ChatGPT와 함께하는 딥러닝 5일 완성

합성곱 신경망 기초와 이미지 분류 모델 만들기

1일차

서울대학교 평생교육원
Extension College Seoul National University

Topics

- 딥러닝과 합성곱 신경망(CNN)의 기본 구조
- 합성곱, 풀링, 완전 연결층 역할 파악
- 필터, 스트라이드, 패딩 개념 이해
- 학습 결과 분석을 위한 모델 평가 방법
- MNIST로 CNN 모델 구현 실습
- ChatGPT로 디버깅 및 개선 연습

ChatGPT

- OpenAI 에서 개발한 자연어 처리기술을 활용한 대화형 AI 모델
- 사용자와 대화를 통해 상호작용하며, 질문에 대답하고 자연스러운 대화를 제공
- GPT-3.5, GPT-4, 최신 GPT-4o까지 발전
- 사용자의 피드백과 입력을 바탕으로 계속해서 학습하고 발전하는 능력을 갖춤
- 코드 작성, 글쓰기, 요약 등 다용도에 강함

GPT = Generative + Pre-trained + Transformer

코드 생성 예제

- 사용자로부터 이름과 나이를 입력받아, 다음과 같은 인사 메시지를 출력하는 프로그램 코드를 작성하시오.
- 조건:
  - 나이가 20세 이상이면 "성인입니다." 출력
  - 20세 미만이면 "청소년입니다." 출력
- 예상 출력:
  - 이름을 입력하세요: Kim
  - 나이를 입력하세요: 18
  - 안녕하세요 Kim님! 청소년입니다.

예상 출력 코드

```
name = input("이름을 입력하세요: ")
age = int(input("나이를 입력하세요: "))

if age >= 20:
    status = "성인입니다."
else:
    status = "청소년입니다."

print(f"안녕하세요 {name}님! {status}")
```

Classification of Artificial Intelligence
인공지능의 분류

Artificial Intelligence

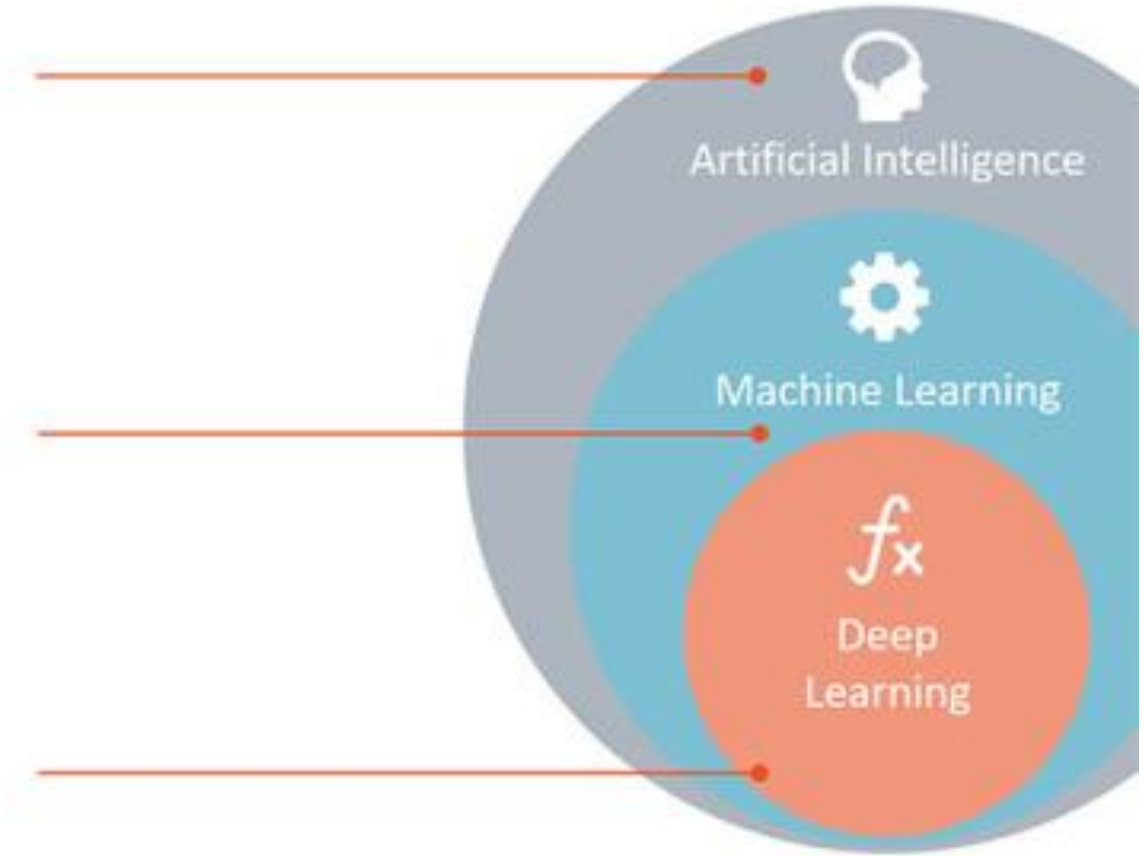
Any technique which enables computers to mimic human behavior.
컴퓨터가 사람의 행동을 흉내내는 모든 기술들

Machine Learning

Subset of AI techniques which use statistical methods to enable machines to improve with experiences.
통계적 방법을 이용하여 기계가 학습하게 하는 인공지능 기술

Deep Learning

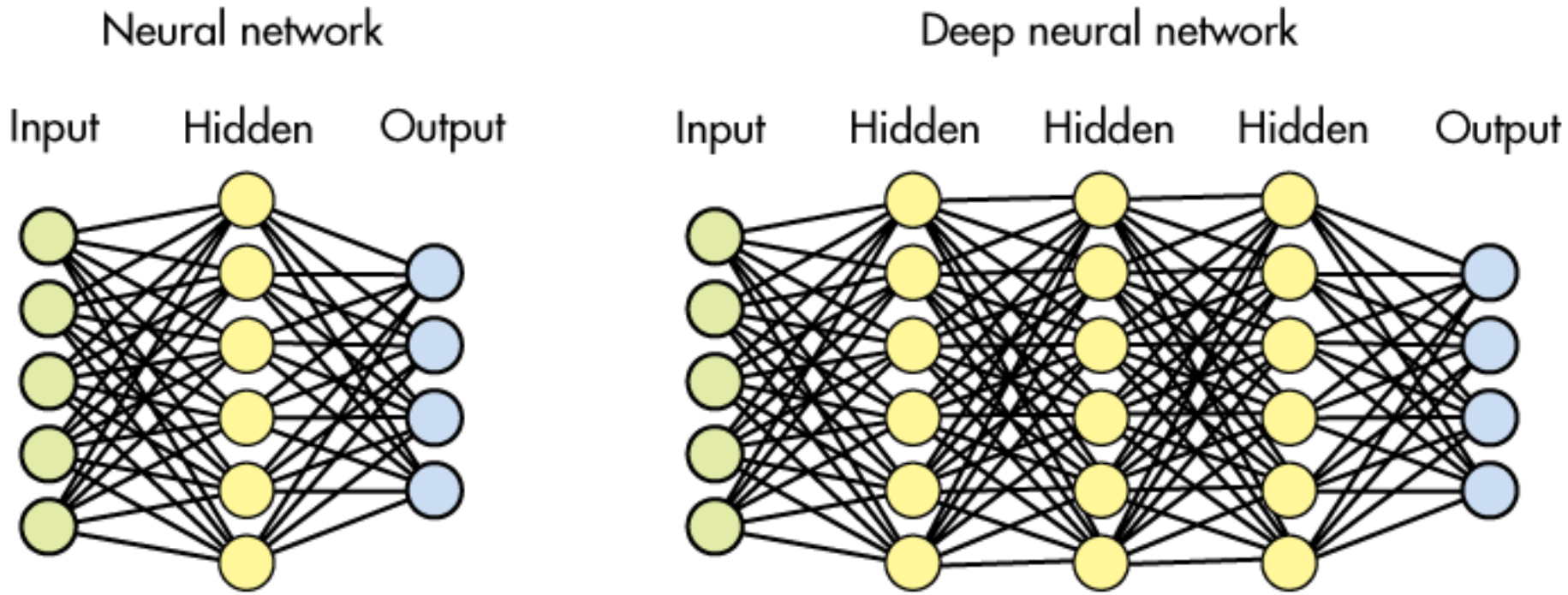
Subset of ML which make the computation of multi-layer neural networks feasible.
다중층 신경망을 이용하는 머신러닝기술



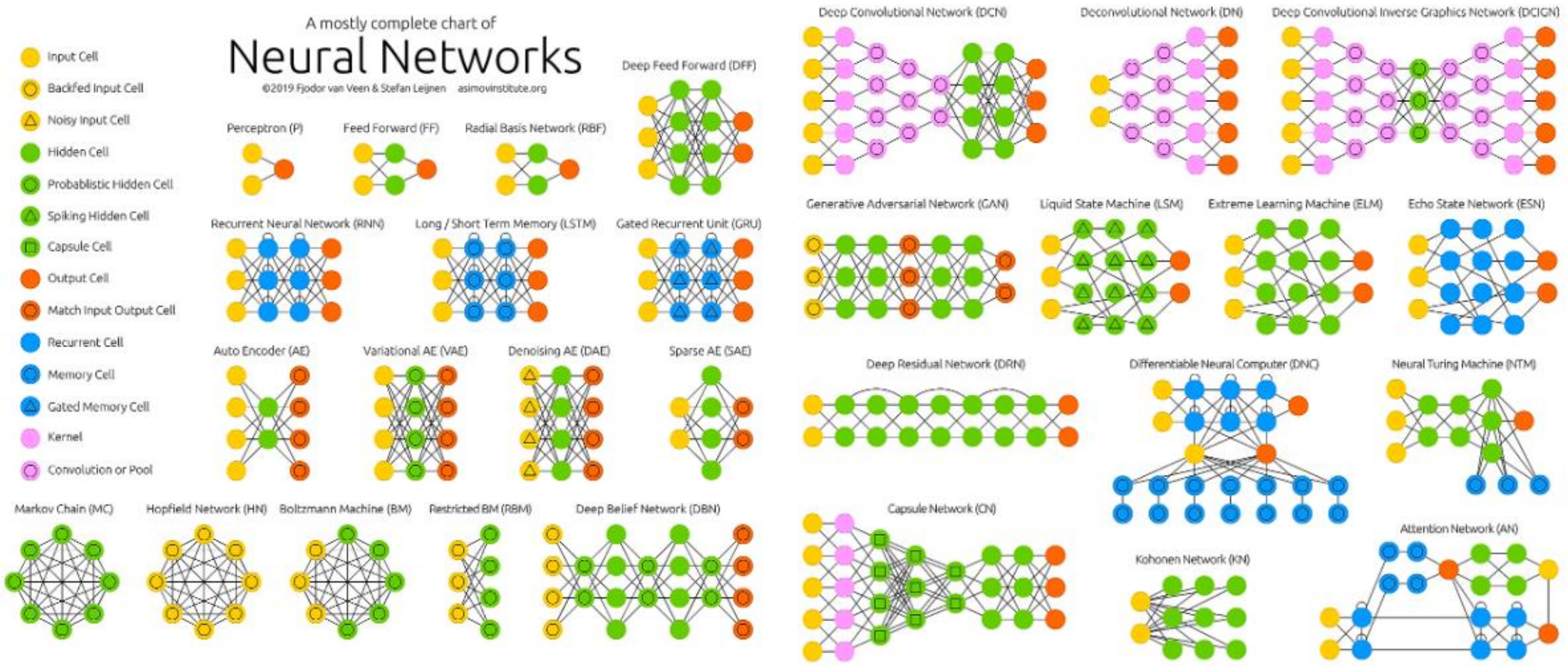

Neural Network
기계학습과 인지과학에서 생물학의 신경망(동물의 중추신경계중 특히 뇌)에서 영감을 얻은 통계학적 학습 알고리즘

Deep Learning 딥러닝

- Structures algorithms in layers to create an “artificial neural network” that can learn and make intelligent decisions on its own. 레이어를 이용한 인공적인 신경망을 구현하고 그것을 이용하여 의사결정을 내리는 알고리즘

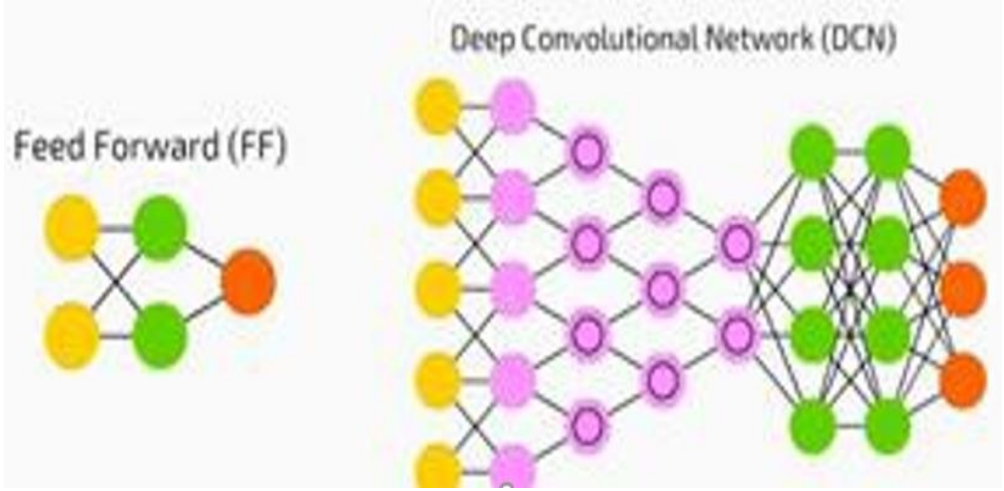


신경망 아키텍쳐의 발전


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

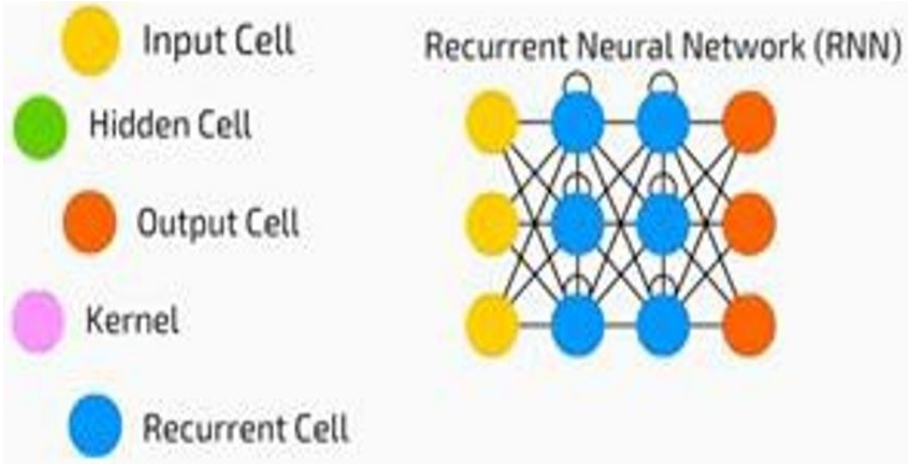
Advance in Deep Learning
딥러닝의 발전

- Convolutional Neural Network
  합성곱 신경망



- Uses convolutional neural networks for image processing. 이미지 처리를 위해 컨볼루셔널 신경망을 사용

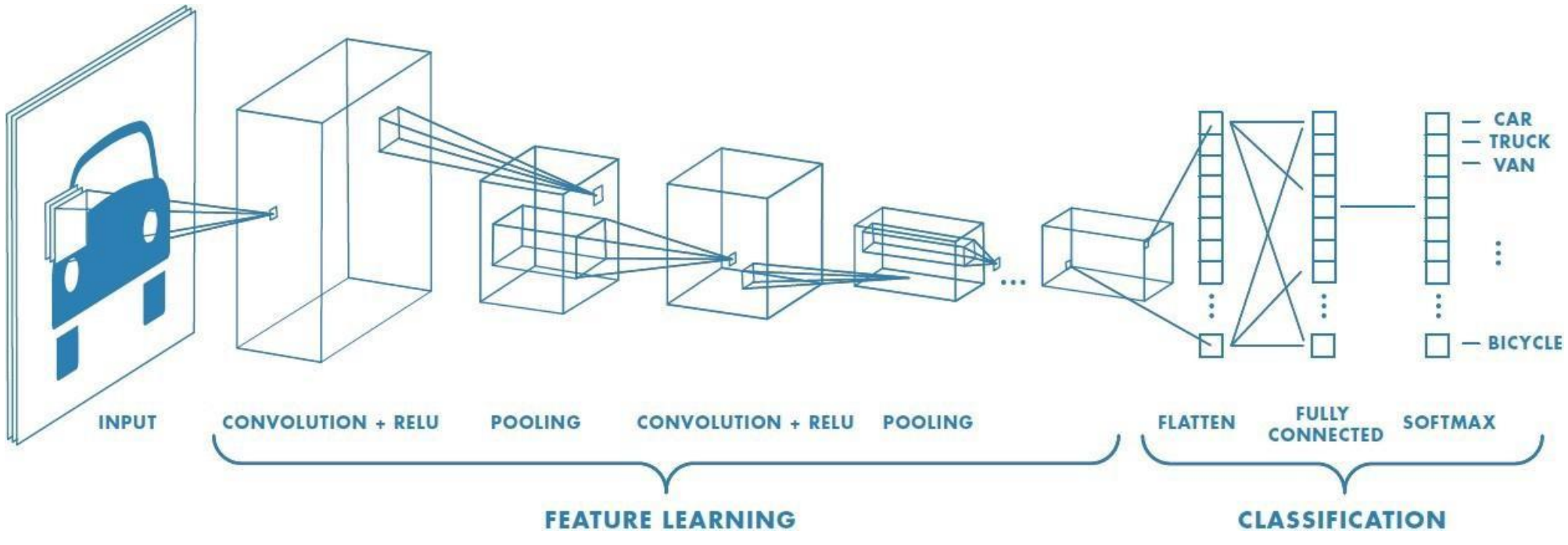
- Recurrent Neural Network
  순환 인공 신경망



- Uses recurrent neural networks for natural language processing. 자연어 처리를 위해 순환 신경망을 사용

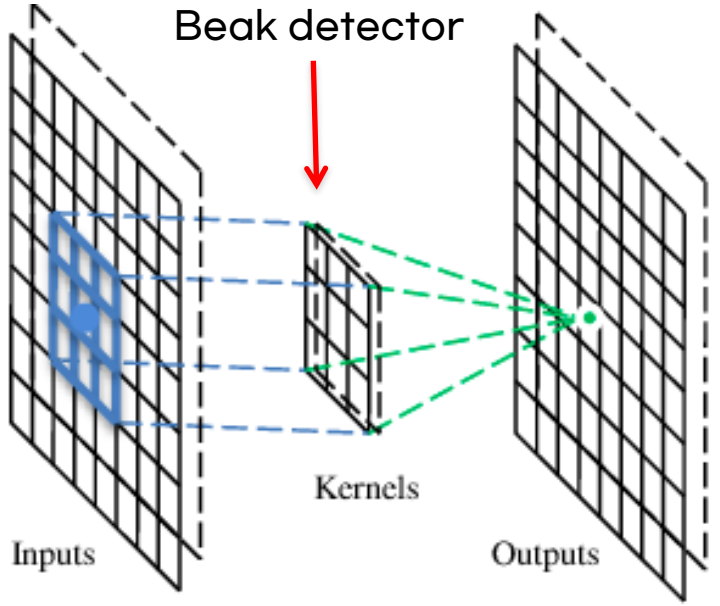
Convolutional Neural Network (CNN)
합성곱 신경망

- A neural network with some convolutional layers (and some other layers).
  컨볼루셔널 레이어(및 일부 다른 레이어)가 있는 신경망
- Convolutional Layers 합성곱층: Responsible for the convolutional operation in which feature maps identifies features in the images. 이미지 안에 있는 특성을 찾아내기 위해 컨볼루션 계산을 하는 층



Convolutional Layer 컨볼루션 레이어

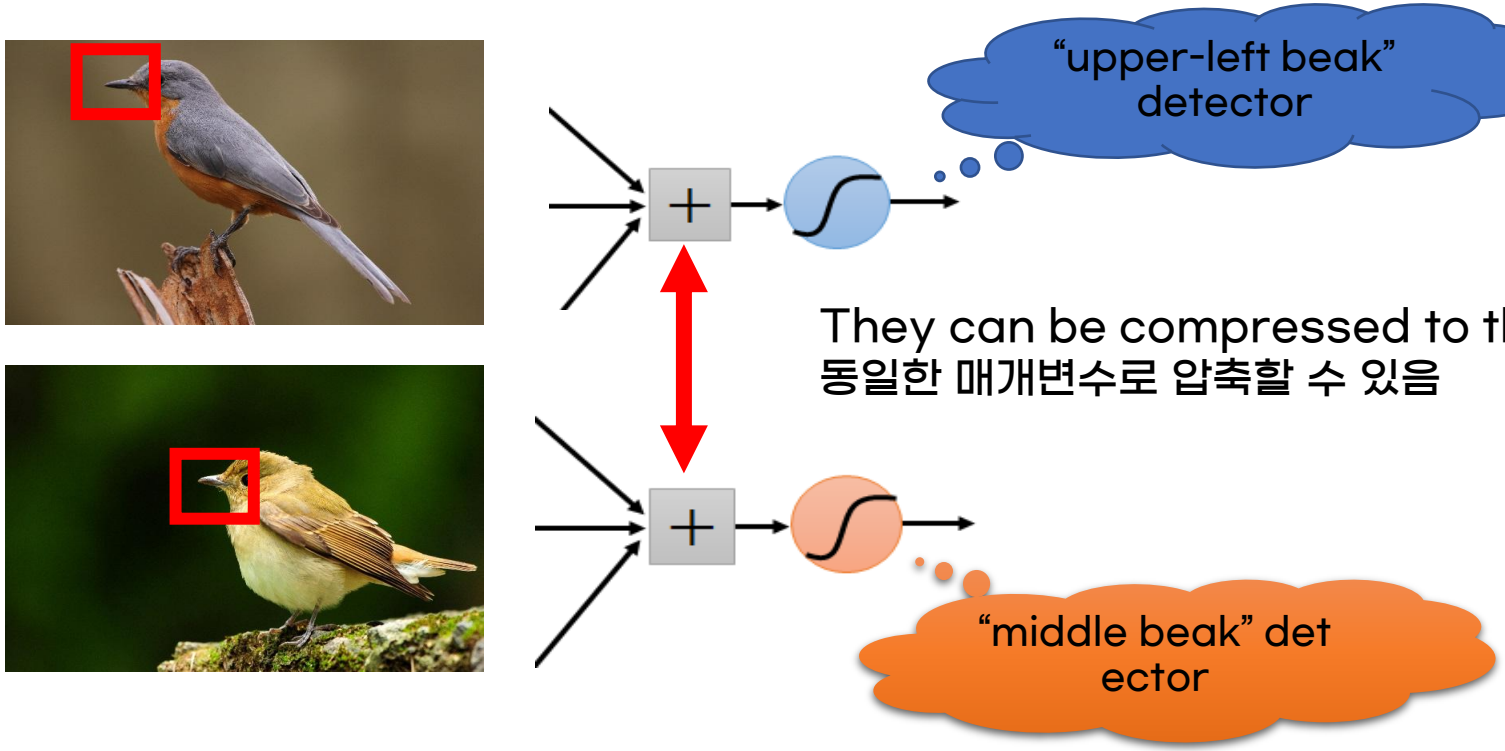
- A convolutional layer has a number of filters that does convolutional operation. 각 레이어에 컨볼루션 연산을 수행하는 필터들을 가짐



or Filters
or channels

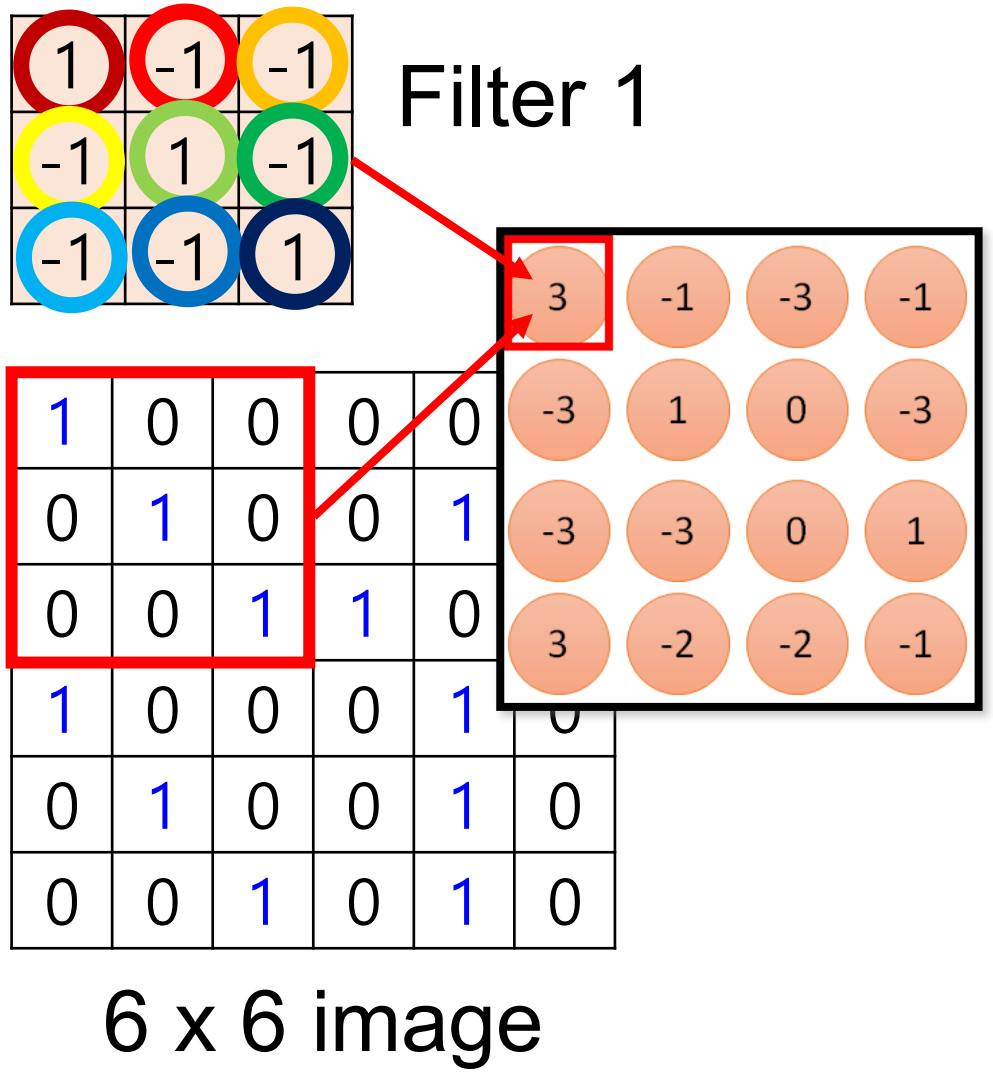
Image Learning 이미지 학습

- Same pattern appears in different places. 같은 패턴이 여러 장소에서 나타남
- Can represent a small region with fewer parameters. 작은 영역은 더 적은 파라미터로 표현될 수 있음
- Each small detector move around to identify features in the images. 이미지 안에 있는 특성을 찾아내기 위해 작은 부리 탐지기가 여기저기 움직임



Convolution Operation 컨볼루션 연산

- A convolution multiplies a matrix of pixels with a filter matrix or ‘kernel’ and sums up the multiplication values. 사진픽셀에 필터행렬을 곱해서 합침
- Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered. 필터가 다음 픽셀로 이동한 후 마지막 픽셀에 이를 때까지 똑같은 계산을 반복



Filter 필터

- Each filter detects a small pattern (3 x 3). 각 필터가 작은 패턴을 탐지

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

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Stride 폭

- stride=1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Dot product → 3 -1

6 x 6 image

- stride=2

1	-1	-1
-1	1	-1
-1	-1	1

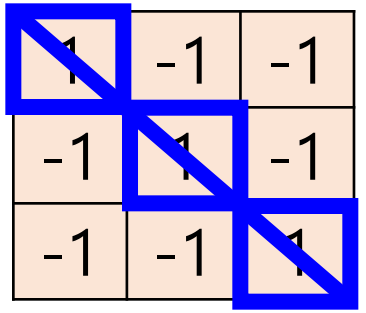
Filter 1

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

Dot product → 3 -3

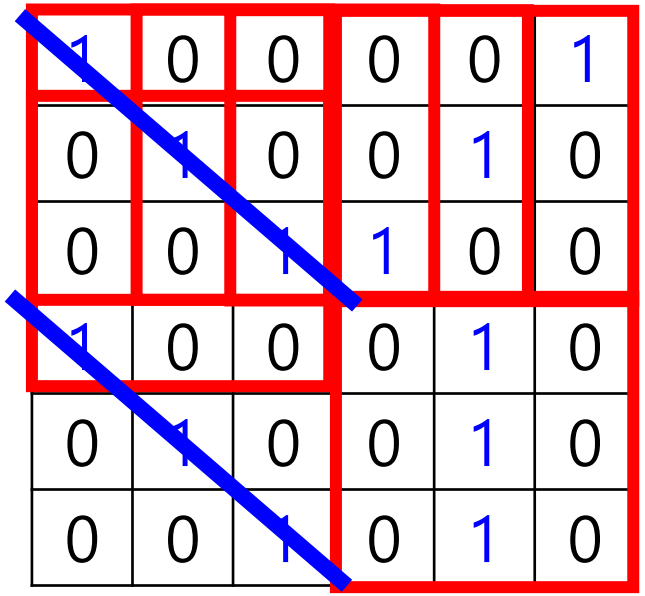
6 x 6 image

Feature Detection

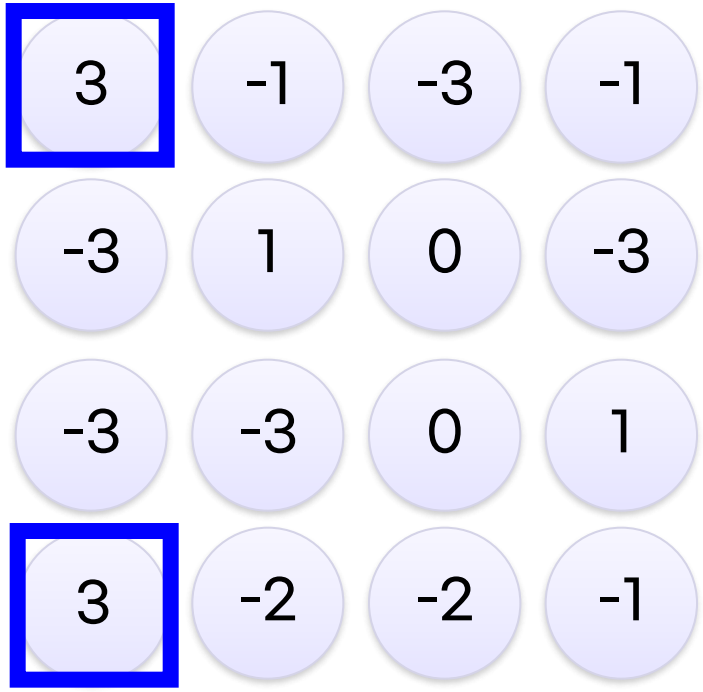


Filter 1

stride=1



6 x 6 image



Feature Map 속성맵

- Repeat this for each filter. 각 필터에 대해서 반복

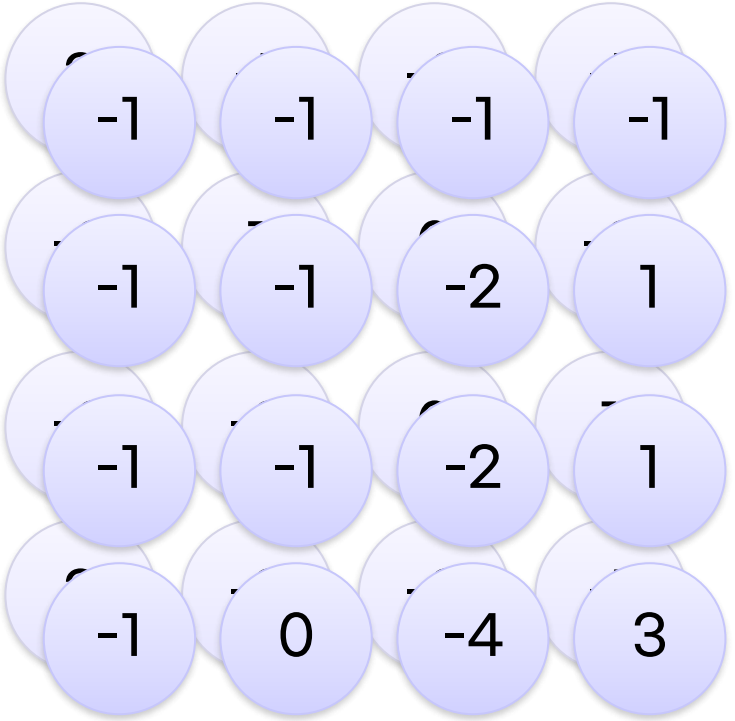
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

stride=1

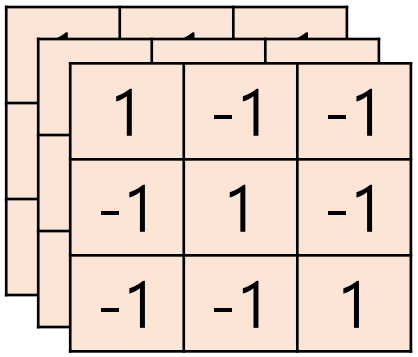
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

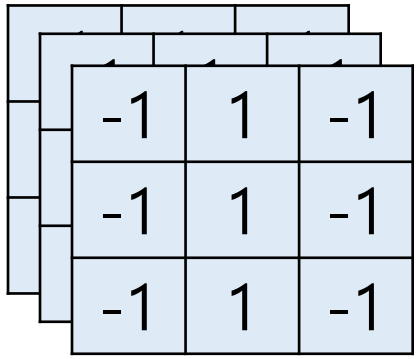


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

RGB 3 Channels

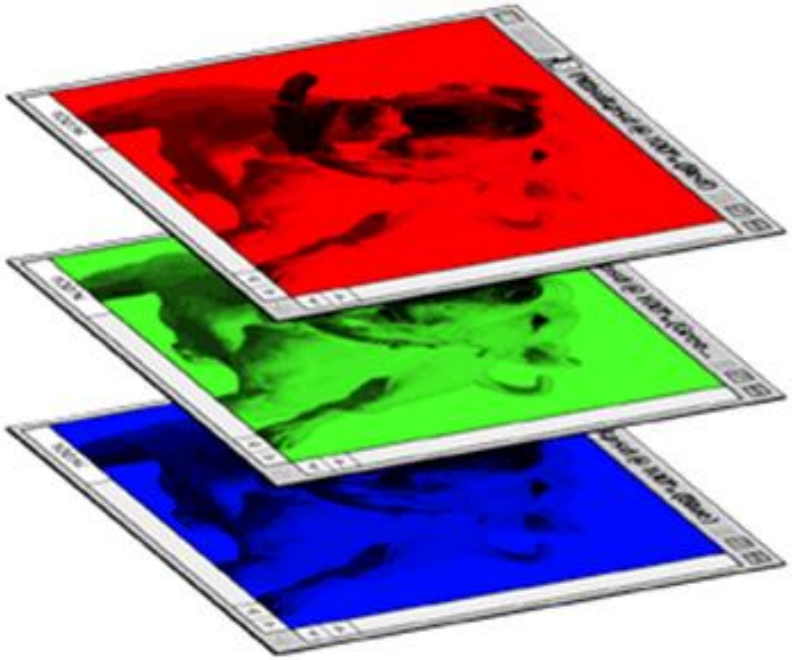


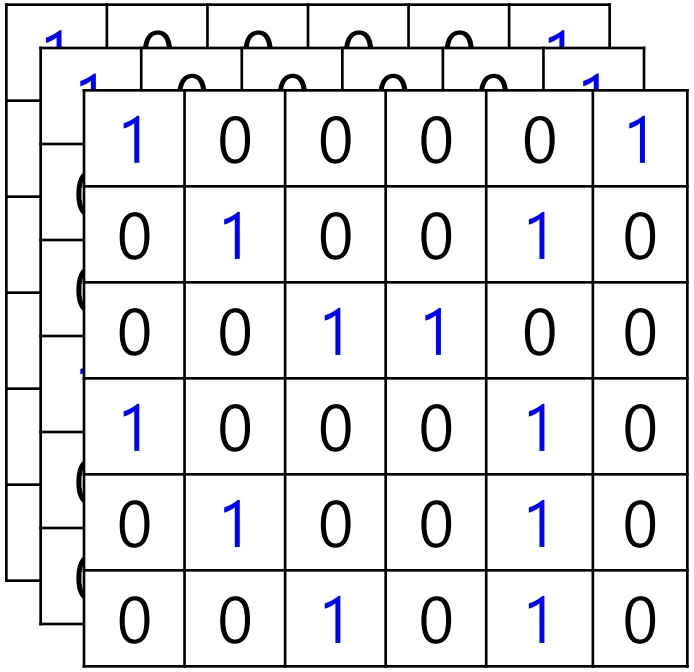
Filter 1



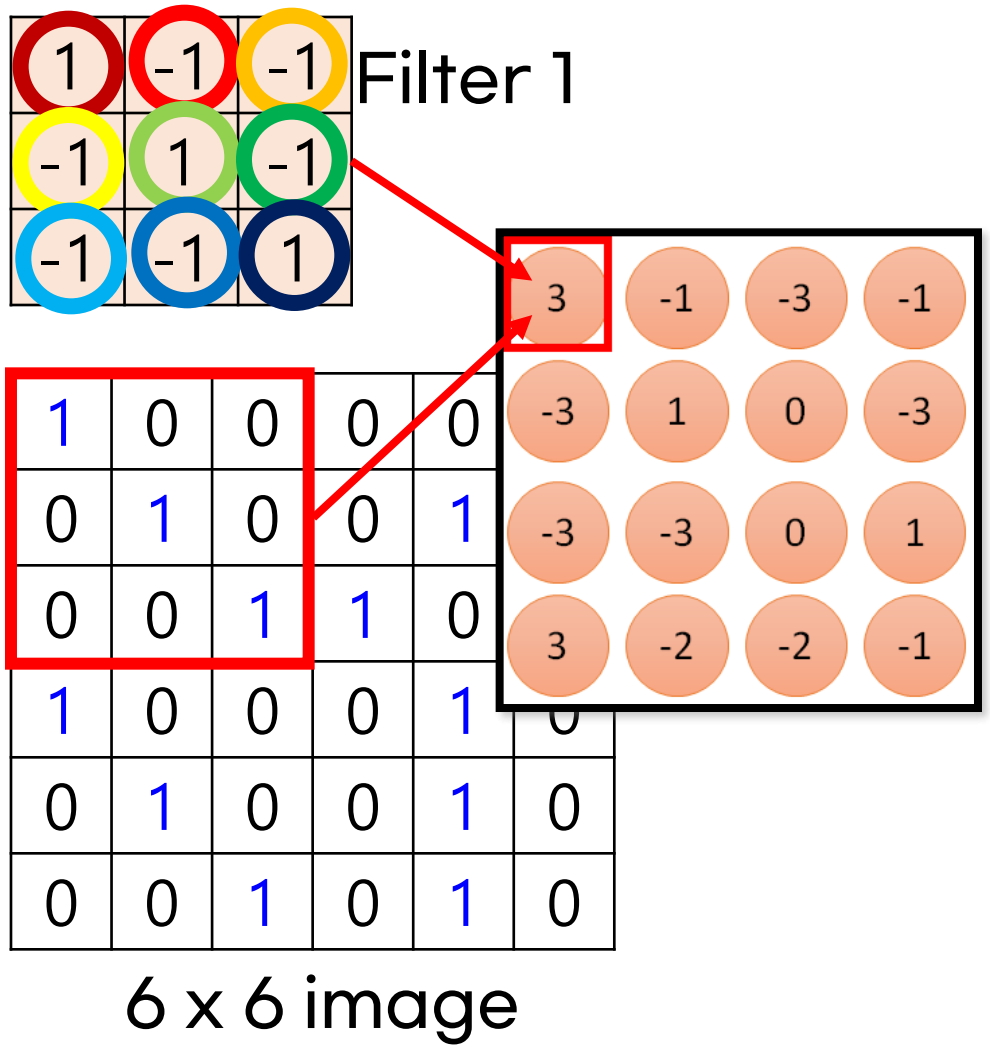
Filter 2

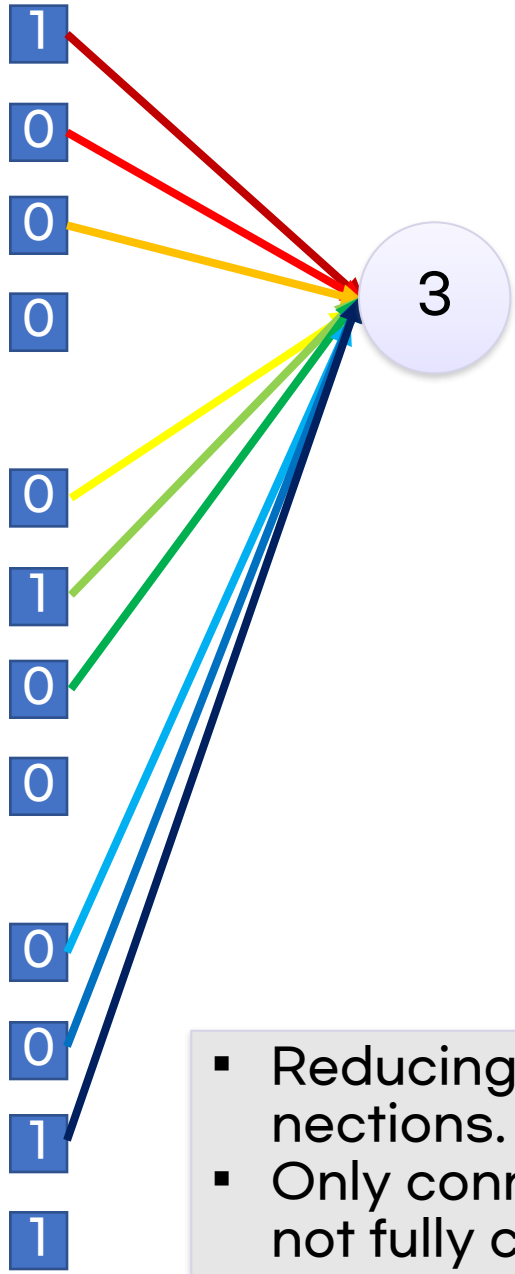
Color image





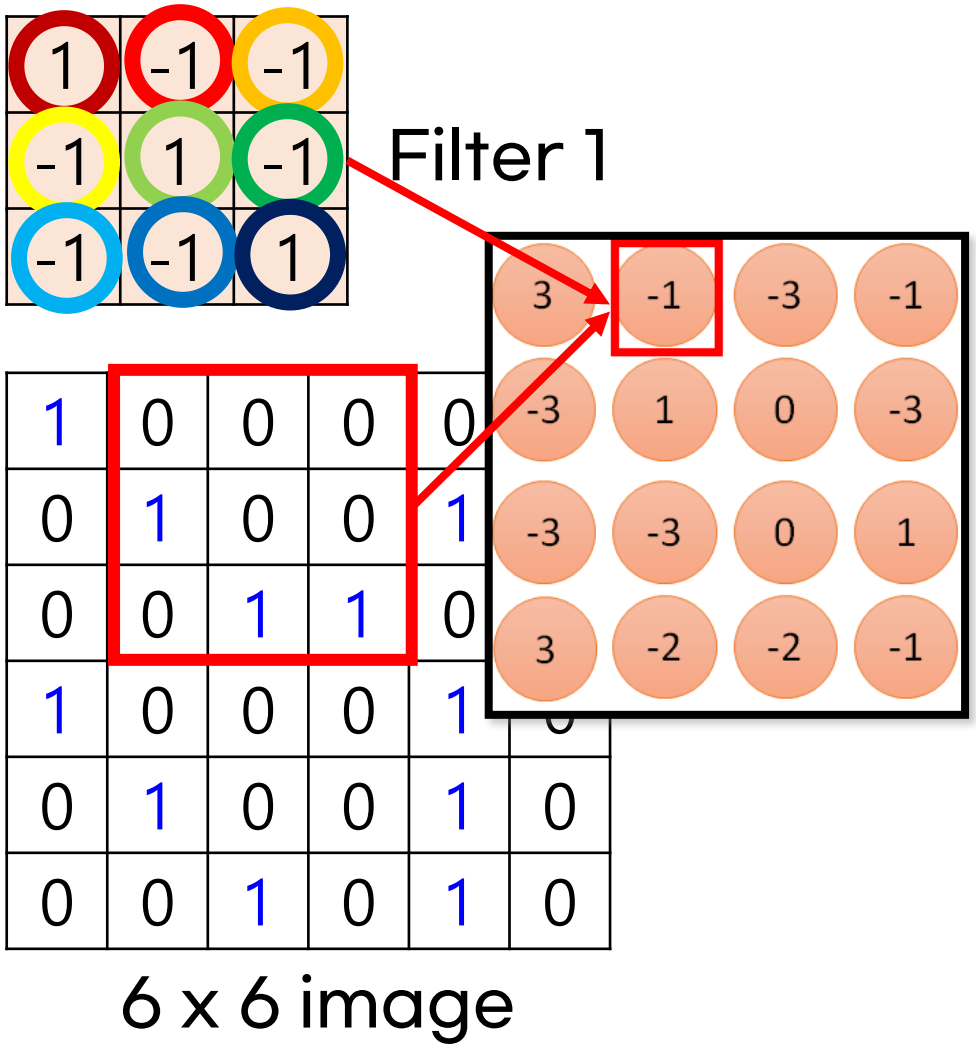
Convolutional Operation
컨볼루션 연산

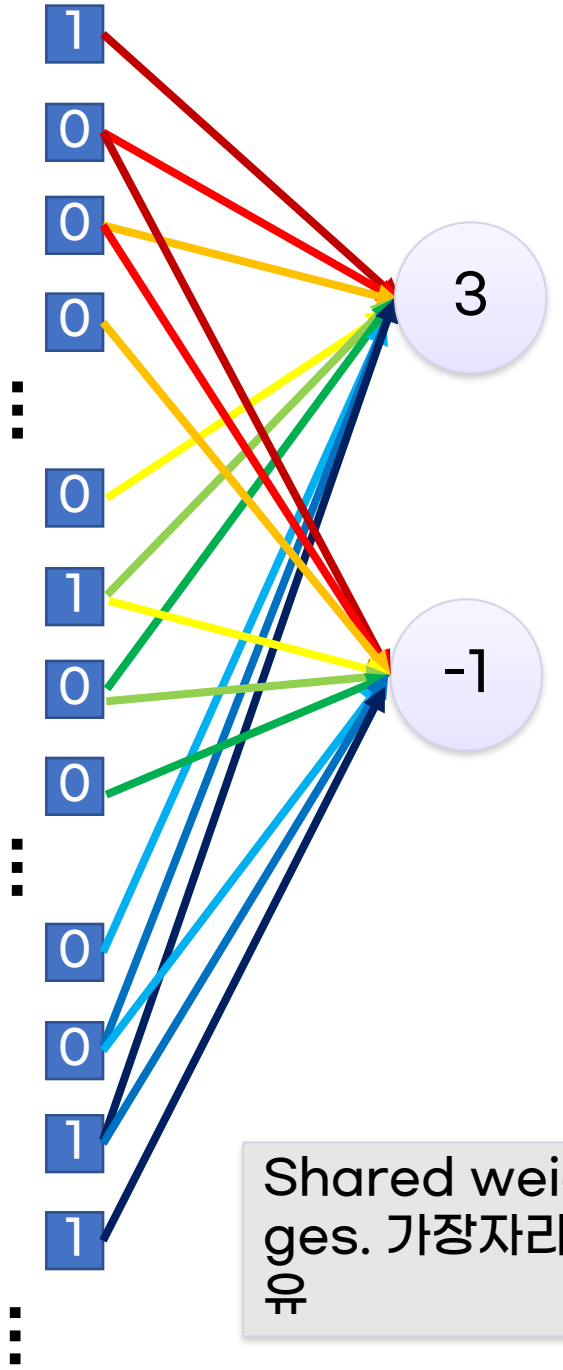




- Reducing number of con nections. 연결수를 줄임
- Only connect to 9 inputs, not fully connected. 9개의 입력값에만 연결

Convolutional Operation
컨볼루션 연산





Shared weights on the ed ges. 가장자리에서 가중치를 공유

Pooling 풀링

- Subsampling pixels will not change the object. 서브샘플링 픽셀은 객체를 변경하지 않음
- We can subsample the pixels to make image smaller ➔ fewer parameters to characterize the image. 이미지를 작게 만들기 위해서 픽셀을 서브샘플하면 이미지를 특징짓는 파라미터가 적어짐

bird

Subsampling

bird

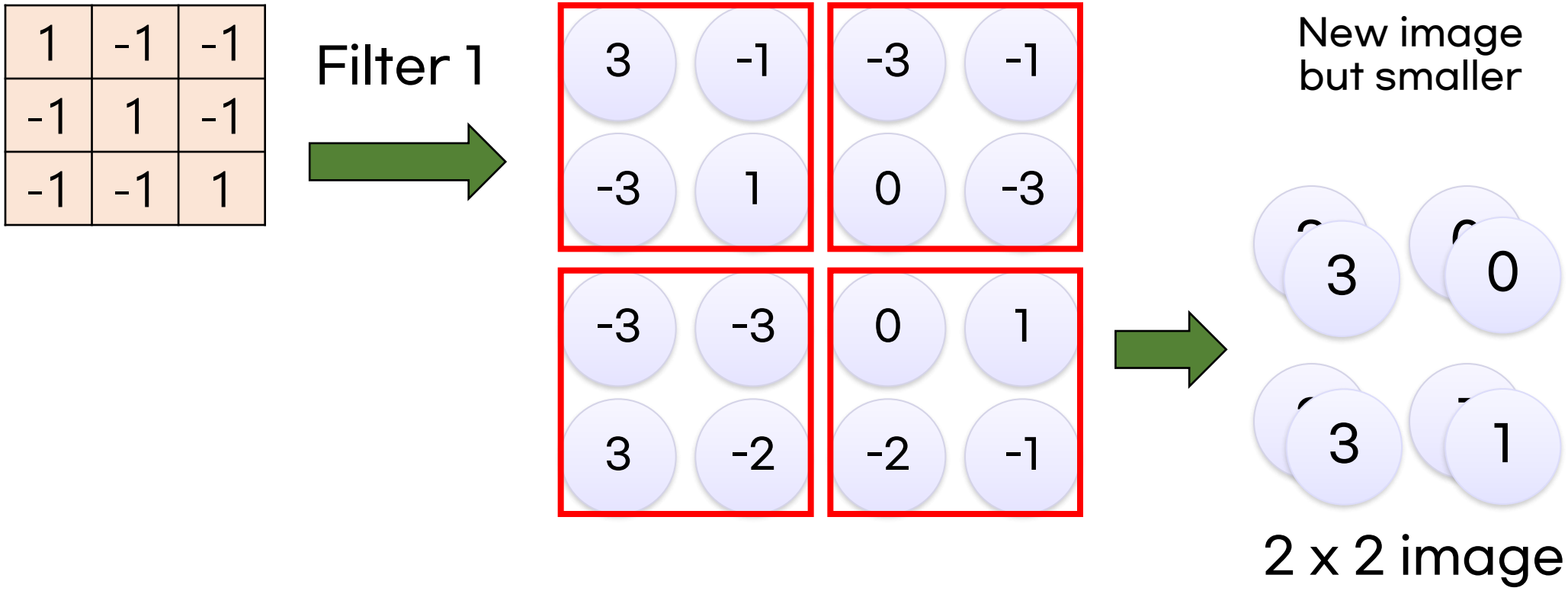
Pooling further reduces the complexity. 풀링으로 복잡도를 줄임

Pooling Layer 풀링 레이어

- Used to downsample the image as reducing the number of parameters. 계수의 수를 줄여서 이미지를 줄여나감
- Reduces the computation and avoids overfitting. 계산도 줄고 오버피팅도 방지
- Max Pooling—Selecting the maximum value. 사이즈를 정해서 그안에서 가장 큰값을 가지고 옴
- Average Pooling—Sum all of the values and dividing it by the total number of values. 전체를 다 더해서 갯수로 나눔

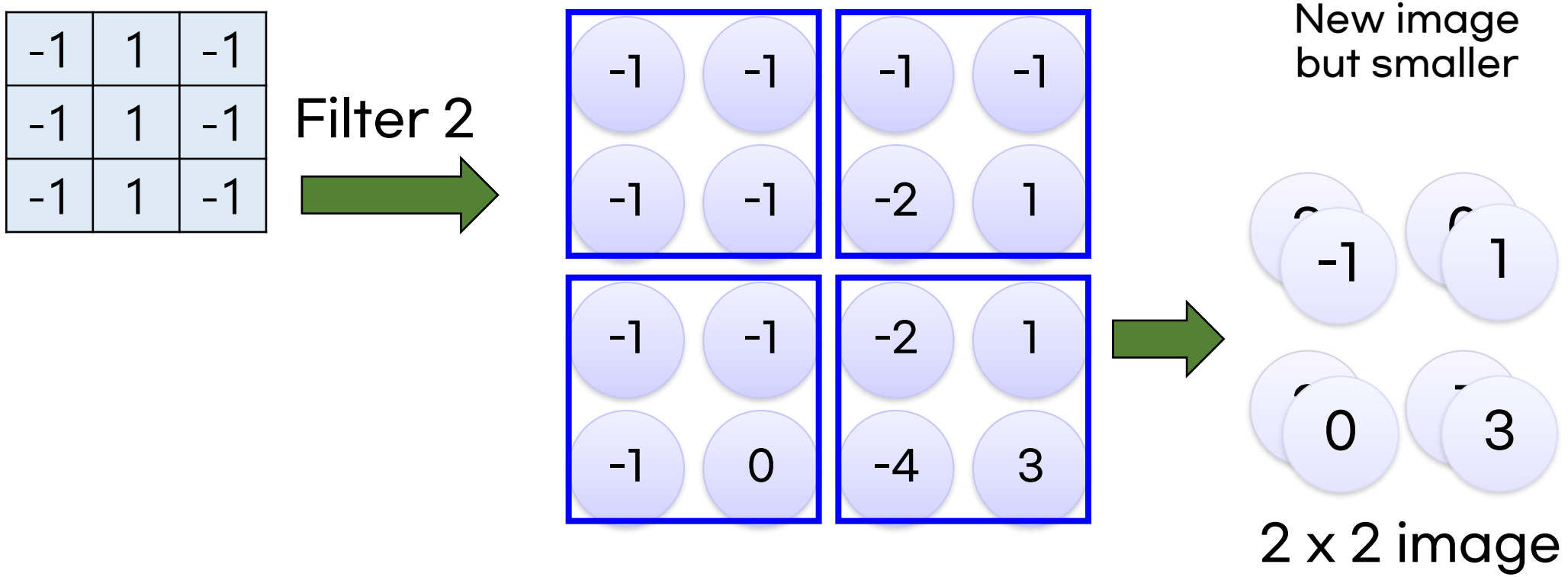
Max Pooling 맥스풀링

- Selecting the maximum value. 가장 큰 값을 선택



Max Pooling 맥스풀링

- Smaller than the original image. 원래 이미지보다 작음



Whole CNN

Convolution

Max Pooling

Convolution

Max Pooling

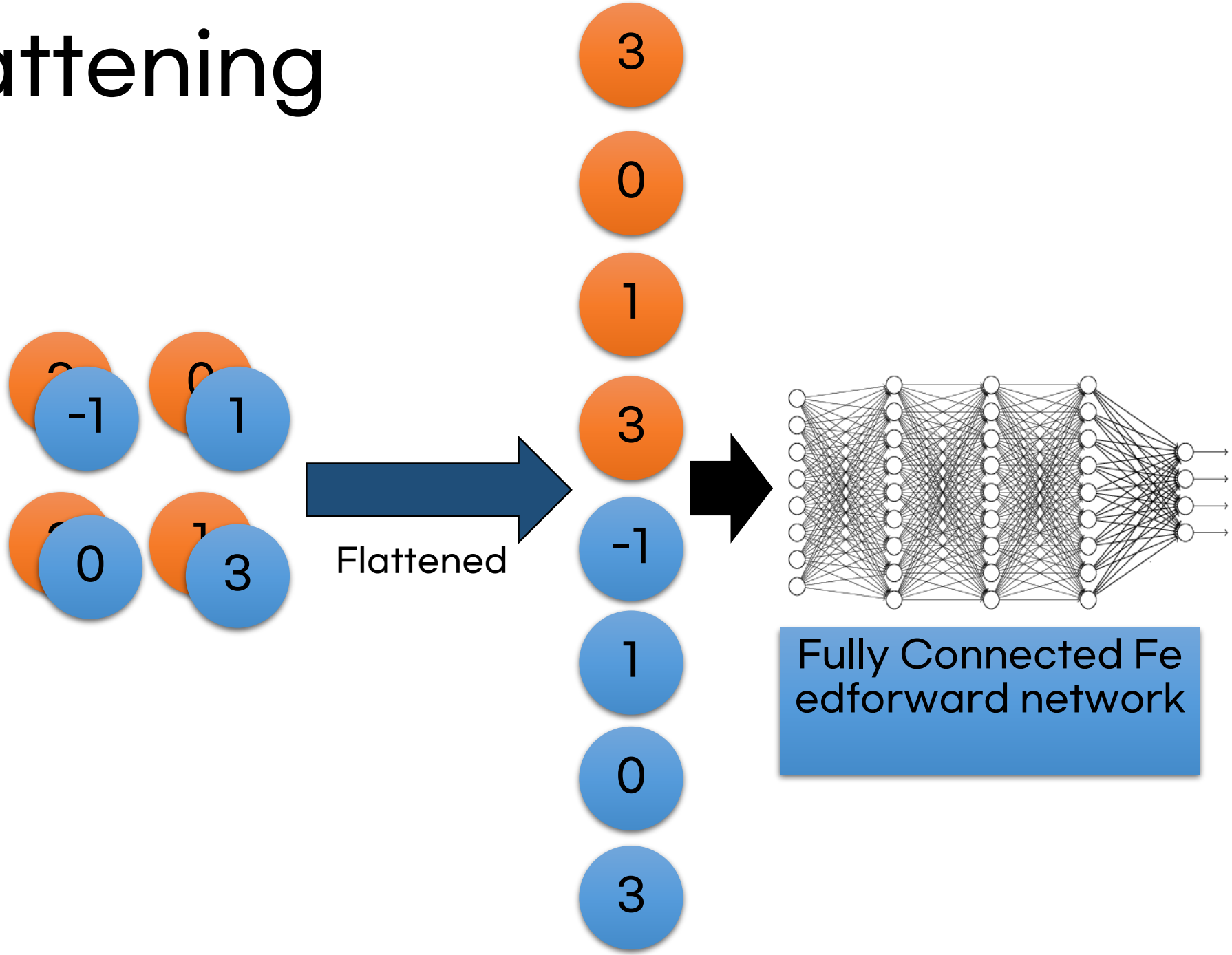
Can repeat many times.
여러 번 반복할 수 있음

output

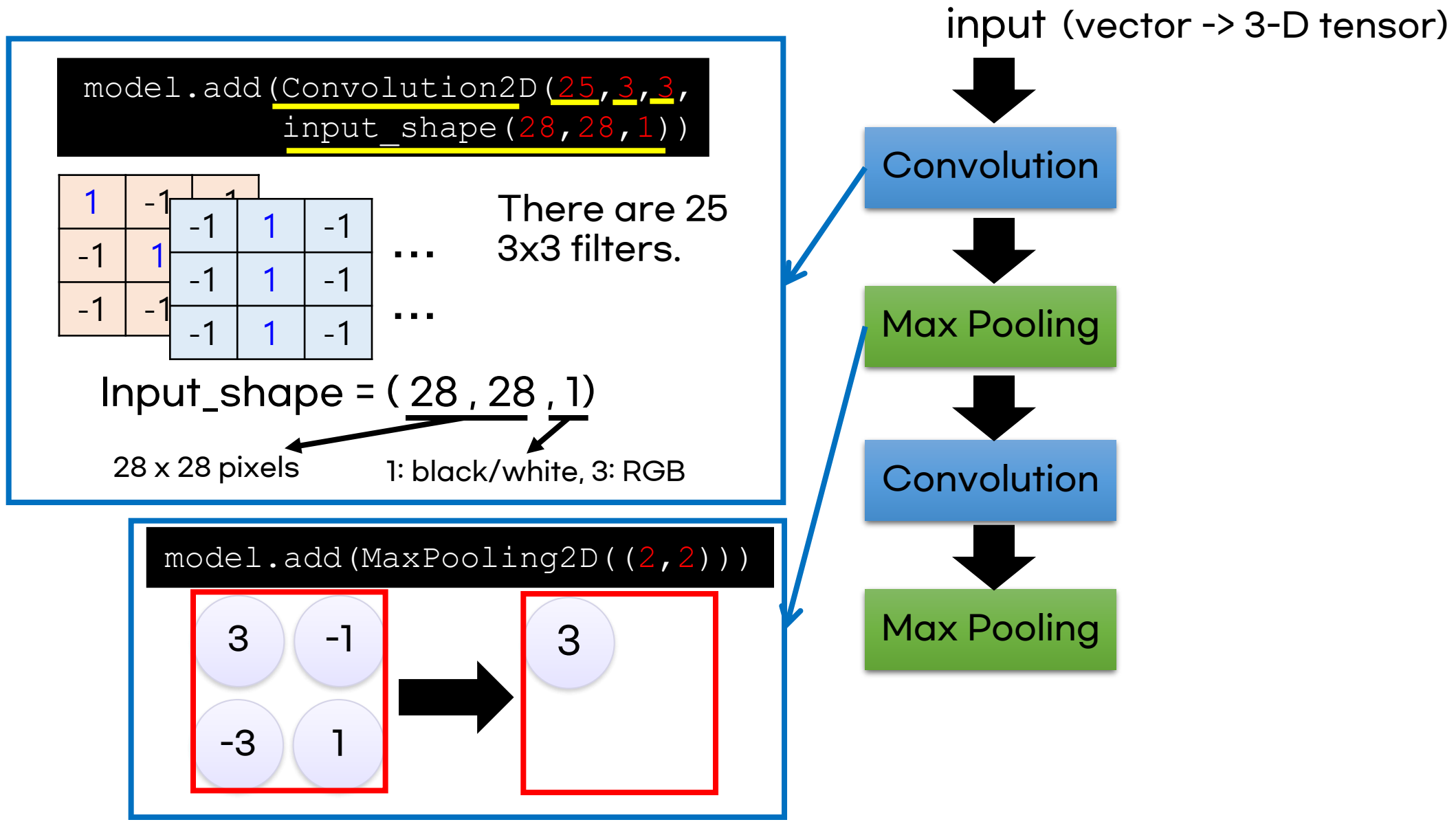
Fully Connected Feed forward network

Flattened

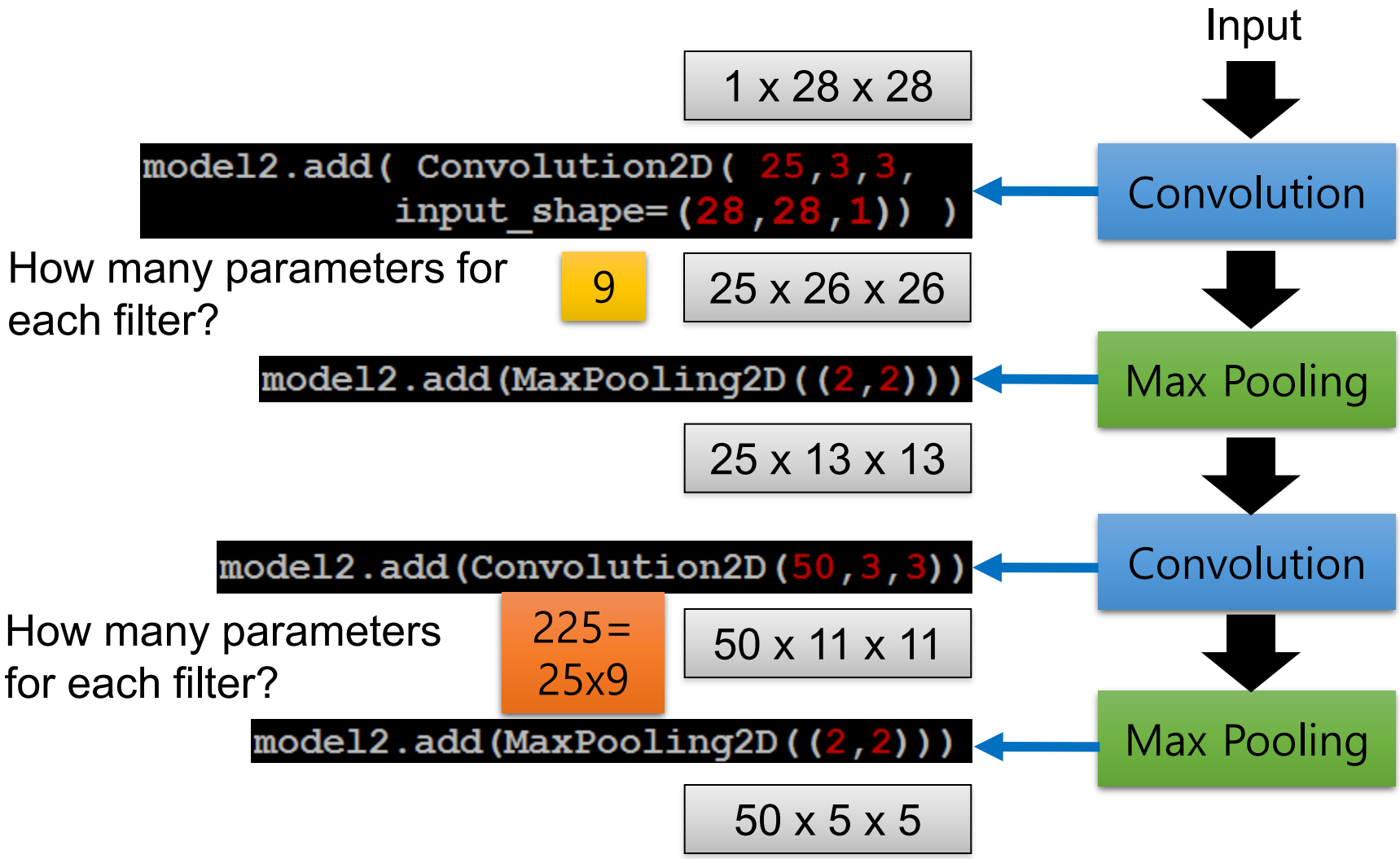
Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected Fe
edforward network

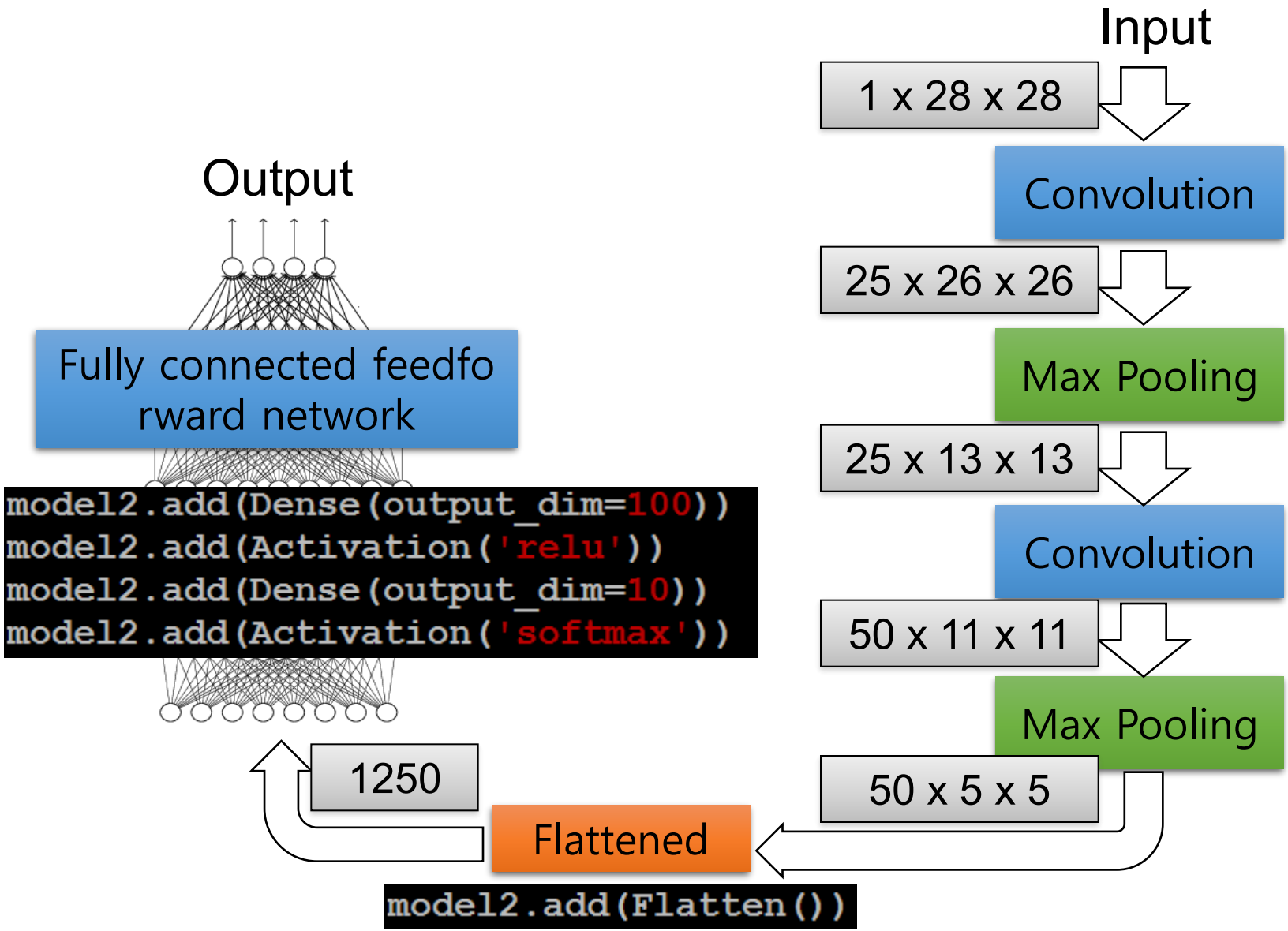
CNN in Keras


input (vector -> 3-D tensor)
model.add(Convolution2D(25,3,3,
input_shape(28,28,1))
There are 25
3x3 filters.
Input_shape = (28 , 28 , 1)
28 x 28 pixels
1: black/white, 3: RGB
model.add(MaxPooling2D((2,2)))
3
-1
-3
1
3
Convolution
Max Pooling
Convolution
Max Pooling

Convolution & Pooling Code

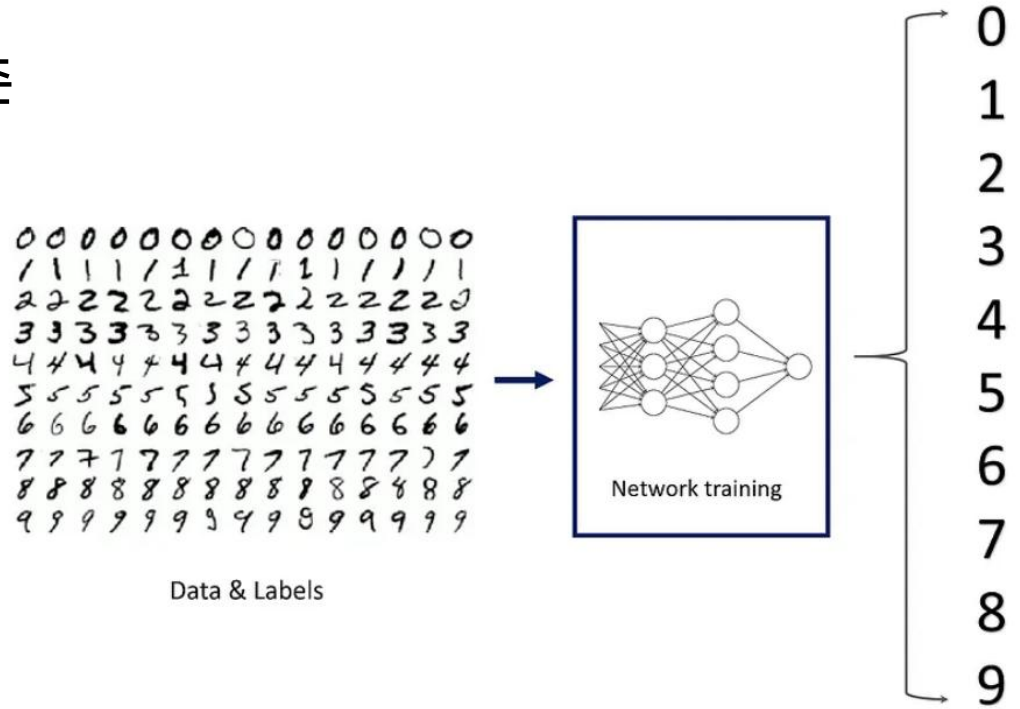


Flatten & Dense Code



MNIST Dataset

- Stands for Modified National Institute of Standards and Technology. 수정된 국가 표준 및 기술 협회의 약자
- A large collection of handwritten digits. 손으로 쓴 숫자의 모음
- 60,000 training datasets and 10,000 test sets. 60,000개의 훈련 데이터셋과 10,000개의 테스트셋으로 이루어짐.
- Numbers are standardized and centered in size in a fixed-size image (28x28 pixels) with values from 0 to 9. 숫자는 0에서 9까지의 값을 갖는 고정 크기 이미지 (28x28 픽셀)로 크기가 표준화되고 중심에 배치됨

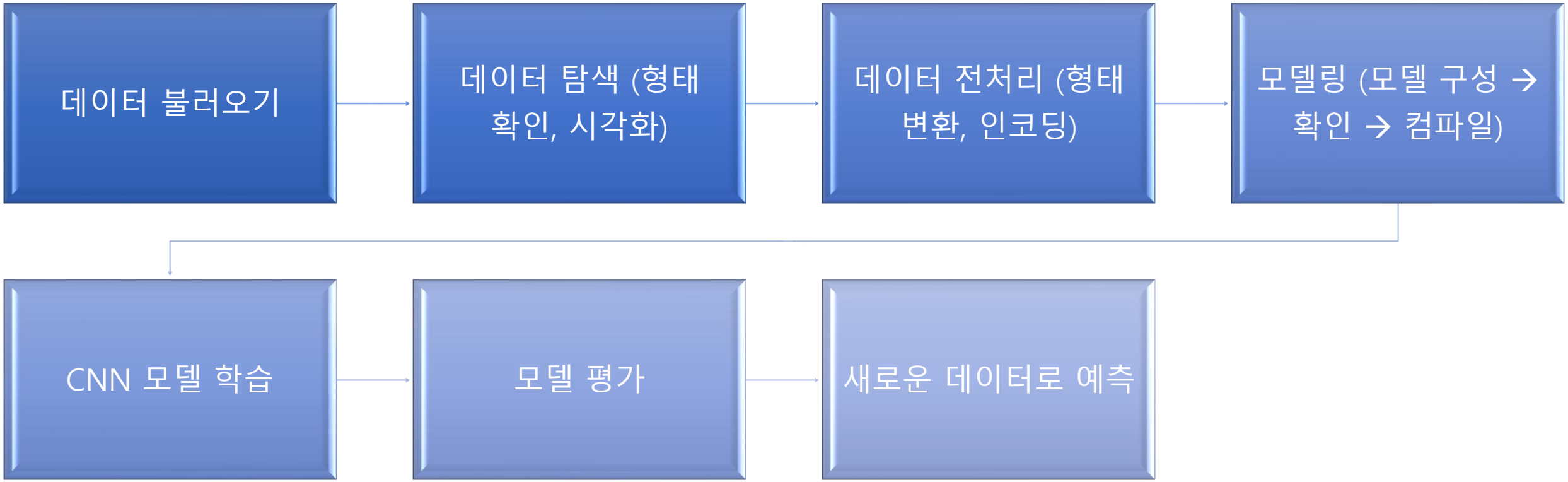


Lab #1

- 캐라스 투토리얼을 마치시오. Complete the Keras tutorial on https://medium.com/data-science/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5.

  - Evaluate the model (model.evaluate(x_train, y_train)).
  - Save the model and load again to predict the second test data (cnn_model.model).
  - Tensorboard and history graph

작업순서


데이터 불러오기
데이터 탐색 (형태 확인, 시각화)
데이터 전처리 (형태 변환, 인코딩)
모델링 (모델 구성 → 확인 → 컴파일)
CNN 모델 학습
모델 평가
새로운 데이터로 예측

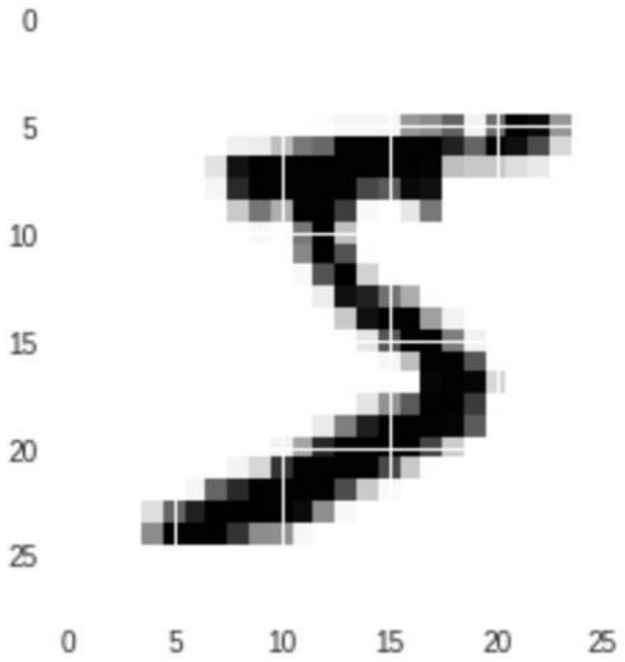
Loading Dataset 데이터 불러오기

```
from keras.datasets import mnist


#download mnist data and split into train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Exploratory Data Analysis 탐색분석

```
import matplotlib.pyplot as plt


#plot the first image in the
dataset

plt.imshow(X_train[0])

plt.show()


#check image shape
X_train[0].shape
```

```
<matplotlib.image.AxesImage at 0x7f83bcfbdc88>
```




```
(28, 28)
```

Data Pre-processing
데이터 전처리

```
#reshape data to fit model
X_train = X_train.reshape(60000,28,28,1)
X_test = X_test.reshape(10000,28,28,1)


#one-hot encode target column

from keras.utils import to_categorical

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_train[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

Building Model 모델 구축

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten

#create model
model = Sequential()

#add model layers
model.add(Conv2D(64, kernel_size=3, activation='relu',
input_shape=(28,28,1)))
model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

모델 확인

- 전체 구조를 요약

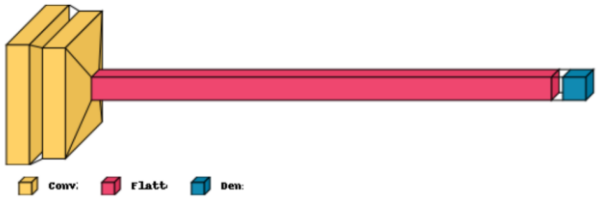
```
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 64)	640
conv2d_1 (Conv2D)	(None, 24, 24, 32)	18,464
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 10)	184,330

Total params: 203,434 (794.66 KB)
Trainable params: 203,434 (794.66 KB)
Non-trainable params: 0 (0.00 B)

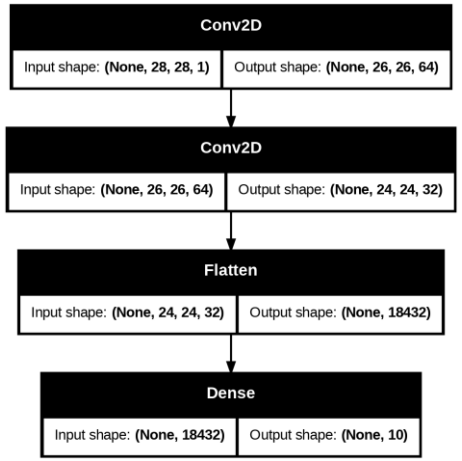
- 시각화 시켜서 구조 확인

```
!pip install visualkeras
import visualkeras
visualkeras.layered_view(model, legend=True)
```



- 이미지 파일로 시각화

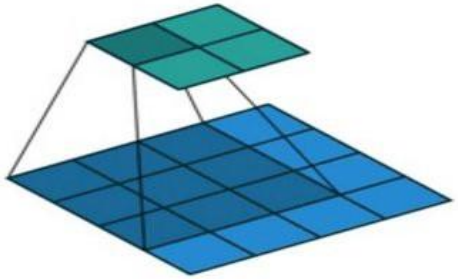
```
from tensorflow.keras.utils import plot_model
plot_model(model, 'model.png', show_shapes=True)
```



Convolutional Layer 컨볼류션 레이어

- Contains a set of filters that deal with input images, 2-dimensional matrices. 2차 메트릭스의 이미지를 처리하는 필터들을 포함

```
model.add(Conv2D(64, kernel_size=3, activation='relu',
input_shape=(28,28,1)))

model.add(Conv2D(32, kernel_size=3, activation='relu'))
```

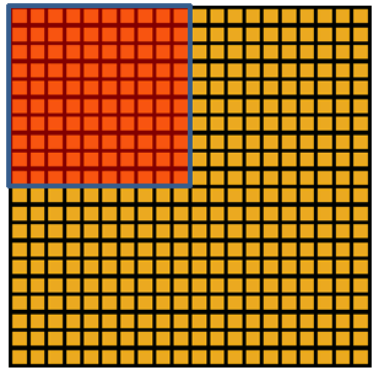


Convolutional Layer Options 옵션

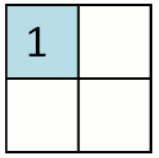
- 64 and 32 - the number of nodes in each layer, which can be adjusted depending on the size of the dataset. 데이터셋의 사이즈에 따라 각층에 있는 노드의 갯수를 조정
- Kernel size - the size of the filter matrix for the convolution (i.e., 3 is 3x3 filter matrix). 컨볼루션을 위한 필더 행렬의 크기
- Activation function - ReLU (Rectified Linear Activation)
- Input shape - only for the first layer. 1 indicates that the images are greyscale. 첫번째 레이어에서만 지정. 1은 이미지가 흑백이라는 의미

Pooling Layer 풀링 레이어

```
from keras.layers import MaxPooling2D
model.add(MaxPooling2D(pool_size=(2, 2)))
```

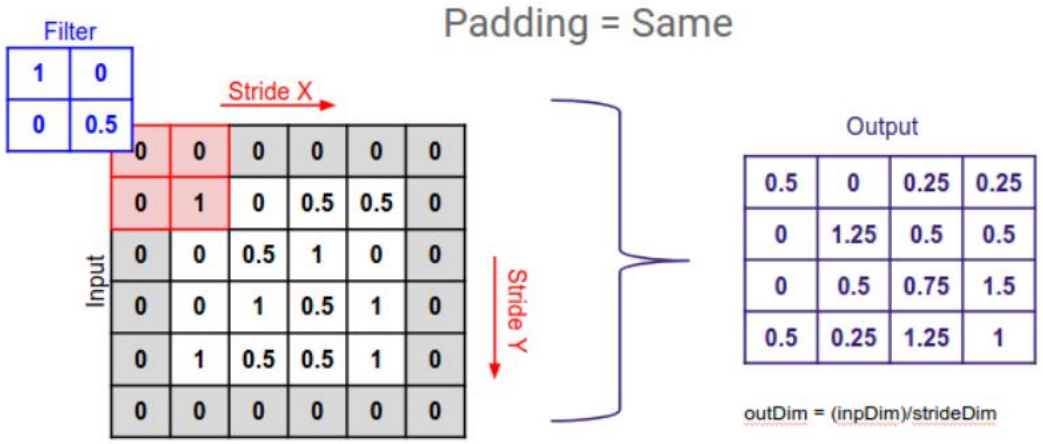


Convolved feature



Pooled feature

Padding 패딩



- An additional layer that added to the border of an image to overcome shrinking outputs and loosing information on corners of the image. 레이어 하나를 더해서 컨볼루션, 맥스풀링시 데이터가 손실되는 것 또는 코너의 데이터를 잃는 것을 방지

```
model.add(Conv2D(32, kernel_size=3, padding='same', activation='relu'))
```

Compiling Model 컴파일

```
#compile model using accuracy to measure model performance
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

- For one-hot encoded y, use categorical_crossentropy. 원핫 인코드된 y값은 카테고리컬 크로스엔트로피 사용
  - [1,0,0]
  - [0,1,0]
  - [0,0,1]

- For integer y, use sparse_categorical_crossentropy. 와이값이 정수이면 스팔스 카테고리컬 크로스엔트로피 사용
  - 1
  - 2
  - 3

Compile Options 컴파일 옵션

- Optimizer - controls the learning rate. The adam optimizer adjusts the learning rate throughout training. A smaller learning rate may lead to more accurate weights, but it takes longer time. 옵티마이저가 학습률을 조절. 적은 학습률 일수록 더 정확한 가중치를 찾아내지만, 오래 걸림. 아담 옵티마이저는 훈련하는 동안 학습률을 조절함
- Loss function - a lower score indicates that the model is performing better. Categorical_crossentropy is the most common choice for classification. 점수가 낮을수록 모델의 성과가 좋음. 분류문제에 가장 좋은 categorical_crossentropy

Training Model 모델 훈련

```
#train the model
history = model.fit(X_train, y_train, validation_data=(X_test,
y_test), epochs=3)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [==============================] - 22s 363us/step - loss: 1.3991 - acc: 0.8830 - val_loss: 0.0882 - val_acc: 0.9738
Epoch 2/3
60000/60000 [==============================] - 20s 334us/step - loss: 0.0712 - acc: 0.9790 - val_loss: 0.0874 - val_acc: 0.9729
Epoch 3/3
60000/60000 [==============================] - 20s 334us/step - loss: 0.0484 - acc: 0.9854 - val_loss: 0.0898 - val_acc: 0.9757
<keras.callbacks.History at 0x7fc38442e240>
```

Evaluate Model 모델 평가

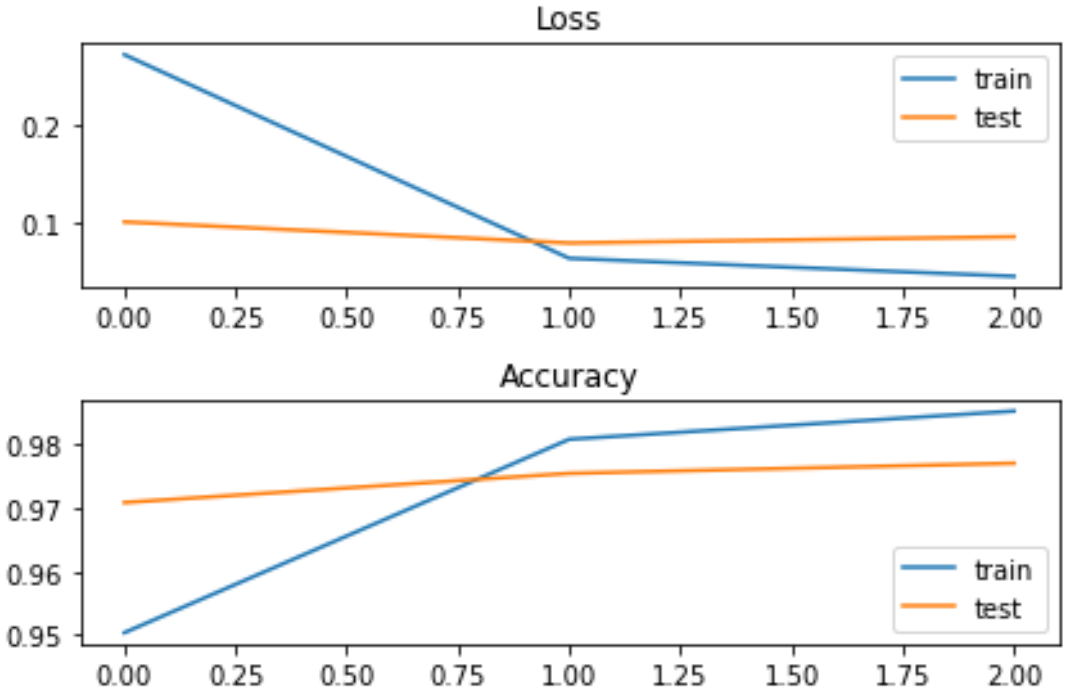
```
# 모델 평가
score = model.evaluate(x_test, y_test, verbose=0)
print(model.metrics_names[0], score[0])
print(model.metrics_names[1], score[1])
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9745 - loss: 0.1042
('compile_metrics', 0.9772999882698059)
```

```
# 정확도 계산
from sklearn.metrics import accuracy_score
y_pred = np.argmax(model.predict(x_test), axis=1)
accuracy_score(np.argmax(y_test, axis=1), y_pred)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
0.9773
```

Loss and Accuracy Graph
손실 및 정확도 그래프



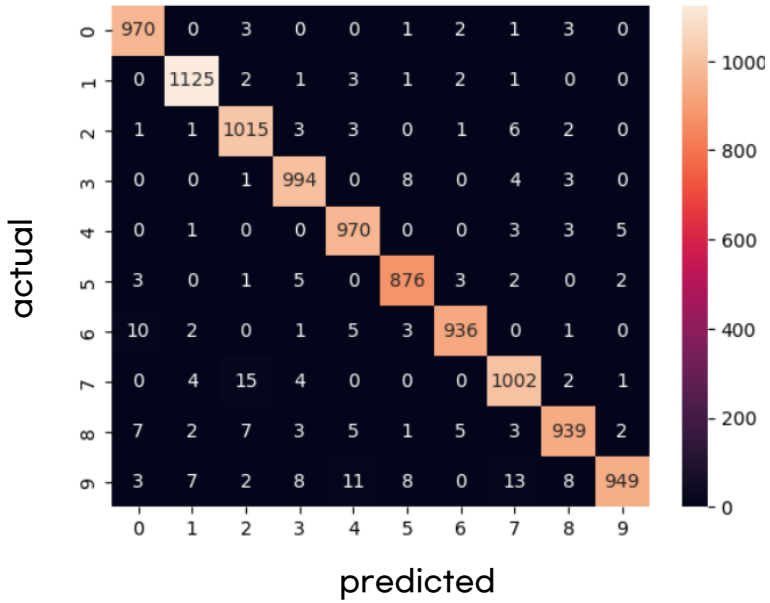

```
import matplotlib.pyplot as plt

plt.subplot(211)

plt.title('Loss')

plt.plot(history.history['loss'], label='train')

plt.plot(history.history['val_loss'], label='test')

plt.legend()

plt.subplot(212)

plt.title('Accuracy')

plt.plot(history.history['accuracy'], label='train')

plt.plot(history.history['val_accuracy'],
   label='test')

plt.legend()

plt.tight_layout()

plt.show()
```

Confusion Matrix 혼동행렬

```
import seaborn as sns

from sklearn.metrics import confusion_matrix

sns.heatmap(confusion_matrix(np.argmax(y_test, axis=1), y_pred),
annot=True, fmt='g')

plt.show()
```

Predicted 예측값

Actual 실제값	negative	positive
negative	TN (True Negative)	FP (False Positive)
positive	FN (False Negative)	TP (True Positive)



Classification Report

```
# 분류리포트

from sklearn.metrics import
classification_report

print(classification_report(np.a
rgmax(y_test, axis=1), y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.97	0.98	0.98	1032
3	0.98	0.98	0.98	1010
4	0.97	0.99	0.98	982
5	0.98	0.98	0.98	892
6	0.99	0.98	0.98	958
7	0.97	0.97	0.97	1028
8	0.98	0.96	0.97	974
9	0.99	0.94	0.96	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Precision and Recall 정밀도와 재현율

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

양성이라고 나온 사람 중에 진짜 양성

		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

진짜 양성인 사람 중에 양성이라고 나온 사람

		Predicted: Negative	Predicted: Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

F1-Score = 0 ---> Poor (P=0 or R=0)
F1-Score = 1 ---> Perfect (P=1 or R=1)

Make Predictions 예측

- Predicted 7, 2, 1 and 0 for the first four images. 처음 네개의 이미지를 7,2,1,0으로 예측

```
model.predict(X_test[:4])
```

```
array([[1.6117248e-09, 8.6684462e-16, 6.8095707e-10, 1.5486043e-08,
        6.2878847e-14, 1.2934288e-15, 1.1453808e-16, 9.9999928e-01,
        1.0626109e-08, 6.9729606e-07],
       [1.3555871e-07, 2.6465393e-06, 9.9999511e-01, 2.0351818e-08,
        1.9796262e-11, 1.6996018e-12, 2.1163373e-06, 1.2008194e-17,
        4.8792381e-10, 2.6086671e-13],
       [6.7238901e-08, 9.9785548e-01, 1.9031411e-04, 3.9194603e-08,
        1.2894072e-04, 1.5791730e-06, 1.2754040e-06, 4.1349044e-09,
        1.8221687e-03, 5.5910935e-08],
       [9.9999356e-01, 1.6909821e-12, 8.2496926e-10, 1.7359107e-11,
        1.7359230e-12, 1.8865266e-13, 6.4659162e-06, 2.3738855e-11,
        1.1319052e-08, 2.6948474e-08]], dtype=float32)
```

```
np.argmax(model.predict(X_test[:4]), axis=1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 365ms/step
array([7, 2, 1, 0])
```

- Actual results for first 4 images in test set. 테스트셋에서 처음 4개 이미지에 대한 실제결과

```
y_test[:4]
```

```
array([[0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

```
np.argmax(y_test[:4], axis=1)
```

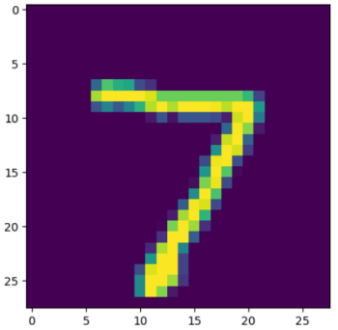
argmax

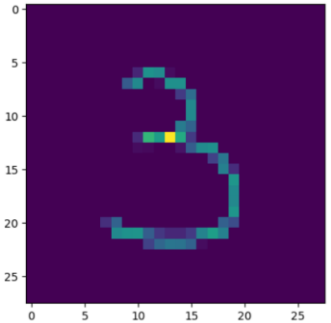
- Returns the indices of the maximum values along an axis. 가장 큰값의 인덱스넘버
- Only the first occurrence is returned if there is a same value appearing multiple times. 여러개의 최대값이 있으면 처음 나온 최대값의 인덱스

```
b = np.arange(6)
b[1] = 5
b #array([0, 5, 2, 3, 4, 5])
np.argmax(b) #1
a = np.arange(6).reshape(2,3) + 10
a #array([[10, 11, 12], [13, 14, 15]])
np.argmax(a) #5
```

이미지 파일로 예측하는 방법

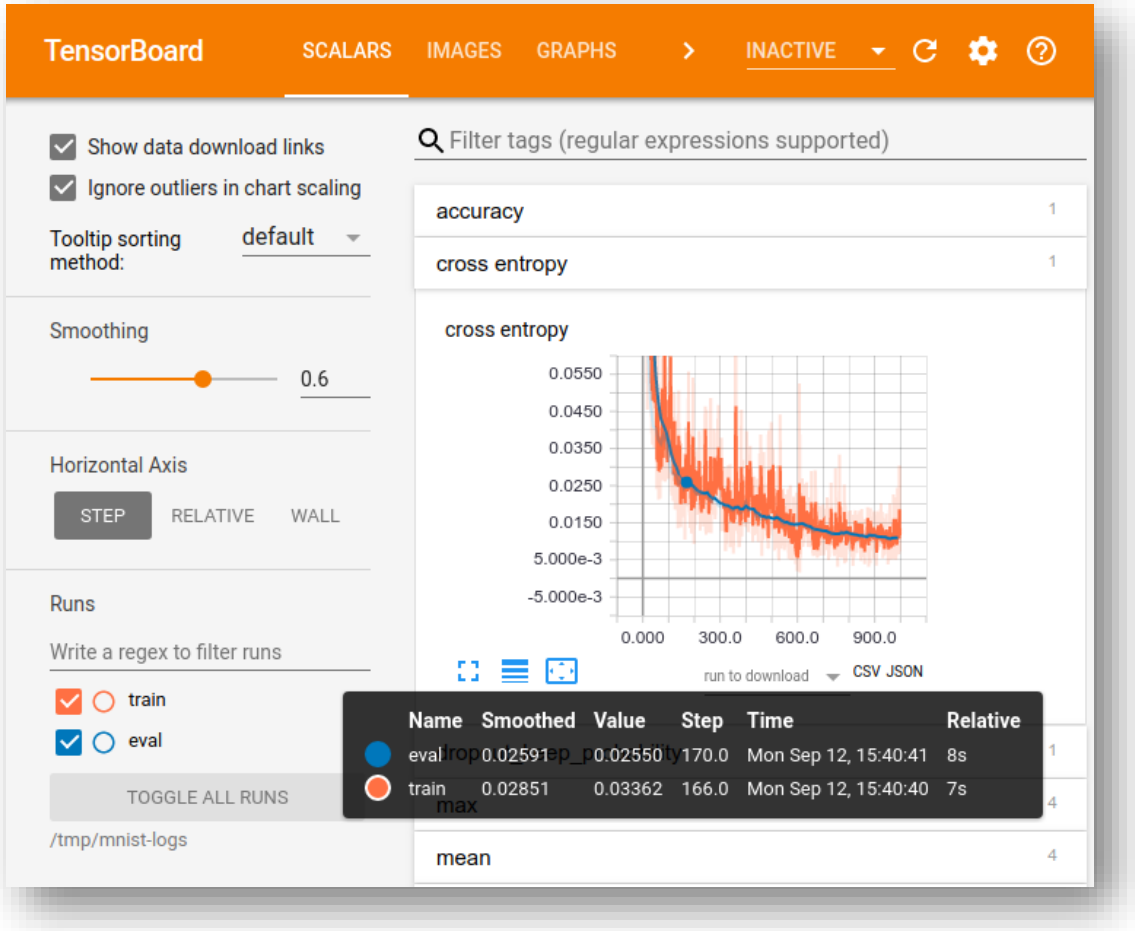
```
# 이미지 저장
from PIL import Image
import numpy as np

seven_array = Image.fromarray(x_test[0].reshape(28,28))
seven_array.save('seven.jpg')

# 테스트 이미지 준비하는 방법
seven = Image.open('seven.jpg').convert('L') #흑백변환
test_data = np.asarray(seven).reshape(1,28,28,1)

# 예측
np.argmax(model.predict(test_data), axis=1)
```





TensorBoard 텐서보드

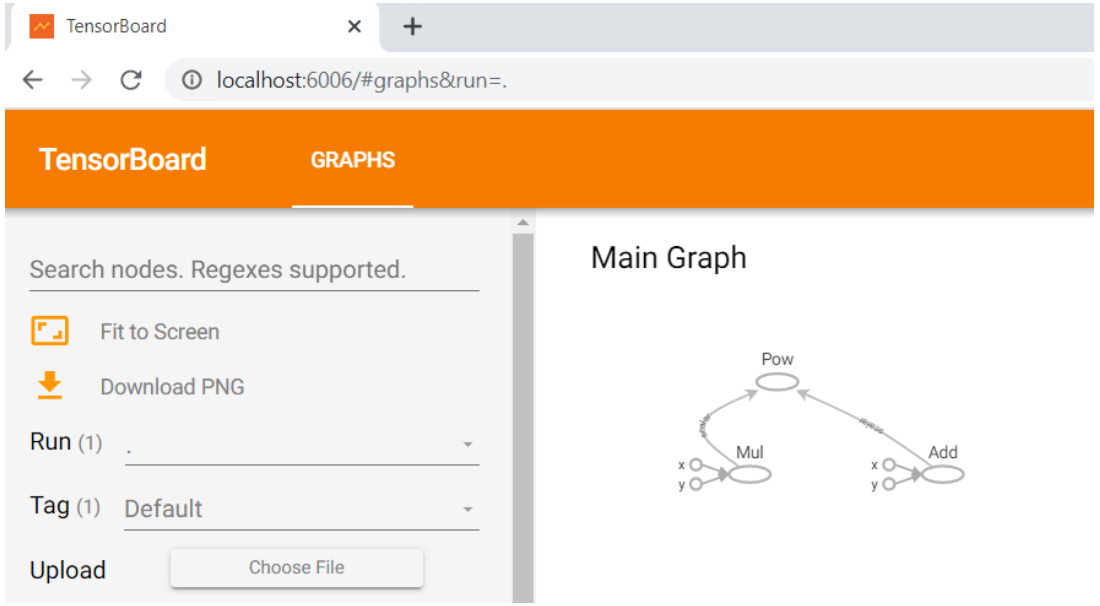
- A visualization software that comes with any standard TensorFlow installation. 텐서플로우 설치하면 같이 오는 시각화 소프트웨어



Visualizing Graph 그래프시각화

1

```
from keras.callbacks import
TensorBoard

from time import time

tensorboard =
TensorBoard(log_dir='logs\\{}'.for
mat(time()))

model.fit(X_train, y_train,
validation_data=(X_test, y_test),
epochs=3,callbacks=[tensorboard])
```



2

``명령프롬프트에서 텐서보드 실행

```
tensorboard --logdir=logs/
```

3

브라우저에 로컬호스트치고 텐서보드 들어감

```
http://localhost:6006
```

Save and Load Model 모델 저장 및 로딩

```
# 모델 저장할 때
Model.save('mnist_model.h5')
```

.h5와 .hdf5는 모두 Hierarchical Data Format (HDF) 파일의 확장자. Keras에서는 .h5 확장자를 사용

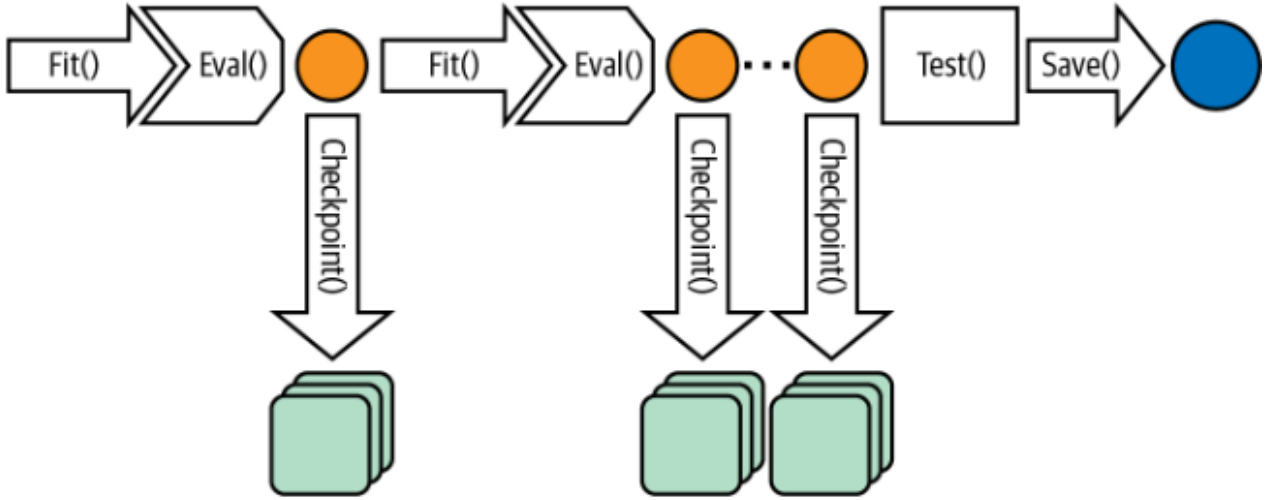
```
# 나중에 사용할 때 (컴파일 할 필요 없음)
from keras.models import load_model

new_model = load_model('mnist_model.h5')
pred = np.argmax(new_model.predict(X_test), axis=1)
pred[0]
```

Save and Load Weights
가중치 저장 및 로딩

```
# 저장할 때
model.save_weights('mnist_model.h5')

# 나중에 사용할 때 (반드시 컴파일 해서 사용
model.load_weights('mnist_model.h5')
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
pred = np.argmax(model.predict(X_test), axis=1)
pred[0]
```

ModelCheckpoint 가중치 저장

- Checkpointing saves the full model state at the end of every epoch. 체크포인트는 모든 Epoch가 끝날 때 전체 모델 상태를 저장
- You can setup to save the network weights only when there is an improvement in classification accuracy on the validation dataset. 테스트 데이터 세트의 정확도가 높아질 때만 가중치를 저장할 수 있게 할 수 있음



Checkpoint Code 체크 포인트 코드

```
from keras.callbacks import ModelCheckpoint


# checkpoint 객체 생성
ckpt_model=“mnist_weights.h5"
checkpoint = ModelCheckpoint(ckpt_model, monitor='val_accuracy',
verbose=1, save_best_only=True, mode='max’)


# Fit the model
model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10,
callbacks=[checkpoint], verbose=0)
```

Loading Checkpoint 가중치 로딩

```
# create model

…


# load weights

model.load_weights(“mnist_weights.h5")


# train the model (이어서 학습함)

history = model.fit(X_train, y_train, validation_data=(X_test,
  y_test), epochs=3)
```

ChatGPT로 디버깅 및 개선 연습

- 다음 코드에 대한 개선된 코드 받아보기. 해설도 요청하세요

```
# 잘못된 코드
test_img = Image.open('seven.jpg').convert('L')
test_array = np.array(test_img).reshape(28, 28, 1)
prediction = model.predict(test_array)
```

- 출력 결과

❗ 문제점

- model.predict()는 입력으로 **4차원 텐서(batch 포함)**를 기대합니다.
  - 기대하는 입력 shape: (1, 28, 28, 1) ← 배치가 1장인 이미지
- 하지만 위 코드는 3차원: (28, 28, 1) → 그래서 ValueError 발생
- 또한 **정규화(0255 → 01)**가 되어 있지 않으면 예측 정확도가 떨어질 수 있습니다.

ChatGPT로 디버깅 및 개선 연습

- 다음 코드에 대한 개선된 코드 받아보기. 해설도 요청하세요

```
model = Sequential()
model.add(Dense(128, input_shape=(784,),
activation='relu'))
model.add(Dense(10, activation='softmax'))
```

- 출력 결과

❗ **한계점**

- 이미지의 공간 구조(위치, 패턴 등)를 고려하지 못함 → 성능이 CNN보다 떨어짐
- 이미지를 평탄화(Flatten)해야 함 → 전처리가 필요함