딥러닝/클라우드

Chapter 13

Keras CNN
(Convolution Neural Network)

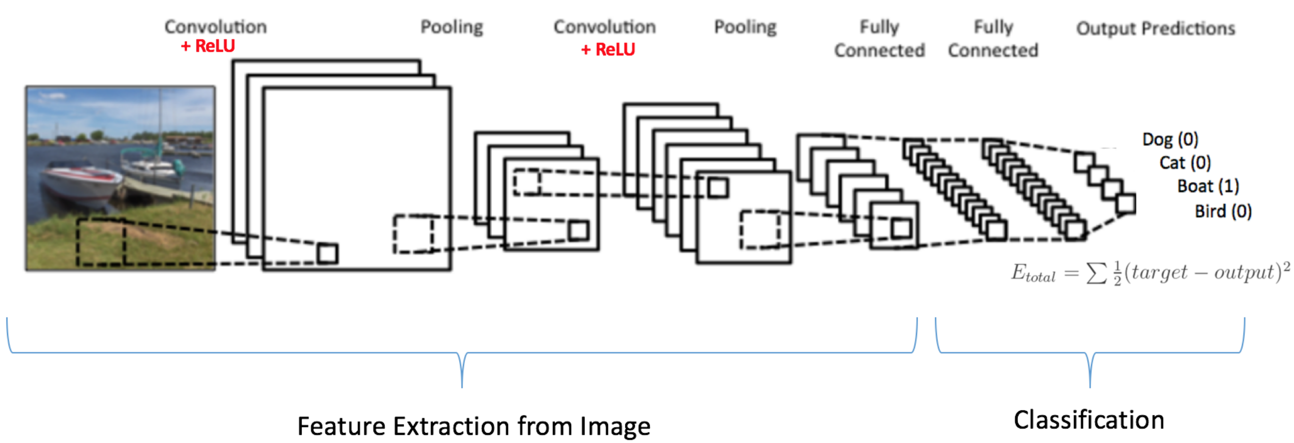
오 세 종

DANKOOK UNIVERSITY

Contents

1. Basis of image processing
2. CNN concepts
3. CNN building
4. MNIST classification by CNN

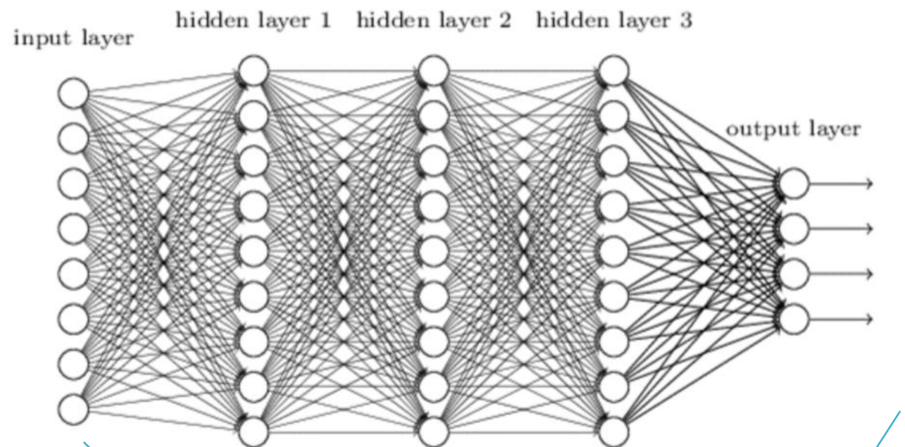
CNN : Convolutional Neural Network



https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3

What is CNN ?

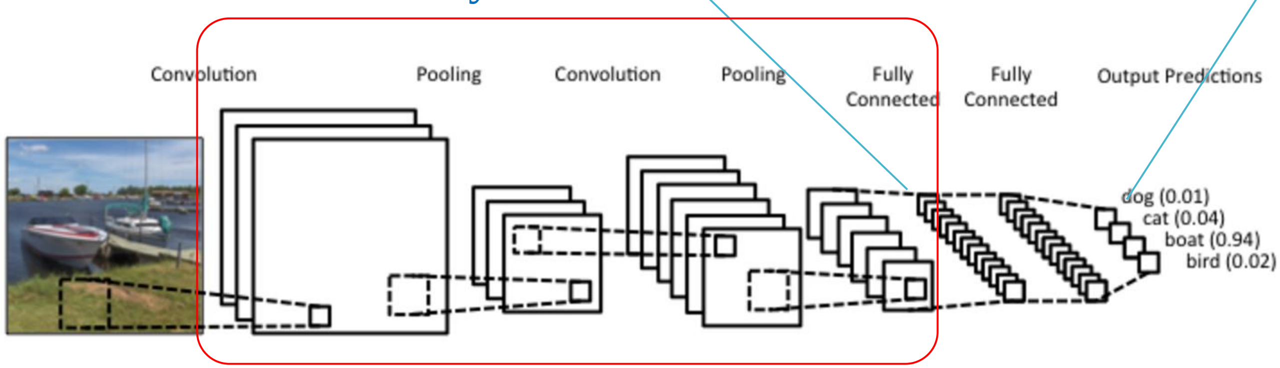
- Image 분류에 우수한 성능을 보이는 deep neural network
- Fully connected layers 앞에 convolution layers 존재



(http://physics2.mju.ac.kr/juhapruwp/?p=1517)

DNN

convolution layers

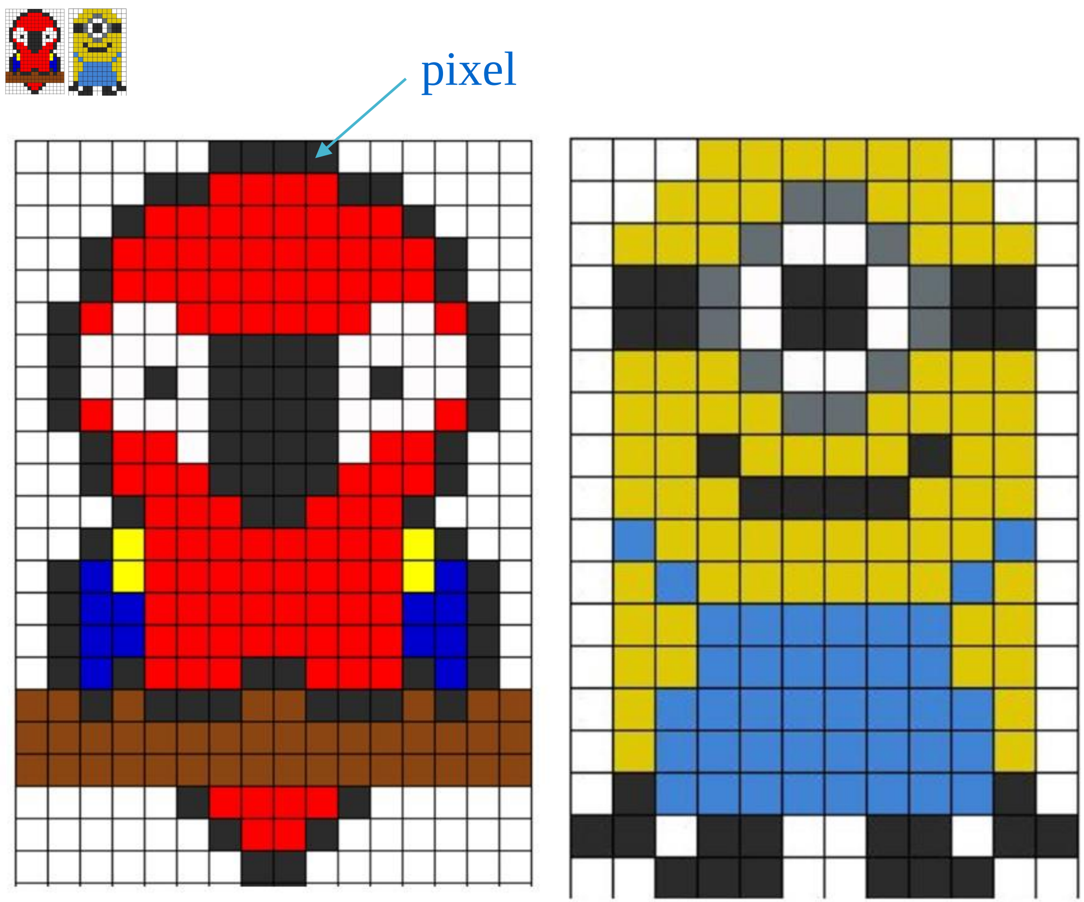


CNN

https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f

1. Image processing

- Image expression

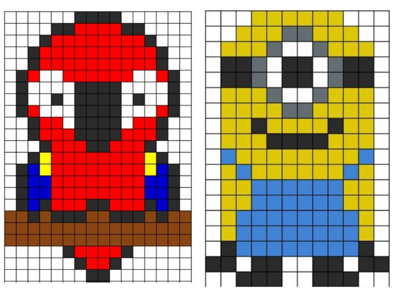


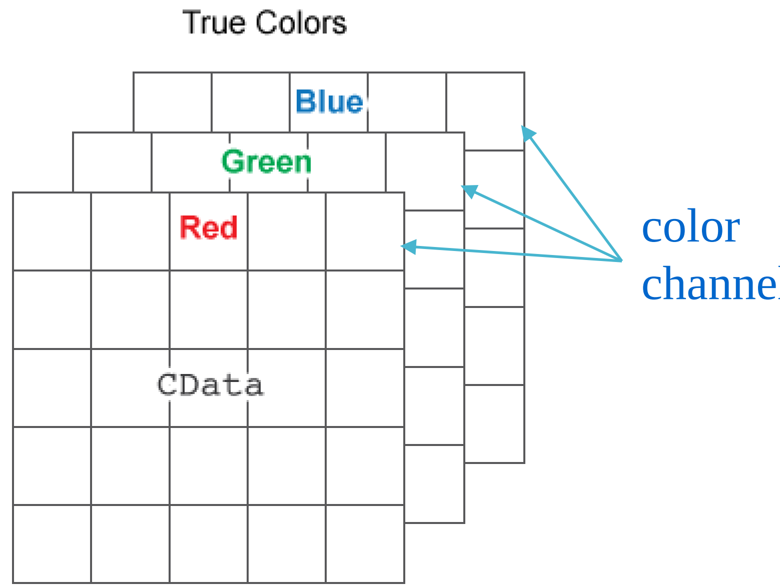
http://www.qygjxz.com/pixel.html

R,G,B 값의
조합에 의해서
Color 표현

1. Image processing

- Image expression
  - 컬러 이미지 데이터의 저장 → 3차원 array



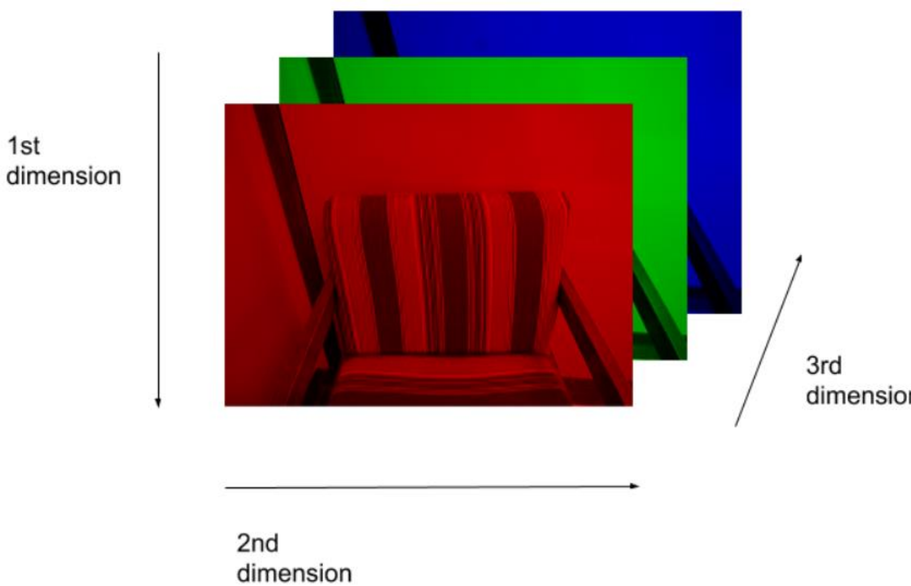


https://uk.mathworks.com/help/matlab/ref/image.html

1. Image processing

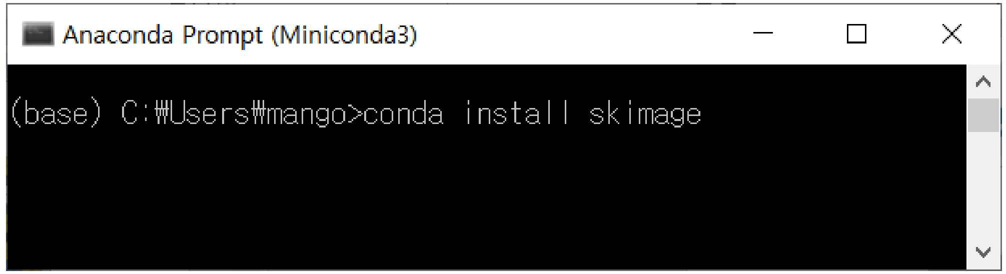
- Image can be stored into numpy array

A visual representation of how this image is stored as a NumPy array is:



1. Image processing

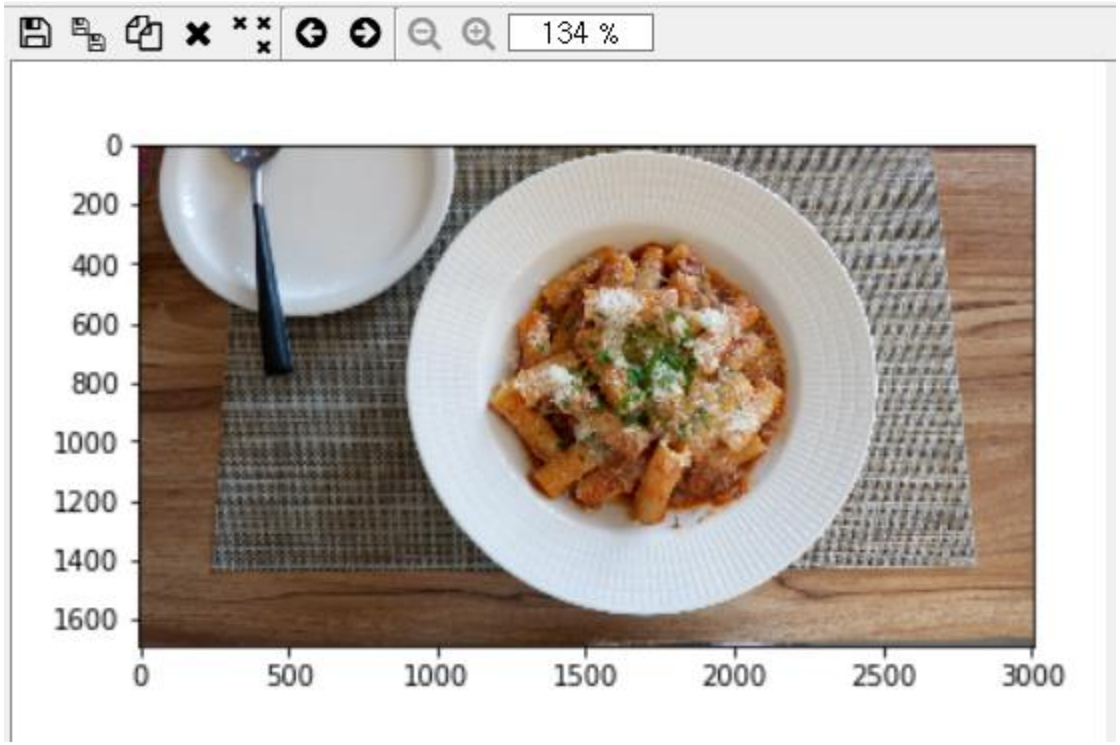
- Python package : scikit-image (skimage)
- Install skimage



- Read image file

```
from skimage import io
from tkinter.filedialog import askopenfilename

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

fname = askopenfilename()                # choose image file
image = mpimg.imread(fname)              # read image
plt.imshow(image)                        # display image
```

1. Image processing



1. Image processing

- Image info

```
# type of image object
type(image)

# image shape
print(image.shape)

# image data
print(image[:,:,1])  # red channel
```

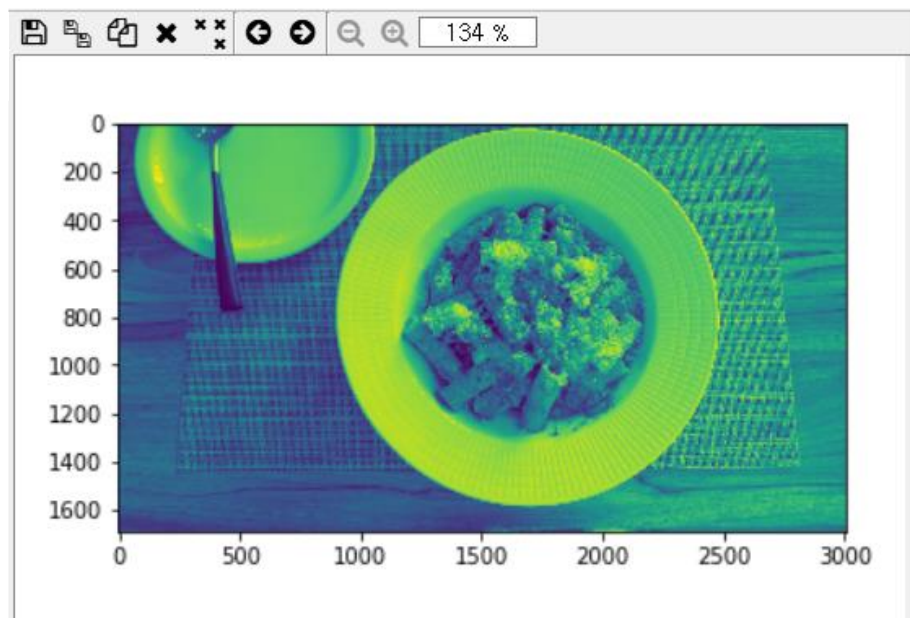
```
In [9]: type(image)
Out[9]: numpy.ndarray

In [10]: print(image.shape)
(1688, 3008, 3)

In [11]: print(image[:,:,1])  # red channel
[[ 23  15  11 ... 112 113 111]
 [ 20  15  12 ... 103 102 103]
 [ 20  17  15 ...  99  98  98]
 ...
 [ 62  63  64 ... 106 103 103]
 [ 58  59  60 ... 107 106 105]
 [ 62  60  60 ... 105 107 106]]
```

1. Image processing

- Color to gray

```
# color to gray
from skimage import color

gray_image = color.rgb2gray(image)
print(gray_image.shape)
plt.imshow(gray_image)
```

```
In [14]: print(gray_image.shape)
(1688, 3008)
```

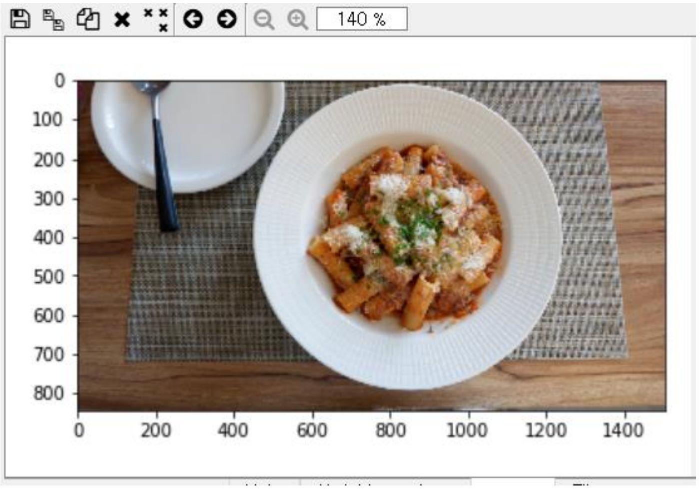


1. Image processing

- Resize

```
# resize
from skimage import transform

new_shape = (image.shape[0] // 2, image.shape[1] // 2,
             image.shape[2])
small = transform.resize(image=image,
        output_shape=new_shape)

print(small.shape)
plt.imshow(small)
```

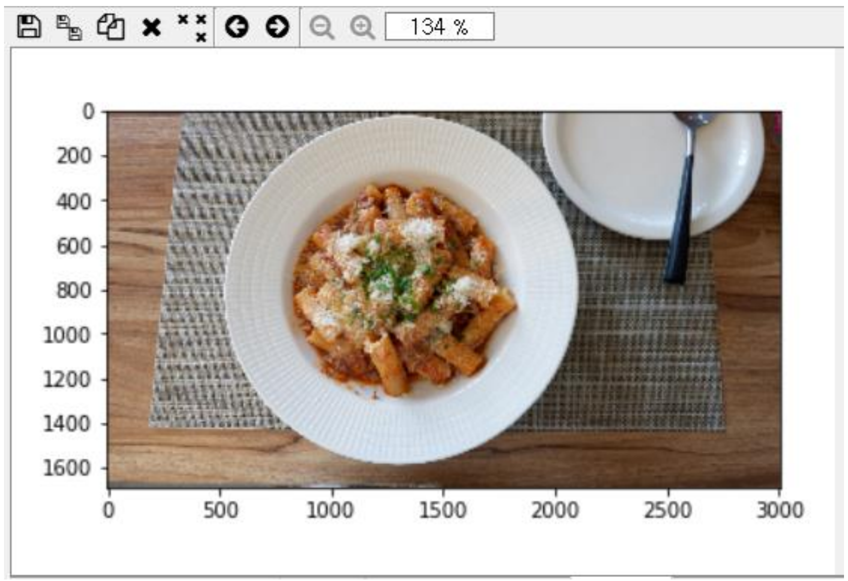
```
In [18]: print(small.shape)
(844, 1504, 3)
```



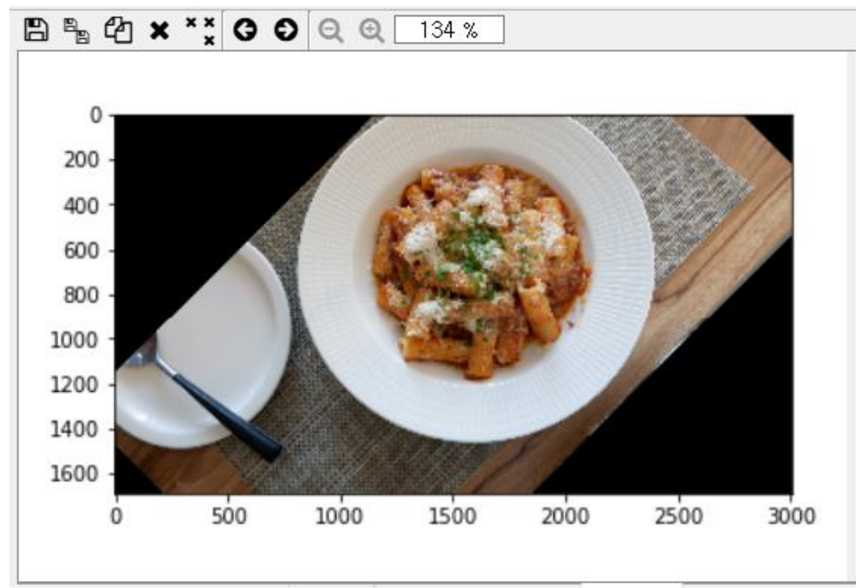
1. Image processing

- Filp, rotate

```
from skimage.transform import rotate

plt.imshow(np.fliplr(image))            # filp
plt.imshow(rotate(image, angle=45))  # rotate
```



flip

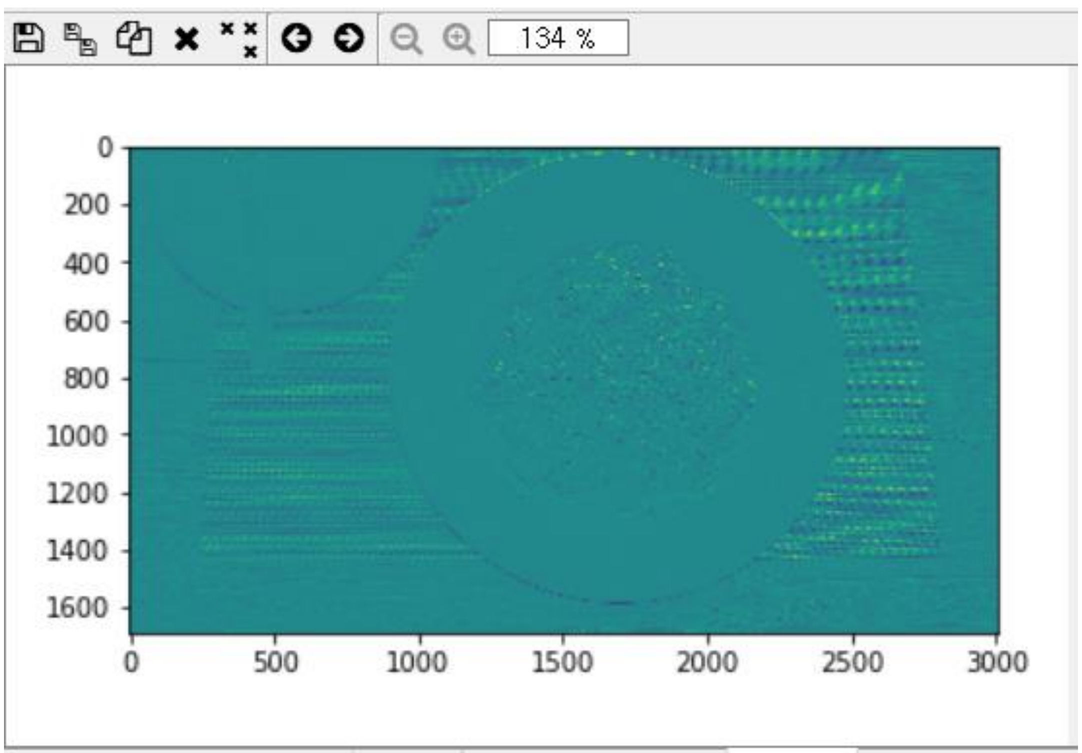


rotate

1. Image processing

- Filters

```
# filters
from skimage.filters import sobel_h

plt.imshow(sobel_h(gray_image))
```



1. Image processing

- Save image

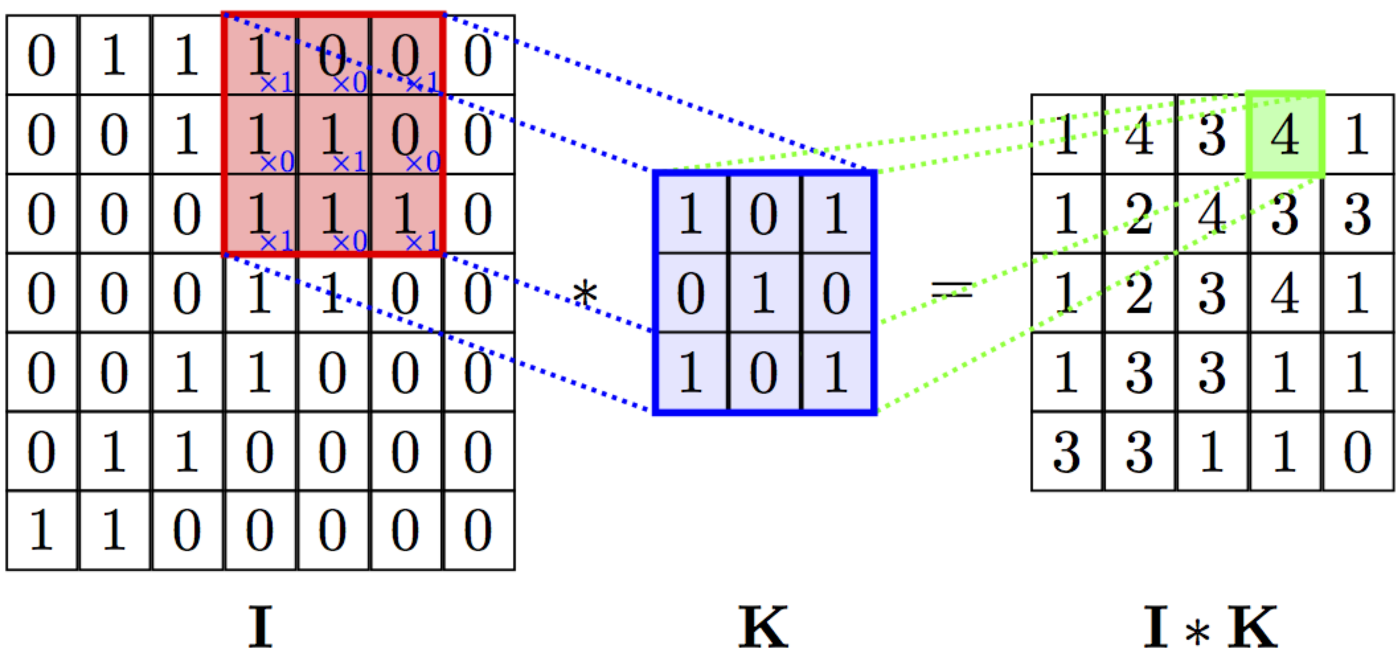
```
# save image
io.imsave("d:/data/test2.png", image)
```

9 Powerful Tips and Tricks for Working with Image Data using skimage in Python

https://www.analyticsvidhya.com/blog/2019/09/9-powerful-tricks-for-working-image-data-skimage-python/

2. CNN concepts

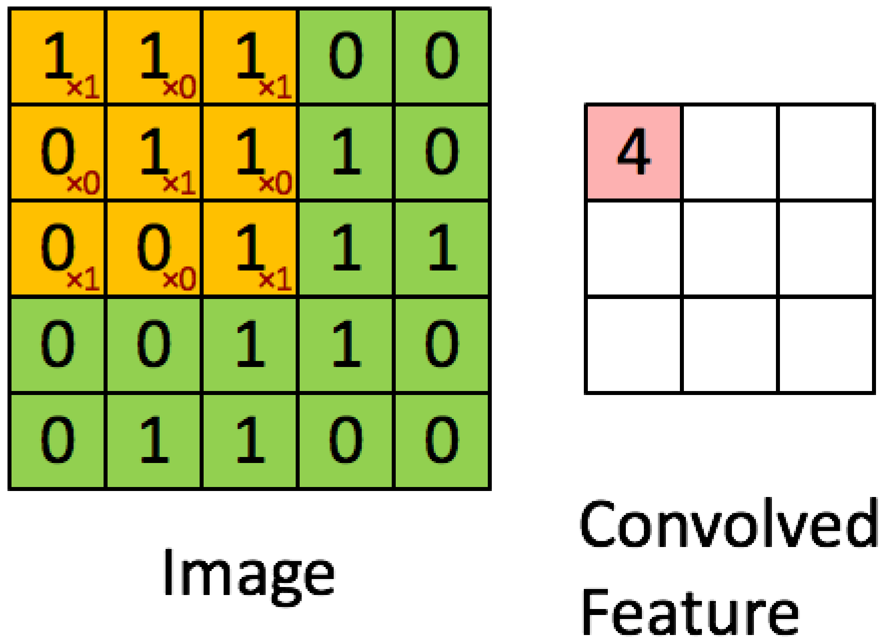
- Convolution (합성곱)
  - 영상처리의 일종으로 이미지에 특정 필터를 적용하여 새로운 이미지를 만든다



이미지 원본 필터 New Image

2. CNN concepts

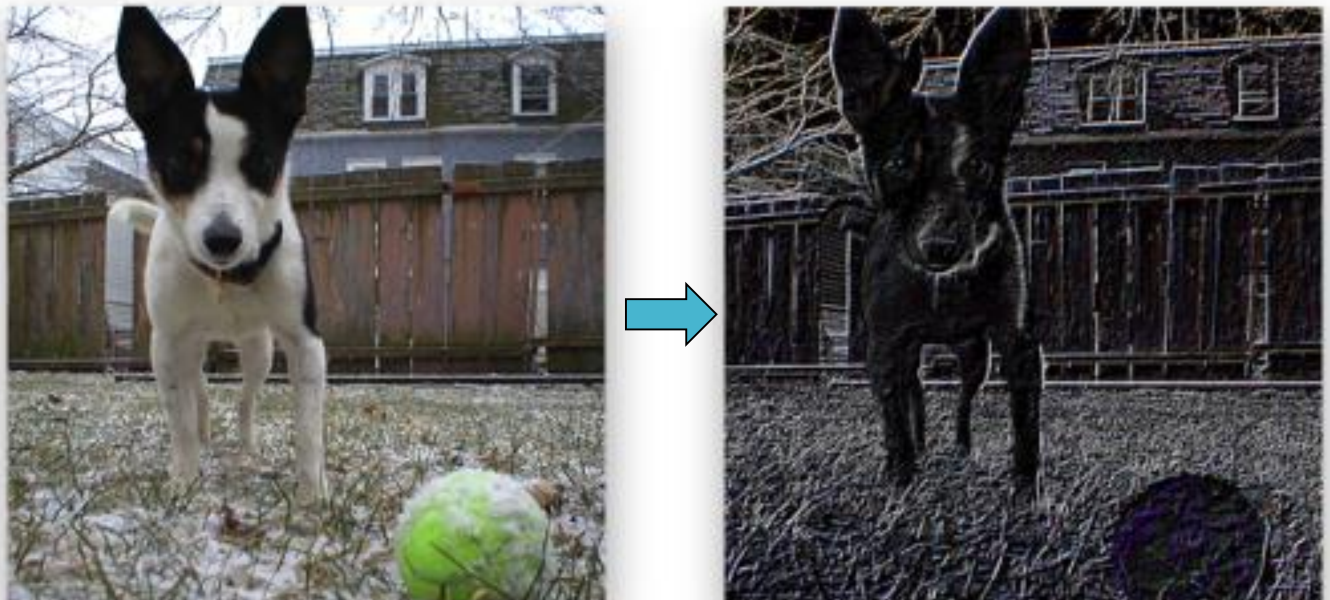
- Convolution (합성곱)



(http://taewan.kim/post/cnn/)

2. CNN concepts

- Convolution 의 예

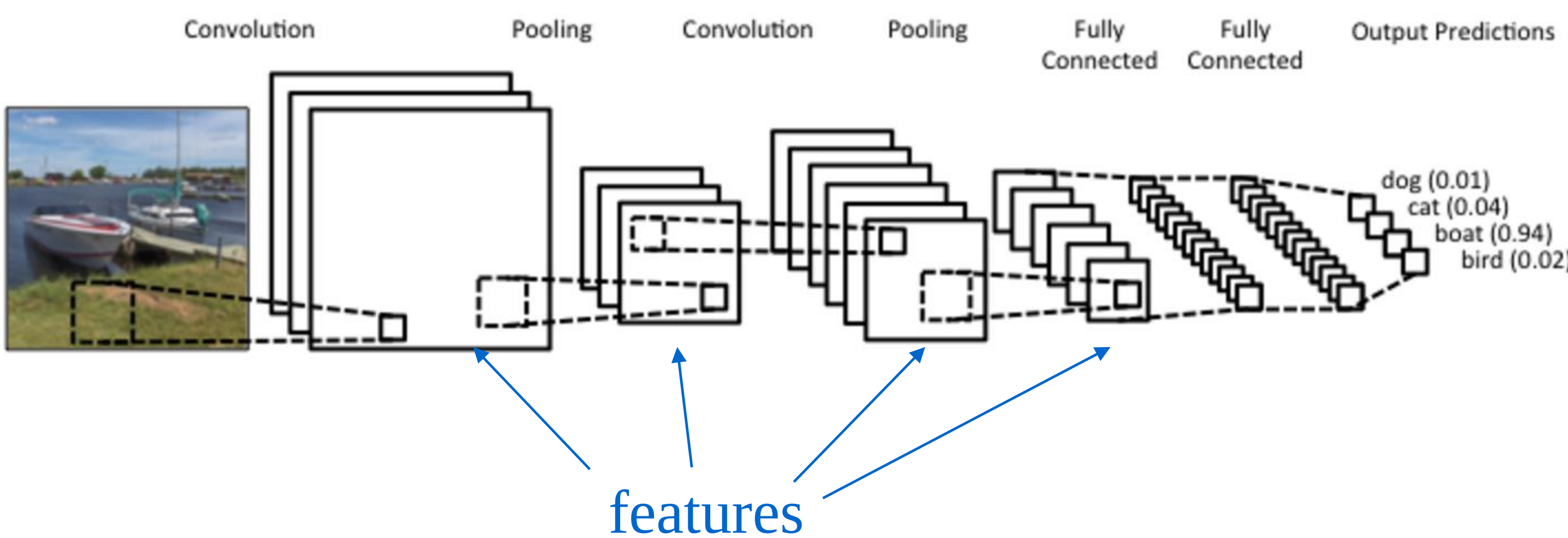


Original　　　　Emboss

https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

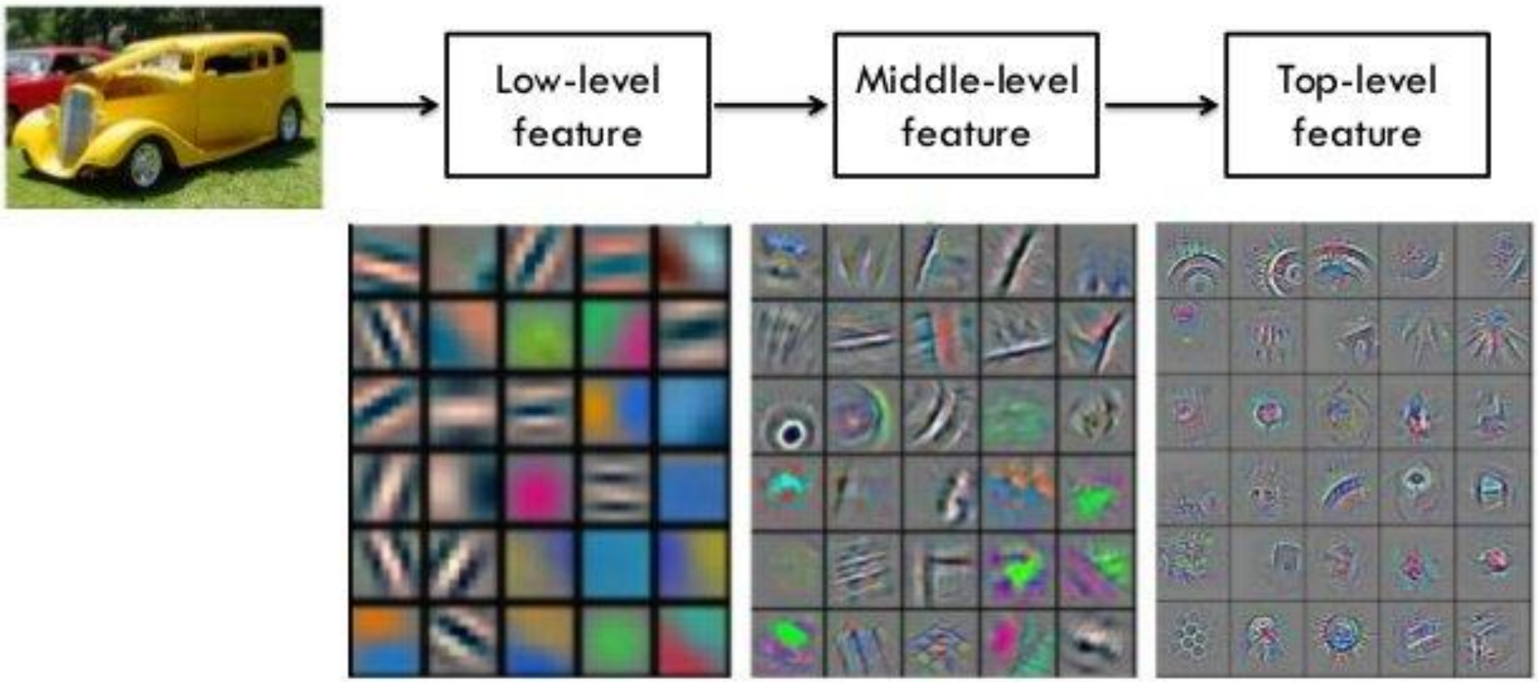
2. CNN concepts

- Convolution 연산의 의미
  - Convolution 에 의해 도출된 이미지를 feature 라고 한다
  - 한 이미지에서 여러 feature 를 만들 수 있으며, 각 feature 는 이미지의 특징 정보를 저장하게 된다
  - 결국 Convolution 은 이미지의 특징을 학습하는 과정이며, 이 과정은 자동적으로 이루어진다.



2. CNN concepts

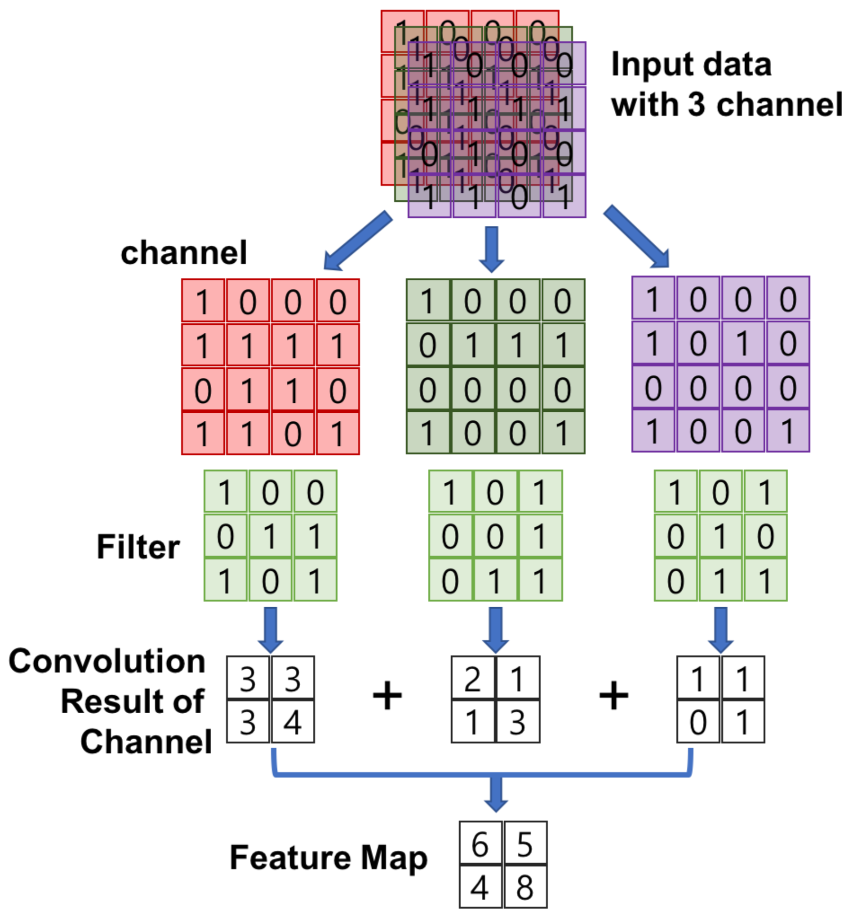
- Convolution 연산의 의미



https://arxiv.org/abs/1311.2901

2. CNN concepts

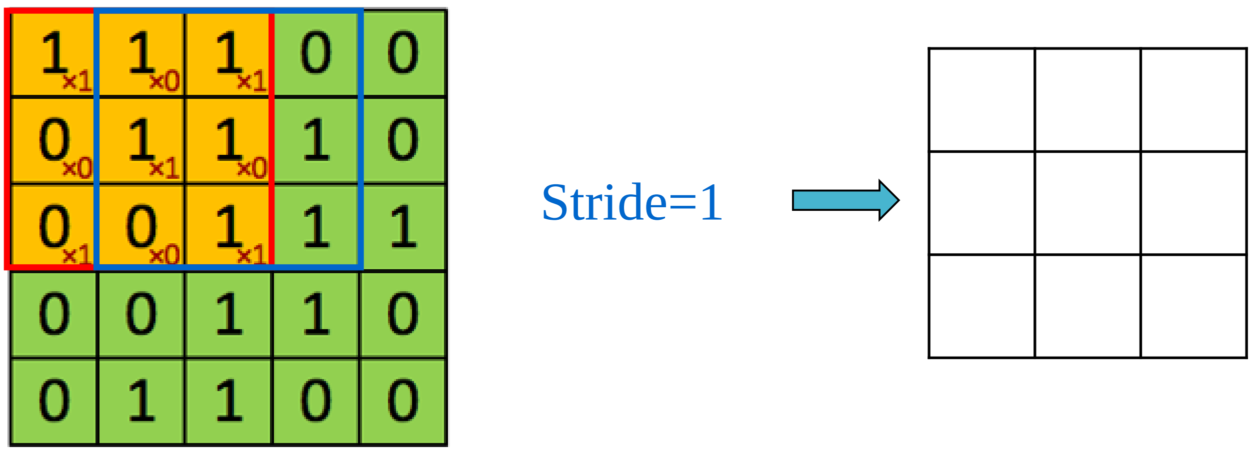
- Convolution (합성곱)
  - For 3 channel color image

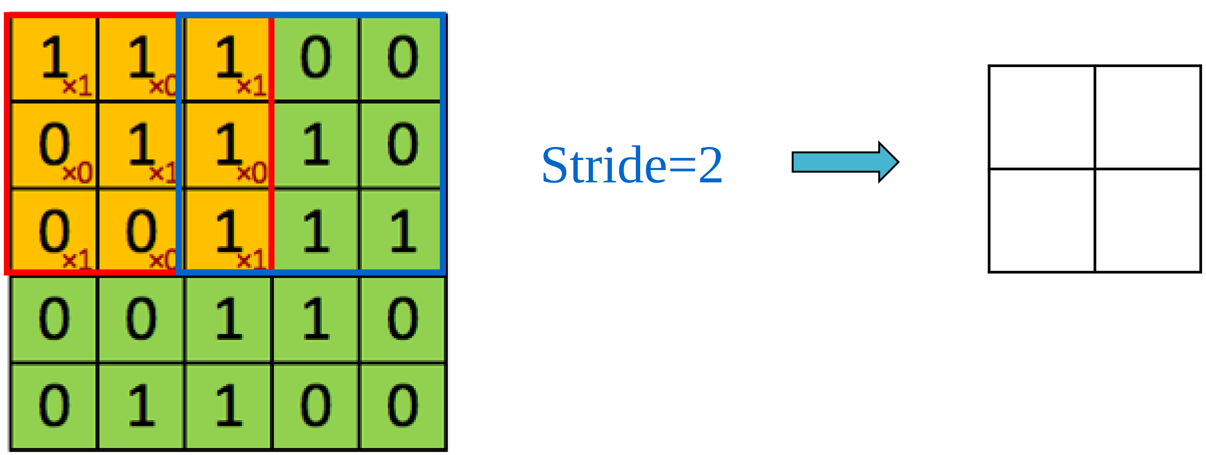


2. CNN concepts

- Stride
  - 이미지에 필터를 적용할 때 몇칸씩 옮겨가며 적용할 것인가

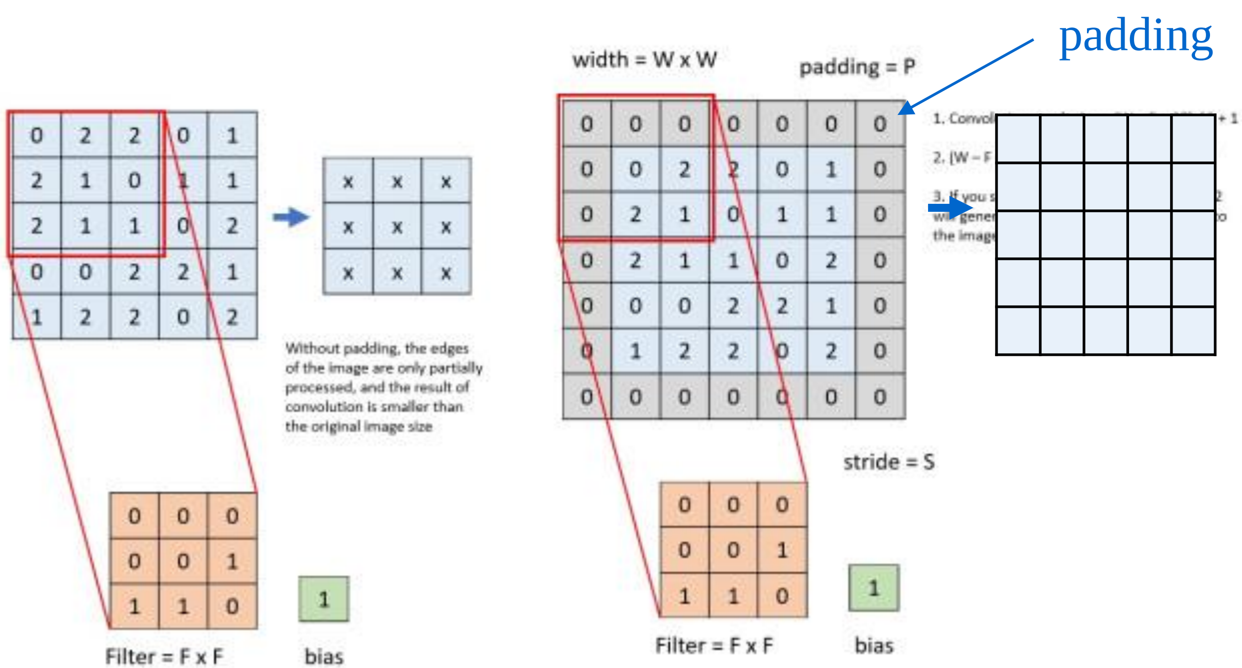
  - 3x3 필터





2. CNN concepts

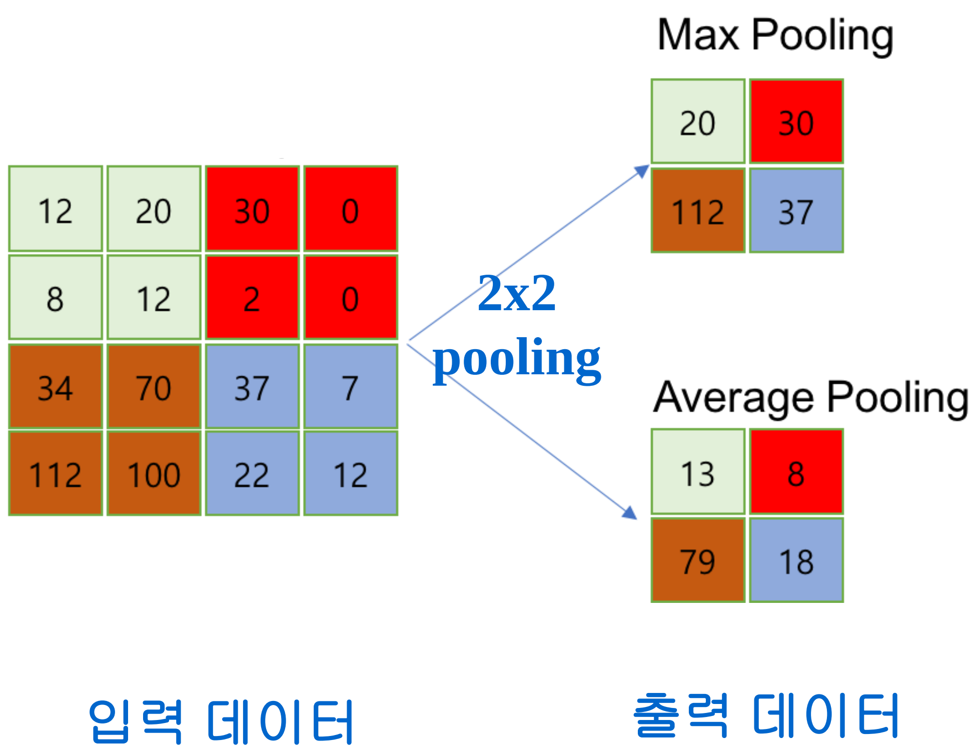
- Padding
  - Convolution 연산을 적용하면 그 특성상 원본 이미지의 크기가 줄어든다.
  - 원본 이미지의 크기를 유지하기 위한 장치가 padding
  - 입력 데이터의 외각에 지정된 픽셀만큼 특정 값으로 채워 넣는다



https://jamesmccaffrey.wordpress.com/2018/05/30/convolution-image-size-filter-size-padding-and-stride/

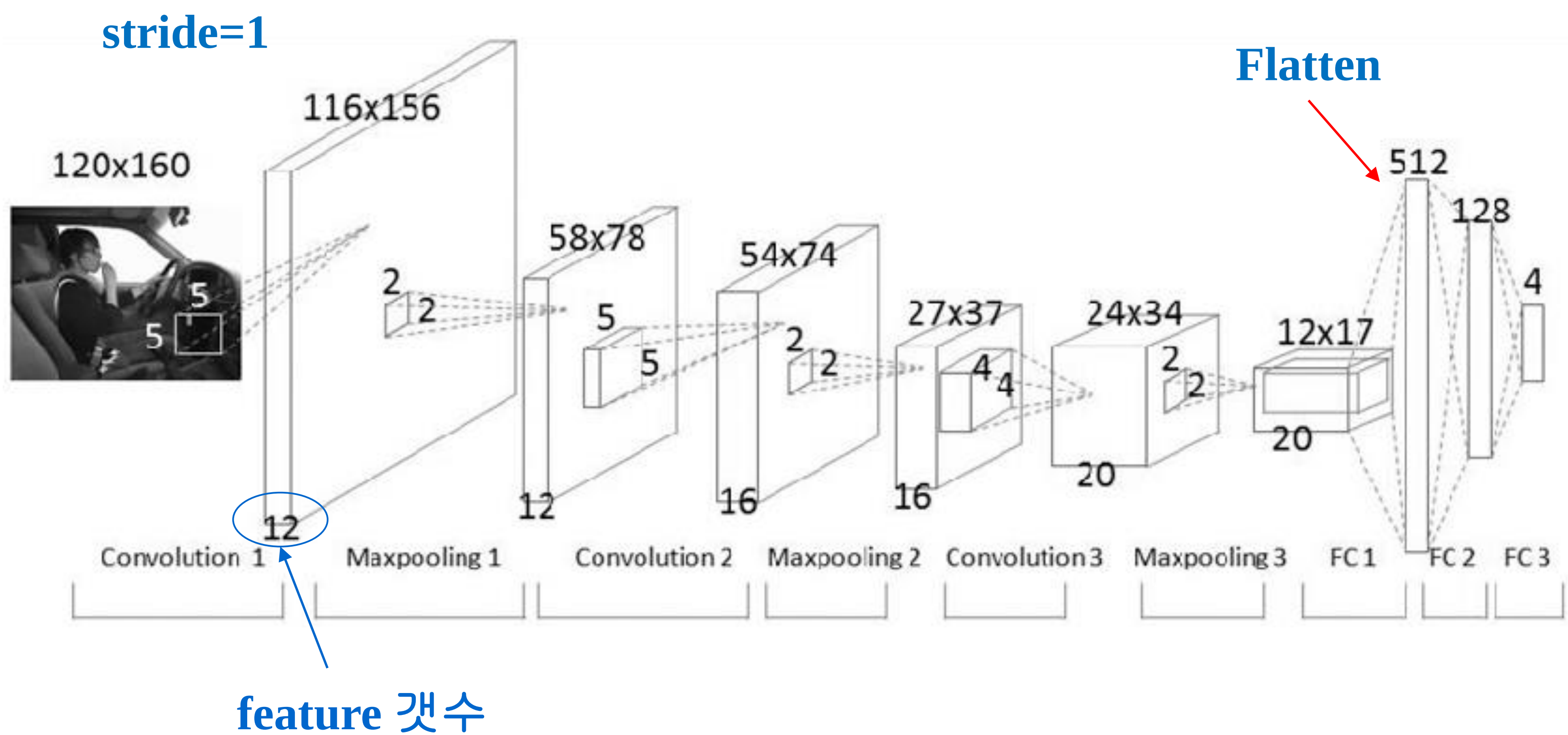
2. CNN concepts

- Pooling
  - 출력 데이터(이미지)를 입력으로 받아서 출력 데이터의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용
  - Max pooling (주로 사용됨), Average pooling, min pooling



3. CNN building

- CNN example



https://www.researchgate.net/figure/Architecture-of-our-unsupervised-CNN-Network-contains-three-stages-each-of-which_283433254

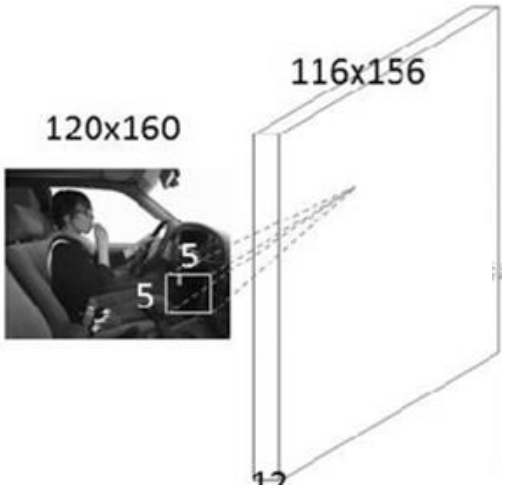
3. CNN building

- (1) Convolution 레이어 출력 데이터 크기 계산

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

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

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



입력 데이터 높이: 120
입력 데이터 폭: 160
필터 높이: 5
필터 폭: 5
Stride 크기: 1
패딩 사이즈: 0

$$OH = \frac{120+2*0-5}{1} + 1 = 116$$

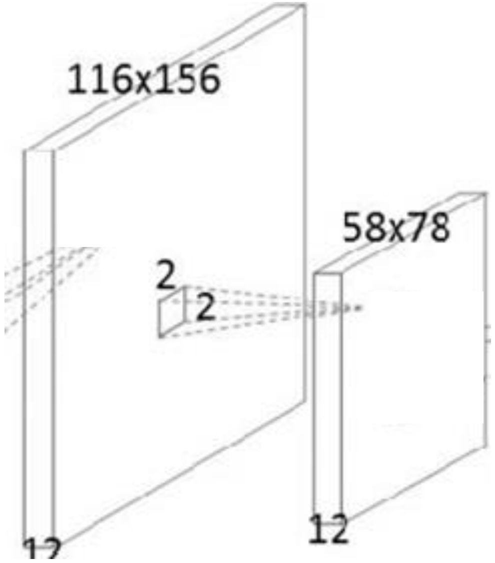
$$OW = \frac{160+2*0-5}{1} + 1 = 156$$

3. CNN building

- (2) pooling layer 크기 계산

$$OutputHeight = \frac{InputHeight}{PoolingSize\_height}$$

$$OutputWidth = \frac{InputWidth}{PoolingSize\_width}$$



2x2 pooling

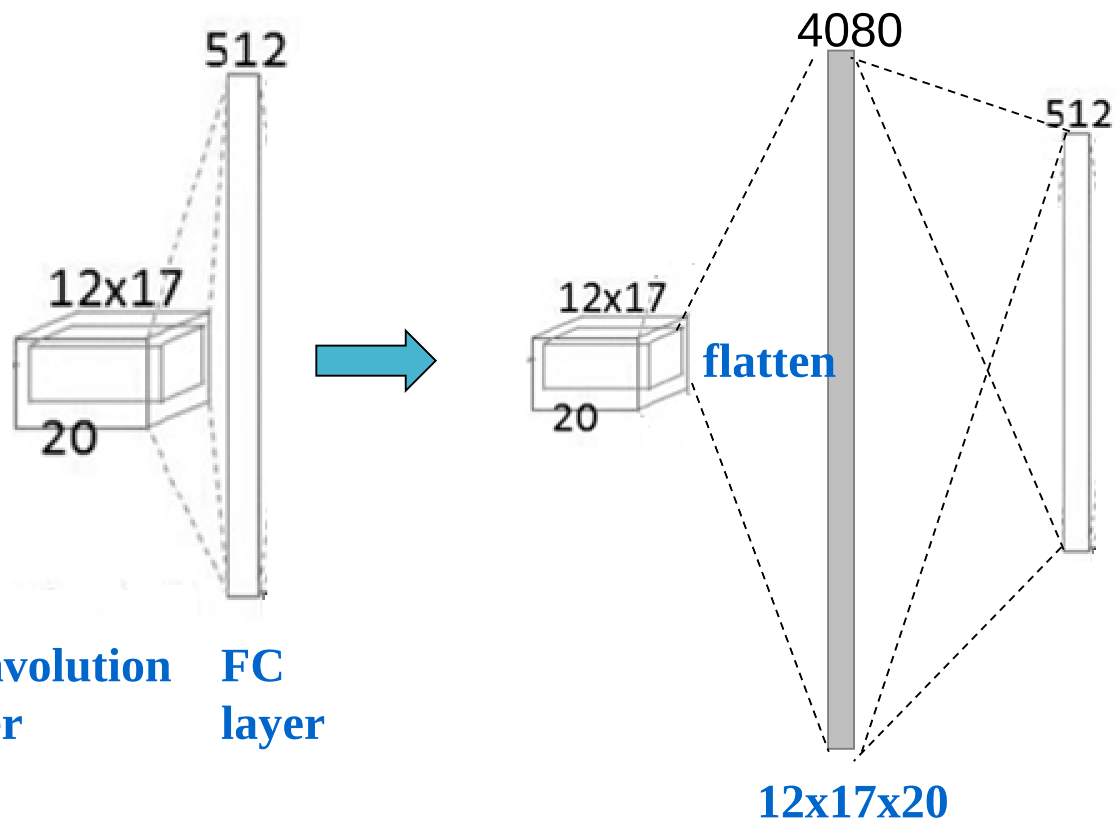
$$OH = \frac{116}{2} = 58$$

$$OW = \frac{156}{2} = 78$$

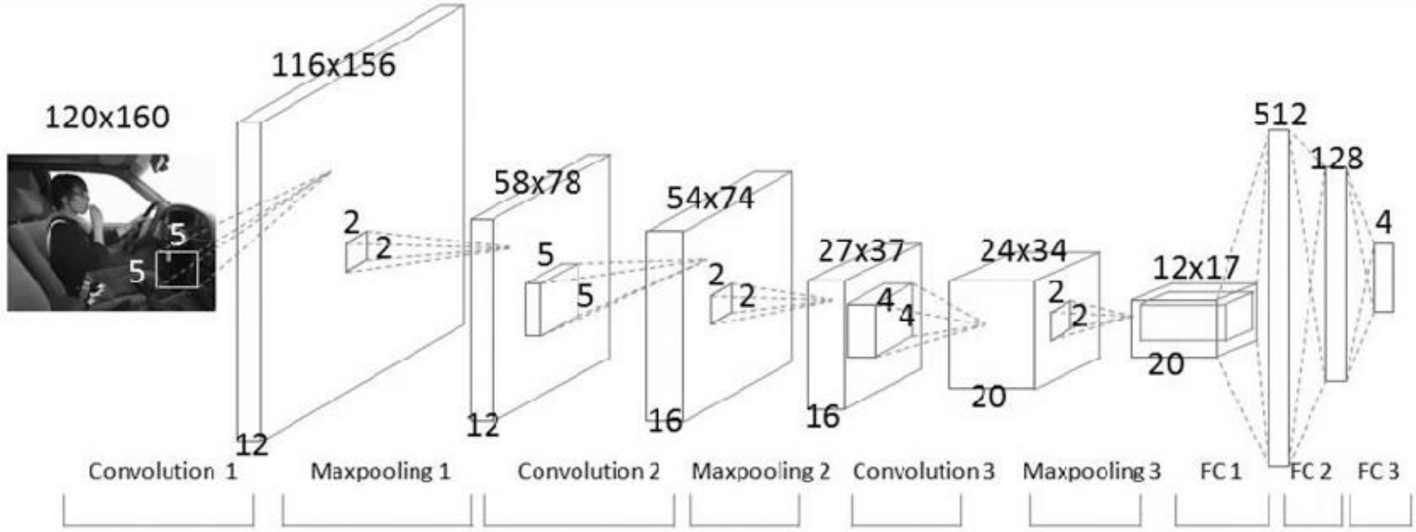
3. CNN building

- (3) Convolution layer -> Fully Connected layer

12x17 image 가
20개

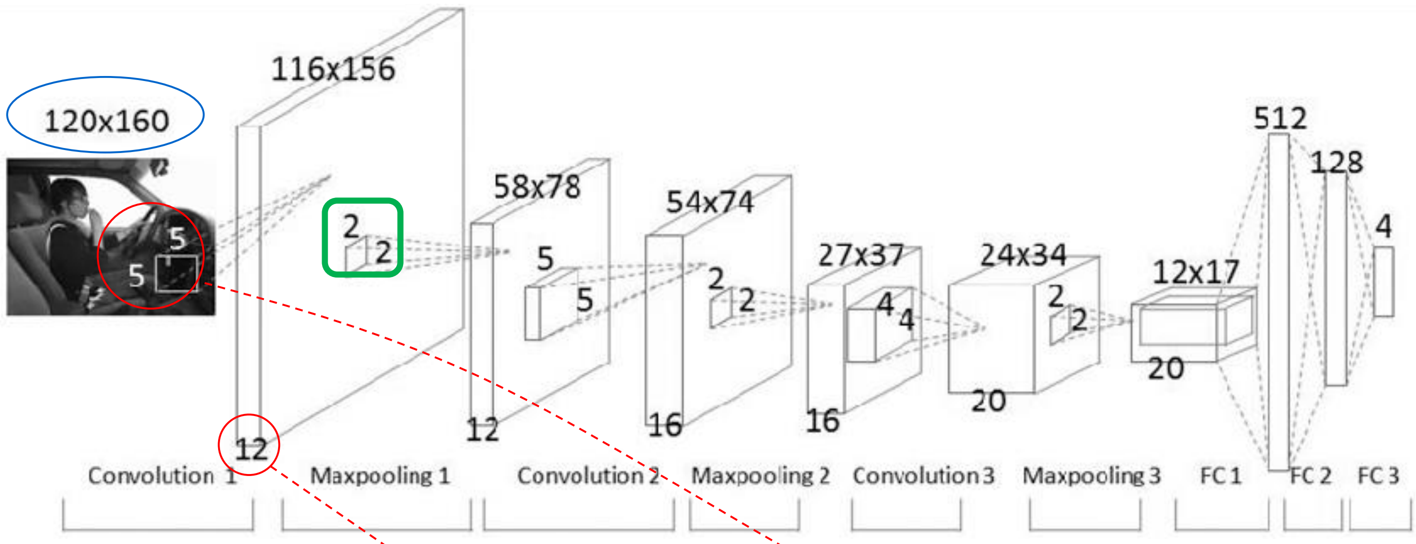


이것을 일렬로 세운후 (flatten) FC 에 연결




```
model = Sequential()
model.add(Convolution2D(12, kernel_size=(5, 5),
                        input_shape=(120, 160, 1),
                        activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Convolution2D(16, kernel_size=(5, 5),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Convolution2D(20, kernel_size=(4, 4),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation='softmax'))
```

3. CNN building




```
model = Sequential()
model.add(Convolution2D(12, kernel_size=(5, 5),
                               input_shape=(120, 160, 1),
                               activation='relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))
 …
```

Color channel 수
(흑백:1, 컬러:3)

4. MNIST classification by CNN

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import  numpy as np
import matplotlib.pyplot as plt

# define image size
img_rows=28
img_cols=28

# load dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0
```

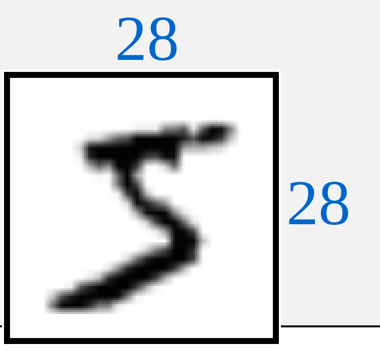
4. MNIST classification by CNN

```
# reshape
X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, 1)
X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, 1)


# one hot encoded
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)


# fix random seed for reproducibility
seed = 100
np.random.seed(seed)
num_classes = 10
```

★

4. MNIST classification by CNN

```
# create CNN model
def cnn_model():
    # define model
    model = Sequential()
    model.add(Convolution2D(32, kernel_size=(5, 5),
                            border_mode='valid',      # same
                            strides=(1, 1),
                            input_shape=(img_rows, img_cols, 1),
                            activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(127, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    # Compile model
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
```



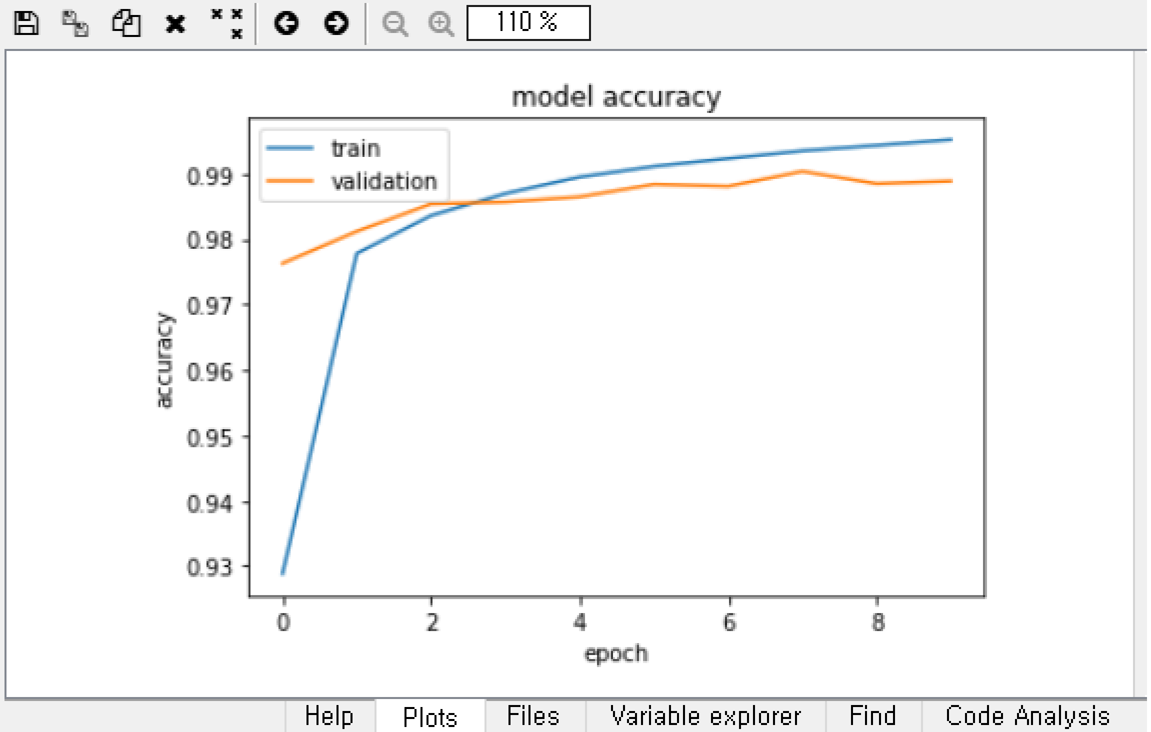
4. MNIST classification by CNN

```
# build the model
model = cnn_model()

# Fit the model
disp = model.fit(X_train, y_train,
          validation_data=(X_test, y_test),
          nb_epoch=10,
          batch_size=200,
          verbose=1)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("loss: %.2f" % scores[0])
print("acc: %.2f" % scores[1])
```

```
In [34]: print("loss: %.2f" % scores[0])
loss: 0.03

In [35]: print("acc: %.2f" % scores[1])
acc: 0.99
```

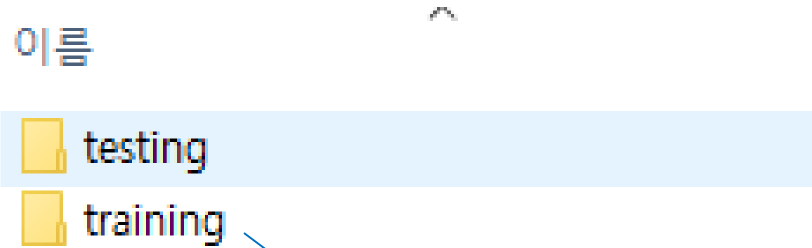
```
Epoch 4/10
60000/60000 [==============================] - 15s 255us/step - loss: 0.0418 -
accuracy: 0.9872 - val_loss: 0.0410 - val_accuracy: 0.9857
Epoch 5/10
60000/60000 [==============================] - 16s 262us/step - loss: 0.0333 -
accuracy: 0.9897 - val_loss: 0.0394 - val_accuracy: 0.9867
Epoch 6/10
60000/60000 [==============================] - 15s 255us/step - loss: 0.0284 -
accuracy: 0.9910 - val_loss: 0.0329 - val_accuracy: 0.9876
Epoch 7/10
60000/60000 [==============================] - 16s 259us/step - loss: 0.0238 -
accuracy: 0.9923 - val_loss: 0.0346 - val_accuracy: 0.9883
Epoch 8/10
60000/60000 [==============================] - 16s 259us/step - loss: 0.0197 -
accuracy: 0.9936 - val_loss: 0.0343 - val_accuracy: 0.9896
Epoch 9/10
60000/60000 [==============================] - 16s 264us/step - loss: 0.0170 -
accuracy: 0.9947 - val_loss: 0.0319 - val_accuracy: 0.9893
Epoch 10/10
60000/60000 [==============================] - 18s 304us/step - loss: 0.0147 -
accuracy: 0.9951 - val_loss: 0.0326 - val_accuracy: 0.9895
```

```
# summarize history for accuracy
plt.plot(disp.history['accuracy'])
plt.plot(disp.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

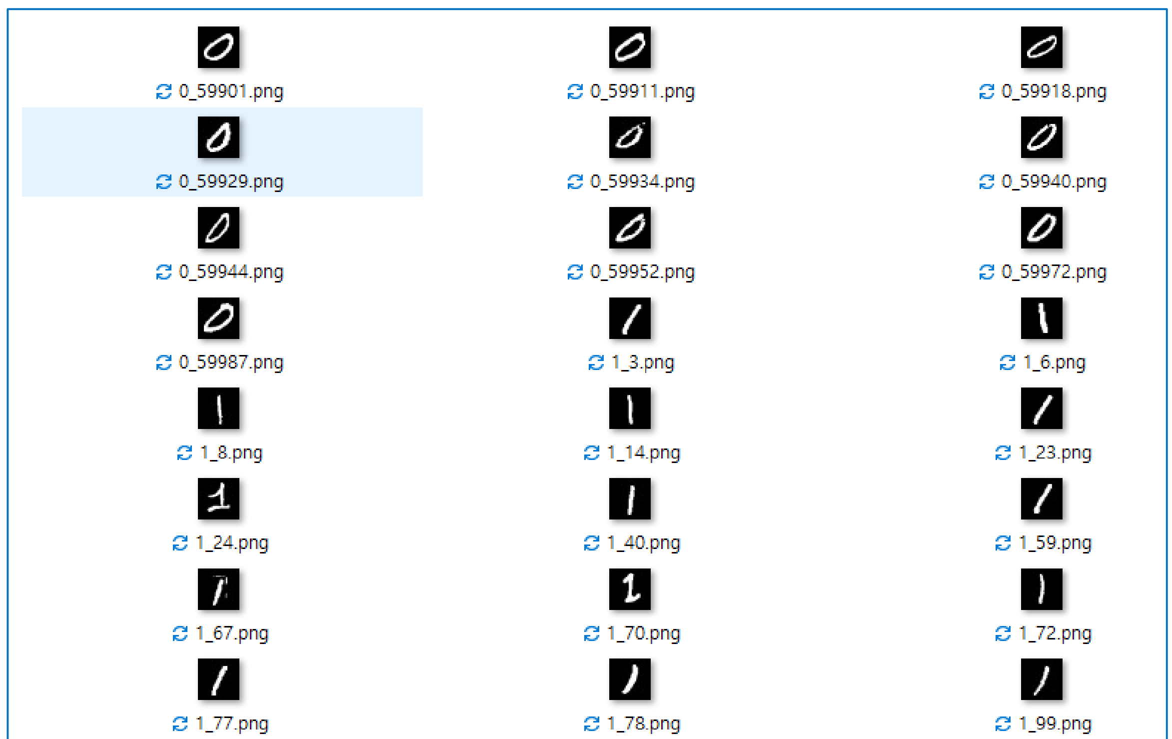


5. Read MNIST image file

- 이미지 파일을 읽어서 training/test dataset 을 만들어보자



파일 이름의 첫글자가 레이블



5. Read MNIST image file

```
# load MNIST image files

import numpy as np
from keras.utils import np_utils
import os      # operating system interfaces

# Set data folder
img_dir_train = 'D:\\mnist\\training'
img_dir_test = 'D:\\mnist\\testing’

# get file names
flist_train = os.listdir(img_dir_train)
flist_test  = os.listdir(img_dir_test)
```

5. Read MNIST image file

```
# Preprocess the image into a 4D array using
keras.preprocessing
from keras.preprocessing import image

# load train images
X_train = np.zeros(shape=(len(flist_train), 28,28,3))
y_train = np.zeros(shape=(len(flist_train)))

for idx, fname in enumerate(flist_train):
    img_path = os.path.join(img_dir_train, fname)
    img = image.load_img(img_path, target_size=(28,28))
    img_array_train = image.img_to_array(img)
    img_array_train = np.expand_dims(img_array_train,
axis=0)
    X_train[idx] = img_array_train
    y_train[idx] =  flist_train[idx][:1]

# scaling into [0, 1]
X_train = X_train / 255.0
```


* image.img_to_array: return 3D numpy array