된 것에 비해 ResNet은 층이 총 152개로 구성되어 기울기 소멸 문제가 발생할 수 있기 때문임
  - 다음 그림과 같이 숏컷을 두어 기울기 소멸 문제를 방지했다고 이해하면 됨

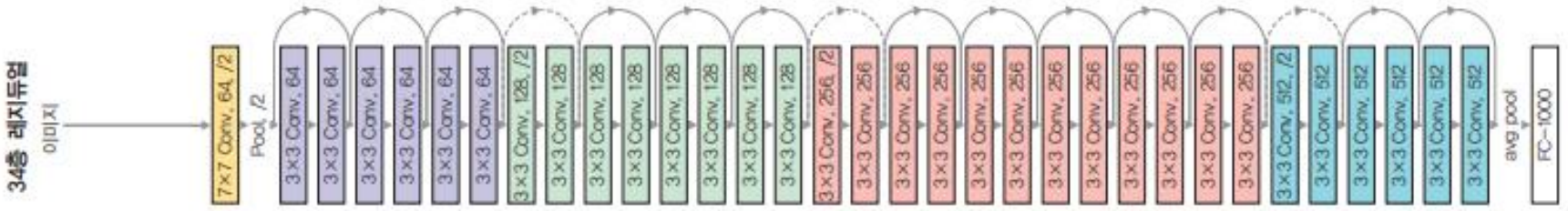
▼ 그림 6-25 ResNet 구조



7.1 이미지 분류를 위한 신경망

- ResNet
  - 블록은 합성곱 계층의 묶음
  - 그림 6-26에서 색상별(보라색, 노란색 등)로 블록을 구분했는데 이렇게 묶인 계층들을 하나의 레지듀얼 블록(residual block)이라고 함
  - 레지듀얼 블록을 여러 개 쌓은 것을 ResNet이라고 함

▼ 그림 6-26 ResNet 모델 전체 네트워크



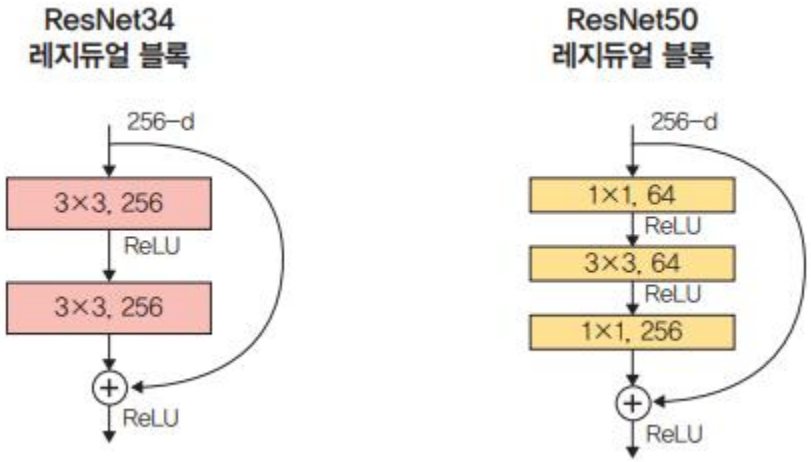
7.1 이미지 분류를 위한 신경망

- ResNet
  - 이렇게 계층을 계속해서 쌓아 늘리면 파라미터 수가 문제가 됨
  - 계층이 깊어질수록 파라미터는 증가
  - 예를 들어 ResNet34는 합성곱층이 34개와 16개의 블록으로 구성되어 있음
  - 첫 번째 블록의 파라미터가 1152K라면 전체 파라미터 수는 2만 1282K
  - 이와 같이 계층의 깊이가 깊어질수록 파라미터는 무제한으로 커질 것
  - 이러한 문제를 해결하기 위해 병목 블록(bottleneck block)이라는 것을 두었음

7.1 이미지 분류를 위한 신경망

- ResNet
  - ResNet34는 기본 블록(basic block)을 사용하며, ResNet50은 병목 블록을 사용
  - 기본 블록의 경우 파라미터 수가 39.3216M인 반면, 병목 블록의 경우 파라미터 수가 6.9632M
  - 깊이가 깊어졌음에도 파라미터 수는 감소한 것

▼ 그림 6-27 기본 블록과 병목 블록



7.1 이미지 분류를 위한 신경망

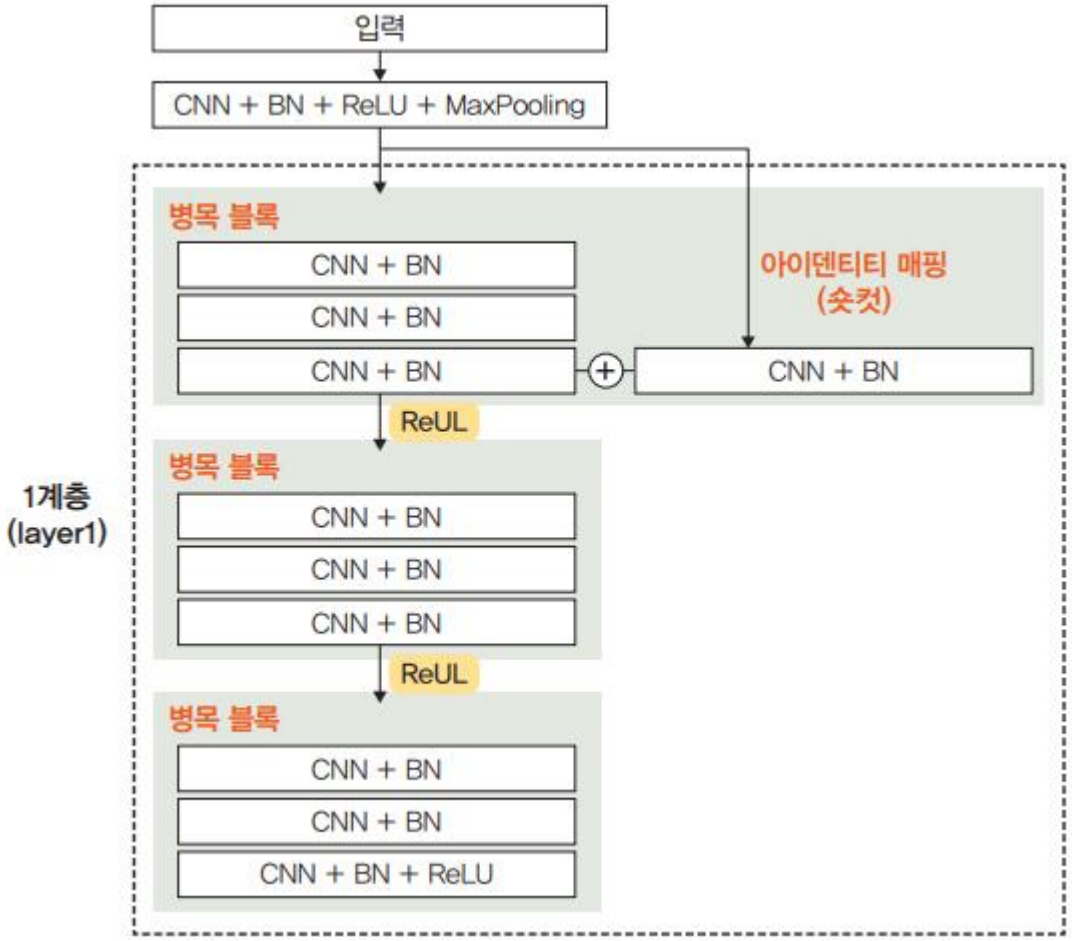
- ResNet
  - 어떻게 가능한 것일까?
  - 앞에서 분명 깊이가 깊어질수록 파라미터 수가 증가한다고 했음
  - 병목 블록을 사용하면 파라미터 수가 감소하는 효과를 줄 수 있음
  - 합성곱층을 자세히 보면 ResNet34와는 다르게 ResNet50에서는 3×3 합성곱층 앞뒤로 1×1 합성곱층이 붙어 있는데, 1×1 합성곱층의 채널 수를 조절하면서 차원을 줄였다 늘리는 것이 가능하기 때문에 파라미터 수를 줄일 수 있었던 것
  - 이 부분이 병목과 같다고 하여 병목 블록이라고 함

7.1 이미지 분류를 위한 신경망

- ResNet
  - 아이덴티티 매핑(identity mapping)(혹은 숏컷(shortcut), 스킵 연결(skip connection)이라고도 함)의 활용
  - 그림 6-27의 아래쪽에 + 기호가 있음(기본 블록과 병목 블록 모두에서 사용)
  - 이 부분을 아이덴티티 매핑이라고 함
  - 아이덴티티 매핑이란 입력 x가 어떤 함수를 통과하더라도 다시 x라는 형태로 출력되도록 함

7.1 이미지 분류를 위한 신경망

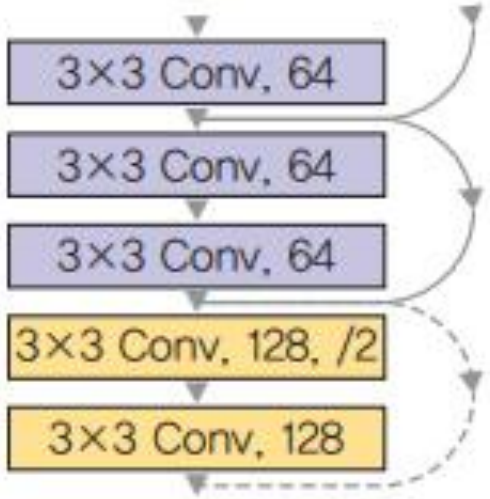
▼ 그림 6-28 아이덴티티 매핑(숏컷)



7.1 이미지 분류를 위한 신경망

- ResNet
  - ResNet의 또 다른 핵심 개념으로 다운샘플(downsample)이 있음(스트라이드 활용)
  - 다운샘플은 특성 맵(feature map) 크기를 줄이기 위한 것으로 풀링과 같은 역할을 한다고 이해하면 됨
  - 다음 그림은 ResNet 네트워크의 일부를 가져온 것

▼ 그림 6-29 ResNet 네트워크의 일부



7.1 이미지 분류를 위한 신경망

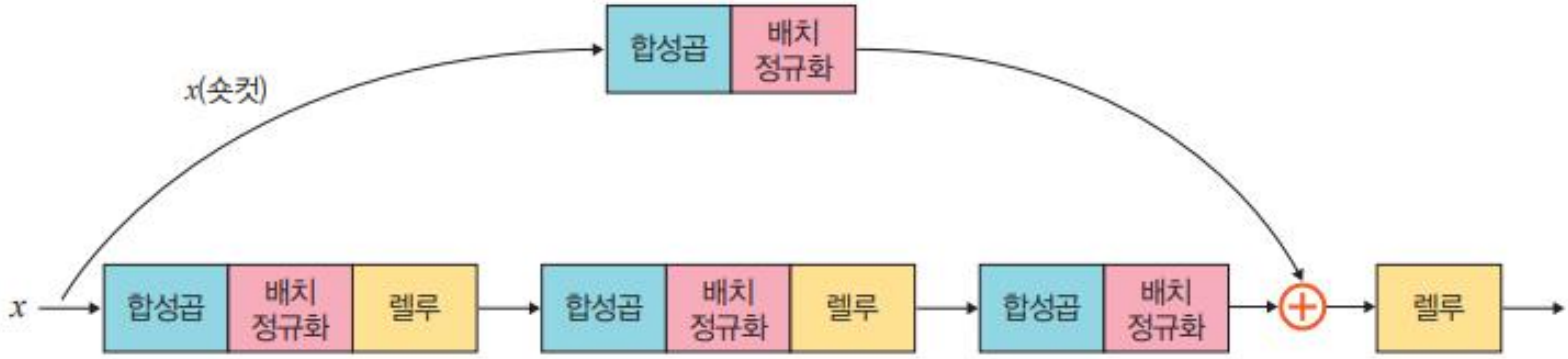
- ResNet
  - 보라색 영역의 첫 번째 블록에서 특성 맵의 형상이 (28, 28, 64)였다면 세 번째 블록의 마지막 합성곱층을 통과하고 아이덴티티 매핑(identity mapping)까지 완료된 특성 맵의 형상도(28, 28, 64)
  - 노란색 영역의 시작 지점에서는 채널 수가 128로 늘어났고, /2라는 것으로 보아 첫 번째 블록에서 합성곱층의 스트라이드가 2로 늘어나 (14, 14, 128)로 바뀐다는 것을 알 수 있음
  - 즉, 보라색과 노란색의 형태가 다른데 이들 간의 형태를 맞추지 않으면 아이덴티티 매핑을 할 수 없게 됨
  - 아이덴티티에 대해 다운샘플이 필요함

7.1 이미지 분류를 위한 신경망

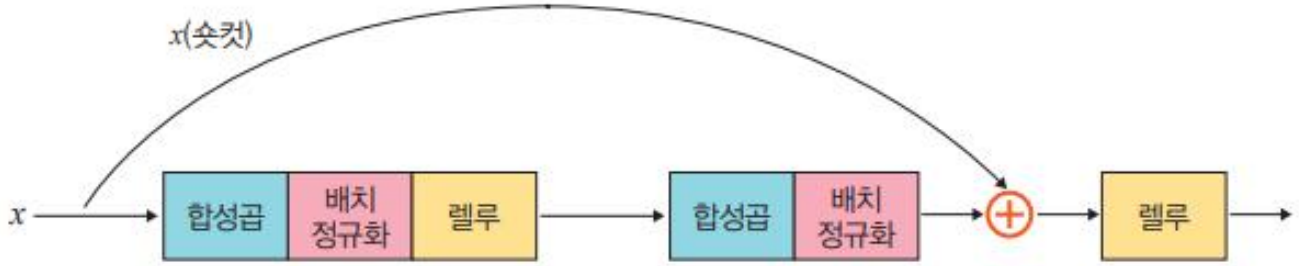
- ResNet
  - 참고로 입력과 출력의 형태를 같도록 맞추어 주기 위해서는 스트라이드(stride) 2를 가진 1×1 합성곱 계층을 하나 연결해 주면 됨
  - 이와 같이 입력과 출력의 차원이 같은 것을 아이덴티티 블록이라고 함
  - 입력 및 출력 차원이 동일하지 않고 입력의 차원을 출력에 맞추어 변경해야 하는 것을 프로젝션 숏컷(projection-shortcut) 혹은 합성곱 블록이라고 함
  - 정리하면 ResNet은 기본적으로 VGG19 구조를 뼈대로 하며, 거기에 합성곱층들을 추가해서 깊게 만든 후 숏컷들을 추가하는 것이 사실상 전부라고 생각하면 됨

7.1 이미지 분류를 위한 신경망

▼ 그림 6-30 합성곱 블록

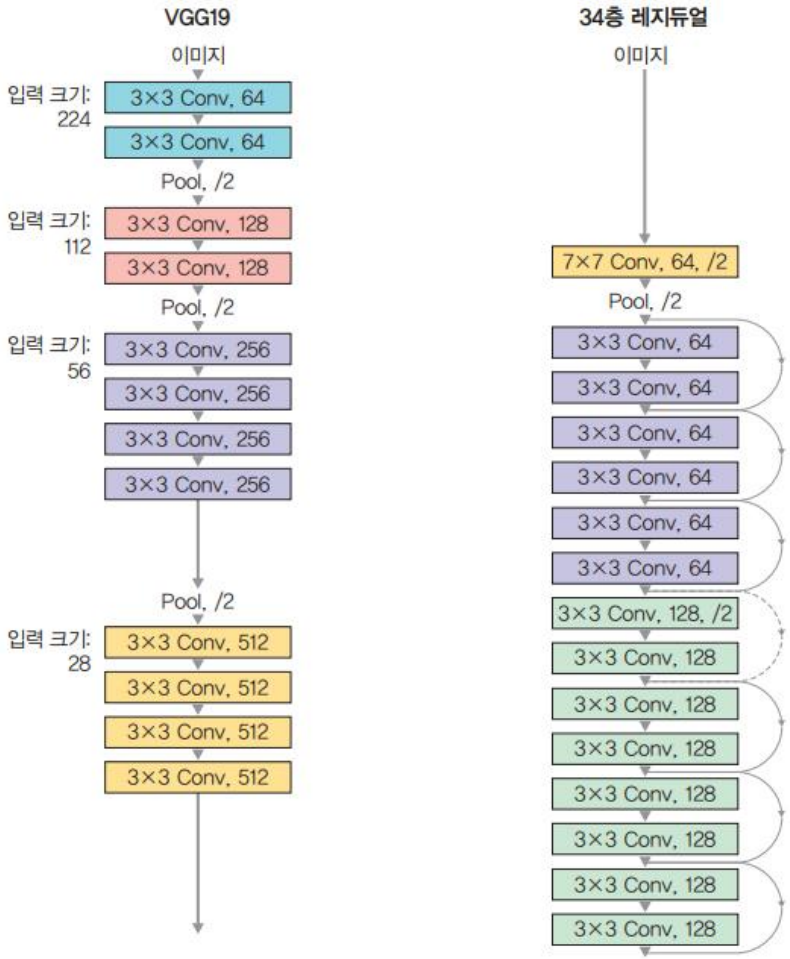


▼ 그림 6-31 아이덴티티 블록



7.1 이미지 분류를 위한 신경망

▼ 그림 6-32 VGG19와 ResNet 비교



7.2 객체 인식을 위한 신경망

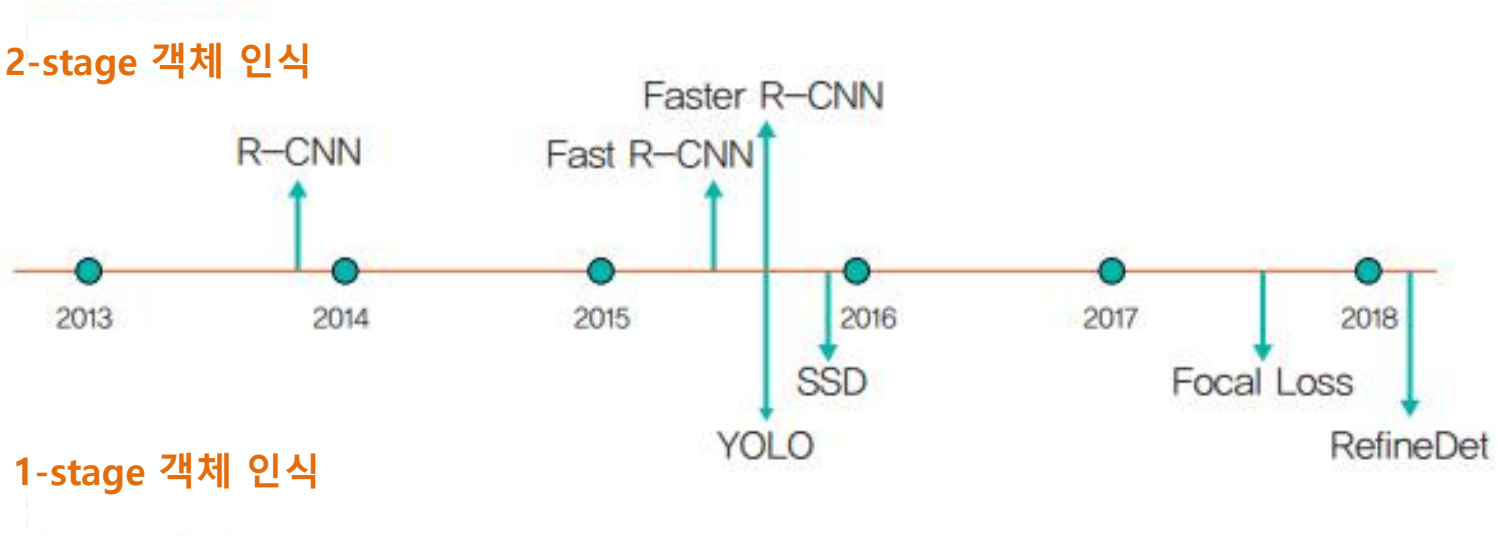
- 객체 인식을 위한 신경망
  - 객체 인식(object detection)은 이미지나 영상 내에 있는 객체를 식별하는 컴퓨터 비전 기술
  - 즉, 객체 인식이란 이미지나 영상 내에 있는 여러 객체에 대해 각 객체가 무엇인지 분류하는 문제와 그 객체 위치가 어디인지 박스(bounding box)로 나타내는 위치 검출(localization) 문제를 다루는 분야

객체 인식 = 여러 가지 객체에 대한 분류 + 객체의 위치 정보를 파악하는 위치 검출

7.2 객체 인식을 위한 신경망

- 객체 인식을 위한 신경망
  - 딥러닝을 이용한 객체 인식 알고리즘은 크게 1단계 객체 인식(1-stage detector)과 2단계 객체 인식(2-stage detector)으로 나눌 수 있음

▼ 그림 6-35 1-stage 객체 인식 vs 2-stage 객체 인식 흐름도



7.2 객체 인식을 위한 신경망

- 객체 인식을 위한 신경망
  - 1-stage 객체 인식은 이 두 문제(분류와 위치 검출)를 동시에 행하는 방법이고, 2-stage 객체 인식은 이 두 문제를 순차적으로 행하는 방법
  - 1-stage 객체 인식은 비교적 빠르지만 정확도가 낮고, 2-stage 객체 인식은 비교적 느리지만 정확도가 높음
  - 2-stage 객체 인식은 CNN을 처음으로 적용시킨 R-CNN 계열이 대표적이며, 1-stage 객체 인식에는 YOLO(You Only Look Once) 계열과 SSD 계열 등이 포함
  - 참고로 객체 인식은 자율 주행 자동차, CCTV, 무인 점포 등 많은 곳에서 활용

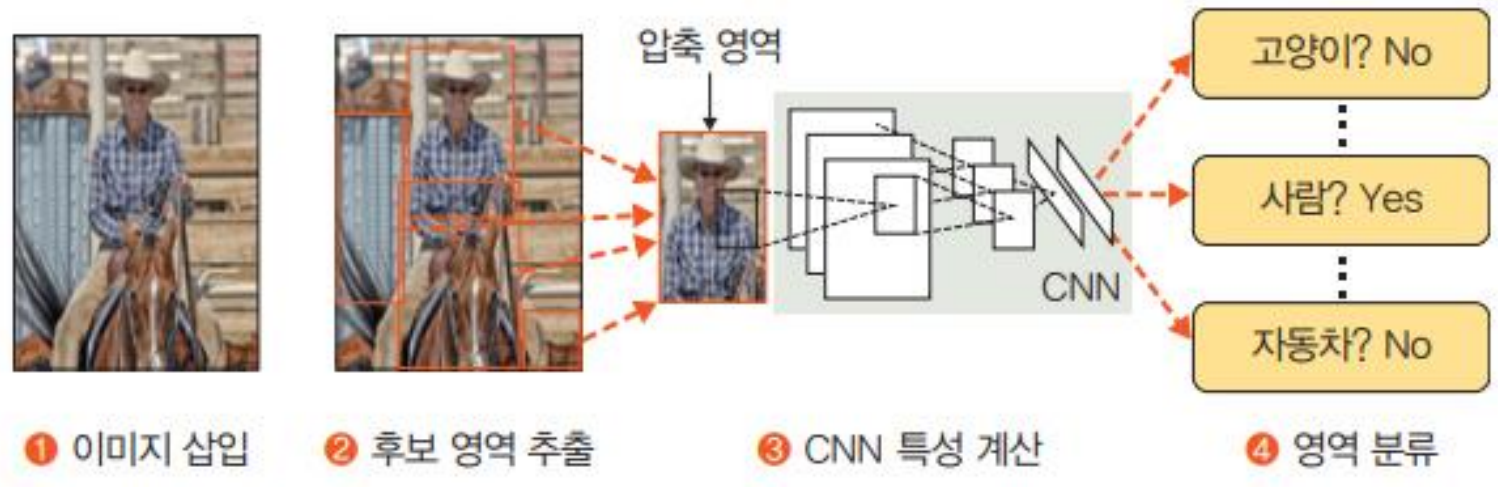
7.2 객체 인식을 위한 신경망

- R-CNN
  - 예전의 객체 인식 알고리즘들은 슬라이딩 윈도우(sliding window) 방식, 즉 일정한 크기를 가지는 윈도우(window)를 가지고 이미지의 모든 영역을 탐색하면서 객체를 검출해 내는 방식
  - 알고리즘의 비효율성 때문에 많이 사용하지 않았으며, 현재는 선택적 탐색(selective search) 알고리즘을 적용한 후보 영역(region proposal)을 많이 사용

7.2 객체 인식을 위한 신경망

- R-CNN
  - R-CNN(Region-based CNN)은 이미지 분류를 수행하는 CNN과 이미지에서 객체가 있을 만한 영역을 제안해 주는 후보 영역 알고리즘을 결합한 알고리즘
  - R-CNN의 수행 과정은 다음 그림과 같음

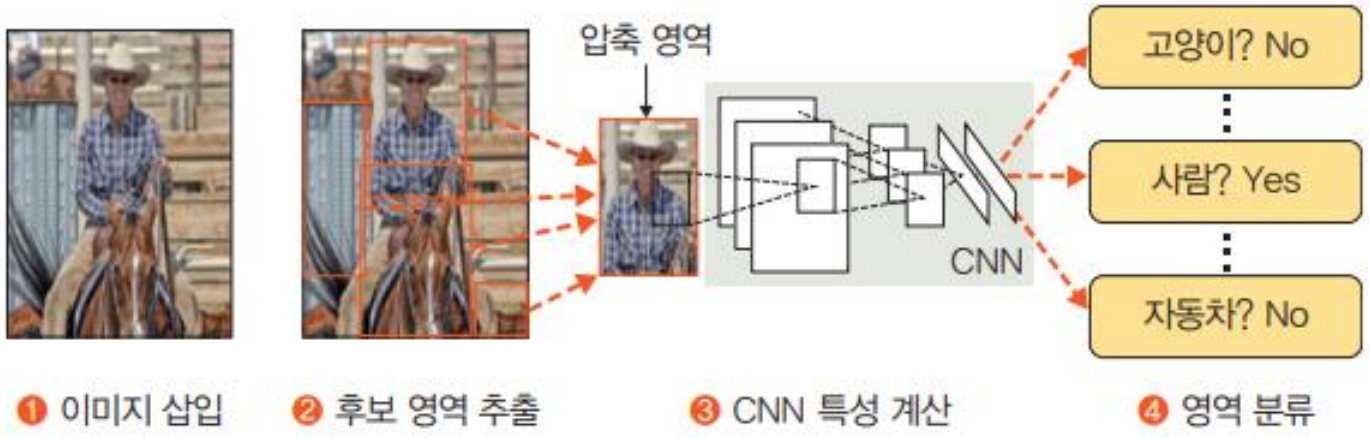
▼ 그림 6-36 R-CNN 학습 절차



7.2 객체 인식을 위한 신경망

- R-CNN
  - 수행 과정
    - 1. 이미지를 입력으로 받음
    - 2. 2000개의 바운딩 박스(bounding box)를 선택적 탐색 알고리즘으로 추출한 후 잘라내고(cropping), CNN 모델에 넣기 위해 같은 크기(227×227 픽셀)로 통일(warping)
    - 3. 크기가 동일한 이미지 2000개에 각각 CNN 모델을 적용
    - 4. 각각 분류를 진행하여 결과를 도출

▼ 그림 6-36 R-CNN 학습 절차



7.2 객체 인식을 위한 신경망

- R-CNN
  - 선택적 탐색
    - 선택적 탐색은 객체 인식이나 검출을 위한 가능한 후보 영역(객체가 있을 만한 위치, 영역)을 알아내는 방법
    - 선택적 탐색은 분할 방식을 이용하여 시드(seed)를 선정하고, 그 시드에 대한 완전 탐색을 적용
    - 선택적 탐색은 다음 세 단계 과정을 거침

7.2 객체 인식을 위한 신경망

- R-CNN
  - 선택적 탐색
    - 1단계. 초기 영역 생성(sub-segmentation)
      - 각각의 객체가 영역 한 개에 할당될 수 있도록 많은 초기 후보 영역을 생성
      - 즉, 입력된 이미지를 영역 다수 개로 분할하는 과정

▼ 그림 6-37 R-CNN 학습 1단계

입력 이미지

분할

후보 영역

7.2 객체 인식을 위한 신경망

- R-CNN
  - 선택적 탐색
    - 2단계. 작은 영역의 통합
      - 1단계에서 영역 여러 개로 나눈 것들을 비슷한 영역으로 통합하는데, 이때 탐욕(greedy) 알고리즘을 사용하여 비슷한 영역이 하나로 통합될 때까지 반복

▼ 그림 6-38 R-CNN 학습 2단계

7.2 객체 인식을 위한 신경망

- R-CNN
  - 선택적 탐색
    - 3단계. 후보 영역 생성
      - 2단계에서 통합된 이미지들을 기반으로 다음 그림과 같이 통합된 후보 영역(바운딩 박스)을 추출

▼ 그림 6-39 R-CNN 학습 3단계

7.2 객체 인식을 위한 신경망

- R-CNN
  - 여기에서 사용되는 용어 의미는 다음과 같음
    - 완전 탐색(exhaustive search): 후보가 될 만한 대상의 크기 및 비율이 모두 다른 상황을 고려하여 후보 영역을 찾는 기법
    - 분할(segmentation): 영상 데이터의 특성(색상, 모양, 무늬 등)에 따라 분할하여 후보 영역을 선정하는 기법
    - 후보 영역(바운딩 박스): 3D 객체의 형태를 모두 포함할 수 있는 최소 크기의 박스
    - 시드(seed): 영상에서는 특정 기준점의 픽셀에서 점점 의미가 같은 영상 범위까지 픽셀을 확장해 나가면서 분할하는데, 이때 특정 기준점이 되는 픽셀

7.2 객체 인식을 위한 신경망

- R-CNN
  - R-CNN에서는 AlexNet, VGG, GoogLeNet 등의 딥러닝 모델을 사용할 수 있음
  - R-CNN은 성능이 뛰어나기는 하지만 다음과 같은 단점으로 크게 발전하지는 못했음
    - 1. 앞서 언급한 세 단계의 복잡한 학습 과정
    - 2. 긴 학습 시간과 대용량 저장 공간
    - 3. 객체 검출(object detection) 속도 문제
  - 이러한 문제를 해결하기 위해 Fast R-CNN, Faster R-CNN, Mask R-CNN 등이 제안됨

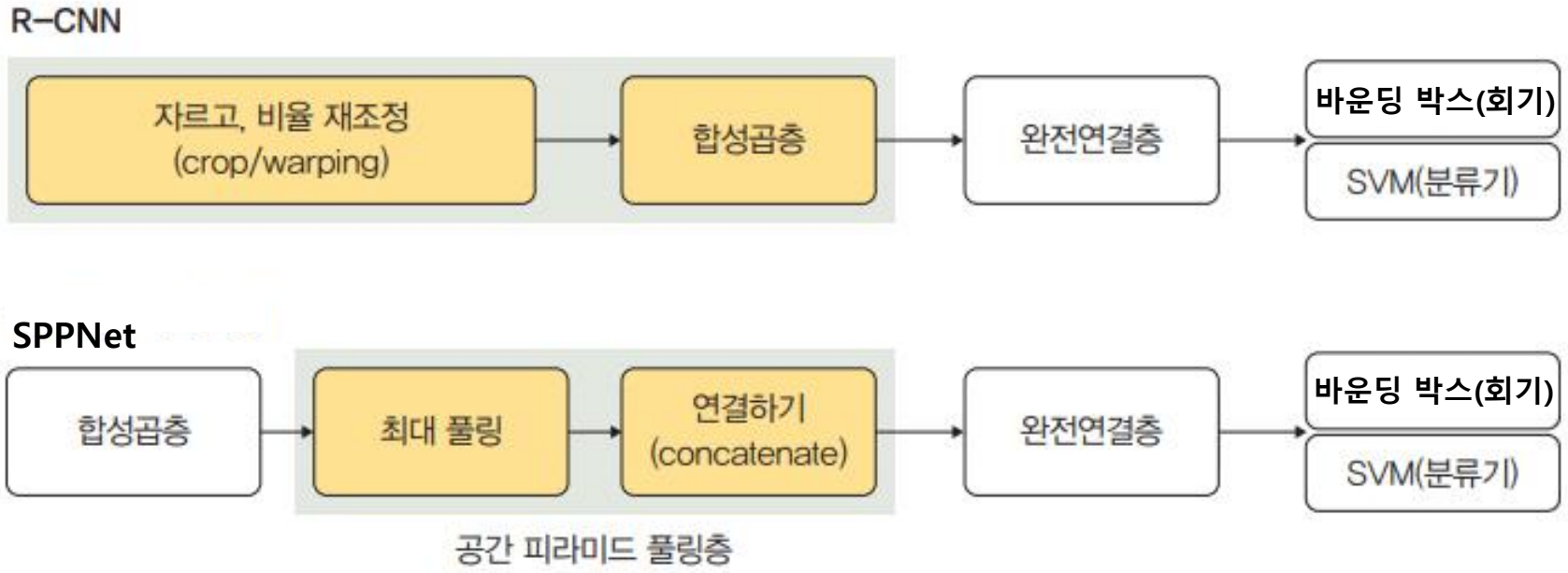
7.2 객체 인식을 위한 신경망

- SPPNet
  - 기존 CNN 구조(예: 다음 그림의 R-CNN)들은 모두 완전연결층을 위해 입력 이미지를 고정해야 했음
  - 그렇기 때문에 신경망을 통과시키려면 이미지를 고정된 크기로 자르거나(crop) 비율을 조정(warp)해야 했음
  - 이렇게 하면 물체의 일부분이 잘리거나 본래의 생김새와 달라지는 문제점이 있음
  - 이러한 문제를 해결하고자 공간 피라미드 풀링(spatial pyramid pooling)을 도입

7.2 객체 인식을 위한 신경망

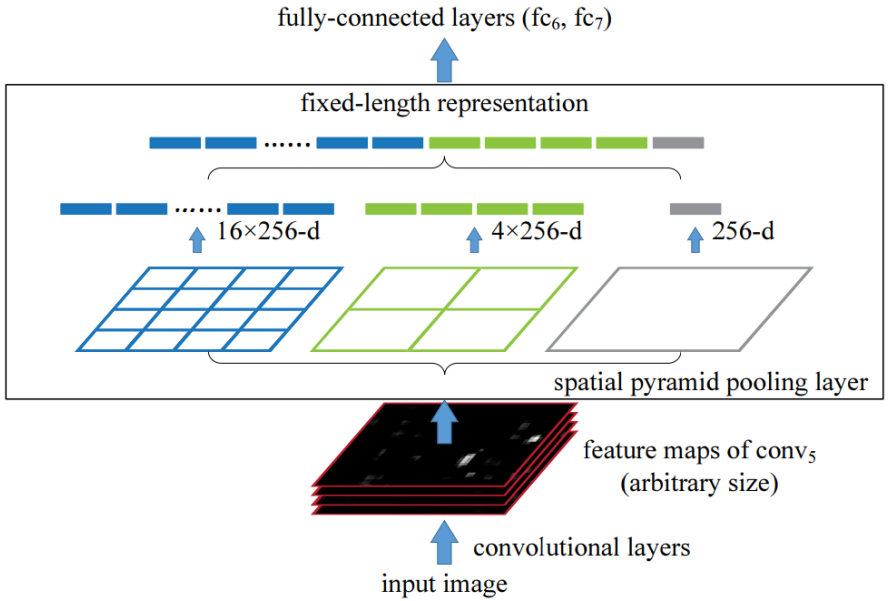
- SPPNet

▼ 그림 6-40 공간 피라미드 풀링



7.2 객체 인식을 위한 신경망

- SPPNet
  - 즉, 공간 피라미드 풀링은 입력 이미지의 크기에 관계없이 합성곱층을 통과시키고, 완전연결층에 전달되기 전에 특성 맵들을 동일한 크기로 조절해 주는 풀링층을 적용하는 기법
  - 입력 이미지의 크기를 조절하지 않고 합성곱층을 통과시키기 때문에 원본 이미지의 특징이 훼손되지 않는 특성 맵을 얻을 수 있음
  - 또한, 이미지 분류나 객체 인식 같은 여러 작업에 적용할 수 있다는 장점이 있음

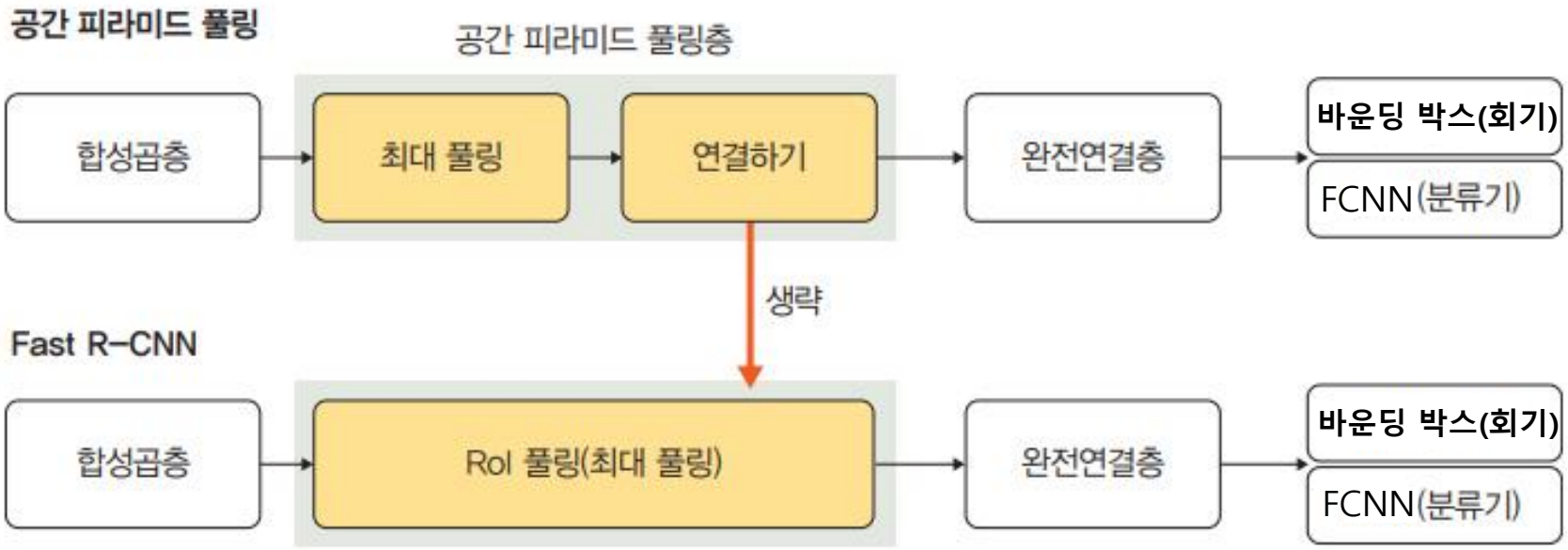


7.2 객체 인식을 위한 신경망

- Fast R-CNN
  - R-CNN은 바운딩 박스(RoI, Region of Interest)마다 CNN을 돌리고, 분류를 위한 긴 학습 시간이 문제였음
  - Fast R-CNN(Fast Region-based CNN)은 R-CNN의 속도 문제를 개선하려고 RoI 풀링을 도입
  - 즉, 선택적 탐색에서 찾은 바운딩 박스 정보가 CNN을 통과하면서 유지되도록 하고 최종 CNN 특성 맵은 풀링을 적용하여 완전연결층을 통과하도록 크기를 조정
  - 이렇게 하면 바운딩 박스마다 CNN을 돌리는 시간을 단축할 수 있음

7.2 객체 인식을 위한 신경망

▼ 그림 6-41 Fast R-CNN



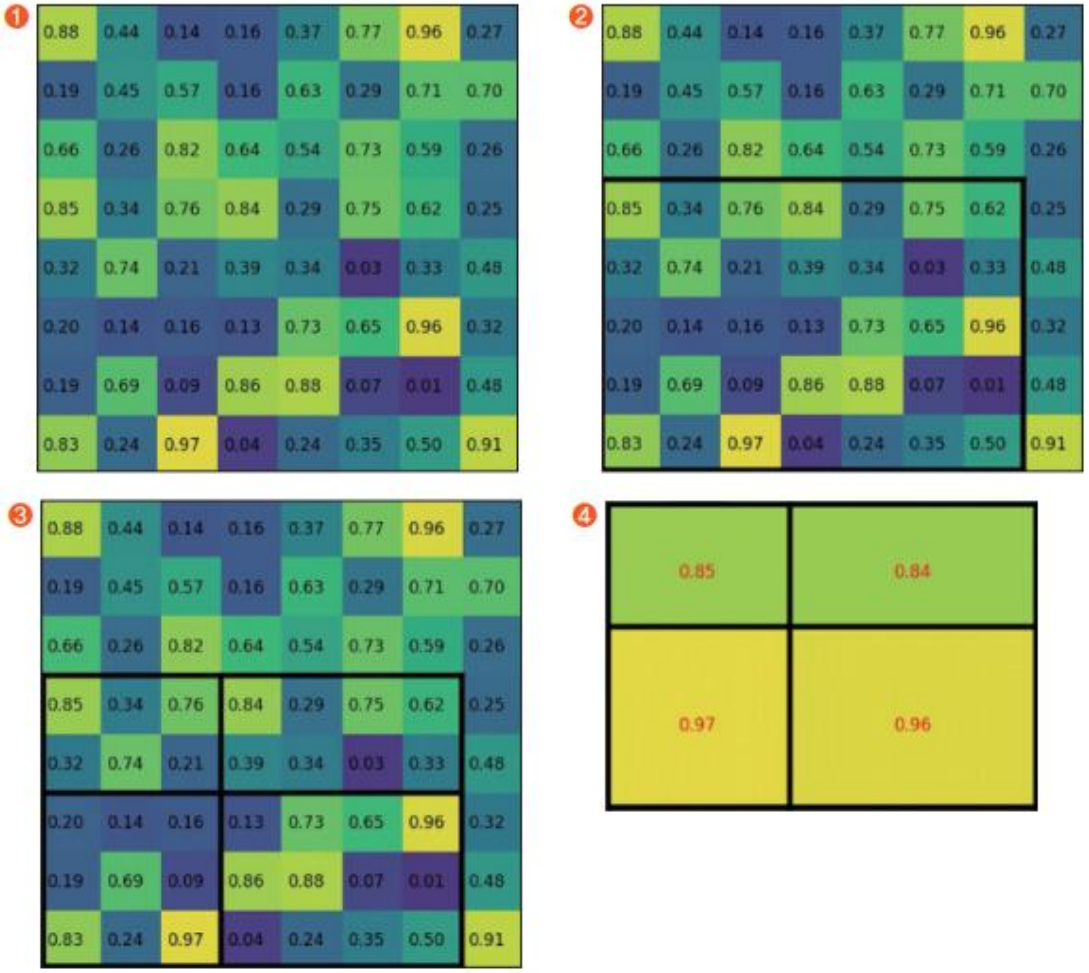
7.2 객체 인식을 위한 신경망

- Fast R-CNN
  - RoI 풀링(RoI pooling)은 크기가 다른 특성 맵의 영역마다 윈도우 크기를 다르게 최대 풀링을 적용하여 결괏값 크기를 동일하게 맞추는 방법
  - 예를 들어 다음 그림과 같이 박스 한 개가 픽셀 한 개를 뜻하는 특성 맵이 있다고 하자
  - 즉, 8×8 특성 맵(①)에서 선택적 탐색으로 뽑아냈던 7×5 후보 영역(②)이 있으며, 이것을 2×2로 만들기 위해 적절한 윈도우 크기(7/2=3, 5/2=2)로 풀링 영역(③)을 정하고 겹치는 영역이 없도록 최대 풀링을 적용하면 2×2 결과(④)를 얻을 수 있음

7.2 객체 인식을 위한 신경망

- Fast R-CNN

▼ 그림 6-42 RoI 풀링



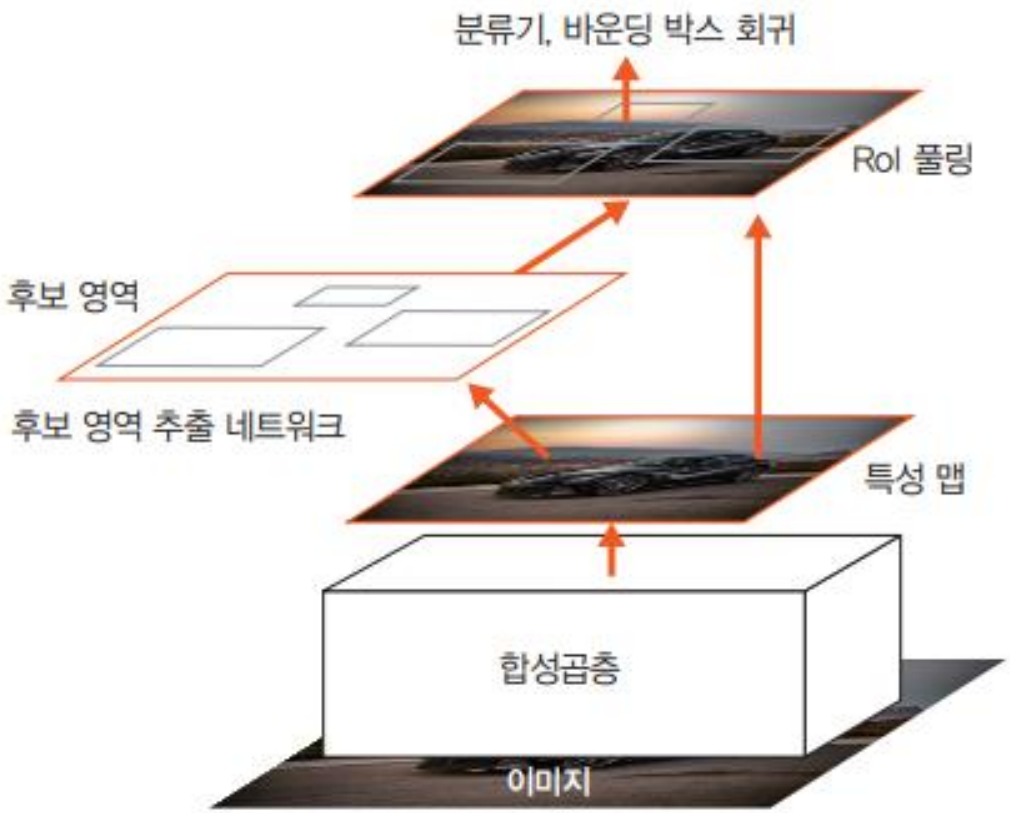
7.2 객체 인식을 위한 신경망

- Faster R-CNN
  - Faster R-CNN은 '더욱 빠른' 객체 인식을 수행하기 위한 네트워크
  - 기존 Fast R-CNN 속도의 걸림돌이었던 후보 영역 생성을 CNN 내부 네트워크에서 진행할 수 있도록 설계
  - 즉, Faster R-CNN은 기존 Fast R-CNN에 후보 영역 추출 네트워크(Region Proposal Network, RPN)를 추가한 것이 핵심이라고 할 수 있음
  - Faster R-CNN에서는 외부의 느린 선택적 탐색(CPU로 계산) 대신 내부의 빠른 RPN(GPU로 계산)을 사용
  - RPN은 다음 그림과 같이 마지막 합성곱층 다음에 위치하고, 그 뒤에 Fast R-CNN과 마찬가지로 RoI 풀링과 분류기(classifier), 바운딩 박스 회귀(bounding-box regression)가 위치

7.2 객체 인식을 위한 신경망

- Faster R-CNN

▼ 그림 6-43 Faster R-CNN



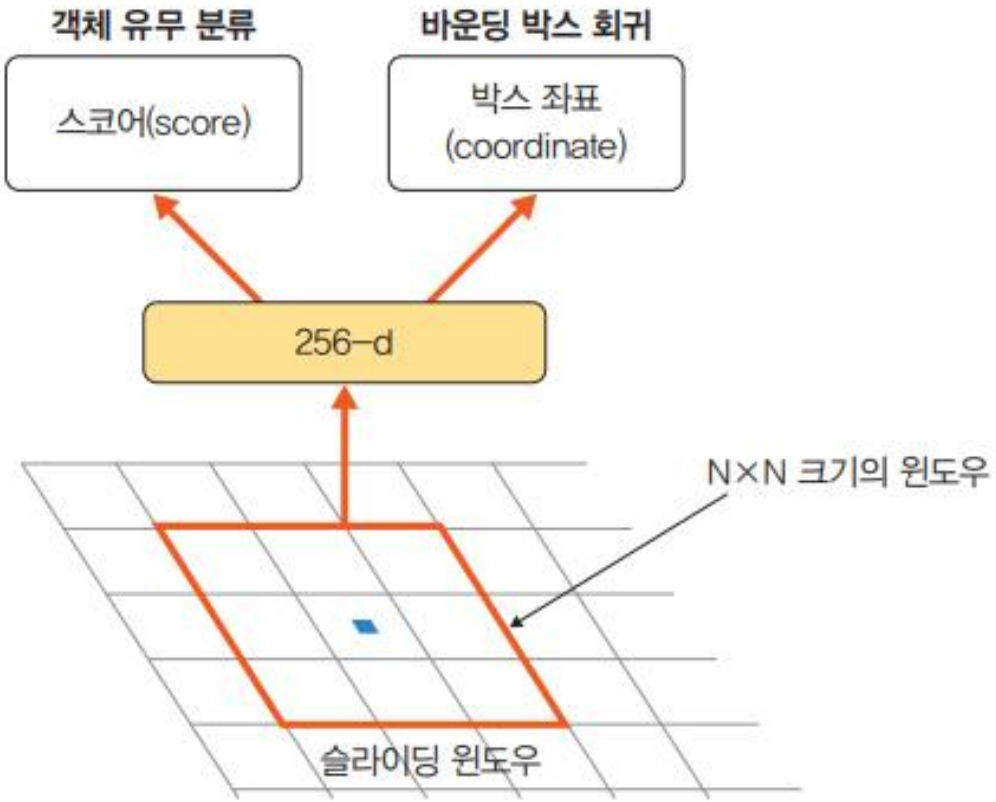
7.2 객체 인식을 위한 신경망

- Faster R-CNN
  - 후보 영역 추출 네트워크는 특성 맵 N×N 크기의 작은 윈도우 영역을 입력으로 받고, 해당 영역에 객체의 존재 유무 판단을 위해 이진 분류(binary classification)를 수행하는 작은 네트워크를 생성
  - R-CNN, Fast R-CNN에서 사용되었던 바운딩 박스 회귀 또한 위치 보정(좌표점 추론)을 위해 추가
  - 또한, 하나의 특성 맵에서 모든 영역에 대한 객체의 존재 유무를 확인하기 위해서는 슬라이딩 윈도우 방식으로 앞서 설계한 작은 윈도우 영역(N×N 크기)을 이용하여 객체를 탐색

7.2 객체 인식을 위한 신경망

- Faster R-CNN

▼ 그림 6-44 후보 영역 추출 네트워크



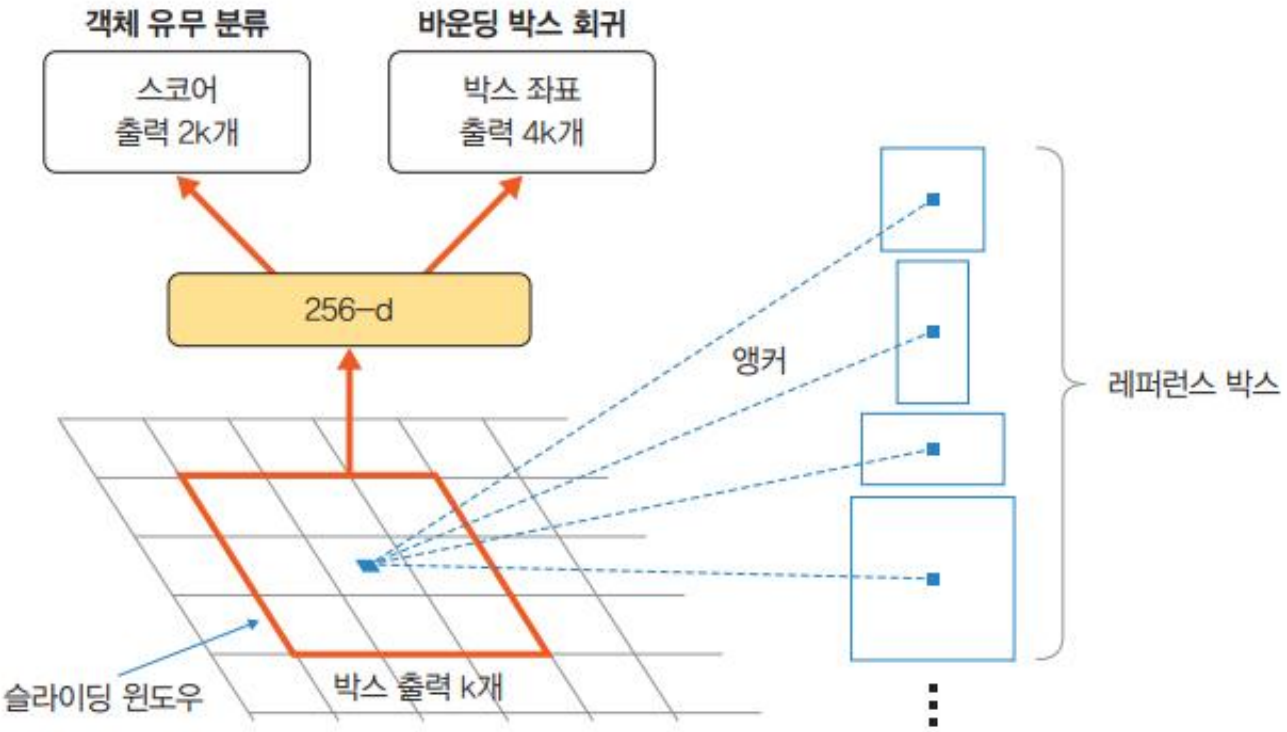
7.2 객체 인식을 위한 신경망

- Faster R-CNN
  - 후보 영역 추출 네트워크는 이미지에 존재하는 객체들의 크기와 비율이 다양하기 때문에 고정된 N×N 크기의 입력만으로 다양한 크기와 비율의 이미지를 수용하기 어려운 단점이 있음
  - 이러한 단점을 보완하기 위해 여러 크기와 비율의 레퍼런스 박스(reference box) k개를 미리 정의하고 각각의 슬라이딩 윈도우 위치마다 박스 k개를 출력하도록 설계하는데, 이 방식을 앵커(anchor)라고 함
  - 즉, 후보 영역 추출 네트워크의 출력 값은 모든 앵커 위치에 대해 각각 객체와 배경을 판단하는 2k개의 분류에 대한 출력과 x, y, w, h 위치 보정 값을 위한 4k개의 회귀 출력을 갖음
  - 예를 들어, 특성 맵 크기가 w×h라면 하나의 특성 맵에 앵커가 총 w×h×k개 존재

7.2 객체 인식을 위한 신경망

- Faster R-CNN

▼ 그림 6-45 앵커



7.3 이미지 분할을 위한 신경망

- 이미지 분할을 위한 신경망
  - 이미지 분할(image segmentation)은 신경망을 훈련시켜 이미지를 픽셀 단위로 분할하는 것
  - 즉, 이미지를 픽셀 단위로 분할하여 이미지에 포함된 객체를 추출
  - 이미지 분할의 대표적 네트워크는 완전 합성곱 네트워크, 합성곱 & 역합성곱 네트워크, U-Net, PSPNet, DeepLabv3/v3+가 있음

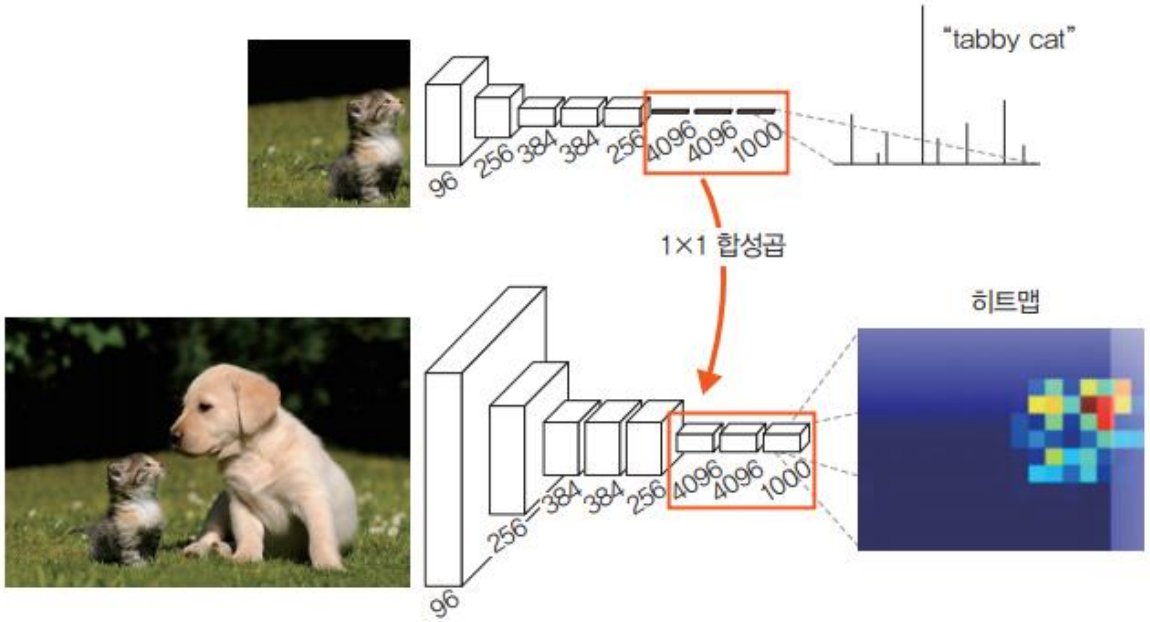
7.3 이미지 분할을 위한 신경망

- 완전 합성곱 네트워크
  - 완전연결층의 한계는 고정된 크기의 입력만 받아들이며, 완전연결층을 거친 후에는 위치 정보가 사라진다는 것
  - 이러한 문제를 해결하기 위해 완전연결층을 1×1 합성곱으로 대체하는 것이 완전 합성곱 네트워크
  - 즉, 완전 합성곱 네트워크(Fully Convolutional Network, FCN)는 이미지 분류에서 우수한 성능을 보인 CNN 기반 모델(AlexNet, VGG16, GoogLeNet)을 변형시켜 이미지 분할에 적합하도록 만든 네트워크
  - 합성곱층으로 사용되기 때문에 입력 이미지에 대한 크기 제약이 사라지는 장점이 있음

7.3 이미지 분할을 위한 신경망

- 완전 합성곱 네트워크
  - 예를 들어, 다음 그림과 같이 AlexNet 아래쪽에서 사용되었던 완전연결층 세 개를 1×1 합성곱으로 변환하면 위치 정보가 남아 있기 때문에 히트맵(heatmap) 그림과 같이 고양이의 위치를 확인할 수 있음

▼ 그림 6-46 완전 합성곱 네트워크



7.3 이미지 분할을 위한 신경망

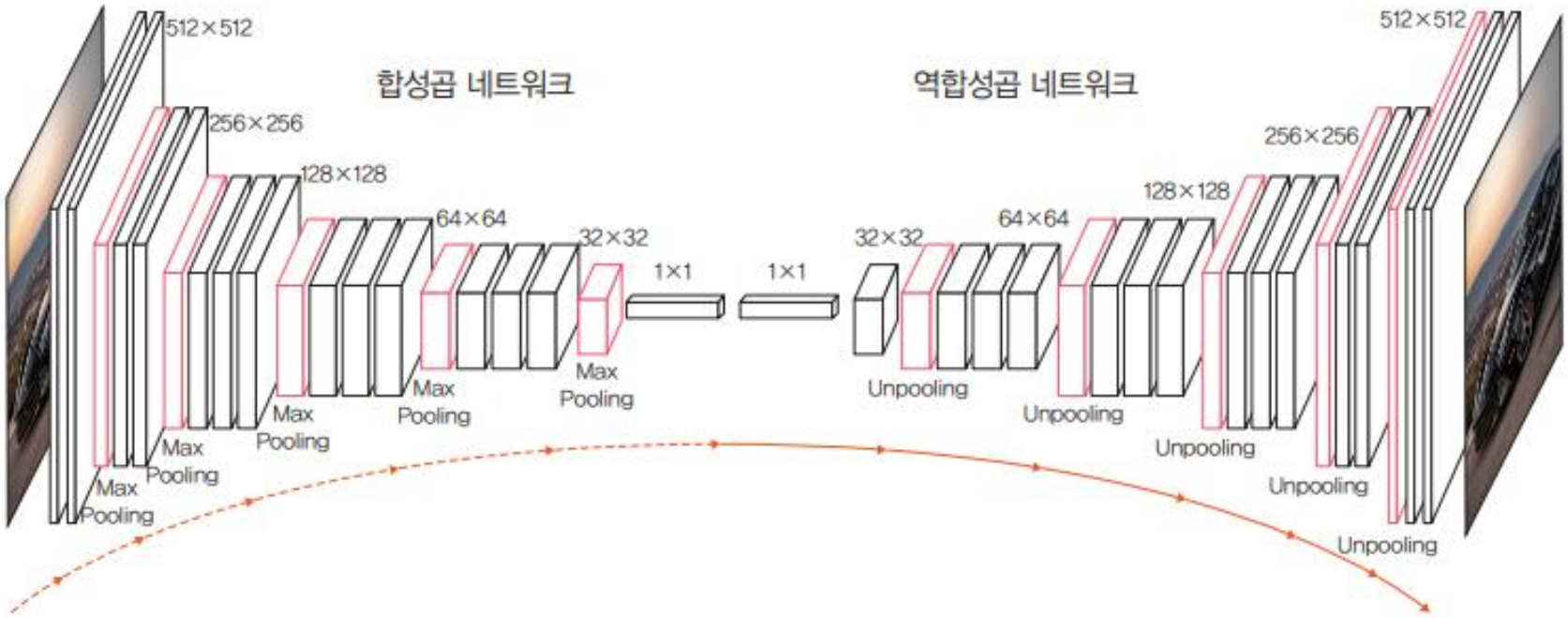
- 합성곱 & 역합성곱 네트워크
  - 완전 합성곱 네트워크는 위치 정보가 보존된다는 장점에도 다음과 같은 단점이 있음
    - 여러 단계의 합성곱층과 풀링층을 거치면서 해상도가 낮아짐
    - 낮아진 해상도를 복원하기 위해 업샘플링 방식을 사용하기 때문에 이미지의 세부 정보들을 잃어버리는 문제가 발생

7.3 이미지 분할을 위한 신경망

- 합성곱 & 역합성곱 네트워크
  - 이러한 문제를 해결하기 위해 역합성곱 네트워크를 도입한 것이 합성곱 & 역합성곱 네트워크(convolutional & deconvolutional network)
  - 역합성곱은 CNN의 최종 출력 결과를 원래의 입력 이미지와 같은 크기로 만들고 싶을 때 사용
  - 시멘틱 분할(semantic segmentation)등에 활용할 수 있으며, 역합성곱을 업샘플링(upsampling)이라고도 함

7.3 이미지 분할을 위한 신경망

▼ 그림 6-47 합성곱 & 역합성곱 네트워크



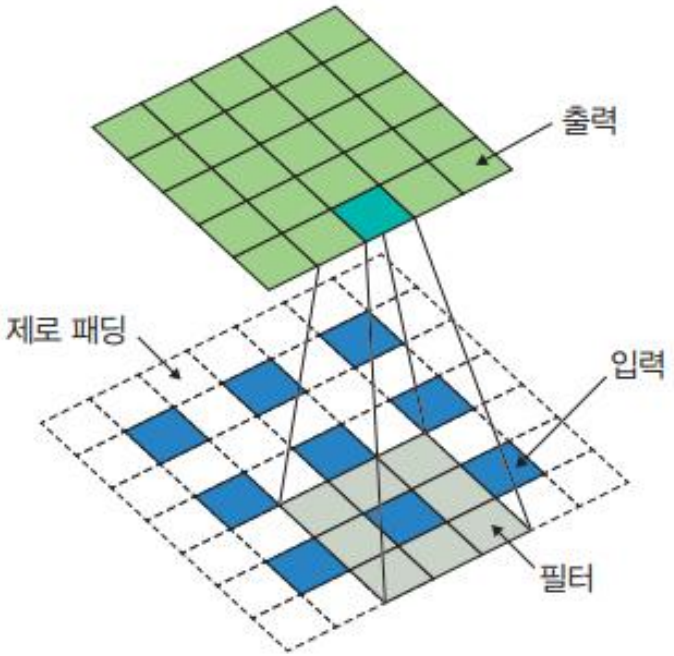
7.3 이미지 분할을 위한 신경망

- 합성곱 & 역합성곱 네트워크
  - CNN에서 합성곱층은 합성곱을 사용하여 특성 맵 크기를 줄임
  - 역합성곱은 이와 반대로 특성 맵 크기를 증가시키는 방식으로 동작
  - 역합성곱은 다음 방식으로 동작
    1. 각각의 픽셀 주위에 제로 패딩(zero-padding)을 추가
    2. 이렇게 패딩된 것에 합성곱 연산을 수행

7.3 이미지 분할을 위한 신경망

- 합성곱 & 역합성곱 네트워크
  - 오른쪽 그림에서 아래쪽의 파란색 픽셀이 입력이며, 초록색 픽셀이 출력
  - 이 파란색 픽셀 주위로 흰색 제로 패딩을 수행하고, 회색 필터로 합성곱 연산을 수행하면 초록색이 출력

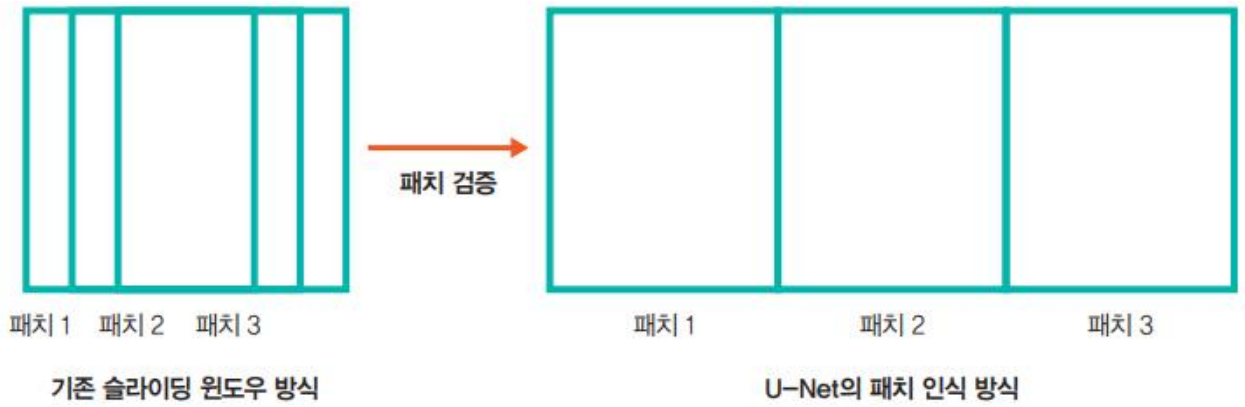
▼ 그림 6-48 역합성곱 진행 방식



7.3 이미지 분할을 위한 신경망

- U-Net
  - U-Net은 바이오 메디컬 이미지 분할을 위한 합성곱 신경망
  - 메디컬 이미지의 분할과 관련해서 항상 회자되는 네트워크가 U-Net
  - U-Net은 다음 특징이 있음
    - 속도가 빠르다: 기존 슬라이딩 윈도우 방식은 이전 패치(patch)에서 검증이 끝난 부분을 다음 패치에서 또 검증하기 때문에 속도가 느렸음
    - U-Net은 이미 검증이 끝난 패치는 건너뛰기 때문에 속도가 빠름

▼ 그림 6-49 슬라이딩 윈도우 방식



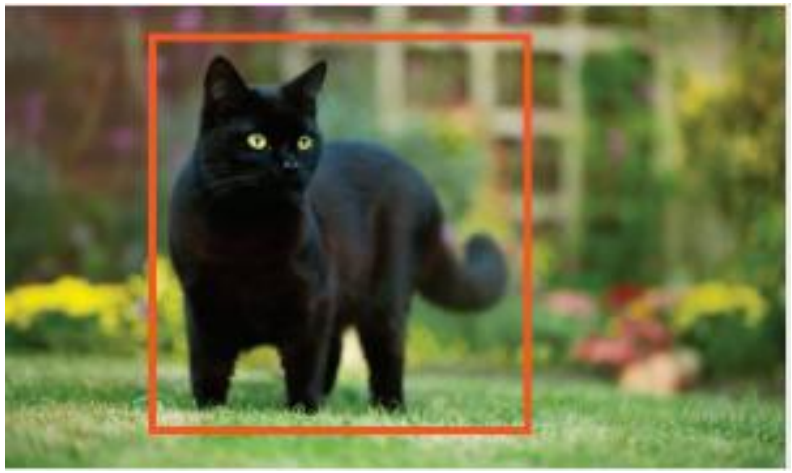
7.3 이미지 분할을 위한 신경망

- U-Net
  - U-Net은 다음 특징이 있음
    - 트레이드오프(trade-off)에 빠지지 않음
      - 일반적으로 패치 크기가 커진다면 넓은 범위의 이미지를 인식하는 데 뛰어나기 때문에 컨텍스트(context) 인식에 탁월하지만 지역화에는 한계가 있음
    - 즉, 너무 넓은 범위를 한 번에 인식하기 때문에 지역화에는 약하기 마련인데, U-Net은 컨텍스트 인식과 지역화 트레이드오프 문제를 개선

7.3 이미지 분할을 위한 신경망

- U-Net
  - 지역화
    - 지역화(localization)는 오른쪽 그림과 같이 이미지 안에 객체(고양이) 위치 정보를 출력해 주는 것으로, 주로 바운딩 박스를 많이 사용
    - 바운딩 박스의 네 꼭지점 픽셀 좌표가 출력되는 것이 아닌 왼쪽 위(left top), 오른쪽 아래(right bottom) 좌표를 출력

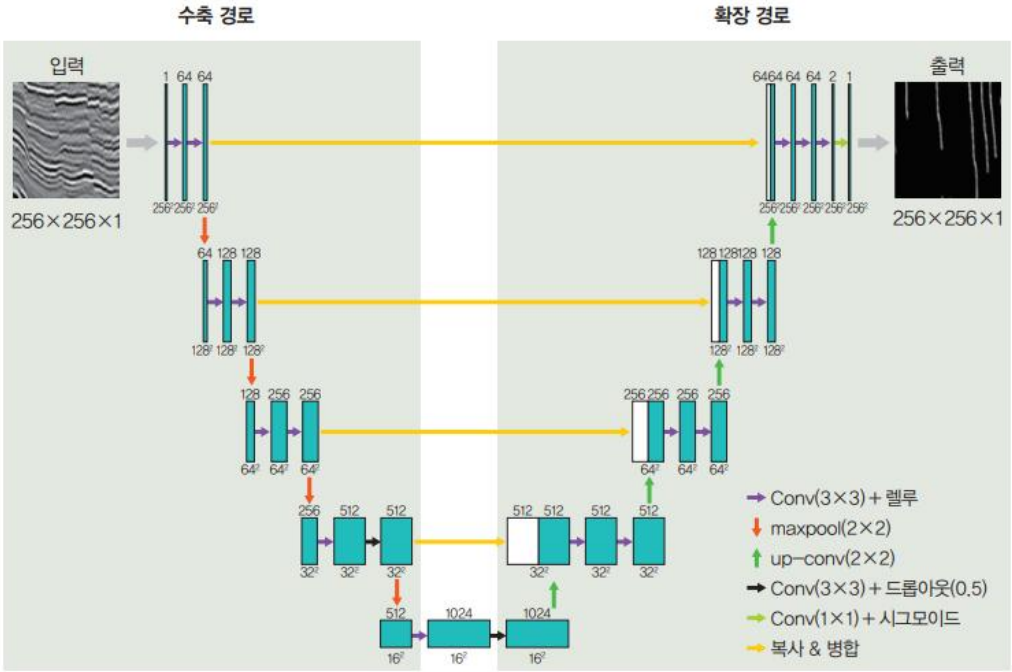
▼ 그림 6-50 지역화



7.3 이미지 분할을 위한 신경망

- U-Net
  - 다음은 U-Net 구조
    - U-Net은 FCN을 기반으로 구축되었으며, 수축 경로(contracting path)와 확장 경로(expansive path)로 구성되어 있음
    - 수축 경로는 컨텍스트를 포착하며, 확장 경로는 특성 맵을 업 샘플링하고 수축 경로에서 포착한 특성 맵의 컨텍스트와 결합하여 정확한 지역화를 수행

▼ 그림 6-51 U-Net



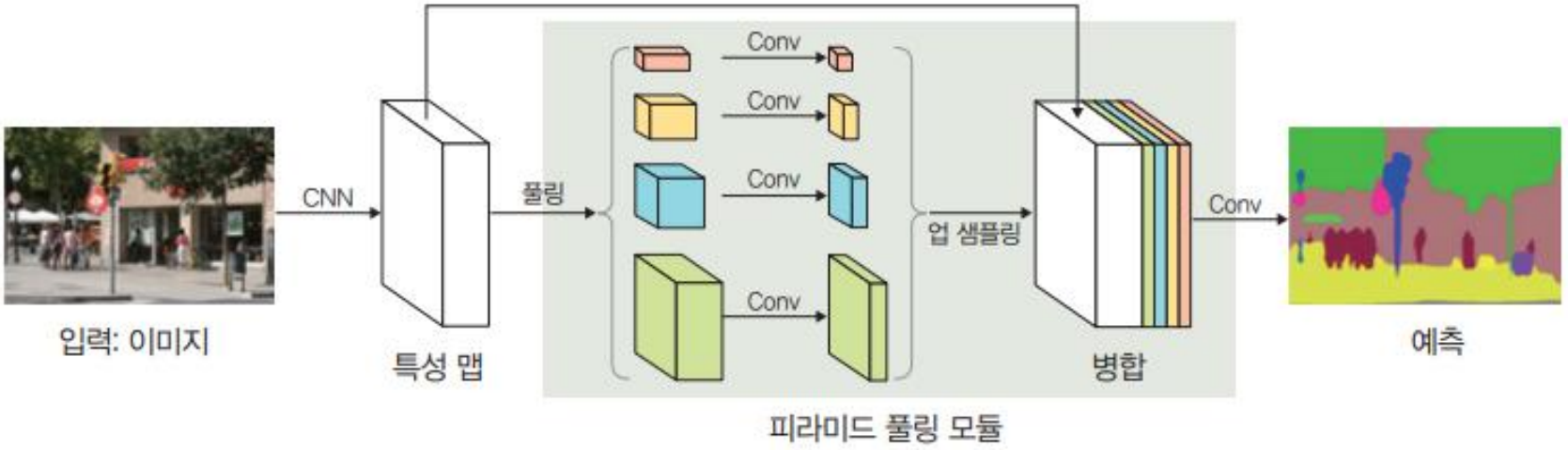
7.3 이미지 분할을 위한 신경망

- U-Net
  - 다음은 U-Net 구조
    - U-Net은 3×3 합성곱이 주를 이루는데 각 합성곱 블록은 3×3 합성곱 두 개로 구성되어 있으며, 그 사이에 드롭아웃(dropout)이 있음
    - 다음 그림의 왼쪽 수축 경로에서의 블록은 3×3 합성곱 두 개로 구성된 것이 네 개가 있는 형태
    - 각 블록은 최대 풀링(maxpool)을 이용하여 크기를 줄이면서 다음 블록으로 넘어감
    - 반면 다음 그림의 오른쪽 확장 경로에서는 합성곱 블록에 up-conv라는 것을 앞에 붙였음
    - 수축 과정에서 줄어든 크기를 다시 키워 가면서 합성곱 블록을 이용하는 형태
    - 즉, 크기가 다양한 이미지의 객체를 분할하기 위해 크기가 다양한 특성 맵을 병합할 수 있도록 다운 샘플링과 업 샘플링을 순서대로 반복하는 구조로 되어 있음

7.3 이미지 분할을 위한 신경망

- PSPNet
  - PSPNet(Pyramid Scene Parsing Network)은 CVPR(The IEEE Conference on Computer Vision and Pattern Recognition) 2017에서 발표된 시멘틱 분할 알고리즘

▼ 그림 6-52 PSPNet



7.3 이미지 분할을 위한 신경망

- PSPNet
  - PSPNet 역시 완전연결층의 한계를 극복하기 위해 피라미드 풀링 모듈을 추가했으며 훈련 과정은 다음과 같음
    - 1. 이미지 출력이 서로 다른 크기가 되도록 여러 차례 풀링을 함. 즉, 1×1, 2×2, 3×3, 6×6 크기로 풀링을 수행하는데, 이때 각각 다른 크기의 특성 맵은 서로 다른 영역들의 정보를 담는다고 이해하면 됨
    - 2. 이후 1×1 합성곱을 사용하여 채널 수를 조정. 풀링층 개수를 N이라고 할 때 출력 채널 수=입력 채널 수/N이 됨
    - 3. 이후 모듈의 입력 크기에 맞게 특성 맵을 업샘플링. 이 과정에서 양선형 보간법(bilinear interpolation)이 사용
    - 4. 원래의 특성 맵과 1~3 과정에서 생성한 새로운 특성 맵들을 병합

7.3 이미지 분할을 위한 신경망

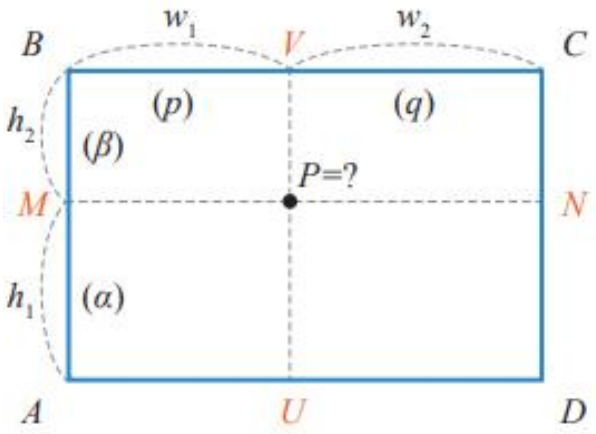
- PSPNet
  - 양선형 보간법
    - 보간법(interpolation)이란 화소 값을 할당받지 못한 영상(예: 영상의 빈 공간)의 품질은 안 좋을 수밖에 없는데, 이때 빈 화소에 값을 할당하여 좋은 품질의 영상을 만드는 방법을 의미
    - 보간법에는 선형 보간법(linear interpolation)과 양선형 보간법(bilinear interpolation)이 있음
    - 선형 보간법은 원시 영상의 화소 값 두 개를 사용하여 원하는 좌표에서 새로운 화소 값을 계산하는 방법
    - 반면 양선형 보간법은 화소당 선형 보간을 두 번 수행하며, 새롭게 생성된 화소는 가장 가까운 화소 네 개에 가중치를 곱한 값을 합해서 얻음

7.3 이미지 분할을 위한 신경망

- PSPNet
  - 양선형 보간법
    - 점 P에서 x축 방향으로 거리를 w_1, w_2라고 하며, y축 방향으로 거리를 h_1, h_2라고 하자
    - 이때 알려진 네 점에서 데이터 값을 A, B, C, D라고 할 때, 양선형 보간법에 따라 점 P의 값은 다음과 같이 계산(단 $\alpha = h_1/(h_1+h_2)$, $\beta = h_2/(h_1+h_2)$, $p = w_1/(w_1+w_2)$, $q = w_2/(w_1+w_2)$)

$$P = q(\beta A+\alpha B) + p(\beta D+\alpha C)$$
$$= q\beta A + q\alpha B + p\beta D + p\alpha C$$

▼ 그림 6-53 양선형 보간법



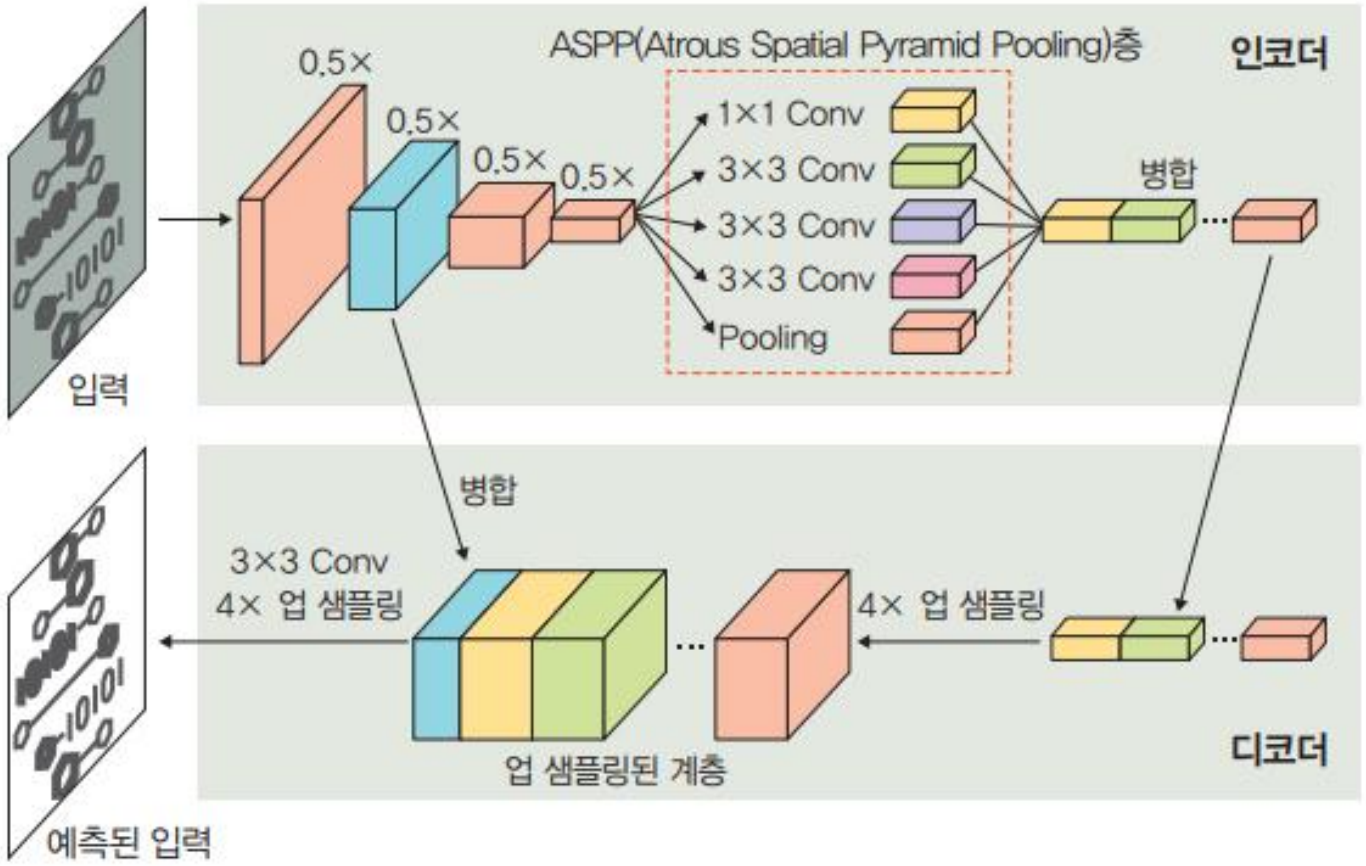
7.3 이미지 분할을 위한 신경망

- DeepLabv3/DeepLabv3+
  - DeepLabv3/DeepLabv3+ 역시 완전연결층의 단점을 보완하기 위해 Atrous 합성곱을 사용하는 네트워크
  - 인코더과 디코더 구조를 가지며, 일반적으로 인코더-디코더 구조에서는 불가능했던 인코더에서 추출된 특성 맵의 해상도를 Atrous 합성곱을 도입하여 제어할 수 있도록 했음

7.3 이미지 분할을 위한 신경망

- DeepLabv3/DeepLabv3+

▼ 그림 6-54 DeepLab의 인코더-디코더 구조



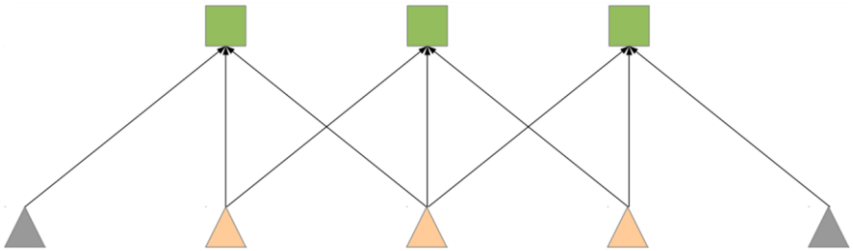
7.3 이미지 분할을 위한 신경망

- DeepLabv3/DeepLabv3+
  - Atrous 합성곱은 다음 그림과 같이 필터 내부에 빈 공간을 둔 채로 작동
  - 얼마나 많은 빈 공간을 가질지 결정하는 파라미터로 (dilation) rate가 있음
  - rate r=1일 경우 기존 합성곱과 동일하게 빈 공간을 가지며, r이 커질수록 빈 공간은 더 많아짐

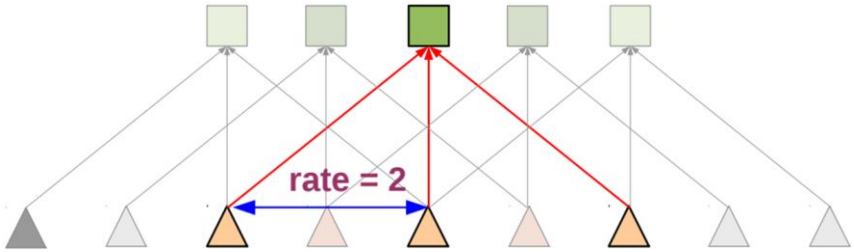
▼ 그림 6-55 Atrous 합성곱

7.3 이미지 분할을 위한 신경망

- DeepLabv3/DeepLabv3+
  - Atrous 합성곱은 필터의 수용 영역이 커지지만 패딩과 스트라이드를 통해 출력 크기가 입력 크기 대비 줄어들지 않도록 조절함



일반 합성곱 (입력3→출력3)



7.3 이미지 분할을 위한 신경망

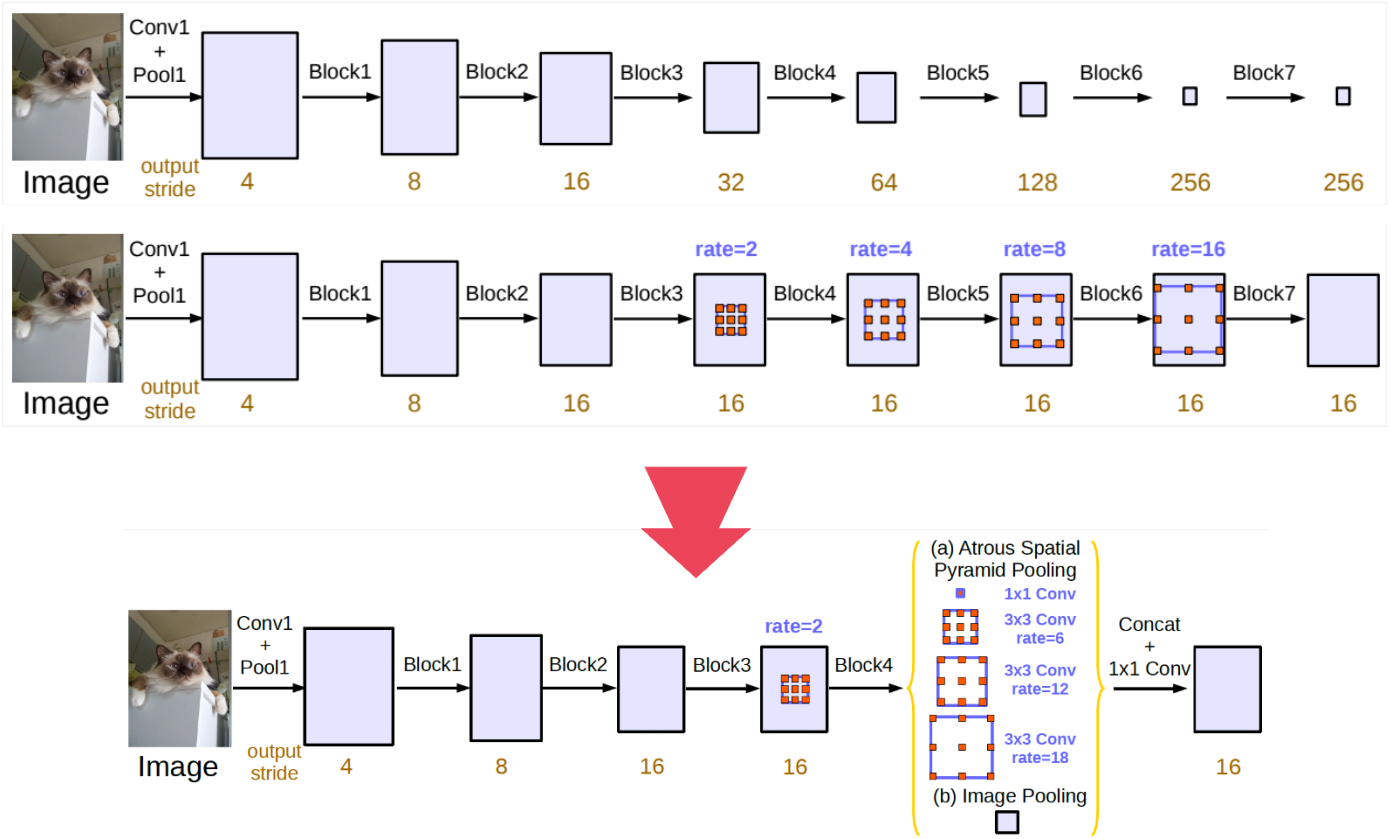
- DeepLabv3/DeepLabv3+
  - 보통 이미지 분할에서 높은 성능을 내려면 수용 영역(receptive field)의 크기가 중요한데, 수용 영역을 확대하여 특성을 찾는 범위를 넓게 해 주기 때문임
  - Atrous 합성곱을 활용하면 파라미터 수를 늘리지 않으면서도 수용 영역을 크게 키울 수 있기 때문에 이미지 분할 분야에서 많이 사용
  - Atrous Spatial Pyramid Pooling을 통해 다양한 수용 영역의 특징을 병렬로 추출함으로써, 파라미터 증가 없이 연산을 병렬화하여 고해상도 이미지 분할에서 효율성과 정확도를 동시에 향상시킴

7.3 이미지 분할을 위한 신경망

- DeepLabv3/DeepLabv3+

▼ 그림 6-56 Atrous 합성곱 효과

출력 스트라이드는 원본 입력 이미지 크기 대비 출력 특성 맵의 크기 비율의 역수



경청해주셔서 감사합니다.