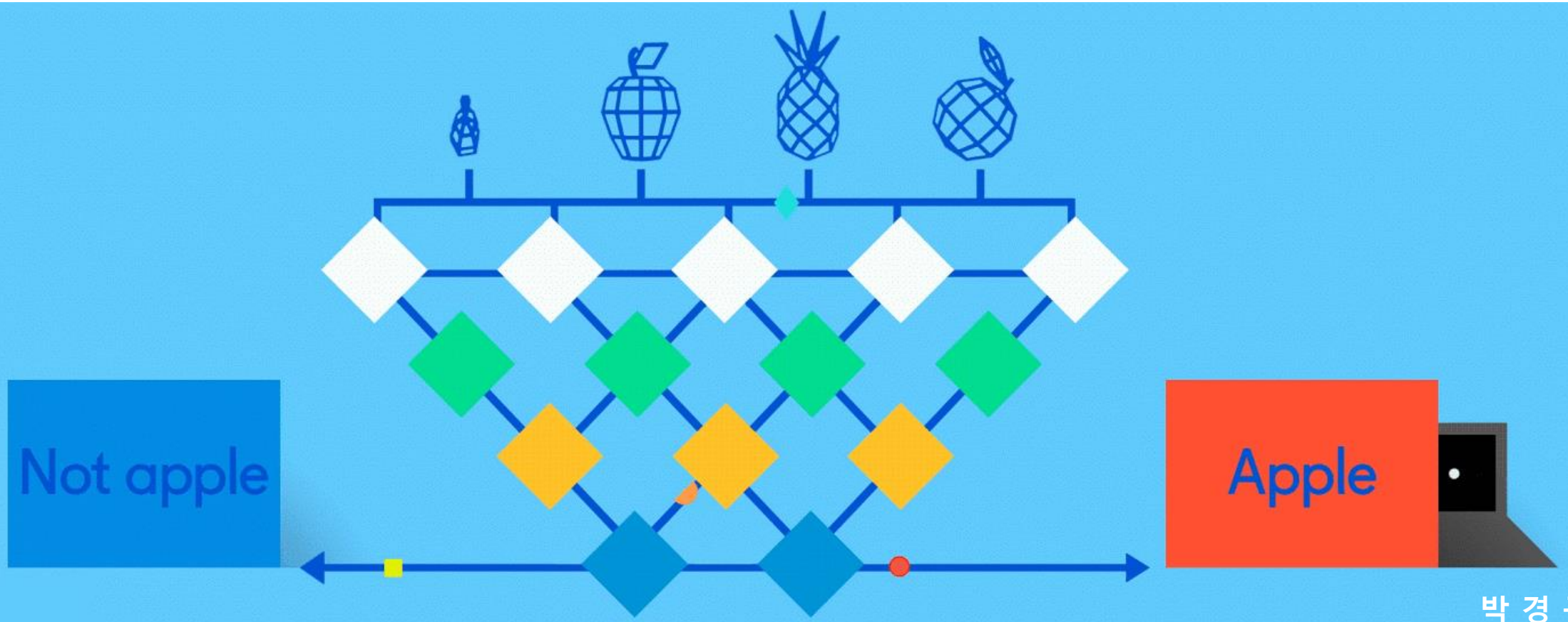
# 인공신경망 소개




박 경 규

본 챕터 전체 내용의 출처는 도서 "핸즈온 머신러닝 2판(Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition)"입니다.

# 목 차

# 1. 인공뉴런

# 인공신경망(Artificial Neural Network)

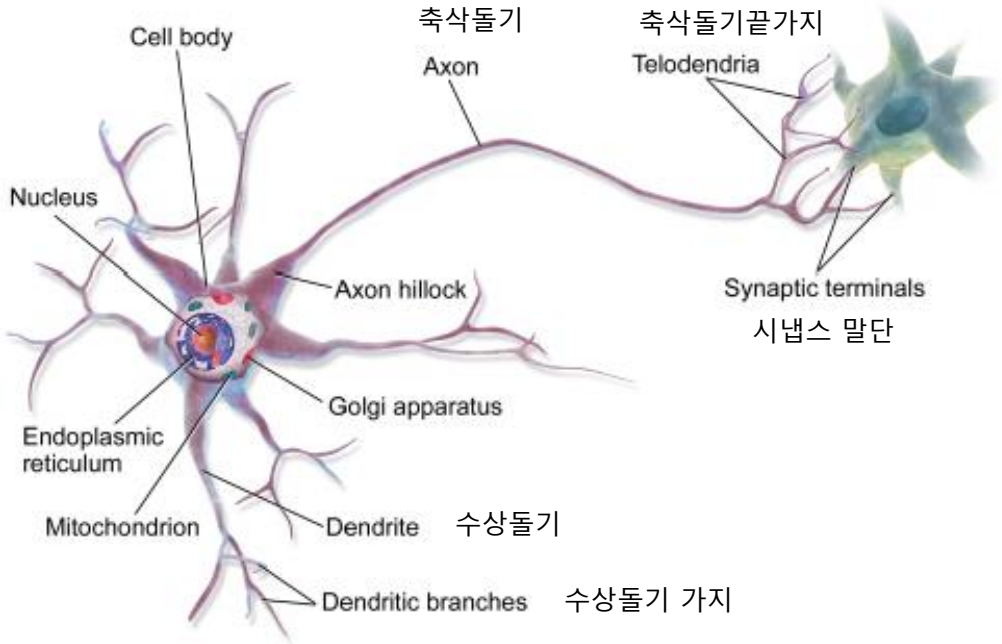
**딥러닝의 핵심인 인공신경망은 강력하고 확장성이 좋아서 이미지 분류, 음성인식서비스 성능 향상, 동영상 추천 등 복잡한 대규모 문제를 다루는데 적합**

## ■ 인공신경망 발전

- 빅데이터
- 머신러닝 기법보다 좋은 성능
- 컴퓨터 하드웨어 발전(GPU, CLOUD)
- 훈련 알고리즘 발전
- AI 분야 투자 활성화

## ■ 생물학적 뉴런

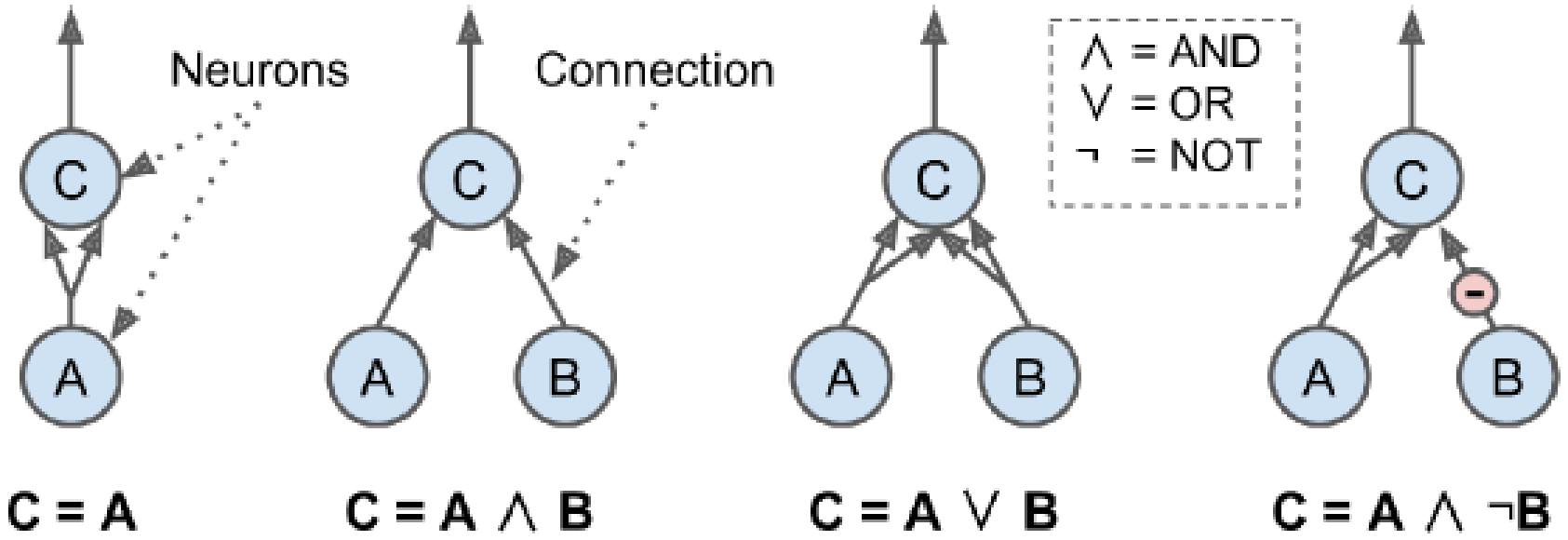
- 단순한 뉴런으로 구성된 거대한 네트워크가 매우 복잡한 계산을 수행할 수 있음



# 인공뉴런(Artificial Neuron)

인공뉴런은 하나 이상의 이진(on/off) 입력과 이진 출력 하나를 가지며, 입력이 일정 개수만큼 활성화되면 출력이 나옴

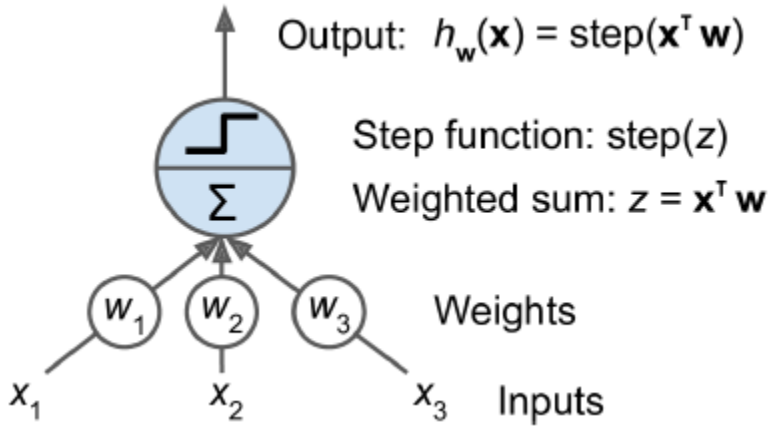
## ■ 뉴런을 사용한 논리연산



# 퍼셉트론(Perceptron)

입력과 출력이 숫자로 입력은 가중치와 연관되어 있으며 입력의 선형조합 계산이 임계값을 넘으면 양성클래스를 출력

## ■ 퍼셉트론



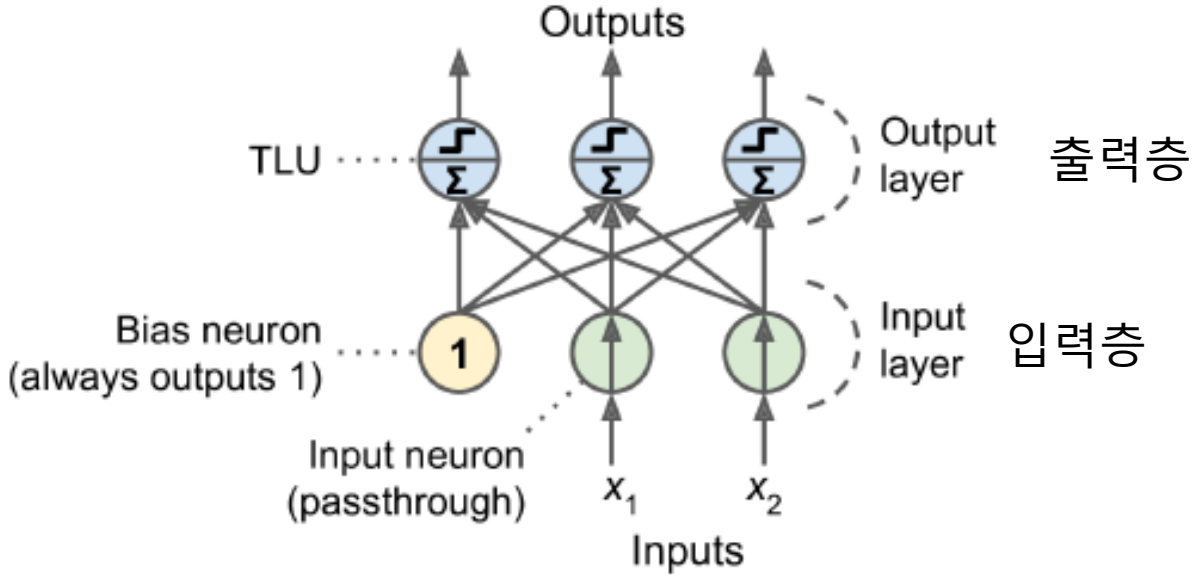

출력 계산 $h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b})$

계단 함수 (Step Function)

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

$$\text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

## ■ 퍼셉트론 구조



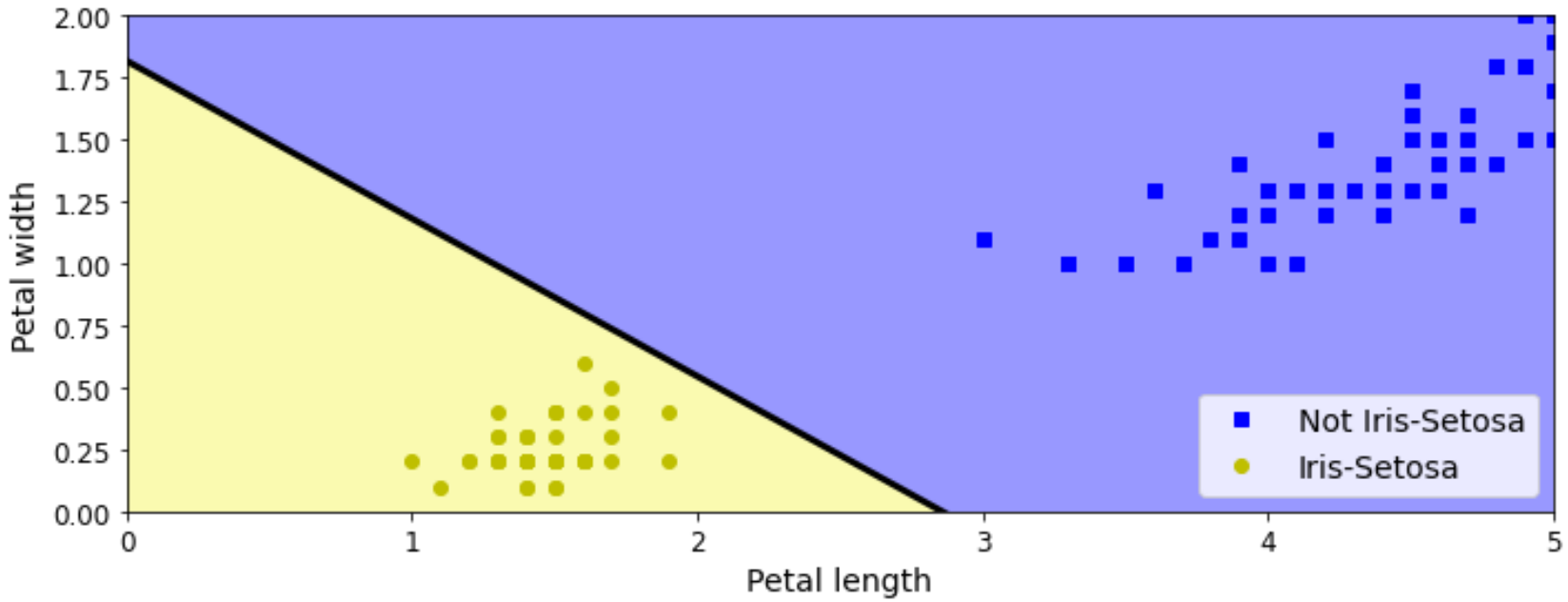

퍼셉트론 학습규칙(가중치 업데이트)

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\left(y_j - \hat{y}_j\right)x_i$$

= 가중치 + 학습율(출력값-타깃값)입력값

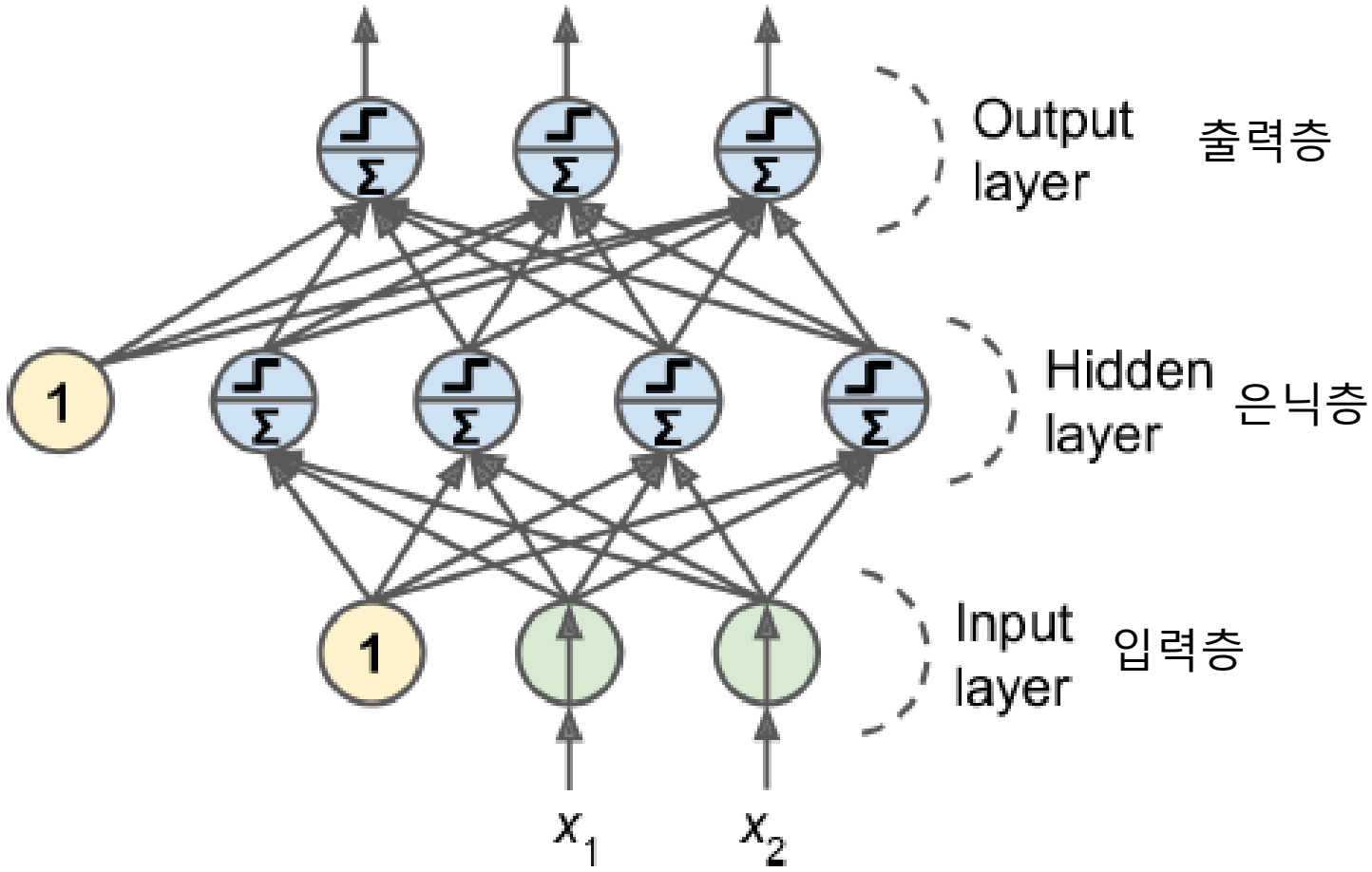
# 퍼셉트론(Perceptron)

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron

iris = load_iris()
X = iris.data[:, (2, 3)]  # 꽃잎 길이, 꽃잎 너비
y = (iris.target == 0).astype(np.int)

per_clf = Perceptron(max_iter=1000, tol=1e-3, random_state=42)
per_clf.fit(X, y)

y_pred = per_clf.predict([[2, 0.5]])
print(y_pred)
```



# 다층 퍼셉트론(MLP : Multilayer Perceptron)

입력층과 하나 이상의 은닉층과 출력층으로 구성되며, 은닉층을 여러 개 쌓아 올린 인공신경망을 심층신경망이라고 함



# 역전파(Backpropagation)

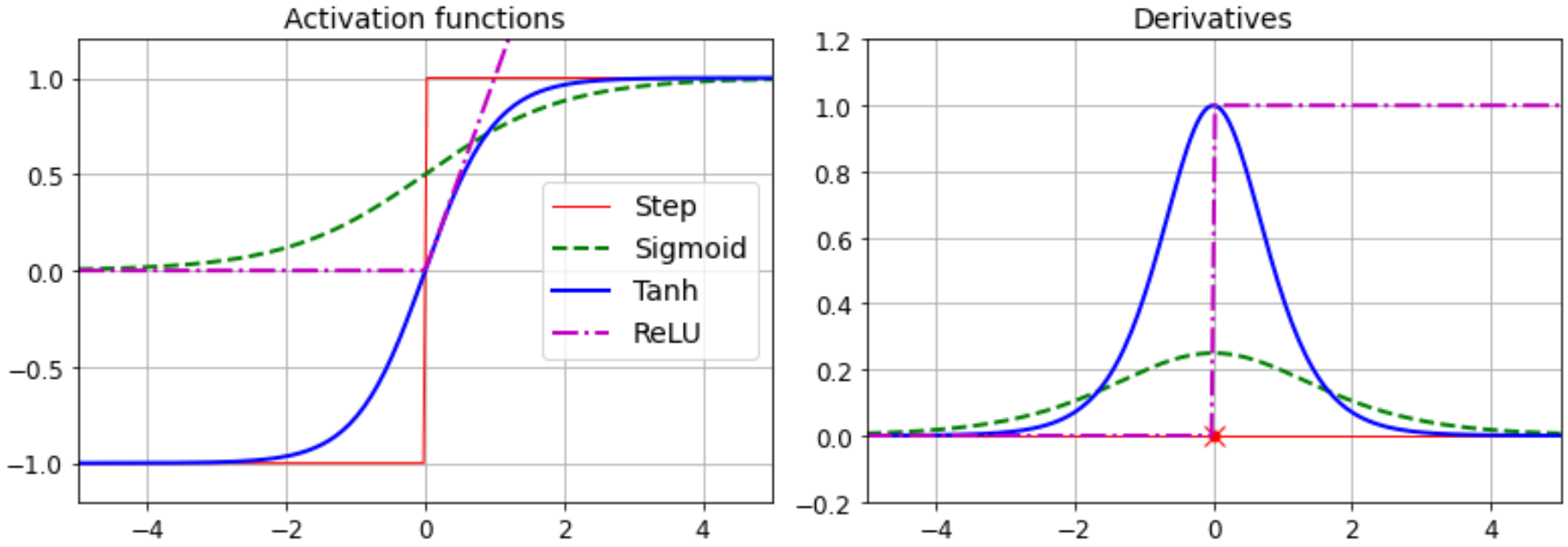
**역전파 알고리즘은 효율적인 기법으로 그레이디언트를 자동으로 계산하는 경사하강법**

## ■ 역전파 알고리즘

- 각 훈련 샘플에 대해 역전파 알고리즘이 먼저 예측을 만들고(정방향 계산) 오차를 측정
- 역방향으로 각 층을 거리면서 각 연결이 오차에 기여한 정도를 계산(역방향 계산)
- 이 오차가 감소하도록 가중치를 조정(경사하강법 단계)

# 활성화 함수(Activation function)

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def relu(z):
    return np.maximum(0, z)

def derivative(f, z, eps=0.000001):
    return (f(z + eps) - f(z - eps))/(2 * eps)
```



# 회귀 MLP(Regression MLP)

값 하나를 예측하는 데 출력 뉴런이 하나만 필요하며 다변량회귀에서는 출력차원마다 출력뉴런이 하나씩 필요

## ■ 회귀 MLP(Multi Layer Perceptron) 구조

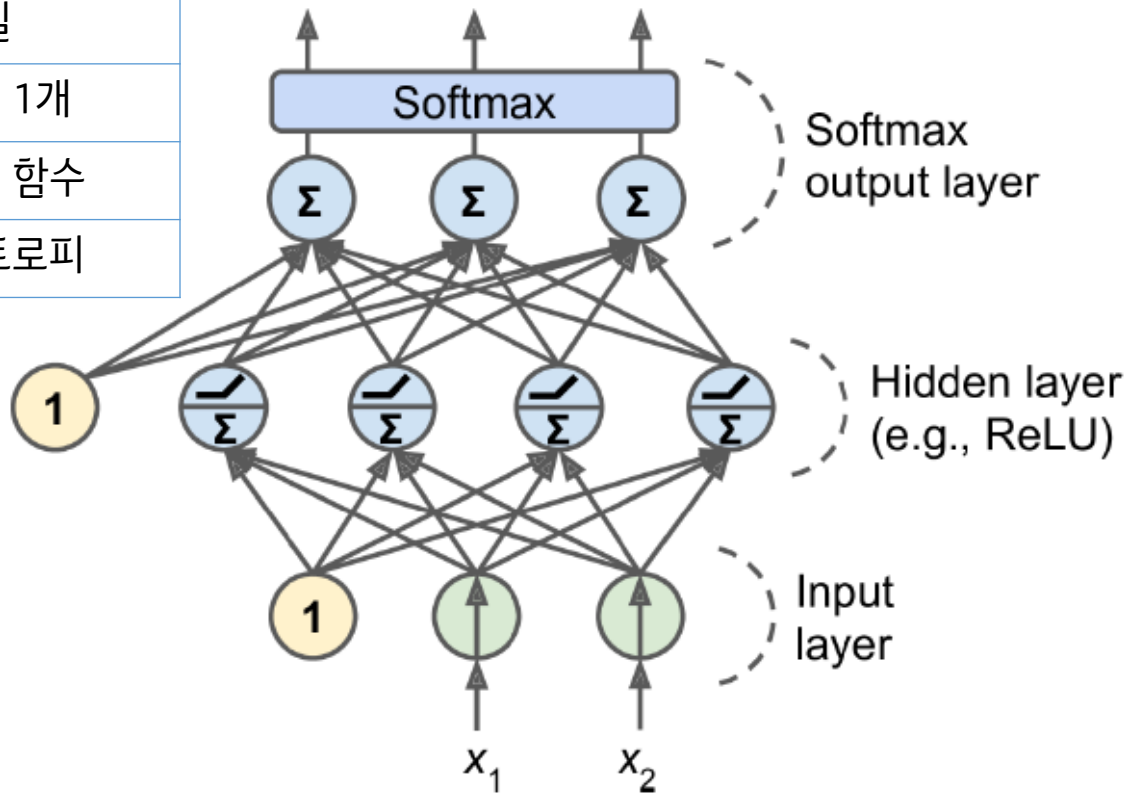
| 하이퍼파라미터 | 일반적인 값 |
|---|---|
| 입력 뉴런 수 | 특성(input feature)마다 1개 |
| 은닉층 수 | 문제에 따라 다름, 일반적으로 1~5개 |
| 은닉층의 뉴런 수 | 문제에 따라 다름, 일반적으로 10~100개 |
| 출력 뉴런 수 | 예측 차원(dimension) 마다 하나 |
| 은닉층의 활성화 함수 | ReLU |
| 출력층의 활성화 함수 | 없음, 또는 ReLU/softplus (양의 출력) 또는 logistic/tanh (특정 범위 출력)) |
| 손실 함수 | MSE 또는 MAE/Huber (이상치가 있는 경우) |

# 분류 MLP(Classification MLP)

이진분류에서는 로지스틱 활성화 함수를 가진 하나의 출력뉴런만 필요하며, n개의 클래스가 있는 경우에는 클래스마다 출력뉴런이 필요하며 출력층에는 소프트맥스 활성화 함수를 사용

## ■ 분류 MLP 구조

| 하이퍼파라미터 | 이진분류 | 다중 레이블 분류 | 다중분류 |
|---|---|---|---|
| 입력층과 은닉층 | 회귀와 동일 | 회귀와 동일 | 회귀와 동일 |
| 출력 뉴런수 | 1개 | 레이블마다 1개 | 클래스마다 1개 |
| 출력층의 활성화 함수 | 로지스틱 함수 | 로지스틱 함수 | 소프트맥스 함수 |
| 손실함수 | 크로스 엔트로피 | 크로스 엔트로피 | 크로스 엔트로피 |

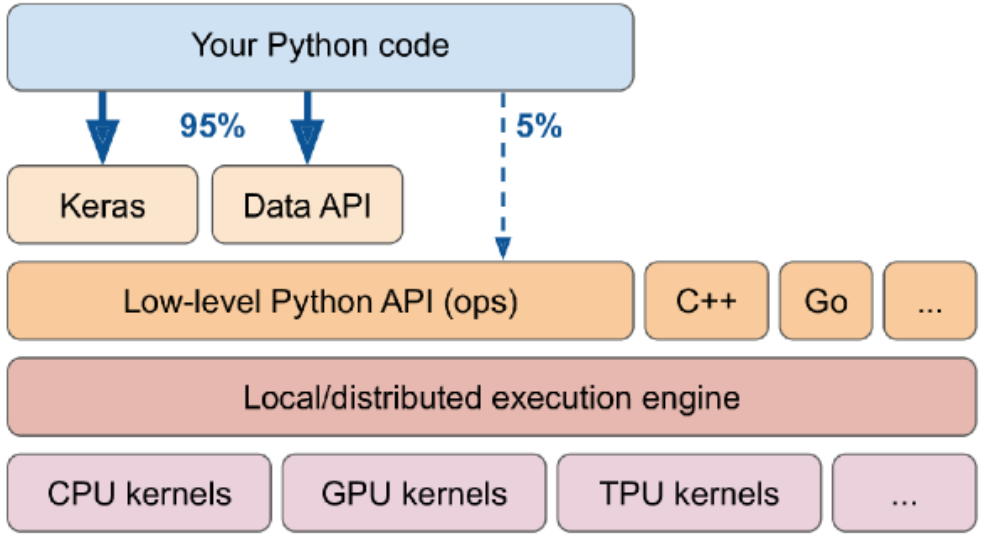


# 2. 다층 퍼셉트론 구현하기

# 텐서플로(Tensorflow)

텐서플로는 구글브레인에서 개발한 강력한 수치계산용 라이브러리로 현재 가장 인기있는 딥러닝 라이브러리입니다.
텐서플로 API는 텐서(Tensor)를 순환시킵니다. 텐서는 한 연산에서 다른 연산으로 흐릅니다. 그래서 텐서플로라고 부릅니다.

## ■ 텐서플로 구조




TensorFlow

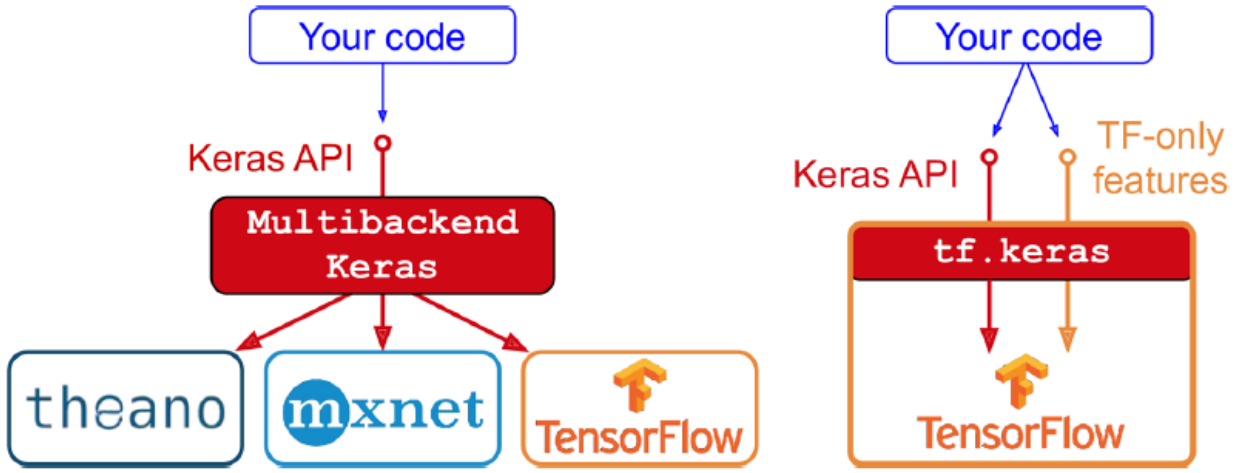
## ■ 텐서플로 파이썬API

| | | | |
|---|---|---|---|
| `tf.keras`<br>`tf.estimator` | High-level Deep Learning APIs | `tf.distribute`<br>`tf.saved_model`<br>`tf.autograph`<br>`tf.graph_util`<br>`tf.lite`<br>`tf.quantization`<br>`tf.tpu`<br>`tf.xla` | Deployment and optimization |
| `tf.nn`<br>`tf.losses`<br>`tf.metrics`<br>`tf.optimizers`<br>`tf.train`<br>`tf.initializers` | Low-level Deep Learning APIs | | |
| `tf.GradientTape`<br>`tf.gradients()` | Autodiff | `tf.lookup`<br>`tf.nest`<br>`tf.ragged`<br>`tf.sets`<br>`tf.sparse`<br>`tf.strings` | Special data structures |
| `tf.data`<br>`tf.feature_column`<br>`tf.audio`<br>`tf.image`<br>`tf.io`<br>`tf.queue` | I/O and preprocessing | `tf.math`<br>`tf.linalg`<br>`tf.signal`<br>`tf.random`<br>`tf.bitwise` | Mathematics, including linear algebra and signa processing |
| `tf.summary` | Visualization with TensorBoard | `tf.compat`<br>`tf.config` *& more* | Miscellaneous |

# 케라스(Keras)

케라스(Keras)는 모든 종류의 신경망을 쉽게 만들고 훈련, 평가, 실행할 수 있는 고수준 딥러닝 API입니다.

## ■ 케라스 API




## ■ 텐서플로2 설치

pip install tensorflow

## ■ 버전 확인 : 파이썬셀, 주피터 노트북

```
import tensorflow as tf
from tensorflow import keras
tf.__version__
keras.__version__
```

# 이미지 분류기 구현

```
# %matplotlib inline
%pylab

import os
import pandas as pd
import tensorflow as tf
from tensorflow import keras
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# 그래프 출력
# 그림을 저장할 위치
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "ann"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("그림 저장:", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

# 이미지 분류기 구현

```
# 데이터셋 로드
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()

# 트레이닝세트를 검증세트와 트레이닝세트로 분리
X_valid, X_train = X_train_full[:5000] / 255., X_train_full[5000:] / 255.
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
X_test = X_test / 255.

# 이미지 출력
plt.imshow(X_train[0], cmap="binary")
plt.axis('off')
plt.show()

# 클래스 이름
class_names = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
               "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"]

# 모델 만들기
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

# 이미지 분류기 구현

```
# 모델 출력
print(model.summary())
keras.utils.plot_model(model, "my_fashion_mnist_model.png", show_shapes=True)

# 모델 컴파일
model.compile(loss=keras.losses.sparse_categorical_crossentropy,
              optimizer=keras.optimizers.SGD(),
              metrics=[keras.metrics.sparse_categorical_accuracy])

# 모델 훈련과 평가
history = model.fit(X_train, y_train, epochs=30,
                    validation_data=(X_valid, y_valid))

# 학습 곡선 출력
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
save_fig("keras_learning_curves_plot")
plt.show()

# 모델 평가 : 상용환경으로 배포하기 전에 테스트세트로 평가
model.evaluate(X_test, y_test)

# 예측
X_new = X_test[:3]
y_proba = model.predict(X_new)
y_proba.round(2)
```

# 주택 가격 예측기 구현

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# 데이터셋 로드
housing = fetch_california_housing()

# 훈련 세트를 검증 세트와 훈련 세트로 분리
X_train_full, X_test, y_train_full, y_test = train_test_split(housing.data, housing.target, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full, y_train_full, random_state=42)

# 모든 특성의 스케일 조정
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)
```

# 주택 가격 예측기 구현

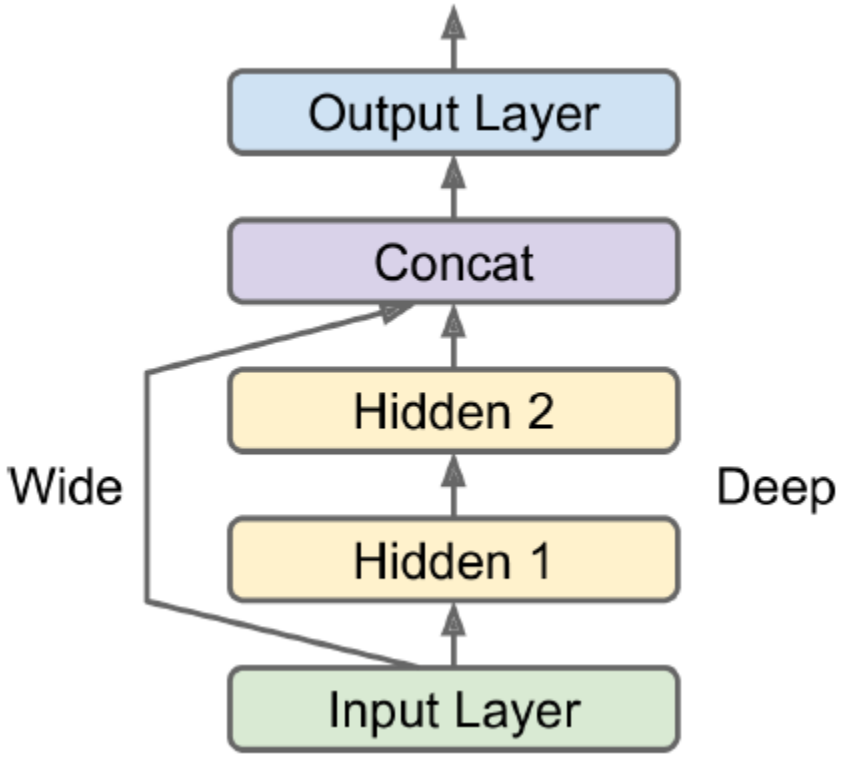
## 회귀MLP는 분류MLP와 비슷하며, 차이점은 출력층이 활성화 함수가 없는 하나의 뉴런이고 손실함수로 MSE를 사용

```
np.random.seed(42)
tf.random.set_seed(42)

# 모델 구출, 훈련, 평가, 예측
model = keras.models.Sequential([
    keras.layers.Dense(30, activation="relu", input_shape=X_train.shape[1:]),
    keras.layers.Dense(1)
])
model.compile(loss="mean_squared_error", optimizer=keras.optimizers.SGD(lr=1e-3))
history = model.fit(X_train, y_train, epochs=20, validation_data=(X_valid, y_valid))
mse_test = model.evaluate(X_test, y_test)
X_new = X_test[:3]
y_pred = model.predict(X_new)
print(y_pred)
```

# 와이드 & 딥(Wide & Deep) 신경망

입력의 일부 또는 전체가 출력층에 바로 연결되는 구조로 복잡한 패턴과 간단한 규칙을 모두 학습할 수 있습니다.

## ■ 와이드 & 딥 신경망




```
input_ = keras.layers.Input(shape=X_train.shape[1:])
hidden1 = keras.layers.Dense(30, activation="relu")(input_)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)
concat = keras.layers.concatenate([input_, hidden2])
output = keras.layers.Dense(1)(concat)
model = keras.models.Model(inputs=[input_], outputs=[output])
```

# 모델 저장과 복원

케라스는 HDF5 포맷을 사용하여 모델 구조와 모델 파라미터를 저장합니다.

## ■ 모델 저장

```
model = keras.models.Sequential([...]) # or keras.Model([...])
model.compile([...])
model.fit([...])
model.save("my_keras_model.h5")
```

## ■ 모델 복원

```
model = keras.models.load_model("my_keras_model.h5")
```

# 콜백 사용하기

Fit() 메서드의 callbacks 매개변수를 사용하여 훈련의 시작이나 끝에 호출할 객체리시트를 지정할 수 있습니다.

## ■ 모델 체크포인트 저장

```
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model.h5")
history = model.fit(X_train, y_train, epochs=10, callbacks=[checkpoint_cb])
```

## ■ 최상 검증 세트 점수에서만 모델 저장

```
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model.h5", save_best_only=True)
history = model.fit(X_train, y_train, epochs=10,
                    validation_data=(X_valid, y_valid),
                    callbacks=[checkpoint_cb])
model = keras.models.load_model("my_keras_model.h5") # 최상의 모델로 복원
```

## ■ 훈련 조기종료 구현

```
early_stopping_cb = keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True)
history = model.fit(X_train, y_train, epochs=100,
validation_data=(X_valid, y_valid),
callbacks=[checkpoint_cb, early_stopping_cb])
```

# 텐서보드 시각화

텐서보드는 인터렉티브 시각화 도구로 계산 그래프 시각화와 훈련 통계 분석 등 많은 기능을 제공합니다.

## ■ 모델 체크포인트 저장

```
import os
root_logdir = os.path.join(os.curdir, "my_logs")

def get_run_logdir():
    import time
    run_id = time.strftime("run_%Y_%m_%d-%H_%M_%S")
    return os.path.join(root_logdir, run_id)

run_logdir = get_run_logdir()
```

## ■ 모델 구성과 컴파일

```
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model.h5", save_best_only=True)
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)
history = model.fit(X_train, y_train, epochs=30,
                    validation_data=(X_valid, y_valid),
                    callbacks=[checkpoint_cb, tensorboard_cb])
```

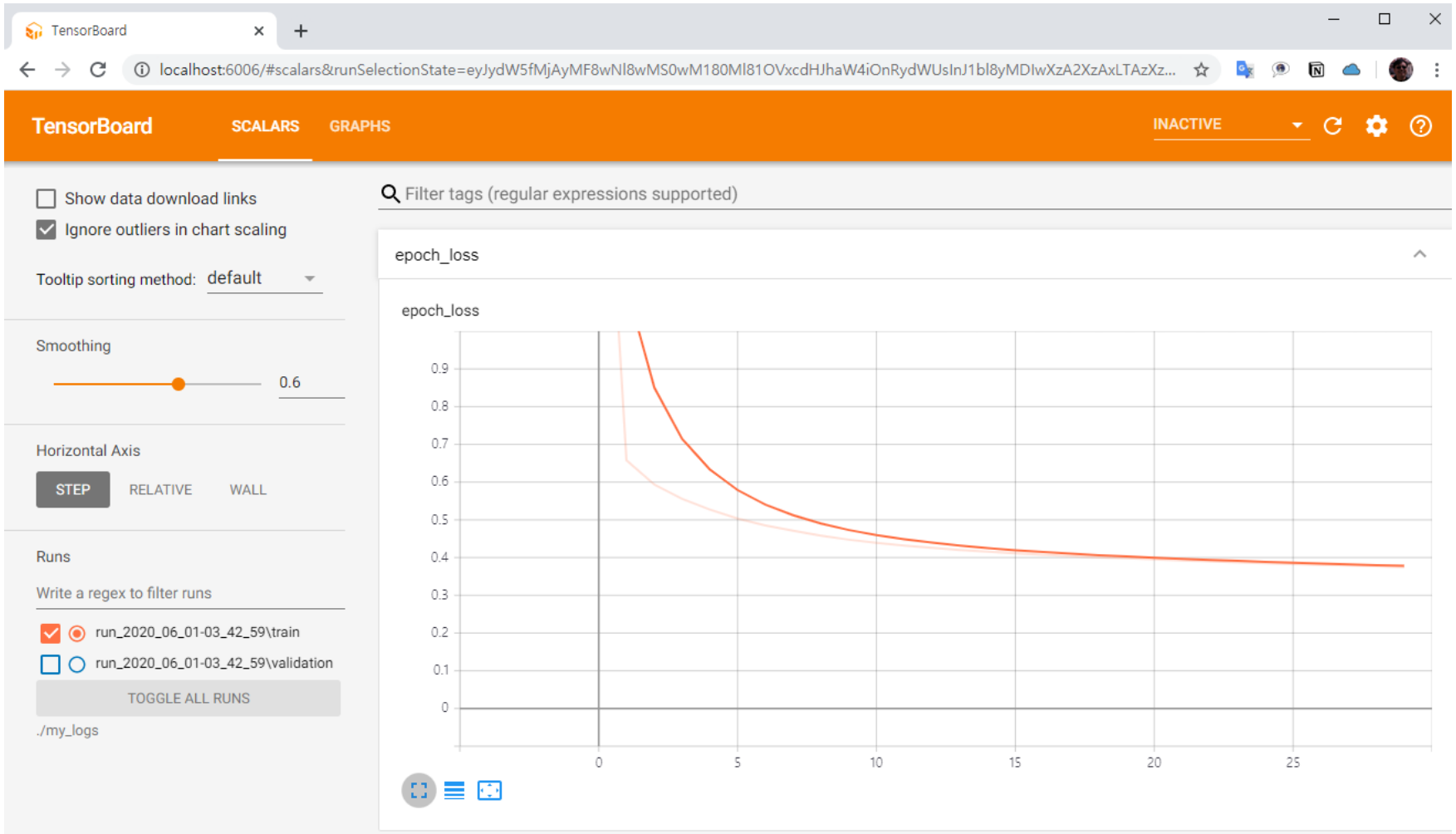
# 텐서보드 시각화

## ■ 텐서보드 서버 실행

tensorboard --logdir=./my_logs --port=6006

## ■ 모델 구성과 컴파일

http://localhost:6006/



# Thank you

박경규, 010-7240-1128
kgpark88@gmail.com