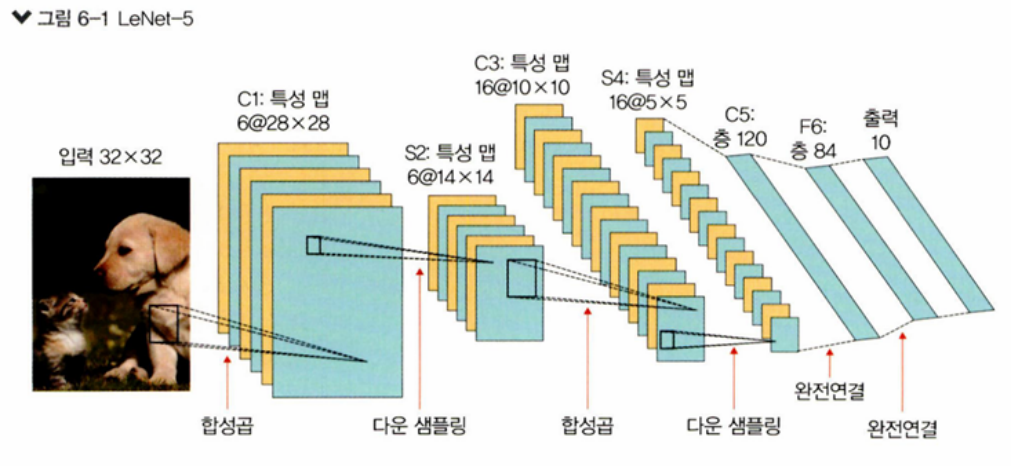
6장 합성곱 신경망 2

생성일	@2024년 9월 22일 오후 4:06
주차	Week 3
완료여부	☐

6.1 이미지 분류를 위한 신경망

- 입력데이터로 이미지를 사용한 분류는 특성 대상이 영상 내에 존재하는지 여부를 판단하는 것

6.1.1 LeNet -5

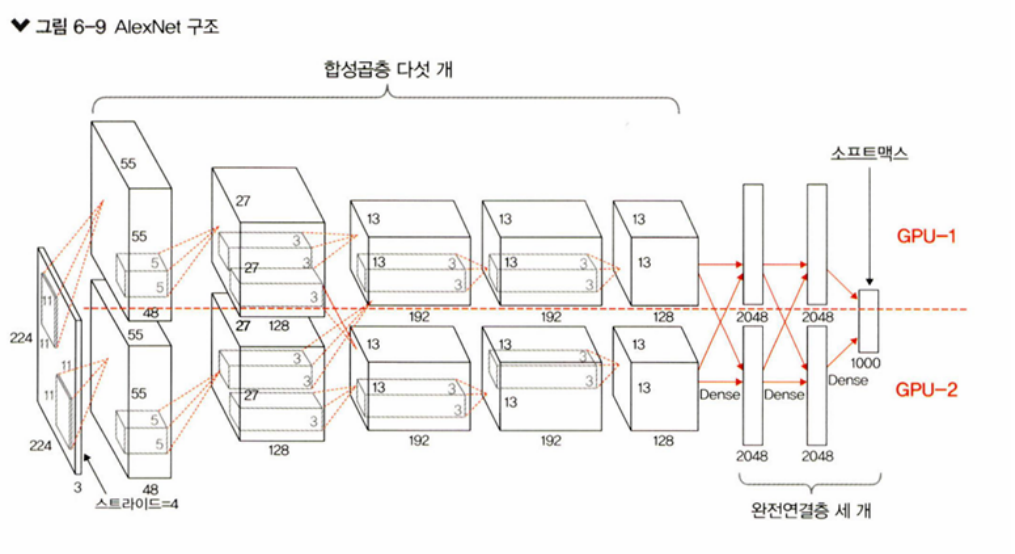


- CI에서 5x5 합성곱 연산 후 28x28 크기의 feature map을 6개 생성함. S2에서 다운샘플링하여 특성맵의 크기를 14x14로 줄이고 C3에서 다시 5x5 합성곱 연산하여 10x10 크기의 feature map 16개를 생성하고 S4에서 다시 다운샘플링하여 5x5 크기로 줄임. C5에서 다시 5x5 합성곱 연산하여 1x1 크기그이 특성맵 120개를 생성하고 마지막 F6에서 완전연결층으로 C5의 결과를 노드 84개에 연결시킴.

6.1.2 AlexNet

- CNN은 3차원 구조를 가짐(너비, 높이, 깊이).
- Alexnet은 합성곱층 5개와 완전연결층 3개로 구성되어 있고 맨 마지막 완전연결층은 카테고리 1000개를 분류하기 위한 소프트맥스 활성화 함수를 사용하고 있음

❤ 그림 6-9 AlexNet 구조



❤ 표 6-2 AlexNet 구조 상세

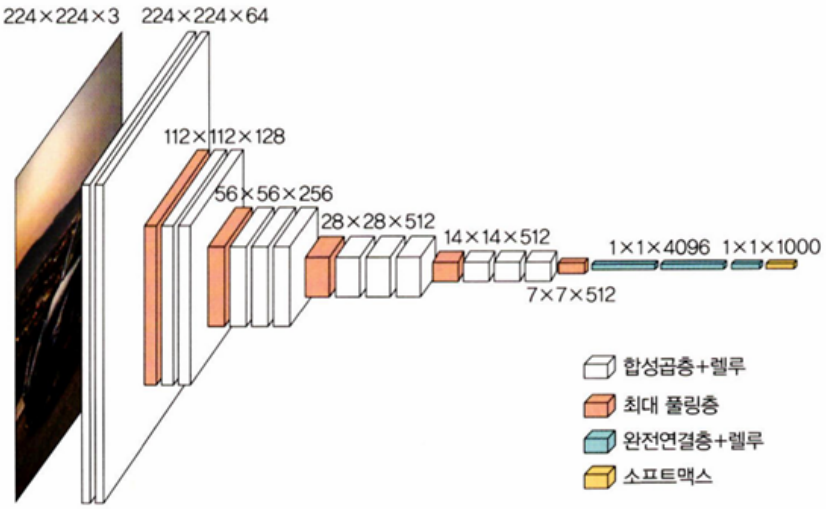
계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
이미지	1	227×227	–	–	–
합성곱층	96	55×55	11×11	4	렐루(ReLU)
최대 풀링층	96	27×27	3×3	2	–
합성곱층	256	27×27	5×5	1	렐루(ReLU)
최대 풀링층	256	13×13	3×3	2	–
합성곱층	384	13×13	3×3	1	렐루(ReLU)
합성곱층	384	13×13	3×3	1	렐루(ReLU)
합성곱층	256	13×13	3×3	1	렐루(ReLU)
최대 풀링층	256	6×6	3×3	2	–
완전연결층	–	4096	–	–	렐루(ReLU)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

- AlexNet의 활성화 함수는 ReLU이고 첫번째 합성곱층 커널의 크기는 11x11×3이고 스트라이드 4를 적용하여 특성맵 96개를 생성하기 때문에 55x55×96의 출력을 가짐
- 첫번째 계층을 거치면서 GPU-1에서는 주로 컬러와 상관없는 정보를 추출하기 위한 커널이 학습되고 GPU-2에서는 컬러와 관련된 정보를 추출하기 위한 커널이 학습됨

6.1.3 VGGNet

- 네트워크 계층의 총 개수에 따라 VGGNet 16,19 등이 있고 이중 VGG16을 상요할 것
- 1억개가 넘는 파라미터가 있는데 여기서 주의할 것은 모든 합성곱 커널의 크기는 3x3, 최대풀링 커널의 크기는 2x2, 스트라이드는 2라는 것. 또한 마지막 16번째 계층을 제외하곤은 모두 ReLU 활성화 함수가 적용됨

❤ 그림 6-13 VGG16 구조



❤ 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)
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	–

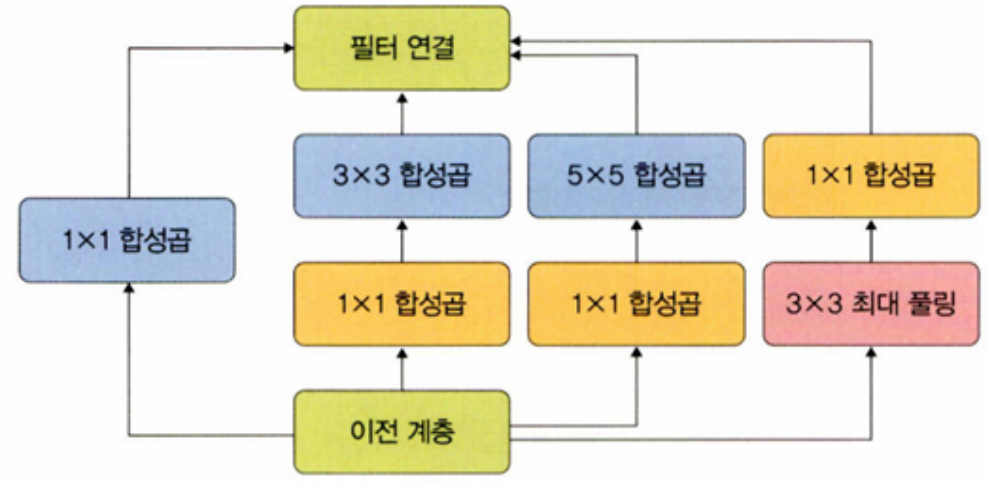
계속

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)

6.1.4 Google Net

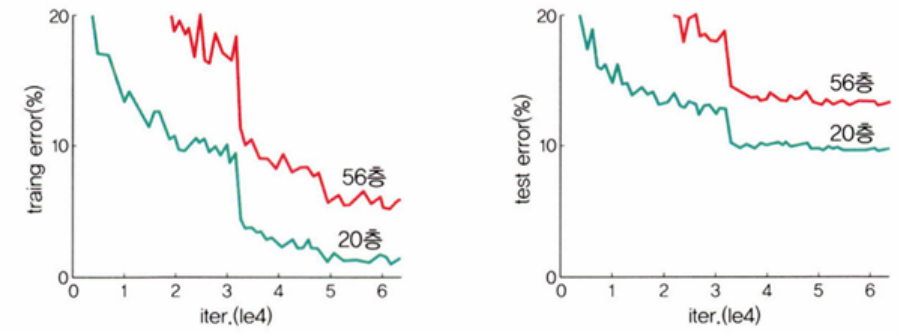
- 깊고 넓은 신경망을 위해 inception을 모듈로 추가함
- 이는 특징을 효율적으로 추출하기 위해 1x1, 3×3, 5×5 합성곱 연산을 각각 수행함
- 3×3 최대 풀링은 입력과 출력의 높이와 너비가 같아야 하므로 풀링 연산에서는 드물게 패딩을 추가해야 함
- 결과적으로 googlenet에 적용된 해결 방밥은 sparse connection임

그림 6-23 GoogLeNet의 인셉션 모듈



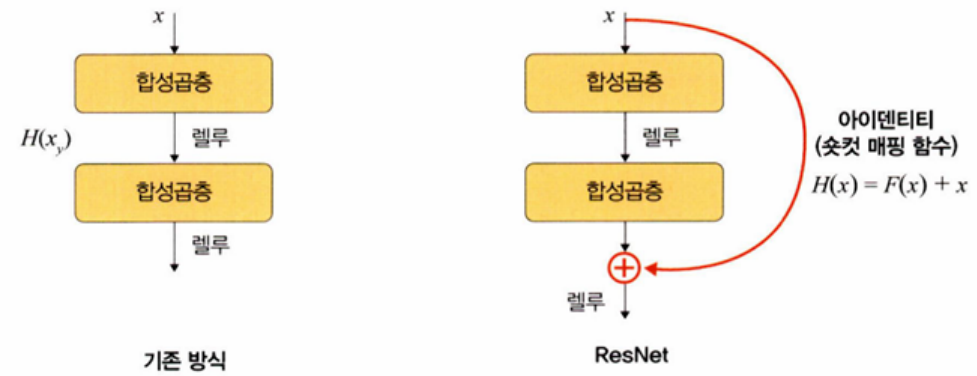
6.1.5 ResNet

❯ 그림 6-24 네트워크 56층이 20층보다 더 나쁜 성능을 보임



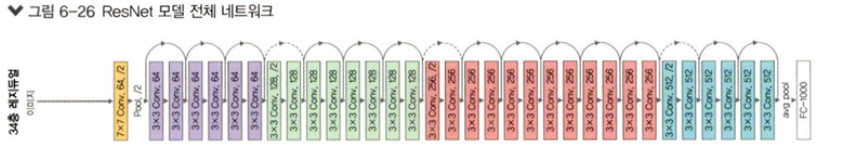
- 네트워크 깊이가 깊다고 해서 무조건 성능이 좋아지는 것은 아님 → 기울기 소멸 문제가 발생할 수 있음
- Residual Block을 도입해서 기울기가 잘 전파될 수 있도록 shortcut을 만들어줌

❯ 그림 6-25 ResNet 구조

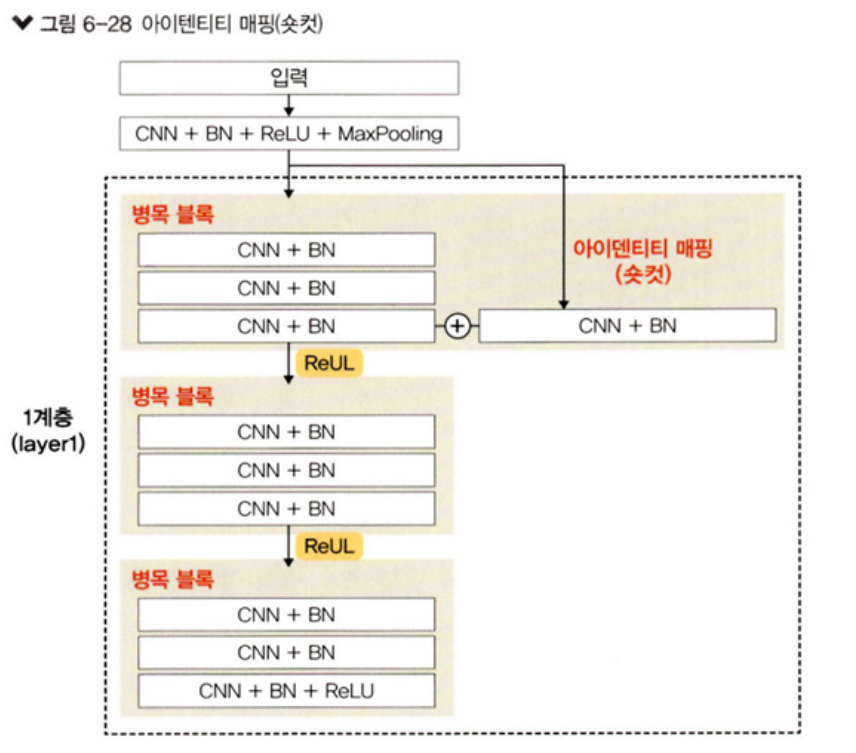


- Block은 계층의 묶음으로 합성곱층 하나를 블록으로 묶은 것. 이렇게 묶인 block의 계층들을 하나의 residual block이라고 함
- 계층을 계속해서 쌓아올리면 파라미터가 너무 많이 증가하는 문제가 생김. 이를 위해 bottleneck block를 만듦
- ResNet34에서 사용하는 1x1 합성곱층의 채널 수를 조절하면 차원을 늘렸다 줄이는 것이 가능하기 때문에 파라미터 수를 줄일 수 있음

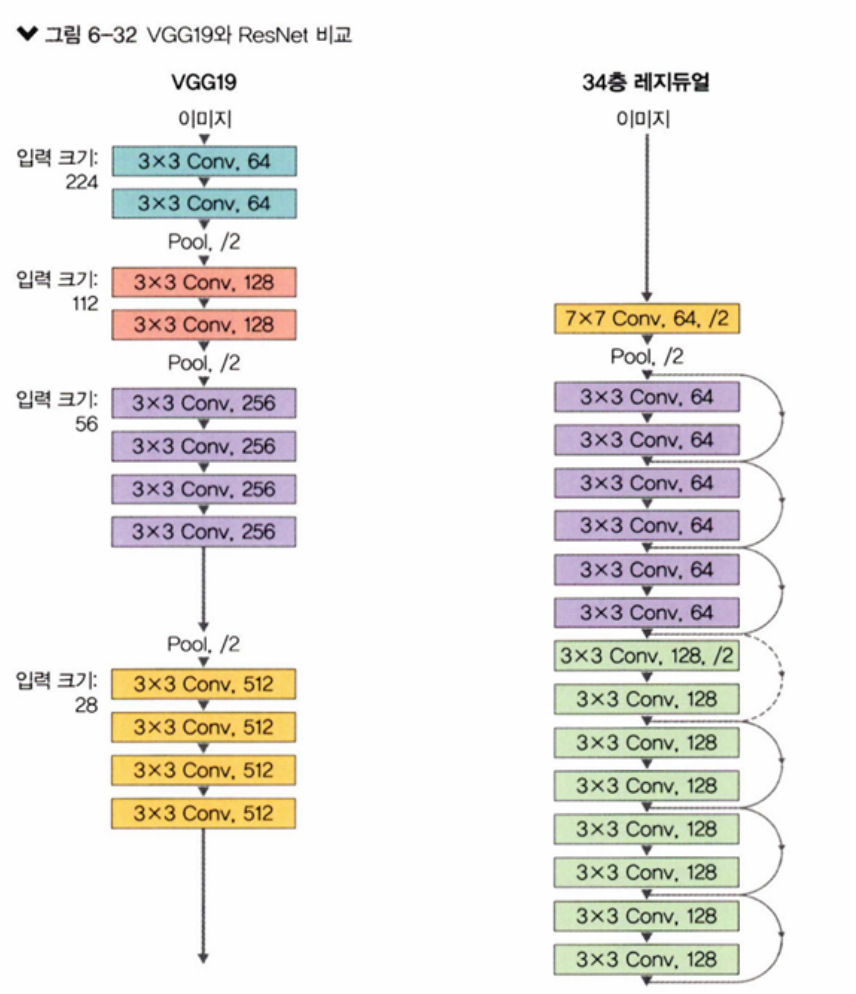
❯ 그림 6-26 ResNet 모델 전체 네트워크



- Identity Mapping/ Shortcut은 입력 x가 어떤 함수를 통과하더라고 다시 x란은 형태로 출력이 되게 함



- DownSampling이란 featuremap의 크기를 줄이는 pooling과 같은 역할임



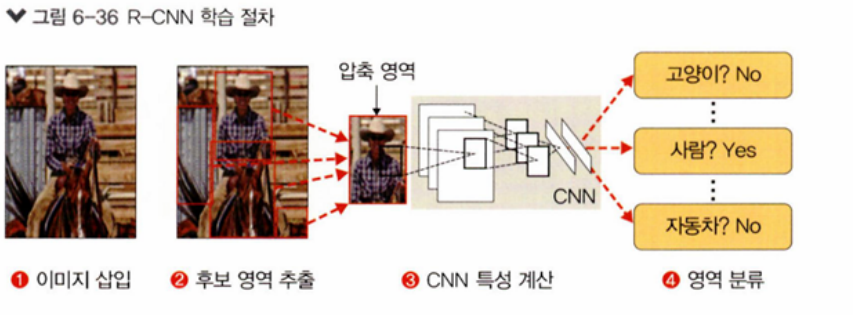
6.2 객체 인식을 위한 신경망

- 객체 인식이란 이미지나 영상 내에 있는 객체를 식별하는 컴퓨터 비전 기술로 각 객체가 무엇인지 분류하는 문제와 그 객체의 위치가 어디인지 bouding box와 localization을 다루는 분야임
- 1 stage detector와 2 stage detector로 객체 인식 알고리즘을 나누는데 1 stage detector은 분류와 localization을 동시에 행하는 것이고 2 stage detector은 순차적으로 행하는 방식힘
- 1 stage detector의 예시로는 yolo가 있고 2 stage detector의 예시로는 R-CNN 등이 있음

6.2.1 R-CNN

- 객체 인식 알고리즘은 sliding window 방식으로 일정한 크기를 가지는 window가 이미지의 모든 영역을 탐색하면서 객체를 검출해내는 방식임
- 하지만 알고리즘의 비효율성 때문에 현재는 selective search와 region proposal 방식을 많이 사용함

- R-CNN은 이미지 분류를 수행하는 CNN과 이미지에서 객체가 있을만한 영역을 제안해주는 후보 영역 알고리즘을 결합한 알고리즘 방식임



→ 1) 이미지를 입력으로 받아 2) 2000개 정도의 bounding box를 selective search 알고리즘으로 추출한 후 잘라내고 CNN 모델에 넣기 위해 크기를 통일시킴 2) 크기가 동일한 2000개의 이미지에 대해 CNN 모델을 적용하여 각각 분류를 진행해 결과를 도출함

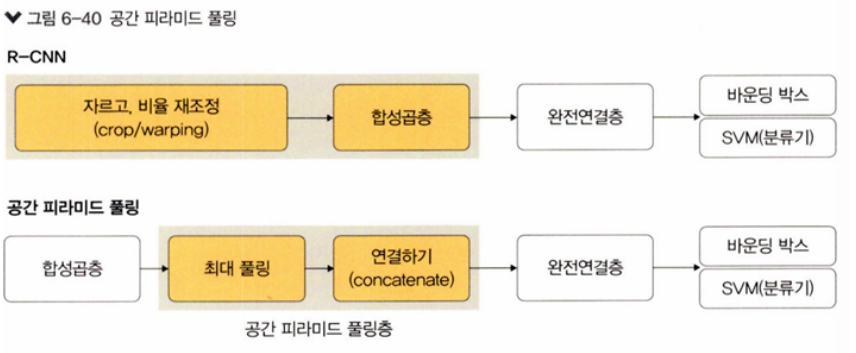
? selective search란

- 객체 인식이나 검출을 위한 후보 영역을 알아내는 방법으로 각각의 객체가 한개 영역에 할당될 수 있도록 많은 초기 영역을 생성한 후 입력된 이미지를 영역 다수개로 분리하는 과정
- 영역을 여러개로 나눈 것을 비슷한 영역으로 통합하는데 이때 greedy 알고리즘을 사용하여 비슷한 영역이 하나로 통합될 수 있도록 함

R-CNN의 단점(복잡한 학습 과정. 긴 학습 시간과 대용량 저장 공간, 객체 검출 속도 문제)로 크게 발전하지 못함

6.2.2 공간 피라미드 풀링

- 기존의 CNN 구조는 완전 연결층을 위해 입력 이미지를 고정해야 했기 때문에 본래의 생김새와 달라지는 문제점이 있었음

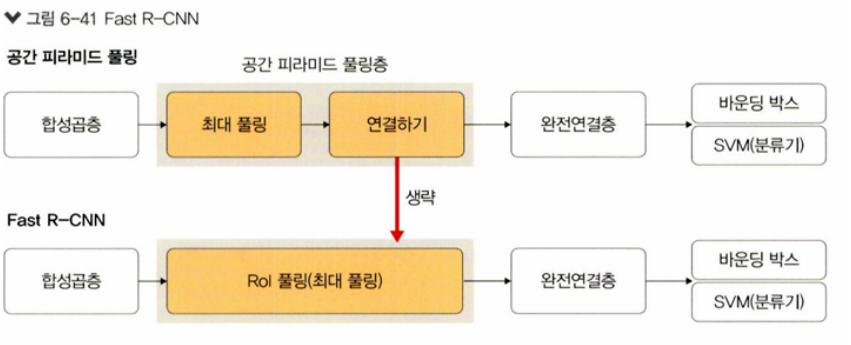


- 공간 피라미드 풀링은 입력 이미지의 크기에 관계없이 합성곱층을 통과시키고, 완전 연결층에 전달되기 전에 특성맵을 동일한 크기로 조절해주는 풀링층을 적용하는 기법

- 이미지 훼손이 적어지는 장점이 있음

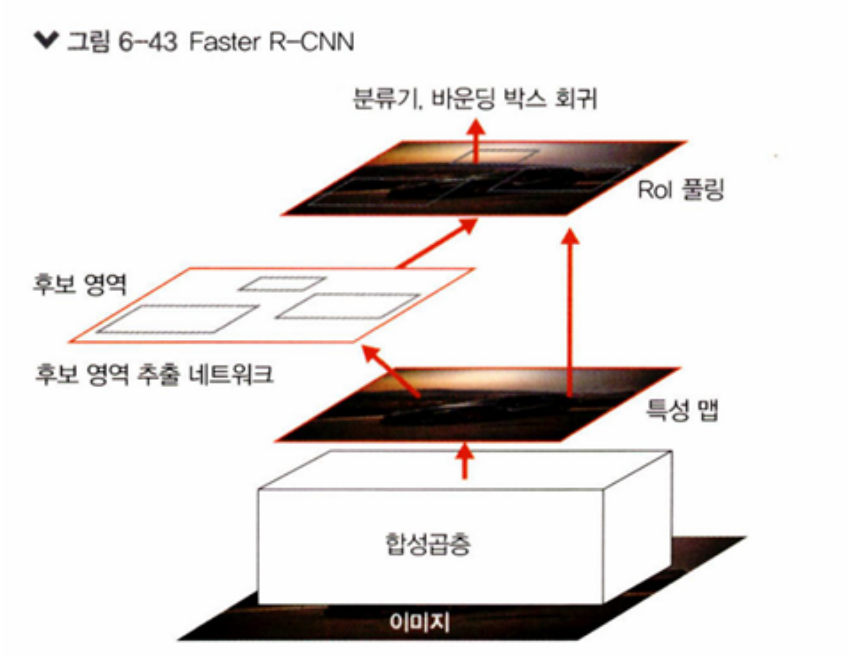
6.2.3 Fast R-CNN

- bounding box마다 CNN을 돌려 긴 학습 시간이 필요한 R-CNN 문제를 해결하기 위해 RoI 풀링을 도입함
- RoI 풀링은 크기가 다른 특성맵의 영역마다 스트라이드를 다르게 최대 풀링을 적용하여 결과값의 크기를 동일하게 맞추는 방법으로 최대 풀링과 비슷한 매커니즘



6.2.4 Faster R-CNN

- 기존의 Fast R-CNN의 속도를 낮추게 하는 원인인 후보영역 생성을 CNN 내부 네트워크에서 진행할 수 있도록 설계함



- 후보 영역 추출 네트워크는 feature map nxn 크기의 작은 윈도우 영역을 입력으로 받고 해당 영역의 객체 존재 유무 판단을 위해 binary classification을 수행하는 작은 네

트워크를 생성함

- 하나의 feature map에 대해 모든 영역에 대한 객체 존재 여부를 판단하기 위해 슬라이딩 윈도우 방식으로 탐색함
- 이미지에 존재하는 객체의 크기와 비율이 다양하다는 점을 고려하여 여러 크기와 비율의 reference box를 k개 미리 정의하고 각각의 슬라이딩 윈도우 위치마다 k개를 출력하도록 설계함 → aka anchor 방식