

# 합성곱 신경망(CNN)

2019.11



### ■ 심층 신경망(DNN: Deep Neural Network)

#### ❖ 딥러닝 학습의 일반적 절차

- 적절한 네트워크
  - 구조
  - 비선형성 획득 방법: 활성화 함수
- 그래디언트 체크
- 학습 파라미터 초기화
- 학습 파라미터 최적화
- 과적합 방지
- 기타
  - 학습률 감소

## ■ MNIST 손글씨 데이터



- 미국 국립표준기술원(NIST)이 고등학생과 인구조사국 직원 등이 쓴 손글씨를 이용해 만든 데이터
- 70,000개의 글자 이미지에 각각 0부터 9까지 이름표를 붙인 데이터셋



## ■ MNIST 손글씨 데이터

### ❖ 데이터 전처리

- MNIST 데이터 불러오기

```
from tensorflow.keras.datasets import mnist

(X_train, Y_class_train), (X_test, Y_class_test) = mnist.load_data()

print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
```

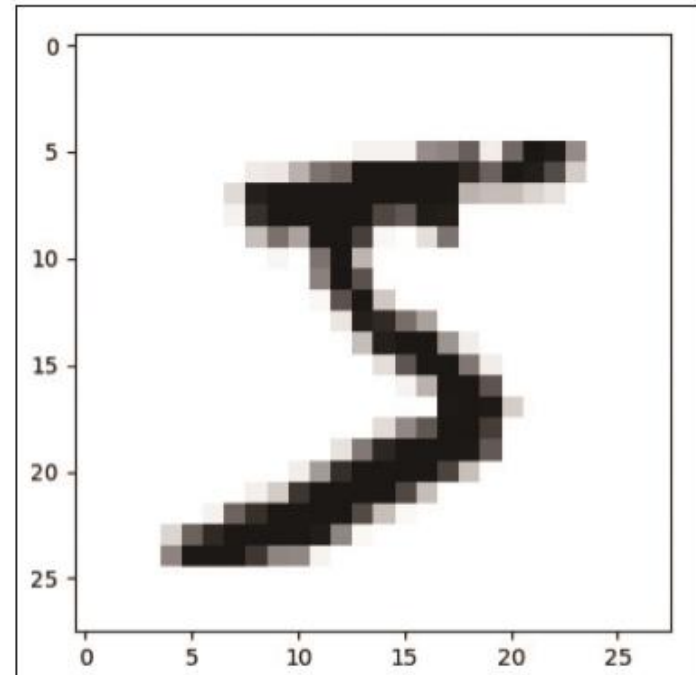
```
학습셋 이미지 수: 60000 개
테스트셋 이미지 수: 10000 개
```

### ■ MNIST 손글씨 데이터

#### ❖ 데이터 전처리

- MNIST 데이터 이미지 보기

```
import matplotlib.pyplot as plt
plt.imshow(X_train[0], cmap='Greys')
plt.show()
```





## ■ MNIST 손글씨 데이터

### ❖ 데이터 전처리: 정규화(Normalization)

- 케라스는 데이터가 0에서 1 사이의 값일 때 최적의 성능을 보임
- 0~255 사이의 값으로 이루어진 값을 0~1 사이의 값으로 바꿔야 함

```
X_train = X_train.astype('float64')  
X_train = X_train / 255
```

```
X_test = X_test.astype('float64') / 255
```

## ■ MNIST 손글씨 데이터

❖ 데이터 전처리: 원 핫 인코딩(One-hot encoding)

- 0~9까지의 정수형 값을 갖는 현재 상태에서 0 또는 1로만 이루어진 벡터로 값을 수정해야 함
- 예를 들어 class가 '3'이라면, [3]을 [0,0,0,1,0,0,0,0,0,0]로 바꿔 주어야 함

```
Y_train = tf.keras.utils.to_categorical(Y_class_train, 10)  
Y_test = tf.keras.utils.to_categorical(Y_class_test, 10)
```



## ■ MNIST 손글씨 인식을 DNN으로 해결

### ❖ 1차원 배열로 전환

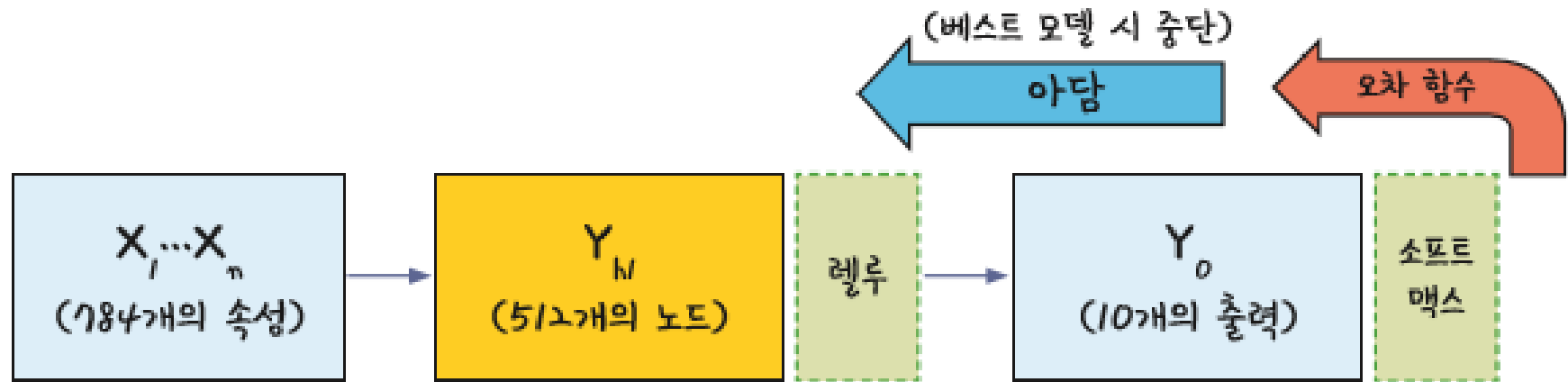
- 가로 28, 세로 28의 2차원 배열을 784개의 1차원 배열로 바꿔 주어야 함

```
X_train = X_train.reshape(X_train.shape[0], 784)
```

- 1차원 배열로 전환하고 정규화

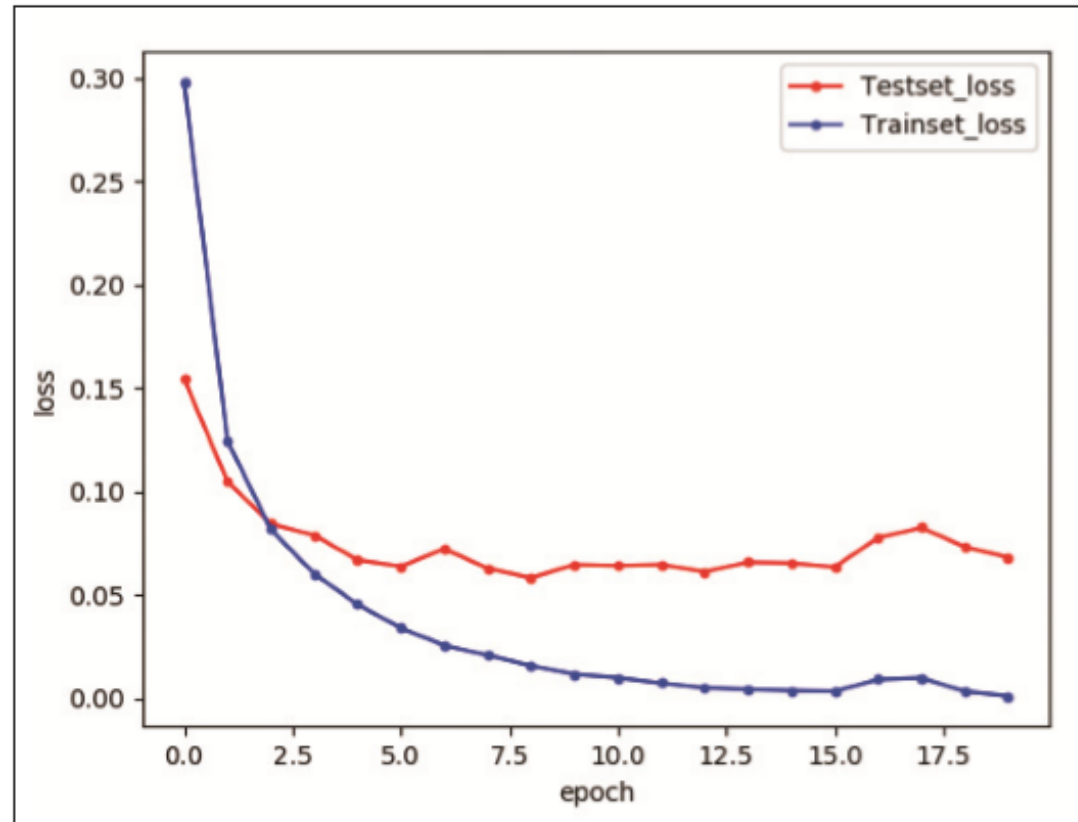
```
X_train = X_train.reshape(X_train.shape[0], 784).astype('float64') / 255
```

## ■ MNIST 손글씨 인식을 DNN으로 해결



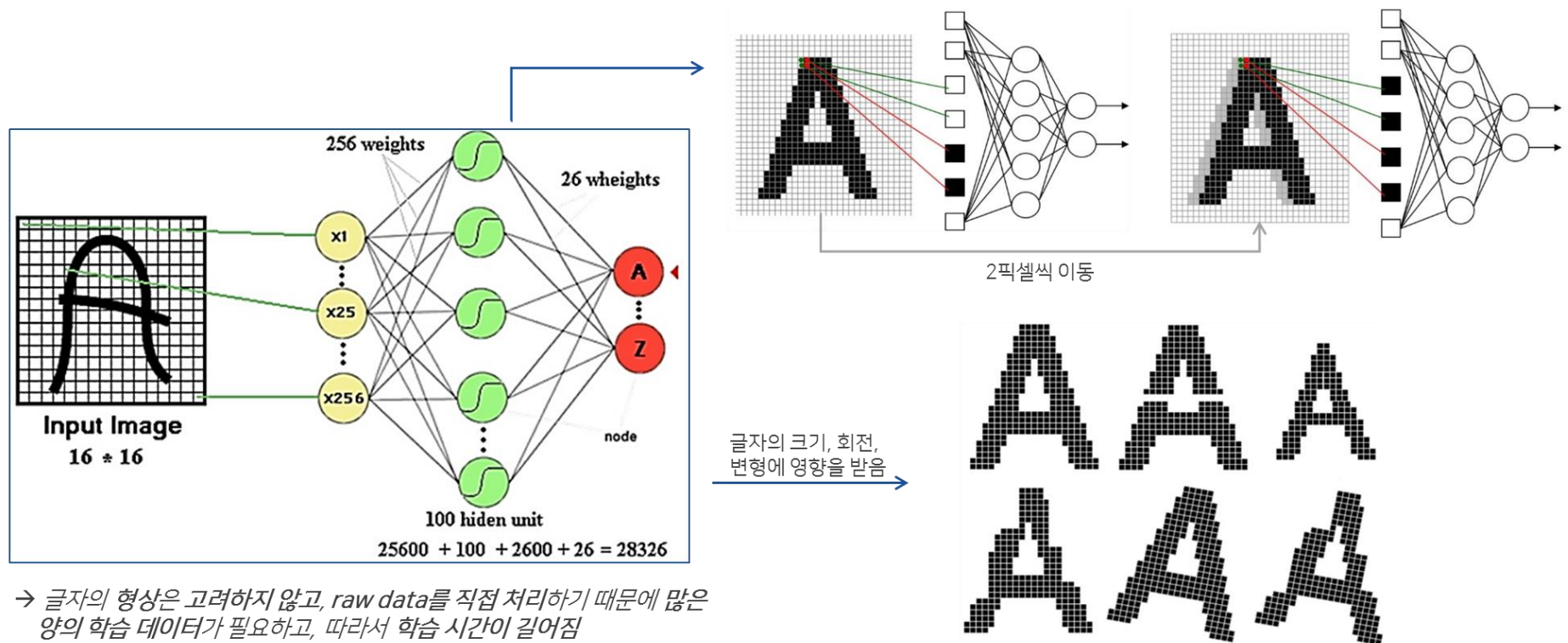
## ■ MNIST 손글씨 인식을 DNN으로 코딩

▶ 01\_MNIST\_딥 러닝.ipynb



## ■ 심층 신경망(DNN)으로 구현했을 때 문제점

- 변수의 개수
- 네트워크의 크기
- 학습 시간
- 글자의 형상은 고려하지 않고, 글자의 크기, 회전, 변형에 취약함



## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ Mask(Filter, Window, Kernel)

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

주어진 이미지

x1	x0
x0	x1

필터

## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

### ❖ Convolution 과정

1x1	0x0	1	0
0x0	1x1	1	0
0	0	1	1
0	0	1	0

$$(1 \times 1) + (0 \times 0) + (0 \times 0) + (1 \times 1) = 2$$

1x1	0x0	1	0
0x0	1x1	1	0
0	0	1	1
0	0	1	0

1	0x1	1x0	0
0	1x0	1x1	0
0	0	1	1
0	0	1	0

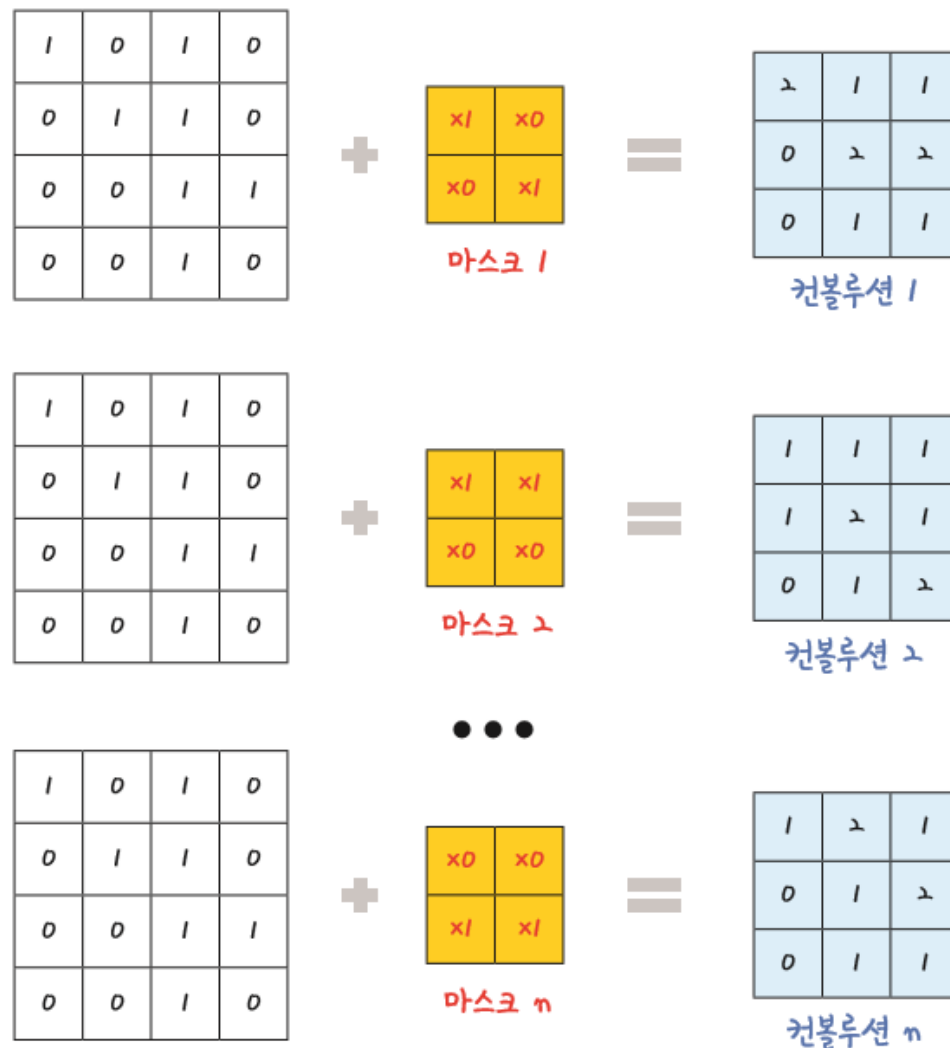
1	0	1x1	0x0
0	1	1x0	0x1
0	0	1	1
0	0	1	0

1	0	1	0
0x1	1x0	1	0
0x0	0x1	1	1
0	0	1	0
1	0	1	0
0	1	1	0
0x1	0x0	1	1
0x0	0x1	1	0
1	0	1	0
0	1	1	0
0	0	1x1	1x0
0	0	1x0	0x1

## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

### ❖ Convolution 과정

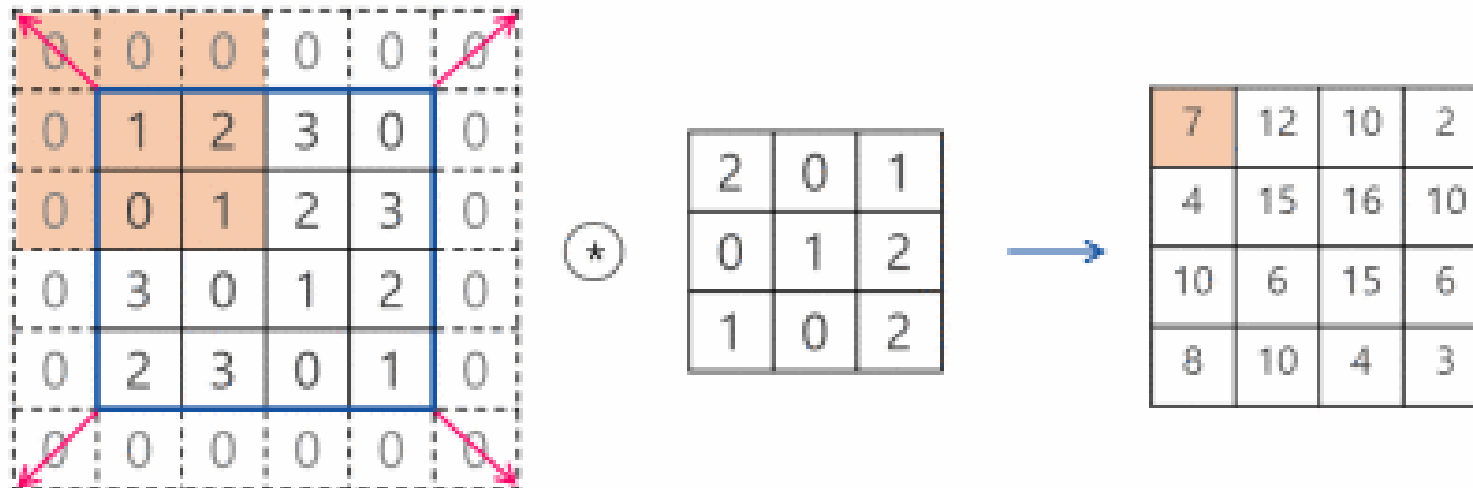
- 컨볼루션을 만들면 입력 데이터로부터 더욱 정교한 특징을 추출할 수 있음
- 이러한 마스크를 여러 개 만들 경우 여러 개의 컨볼루션이 만들어짐 (Feature Map)



## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

### ❖ Padding, Stride

- 패딩: 합성곱 연산을 수행하기 전, 입력데이터 주변을 특정값으로 채워 늘리는 것
- 스트라이드: 입력데이터에 필터를 적용할 때 이동할 간격을 조절하는 것

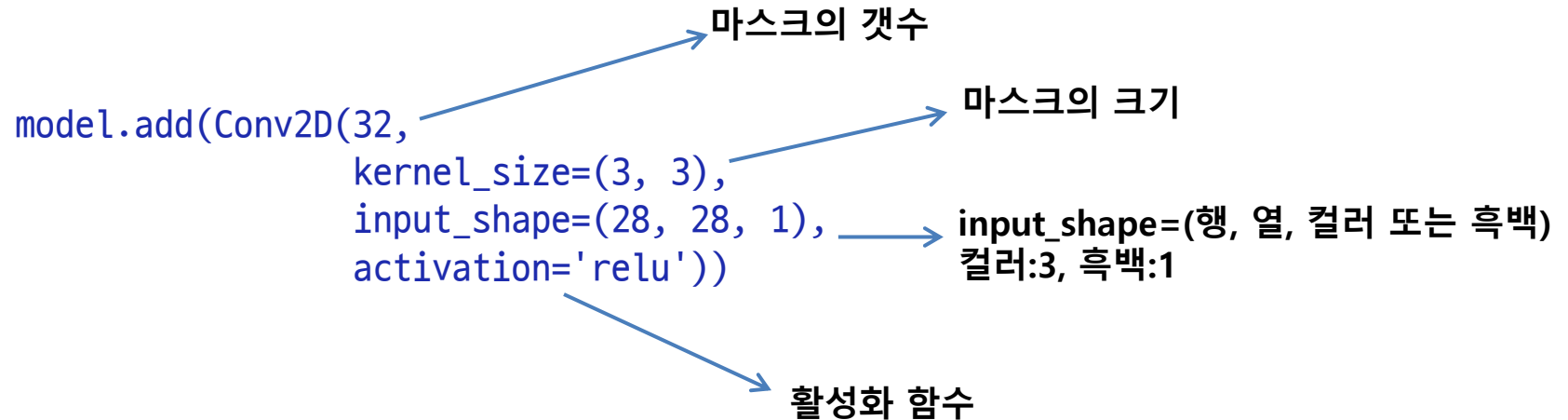




## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

### ❖ Convolution 층 추가

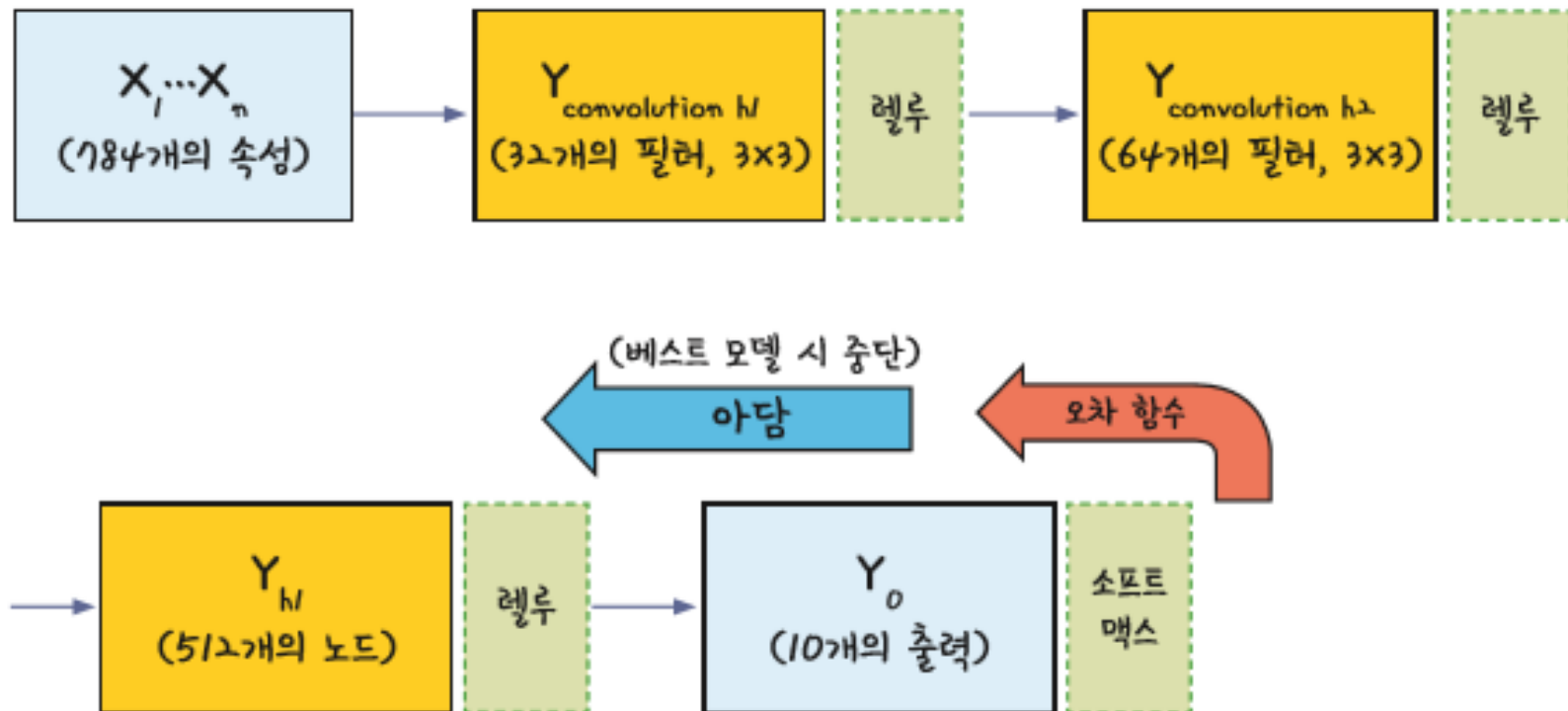
#### ▪ Conv2D()



#### ▪ 컨볼루션 층을 하나 더 추가

```
model.add(Conv2D(64, (3, 3), activation='relu'))
```

## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)



## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

### ❖ 풀링(Pooling)

- Convolution 결과를 축소하는 것
- 풀링 기법 중 가장 많이 사용되는 방법이 맥스 풀링(max pooling)
- 맥스 풀링은 정해진 구역 안에서 가장 큰 값만 다음 층으로 넘기고 나머지는 버림

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

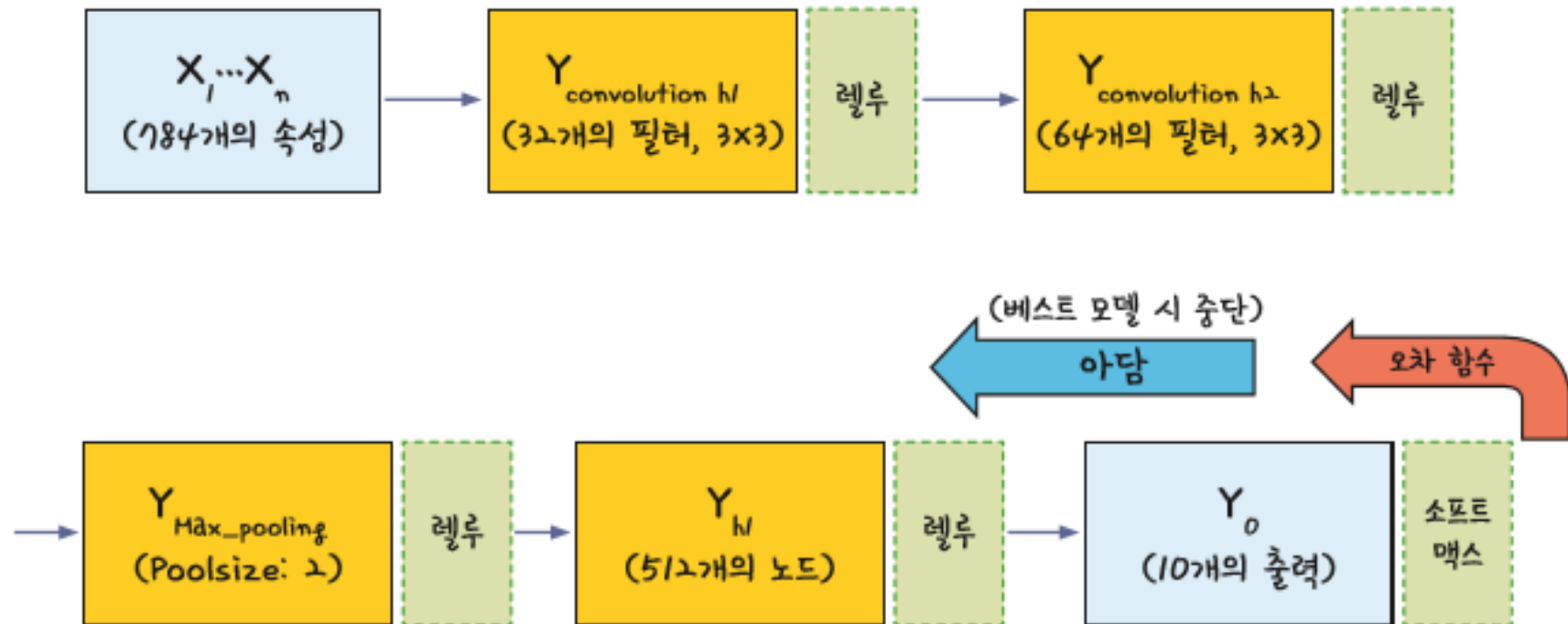
1	0	1	0
0	4	2	0
0	1	6	1
0	0	1	0

4	2
1	6

- 불필요한 정보를 간추림

```
model.add(MaxPooling2D(pool_size=2))
```

## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)



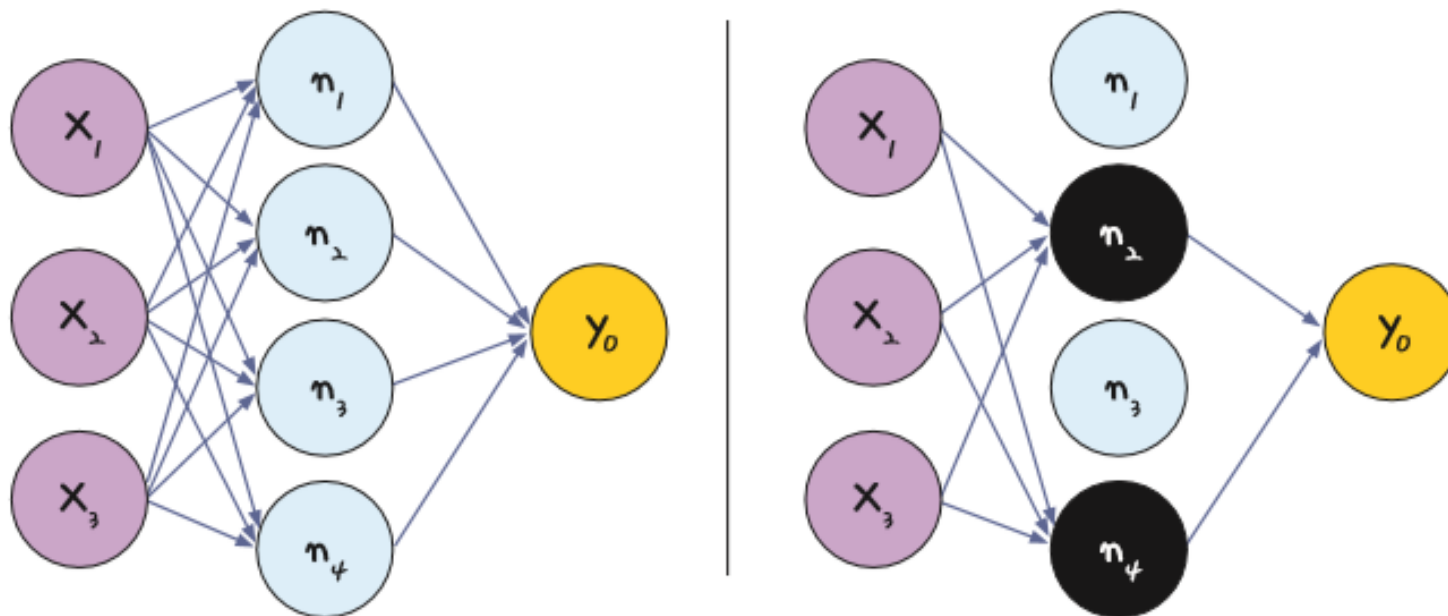
## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 드롭아웃(Drop out) , 플래튼(Flatten)

- 노드가 많아지거나 층이 많아진다고 해서 학습이 무조건 좋아지는 것이 아니다  
→ 과적합 발생
- 과적합을 피하는 간단하지만 효과가 큰 기법이 바로 드롭아웃(drop out) 기법
- 드롭아웃은 은닉층에 배치된 노드 중 일부를 임의로 꺼주는 것

## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 드롭아웃(Drop out) , 플래튼(Flatten)



- 25%의 노드를 끄려면 다음과 같이 코드를 작성  
`model.out(Dropout(0.25))`

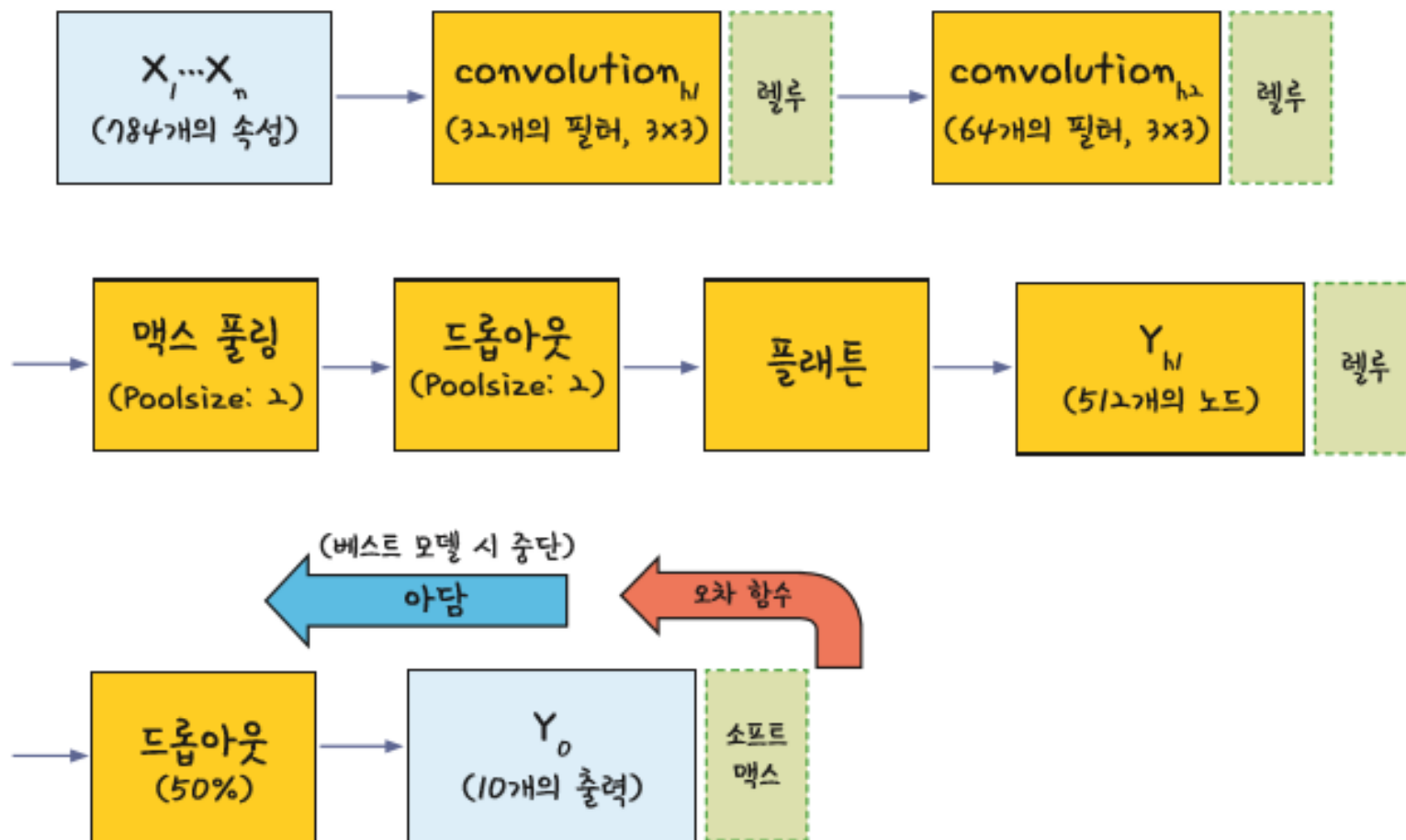
### ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)

❖ 드롭아웃(Drop out) , 플래튼(Flatten)

- 컨볼루션, 맥스풀링, 드롭아웃 층을 거친 후 기본 층에 연결
  - 컨볼루션, 맥스풀링: 2차원
  - 기본 층: 1차원
- 2차원 → 1차원 변환

```
model.add(Flatten())
```

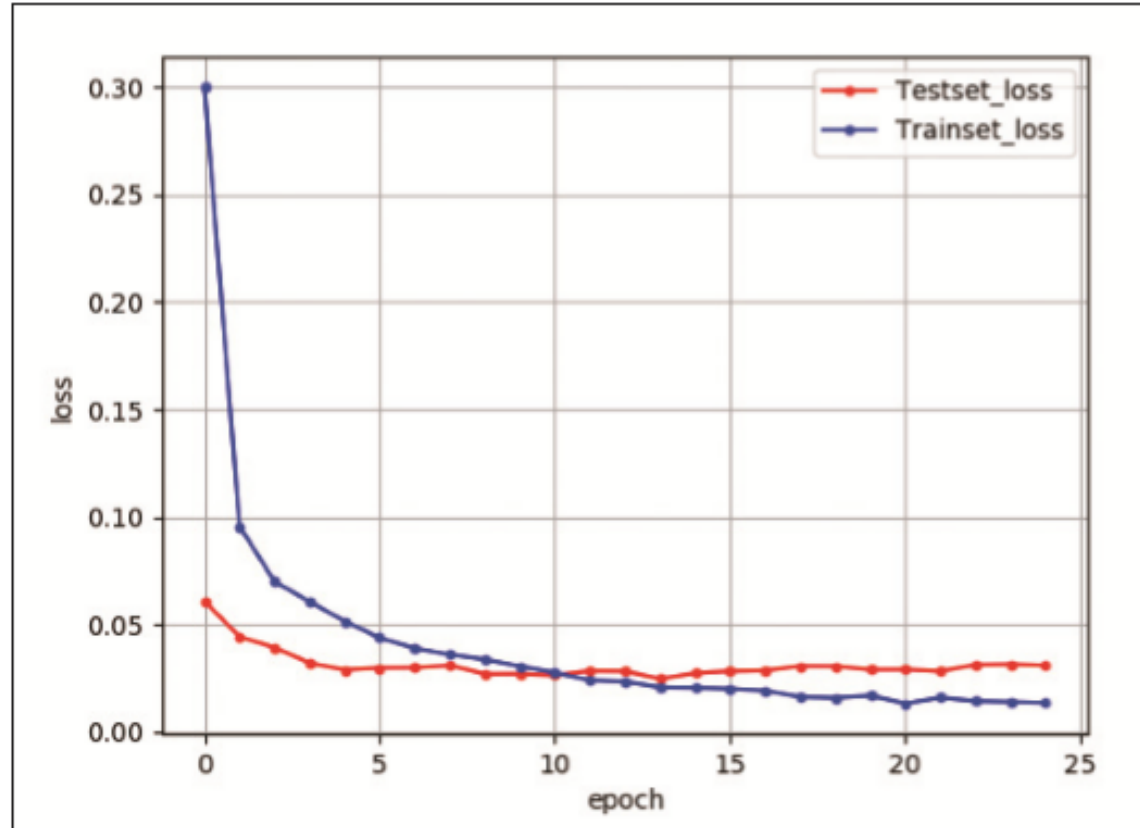
## ■ 컨볼루션 신경망(Convolutional Neural Network, CNN)





## ■ 코딩으로 확인하는 MNIST 이미지 인식

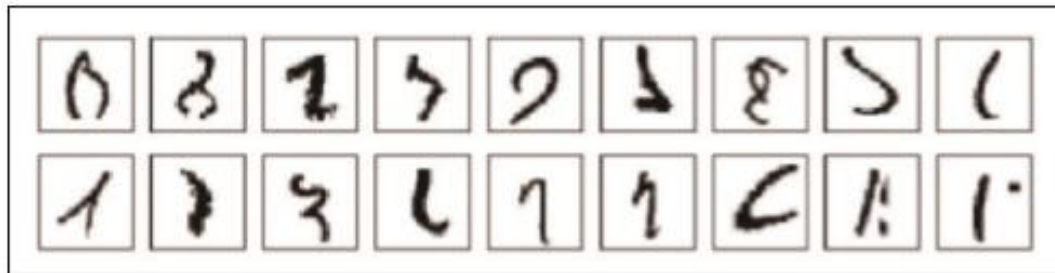
▶ 11\_MNIST\_CNN.ipynb



## ■ 코딩으로 확인하는 MNIST 이미지 인식

### ❖ 결과 리뷰

- 0.9934, 즉 99.34%의 정확도
- 심층 신경망 코드에서는 정확도가 98.39%
- 100% 다 맞이지 못한 이유는 데이터 안에 다음과 같이 확인할 수 없는 글씨가 들어있었기 때문



## ■ 개, 고양이 구분

### ❖ [Kaggle site](#)

- 2010년 설립된 빅데이터 솔루션 대회 플랫폼 회사
- 2017년 Google이 인수 ([ZDNet 기사](#))
- 기업 및 단체에서 Prize를 걸고 데이터와 해결 과제를 등록하면, 데이터 사이언티스트들이 이를 해결하기 위해 모델을 개발하고 경쟁하게 되는 시스템

### ❖ [Kaggle 구성 요소](#)

- Overview: 문제에 대한 간략한 소개와 문제 정의
- Dataset: 예측 모델을 만들기 위해 필요한 데이터셋 및 field에 대한 설명
- Kernels: 다른 사람들이 어떤 모델을 써서 구현을 했는지 힌트를 얻을 수 있고, 또한 내가 구현한 모델이 과연 올바른지에 관해서 코멘트를 주고받을 수 있음
- Discussion: 게시판 역할
- Leaderboard: 모델 예측 정확도 랭킹

## ■ 개, 고양이 구분

### ❖ 데이터 셋

- 훈련 셋: 개, 고양이 사진 각각 12,500개, 총 25,000개
- 테스트 셋: 개, 고양이 사진 합쳐서 12,500개

```
datasets
├── dogs-vs-cats
│   ├── train
│   │   ├── cat.0.jpg
│   │   ├── ...
│   │   ├── cat.12499.jpg
│   │   ├── dog.0.jpg
│   │   ├── ...
│   │   └── dog.12499.jpg
│   └── test1
│       ├── 1.jpg
│       ├── ...
│       └── 12500.jpg
```

#### ■ 개, 고양이 구분

##### ❖ 데이터 셋

- 훈련 셋: 개, 고양이 사진 각각 12,500개, 총 25,000개
- 테스트 셋: 개, 고양이 사진 합쳐서 12,500개

▶ [21\\_Dogs-vs-Cats-partial.ipynb](#)