AI 프로그래밍

- 13주차

인하 공전 컴퓨터 정보과

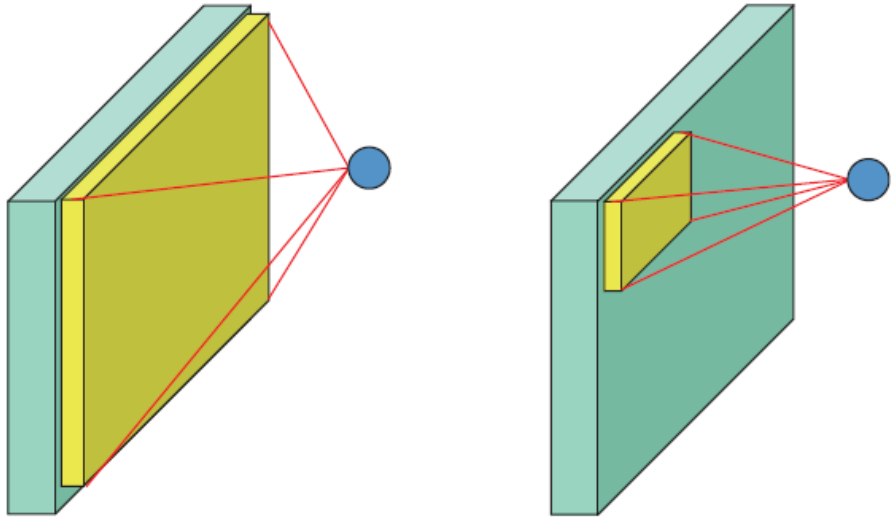
민 정혜

오늘 수업 순서- 딥러닝 예제

- **컨볼루션 신경망 (CNN: Convolutional Neural Network)**

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 컨볼루션(Convolution Neural Network: CNN) 신경망에서는 하위 레이어의 노드들과 상위 레이어의 노드들이 부분적으로만 연결되어 있다.

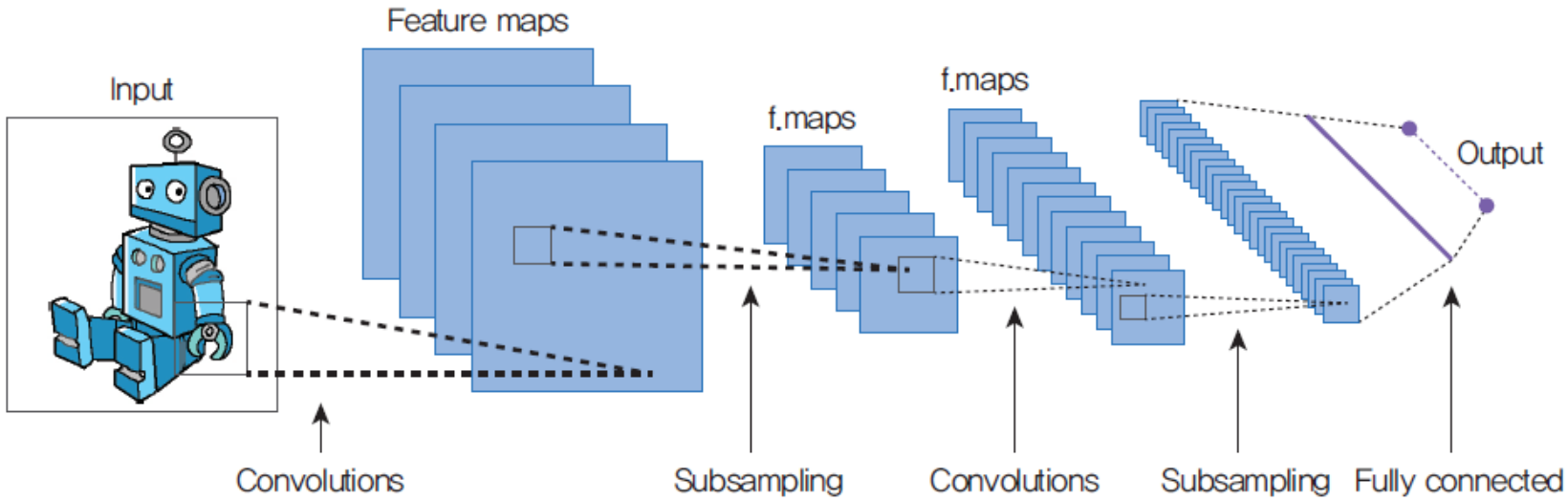


(a) 완전연결 신경망 (b) 컨볼루션 신경망

https://keras.io/api/layers/core_layers/

컨볼루션 신경망 (CNN: Convolutional Neural Network)

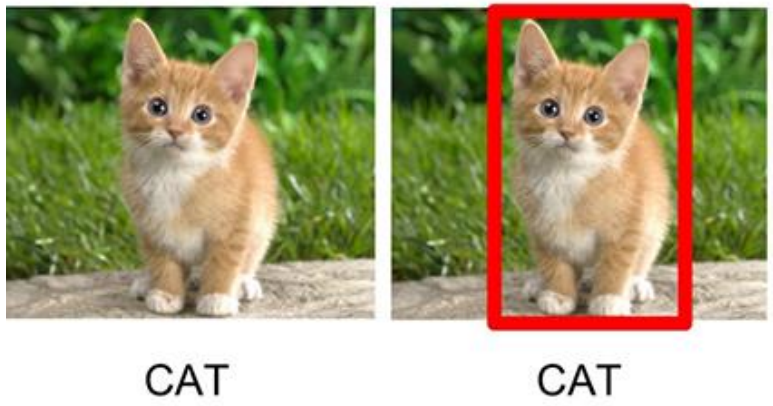
- 컨벌루션 신경망은 모든 신경망 구조 중에서 가장 강력한 성능을 보여주는 신경망 중의 하나이다.
- 컨벌루션 신경망은 2차원 형태의 입력을 처리하기 때문에, 이미지 처리에 특히 적합하다. 신경망의 각 레이어에서 일련의 필터가 이미지에 적용된다.



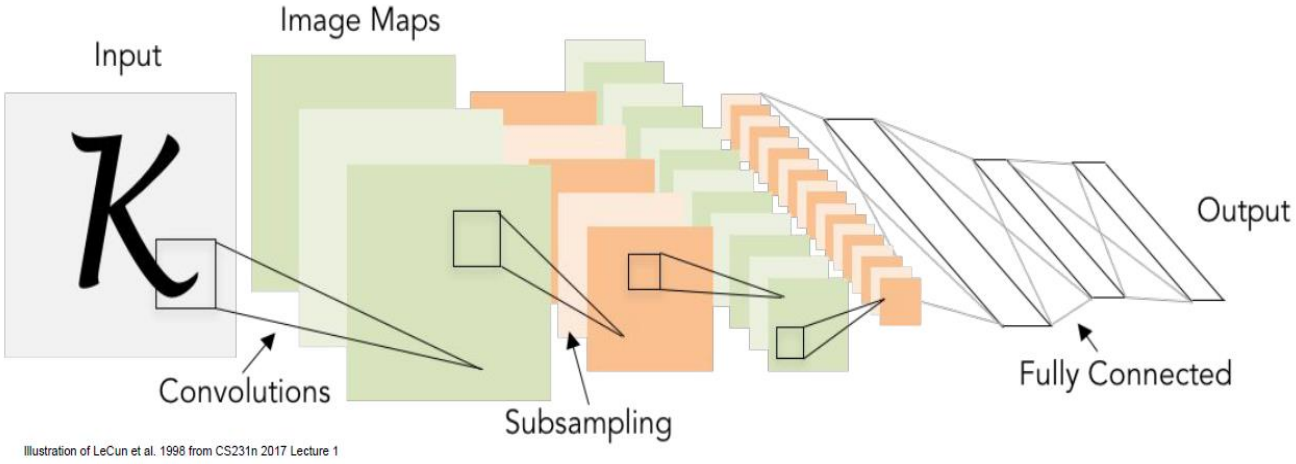
https://keras.io/api/layers/core_layers/

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 영상 인식과 처리에 사용되는 딥러닝 기술
- 합성곱 연산을 신경망에 적용한 영상 데이터에 최적화된 구조
- 신경망의 input이 영상데이터 임



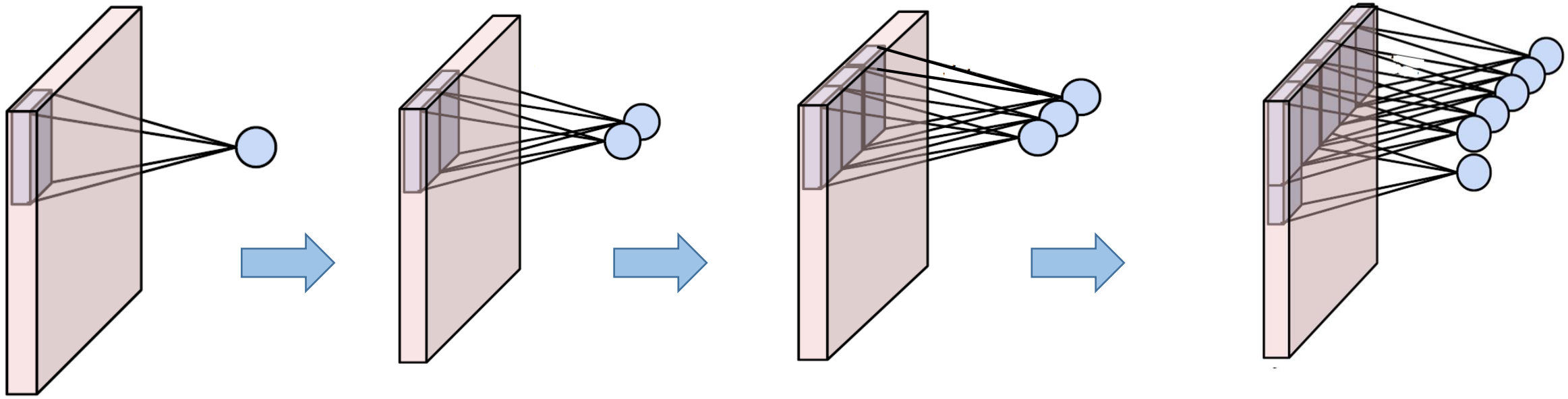
[영상 인식]



[합성곱 신경망 (Stanford cse231)]

[영상 변환]

컨볼루션 신경망 (CNN: Convolutional Neural Network)



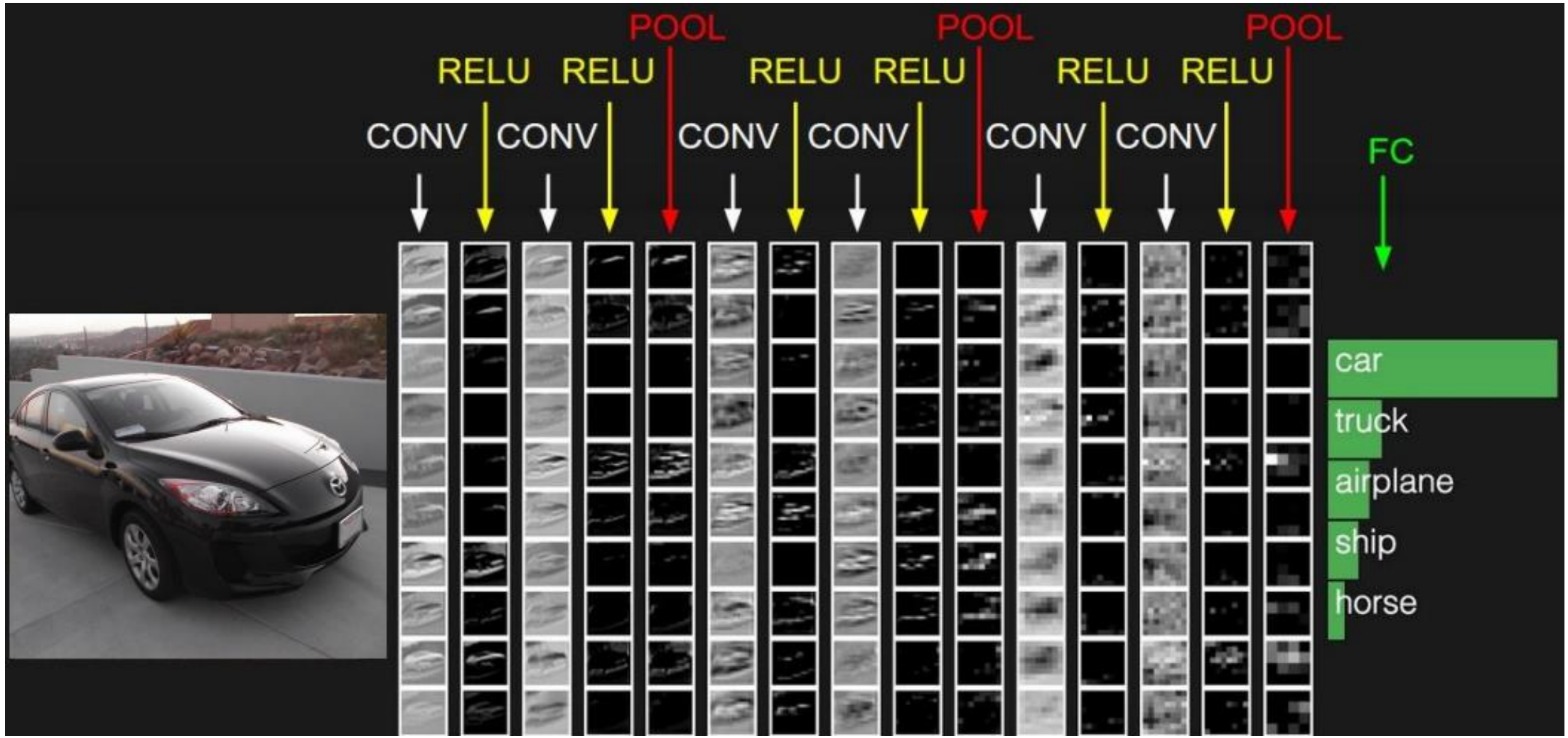
[CNN (Stanford cse231)]

신경망 안에서 합성곱(Convolution) 연산을 수행


THE SOLOMON R GUGGENHEIM MUSEUM

THE SOLOMON R GUGGENHEIM MUSEUM

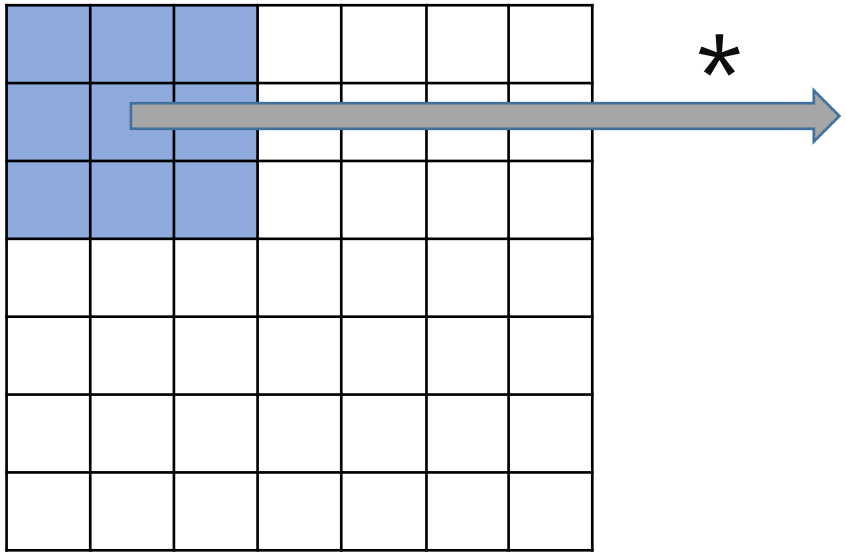
컨볼루션 신경망 (CNN: Convolutional Neural Network)



CONV: Convolution Layer
POOL : Pooling Layer

컨볼루션 (Convolution)

7x7 input image



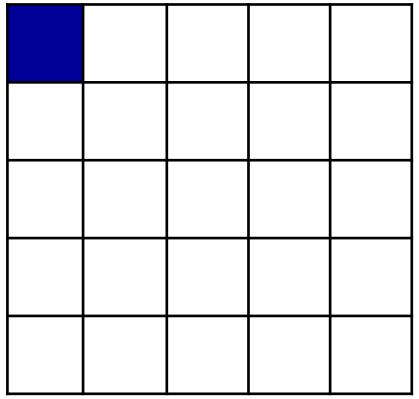
*

3x3 커널

w00	w01	w02
w10	w11	w21
w20	w21	w22

-1.2	2.0	3.1
0.1	1.5	-1.5
1.3	1.6	-0.7

output image

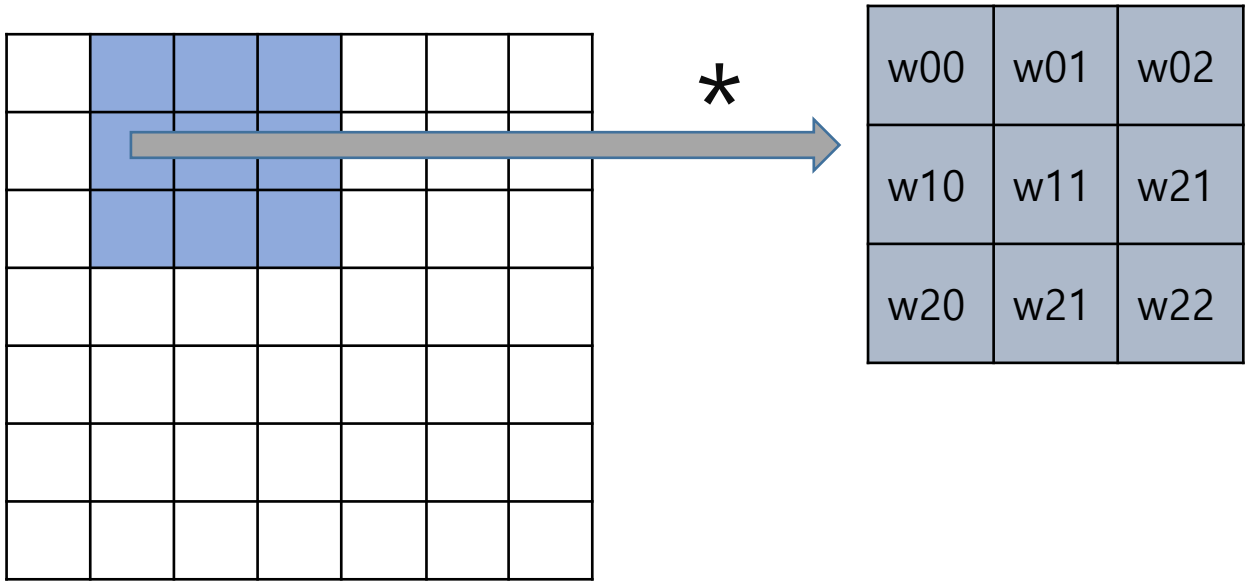


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

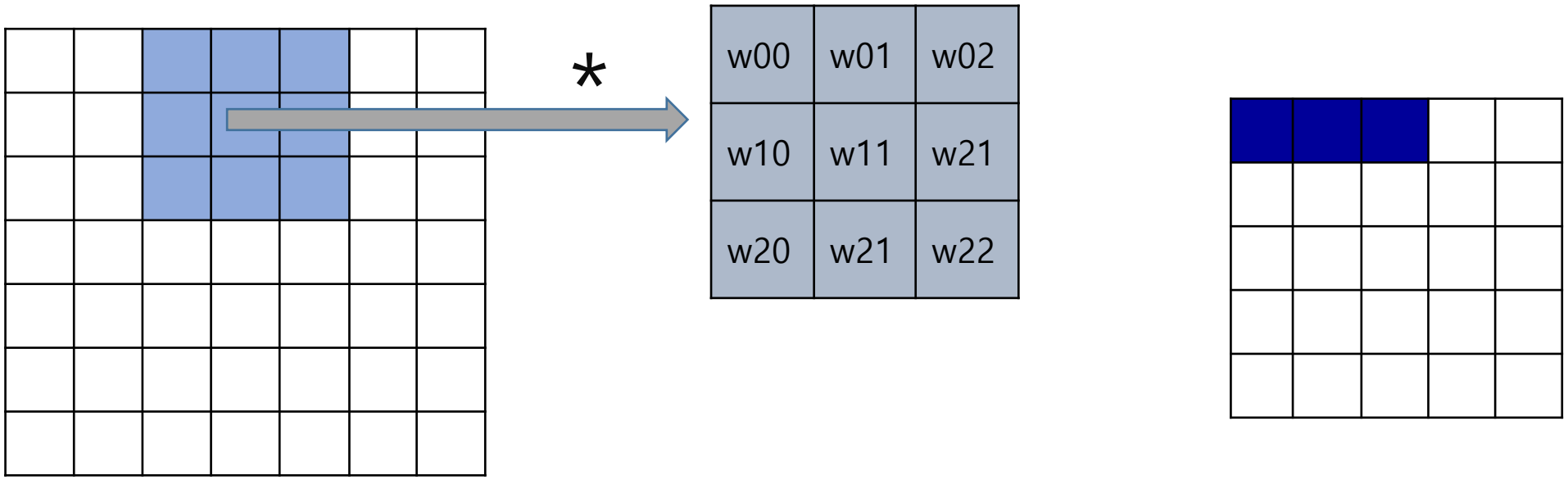


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

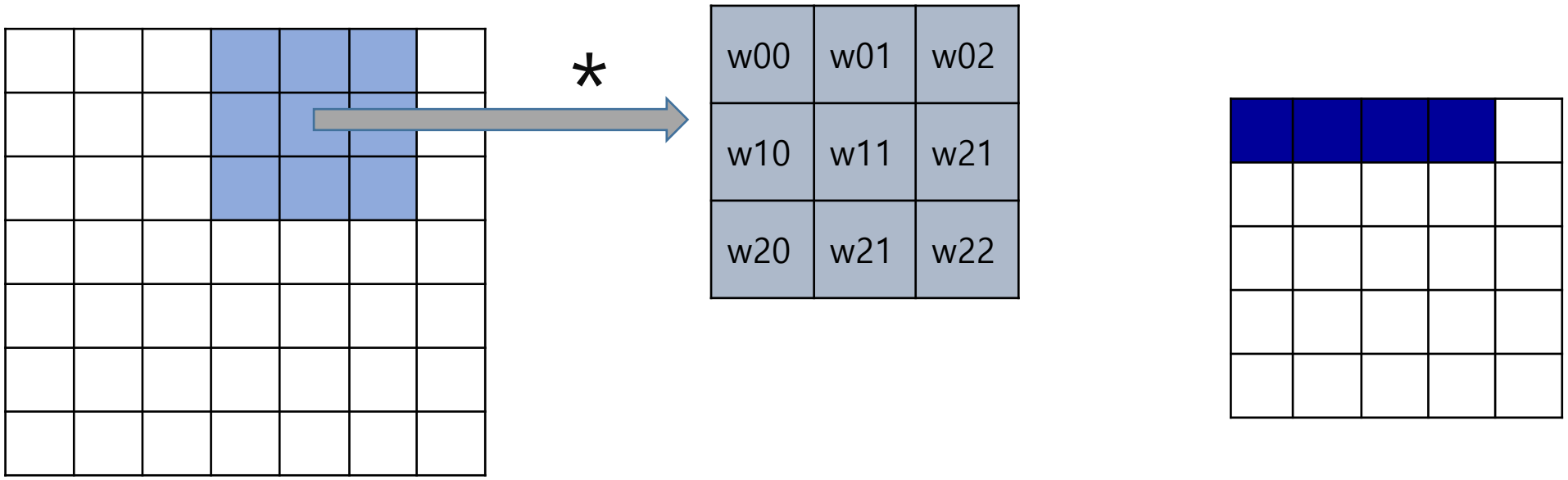


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

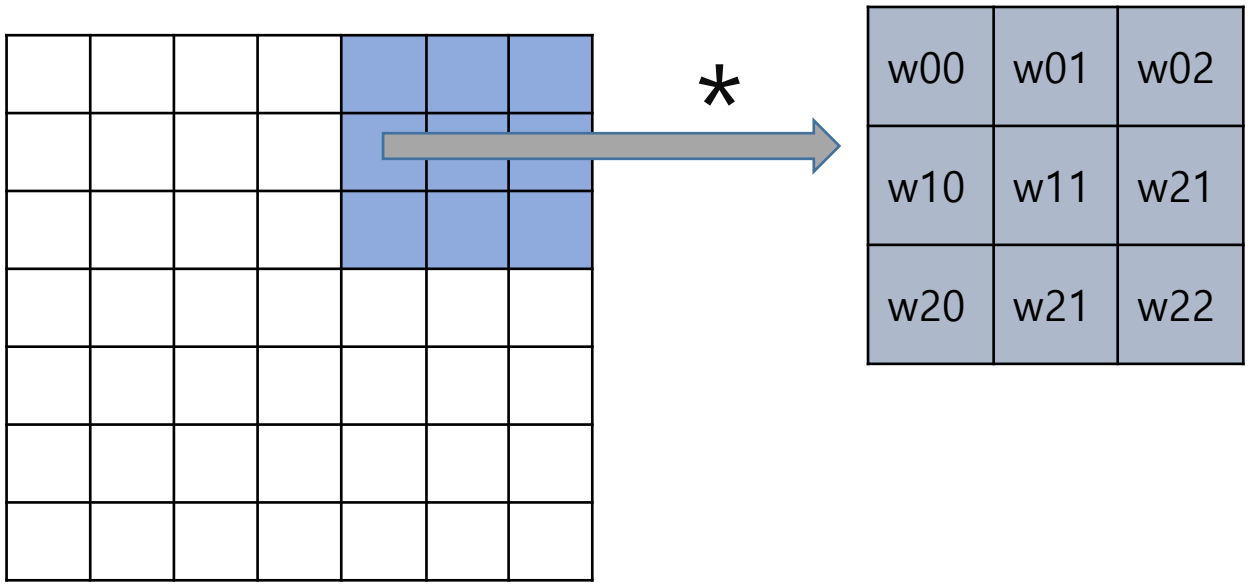


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

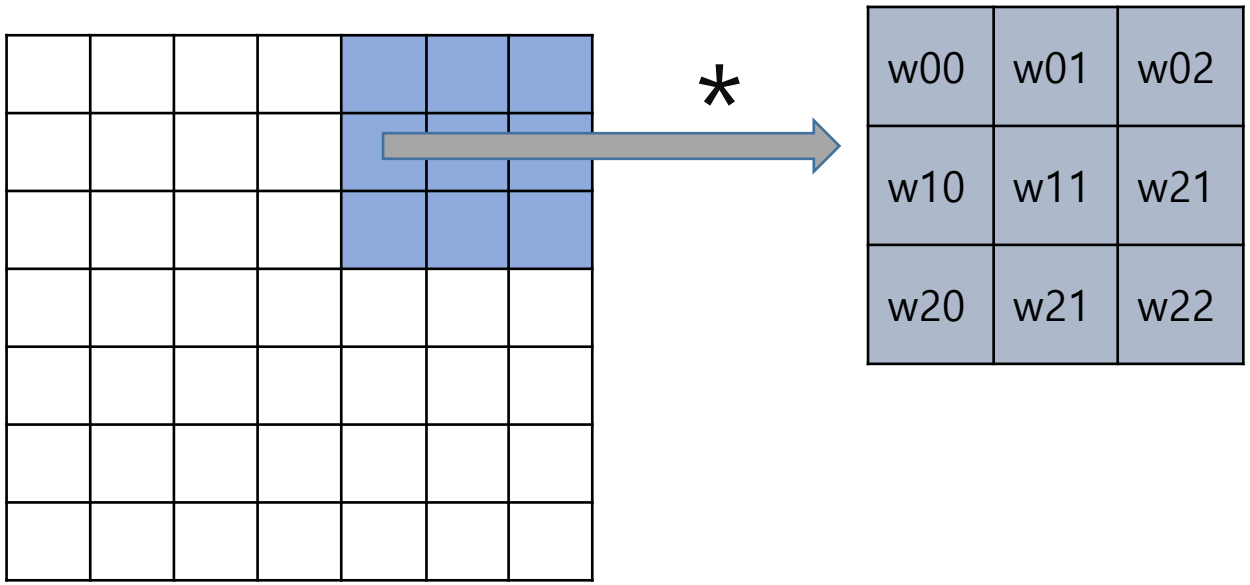


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

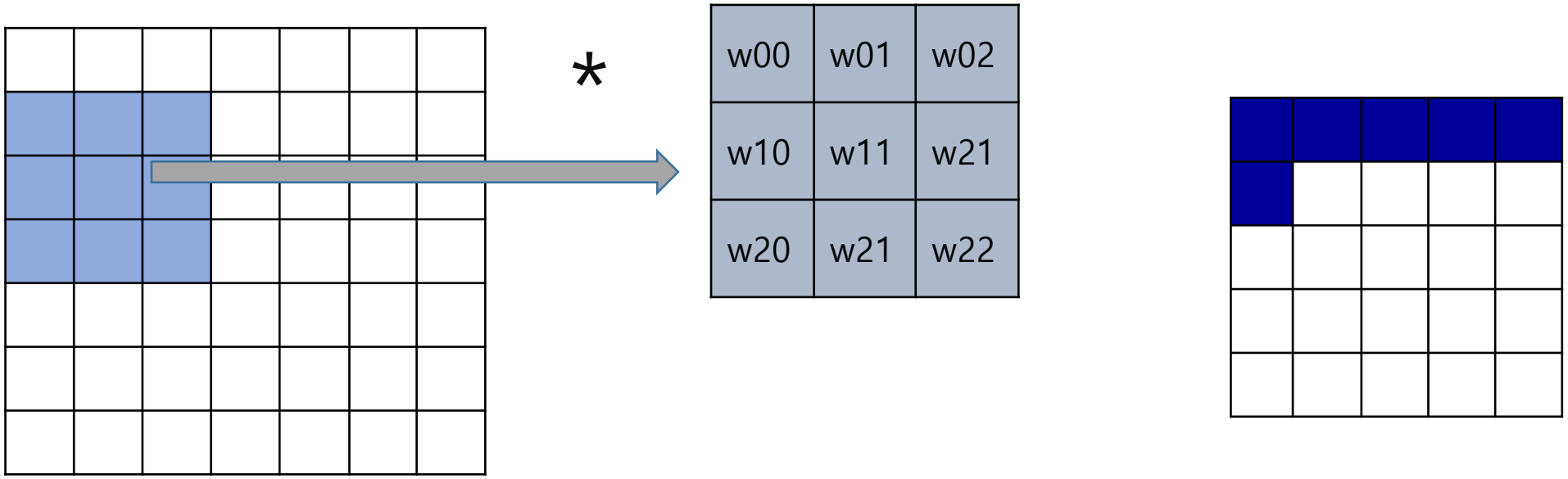


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

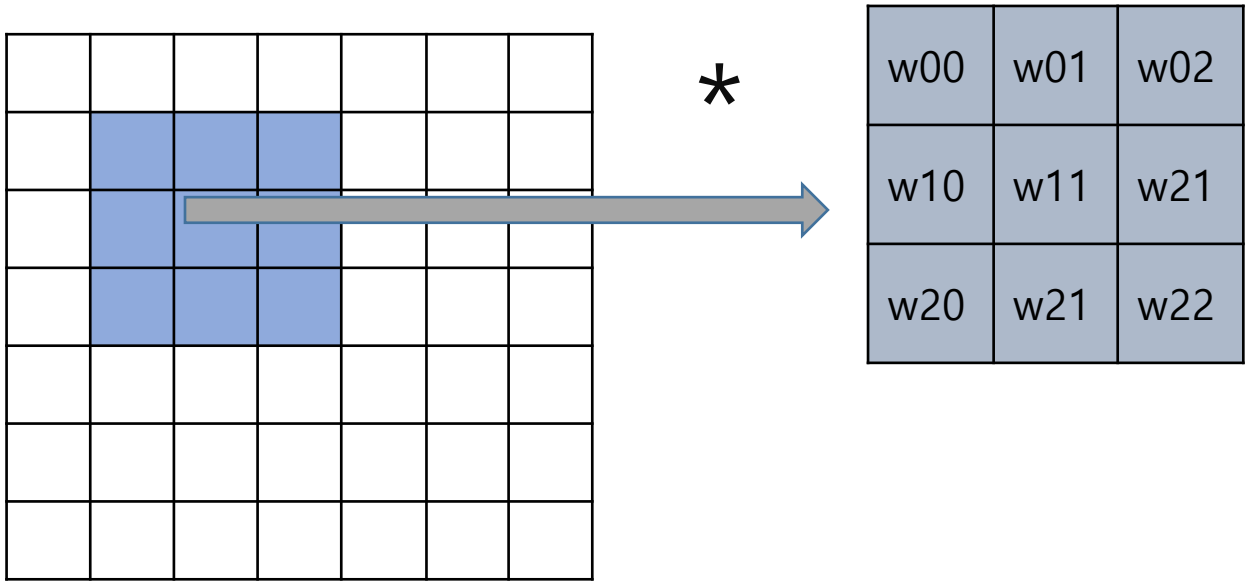


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image



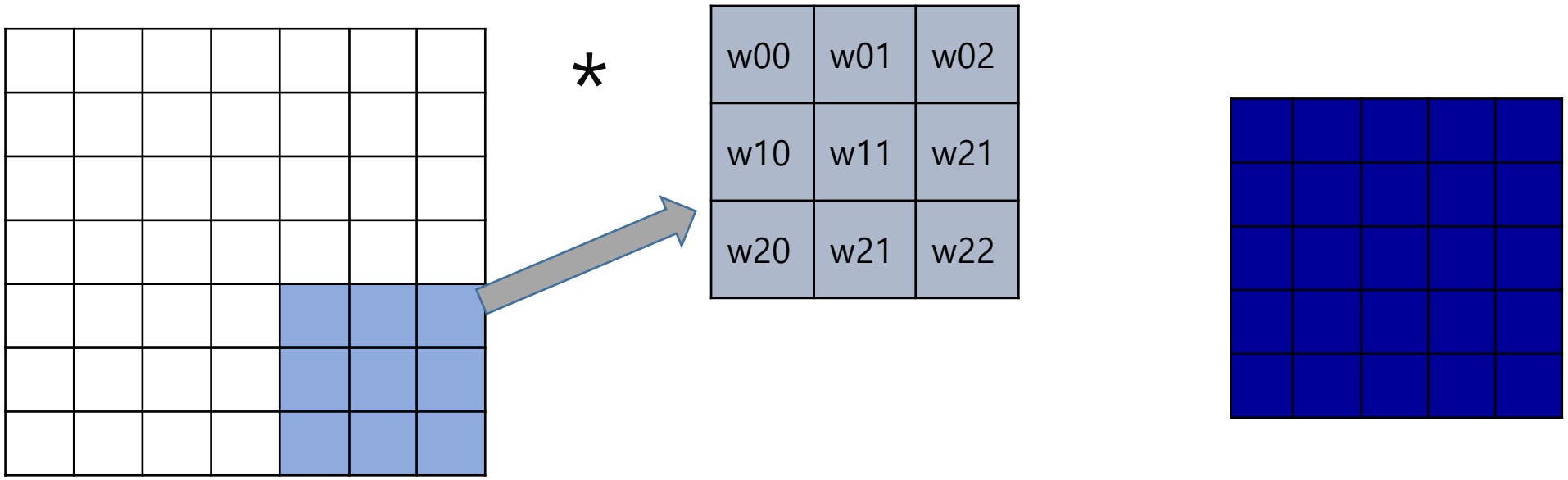
컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

5X5 image



출력 사이즈= (입력 사이즈 – 커널 사이즈)+1=(7-3)+1=5

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 컨벌루션 (Convolution)
  - **컨벌루션은 주변 화소값들에 가중치를 곱해서 더한 후에 이것을 새로운 화소값으로 하는 연산이다.**

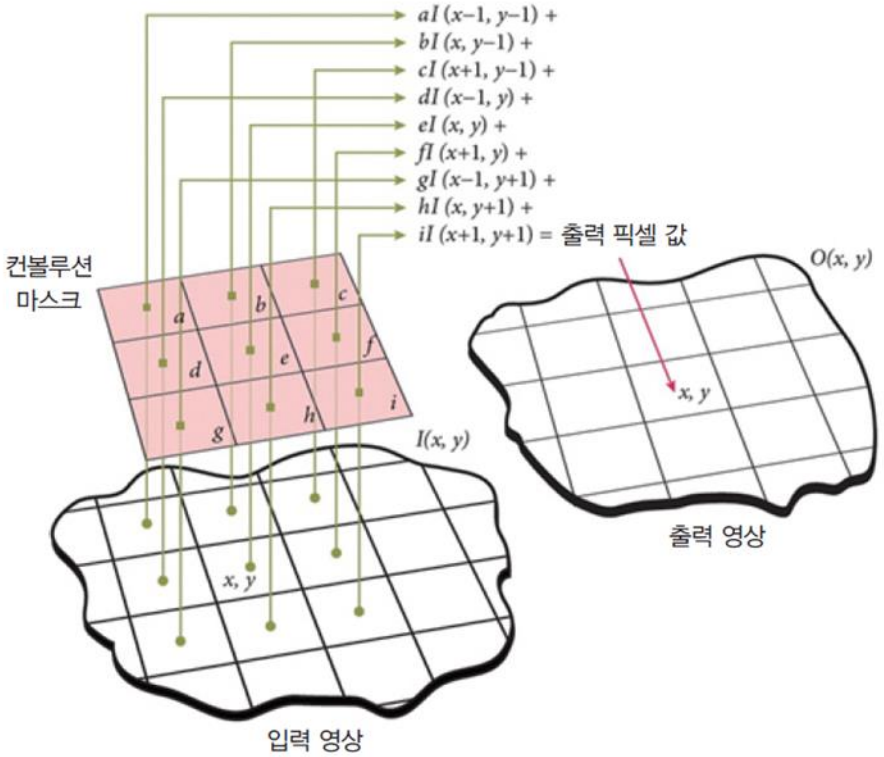


그림 9-6 영상 처리에서 컨벌루션 연산

컨볼루션 신경망 (CNN: Convolutional Neural Network)

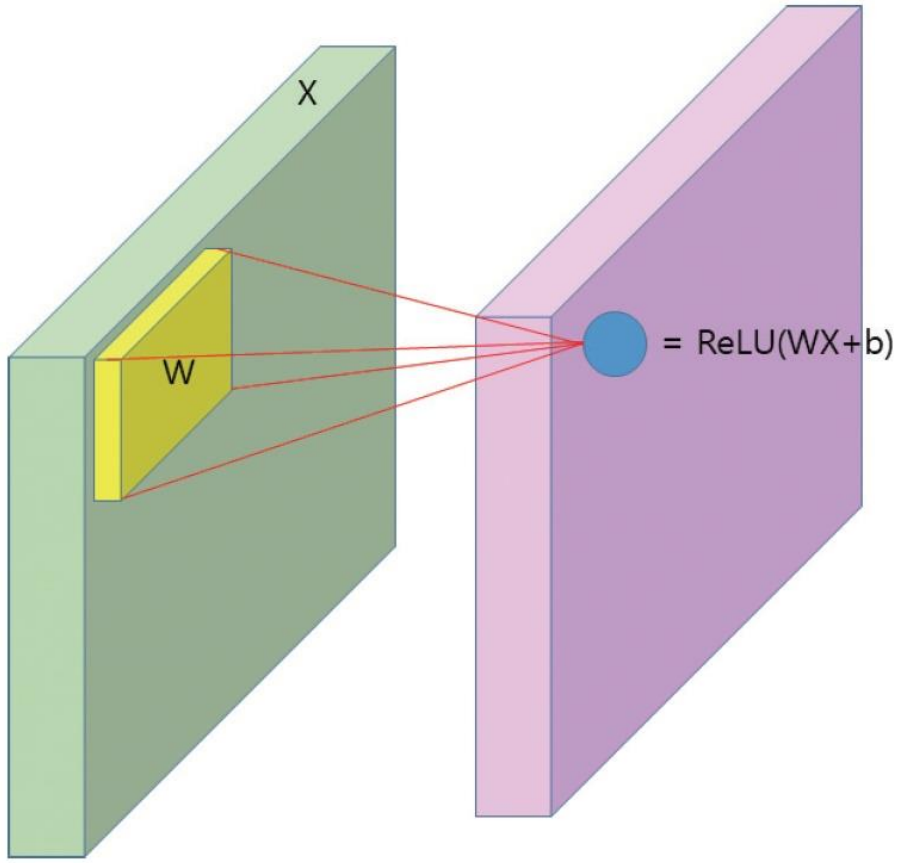


그림 15–15 신경망에서의 컨볼루션 연산

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 컨벌루션를 수행한 결과는 특징맵(feature map)이라고 불리는 데, 그림 9-9가 그 이유를 보여준다.

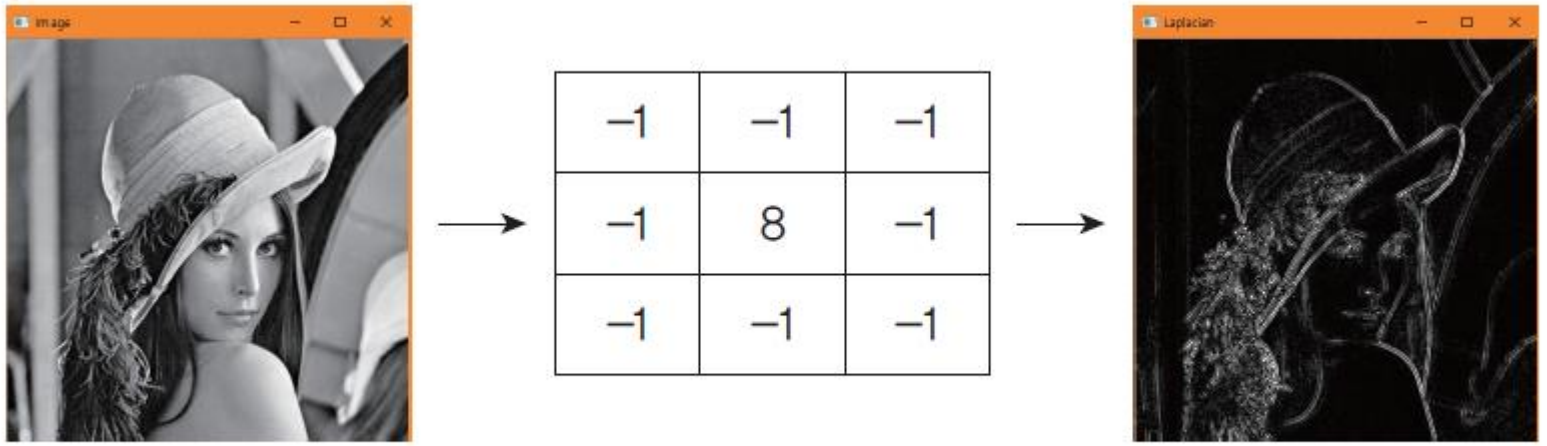


그림 9-9 영상 처리에서의 컨벌루션 연산

컨볼루션 신경망 (CNN: Convolutional Neural Network)

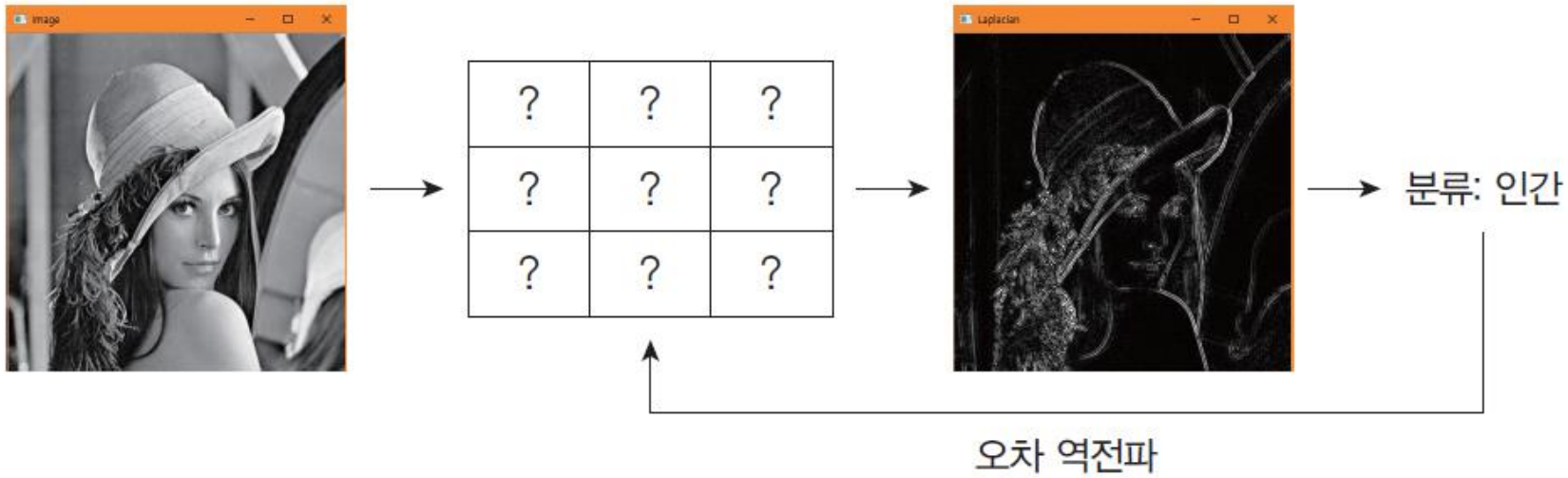


그림 9-10 컨벌루션 신경망에서는 커널의 가중치들이 학습된다.

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 컨벌루션 신경망에서도 커널이 입력층의 각 화소를 중심으로 덮여 씌워진다. 앞 레이어의 값 X는 각커널 W와 곱해져서 더해져서 WX+b가 된다.
- 이 계산값은 ReIU()와 같은 활성화 함수를 통과해서, 다음 레이어의 동일한 위치에 ReLU(WX+b)로 저장된다.

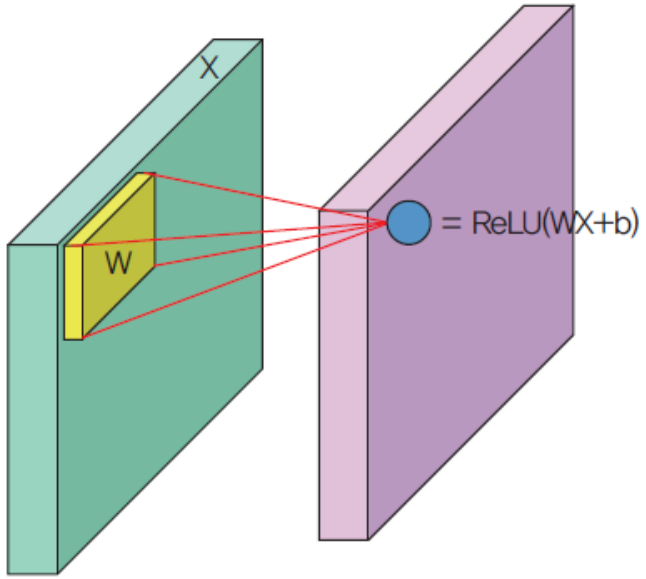
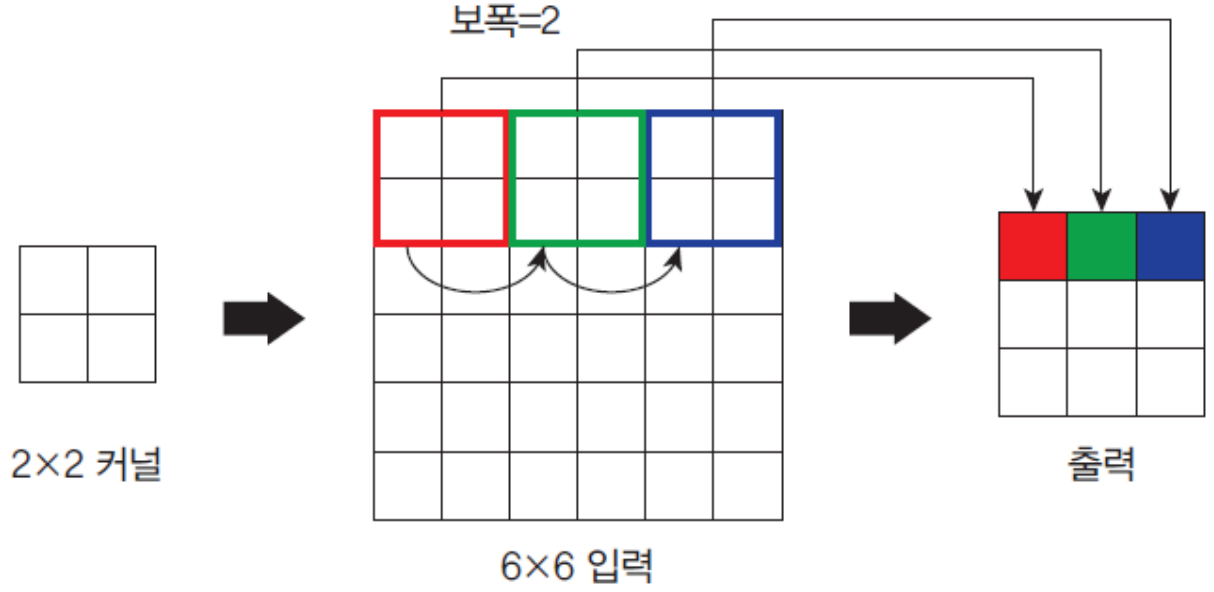


그림 9-11 신경망에서의 컨벌루션 연산

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 보폭(stride) 은 커널을 적용하는 거리이다. 보폭이 1이면 커널을 한 번에 1픽셀씩 이동하면서 커널을 적용하는 것이다.
- 보폭이 2라는 것은 하나씩 건너뛰면서 픽셀에 커널을 적용한다



컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 패딩(padding)은 이미지의 가장자리를 처리하기 위한 기법이

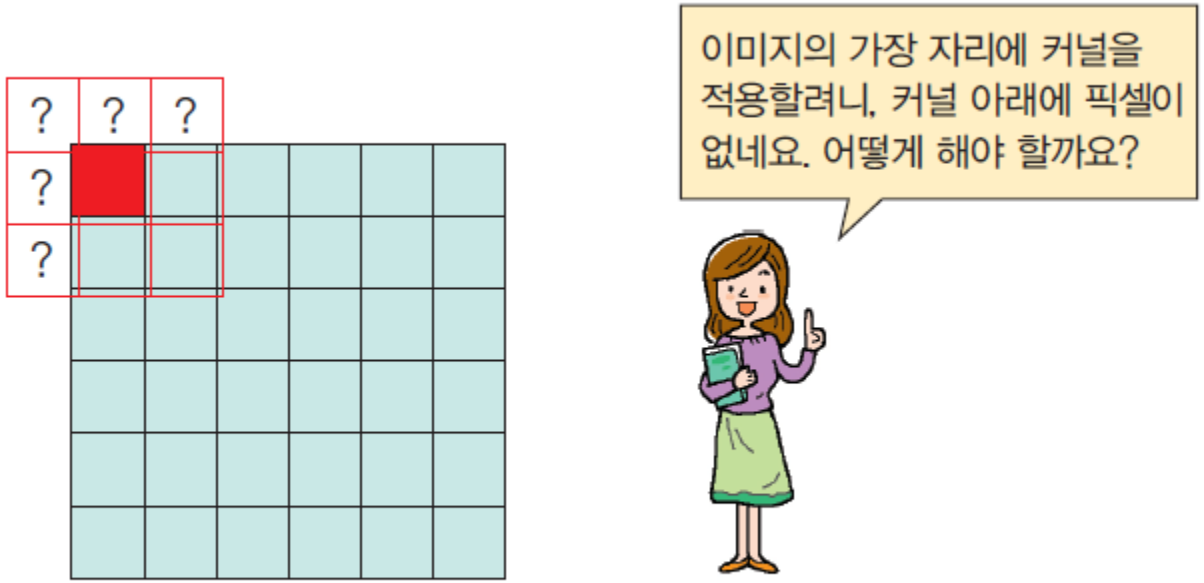
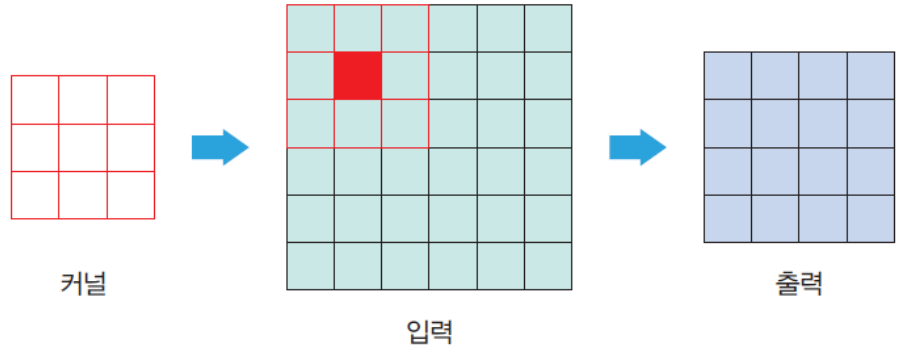


그림 9-13 패딩이 필요한 이유

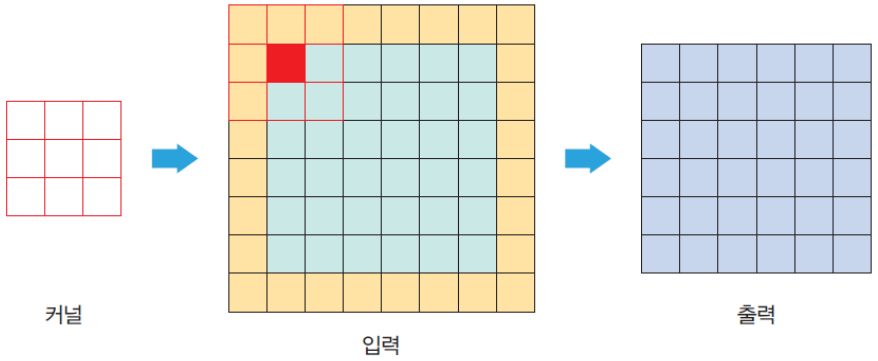
컨볼루션 신경망 (CNN: Convolutional Neural Network)

- **Valid:** 커널을 입력 이미지 안에서만 움직인다.



tf.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), activation=None, input_shape, padding='valid')

- **Same :**입력 이미지의 주변을 특정값(예를 들면 0, 또는 이웃 픽셀값)으로 채우는 것



tf.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), activation=None, input_shape, padding='same')

컨볼루션 신경망 (CNN: Convolutional Neural Network)

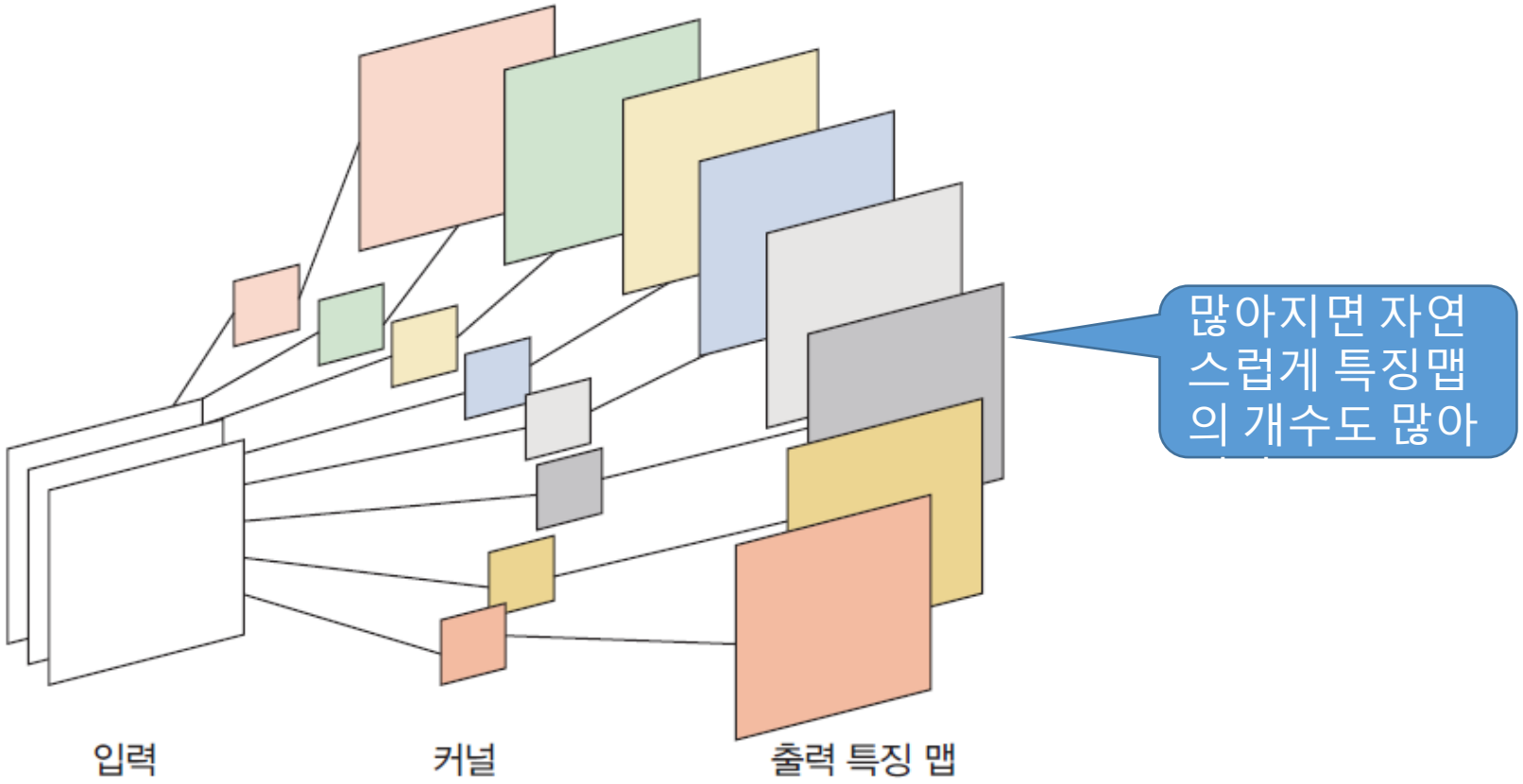
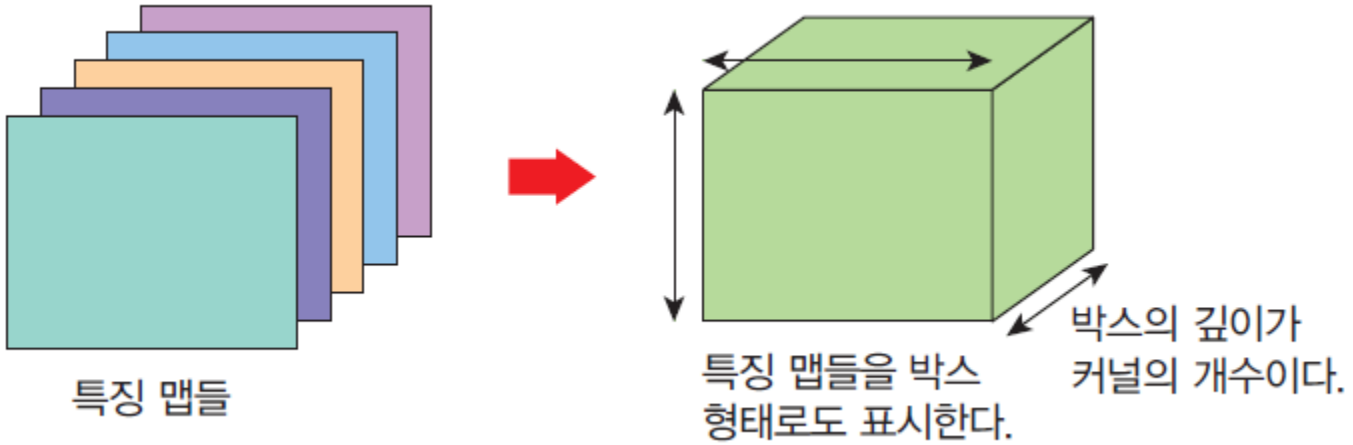


그림 9-14 필터가 여러 개일 때의 컨벌루션 레이어

컨볼루션 신경망 (CNN: Convolutional Neural Network)



컨볼루션 신경망 (CNN: Convolutional Neural Network)

- tf.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), activation=None, input_shape, padding='valid')

  - filters: 필터의 개수이다.
  - kernel_size: 필터의 크기이다.
  - strides: 보폭이다.
  - activation: 유닛의 활성화 함수이다.
  - input_shape: 입력 배열의 형상
  - padding: 패딩 방법을 선택한다. 디폴트는 "valid"이다.

컨볼루션 신경망 (CNN: Convolutional Neural Network)


```
shape = (4, 28, 28, 3)
x = tf.random.normal(shape)
y = tf.keras.layers.Conv2D(2, 3, activation='relu', input_shape=shape[1:])(x)
print(y.shape)
```

```
(4, 26, 26, 2)
```

풀링 (Pooling)

- 풀링(Pooling)이란 서브 샘플링이라고도 하는 것으로 입력 데이터의 크기를 줄이는 것이다.

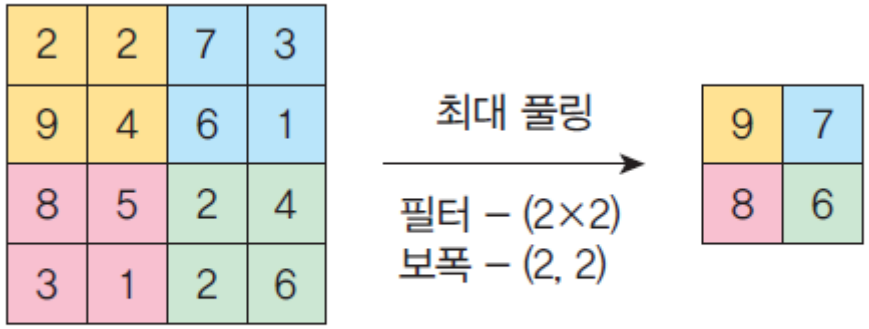


그림 9–17 폴링 레이어

Max Pooling (맥스 풀링)

- 컨벌루션처럼 윈도우를 움직여서 윈도우 안에 있는 숫자 중에서 가장 큰 값만 출력하는 연산이다.

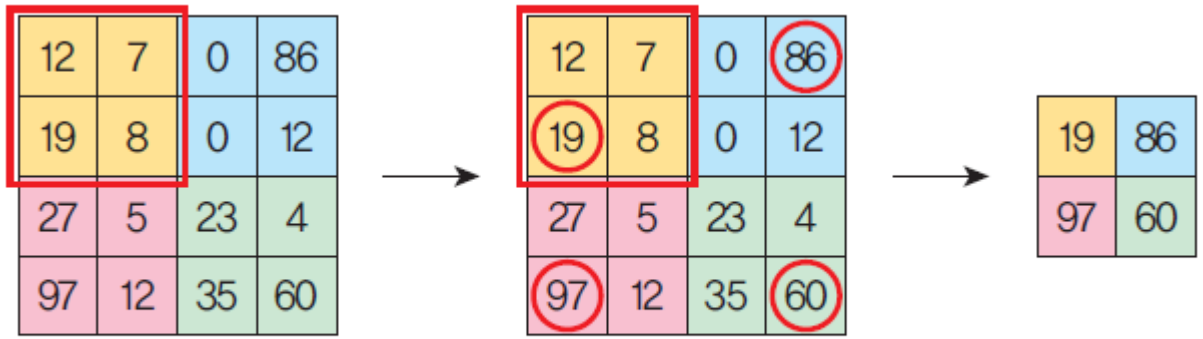


그림 9-18 폴링 연산

풀링의 종류

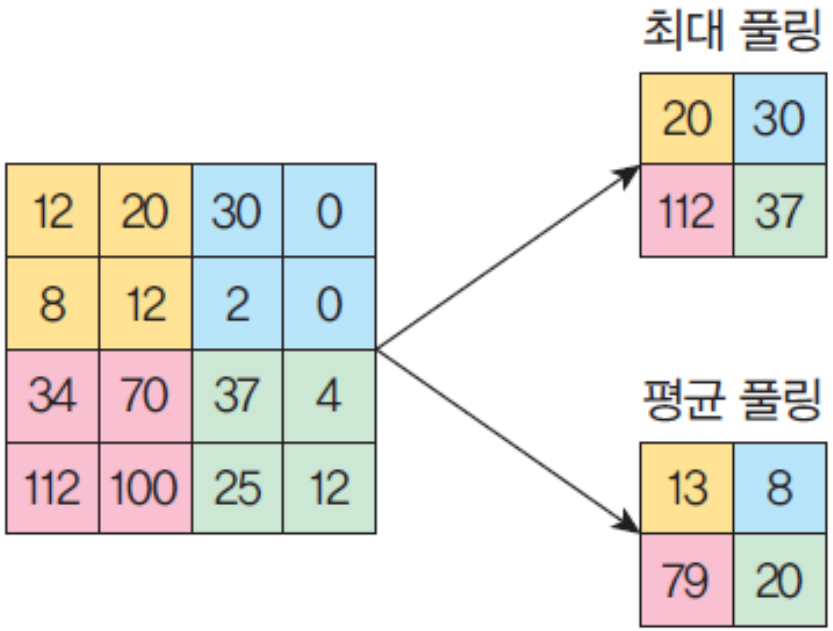
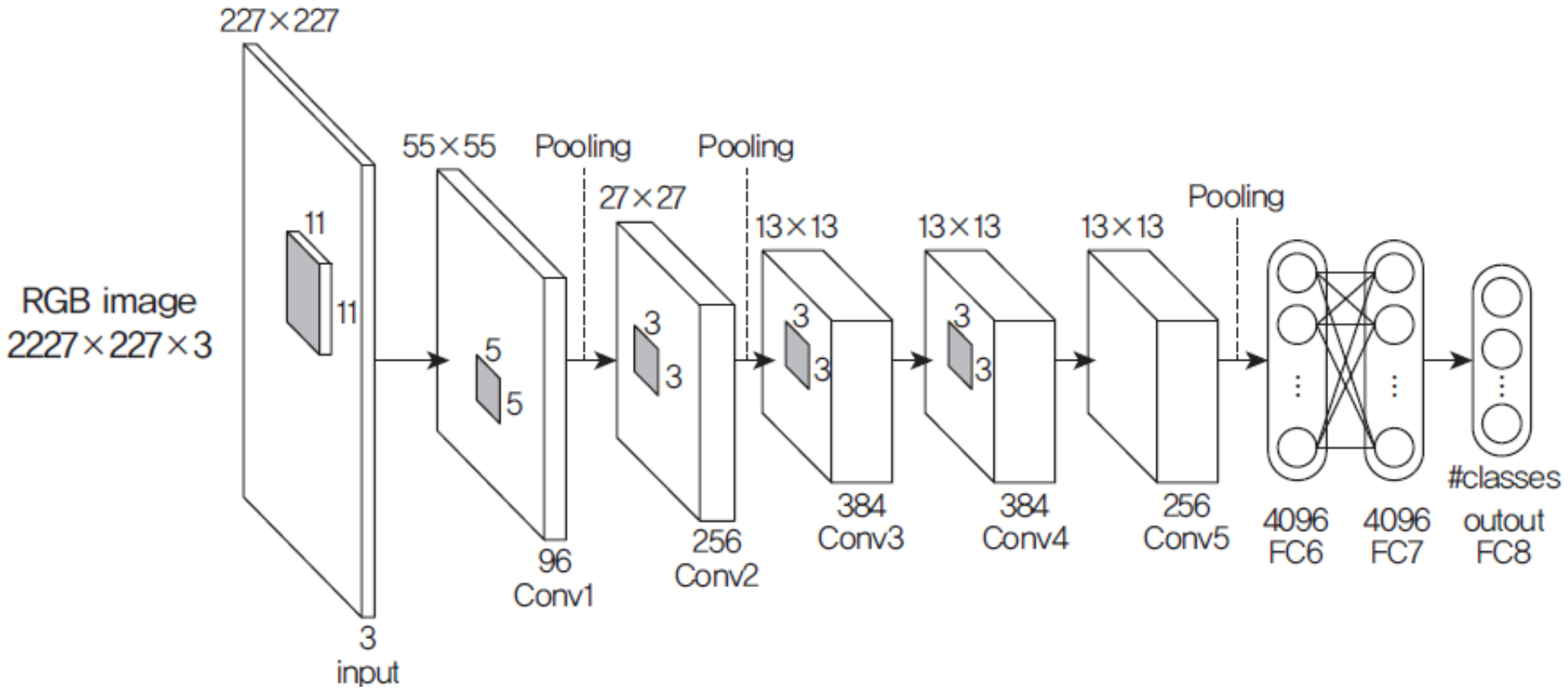


그림 9-19 폴링의 종류

컨볼루션 신경망 (CNN: Convolutional Neural Network)



컨볼루션 신경망 (CNN: Convolutional Neural Network)

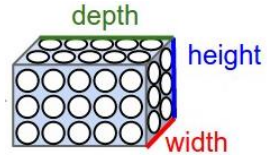
커널 (kernel)

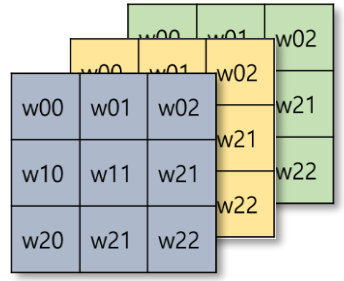
- 합성 곱 (Convolution)과정에서 사용되는 행렬

w00	w01	w02
w10	w11	w21
w20	w21	w22

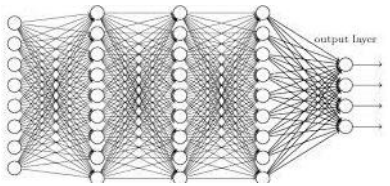
채널 (channel)

- 커널의 수
- 채널의 수가 output의 depth가 됨

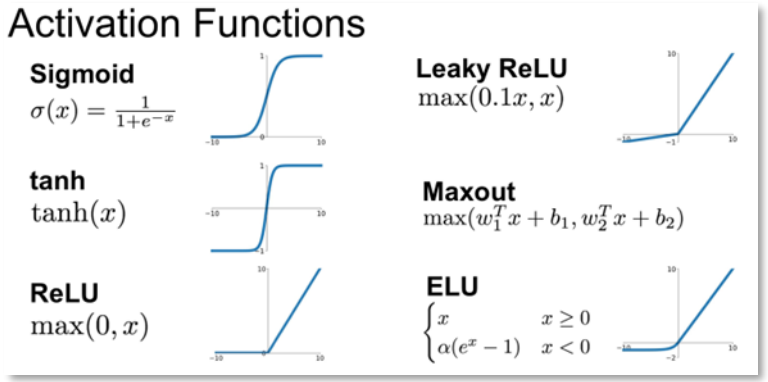


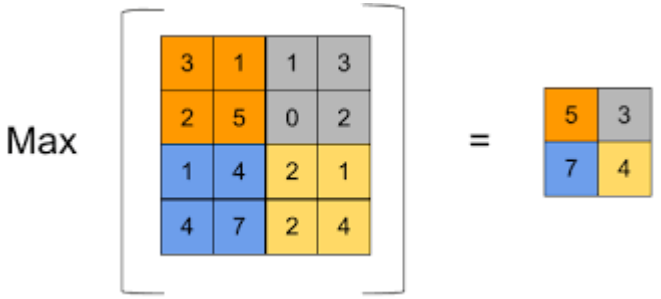


레이어 (layer)



- 신경망 안에서 연산을 수행하는 각 단계
- 레이어의 종류
  - . **Convolution Layer** : 합성곱 연산을 수행 하는 layer
  - . **Activation Layer** : 신경망 학습의 효율을 높이기 위하여 데이터 분포를 변형 해주는 layer
  - . **Pooling Layer** : 정보의 크기를 줄여주는 Layer. Ex) Average Pooling, Max Pooling

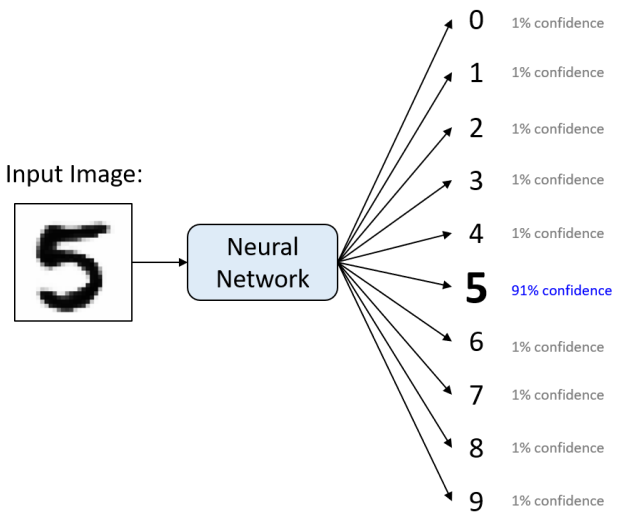




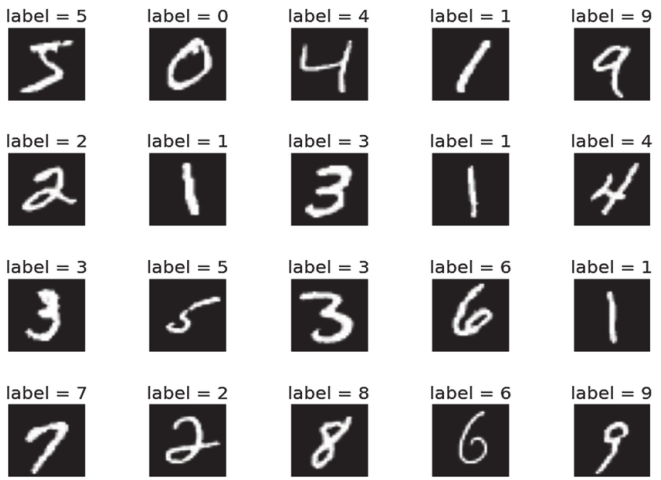
[max pooling]

MNIST 손 글씨 이미지 분류

- **필기체 이미지를 입력해 숫자를 인식 함**
- **MNIST dataset**
  - Image size : 28x28
  - Image와 label (category)가 같이 저장되어 있음.
  - Training image 60000 장, test image 10000 장



[MNIST 손 글씨 이미지 분류]



[MNIST dataset]

MNIST 필기체 숫자 인식


```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# 픽셀 값을 0~1 사이로 정규화한다.
train_images, test_images = train_images / 255.0, test_images / 255.0
```

MNIST 필기체 숫자 인식

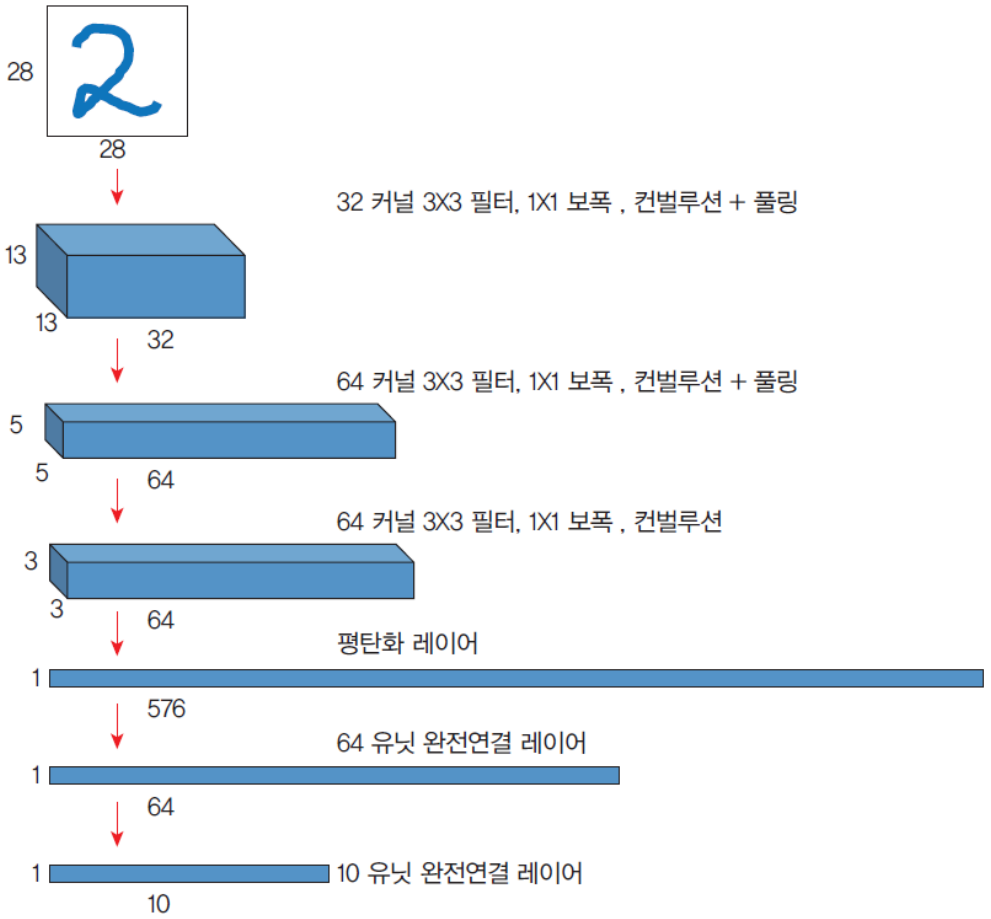

```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

MNIST 필기체 숫자 인식


```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
conv2d_3 (Conv2D) (None, 26, 26, 32) 320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 32) 0
_________________________________________________________________
conv2d_4 (Conv2D) (None, 11, 11, 64) 18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64) 0
_________________________________________________________________
conv2d_5 (Conv2D) (None, 3, 3, 64) 36928
_________________________________________________________________
```

MNIST 필기체 숫자 인식

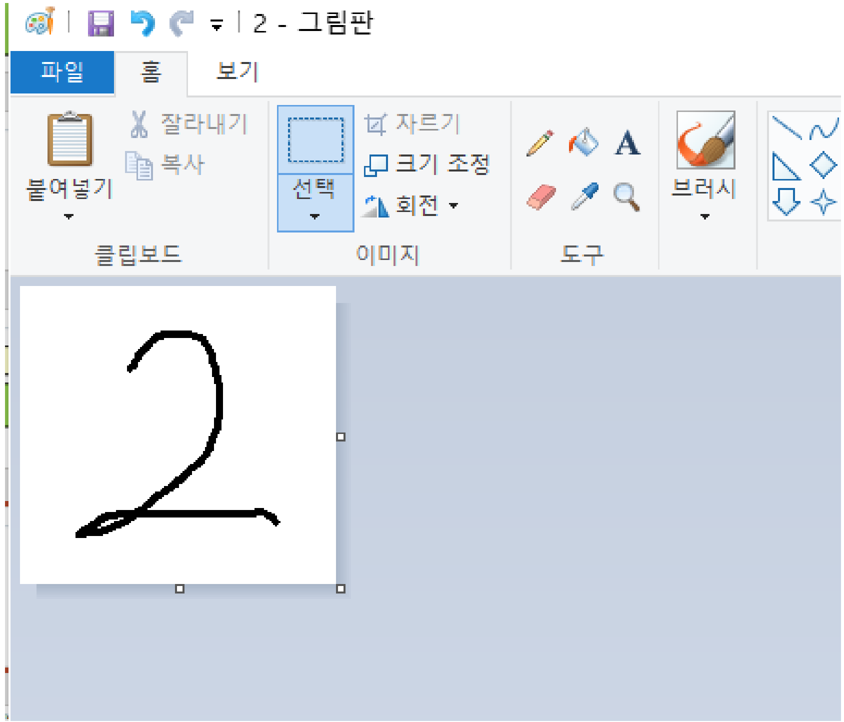


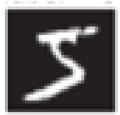
MNIST 필기체 숫자 인식


```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1414 -
accuracy: 0.9560
...
Epoch 5/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0194 -
accuracy: 0.9940
```

MNIST 필기체 숫자 인식



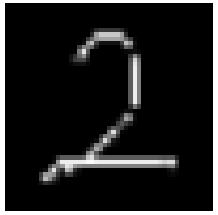


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	14	1	154	253	90	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	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	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	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	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	249	253	249	64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	136	253	253	253	212	135	132	16	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	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	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MNIST 필기체 숫자 인식

(과제1)


```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from google.colab.patches import cv2_imshow

model=load_model('/content/mnist_model1.hdf5')
img=cv2.imread('2.png',cv2.IMREAD_GRAYSCALE)
img=cv2.resize(img,(28,28))
img=img.astype('float32')
cv2_imshow(img)
img=255-img
cv2_imshow(img)
img=img/255.0
img=img[np.newaxis,:,:,np.newaxis]
test_pred=model.predict(img)
print(np.round(test_pred,2))
```



자료실 : mnist_model1.hdf5 이용

MNIST 필기체 숫자 인식

https://transcranial.github.io/keras-js/#/mnist-cnn

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

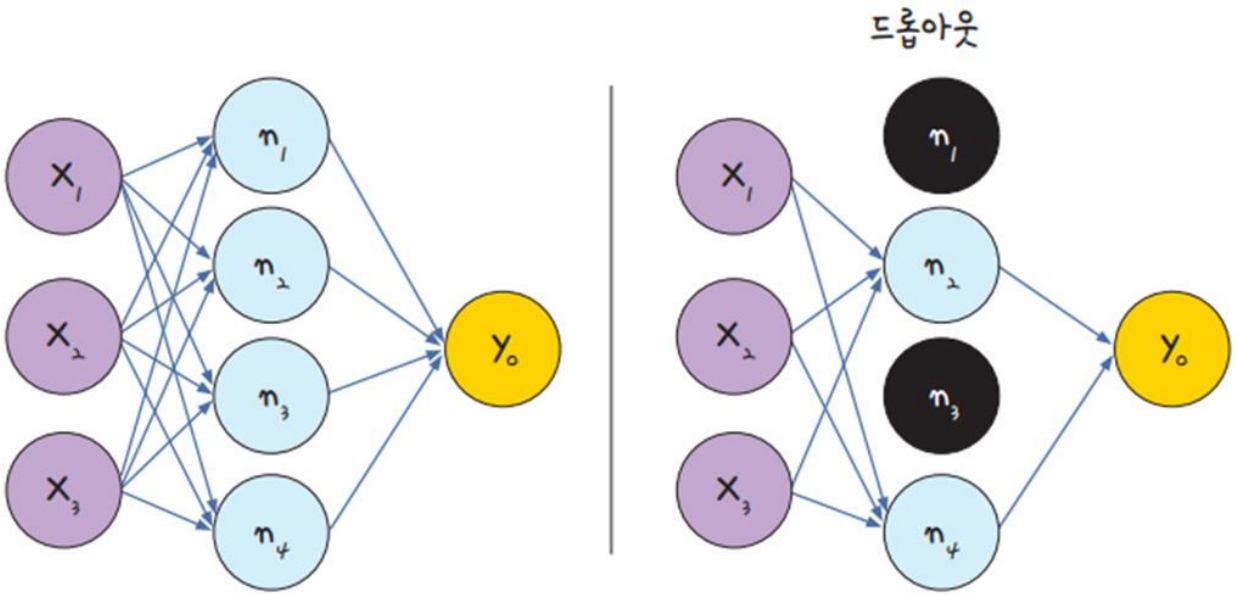
```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

드롭아웃,

- 노드가 많아지거나 층이 많아진다고 해서 학습이 무조건 좋아지는 것이 아니라는 점을 과적합 의미를 공부하며 배웠음
- 딥러닝에서 학습을 진행할 때 가장 중요한 것은 과적합을 얼마나 효과적으로 피해 가는지에 달려 있다고 해도 과언이 아님
- 과적합을 피하기 위한 방법중 하나 => 드롭 아웃 (drop out) 기법
- 드롭아웃은 은닉층에 배치된 노드 중 일부를 임의로 꺼 주는 것



MNIST 손 글씨 이미지 분류- CNN 케라스 구현

- **맥스 풀링, 드롭아웃, 플래튼**
  - 이렇게 랜덤하게 노드를 꺼 주면 학습 데이터에 지나치게 치우쳐서 학습되는 과적합을 방지할 수 있음
  - 케라스는 이러한 드롭아웃을 손쉽게 적용하도록 도와줌
  - 예를 들어 25%의 노드를 끄려면 다음과 같이 코드를 작성하면 됨

```
model.add(Dropout(0.25))
```

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

- **맥스 풀링, 드롭아웃, 플래튼**
  - 이제 이러한 과정을 지나 다시 앞에서 Dense() 함수를 이용해 만들었던 기본 층에 연결해 볼까?
  - 이때 주의할 점은 컨볼루션 층이나 맥스 풀링은 주어진 이미지를 2차원 배열인 채로 다룬다는 것
  - 이를 1차원 배열로 바꾸어 주어야 활성화 함수가 있는 층에서 사용할 수 있음
  - Flatten() 함수를 사용해 2차원 배열을 1차원으로 바꾸어 줌

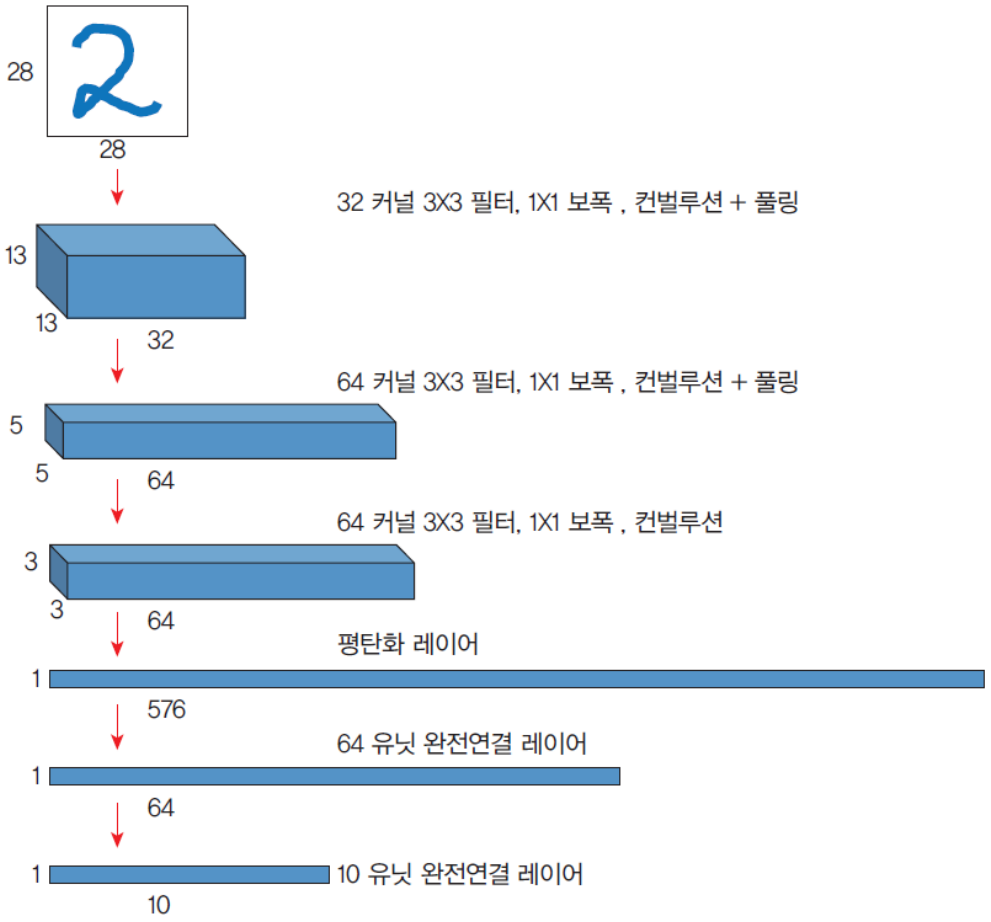
```
model.add(Flatten())
```

CNN output shape

- **맥스 풀링, 드롭아웃, 플래튼**
  - 이제 이러한 과정을 지나 다시 앞에서 Dense() 함수를 이용해 만들었던 기본 층에 연결해 볼까?
  - 이때 주의할 점은 컨볼루션 층이나 맥스 풀링은 주어진 이미지를 2차원 배열인 채로 다룬다는 것
  - 이를 1차원 배열로 바꾸어 주어야 활성화 함수가 있는 층에서 사용할 수 있음
  - Flatten() 함수를 사용해 2차원 배열을 1차원으로 바꾸어 줌

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

Convolution Neural Network (CNN)



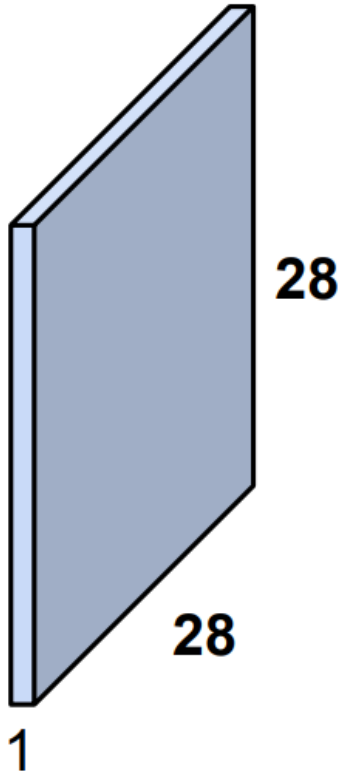

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=========Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
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
_________________________________________________________________
dropout (Dropout)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

Convolution Neural Network (CNN)

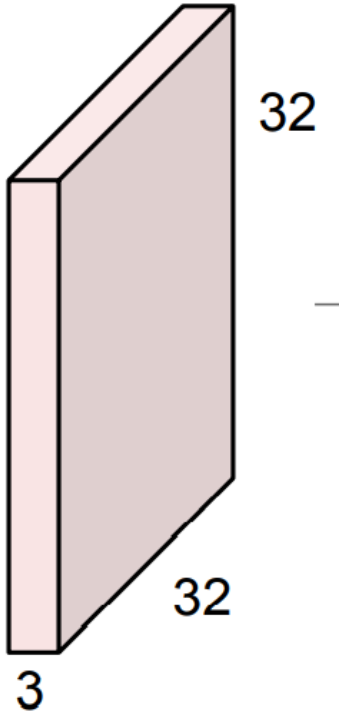
Input size = (28,28,1)

Input size = (32,32,3)

input depth



- **Output Shape**



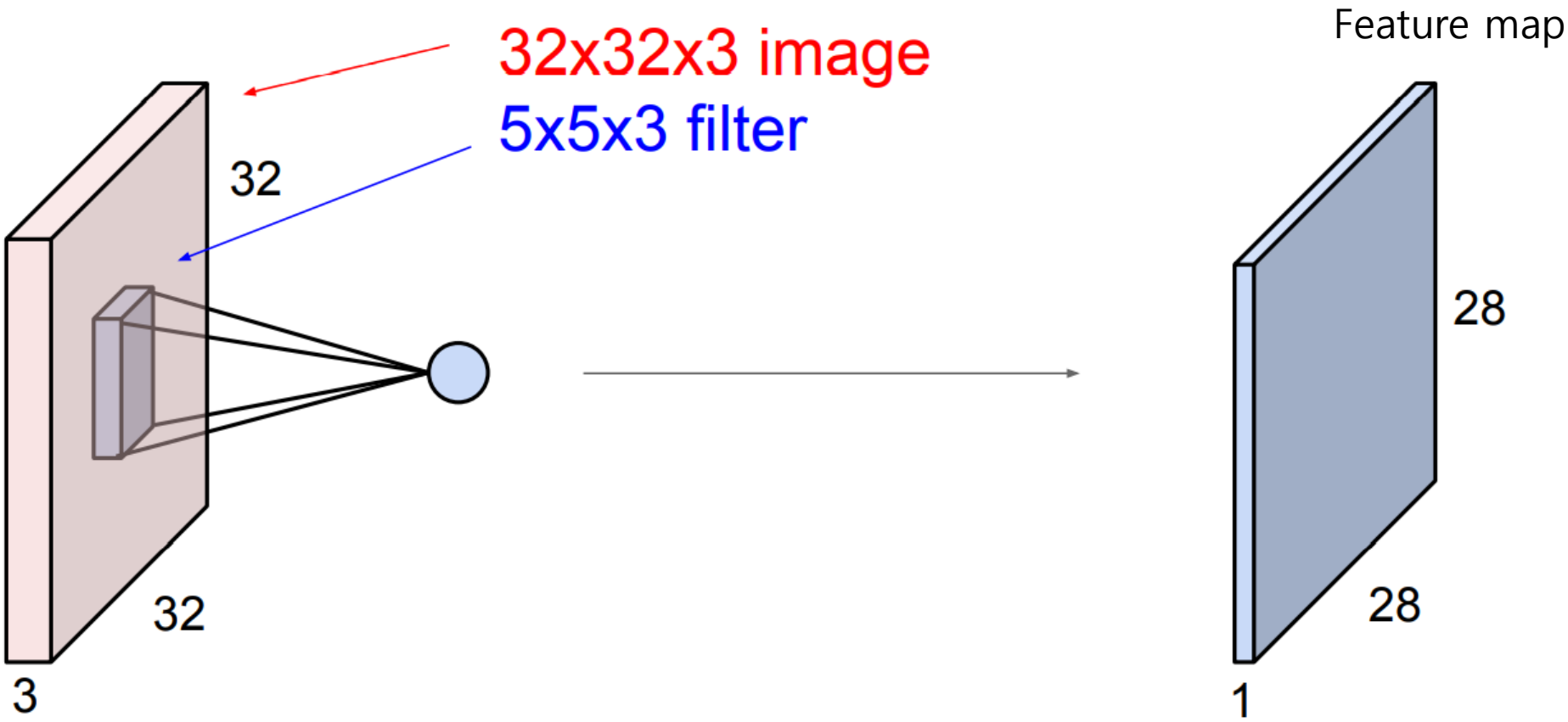
Convolution Neural Network (CNN)

- **커널(kernel)=필터(filter)**
- **필터의 수 = 채널 (channel)**

  - tf.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), activation=None, input_shape, padding='valid')
    - filters: 필터의 개수이다.

Convolution Neural Network (CNN)

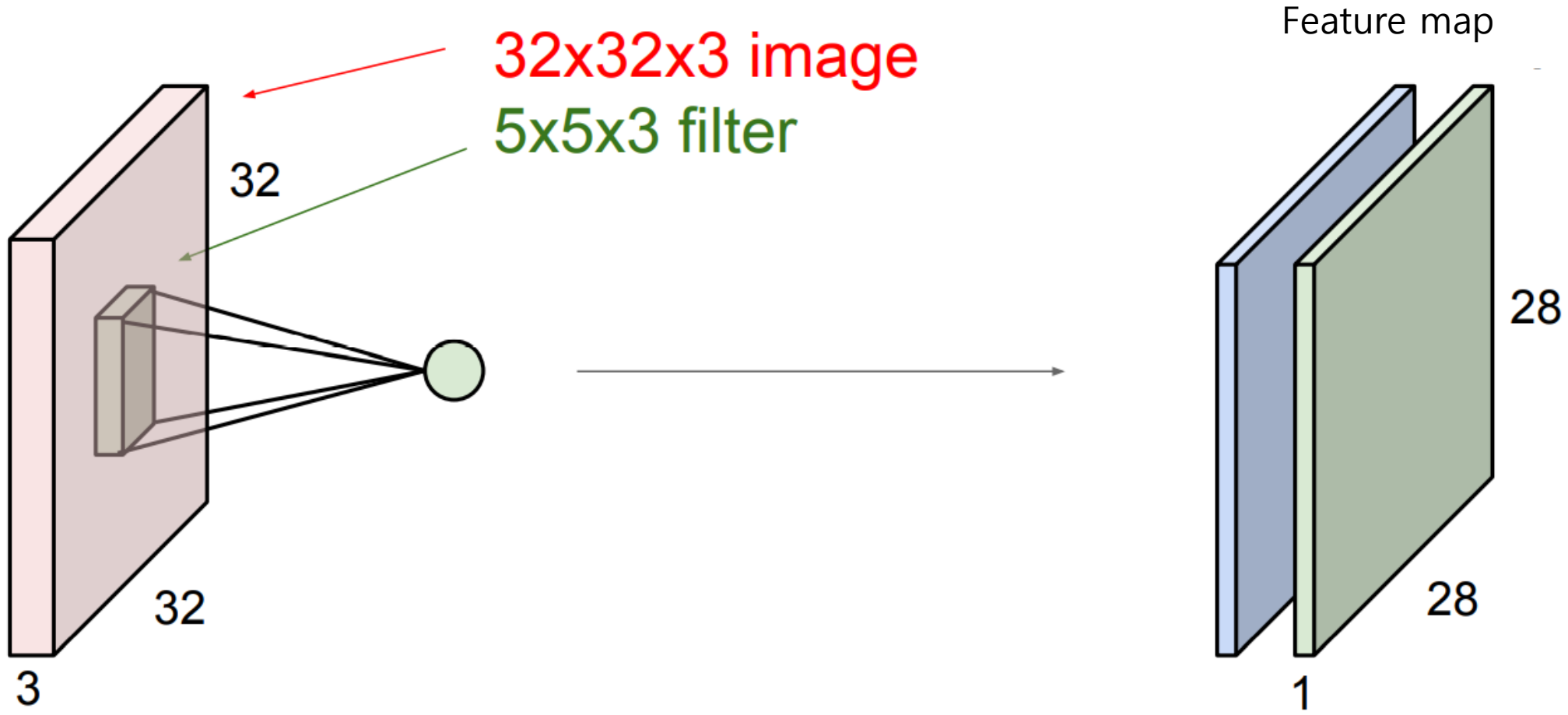
- **Output Shape**



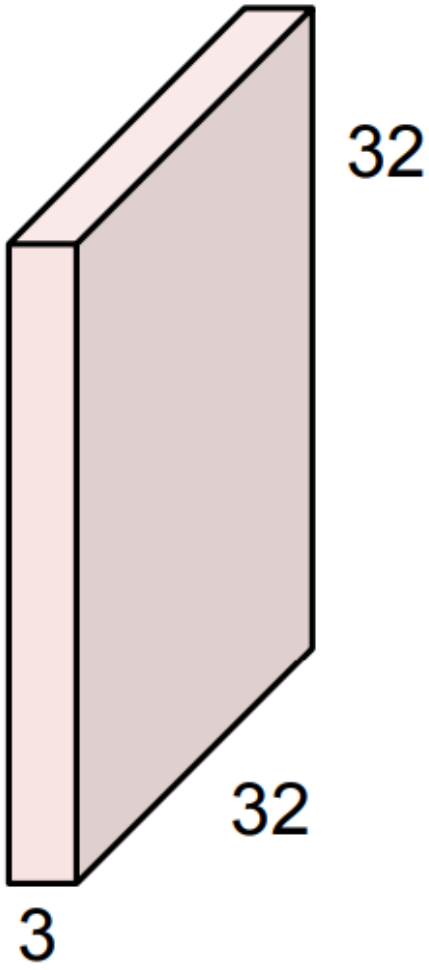
출력 사이즈= (입력 사이즈 – 커널 사이즈)+1=32-5+1=28

Convolution Neural Network (CNN)

- **Output Shape**

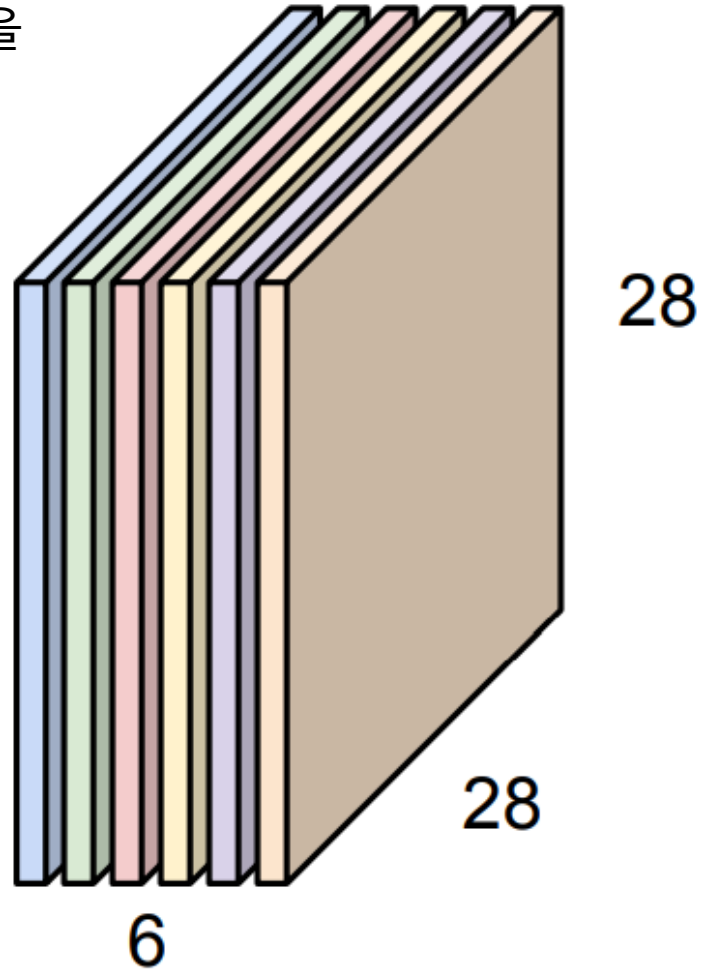


Convolution Neural Network (CNN)



6개의 filter가 있으면 6개의 feature map을 가지게 됨

Feature map



채널 :6

(32,32,3) => 6개의 5x5 filter => (28,28,6)

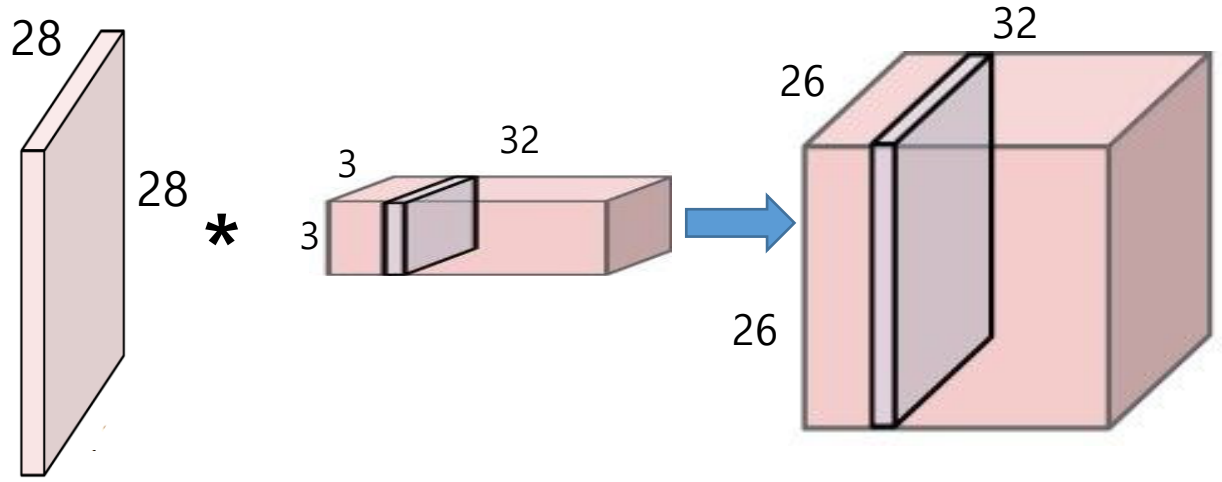
MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Filter size Filter 개수=32

Input=28x28x1, kernel=3x3, channel=32
=> output size=26x26x32

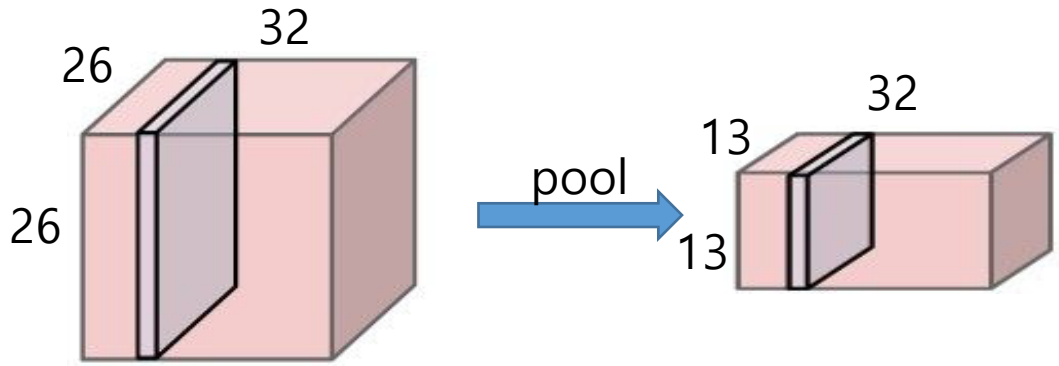


MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Input=26x26x32, MaxPool =(2,2)
=> output size=13x13x32



MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Input=13x13x32, kernel=3x3, channel=64
=> output size=11x11x64

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Input=11x11x64, MaxPool =(2,2)
=> output size=5x5x64

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Input=5x5x64, => output size=1600

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Input=1600, => output size=1600

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Number of parameters
=input size*number of class +number of class
=>1600*10+10=16010

Input=1600, => output size=10

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)



model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

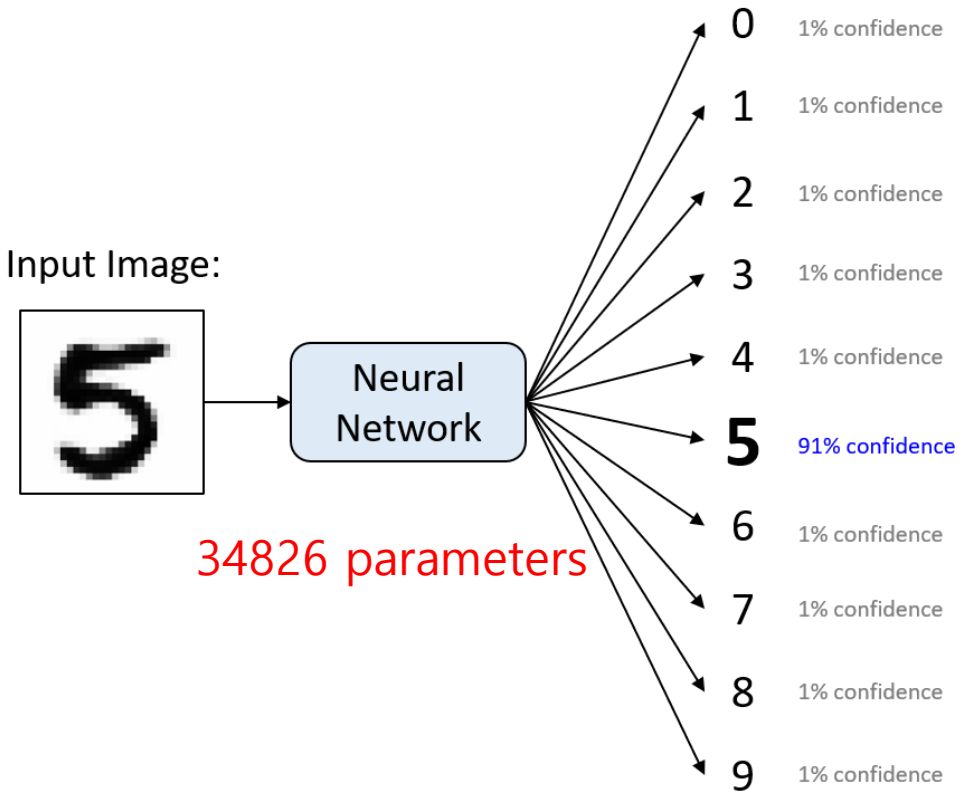
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=========Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
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
_________________________________________________________________
dropout (Dropout)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

```
batch_size = 128
epochs = 15

model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
validation_split=0.1)
```

Compile : Model을 기계가 이해할 수 번역

Fit : 학습시킴 (training)




```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

학습된 model의 성능 테스트

Convolution Neural Network (CNN)

- **모델 디자인 (mnist_paratest.ipynb)** (과제2)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 16)        160

max_pooling2d_2 (MaxPooling  (None, 13, 13, 16)        0
2D)

conv2d_4 (Conv2D)            (None, 13, 13, 32)        4640

max_pooling2d_3 (MaxPooling  (None, 6, 6, 32)          0
2D)

conv2d_5 (Conv2D)            (None, 4, 4, 64)          18496

flatten_1 (Flatten)          (None, 1024)              0

dense_2 (Dense)              (None, 64)                65600

dense_3 (Dense)              (None, 10)                650

=================================================================
```

Convolution Neural Network (CNN)

- **모델 디자인 (mnist_paratest.ipynb)**

(과제3)

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	51264
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 64)	73792
dense_1 (Dense)	(None, 10)	650

케라스 신경망 실습 – 패션 아이템 분류

- 이미지는 28x28 크기이고 픽셀 값은 0과 255 사이의 값이다. 레이블(label)은 0에서 9까지의 정수로서 패션 아이템의 범주를 나타낸다.

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

케라스 신경망 실습 – 패션 아이템 분류


```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

케라스 신경망 실습 – 패션 아이템 분류


```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

```
10000/10000 [==============================] - 0s 32us/sample - loss: 0.3560 -
acc: 0.8701
정확도: 0.8701
```

CNN 실습 – 패션 아이템 분류

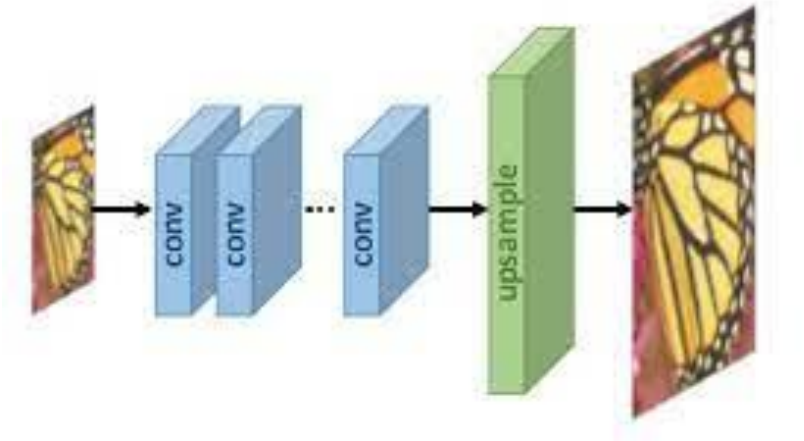
fashio_cnn.ipynb

```
model.add(keras.layers.Conv2D(32, kernel_size=3, activation='relu',
                              padding='same', input_shape=(28,28,1)))
model.add(keras.layers.MaxPooling2D(2))
model.add(keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu',
                              padding='same'))
model.add(keras.layers.MaxPooling2D(2))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(100, activation='relu'))
model.add(keras.layers.Dropout(0.4))
model.add(keras.layers.Dense(10, activation='softmax'))
model.summary()
```

Accuracy?

CNN 실습 – super resolution

https://transcranial.github.io/keras-js/#/mnist-cnn



CNN 실습 – style transfer

원본 영상 style영상 결과 영상

원본 영상 style영상 결과 영상

수고 하셨습니다

jhmin@inhatc.ac.kr