합성망 신경망

Convolutional Neural Networks

본 자료는 한빛미디어에서 나온 〈비전 시스템을 위한 딥러닝〉 책을
참고하여 제작 되었음을 알립니다.

3. 합성곱 신경망

- **학습 목표**
  - 다층 퍼셉트론을 이용해서 이미지 분류하기
  - 합성곱 신경망 구조를 이용해서 이미지 분류하기
  - 컬러 이미지에 대한 합성곱 이해하기

- **3. 합성곱 신경망** convolutional neural network, CNN
  - 합성곱 신경망 은 이미지를 다루는데 특화해서 진화된 MLP 구조임
  - 신경망이 학습하고 파리미터를 최적화하는 기본원리는 CNN과 MLP 모두 같음

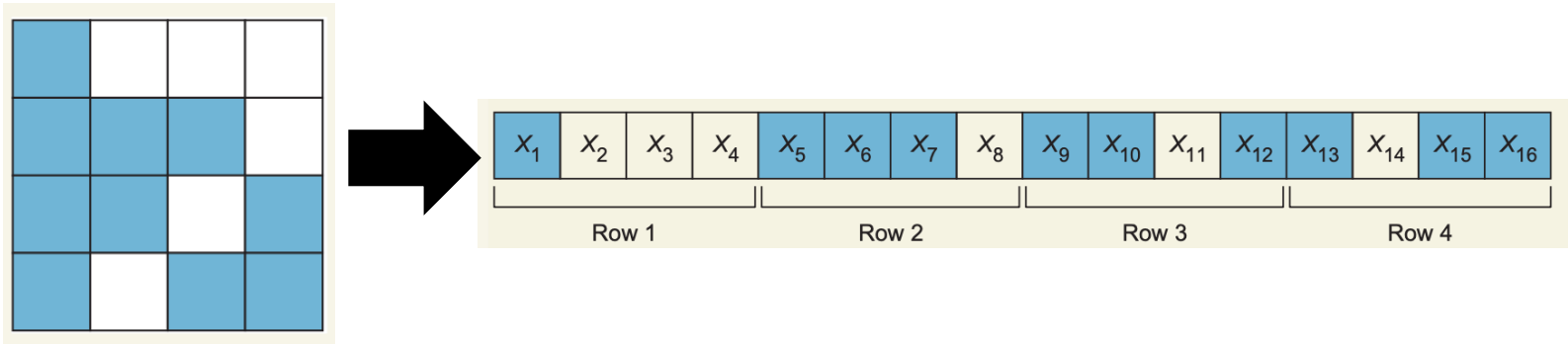
3. 합성곱 신경망

- **3.1 다층 퍼셉트론을 이용한 이미지 분류**
  - MLP를 이용하여 MNIST 데이터셋을 대상으로 이미지 분류 문제 풀기
  - 이미지 분류 문제의 목표는 〈손글씨 이미지를 0부터 9까지의 숫자〉로 분류하는 것
- **3.1.1 입력층**
  - 신경망으로 2차원 이미지를 다루려면 이미지를 제공하기 전에 신경망이 이해할 수 있는 형태로 변환 필요

28X28 크기의 이미지는 28x28 크기의 행렬이 되며, 행렬의 요소값은 0부터 255의 범위를 가짐 (0은 검은색, 255는 흰색, 그 사이 값은 회색조)

3. 합성곱 신경망

- **3.1.1 입력층**
  - MLP는 모양이 (1,n) 인 1차원 벡터만 입력 받으므로 모양이 (x,y)인 2차원 이미지 행렬은 입력할 수 없음
  - 2차원 행렬을 MLP 입력층에 입력하려면 행렬을 모양이 (1, n) 벡터로 변환해야 하며, 이 과정을 이미지 벡터 변환 image flattening 이라고 함
    - 즉, (28x28) 크기의 영상을 MLP로 입력하려면 (1x784) 인 벡터로 변환이 필요
  - 입력 벡터 시각화하기



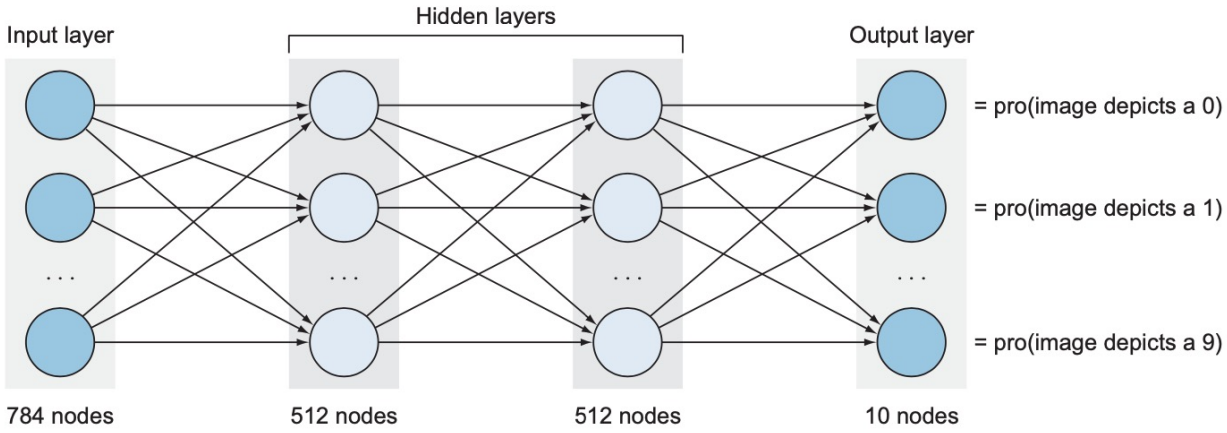
3. 합성곱 신경망

- **3.1.2 은닉층**
  - 신경망은 하나 이상의 은닉층을 가질 수 있음
  - 각 층은 하나 이상의 뉴런으로 구성됨
  - 이 문제에서는 노드 512개를 가진 은닉층을 두 층 만들고, 각 층마다 ReLU활성화 함수를 추가

- **3.1.3 출력층**
  - 분류 문제의 출력층 노드 수는 분류 대상 클래스 수와 같음
  - 이 문제는 0부터 9까지의 숫자 10개가 분류 대상이므로 노드 10개를 갖는 Dense층을 추가하면 됨

3. 합성곱 신경망

- 3.1.4 모델 완성하기
  - 입력층, 은닉층, 출력층을 모두 합쳐 신경망을 완성해보면 아래와 같음




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

해당 층의 파라미터(가중치) 수

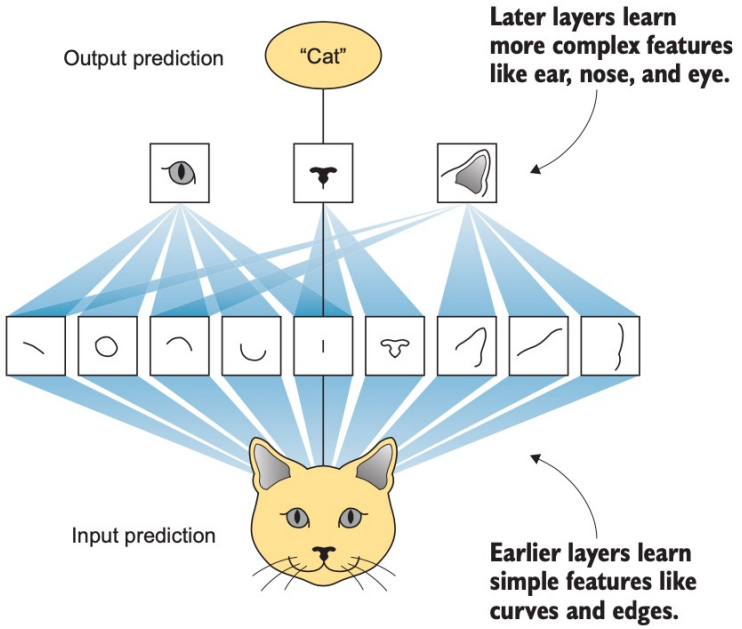
784x512+512

512x512+512

512x10+10

3. 합성곱 신경망

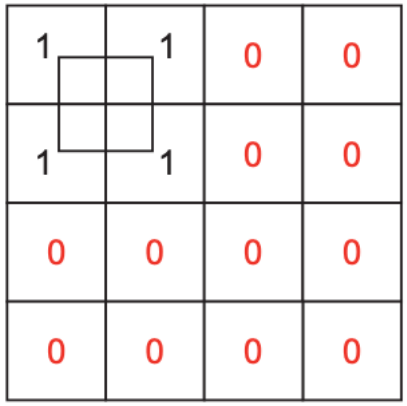
- **3.1.5 MLP로 이미지를 다룰 때의 단점**
  - MLP로 이미지를 다룰 때 나타나는 단점
    - 공간적 특징의 손실
    - 복잡한 신경망 학습
  - CNN은 MLP의 단점을 개선하기 위해 고안된 구조



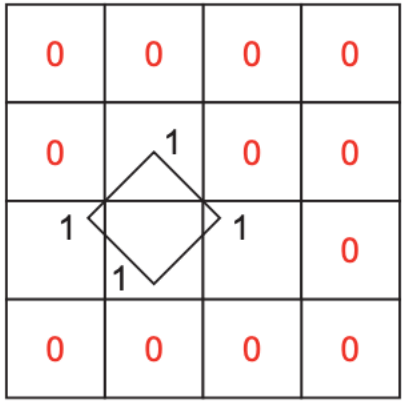
3. 합성곱 신경망

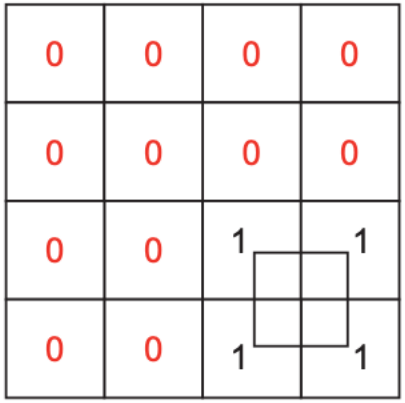
- **3.1.5.1 공간적 특징의 손실**
  - 2차원 이미지를 1차원 벡터로 변형하면 이미지 내 공간적 특징이 손실됨
    - 즉, 서로 가까이 위치한 픽셀 간의 관계를 알 수 없기 때문에 정보의 손실이 발생

MLP 모델의 공간적 특징 손실의 예



MLP 학습에 사용한 이미지 행렬

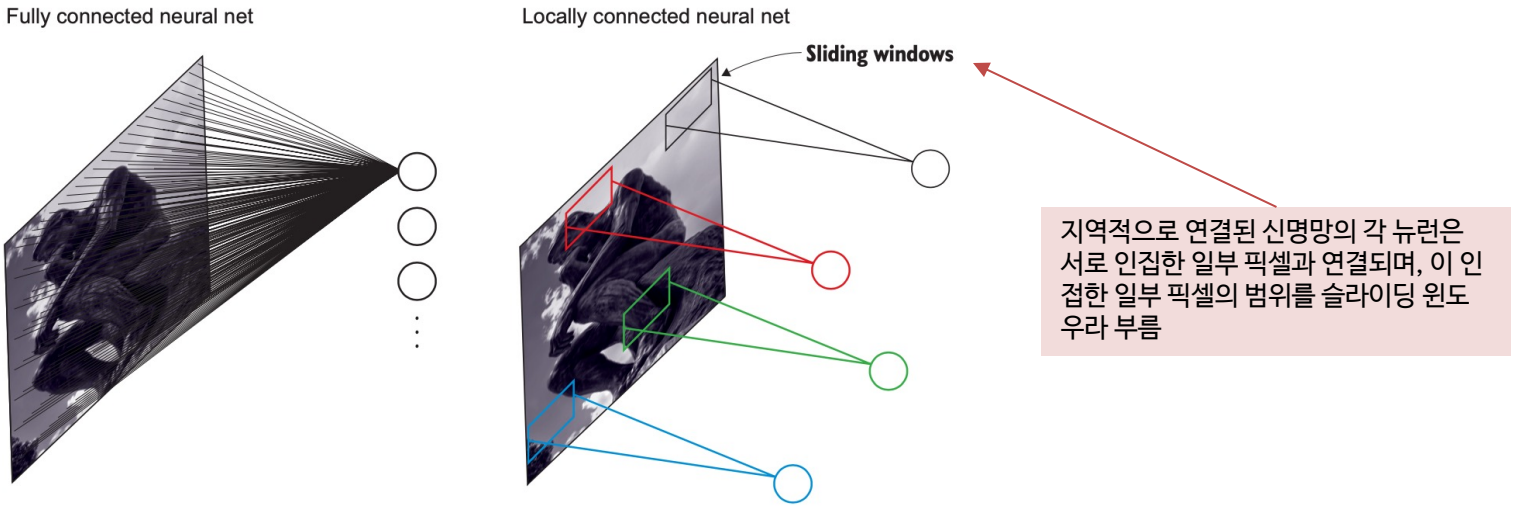




다양한 자리에 위치한 정사각형의 예
➔ 정사각형의 포함 유무를 알지 못하는 MLP

3. 합성곱 신경망

- 3.1.5.2 전결합층
  - MLP는 전결합 fully-connected 층으로 구성
    - 전결합이란 이전 층의 모든 노드가 다음 층의 모든 노드와 연결된다는 뜻
  - CNN은 지역적 locally connected 으로 연결된 구조의 층을 가짐
    - 즉, CNN의 노드는 이전 층의 노드 중 일부하고만 연결됨
    - 지역적 연결을 가진 층은 전결합층에 비해 파라미터 수가 훨씬 적음



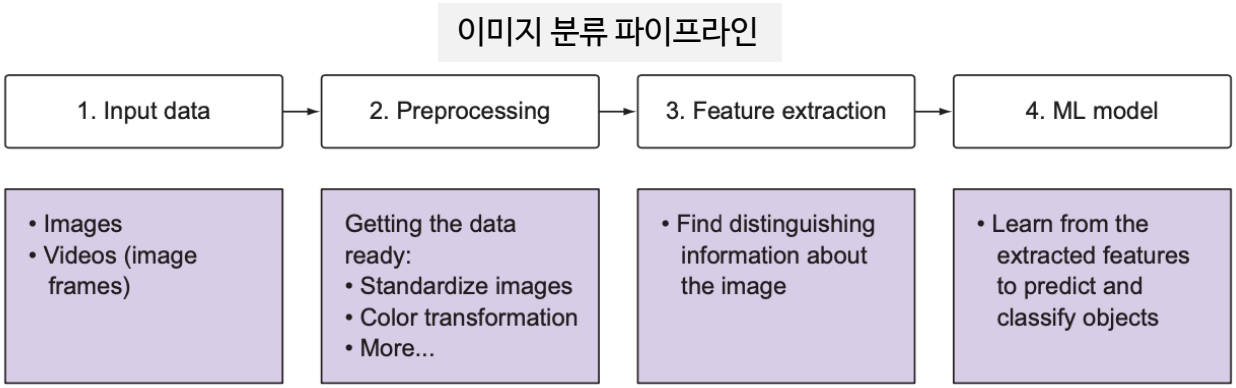
3. 합성곱 신경망

- **3.1.5.3 합성곱 신경망 구조의 의의**
  - 2차원 행렬인 이미지를 1차원 벡터로 변환하면서 손실되는 정보와 전결합층의 계산 복잡도를 고려하며 이미지 입력을 다루려면 새로운 형태의 신경망이 필요
  - CNN은 이런 필요로부터 등장했으며, 2차원 이미지 행렬을 그대로 입력으로 받을 수 있으므로 픽셀값에 숨어 있는 패턴을 이해할 수 있음

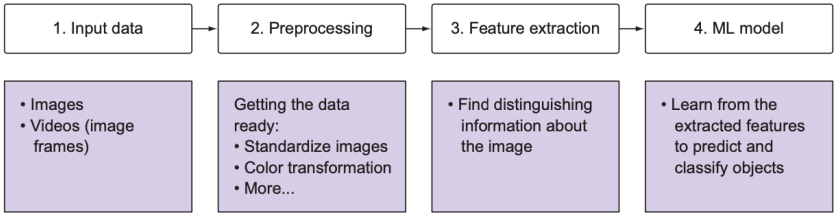
3. 합성곱 신경망

- **3.2 합성곱 신경망 구조**
  - 합성곱 신경망(CNN) 역시 일반적인 신경망(MLP)의 학습과 크게 다르지 않음

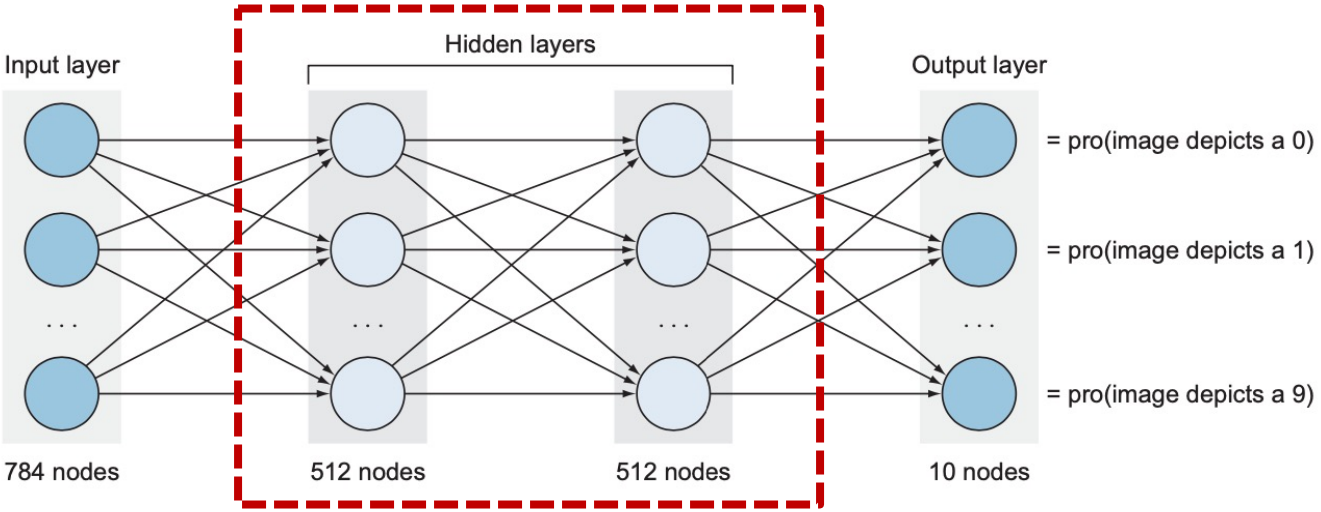
- **3.2.1 전체 학습 과정**
  - 딥러닝 이전에는 특징 추출을 사람이 직접 수행 Handcraft Feature, Human Inspired Feature 하고 추출된 특징 벡터를 분류기(SVM 등의 일반적인 머신러닝 알고리즘) 에 입력했음
  - 특징 학습과 분류를 동시에(3단계, 4단계)할 수 있는 신경망(MLP와 CNN)의 마법 같은 힘 덕분에 세번째 단계를 사람이 직접 할 필요가 없어짐



3. 합성곱 신경망



- 3.2.1 전체 학습 과정
  - MLP는 〈특징을 학습 3단계〉하고 〈이미지를 분류 4단계〉하는 과정을 함께 수행함
  - 전결합층 FC Layer 의 문제 공간적 특징의 손실, 복잡한 신경망 학습 는 〈분류 단계〉 보다는 〈특징 학습 단계〉에 있음
    - 전결합층의 특징 추출을 지역적 연결을 가진 합성곱층으로 변경
    - 전결합층은 추출된 특징을 잘 분류하므로 그대로 사용

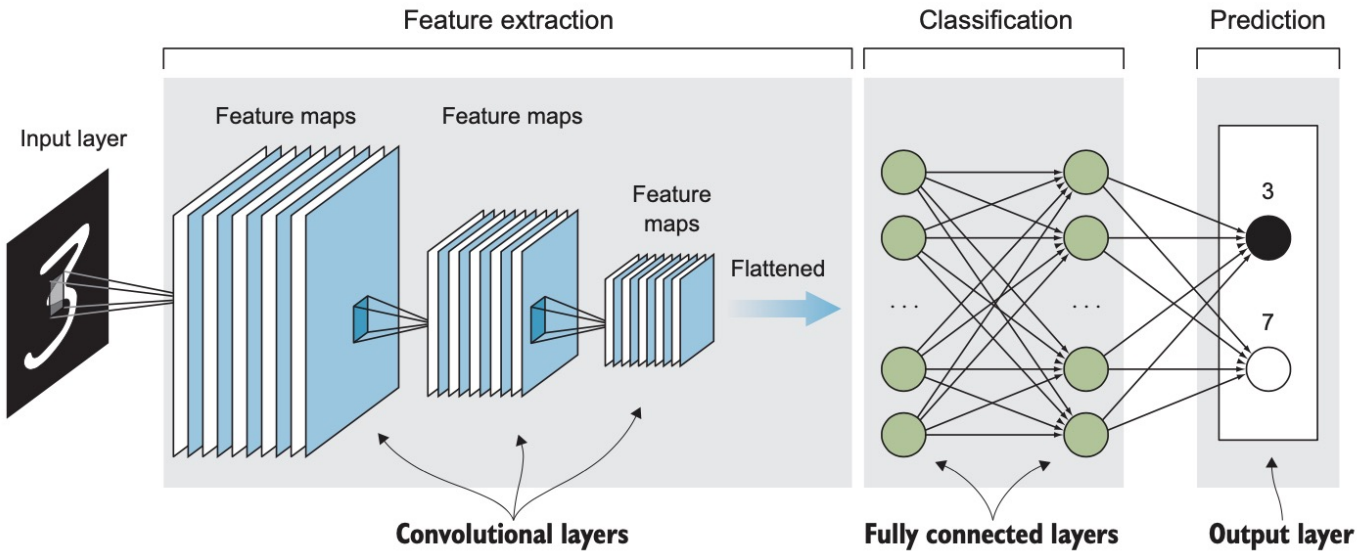


3. 합성곱 신경망

- 3.2.1 전체 학습 과정
  - CNN의 추상적인 구조
    - 입력층
    - 특징 추출을 담당하는 합성곱층
    - 분류를 담당하는 전결합층
    - 예측 결과 출력

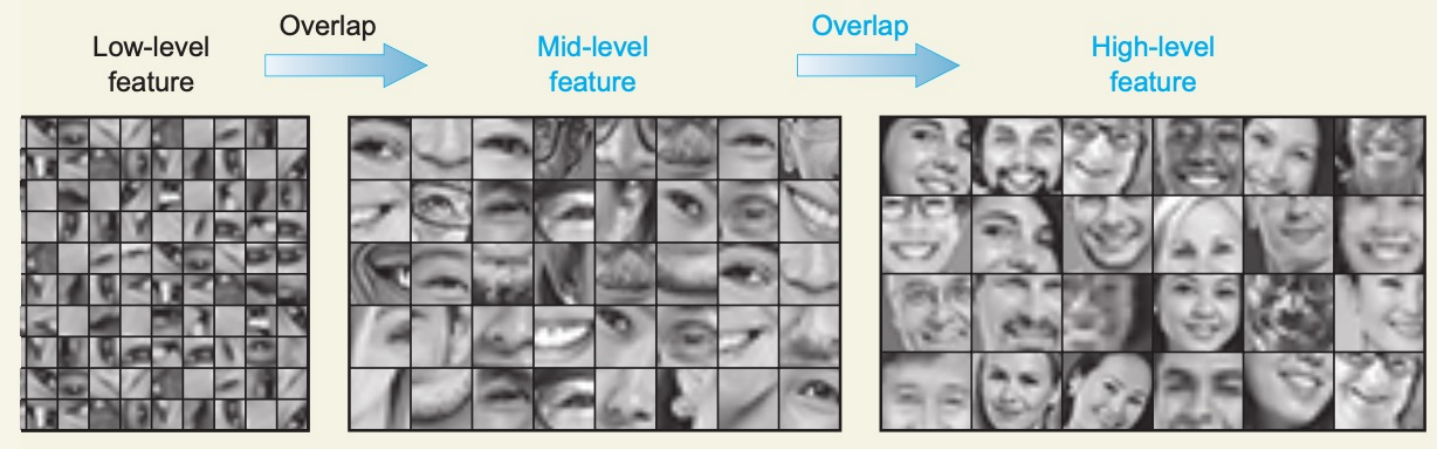
CNN의 특징맵이란?
- 이전 층에서 적용된 필터의 출력

특징맵이라는 이름이 붙은 이유는?
- 해당 특징이 이미지의 어느 부분에서 발견되었는지 나타내기 때문



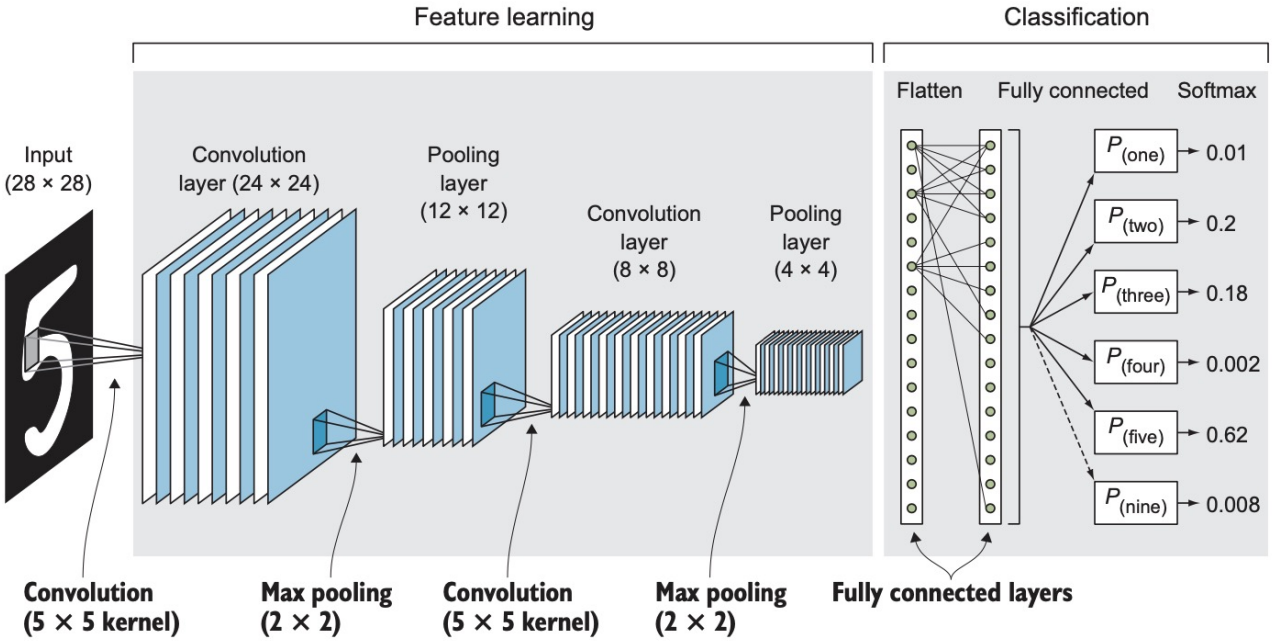
3. 합성곱 신경망

- **3.2.4 합성곱 신경망은 어떻게 패턴을 학습하는가**
  - CNN 에서는 한 층만에 바로 특징이 추출되는 것이 아님
  - 이미지에서 특징을 제대로 추출하려면 수십 또는 수백 층을 거쳐야 함
  - 특징이 학습되는 과정은 은닉층 하나하나를 거치며 점진적으로 진행됨
  - 첫번째 은닉층은 대개 직선이나 모서리 같은 아주 간단한 특징을 학습함
  - 두번째 은닉층은 이들 특징을 조합하여 도형, 꼭짓점, 원 등의 특징을 인식함
  - 은닉층은 깊어질 수록 사람의 이목구비와 같은 복잡한 형태를 학습할 수 있음



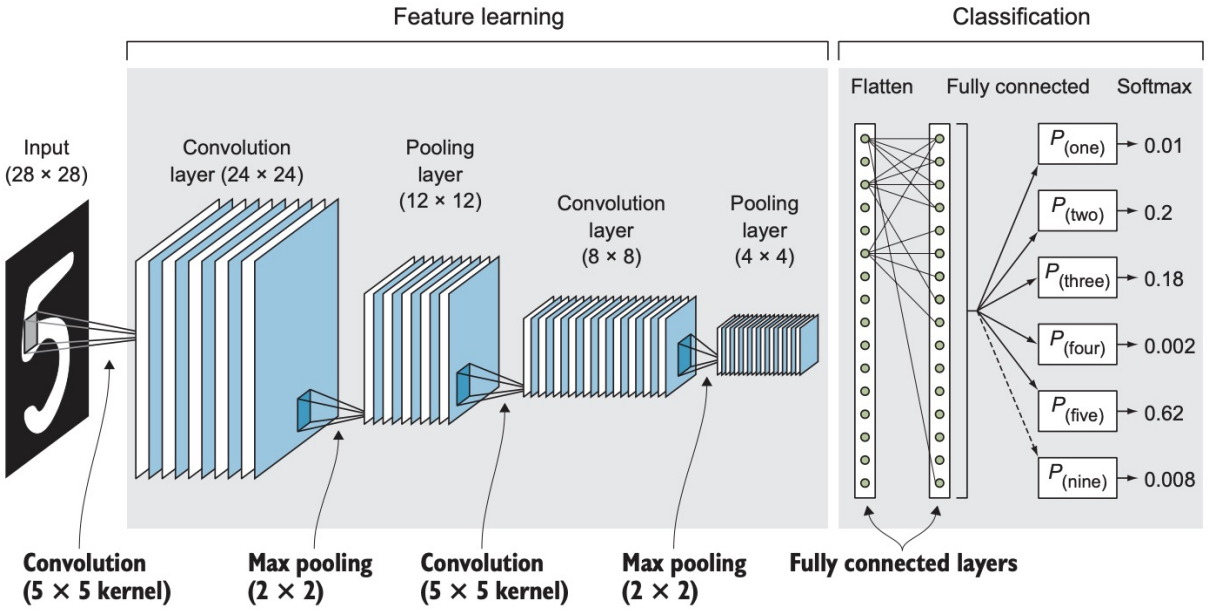
3. 합성곱 신경망

- **3.3 합성곱 신경망의 기본 요소**
  - 합성곱 신경망의 주요 구성 요소
    - 합성곱층 convolutional layer, CONV ➔ 특징 추출
    - 풀링층 pooling layer, POOL
    - 전결합층 fully connected layer, FC ➔ 분류



3. 합성곱 신경망

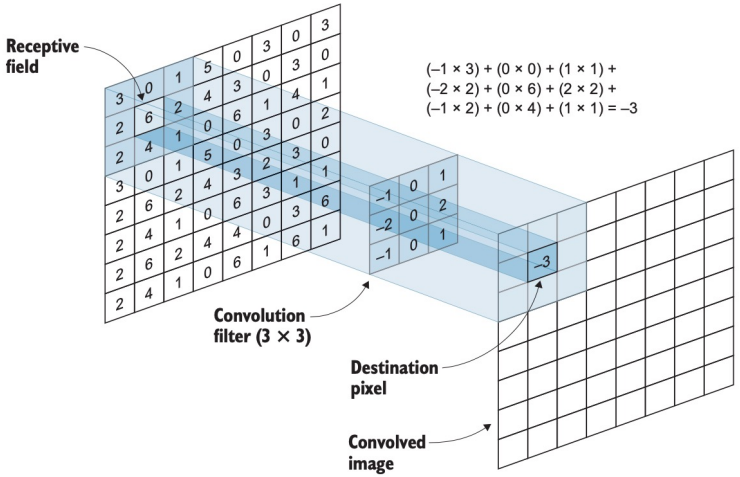
- **3.3 합성곱 신경망의 기본 요소**
  - 텍스트로 표현한 합성곱 신경망의 구조 (아래의 그림)
    - 입력➔CONV➔RELU➔POOL➔ CONV➔RELU➔POOL➔FC➔SOFTMAX
    - 합성곱 2개와 전결합층 1개를 가진 CNN 구조
    - 합성곱과 전결합층은 원하는 만큼 둘 수 있음
    - ReLU는 CONV와 소프트맥스는 FC와 함께 사용되는 층이지 독립적인 층이 아님



3. 합성곱 신경망

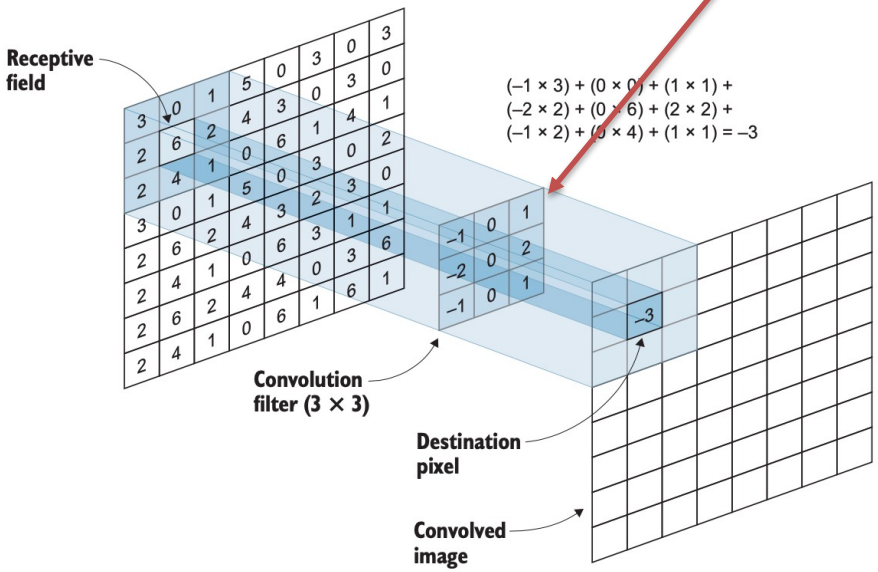
- 3.3.1 합성곱층
  - 3.3.1.1 합성곱이란
    - 수학에서 말하는 **합성곱**은 두 함수를 인수로 새로운 함수를 만들어 내는 연산
    - 합성곱 신경망에서 말하는 합성곱은 첫 번째 인수는 입력 이미지 이고, 두 번째 인수는 합성곱 필터에 해당하며, 이를 대상으로 수학적 연산을 통해 새로운 이미지를 만들어 냄
    - 즉, 합성곱 필터를 입력 이미지 위로 이동시키며 합성곱 필터가 위치한 부분에 해당하는 작은 이미지 조각을 처리한 결과를 모아서 새로운 이미지인 특징 맵을 만듬

3x3 크기의 합성곱 필터가 이미지 위를 이동하는 예



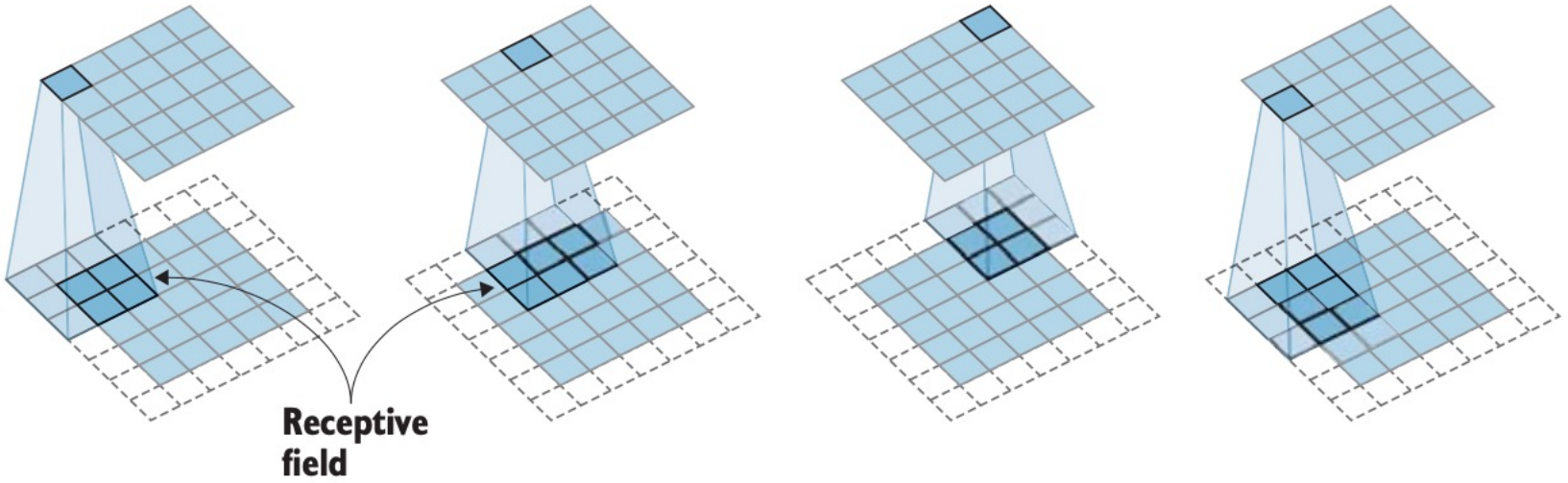
3. 합성곱 신경망

- 3.3.1 합성곱층
  - 3.3.1.1 합성곱이란
    - 가운데에 위치한 3x3 크기의 행렬이 합성곱 필터이며, 이를 **커널** kernel 이라고 부름
    - 합성곱 신경망에서는 합성곱 행렬이 바로 가중치
    - 합성곱 행렬은 무작위 값으로 초기화 되며 신경망에 의해 학습되는 값임



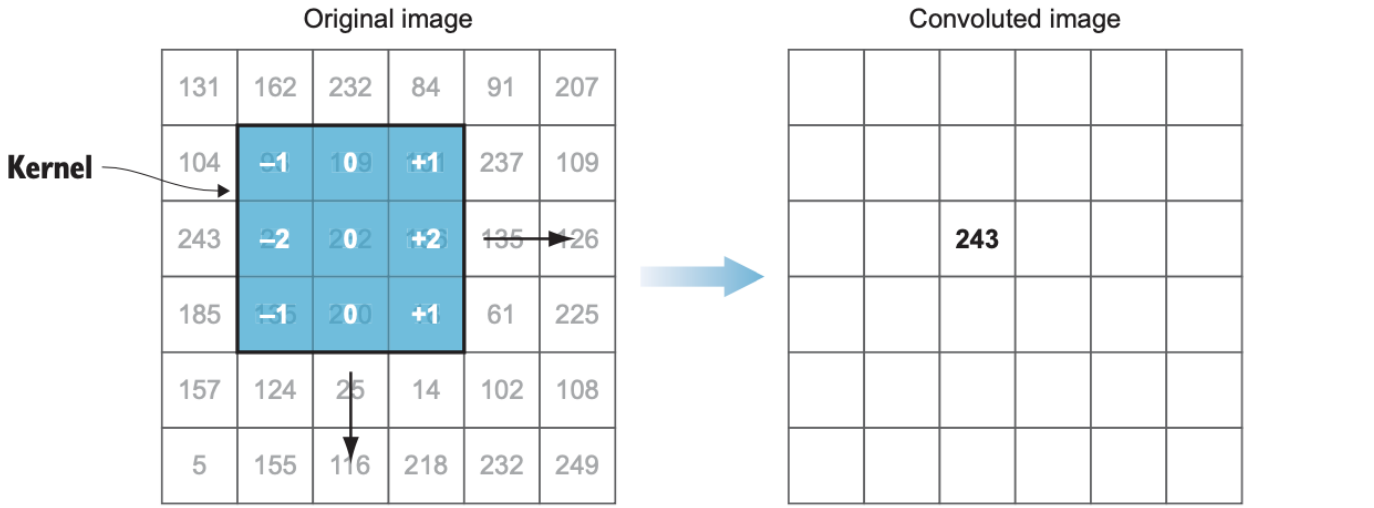
3. 합성곱 신경망

- **3.3.1 합성곱층**
  - 3.3.1.1 합성곱이란
    - 커널은 입력 이미지 위를 픽셀 단위로 움직이며 연산을 수행하며 픽셀값을 계산함
    - 각 위치에서 연산된 픽셀값을 모아 합성곱 연산을 거친 새로운 이미지를 만들어 다음 층으로 전달
    - 필터가 위치한 입력 이미지상의 범위를 **수용 영역** receptive field 이라고 함



3. 합성곱 신경망

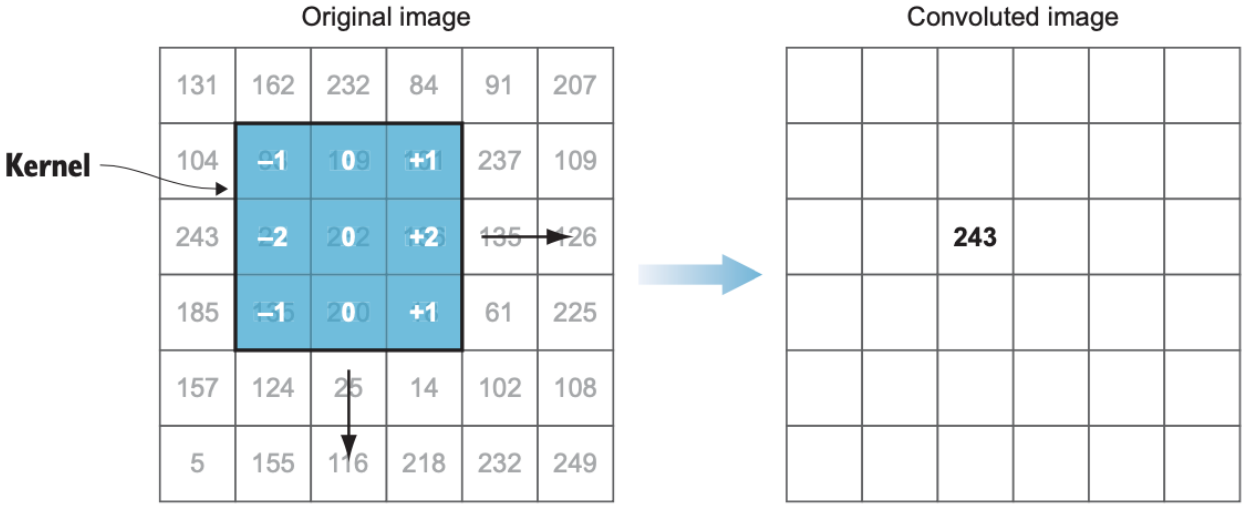
- 3.3.1 합성곱층
  - 3.3.1.2 합성곱 연산
    - 합성곱 연산은 다층 퍼셉트론의 가중치합과 동일
      - 입력에 가중치를 곱하고 그 결과를 합하여 계산
      - 다만 뉴런과 가중치가 행렬처럼 배열되어 있다는 점이 다름



(93x-1)+(139x0)+(101x1)+(26x-2)+(252x0)+(196x2)+(135x-1)+(240x0)+(48x1) = 243

3. 합성곱 신경망

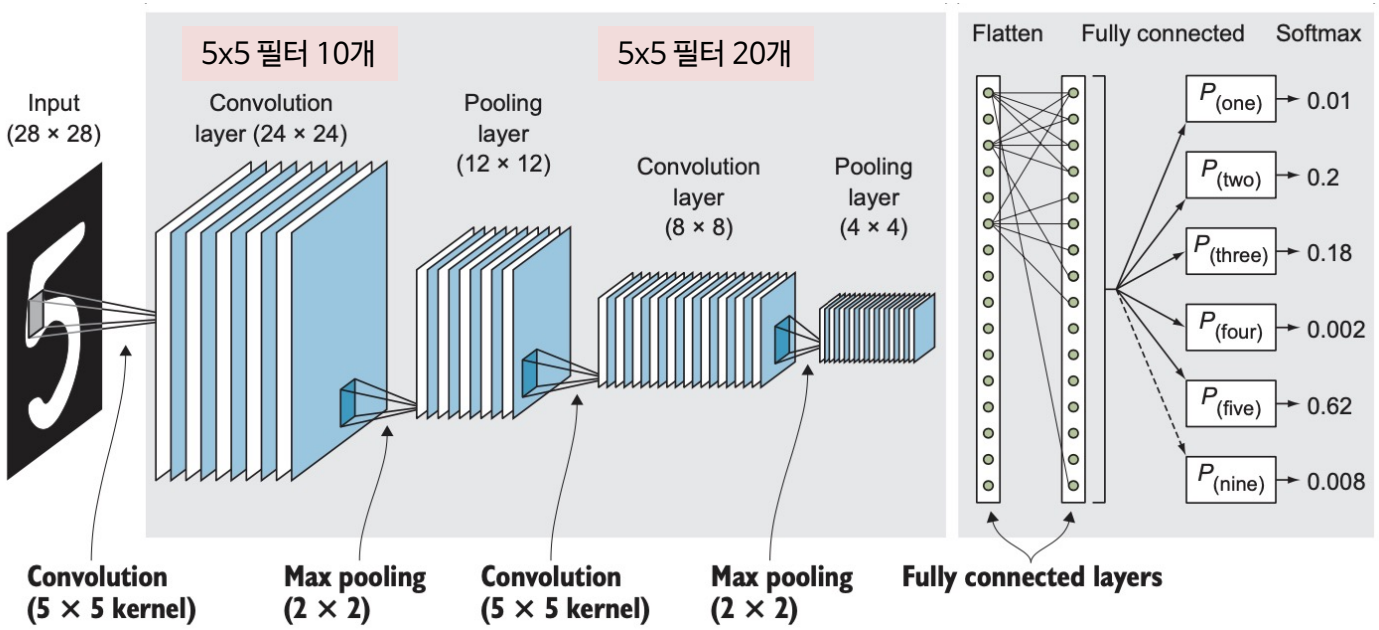
- **3.3.1 합성곱층**
  - 3.3.1.2 합성곱 연산
    - 합성곱 필터(또는 커널)은 입력 이미지 위를 이동하며 입력 이미지 전체를 커버함
    - 합성곱 필터가 한 번 움직일 때마다 위와 같은 픽셀 단위로 가중합이 계산되어 필터 중심에 해당하는 픽셀의 새로운 값이 결정 됨
    - 이런 방법으로 만들어진 이미지를 특징 맵feature map 또는 활성화 맵activation map 이라고 함



(93x-1)+(139x0)+(101x1)+(26x-2)+(252x0)+(196x2)+(135x-1)+(240x0)+(48x1) = 243

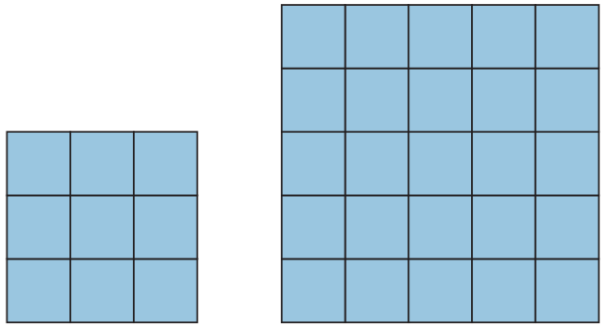
3. 합성곱 신경망

- 3.3.1 합성곱층
  - 3.3.1.3 합성곱 필터 수
    - 합성곱층에는 하나 또는 그 이상의 합성곱 필터가 있음
    - 합성곱층의 합성곱 필터 수만큼 특징 맵이 출력되기 때문에 이 **필터 수가 다음 층의 깊이를 결정**함



3. 합성곱 신경망

- **3.3.1 합성곱층**
  - 3.3.1.4 커널 크기
    - 합성곱 필터를 커널이라고 함
    - 커널은 가중치가 담긴 행렬로 입력 이미지 위를 이동하며 특징을 추출하는 역할을 함
    - 커널 크기는 합성곱 필터 크기(너비x높이)를 의미함
    - 커널 크기는 합성곱층을 만들기 위한 하이퍼파라미터 중 하나
    - 크기가 작을 수록 이미지의 세세한 부분까지 잡을 수 있고, 클수록 신경망이 복잡한 패턴을 학습할 수 있음 (그러나 연산복잡도가 올라가 과적합을 일으키기 쉬움)
    - 커널은 대부분 정사각형이며, 최소 2x2, 최대 5x5 크기의 필터가 사용됨

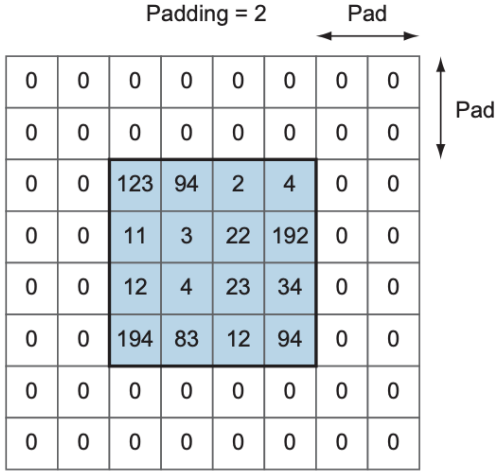


3. 합성곱 신경망

- **3.3.1 합성곱층**
  - 3.3.1.4 스트라이드와 패팅
    - 스트라이드와 패딩은 합성곱층의 출력 모양을 결정하는 하이퍼파라미터임
    - 스트라이드stride
      - 필터가 입력 이미지 위를 한번에 이동하는 픽셀 수를 의미함
      - 예를들어, 합성곱 필터가 입력 이미지 위를 한번에 한 픽셀씩 이동한다면 스트라이드 값은 1이고, 한번에 두 픽셀씩 이동한다면 스트라이드 값은 2가 됨
      - 여러 픽셀을 건너뛰면 출력의 크기가 작아지기 때문에 3 이상의 스트라이드 값은 잘 사용하지 않음
      - 스트라이드 값을 1로 설정하면 입력 이미지와 거의 같은 크기의 출력 이미지가 얻어지고, 2로 설정하면 출력 이미지가 입력 이미지의 거의 절반 크기가 됨. 여기서 거의 라고 한 이유는 패딩 설정에 따라 출력 이미지의 크기가 달라질 수 있기 때문임

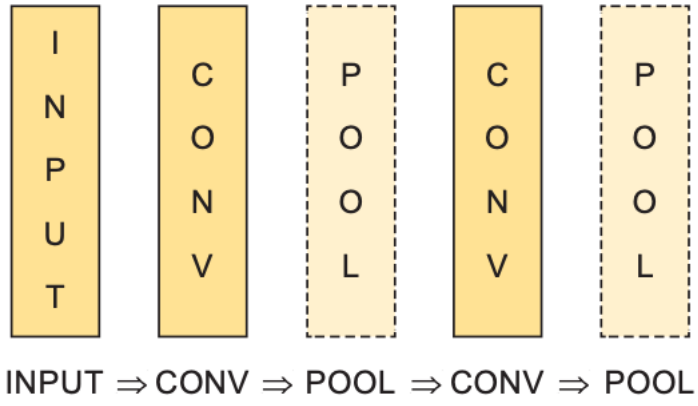
3. 합성곱 신경망

- **3.3.1 합성곱층**
  - 3.3.1.4 스트라이드와 패팅
    - 스트라이드와 패딩은 합성곱층의 출력 모양을 결정하는 하이퍼파라미터임
    - 패딩 padding
      - 〈제로 패딩〉은 이미지 둘레에 픽셀값 이 0인 추가 픽셀을 덧붙이는 것을 말함
      - 패딩의 주된 목적은 합성곱 연산을 거친 출력 이미지의 크기를 입력 이미지의 크기와 같게 유지하는 것임
      - 이미지의크기를 유지하며 합성곱 층을 여러 개 사용할 수 있음



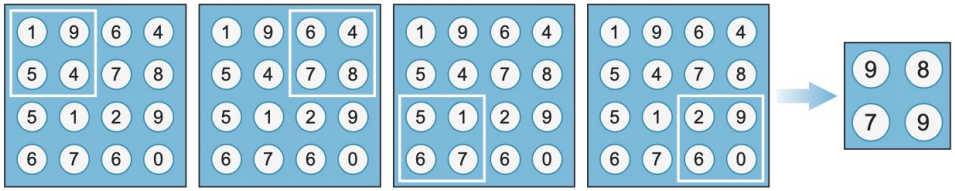
3. 합성곱 신경망

- **3.3.2 풀링층과 서브샘플링**
  - 합성곱층 수를 늘리면 파라미터(가중치) 수가 매우 많아 지며, 학습에 필요한 계산 복잡도가 상승하고 학습 시간도 오래 걸림
  - 이런 단점을 해결해주는 것이 풀링층이며, 풀링 Pooling 은 다음 층으로 전달되는 파라미터 수를 감소시키는 방법으로 신경망의 크기를 줄임
  - 풀링 연산은 최대 또는 평균 같은 통계함수로 입력 크기를 축소
  - CNN 구조는 일반적으로 합성곱층 사이에 풀링층을 끼워 넣는 경우가 많음

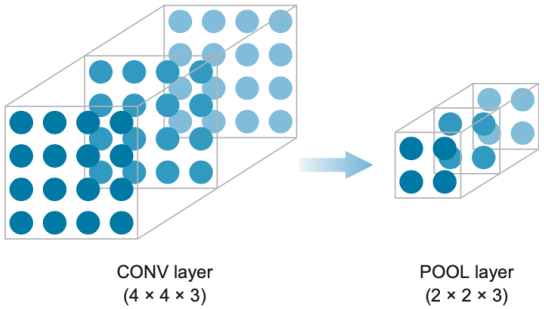


3. 합성곱 신경망

- ## 3.3.2 풀링층과 서브샘플링
  - ### 3.3.2.1 최대 풀링 max pooling
    - 최대풀링 커널에도 윈도우크기와 스트라이드 라는 하이퍼파라미터 존재
    - 다만 행렬에 별도의 가중치가 없음
    - 자기 앞의 합성곱층에서 출력한 특징 맵을 입력 받아 커널을 입력 이미지 위로 이동시키면서 윈도우 내 픽셀 값의 **최대값**을 찾아 이를 출력 이미지의 픽셀 값으로 삼음

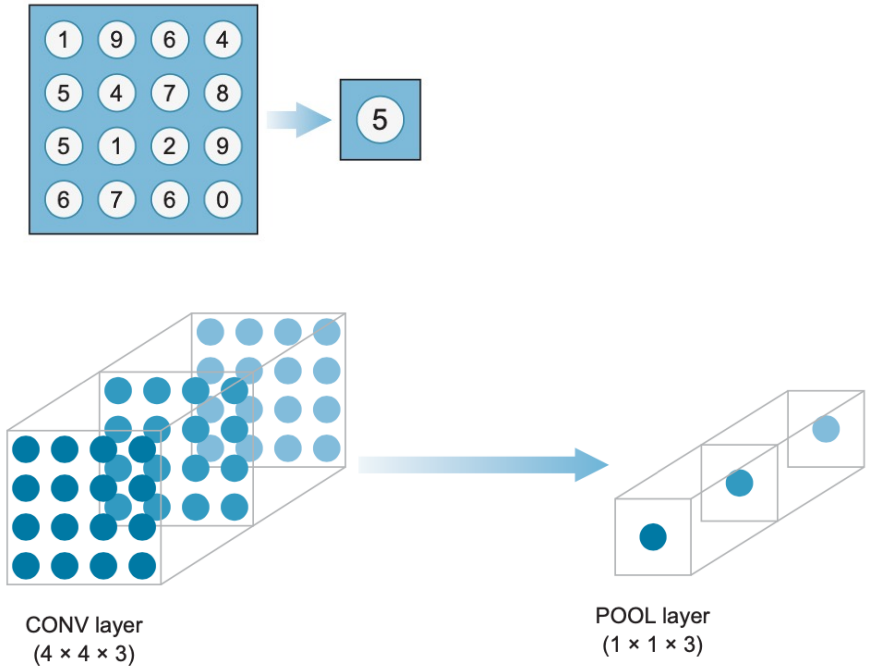


    - 합성곱층이 출력한 모든 특징 맵에 풀링을 적용하면 특징 맵의 크기는 작아지지만 **특징 맵의 개수는 유지됨**



3. 합성곱 신경망

- **3.3.2 풀링층과 서브샘플링**
  - 3.3.2.1 평균 풀링
    - 전역 평균 풀링은 특징 맵 크기를 극단적으로 줄이는 방식
    - 윈도우 크기와 스트라이드를 설정하지 않고 전체 특징 맵 픽셀 값의 평균을 구함

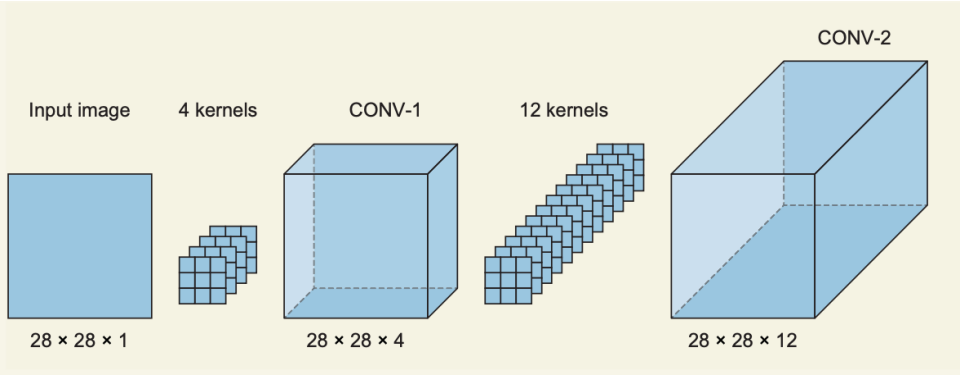


3. 합성곱 신경망

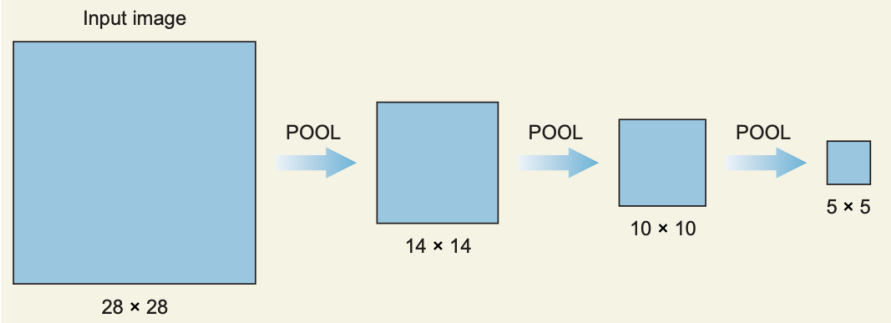
- **3.3.2 풀링층과 서브샘플링**
  - 3.3.2.2 풀링층을 사용하는 이유
    - 풀링층은 합성곱층의 규모를 줄여주는 효과가 있음
    - 풀링층을 사용하면 중요한 특징을 잃지 않으면서 이미지 크기를 줄여 다음 층에 전달
    - 일종의 이미지 압축 프로그램이라고 생각할 수 있음
    - 즉, 중요한 특징은 유지하면서 이미지의 해상도를 떨어뜨리는 것

3. 합성곱 신경망

- ## 3.3.2 풀링층과 서브샘플링
  - ### 3.3.2.3 합성곱층의 입력과 출력
    - 28x28 인 입력 이미지가 필터 수가 4, 스트라이드와 패딩이 1로 설정된 합성곱층 CONV_1을 통과하면 출력 이미지의 크기는 입력 이미지와 동일하지만 이미지의 깊이는 4가 됨

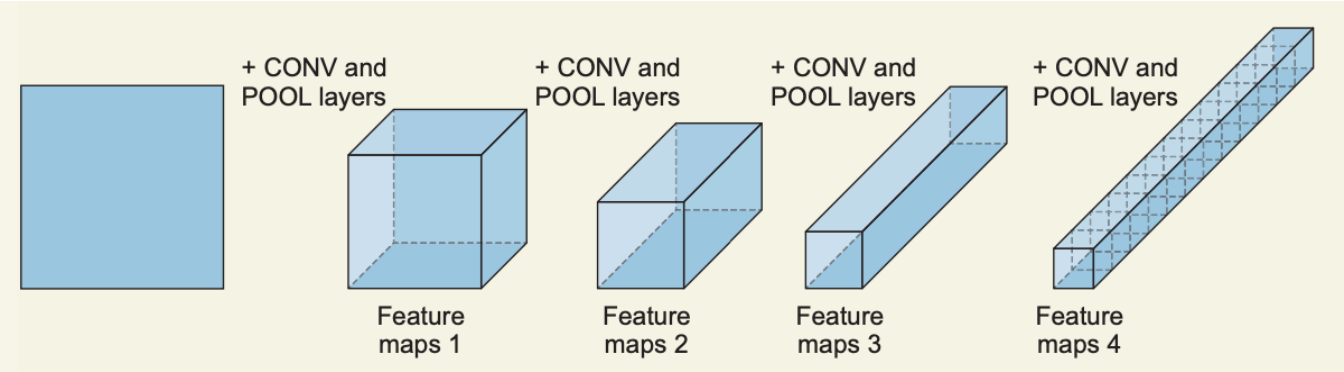


    - 풀링층을 통과한 출력은 입력과 비교해 깊이는 그대로 유지되지만 가로세로 크기는 줄어듬



3. 합성곱 신경망

- ## 3.3.2 풀링층과 서브샘플링
  - ### 3.3.2.3 합성곱층의 입력과 출력
    - 합성곱층과 풀링층을 번갈아 배치하면 출력은 다음과 같음
    - 작은 이미지가 모든 특징이 늘어선 매우 깊게 이어진 대롱과 같은 모양이 될 때까지 반복

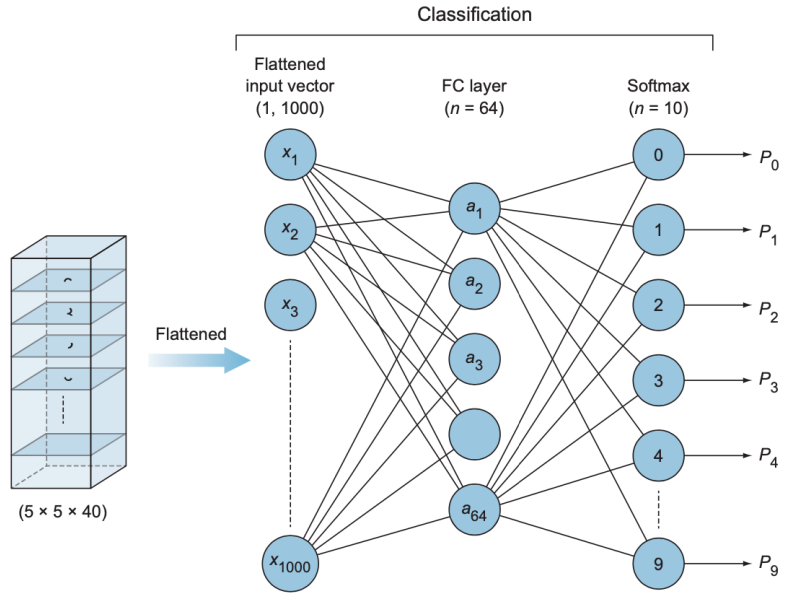


- 예를들어 출력 특징이 대롱(5x5x40) 형태가 되면 이 특징은 분류를 거칠 준비를 마친 상태임
- 이 특징의 대롱을 1차원 벡터로 변환(5x5x40 =1000)해서 전결합층에 입력함
- 변환된 벡터의 길이가 1000이므로 전결합층의 유닛 수도 1000임

3. 합성곱 신경망

- **3.3.3 전결합층**
  - 합성곱과 풀링층으로 구성된 특징 학습 과정을 거친 후, 대롱과 같은 모양으로 추출된 특징을 사용해서 실제로 이미지를 분류해야 함
  - 이미지 분류를 위해 일반적인 신경망 구조인 MLP를 사용함
- **3.3.3.1 전결합층을 사용하는 이유**
  - MLP는 분류에 효과적임

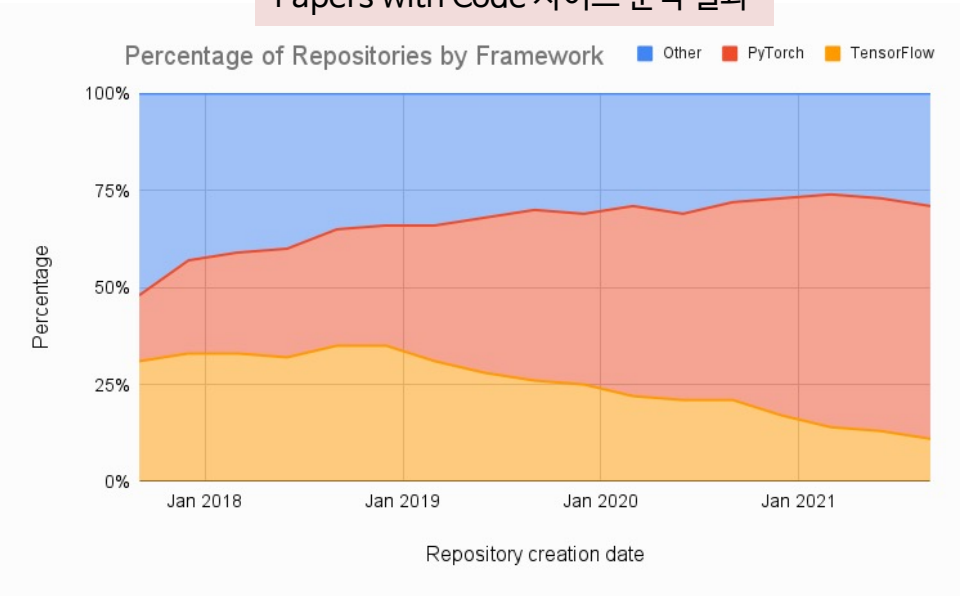
- 다층 퍼셉트론을 전결합층이라고 부르기도 함
- 혹은 밀집층, MLP, 피드포워드 신경망 이라고 부르기도 함



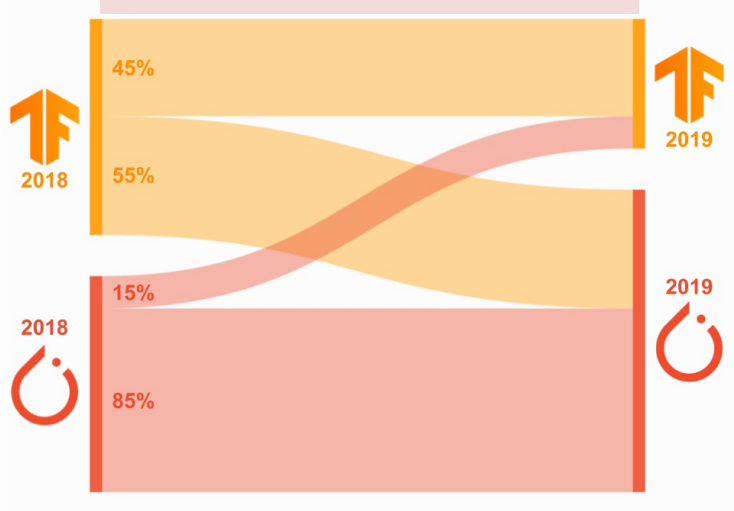
3. 합성곱 신경망

- 딥러닝 프레임워크
  - 모델 가용성 (Pytorch 〉 TensorFlow)

Papers with Code 사이트 분석 결과



빠른 모델 구현을 위한 연구자들의 선택



https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022
https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/

3. 합성곱 신경망

- **딥러닝 프레임워크**
  - 배포 인프라 (TensorFlow 〉 Pytorch)
    - 텐서플로우는 처음부터 배포 지향 애플리케이션을 위한 필수 프레임워크 였음
    - 파이토치는 배포 관점에서 극도로 부진했지만 최근 몇 년 동안 격차를 좁히기 위해 노력함

TensorFlow	Pytorch
▪ TensorFlow Serving (클라우드 배포) ▪ TensorFlow Lite (모바일 배포)	▪ TorchServe (클라우드 배포) ▪ PyTorch Live (모바일 배포)

https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022
https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/

3. 합성곱 신경망

- 딥러닝 프레임워크
  - 생태계 (TensorFlow == Pytorch)
    - 산업 환경에서 딥러닝 엔지니어링을 수행하는 경우 TensorFlow를 사용할 가능성이 높음
    - 최근 SOTA 모델을 액세스해야 하는 경우 Pytorch를 사용할 가능성이 높음

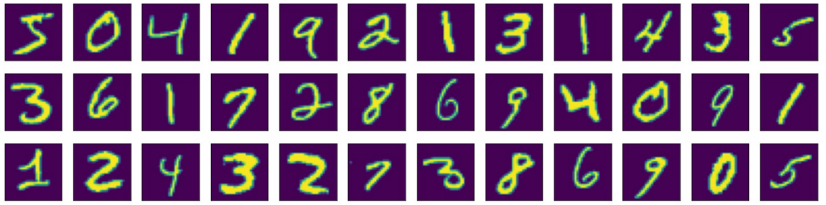
TensorFlow	Pytorch
▪ TensorFlow Hub (이미지, 비디오, 오디오, 텍스트 등 지원) ▪ Model Garden ▪ TensorFlow Extended (배포용) ▪ Vertex AI (교육용) ▪ TensorFlow.js (자바스크립트 지원용) ▪ TensorFlow Cloud (클라우드 지원용) ▪ Colab (클라우드기반 노트북 환경)	▪ Pytorch Hub (비전, 오디오, NLP등) ▪ SpeechBrain (음성툴킷) ▪ TorchElastic (분산 훈련용) ▪ TorchX (배포용 SDK) ▪ Lightning (교육용)

https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022
https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/

3. 합성곱 신경망

- **수업에서 사용하는 딥러닝 프레임워크**
  - 케라스 소개 (Keras)
    - TensorFlow 2.0에 포함
    - 딥러닝 모델을 간편하게 만들고 훈련시킬 수 있는 파이썬을 위한 딥러닝 프레임워크
    - 신속하게 실험을 해야 하는 연구자들을 위해 개발되었음
    - MIT 라이센스를 따르므로 상업적인 프로젝트에도 자유롭게 사용 가능 (2018년 기준)
    - 파이썬 2.7에서 3.6까지 모든 버전과 호환
    - 공홈: https://keras.io/
    - 공홈-한글화: https://keras.io/ko/
  - 케라스의 특징
    - 동일한 코드로 CPU와 GPU 에서 실행할 수 있음
    - 사용하기 쉬운 APU를 가지고 있어 딥러닝 모델의 프로토타입을 빠르게 만들 수 있음
    - (컴퓨터 비전을 위한) 합성곱 신경망, (시퀀스 처리를 위한) 순환 신경망을 지원하며 이 둘을 자유롭게 조합하여 사용할 수 있음

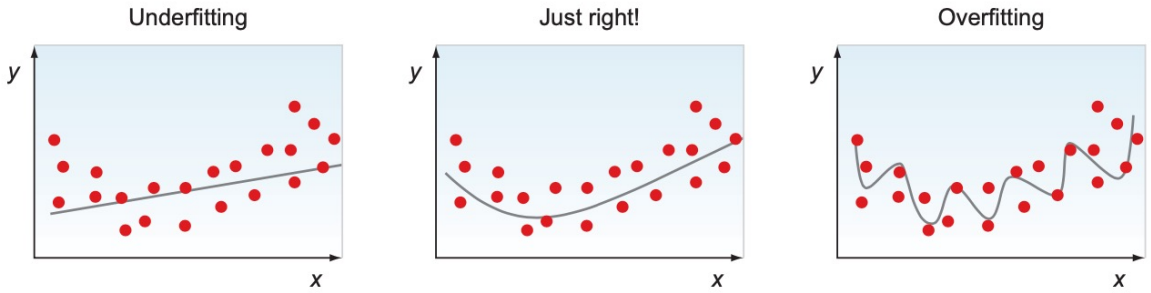
3. 합성곱 신경망



- **3.4 CNN을 이용한 이미지 분류**
  - CNN 모델을 학습하여 MNIST 데이터셋의 손글씨 이미지를 분류하는 것
  - 학습데이터: MNIST
    - 딥러닝계의 'Hello World' 데이터 셋
    - 0~9숫자 손글씨 데이터
    - 28x28 크기의 흑백 이미지 10가지 클래스로 구성
  - 학습모델
    - CNN 2개 + FC 2개
  - 테스트결과: 약 99% 정확도
  - 실습코드
    - CPU 를 활용한 교재 코드
      - https://www.kaggle.com/yukyungchoi/2022-dl-w3p1
    - Keras 메뉴얼 샘플 코드
      - https://keras.io/examples/vision/mnist_convnet/

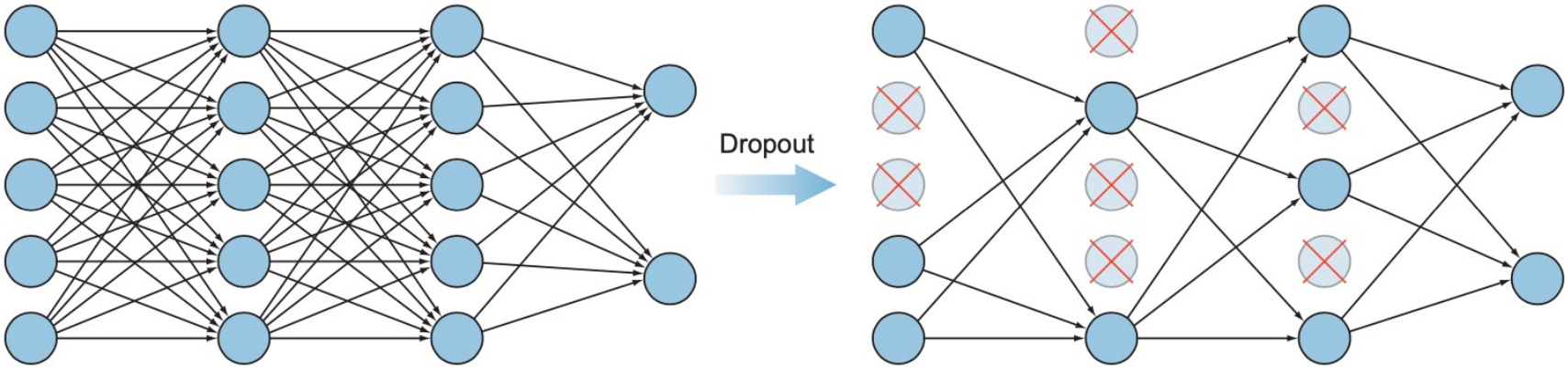
3. 합성곱 신경망

- **3.5 과적합을 방지하기 위해 드롭아웃층 추가하기**
- **3.5.1 과적합이란**
  - 머신러닝에서 모델의 성능이 잘 나오지 않는 이유 중 한가지
  - 과소적합 underfitting
    - 모델이 학습 데이터에 부합하지 못하는 현상
    - 모델이 데이터를 나타내기에 너무 단순한 경우에 발생
  - 과적합 overfitting
    - 모델이 학습 데이터에 지나지게 부합하는 현상 (학습 오차가 매우 적음)
    - 모델이 표현력이 매우 좋은 신경망
    - 처음 보는 데이터는 잘 예측하지 못해 일반화 성능이 떨어지는 경우



3. 합성곱 신경망

- **3.5.2 드롭아웃층이란**
  - 과적합을 방지하기 위한 수단 중 가장 널리 사용
  - 드롭아웃층을 적용하면 층을 구성하는 뉴런(노드)의 일정 비율을 비활성화 함
    - 비활성화란 순방향 혹은 역전파 계산에 참여하지 않는다는 뜻
  - 이 비율은 하이퍼 파라미터로 지정
  - 학습단계에서만 사용되는 층 (테스트 단계에서는 사용되지 않음)
  - 모든 노드가 협력하여 영향력이 지나치게 약하거나 강한 노드가 생기는 것을 방지



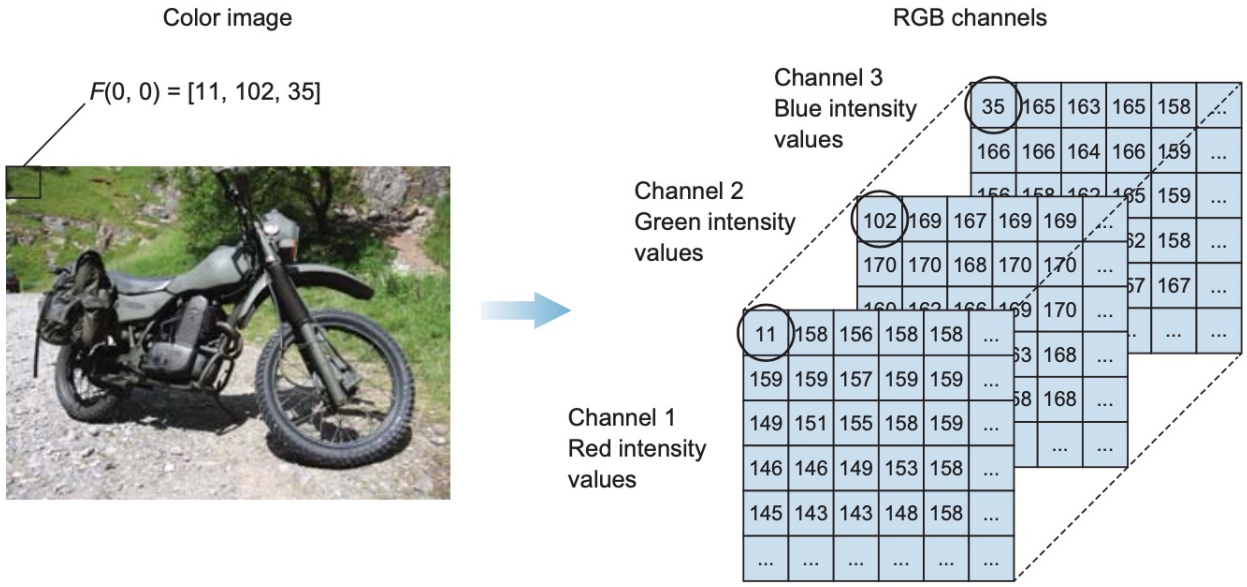
3. 합성곱 신경망

- **3.5.3 드롭아웃층이 필요한 이유**
  - 뉴런은 학습 과정을 거치며 상호 의존 관계를 구축
  - 상호 의존관계는 각 뉴런의 영향력을 결정하며 과적합으로 이어지는 원인이 됨
  - 노드 중 일부를 무작위로 비활성화한다면 다른 노드는 비활성화된 노드가 가진 특징 없이 패턴을 학습해야 하며, 모든 특징이 이런식으로 비활성화될 수 있으므로 가중치가 특징 간에 고르게 분산되는 효과가 발생하며 더 잘 학습된 뉴런으로 이어짐
    - 드롭아웃은 일종의 앙상블 학습 기법임

- **3.5.4 CNN 구조 중 어디에 드롭아웃층을 끼워 넣어야 할까**
  - 드롭아웃층은 추출된 특징의 1차원 벡터 변환이 끝난 다음부터 마지막 출력 층 사이에 배치하는 것이 일반적임

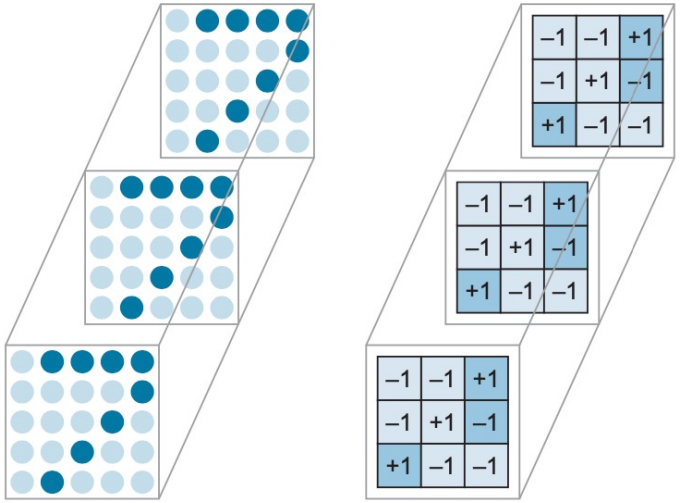
3. 합성곱 신경망

- 3.6 컬러 이미지의 합성곱 연산
  - 컴퓨터가 본 회색조 이미지는 너비, 높이를 가진 2차원 행렬
    - 각 픽셀 값은 원색의 강도, 밝기값을 의미
  - 컴퓨터가 본 컬러 이미지는 너비, 높이, 깊이를 가진 3차원 행렬
    - 2차원 행렬 3개 (빨간색, 녹색, 파란색에 해당)



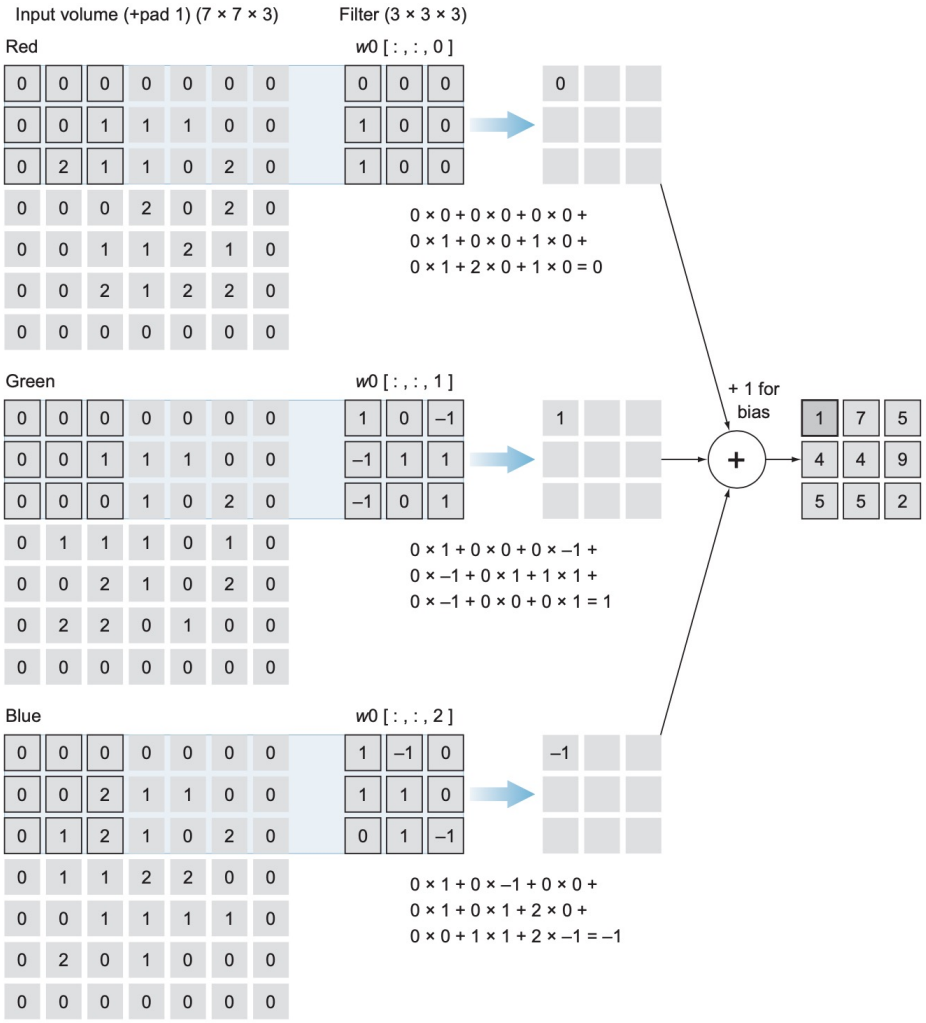
3. 합성곱 신경망

- **3.6.1 컬러 이미지를 대상으로 합성곱 연산하기**
  - 컬러 이미지의 합성곱은 회색조 이미지와 마찬가지로 합성곱 커널을 입력 이미지 위로 이동시키며 특징 맵을 계산
  - 컬러 이미지의 합성곱 연산은 합해야 할 항수가 세배로 늘어난 것만 제외하면 회색조 이미지의 합성곱과 동일
    - 색상 채널별로 별도의 필터를 갖음



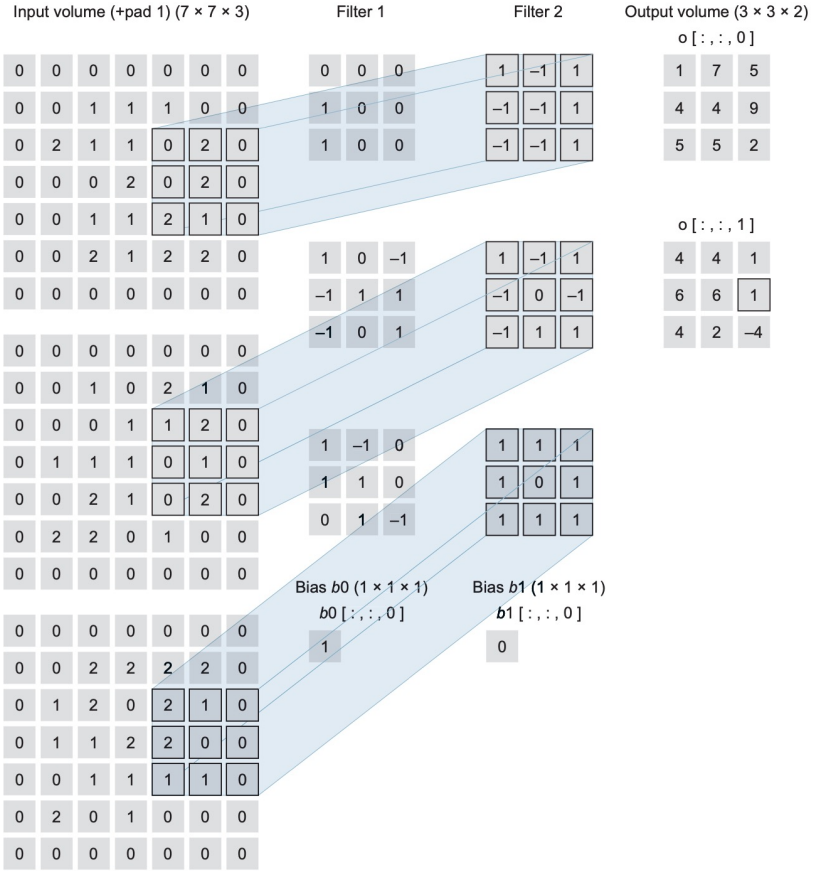
3. 합성곱 신경망

- 3.6.1 컬러 이미지를 대상으로 합성곱 연산하기

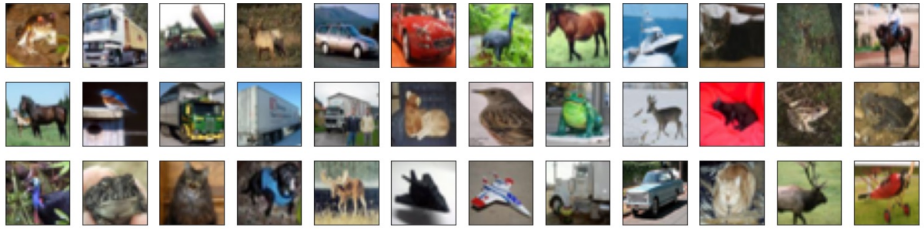


3. 합성곱 신경망

- **3.6.2 계산 복잡도의 변화**
  - 컬러 이미지는 회색조 이미지보다 공간 복잡도를 크게 증가시킴
  - 따라서 색상이 큰 영향을 못 미치는 경우에는 학습 데이터를 회색조 이미지로 변환



3. 합성곱 신경망



- **3.7 프로젝트: 컬러 이미지 분류 문제**
  - CNN 모델을 학습하여 CIFAR-10 데이터셋의 이미지를 분류하는 것
  - 학습데이터: CIFAR-10
    - 컴퓨터 비전 분야에서 널리 알려진 사물 인식 문제 데이터 셋
    - 8천만 장으로 구성된 '80 Million Tiny Image' 데이터셋의 서브셋
    - 32x32 크기의 컬러 이미지 6만장으로 클래스당 6천 장씩 10가지 클래스로 구성
  - 학습모델: AlexNet 간략버전
    - CNN 3개 + FC 2개
  - 테스트결과: 약 67% 정확도
  - 실습코드
    - CPU 를 활용한 코드 (한 에포크당 평균 23초)
      - https://www.kaggle.com/yukyungchoi/2022-dl-w3-project-cpu
    - GPU 를 활용한 코드 (한 에포크당 평균 5초)
      - https://www.kaggle.com/yukyungchoi/2022-dl-w3-project-gpu