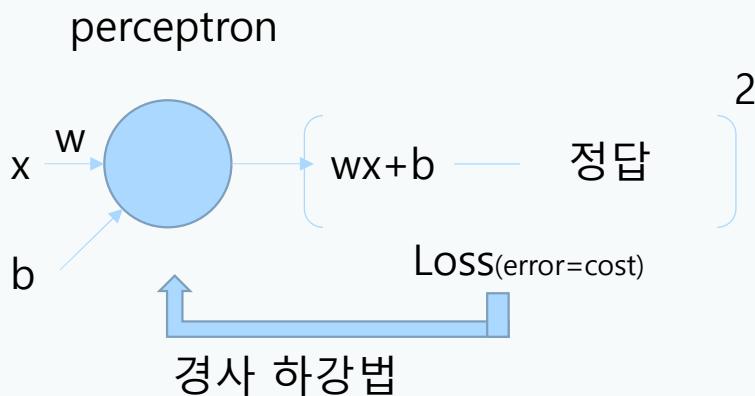


DAY 4

이미지 데이터 Neural Network 한계
CNN 이론
CNN 실습

0. 복습

1. Perceptron

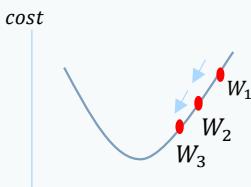


$$\begin{aligned} & (WX - Y)^2 \\ & = W^2X^2 - 2WXY + Y^2 \\ & = AW^2 - BW + C \\ & = \text{cost(error)} \end{aligned}$$

기울기

$$W_{\text{update}} = W - \alpha \frac{\partial}{\partial W} \text{cost}(W)$$

러닝 레이트



Regression

출력 함수 = None

loss = mse

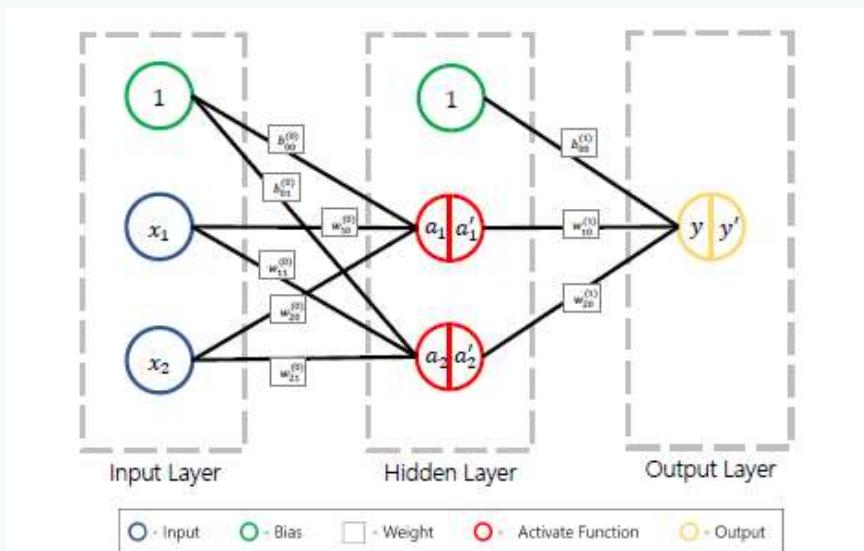
Binary classification

출력함수 = sigmoid

loss = binary crossentropy

0. 복습

2. Multi Layer Perceptron



다층 구조가 가능 하기 위해

activation function 사용 = sigmoid

hidden layer의 역할

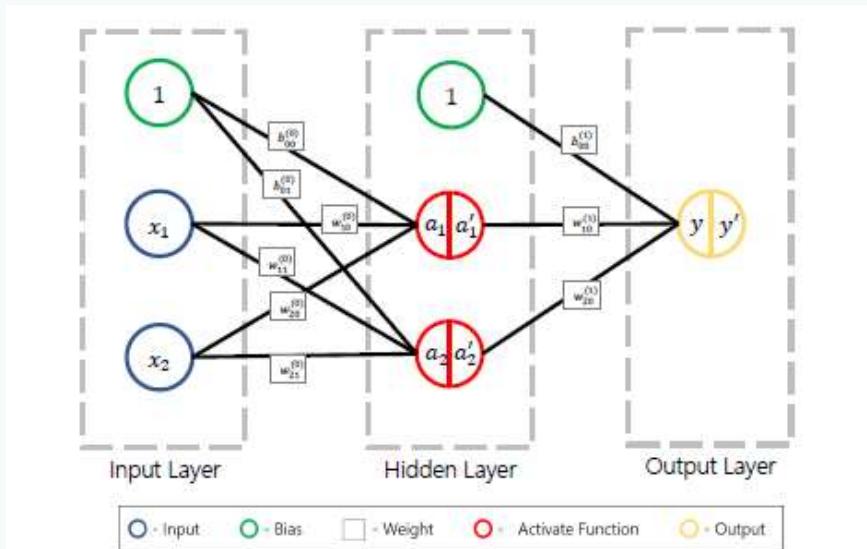
output layer가 잘 동작하도록 입력 데이터를
representation 해주는 역할

입력 차원 : (batch size, 데이터 변수)

모델 학습 알고리즘 : backpropagation

0. 복습

2. Multi Layer Perceptron



Regression

출력 함수 = None

loss = mse

Binary classification

출력함수 = sigmoid

loss = binary crossentropy

Categorical classification

출력함수 = softmax

loss = categorical crossentropy

(one hot encoding 사용)

0. 복습

2. Multi Layer Perceptron

다층 구조가 가능 하기 위해

여기서 hidden layer의 activation function과 output layer의 activation function은 다르다!

Hidden layer의 activation function

1. 여러 layer를 쌓는 효과를 보기위해 비선형적이어야함.
2. Layer들의 w,b가 잘 업데이트 되기 위해 오차 역전파에서 오차의 미분값이 잘 전달되도록해야함
3. 미분 가능해야함

Output layer의 activation function은

Task에 따라 선정.

Regression 의 activation function은 None (regression은 출력값이 숫자값으로 입력*w + b값이 예측값)

Classification의 activation function은 sigmoid or softmax (classification은 0 or 1의 제한된 출력)

0. 복습

2. Multi Layer Perceptron

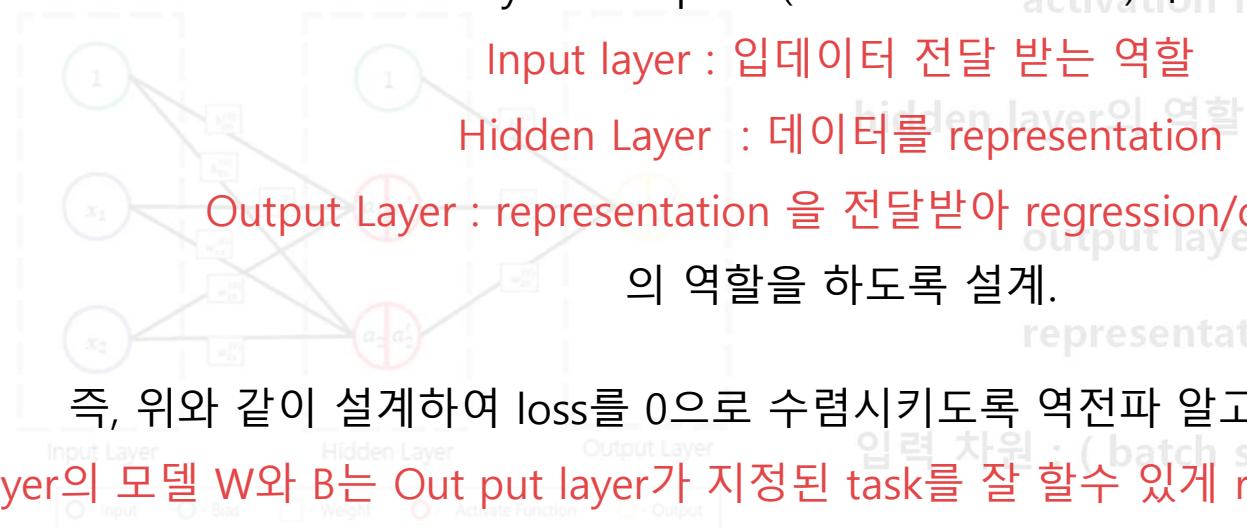
다층 구조가 가능 하기 위해

Multi Layer Perceptron(=Neural Network) 구조 할때

Input layer : 입데이터 전달 받는 역할

Hidden Layer : 데이터를 representation

Output Layer : representation 을 전달받아 regression/classification
의 역할을 하도록 설계.



즉, 위와 같이 설계하여 loss를 0으로 수렴시키도록 역전파 알고리즘으로 학습하면

Hidden Layer의 모델 W와 B는 Out put layer가 지정된 task를 잘 할수 있게 representation되도록 학습되어 짐

Output Layer의 w와 b는 지정된 task를 잘하도록 학습 되어 짐.

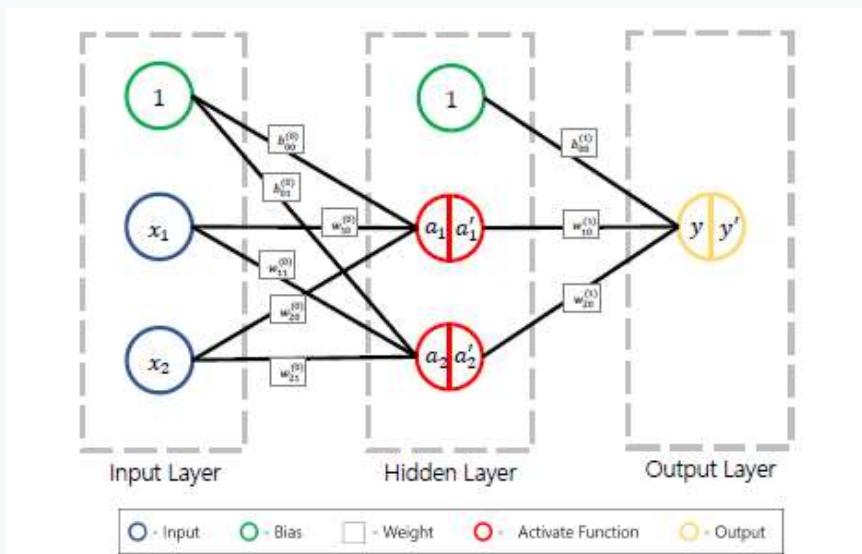
즉, 결론적으로 학습이 잘되면 각 layer들은 각각의 역할을 잘 수행한다고 해석함.

학습이 잘 안되면 위의 역할을 잘 수행 못한다고 해석함.

0. 복습

2. Multi Layer Perceptron

Multi Layer Perceptron 문제



1. 오비피팅

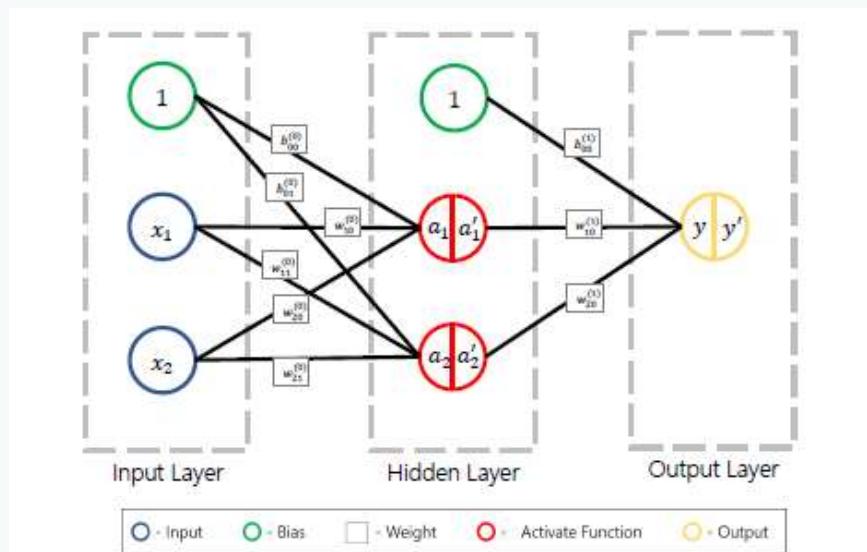
2. 느린 학습 속도

3. local minima

4. vanishing gradient

0. 복습

2. Multi Layer Perceptron



Multi Layer Perceptron 대표적 해결법

1. batch normalization

2. dropout

3. regularization

4. small model

5. data augmentation

6. 새로운 activation 함수

7. 좋은 optimizer

0. 복습

3. 학습에서 중요한 것

데이터를 모을 때 현장과 비슷하고 발생할 모든 케이스의 데이터를 많이 모아야함.

학습/검증/시험 데이터 나눠서 학습시 성능 확인을 하고 오버피팅 확인을 해야함

데이터를 나눌 때 분포가 고르게 되어야함

학습 모델 웨이트가 고루 잘 퍼져잇는게 일반적으로 잘 학습되었다고 함.

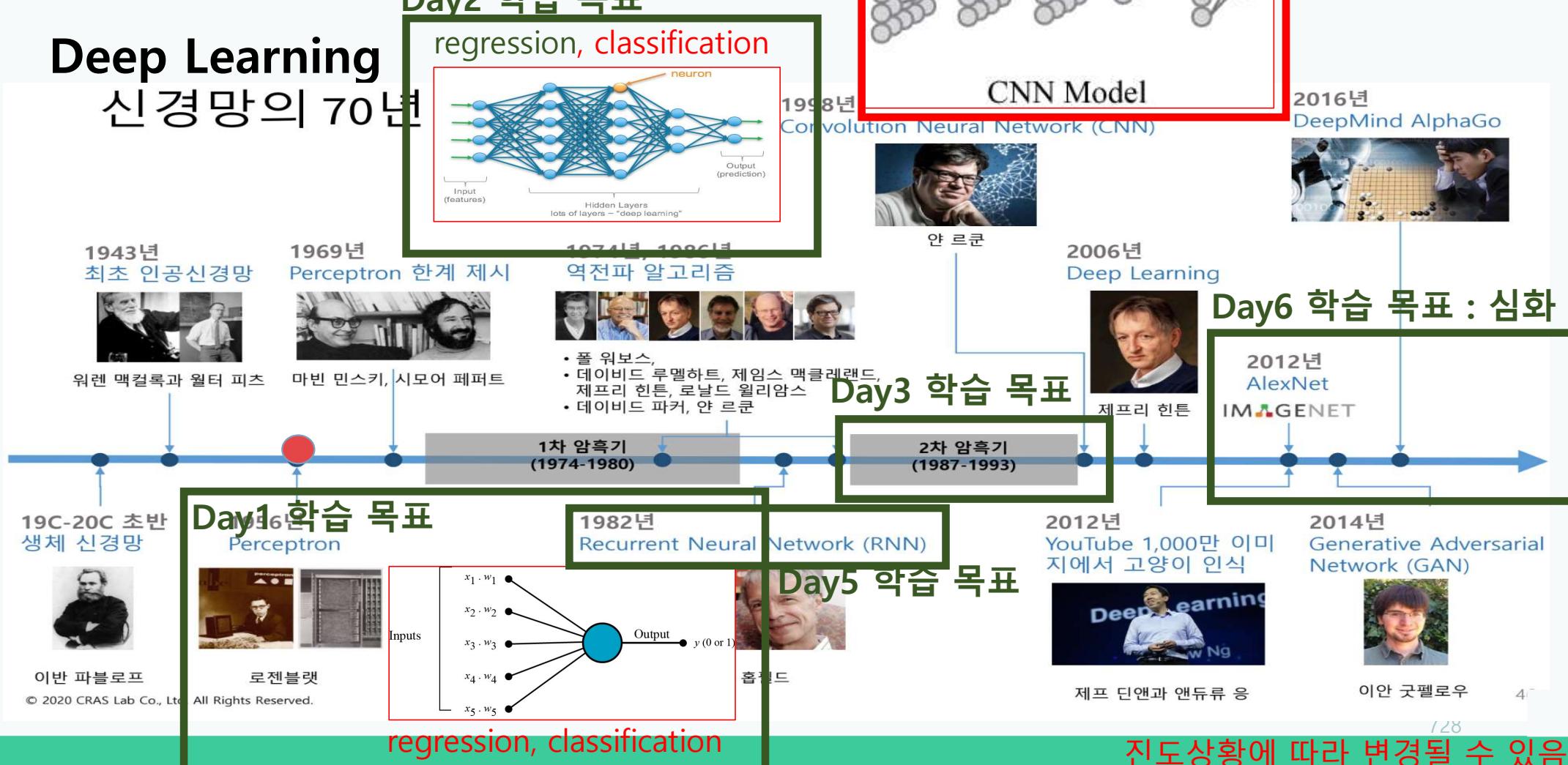
러닝 레이트, 모델 파라미터 초기값, 모델 설계, batch size, epoch 등이 중요

검증데이터는 학습시 모델 업데이트에 적용되지 않음

0. 오늘 하루 동안 무엇을

Deep Learning

신경망의 70년



0. 오늘 하루 동안 무엇을 하나요?

1. 이미지 Neural Network 실습

- NN 문제
- NN 해결
- MNIST DATA
- MNIST NN Classification
- MINIST NN Test
- NN 성능 한계

2. Convolution Neural Network

- Convolution Filter
- Max pooling
- CNN 장점

3. CNN 실습

- CNN 기반 MNIST 학습 및 테스트
- CNN 기반 binary classification
- CNN 기반 Multi classification
- Image data augmentation

1. 이미지 Neural Network 실습

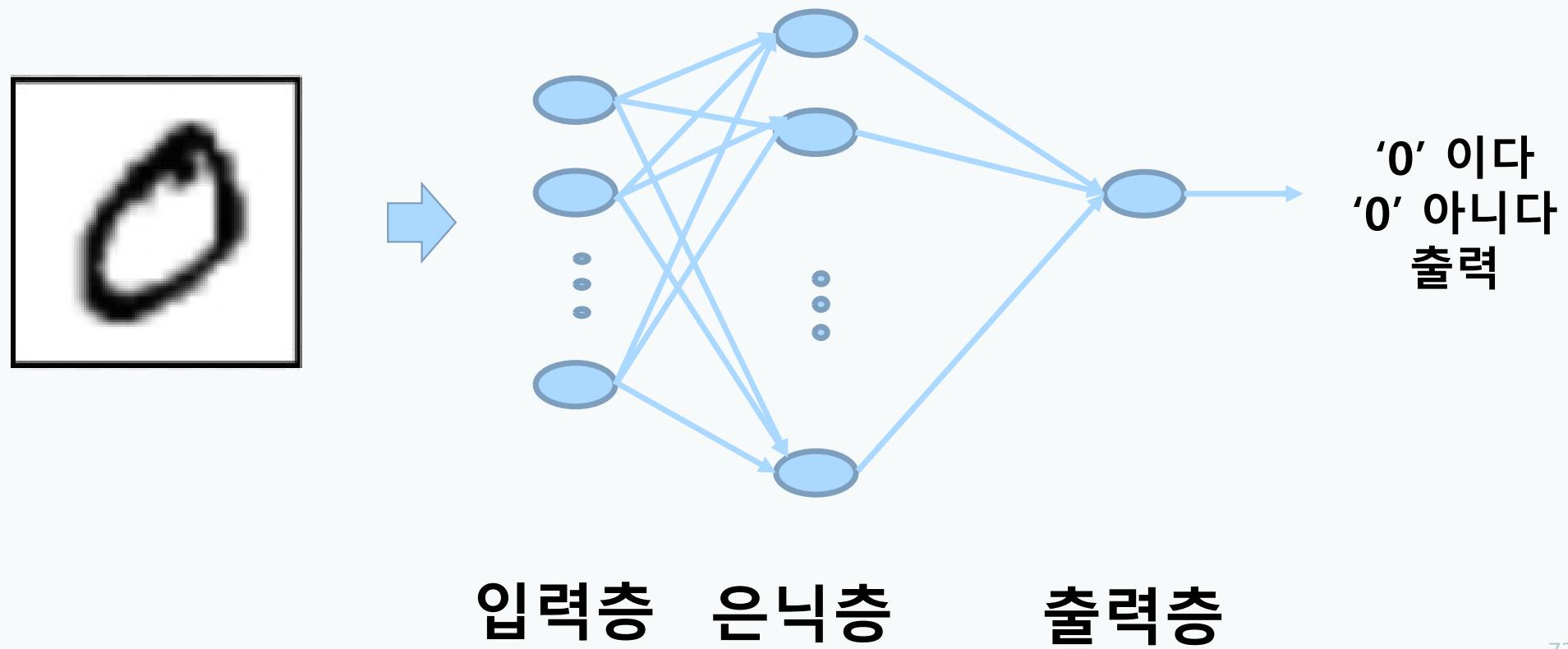
Neural Network image 분류

- MNIST dataset
- 0 ~ 9의 손 글씨 데이터
- 28 x 28 grayscale 이미지
- 6만장 학습 데이터, 1만장 테스트 데이터



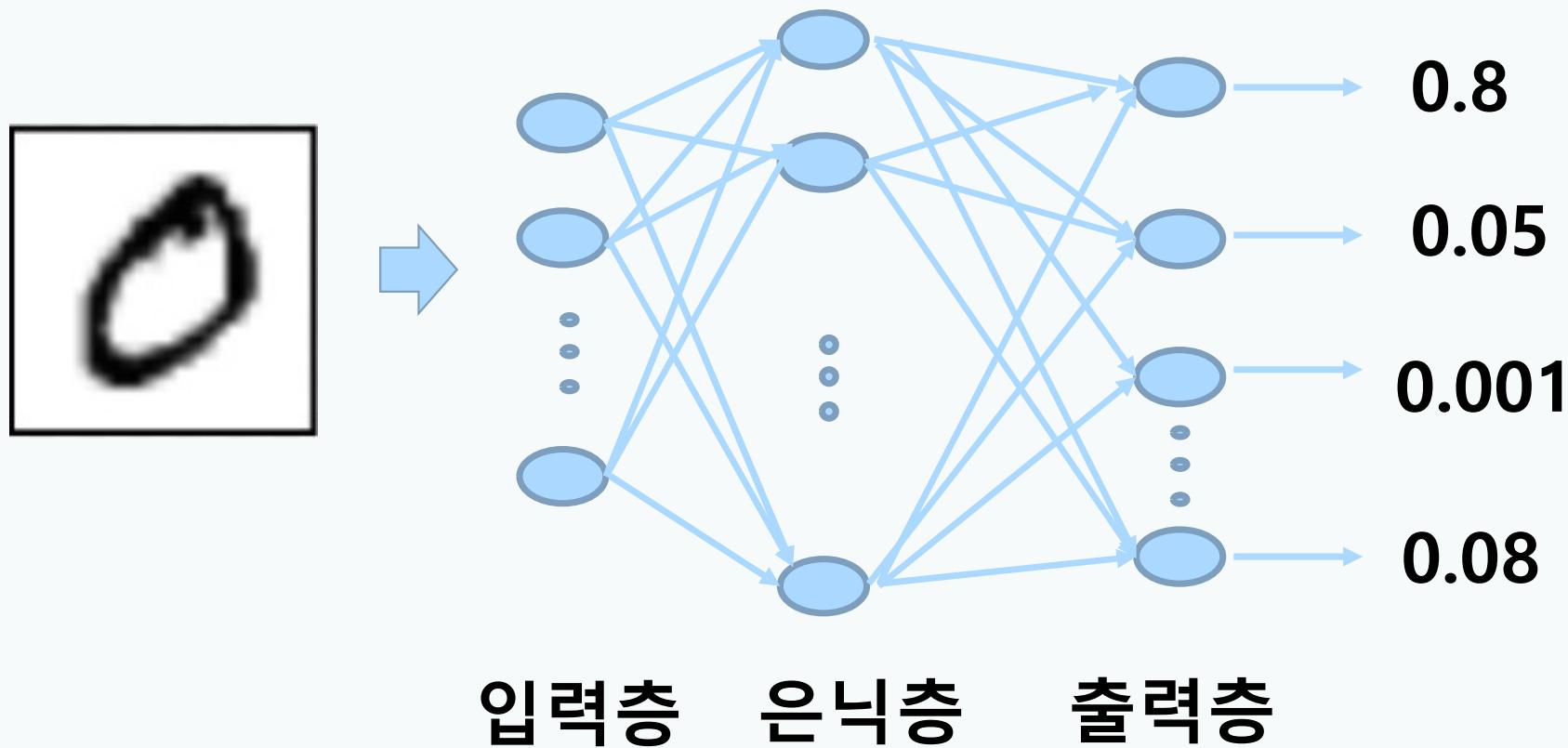
1. 이미지 Neural Network 실습

Neural Network image 분류



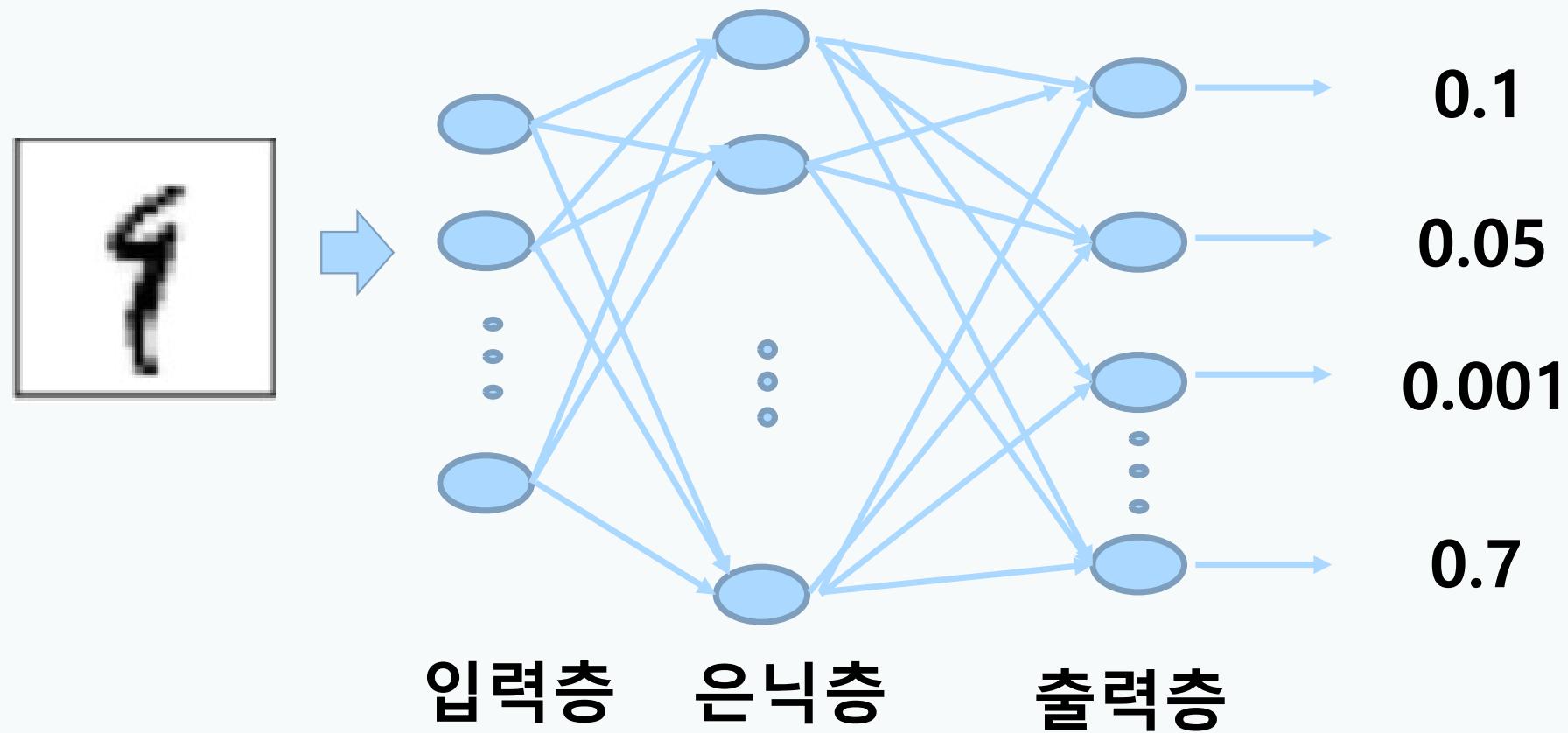
1. 이미지 Neural Network 실습

Neural Network image 분류



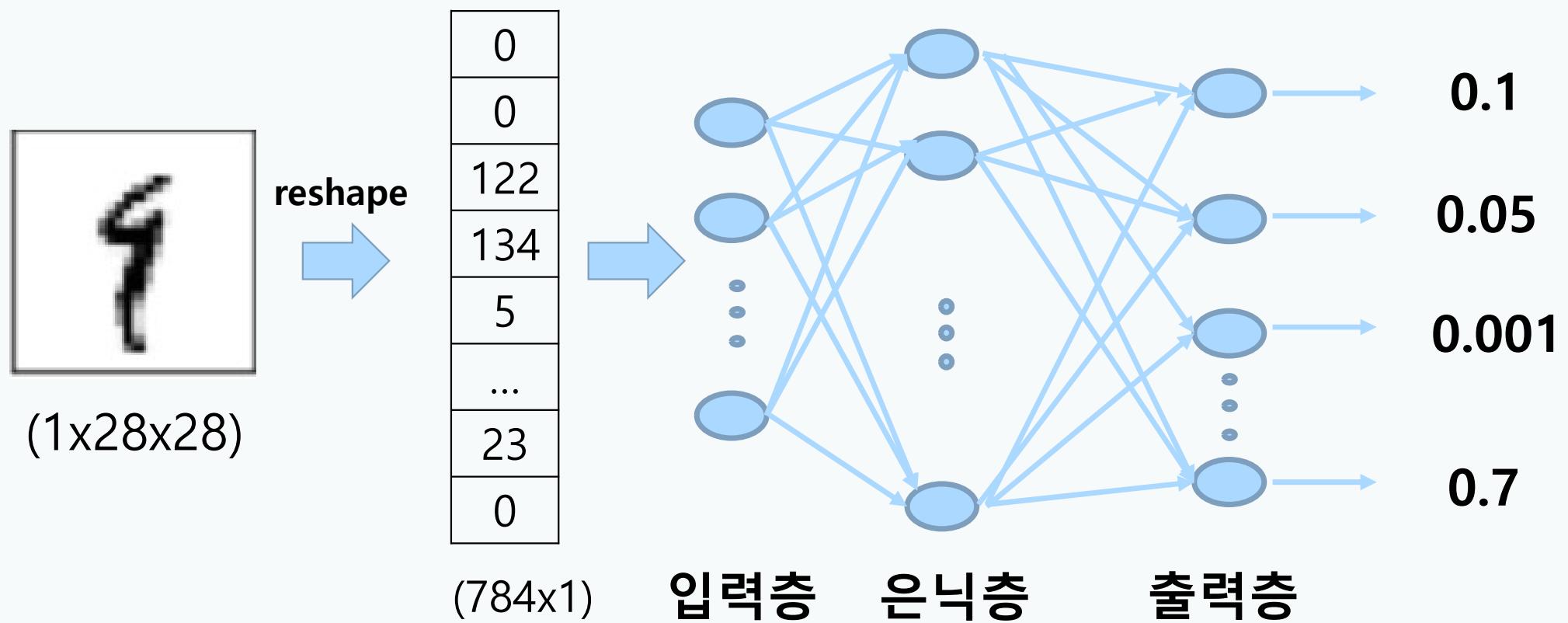
1. 이미지 Neural Network 실습

Neural Network image 분류



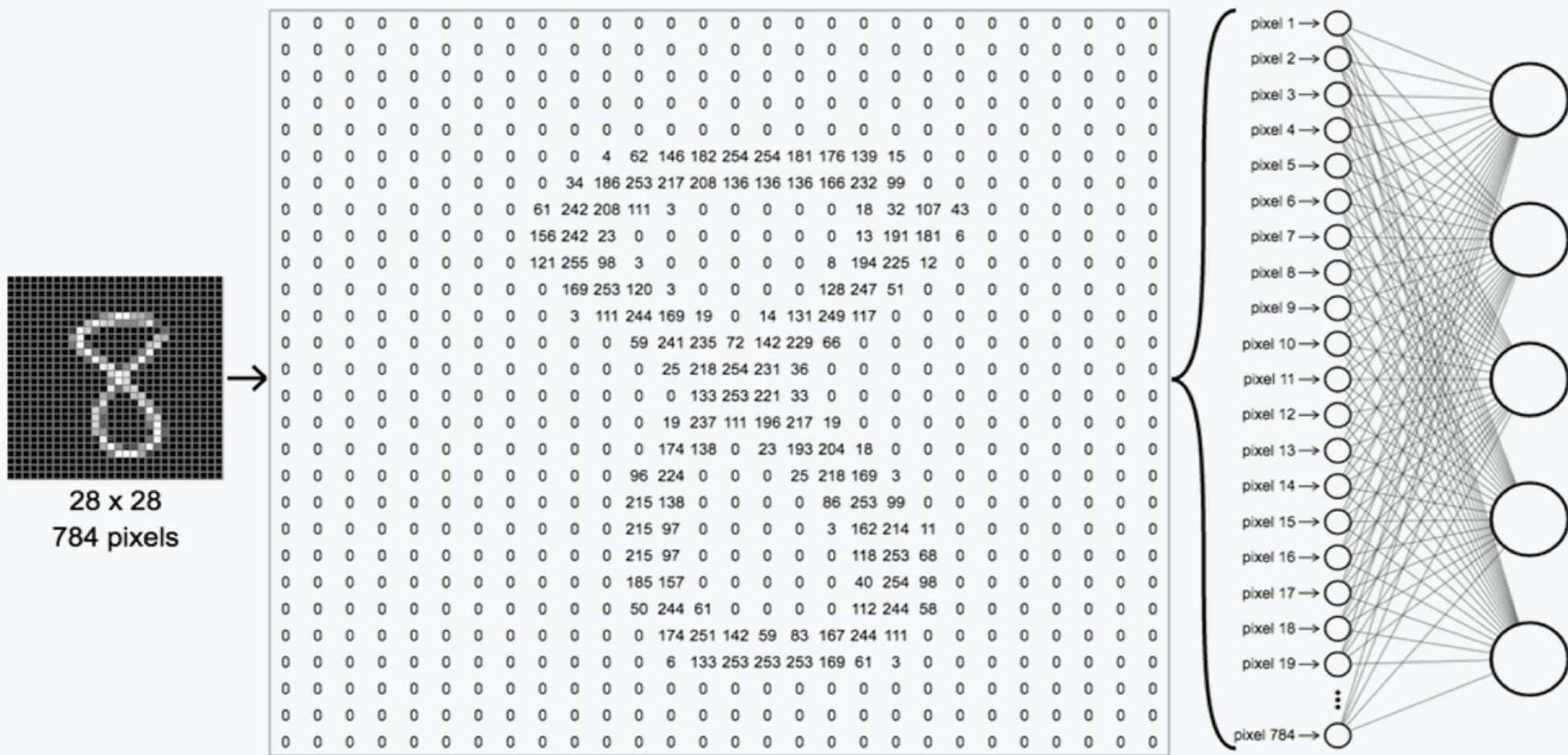
1. 이미지 Neural Network 실습

Neural Network image 분류



1. 이미지 Neural Network 실습

Neural Network image 분류



1. 이미지 Neural Network 실습

Neural Network image 분류

MNIST DATA를 확인해 보겠습니다.

1. 이미지 Neural Network 실습

Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그래프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 차원 변환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784 ) 1000은 이미지의 총 갯수.
23
24
25
26 X_train = X_train.astype('float64') ## types를 float 64로 변환
27 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
28
29 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
30
31 # 클래스 값 확인
32 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0]이므로 여기서도 [0]을 가져온다.
33
34 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
35 Y_train = tf.keras.utils.to_categorical(Y_class_train)
36 Y_test = tf.keras.utils.to_categorical(Y_class_test)
37 print(Y_train[0])
```

keras에서 data 가져오기
x_train shape : (60000, 28, 28) -> (이미지장수, 가로, 세로)
-> 3차원
-> 28x28 이미지 60000장

1. 이미지 Neural Network 실습

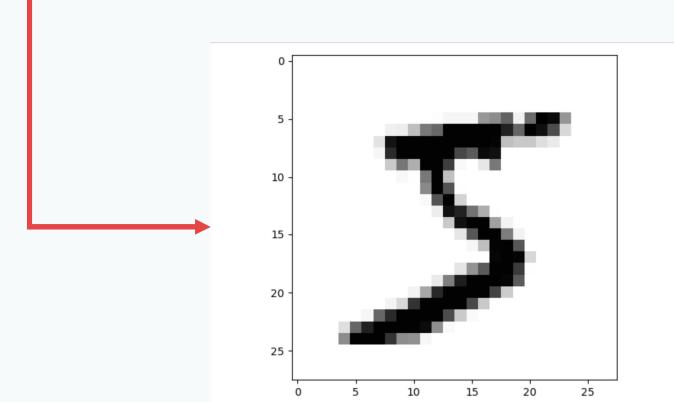
Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그레프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 차원 변환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784 ) 1000은 이미지의 총 갯수.
23
24
25
26 X_train = X_train.astype('float64') ## types를 float 64로 변환
27 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
28
29 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
30
31 # 클래스 값 확인
32 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0]이므로 여기서도 [0]을 가져온다.
33
34 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
35 Y_train = tf.keras.utils.to_categorical(Y_class_train)
36 Y_test = tf.keras.utils.to_categorical(Y_class_test)
37 print(Y_train[0])
```

keras에서 data 가져오기

x_train shape : (60000, 28, 28) -> (이미지장수, 가로, 세로)
-> 3차원

-> 28x28 이미지 60000장



X_train 이미지 60000장 중에 0번째 이미지 가져와서
display

1. 이미지 Neural Network 실습

Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그래프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 차원 변환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784) 1000은 이미지의 총 갯수.
23
24
25
26 X_train = X_train.astype('float64') ## types를 float 64로 변환
27 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
28
29 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
30
31 # 클래스 값 확인
32 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0]이므로 여기서도 [0]을 가져온다.
33
34 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
35 Y_train = tf.keras.utils.to_categorical(Y_class_train)
36 Y_test = tf.keras.utils.to_categorical(Y_class_test)
37 print(Y_train[0])
```

keras에서 data 가져오기
x_train shape : (60000, 28, 28) -> (이미지장수,가로,세로)
-> 3차원
-> 28x28 이미지 60000장

1. 이미지 Neural Network 실습

Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그래프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 자원 복환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784 ) 1000은 0이지의 총 갯수.
23
24
25 X_train = X_train.astype('float64') ## types를 float 64로 변환
26 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
27
28
29 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
30
31 # 클래스 값 확인
32 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0]이므로 여기서도 [0]을 가져온다.
33
34 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
35 Y_train = tf.keras.utils.to_categorical(Y_class_train)
36 Y_test = tf.keras.utils.to_categorical(Y_class_test)
37 print(Y_train[0])
```

keras에서 data 가져오기
x_train shape : (60000, 28, 28) -> (이미지장수,가로,세로)
-> 3차원
-> 28x28 이미지 60000장

이미지 장수, 이미지가로, 이미지 세로 -> 이미지장수, 이미지 데이터

1. 이미지 Neural Network 실습

Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그레프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 차원 변환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784 ) 1000은 0~255의 총 갯수.
23
24
25 X_train = X_train.astype('float64') ## types를 float 64로 변환
26 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
27
28 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
29
30
31 # 클래스 값 확인
32 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0]이므로 여기서도 [0]을 가져온다.
33
34 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
35 Y_train = tf.keras.utils.to_categorical(Y_class_train)
36 Y_test = tf.keras.utils.to_categorical(Y_class_test)
37 print(Y_train[0])
```

keras에서 data 가져오기
x_train shape : (60000, 28, 28) -> (이미지장수,가로,세로)
-> 3차원
-> 28x28 이미지 60000장

이미지는 0~255 사이값을 가지는데
255로 나누어서 0~1사이 값으로 normalization

1. 이미지 Neural Network 실습

Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그레프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 차원 변환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784 ) 1000은 0~255의 총 갯수.
23
24
25 X_train = X_train.astype('float64') ## types를 float 64로 변환
26 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
27
28 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
29
30
31 # 클래스 값 확인
32 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0]이므로 여기서도 [0]를 가져온다.
33
34 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
35 Y_train = tf.keras.utils.to_categorical(Y_class_train)
36 Y_test = tf.keras.utils.to_categorical(Y_class_test)
37 print(Y_train[0])
```

keras에서 data 가져오기
x_train shape : (60000, 28, 28) -> (이미지장수,가로,세로)
-> 3차원
-> 28x28 이미지 60000장

이미지는 0~255 사이값을 가지는데
255로 나누어서 0~1사이 값으로 normalization

Y_class_train shape : (60000,) -> 정답 개수
-> 1차원

첫번째 저장된 정답 값 가져오기

practice : P_04_00_MNIST_DATA.ipynb 742

1. 이미지 Neural Network 실습

Neural Network image 분류

```
5 (X_train, Y_class_train), (X_test, Y_class_test) = tf.keras.datasets.mnist.load_data()
6 print(X_train.shape)
7 print("학습셋 이미지 수 : %d 개" % (X_train.shape[0]))
8 print("테스트셋 이미지 수 : %d 개" % (X_test.shape[0]))
9
10 # 그레프로 확인
11 import matplotlib.pyplot as plt
12 plt.imshow(X_train[0], cmap='Greys')
13 plt.show()
14
15 # 코드로 확인
16 for x in X_train[0]:
17     for i in x:
18         sys.stdout.write('%d\t' % i)
19     sys.stdout.write('\n')
20
21 # 차원 변환 과정
22 X_train = X_train.reshape(X_train.shape[0], 784) # (6000 x 28 x 28) -> (6000 x 784 ) 1000은 0~255의 총 갯수.
23
24 X_train = X_train.astype('float64') ## types를 float 64로 변환
25 X_train = X_train / 255 ## 0~255 값으로 노말라이즈 해주기
26
27 X_test = X_test.reshape(X_test.shape[0], 784).astype('float64') / 255
28
29
30 # 클래스 값 확인
31 print("class : %d " % (Y_class_train[0])) ## 이미지 하나 확인하는게 [0] 이므로 여기서도 [0]를 가져온다.
32
33 # 바이너리화 과정 0 ->[1 0 0 0 0 0 0 0]
34 Y_train = tf.keras.utils.to_categorical(Y_class_train)
35 Y_test = tf.keras.utils.to_categorical(Y_class_test)
36
37 print(Y_train[0])
```

keras에서 data 가져오기
x_train shape : (60000, 28, 28) -> (이미지장수,가로,세로)
-> 3차원
-> 28x28 이미지 60000장

이미지는 0~255 사이값을 가지는데
255로 나누어서 0~1사이 값으로 normalization

Y_class_train shape : (60000,) -> 정답 개수
-> 1차원

첫번째 저장된 정답 값 가져오기

One hot encoding

1. 이미지 Neural Network 실습

Neural Network image 분류

MNIST DATA로 classification을 해보겠습니다.

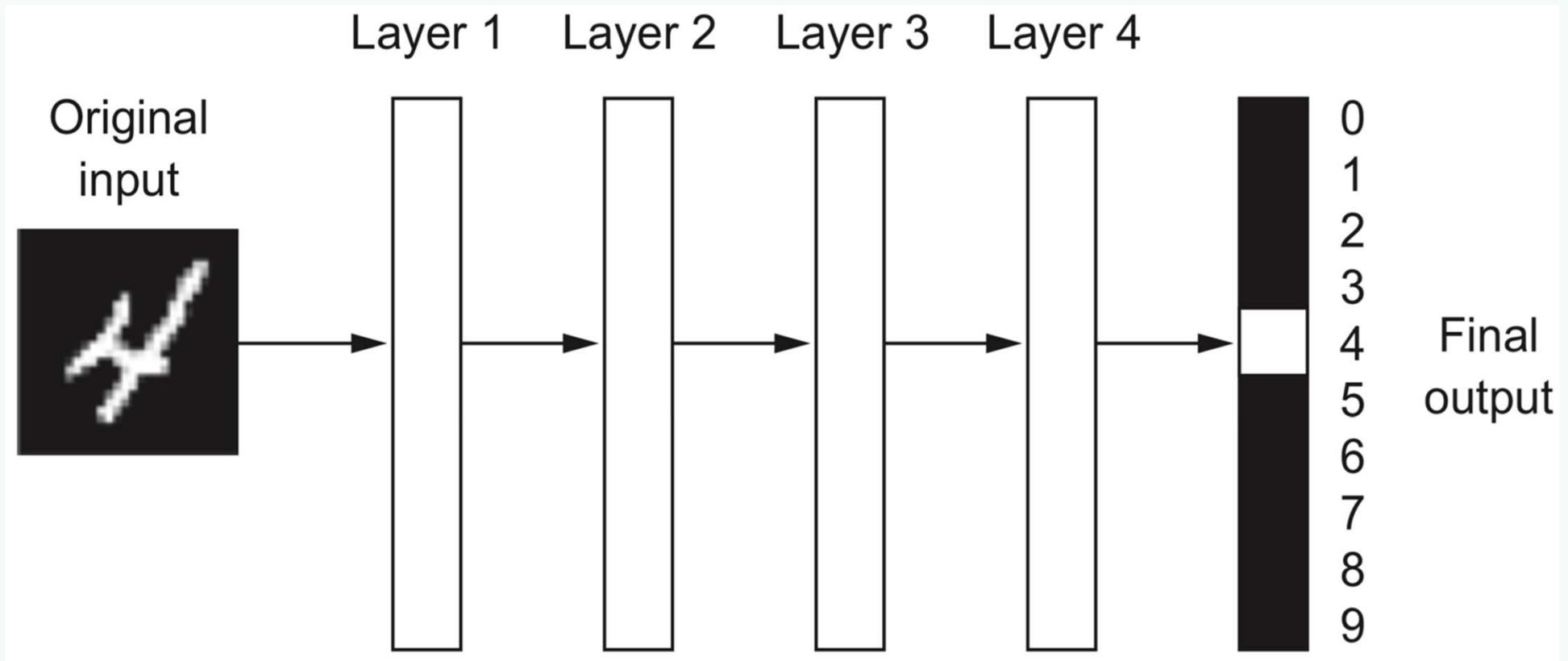
1. 이미지 Neural Network 실습

Neural Network image 분류

어떻게 모델을 구성하면 될까요~?

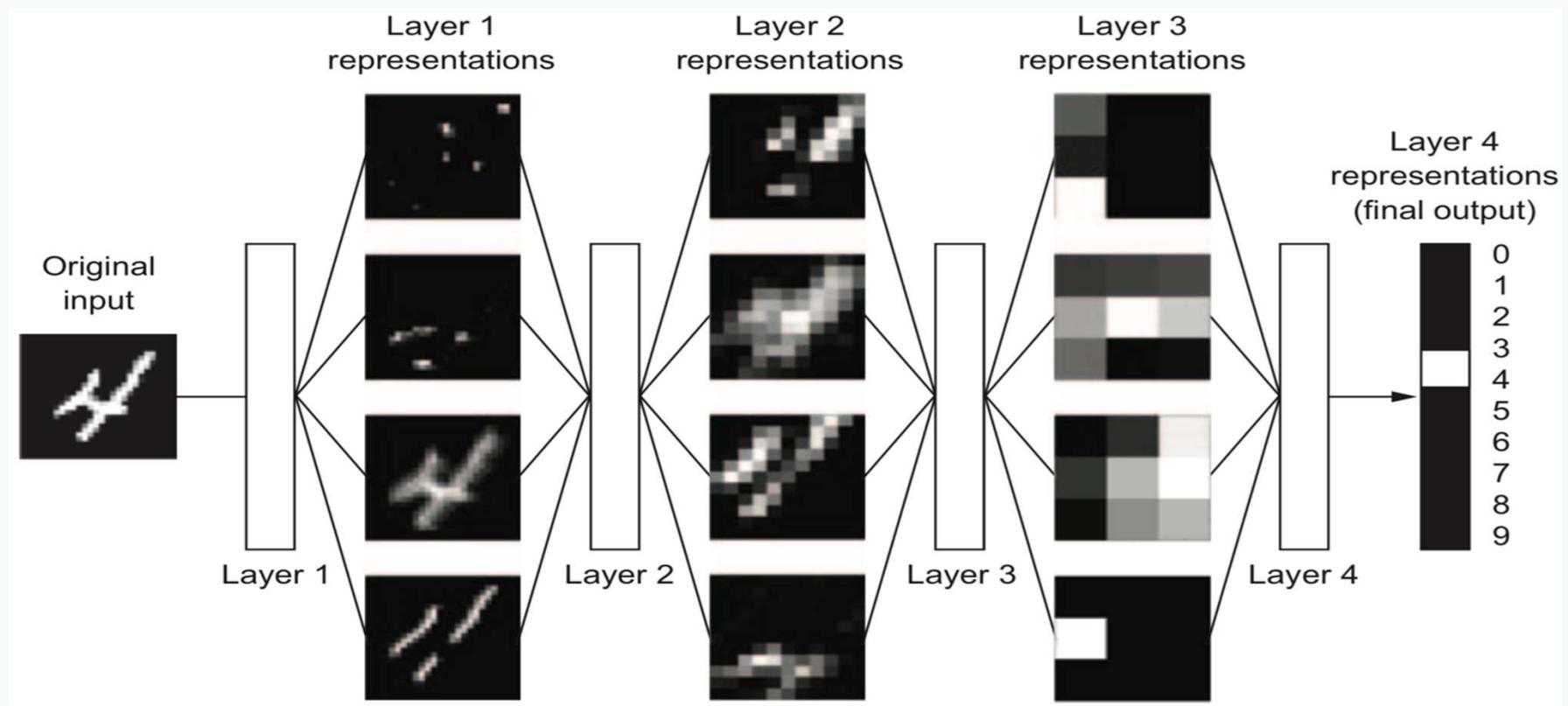
1. 이미지 Neural Network 실습

Neural Network image 분류



1. 이미지 Neural Network 실습

Neural Network image 분류



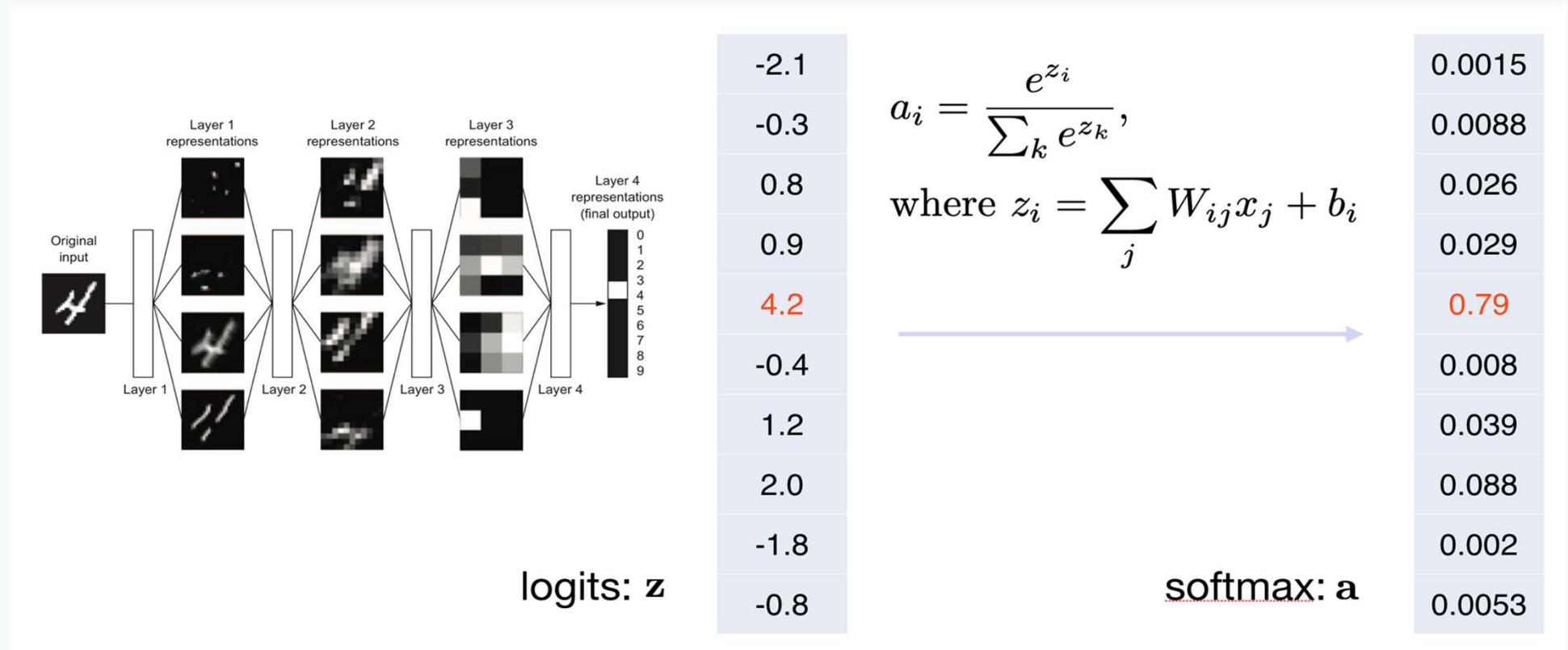
1. 이미지 Neural Network 실습

Neural Network image 분류

loss 함수는 무엇을 쓰면 될까요??

1. 이미지 Neural Network 실습

Neural Network image 분류



1. 이미지 Neural Network 실습

Neural Network image 분류

	prediction	label
	0.0015	0
	0.0088	0
	0.026	0
	0.029	0
	0.79	1
	0.008	0
	0.039	0
	0.088	0
	0.002	0
	0.0053	0

Mean Squared Error (MSE)

$$\mathcal{L} \equiv \frac{1}{2N} \sum_i ||\hat{\mathbf{y}}_i - \mathbf{y}_i||^2 = \frac{1}{2N} \sum_i \sum_k (\hat{y}_{ik} - y_{ik})^2$$

Cross Entropy Loss

$$\mathcal{L} \equiv -\frac{1}{N} \sum_i \mathbf{y}_i \log \hat{\mathbf{y}}_i = -\frac{1}{N} \sum_i \sum_k y_{ik} \log \hat{y}_{ik}$$

label: y

prediction: \hat{y}

dimension: k

1. 이미지 Neural Network 실습

Neural Network image 분류

minist 데이터 기반으로 0~9를
분류하는 모델을 작성해 보겠습니다.

1. 이미지 Neural Network 실습

Neural Network image 분류

minist 데이터 기반으로 0~9를
분류하는 모델을 짜보겠습니다.
여기서 잠깐!!

Neural Network 설계시 입력 차원이 어떻게 되었나요?

1. 이미지 Neural Network 실습

Neural Network image 분류

미리 데이터셋으로 (None, 3)를
분류하는 모델을 구현해 보겠습니다.
즉, 배치사이즈를 제외하면 1차원 입니다.

1. 이미지 Neural Network 실습

Neural Network image 분류

그런데 Gray 이미지는 2차원입니다.

분류하는 모델은 가로 x 세로 보겠습니다.

그러므로 `reshape`를 통해 2차원을 1차원으로 바꿔주고

그것을 입력 `shape`과 맞춰줘야 합니다

1. 이미지 Neural Network 실습

Neural Network image 분류

```
# MNIST 데이터 불러오기
(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()

X_train = X_train.reshape(X_train.shape[0], 784).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 784).astype('float32')

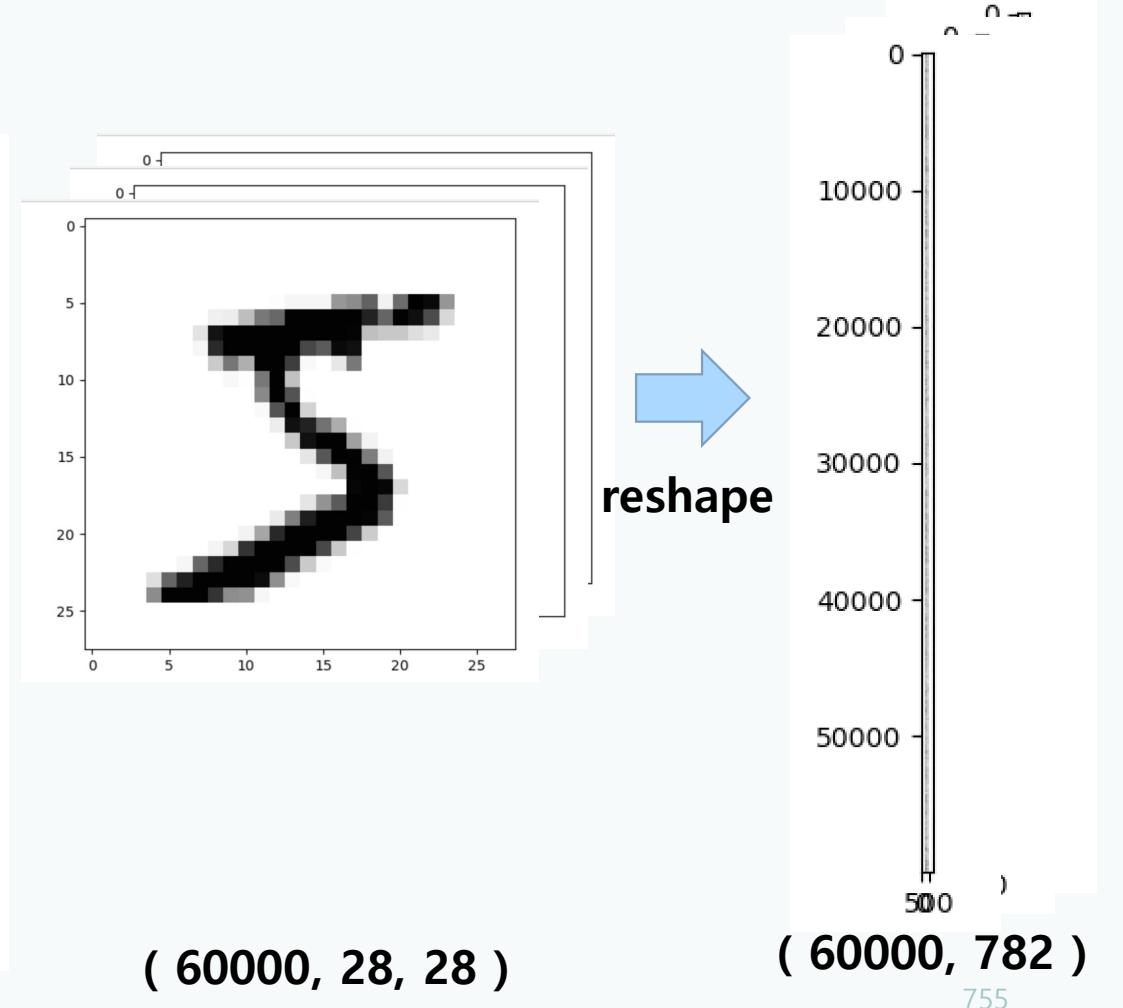
Y_train = tf.keras.utils.to_categorical(Y_train, 10)
Y_test = tf.keras.utils.to_categorical(Y_test, 10)

# 모델 설계
# method 1
input_Layer = tf.keras.layers.Input(shape=(784,))
x = tf.keras.layers.Dense(512, activation='relu')(input_Layer)
Out_Layer= tf.keras.layers.Dense(10, activation='softmax')(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()

# 모델 컴파일
loss=tf.keras.losses.categorical_crossentropy
optimizer = tf.keras.optimizers.Adam(lr=0.001)
model.compile(loss=loss,
              optimizer=optimizer,
              metrics=['accuracy'])


```



1. 이미지 Neural Network 실습

Neural Network image 분류

```
# MNIST 데이터 불러오기
(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()

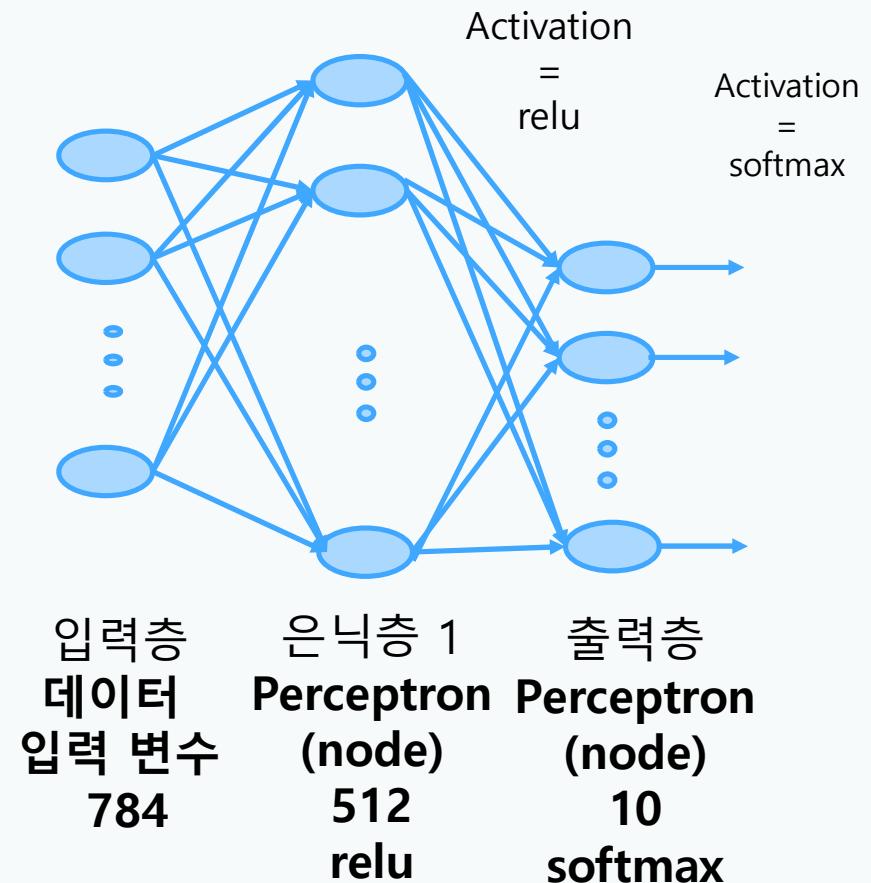
X_train = X_train.reshape(X_train.shape[0], 784).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 784).astype('float32')

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

#모델 설계
# method 1
input_Layer = tf.keras.layers.Input(shape=784))
x = tf.keras.layers.Dense(512, activation='relu')(input_Layer)
Out_Layer= tf.keras.layers.Dense(10, activation='softmax')(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()
```

```
# 모델 컴파일
loss=tf.keras.losses.categorical_crossentropy
optimizer = tf.keras.optimizers.Adam(lr=0.001)
model.compile(loss=loss,
              optimizer=optimizer,
              metrics=['accuracy'])
```



1. 이미지 Neural Network 실습

Neural Network image 분류

지금까지는 모델을 학습하고 평가하여 수치적으로만 확인.
그렇다면 이미지를 입력으로 넣어서 결과를 visual로 확인하고 싶다면
어떻게 해야 할까요?

1. 이미지 Neural Network 실습

Neural Network image 분류

```
# MNIST 데이터 불러오기  
(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()
```

```
## TEST할 이미지 선택  
test_image = X_test[0] ## image matrix 곱
```

60000장의 이미지 중
테스트할 1장의 이미지 선택

```
## NN 이미로 이차원으로 데이터를 넣어주어야 해서 1x784 형태로 reshape하고 노말라이제이션  
test_image_reshape = test_image.reshape(1,784).astype('float64') ## 이미지가 2-d이므로 1-d로 변환하여 nn으로 전달
```

NN 입력 차원 맞춰주기

```
## 모델 불러오기  
model = tf.keras.models.load_model('./MNIST_model\\15-0.4875.hdf5') # 모델을 새로 불러옴
```

모델 불러오기

```
Y_prediction = model.predict(test_image_reshape)
```

모델 입력해서 출력 값 받기

```
## 10개의 class가 각 확률 값으로 나오기 때문에 가장 높은값을 출력하는 인덱스를 추출. 그럼 이것이 결국 class임.  
### np.argmax는 들어온 행렬에서 가장 높은값이 있는 index를 반환해주는 함수.
```

출력 값중에 가장 큰값을 가지는
index 가져오기

```
index=np.argmax(Y_prediction)  
vlaue=Y_prediction[:, index]  
plt.imshow(test_image, cmap='Greys')  
plt.xlabel(str(index)+" "+str(vlaue))  
plt.show()
```

(가장 활성화된 perceptron 위치)
가져온 index로 perceptron의
출력값 가져오기

1. 이미지 Neural Network 실습

Neural Network image 분류

```
# MNIST 데이터 불러오기
(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()

# TEST할 이미지 선택
test_image = X_test[0] ## image matrix

# NN 입력 이미지 차원으로
test_image_reshape = test_image.reshape(1,784).astype('float64') ## 이미지가 2-d이므로 1-d로 변환하여 깊이로 전달

# 모델 불러오기
model = tf.keras.models.load_model('./MNIST_model\\15-0.4875.hdf5') # 모델을 새로 불러옴

# 모델 예측
Y_prediction = model.predict(test_image_reshape)

# 10개의 class가 각 위치를 나누기 때문에 가장 높은값을 출력하는 인덱스를 추출하고 그에 맞는 class는?
index=np.argmax(Y_prediction)
vlaue=Y_prediction[:, index]

plt.imshow(test_image, cmap='Greys')
plt.xlabel(str(index)+" "+str(vlaue))
plt.show()
```

그런데 입력 이미지가 과연 올바른 방향으로만 들어 올까요?
회전되어 들어오는 경우가 있습니다.

그러면 모델 테스트를 할때 이미지를 회전해서 넣어보고 싶을때는
어떻게 해야할까요?

60000장의 이미지 중
테스트할 1장의 이미지 선택

NN 입력 차원 맞춰주기

모델 불러오기

모델 입력해서 출력 값 받기

(출력 값중에 가장 큰값을 가지는
index 가져오기)

(가장 활성화된 perceptron 위치)
가져온 index로 perceptron의

출력값 가져오기

1. 이미지 Neural Network 실습

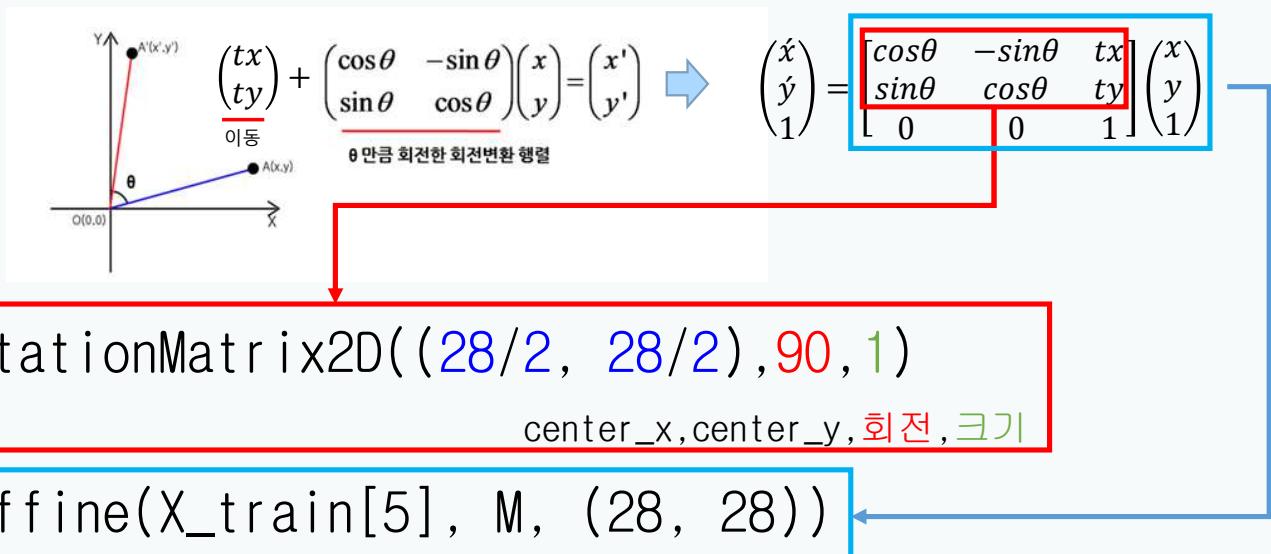
Neural Network image 분류

opencv / python-opencv 를 pip으로 설치하여
(conda cmd 창에서 [pip install opencv-python](#))
import cv2 하여
이미지 데이터를 다루도록 하겠습니다.
(opencv는 영상 데이터를 다루고 이미지 처리 할 수 있는 패키지)

1. 이미지 Neural Network 실습

Neural Network image 분류

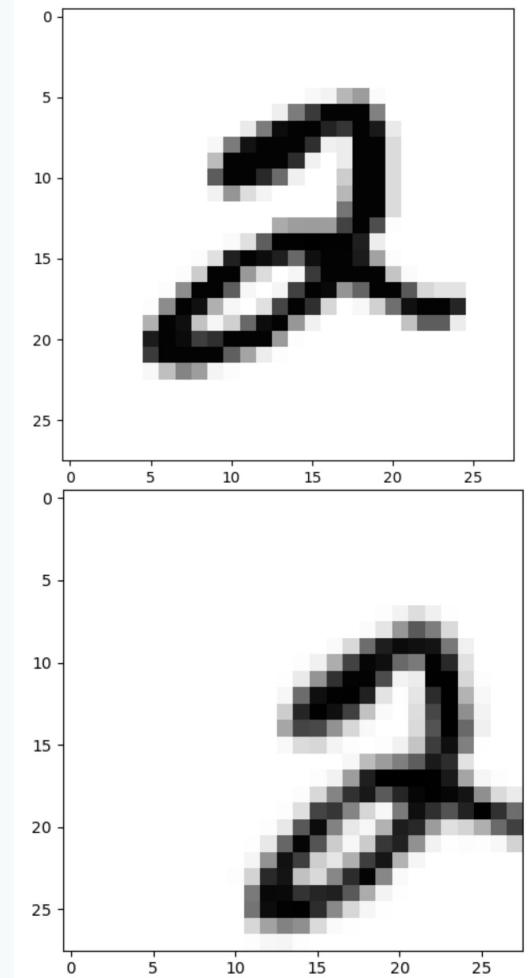
cv2로 이미지 회전시키기



1. 이미지 Neural Network 실습

Neural Network image 분류

```
8 # MNIST 데이터 불러오기
9 (X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()
10
11 ## 이미지 회전 변환 매트릭스 구하기
12 M= cv2.getRotationMatrix2D((28/2, 28/2), 10, 1) ## Matrix 생성
13 ## 이미지 회전 변환 매트릭스 적용
14 M[0, 2]=M[0, 2]+5
15 M[1, 2]=M[1, 2]+3
16
17 ## 이미지 변환 매트릭스 적용
18 test_image = cv2.warpAffine(X_train[5], M, (28, 28)) ## image // matrix 곱
19
20 ## NN 이미로 이차원으로 데이터를 넣어야 해서 1x784 형태로 reshape하고 노말라이제이션
21 test_image_reshape = test_image.reshape(1,784).astype('float64') ## 이미지가 2-d이므로 1-d로 변환하여!
22
23 ## 모델 불러오기
24 model = tf.keras.models.load_model('./MNIST_model\\15-0.4875.hdf5') # 모델을 새로 불러옴
25 # 불러온 모델로 값 예측하기.
26 Y_prediction = model.predict(test_image_reshape)
27
28 ## 10개의 class가 각 확률 값으로 나오기 때문에 가장 높은값을 출력하는 인덱스를 추출. 그럼 이것이 결국 class
29 ##### np.argmax는 들어온 행렬에서 가장 높은값이 있는 index를 반환해주는 함수.
30 index=np.argmax(Y_prediction)
31 vlaue=Y_prediction[:, index]
32 plt.imshow(test_image, cmap='Greys')
33 plt.xlabel(str(index)+" "+str(vlaue))
34 plt.show()
```



762

Exersice : 04_03_NN_MNIST_rotation.ipynb practice : P_04_03_NN_MNIST_rotation.ipynb

1. 이미지 Neural Network 실습

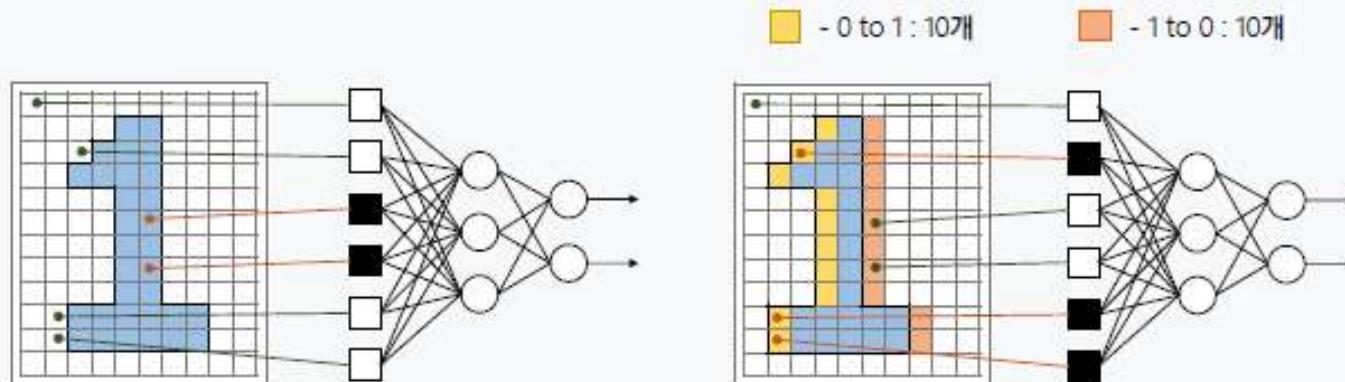
Neural Network image 분류

성능이 저하되거나
잘못된 결과를 출력하기도 합니다.
왜 이럴까요~?

1. 이미지 Neural Network 실습

Neural Network image 분류

MLP 의 문제점

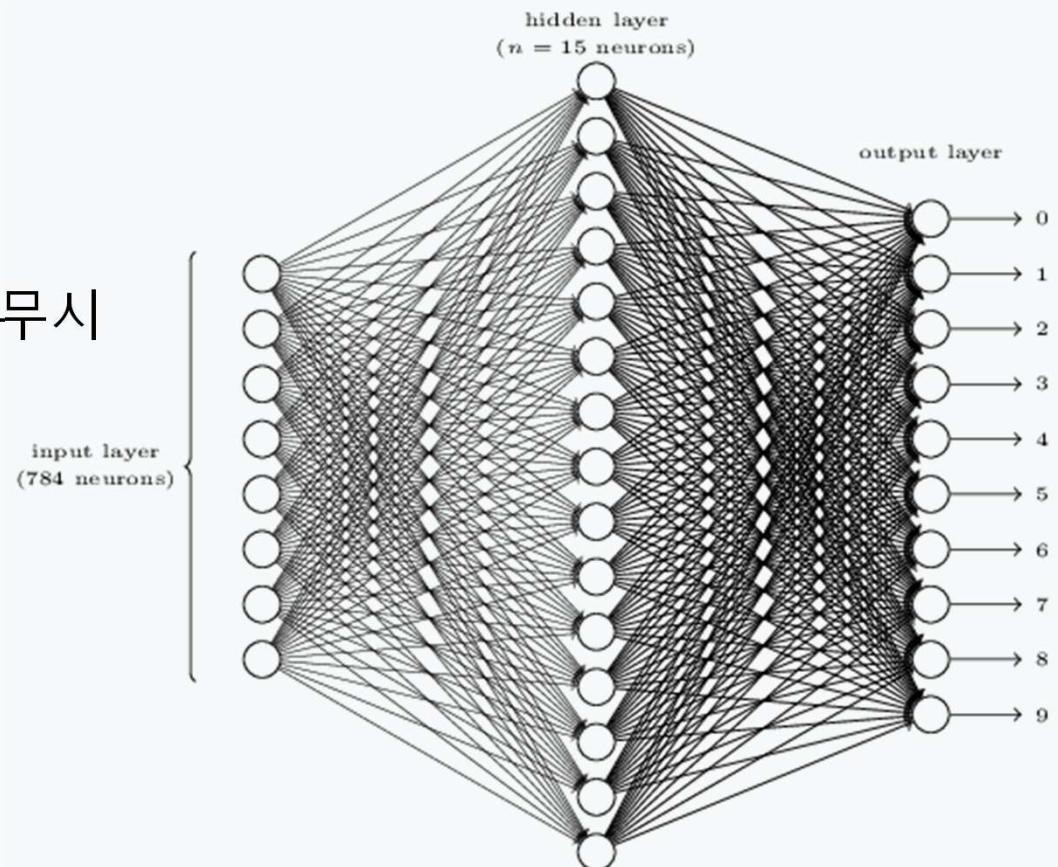


한 칸씩만 움직였는데
변화하는 인풋값이 20개

1. 이미지 Neural Network 실습

Neural Network image 분류

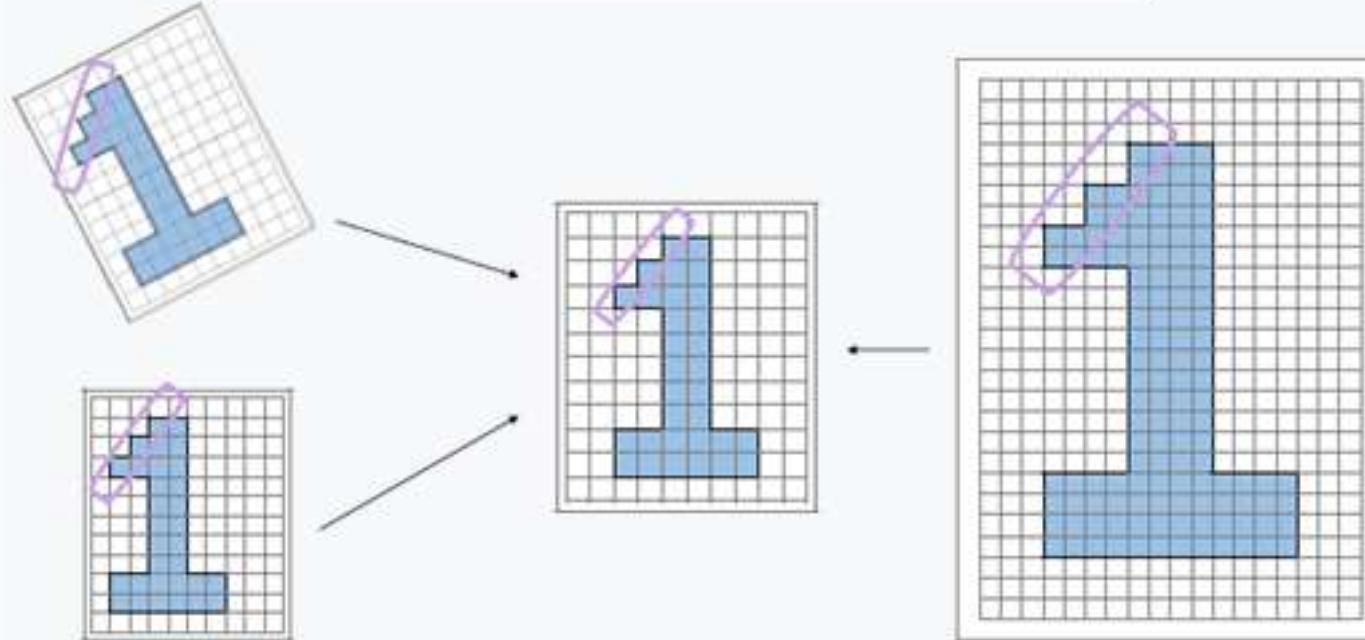
- Fully connected layers의 단점
 - 데이터 형상의 무시
이미지의 공간적(spatial)한 정보 무시
 - 학습해야 할 가중치(W)가 많다



1. 이미지 Neural Network 실습

image 특징점

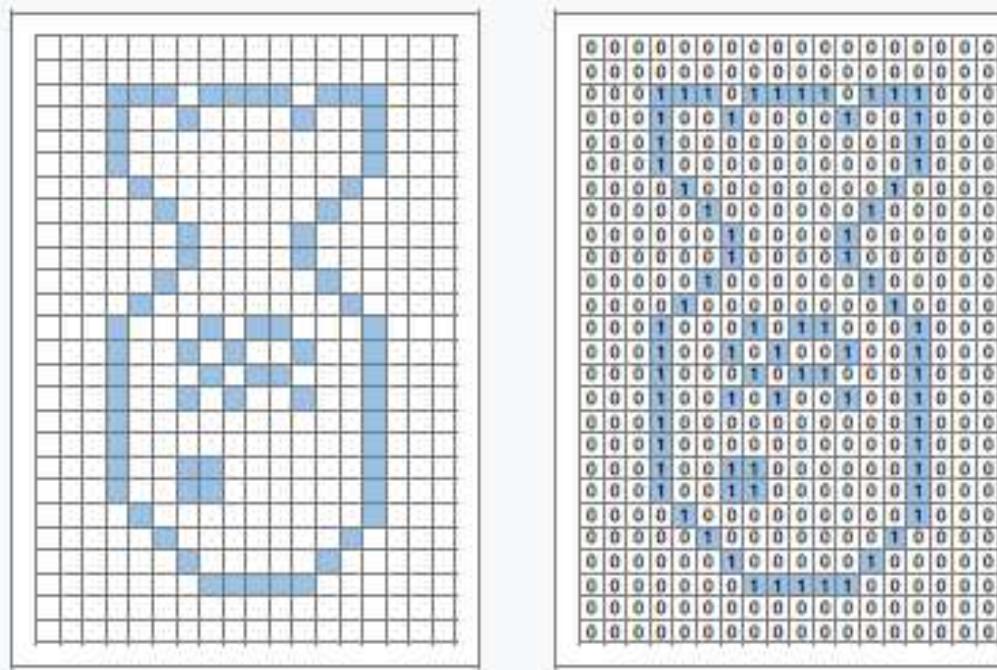
특징을 추출해내서 변화에 대응하자



1. 이미지 Neural Network 실습

image 특징점

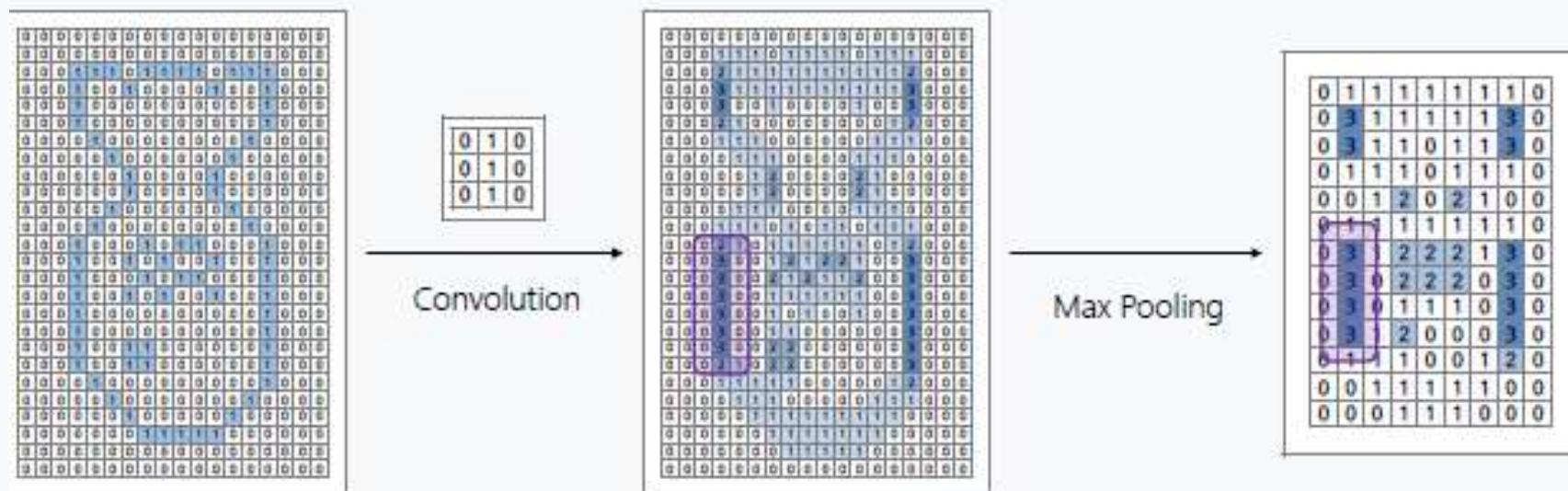
생선 그림을 가지고 CNN 과정을 살펴 봅시다



1. 이미지 Neural Network 실습

image 특징점

Convolution : 세로 필터

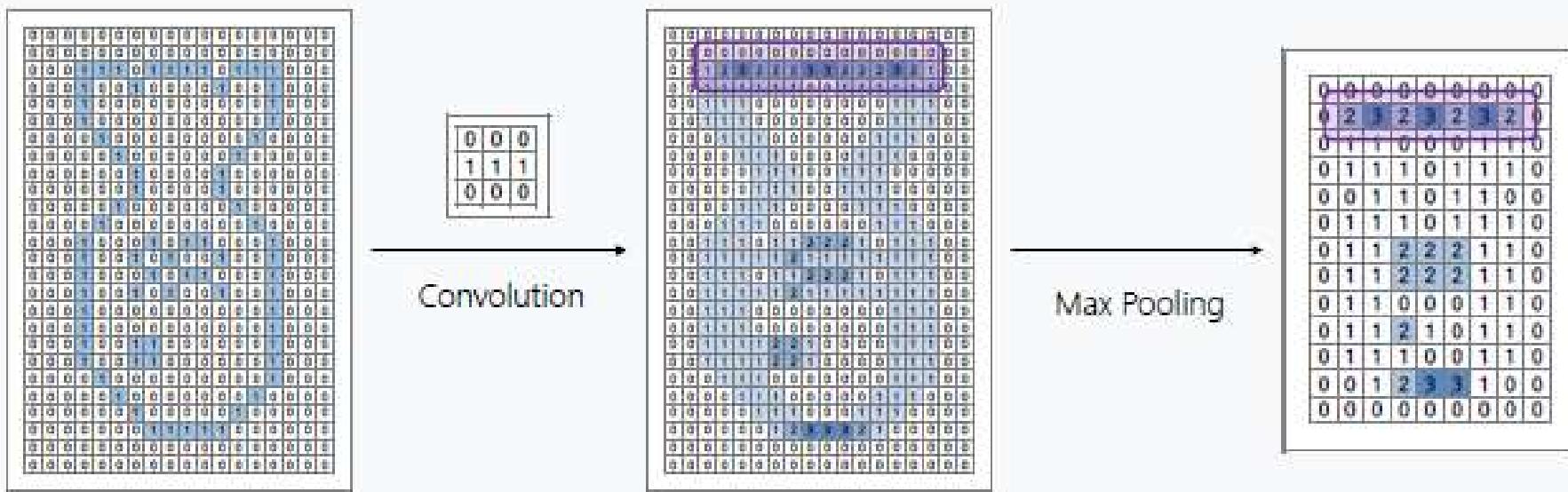


Max Pooling

1. 이미지 Neural Network 실습

image 특징점

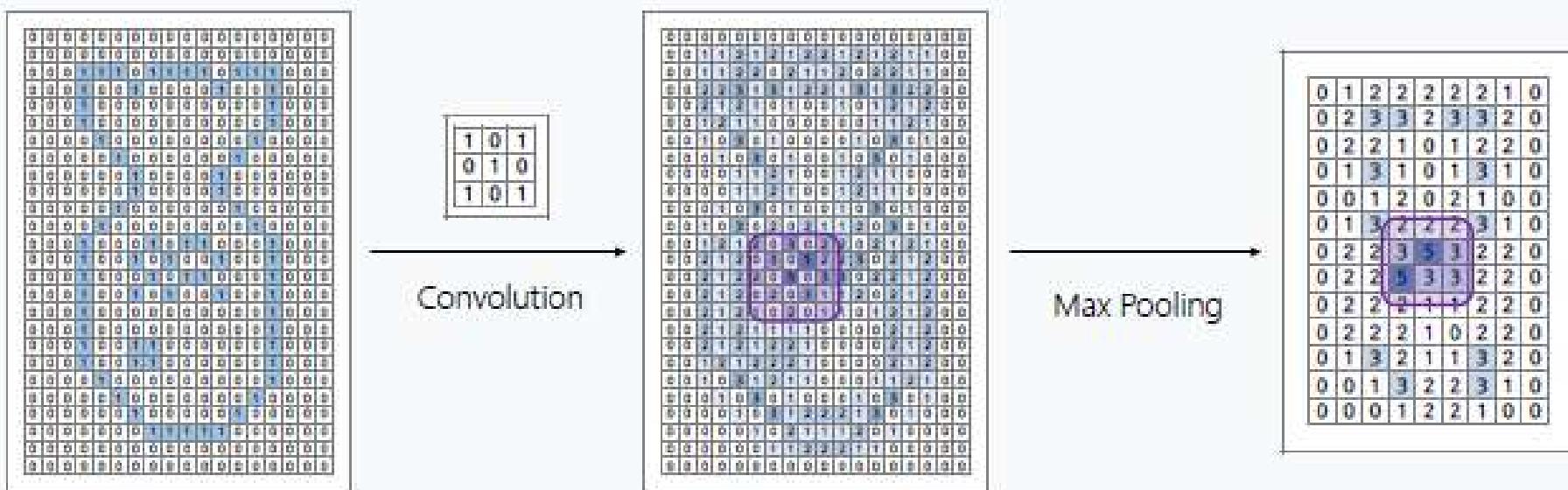
Convolution : 가로 필터



1. 이미지 Neural Network 실습

image 특징점

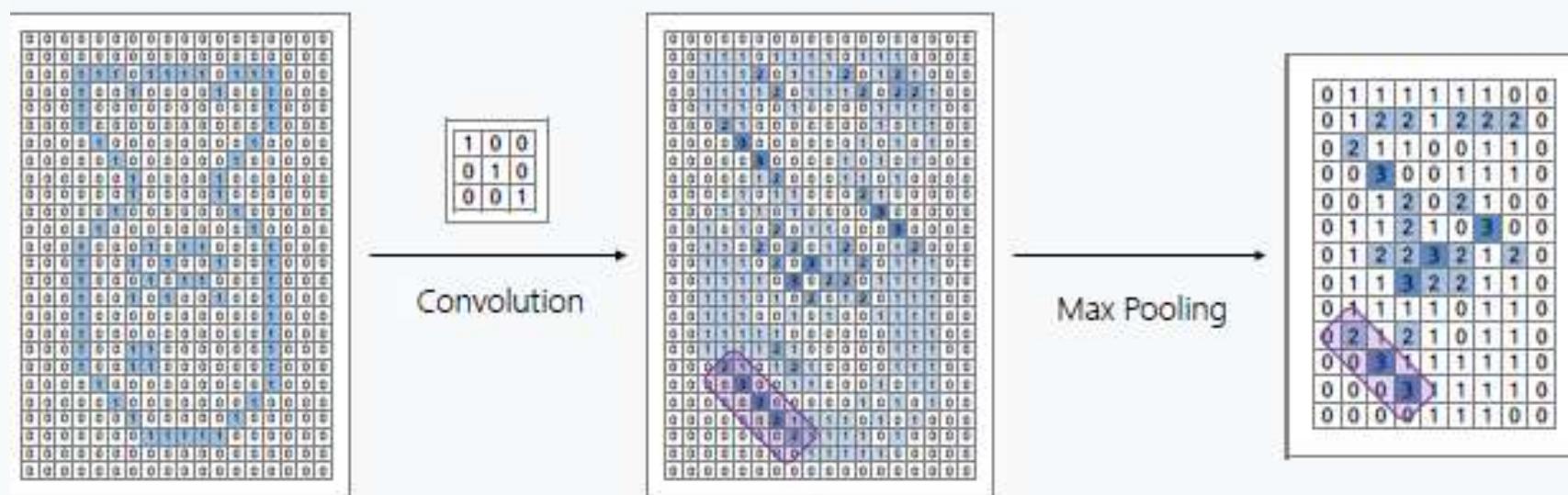
Convolution : 지느러미 필터



1. 이미지 Neural Network 실습

image 특징점

Convolution : 대각선 필터



1. 이미지 Neural Network 실습

image 특징점

기우시안			사프닝		
박스			0 -1 0 -1 5 -1 0 -1 0		
.0000	.0000	.0002	.0000	.0000	
.0000	.0113	.0837	.0113	.0000	
.0002	.0837	.6187	.0837	.0002	
.0000	.0113	.0837	.0113	.0000	
.0000	.0000	.0002	.0000	.0000	
모션			0304 .0501 0 0 0 .0501 .1771 .0519 0 0 0 .0519 .1771 .0519 0 0 0 .0519 .1771 .0501 0 0 0 .0501 .0304		
수평 예지			수직 예지		
1 1 1	1 0 -1		1 0 -1	1 0 -1	
0 0 0	1 0 -1		1 0 -1	1 0 -1	
-1 -1 -1	1 0 -1		1 0 -1	1 0 -1	



(a) 원래 영상과 여러 가지 마스크들



> 박스



> 가우시안



> 샤프닝



> 수평 예지



> 수직 예지



> 모션

1. 이미지 Neural Network 실습

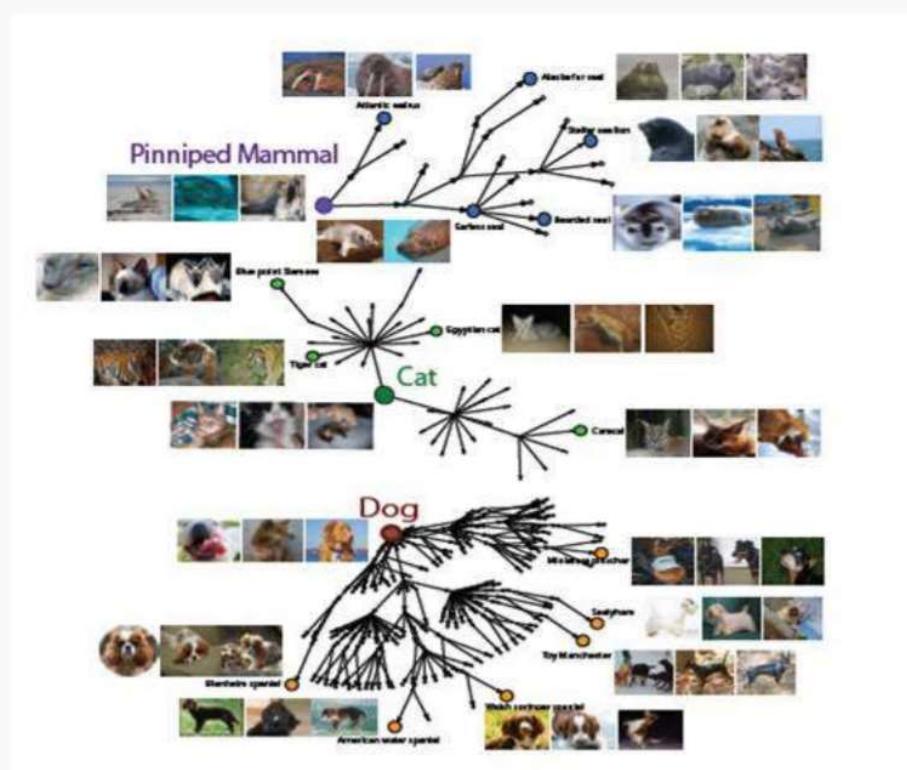
Image 특징점 기반 Neural Network image 분류



- Hand-crafted 특징 추출 대표 방법
 - SHIF
 - SURF
 - HOG
 - PCA
 - ETC

1. 이미지 Neural Network 실습

Image 특징점 기반 Neural Network image 분류



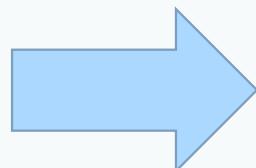
1. 이미지 Neural Network 실습

Image 특징점 기반 Neural Network image 분류



- Hand-crafted 특징 추출 대표 방법

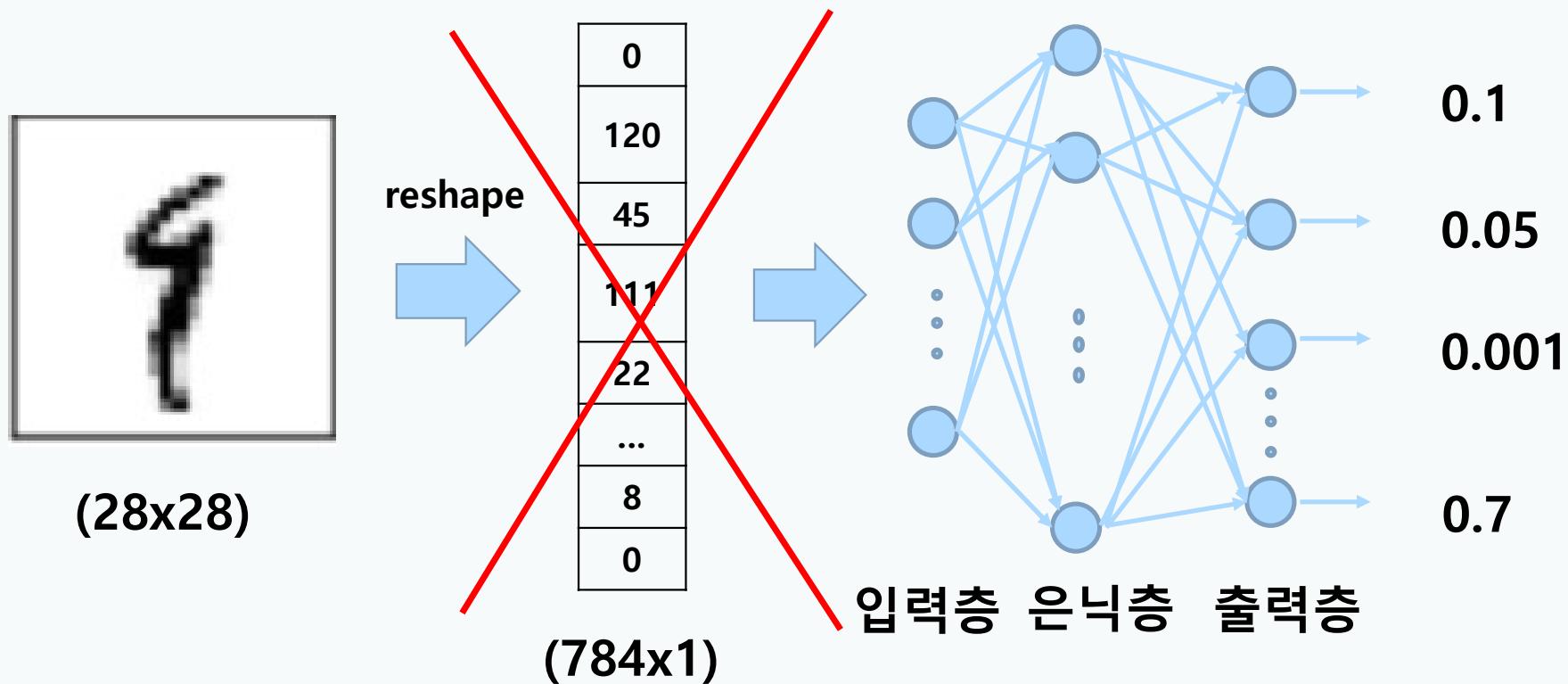
- SHIF
- SURF
- HOG
- PCA
- ETC



학습을 통해서 특징점을 찾아주면?

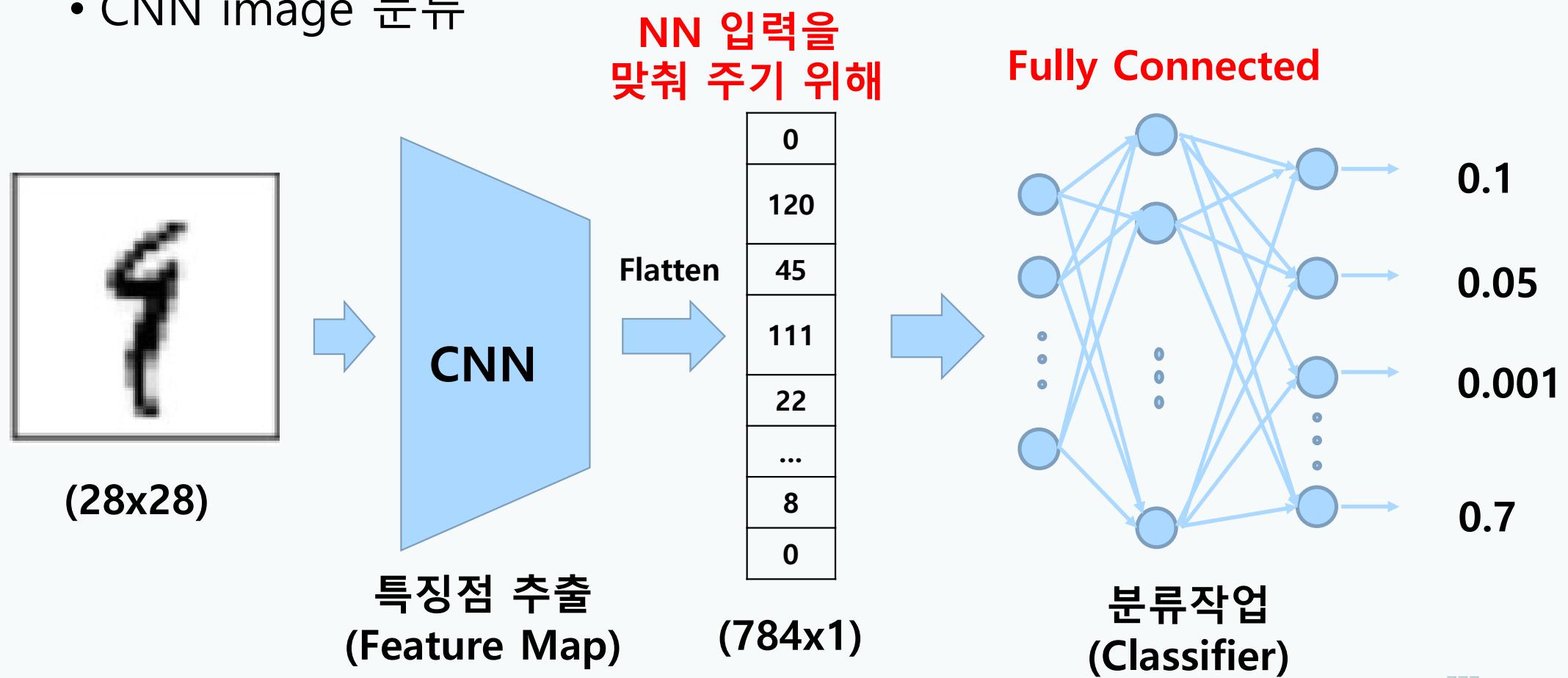
2. Convolution Neural Network(CNN)

- CNN image 분류



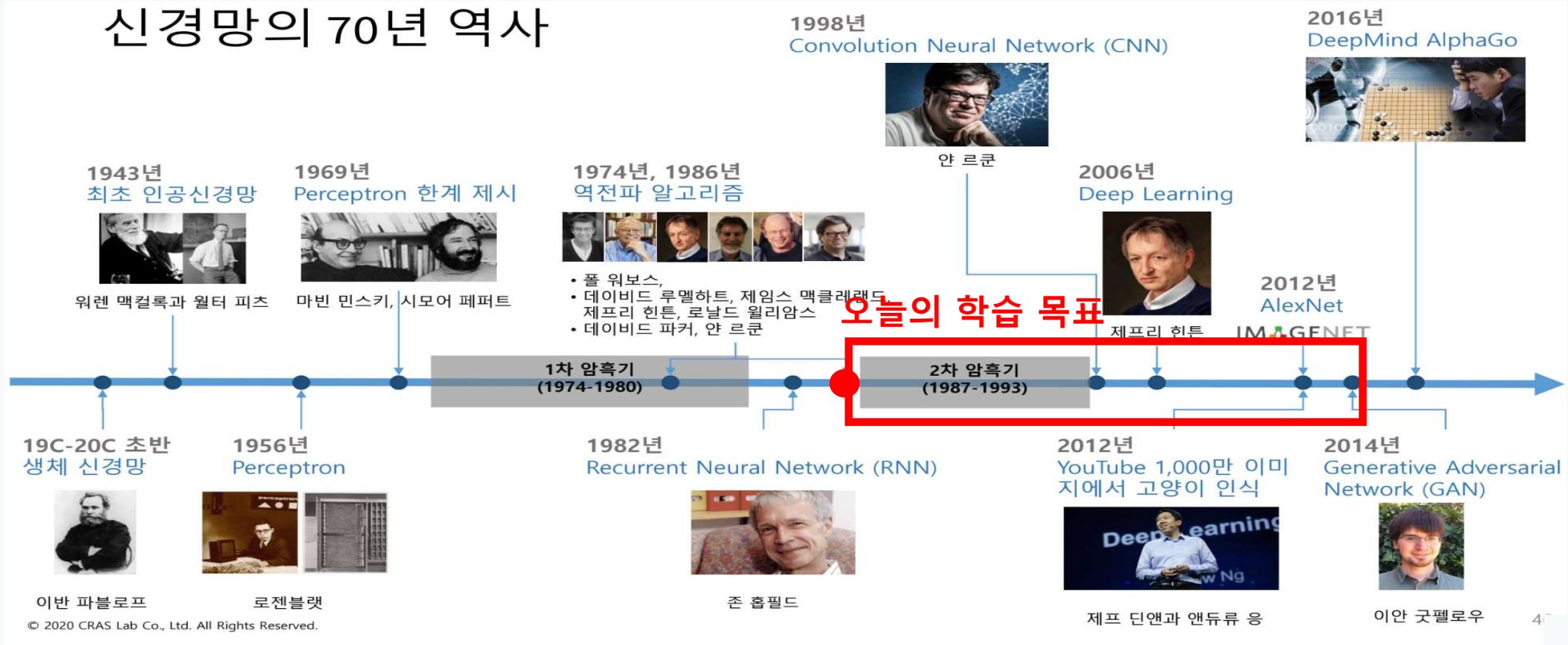
2. Convolution Neural Network(CNN)

- CNN image 분류



2. Convolution Neural Network(CNN)

신경망의 70년 역사



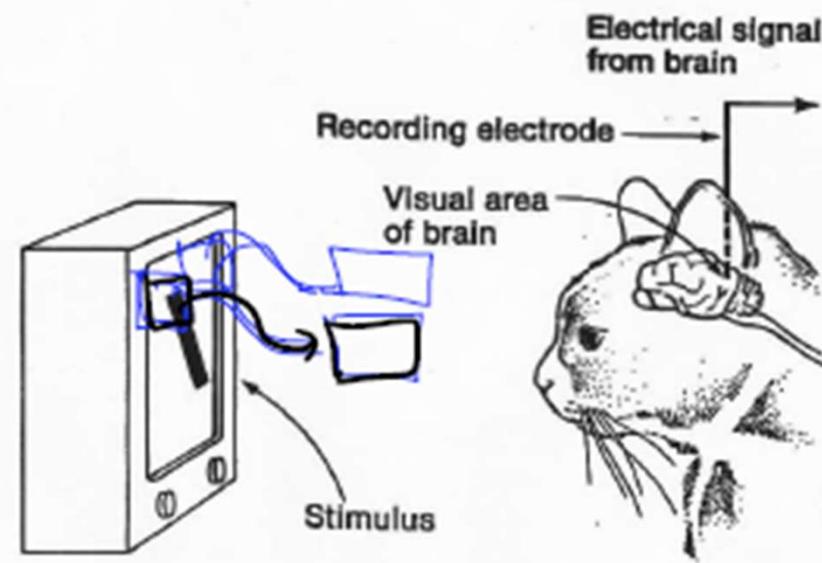
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



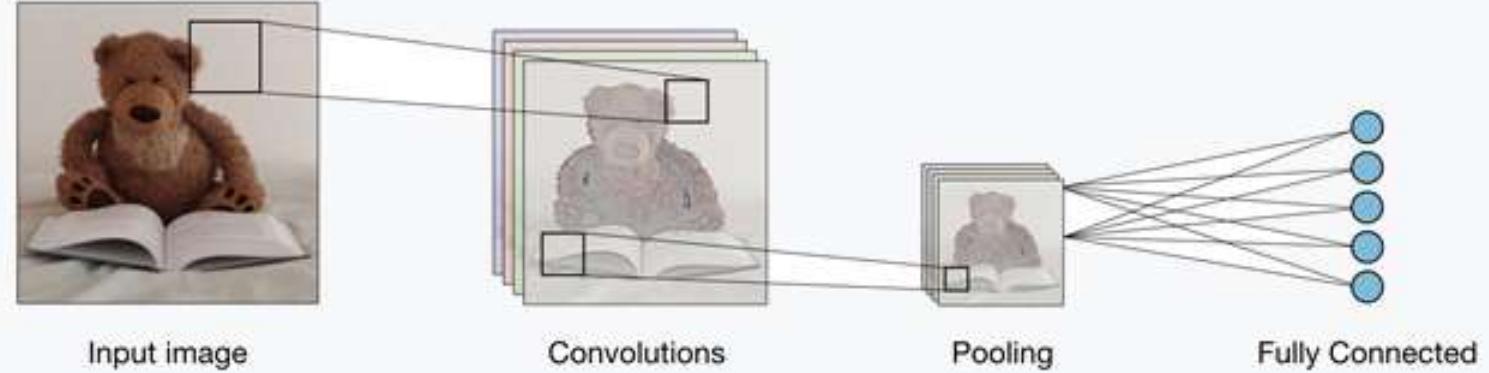
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



Convolution Neural Network 는 크게
Convolution / pooling 으로 이루어져 있다

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

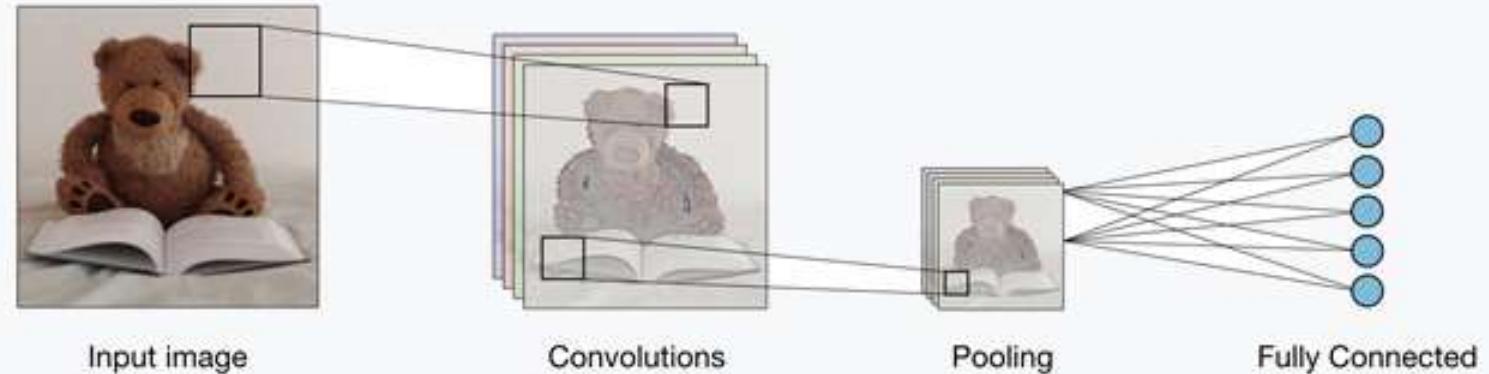
Pooling Layer(pool)

Fully Connected Layer(FC)

선언된 크기의
Filter를 image에
Override하면서
Feature를 추출

추출된 Feature에서
도드라지(중요한)는
Feature를 추출

최종 추출된 Feature
기반으로 분류



Convolution Neural Network 는 크게
Convolution / pooling 으로 이루어져 있다

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

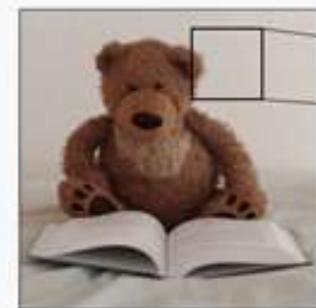
Pooling Layer(pool)

Fully Connected Layer(FC)

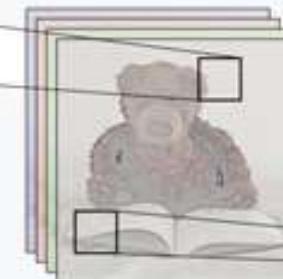
선언된 크기의
Filter를 image에
Override하면서
Feature를 추출

추출된 Feature에서
도드라지(중요한)는
Feature를 추출

최종 추출된 Feature
기반으로 분류



Input image



Convolutions



Pooling



Fully Connected

먼저 Filter(=kernel)을 선언하여
image에 override 계산하는 과정을 neural Network 는 크게
살펴보겠습니다. convolution / pooling 으로 이루어져 있다

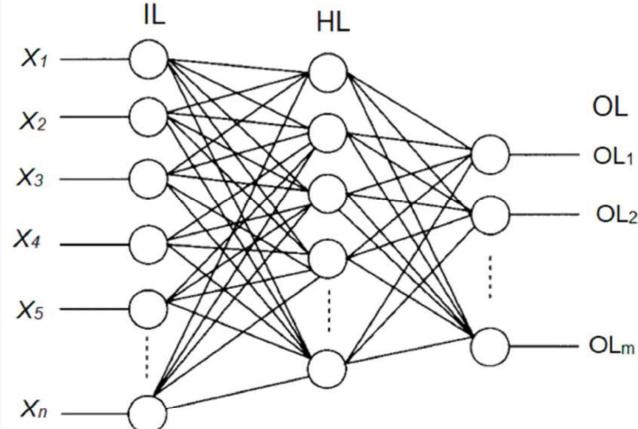
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

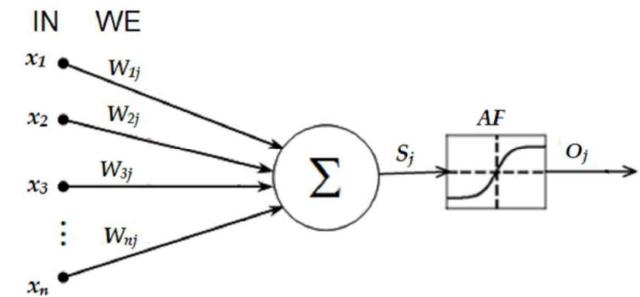
Convolution Layer(conv)

Pooling Layer(pool)

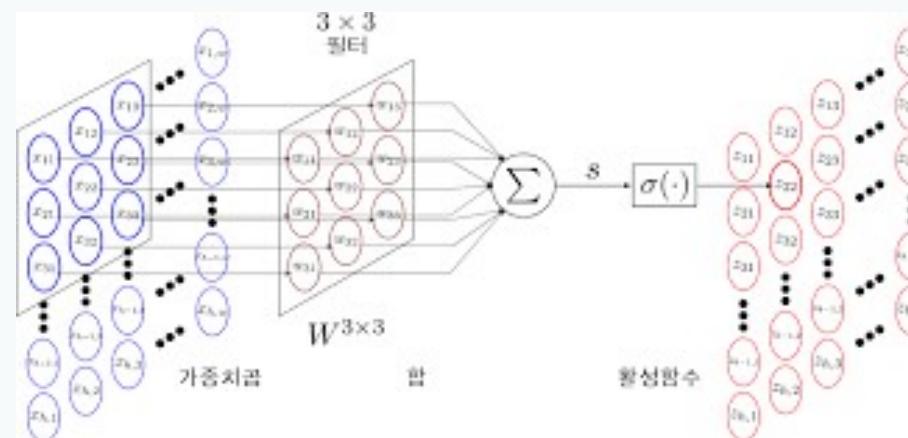
Fully Connected Layer(FC)



(a)



(b)



2. Convolution Neural Network(CNN)

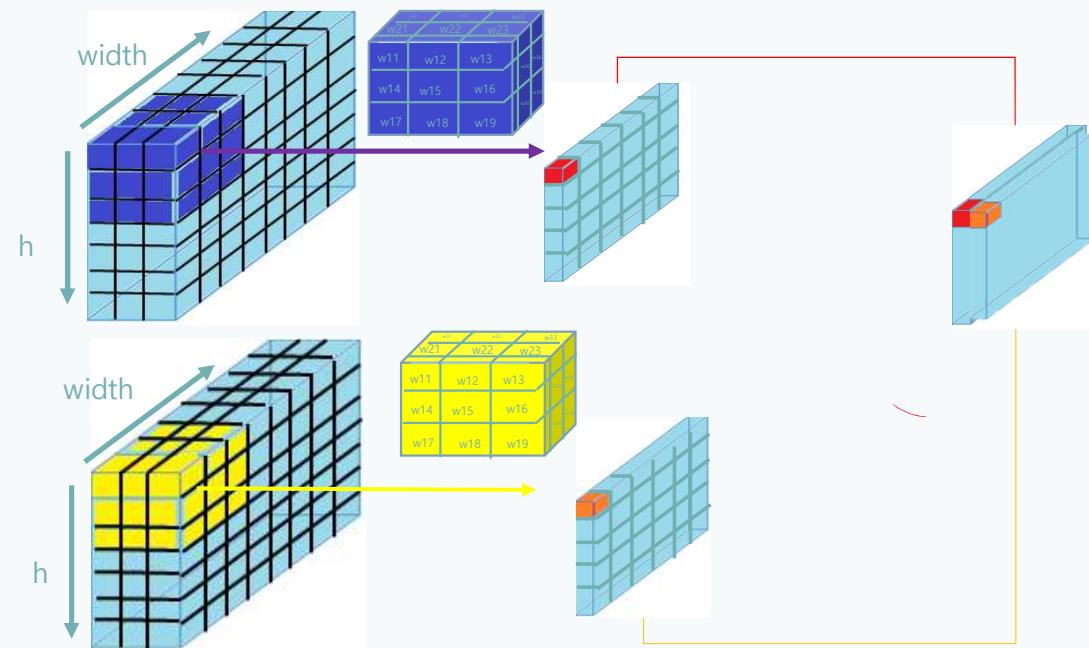
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

IMAGE(R,G,B) 에 (3X3) filter를 선언하여
Feature을 계산하는 과정



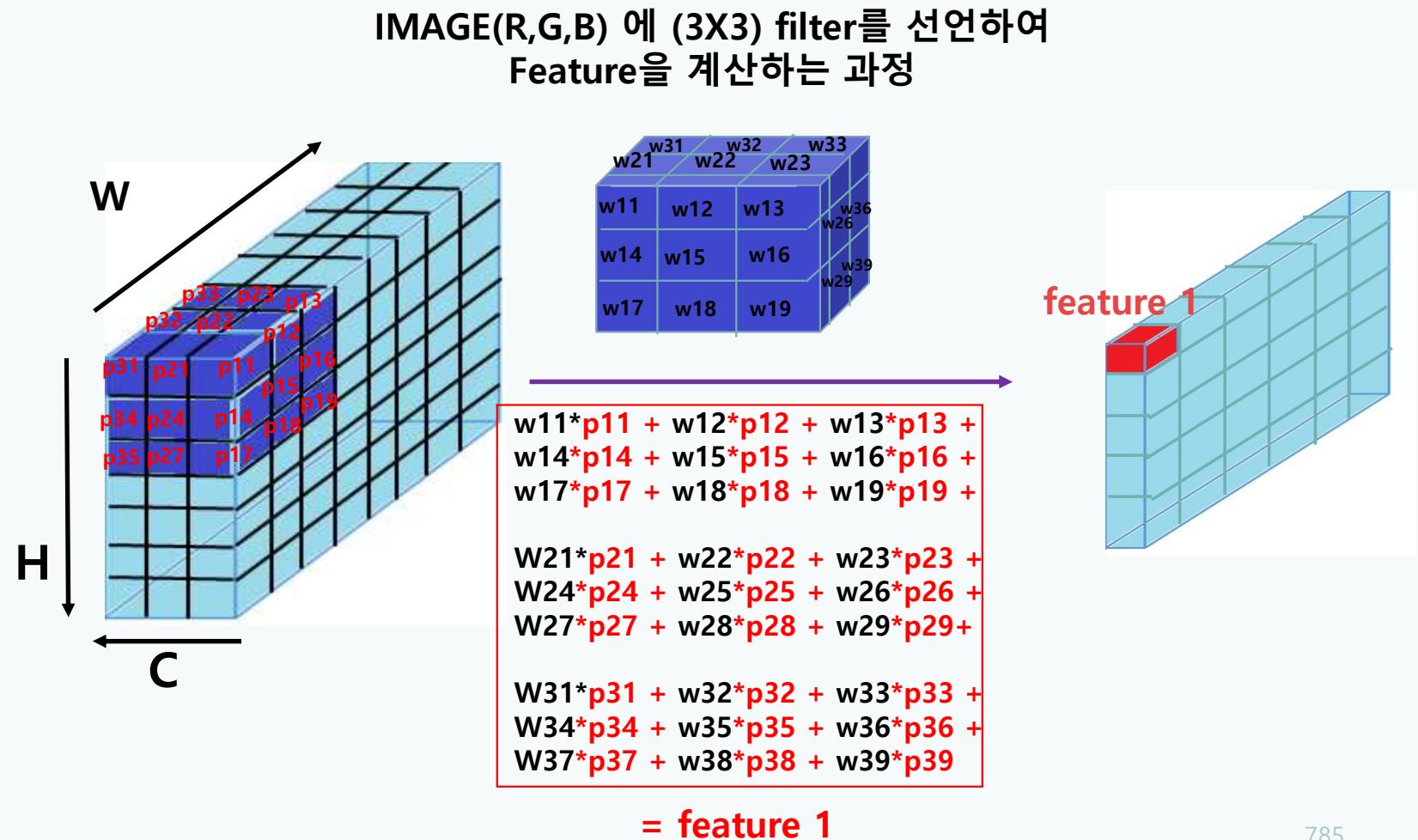
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



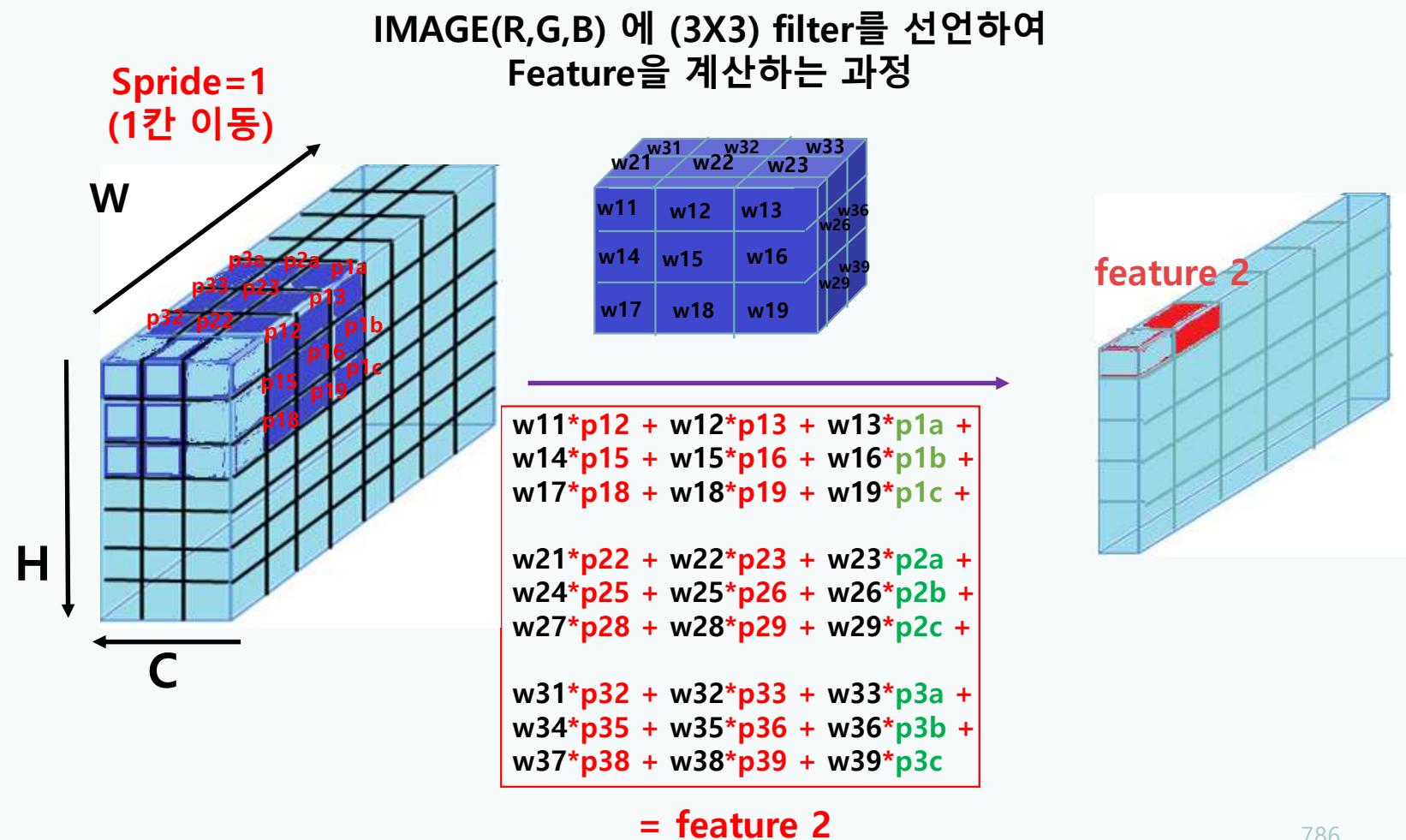
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



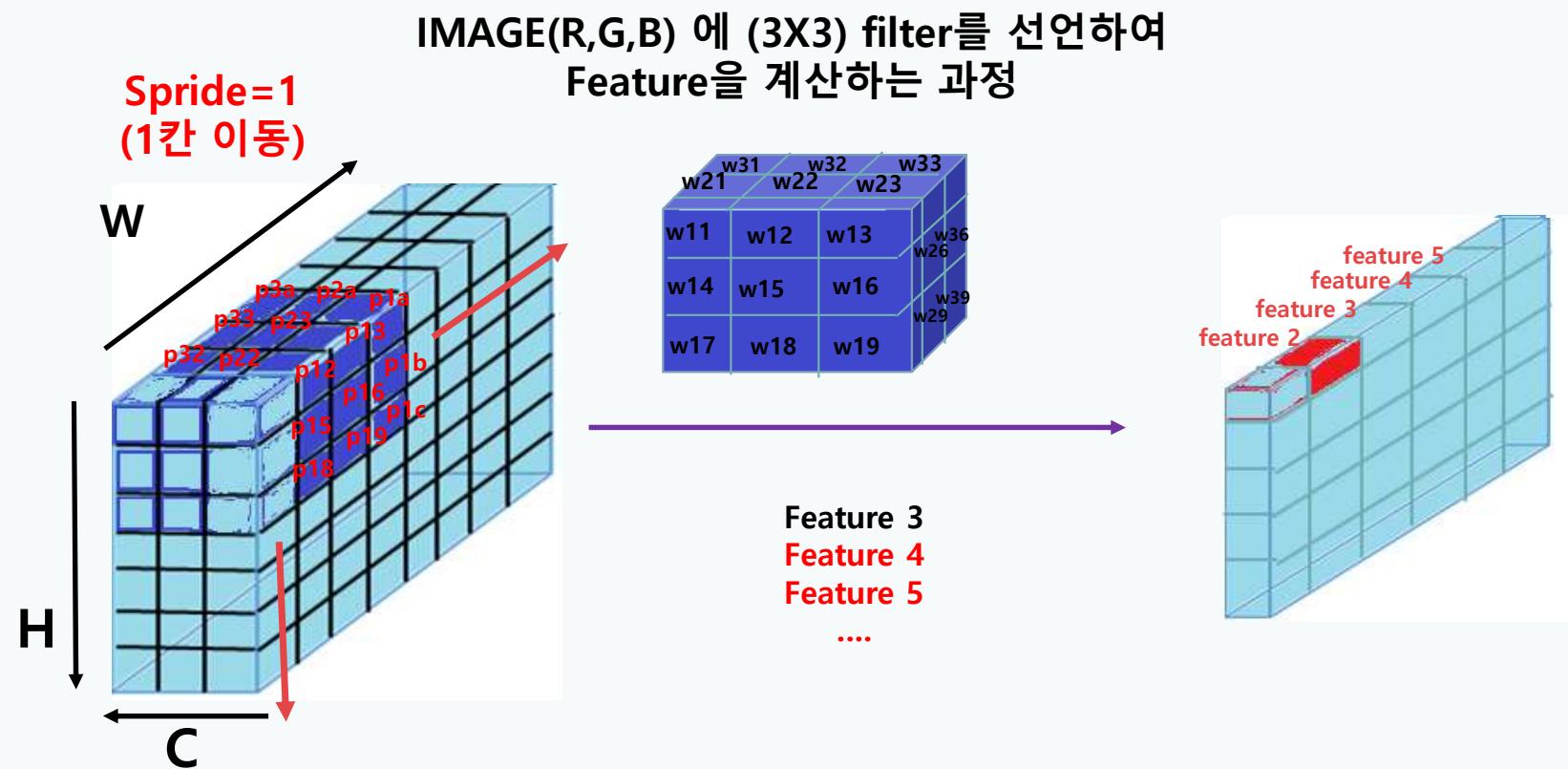
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



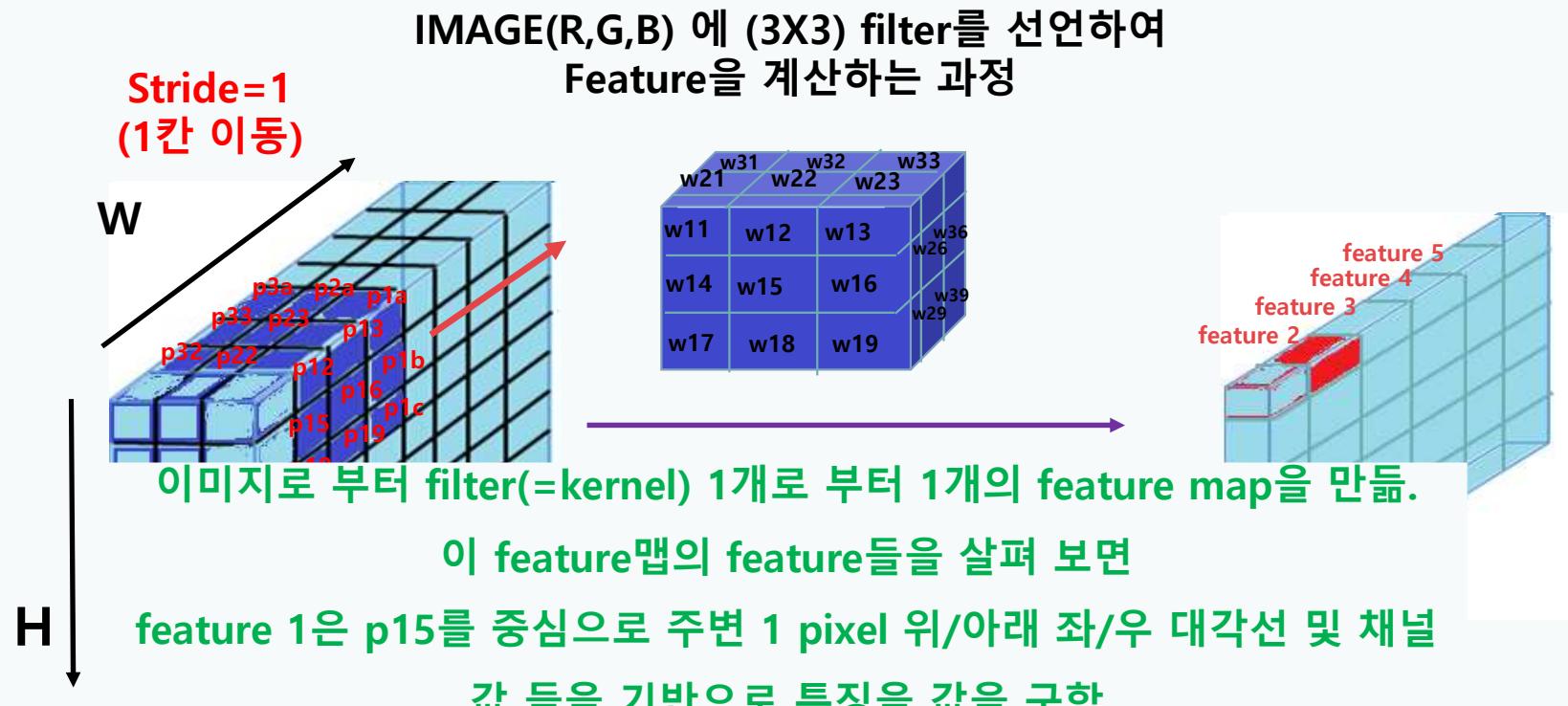
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



분류자가 잘 분류 할 수 있는 특징값들이 모여 있는 activation map을 만들수 있도록 하는 filter(=kernel)의 w값으로 학습으로 찾는것이 목표

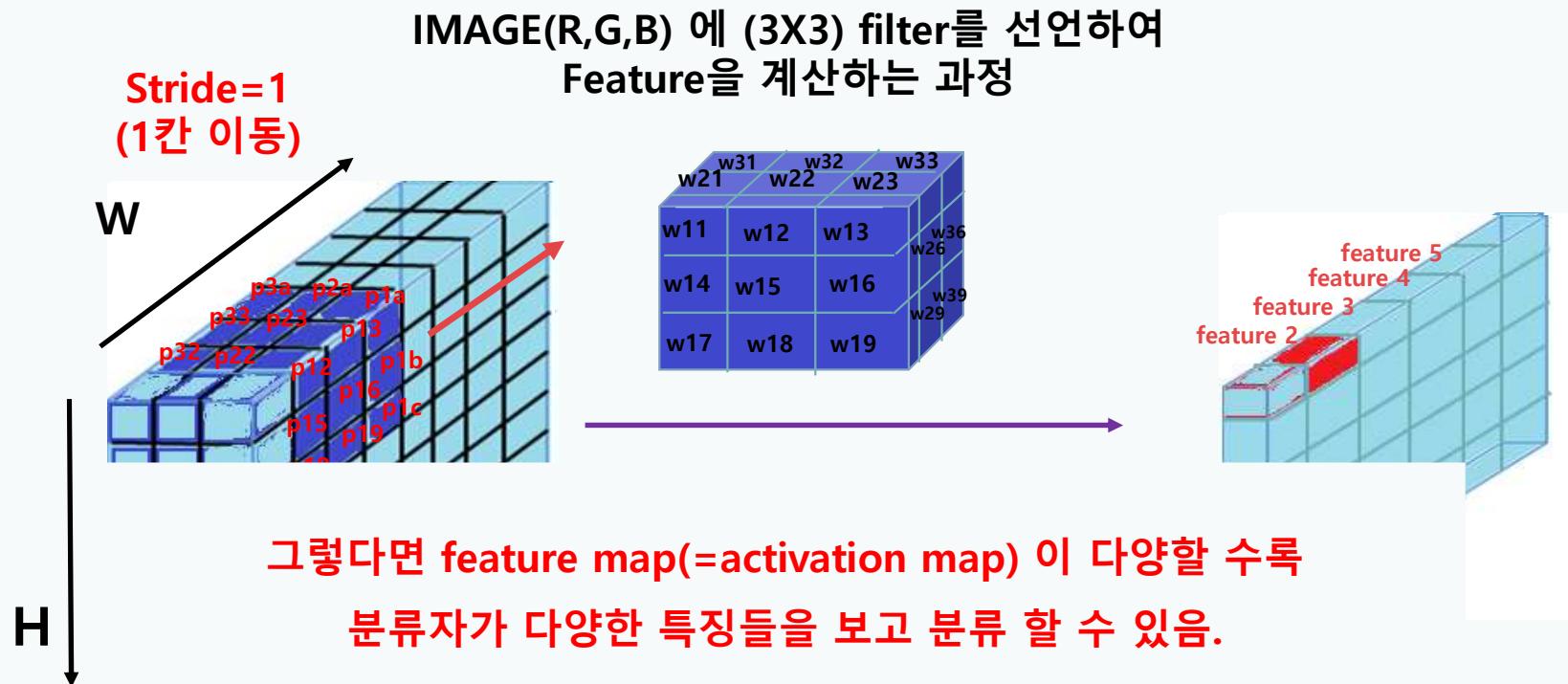
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



즉, 다양한 feature map을 생성 할 수 있도록 kernel을 여러 개 생성하여
다양한 feature map을 나타낼 수 있도록 학습

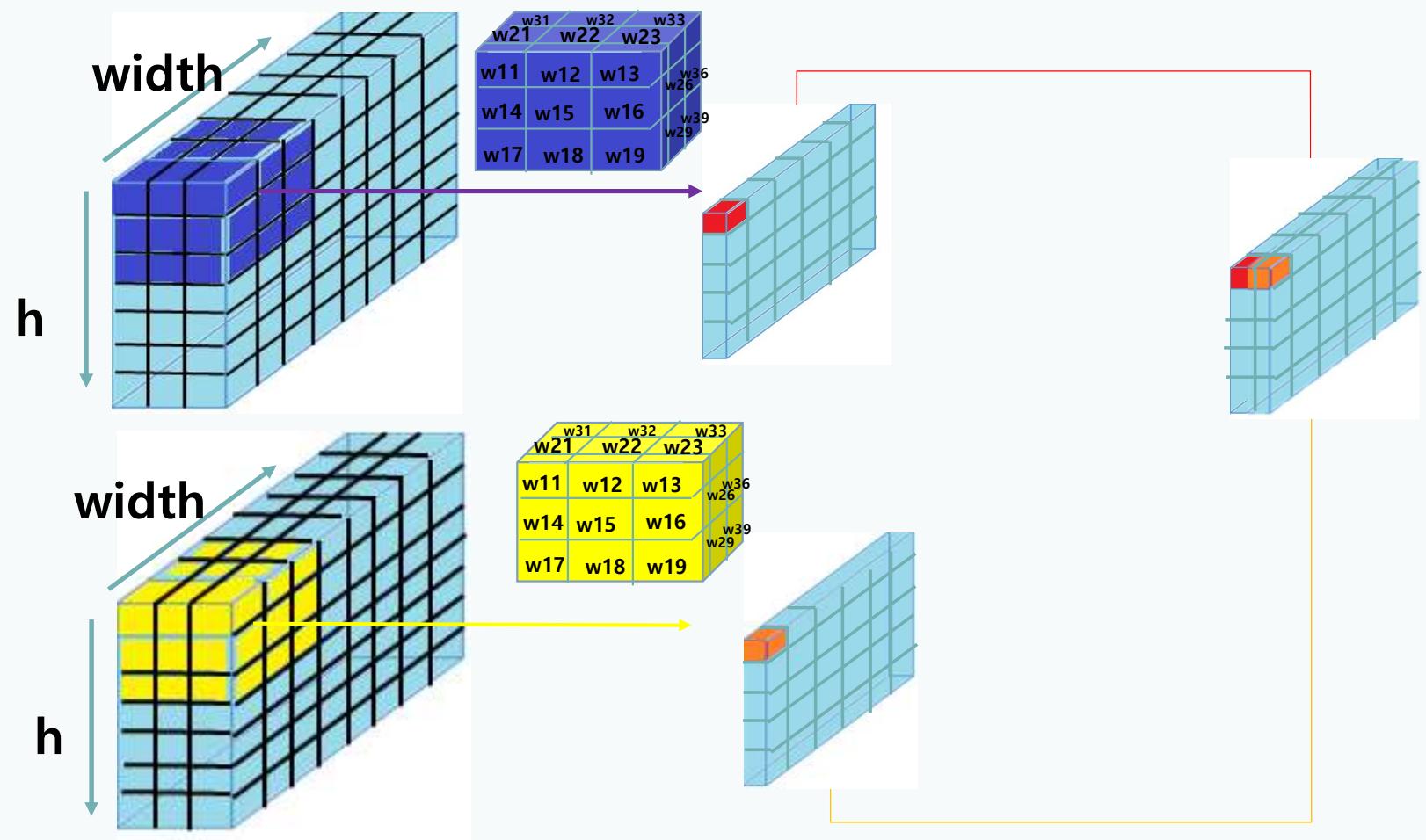
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



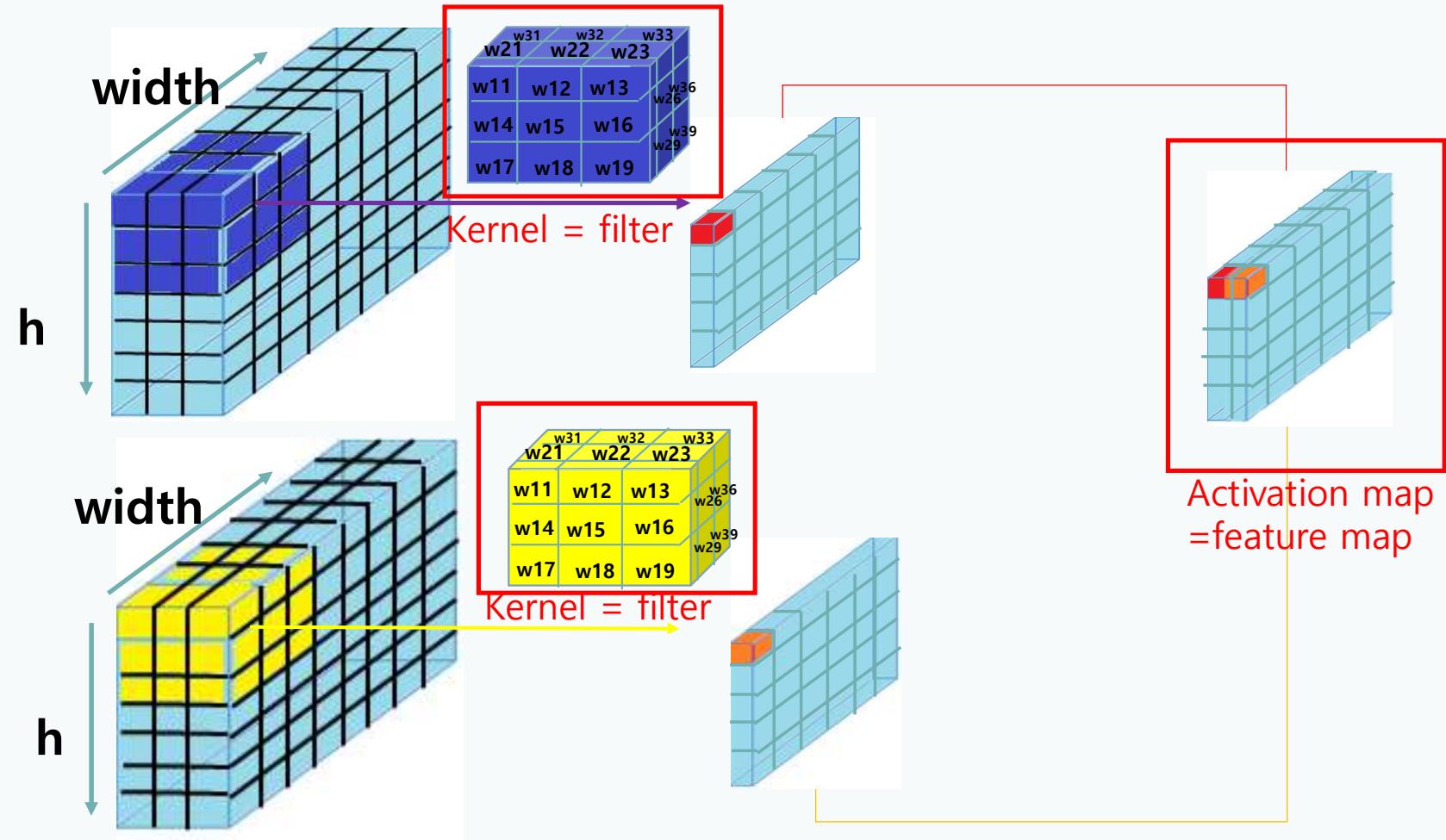
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



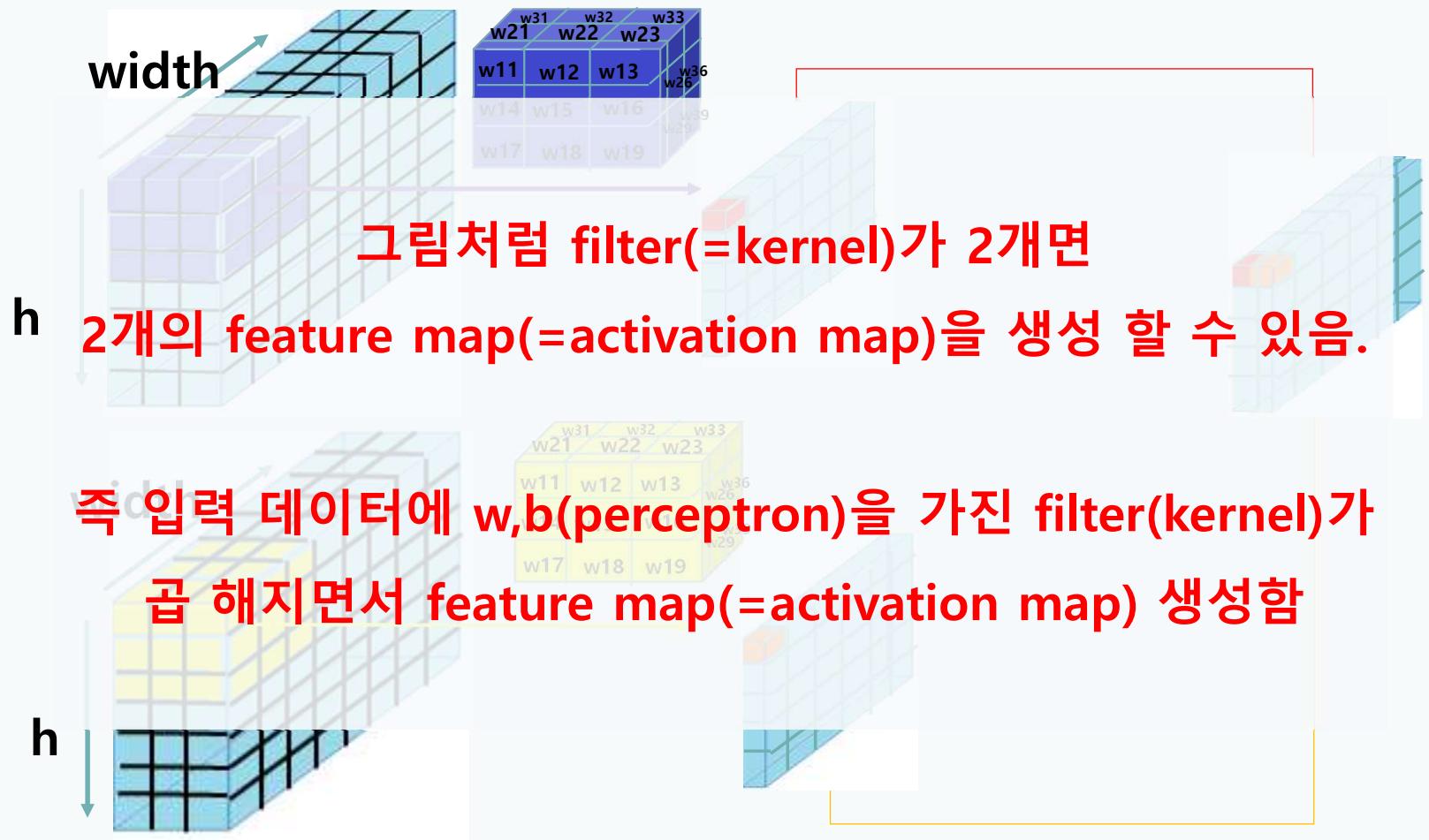
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



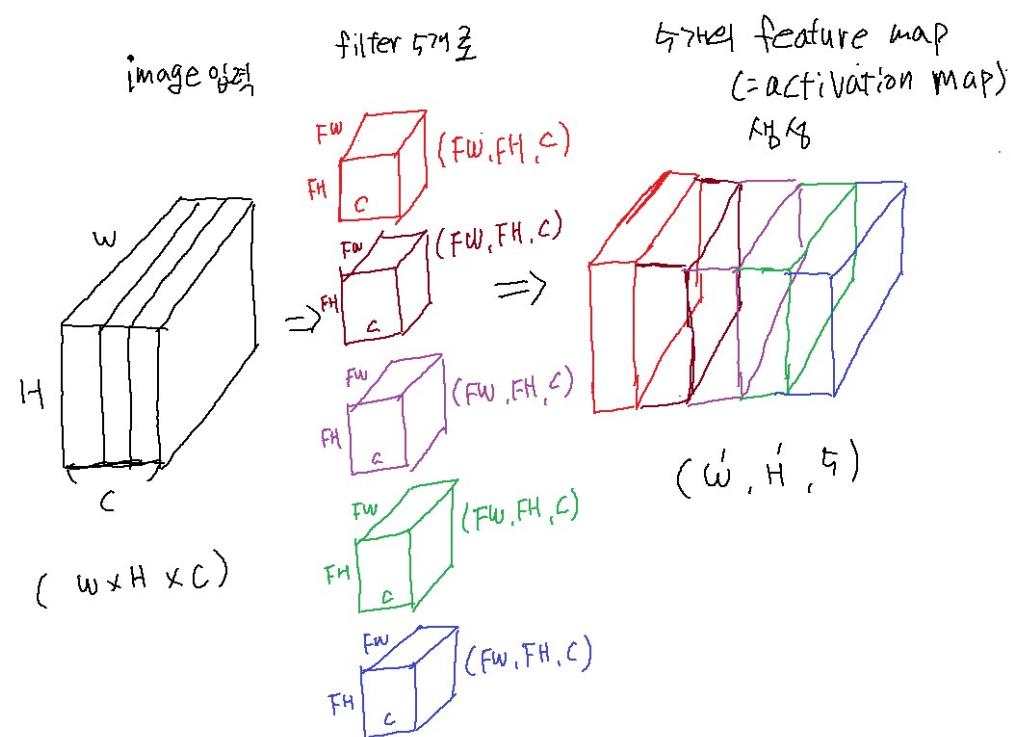
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



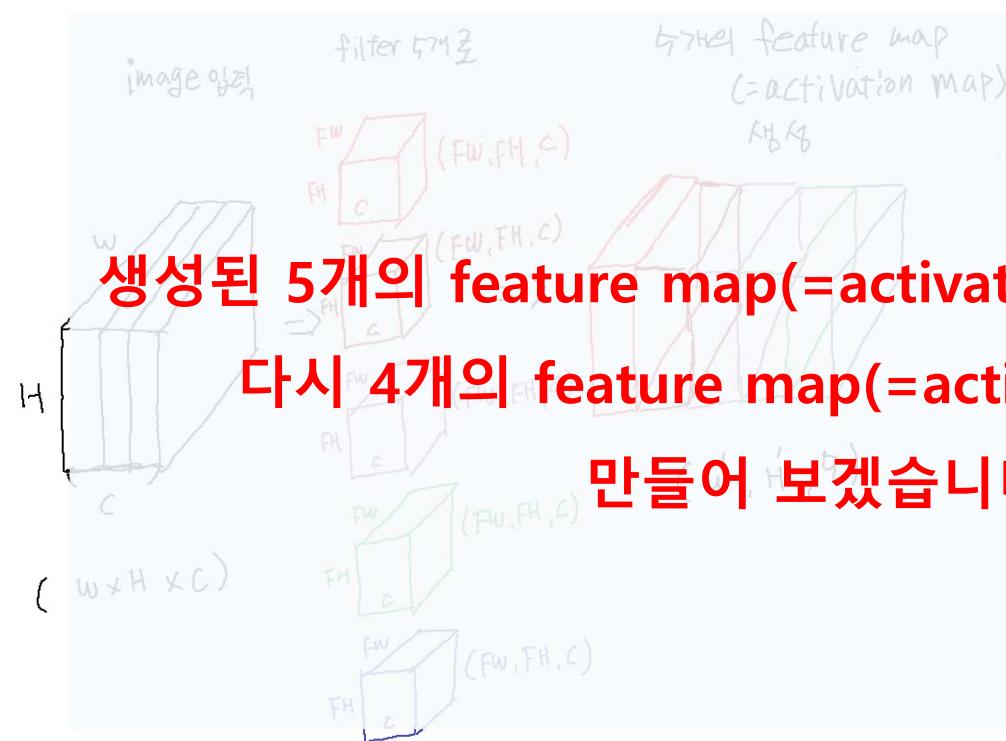
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



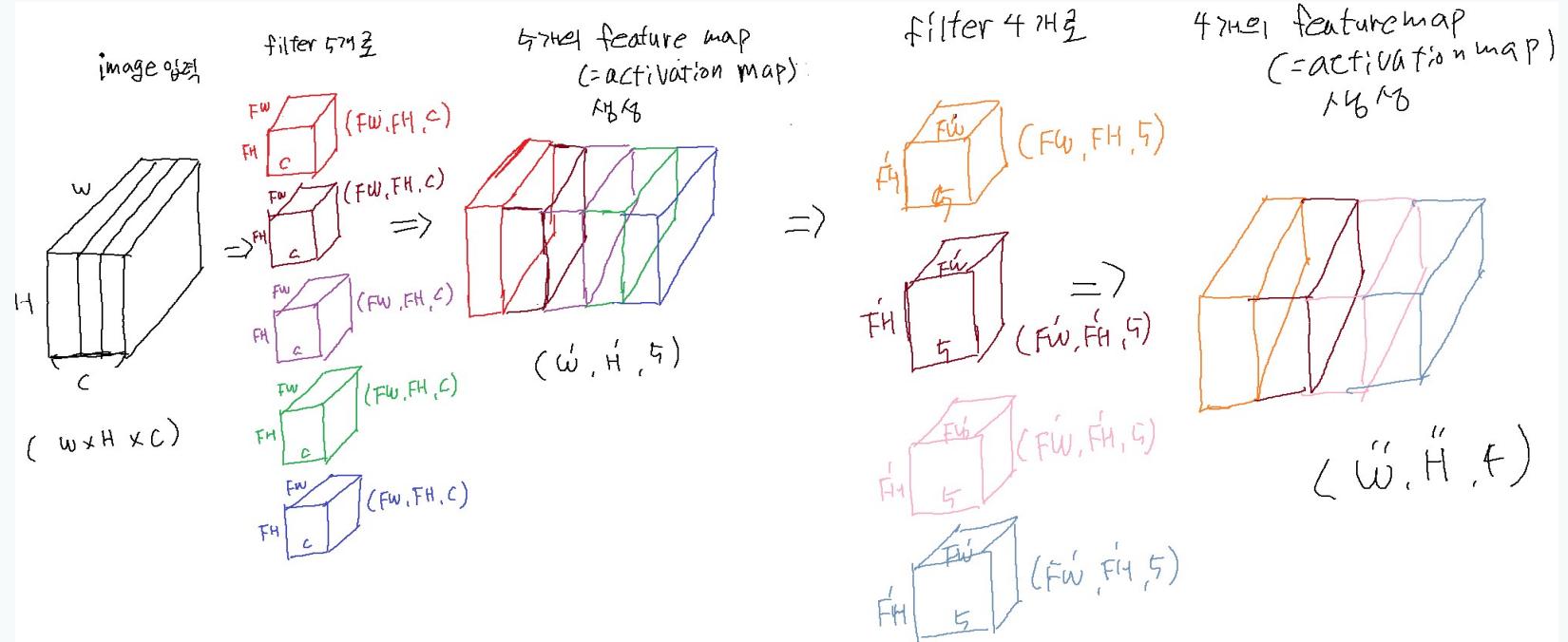
2. Convolution Neural Network(CNN)

**Convolutional
Neural Network**

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



FW, FH 는
filter (=kernel)의 크기로
그림에 표시한 것과 같다.

FW, FH 는
filter (=kernel)의 크기로
그림에 표시한 것과 같다.

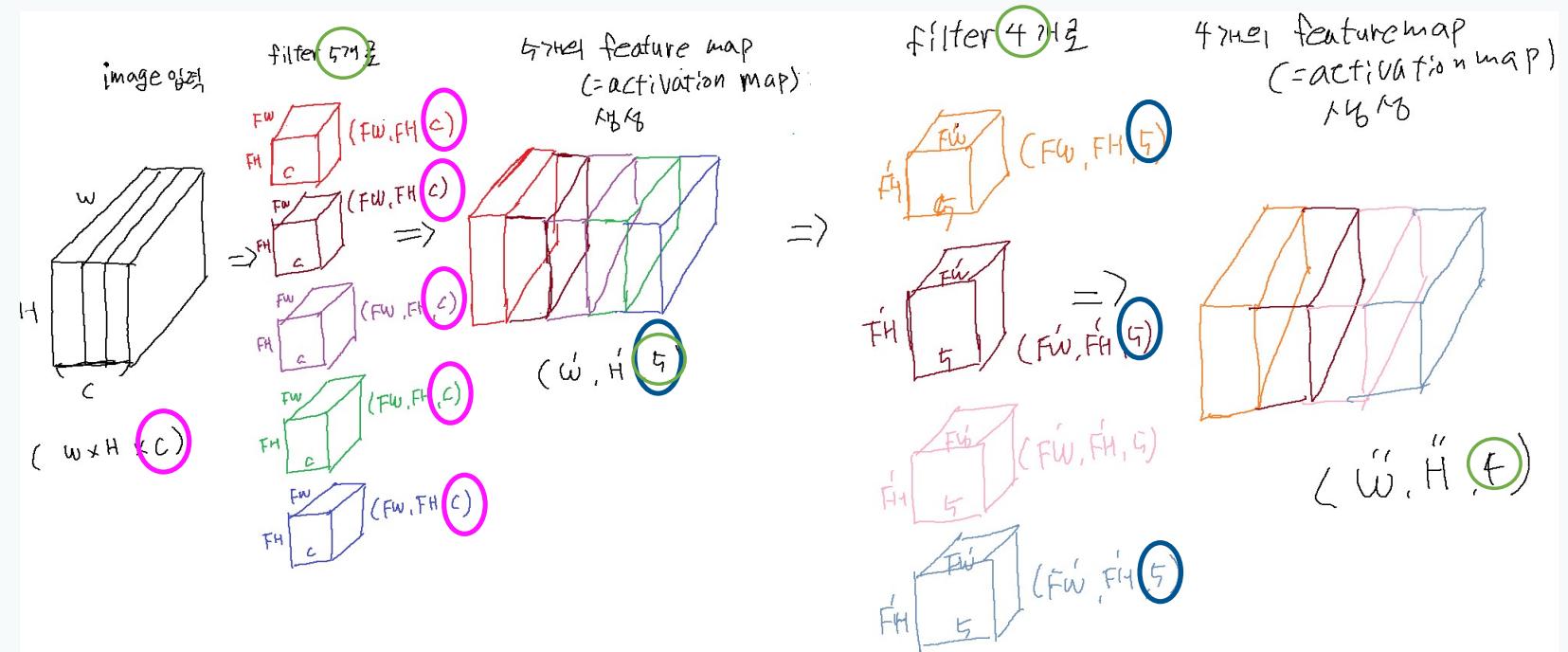
2. Convolution Neural Network(CNN)

**Convolutional
Neural Network**

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



FW, FH 는
filter (=kernel)의 크기로
그림에 표시한 것과
같다.

FW, FH 는
filter (=kernel)의 크기로
그림에 표시한 것과
같다.

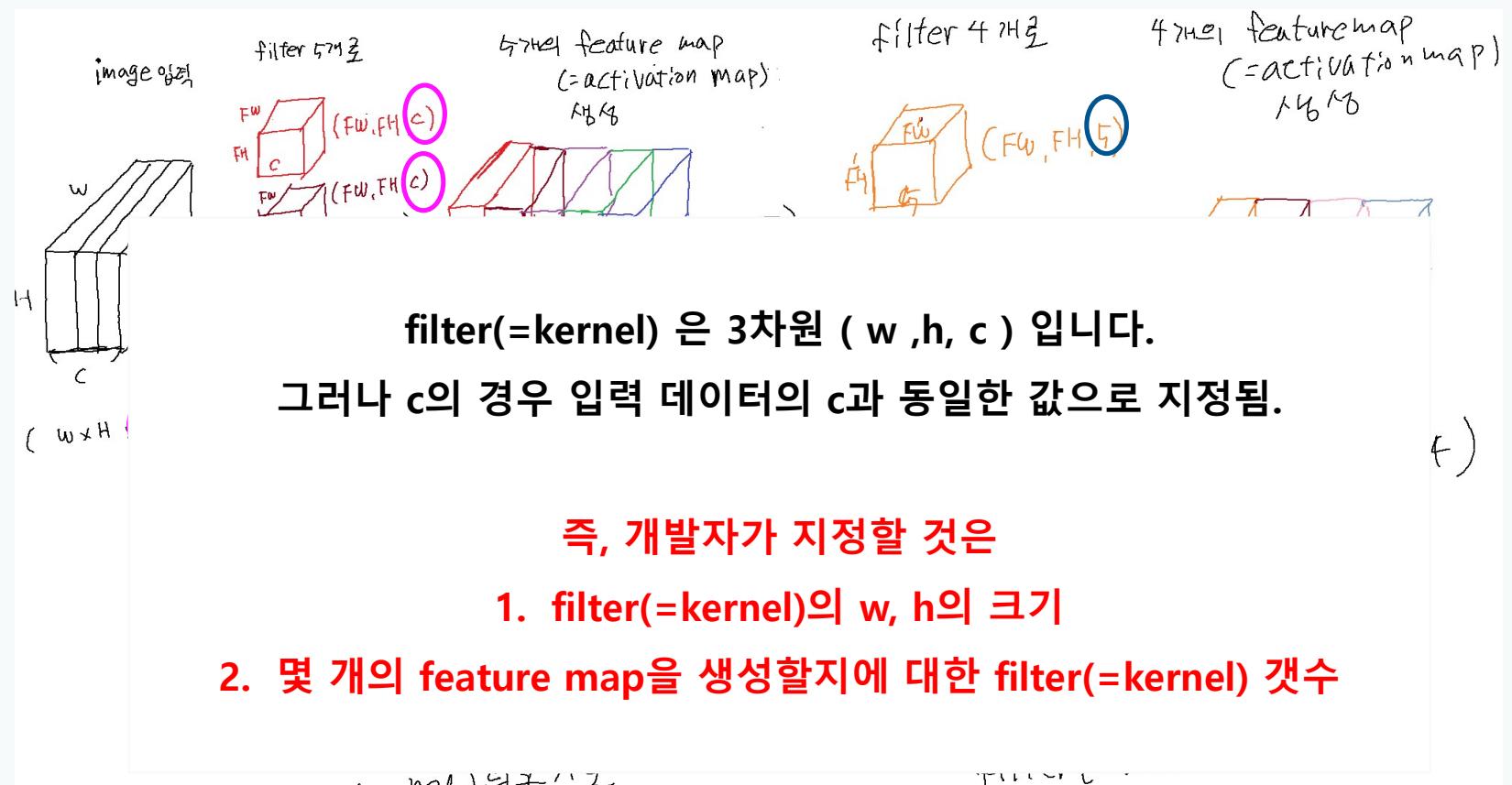
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



filter(=kernel)의 크기
개발자 지정

개수 지정

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

(3x3) kernel , stride=1 선언한 연산

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

Image

4		

Convolved
Feature

2. Convolution Neural Network(CNN)

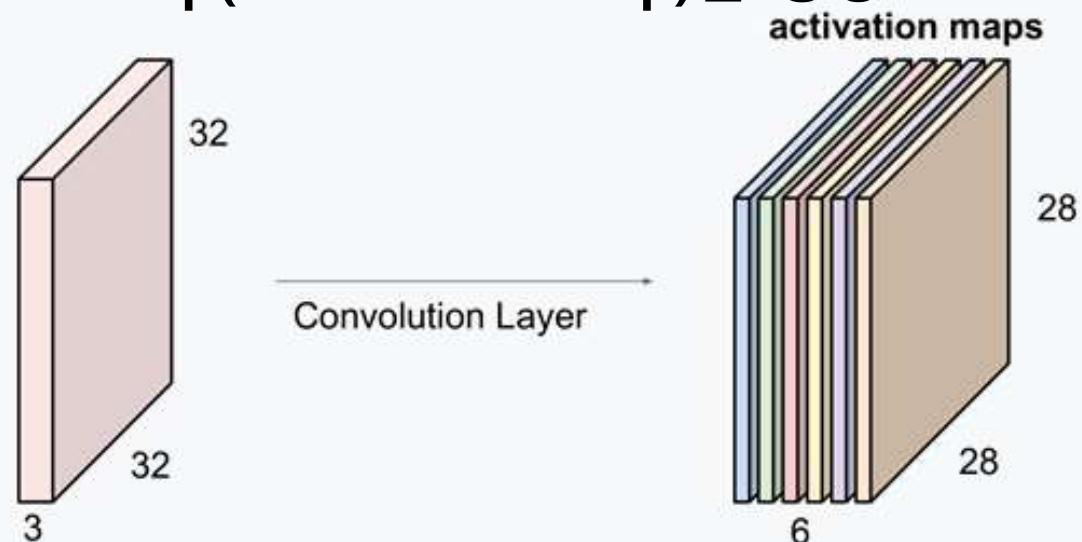
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

(5x5) filter 6개를 만들면 6개의 feature
map(=activation map)을 생성



We stack these up to get a “new image” of size 28x28x6!

2. Convolution Neural Network(CNN)

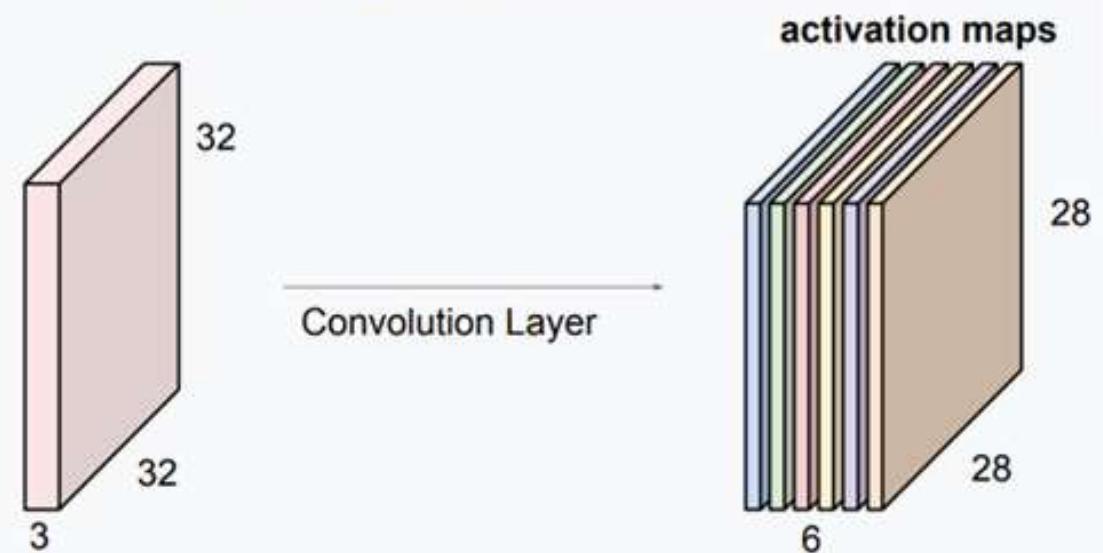
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

- 6개의 5x5필터를 convolution을 적용하면 6장의 activation map이 나온다.



Activation map의 채널 수는 필터의 개수와 동일!

2. Convolution Neural Network(CNN)

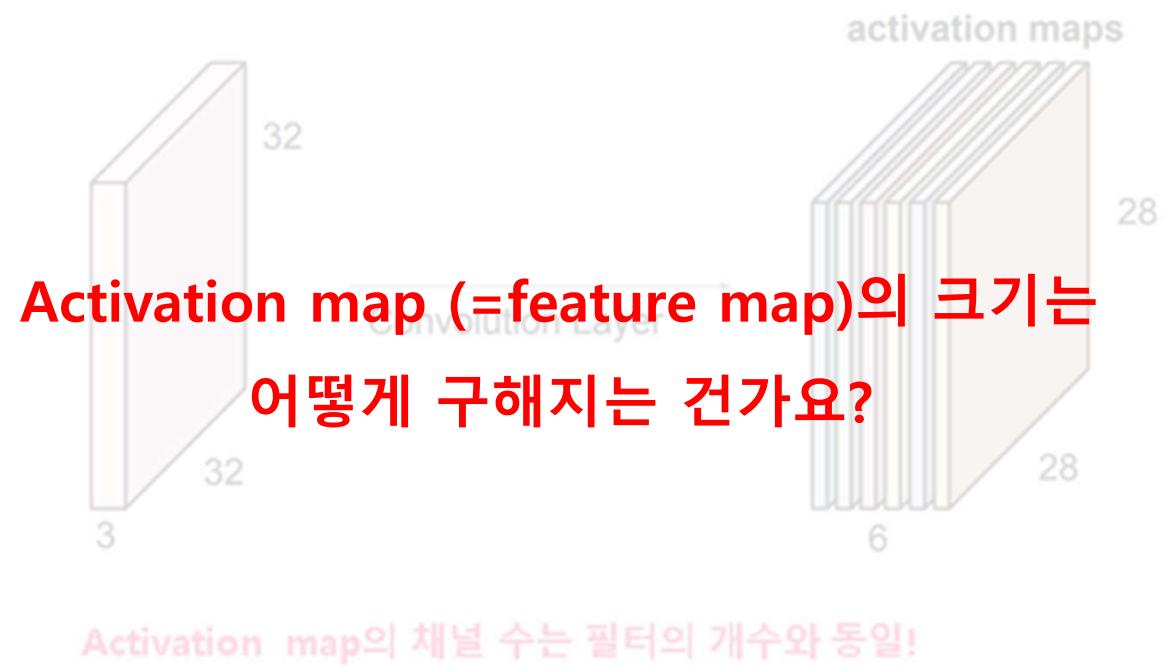
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

- 6개의 5x5필터를 convolution을 적용하면 6장의 activation map이 나온다.



2. Convolution Neural Network(CNN)

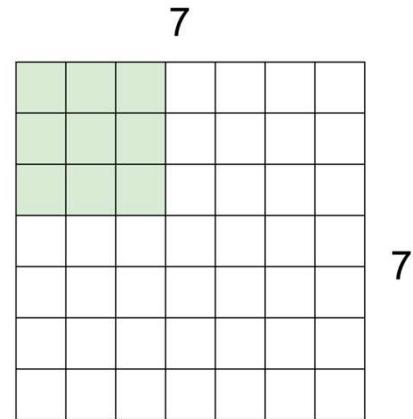
**Convolutional
Neural Network**

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

2. Convolution Neural Network(CNN)

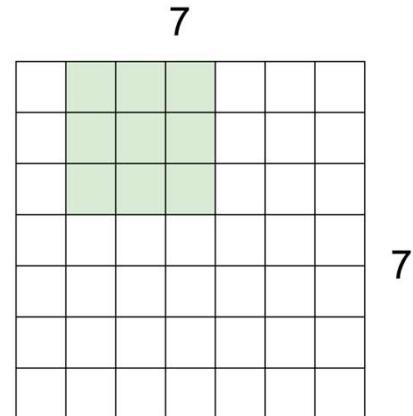
**Convolutional
Neural Network**

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

2. Convolution Neural Network(CNN)

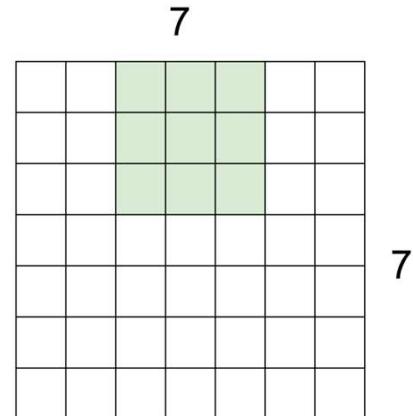
**Convolutional
Neural Network**

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

2. Convolution Neural Network(CNN)

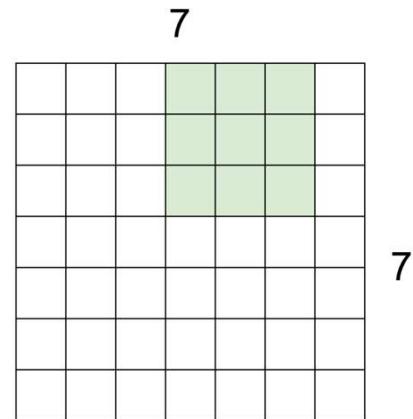
**Convolutional
Neural Network**

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

2. Convolution Neural Network(CNN)

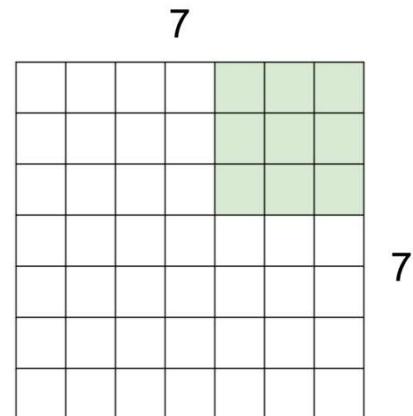
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

2. Convolution Neural Network(CNN)

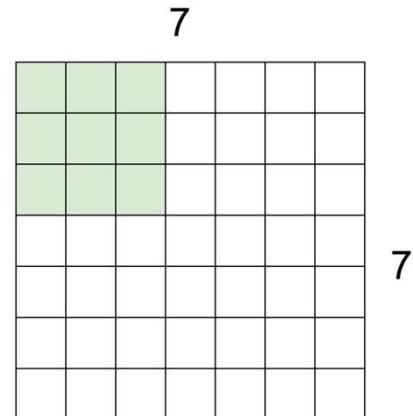
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

2. Convolution Neural Network(CNN)

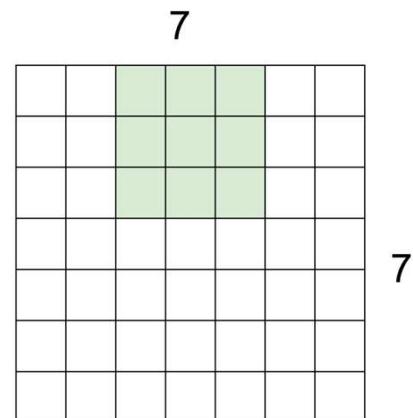
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

2. Convolution Neural Network(CNN)

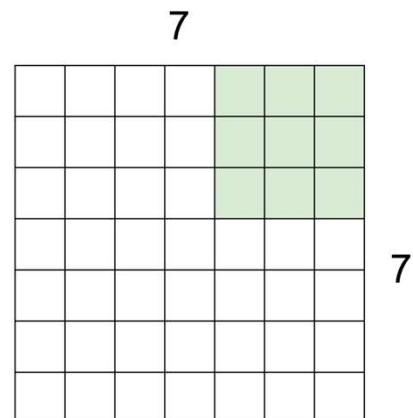
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

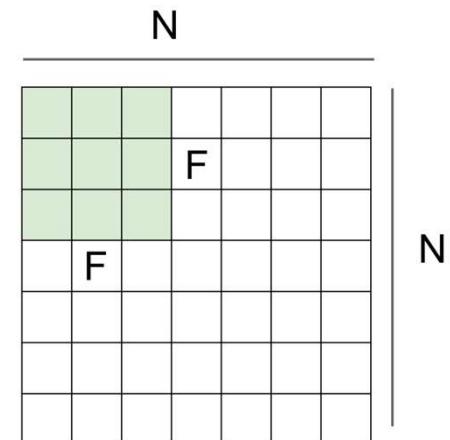
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7$, $F = 3$:
stride 1 $\Rightarrow (7 - 3)/1 + 1 = 5$
stride 2 $\Rightarrow (7 - 3)/2 + 1 = 3$
stride 3 $\Rightarrow (7 - 3)/3 + 1 = 2.33 : \backslash$

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



N

F

F

N

Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7, F = 3:$
 $\text{stride } 1 \Rightarrow (7 - 3)/1 + 1 = 5$
 $\text{stride } 2 \Rightarrow (7 - 3)/2 + 1 = 3$
 $\text{stride } 3 \Rightarrow (7 - 3)/3 + 1 = 2.33 \rightarrow 1$

그런데 image를 받아서 filter를 만들어서
Activation map 만들고 다시 filter를 만들어서
Activation map을 또 만드는 과정을 거칠 수록
만들어지는 activation map의 크기는 작아짐.

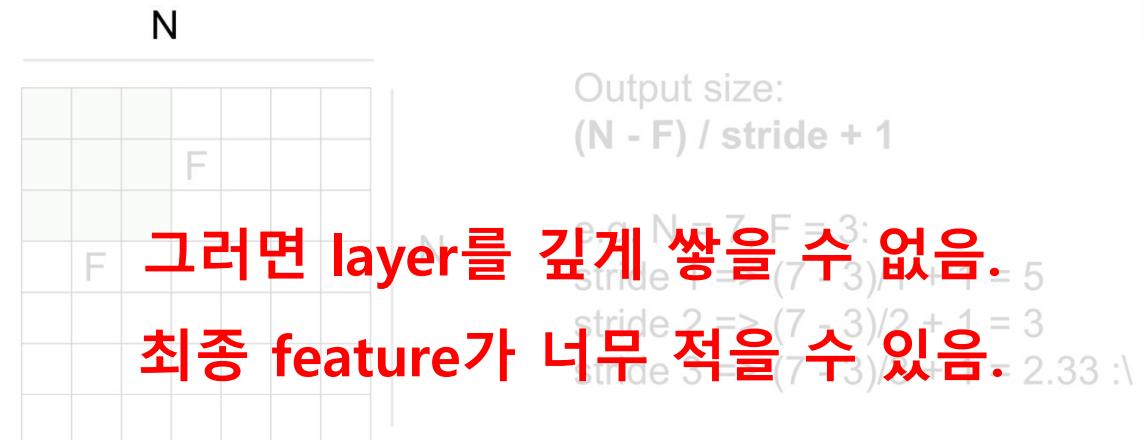
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



어떻게 해결 할 수 없을까?

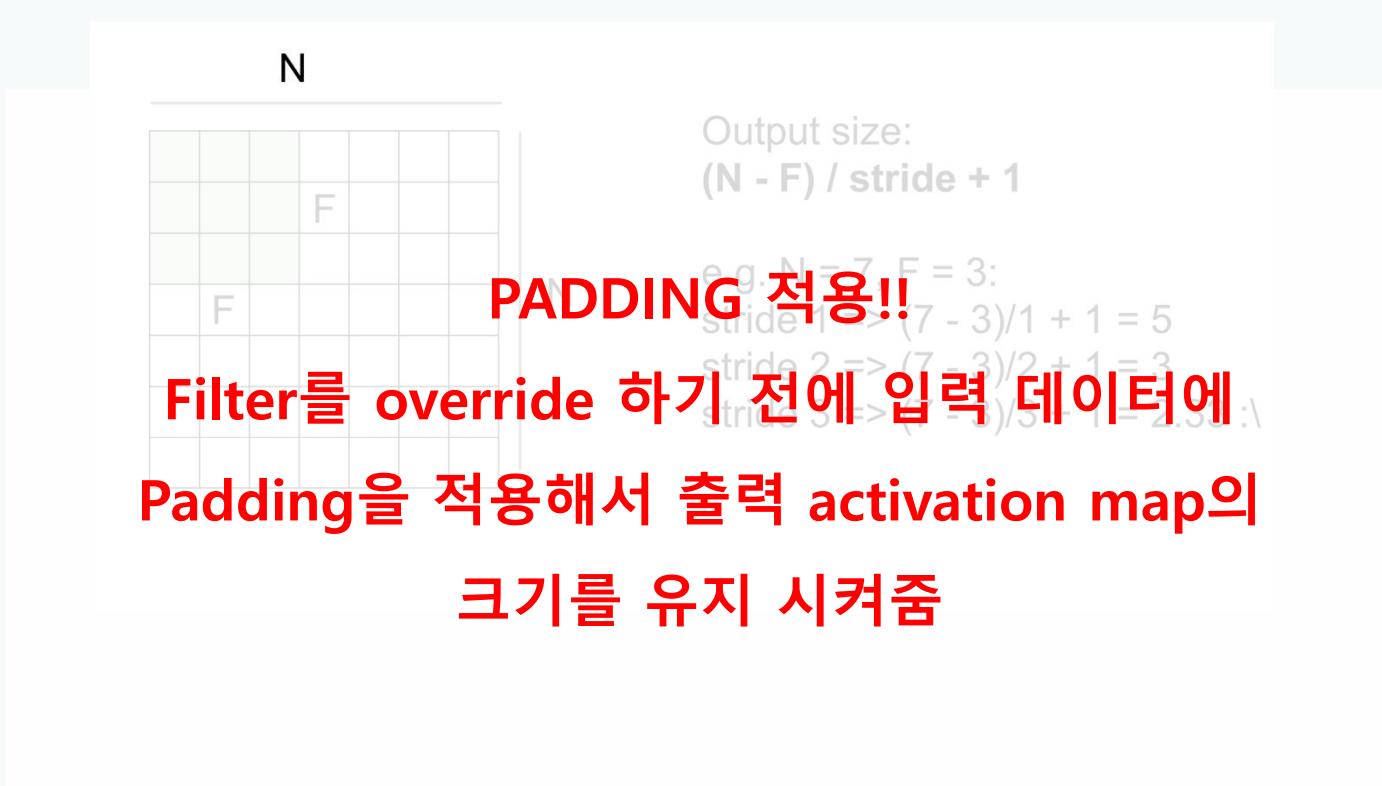
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

(recall:)
$$(N - F) / \text{stride} + 1$$

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

In practice: Common to zero pad the border

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

7x7 output!

in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with $(F-1)/2$. (will preserve size spatially)

e.g. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

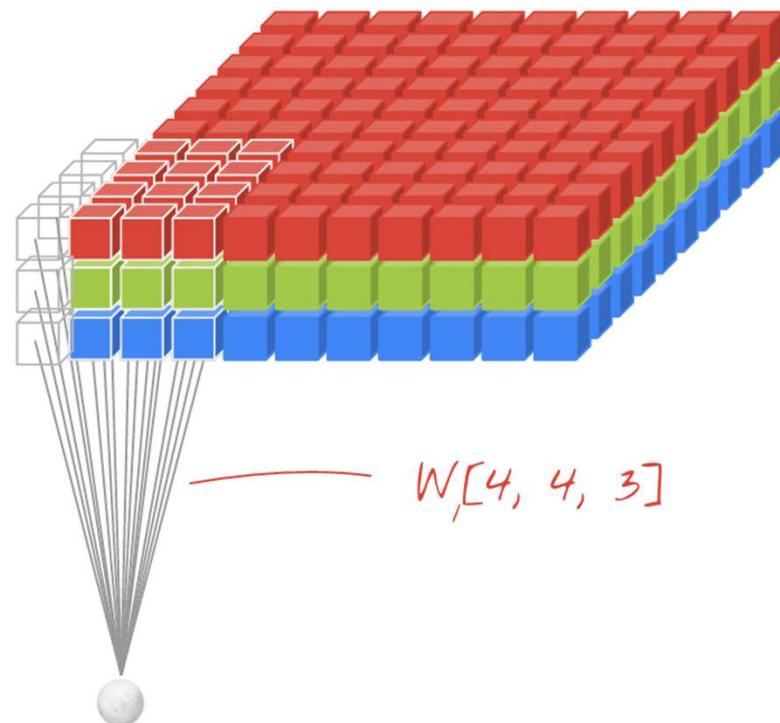
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



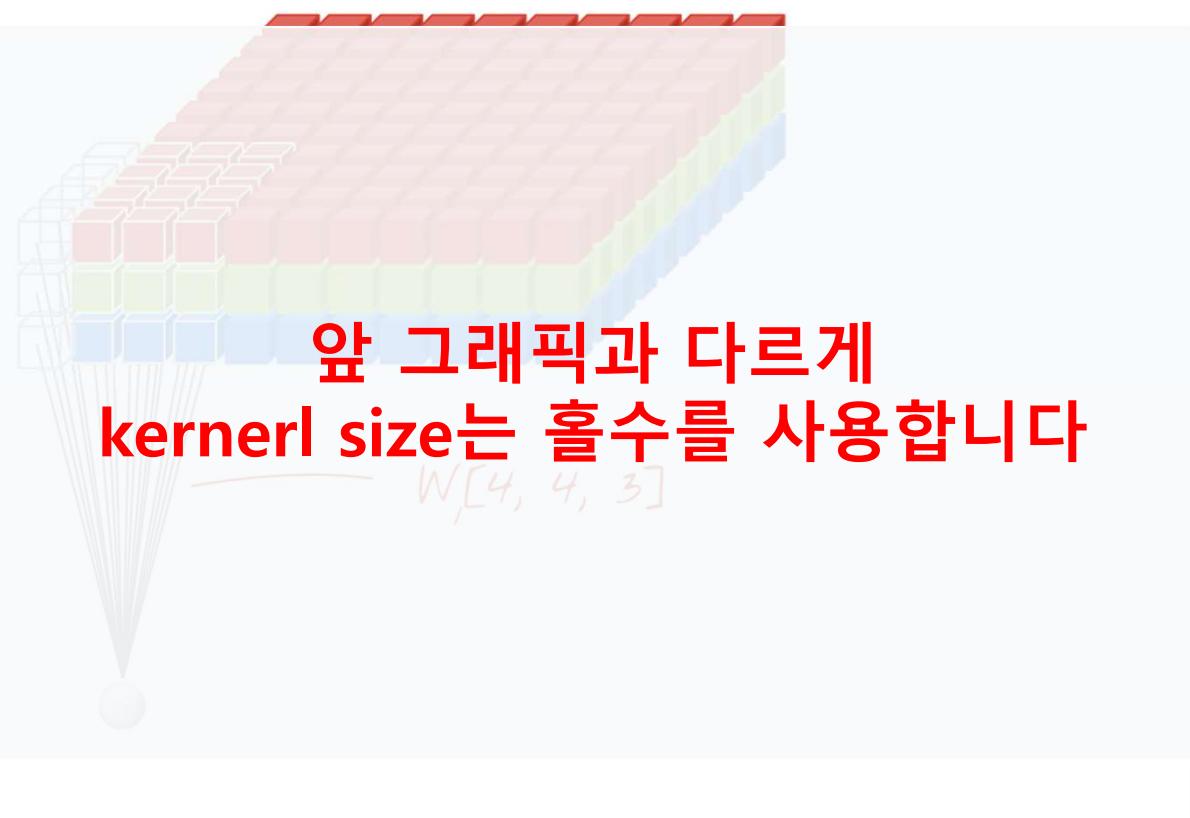
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



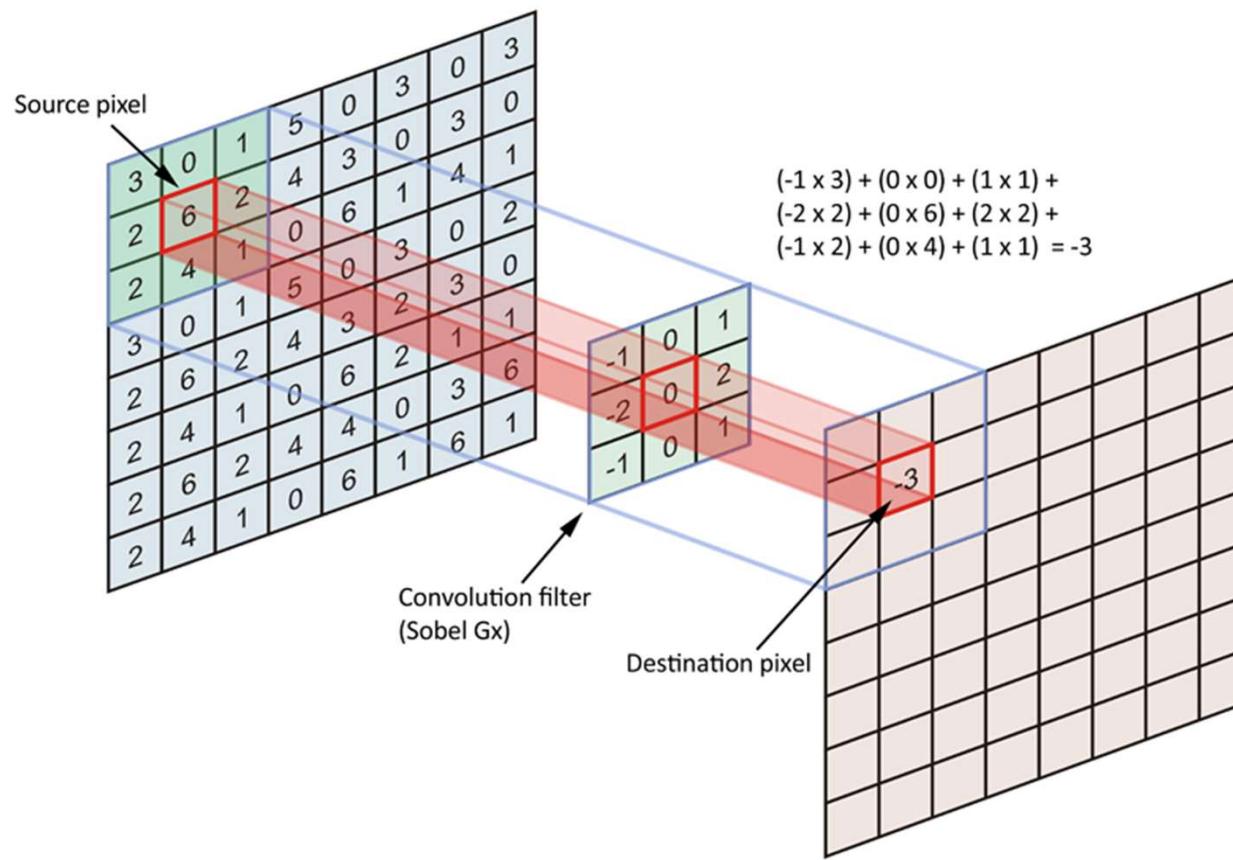
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



픽셀 하나를
중심으로 주변의
특징점 찾는것을
기본으로 하기
때문에

그러므로
나를 중심으로
각면의 갯수+1
 $2*n+1$

그러므로 홀수

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

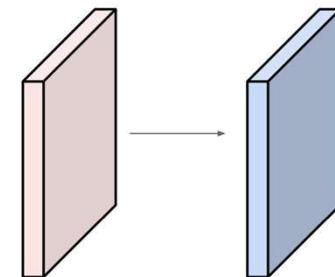
Pooling Layer(pool)

Fully Connected Layer(FC)

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?



2. Convolution Neural Network(CNN)

Convolutional
Neural Network

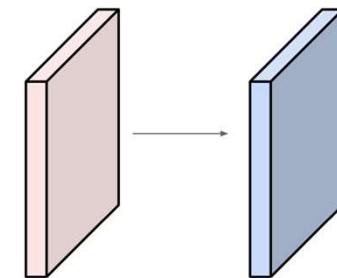
Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride **1**, pad **2**



Output volume size:
 $(32+2*2-5)/1+1 = 32$ spatially, so
32x32x10

Output size:
(N - F) / stride + 1

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

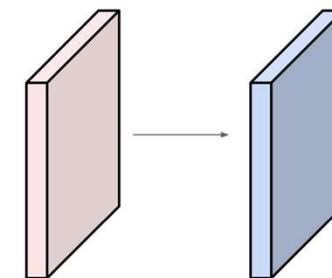
Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

2. Convolution Neural Network(CNN)

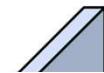
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

Examples time:



kernel(=filter) 사이즈가
 3×3 이 아니라 1×1 이면 어떻게 될까요?

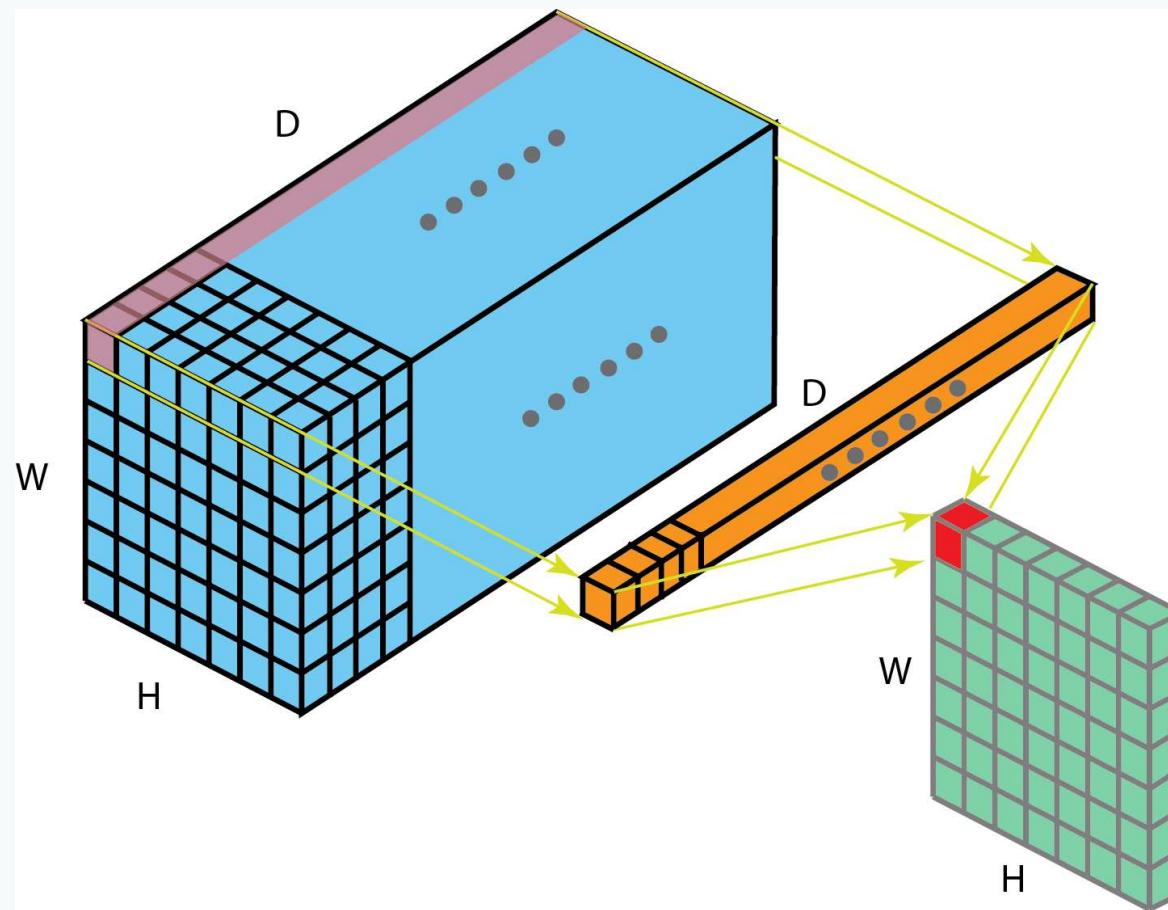
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



만약 kernel을
1x1으로 선언하면
왼쪽과 같이 계산

2. Convolution Neural Network(CNN)

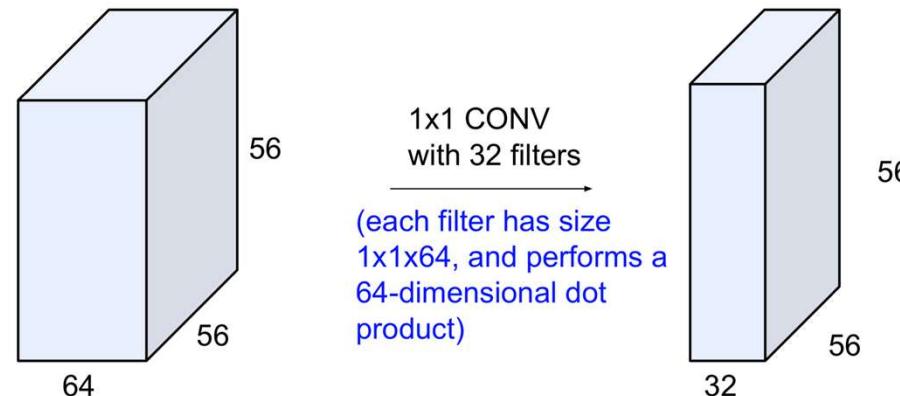
Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

(btw, 1x1 convolution layers make perfect sense)



즉 feature 픽셀 하나당 w가 곱해지는 형태
feture map의 크기는 padding 없이 유지 하되
feature의 채널의 크기만 줄어는 형태.

결과적으로 feture 갯수를 줄이는 형태(모델 크기 감소에 사용 되기도 함)

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

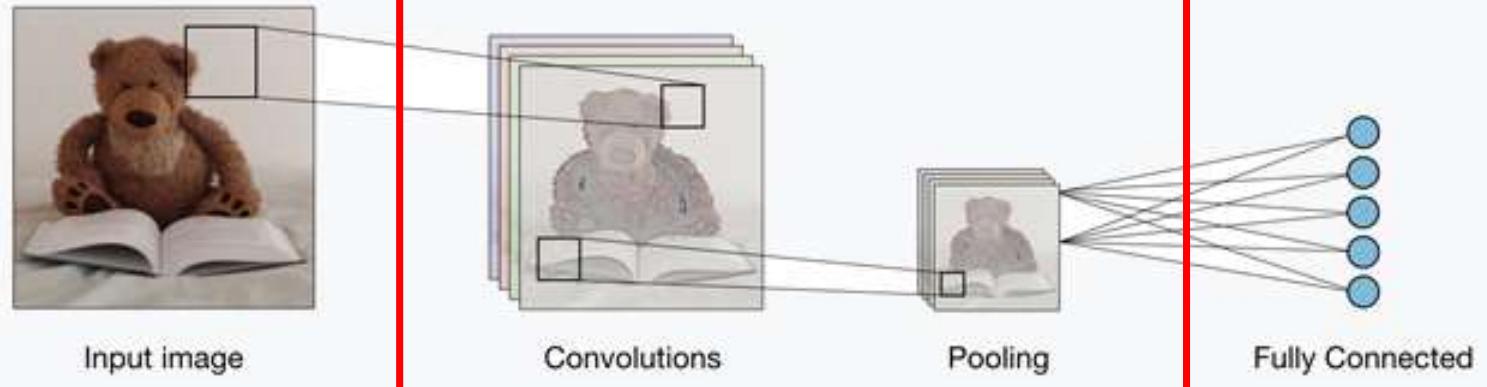
Pooling Layer(pool)

Fully Connected Layer(FC)

선언된 크기의
Filter를 image에
Override하면서
Feature를 추출

추출된 Feature에서
도드라지(중요한)는
Feature를 추출

최종 추출된 Feature
기반으로 분류



Filter를 통해 activation map(=feature map)에
서 더 도드라지는 특징을 추출합니다. 이 과정을
Convolution / pooling이라 하며 이루어져 있다

Max pooling or avg pooling 이 대표적입니다.

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

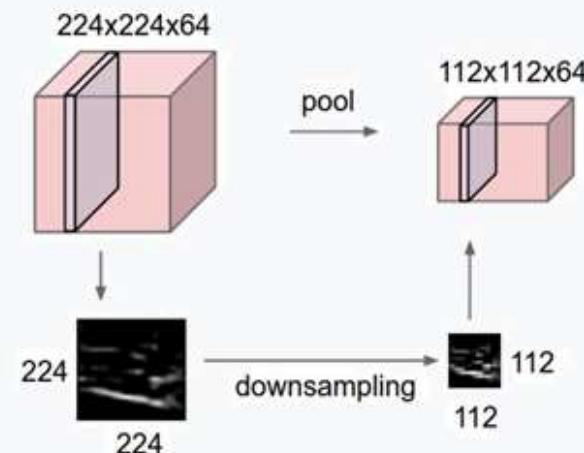
Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



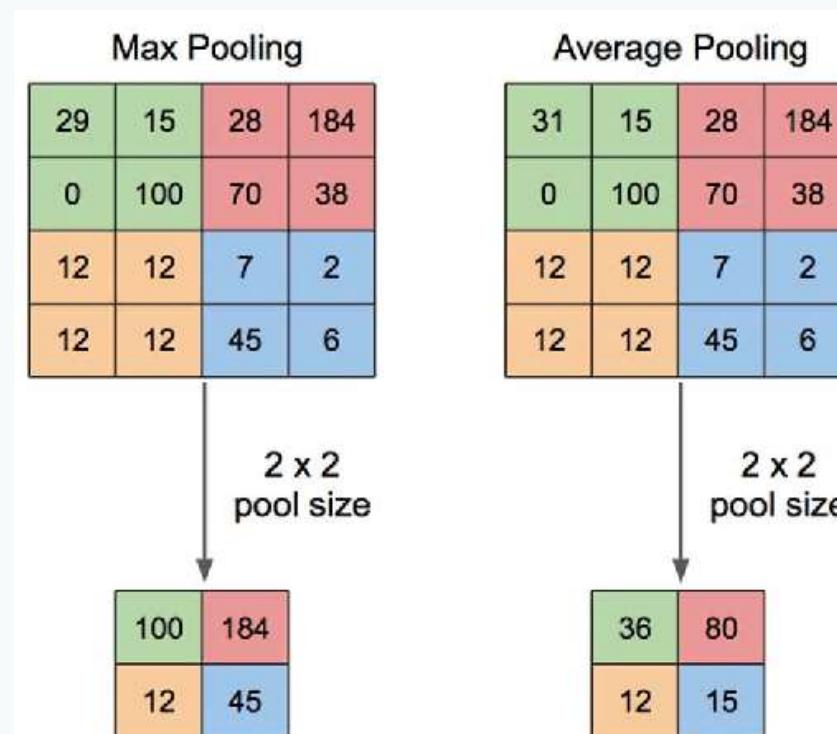
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



즉 filter(=kernel) 을 통해 feature을 만들었다면 만들어진 feature map에서

- 큰 특징을 가지는 값만 모아서 새로운 feature map을 만드는 과정
- 평균적인 특징값을 추출하여 새로운 feature map을 만드는 과정

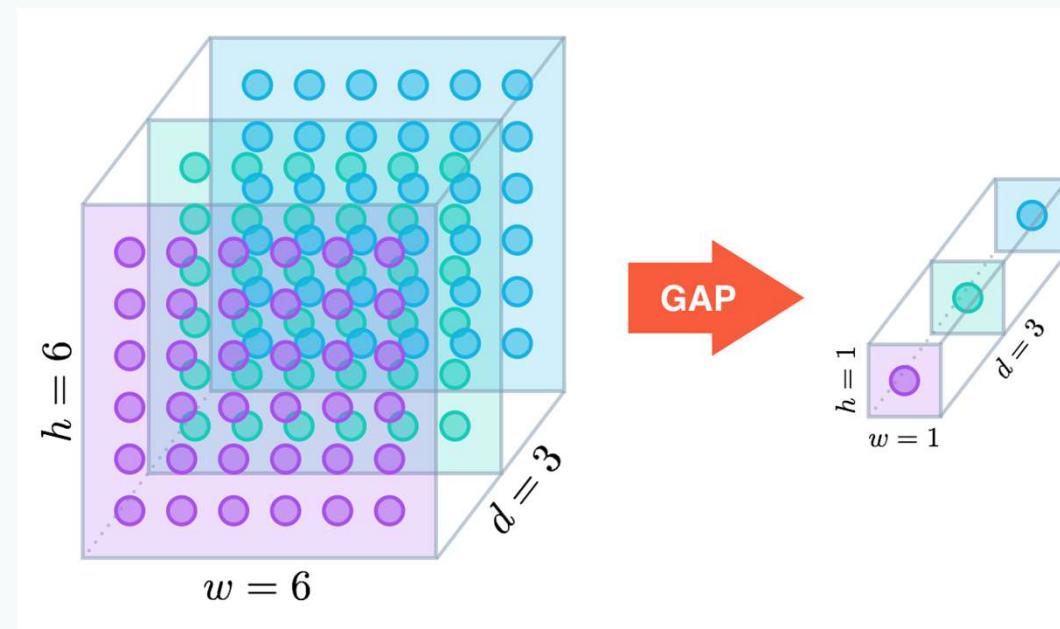
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



Average Pooling은 Global Average Pooling으로 주로 사용되어지고 있음.

이것은 Fully Connected 대신 분류 역할기 대체방법 중 하나로 쓰임

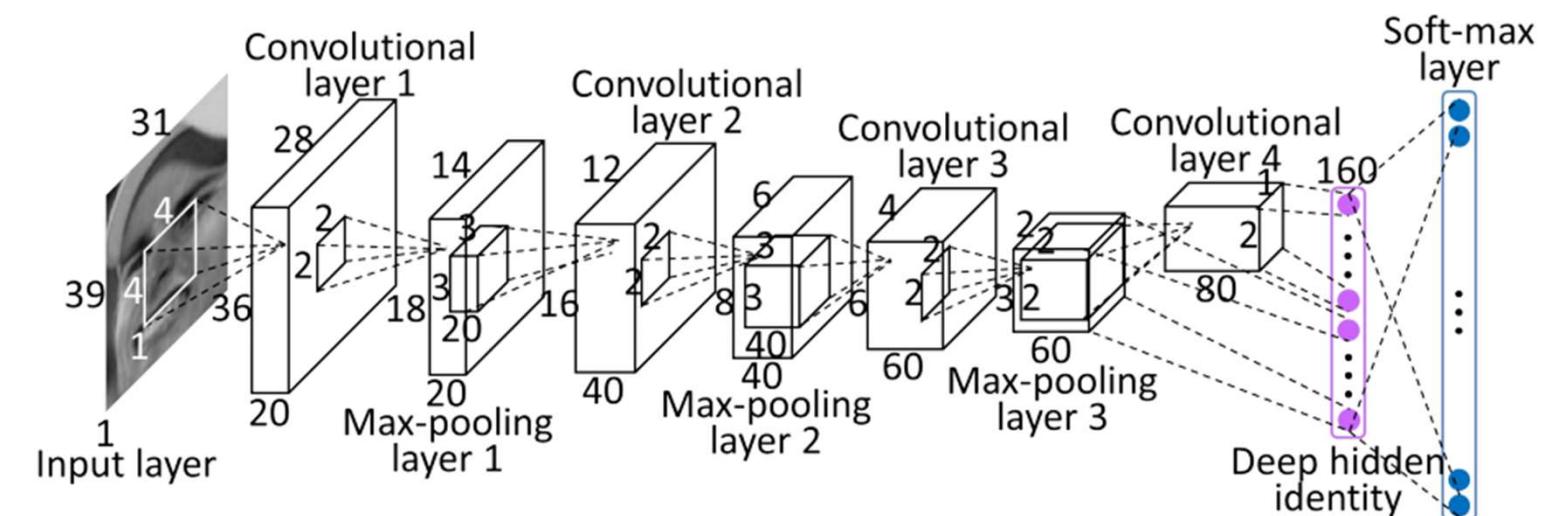
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



Reference: <https://www.cnblogs.com/yxwkf/p/3831310.html>

<http://blog.cs'enwei2>

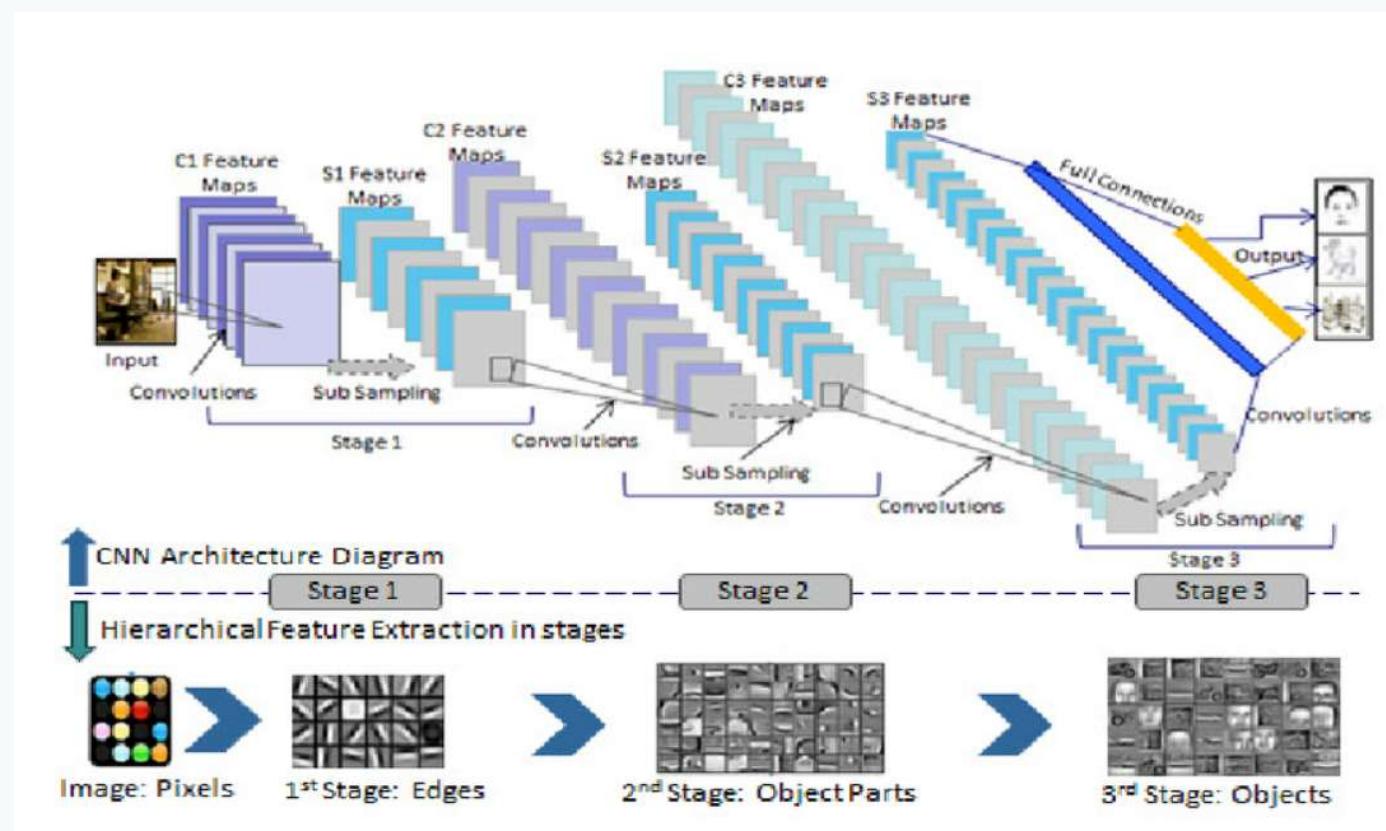
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



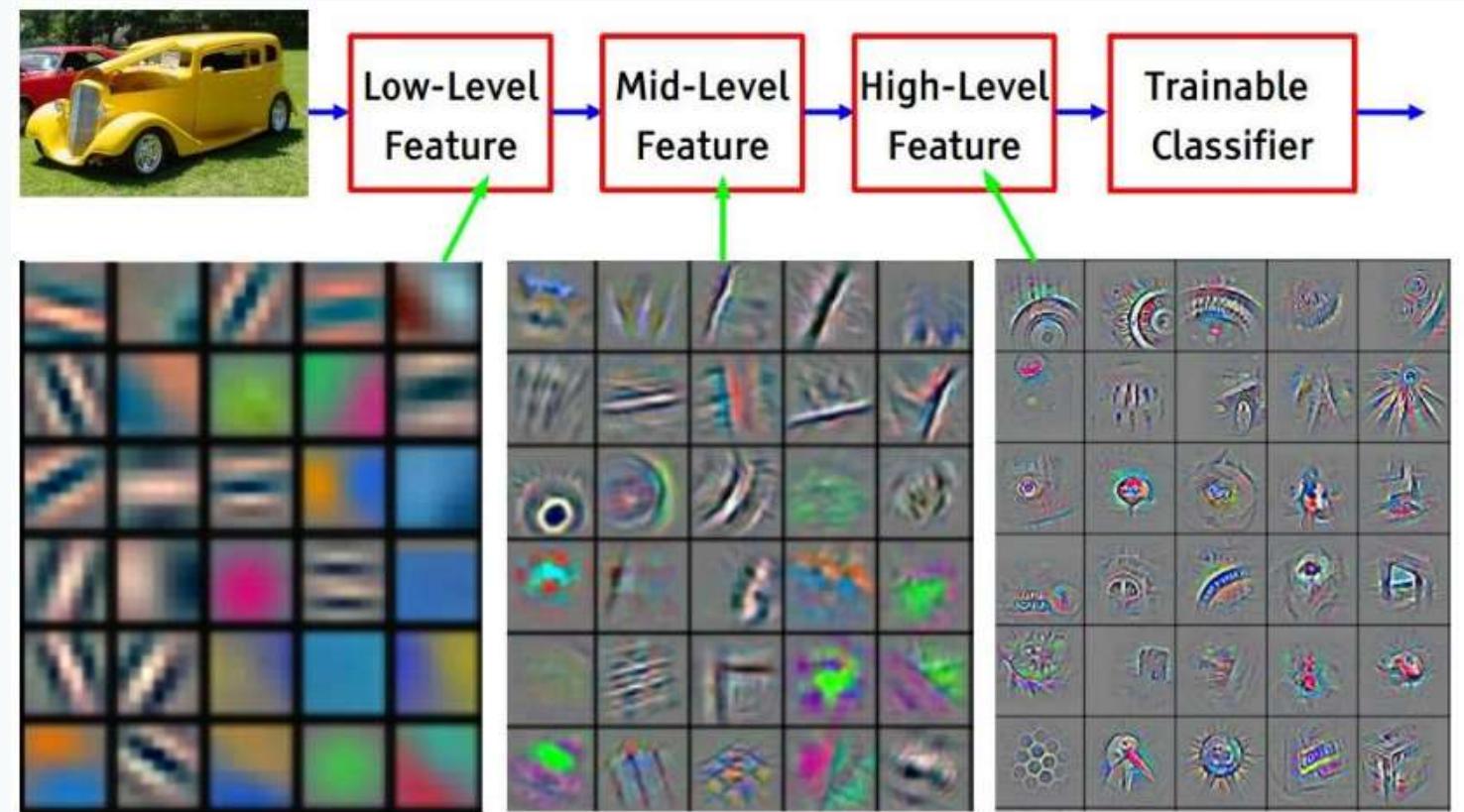
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



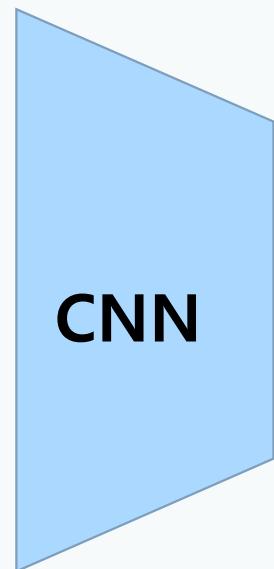
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



NN 입력을
맞춰 주기 위해

Flatten

0
120
45
111
22
...
8
0

Fully Connected

분류작업
(Classifier)

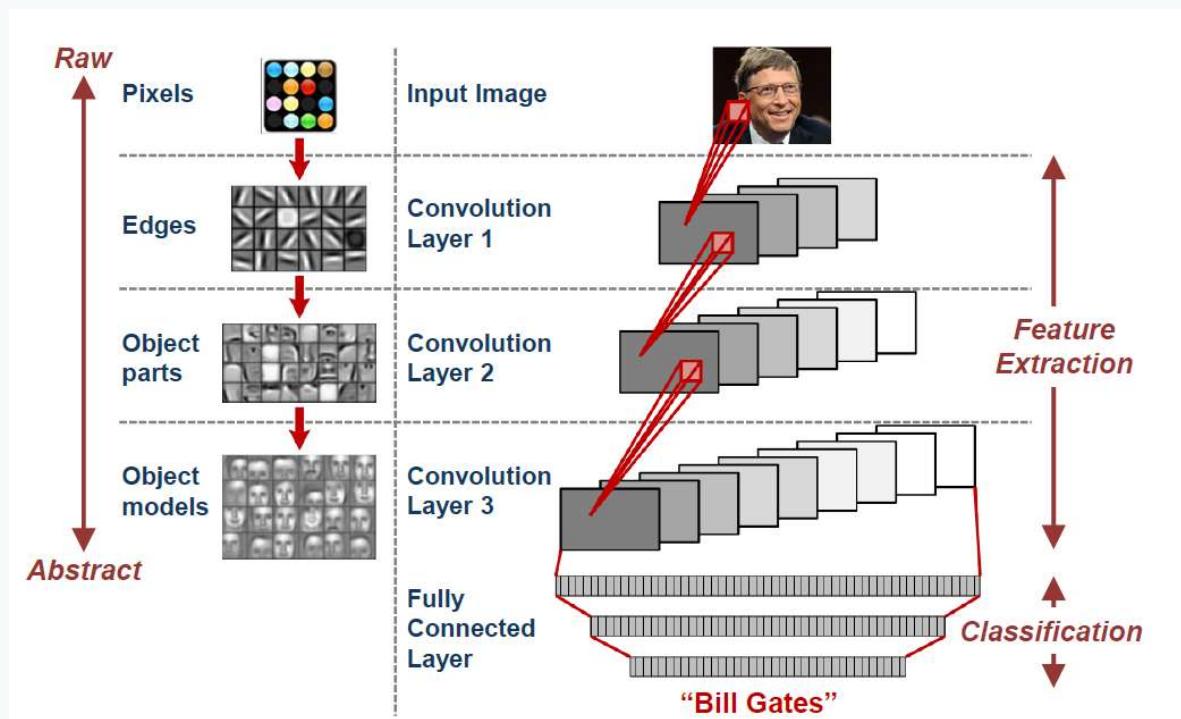
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



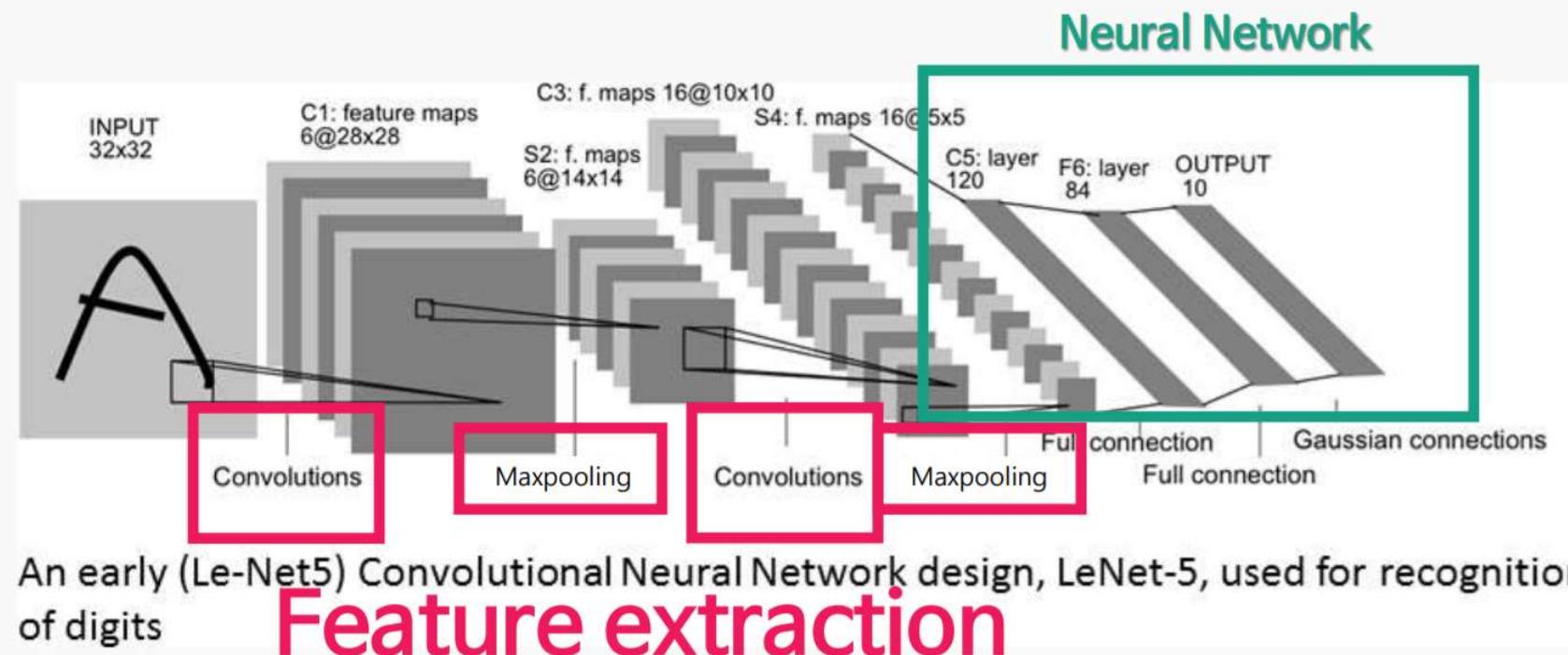
2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)



2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

Convolution Layer는 filter(w, b)을 통해 지역적 특징을 추출.

여기서, 학습은 filter가 가진 w 와 b 가 학습되어짐.

즉, 학습되어진 filter의 w 와 b 가 입력 이미지에 곱해져서 activation map(=feature map) 생성

Max pooling은 학습 되는 파라미터가 없음.

단순히 입력받은 이미지의 특징맵에서 더 도드라지는 특징 추출

Fully Connected = Neural Network = Multi Layer Perceptron 은 앞서 학습한 regression/classification을 함.

중요한 것은 사람이 설계한 특징을 추출하는 것이 아닌 학습을 통해
FC가 Regression/classification을 잘하도록 feature라고 하는
activation map(=feature map)을 잘 추출하도록 Filter의 w, b 가 학습을 통해 업데이트 됨

2. Convolution Neural Network(CNN)

Convolutional
Neural Network

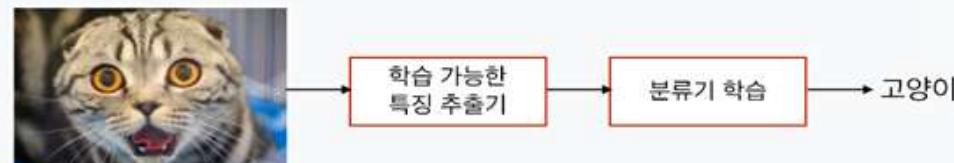
Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

CNN의 장점

- 딥러닝 기반 이미지 분류 프로세스
 - 특징 추출 모듈과 분류기를 하나의 거대한 뉴럴 네트워크로 구성
 - 특징 추출은 컨볼루션 레이어가, 분류기는 FC 레이어가 주로 담당
 - 특징 추출과 분류를 한번에 학습할 수 있음 (end-to-end model)
 - 높은 성능, 그러나 데이터가 많이 필요



2. Convolution Neural Network(CNN)

Convolutional
Neural Network

Convolution Layer(conv)

Pooling Layer(pool)

Fully Connected Layer(FC)

왜 CNN을 사용해야 하는가?

- **Local connectivity**

- 뉴런은 이전 레이어의 뉴런과 모두 연결되지 않고, 작은 영역만 연결
- 이 작은 영역을 receptive field이라고 함.
- 뇌에서 시각 신호를 처리할 때 하나의 뉴런이 시각 정보의 일정 부분만 처리하는 과정과 유사한 매커니즘

- **Parameter sharing**

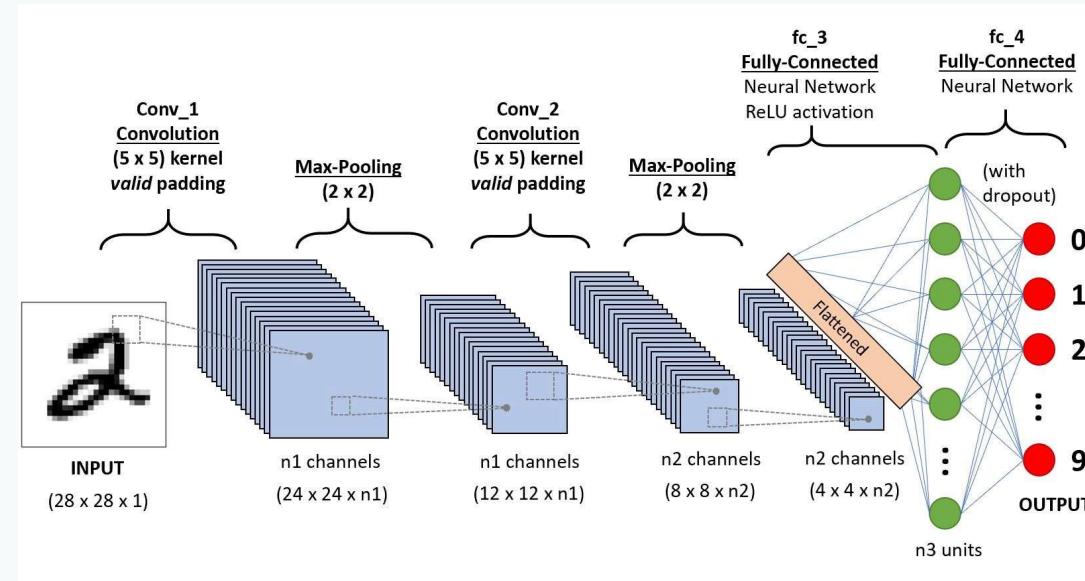
- 동일한 계수를 갖는 filter를 전체 영상에 반복 적용하여 변수를 획기적으로 줄임

- **Equivariant representation**

- Topology 변화(translational)에 무관한 invariance를 얻을 수 있음.

3. ConvolutionNeuralNetwork 실습

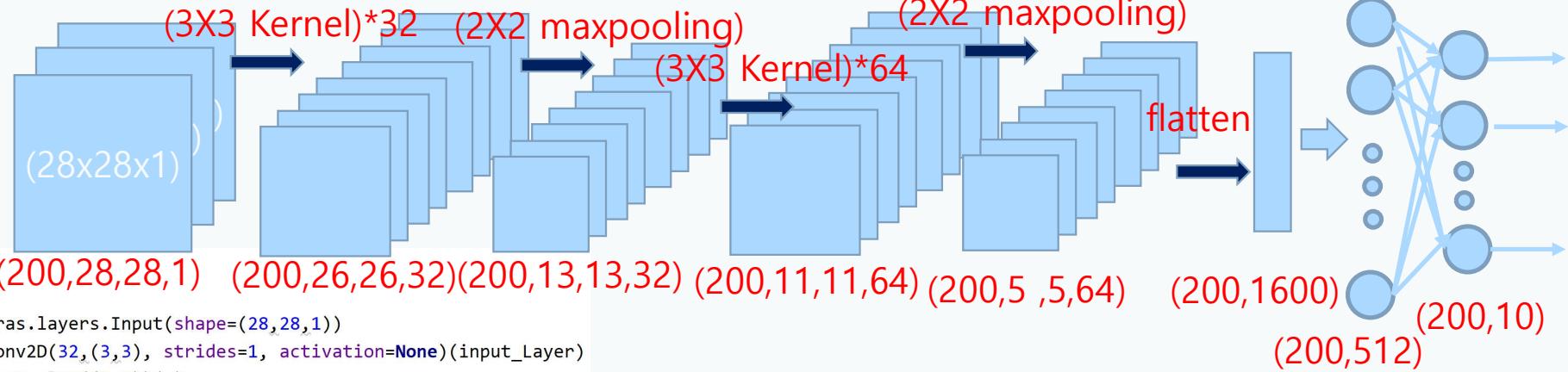
CNN(Convolution Neural Network) 기반 이미지 분류



나만의 MNIST를 분류하는
CNN 모델을 설계하고 학습해 보겠습니다.

3. ConvolutionNeuralNetwork 실습

CNN(Convolution Neural Network) 기반 MNIST 분류



```
# CNN 모델 설계.
## 모델
input_Layer = tf.keras.layers.Input(shape=(28,28,1))
x=tf.keras.layers.Conv2D(32,(3,3), strides=1, activation=None)(input_Layer)
x=tf.keras.layers.MaxPool2D((2,2))(x)
x=tf.keras.layers.Conv2D(64,(3,3),strides=1,activation=None)(x)
x=tf.keras.layers.MaxPool2D((2,2))(x)
x=tf.keras.layers.Flatten()(x)
x= tf.keras.layers.Dense(512, activation='relu')(x)
Out_Layer= tf.keras.layers.Dense(10, activation='softmax')(x)

model = tf.keras.Model(inputs=[input_Layer], outputs=[Out_Layer])
model.summary()

modelpath="./CNN_MNIST_model/{epoch:02d}-{val_loss:.4f}.hdf5"
callback_list=[tf.keras.callbacks.ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1, save_best_only=True),
              tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)]
]

# 모델의 실행
history = model.fit(X_train, Y_train, validation_split=0.2, epochs=5, batch_size=200, verbose=1, callbacks=callback_list)
```

```
loss_function=tf.keras.losses.categorical_crossentropy
optimizer=tf.keras.optimizers.RMSprop(lr=0.0001)
metric=tf.keras.metrics.categorical_accuracy
model.compile(loss=loss_function,
              optimizer=optimizer,
              metrics=[metric])
```

입력 데이터의 20%를 검증 데이터로 활용

3. ConvolutionNeuralNetwork 실습

CNN(Convolution Neural Network) 기반 Fashion MNIST 실습



This is a dataset of 60,000 28x28 grayscale images of 10 fashion categories, along with a test set of 10,000 images. This dataset can be used as a drop-in replacement for MNIST. The class labels are:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Returns

Tuple of Numpy arrays: (`x_train`, `y_train`), (`x_test`, `y_test`).

`x_train`, `x_test`: uint8 arrays of grayscale image data with shape (`num_samples`, 28, 28).

`y_train`, `y_test`: uint8 arrays of labels (integers in range 0-9) with shape (`num_samples`,).

3. ConvolutionNeuralNetwork 실습

CNN(Convolution Neural Network) 기반 MNIST 분류 테스트

```
# MNIST 데이터 불러오기  
(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.mnist.load_data()
```

데이터 로드

```
M= cv2.getRotationMatrix2D((28/2, 28/2),70, 1)  
test_image = cv2.warpAffine(X_train[3], M,(28, 28))  
test_image_reshape = test_image.reshape(1, 28, 28, 1).astype('float32')
```

회전 행렬 구하기
TEST할 이미지 하나에
회전행렬 곱하기

```
model = tf.keras.models.load_model('./CNN_MNIST_model\\05-0.1416.hdf5') # 모델을 새로 불러옴  
Y_prediction = model.predict(test_image_reshape)  
index=np.argmax(Y_prediction)  
plt.imshow(test_image, cmap='Greys')  
plt.xlabel(str(index))  
plt.show()
```

모델 불러오고 테스트하기
출력 값중에 가장 큰값을 가지는
index 가져오기
(가장 활성화된 perceptron 위치)
가져온 index로 perceptron의
출력값 가져오기

3. ConvolutionNeuralNetwork 실습

CNN MODEL VISUAL

모델을 잘 설계했는지 확인하고 싶을때 어떻게 할까요???

3. ConvolutionNeuralNetwork 실습

CNN MODEL VISUAL

1. model.summary()
2. Netron (<https://github.com/gaussian37/netron>)
3. Graphviz

Ubuntu : pip3 install graphviz==0.8.1

pip3 install pydot==1.2.3

window : https://graphviz.gitlab.io/_pages/Download/Download_windows.html

에서 msi 다운받아서 설치

pycharm에서 터미널 열어서 pip install pydot==1.2.3

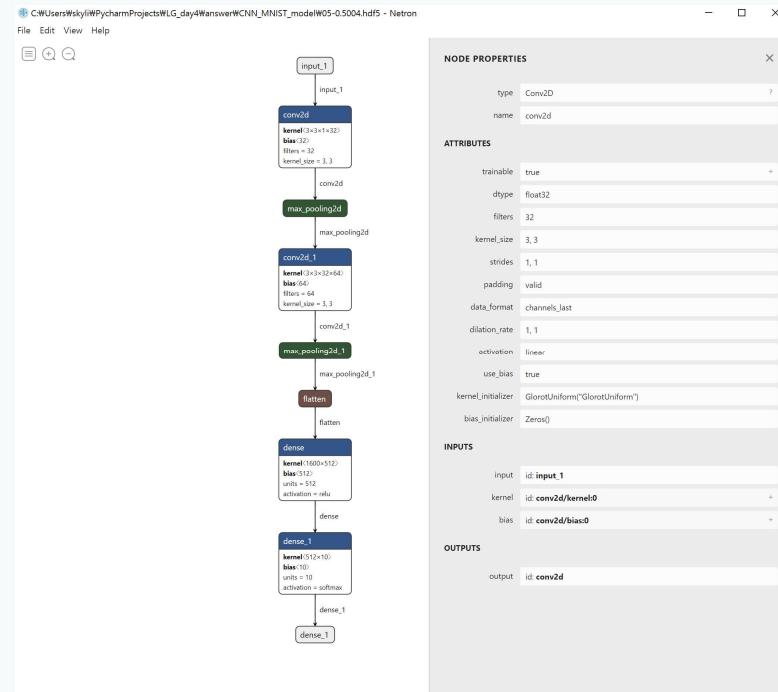
3. ConvolutionNeuralNetwork 실습

CNN MODEL VISUAL

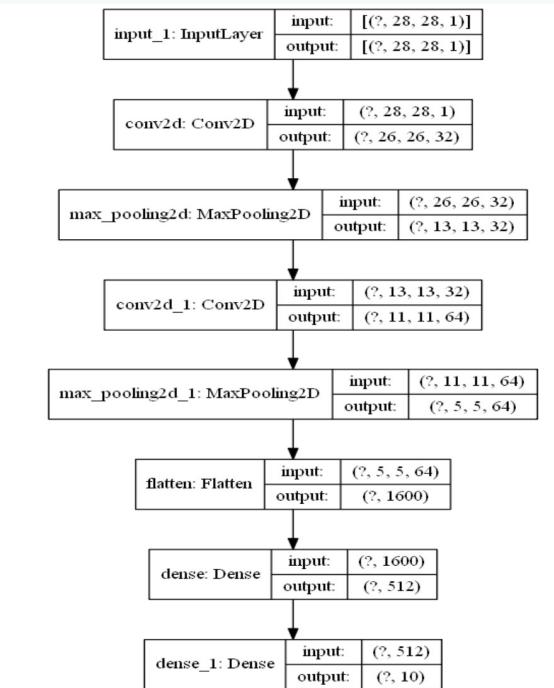
model.summary()

```
Model: "model"
Layer (type)      Output Shape       Param #
=====
input_1 (InputLayer) [(None, 28, 28, 1)] 0
conv2d (Conv2D)    (None, 26, 26, 32) 320
max_pooling2d (MaxPooling2D) (None, 13, 13, 32) 0
conv2d_1 (Conv2D)  (None, 11, 11, 64) 18496
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64) 0
flatten (Flatten) (None, 1600) 0
dense (Dense)     (None, 512) 819712
dense_1 (Dense)   (None, 10) 5130
=====
Total params: 843,658
Trainable params: 843,658
Non-trainable params: 0
=====
Train on 48000 samples, validate on 12000 samples
```

netron

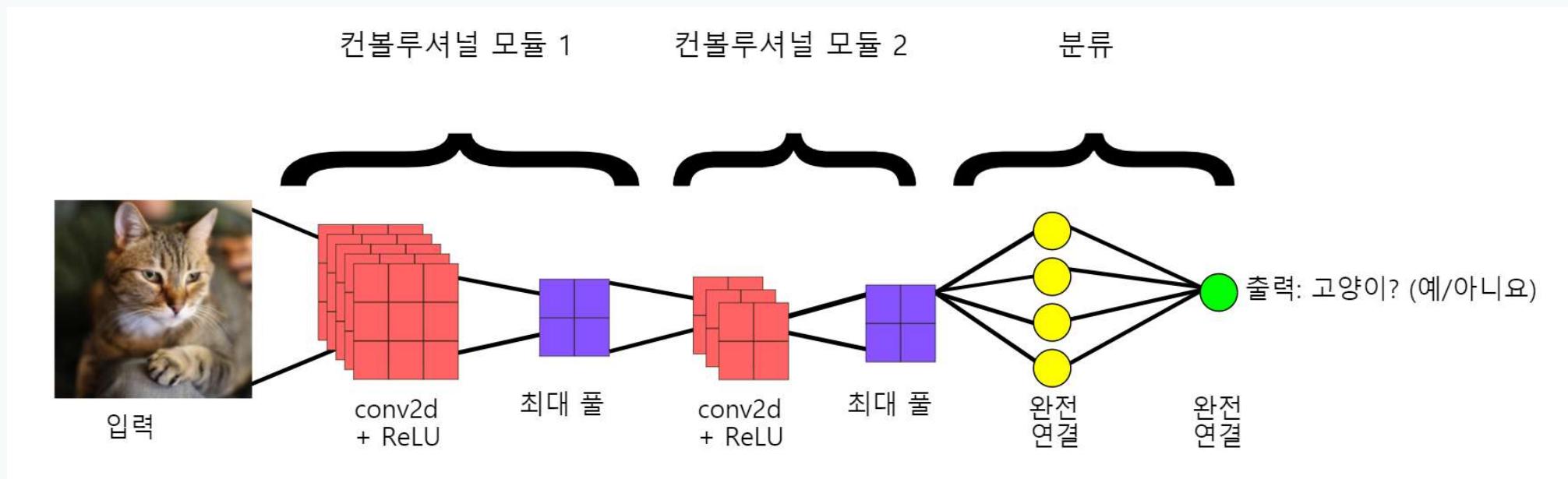


graphviz



3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델



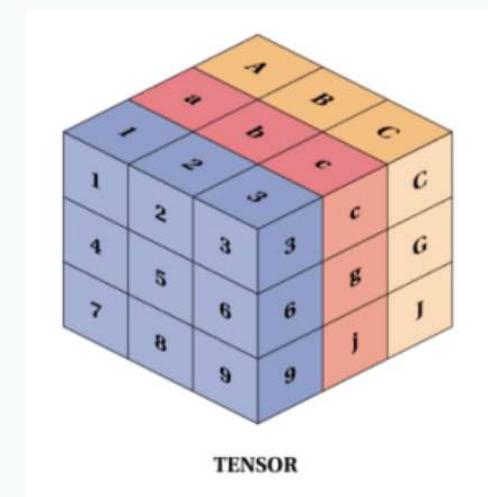
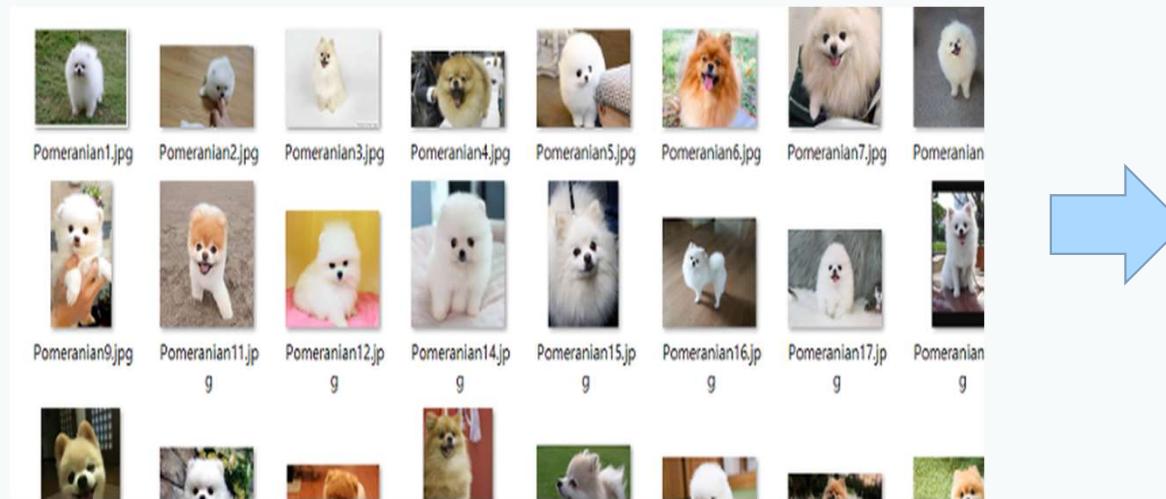
3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

mnist는 1채널(흑백)

이제부터는 RGB(컬러) 이미지를 다룰 예정.

IMAGE가 담긴 폴더에서 이미지를 가져와서 학습



3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

이미지 읽기 → RGB 픽셀로 디코딩 → 텐서화



ImageDataGenerator API 사용

```
train_datagen = ImageDataGenerator
```

```
train_generator = train_datagen.flow_from_directory
```

```
model.fit_generator (구버전)
```

```
model.fit ( 신버전 tensorflow2.x)
```

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

ImageDataGenerator API

라벨 읽어 올 때 MODE

- `class_mode`: one of "categorical", "binary", "sparse", "input", "other" or None. Default: "categorical". Mode for yielding the targets:
 - "binary" : 1D numpy array of binary labels,
 - "categorical" : 2D numpy array of one-hot encoded labels. Supports multi-label output.
 - "sparse" : 1D numpy array of integer labels,
 - "input" : images identical to input images (mainly used to work with autoencoders),
 - "other" : numpy array of `y_col` data,
 - `None` , no targets are returned (the generator will only yield batches of image data, which is useful to use in `model.predict_generator()`).

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

ImageDataGenerator API 사용해서 분류 모델 작성하기.

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

```
## DATA 만들기
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator()

## 학습데이터를 불러와서 파싱함.
train_generator = train_datagen.flow_from_directory(
    directory=train_dir,           # 타깃 디렉터리
    target_size=(150, 150),         # 모든 이미지를 150 × 150 크기로 바꿉니다
    batch_size=20,
    interpolation='bilinear',     ## resize / interpolation 기법
    color_mode='rgb',
    shuffle='True',
    # binary_crossentropy 손실을 사용하기 때문에 이진 레이블이 필요합니다
    class_mode='binary') # binary, categorical , sparse , input

## class의 인덱스를 확인.
print(train_generator.class_indices)
print(train_generator.classes)

validation_generator = valid_datagen.flow_from_directory(
    directory=validation_dir,
    target_size=(150, 150),
    batch_size=20,
    shuffle='True',|
    interpolation='bilinear', ## resize / interpolation 기법
    color_mode='rgb',
    class_mode='binary')
```

이미지 데이터를 읽어올 api 선언

flow_from_directory() 로 부터

1. **directory** 옵션에 지정된 폴더로 부터 데이터를 읽어 옴
2. 읽어올 때 이미지 사이즈, 배치 사이즈, 컬러 모드, 셔플, 라벨 모드를 지정 할 수 있음.

이것은 **generator** 함수로 위의 함수가 call이 될 때마다 동작 함.

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

```
## DATA 만들
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator()

## 데이터를 불러와서 파싱함.
train_generator = train_datagen.flow_from_directory(
    directory=train_dir,           # 타깃 디렉터리
    target_size=(150, 150),         # 모든 이미지를 150 × 150 크기로 바꿉니다
    batch_size=20,
    interpolation='bilinear',     ## resize / interpolation 기본
    color_mode='rgb',
    shuffle='True',
    # binary_crossentropy 손실을 사용하기 때문에 이진 레이블이 필요합니다
    class_mode='binary') # binary, categorical , sparse , input
```

```
history = model.fit(
    train_generator,
    steps_per_epoch=(len(os.listdir(train_cats_dir)) + len(os.listdir(train_dogs_dir)))/batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    # callbacks=callbacks_list,
    validation_freq=1
)
```

model.fit_generator()를 사용해 학습(구버전)

model.fit() 사용해 학습(신버전-2.1이상)

train_generator를 call해서 데이터를 가져와 학습

validation_generator를 call해서 데이터를 가져와
검증.

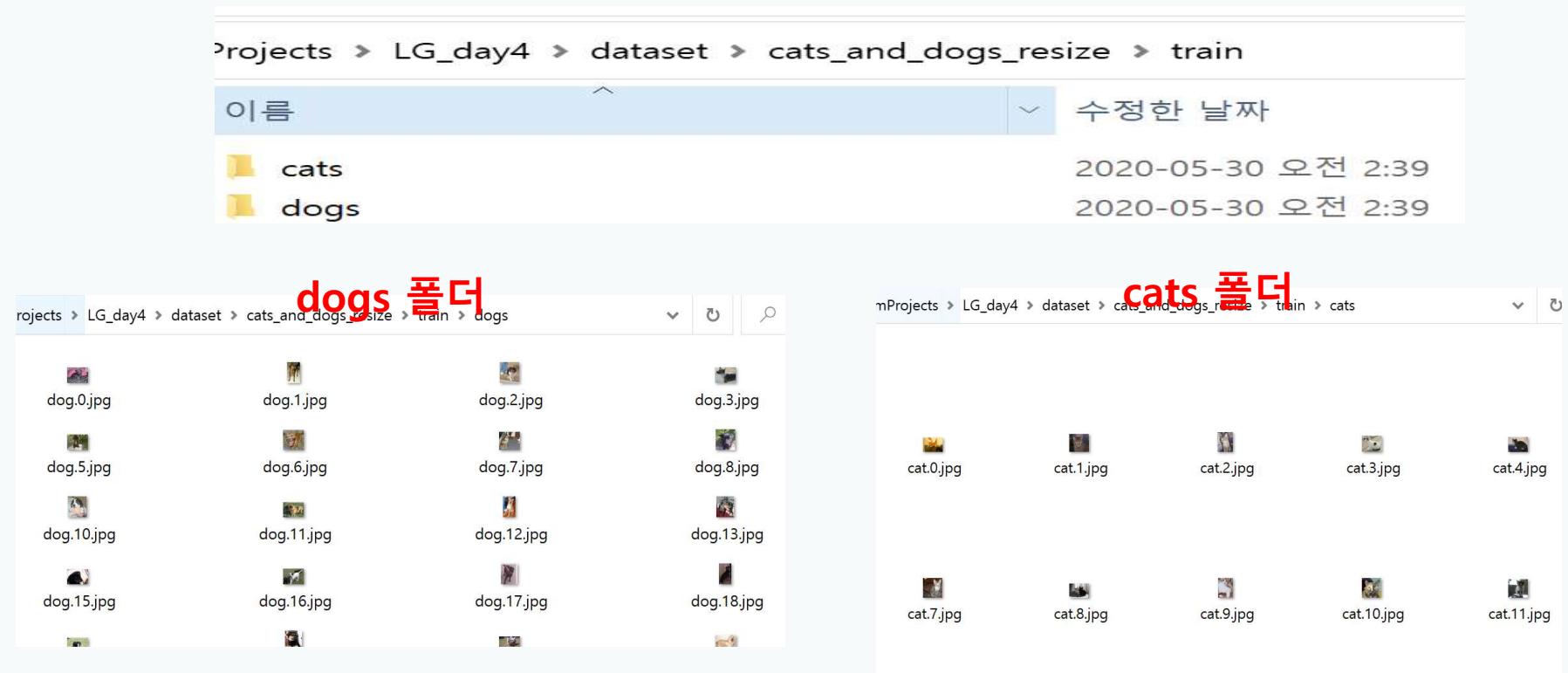
validation_freq을 통해 몇번의 epochs 마다
검증할지 지정 가능

step_per_epoch는 1번의 에포크에서 **batch_size**만큼
데이터를 가져와서 몇번 학습할것인지 지정
(보통 전체 데이터셋 길이/ 배치사이즈로 지정)

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

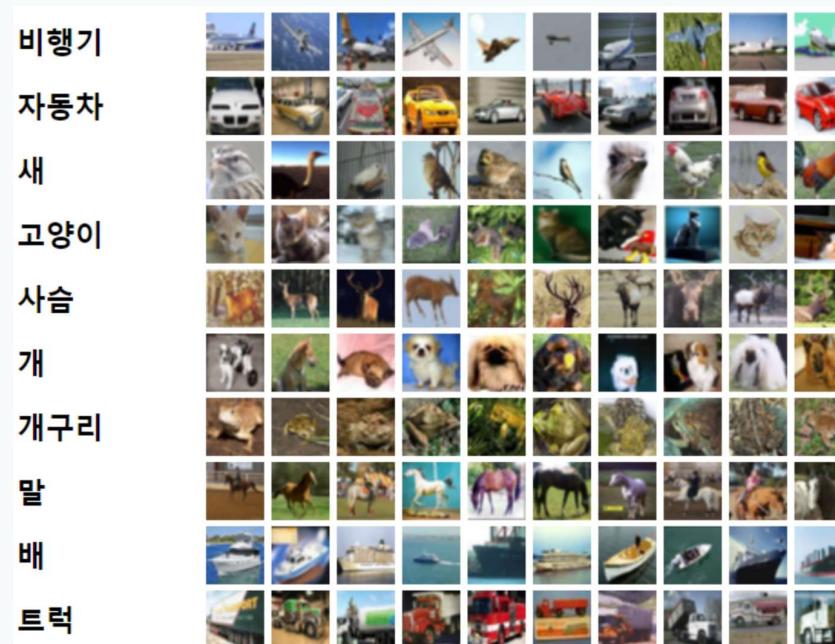
데이터가 담긴 폴더



Practice : P_04_07_dog_cat_binary_classification.ipynb Practice : P_04_07_dog_cat_multi_classification.ipynb

3. ConvolutionNeuralNetwork 실습

CNN image - cifar 10 데이터 분류



cifar10 data

3. ConvolutionNeuralNetwork 실습

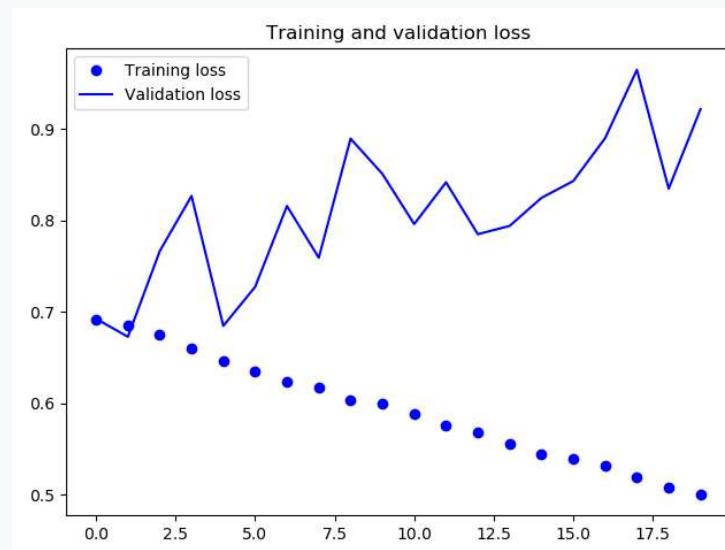
CNN image - cifar 10 데이터 분류

본 실습에서는 cpu 연산을 고려하여 이미지 크기를 줄이고
4개의 class(비행기, 새, 자동차, 고양이) 의 데이터만 dataset 폴더
안에 cifar_10_small 폴더안에 이미지를 저장
이 데이터를 가지고 분류하는 모델 설계 후 학습

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

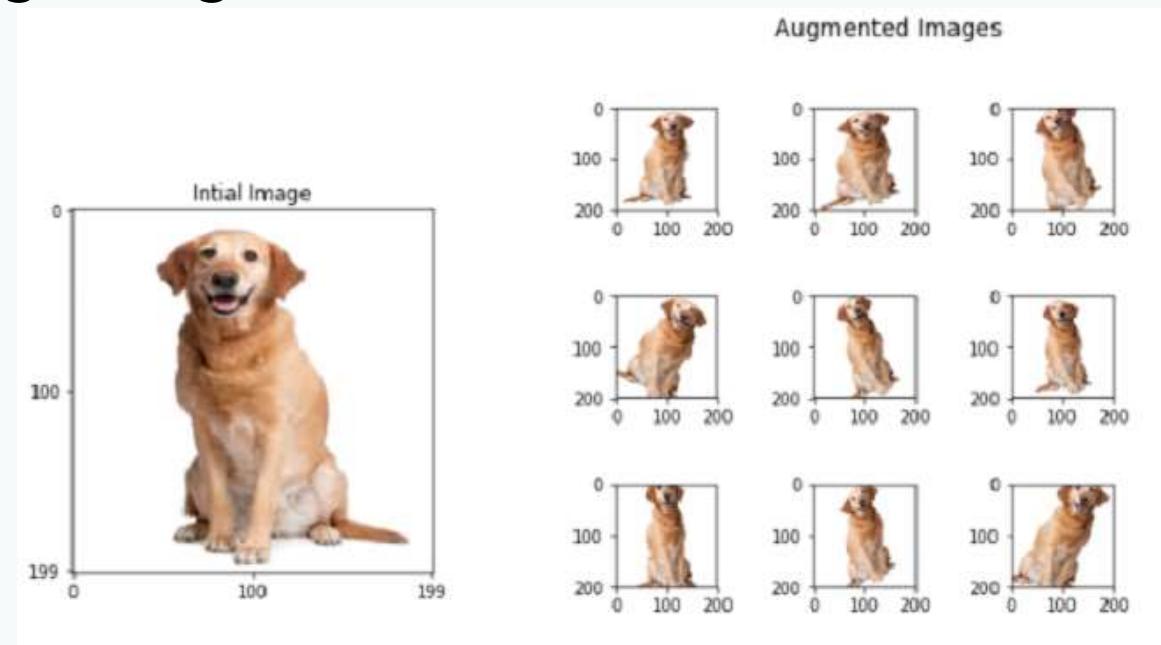
overfitting 경향성이 보입니다.



3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

image augmentation 적용하여 오버피팅 막기



3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

```
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator()  
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator()  
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
```

데이터를 읽어올때 옵션이 없음



```
argu_train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
    rotation_range=40,  
    width_shift_range=0.2, ## 가로로 이동 비율  
    height_shift_range=0.2, ## 세로로 이동 비율  
    shear_range=0.2, # 전단의 강도  
    zoom_range=0.2, ## 확대 와 축소 범위 [1-0.2 ~ 1+0.  
    horizontal_flip=True) ## 수평기준 플립  
  
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator()  
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
```

- rotation_range : 이미지 회전 범위 (degrees)
- width_shift , height_shift : 그림을 수평 또는 수직으로 랜덤하게 평행 이동시키는 범위 (원본 가로, 세로 길이에 대한 비율 값)
- rescale : 원본 영상은 0-255의 RGB 계수로 구성되는데, 이 같은 입력값은 모델을 효과적으로 학습시키기에 너무 높습니다 (통상적인 learning rate를 사용할 경우). 그래서 이를 1/255로 스케일링하여 0-1 범위로 변환시켜줍니다. 이는 다른 전처리 과정에 앞서 가장 먼저 적용됩니다.
- shear_range : 임의 전단 변환 (shearing transformation) 범위
- zoom_range : 임의 확대/축소 범위
- horizontal_flip : True로 설정할 경우, 50% 확률로 이미지를 수평으로 뒤집습니다. 원본 이미지에 수평 비대칭성이 없을 때 효과적입니다. 즉, 뒤집어도 자연스러울 때 사용하면 좋습니다.
- fill_mode 이미지를 회전, 이동하거나 축소할 때 생기는 공간을 채우는 방식

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

학습 모델을 불러와 강아지 고양이 분류 모델

테스트 하기

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

```
2 names=['cat', 'dog'] ## names[0]= cat, names[1]=dog
3 fnames = sorted([os.path.join(test_dogs_dir, fname) for fname in os.listdir(test_dogs_dir)])
4 # fnames = sorted([os.path.join(test_cats_dir, fname) for fname in os.listdir(test_cats_dir)])
5
6 img_path = fnames[3]
7
8 # 이미지를 읽고 크기를 변경합니다
9 img=cv2.imread(img_path)
0 img_resize=cv2.resize(img,(128,128))
1
2 ## 이미지 회전 변환 메트릭스 구하기
3 M= cv2.getRotationMatrix2D((128/2, 128/2), 0, 1) ## Matrix 생성
4 ## 이미지 이동 변환 메트릭스 구하기
5 M[0, 2]=M[0, 2]+5
6 M[1, 2]=M[1, 2]+3
7 test_image = cv2.warpAffine(img_resize, M,(128, 128) )
8 print(test_image.shape)
9 test_image_reshape = test_image.reshape(1, 128, 128, 3)
0
1 model = tf.keras.models.load_model('cats_and_dogs_binary_classification.hdf5') # 모델을 새로 불러옴
2 y = model.predict(test_image_reshape)
3 if y[0]<=0.5:
4     class_num=0
5 else:
6     class_num=1
7 plt.imshow(test_image)
8 plt.xlabel(names[class_num] + str(y[0]).)
```

지정된 폴더에서 폴더에 있는 이미지 리스트를 읽어와서 주소를 결합해서 frames에 저장

frames에서 테스트할 이미지 한장 선택

cv2.imread를 통해 이미지 읽고
cv2.resize를 통해 학습 모델 입력 차원 맞춰줌

이미지 회전 + 이동

모델의 입력은 (batch_size, w, h, c) 이므로 batch_size를 1로 하여 4차원으로 차원을 조정

모델 로드후에 prediction 하여 결과 받기

test image 보여주고

출력값이 0이면 names에서 0번째 cat,
출력값이 1이면 names에서 1번째 dog
라벨 읽어와서 표기하기

3. ConvolutionNeuralNetwork 실습

CNN image - 강아지/고양이 분류 모델

```
import cv2
names=['cat', 'dog'] ## names[0]= cat, names[1]=dog

model = tf.keras.models.load_model('cats_and_dogs_categorical_classification.hdf5') # 모델을 불러오기

capture = cv2.VideoCapture(0)
capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480) # 640x480 사이즈로 Webcam 불러오기

while True:
    ret, frame = capture.read() # Webcam에서 불러온 영상을 이미지로 읽어오기

    img_resize = cv2.resize(frame, (128, 128)) # 이미지 128x128로 resize 한 후
    test_image_reshape = img_resize.reshape(1, 128, 128, 3).astype('float32') # (batch_size, w, h, c) 차원으로 reshape 하기

    y = model.predict(test_image_reshape) # 불러온 모델 입력 데이터 넣어서 예측값 받아오기
    class_name=names[np.argmax(y)] # [y1, y2, ..] 처럼 class 개수만큼 예측값이 나옴. 여기서 가장 큰값을 출력하는 index 추출하여 name 가져오기

    location = (100, 100 + 20) # 이미지에 text 입력하여 display 하기
    cv2.putText(frame, str(class_name), location, cv2.FONT_ITALIC, 2, (0, 255, 0), 2)

    cv2.imshow("VideoFrame", frame) # Esc 누르면 종료

    if cv2.waitKey(1) > 0: break
```

모델 불러오기

640x480 사이즈로 Webcam 불러오기

Webcam에서 불러온 영상을 이미지로 읽어오기

이미지 128x128로 resize 한 후
(batch_size, w, h, c) 차원으로 reshape 하기

불러온 모델 입력 데이터 넣어서 예측값 받아오기
[y1, y2, ..] 처럼 class 개수만큼 예측값이 나옴.
여기서 가장 큰값을 출력하는 index 추출하여 name 가져오기

이미지에 text 입력하여 display 하기

Esc 누르면 종료

오늘 하루 고생 하셨습니다.
Q & A

참 고 자 료

- cs231n 강의
- 인공지능연구원 AIRI400 이광희 박사님 강의 자료
- 곰가드의 라이브러리 블로그 그림
- 기타 구글 딥러닝 이미지 및 블로그 자료