

---

배워서 바로 쓰는

---

# DNN, CNN, RNN을 활용한 딥러닝 실무

---

# NOTICE

- 본 자료는 저작권법에 의거해 허가 받지 않은 복사, 전재, 편집, 재배포 등을 금지함을 알려 드립니다.
- 본 자료는 온/오프라인 또는 동영상 강의를 전제로 제작되었습니다.  
강사의 부가 설명 없이 본 자료의 내용을 **임의 해석할 경우 잘못된 결론**에 이를 수 있음을 유념하십시오.
- 강의를 캡처, 녹음, 녹화하는 등의 콘텐츠의 원천 제공 방식 이외의 **저장 행위를 엄격히 금지**하오니  
필요하신 내용은 수업 틈틈이 개별적으로 메모하시기 바랍니다.

## 딥러닝 개발 과정

- 딥러닝 개발 과정

## 딥러닝 알고리즘 종류

- 딥러닝 알고리즘 종류 - DNN, CNN, RNN

## 딥러닝을 이용한 회귀와 분류

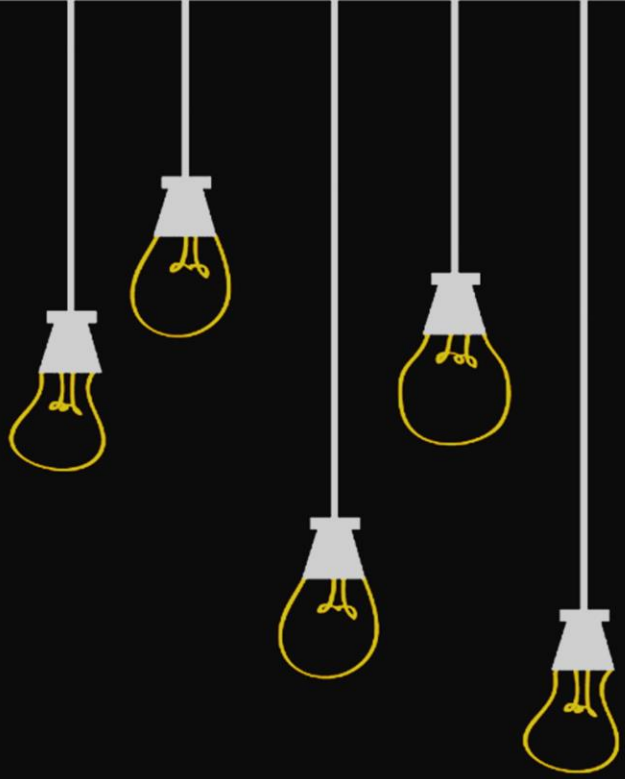
- 딥러닝을 이용한 분류
- 딥러닝을 이용한 회귀
- 딥러닝(DNN)을 이용한 MNIST 손글씨 인식하기

## CNN(합성곱 신경망)

- CNN(합성곱 신경망) 개념
- CNN(합성곱 신경망) 구성 요소 및 연산
- CNN을 이용한 MNIST 손글씨 분류하기

## RNN(순환 신경망), LSTM

- RNN 개념, LSTM 개념
- RNN을 이용한 영화리뷰 감성 분석
- LSTM을 이용한 뉴스 카테고리 분류



Actionable Content  
**MASO CAMPUS**

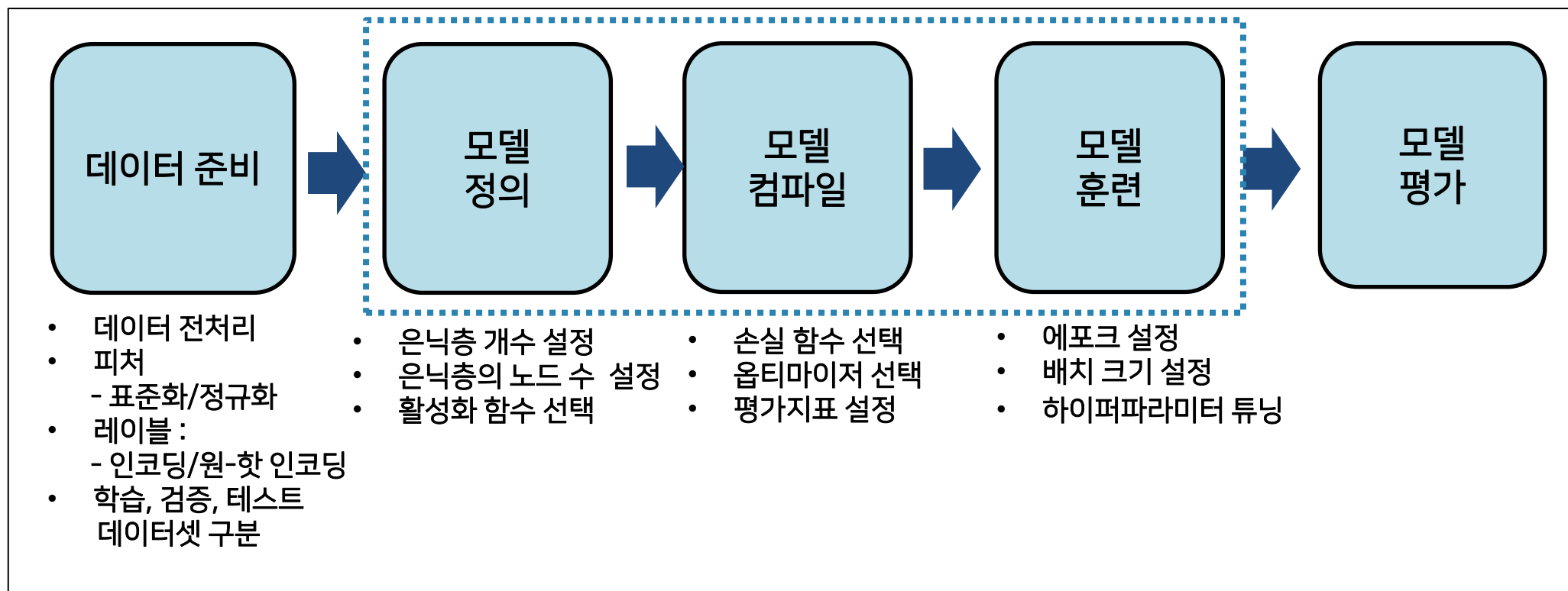
# 딥러닝 개발 과정



# 딥러닝 개발 과정

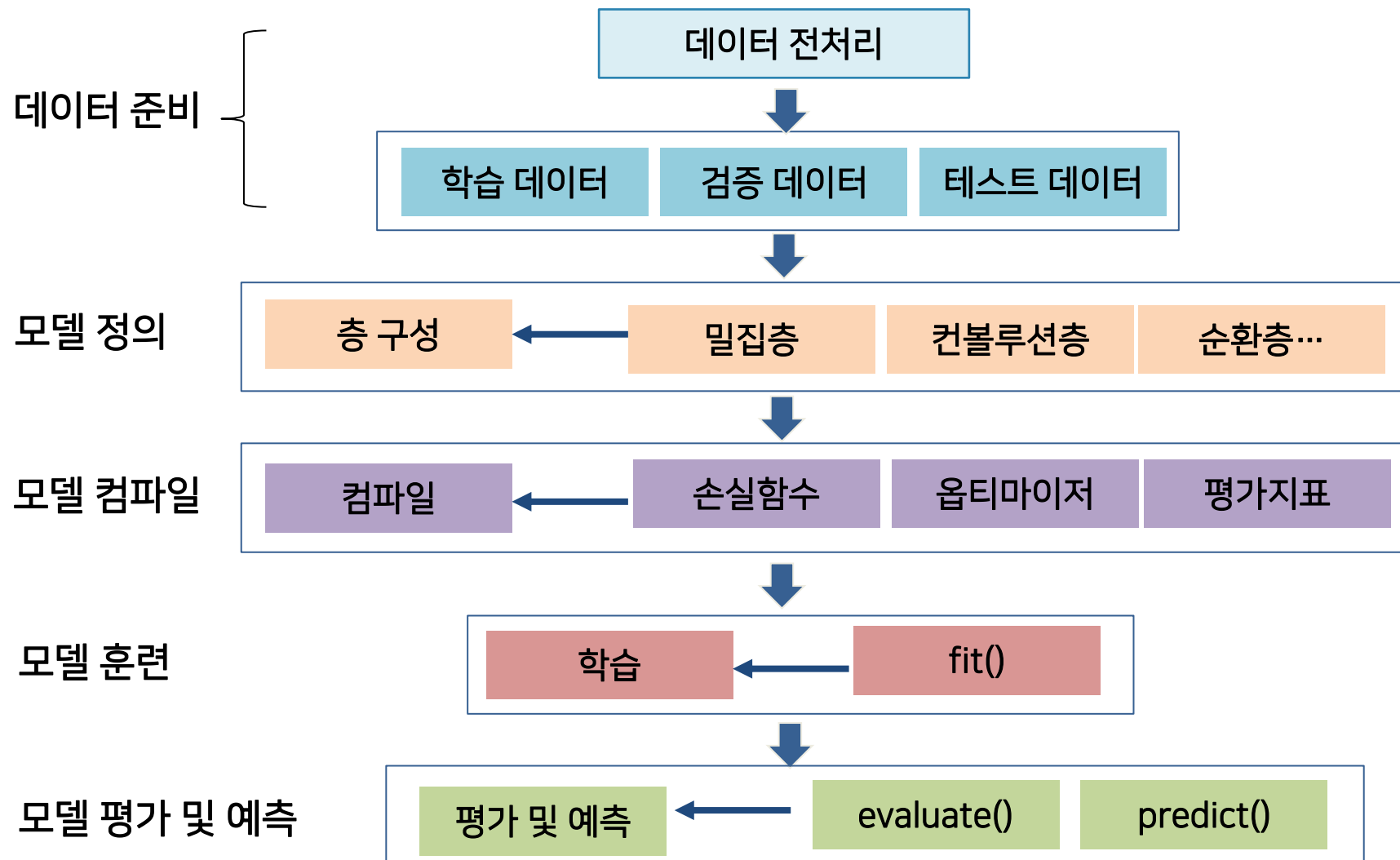


## ■ 딥러닝(Deep Learning) 개발 과정





## ■ 딥러닝(Deep Learning) 개발 과정







## ■ 딥러닝(Deep Learning) 개발 과정

### 1. 데이터 준비

- 데이터 전처리 - 피처와 레이블 구분, 피처(표준화/정규화), 레이블(인코딩/원-핫인코딩)
- 학습데이터, 검증데이터, 테스트데이터로 구분

### 2. 모델 정의 → 층(입력층, 은닉층, 출력층) 구성, 활성화함수

- `model = Sequential()`
- `model.add(Dense(32, input_shape = (2, ), activation = 'relu'))` # (입력층 + 은닉층), 활성화 함수
- `model.add(Dense(1, activation = 'sigmoid'))` # 출력층, 활성화 함수

### 3. 모델 컴파일 → 옵티마이저, 손실함수, 평가지표 설정

- 회귀문제 - 평균제곱오차(MSE)  
`model.compile(optimizer= Adam(), loss='mse' )`
- 이항 분류 문제(크로스 엔트로피)  
`model.compile(optimizer= Adam(), loss='binary_crossentropy', metrics=['acc'])`
- 다항 분류 문제(크로스 엔트로피)  
`model.compile(optimizer= Adam(), loss='categorical_crossentropy', metrics=['acc'])`

### 4. 모델 훈련

- `model.fit(data, label, epochs = 100)`
- `model.fit(data, label, epochs = 100, validation_data=(val_data, val_label))`

### 5. 모델 평가 및 예측

- `model.evaluate(data, label)`
- `model.predict(data)`



## ■ 데이터 준비

### 1) 데이터 전처리

- 피처(X)와 레이블(y) 구분
- 피처 스케일링(Feature Scaling)
  - 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업, 표준화, 정규화가 방법이 있음
  - 표준화(Standardization)
    - 값의 분포를 평균이 0이고 분산이 1인 가우시안 정규분포를 가진 값으로 변환, 표준화 =  $\frac{x_i - \text{mean}(x)}{\text{std}(x)}$
  - 정규화(Normalization)
    - 서로 다른 피처의 크기를 통일하기 위해 크기를 변환해 주는 개념, 정규화 =  $\frac{x_i - \min(x)}{\max(x) - \min(x)}$
- 레이블 - 레이블 인코딩, 원-핫 인코딩
  - 레이블 인코딩 - 문자열을 숫자로 변환하는 인코딩
  - 원-핫 인코딩(One-Hot Encoding)
    - 하나의 클래스만 1이고 나머지 클래스는 전부 0인 인코딩을 의미
    - 다중분류의 레이블인 경우는 반드시 원-핫 인코딩 형태로 변환해야 함

### 2) 학습 데이터셋, 검증 데이터셋, 테스트 데이터셋으로 구분

Train Dataset		Validation Dataset		Test Dataset	
Train Data	Train Label	Val Data	Val Label	Test Data	Test Label



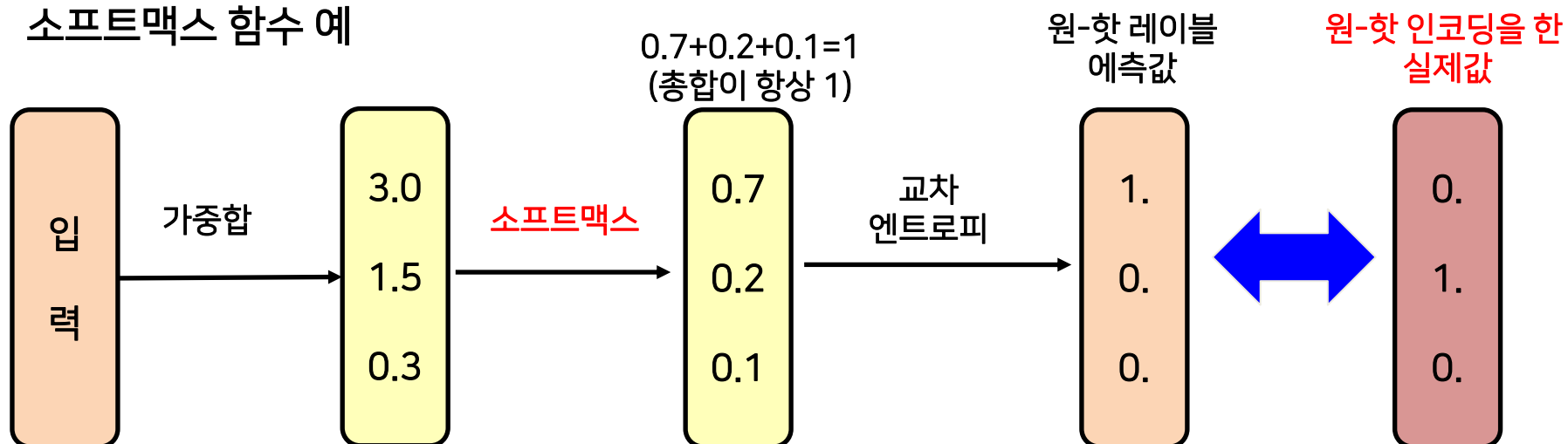
## ■ 데이터 전처리

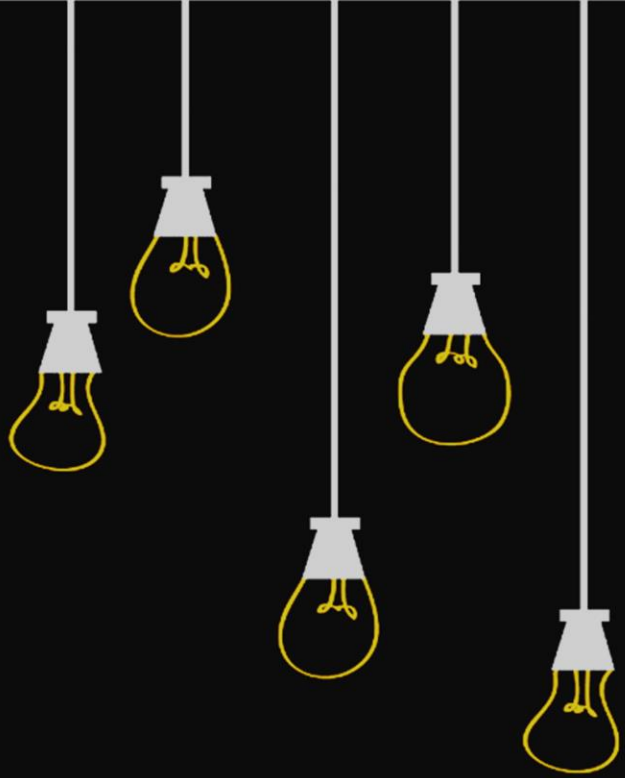
### ■ 다중분류에서 레이블을 원-핫 인코딩을 하는 이유

#### ■ 소프트맥스(softmax) 활성화 함수

- 소프트맥스는 총합이 1인 형태로 바꿔서 계산해 주는 활성화 함수
- 합계가 1인 형태로 변환하면 큰 값이 두드러지게 나타나고 작은 값이 더 작아짐
- 이 값이 교차 엔트로피를 지나  $[1., 0., 0.]$ 으로 변화하게 되면 우리가 원하는 원-핫 인코딩 값, 즉, 하나만 1이고 나머지는 모두 0인 형태로 전환시킬 수 있음

#### ■ 소프트맥스 함수 예





Actionable Content  
**MASO CAMPUS**

# 딥러닝 알고리즘 종류



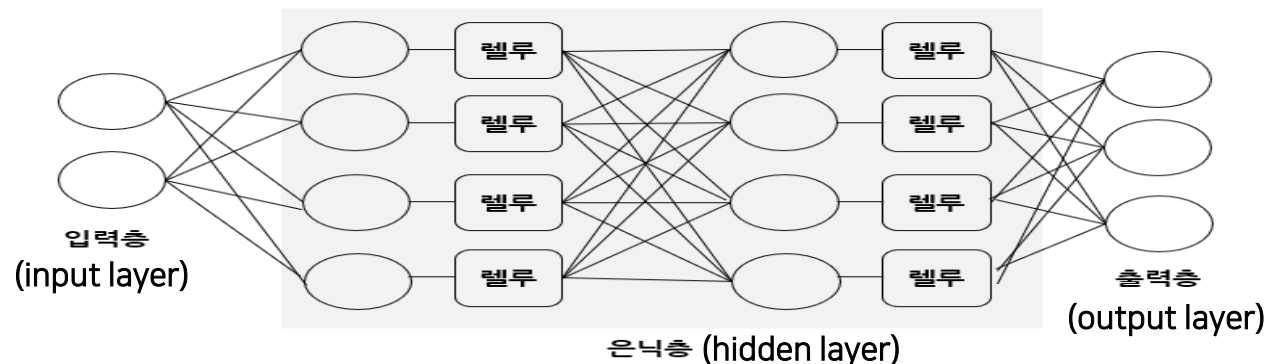
# 딥러닝 알고리즘 종류



## ■ 딥러닝 알고리즘 종류 : 심층 신경망(DNN), 합성곱 신경망(CNN), 순환 신경망(RNN)

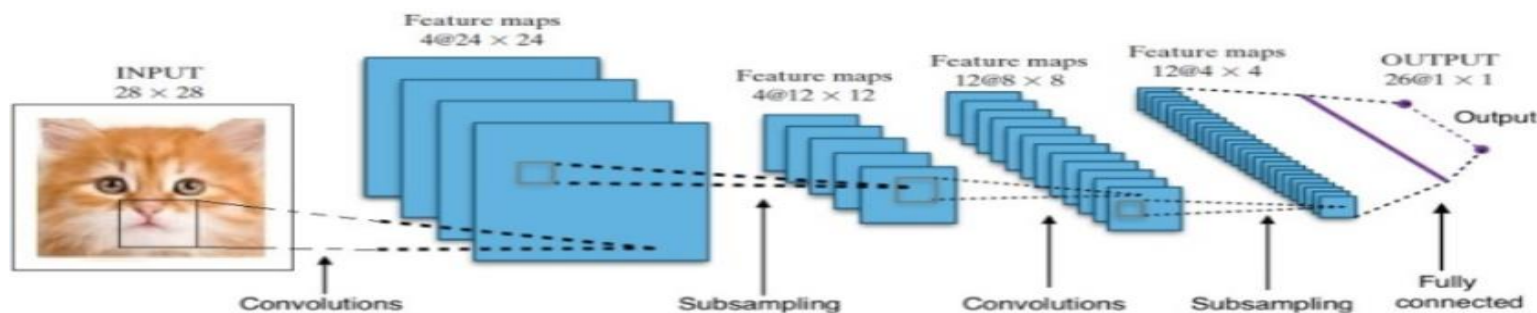
### 1) 심층 신경망(DNN, Deep Neural Network)

- 입력층과 출력층 사이에 다수의 은닉층을 포함하는 인공 신경망



### 2) 합성곱 신경망(CNN, Convolutional Neural Network)

- 합성곱과 풀링층을 포함하는 이미지 처리 성능이 좋은 인공 신경망 알고리즘
- 영상 및 사진이 포함된 이미지 데이터에서 객체를 탐지하거나 객체 위치를 찾아내는데 유용한 신경망





## ■ 딥러닝 알고리즘 - 심층 신경망(DNN), 합성곱 신경망(CNN), 순환 신경망(RNN)

### 3) 순환 신경망(RNN, Recurrent Neural Network)

- 시계열(음악, 영상) 같은 시간 흐름에 따라 변환하는 데이터를 학습하기 위한 인공 신경망
- 순환(Recurrent)은 자기 자신을 참조한다는 것으로, 현재 결과가 이전 결과와 연관이 있다는 의미

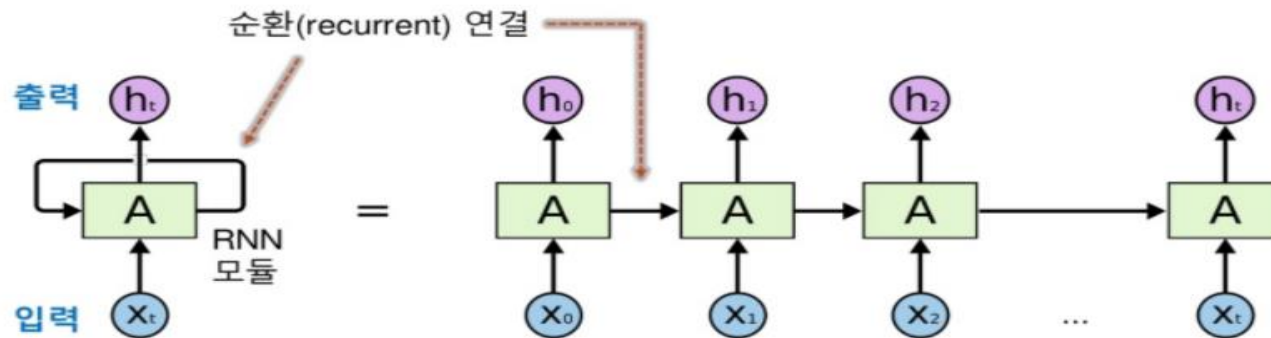


그림 출처 : <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>

### 4) LSTM(Long-Short Term Memory)

- 순환신경망(RNN)의 기울기 소실 문제로 학습이 제대로 되지 않는 문제를 해결하고자 메모리 개념을 도입한 순환 신경망

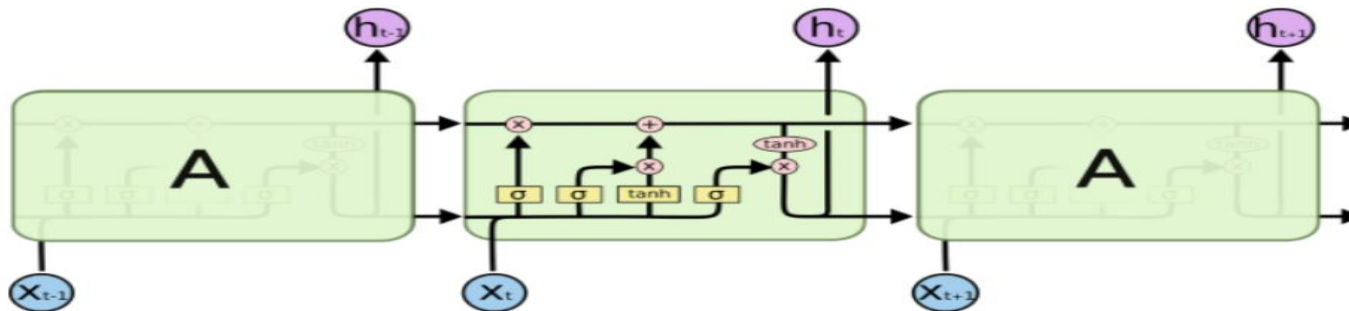
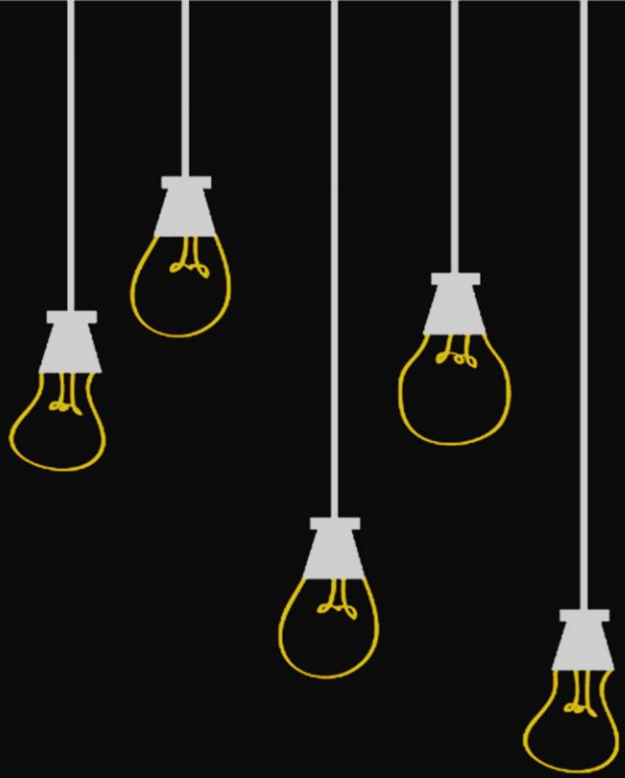


그림 출처 :

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=apr407&logNo=221237917815>



Actionable Content  
**MASO CAMPUS**

# 딥러닝을 이용한 회귀와 분류





# 딥러닝을 이용한 분류



## ■ 아이리스(iris) 다중 품종 예측 : 다중 분류 문제

### ■ 아이리스 품종 예측

- Iris 데이터 셋

**iris setosa**



petal sepal

**iris versicolor**



petal sepal

**iris virginica**



petal sepal

- 아이리스는 꽃잎의 모양과 길이에 여러가지 품종으로 나뉨
- 딥러닝을 사용하여 아이리스의 품종을 구별해 낼 수 있을까?
- 아이리스 품종이 3개의 품종으로 구분되어야 하므로 이항 분류가 아닌 다중 분류를 사용

그림 출처 : <https://velog.io/@sp1rit/Iris%EC%9D%98-%EC%84%B8%EA%B0%80%EC%A7%80-%ED%92%88%EC%A2%85-%EB%B6%84%EB%A5%98%ED%95%98%EA%B8%B0>



## ■ 아이리스(iris) 다중 품종 예측 : 다중 분류 문제

### ■ 아이리스(iris) 데이터 셋

	4개 속성				레이블(정답)
	정보1	정보2	정보3	정보4	품종
0번째 아이리스	5.1	3.5	4.0	0.2	iris-setosa
1번째 아이리스	4.9	3.0	1.4	0.2	iris-setosa
2번째 아이리스	4.7	3.2	1.3	0.3	iris-setosa
...	...	...	...	...	...
149번째 아이리스	5.9	3.0	5.1	1.8	iris-viginica

- 샘플 수 : 150
- 속성 수 : 4
  - 정보 1 : 꽃받침 길이(sepal length)    - 정보 2 : 꽃받침 넓이(sepal width)
  - 정보 3 : 꽃잎 길이(petal length)       - 정보 4 : 꽃잎 넓이(petal width)
- 클래스(정답) : 꽃의 품종(iris-setosa, iris-versicolor, iris-viginica)

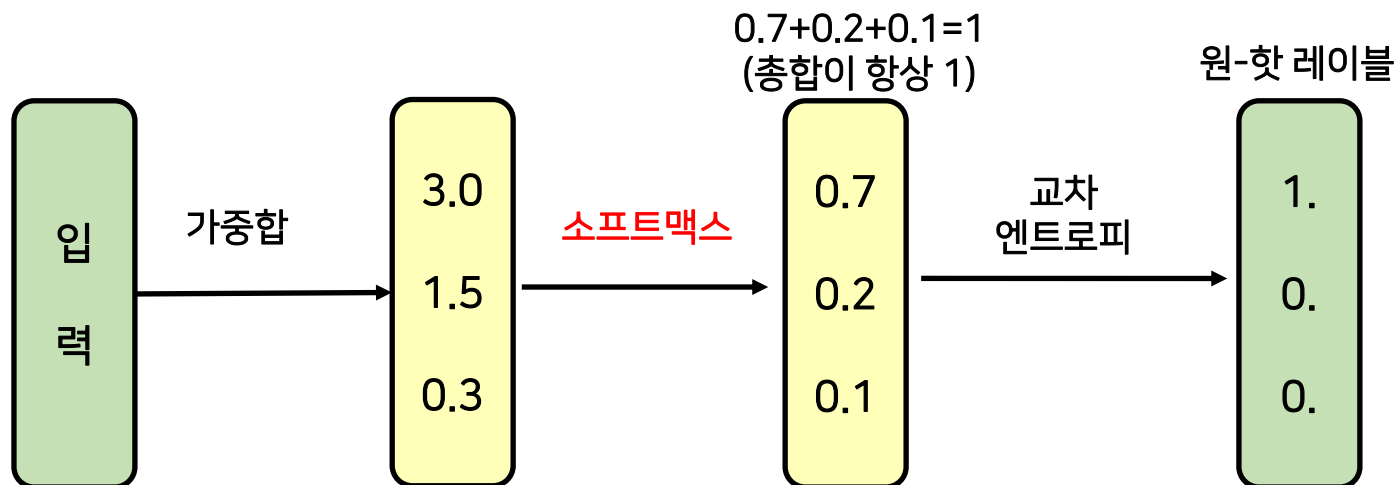


## ■ 아이리스(iris) 다중 품종 예측 : 다중 분류 문제

### ■ 소프트맥스(softmax) 활성화 함수

- 소프트맥스는 총합이 1인 형태로 바꿔서 계산해 주는 활성화 함수
- 합계가 1인 형태로 변환하면 큰 값이 두드러지게 나타나고 작은 값이 더 작아짐
- 이 값이 교차 엔트로피를 지나  $[1., 0., 0.]$ 으로 변화하게 되면 우리가 원하는 원-핫 인코딩 값, 즉, 하나만 1이고 나머지는 모두 0인 형태로 전환시킬 수 있음

### ■ 소프트맥스 함수의 원리





## ■ 아이리스(iris) 다중 품종 예측 실습 : 다중 분류 문제

- 데이터셋을 구분 : 학습 데이터셋, 검증 데이터셋, 테스트 데이터셋)하여 딥러닝 실습

참고) 원-핫 인코딩(One-Hot Encoding)

- 하나의 클래스만 1이고 나머지 클래스는 전부 0인 인코딩을 의미
- 범주형 데이터에 주로 사용

품종	iris-setosa	iris-versicolor	iris-viginica
iris-setosa	1	0	0
iris-versicolor	0	1	0
iris-viginica	0	0	1



# 딥러닝을 이용한 회귀



## ■ 딥러닝을 이용한 보스톤 집값 예측 : 다중선형회귀

### ■ 보스톤 주택 가격 예측

- 1978년, 집값에 가장 큰 영향을 미치는 것이 '깨끗한 공기'라는 연구 결과가 하버드대학교 도시개발학과에서 발표됨
- 이들은 자신의 주장을 뒷받침하기 위해 집값의 변동에 영향을 미치는 여러 가지 요인을 모아서 환경과 집값의 변동을 보여주는 데이터셋을 만듦
- 이 데이터가 현재 선형 회귀를 테스트하는 가장 유명한 데이터로 쓰이고 있음

### ■ 선형 회귀(Linear Regression)

- 레이블(클래스)가 수치형 자료일 때 사용하는 머신러닝으로 수치를 예측하는 문제
- 단순 선형 회귀 : 레이블을 설명하는 설명변수(피쳐)가 하나만 있는 경우,  $y = wx + b$
- 다중 선형 회귀 : 레이블을 설명하는 설명변수(피쳐)가 두 개 이상인 경우,  $y = w_1x_1 + w_2x_2 + \dots + b$
- 예) 보스톤 주택가격 데이터
  - 주어진 환경 요인과 집값의 변동을 학습해서 새로운 환경요인에 대한 집값을 예측하는 것



## ■ 딥러닝을 이용한 보스턴 집값 예측 : 다중선형회귀

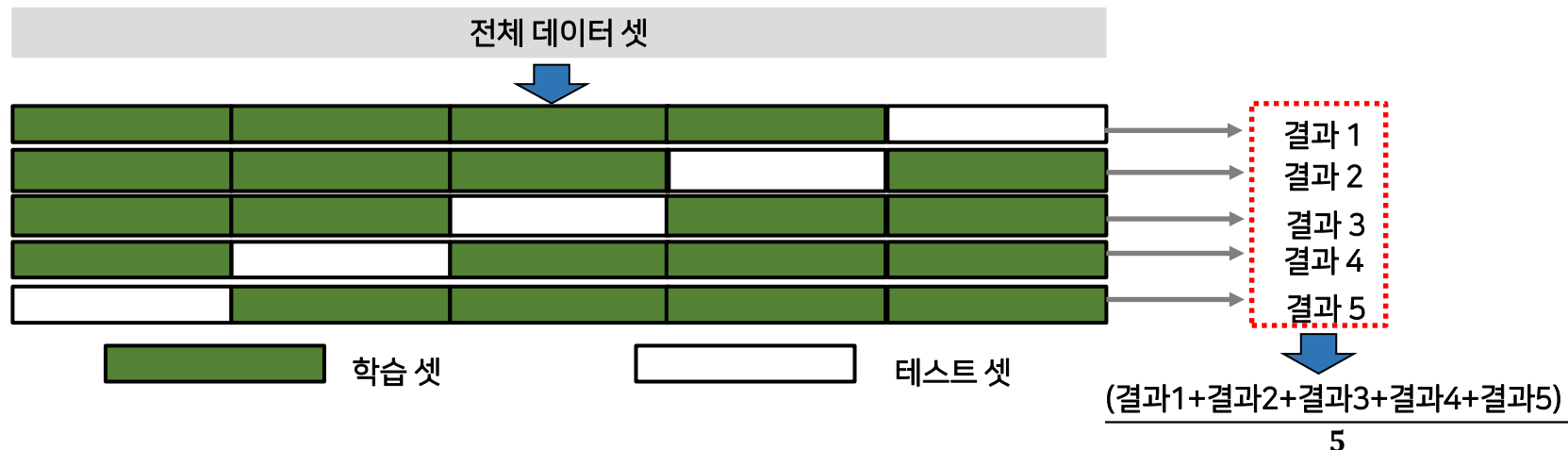
### 참고) 데이터 표준화

- 피쳐 스케일링(Feature Scaling)
  - 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업, 표준화, 정규화가 방법이 있음
- 표준화(Standardization)
  - 값의 분포를 평균이 0이고 분산이 1인 가우시안 정규분포를 가진 값으로 변환

$$\frac{x_i - \text{mean}(x)}{\text{std}(x)}$$

### 참고) 교차 검증(Cross Validation)

- 데이터의 약 70%를 학습 데이터 셋으로 쓰면, 테스트 데이터 셋은 겨우 전체 데이터의 30% 사용  
→ 단 한번의 테스트만으로 실제로 얼마나 잘 작동하는지 확신하기는 쉽지 않음  
→ 여러 번 학습과 테스트를 하는 방법 : K-겹 교차 검증(K-fold Cross Validation)







# 딥러닝(DNN)을 이용한 MNIST 손글씨 인식하기



### ■ 딥러닝(DNN)을 이용한 MNIST 손글씨 인식하기

- MNIST 손글씨 인식 데이터셋



- MNIST 데이터셋은 미국 국립표준기술원(NIST)이 고등학생과 인구조사국 직원 등이 쓴 손 글씨를 이용해 만든 데이터로 구성
- 70,000개의 28\*28 글자 이미지에 각각 0부터 9까지 이름표를 붙인 데이터
- 70,000개 = 60,000개의 학습 데이터셋 + 10,000개의 테스트 데이터셋으로 구성




### ■ 딥러닝(DNN)을 이용한 MNIST 손글씨 인식하기

- 웹에서 MNIST 테스트 해보기 : <http://myselfph.de/neuralNet.html>

myselfph.de Rigid Body Game Physics ▾ Hodgkin-Huxley Neural Net for MNIST

## Neural Net for Handwritten Digit Recognition in JavaScript

Draw a digit in the box below and click the "recognize" button.

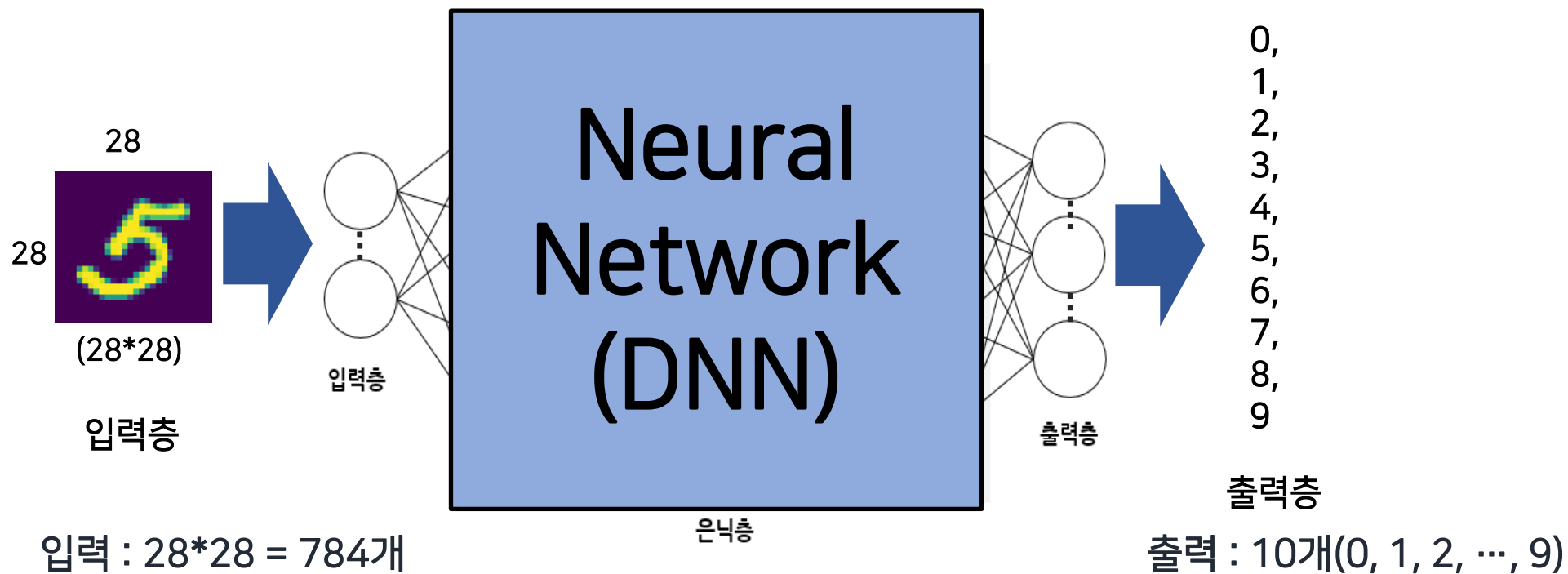


☐ Display Preprocessing

☒ Scale Stroke Width



### ■ 딥러닝(DNN)을 이용한 MNIST 손글씨 인식하기





### ■ 딥러닝(DNN)을 이용한 MNIST 손글씨 인식하기

#### 참고) 데이터 표준화

- 피쳐 스케일링(Feature Scaling)
  - 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업, 표준화, 정규화가 방법이 있음
- 표준화(Standardization)
  - 서로 다른 피쳐의 크기를 통일하기 위해 크기를 변환해 주는 개념

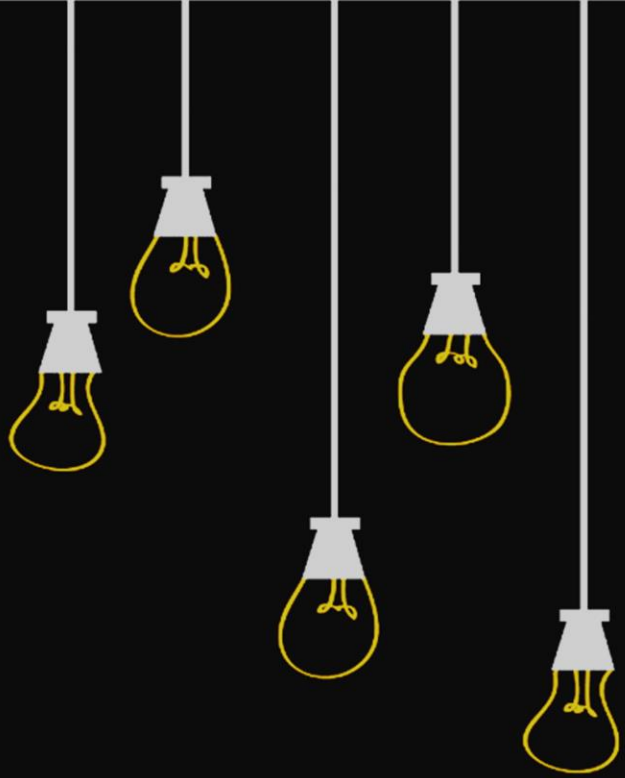
$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- 예) 피쳐 A 거리 0~100km, 피쳐 B 금액 0~100,000,000원 -> 정규화 -> 최소 0 ~ 최대 1로 변환

#### 참고) 원-핫 인코딩(One-Hot Encoding)

- 하나의 클래스만 1이고 나머지 클래스는 전부 0인 인코딩을 의미

color	color red	color blue	color green
red	1	0	0
green	0	0	1
blue	0	1	0



Actionable Content  
**MASO CAMPUS**

# 합성곱 신경망(CNN)



# 합성곱 신경망

## (CNN, Convolutional Neural Network)



## ■ 다층 퍼셉트론(Multi Layer Perceptron) 한계

- 심층 신경망의 계산 과정은 복잡하고 많은 자원(CPU 또는 GPU, 메모리)을 요구, 계산 시간도 오래 걸림
- 같은 이미지라도 이미지 내 물체의 위치가 조금만 달라져도 동일한 이미지에 대해 전혀 다른 입력값으로 인식하여 매번 연산을 수행하는 비효율적인 문제점
- 다층 퍼셉트론의 한계 : 이미지 학습
- 이미지 학습 알고리즘 도입 → CNN(Convolutional Neural Network, 합성곱)

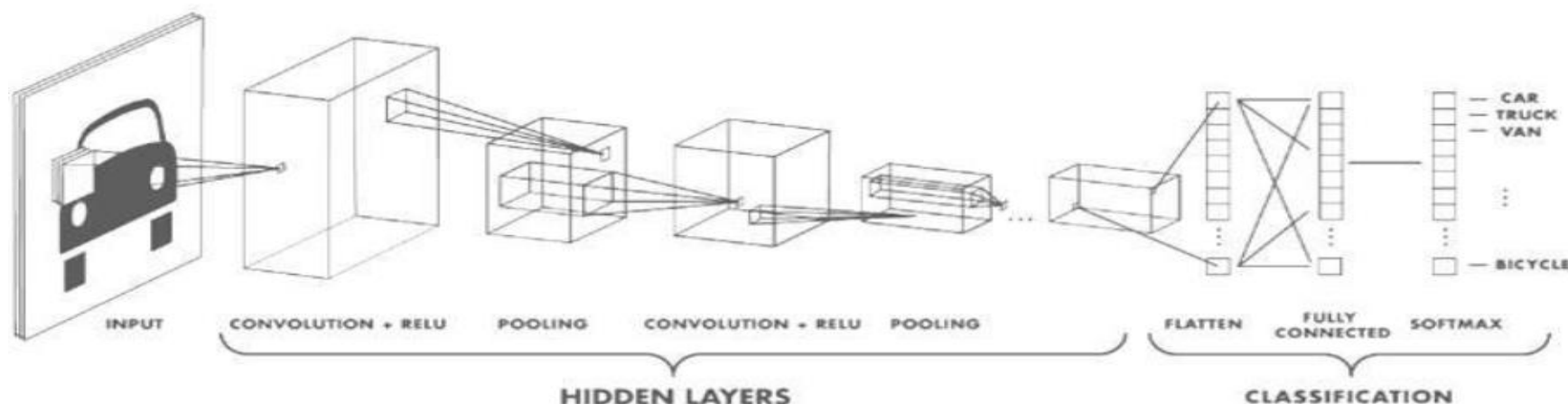


그림 출처 : <https://kr.mathworks.com/solutions/deep-learning/convolutional-neural-network.html>

## ■ CNN(Convolutional Neural Network, 합성곱)

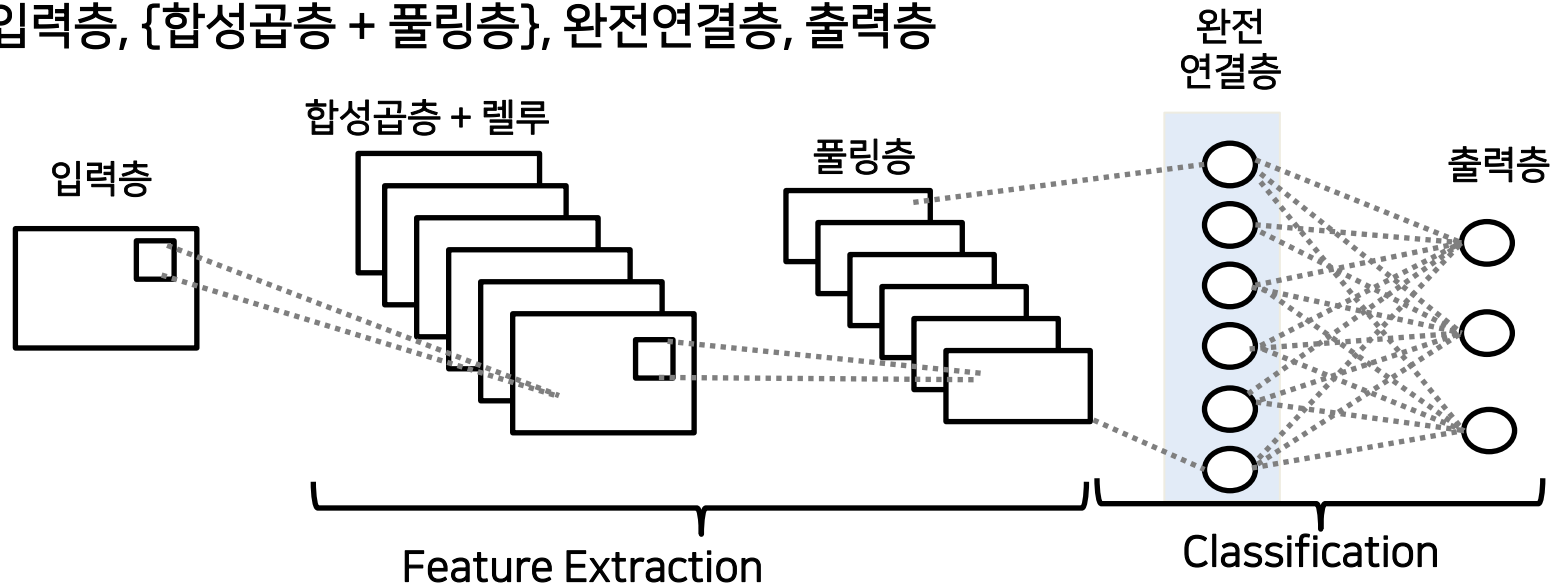
- 이미지 인식에 강력한 성능을 발휘하는 알고리즘
- 이미지나 영상을 입력으로 받고 특징을 추출하여 분류하는 모델
- 합성곱 신경망은 이미지 전체를 한 번에 계산하는 것이 아니라 이미지의 국소적 부분을 계산함으로써 시간과 자원을 절약하여 이미지의 세밀한 부분까지 분석할 수 있는 신경망
- 입력된 이미지에서 특징을 추출하기 위해 필터(마스크, 윈도우, 커널)를 도입하는 기법





## ■ 합성곱 신경망(CNN, 컨볼루션 신경망)

- 입력층, {합성곱층 + 풀링층}, 완전연결층, 출력층



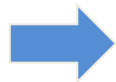
CNN = {컨볼루션층(Convolution layer) + 풀링층(Pooling layer)} + 완전연결층(Fully-Connected layer)

Feature Extraction(데이터 특징 추출)

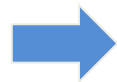
Classification(데이터 분류)

CNN = {Convolution layer, Pooling layer 반복} → Fully-Connected layer → Softmax

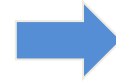
데이터 입력



Conv2D  
MaxPool2D



Dense

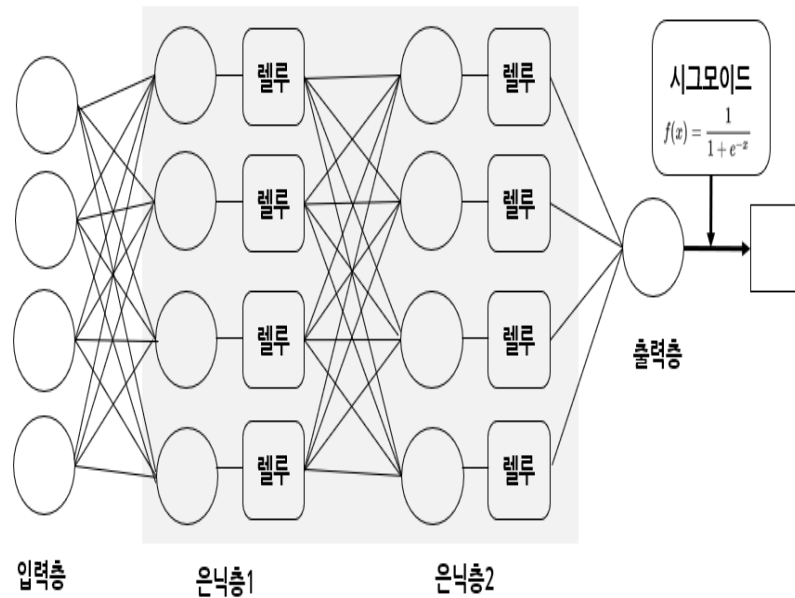


결과값 출력



## ■ 컨볼루션층(CNN)을 사용하는 이유

### ■ 완전연결층 vs 컨볼루션층 비교



완전 연결층  
Multi-layer Neural Network  
(Fully-connected Neural Network)



그림 출처 : <http://cs231n.github.io/convolutional-networks>

컨볼루션층  
Convolutional Neural Network



### ■ 컨볼루션층(CNN)을 사용하는 이유

#### ■ 완전연결층 vs 컨볼루션층 비교

##### 1) 완전연결층 단점

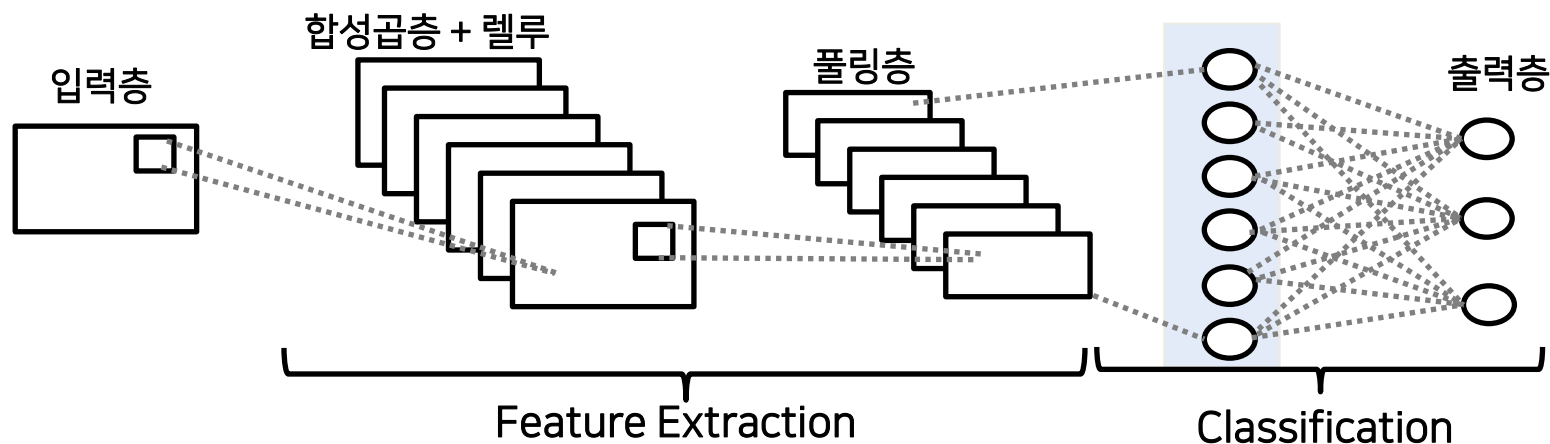
- 완전연결층(fully-connected layer)은 1차원 배열 형태의 데이터를 통해 학습
- 데이터를 단순히 1차원으로 펼쳐서 사용하기 때문에 각 이미지 픽셀의 관계를 고려하지 않는다.
- 2차원 배열 형태(28, 28)를 1차원 배열 형태의 데이터(28\*28)로 변환하면서 본래 데이터의 특징을 잃어버리게 된다.
- 지역적 특징이 아닌 전역적 특징을 학습하게 되는 것 -> '공간 정보를 손실한다'
- 완전연결층의 은닉 노드 수를 늘려 문제를 해결할 수도 있지만, 급격히 증가하는 파라미터 수로 인해 과대적합 문제가 발생할 수도 있다.

##### 2) 컨볼루션층을 사용하는 이유

- 이미지 픽셀 사이의 관계를 고려하기 때문에 지역적 특징을 학습하게 됨 -> '공간 정보를 유지한다'
- 각 필터의 파라미터가 공유되기 때문에 완전연결층에 비해 적은 수의 파라미터를 요구



## ■ 컨볼루션 신경망(CNN) 구조



### 1) 입력층(input layer)

- 입력 이미지 데이터가 최초로 거치게 되는 계층
- 이미지는 3차원 데이터 - (높이, 너비, 채널) - 이미지가 흑백이면(gray scale)이면 채널(Channel)은 1개, 컬러(RGB)이면 채널은 3개

### 2) 합성곱 층(convolutional layer)

- 입력 데이터에서 특성을 추출하는 역할을 수행
- 이미지에 대한 특성을 감지하기 위해 커널(kernel)이나 필터(Filter)를 이용

#### (1) 커널(kernel, 필터)

- 이미지의 모든 영역을 훑으면서 특성을 추출하게 되고, 추출된 결과물이 특성 맵(feature map)이 됨
- 3X3, 5X5 크기로 적용

#### (2) 스트라이드(stride)

- 필터가 지정된 간격에 따라 순차적으로 이동

### 3) 풀링층(Pooling Layer)

- 컨볼루션 층에서 이미지 특징을 도출하지만, 컨볼루션의 결과가 여전히 크고 복잡하여 풀링(pooling) ]을 사용하여 이미지를 축소
- 맥스 풀링(max pooling) - 정해진 구역 안에서 가장 큰 값만 다음 층으로 넘기고 나머지는 버림

### 4) 완전 연결층(fully-connected layer)

### 5) 출력층(output layer)



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

- 컨볼루션 필터(= 커널, 윈도우)
- 컨볼루션 연산
- 스트라이드(stride)
- 컨볼루션 신경망 층(Conv2D)
- 맥스 풀링(max pooling)
- 패딩(padding)
- 드롭 아웃(drop out)
- 플래튼(Flatten())



## ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

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

- 입력된 이미지에서 다시 한번 특징을 추출하기 위해 마스크(필터, 윈도우 또는 커널)를 도입하는 기법
- 컨볼루션을 만들면 입력 데이터로부터 더욱 정교한 특징을 추출할 수 있음
- (예) 입력 X 필터 = 출력

### ■ 컨볼루션 연산과 스트라이드

- 컨볼루션층은 주어진 입력 데이터에서 컨볼루션 필터를 활용하여 원소별 곱과 윈도우 슬라이딩을 행하는 컨볼루션 연산을 통해 특징맵(Feature Map)을 만들게 된다.
- 스트라이드 : 필터가 움직일 때 몇 칸씩 뛰면서 움직이는 지의 수치
- 예) 입력 4x4이미지, 필터 2X2, 스트라이드 1 → 출력 3X3

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

4x4 입력 이미지

X

1	0
0	1

2x2 필터

=

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

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

2	1	1
0	2	2
0	1	1

3x3 출력



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 컨볼루션 필터(= 커널, 윈도우)

- 컨볼루션층에서 '필터'라는 개념 사용
- 사진을 찍고 난 후, 특정 사진 앱을 사용하여 얼굴을 변형해주는 필터와 동일한 개념 → 이미지 필터
- 컨볼루션층은 여러 개의 컨볼루션 필터를 활용하여 이미지에 내포된 다양한 정보를 인식할 수 있다.

#### ■ 컨볼루션 연산 과정



그림 출처

[http://deeplearning.stanford.edu/wiki/index.php/Feature\\_extraction\\_using\\_convolution](http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution)

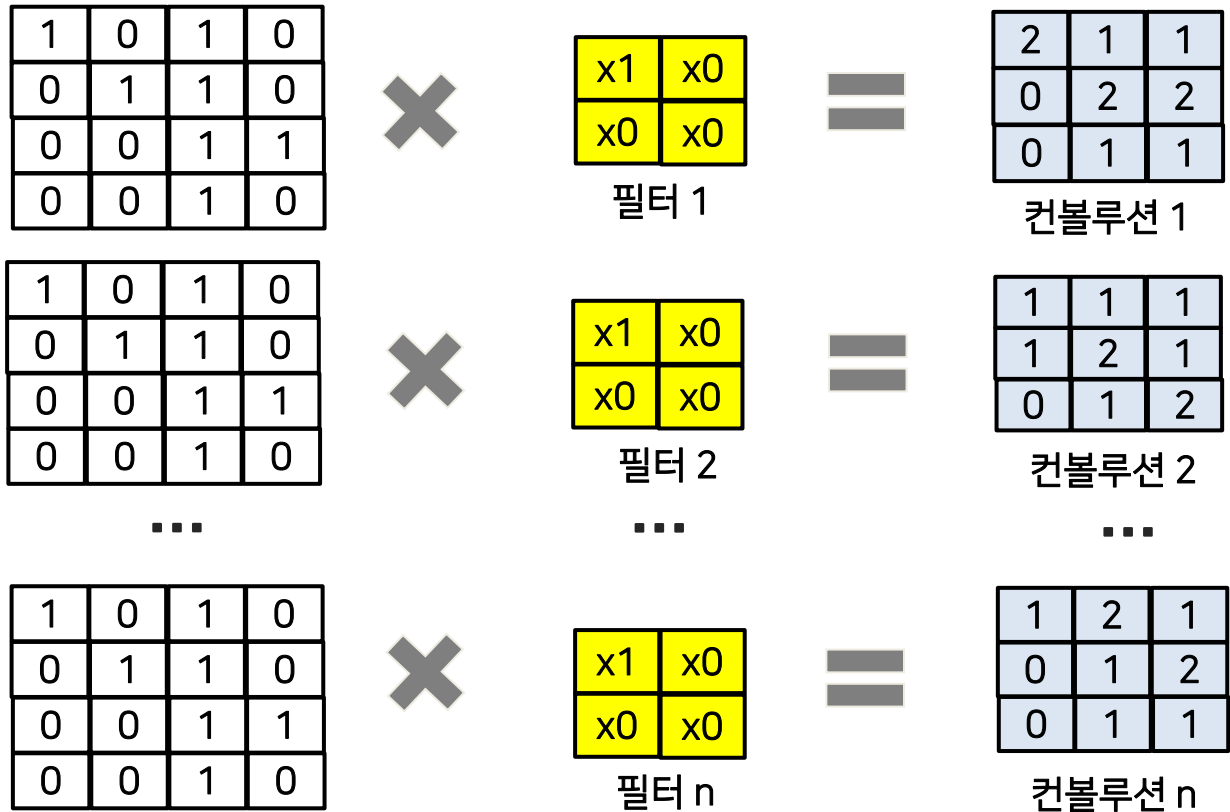


### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 컨볼루션 연산 과정

- 컨볼루션 필터의 크기는 2x2, 3x3을 주로 이용
- 컨볼루션 필터는 지정해 준 스트라이드 크기만큼 움직이게 된다.

- 필터(커널, 마스크)를 여러 개 사용하는 경우 → 여러 개의 컨볼루션이 만들어짐.







### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 컨볼루션 연산 과정

- (예) 이미지 크기 4X4, 채널 3개(RGB컬러) → 필터 3X3가 3개, 스트라이프 1 → 컨볼루션 2X2 3개 → 1개의 특징맵

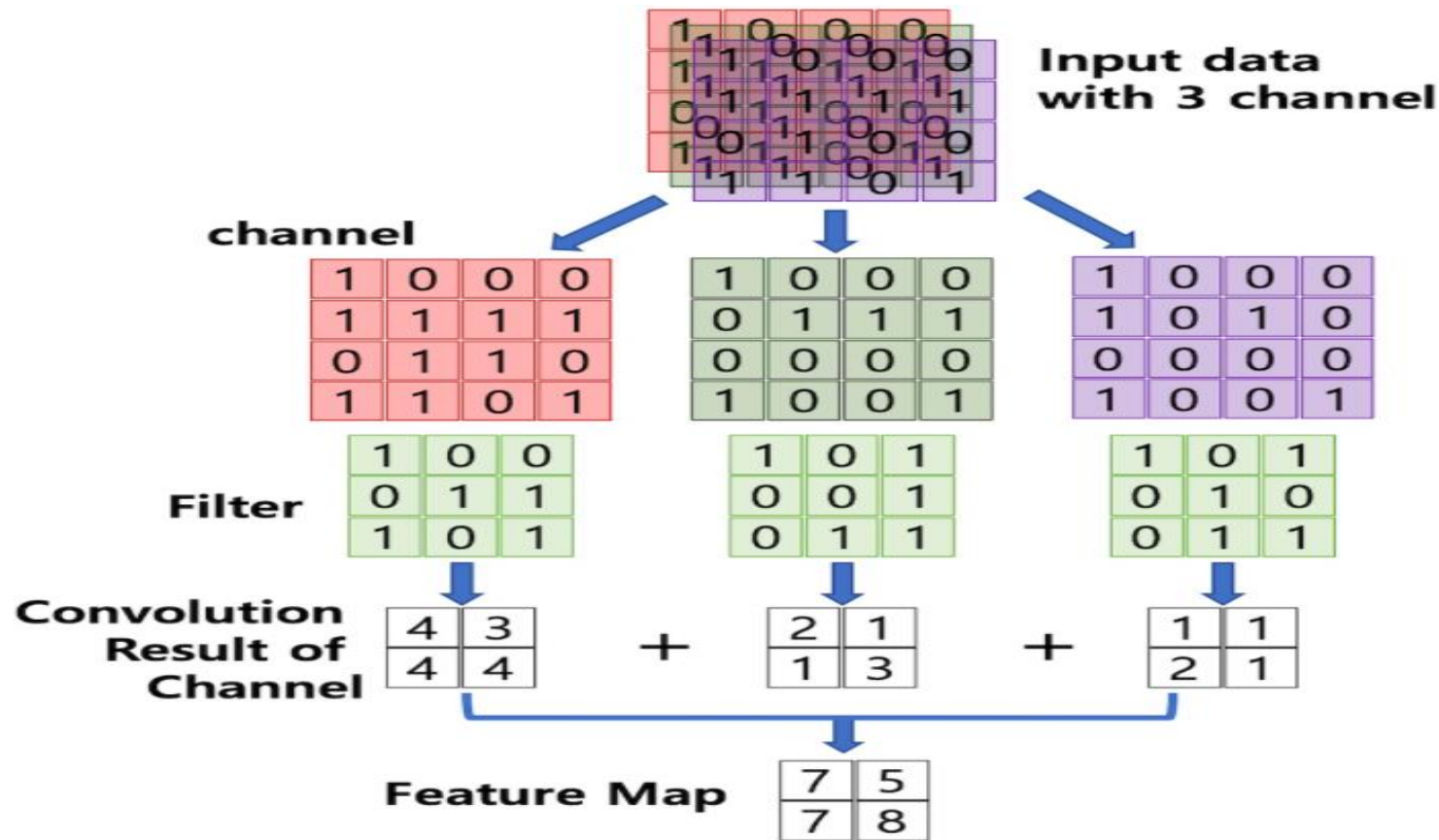


그림 출처 : <http://taewan.kim/post/cnn/>



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ▪ 컨볼루션 신경망(CNN) 층

##### ▪ Conv2D() 함수 : Conv2D(필터 수, 커널 사이즈, 활성화함수)

```
Conv2D(filters=16, kernel_size=3, strides=(1,1), padding='same',  
        activation='relu', input_size=(28,28,1))
```

- filters : 특징 맵의 차원을 결정
  - filters -16은 16개의 컨볼루션 필터를 사용한다는 의미
  - 특징맵의 형태는 (batch\_size, rows, cols, filters)가 된다.
- kernel\_size : (3,3)과 같이 튜플 형태로 필터의 크기를 설정
  - kernel\_size=3은 kernel\_size =(3,3)과 동일
- strides : 스트라이드 크기를 지정, 기본값은 (1,1)
  - kernel\_size와 같이 하나의 숫자 형태로도 사용할 수 있다.
- padding : 패딩에 대한 결정 여부 지정, 'same', 'valid'가 있으며 'valid'가 기본값
  - 'same' : 출력형태와 입력 형태가 동일하도록 조절
  - 'valid' : 패딩을 사용하지 않음
- activation : 사용할 활성화 함수를 문자열 또는 클래스 형태로 제공
- input\_shape : 입력 값의 형태



### ■ 컨볼루션 신경망(CNN) 구성 요소

#### ■ 컨볼루션 신경망(CNN) 층 : Conv2D(필터 수, 커널 사이즈, 활성화함수)

- Conv2D()

- 케라스에서 컨볼루션 층을 추가하는 함수

- 입력층+은닉층(합성곱층)

```
model.add(Conv2D(32, kernel_size=(3,3), input_shape=(28,28,1), activation='relu'))
```

- 첫번째 인자 : 필터의 개수, 32개의 필터를 적용함
  - kernel\_size : 필터의 크기, kernel\_size=(행,열) 형식, 여기서는 3x3 크기의 필터 사용
  - Input\_shape : Dense 층과 마찬가지로 맨 처음 층에는 입력되는 값을 알려주어야 함
  - Input\_shape=(행, 열, 색상 또는 흑백), 만약 입력 이미지가 색상이 있으면(RGB) 3, 흑백 1을 지정
  - activation : 활성화 함수를 정의, 여기서는 relu 함수

- 은닉층(합성곱층)

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

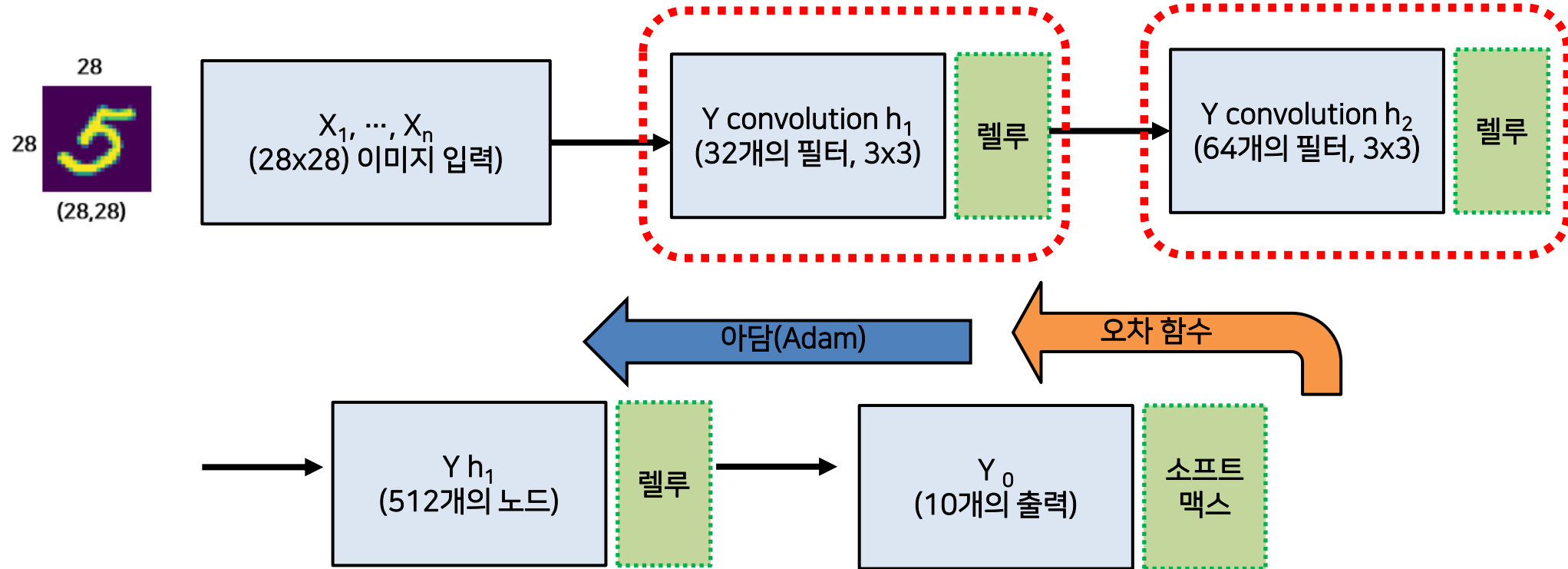
- 필터 64개를 적용한 새로운 컨볼루션 층 추가



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 케라스(Keras)에서 컨볼루션 층을 추가 : Conv2D() 함수 이용

- 1) 입력층+은닉층 : `model.add(Conv2D(32, kernel_size=(3,3), input_shape=(28,28,1), activation='relu'))`
- 2) 은닉층 : `model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))`
- 3) 완전연결층
- 4) 출력층





## ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

### ■ 풀링(Pooling) 연산

- 컨볼루션 층에서 이미지 특징을 도출하지만, 컨볼루션의 결과가 여전히 크고 복잡함  
→ 풀링(pooling) 또는 서브 샘플링(sub sampling)을 사용 이미지를 축소
- 풀링 연산 종류 - 평균 풀링(Average Pooling), 맥스 풀링(Max Pooling)
  - 평균 풀링(Average Pooling) : 풀링 연산에 평균값을 사용
  - 맥스 풀링(Max Pooling) : 풀링 연산에 최댓값을 사용

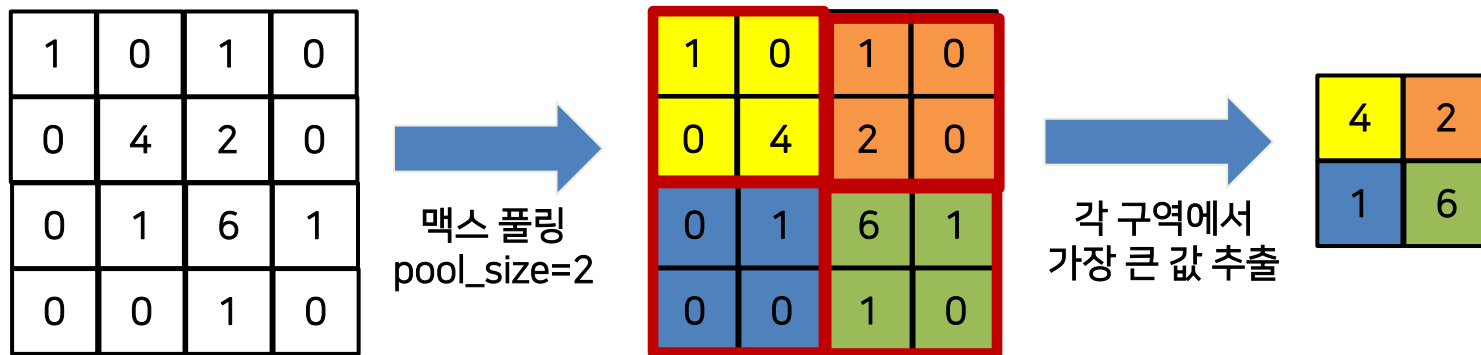
### ■ 맥스 풀링(max pooling)

- 풀링 기법 중 가장 많이 사용되는 방법
- 맥스 풀링은 정해진 구역 안에서 가장 큰 값만 다음 층으로 넘기고 나머지는 버림
- 맥스풀링은 일반적으로 2X2 스트라이드와 2X2 윈도우 크기를 사용하여 특징맵의 크기를 절반으로 줄이는 역할을 함(다운 샘플링)
- 맥스 풀링(max pooling)의 장점
  - 모델이 물체의 주요한 특징을 학습할 수 있도록 도와주며, 컨볼루션 신경망이 이동 불변성 특성을 가지도록 한다.
    - 이동 불변성 : 물체가 어느 위치에 있어도 그 물체를 인식할 수 있다는 것을 의미
  - 모델 파라미터 수를 줄여준다.
    - 계산 속도 향상, 과대적합 문제 노출될 위험을 줄임



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ▪ 예) 맥스 풀링(max pooling)



- MaxPooling2D() 함수를 이용하여 맥스 풀링 진행
- model.add(MaxPooling2D(pool\_size=2))
  - pool\_size : 풀링 창 크기를 정하는 것
    - pool\_size = 2로 하면, 2X2 스트라이드와 2X2 윈도우 크기를 사용하여 전체 크기가 절반으로 줄어듦



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

- 예) 맥스 풀링과 평균 풀링 비교

Activation Map

12	20	30	0
8	12	2	0
34	70	37	7
112	100	22	12

window size: 2x2, stride : 2

맥스 풀링(Max Pooling)

20	30
112	37

$\max(30, 0, 2, 0) = 30$

평균 풀링(Average Pooling)

13	8
79	18

$(30+0+2+0)/4 = 8$



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ▪ 케라스의 MaxPool2D() 함수

```
MaxPool2D(pool_size=(2,2), strides=2, padding='same')
```

- pool\_size : 풀링 층에서 사용할 커널의 크기를 설정  
Conv2D의 kernel\_size처럼 하나의 숫자 형태로도 사용할 수 있음
- strides : 스트라이드 크기를 지정, 기본값은 None
  - 이 값이 주어지지 않는 경우, pool\_size의 크기와 동일한 크기로 지정
  - 예를 들어, pool\_size=(2,2)이고, strides=None이면 실제 최대 풀링 층의 스트라이드는 (2,2)로 적용
- padding : 패딩에 대한 결정 여부 지정, 'same', 'valid'가 있으며 'valid'가 기본값
  - 'same' : 출력형태와 입력 형태가 동일하도록 조절
  - 'valid' : 패딩을 사용하지 않음





### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 패딩(Padding)

- 특징맵의 크기가 감소하지 않고, 입력 데이터의 형태와 동일한 형태를 출력값으로 얻고 싶을 경우 패딩을 사용
- 패딩을 사용하게 되면 이미지 가장자리 부분에 해당하는 정보를 손실하지 않고 전달해줄 수 있는 효과가 있다.
- 패딩은 행과 열에 특정 숫자를 추가하는 것
- Zero Padding은 원본 이미지에 0으로 이루어진 부분을 추가하여 이미지를 크게 만들어 주는 것

#### ■ 제로 패딩(Zero Padding)의 예

0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

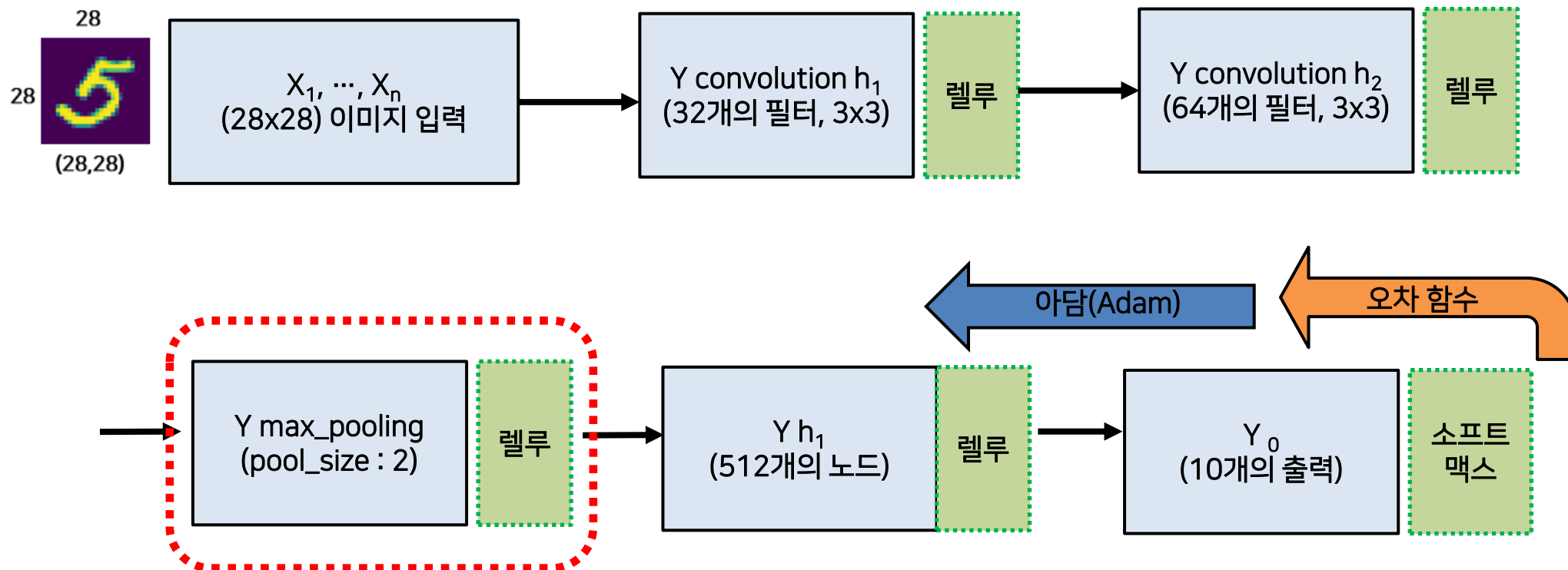
최종 사이즈 8x8

제로 패딩(Zero Padding)



### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

- 맥스 풀링(Max pooling) 층을 추가한 컨볼루션 신경망(CNN)





### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 레이어별 출력 데이터 산정

##### 1) 컨볼루션층 출력 데이터 크기 산정

- 입력 데이터 높이 :  $H$
- 입력 데이터 폭 :  $W$
- 필터 높이 :  $FH$
- 필터 폭 :  $FW$
- Stride 크기 :  $S$
- 패딩 사이즈 :  $P$

$$OutputWeight = OW = \frac{(W+2P-FW)}{S} + 1$$

$$OutputHeight = OH = \frac{(H+2P-FH)}{S} + 1$$

##### 2) 풀링층 출력 데이터 크기 산정

$$OutputRowSize = \frac{InputRowSize}{PoolingSize}$$

$$OutputColumnSize = \frac{InputColumnSize}{PoolingSize}$$



## ■ 레이어별 출력 데이터 산정

### 1) 컨볼루션층 출력 데이터 크기 산정

- 입력 데이터 높이 : H
- 입력 데이터 폭 : W
- 필터 높이 : FH
- 필터 폭 : FW
- Stride 크기 : S
- 패딩 사이즈 : P

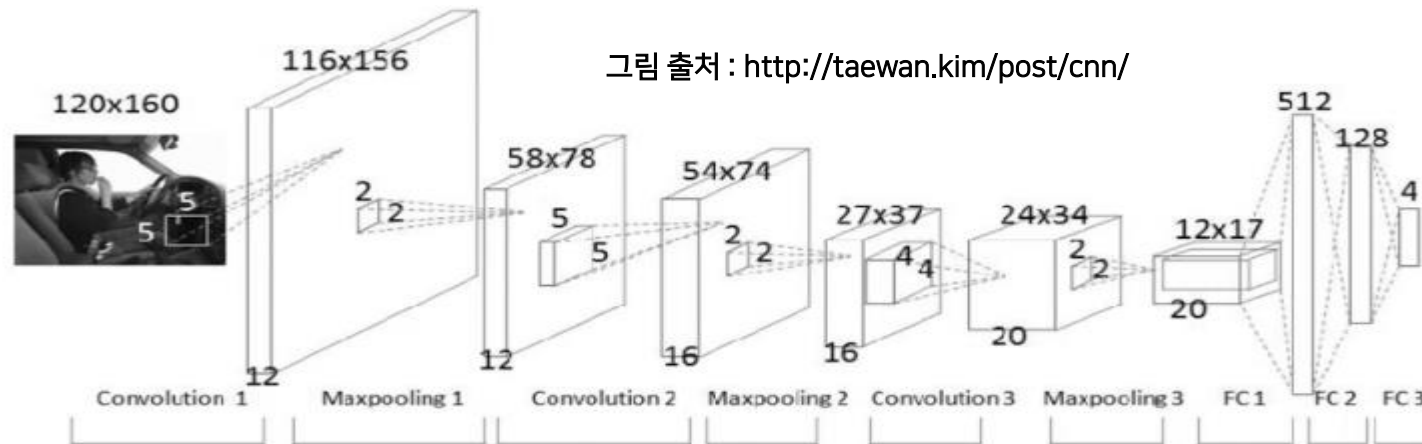
$$OutputWeight = OW = \frac{(W+2P-FW)}{S} + 1$$

$$OutputHeight = OH = \frac{(H+2P-FH)}{S} + 1$$

$$OutputRowSize = \frac{InputRowSize}{PoolingSize}$$

$$OutputColumnSize = \frac{InputColumnSize}{PoolingSize}$$

### 2) 풀링층 출력 데이터 크기 산정

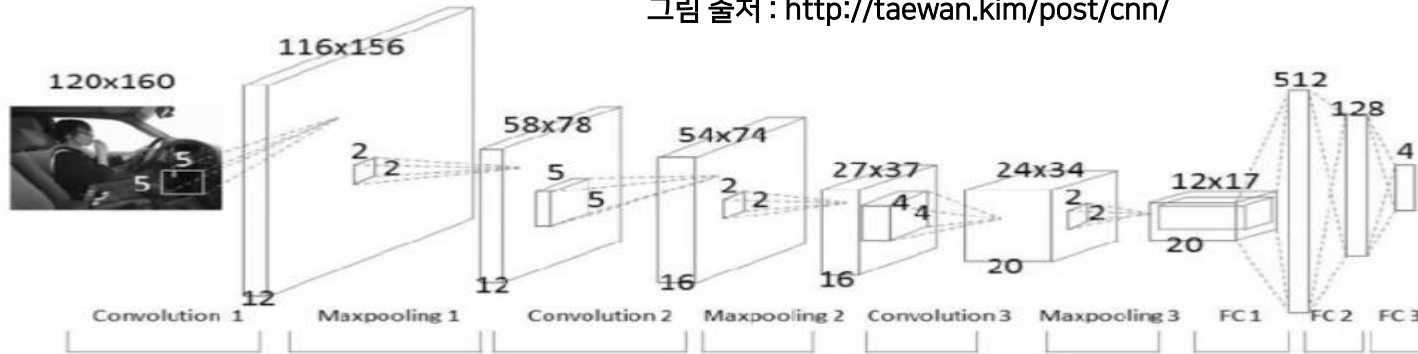


- 컨볼루션층1 결과 데이터의 높이 =  $(160 + 2*0 - 5)/1 + 1 = 156$
- 컨볼루션층1 결과 데이터의 너비 =  $(120 + 2*0 - 5)/1 + 1 = 116$
- 맥스풀링1 결과 데이터의 너비 =  $116/2 = 58$
- 맥스풀링1 결과 데이터의 높이 =  $156/2 = 78$



## 레이어별 출력 데이터 산정

그림 출처 : <http://taewan.kim/post/cnn/>



- 컨볼루션층1 결과 데이터의 너비 =  $(120 + 2*0 - 5)/1 + 1 = 116$
- 컨볼루션층1 결과 데이터의 높이 =  $(160 + 2*0 - 5)/1 + 1 = 156$
- 맥스풀링1 결과 데이터의 너비 =  $116/2 = 58$
- 맥스풀링1 결과 데이터의 높이 =  $156/2 = 78$
- 컨볼루션층2 결과 데이터의 너비 =  $(58 + 2*0 - 5)/1 + 1 = 54$
- 컨볼루션층2 결과 데이터의 높이 =  $(78 + 2*0 - 5)/1 + 1 = 74$
- 맥스풀링2 결과 데이터의 너비 =  $54/2 = 27$
- 맥스풀링2 결과 데이터의 높이 =  $74/2 = 37$
- 컨볼루션층3 결과 데이터의 너비 =  $(27 + 2*0 - 4)/1 + 1 = 24$
- 컨볼루션층3 결과 데이터의 높이 =  $(37 + 2*0 - 4)/1 + 1 = 34$
- 맥스풀링3 결과 데이터의 너비 =  $24/2 = 12$
- 맥스풀링3 결과 데이터의 높이 =  $34/2 = 17$
- 밀집층1 : 512
- 밀집층2 : 128
- 출력층 : 4



### ■ CNN 입출력 파라미터 계산

- 입력 데이터 shape : (39,31,1)
- 분류 클래스 : 100
- 모델 구성

layer	Input Channel	Filter	Output Channel	Stride	Max Pooling	Activation function
Convolution Layer 1	1	(4, 4)	20	1	x	relu
Max Pooling Layer 1	20	x	20	2	(2, 2)	x
Convolution Layer 2	20	(3, 3)	40	1	x	relu
Max Pooling Layer 2	40	x	40	2	(2, 2)	x
Convolution Layer 3	40	(2, 2)	60	1	1	relu
Max Pooling Layer 3	60	x	60	2	(2, 2)	x
Convolution Layer 4	60	(2, 2)	60	1	1	relu
Flatten	x	x	x	x	x	x
Fully Connected Layer	x	x	x	x	x	softmax



## ■ CNN 입출력 파라미터 계산

- 컨볼루션층의 학습 파라미터 수 = 입력채널 x 필터폭 x 필터 높이 x 출력 채널수

- (예) 컨볼루션층1의 기본 정보

- 입력 데이터 Shape = (39,31,1)
- 입력 채널 = 1
- 필터 = (4,4)
- 출력 채널 = 20
- Stride = 1

- 입력 데이터 높이 : H
- 입력 데이터 폭 : W
- 필터 높이 : FH
- 필터 폭 : FW
- Stride 크기 : S
- 패딩 사이즈 : P

- 컨볼루션층1의 결과 데이터 너비 =  $(39 + 2*0 - 4)/1 = 36$
- 컨볼루션층1의 결과 데이터 높이 =  $(31 + 2*0 - 4)/1 = 28$
- 컨볼루션층1의 결과 데이터 shape = **(36, 28, 20)**

$$OutputWeight = OW = \frac{(W+2P-FW)}{S} + 1$$

$$OutputHeight = OH = \frac{(H+2P-FH)}{S} + 1$$

- 컨볼루션층1에서 학습 파라미터 수 계산하기

- 입력 채널 : 1
- 필터 사이즈 : (4,4)
- 출력 채널 : 20
- 학습 파라미터 = 입력채널 x 필터폭 x 필터 높이 x 출력 채널수 =  $4 \times 4 \times 20 = 320$ 개

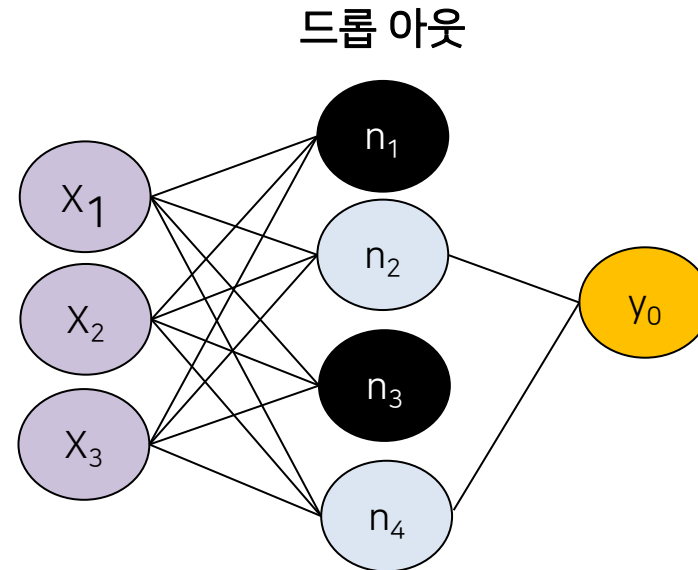
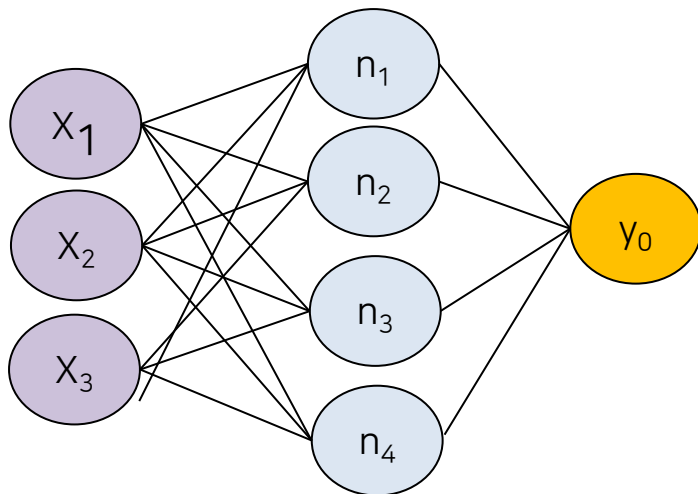


## ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

### ■ 드롭 아웃(drop out)

- 노드가 많아지거나 층이 많아진다고 해서 학습 결과가 무조건 좋아지는 것이 아니다.  
→ 과적합(overfitting) 발생!
- 드롭 아웃(drop out) 기법
  - 과적합을 피하는 간단하지만 효과가 큰 기법
  - 은닉층에 배치된 노드 중 일부를 임의로 꺼주는 것

### • 드롭 아웃(drop out)의 예



- 랜덤하게 노드를 끄으로써 학습 데이터에 지나치게 치우쳐서 학습되는 과적합을 방지  
(예) `model.add(Dropout(0.25))`  
-----→ 25%의 노드를 드롭 아웃

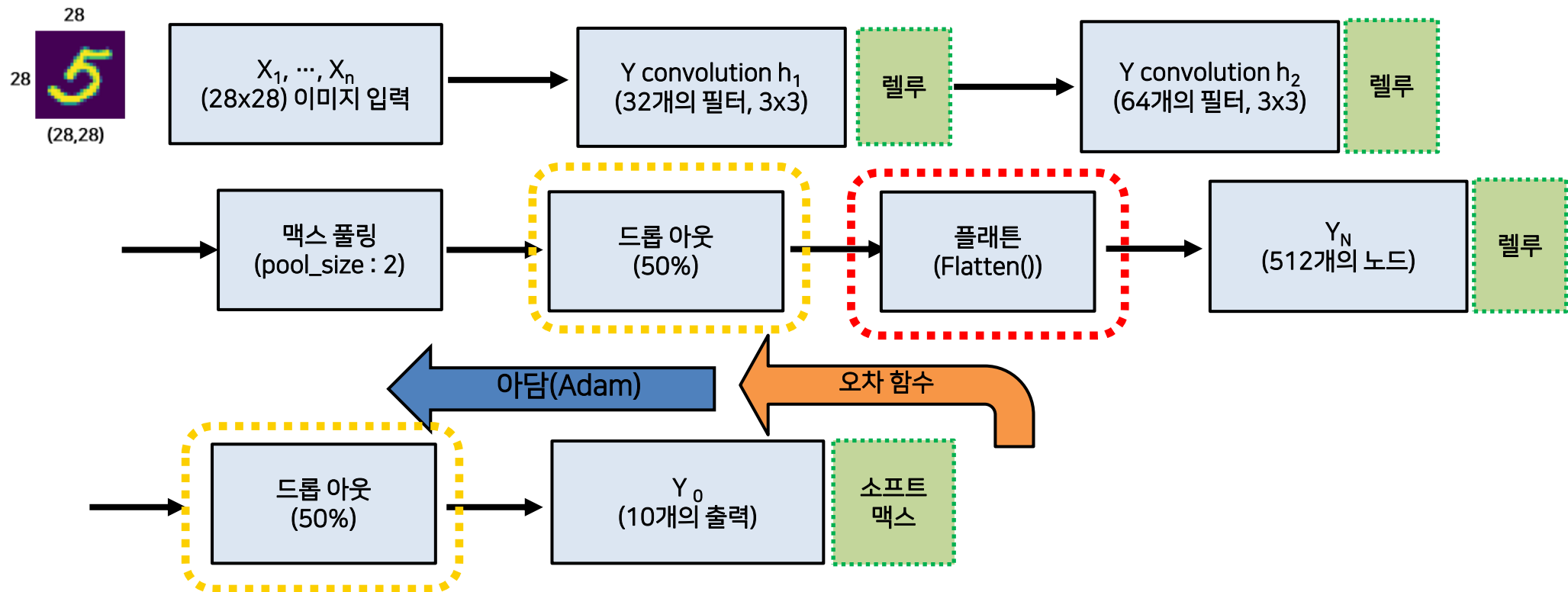




### ■ 컨볼루션 신경망(CNN) 구성 요소와 연산

#### ■ 플래튼 : Flatten() 함수

- 드롭 아웃 과정을 지나 다시 앞에서 Dense() 함수를 이용해 만들었던 기본 층에 연결할 때
- 컨볼루션 층이나 맥스 풀링은 주어진 이미지를 2차원 배열인 채로 다루어 진다.
- 이를 1차원으로 바꿔주는 함수가 플래튼 함수 → Flatten() 함수
- (예) model.add(Flatten())

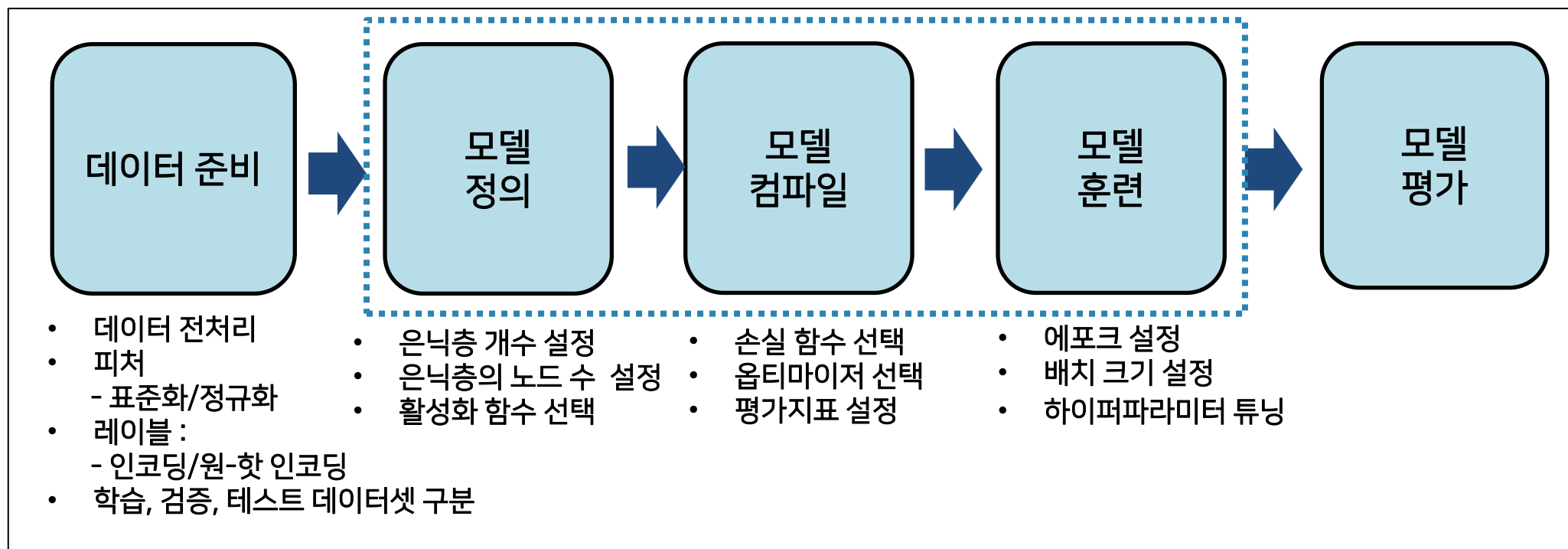




# CNN으로 MNIST 손글씨 분류하기



### ■ 딥러닝(Deep Learning) 개발 과정





### ■ CNN으로 MNIST 손글씨 분류하기

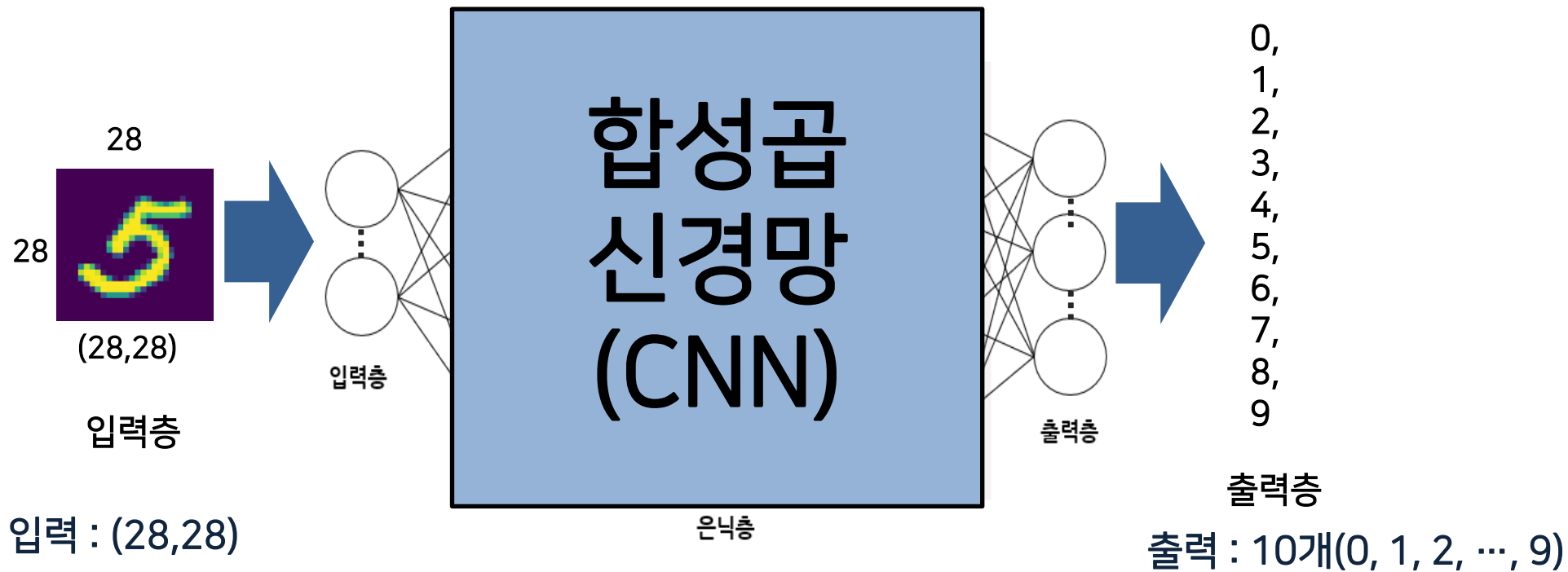
- MNIST 손글씨 인식 데이터 셋



- MNIST 데이터셋은 미국 국립표준기술원(NIST)이 고등학생과 인구조사국 직원 등이 쓴 손 글씨를 이용해 만든 데이터로 구성
- 70,000개의 28\*28 글자 이미지에 각각 0부터 9까지 이름표를 붙인 데이터
- 70,000개 = 60,000개의 학습 데이터셋 + 10,000개의 테스트 데이터셋으로 구성



### ■ CNN으로 MNIST 손글씨 분류하기





### ■ CNN으로 MNIST 손글씨 분류하기

#### ■ 참고) 케라스 콜백(Keras Callback)

- 모델의 학습 방향, 저장 시점, 학습 정지 시점 등에 관한 상황을 모니터링하기 위해 주로 사용
- 모델의 fit() 함수를 통해 반환되는 History 객체를 활용하여 학습 과정 시각화에 사용  
→ 케라스 콜백 중 하나인 History 콜백이 모든 케라스 모델에 자동으로 적용되어 있기 때문
- 케라스 콜백 4가지 : ModelCheckpoint, EarlyStopping, ReduceROnPlateau, TensorBoard

#### 1) ModelCheckpoint 콜백

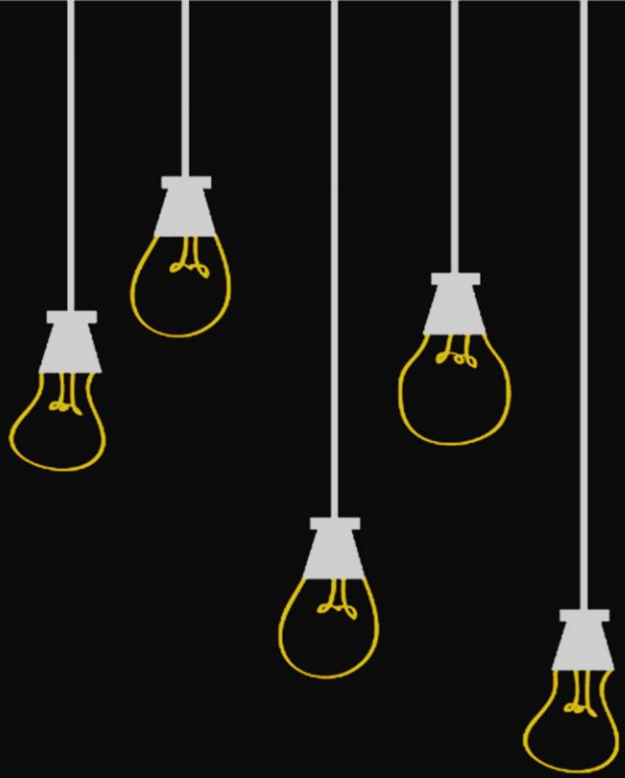
- 지정한 평가 지표를 기준으로 가장 뛰어난 성능을 보여주는 모델을 저장할 때 사용
- `ModelCheckpoint(filepath, monitor= ' val_loss ' , verbose=0, save_best_only=False, save_weights_only=False, mode='auto')`
  - `filepath` : 모델의 저장 경로 지정
  - `monitor` : 모니터링할 평가 지표를 설정, 모델이 포함하고 있는 모든 지표를 사용할 수 있음
  - `verbose` : 콜백의 수행 과정 노출 여부를 지정, 0(아무런 표시하지 않음), 1(프로그래스바로 나타남), 2(매 에폭마다 수행과정을 설명)
  - `save_best_only` : True인 경우, 가장 뛰어난 모델만 저장. 그보다 좋지 않은 모델의 경우에는 덮어쓰지 않음
  - `save_weights_only` : 모델의 가중치만 저장
  - `mode` : ['auto', 'min', 'max'] 중 하나를 사용, `monitor`에서 지정한 평가지표를 기준으로 작동
    - 평가지표가 `val_acc` 인 경우 `max`를 선택, 평가지표가 `val_loss`인 경우 `min`을 선택
    - `auto`인 경우 평가지표의 이름을 통해 자동으로 유추하여 결정



### ■ CNN으로 MNIST 손글씨 분류하기

#### 2) EarlyStopping 콜백

- 이른 멈춤
- 모델 학습 시에 지정된 기간 동안 모니터링하는 평가지표에서 성능 향상이 일어나지 않은 경우 학습을 중단
- `EarlyStopping(monitor='val_loss', patience=0, verbose=0, mode='auto')`
  - `patience` : 지정한 수만큼의 기간에서 평가지표의 향상이 일어나지 않을 경우 학습을 중단함  
(예) `patience=5`일 때, 5에폭 동안 성능 향상이 일어나지 않으면 학습을 중단한다.
  - `verbose` : 콜백의 수행 과정 노출 여부를 지정, 0(아무런 표시하지 않음),  
1(프로그래스바로 나타남), 2(매 에폭마다 수행과정을 설명)
  - `mode` : ['auto', 'min', 'max'] 중 하나를 사용, `monitor`에서 지정한 평가지표를 기준으로 작동
    - 평가지표가 `val_acc` 인 경우 `max`를 선택, 평가지표가 `val_loss`인 경우 `min`을 선택
    - `auto`인 경우 평가지표의 이름을 통해 자동으로 유추하여 결정



Actionable Content  
**MASO CAMPUS**

# 순환 신경망(RNN)





# 순환 신경망 (RNN, Recurrent Neural Network)

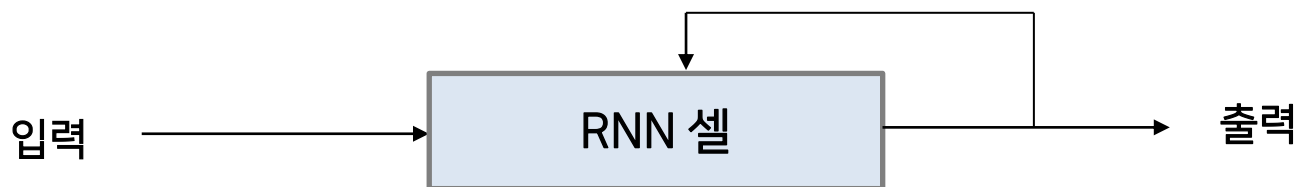


## ■ RNN(순환 신경망, Recurrent Neural Network)

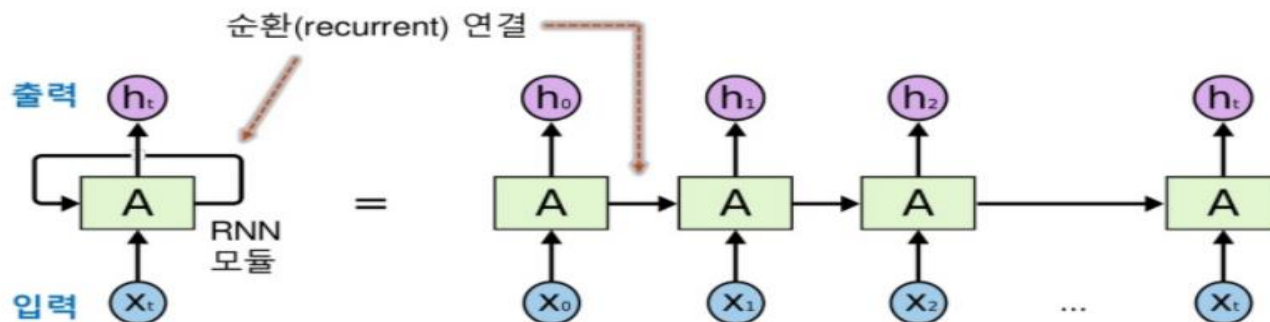
### ■ 순환 신경망(Recurrent Neural Network, RNN)

- 여러 개의 데이터가 순서대로 입력되었을 때 앞서 입력 받은 데이터를 잠시 기억해 놓은 방법
- 기억된 데이터가 얼마나 중요한지를 판단하여 별도의 가중치를 줘서 다음 데이터 넘어 감
- 모든 입력 값에 이 작업을 순서대로 실행하므로 다음 층으로 넘어가기 전에 같은 층을 맴도는 것처럼 보임 → 이렇게 같은 층 안에서 맴도는 성질 때문에 '순환신경망'이라고 부름

순환



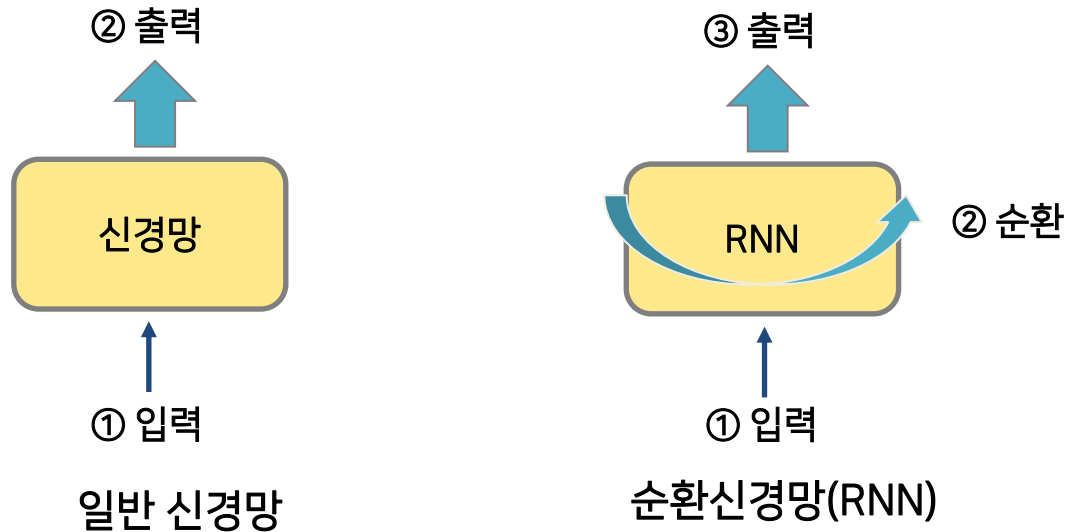
- 순환신경망(RNN)은 노드가 출력 값을 반환하는 동시에 이전 상태(state)를 기억하는 메모리 역할을 수행  
→ RNN 셀
- RNN 셀의 상태 : 은닉 상태(hidden state)





## ■ RNN(순환 신경망, Recurrent Neural Network)

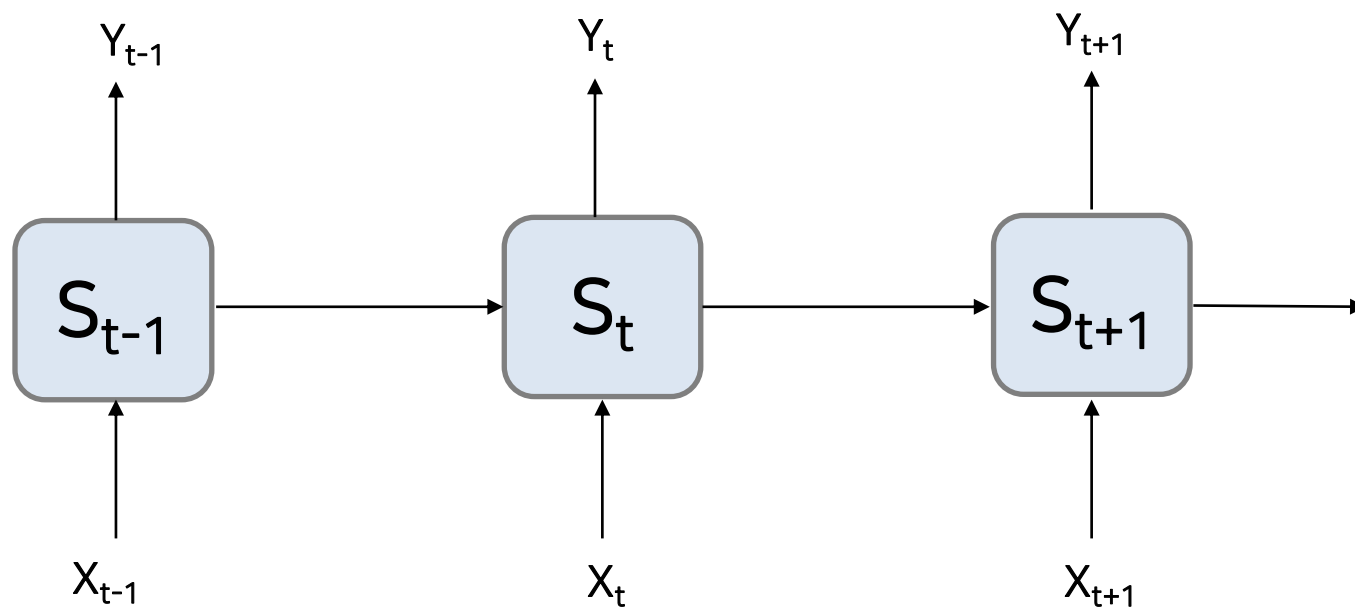
### ▪ 일반 신경망과 순환 신경망의 차이



- 일반 신경망(완전 연결층, 컨볼루션 신경망)은 피트 포워드 네트워크(feed-forward network)로 모든 출력 값이 마지막 층인 출력층을 향함
- 순환신경망(RNN)은 각 층의 결과값이 출력층을 향하면서도 동시에 현재 층의 다음 계산에 사용됨



## ■ RNN(Recurrent Neural Network) 구조



순환신경망(RNN)의 구조

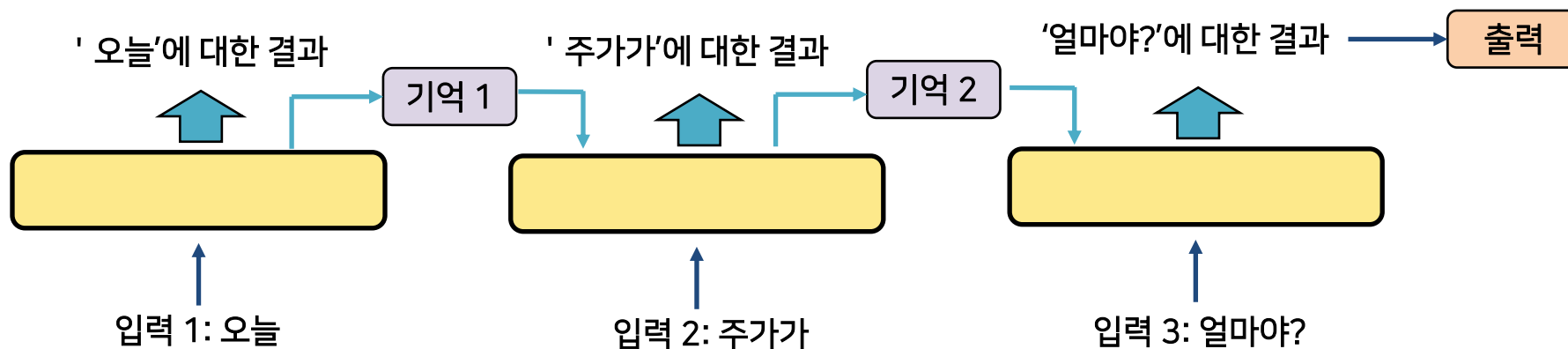
- $X$  : 입력,  $Y$  : 출력,  $t$  : 현재 시점
- 현재 시점  $t$ 에서 은닉 상태는 이전 시점  $t-1$ 의 은닉상태를 활용하여 업데이트 됨



## ■ RNN(순환 신경망, Recurrent Neural Network)

- 순환신경망(RNN)의 처리 방식과 사용하는 이유

(예) 인공지능 비서에게 '오늘 주가가 얼마야?'를 RNN이 처리하는 방식



- 순환이 되는 가운데 앞서 나온 입력에 대한 결과가 뒤에 나오는 입력 값에 영향을 주는 것을 알 수 있음

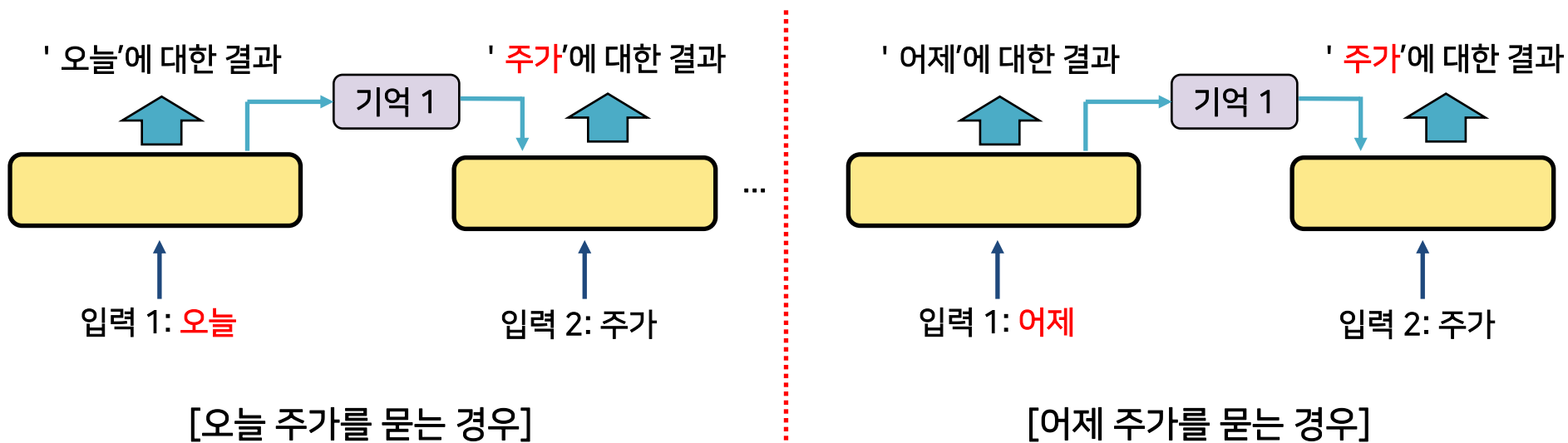


### ■ RNN(순환 신경망, Recurrent Neural Network)

#### ▪ 순환신경망(RNN)의 처리 방식과 사용하는 이유

(예) 입력 2의 값은 왼쪽과 오른쪽 그림 모두 '주가' 이지만,  
왼쪽의 주가는 오늘을 기준으로, 오른쪽은 어제를 기준으로 계산되어야 함

#### RNN을 사용하는 이유



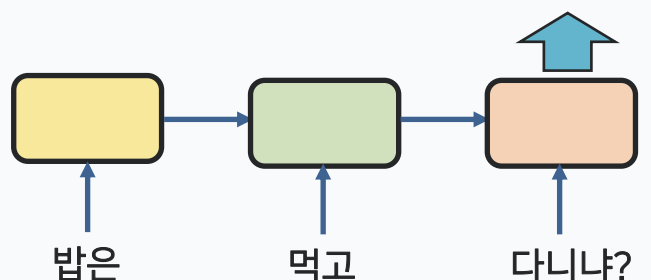


## ■ RNN(순환 신경망, Recurrent Neural Network)

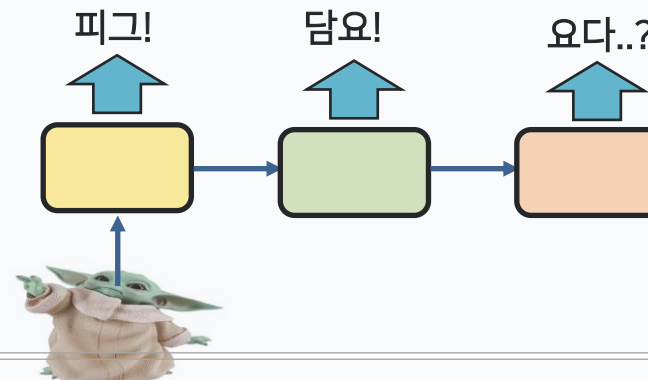
### ■ 순환 신경망(RNN)의 활용

- 1) 다수 입력, 단일 출력 - (예) 문장의 의도 파악
- 2) 단일 입력, 다수 출력 - (예) 사진의 캡션 달기
- 3) 다수 입력, 다수 출력 - (예) 문장 번역

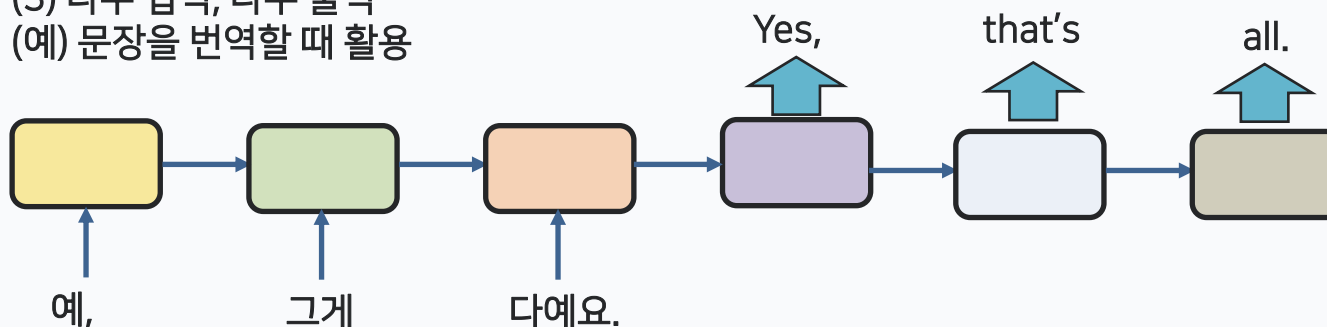
(1) 다수 입력, 단일 출력  
(예) 문장을 읽고 뜻을 파악할 때 활용



(2) 단일 입력, 다수 출력  
(예) 사진의 캡션을 만들 때 활용



(3) 다수 입력, 다수 출력  
(예) 문장을 번역할 때 활용





# IMDB 데이터셋을 RNN으로 실행하기

## - 영화 리뷰 감성 분석





# LSTM

## (Long Short-Term Memory)



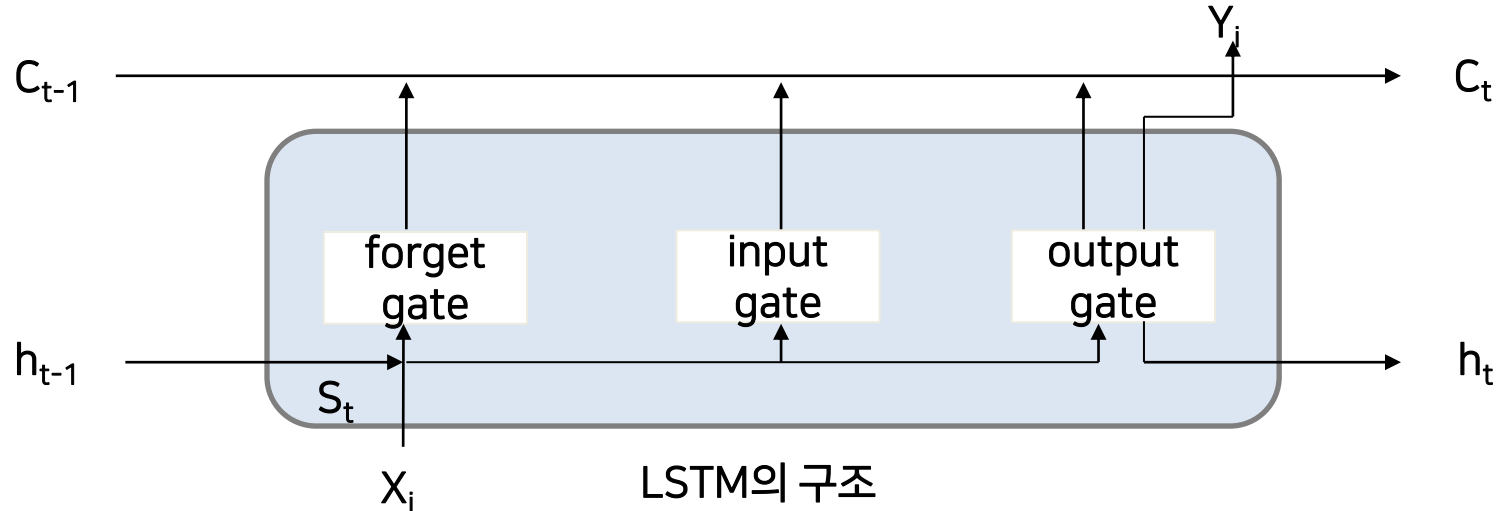
## ■ LSTM(Long Short-Term Memory)

### ■ LSTM(Long Short-Term Memory)

- Simple RNN은 Dense 층과 같이 기능적으로 매우 단순하게 구성되어 있기 때문에 깊이 쌓을수록 학습에 큰 문제가 존재
- “시점이 흐를수록 지속해서 기억하지 못한다”  
→ Simple RNN 층에 그래디언트 손실 문제가 존재한다는 것을 의미
- 이를 해결하기 위해 고안된 것 → LSTM(Long Short-Term Memory)
- 1997년 호흐라이터(Hochreite)와 슈미트후버(Schmidhuber)에 의해 만들어짐

### ■ LSTM의 원리

- LSTM 핵심 : 정보를 여러 시점에 걸쳐 나르는 장치가 추가된 점
- 이로 인해 그래디언트를 보존할 수 있어 그래디언트 손실 문제가 발생하지 않도록 도와준다.





## ■ LSTM(Long Short-Term Memory)

### ▪ LSTM(Long Short-Term Memory)의 원리

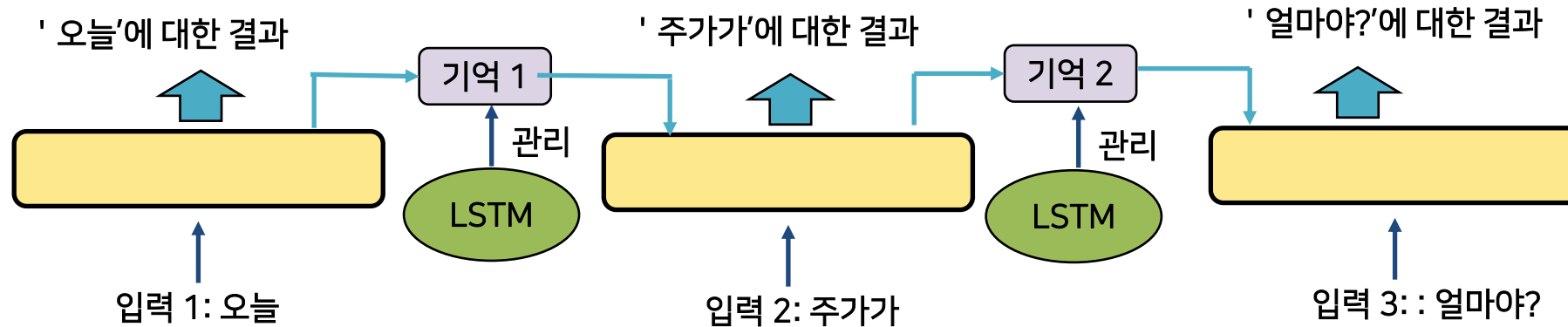
- 'C' : LSTM에는 정보를 여러 시점에 걸쳐 나르는 'Cell state' 가 있다.
- 'h' : 은닉 상태를 의미
- 정보를 나르는 작업을 도와줄 3개의 게이트(gate)가 존재 : forget gate, input gate, output gate
  - 1) forget gate
    - Cell state가 나르는 정보 중 관련 없는 시그모이드 함수를 사용하여 제거
  - 2) input gate
    - 두가지 작업 진행
      - (1) 현 시점의 정보( $x_i$ )와 이전 시점의 상태( $h_{t-1}$ )에 시그모이드 함수를 활용하여 어떤 정보를 업데이트 할 지를 결정
      - (2) 현재 시점의 정보와 이전 시점의 상태에 tanh 함수를 활용하여 새로운 정보를 만든다.
    - 이 둘을 곱한 뒤 forget gate를 통해 걸러진 정보와 더해져 현재 시점의 Cell state를 만들게 된다. ( $c_i$ )
  - 3) output gate
    - 출력 값과 현재 시점의 상태( $h_i$ )를 만든다.
- 모든 연산이 'Cell state'를 중심으로 수행된다는 것을 기억하고 사용
- LSTM의 핵심적인 기능은 'Cell state'를 통해 이전 정보를 계속해서 사용하여 그래디언트 손실 문제를 방지하는 것
- 반복되기 직전에 다음 층으로 기억된 값을 넘길지 안 넘길지를 관리하는 단계를 하나 더 추가한 것



## ■ LSTM(Long Short-Term Memory)

- LSTM(Long Short-Term Memory)의 원리

LSTM은 기억 값의 가중치를 관리하는 장치





# LSTM을 이용한 뉴스 카테고리 분류

# THANK YOU

---

마소캠퍼스  
이메일 문의  
전화 문의

[www.masocampus.com](http://www.masocampus.com)  
[biz@masocampus.com](mailto:biz@masocampus.com)  
02-6080-2022