14.1 ~ 14.4

나민주

2022. 08. 24.

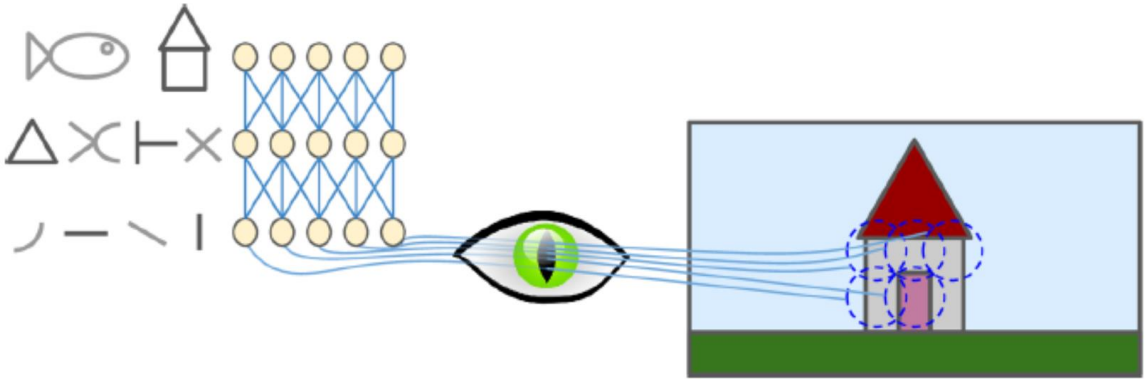
14. 합성곱 신경망을 사용한 검퓨터 비전

합성곱 신경망 (CNN)

- 이미지 인식 분야에 사용되는 딥러닝 알고리즘
- 이미지 검색 서비스, 자율주행 자동차, 영상 자동 분류 시스템 등에 큰 기여
- 시각 분야 외에도 음성인식, 자연어 처리 (NLP) 등에도 사용

14.1 시각 피질 구조

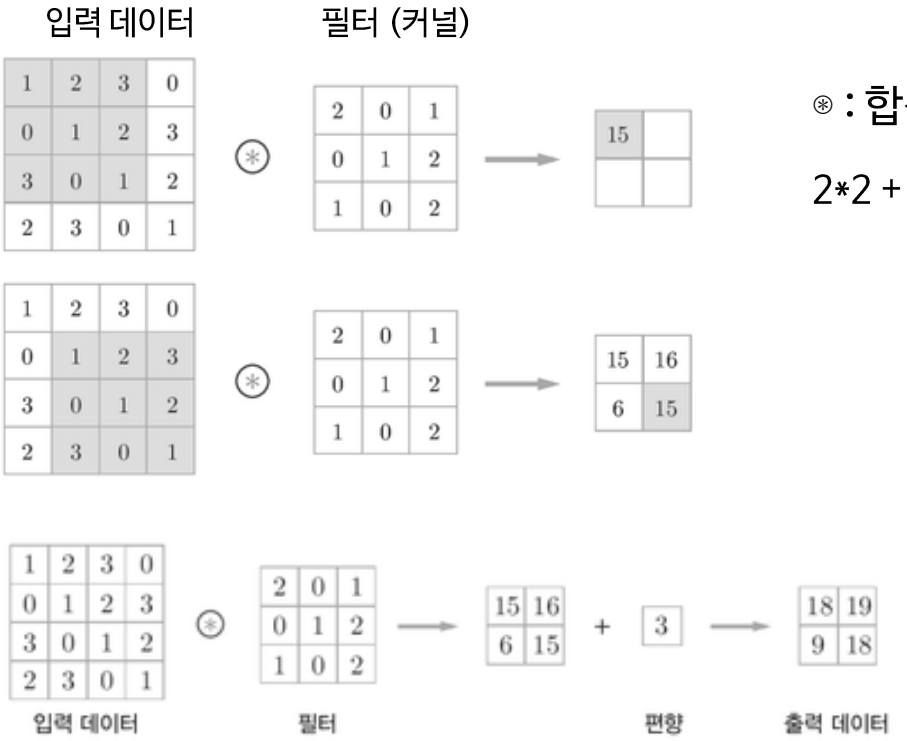
- 시각 피질 구조에 대한 연구 결과, 시각 피질 안의 많은 뉴런이 작은 국부 수용장을 가진다는 것을 발견
- 즉, 뉴런들이 일부 범위 내에 있는 시각 자극에만 반응 한다는 뜻



- 뉴런 수용장들은 서로 겹칠 수 있으며, 합치며 전체 시야를 감싸게 됨
- 수평선의 이미지 / 다른 각도의 선분 / 큰 수용장을 가져서 저수준 패턴이 조합된 복잡한 패턴 등에 반응

→ 고수준 뉴런이 이웃한 저수준 뉴런의 출력에 기반하며, 이러한 구조로 전체 시야 영역의 복잡한 패턴 감지 가능

14. 합성곱 신경망을 사용한 검퓨터 비전



⊛ : 합성곱 연산

2*2 + 3*1 + 1*1 + 2*2 + 3*1 + 1*2 = 15

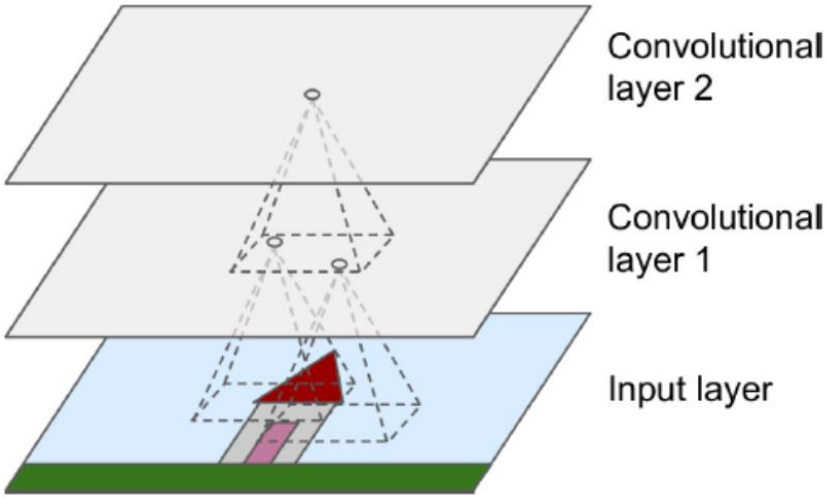
윈도우를 옮기면서 입력과 필터에 대응하는 원소끼리 곱하고 그 총합을 구함
CNN에서는 필터의 매개변수 (원소 값)가 “가중치”에 해당

14. 2 합성곱 층

CNN의 가장 중요한 구성 요소인 합성곱 층

합성곱 : 한 함수가 다른 함수 위를 이동하면서 원소별 곱셈의 적분을 계산하는 수학 연산

- 첫 번째 합성곱 층의 뉴런은 입력 이미지의 모든 픽셀에 연결되지 않고, 합성곱 뉴런의 수용장 안에 있는 픽셀에만 연결
  → 작은 저수준 특성에 집중
- 두 번째 합성곱 층에 있는 각 뉴런은 첫 번째 층의 작은 사각 영역 안에 위치한 뉴런에 연결
  → 더 큰 고수준 특성으로 조합해 나가도록 도와줌

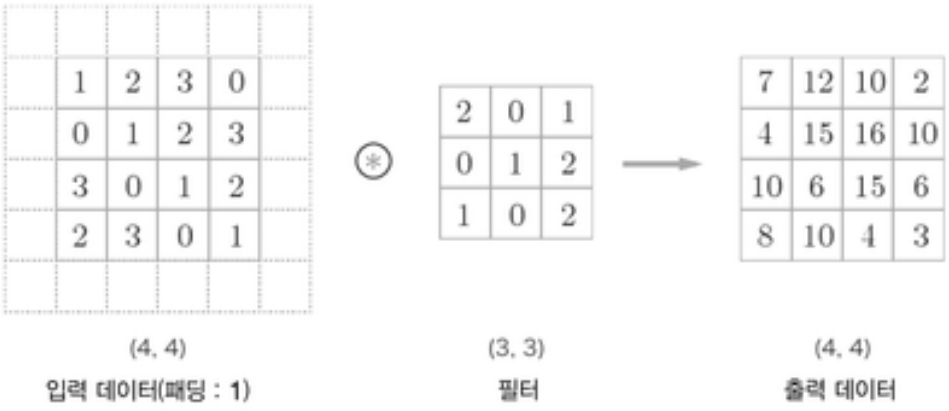


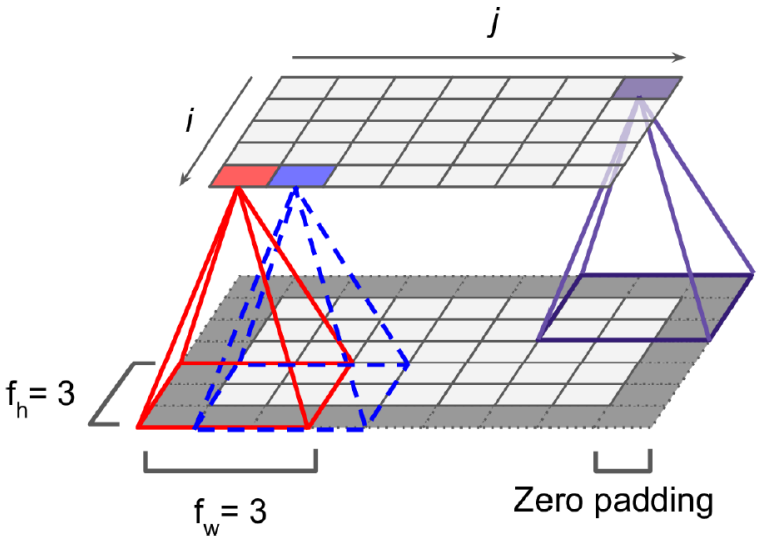
14. 2 합성곱 층

패딩 (Padding)

합성곱 연산 수행 전, 입력 데이터 주변을 특정값으로 채워 늘리는 것

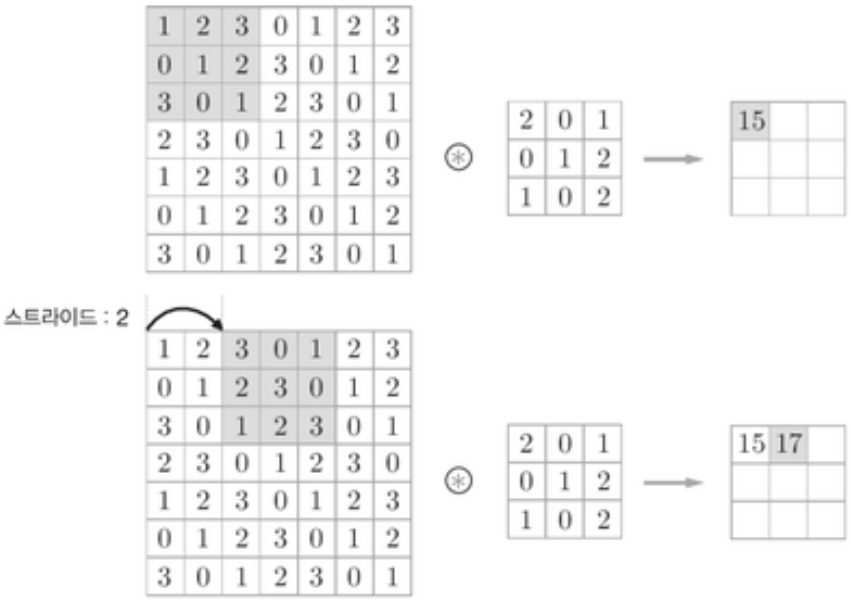
층의 높이와 너비를 이전 층과 같게 하기 위해 입력의 주위에 0을 추가하는 “제로 패딩”이 일반적

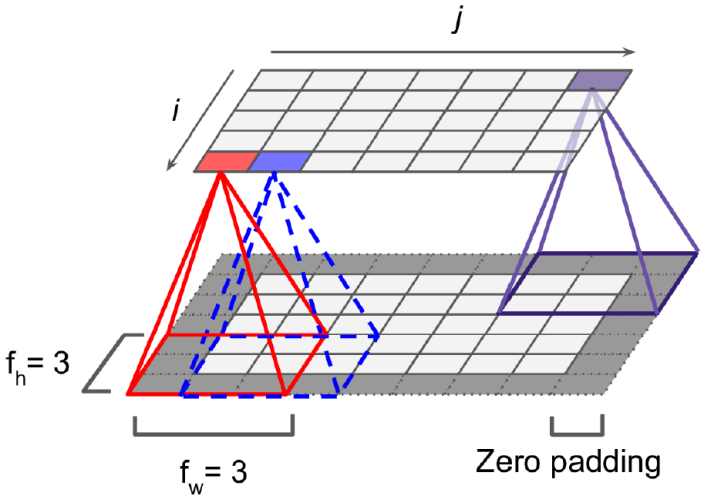




스트라이드 (Stride)

입력데이터에 필터를 적용할 때, 필터가 이동할 간격





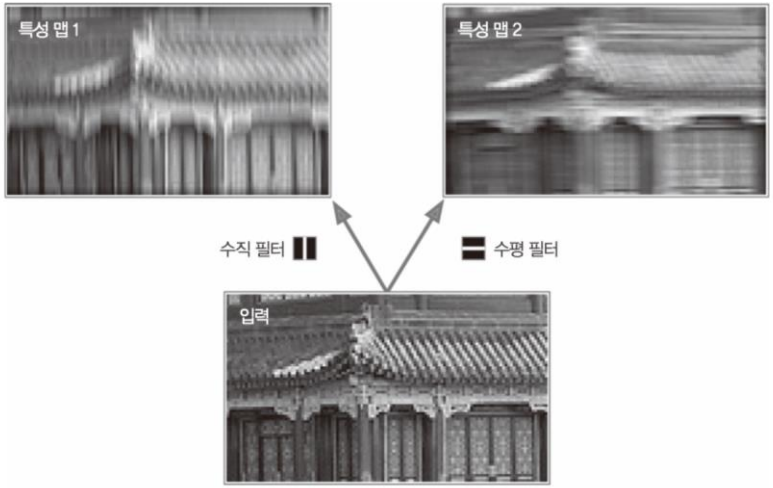
14.2.1 필터

필터 (합성곱 커널) : 입력뉴런에 사용될 **가중치** 역할 수행

- 필터의 모양과 크기가 국부수용장의 모양과 크기를 지정
- 결국 딥러닝은 학습하면서 필터의 값을 찾는 것!

특성맵(feature map) : 필터 각각을 사용하여 생성된 출력값(=결과값)

- 각 특성지도의 픽셀이 하나의 뉴런에 해당
- 필터에 포함된 모든 뉴런은 동일한 가중치와 편향 사용
- 이 맵은 필터를 가장 크게 활성화시키는 이미지의 영역을 강조한다.
- 훈련하는 동안 합성곱 층이 자동으로 해당 문제에 가장 유용한 필터를 찾고 상위층은 이들을 연결하여 더 복잡한 패턴을 학습

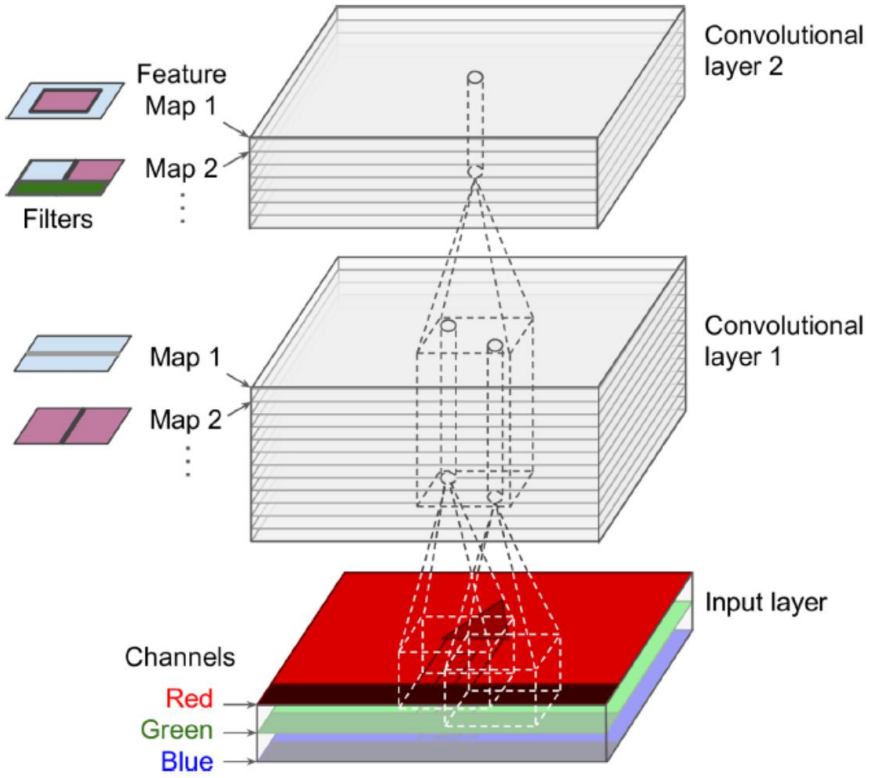


14.2.2 여러 가지 특성 맵 쌓기

각 특성 맵의 픽셀은 하나의 뉴런에 해당하고 하나의 특성 맵 안에서는 모든 뉴런이 같은 파라미터를 공유

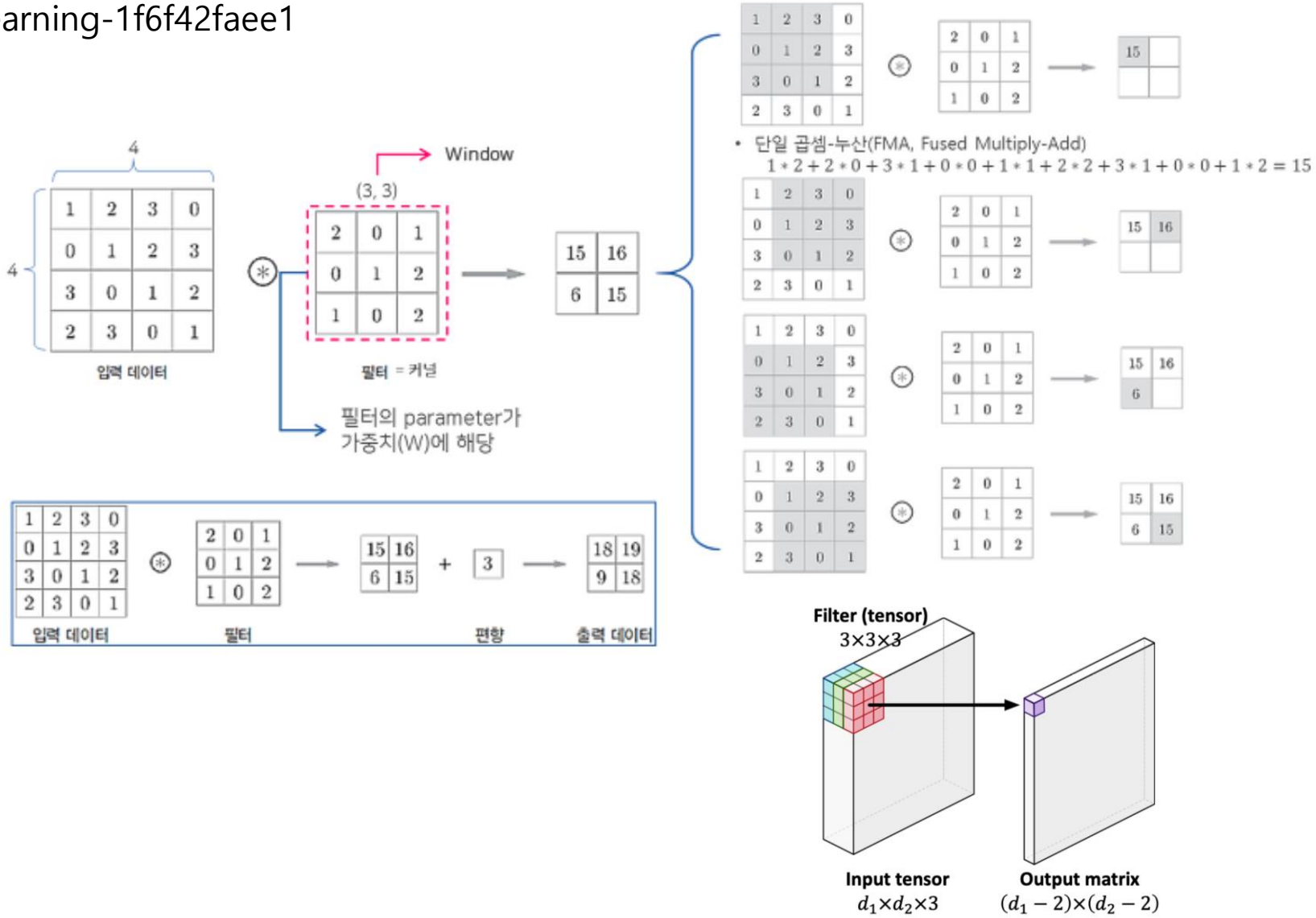
- 다른 특성 맵에 있는 뉴런은 다른 파라미터를 사용
- 한 뉴런의 수용장은 이전 층에 있는 모든 특성 맵에 걸쳐 확장됨

하나의 합성곱 층이 입력에 따라 필터를 동시에 적용하여 입력에 있는 여러 특성을 감지



14.2.2 여러 가지 특성 맵 쌓기

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1



14.2.3 텐서플로 구현

```
from sklearn.datasets import load_sample_image
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

# 샘플 이미지 로드
china = load_sample_image('china.jpg') / 255
flower = load_sample_image("flower.jpg") / 255
images = np.array([china, flower])
batch_size, height, width, channels = images.shape

filters = np.zeros(shape=(7, 7, channels, 2), dtype=np.float32)
filters[:, 3, :, 0] = 1 # 수직선
filters[3, :, :, 1] = 1 # 수평선

outputs = tf.nn.conv2d(images, filters, strides=1, padding='SAME') # 저수준 API에서 padding의 매개변수를 항상 대문자로!

plt.imshow(outputs[0, :, :, 1], cmap='gray') # 첫번째 이미지의 두 번째 특성 맵 선택
plt.show()
```

keras.layers.Conv2D 사용

```
# 3x3개의 필터 32개, padding='same'은 제로 패딩 사용
conv = keras.layers.Conv2D(filters=32, kernel_size=3, strides=1,
                           padding='same', activation='relu')
```

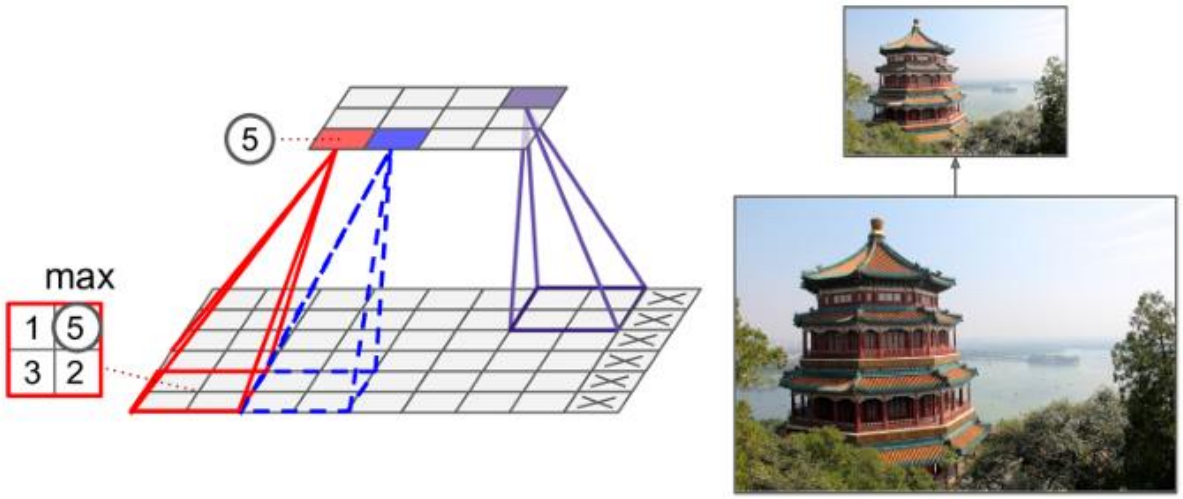
14.3 풀링 층

풀링 층(pooling layer)의 목적 : 계산량과 메모리 사용량, 파라미터 수를 줄이기 위해 입력 이미지의 부표본(subsample)을 만들기 위함

- 풀링 층의 각 뉴런은 이전 층의 작은 사각 영역의 수용장 안에 있는 뉴런의 출력과 연결되어 있음

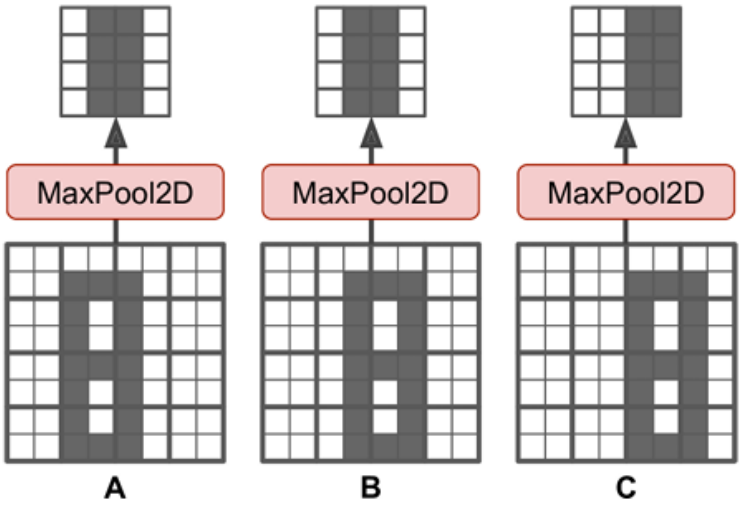
최대 풀링 층(max pooling layer)을 통해 각 수용장에서 가장 큰 입력값이 다음 층으로 전달되고 다른 값은 버려짐

- 특징맵(feature map)의 크기를 절반으로 줄여줌
- 큰 특징들만 추출. 즉, 점점 샘플링해서 줄여주는 역할
- 계산량, 메모리 사용량을 줄어줌
- 많은 정보를 잃게 되지만 그대로 잘 작동됨



14.3 풀링 층

최대 풀링은 작은 변화에도 일정 수준의 **불변성(invariance)**을 만들어 줌



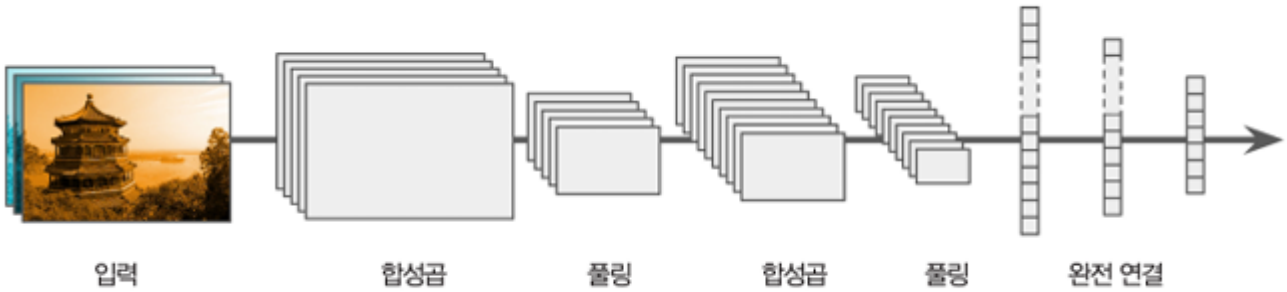
평균 풀링 층(average pooling layer) : AvgPool2D 사용

- But, 최대 풀링층에 비해 성능이 떨어짐

전역 평균 풀링 층(global average pooling layer) : 각 특성 맵의 평균을 계산

14.4 CNN 구조

전형적인 CNN 구조 : 합성곱 층을 몇 개 쌓고(각각 ReLU 층을 그 뒤에 놓고), 그다음에 풀링층을 쌓고, 그다음에 또 합성곱 층(+ReLU)을 몇 개 더 쌓고, 그다음에 다시 풀링 층을 쌓는 식.




```
model = keras.models.Sequential([
    DefaultConv2D(filters=64, kernel_size=7, input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(pool_size=2),
    DefaultConv2D(filters=128),
    DefaultConv2D(filters=128),
    keras.layers.MaxPooling2D(pool_size=2),
    DefaultConv2D(filters=256),
    DefaultConv2D(filters=256),
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=10, activation='softmax'),
])
```

14.4.1 LeNet-5

1998년에 만들어져 MNIST에 널리 사용된 CNN 구조

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

LeNet-5 구조

- **C1 layer** : 32x32 이미지를 6개의 필터(5x5)와 합성곱 연산 수행 -> 6장의 28x28 특성 맵을 얻음
  - 특성맵이 6개인 이유는 각 필터마다 특성맵이 나오기 때문
  - 제로패딩이므로, 32x32에서 28x28로 줄어듬 (이미지가 네트워크를 진행하면서 점점 크기가 줄어듦)

- **S2 layer** : 평균 풀링 층에서 각 뉴런은 입력의 평균을 계산한 다음, 그 값에 학습되는 계숫값을 곱함 그리고 학습되는 값인 편향을 더한 후, 마지막으로 활성화 함수를 적용
  - 2x2 필터를 stride 2로 설정하여 특성맵이 절반으로(14x14)으로 축소됨

- **C3 layer** : 6장의 14x14 특성맵으로부터 16장의 10x10 특성맵이 산출됨
  - C3의 대부분의 뉴런은 S2의 6개 맵 전체가 아니라 3~4개 맵에 있는 뉴런에만 연결 됨

14.4.1 LeNet-5

1998년에 만들어져 MNIST에 널리 사용된 CNN 구조

- 6장의 14x14 특성맵에서 연속된 3장씩을 모아 5x5x3 사이즈의 필터와 합성곱 -> 6장의 10x10 특성맵 산출
- 6장의 14x14 특성맵에서 연속된 4장씩을 모아 5x5x4 사이즈의 필터와 합성곱 -> 6장의 10x10 특성맵 산출
- 6장의 14x14 특성맵에서 불연속 4장씩을 모아 5x5x4 사이즈의 필터와 합성곱 -> 3장의 10x10 특성맵 산출
- 6장의 14x14 특성맵 모두를 가지고 5x5x6 사이즈의 필터와 합성곱 -> 1장의 10x10 특성맵 산출
- => 총 16장의 10x10 특성맵 산출

- **S4 layer** : 평균 풀링 층, 특성 맵이 5x5 로 축소

- **C5 layer** : 16개의 5x5 를 받아 5x5 커널 크기의 합성곱을 수행하기 때문에 출력은 1x1 크기의 특성맵
  - 이것들을 fully connected 형태로 연결하여 총 120개의 특성맵을 생성
  - 이전 단계의 16장의 특성맵이 합성곱을 거치면서 다시 전체적으로 섞이게 됨

- **F6 layer** : Fully connected, C5의 결과를 84개의 unit에 연결

- **출력 layer** : 입력과 가중치 벡터를 행렬 곱셈하는 대신 각 뉴런에서 입력 벡터와 가중치 벡터 사이의 유클리드 거리를 출력
  - 각 출력은 이미지가 얼마나 특정 숫자 클래스에 속하는지 측정
  - 요즘은 잘못된 예측을 줄여주고 그래디언트 값이 크고 빠르게 수렴되기 때문에 크로스 엔트로피 비용함수를 선호

14.4.2 AlexNet

- 2012년 17%의 에러율을 달성하며 이미지넷 대회에서 우승
- LeNet-5 와 비슷하며 더 크고 깊은 구조
- 처음으로 합성곱 층 위에 풀링 층을 쌓지 않고 합성곱 층끼리 쌓음

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13 × 13	3 × 3	1	SAME	ReLU
C6	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
C5	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
S4	Max Pooling	256	13 × 13	3 × 3	2	VALID	–
C3	Convolution	256	27 × 27	5 × 5	1	SAME	ReLU
S2	Max Pooling	96	27 × 27	3 × 3	2	VALID	–
C1	Convolution	96	55 × 55	11 × 11	4	SAME	ReLU
In	Input	3 (RGB)	224 × 224	–	–	–	–

AlexNet 구조

- 과대적합을 줄이기위해 두 가지 규제 기법 사용
  - 훈련하는 동안 F9, F10 출력에 **드롭아웃**을 50% 비율로 적용
  - 훈련이미지를 랜덤하게 이동하거나 수평으로 뒤집고 조명을 바꾸는 식으로 **데이터 증식**을 수행

14.4.2 AlexNet

- **LRN** 이라 부르는 경쟁적인 정규화 단계 사용 (C1, C3층의 ReLU 단계 후에 사용)
  - 가장 강하게 활성화된 뉴런이 다른 특성 맵에 있는 같은 위치의 뉴런을 억제함
  - 특성 맵을 각기 특별하게 다른 것과 구분되게 하고, 더 넓은 시각에서 특징을 탐색하도록해 일반화 성능을 향상시킴

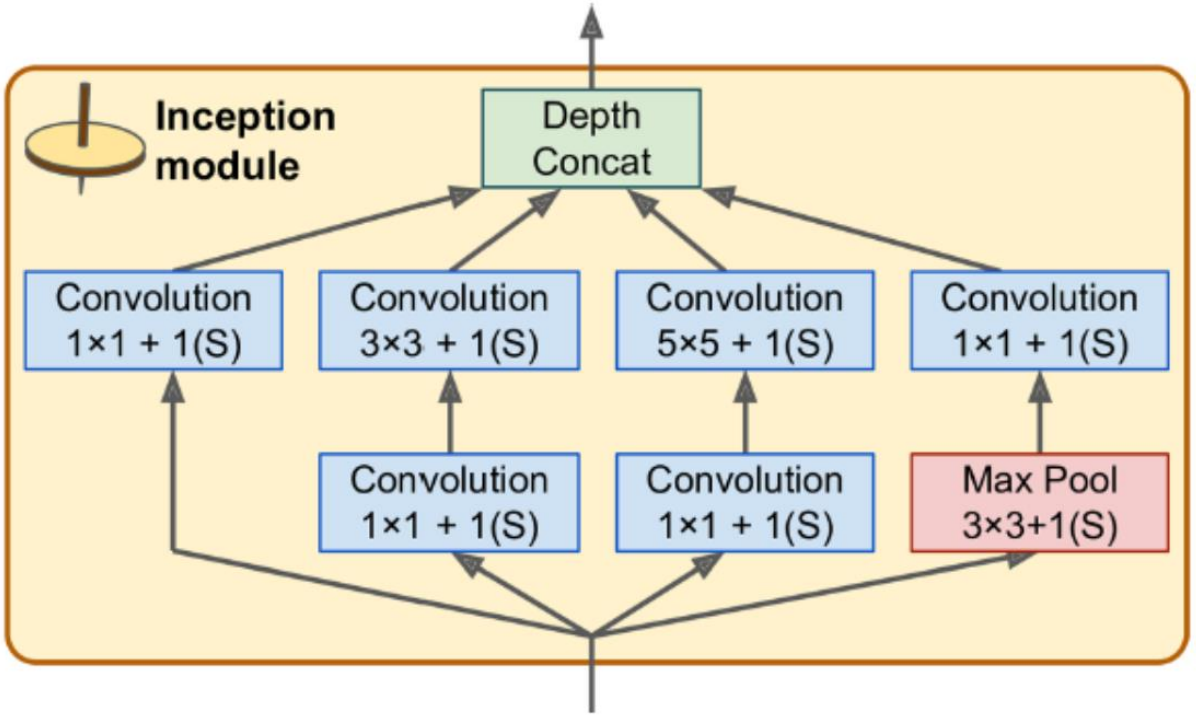
$$b_i = a_i\left(k + \alpha \sum_{j=j_{\text{low}}}^{j_{\text{high}}} a_j^2\right)^{-\beta} \quad \text{with} \quad \begin{cases} j_{\text{high}} = \min\left(i + \frac{r}{2}, f_n - 1\right) \\ j_{\text{low}} = \max\left(0, i - \frac{r}{2}\right) \end{cases}$$

LRN

※ LRN은 현재는 많이 사용되지 않음

14.4.3 GoogLeNET

- 구글 리서치 개발, top-5 에러율 7% 이하
- 네트워크가 이전 CNN보다 훨씬 깊음
- **인셉션 모듈** 이라는 서브 네트워크를 통해 이전의 구조보다 효과적으로 파라미터를 사용 (AlexNet보다 10배 적은 파라미터를 가짐)

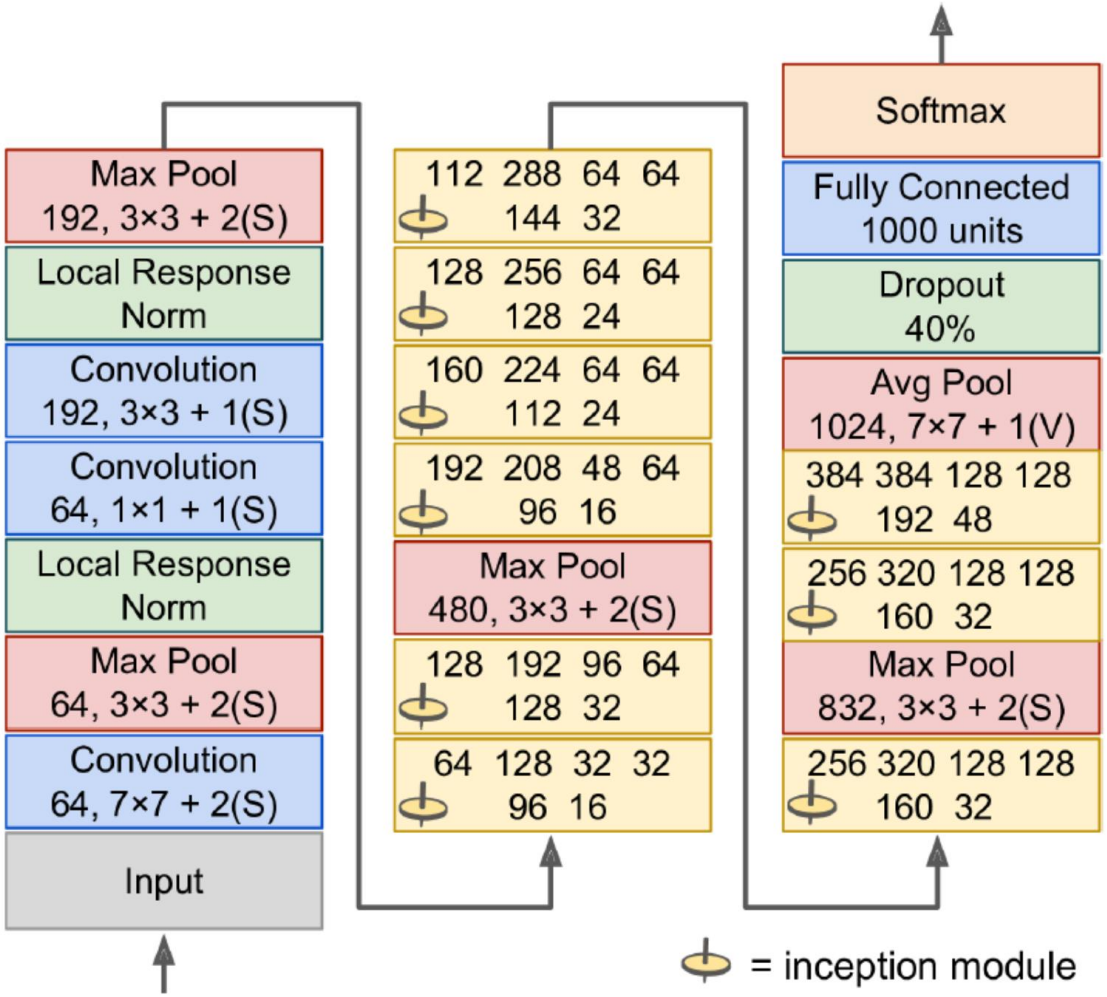


인셉션 모듈

14.4.3 GoogLeNET

GoogLeNet 구조

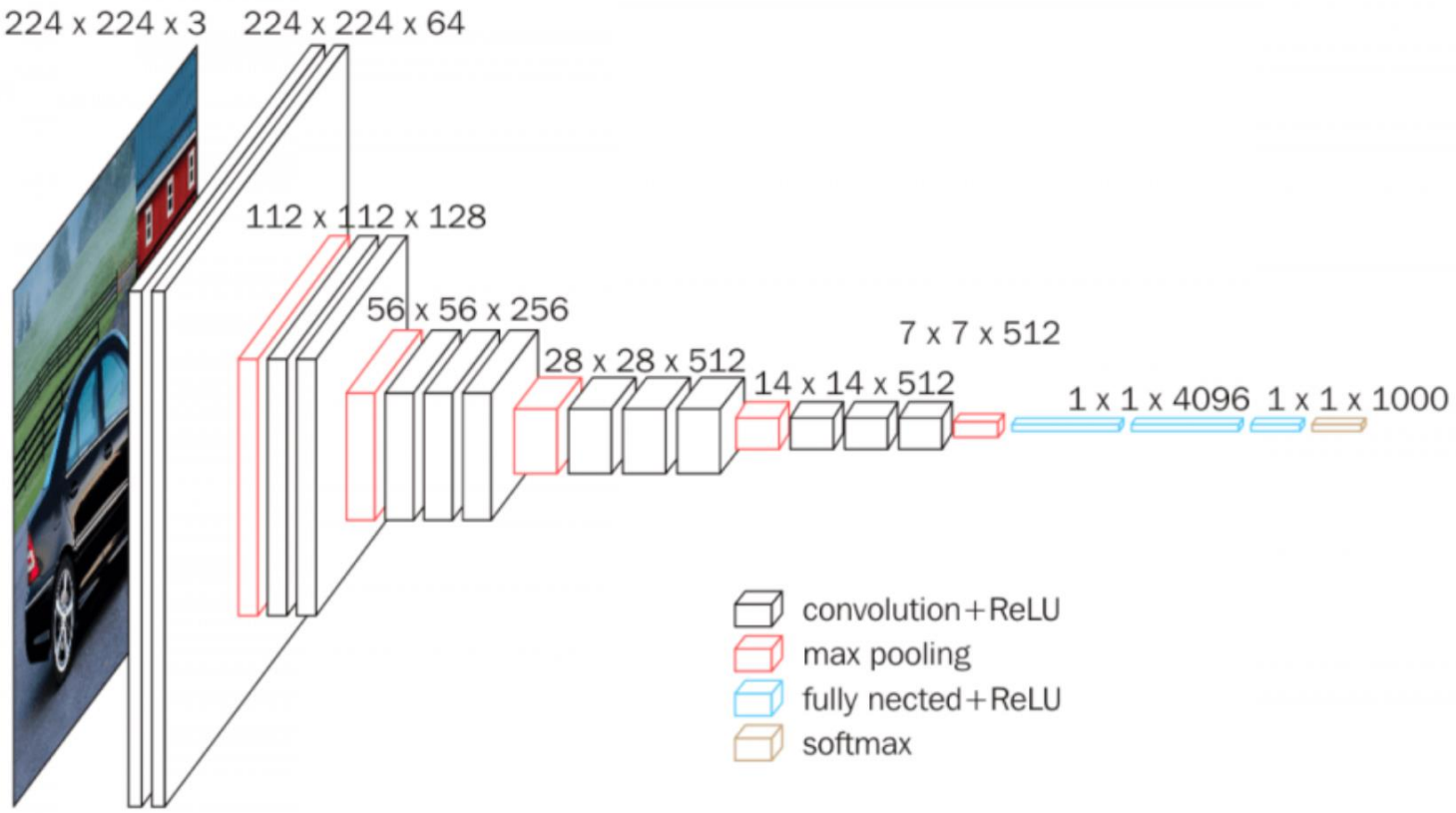
- 합성곱 층과 풀링 층에서 출력되는 특성 맵 수는 커널 크기 앞에 표시
- 네트워크를 하나로 길게 쌓은 구조
- 9개의 인셉션 모듈 포함
- 인셉션 모듈에 있는 여섯 개 숫자는 모듈 안에 있는 합성곱 층에서 출력하는 특성 맵의 수



GoogLeNet 구조

14.4.4 VGGNet

- 네트워크의 깊이가 주는 영향에 대한 연구로부터 개발됨
- 2~3개의 합성곱 층 뒤에 풀링 층이 나오고, 다시 2~3개의 합성곱 층과 풀링 층이 등장하는 방식
- 마지막 dense 네트워크는 2개의 은닉층과 출력층으로 이루어짐
- VGGNet은 많은 개수의 필터를 사용하지만 3x3 필터만 사용함 => 깊은 네트워크 구조를 위해 작은 필터 사용 (3x3은 left, right, up, down을 고려할 수 있는 최소한의 receptive field임)
- 결과적으로 깊은 네트워크를 통해 높은 정확도를 달성할 뿐만 아니라, 단순한 파이프라인으로 우수한 성능을 보여준 모델



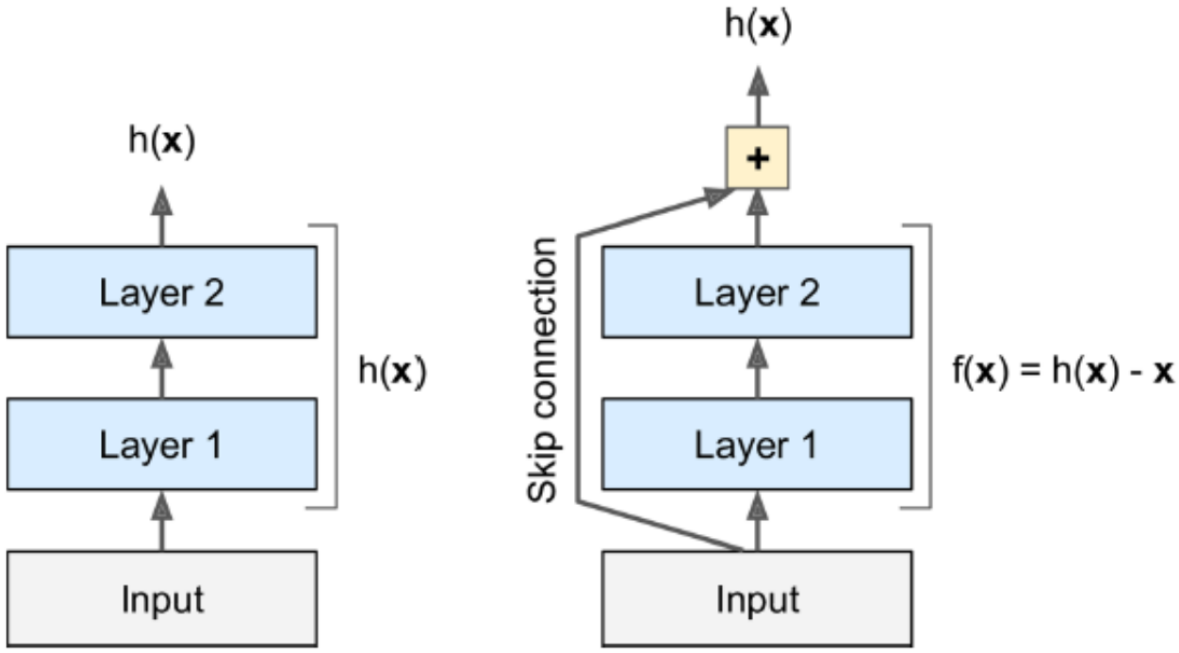
VGG16 구조

14.4.5 ResNet

- ILSVRC 2015 대회에서 우승
- 3.6% 이하의 에러율
- 우승한 네트워크는 152개 층으로 구성된 매우 깊은 CNN 사용
- 더 적은 파라미터를 사용해 더 깊은 네트워크로 모델을 구성하는 트렌드를 만듦

=> 깊은 네트워크를 훈련시킬 수 있는 핵심 요소는 **스킵 연결**(숏컷 연결) - 어떤 층에 주입되는 신호를 상위 층의 출력에도 더해줌

잔차학습



잔차학습, Residual Learning

14.4.6 Xception

- GoogLeNet 과 ResNet의 아이디어를 합쳤지만
- 인셉션 모듈을 **깊이별 분리 합성곱 층** 이라는 층으로 대체 (Depthwise separable convolution)
- 일반적인 합성곱 층이 공간상의 패턴(ex. 타원 형태)과 채널 사이의 패턴(ex. 입+코+눈=얼굴)을 동시에 잡기 위해 필터를 사용
- 분리 합성곱층은 공간 패턴과 채널 사이 패턴을 분리하여 모델링할 수 있다고 가정

깊이별 분리 합성곱 층(Depthwise separable convolution)