AI 프로그래밍

- 2024

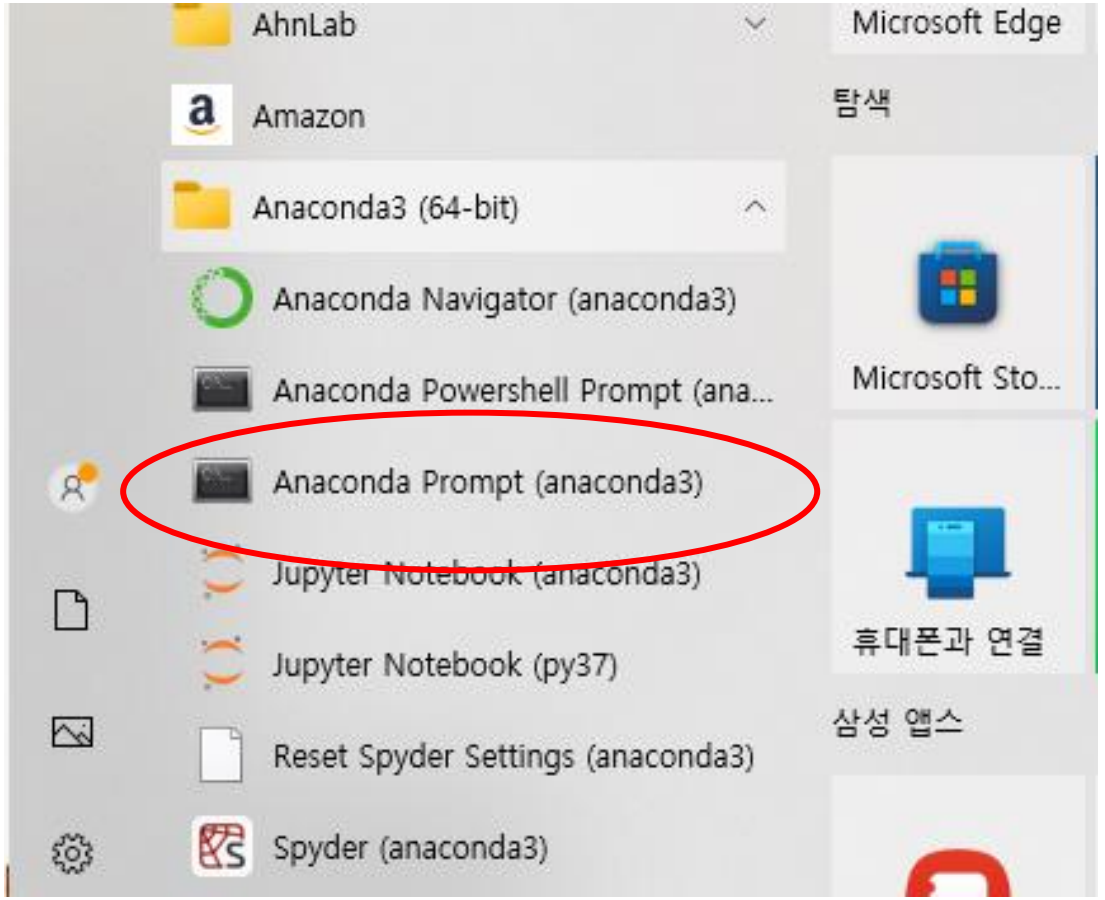
13주차

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

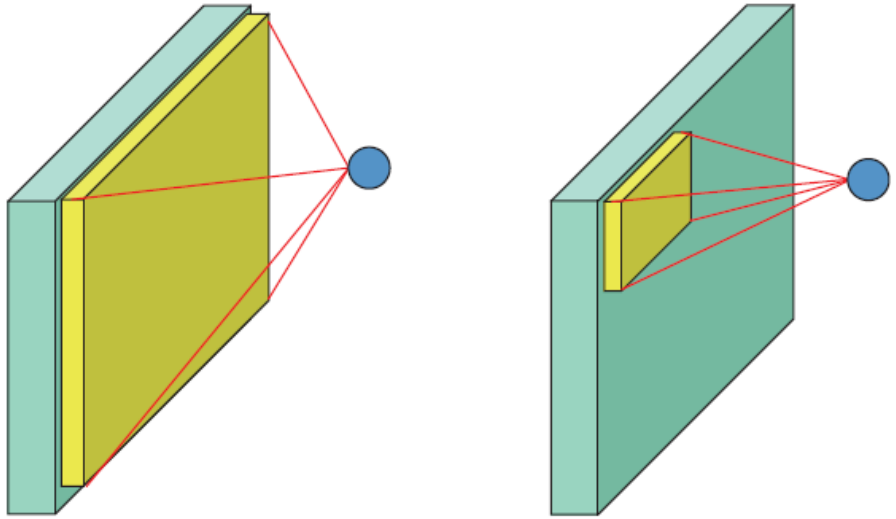
- **(base)>conda activate ml_jhmin2**
- **(ml_jhmin2)> conda install matplotlib**
- **(ml_jhmin2)> conda install scikit-learn**
- **(ml_jhmin2)> pip list # 가상 환경에 설치된 library 확인**
- **(ml_jhmin2)> conda deactivate ##기본 가상 환경 실행**
- **(base)**

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

스크린샷 세 장 제출

컨볼루션 신경망 (CNN: Convolutional Neural Network)

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

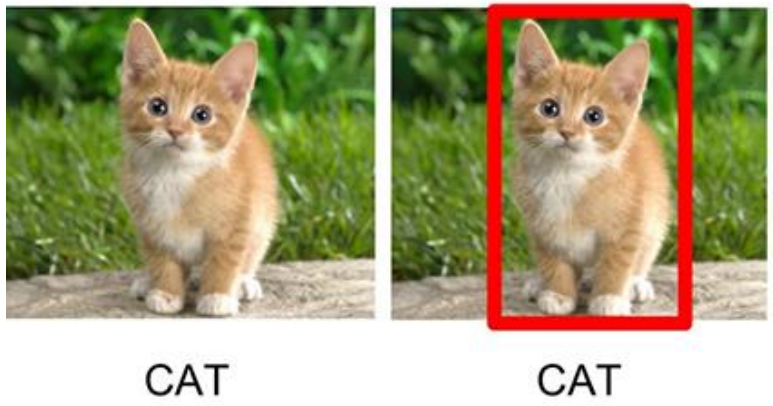


(a) 완전연결 신경망 (b) 컨볼루션 신경망

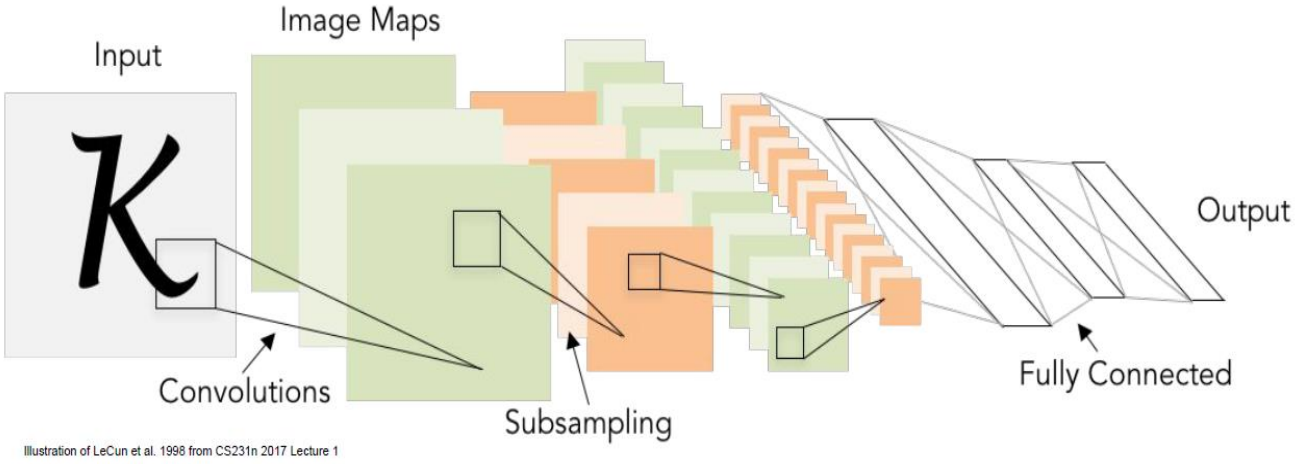
https://keras.io/api/layers/core_layers/

컨볼루션 신경망 (CNN: Convolutional Neural Network)

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



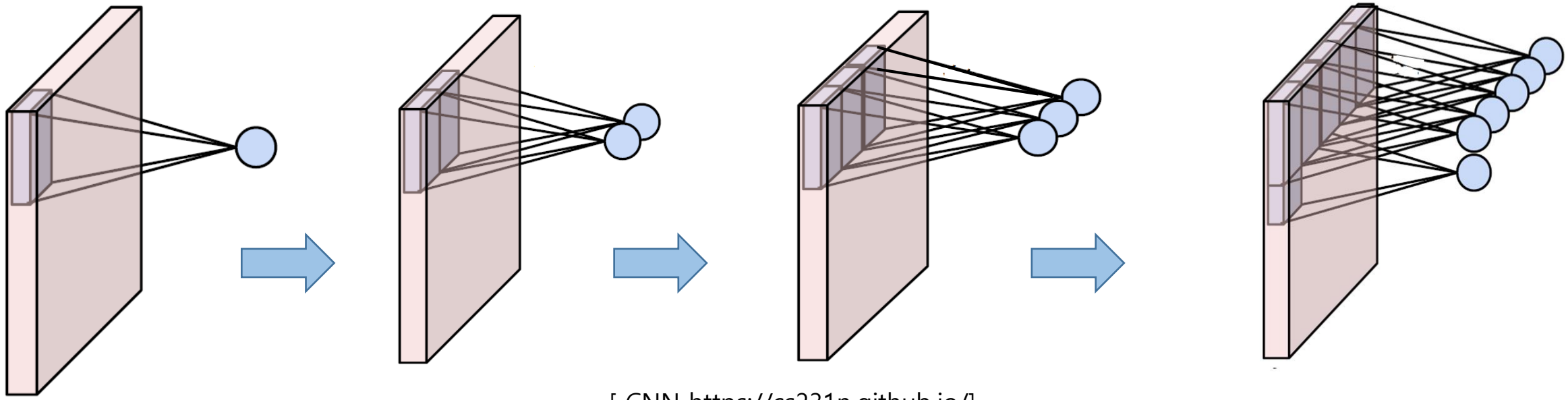
[영상 인식]



[합성곱 신경망 (Stanford cse231)]

[영상 변환]

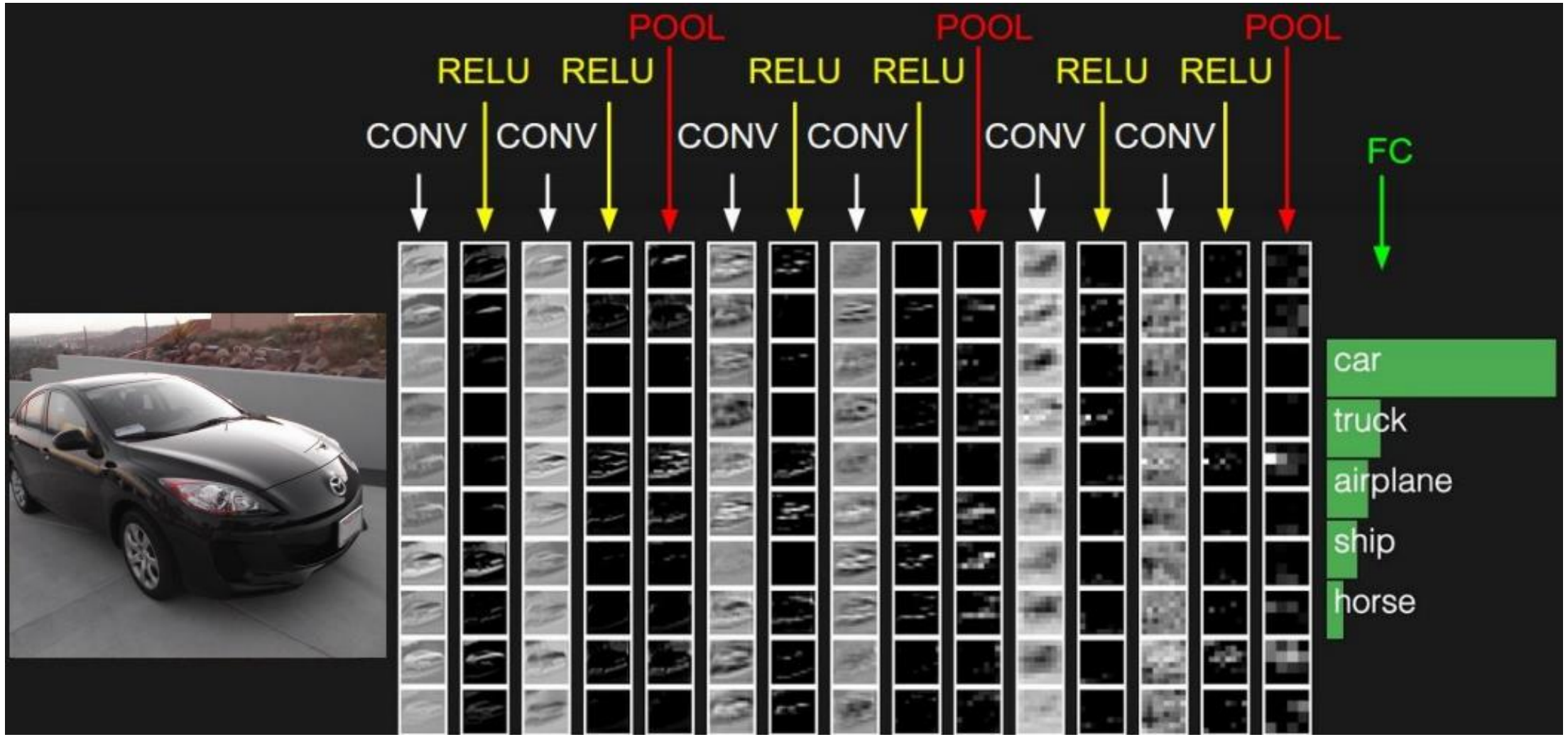
컨볼루션 신경망 (CNN: Convolutional Neural Network)



[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

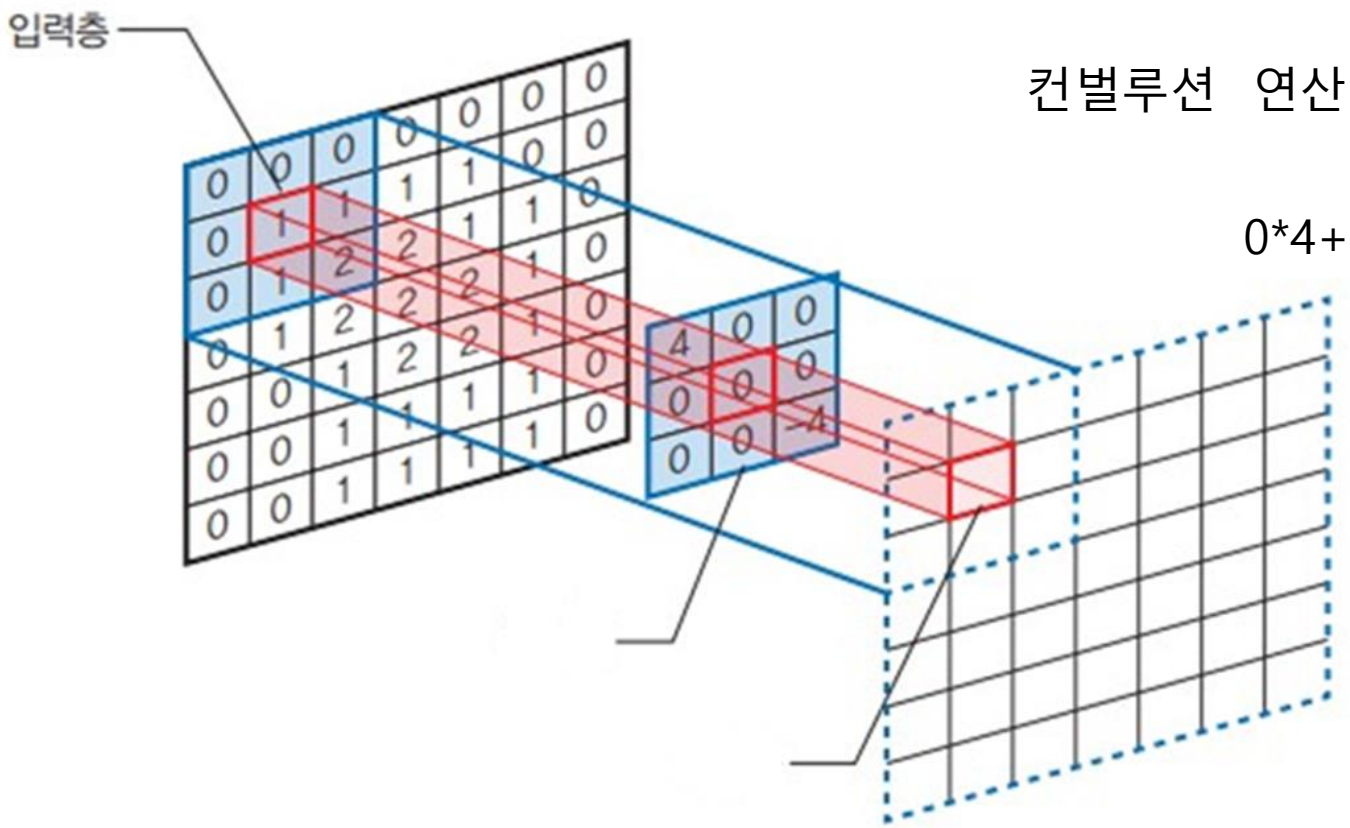
커널

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

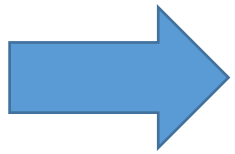
참조 딥러닝 익스프레스

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

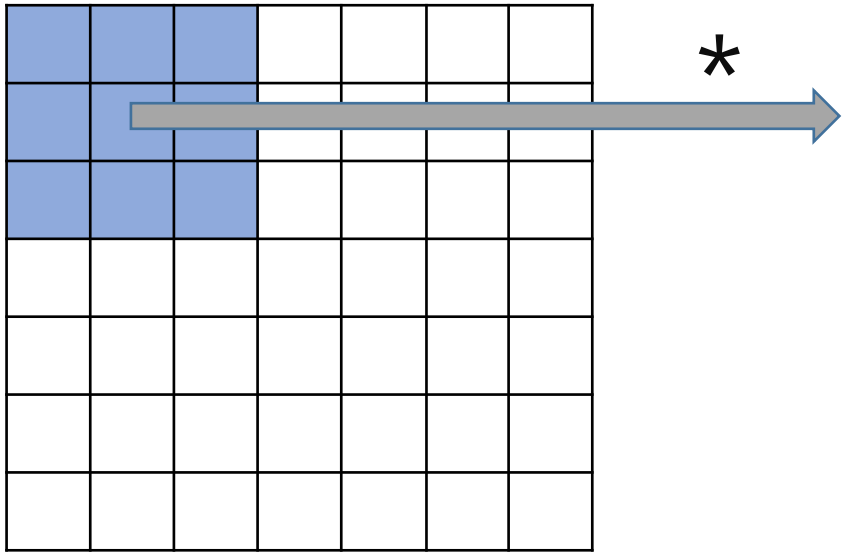


47	8	42
41	12	38
43	14	46

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

컨볼루션 (Convolution)

7x7 input image



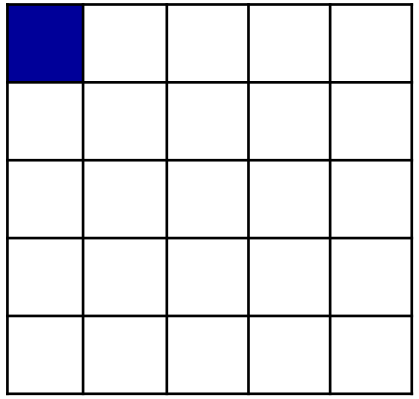
*

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

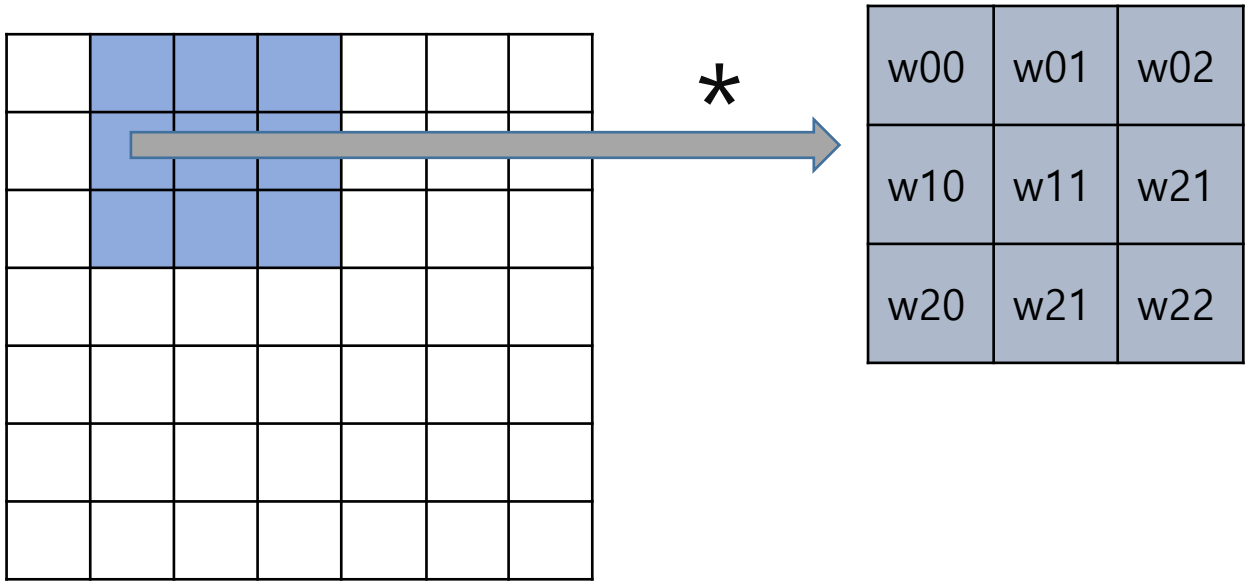


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

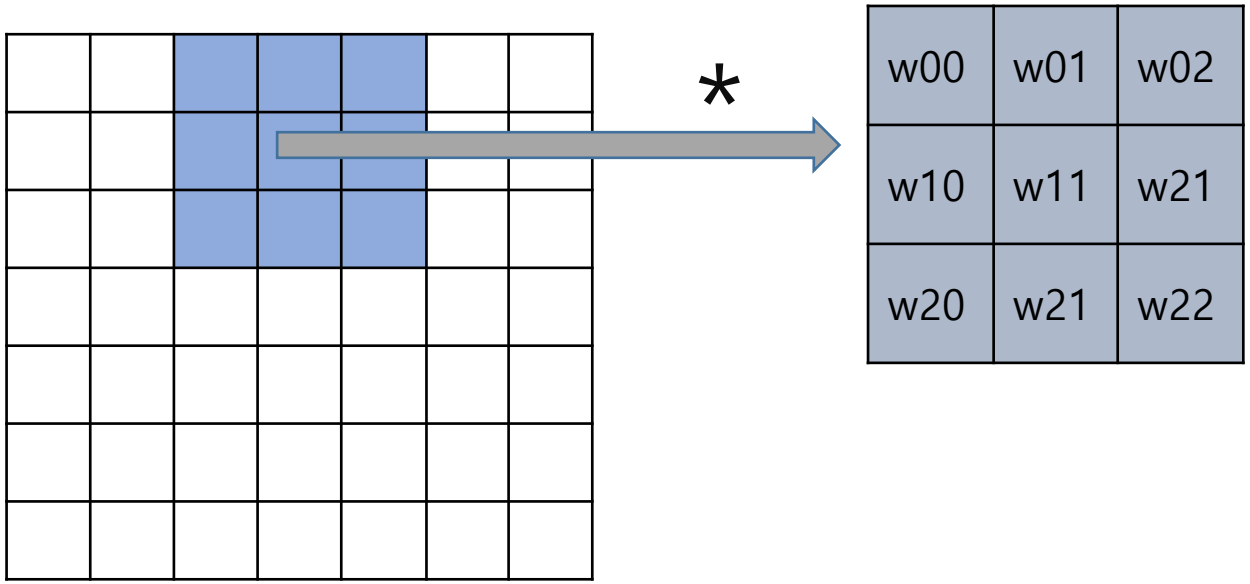


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

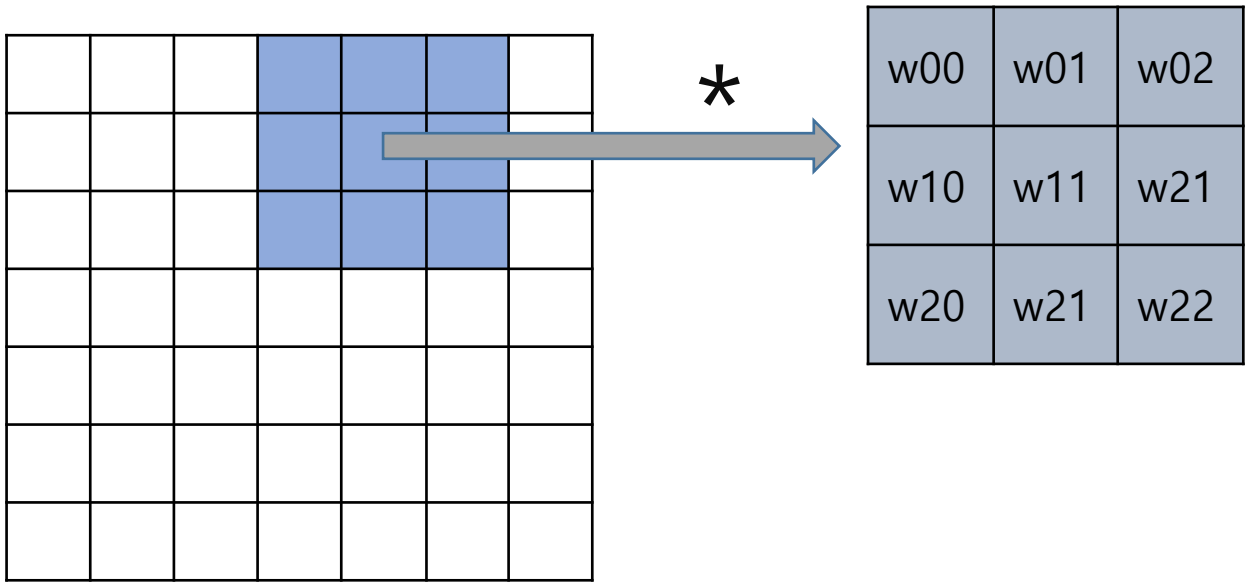


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

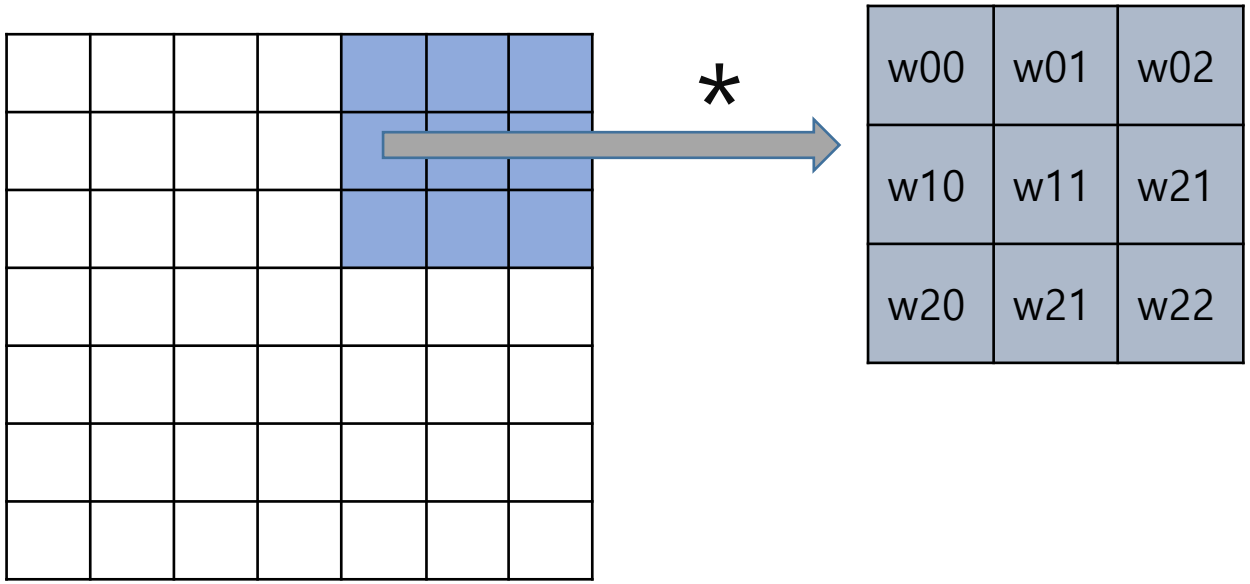


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

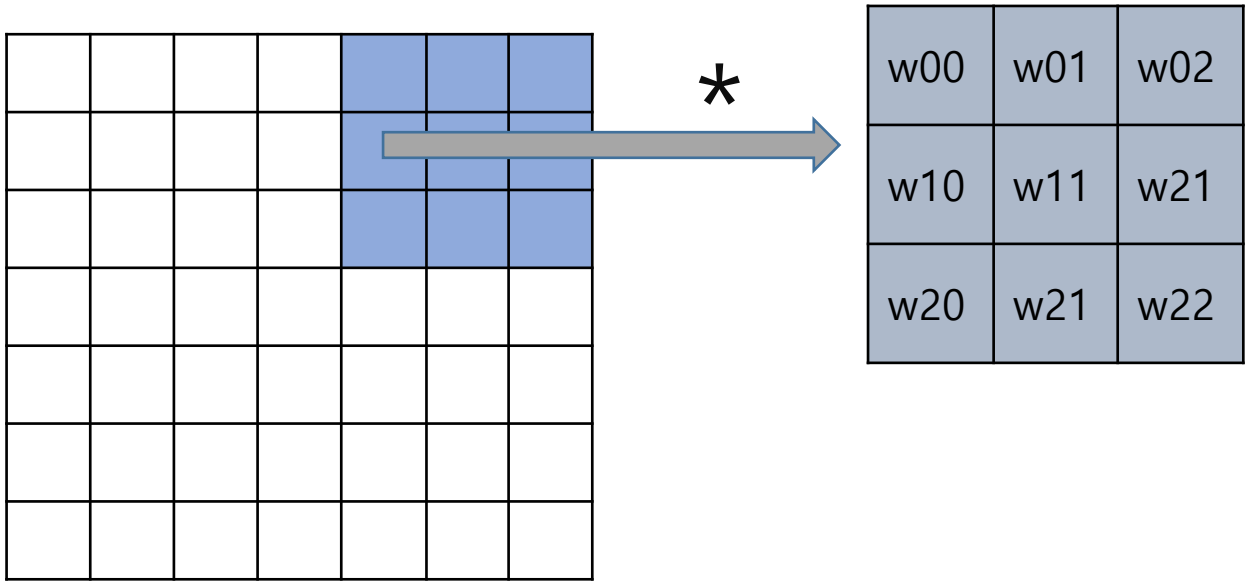


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

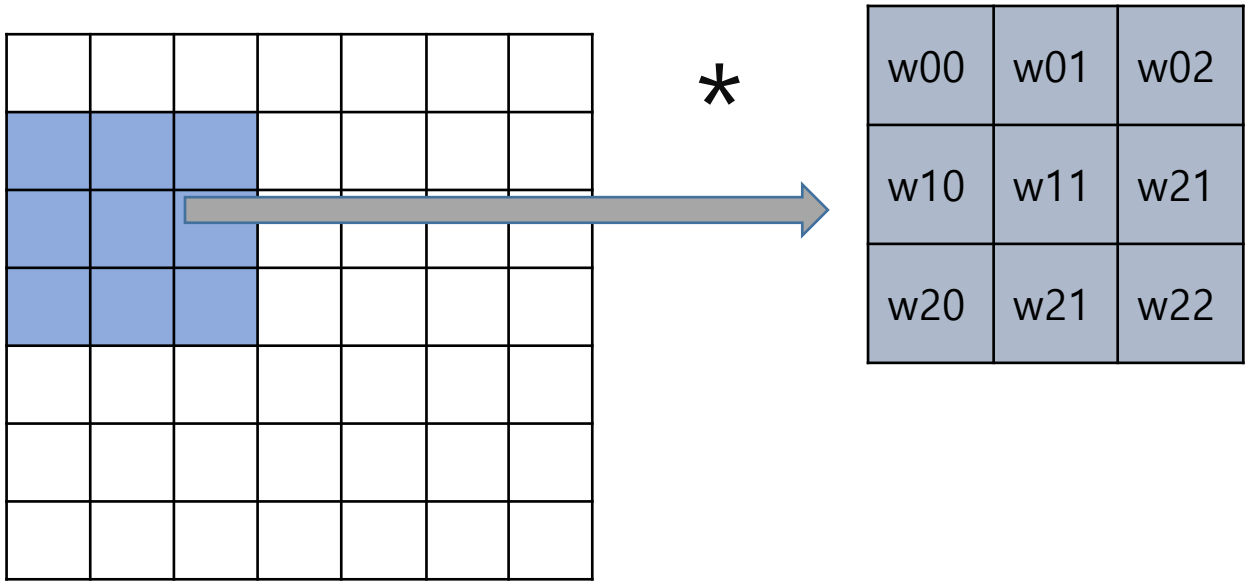


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image

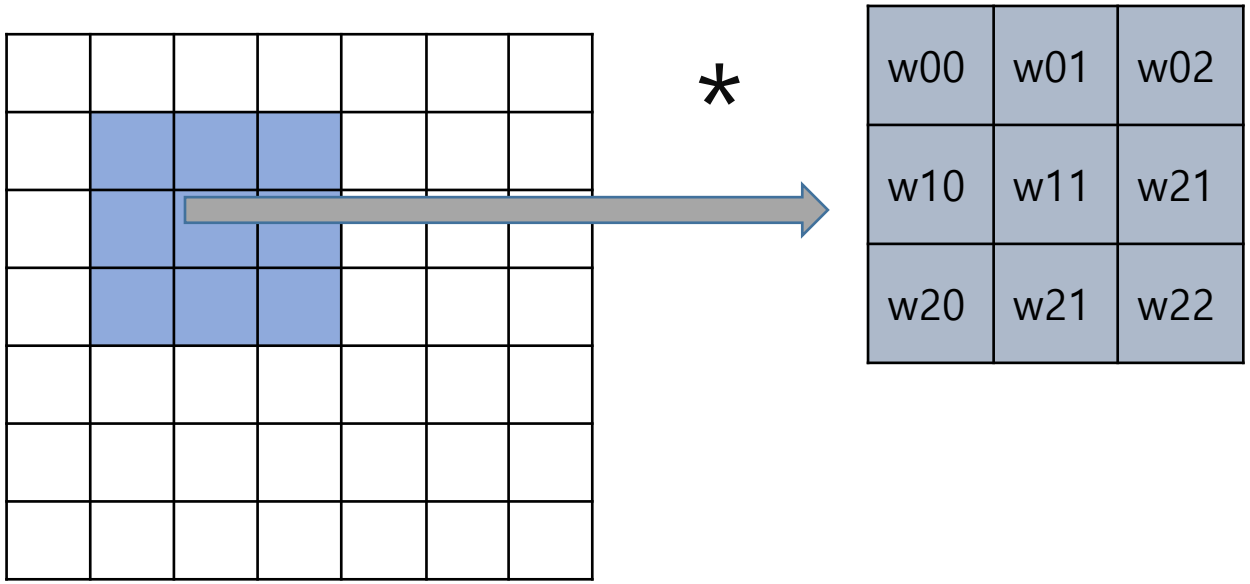


컨볼루션 신경망 (CNN: Convolutional Neural Network)

7x7 input image

3x3 kernel

output image



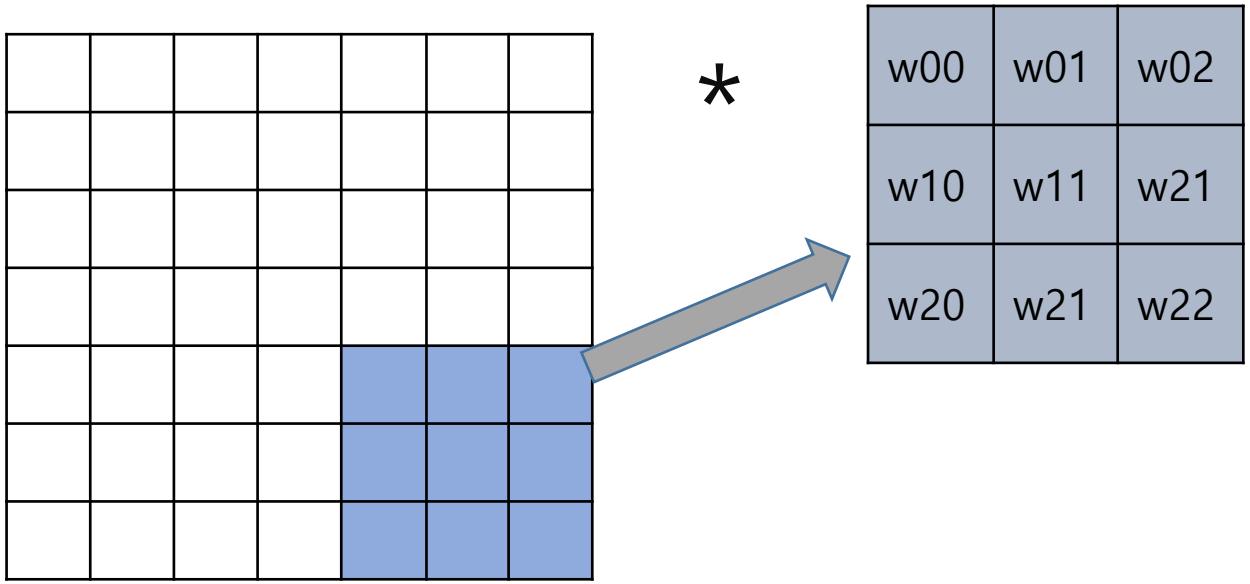
컨볼루션 신경망 (CNN: Convolutional Neural Network)

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)

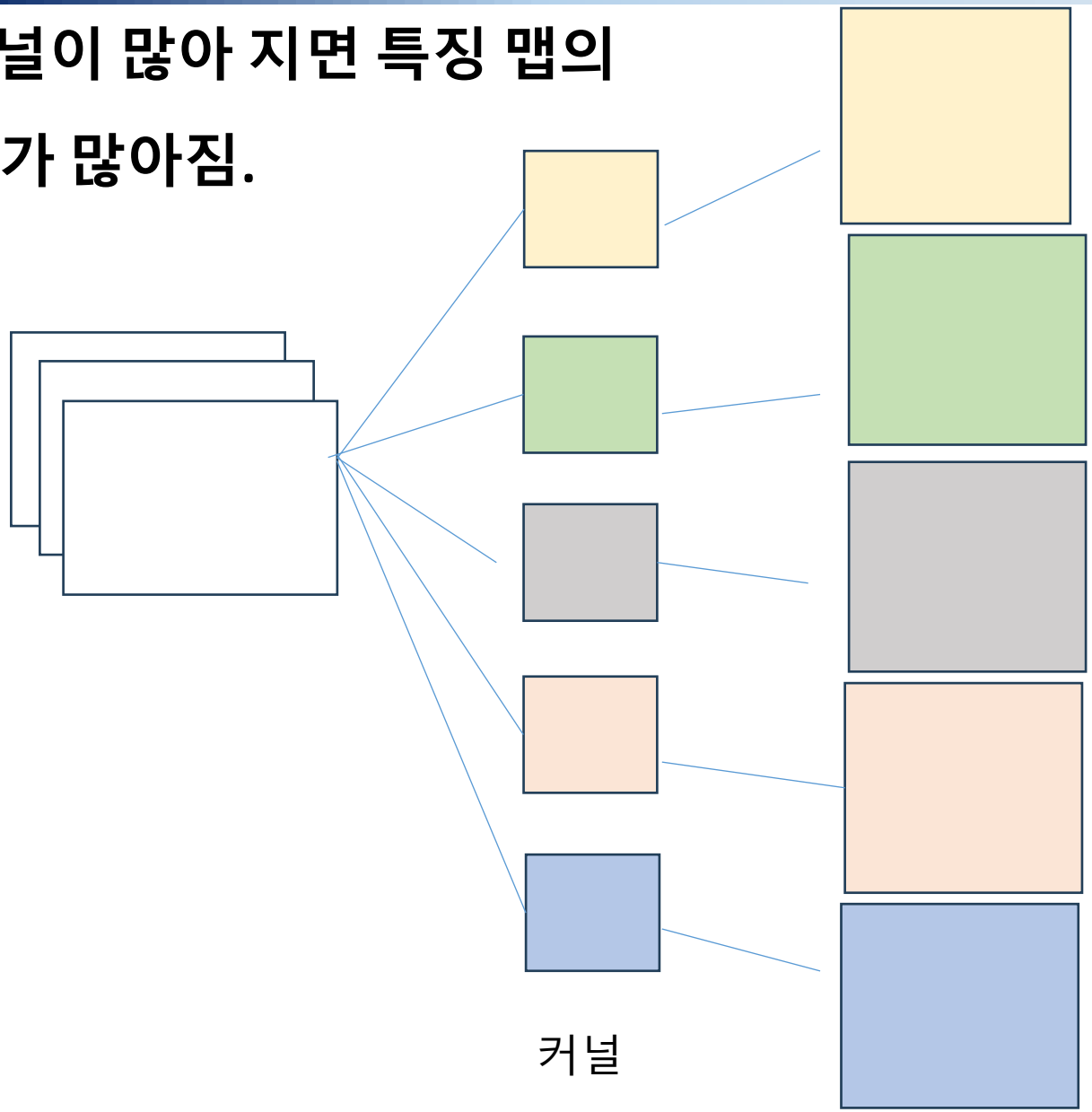
- 보폭(stride) 은 커널을 적용하는 거리이다. 보폭이 1이면 커널을 한 번에 1픽셀씩 이동하면서 커널을 적용하는 것이다. 보폭이 2라는 것은 하나씩 건너뛰면서 커널을 적용

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.

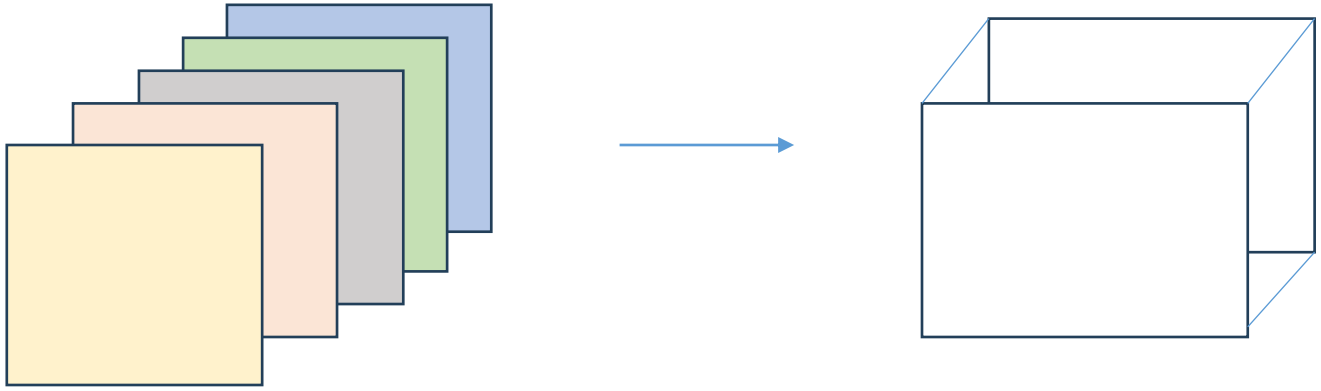
컨볼루션 신경망 (CNN: Convolutional Neural Network)

- 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)

인하공전 컴퓨터 정보과

```
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)

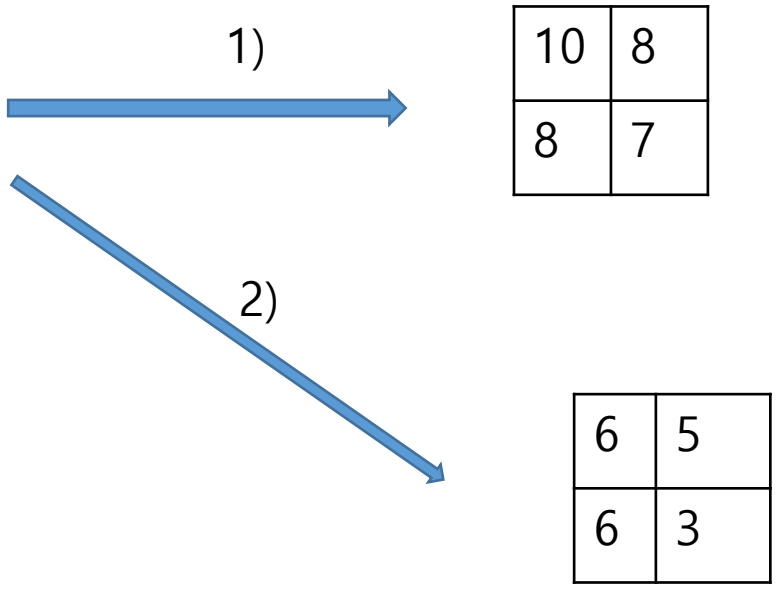
인하공전 컴퓨터 정보과

- 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



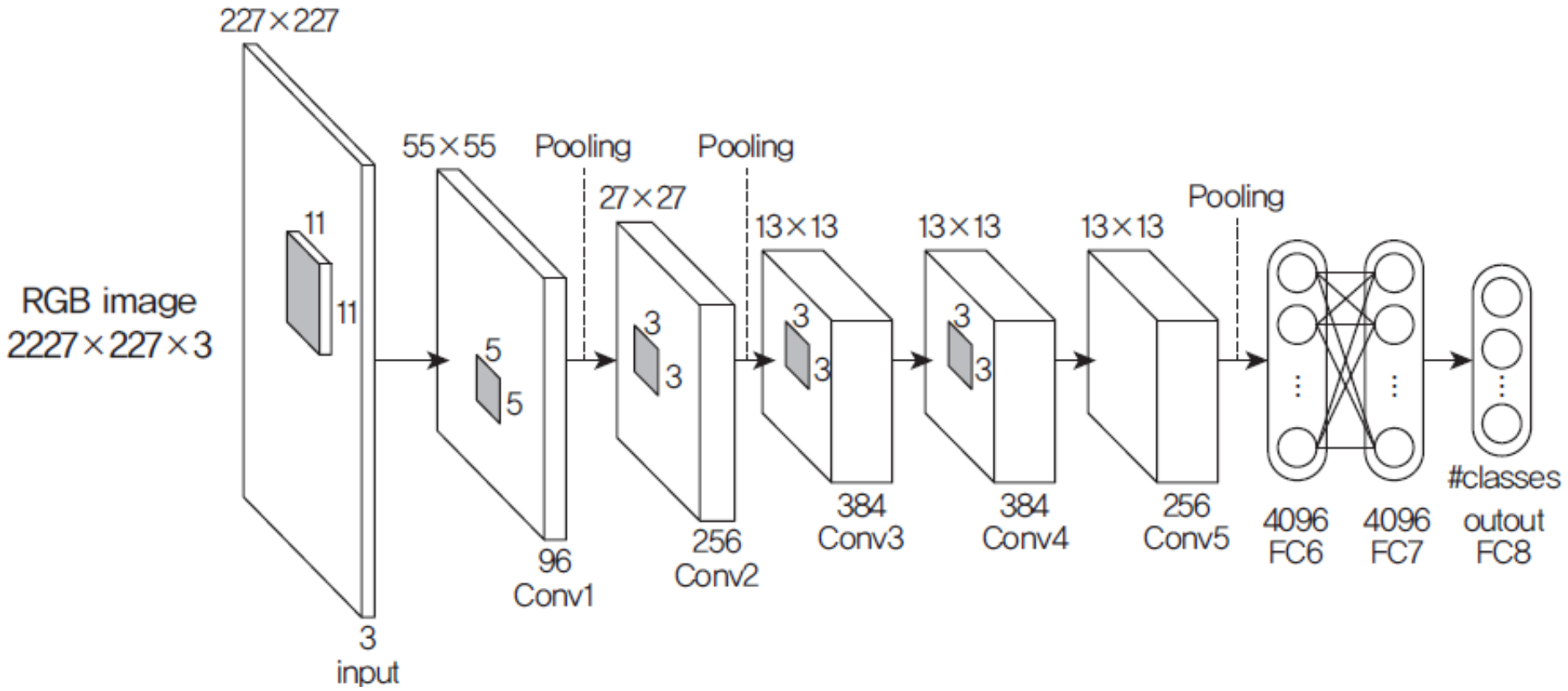
10	8
8	7

6	5
6	3

풀링의 장점

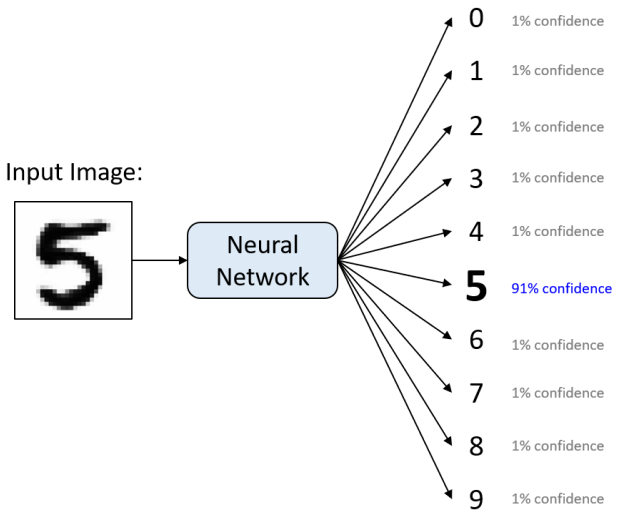
- 레이어의 크기가 작아짐.
- 계산이 빨라짐.
- 정보가 압축됨.

컨볼루션 신경망 (CNN: Convolutional Neural Network)

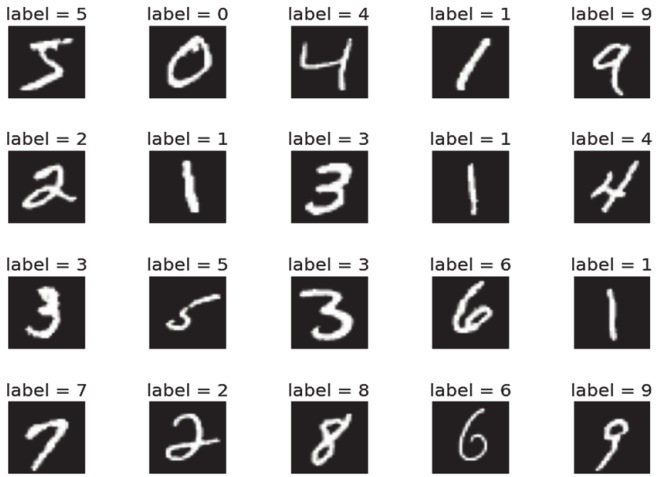


MNIST 손 글씨 이미지 분류

- **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)
```

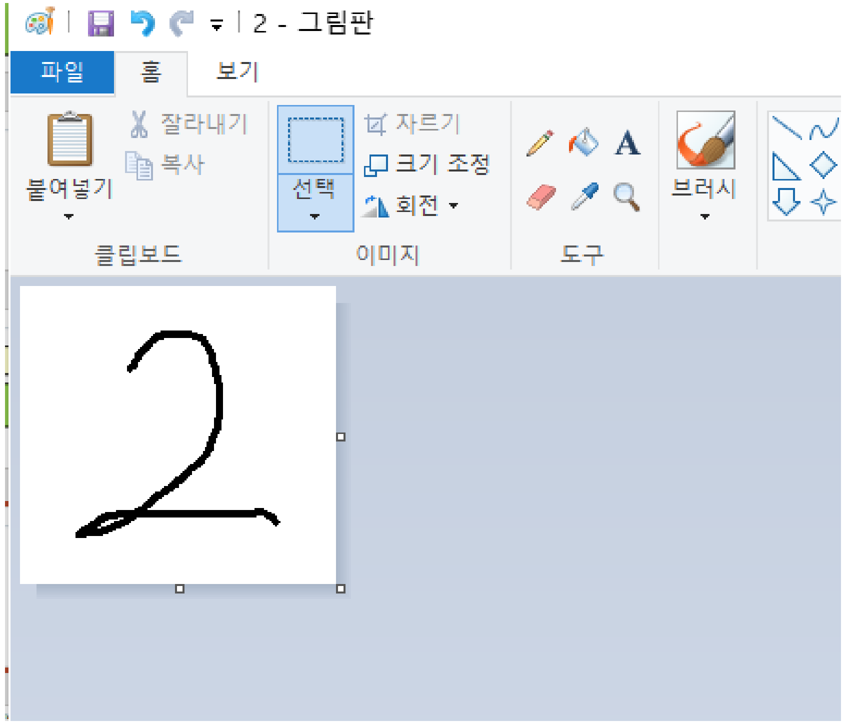
```
Epoch 1/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.1414 -
accuracy: 0.9560
...
Epoch 5/5
1875/1875 [==============================] - 14s 7ms/step - loss: 0.0194 -
accuracy: 0.9940
```

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

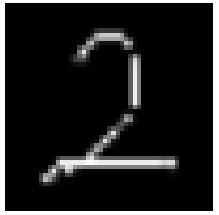


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	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  (None, 7, 7, 64)          0
2D)

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	

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

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

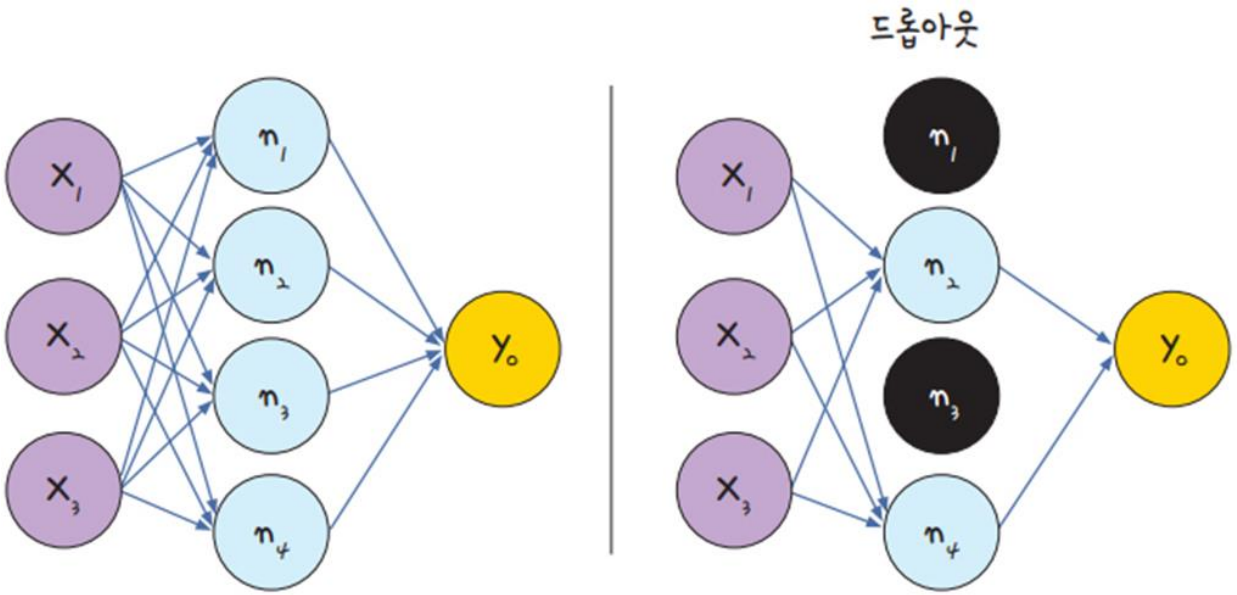
```
input_shape = (28, 28, 1)



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

인하공전 컴퓨터 정보공학과

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())
```

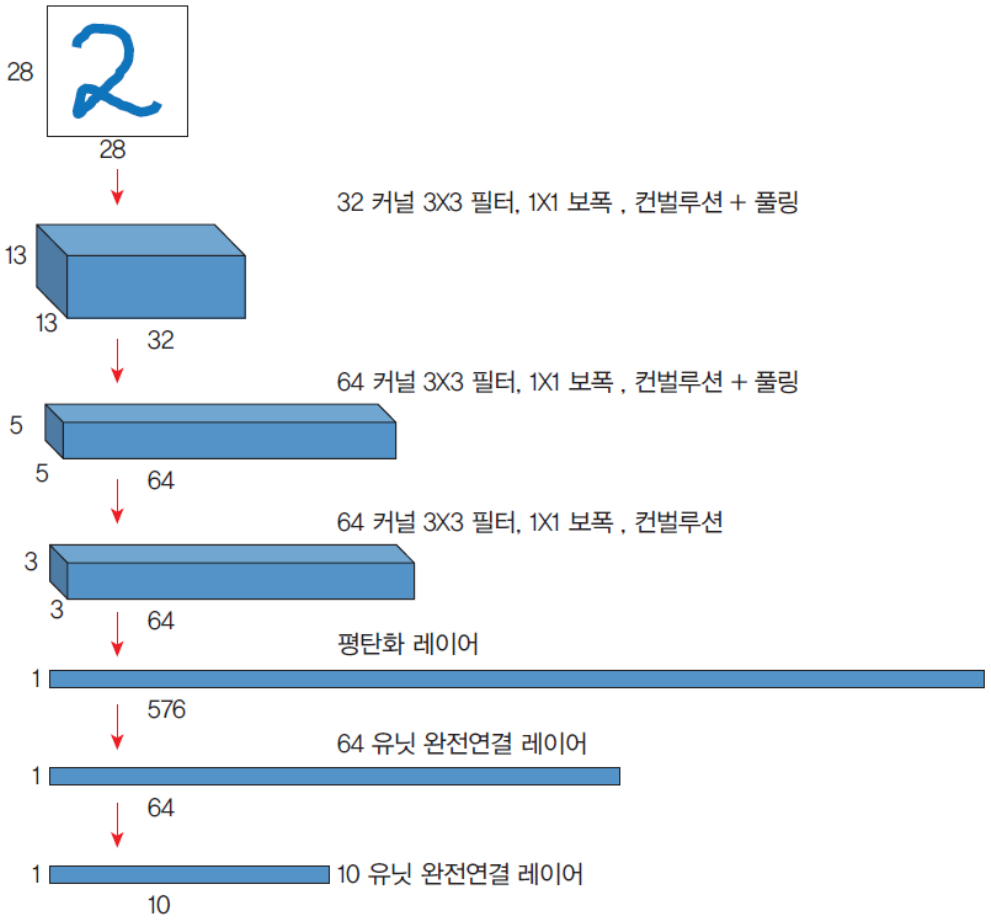
CNN output shape

- **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)

인하공전 컴퓨터 정보공학과




```
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)

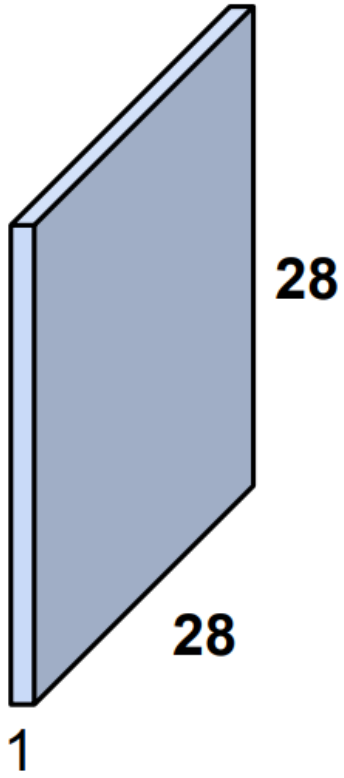
인하공전 컴퓨터 정보공학과

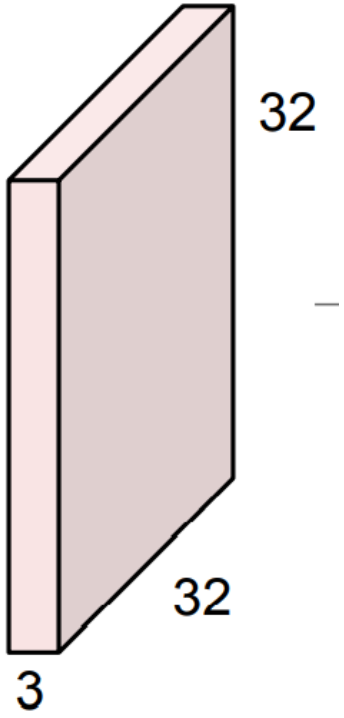
Input size = (28,28,1)

input depth

Input size = (32,32,3)

- **Output Shape**





Convolution Neural Network (CNN)

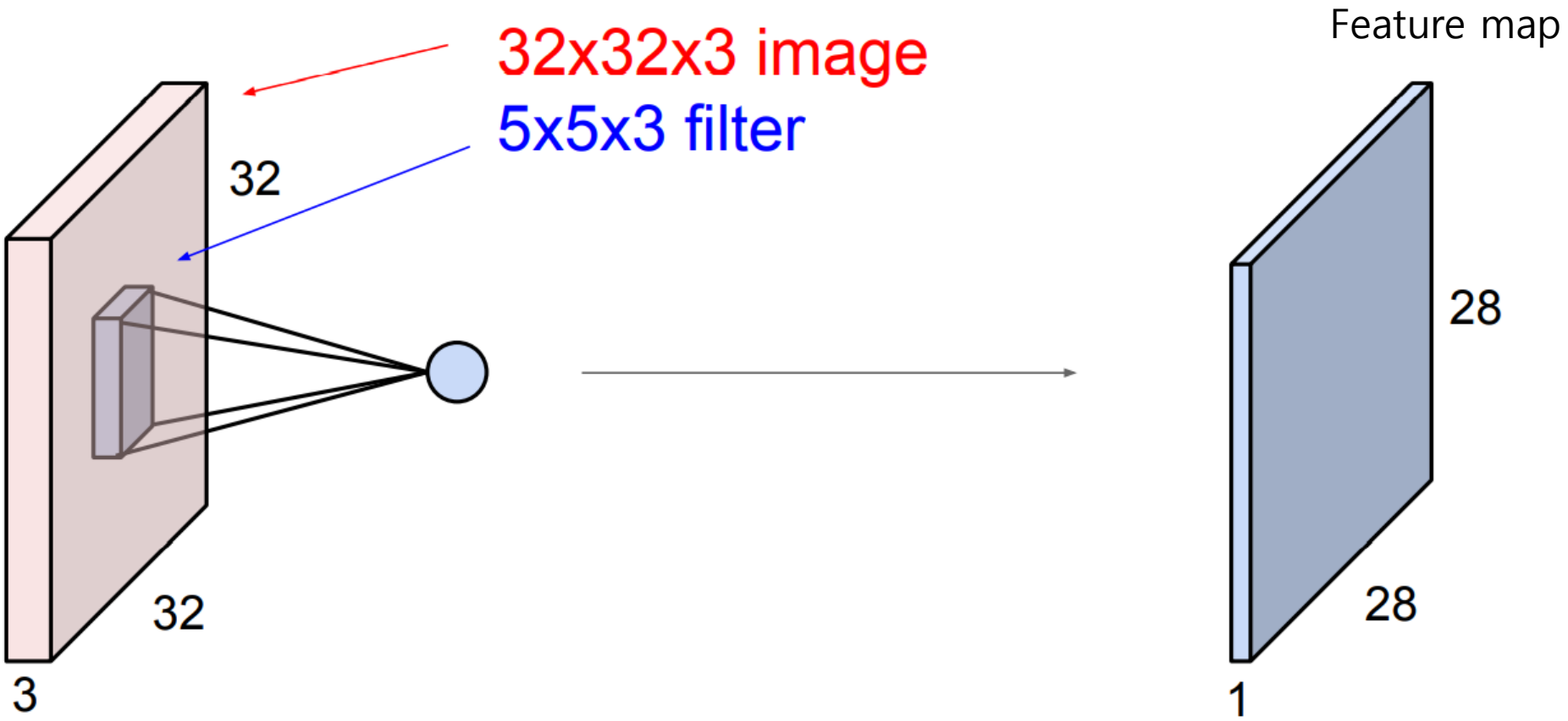
- **커널(kernel)=필터(filter)**
- **필터의 수 = 채널 (channel)**

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

Convolution Neural Network (CNN)

인하공전 컴퓨터 정보공학과

- **Output Shape**

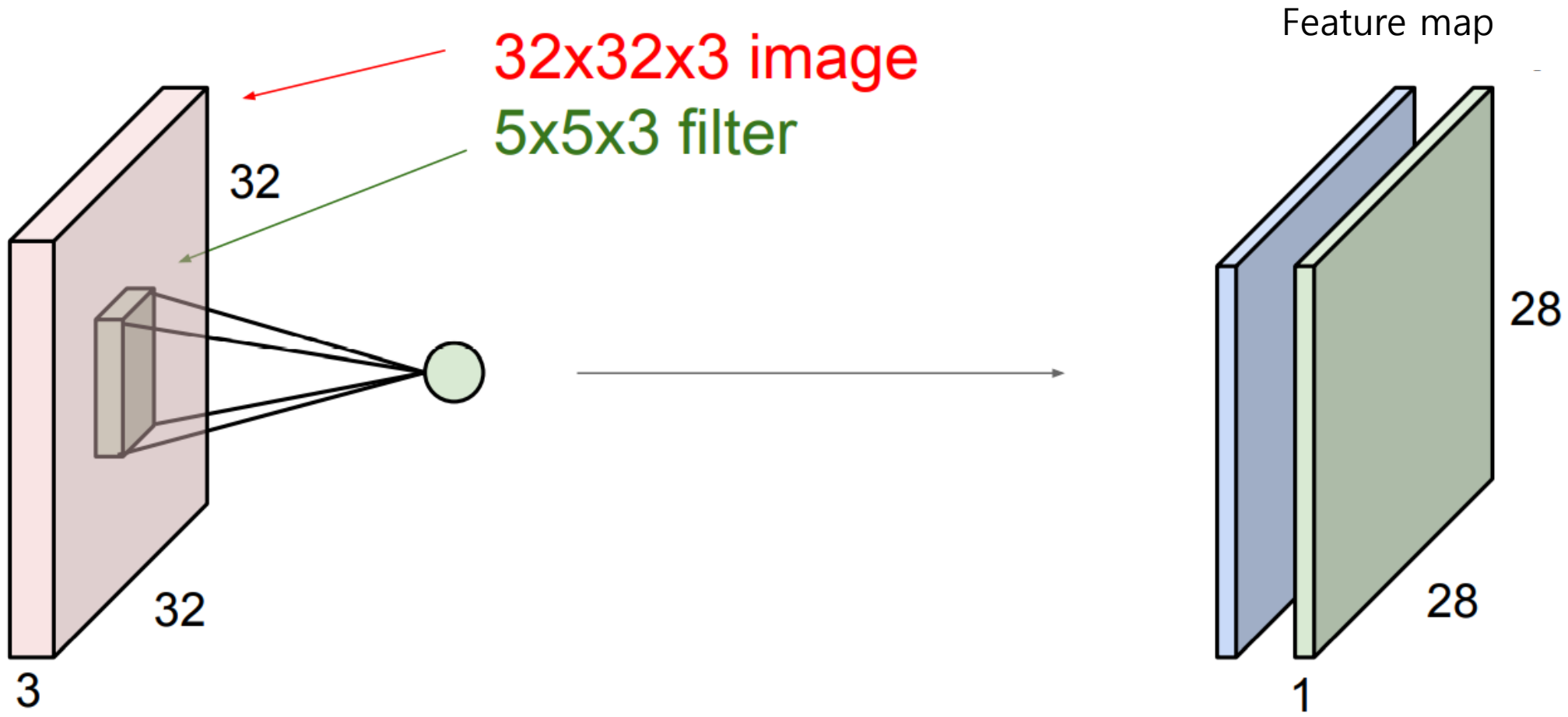


출력 사이즈= (입력 사이즈 – 커널 사이즈)+1=32-5+1=28

Convolution Neural Network (CNN)

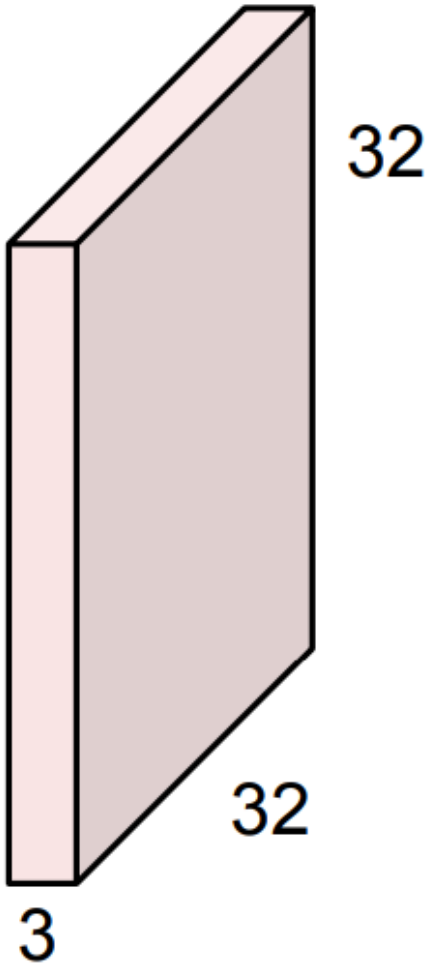
인하공전 컴퓨터 정보공학과

- **Output Shape**



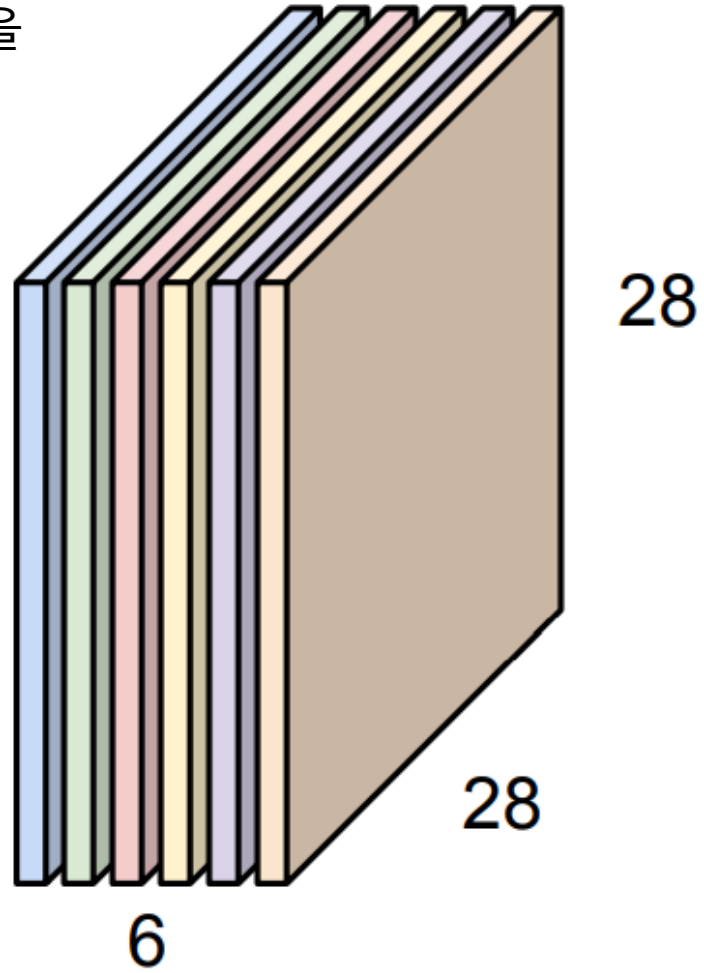
Convolution Neural Network (CNN)

인하공전 컴퓨터 정보과



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

인하공전 컴퓨터 정보공학과

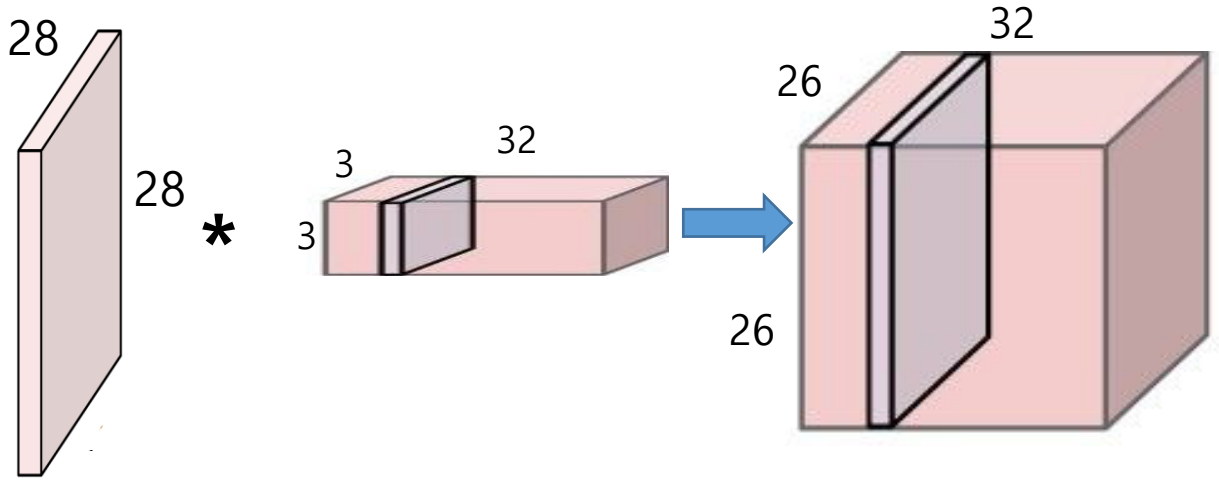
```
from tensorflow import keras
from tensorflow.keras import layers
```

```
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



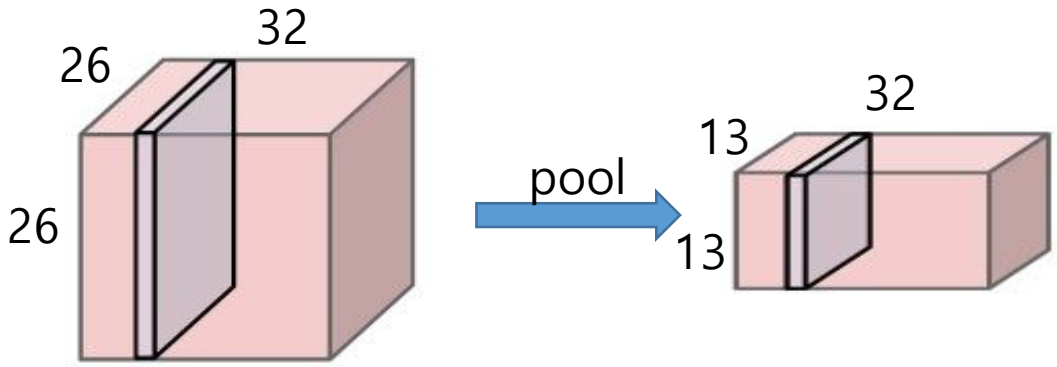
MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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
```

Convolution Neural Network (CNN) (과제1)

인하공전 컴퓨터 정보공학과

- **모델 디자인 (mnist_paratest.ipynb)** **1x1 kernel(filter)은 사용할 수 없음.**

예제 code를 이용하여 model.summar가 다음과 같이 출력되도록 code를 수정하고 제출

code

*.py upload

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 26, 26, 16)	160
max_pooling2d_2 (MaxPooling 2D)	(None, 13, 13, 16)	0
conv2d_4 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_3 (MaxPooling 2D)	(None, 6, 6, 32)	0
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)(과제2)

- **모델 디자인 (mnist_paratest.ipynb)** **1x1 kernel(filter)은 사용할 수 없음.**

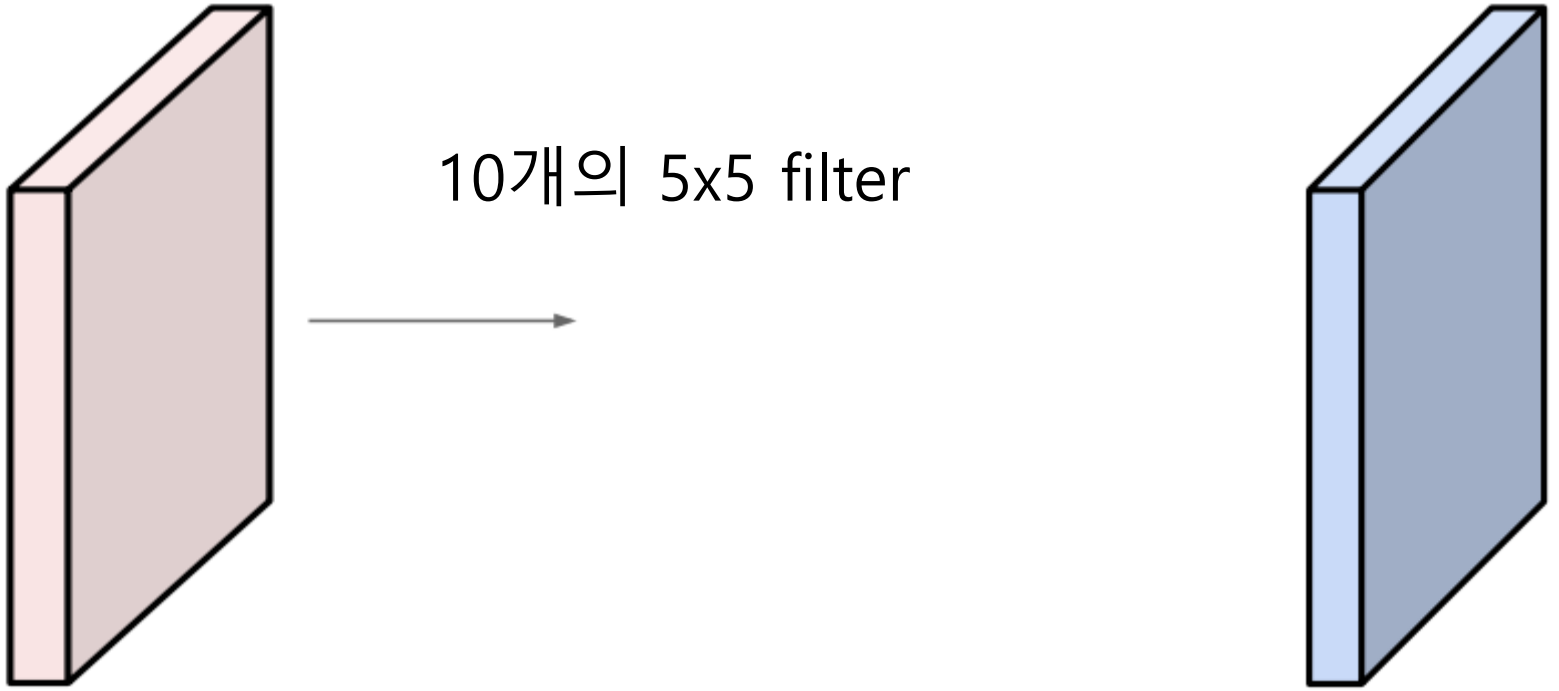
예제 code를 이용하여 model.summar가 다음과 같이 출력되도록 code를 수정하고 제출

code

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

*.py upload

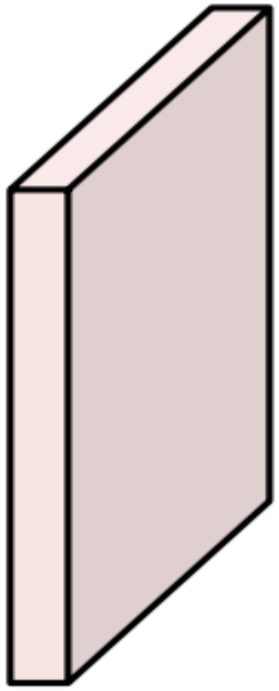
Convolution Layer Parameter



32x32x**3**

Filter는 5x5x**3**

Convolution Layer Parameter



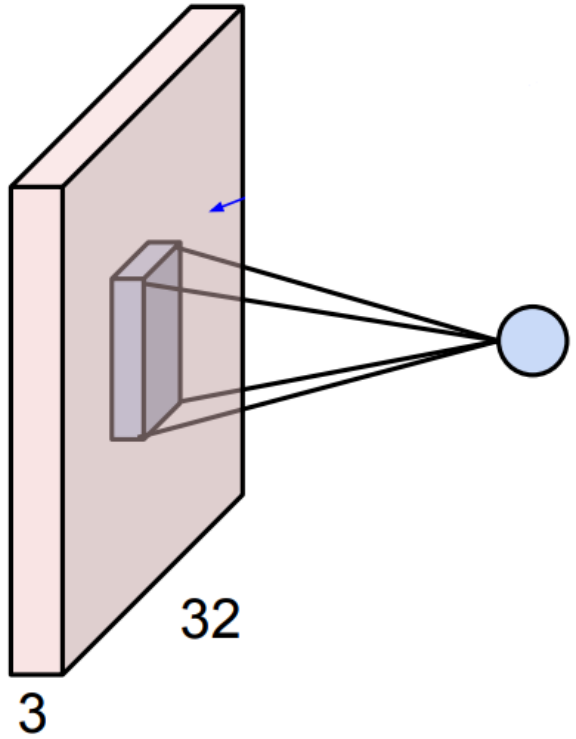
32x32x3

Filter 하나당 bias 하나

10개의 5x5 filter

Filter는 5x5x3 이 됨.

W=5*5*3
B=1



Filter 하나당 parameter 5x5x3+1

Filter 가 10개

10*(5x5x3+1) = 760

채널이 10개

이 layer의 parameter 수는 760

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
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

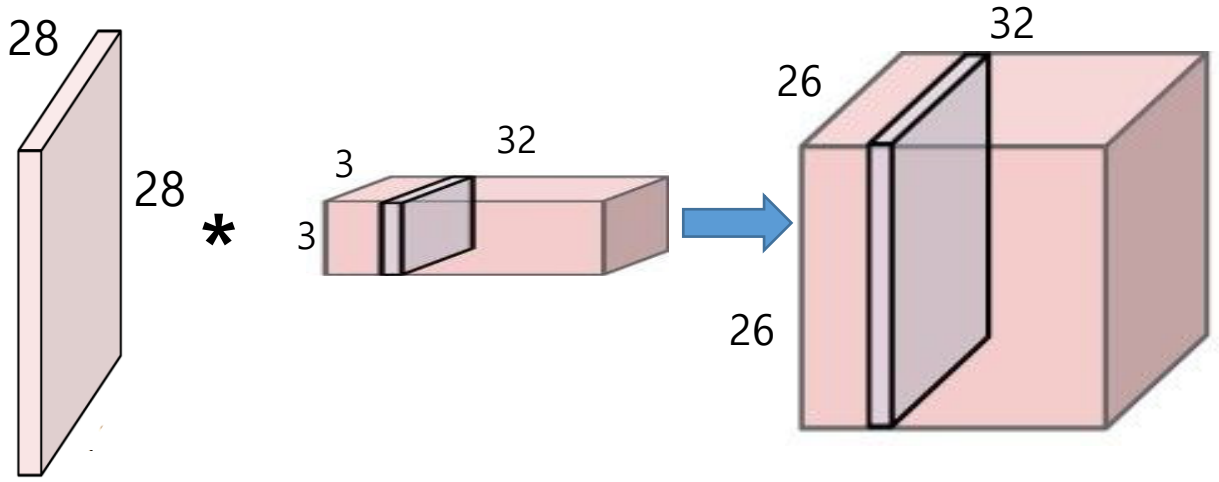
인하공전 컴퓨터 정보공학과

```
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 parameters0
=채널*(input depth*kernel size*kernel size+1
=>32*(1*3*3+1)=320

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



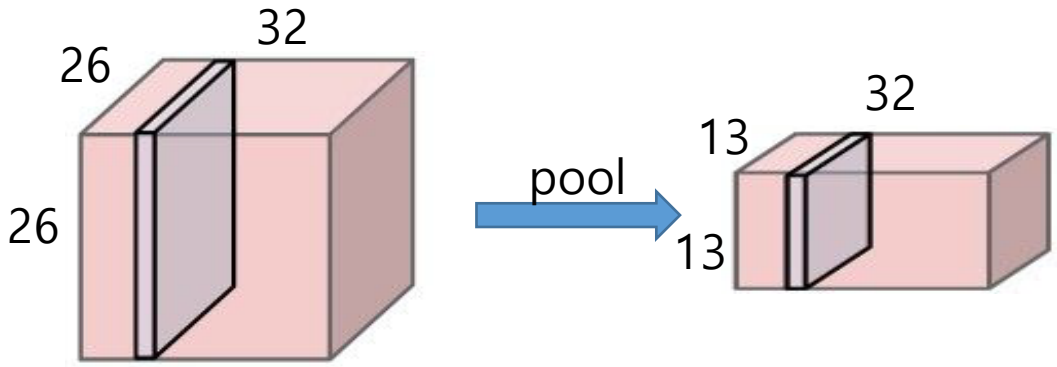
MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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
=채널*(input depth*kernel size*kernel size+1)
=>64*(32*3x3+1)=18496

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

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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

인하공전 컴퓨터 정보공학과

```
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
=앞 layer node수*현재 layer node 수+현재 layer node수
=>1600*10+10=16010

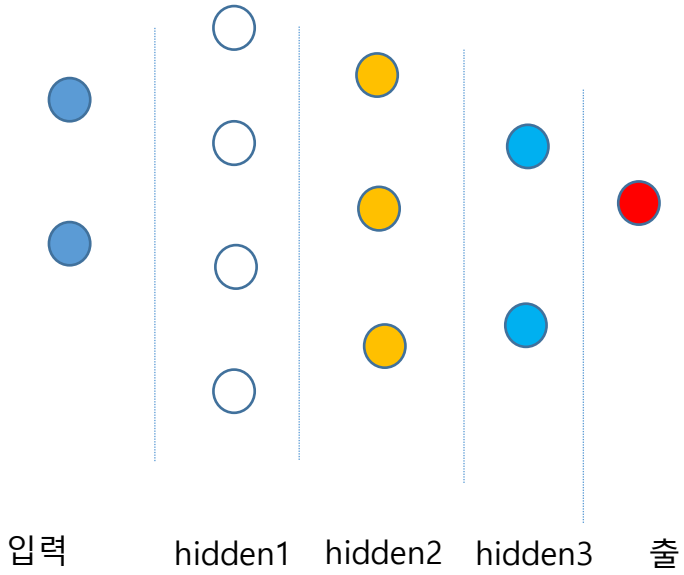
Input=1600, => output size=10

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(4, activation='relu', input_shape=(2,)))
model.add(tf.keras.layers.Dense(3, activation='relu'))
model.add(tf.keras.layers.Dense(2, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

입력 층 : 2
은닉층 1 : 4 (relu)
은닉층 2 : 3 (relu)
은닉층 3 : 2 (relu)
출력 층 : 1

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 4)	12
dense_5 (Dense)	(None, 3)	15
dense_6 (Dense)	(None, 2)	8
dense_7 (Dense)	(None, 1)	3

total params: 38
Trainable params: 38
Non-trainable params: 0



Parameter 수 = 앞 layer node수* 현재 layer node수 + 현재 layer node 수

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

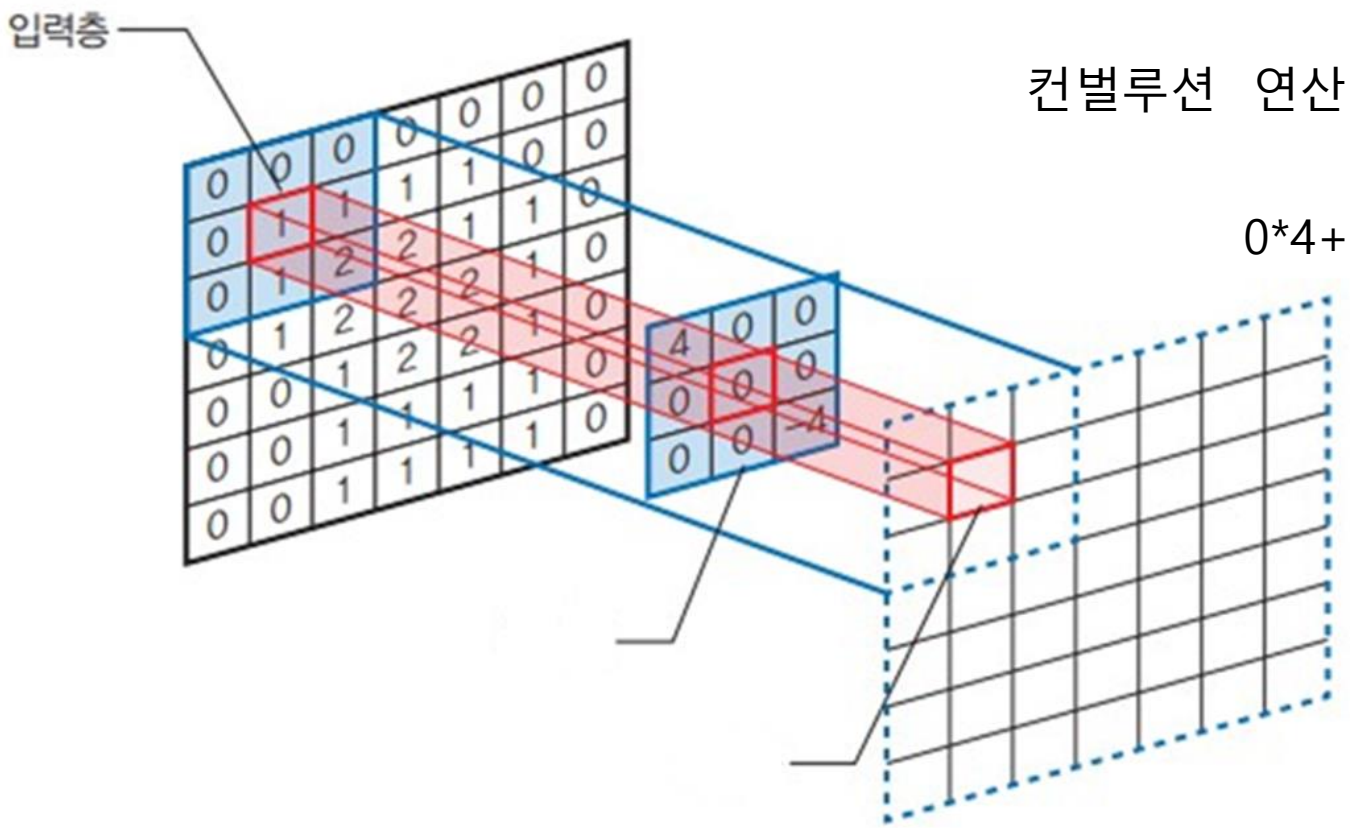
인하공전 컴퓨터 정보공학과

```
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
```

컨볼루션 (Convolution) 계산



컨벌루션 연산 수행 결과

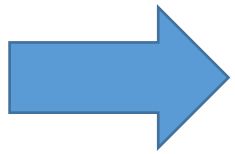
0*4+0*0+0*0+1*0+1*0+0*0+1*0+2*-4= -8

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



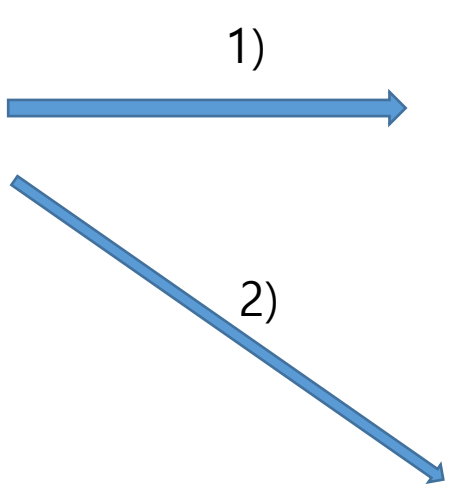
47	8	42
41	12	38
43	14	46

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

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
6	3

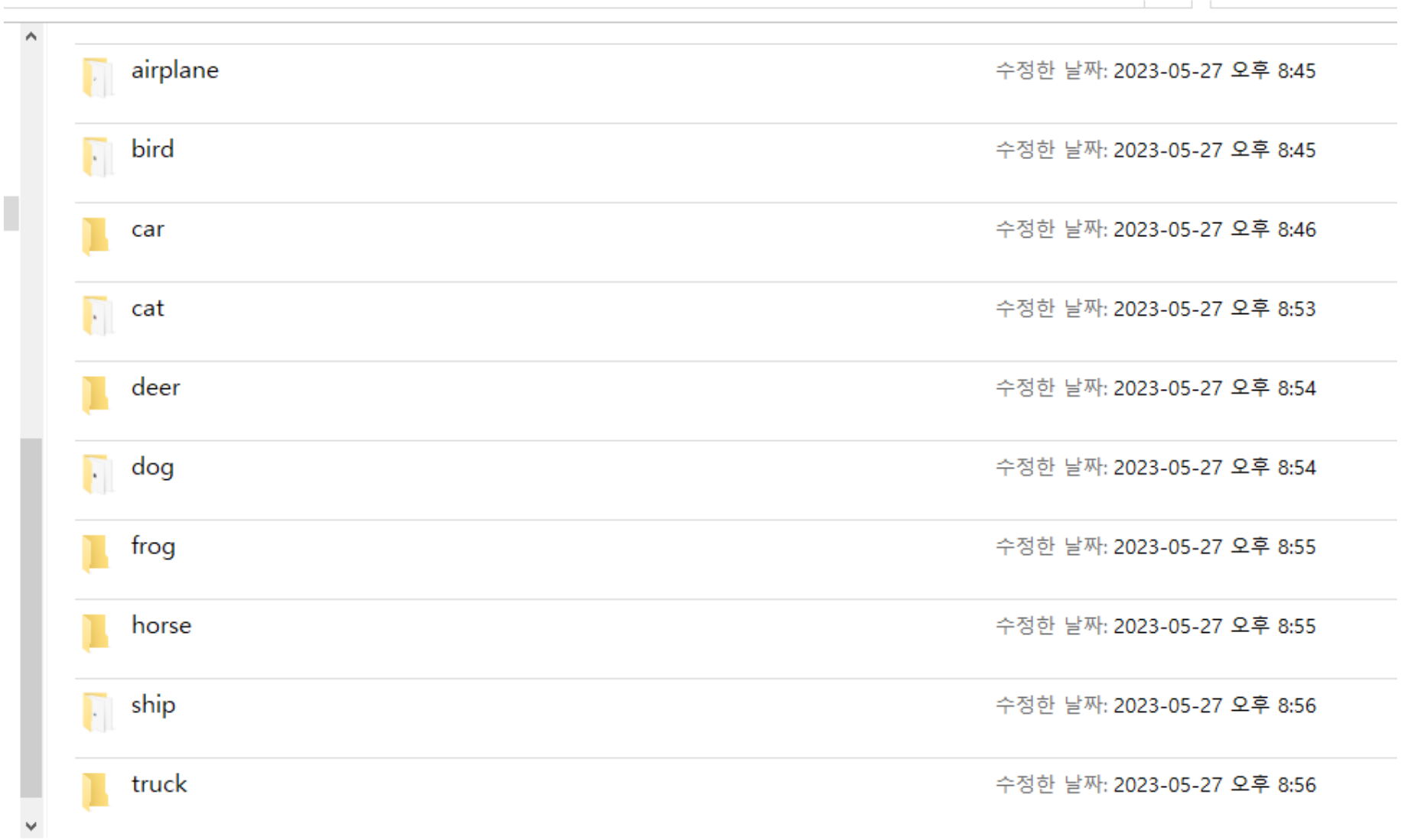
컨볼루션 (Convolution) 계산

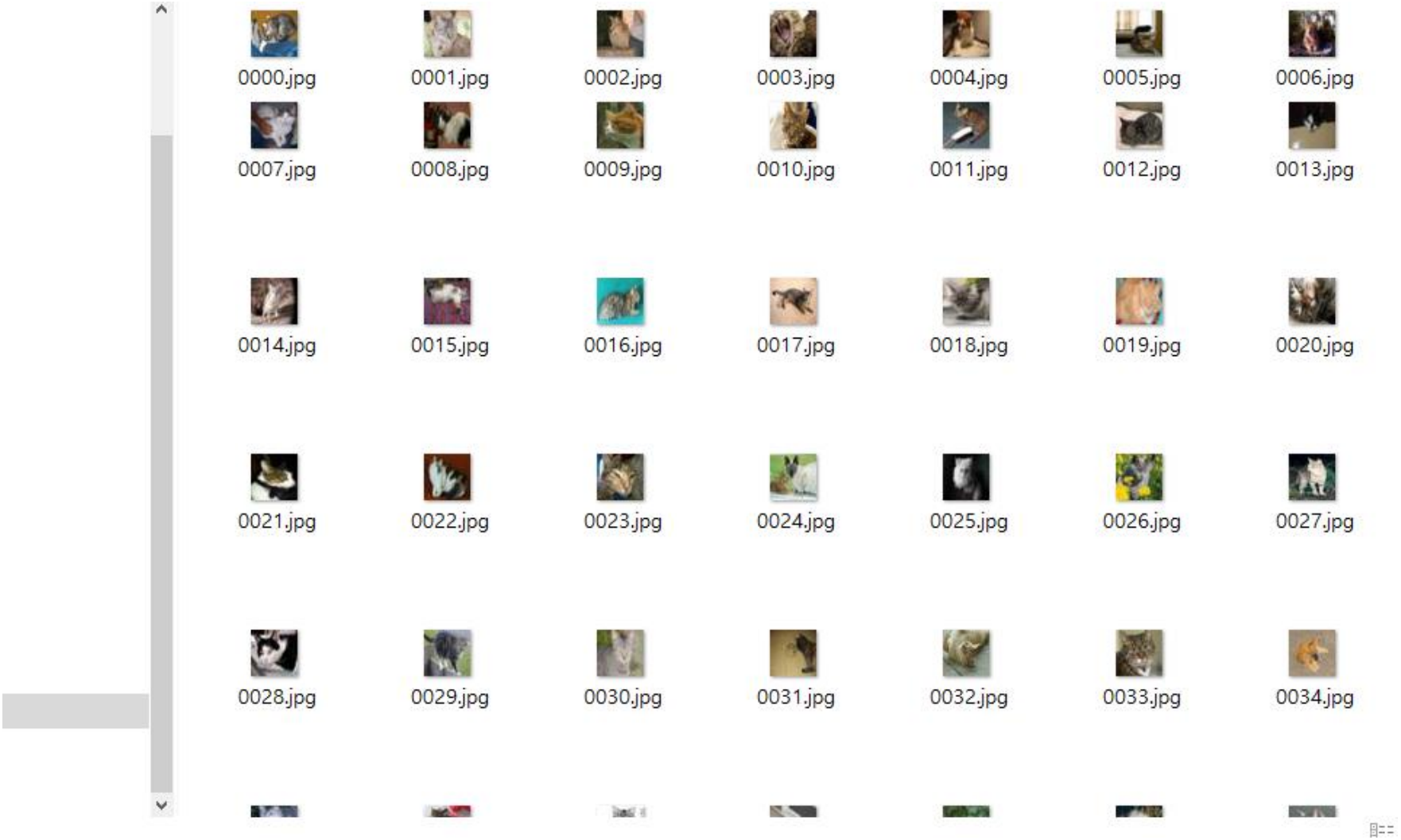
8x8 컬러 영상에 5개의 필터와 'same' padding으로 컨벌루션 연산을 수행 하고 (2,2) 풀링을 통과한 특성 맵의 크기는?

(8,8,3) => (8,8,5)=> (4,4,5)

영상 분류 예제

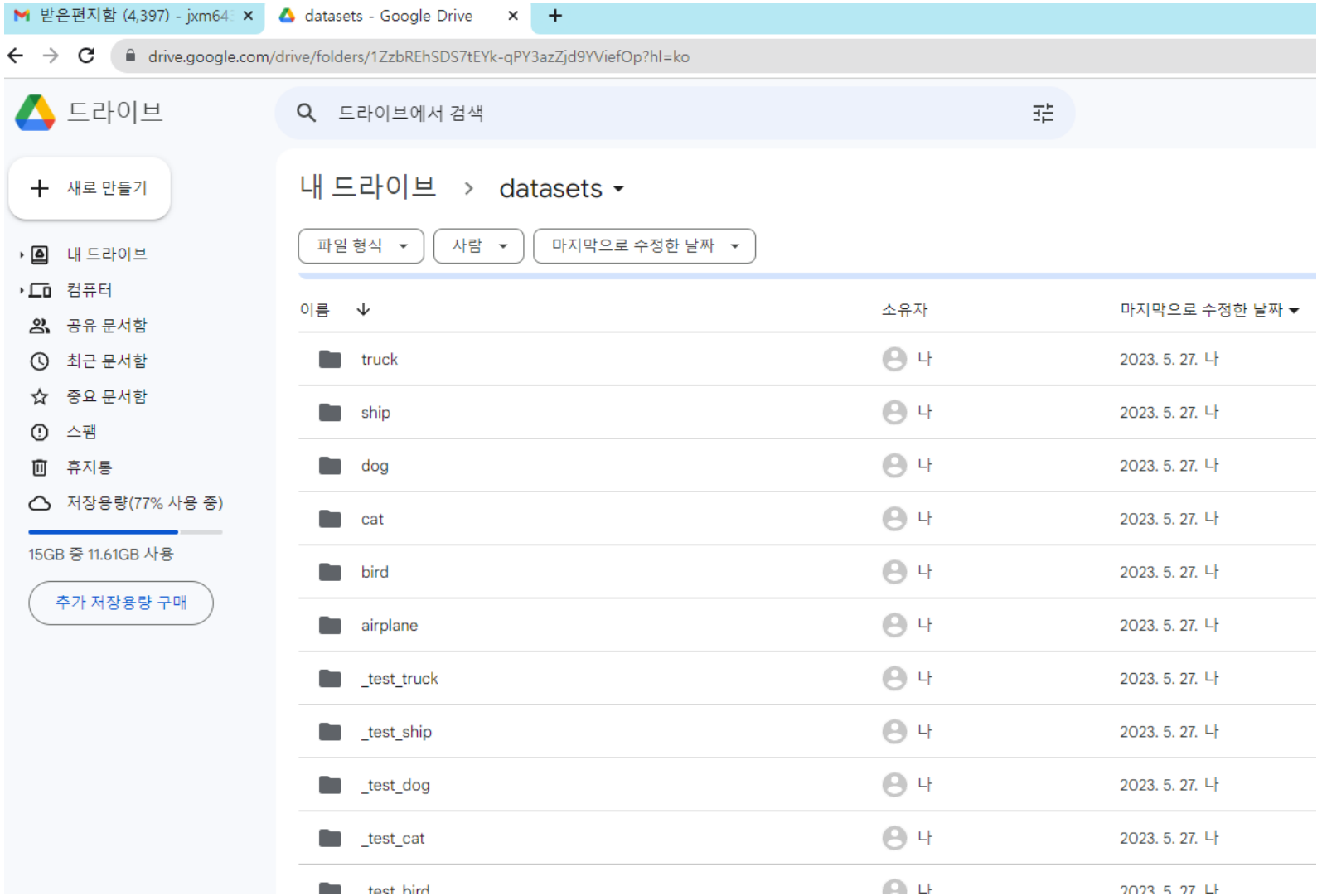
- JPG IMAGE 이용




0000.jpg
0001.jpg
0002.jpg
0003.jpg
0004.jpg
0005.jpg
0006.jpg
0007.jpg
0008.jpg
0009.jpg
0010.jpg
0011.jpg
0012.jpg
0013.jpg
0014.jpg
0015.jpg
0016.jpg
0017.jpg
0018.jpg
0019.jpg
0020.jpg
0021.jpg
0022.jpg
0023.jpg
0024.jpg
0025.jpg
0026.jpg
0027.jpg
0028.jpg
0029.jpg
0030.jpg
0031.jpg
0032.jpg
0033.jpg
0034.jpg

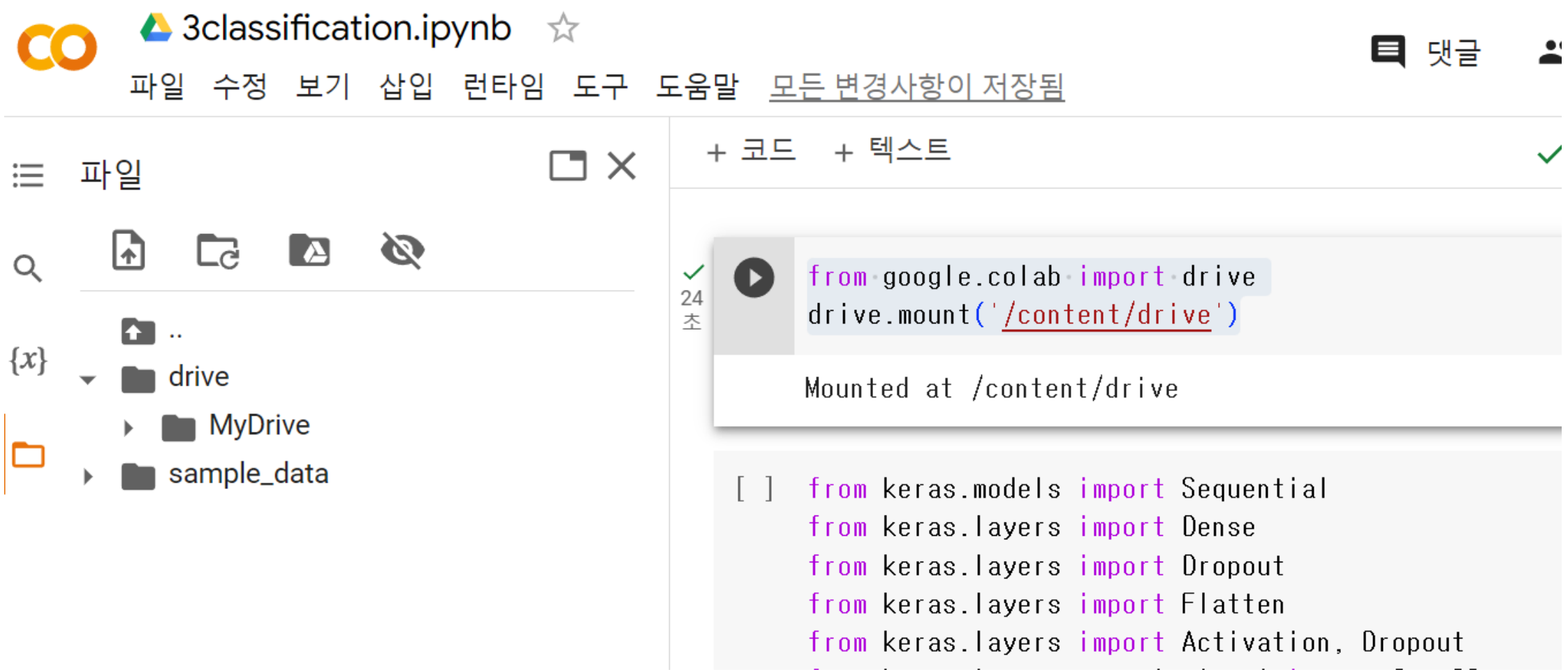
영상 분류 예제

- 영상 구글 드라이브 upload



영상 분류 예제

- **Google drive mount**



영상 분류 예제

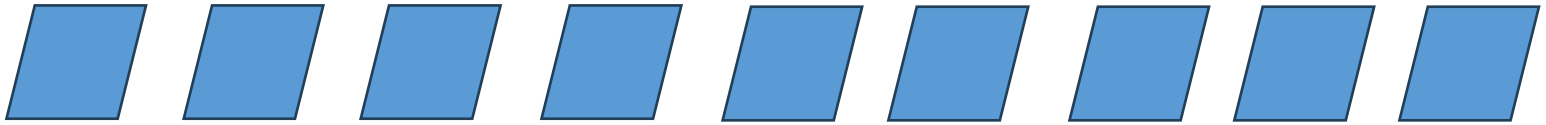

```
dir_path = '/content/drive/MyDrive/datasets'
categories = ['airplane','truck','bird','cat']
data = []

for i in categories:

    paths = os.path.join(dir_path,i)
    print('paths',paths)                                    paths /content/drive/MyDrive/datasets/airplane
    ii=0
    for j in os.listdir(paths):
        img_path = os.path.join(paths,j)
        labels = categories.index(i)
        print('image_pathlabels',img_path,labels)           /content/drive/MyDrive/datasets/truck/0016.jpg 1

        if (os.path.splitext(img_path)[1]=='.jpg'):
          img = cv2.imread(img_path)
          img = cv2.resize(img,(32,32))
          data.append([img,labels])

random.shuffle(data)
print(len(data))
```

영상 분류 예제

categories = ['airplane','truck','bird','cat']

0 1 2 3

data

img	
labels	1 0 2 1 3 2 1 0 2

Train image : 카테고리 별로 20장

Test image : 카테고리 별로 5장

영상 분류 예제


```
x = []
y = []

for features,label in data:
    x.append(features)
    y.append(label)

#Converting lists into numpy arrays
x = np.array(x)
y = np.array(y)
x = x/255.0
x = np.array(x).reshape(-1, 32, 32, 3)
print("Shape of train images is:", x.shape)
print("Shape of labels is:", y.shape)
print(y)
print(x.shape[1:])
```

```
Shape of train images is: (80, 32, 32, 3)
Shape of labels is: (80,)
[3 2 3 0 0 1 3 1 3 2 3 2 2 1 1 2 2 2 1 1 3 0 1 0 0 2 0 3
3 3 0 0 1 2 0 2 2
 0 1 2 0 0 2 3 2 3 1 0 1 2 1 2 1 3 1 0 3 0 2 2 0 2 1 1 3
1 1 0 2 0 3 1 3 3
 1 3 0 0 3 3]
(32, 32, 3)
```

영상 분류 예제


```
model = Sequential()
model.add(Conv2D(64,(3,3), activation='relu', input_shape =
(x.shape[1:])))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128,(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))
model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4,activation='softmax'))
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d_3 (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_5 (MaxPooling2D)	(None, 2, 2, 128)	0
dropout_2 (Dropout)	(None, 2, 2, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 4)	516

영상 분류 예제


```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x, y, epochs=60,batch_size=50,validation_split=0.2)
```

```
Epoch 1/60
5/5 [==============================] - 2s 182ms/step - loss: 1.1215 -
accuracy: 0.3167 - val_loss: 1.0945 - val_accuracy: 0.3000
Epoch 2/60
5/5 [==============================] - 1s 116ms/step - loss: 1.1089 -
accuracy: 0.3583 - val_loss: 1.0947 - val_accuracy: 0.3500
Epoch 3/60
5/5 [==============================] - 1s 110ms/step - loss: 1.0879 -
accuracy: 0.3958 - val_loss: 1.0857 - val_accuracy: 0.4167
Epoch 4/60
5/5 [==============================] - 1s 168ms/step - loss: 1.0916 -
accuracy: 0.3583 - val_loss: 1.0818 - val_accuracy: 0.5333
Epoch 5/60
5/5 [==============================] - 1s 183ms/step - loss: 1.0810 -
accuracy: 0.4083 - val_loss: 1.0788 - val_accuracy: 0.3833
Epoch 6/60
5/5 [==============================] - 1s 177ms/step - loss: 1.0724 -
accuracy: 0.3958 - val_loss: 1.0579 - val_accuracy: 0.4333
Epoch 7/60
```

영상 분류 예제


```
import numpy as np
CATEGORIES = ['airplane', 'truck','bird','cat']

def image(path):
    print('path',path)
    img = cv2.imread(path)

    new_arr = cv2.resize(img, (32, 32))
    new_arr = np.array(new_arr)
    new_arr = new_arr/255.0
    new_arr = new_arr.reshape(-1, 32, 32, 3)

    return new_arr
```

영상 분류 예제


```
prediction = model.predict(image('/content/drive/MyDrive/datasets/_test_bird/0046.jpg'))
#print(prediction.argmax())
print(np.round(prediction,3))
#airplane: 1 0 0 0, truck : 0 1 0 0, bird : 0 0 1 0,cat: 0 0 0 1
```

```
path /content/drive/MyDrive/datasets/_test_bird/0046.jpg
1/1 [==============================] - 0s 87ms/step
[[0.02  0.001 0.84  0.139]]
```

영상 분류 예제

과제3 . airplane, truck ,cat을을 분류하는 code를 작성하고 실행하시오.

airplane, truck, cat data 사용

*.ipynb *.py upload

4classification_2024.ipynb code 사용

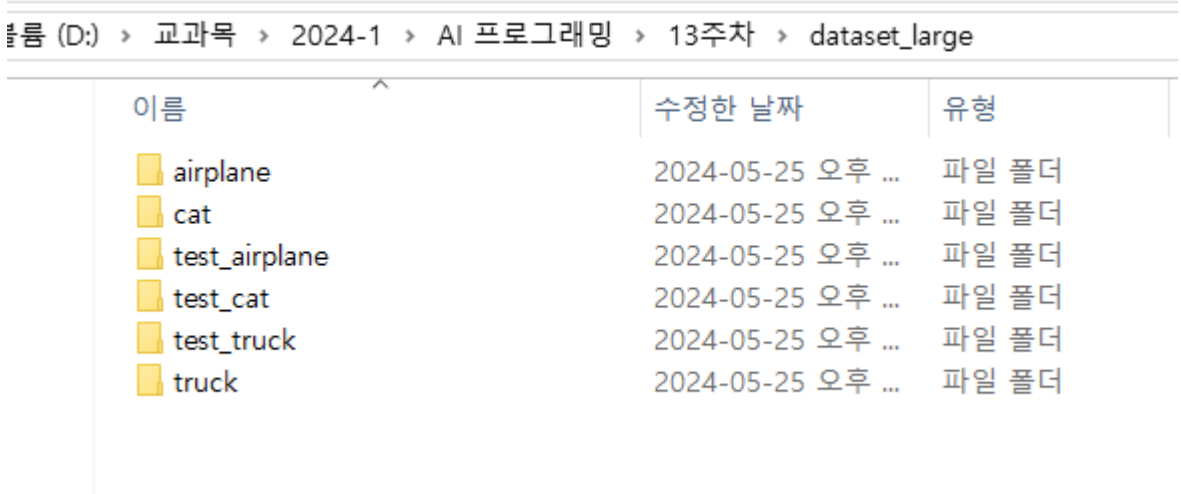
미니 프로젝트

dataset_large image 를 이용하여 airplane, truck ,cat을 분류하는 code를 만드시오,

성능 기준 : 마지막 혹은 best val accurac가 0.8 이상
30개 test 데이터중 25개 이상 예측 과 정답이 같아야 함 (correct_cnt.

MODEL 구조, EPOCH, BATCH SIZE 등을 변경 하여 성능을 높여보시오.

*.ipynb, *,.py upload



수고하셨습니다

jhmin@inhatc.ac.kr