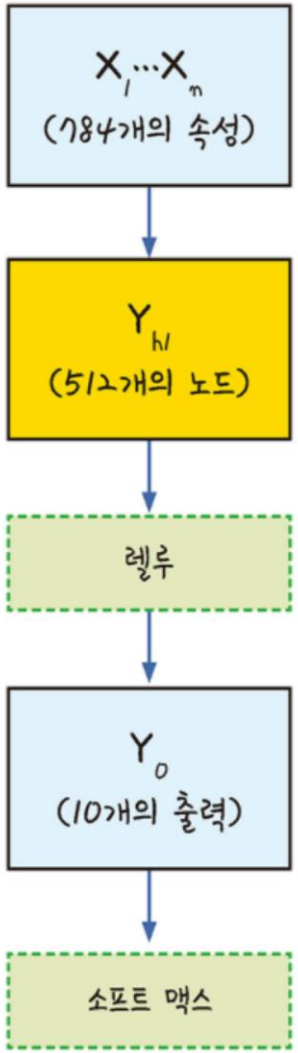
CNN

컨볼루션 신경망

인공신경망 손글씨 모델

- 인공신경망으로 만든 MNIST_Simple.ipynb는 98.21%의 정확도를 갖는 단순신경망 모델
- 도식화
  - 은닉층이 하나인 모델

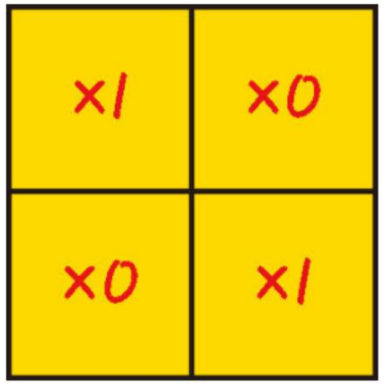


CNN(컨볼루션 신경망)

- 입력된 이미지에 다시 한번 특징(Feature)를 추출하기 위해 마스크(Mask)를 사용하는 기법
- 마스크는 필터, 윈도, 커널이라고 함

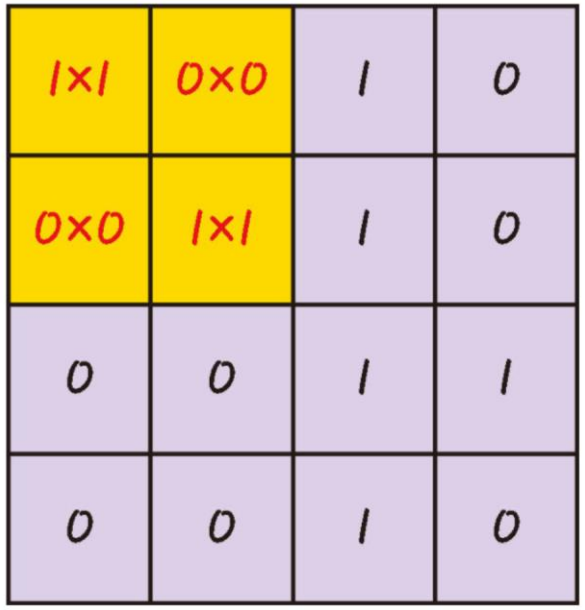
1	0	1	0
0	1	1	0
0	0	1	1
0	0	1	0

원본 이미지

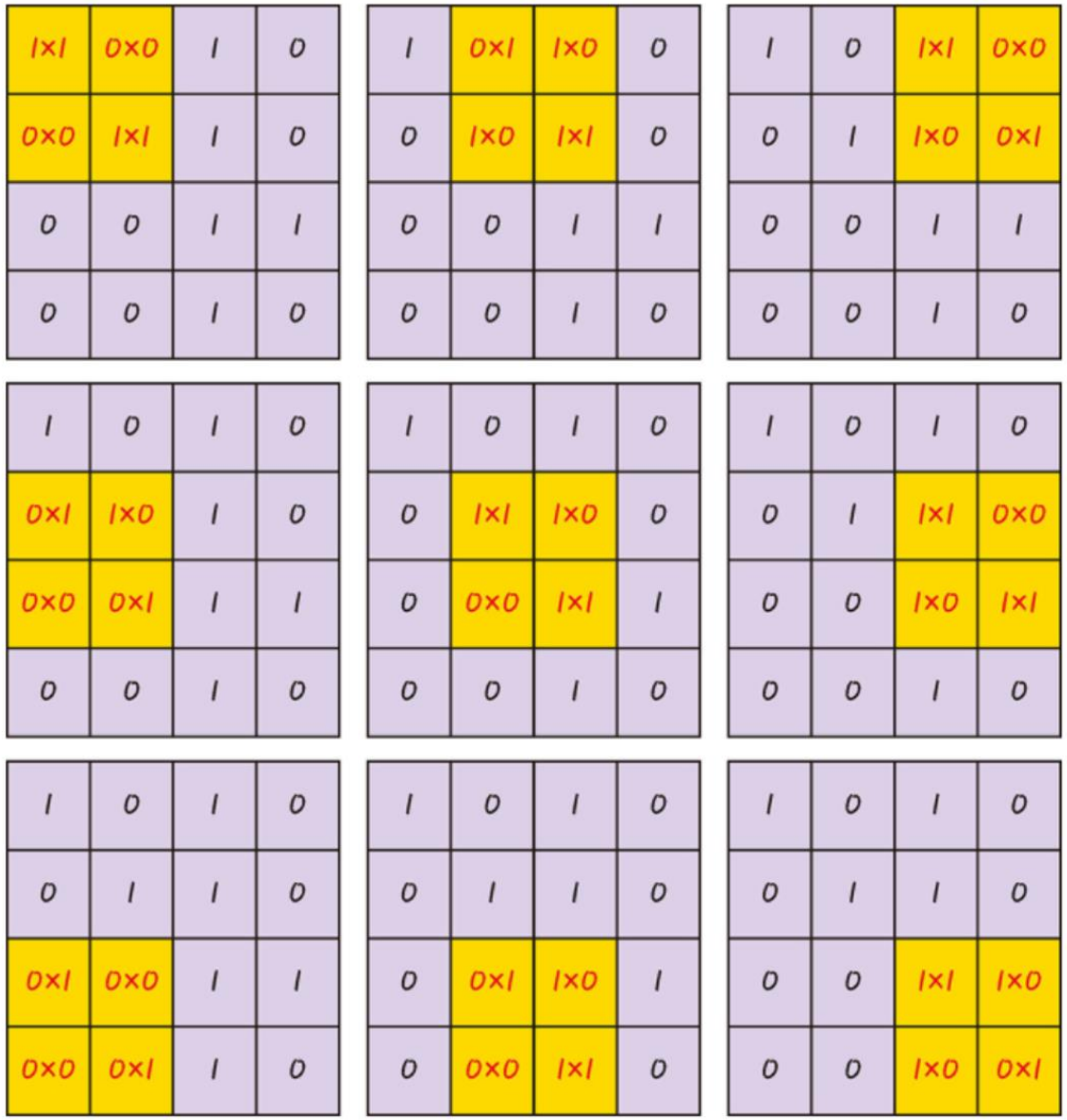


Mask 2 * 2
가중치 포함

CNN(컨볼루션 신경망)

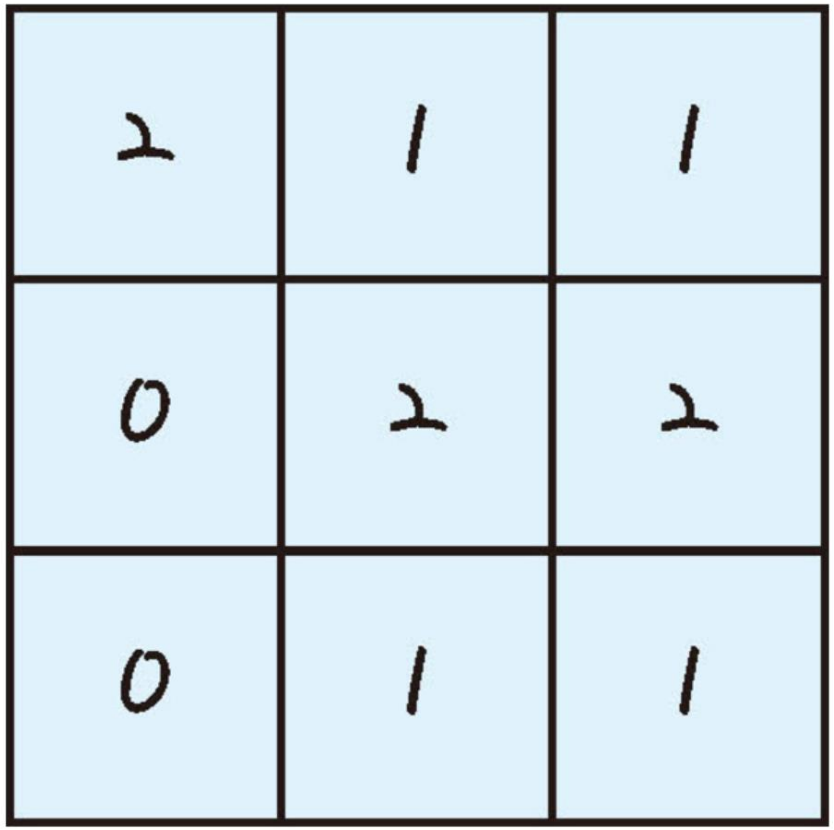


원래 값에 가중치를 곱해 줌
(1*1)+(0*0)+(0*0)+(1*1) = 2

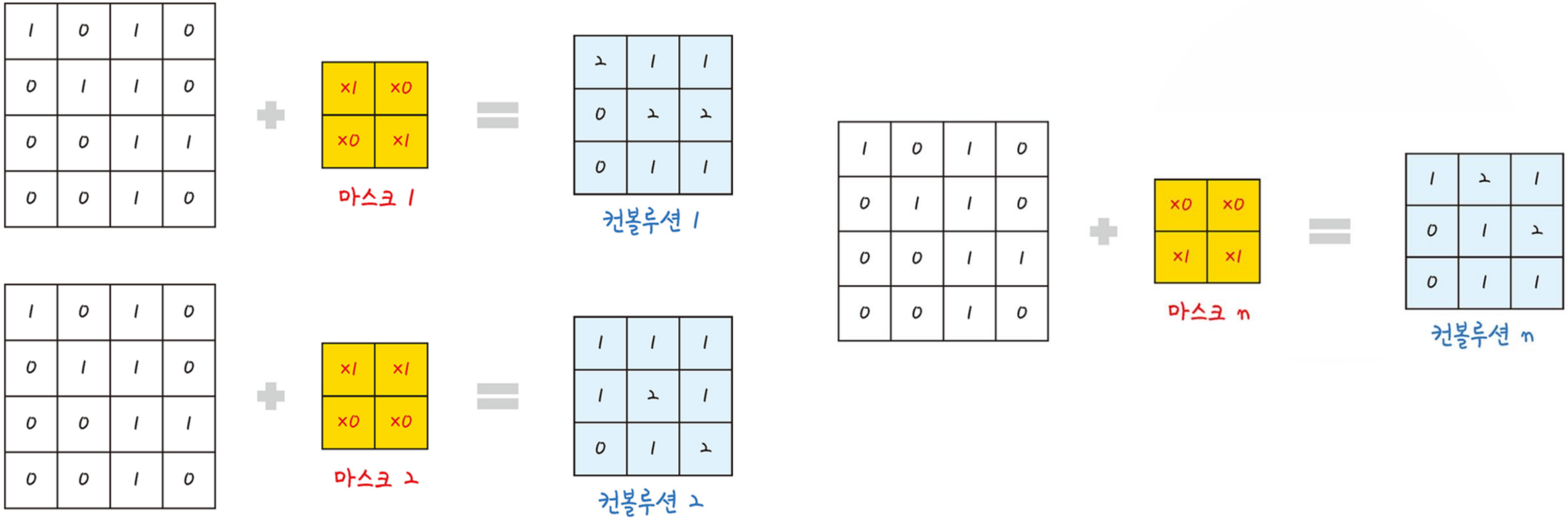


CNN(컨볼루션 신경망)

- 새롭게 만들어진 층
  - 컨볼루션 (합성곱)
  - 여러 개를 만들면 여러 개의 컨볼루션이 만들어 짐



CNN(컨볼루션 신경망)

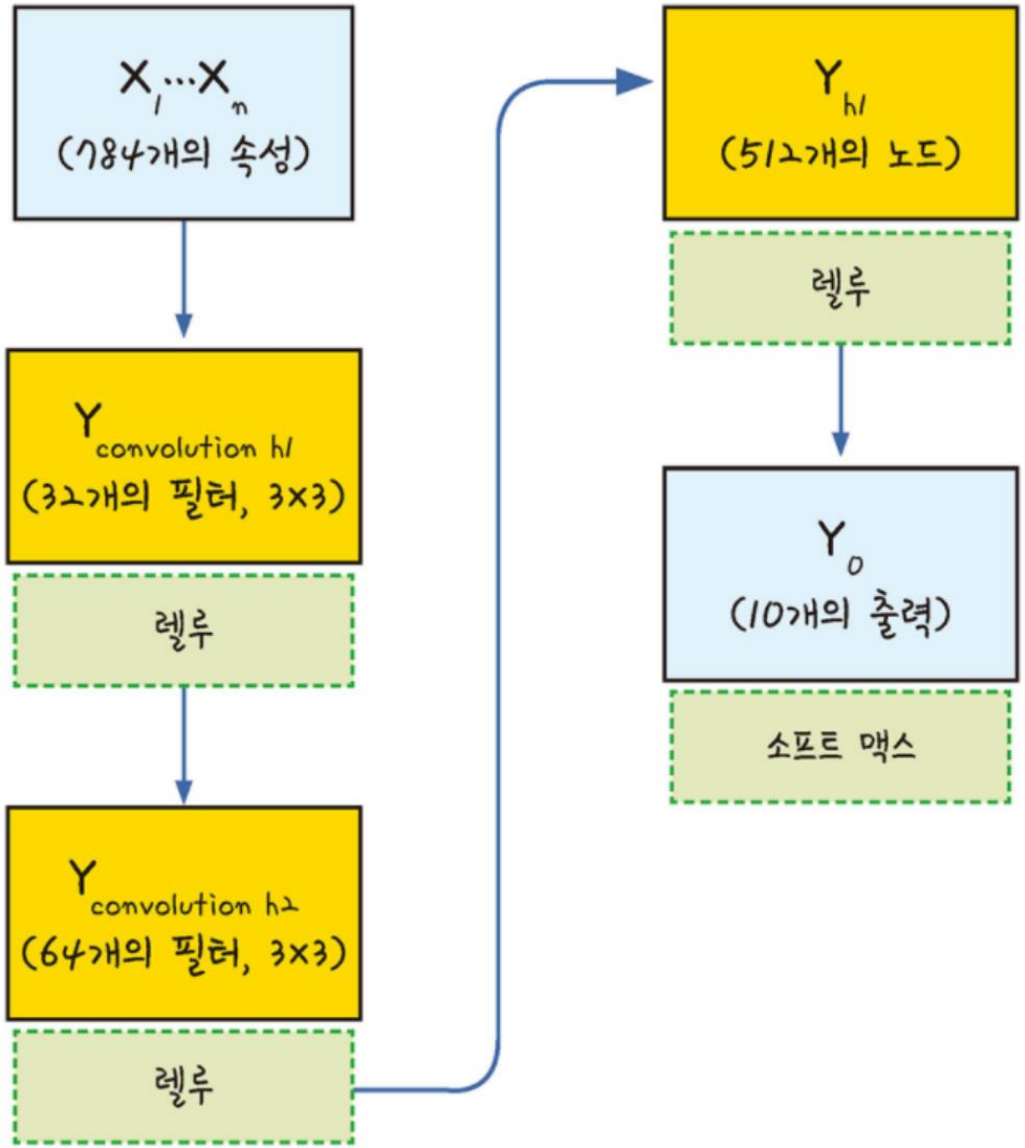

1 0 1 0
0 1 1 0
0 0 1 1
0 0 1 0
×1 ×0
×0 ×1
마스크 1
2 1 1
0 2 2
0 1 1
컨볼루션 1
1 0 1 0
0 1 1 0
0 0 1 1
0 0 1 0
×1 ×1
×0 ×0
마스크 2
1 1 1
1 2 1
0 1 2
컨볼루션 2
1 0 1 0
0 1 1 0
0 0 1 1
0 0 1 0
×0 ×0
×1 ×1
마스크 n
1 2 1
0 1 2
0 1 1
컨볼루션 n

CNN(컨볼루션 신경망)

- 케라스에서 컨볼루션 층을 추가하는 함수는 Conv2D()
  - model.add(Conv2D(32, kernel_size=(3,3), input_shape=(28,28,1),
    activate='relu')
    - 첫번째 매개변수 적용시킬 마스크의 개수 지금 32개의 마스크를 적용
    - kernel_size 마스크의 크기
    - 입력값(행, 열, 색상)
    - 네번째 매개변수 활성함수 현재는 relu 선택
  - model.add(Conv2D(64, (3,3), activation='relu'))
    - 64개 층을 추가한 컨볼루션 층

CNN(컨볼루션 신경망)

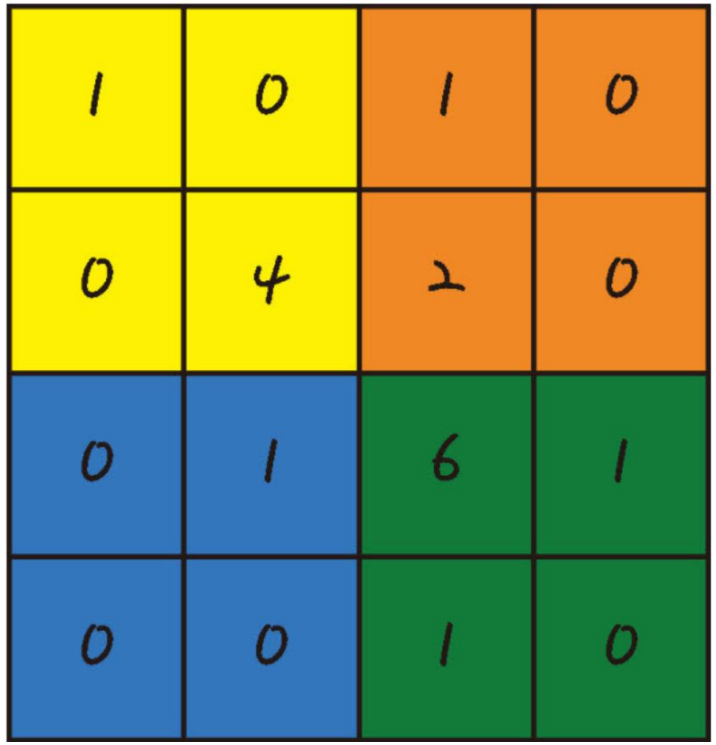
- 컨볼루션 층의 적용

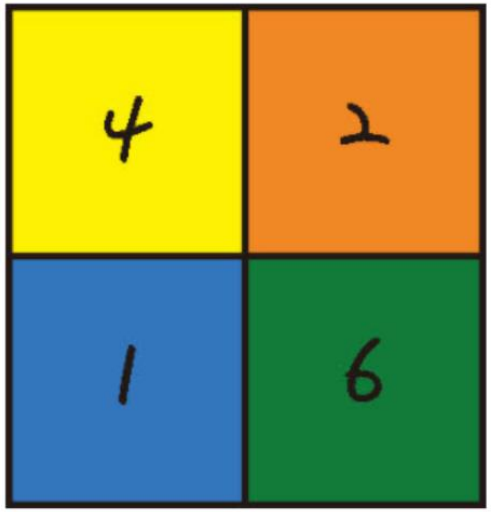


맥스플링(Max pooling)

- 컨볼루션 층을 통한 이미지 특징 추출 후 결과가 크고 복잡하면 축소가 필요
- 축소 과정을 풀링(pooling), 서브 샘플링(sub sampling)
- 풀링 기법의 알고리즘
  - 정해진 구역안에서 최댓값을 구함

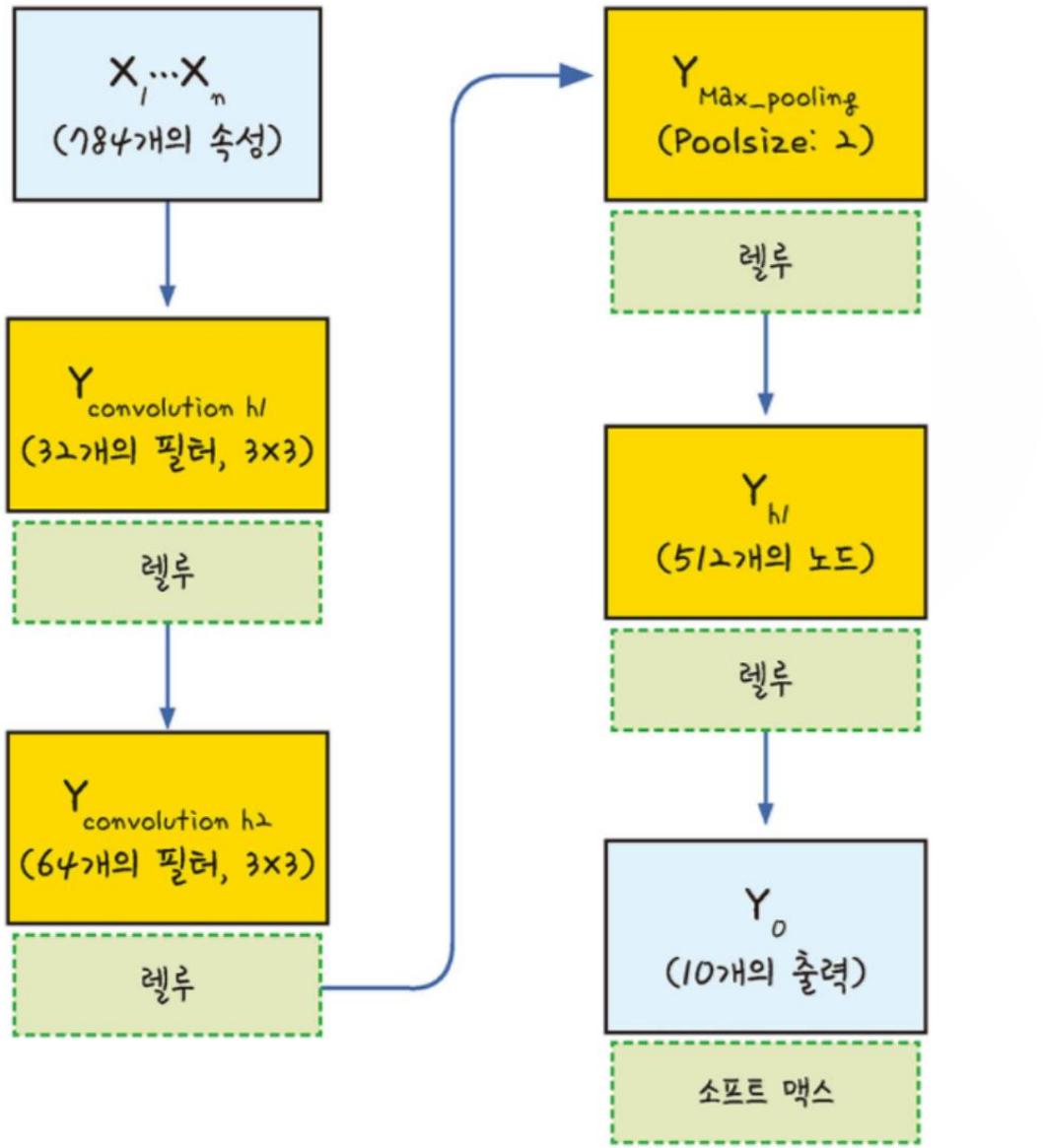
1	0	1	0
0	4	2	0
0	1	6	1
0	0	1	0





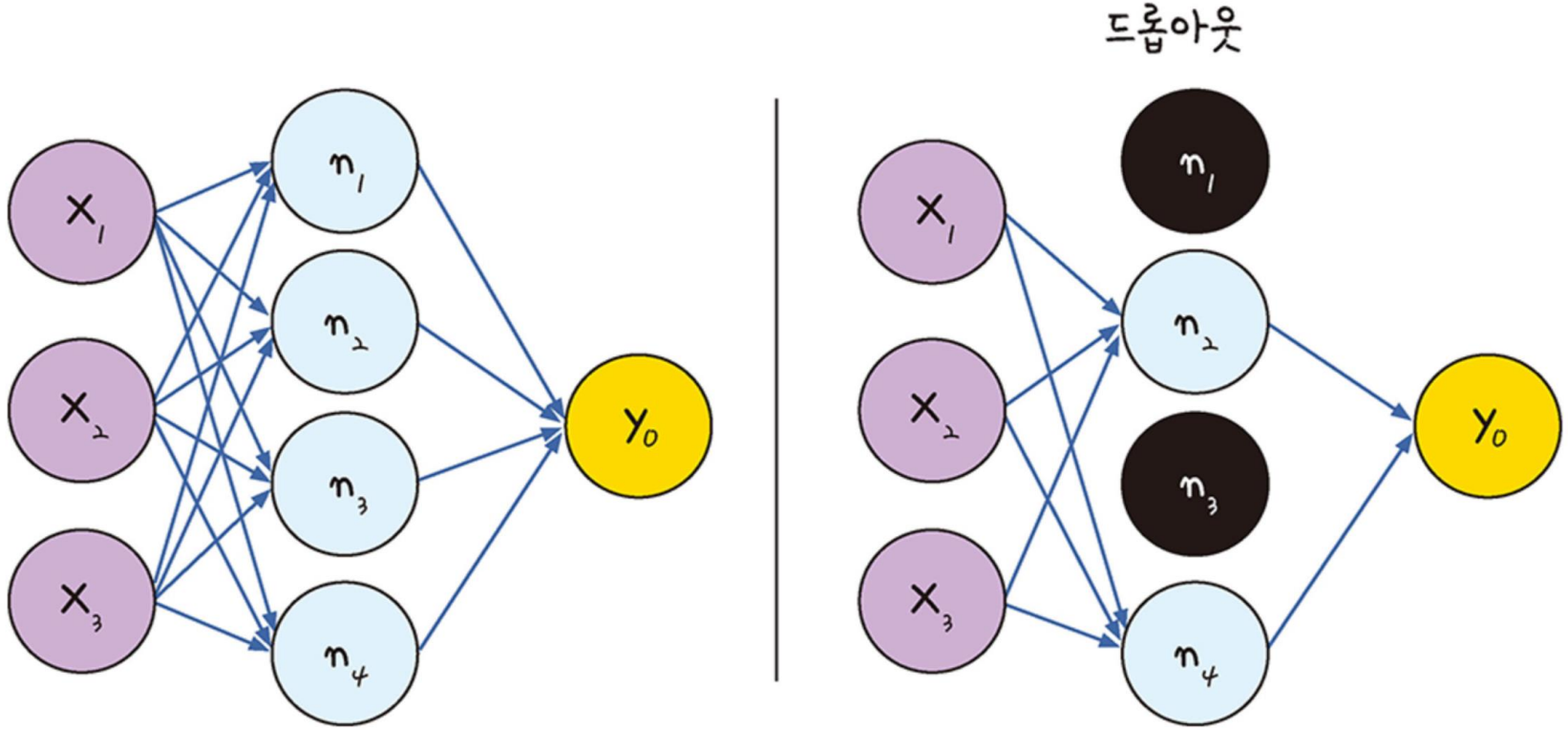
맥스플링(Max pooling)

- Maxpooling2D 함수를 사용함
- model.add(Maxpooling2D(pool_size=2)
  - pool_size는 풀링 창의 크기를 정하는 것 2로 하면 전체 크기가 절반으로 줄어듬



드롭아웃(Dropout)

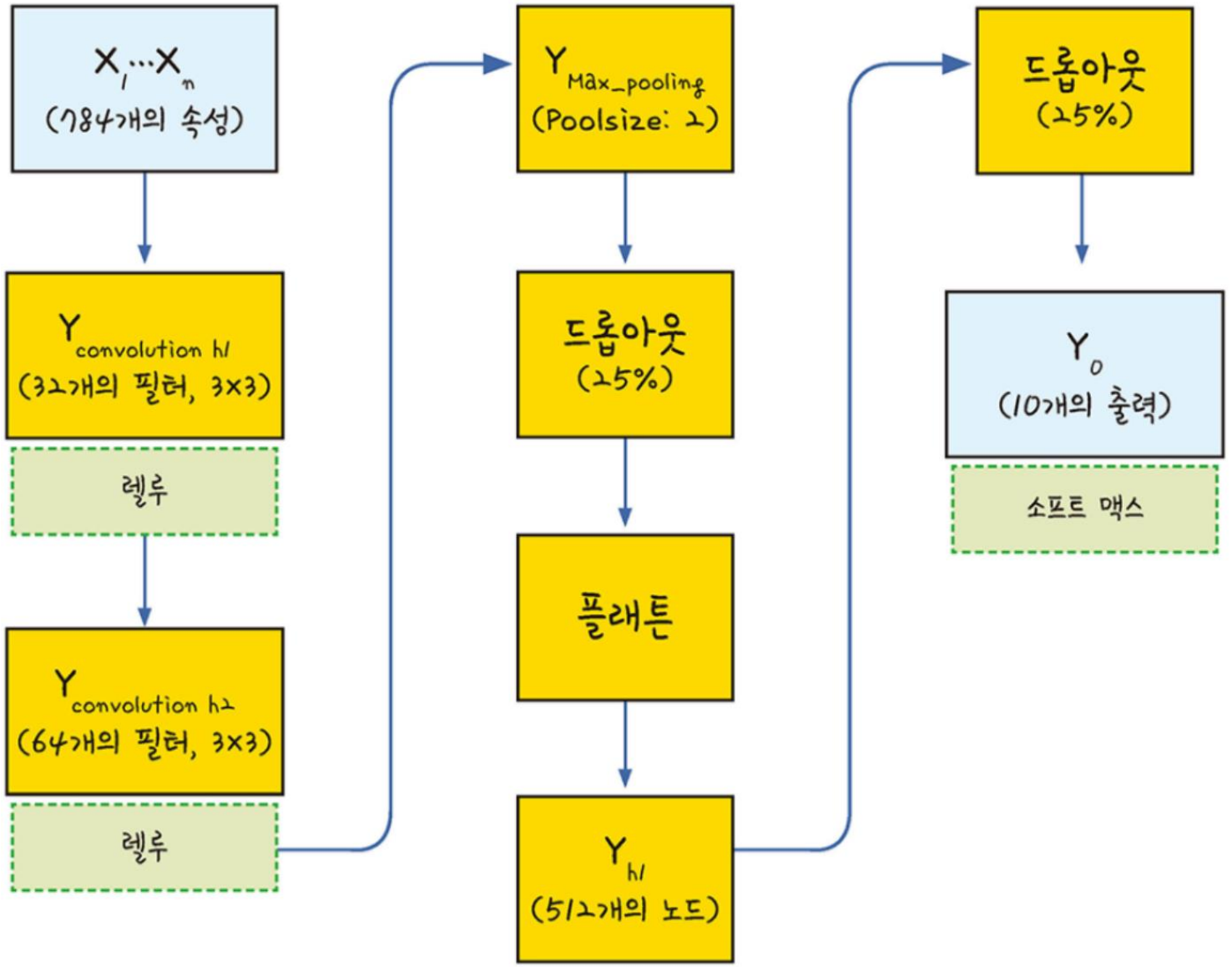
- 노드가 많거나 층이 많다고 무조건 학습이 좋아지지는 않음
- 학습의 효율은 과적합을 효율적으로 피해가는 데 있음
- 과적합을 회피할 수 있는 간단하지만 효율적인 기법
- 은닉층에 배치된 노드 중 일부를 임의로 꺼주는 기법



드롭아웃(Dropout)

- model.add(Dropout(0.25)
  - 25%의 노드를 꺼줌
- model.add(Flatten())
  - 컨볼루션 층이나 맥스풀링 층은 주어진 배열을 2차원 배열인 형태로 다룸
  - 1차원 함수로 변경한 후 사용하여야 활성함수가 있는 층에서 사용 가능함
  - 따라서 2차원 배열을 1차원 배열로 변경해줄 필요가 있음

CNN 작업 과정 도식화

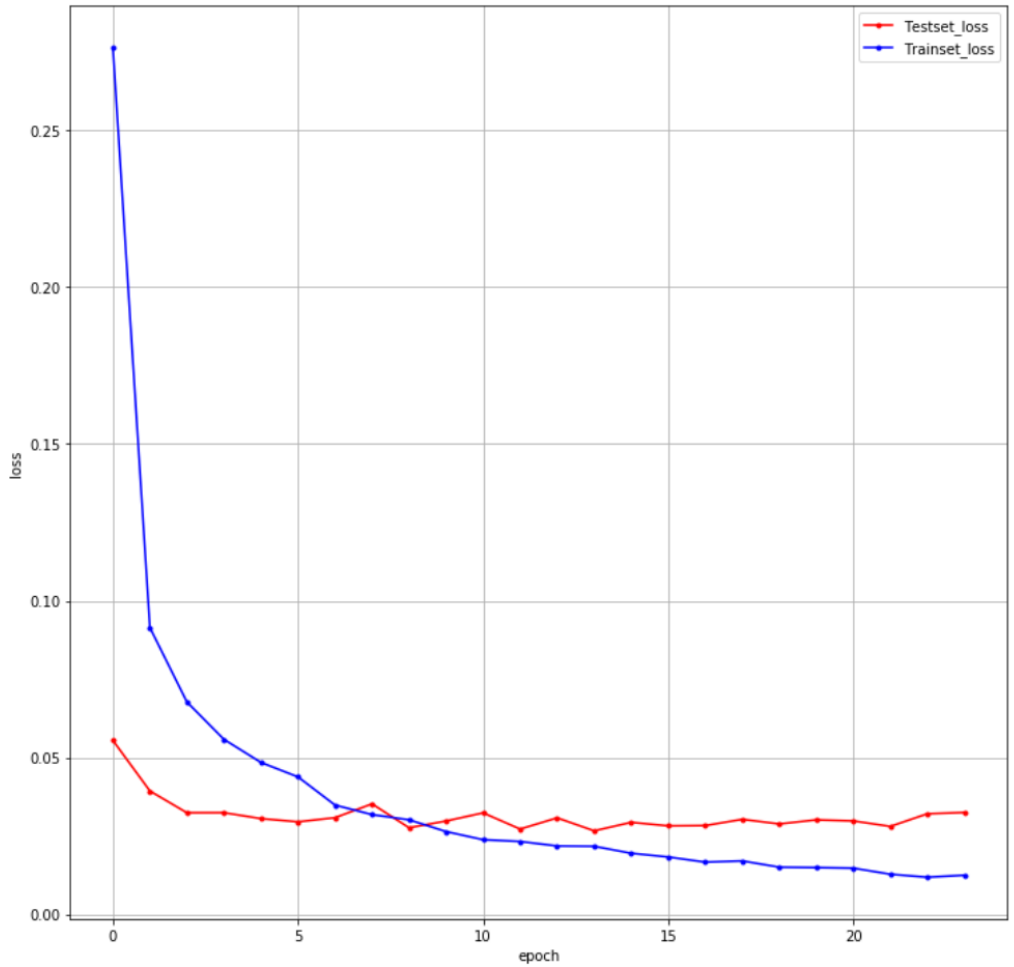

$X_1 \cdots X_n$
(784개의 속성)
$Y_{convolution\ h1}$
(32개의 필터, 3x3)
렐루
$Y_{convolution\ h2}$
(64개의 필터, 3x3)
렐루
$Y_{Max\_pooling}$
(Poolsize: 2)
드롭아웃
(25%)
플래튼
Y_{h1}
(512개의 노드)
드롭아웃
(25%)
Y_o
(10개의 출력)
소프트 맥스

최종 결과

```
In [7]: # 테스트 정확도 출력
        print("₩n Test Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

```
10000/10000 [==============================] - 2s 231us/step

 Test Accuracy: 0.9920
```



- 98%에서 99.20%로 향상
- epoch은 7정도에서 학습중단
  - 과적합 방지

최종결과

- 알아내지 못한 손글씨 샘플