Lab 1: Basic Frameworks

<삼성 AI전문가 교육과정> 실습
서울대학교 바이오지능 연구실 (장병탁 교수)
최원석, 김윤성

2022.06.09

Biointelligence Laboratory
Dept. of Computer Science and Engineering
Seoul National University

BI
BIOINTELLIGENCE

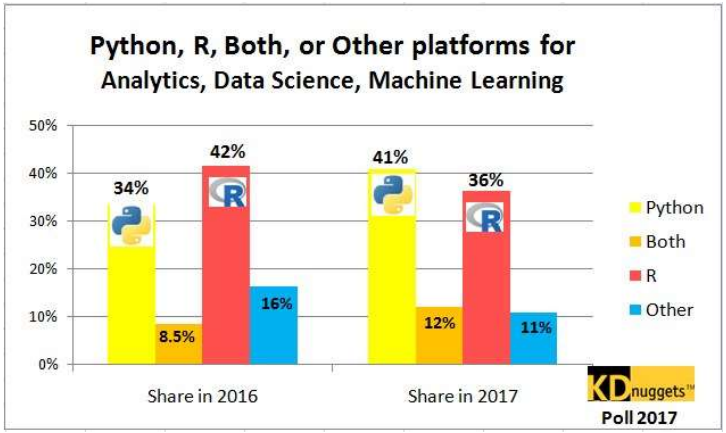
Contents

- Google Colab
- Numpy
- Pytorch basics

1-1

Google Colab 사용법

Programming language for Machine Learning



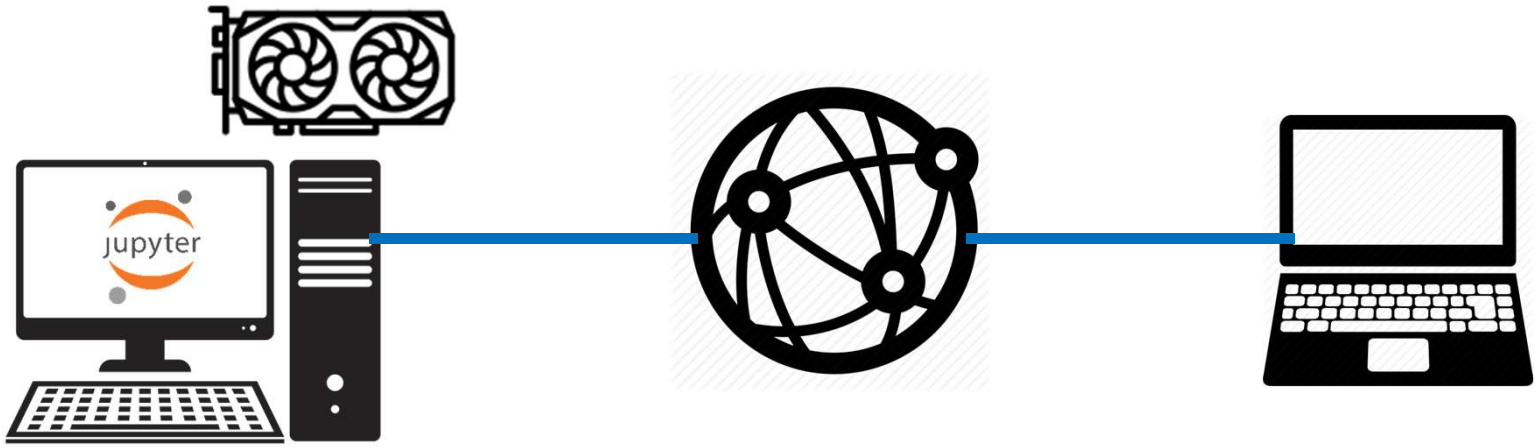
출처 - http://artificialintelligencemania.com/2018/07/02/the-best-programming-language-for-machine-learning/

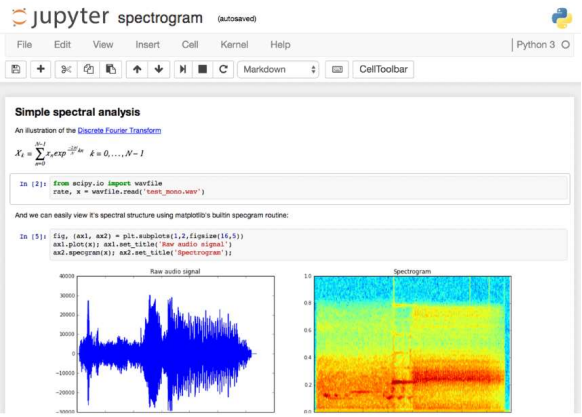
- Python
  - ML 연구 분야에 있어서 대체하기 어려운 프로그래밍 언어
  - Tensorflow, Pytorch 등의 딥러닝 프레임워크
  - Numpy, Jupyter Notebook, Matplotlib, Pandas, …

Jupyter Notebook?

- 웹 브라우저에서 파이썬 코드를 작성하고 실행해 볼 수 있는 개발도구
  - 원격 코딩 가능
  - 코드 블록 단위로 실행 / 디버깅
  - Text block을 이용한 문서화
  - Figure plotting 등 GUI

147.46.123.123:8888

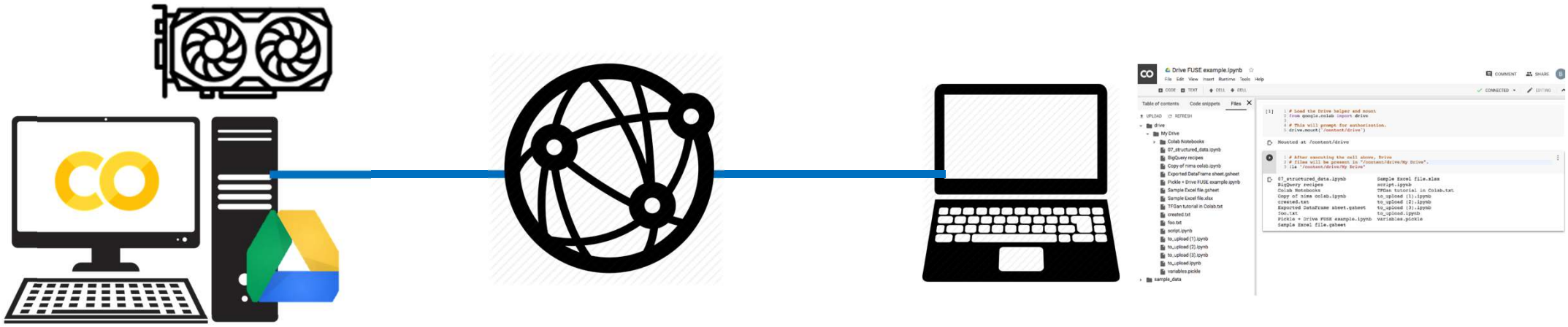




147.46.123.123

Google Colab?

- Google Colaboratory = Google Drive + Jupyter Notebook
  - 구글 계정 전용의 가상 머신 지원 – **GPU 포함**
  - Google drive 문서와 같이 링크만으로 접근 / 협업 가능
  - 약간의 딜레이 존재



Google server

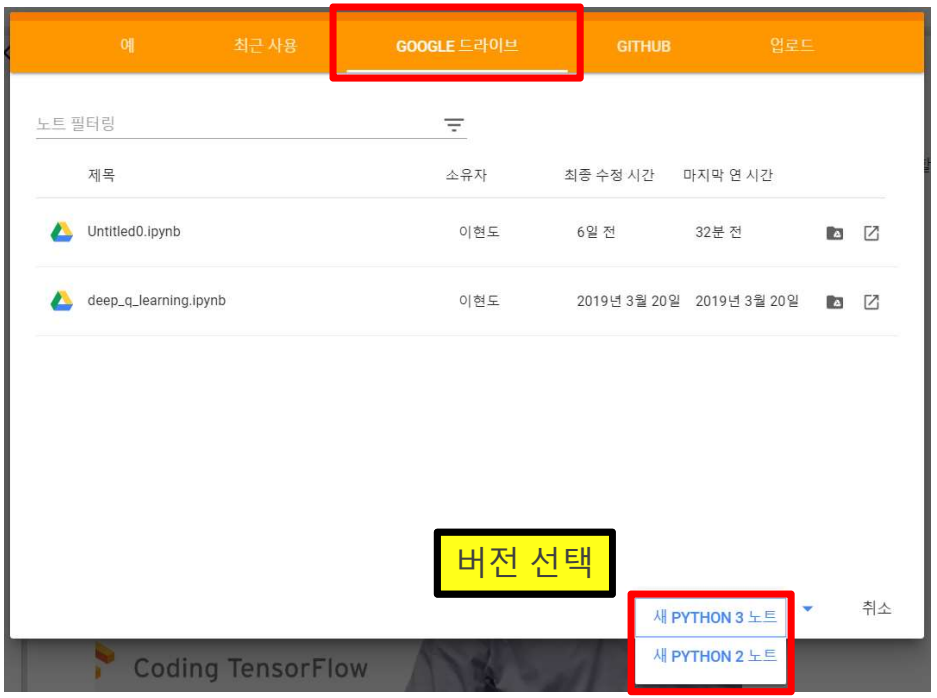
Google Colab - 사용법

- 개인 구글 계정 필요

- Colab과 Jupyter Notebook 사용 방법은 유사한 부분이 많음
  - 실습 수업에서는 Colab 위주로 설명
    - 로컬에서 작업 시 패키지 설치과정 필요
    - Colab의 경우 머신러닝에 요구되는 패키지 다수가 미리 설치되어 있음
  - GPU가 내장된 서버를 사용할 수 있을 시 로컬에서 Jupyter Notebook 사용이 더 편함

Google Colab - 사용법

- 파일 생성/접근 방법
  - 개인 구글 계정으로 접속
  - https://colab.research.google.com 접속
  - GOOGLE 드라이브 탭 이동
  - 새 PYTHON 노트 선택
    - 실습은 python3로 진행할 예정



Google Colab - 사용법

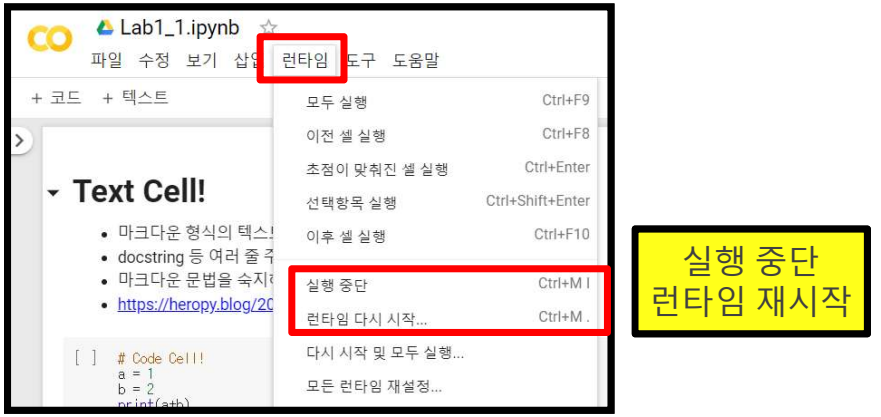
- 파일 이름 변경

- Code cell, Text cell
  - .ipynb 파일은 code cell과 text cell로 구성
  - 각 셀 하단에 마우스를 대거나, 화면 좌상단 버튼으로 셀 추가 가능
  - 셀 선택(마우스) 후 셀 우상단 삭제버튼으로 셀 삭제 가능

Google Colab - 사용법

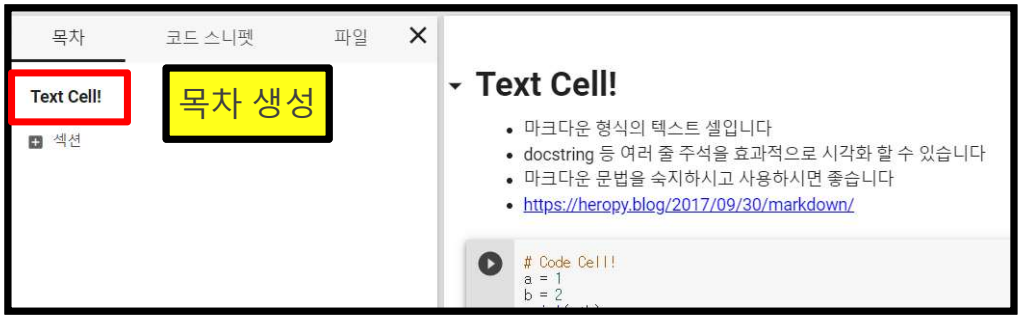
- Code cell
  - 일반적인 파이썬 코딩 방식과 동일
  - 각 셀은 한번에 실행할 단위를 뜻함
  - 실행 이후에도 메모리는 유지되어 다른 셀 실행 시 영향을 줌
    - 런타임 다시 시작 시 초기화

  - 상단 메뉴의 런타임
    - 실행 중인 셀 중단
    - 런타임 다시 시작



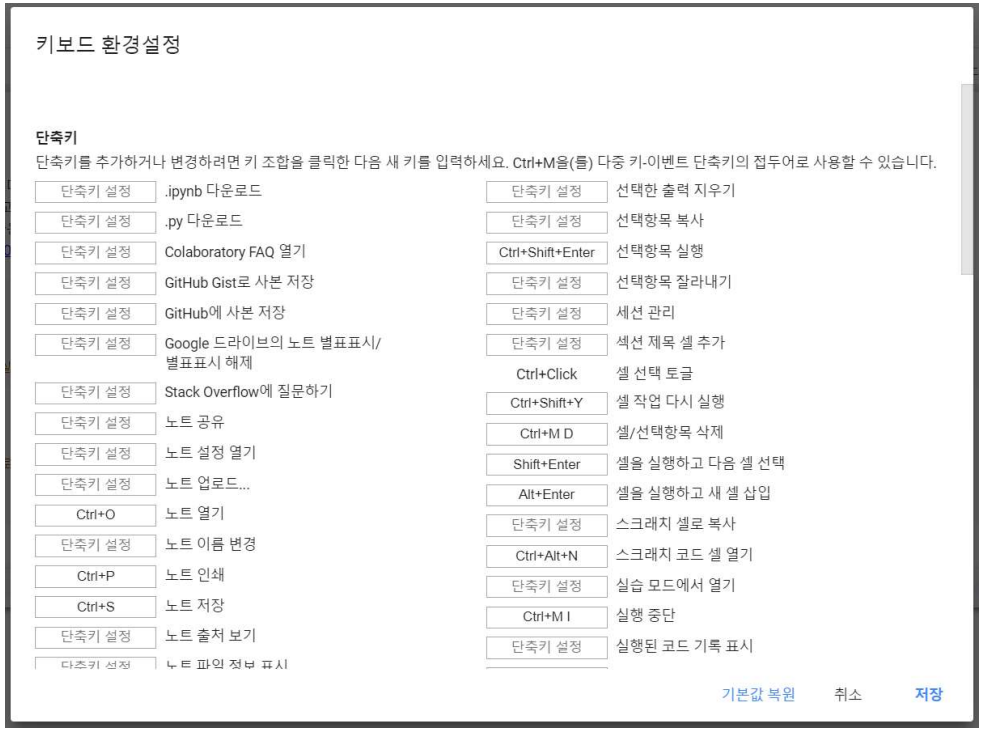
Google Colab - 사용법

- Text cell
  - 여러 줄 주석의 효과적인 시각화
  - 마크다운(Markdown) 문법
  - 자동 목차 생성



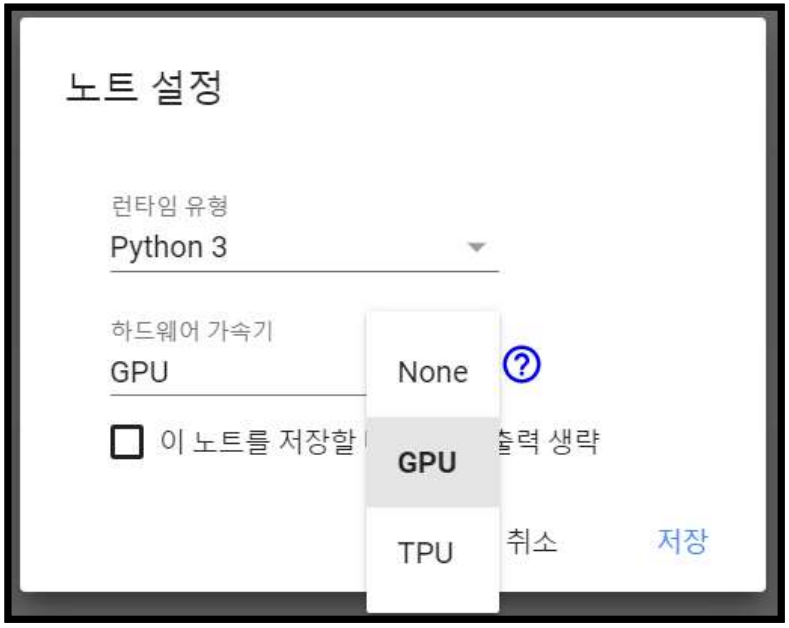
Google Colab - 사용법

- 단축키
  - 대부분의 작업은 단축키로 실행 가능
  - 단축키 설정 가능
  - 단축키 설정화면 – Ctrl+M H
  - 유용한 단축키
    - 코드 셀 생성 – Ctrl+M A(B)
    - 코드 셀 실행 – Ctrl+Enter
    - 셀 삭제 – Ctrl+M D
    - 실행중인 셀 중단 – Ctrl+M I
    - 런타임 다시 시작 – Ctrl+M .
    - 코드(텍스트) 셀로 변환 – Ctrl+M Y(M)
    - 마지막 셀 작업 실행취소 – Ctrl+Shift+Z



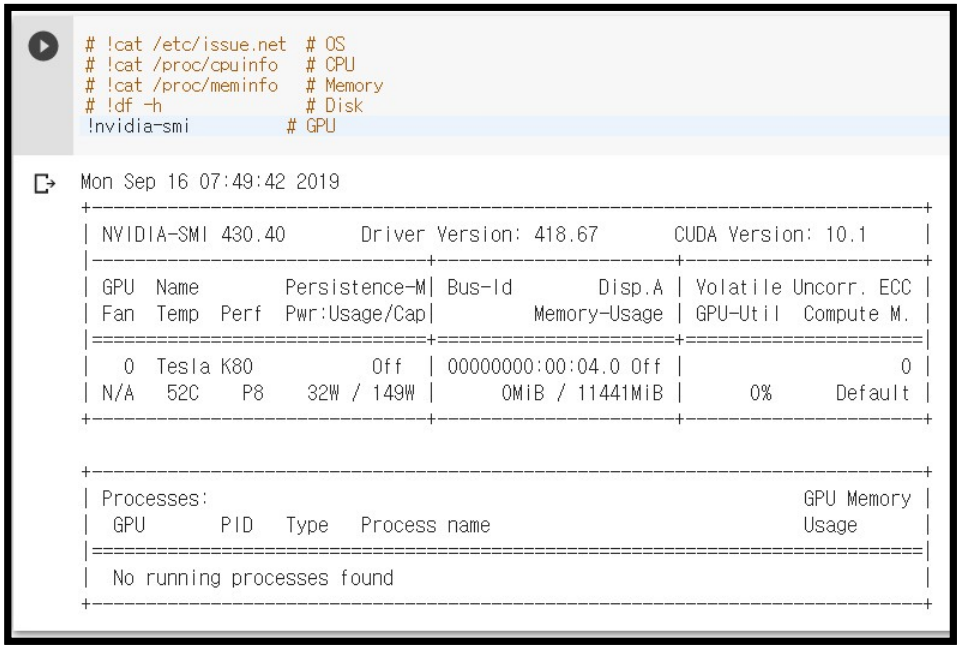
Google Colab - 사용법

- GPU 설정
  - 런타임 -> 런타임 유형 변경 -> 하드웨어 가속기를 GPU로 변경
  - 유의사항 – GPU는 최대 12시간 실행을 지원
    - 12시간 실행 이후에는 런타임 재시작으로 VM을 교체해야 함



Google Colab - 사용법

- 명령어 실행하기
  - 코드 셀에 !를 붙이고 터미널 명령어를 입력하여 실행하면 터미널에서 실행하는 것과 같은 결과가 출력됨
  - 예외로 cd 명령어는 %cd '/your/desired/path'



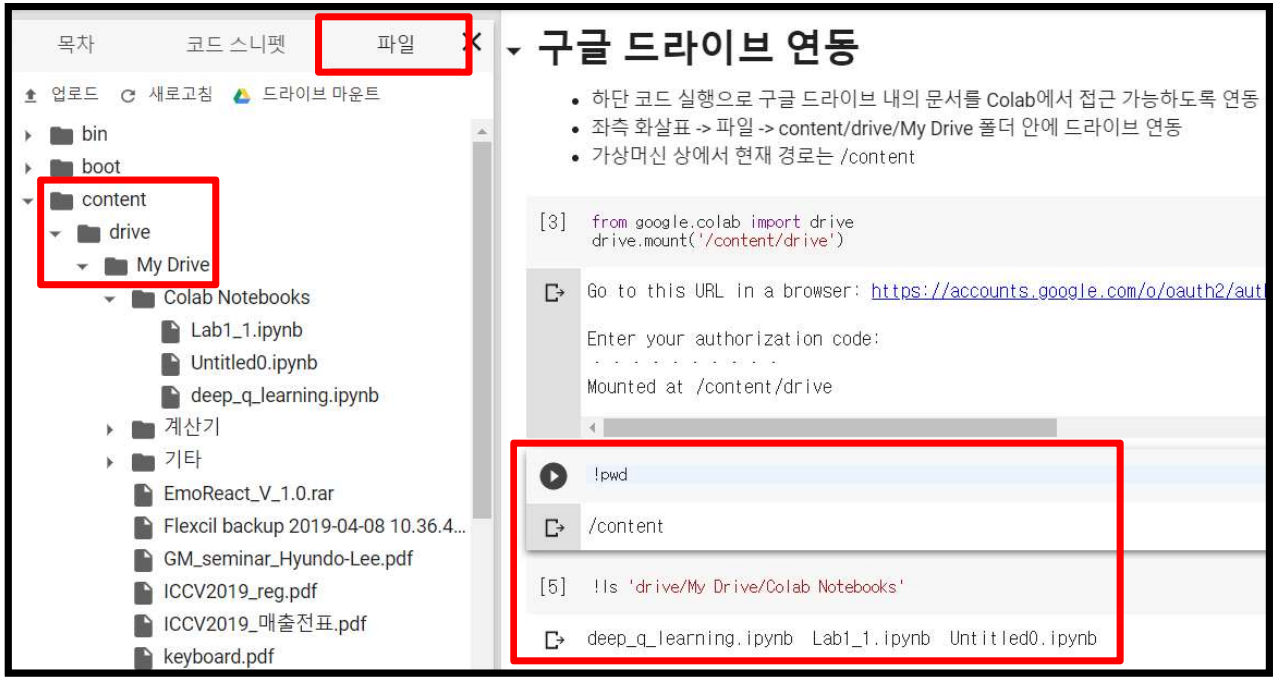
```
# !cat /etc/issue.net  # OS
# !cat /proc/cpuinfo   # CPU
# !cat /proc/meminfo   # Memory
# !df -h               # Disk
!nvidia-smi          # GPU
```

```
Mon Sep 16 07:49:42 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 430.40       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   52C    P8    32W / 149W |      0MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

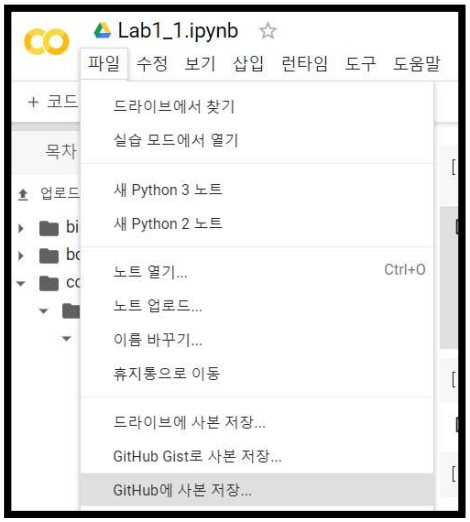
Google Colab - 사용법

- 구글 드라이브 연동
  - 간단한 인증 절차 이후 구글 드라이브의 파일을 Colab에서 접근 가능



Google Colab - 사용법

- Github 연동
  - 단일 .ipynb 파일을 clone 하는 방법
    - https://github.com/~~~ 부분을 https://colab.research.google.com/github/~~~ 로 교체
    - 파일 -> 드라이브에 사본 저장
  - 전체 repository cloning
    - !git clone project.git
  - github repository에 파일을 올리는 방법
    - 파일 - Github에 사본 저장 선택
    - 저장소, 브랜치, 경로 지정

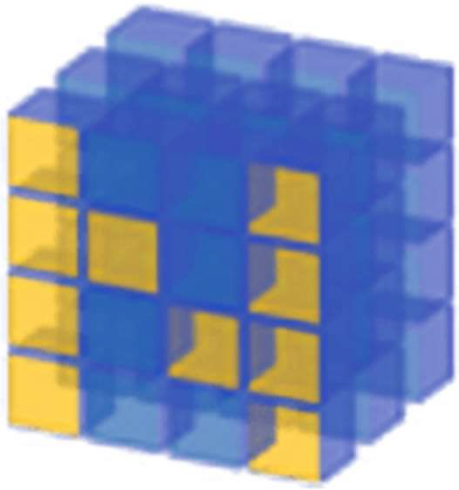


Google Colab - tips

- 장시간 사용할 때의 issue들

> - The **'maximum lifetime'** of a running notebook is **12 hours** (browser open)
> - An **'Idle'** notebook instance cuts-off after **90 minutes**
> - You can have a maximum of **2 notebooks** running concurrently
> - If you close the notebook window and open it while the instance is still running, the cell outputs and variables will still persist. However if the notebook instance has been recycled, your cell outputs and variables will no longer be available.

  - GPU 사용/미사용 관계없이 최대 12시간
  - 아무것도 안 하는 idle 상태 돌입 이후 90분에 런타임 자동으로 shutdown
    - Shutdown 전에 창을 끄고 다시 켜도 런타임은 유지됨
  - 동시에 최대 2개의 notebook 가동 가능

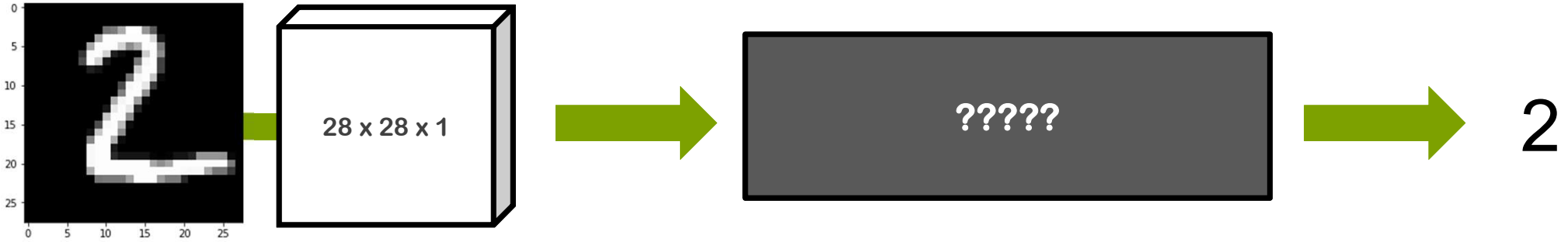


1-2

Numpy, Pytorch tensor

Machine Learning

- 예시 – Object classification

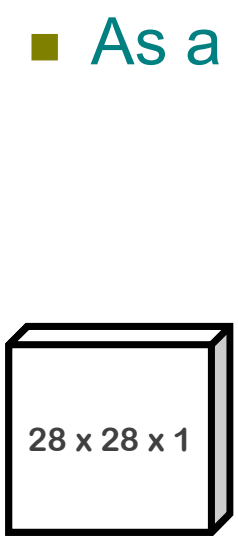


  - Rule-based system
  - Computer vision

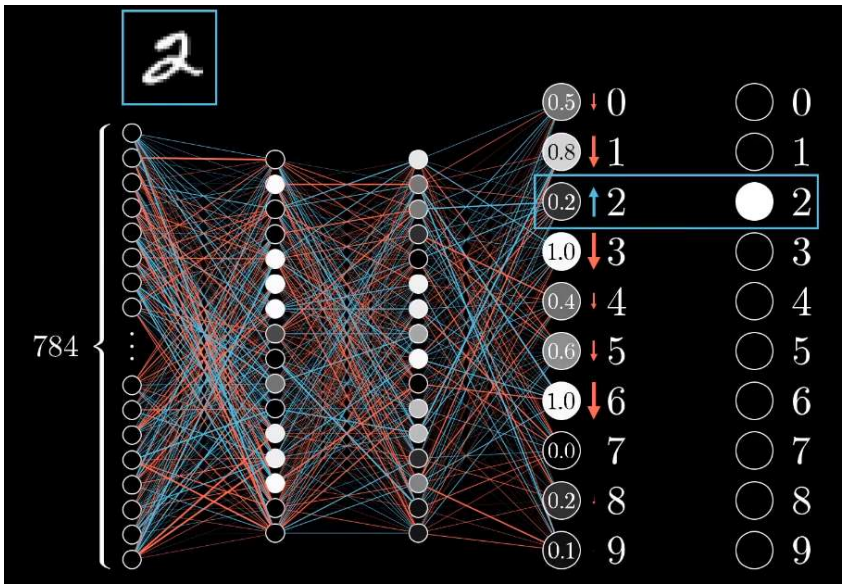
  - Machine Learning
    - Function approximation
    - Generalization

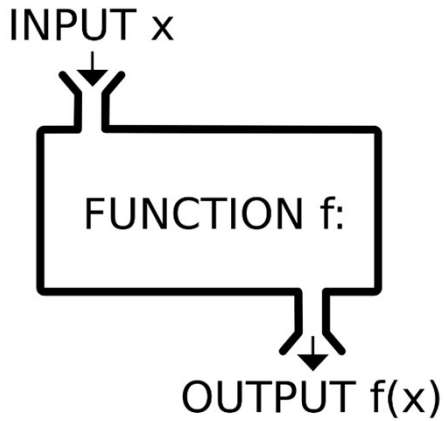
Machine Learning

- As a function approximator



Input:
[28, 28, 1] **tensor**





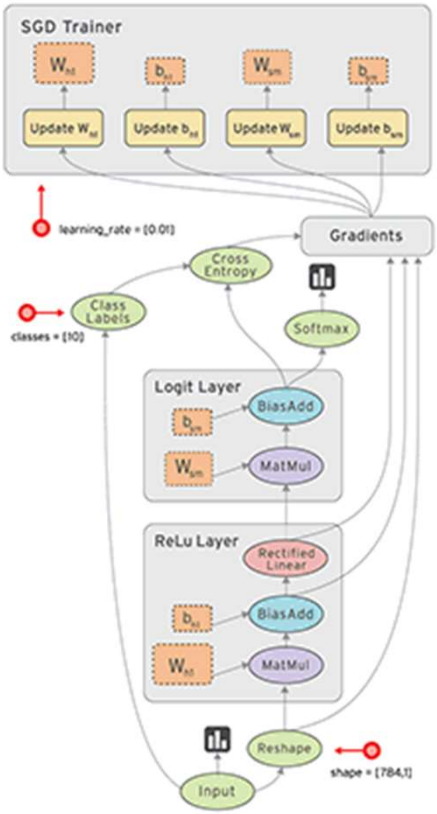
Label:
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]

Output: [10] **tensor**
[0.5, 0.8, 0.2, **1.0**, 0.4, 0.6, **1.0**, 0.0, 0.2, 0.1]

Machine Learning

- Tensor?
  - Multi-dimension array (vector, matrix)
  - 특별한 케이스가 아니라면, 딥러닝 프레임워크는
    - tensor 형태의 입력과
    - tensor 형태의 모델 가중치를 적절히 연산하여
    - tensor 형태의 출력을 내는 과정을 반복함
  - Tensor 형태의 데이터를 잘 다룰 수 있어야 함

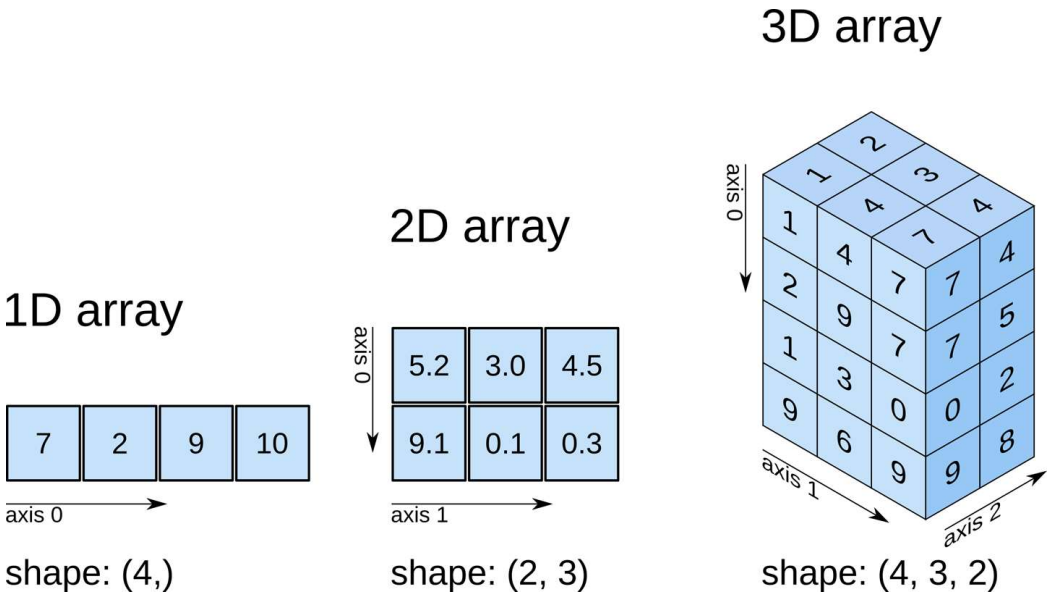
- Graph?
  - 여러 computation을 위한 dataflow의 구조
  - TensorFlow / PyTorch



Numpy

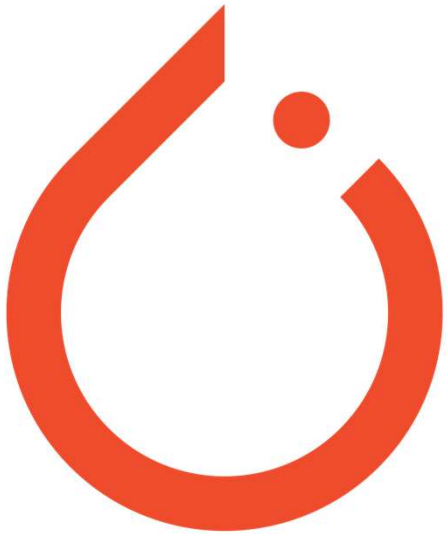
- Numpy
  - Numpy는 multi-dimensional array 연산을 편리하게 하기 위한 라이브러리
  - Numpy 모듈 설치 - *pip3 install numpy*
    - Colab에는 기본적으로 설치되어 있다.
  - Import - *import numpy (as np)*

Numpy array



출처:
https://www.safaribooksonline.com/library/view/elegant-scipy/9781491922927/ch01.html

- numpy array는 모양(shape)과 데이터 타입(dtype)을 가진다.
  - 길이가 n인 정수 벡터 – shape: n, dtype: *np.int32*
  - $n \times m$ 실수 행렬 – shape: (n, m), dtype: *np.float32(64)*
  - $n \times m \times k$ 복소수 텐서 – shape: (n, m, k), dtype: *np.complex64*



1-3

Pytorch basics

Machine Learning Frameworks

- PyTorch : Facebook이 개발한 머신 러닝 프레임워크
  - **Define-by-Run**
  - Dynamic graph

- TensorFlow (v1) : Google이 개발한 머신 러닝 프레임워크
  - **Define-and-Run**
  - Static graph

- TensorFlow (v2) : 2019년 출시
  - PyTorch와 비슷한 방식으로 변경

PyTorch vs Tensorflow - Paradigm

- PyTorch
  - Define-by-Run
    - 정의하면서 값을 함께 넣어줌

- Tensorflow
  - Define-and-Run
    - 그래프를 먼저 생성한 후, 실제로 run 할 때 값을 넣어줌

PyTorch

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
y = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
z = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

TensorFlow

```
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

with tf.device('/gpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

출처 :
https://m.blog.naver.com/PostView.nhn?blogId=fastcampus&logNo=221029365132&proxyReferer=https%3A%2F%2Fwww.google.com%2F

PyTorch vs Tensorflow - Graph

- PyTorch
  - Dynamic graph
    - 매 iteration마다 새로운 graph를 생성

PyTorch: Normal Python

```
N, D, H = 3, 4, 5

x = Variable(torch.randn(N, D))
w1 = Variable(torch.randn(D, H))
w2 = Variable(torch.randn(D, H))

z = 10
if z > 0:
    y = x.mm(w1)
else:
    y = x.mm(w2)
```

- Tensorflow
  - Static Graph
    - 매 iteration마다 처음부터 구축된 graph를 이용해 계산

Tensorflow

```
N, D, H = 3, 4, 5
x = tf.placeholder(tf.float32, (N, D))
w1 = tf.placeholder(tf.float32, (D, H))
w2 = tf.placeholder(tf.float32, (D, H))

z = tf.constant(10)
y = tf.cond(z > 0, lambda: tf.matmul(x, w1), lambda: tf.matmul(x, w2))
```

PyTorch vs Tensorflow - Documentation

- PyTorch
  - Conv2d의 정의와 수식이 좀 더 자세하고, output의 정확한 shape, Conv2d의 용례, 모를 만한 용어에 대한 link도 달려있음.

Conv2d

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

- `dilation` controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but this link has a nice visualization of what `dilation` does.

Shape:

- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

- Tensorflow

tf.nn.conv2d

View source on GitHub

Computes a 2-D convolution given 4-D `input` and `filters` tensors.

Aliases:

- `tf.compat.v2.nn.conv2d`

```
tf.nn.conv2d(
    input,
    filters,
    strides,
    padding,
    data_format='NHWC',
    dilations=None,
    name=None
)
```

Given an input tensor of shape `[batch, in_height, in_width, in_channels]` and a filter / kernel tensor of shape `[filter_height, filter_width, in_channels, out_channels]`, this op performs the following:

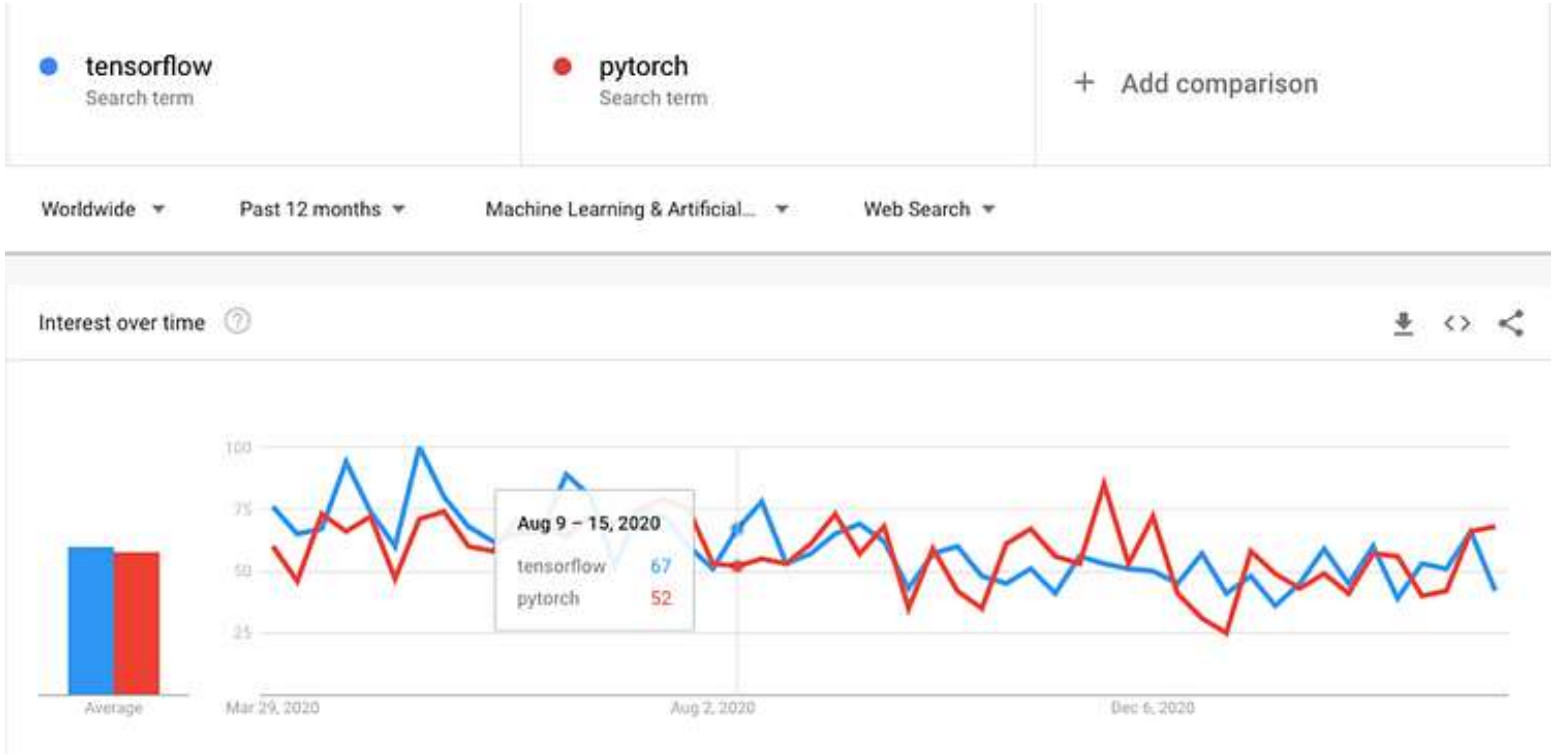
1. Flattens the filter to a 2-D matrix with shape `[filter_height * filter_width * in_channels, output_channels]`.
2. Extracts image patches from the input tensor to form a *virtual* tensor of shape `[batch, out_height, out_width, filter_height * filter_width * in_channels]`.
3. For each patch, right-multiplies the filter matrix and the image patch vector.

In detail, with the default NHWC format,

```
output[b, i, j, k] =
    sum_{di, dj, q} input[b, strides[1] * i + di, strides[2] * j + dj, q] *
                    filter[di, dj, q, k]
```

Must have `strides[0] = strides[3] = 1`. For the most common case of the same horizontal and vertices strides, `strides = [1, stride, stride, 1]`.

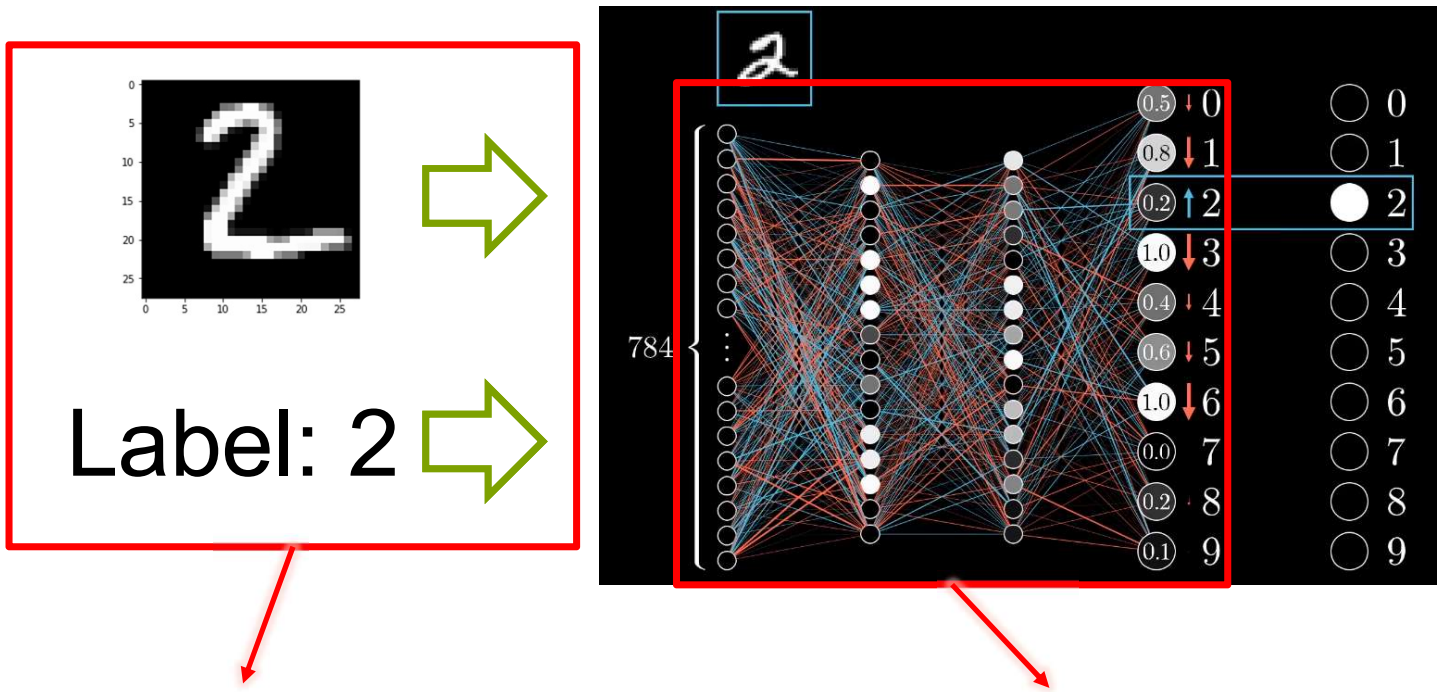
PyTorch vs Tensorflow - Trends



- 거의 비슷한 선호도와 사용빈도를 가짐
  - 실습에서는 PyTorch 프레임워크를 사용

Pytorch basics

- Pytorch를 활용한 딥러닝 프로그램의 기본 구조



Dataset
Input, Labels

Model
Structure, Weights

Output → Label
Optimizing model weight
Training Phase

Test Phase
Evaluating output

Pytorch basics

- Pytorch를 활용한 딥러닝 프로그램의 기본 구조
  - Model(nn.Module) : Class
    - 전체적인 인공신경망의 구조를 선언하는 부분
    - __init__, forward 등의 method는 꼭 필요함
  - Dataset : Class
    - 학습에 필요한 데이터셋을 선언하는 부분
    - __init__, __len__, __getitem__의 method는 꼭 필요함
  - Training
    - 학습을 진행하는 부분
  - {Validation & Test}
    - 학습된 결과를 확인하는 부분

Pytorch basics

- Model

```
Class Model(nn.Module):
    def __init__(self, HYPERPARAMETERS):
        super(Model, self).__init__()
        …SOME DECLAIRATION…


    def forward(self, INPUTS):
        …SOME COMPUTATIONS…
        return OUTPUT
```

Pytorch basics

- Dataset

```
Class Dataset(Dataset):
    def __init__(self, PARAMETERS):
        …SOME DECLAIRATION…


    def __len__(self):
        return length_of_dataset


    def __getitem__(self, index):
        return Dataset[index]
```

Pytorch basics

- Training part

```
DECLAIR DATASET, DATALOADER
DECLAIR HYPERPARAMETERS

model = Model(HYPERPARAMETERS)
optimizer = OPTIMIZER(model.parameters(), lr=LEARNING_RATE)

for epoch in range(NUM_EPOCHES):
    for (INPUTS, REAL_VALUES) in DATALOADER:
        optimizer.zero_grad()
        OUTPUTS = model(INPUTS)
        loss = LOSS_FUNCTION(OUTPUT, REAL_VALUES)
        loss.backward()
        optimizer.step()
```

Codes

- 실습 코드 Github 주소
  - https://github.com/yskim5892/AI_Expert_2022