6장
합성곱 신경망 II

6장 합성곱 신경망 II

6.1 이미지 분류를 위한 신경망

6.2 객체 인식을 위한 신경망

6.3 이미지 분할을 위한 신경망

6.1 이미지 분류를 위한 신경망

6.1 이미지 분류를 위한 신경망

- **이미지 분류를 위한 신경망**
  - 입력 데이터로 이미지를 사용한 분류(classification)는 특정 대상이 영상 내에 존재하는지 여부를 판단하는 것
  - 이미지 분류(image classification)에서 주로 사용되는 합성곱 신경망의 유형을 알아보자

6.1 이미지 분류를 위한 신경망

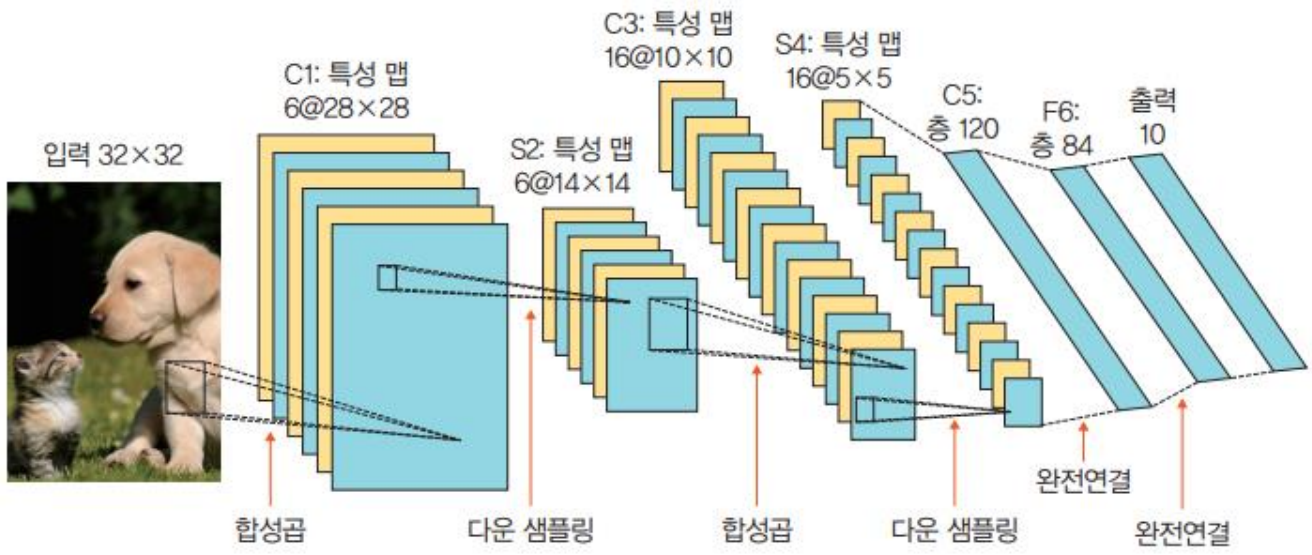
- **LeNet-5**
  - LeNet-5는 합성곱 신경망이라는 개념을 최초로 얀 르쿤(Yann LeCun)이 개발한 구조
  - LeNet-5는 합성곱(convolutional)과 다운 샘플링(sub-sampling)(혹은 풀링)을 반복적으로 거치면서 마지막에 완전연결층에서 분류를 수행

6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - 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

6.1 이미지 분류를 위한 신경망

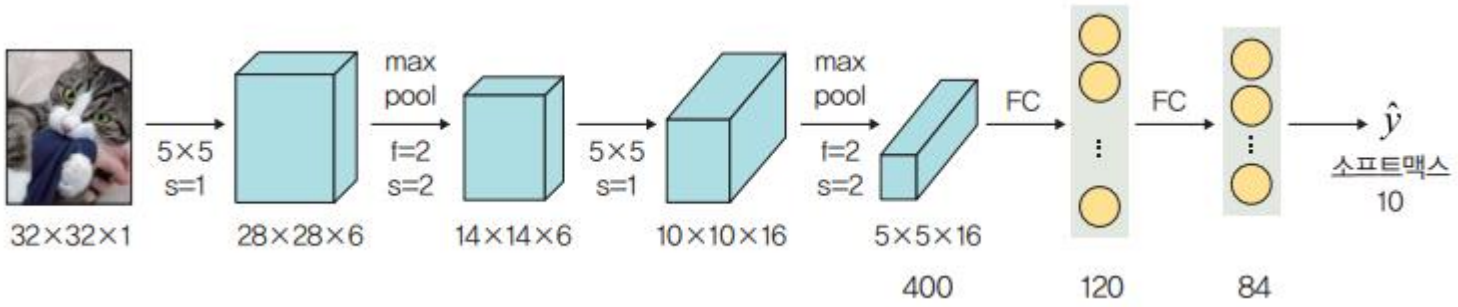
▼ **그림 6-1 LeNet-5**



6.1 이미지 분류를 위한 신경망

- 32×32 크기의 이미지에 합성곱층과 최대 풀링층이 쌍으로 두 번 적용된 후 완전연결층을 거쳐 이미지가 분류되는 신경망

▼ **그림 6-2 LeNet-5 예제 신경망**



6.1 이미지 분류를 위한 신경망

▼ 표 6-1 LeNet-5 예제 신경망 상세

계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
이미지	1	32×32	–	–	–
합성곱층	6	28×28	5×5	1	렐루(ReLU)
최대 풀링층	6	14×14	2×2	2	–
합성곱층	16	10×10	5×5	1	렐루(ReLU)
최대 풀링층	16	5×5	2×2	2	–
완전연결층	–	120	–	–	렐루(ReLU)
완전연결층	–	84	–	–	렐루(ReLU)
완전연결층	–	2	–	–	소프트맥스(softmax)

6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - 네트워크의 각 부분들을 통과할 때마다 입력과 출력의 형태가 바뀌는데, 그 계산은 다음 수식과 같음

Conv2d 계층에서의 출력 크기 구하는 공식

- 출력 크기=(W-F+2P)/S+1

  - **W**: 입력 데이터의 크기(input_volume_size)
  - **F**: 커널 크기(kernel_size)
  - **P**: 패딩 크기(padding_size)
  - **S**: 스트라이드(strides)

6.1 이미지 분류를 위한 신경망

- **LeNet-5**

  MaxPool2d 계층에서의 출력 크기 구하는 공식

  - 출력 크기=IF/F

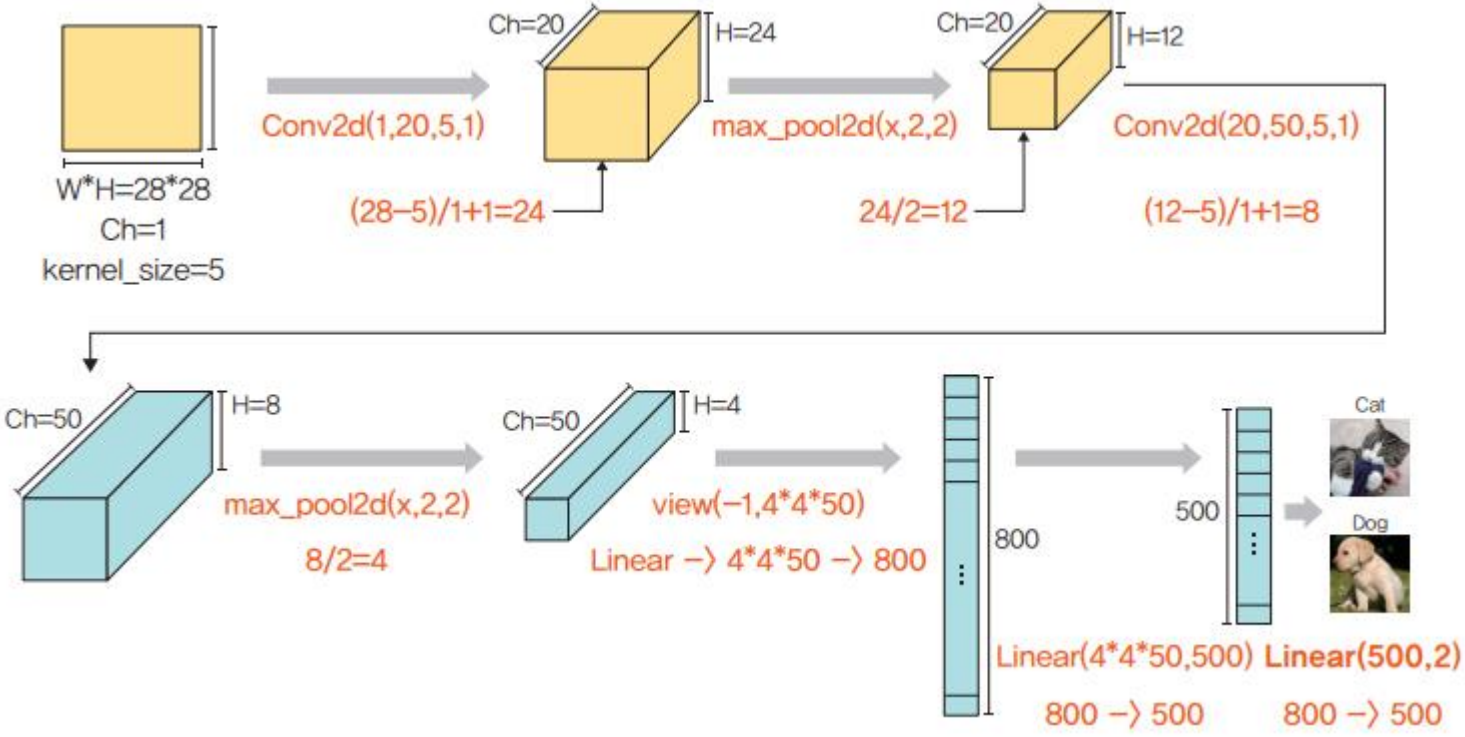
    • IF: 입력 필터의 크기(input_filter_size, 또한 바로 앞의 Conv2d의 출력 크기이기도 함)

    • F: 커널 크기(kernel_size)

6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - 각각의 계층에 대한 결과는 다음 그림과 같음

▼ **그림 6-5 합성곱층과 풀링층의 크기(형태) 계산**



6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - LeNet()을 model이라는 이름으로 객체를 생성하여 모델 학습을 위한 준비를 함

코드 6-11 모델 객체 생성

```
model = LeNet()
print(model)
```

6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - 다음은 생성한 model에 대한 출력 결과

```
LeNet(
  (cnn1): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1))
  (relu1): ReLU()
  (maxpool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (cnn2): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
  (relu2): ReLU()
  (maxpool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=89888, out_features=512, bias=True)
  (relu5): ReLU()
  (fc2): Linear(in_features=512, out_features=2, bias=True)
  (output): Softmax(dim=1)
)
```

6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - 출력 결과가 한눈에 들어오지 않는다면 torchsummary 라이브러리를 사용해 볼 수 있음
  - torchsummary는 케라스와 같은 형태로 모델을 출력해 볼 수 있는 라이브러리
  - 먼저 아나콘다 프롬프트(Anaconda prompt)에서 다음 명령을 실행하여 torchsummary 라이브러리를 설치

```
> pip install torchsummary
```

6.1 이미지 분류를 위한 신경망

- **LeNet-5**
  - 앞에서 말했듯이 캐글에서 내려받은 데이터 전체를 사용한다면 성능이 높아지겠지만 CPU를 사용한다면 훈련 시간은 며칠이 걸릴 수도 있음
  - 지금까지 LeNet의 코드를 살펴보았고 이미지 분류에서 사용되는 또 다른 모델인 AlexNet에 대해 살펴보자
  - 이제부터 AlexNet을 포함한 여러 유형의 컴퓨터 비전 모델들을 살펴볼 텐데 LeNet에서 사용했던 코드와 유사한 부분이 많을 것
  - 모델의 네트워크 위주로 CNN의 다양한 모델들을 살펴보면 학습에 도움이 많이 될 것

6.1 이미지 분류를 위한 신경망

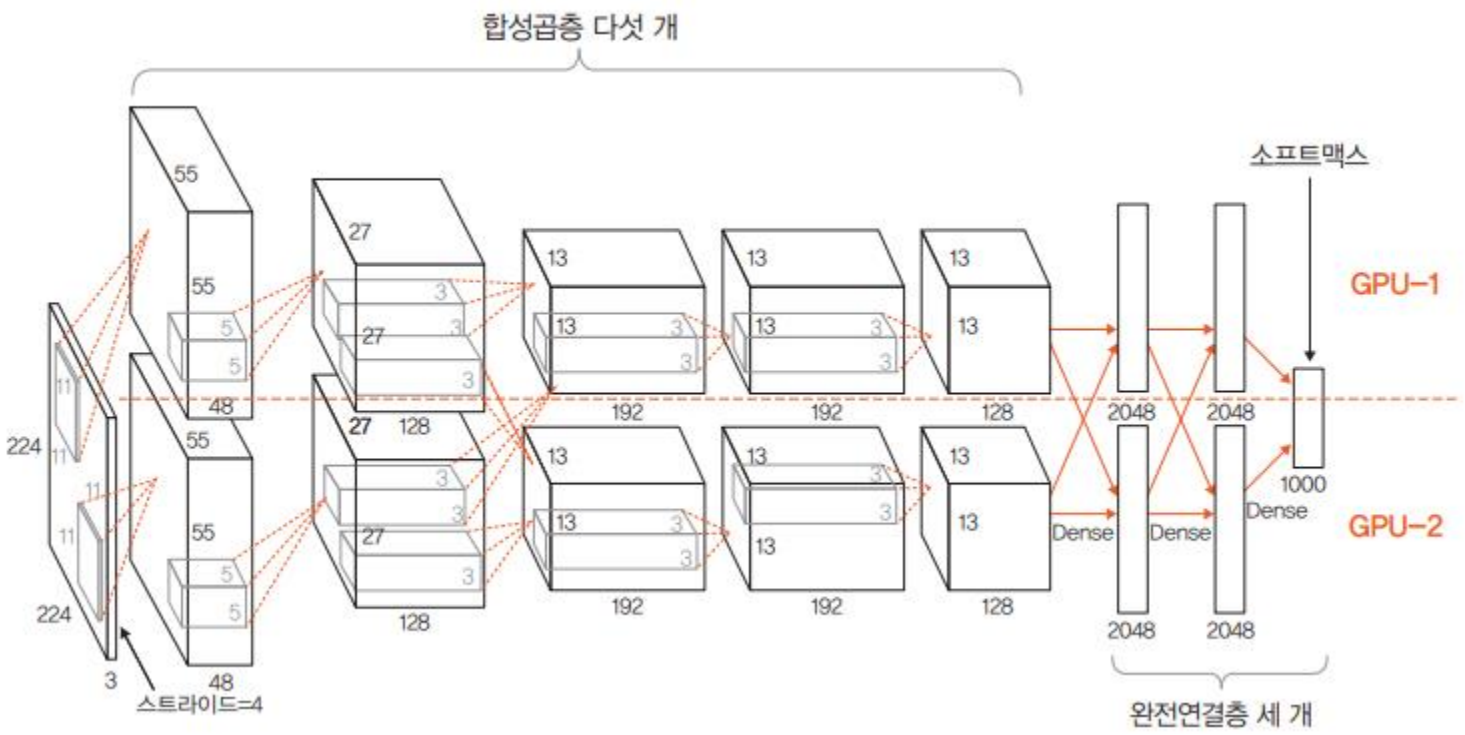
- **AlexNet**
  - AlexNet은 ImageNet 영상 데이터베이스를 기반으로 한 화상 인식 대회인 'ILSVRC 2012'에서 우승한 CNN 구조
  - AlexNet을 설명하기에 앞서 AlexNet의 세부 블록을 이해하고자 CNN 구조를 다시 살펴보자
  - CNN은 다음 그림과 같이 3차원 구조를 갖는다는 것을 이해해야 함(이미지를 다루기 때문에 기본적으로 3차원 데이터를 다룸)
  - 이미지 크기를 나타내는 너비(width)와 높이(height)뿐만 아니라 깊이(depth)를 갖음
  - 보통 색상이 많은 이미지는 R/G/B 성분 세 개를 갖기 때문에 시작이 3이지만, 합성곱을 거치면서 특성 맵이 만들어지고 이것에 따라 중간 영상의 깊이가 달라짐

6.1 이미지 분류를 위한 신경망

- **AlexNet**
  - AlexNet 구조에 있는 숫자 의미에 대한 이해가 가능
  - 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

6.1 이미지 분류를 위한 신경망

▼ **그림 6-9 AlexNet 구조**



6.1 이미지 분류를 위한 신경망

- **AlexNet**
  - AlexNet의 합성곱층에서 사용된 활성화 함수는 렐루(ReLU)로, 각 계층의 구조적 세부 사항은 다음 표를 참고

6.1 이미지 분류를 위한 신경망

▼ **표 6-2 AlexNet 구조 상세**

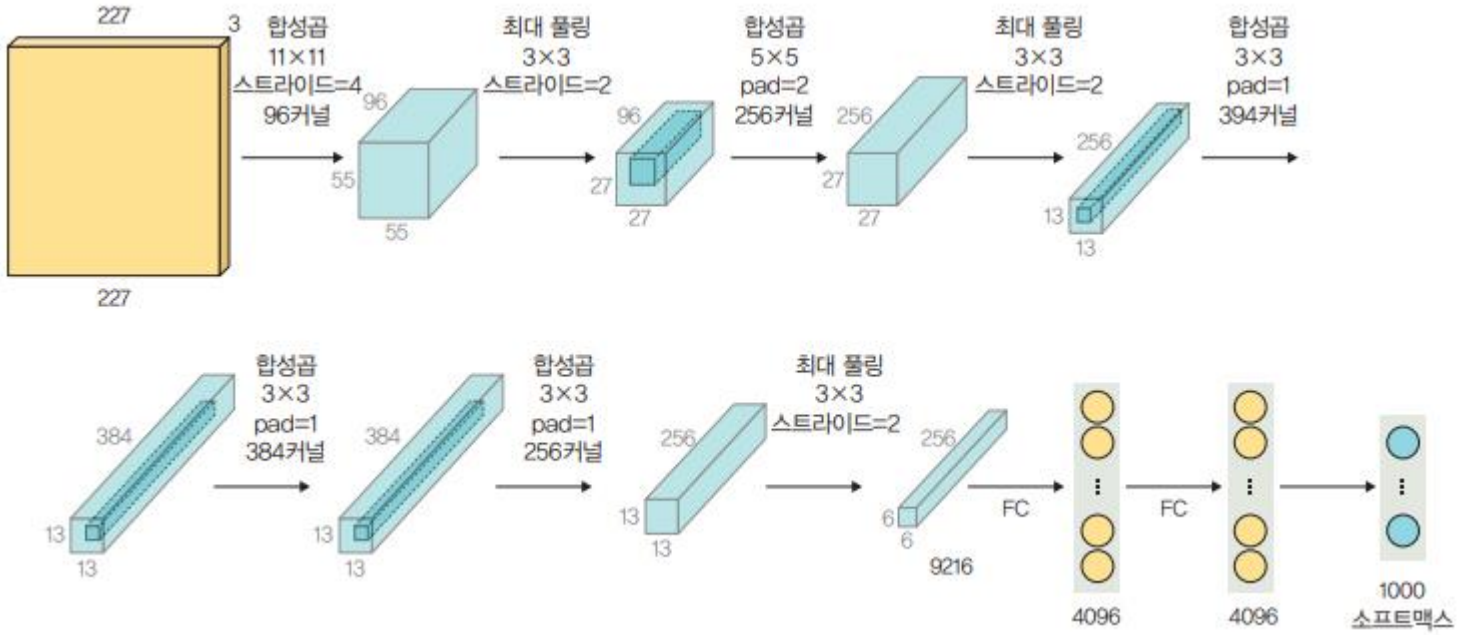
계층 유형	특성 맵	크기	커널 크기	스트라이드	활성화 함수
이미지	1	227×227	–	–	–
합성곱층	96	55×55	11×11	4	렐루(ReLU)
최대 풀링층	96	27×27	3×3	2	–
합성곱층	256	27×27	5×5	1	렐루(ReLU)
최대 풀링층	256	13×13	3×3	2	–
합성곱층	384	13×13	3×3	1	렐루(ReLU)
합성곱층	384	13×13	3×3	1	렐루(ReLU)
합성곱층	256	13×13	3×3	1	렐루(ReLU)
최대 풀링층	256	6×6	3×3	2	–
완전연결층	–	4096	–	–	렐루(ReLU)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

6.1 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
  - 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

6.1 이미지 분류를 위한 신경망

▼ **그림 6-11 AlexNet 예제 네트워크**



6.1 이미지 분류를 위한 신경망

- **AlexNet**
  - nn.AdaptiveAvgPool2d는 nn.AvgPool2d처럼 풀링을 위해 사용
  - AvgPool2d에서는 풀링에 대한 커널 및 스트라이드 크기를 정의해야 동작
  - 예를 들어 nn.AvgPool2d((3, 2), stride=(2, 1))처럼 지정하면 그 결과는 5×5 텐서를 3×3 텐서로, 7×7 텐서를 4×4 텐서로 줄이는 효과를 얻을 수 있음
  - 조금 더 정확히 표현하자면, (N, C, H_{in}, W_{in}) 크기의 입력을 (N, C, H_{out}, W_{out}) 크기로 출력하는 것이 AvgPool2d
  - 이때 H_{out}, W_{out}은 다음과 같은 공식에 의해 계산

$$H_{out} = \left[\frac{H_{in} + 2 \times \text{padding}[0] - \text{kernel\_size}[0]}{\text{stride}[0]} + 1 \right]$$

$$W_{out} = \left[\frac{W_{in} + 2 \times \text{padding}[1] - \text{kernel\_size}[1]}{\text{stride}[1]} + 1 \right]$$

6.1 이미지 분류를 위한 신경망

- **AlexNet**
  - 반면에 AdaptiveAvgPool2d는 풀링 작업이 끝날 때 필요한 출력 크기를 정의
  - 즉, nn.AvgPool2d에서는 커널 크기, 스트라이드, 패딩을 지정했다면 nn.AdaptiveAvgPool2d는 출력에 대한 크기만 지정
  - 앞의 수식에서 H_{out}, W_{out} 값이 정해졌기 때문에 커널 크기, 스트라이드, 패딩 값을 구할 수 있음
  - AdaptiveAvgPool2d를 사용할 경우 출력 크기에 대한 조정이 상당히 쉬워짐
  - 참고로 AdaptiveAvgPool2d는 입력 크기에 변동이 있고 CNN 위쪽에 완전연결층을 사용하는 경우 유용함

6.1 이미지 분류를 위한 신경망

- **AlexNet**
  - 역시 예측 결과가 좋지 않음
  - 앞에서 언급했지만 데이터셋이 많으면 성능이 좋아질 수 있음
  - 성능은 파라미터 튜닝을 통해서도 향상시킬 수 있음

추가내용

6.1 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

6.1 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

6.1 이미지 분류를 위한 신경망

▼ **그림 6-13 VGG16 구조**