

# Android 응용 프로그래밍 - 기계학습(딥러닝)



부산대학교 정보·의생명공학대학  
정보컴퓨터공학부

# 이론

# 강의 목표와 구성

## ❖ 텐서플로 라이트

- Introduction of Tensorflow Lite

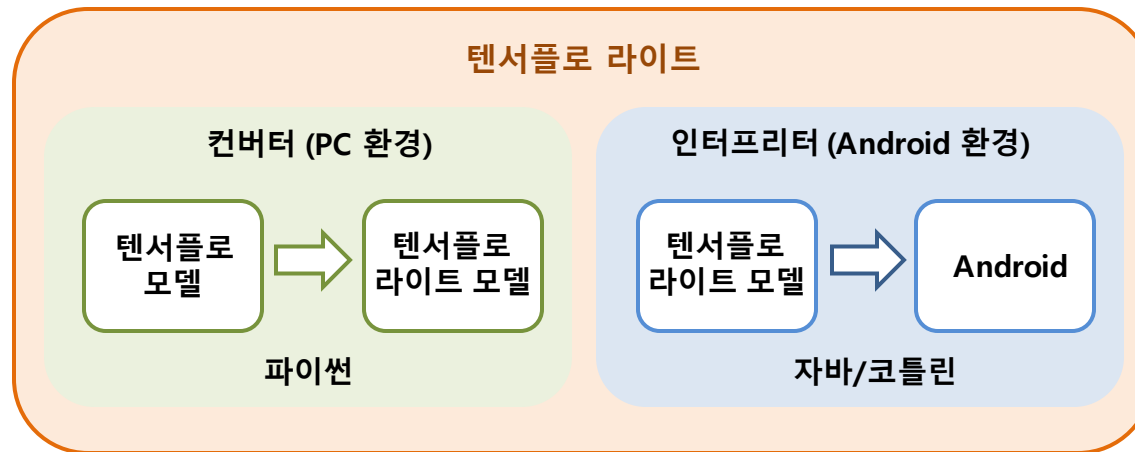
## ❖ 딥러닝

- What is Deep Learning
- Neural Network
- Perceptron
- Loss Function
- Gradient

# Introduction of Tensorflow Lite

## ❖ 텐서플로 라이트(Tensorflow Lite)란?

- 모바일과 IoT 기기에 딥러닝 모델을 배포하고 사용하기 위한 라이브러리
- 텐서플로 모델을 모바일 환경에서 동작 가능하게 하는 컨버터(Converter)와 실제 환경에서 동작하는 인터프리터(Interpreter)로 구성



딥러닝 모델의 학습 및 추론에는 많은 컴퓨팅 자원을 필요하므로  
컨버터(PC 환경)에서 학습 후 텐서플로 라이트 파일로 변환하여 사용

## ❖ 텐서플로 라이트 기반의 개발 Android 앱 프로세스



# Introduction of Tensorflow Lite

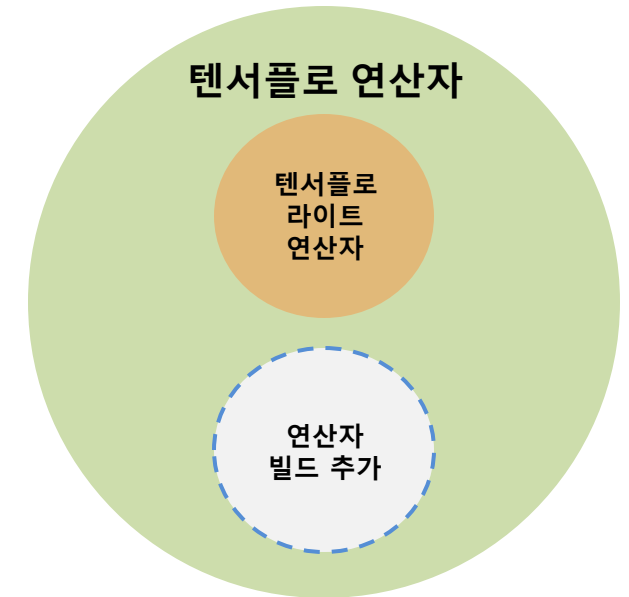
## ❖ 텐서플로 라이트의 특징

- 컴퓨팅 자원이 제한된 환경에서 딥러닝 모델을 실행해야 하므로 **효율성에 초점**
- Android, iOS, 라즈베리 파이 등 다양한 플랫폼 환경과 자바, 파이썬, C++ 등 다양한 언어 지원
- 다양한 **하드웨어 가속 기능을 제공**하여 성능 극대화 가능

## ❖ 텐서플로 vs 텐서플로 라이트

- 개발 프로세스에서의 역할, 개발 언어, 지원하는 연산자 범위에서 차이가 있다.

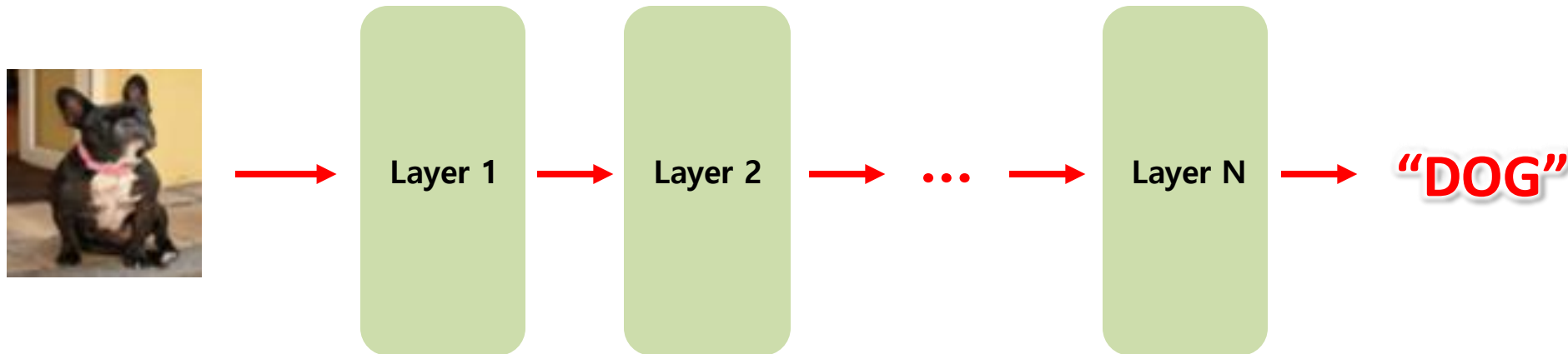
구분	텐서플로	텐서플로 라이트
개발 언어	파이썬	파이썬/자바/코틀린
역할	딥러닝 모델 개발과 학습	딥러닝 모델 변환, Android에서 모델 실행(추론)
연산자	텐서플로의 모든 연산자	텐서플로의 일부 연산자(연산자 추가 가능)



# What is Deep Learning

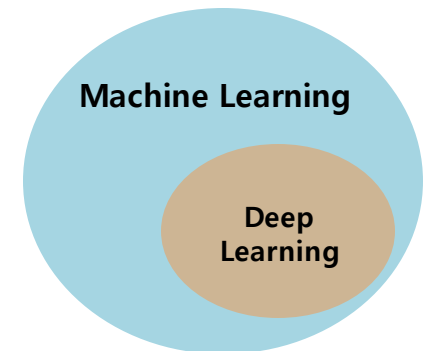
## ❖ 딥러닝(Deep Learning)이란?

- 여러 층을 가진 인공신경망(Artificial Neural Network)을 사용하여 기계학습을 수행
- 기계가 자동으로 대규모 데이터에서 중요한 패턴 및 규칙을 학습하고, 이를 토대로 의사결정이나 예측 등을 수행하는 기술



## ❖ 기계학습(Machine Learning) vs 딥러닝(Deep Learning)

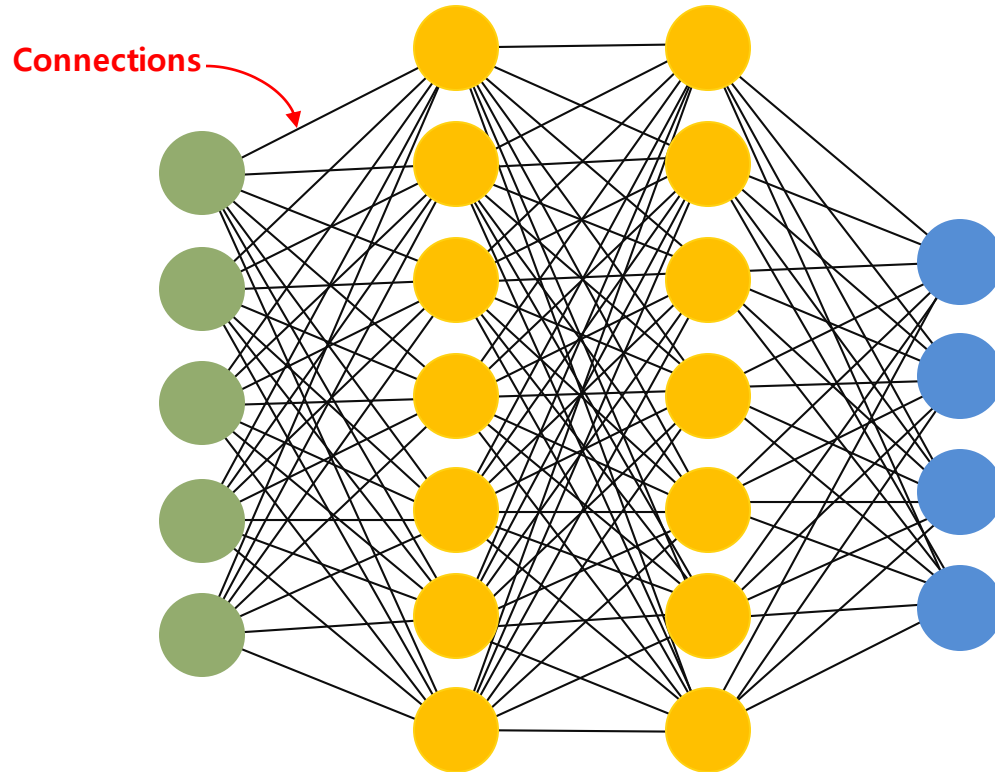
- 기계학습 : 학습하려는 데이터의 여러 특징 중에서 어떤 특징을 추출할지를 사람이 직접 분석하고 판단
- 딥러닝 : 기계가 자동으로 학습하려는 데이터에서 중요한 특징을 추출하여 학습
- 딥러닝은 기계학습의 한 종류






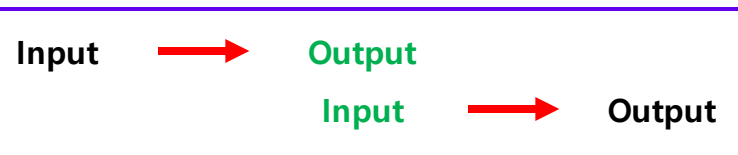
# Neural Network

## ❖ 신경망(Neural Network)이란?

- 여러 뉴런이 서로 **연결**되어 있는 구조의 네트워크
- 연결된 **INPUT/OUTPUT Units**의 집합



-  입력층(Input layer) : 학습 데이터를 입력 받는 레이어
-  은닉층(Hidden layer) : 데이터 연산을 수행하는 레이어
-  출력층(Output layer) : 최종결과를 출력하는 레이어

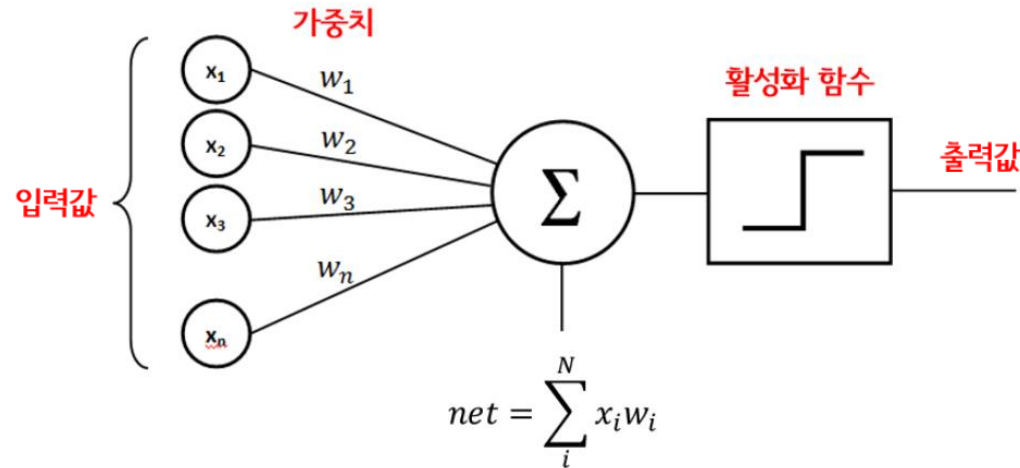


: 이전 레이어의 출력이 다음 레이어의 입력  
[www.tcpschool.com](http://www.tcpschool.com)

# Perceptron

## ❖ 퍼셉트론이란?(perceptron)

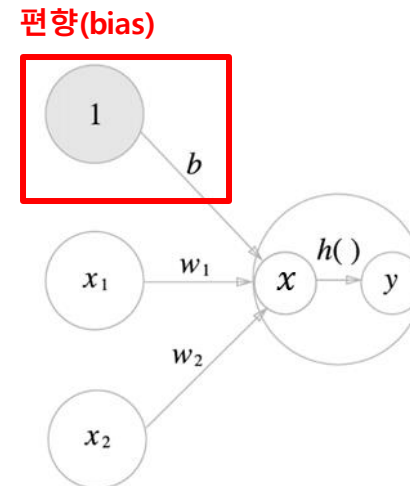
- 신경망(Neural Network)의 기원이 되는 알고리즘으로 다수의 신호를 입력으로 받아 하나의 신호를 출력
- 복수의 입력 신호에 고유한 가중치(weight)를 부여하여 각 입력 신호의 영향력을 제어



[sdc-james.gitbook.io/onebook/4.-and/5.2./5.2.1.](https://sdc-james.gitbook.io/onebook/4.-and/5.2./5.2.1.)

## ❖ 편향(bias)

- 뉴런의 활성화 정도를 조절하기 위한 변수



뉴런의 활성화 제어

$$y = h(\underline{b} + w_1 x_1 + w_2 x_2)$$

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

입력 신호의 총합을 출력 신호로 변환  
= 활성화 함수(Activation Function)



# Perceptron

## ❖ 활성화 함수(Activation Function)

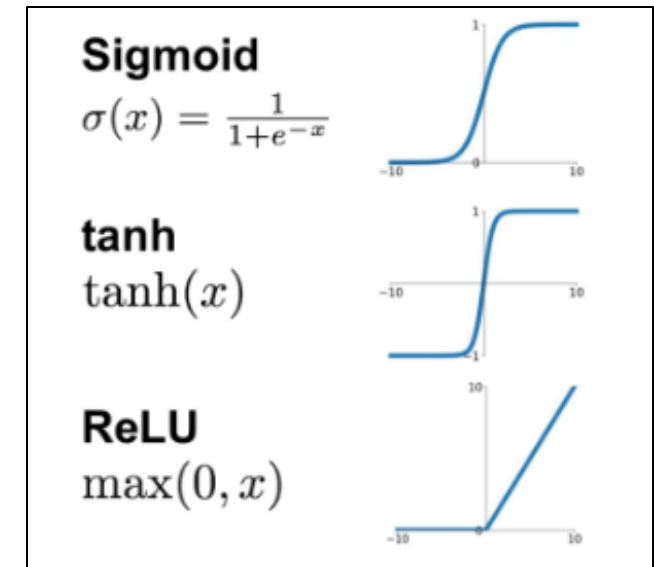
- 선형 함수의 결과를 **비선형 형태로 변형**하기 위한 함수
- XOR과 같은 **비선형 문제를 해결**하기 위한 해결책
- 활성화 함수로 선형함수를 사용할 경우 **레이어가 깊어져도 결국 하나의 선형 함수**이다.
  - Ex) 선형함수인  $f(x) = 3x$ 를 활성화 함수로 사용하여 3층의 레이어를 쌓을 경우  
 $f(f(f(x))) = 3(3(3x)) \rightarrow g(x) = f(f(f(x))) = 27x$

선형 함수로는 XOR 분류 불가능



<선형분류기의 한계>

ganghee-lee.tistory.com/30



<대표적인 활성화 함수>

ANALYSIS OF OPTIMIZING NEURAL NETWORKS AND  
ARTIFICIAL INTELLIGENT MODELS FOR GUIDANCE, CONTROL,  
AND NAVIGATION SYSTEMS

# Loss Function

## ❖ 손실 함수(Loss Function)

- 최적의 매개변수 값(weight, bias)을 탐색하기 위한 지표
  - 손실 함수의 궁극적 목적은 신경망이 높은 정확도를 나타내도록 매개 변수를 찾는 것
- 평균 제곱 오차(MSE), 교차 엔트로피 오차(Cross-Entropy Error)가 주로 사용됨

- 평균 제곱 오차(Mean Squared Error, MSE)

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

Class 수 신경망의 예측 결과 실제 결과

0.00975...

y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
y = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]

0.5975...

- 교차 엔트로피 오차(Cross Entropy Error, CEE)

$$E = - \sum_k t_k \log y_k$$

Class 수 신경망의 예측 결과 실제 결과

0.51082..

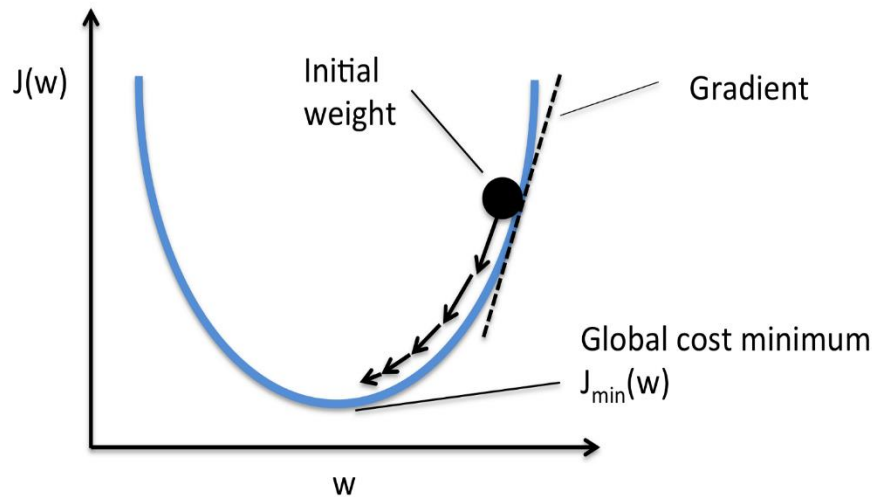
y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0]
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
y = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0]

2.30258

# Gradient

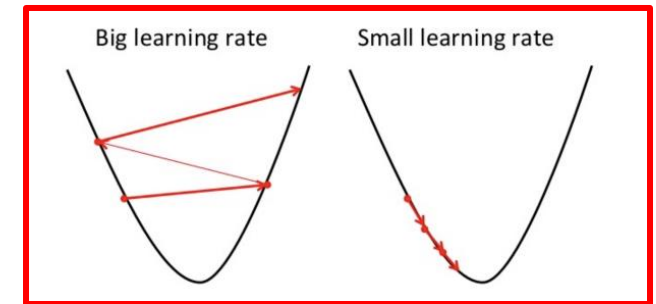
## ❖ 경사 하강법(Gradient Descent)

- 실제 신경망 학습에서는 **손실 함수의 미분(기울기)을 계산**하고 결과에 따라 **매개 변수를 갱신**
  - 기울기 < 0 : 매개변수를 **양**의 방향으로 변화
  - 기울기 > 0 : 매개변수를 **음**의 방향으로 변화
  - 기울기 = 0 : 매개변수 갱신 중지
- 손실 함수의 기울기(경사)를 구하여 **기울기가 낮은 쪽으로 계속 이동**시켜서 **극값**에 이를 때까지 반복시키는 것



[sebastianraschka.com/faq/docs/closed-form-vs-gd.html](http://sebastianraschka.com/faq/docs/closed-form-vs-gd.html)

$$\underbrace{\mathbf{x}_{i+1}}_{\text{변위}} = \mathbf{x}_i - \underbrace{\gamma_i}_{\text{학습 계수 (learning rate)}} \underbrace{\nabla f(\mathbf{x}_i)}_{\text{기울기}}$$



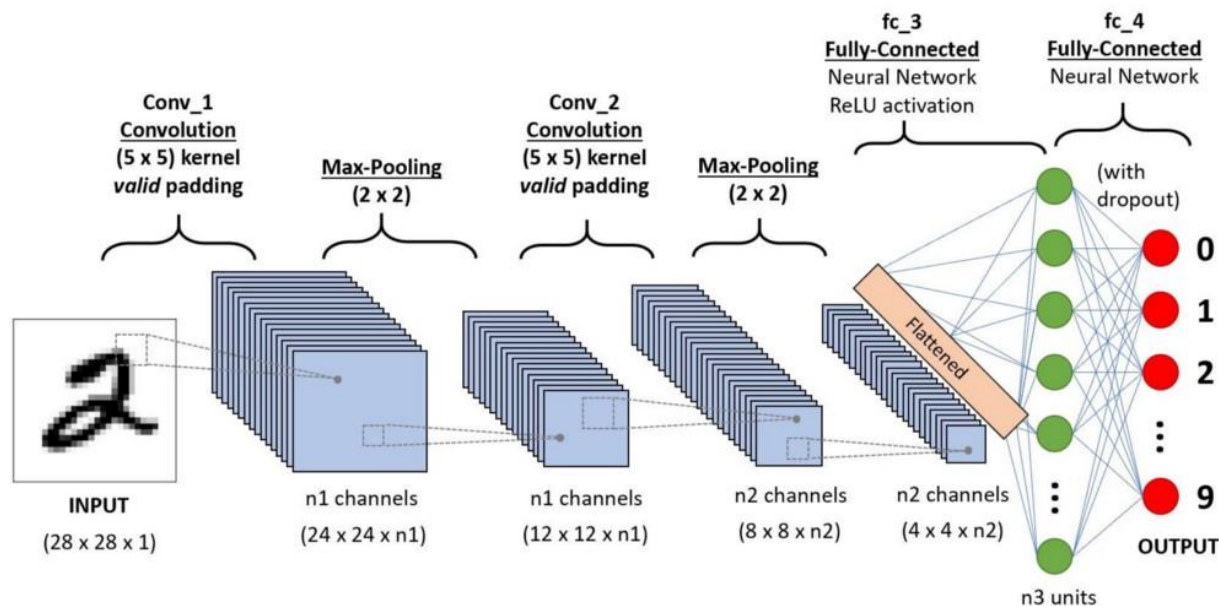
학습 계수가 너무 크면 값이 발산  
학습 계수가 너무 작으면 학습 소요 시간 증가

[resources.experfy.com/ai-ml/linear-regression](http://resources.experfy.com/ai-ml/linear-regression)

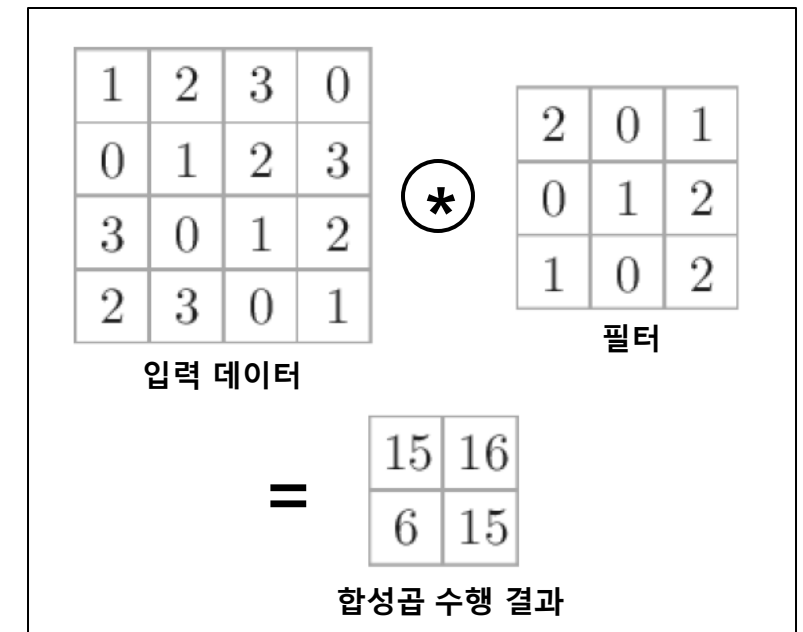
# 대표적인 딥러닝 모델

## ❖ CNN (Convolutional Neural Network)

- **이미지나 영상**을 다루는 컴퓨터 비전에서 가장 대표적으로 사용되는 인공지능망
- CNN의 레이어 구성
  - **Convolutional Layer** : 공간 정보를 학습하기 위해 이미지에 필터를 적용하여 합성곱 연산을 수행
  - **Pooling Layer** : 데이터의 크기를 줄여 연산량을 줄이는 역할(학습 파라미터 감소)을 수행
  - **Fully Connected Layer** : 최종 결과 추출을 위해 출력 값의 형태와 레이블의 길이로 데이터를 변환
  - **ReLU Layer** : 활성화 레이어



<CNN 모델의 구조> [towardsdatascience.com](https://towardsdatascience.com)



<합성곱 연산> [kolikim.tistory.com/53](https://kolikim.tistory.com/53)

# 실습 - 텐서플로 라이트 개발환경 구축

# 실습 목표와 구성

## ❖ 텐서플로 라이트 개발 환경 구축

- 파이썬 개발 환경 구축

## ❖ 딥러닝 모델 실습 1(MNIST)

- Overview
- 데이터 셋
- 모델 생성
- 모델 훈련
- 모델 평가

## ❖ 딥러닝 모델 실습 2

- Overview
- 데이터 처리
- 모델 생성
- 추론 결과 확인

## ❖ 딥러닝 모델 실습 3

# 파이썬 개발환경 구축

## ❖ 아나콘다 설치 (1/2)

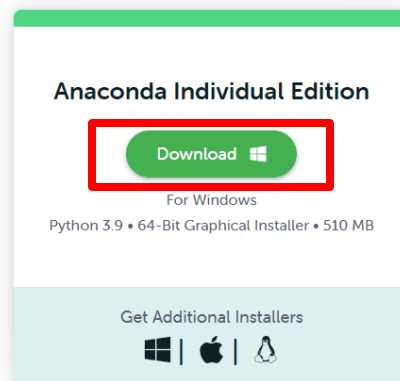
- <https://www.anaconda.com/products/individual>



Individual Edition

## Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.



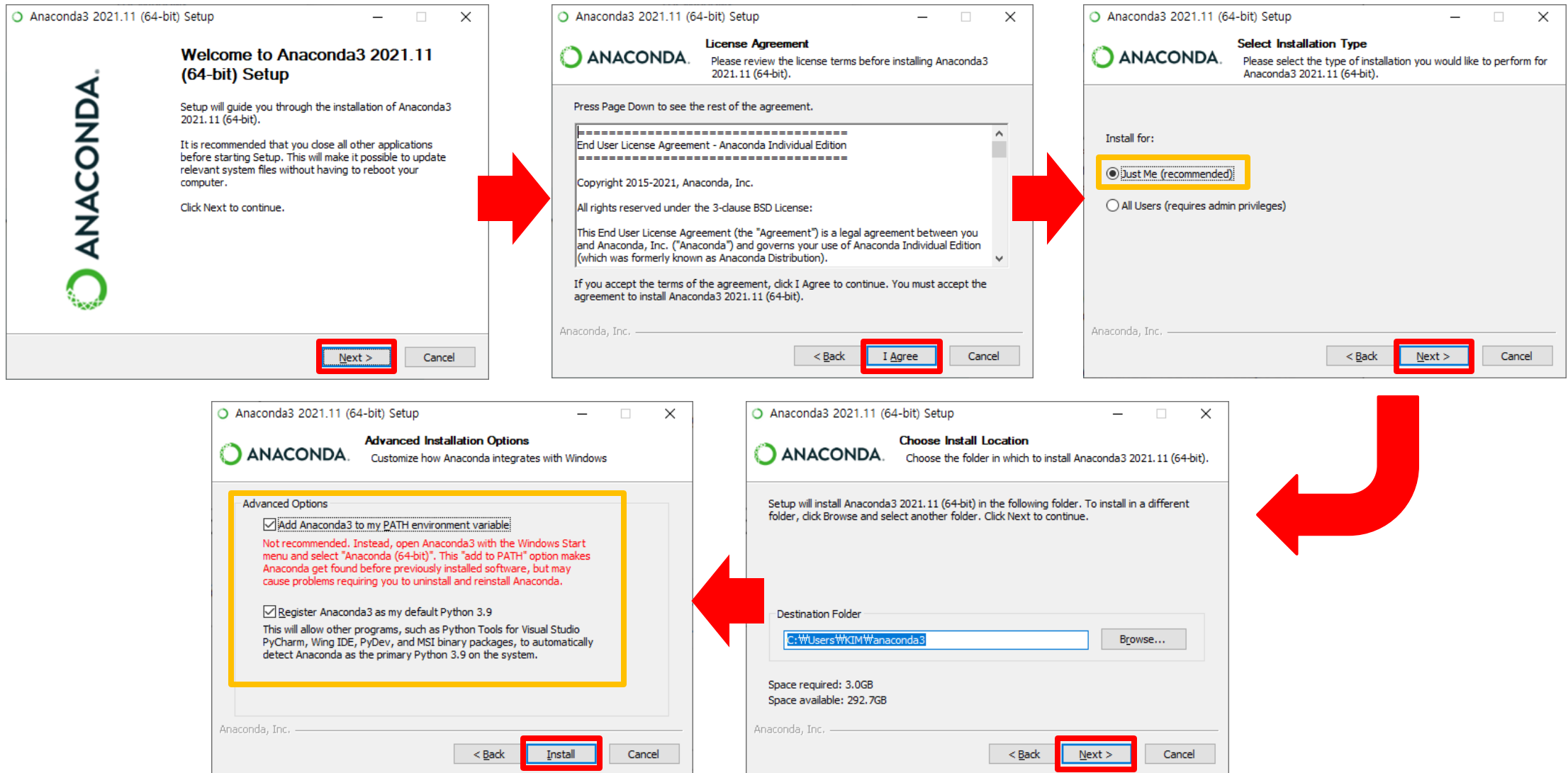
Windows, 64-Bit 버전 설치

Anaconda Installers		
Windows	MacOS	Linux
Python 3.9	Python 3.9	Python 3.9
64-Bit Graphical Installer (510 MB)	64-Bit Graphical Installer (515 MB)	64-Bit (x86) Installer (581 MB)
32-Bit Graphical Installer (404 MB)	64-Bit Command Line Installer (508 MB)	64-Bit (Power8 and Power9) Installer (255 MB)
		64-Bit (AWS Graviton2 / ARM64) Installer (488 MB)
		64-bit (Linux on IBM Z & LinuxONE) Installer (242 MB)

OS별 버전 선택 가능

# 파이썬 개발환경 구축

## ❖ 아나콘다 설치 (2/2)





# 파이썬 개발환경 구축

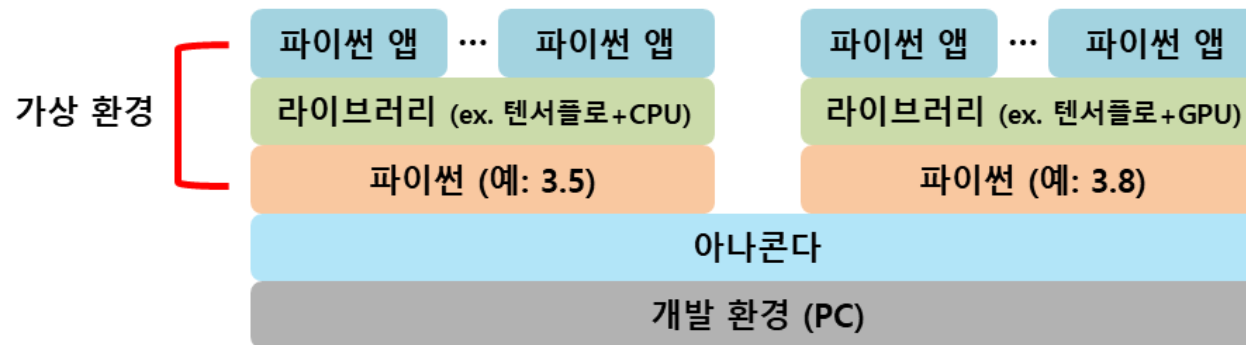
## ❖ 아나콘다를 설치하는 이유

- 머신러닝이나 데이터 분석등에 사용하는 여러가지 패키지가 기본적으로 포함
- 파이썬 가상환경(Virtual Environments) 구축에 용이

다양한 프로젝트 수행 시 프로젝트 별로 요구하는 프로그램이나 모듈의 버전이 상이할 경우 가상환경을 통해 원하는 환경을 별도로 구축할 수 있음.



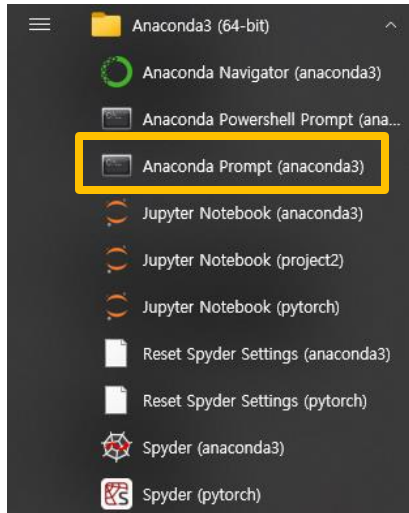
파이썬 정식 배포판 사용 시 개발 환경 구성



아나콘다 사용 시 개발 환경 구성

# 파이썬 개발환경 구축

## ❖ 가상환경 구축



Anaconda Prompt 실행  
(시작 메뉴)

```
$ conda create -n 가상환경이름 python=3.8(버전)
```

```
(base) C:\Users\KIM>conda create -n tf2 python=3.8
```

⋮

```
openssl          pkgs/main/win-64::openssl-1.1.1h2bfff1b_0
pip              pkgs/main/win-64::pip-21.2.2-py38haa95532_0
python          pkgs/main/win-64::python-3.8.12-h6244533_0
setuptools      pkgs/main/win-64::setuptools-58.0.4-py38haa95532_0
sqlite          pkgs/main/win-64::sqlite-3.37.0-h2bfff1b_0
vc              pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime  pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel           pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
wincertstore     pkgs/main/win-64::wincertstore-0.2-py38haa95532_2
```

```
Proceed ([y]/n)?
```

y 입력

가상환경 생성

```
$ conda env list
```

```
(base) C:\Users\KIM>conda env list
# conda environments:
#
base                  C:\Users\KIM\anaconda3
project2             C:\Users\KIM\anaconda3\envs\project2
pytorch              C:\Users\KIM\anaconda3\envs\pytorch
tf2                  C:\Users\KIM\anaconda3\envs\tf2
```

가상환경 list 조회

새로 생성한 가상환경 확인

```
$ activate 가상환경이름
```

```
(base) C:\Users\KIM>activate tf2
```

```
(tf2) C:\Users\KIM>
```

가상환경 활성화

→ 활성화된 가상환경 확인

# 파이썬 개발환경 구축

## ❖ Tensorflow 설치

```
$ pip install tensorflow
```

```
(base) C:\Users\KIM>activate tf2  
(tf2) C:\Users\KIM>pip install tensorflow
```

tensorflow 패키지 설치

```
$ python  
>>> import tensorflow as tf  
>>> print(tf.__version__)
```

```
(tf2) C:\Users\KIM>python  
Python 3.8.12 (default, Oct 12 2021, 03:01:40) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import tensorflow as tf  
>>> print(tf.__version__)  
2.7.0  
>>>
```

tensorflow 설치 확인 및 버전 확인

## ❖ Jupyter notebook 설치 및 ipykernel 생성

```
$ pip install jupyter notebook
```

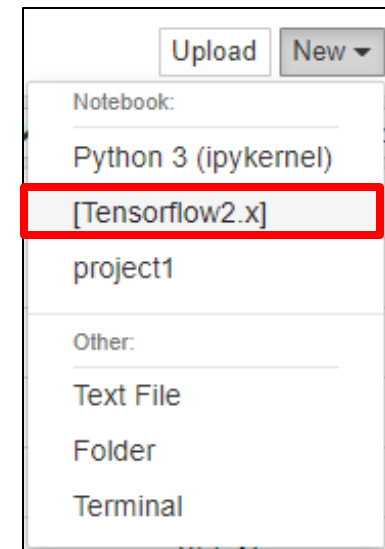
```
(tf2) C:\Users\KIM>pip install jupyter notebook
```

jupyter notebook 설치

```
$ python -m ipykernel install --user --name 가상환경이름 --display-name "커널이름"
```

```
(tf2) C:\Users\KIM>python -m ipykernel install --user --name tf2 --display-name "[Tensorflow2.x]"
```

ipykernel 생성



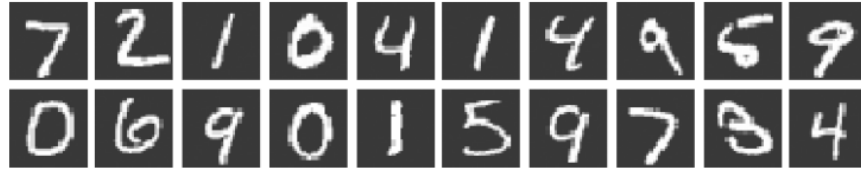
ipykernel 생성 확인  
(Jupyter notebook)

# 딥러닝 모델 실습 1(MNIST)

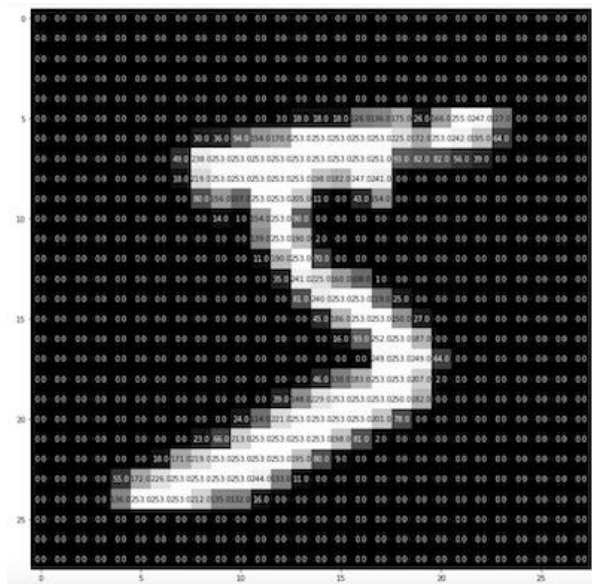
# Overview

## ❖ MNIST(Modified National Institute of Standards and Technology database)

- 손으로 쓰인 0에서 9까지의 숫자로 이루어진 이미지 데이터



- 각 이미지에는 어떤 숫자인지를 나타내는 정답 레이블 정보가 포함
- 이미지 데이터는 0에서 1까지의 값을 갖는 고정 크기의 28x28 행렬 (각 행렬의 원소는 픽셀의 밝기 정보)



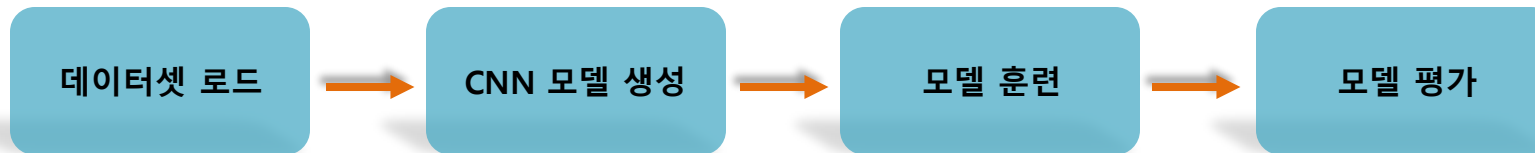
출처 : <https://jiho-ml.com/weekly-nlp-4/>

<손글씨 5 이미지 예시>

# Overview

## ❖ 실습 (따라하기)

- 실습 목표 : Tensorflow로 간단한 CNN 분류 모델 만들기 (MNIST 데이터셋 사용)
- 실습 과정



- 참고
  - 개발 툴 : Jupyter Notebook
  - 데이터셋 : MNIST
  - 딥러닝 라이브러리 : tensorflow

# 데이터 셋

## - Step 1. 라이브러리 불러오기

```
In [1]: import tensorflow as tf
        from matplotlib import pyplot as plt
        %matplotlib inline
```

에러 발생시  
우편 코드 실행 후  
다시 실행

In [0]: !pip install matplotlib

## - Step 2. 데이터셋 불러오기

```
In [2]: mnist = tf.keras.datasets.mnist
        (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Out [2]: Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
        11493376/11490434 [=====] - 1s 0us/step
        11501568/11490434 [=====] - 1s 0us/step
```

## - Step 3. 데이터 수 확인

```
In [3]: print('number of training data: ', len(x_train))
        print('number of test data: ', len(x_test))
```

```
Out [3]: number of training data: 60000
        number of test data: 10000
```

# 데이터 셋

## - Step 4. 데이터 형태 확인

```
In [4]: print('shape of training data: ', x_train.shape)
        print('shape of test data: ', x_test.shape)
        print('shape of data: ', x_train[0].shape)
```

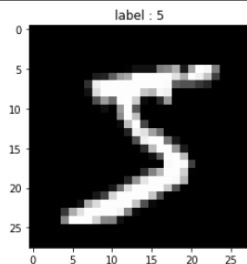
```
Out [4]: shape of training data: (60000, 28, 28)
         shape of test data: (10000, 28, 28)
         shape of data: (28, 28)
```

## - Step 5. 이미지 확인

```
In [4]: image = x_train[0]
        label = y_train[0]

        plt.imshow(image, cmap = 'gray')
        plt.title('label: %s' % label)
        plt.show()
```

Out [4]:





# 모델 생성

## - Step 1. CNN 모델 생성

---

```
In [1]: cnn_model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation = 'relu'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])
```

---

## - Step 2. 모델 컴파일 (Optimizer, 손실 함수, 평가지표 정의)

---

```
In [2]: cnn_model.compile(optimizer = 'adam',
    loss = 'sparse_categorical_crossentropy',
    metrics = ['accuracy'])
```

---

# 모델 생성

## - Step 3. 생성 모델 확인

---

In [3] : `cnn_model.summary()`

---

Out [3] : Model: "sequential\_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_4 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_3 (MaxPooling 2D)	(None, 13, 13, 32)	0
conv2d_5 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_4 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_6 (Conv2D)	(None, 3, 3, 64)	36928
flatten_1 (Flatten)	(None, 576)	0
dense_2 (Dense)	(None, 64)	36928
dense_3 (Dense)	(None, 10)	650
=====		
Total params: 93,322		
Trainable params: 93,322		
Non-trainable params: 0		
-----		

# 모델 훈련

## - Step 1. 데이터 전처리

```
In [1]: x_train_4d = x_train.reshape(-1, 28, 28, 1)
        x_test_4d = x_test.reshape(-1, 28, 28, 1)
        print('shape of x_train_4d: ', x_train_4d.shape)
        print('shape of x_test_4d: ', x_test_4d.shape)
```

```
Out [1]: shape of x_train_4d: (60000, 28, 28, 1)
         shape of x_test_4d: (10000, 28, 28, 1)
```

## - Step 2. 모델 훈련

```
In [2]: cnn_model.fit(x_train_4d, y_train, epochs = 5)
```

```
Out [1]: Epoch 1/5
         1875/1875 [=====] - 8s 4ms/step - loss: 0.2399 - accuracy: 0.9417
         Epoch 2/5
         1875/1875 [=====] - 8s 4ms/step - loss: 0.0639 - accuracy: 0.9803
         Epoch 3/5
         1875/1875 [=====] - 8s 4ms/step - loss: 0.0512 - accuracy: 0.9841
         Epoch 4/5
         1875/1875 [=====] - 8s 4ms/step - loss: 0.0440 - accuracy: 0.9870
         Epoch 5/5
         1875/1875 [=====] - 7s 4ms/step - loss: 0.0344 - accuracy: 0.9895
```

# 모델 평가

## - Step 1. Test 데이터로 모델 성능 평가

---

```
In [1]: cnn_model.evaluate(x_test_4d, y_test, verbose=1)
```

---

```
Out [1]: 313/313 [=====] - 1s 3ms/step - loss: 0.0401 - accuracy: 0.9893
```

---

## - Step 2. 임의 데이터로 모델 결과 확인

---

```
In [2]: eval_data = x_test[500]
eval_data_label = y_test[500]
cnn_model(eval_data.reshape(-1, 28, 28, 1))
```

---

```
Out [2]: <tf.Tensor: shape=(1, 10), dtype=float32, numpy=
array([[7.5083719e-36, 6.7215975e-17, 7.7770675e-20, 1.0000000e+00,
        1.8624271e-28, 4.2284554e-16, 6.2876745e-25, 1.0385777e-17,
        2.7338529e-20, 7.5823021e-17]], dtype=float32)>
```

---

## - Step 3. 실제 레이블 값 확인

---

```
In [3]: print(eval_data_label)
```

---

```
Out [3]: 3
```

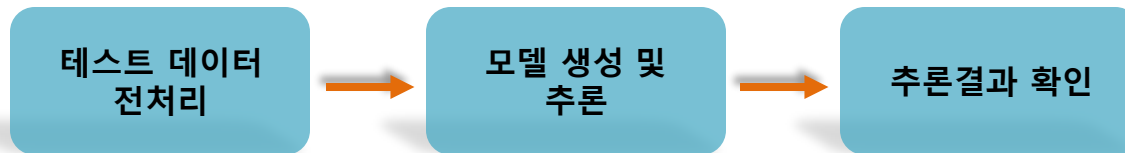
---

# 딥러닝 모델 실습 2

# Overview

## ❖ 실습 (따라하기)

- 실습 목표 : tensorflow에서 제공하는 사전 학습된 모델(Model : MobileNetV2 + Dataset : ImageNet)을 활용하여 데이터 추론
- 실습 과정

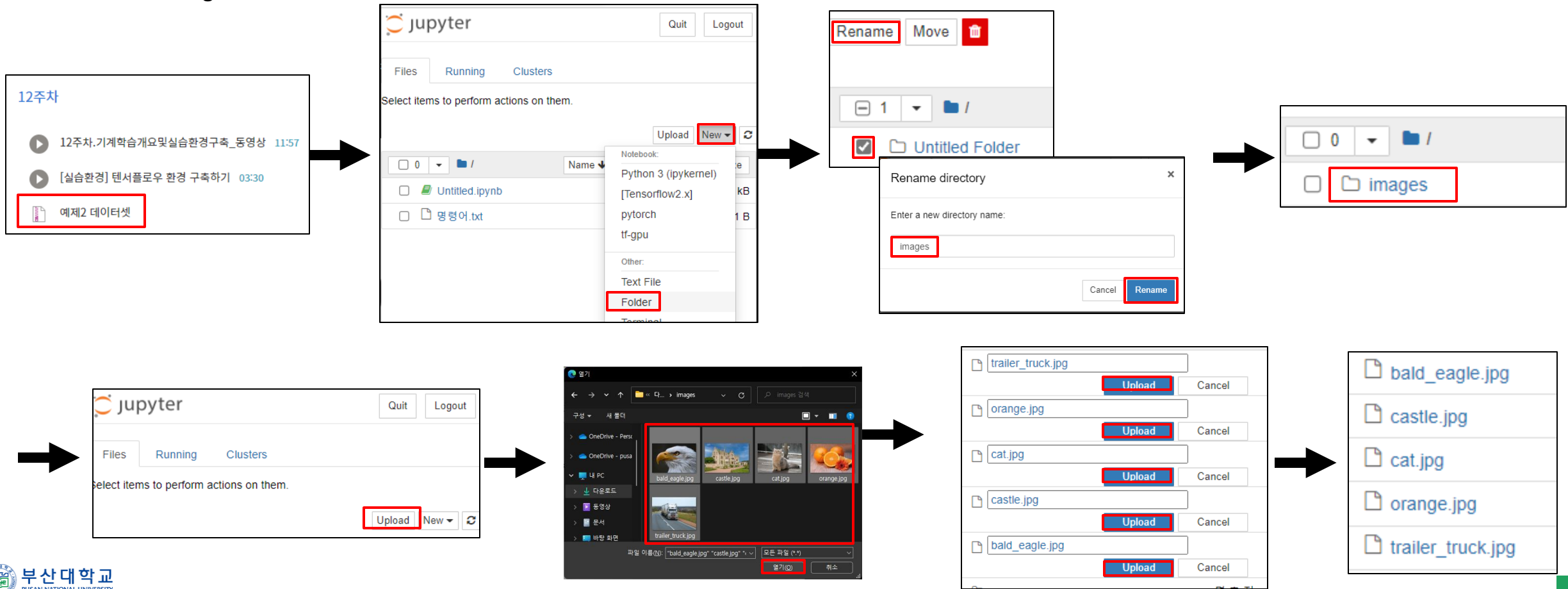


- 참고
  - 개발 툴 : Jupyter Notebook
  - 아키텍처 : MobileNetV2
  - 사전 학습 데이터 셋 : ImageNet
  - 딥러닝 라이브러리 : tensorflow

# Overview

## ❖ 데이터셋 가져오기

- 플라토 교과목 12주차 예제 2 데이터셋 다운로드 및 압축 풀기
- 주피터에 “images” 폴더 만들기
- “images” 폴더에 데이터셋 업로드 하기



# 데이터 처리

## - Step 1. 라이브러리 불러오기

---

```
In [1]: from PIL import Image
import os
import numpy as np
```

---

## - Step 2. 이미지 데이터 경로 추가

---

```
In [2]: data_dir = "./images/"
files = os.listdir(data_dir)

images = []

for file in files:
    path = os.path.join(data_dir, file)
    images.append(np.array(Image.open(path)))
```

---

## - Step 3. 데이터 전처리

---

```
In [3]: import tensorflow as tf

resized_images = np.array(np.zeros((len(images), 224, 224, 3)))
for i in range(len(images)):
    resized_images[i] = tf.image.resize(images[i], [224, 224])

preprocessed_images = tf.keras.applications.mobilenet_v2.preprocess_input(resized_images)
```

---



# 모델 생성

## - Step 4. 모델 생성 및 추론

---

```
In [4]: mobilenet_imagenet_model = tf.keras.applications.MobileNetV2(weights="imagenet")
```

```
        y_pred = mobilenet_imagenet_model.predict(preprocessed_images)
```

```
        topK = 1
```

```
        y_pred_top = tf.keras.applications.mobilenet_v2.decode_predictions(y_pred, top=topK)
```

---

```
Out [4]: Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_t
         f_kernels_1.0_224.h5
         14540800/14536120 [=====] - 1s 0us/step
         14548992/14536120 [=====] - 1s 0us/step
         Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
         40960/35363 [=====] - 0s 2us/step
         49152/35363 [=====] - 0s 1us/step
```

---

## - Step 5. 추론 결과 값 형태 확인

---

```
In [5]: y_pred.shape
```

---

```
Out [5]: (5, 1000)
```

---

→ Input : 5EA , Class : 1000EA

# 추론 결과 확인

## - Step 6. 추론 결과 확인

```
In [6]: from matplotlib import pyplot as plt
import numpy as np

for i in range(len(images)):
    plt.imshow(images[i])
    plt.show()

    for k in range(topK):
        print(f'{y_pred_top[i][k][1]} ({round(y_pred_top[i][k][2] * 100, 1)}%)')
```

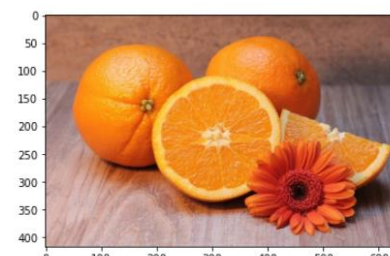
Out [6]:



bald\_eagle (69.1%)



castle (81.0%)



orange (97.5%)



trailer\_truck (52.0%)



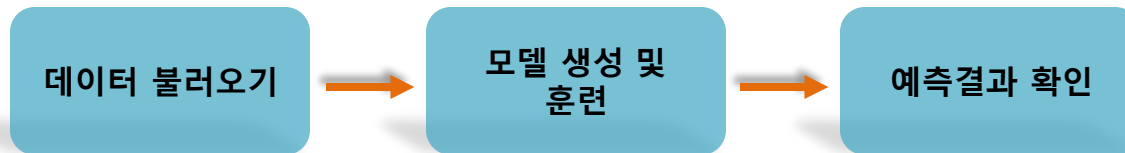
tiger\_cat (35.6%)

# 딥러닝 모델 실습 3

# Overview

## ❖ 실습 (따라하기)

- 실습 목표 : tensorflow에서 제공하는 FashionMNIST 데이터 셋을 활용하여 모델 생성 및 훈련
- 실습 과정



- 참고
  - 개발 툴 : Jupyter Notebook
  - 데이터 셋 : FashionMNIST
  - 딥러닝 라이브러리 : tensorflow

# 데이터셋 불러오기

## - Step 1. 라이브러리 불러오기

```
In [1]: import tensorflow as tf  
  
import numpy as np  
import matplotlib.pyplot as plt
```

## - Step 2. 데이터셋 불러오기

```
In [1]: fashion_mnist = tf.keras.datasets.fashion_mnist  
  
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

## - Step 3. Label 지정

```
In [3]: class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',  
                        'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

## - Step 4. Train image 형태 확인

```
In [4]: train_images.shape
```

```
Out [4]: (60000, 28, 28)
```

# 데이터 전처리

## - Step 5. train\_label 수 확인

---

```
In [5]: len(train_labels)
```

---

```
Out [5]: 60000
```

---

## - Step 6. Test image 형태 확인

---

```
In [6]: test_images.shape
```

---

```
Out [6]: (10000, 28, 28)
```

---

## - Step 7. Image Normalization

---

```
In [6]: train_images = train_images / 255.0  
        test_images = test_images / 255.0
```

---

# 이미지 확인

## - Step 8. 이미지 확인

```
In [8]: plt.figure(figsize=(10,10))
        for i in range(25):
            plt.subplot(5,5,i+1)
            plt.xticks([])
            plt.yticks([])
            plt.grid(False)
            plt.imshow(train_images[i], cmap=plt.cm.binary)
            plt.xlabel(class_names[train_labels[i]])
        plt.show()
```

Out [8]:



# 모델 생성 및 컴파일

## - Step 9. 모델 생성

---

```
In [9]: model = tf.keras.Sequential([  
        tf.keras.layers.Flatten(input_shape=(28, 28)),  
        tf.keras.layers.Dense(128, activation='relu'),  
        tf.keras.layers.Dense(10)  
    ])
```

---

## - Step 10. 모델 컴파일

---

```
In [10]: model.compile(optimizer='adam',  
                        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),  
                        metrics=['accuracy'])
```

---



# 모델 훈련

## - Step 11. 모델 훈련

---

In [11]: `model.fit(train_images, train_labels, epochs=10)`

---

Out  
[11]:

```
Epoch 1/10
1875/1875 [=====] - 2s 932us/step - loss: 0.5028 - accuracy: 0.8241
Epoch 2/10
1875/1875 [=====] - 2s 938us/step - loss: 0.3775 - accuracy: 0.8625
Epoch 3/10
1875/1875 [=====] - 2s 888us/step - loss: 0.3395 - accuracy: 0.8766
Epoch 4/10
1875/1875 [=====] - 2s 932us/step - loss: 0.3133 - accuracy: 0.8849
Epoch 5/10
1875/1875 [=====] - 2s 890us/step - loss: 0.2967 - accuracy: 0.8902
Epoch 6/10
1875/1875 [=====] - 2s 947us/step - loss: 0.2816 - accuracy: 0.8967
Epoch 7/10
1875/1875 [=====] - 2s 890us/step - loss: 0.2697 - accuracy: 0.8996
Epoch 8/10
1875/1875 [=====] - 2s 892us/step - loss: 0.2595 - accuracy: 0.9033
Epoch 9/10
1875/1875 [=====] - 2s 887us/step - loss: 0.2477 - accuracy: 0.9063
Epoch 10/10
1875/1875 [=====] - 2s 905us/step - loss: 0.2395 - accuracy: 0.9097
```

---

# 정확도 평가 및 예측

## - Step 12. 정확도 평가

---

```
In [12]: test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
         print('Test accuracy:', test_acc)
```

---

```
Out      313/313 - 0s - loss: 0.3393 - accuracy: 0.8814 - 294ms/epoch - 939us/step
[12]:
         Test accuracy: 0.8813999891281128
```

---

## - Step 13. 예측하기

---

```
In [13]: probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])

         predictions = probability_model.predict(test_images)

         predictions[0]
```

---

```
Out      array([8.85301858e-08, 2.22813837e-10, 9.56358395e-11, 1.00835484e-13,
[13]:      2.82542687e-11, 2.82390421e-04, 5.06104705e-08, 1.31144025e-03,
         1.32539946e-09, 9.98405993e-01], dtype=float32)
```

---

# 예측 결과 확인 함수 선언

## - Step 14. 예측값 그래프로 표현

---

```
In [14] : def plot_image(i, predictions_array, true_label, img):
            true_label, img = true_label[i], img[i]
            plt.grid(False)
            plt.xticks([])
            plt.yticks([])

            plt.imshow(img, cmap=plt.cm.binary)

            predicted_label = np.argmax(predictions_array)

            if predicted_label == true_label:
                color = 'blue'
            else:
                color = 'red'

            plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                                100*np.max(predictions_array),
                                                class_names[true_label]),
                      color=color)

def plot_value_array(i, predictions_array, true_label):
    true_label = true_label[i]
    plt.grid(False)
    plt.xticks(range(10))
    plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')
```

---

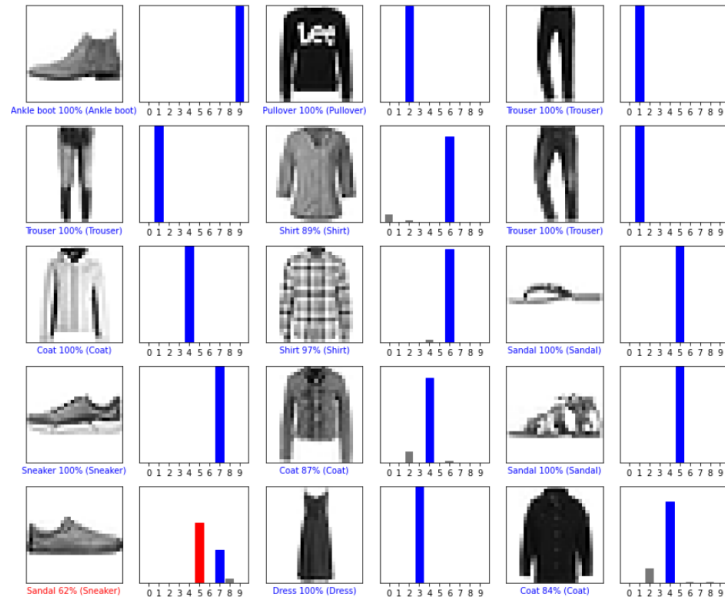
# 예측 결과 확인

## - Step 15. 예측 확인

```
In [15]: num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))

for i in range(num_images):
    plt.subplot(num_rows, 2*num_cols, 2*i+1)
    plot_image(i, predictions[i], test_labels, test_images)
    plt.subplot(num_rows, 2*num_cols, 2*i+2)
    plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```

Out [15]:

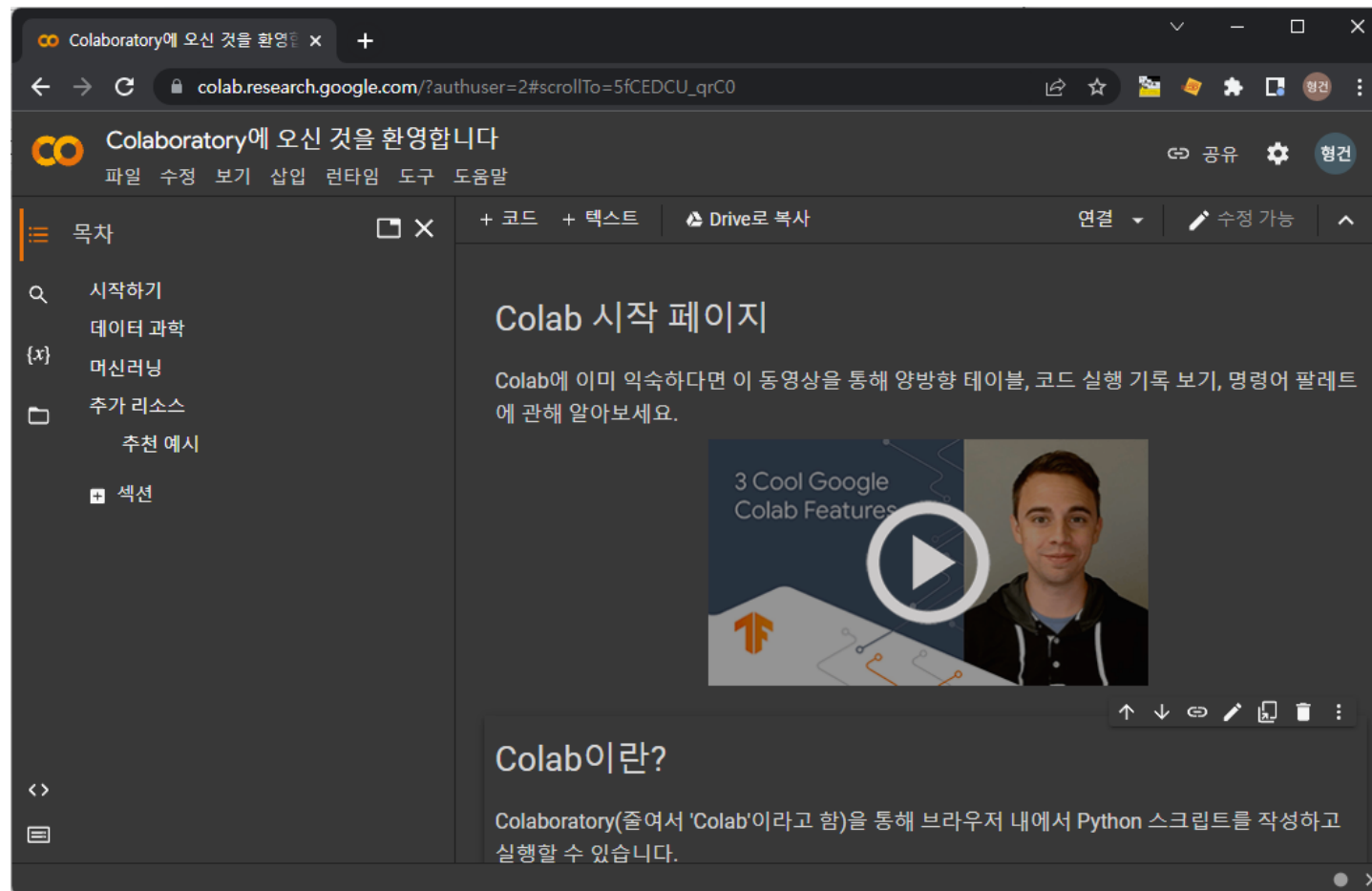


# 부록 - 코랩 개발환경 사용하기

# 부록 - 코랩 개발환경 사용하기

## ❖ Co-laboratory 란?

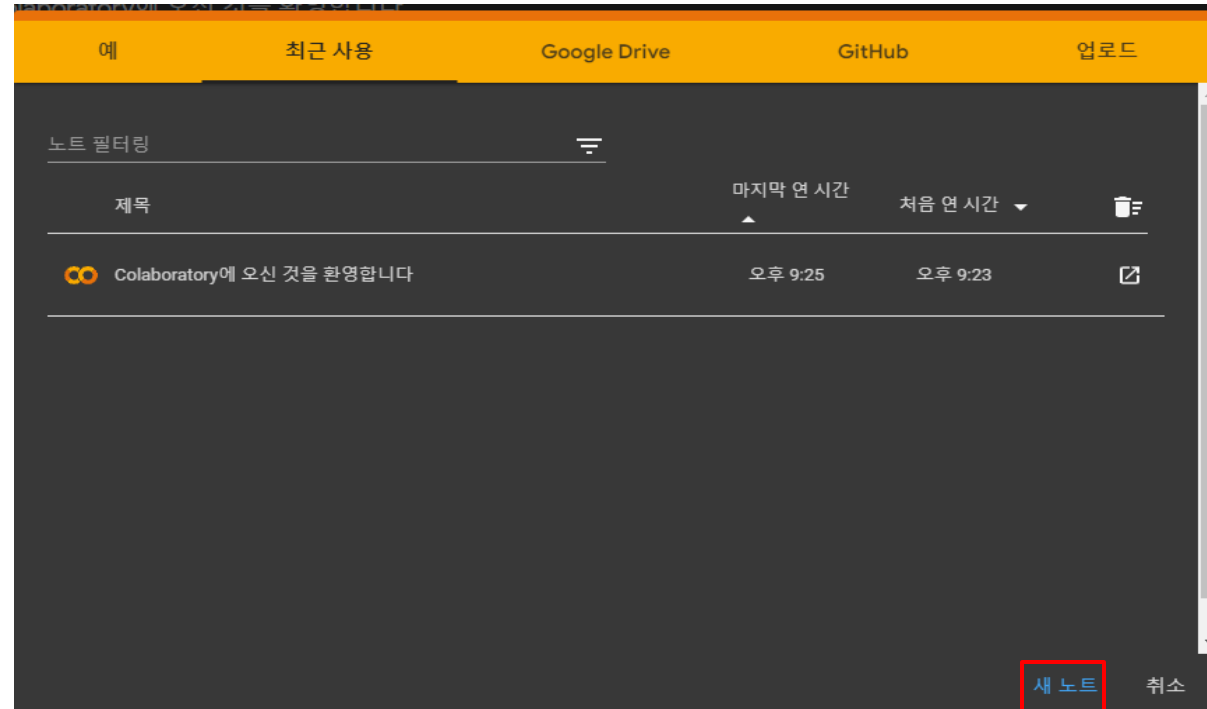
- 구글에서 만든 연구용 서비스 제품으로 Jupyter를 기반으로 만들어진 웹용 서비스
- 간단한 기계학습 예제와 온라인 통합 개발 환경을 제공



# 부록 - 코랩 개발환경 사용하기

## ❖ 개발 환경 설정

- 접속 하기
  - <https://colab.research.google.com>
- 환경 생성하기



# 부록 – 코랩 개발환경 사용하기

## ❖ 개발 환경 설정

- 파이썬 버전 변경하기

---

```
In [1]: !python -V
```

---

```
Out [1]: Python 3.7.13
```

---

---

```
In [2]: !wget https://www.python.org/ftp/python/3.8.13/Python-3.8.13.tgz
!tar xvfz Python-3.8.13.tgz
!Python-3.8.13/configure
!make
!sudo make install
```

---

---

```
Out [2]: ...
Looking in links: /tmp/tmpkdqgwv6q
Processing /tmp/tmpkdqgwv6q/setuptools-56.0.0-py3-none-any.whl
Processing /tmp/tmpkdqgwv6q/pip-22.0.4-py3-none-any.whl
Installing collected packages: setuptools, pip
Successfully installed pip-22.0.4 setuptools-56.0.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
```

---

5분 내외 진행

---

```
In [3]: !python -V
```

---

```
Out [3]: Python 3.8.13
```

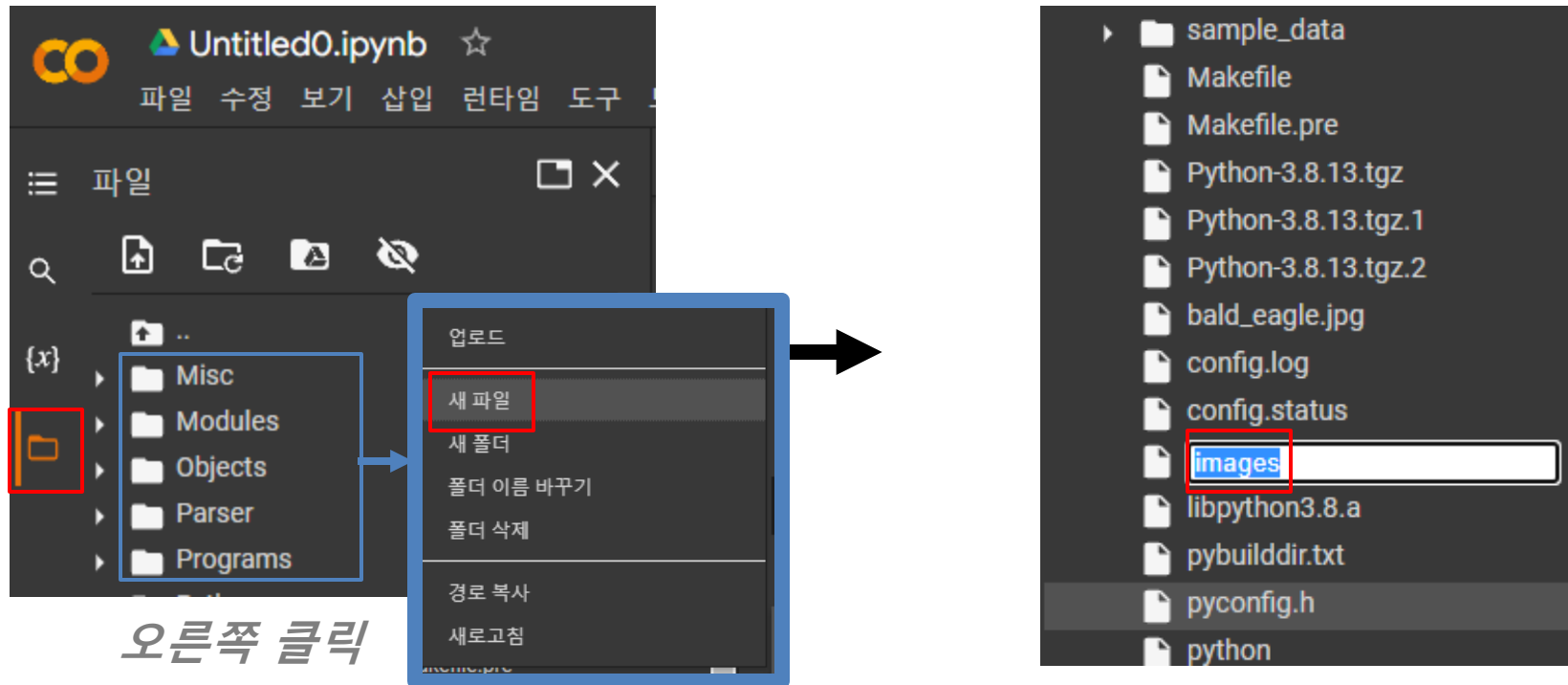
---



# 부록 - 코랩 개발환경 사용하기

## ❖ 개발 환경 설정

- 폴더 생성 하기
  - 현재 파일 경로 : /content

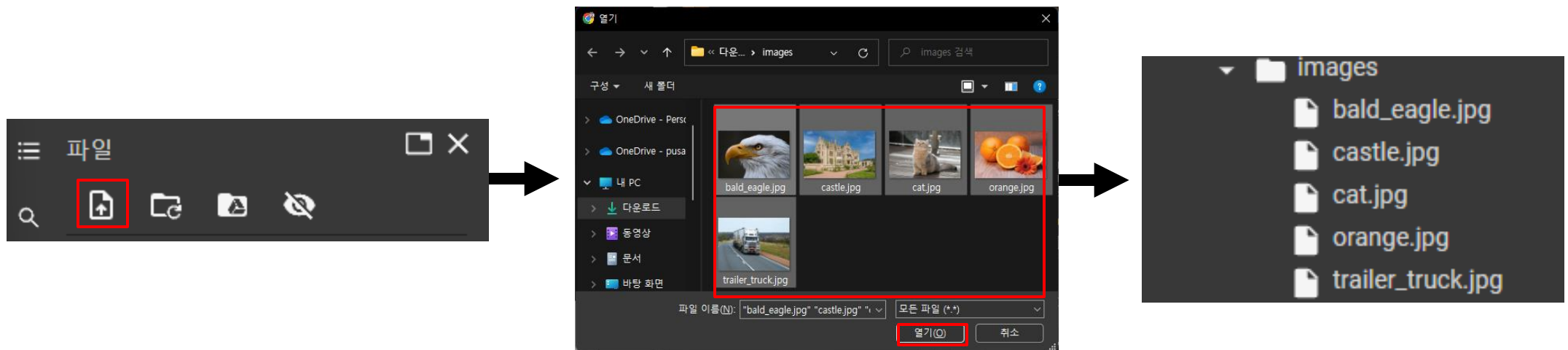


# 부록 - 코랩 개발환경 사용하기

## ❖ 개발 환경 설정

### ▪ 업로드 하기

- /content/images 로 파일 옮기기



# 부록 - 코랩 개발환경 사용하기

## ❖ 개발 환경 설정

- 다운로드 하기

