AI 프로그래밍

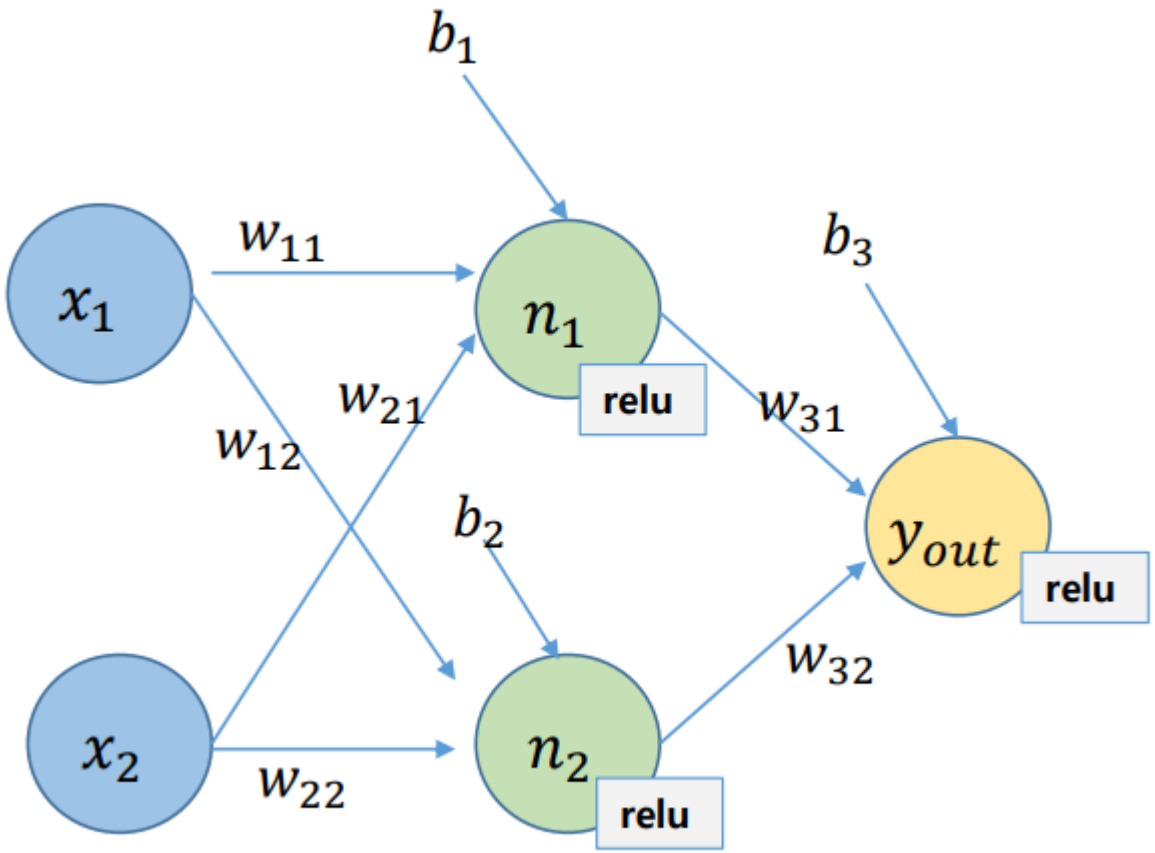
- 2024

12주차

오늘 수업 내용

- **Open CV library**
  - 영상 처리 기초　과제1)
- **Convolutional Neural Network**
  - mnist (과제2)
  - Fashion mnist (과제3)
- **Anaconda 환경 설정 (과제4)**

MLP (다층 퍼셉트론)



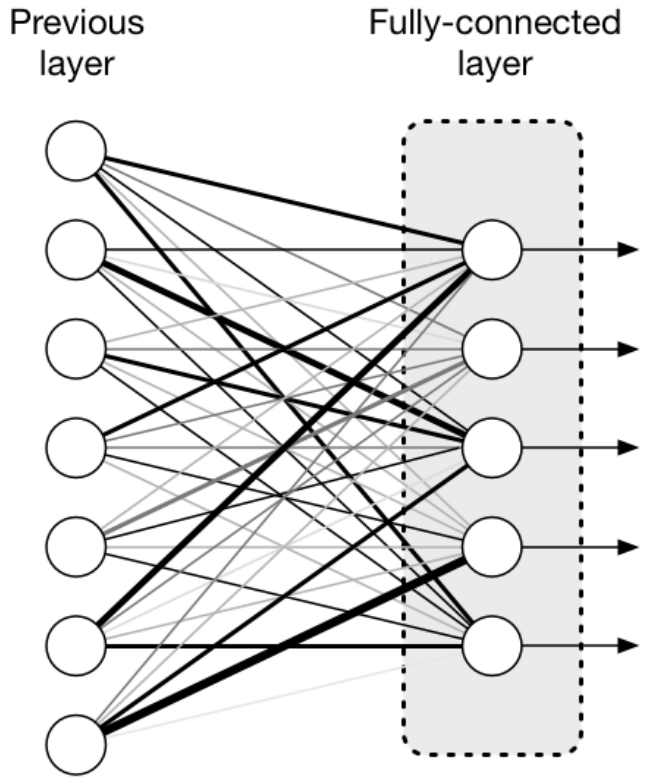
입력 층 : 2
은닉층 1 : 2
출력 층 : 1

완전 연결 신경망 (Fully Connected Neural Network)

tf.keras.layers.**Dense**

완전 연결 계층 : (Fully Connected Layer, Densely connected layer)

한 층 (layer)의 모든 뉴런이 그 다음 층의 모든 뉴런과 연결된 상태



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

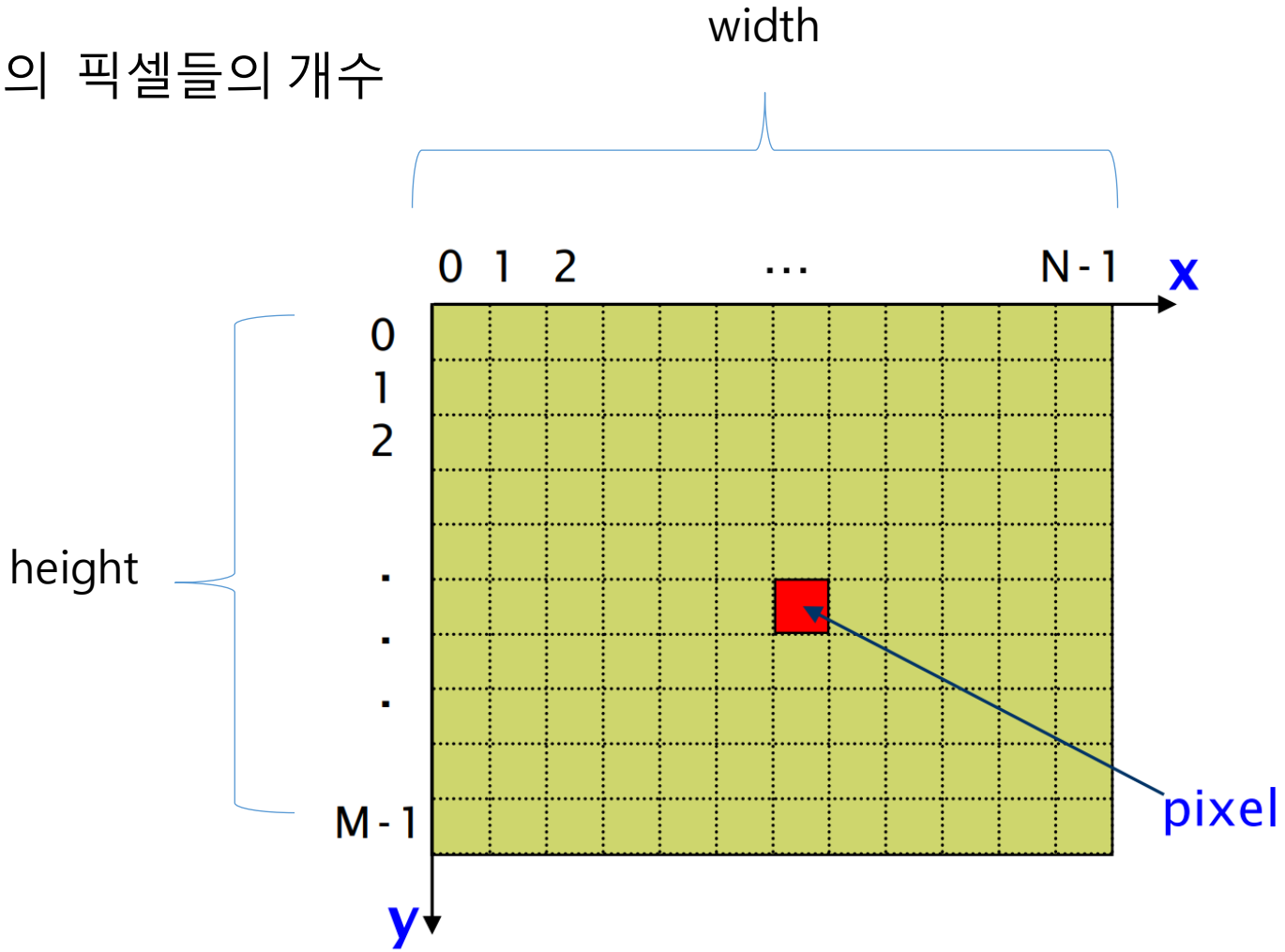
컨볼루션 신경망 (CNN: Convolutional Neural Network)

- **컨볼루션(Convolution Neural Network: CNN) 신경망에서는 앞의 레이어와 현재 레이어가 부분적으로만 연결되어 있다.**

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

영상 처리 기초

- **영상은 픽셀로 이루어짐**
- **해상도 (Resolution)**
  - 2차원 공간 영역에서, x,y 축 방향의 픽셀들의 개수



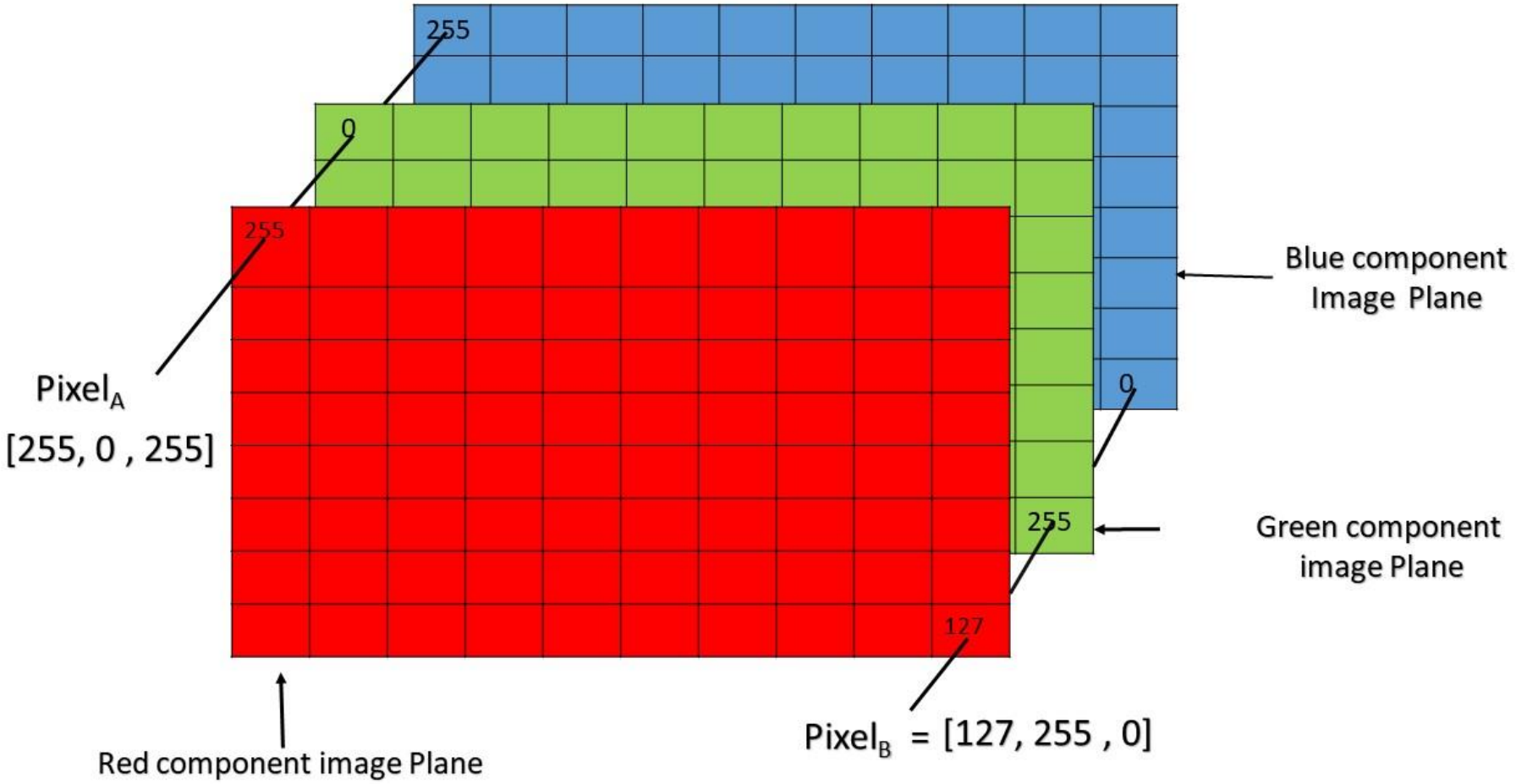
영상 처리 기초

- **영상은 픽셀로 이루어짐.**
- **RGB color space**
  - Red, Green, Blue

White= (255,255,255)
Black = (0,0,0)
Gray =(127,127,127)

Img.shape

=(height, width, rgb==3)



Pixel of an RGB image are formed from the corresponding pixel of the three component images

영상 처리 기초

컬러 이미지

Img.shape

=(height, width, rgb==3)

그레이 이미지

Img.shape

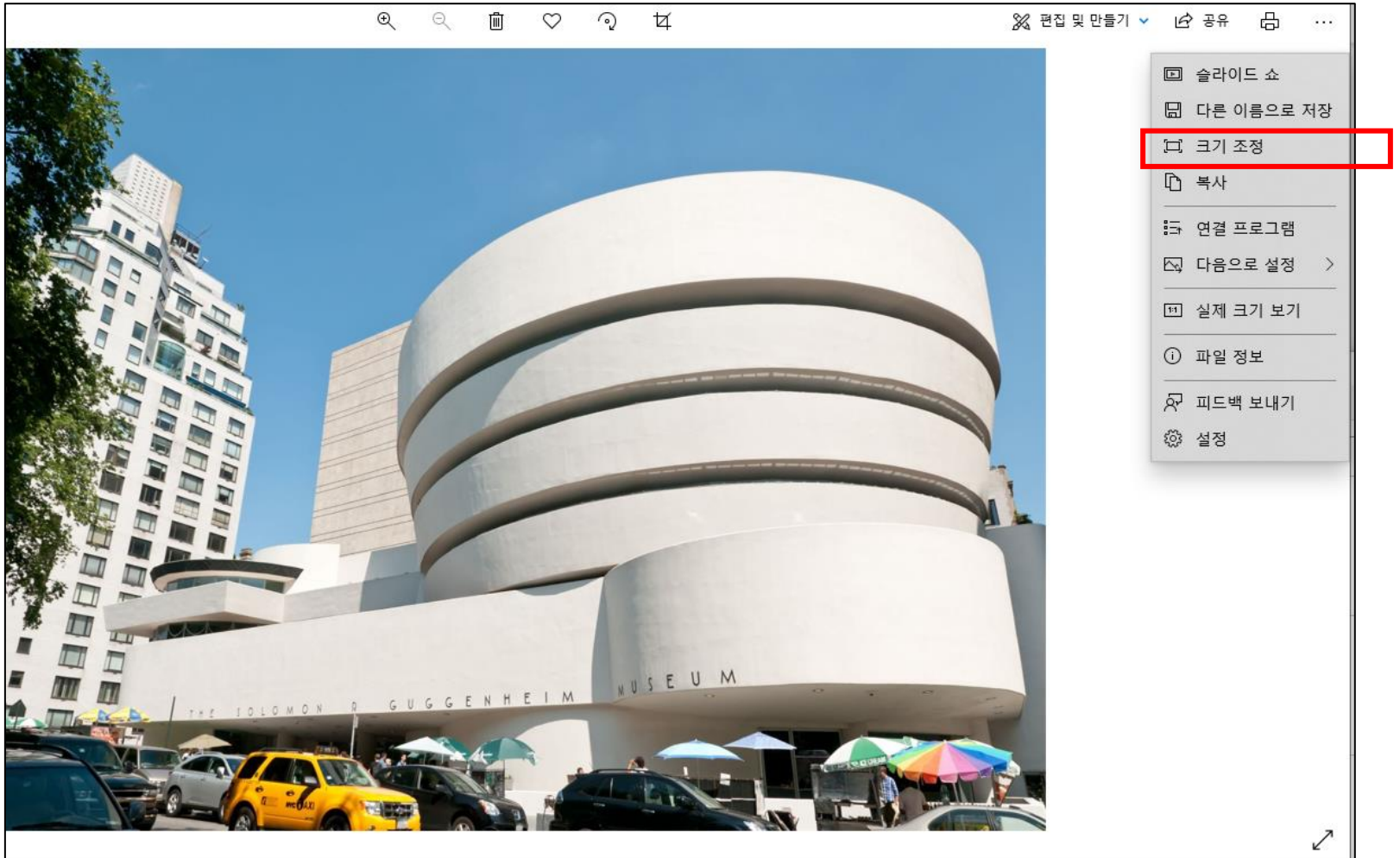
=(height, width)

Gray=0.299*R+0.587*G+0.114*R

영상 처리 기초 – Open CV 실습

인터넷에서 다운 받은 사진 사용

인터넷에서 다운 받은 사진 조정 => 1920x1440 과 비슷한 크기로



영상 처리 기초 – Open CV 실습

Colab에서 실습

```
import cv2
from google.colab.patches import cv2_imshow
img=cv2.imread('/content/test1_1920.jpg')
print('img.ndim =',img.ndim)
print('img.shape =',img.shape)
print('img.dtype = ',img.dtype)
#cv2_imshow(img)
img=cv2.resize(img,(0,0),fx=0.5,fy=0.5)
cv2.imwrite('test1_half.jpg',img)
print(img.shape)
cv2_imshow(img)
```

jupyter

cv2.imshow(img)

영상 처리 기초 – Open CV 실습

test1_half.jpg

test1_half_changed.jpg

test1_halfgray.jpg

test_halfgray_filt1.jpg
test_halfgray_filt2.jpg
test_halfgray_filt3.jpg

test_halfgray_filte.jpg

영상 처리 기초 – Open CV 실습

Colab에서 실습

#colab

```
cv2_imshow(img)
```

#jupyter

```
cv2.imshow(img)
```

영상 처리 기초 – Open CV 실습

Colab에서 실습

```
import cv2
from google.colab.patches import cv2_imshow
img=cv2.imread('/content/test1_1920.jpg')
print('img.ndim =',img.ndim)
print('img.shape =',img.shape)
print('img.dtype = ',img.dtype)
#cv2_imshow(img)
img=cv2.resize(img,(0,0),fx=0.5,fy=0.5)
cv2.imwrite('test1_half.jpg',img)
print(img.shape)
cv2_imshow(img)
```

```
img.ndim = 3
img.shape = (1440, 1920, 3)
img.dtype = uint8

(720, 960, 3)
```

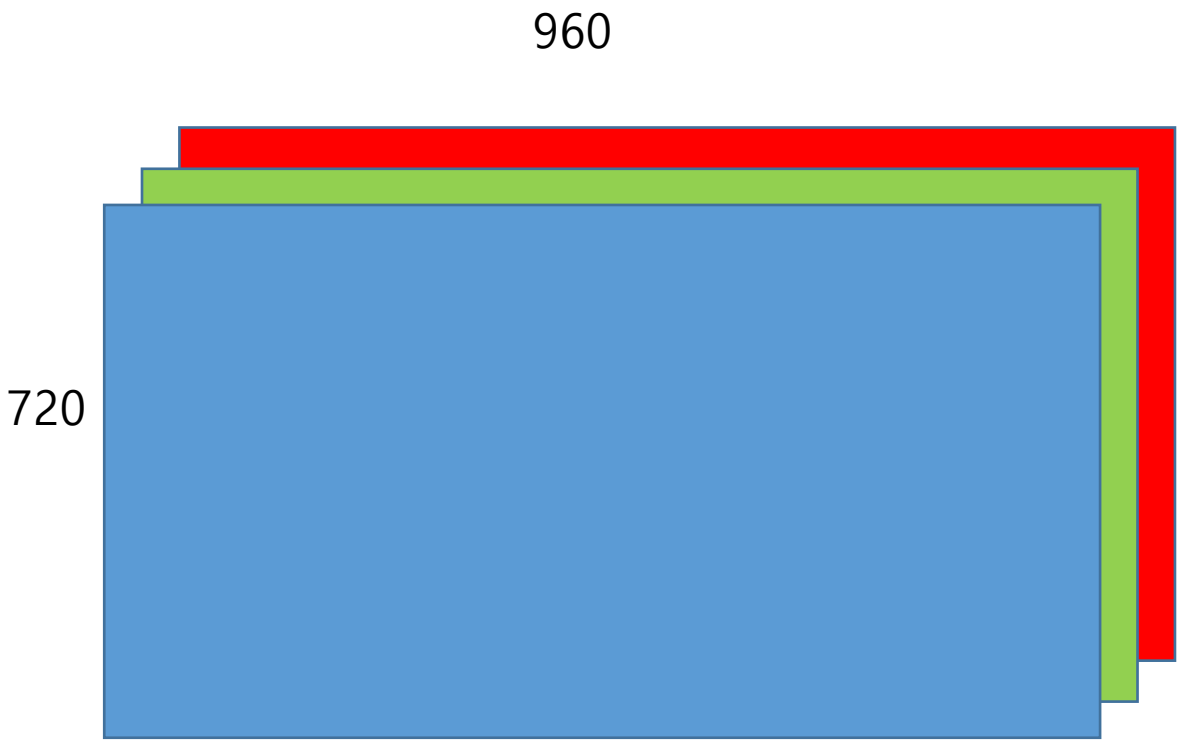
영상 처리 기초 – Open CV 실습


```
img=cv2.imread('/content/test1_half.jpg')
img[100:300,100:300,:]=255
img[100:300,300:500,:]=0
img[100:300,500:700,:]=127

img[300:500,100:300,0]=255
img[300:500,300:500,1]=255
img[300:500,500:700,2]=255

cv2.imwrite('test1_half_changed.jpg',img)

cv2_imshow(img)
```



White= (255,255,255)
Black = (0,0,0)
Gray =(127,127,127)

영상 처리 기초 – Open CV 실습

영상 처리 기초 – Open CV 실습


```
img=cv2.imread('/content/test1_half.jpg')
gimg=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

cv2.imwrite('test1_halfgray.jpg',gimg)
print(gimg.shape)
cv2_imshow(gimg)
```

```
(720, 960)
```

영상 처리 기초 – Open CV 실습

Convolution(컨벌루션)은 주변 화소값들에 가중치를 곱해서 더한 후에 이것을 새로운 화소값으로 하는 연산이다

컨볼루션 연산

입력

2	1	0
1	3	2
4	3	2

커널

1	2	0
1	2	0
3	2	4

컨볼루션 수행 결과 ?"

2*1+1*2+0*0+1*1+3*2+2*0+4*3+3*2+2*4= ?

영상 처리 기초 – Open CV 실습


```
import numpy as np
gimg=cv2.imread('test1_halfgray.jpg')
kernel1=np.ones((3,3),dtype=np.float64)/9.
gimg1=cv2.filter2D(gimg,-1,kernel1)
kernel2=np.ones((5,5),dtype=np.float64)/25.
gimg2=cv2.filter2D(gimg,-1,kernel2)
kernel3=np.ones((7,7),dtype=np.float64)/49.
gimg3=cv2.filter2D(gimg,-1,kernel3)
cv2_imshow(gimg1)
cv2_imshow(gimg2)
cv2_imshow(gimg3)
cv2.imwrite('test_halfgray_filt1.jpg',gimg1)
cv2.imwrite('test_halfgray_filt2.jpg',gimg2)
cv2.imwrite('test_halfgray_filt3.jpg',gimg3)
```

```
kernel1 [[0.11 0.11 0.11]
 [0.11 0.11 0.11]
 [0.11 0.11 0.11]]

kernel2 [[0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]
 [0.04 0.04 0.04 0.04 0.04]]

kernel3 [[0.02 0.02 0.02 0.02 0.02 0.02 0.02]
 [0.02 0.02 0.02 0.02 0.02 0.02 0.02]
 [0.02 0.02 0.02 0.02 0.02 0.02 0.02]
 [0.02 0.02 0.02 0.02 0.02 0.02 0.02]
 [0.02 0.02 0.02 0.02 0.02 0.02 0.02]
 [0.02 0.02 0.02 0.02 0.02 0.02 0.02]
 [0.02 0.02 0.02 0.02 0.02 0.02 0.02]]
```

영상 처리 기초 – Open CV 실습

인하공전 컴퓨터 정보공학과

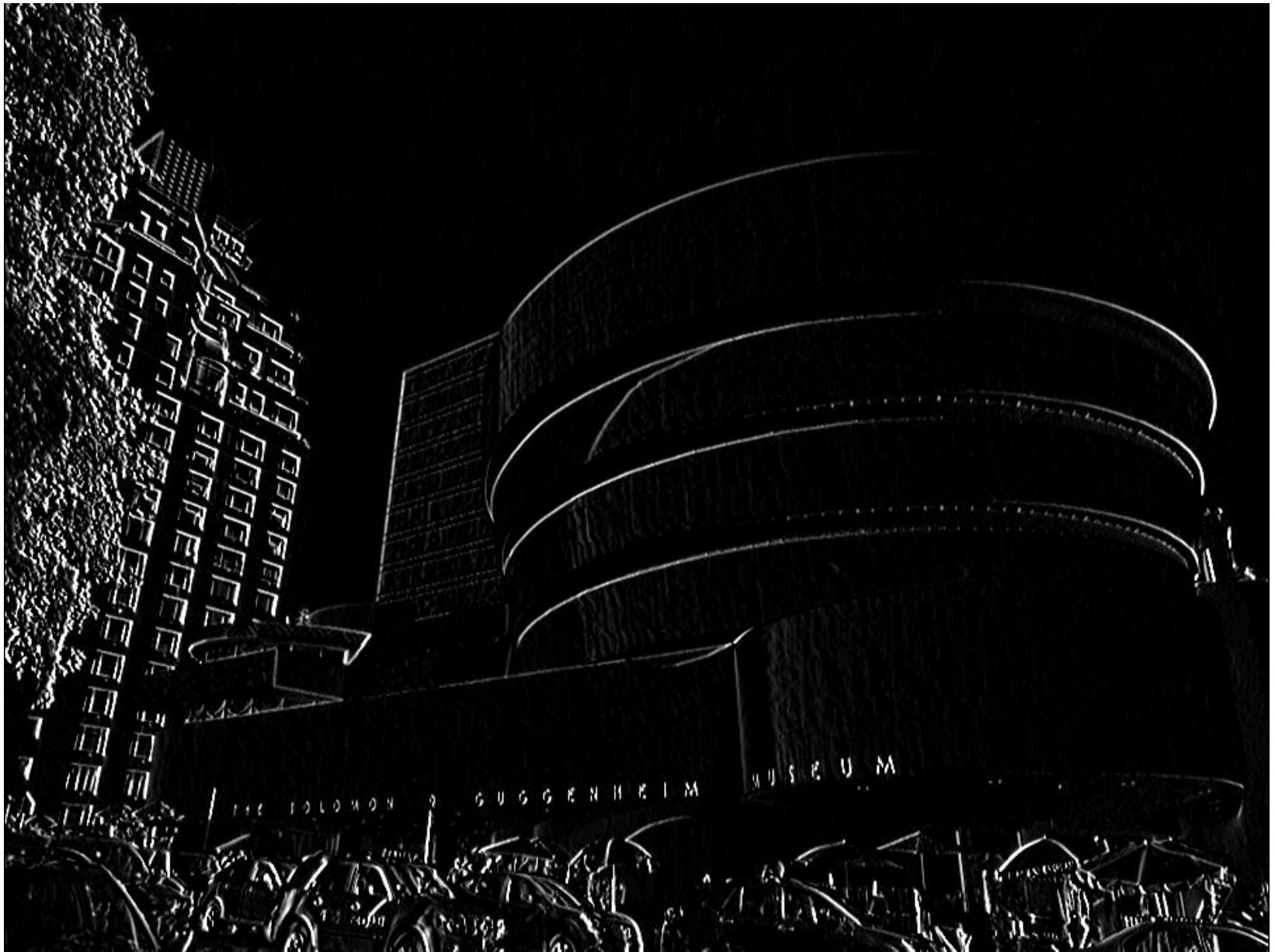
영상 처리 기초 – Open CV 실습

영상 처리 기초 – Open CV 실습

영상 처리 기초 – Open CV 실습


```
import numpy as np
gimg=cv2.imread('test1_halfgray.jpg')
kernel_x = np.array([[-1,0,1], [-2,0,2],[-1,0,1]])
gimge=cv2.filter2D(gimg,-1,kernel_x)
cv2_imshow(gimge)
cv2.imwrite('test_halfgray_filte.jpg',gimge)
```

영상 처리 기초 – Open CV 실습



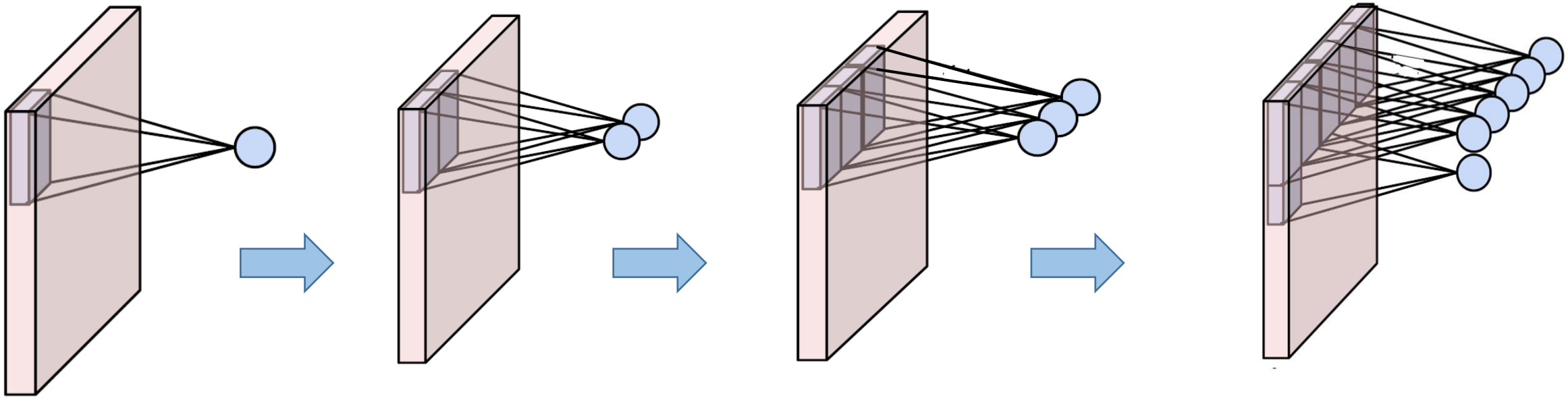
영상 처리 기초 – Open CV 실습

과제1 : 아래 파일 upload

1) **test1_half.jpg**
2) **test1_half_changed.jpg**
3) **test1_halfgray.jpg**
4) **test_halfgray_filt1.jpg**
5) **test_halfgray_filt2.jpg**
6) **test_halfgray_filt3.jpg**
7) test_halfgray_filte.jpg

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 컨벌루션 신경망은 2차원 형태의 입력을 처리하기 때문에, 이미지 처리에 특화되어 있다.

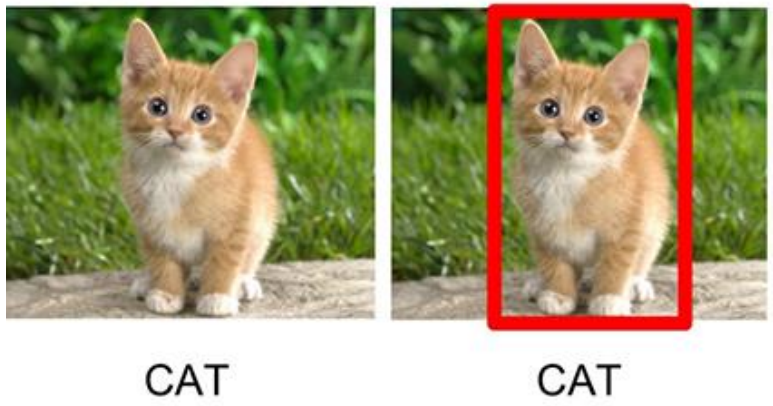


[CNN https://cs231n.github.io/]

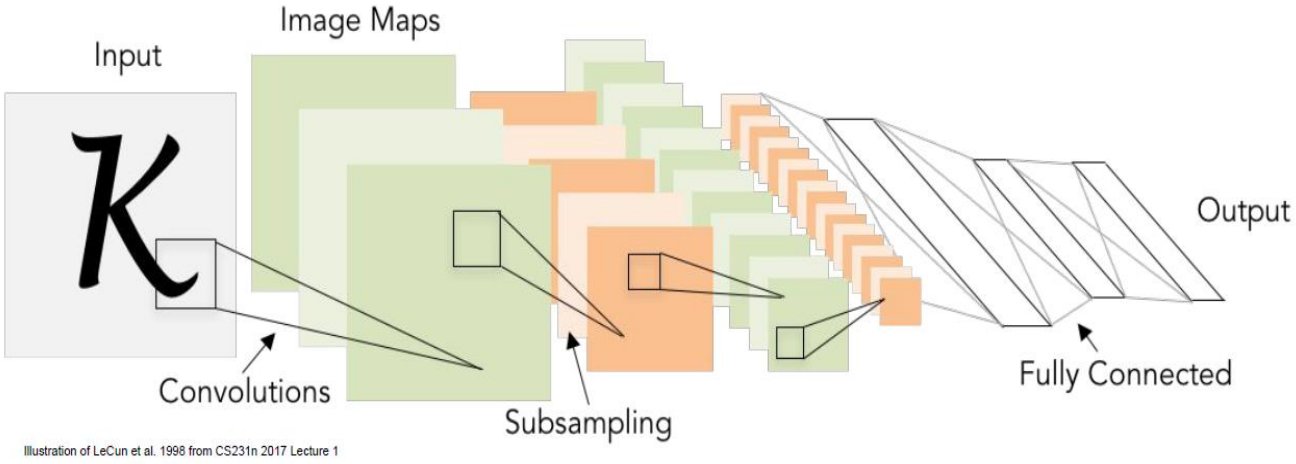
https://keras.io/api/layers/core_layers/

컨볼루션 신경망 (CNN: Convolutional Neural Network)

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



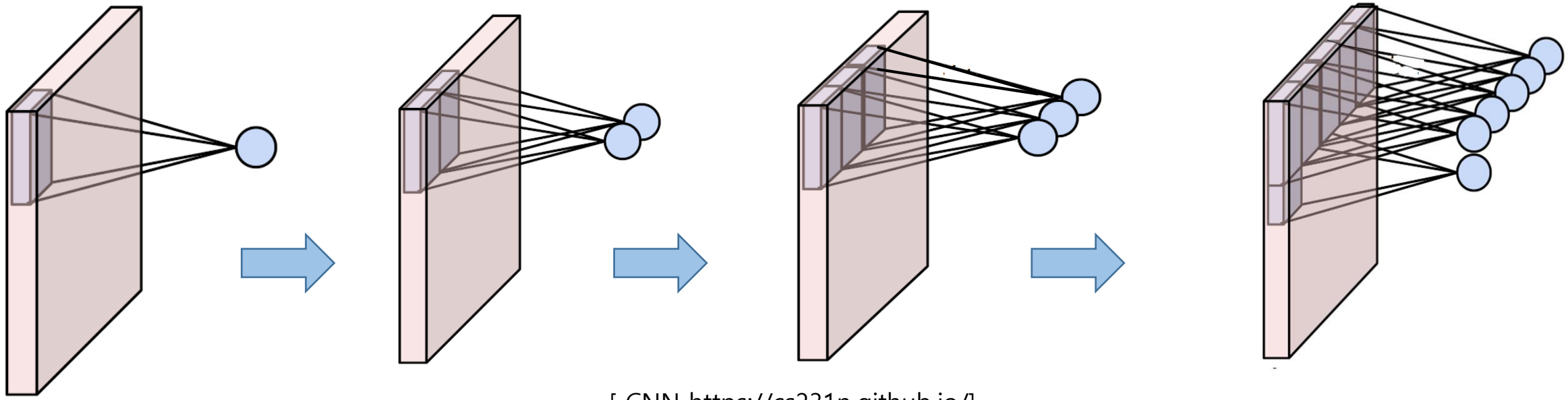
[영상 인식]



[합성곱 신경망 (Stanford cse231)]

[영상 변환]

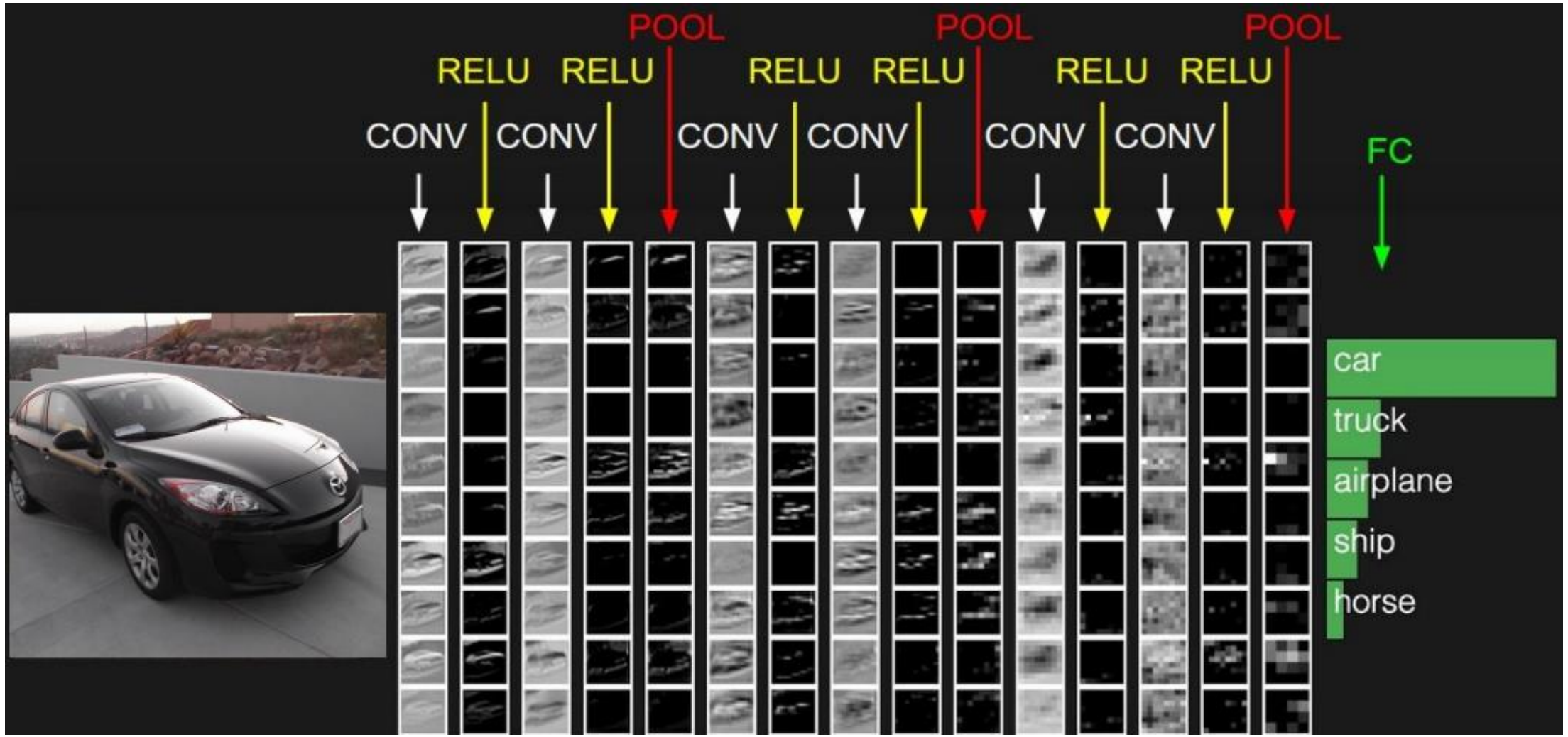
컨볼루션 신경망 (CNN: Convolutional Neural Network)



[CNN https://cs231n.github.io/]

신경망 안에서 합성곱(Convolution) 연산을 수행

컨볼루션 신경망 (CNN: Convolutional Neural Network)



CONV: Convolution Layer
POOL : Pooling Layer

[CNN https://cs231n.github.io/]

컨볼루션 연산

입력

2	1	0
1	3	2
4	3	2

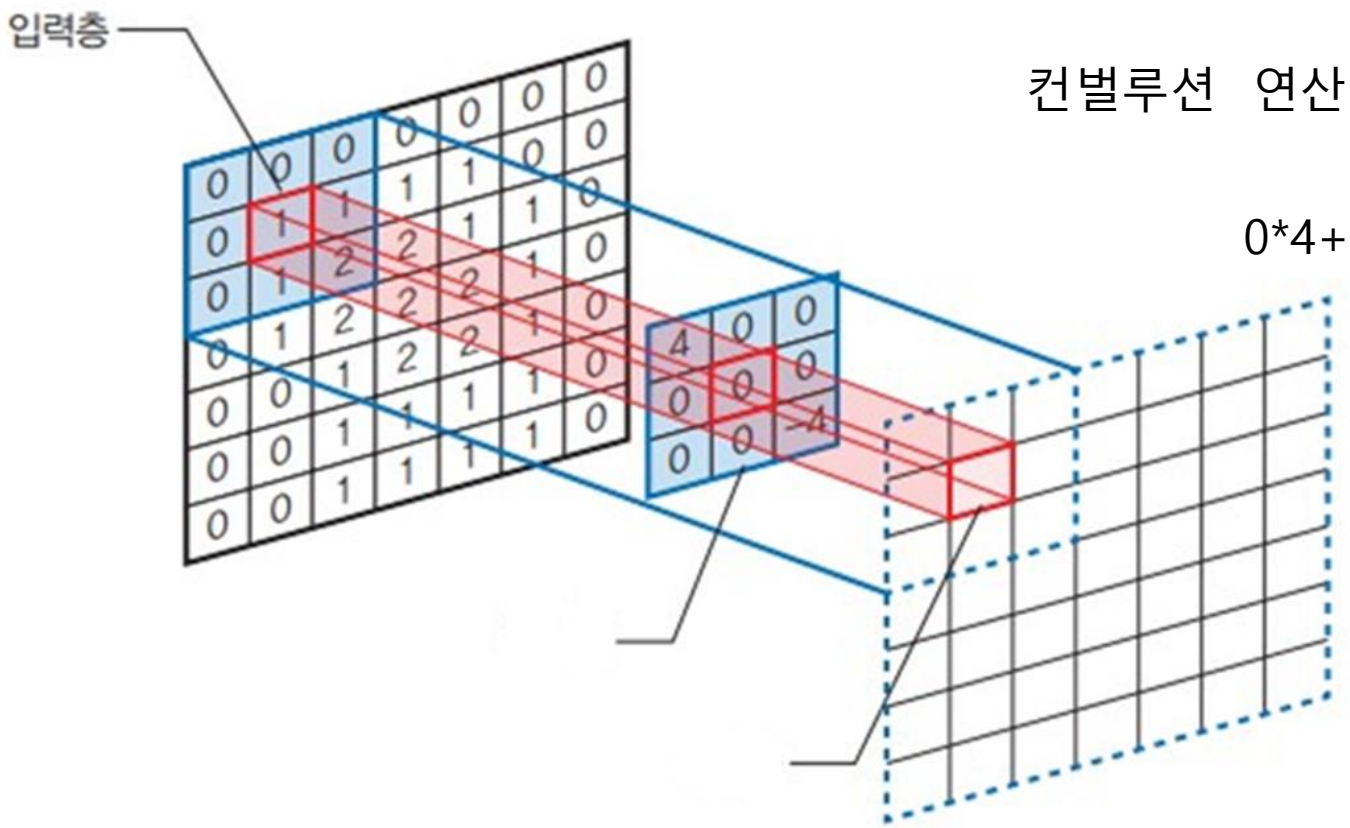
커널

1	2	0
1	2	0
3	2	4

컨볼루션 수행 결과 ?"

2*1+1*2+0*0+1*1+3*2+2*0+4*3+3*2+2*4= ?

컨볼루션 (Convolution) 계산



컨벌루션 연산 수행 결과

0*4+0*0+0*0+1*0+1*0+0*0+1*0+2*-4= -8

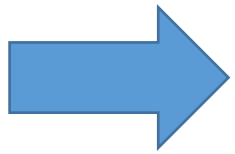
참조 딥러닝 익스프레스

컨벌루션 (Convolution) 계산

다음과 같은 입력에서 (3,3) 커널과 valid 패딩으로 컨벌루션을 수행합니다. 컨벌루션의 결과를 계산해 보시오

3	0	9	1	2
5	1	2	0	7
8	2	4	1	3
2	1	5	3	6
4	1	6	2	7

2	0	1
2	0	1
2	0	1

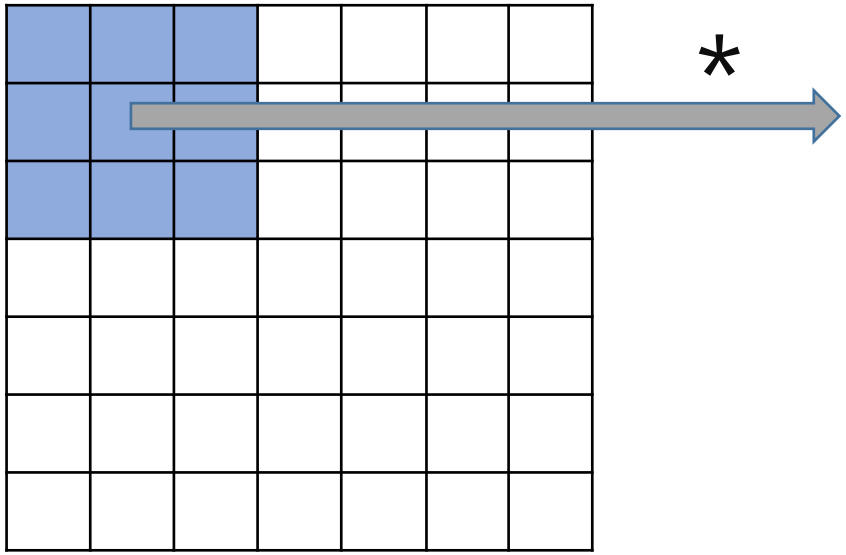


47	8	42
41	12	38
43	14	46

3*2+5*2+8*2+9+2+4=6+10+16+9+2+4=32+15=47

컨볼루션 (Convolution)

7x7 input image



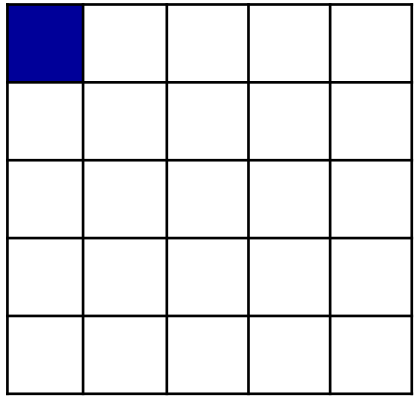
*

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

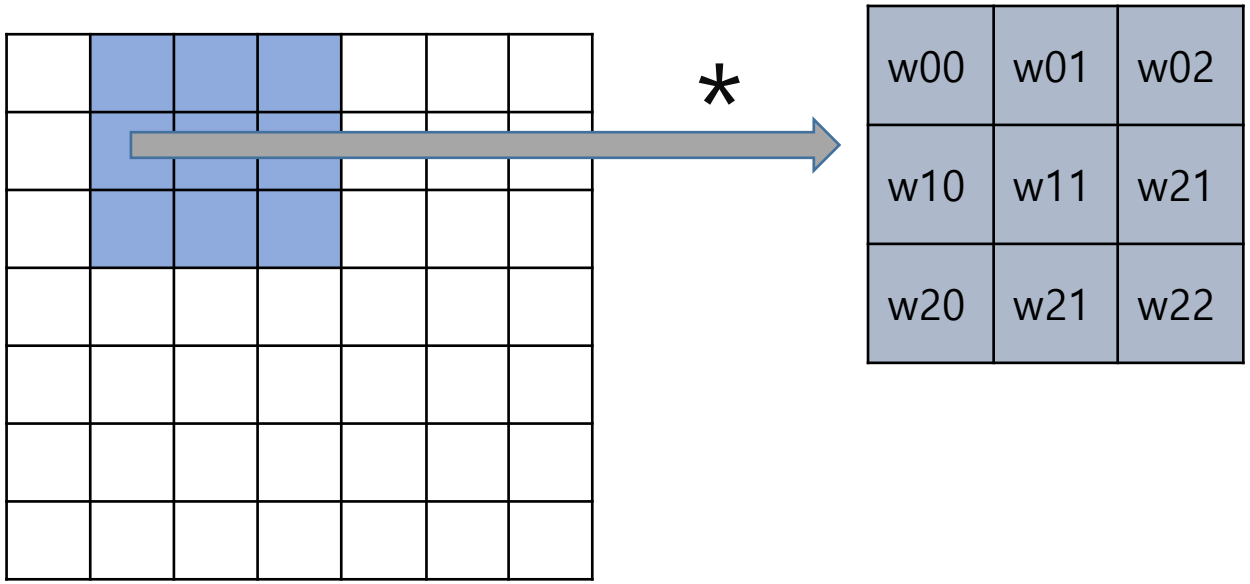


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

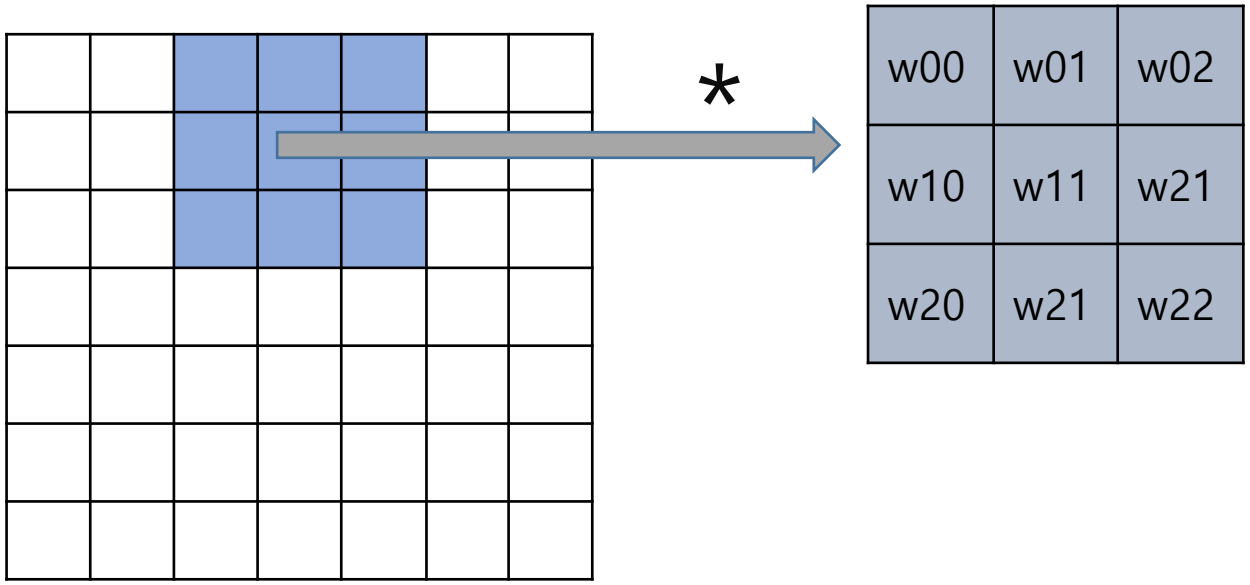


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

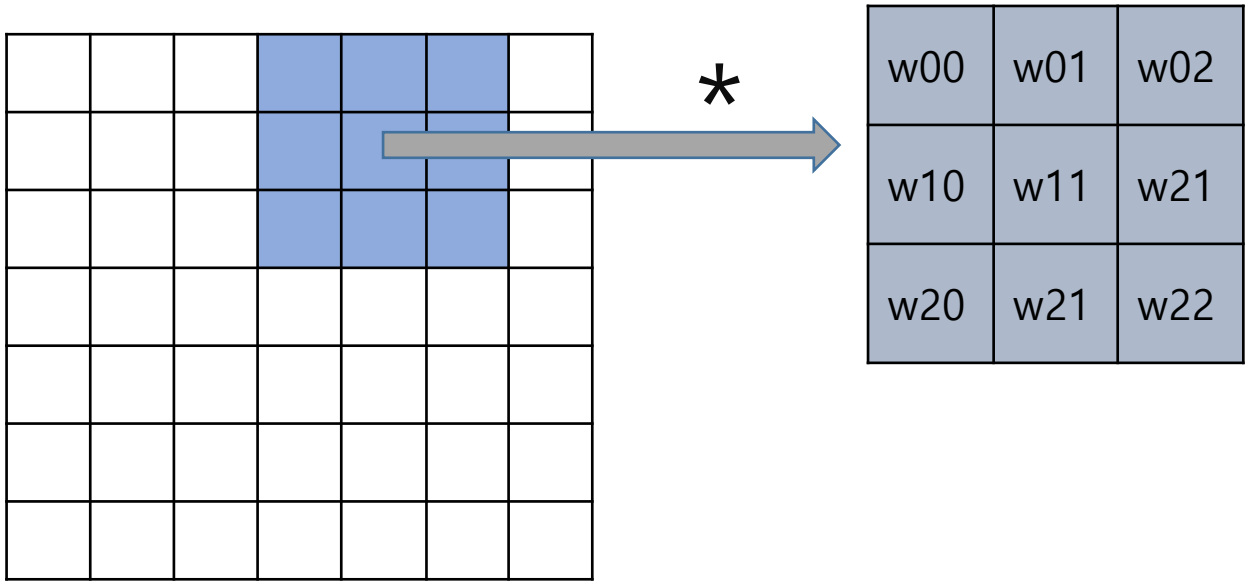


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

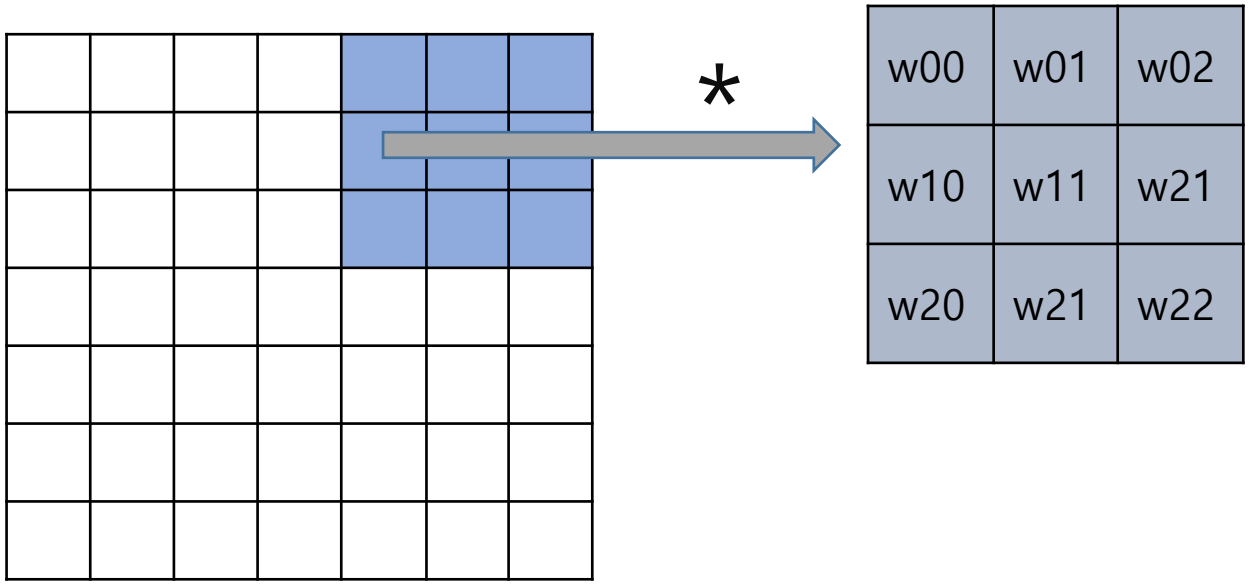


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

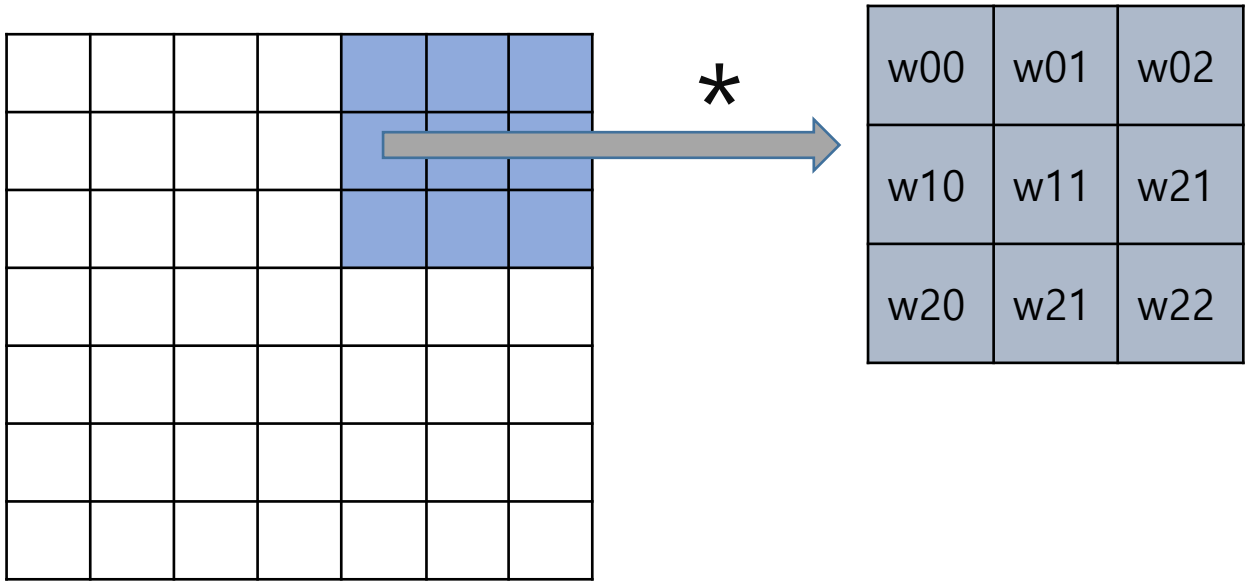


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

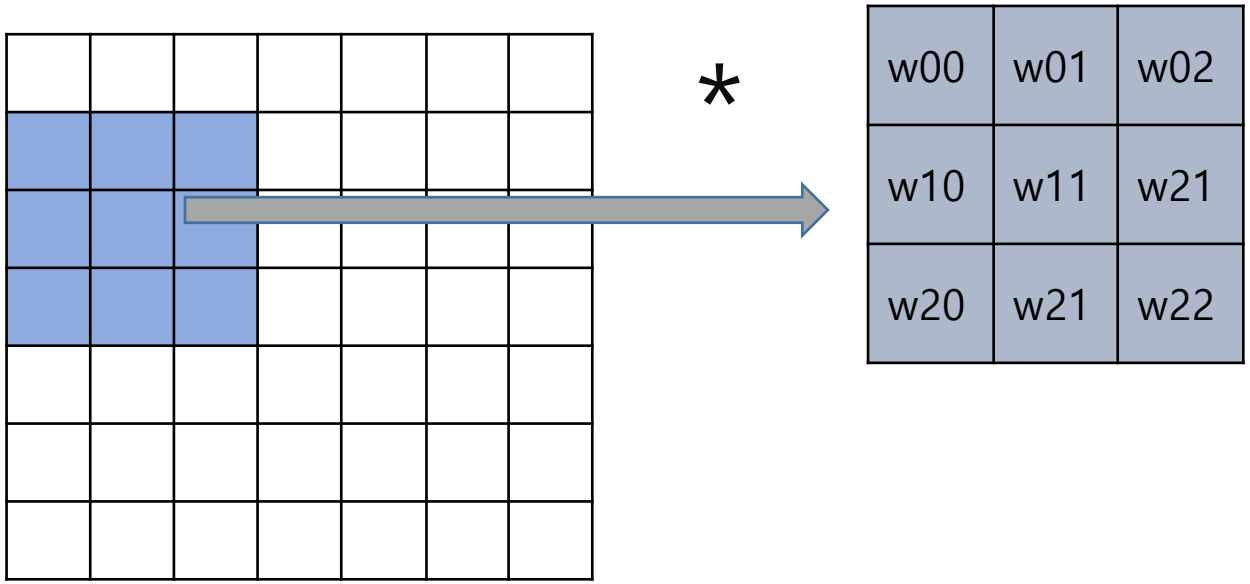


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

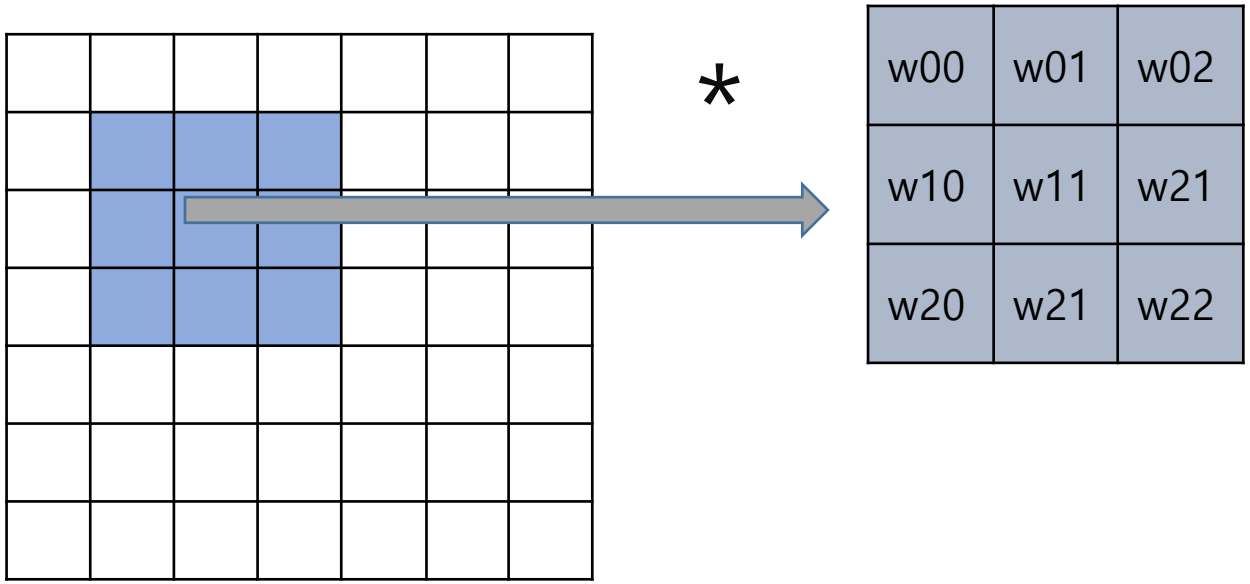


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image



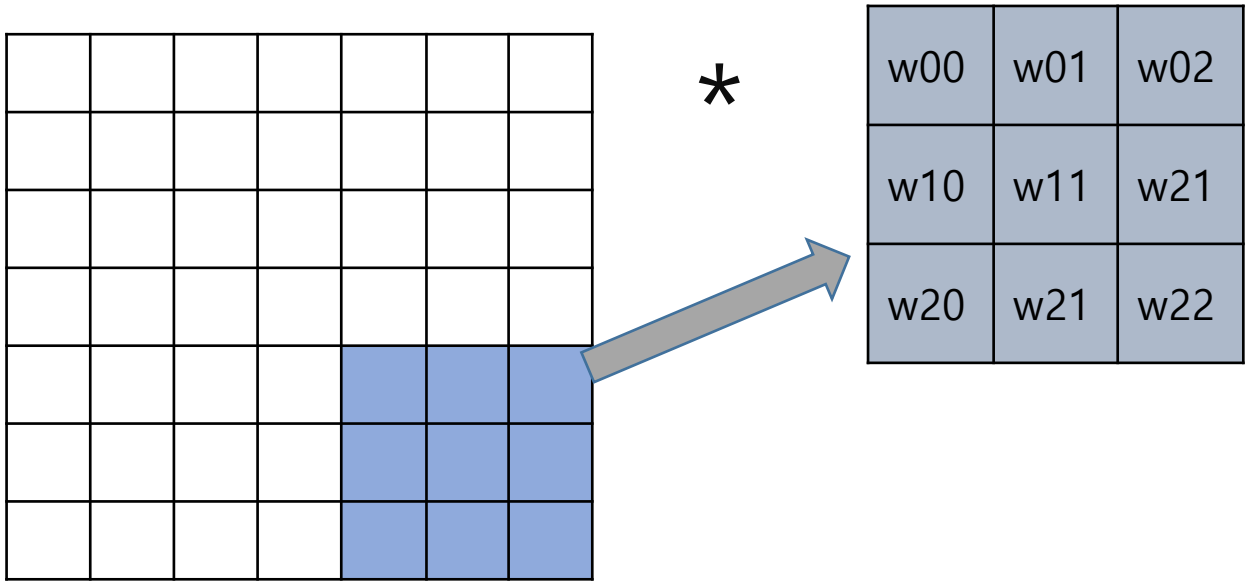
컨볼루션 신경망 (CNN: Convolutional Neural Network)

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.**
  - 컨벌루션를 수행한 결과는 특징맵(feature map)
  - **컨벌루션 신경망에서는 커널의 가중치들이 학습됨.**

컨볼루션 신경망 (CNN: Convolutional Neural Network)

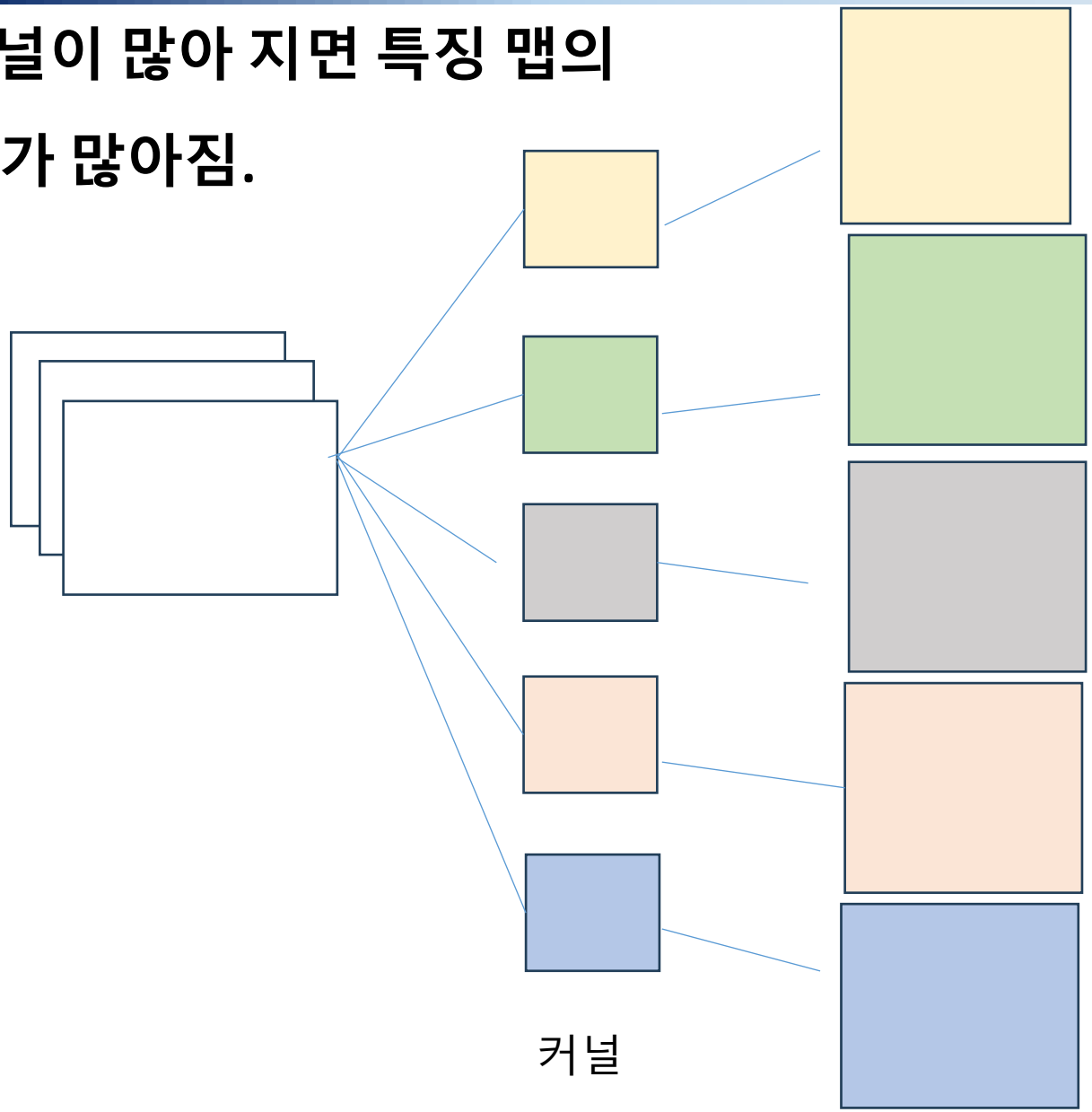
- 보폭(stride) 은 커널을 적용하는 거리이다. 보폭이 1이면 커널을 한 번에 1픽셀씩 이동하면서 커널을 적용하는 것이다. 보폭이 2라는 것은 하나씩 건너뛰면서 커널을 적용

컨볼루션 신경망 (CNN: Convolutional Neural Network)

- **패딩(padding)은 이미지의 가장자리를 처리하기 위한 기법이다.**
  - **Valid:** 커널을 입력 이미지 안에서만 움직인다.
  - **Same :**입력 이미지의 주변을 특정값으로 채우고 움직이기 때문에 결과 이미지가 임력 이미지와 크기가 같음.

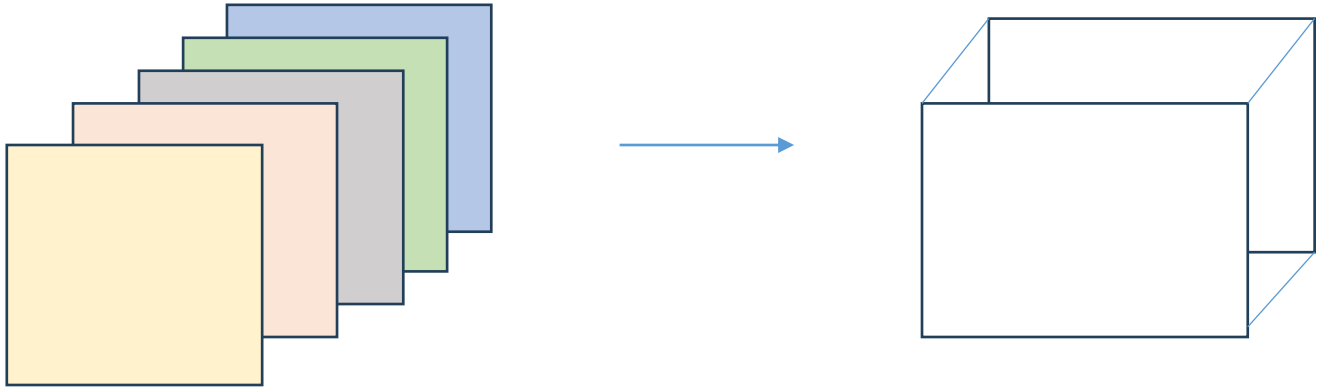
컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 커널이 많아 지면 특징 맵의 개수가 많아짐.



커널의 개수=특징맵 개수

컨볼루션 신경망 (CNN: Convolutional Neural Network)



특징 맵

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)

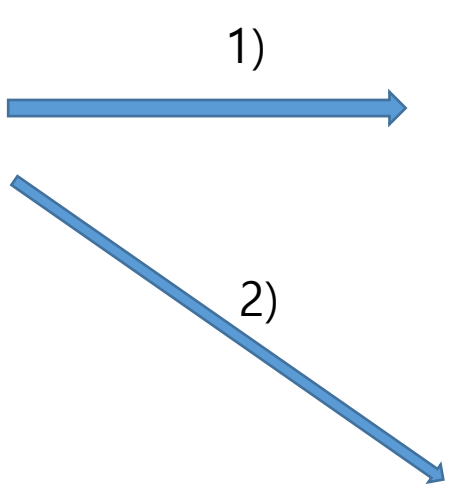
인하공전 컴퓨터 정보과

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

풀링 (Pooling) 계산

1) 다음의 feature 맵의 (2,2) 최대 풀링(Max Pooling) 결과를 구하시오
2) 다음의 feature 맵의 (2,2) 평균 풀링(Average Pooling) 결과를 구하시오

9	10	6	2
4	1	4	8
8	0	7	1
8	8	3	1



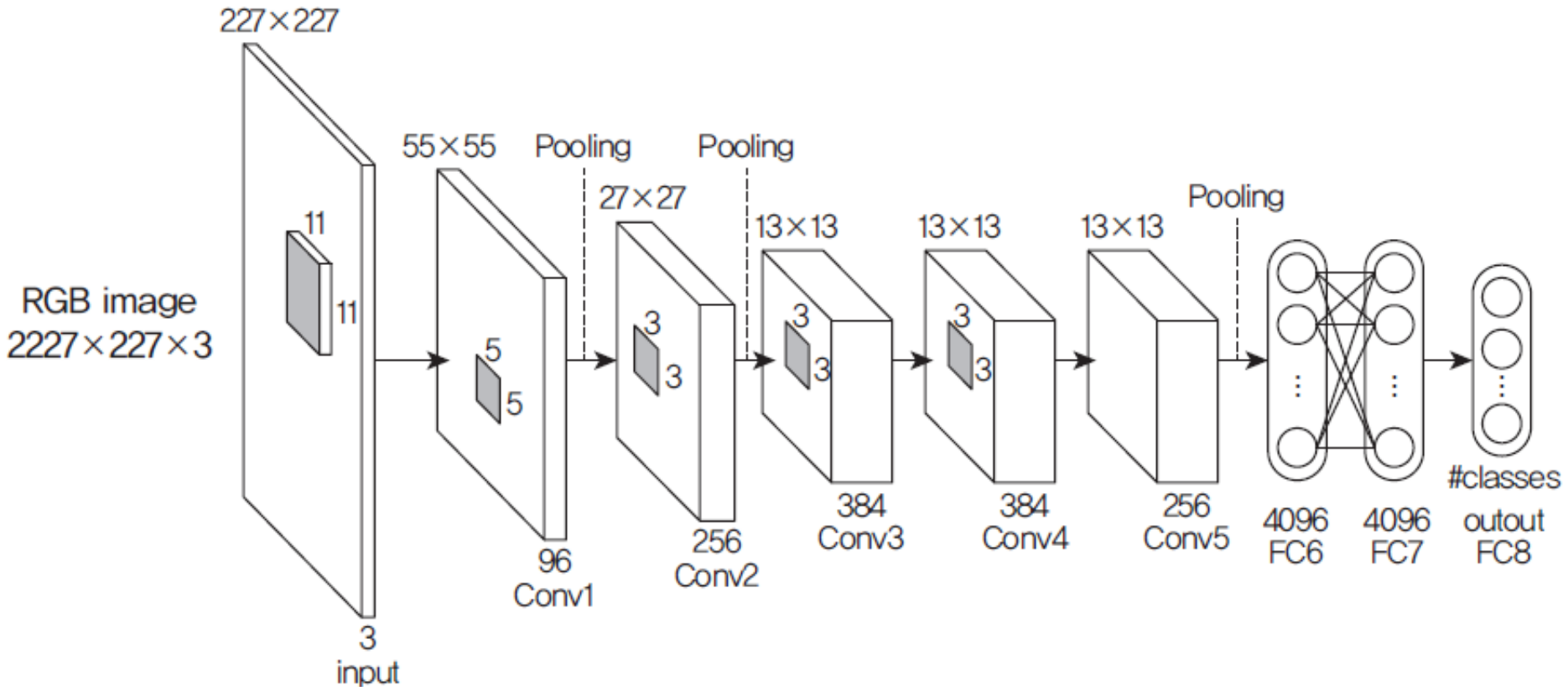
10	8
8	7

6	5
8	3

풀링의 장점

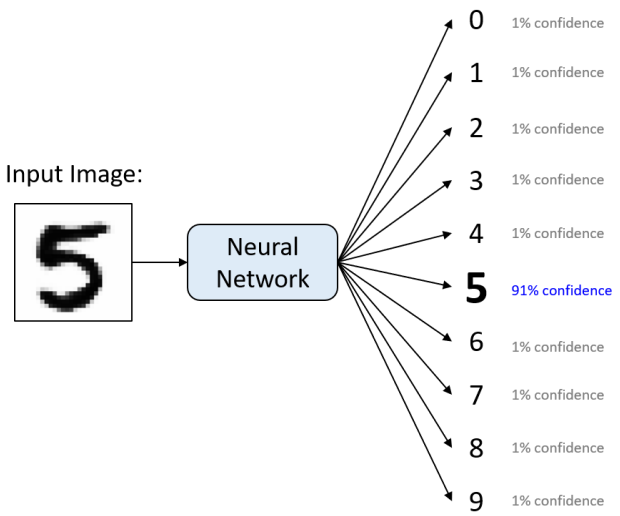
- 레이어의 크기가 작아짐.
- 계산이 빨라짐.
- 정보가 압축됨.

컨볼루션 신경망 (CNN: Convolutional Neural Network)

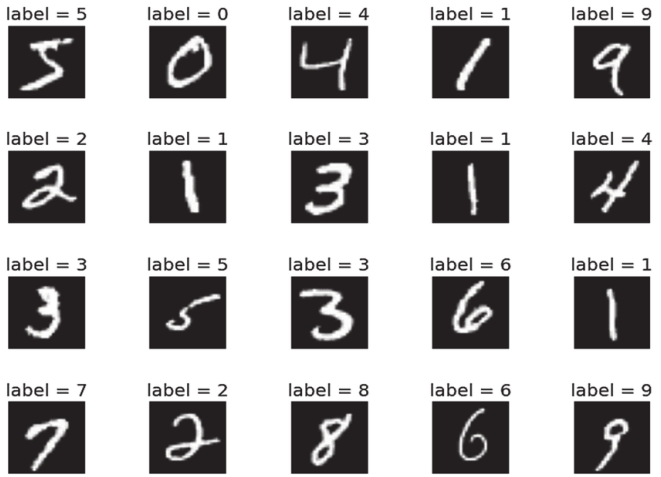


MNIST 손 글씨 이미지 분류

- **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]

Dense layer(fully connected layer) MNIST


```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(512, activation='relu', input_shape=(784,)))
model.add(tf.keras.layers.Dense(10, activation='sigmoid'))
```

Dense layer(fully connected layer) MNIST


```
train_images = train_images.reshape((60000, 784))
train_images = train_images.astype('float32') / 255.0

test_images = test_images.reshape((10000, 784))
test_images = test_images.astype('float32') / 255.0
```

MNIST 필기체 숫자 인식-CNN


```
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-CNN


```
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-CNN


```
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


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

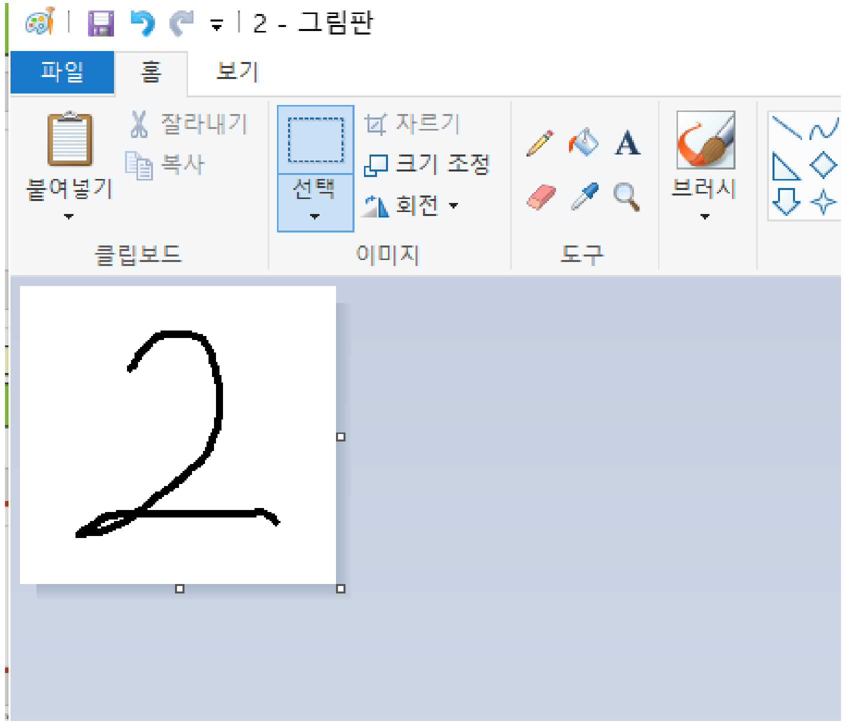
```
Epoch 1/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1414 -
accuracy: 0.9560
...
Epoch 5/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0194 -
accuracy: 0.9940
```

MNIST 필기체 숫자 인식


```
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



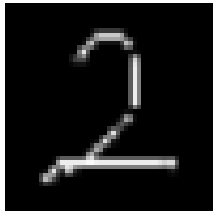
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	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 (과제 2)


```
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



숫자 image와 예측 결과 제출

MNIST 필기체 숫자 인식

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

케라스 신경망 실습 – 패션 아이템 분류

- 이미지는 28x28 크기이고
- 픽셀 값은 0과 255 사이
- 레이블(label)은 0에서 9까지의

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

완전 연결 신경망: FC dense layer

```
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 – 패션 아이템 분류

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_1 (MaxPooling 2D)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 100)	313700
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 10)	1010

```
                callbacks=[checkpoint_cb, early_stopping_cb])
```

Fashion mist 성능 비교 (과제3)

- **CNN으로 구현한 code의 FASION MIST DATA 분류 성능을 측정하시오.**

- **Fashion_cnn.ipynb 사용**

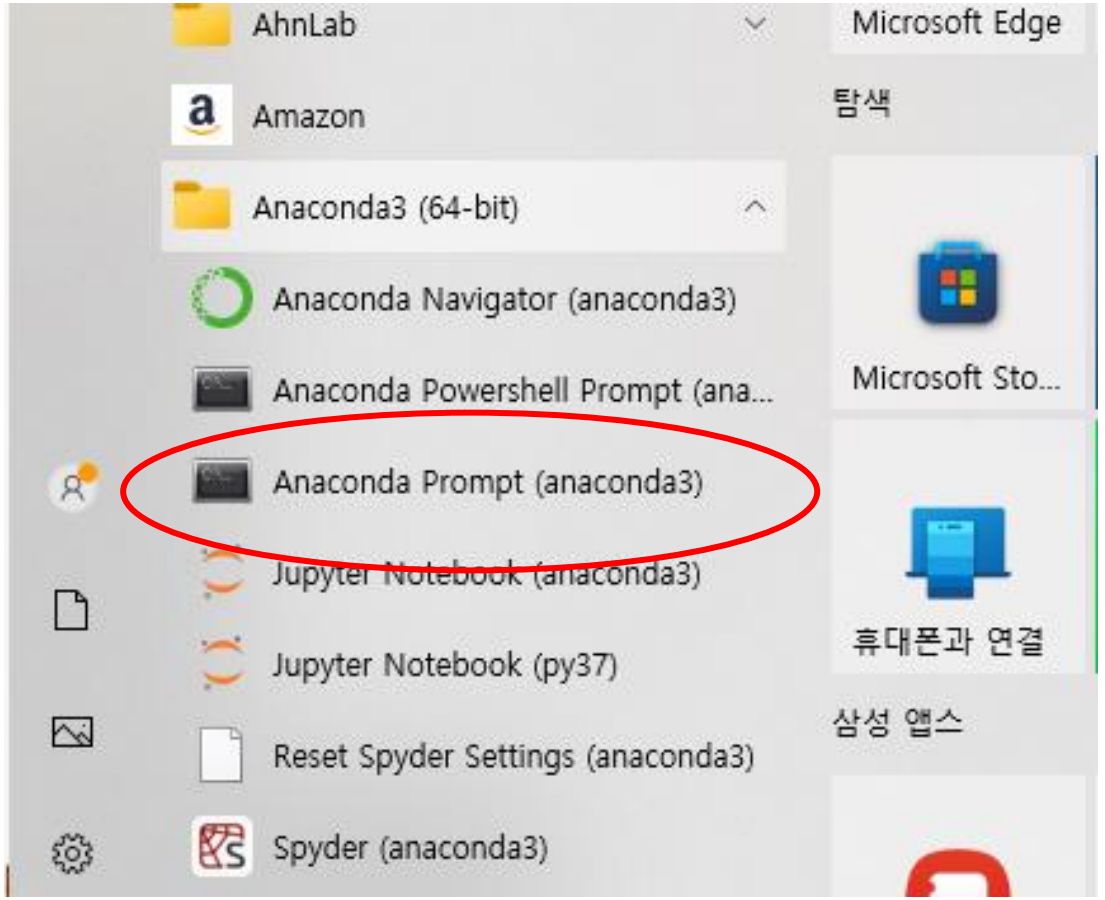
FC network	CNN network
0.8701	

anaconda

파이썬 라이브러리 환경 구축

ml_env1 : tensorflow, keras,numpy
ml_env2 : scikit-learn,numpy

base)
ml_env1)
ml_env2)



2.3 실습 환경 설정 (과제 4)

ml_(본인 initial)1, ml_(본인 initial) 2 라는 가상환경 2개를 만들기
예) ml_jhmin1, ml_jhmin2

ml_jhmin1 : keras,numpy
ml_jhmin2 : scikit-learn,matplotlib

conda install 은 pip install 로 변경 가능

- **(base)>conda create –n ml_jhmin1 python=3.9.0 ## 가상환경 ml_jhmin1 생성**
- **(base)>conte create –n ml_jhmin2 python=3.9.0 ## 가상환경 ml_jhmin2 생성**
- **(base)>conda env list ## 가상 환경 list 확인**
- **(base)>conda activate ml_jhmin1 ## ml_jhmin1 활성화**

>>(ml_jhmin1)> conda install keras
>>(ml_jhmin1)>conda install numpy

2.3 실습 환경 설정

conda install 은 pip install 로 변경 가능

- **(base)>conda activate ml_jhmin2**
- **(ml_jhmin2)> conda install matplotlib**
- **(ml_jhmin2)> conda install scikit-learn**
- **(ml_jhmin2)> pip list # 가상 환경에 설치된 library 확인**
- **(ml_jhmin2)> conda deactivate ##기본 가상 환경 실행**
- **(base)**

과제4 제출) conda env list
ml_jhmin1) pip list
ml_jhmin2) pip list

스크린샷 세 장 제출

수고하셨습니다

jhmin@inhatc.ac.kr