

# Pytorch를 활용한 딥러닝 학습 환경 구축 및 실습

- 1일차 -

강 수 명

smgang.kmu@gmail.com

# C

## ONTENTS

- I \_ 시작하며
- II \_ 딥러닝이란?
- III \_ 딥러닝과 프레임워크
- IV \_ 개발 환경 설정
- V \_ Pytorch 시작하기
- VI \_ Classification 문제 접근하기





배포용 자료 (일 별로 자료 업로드 예정)

<https://url.kr/grs716>

내 드라이브 > KMU\_Pytorch\_특강 ▾



이름 ↑

소유자

내가 마지막으로 수정한 날짜

파일 크기



1일차 참고자료

나

오전 12:08

—

# 강수명 (smgang.kmu@gmail.com)

## 이력

**2007.03~2010.02** 계명대학교 게임모바일콘텐츠학과 졸업 (공학사)

**2010.03~2013.08** 계명대학교 미디어아트학과 (게임모바일전공) 졸업 (게임학석사)

졸업논문 : 방향성 특징 기술자를 이용한 식물 잎 분류 (지도교수 이준재)

**2016.09~2022.02** 계명대학교 컴퓨터공학과 (모바일소프트웨어전공) 졸업 예정 (공학박사)

졸업논문 : Knowledge Distillation을 활용한 Anchor Free Continual Learning 및 응용 (지도교수 이준재)

**2014.03~2017.01** (주)지오씨엔아이 공간정보기술연구소

## 관심 분야

딥러닝, 패턴인식, 영상처리, 게임 응용

## 세부 관심 분야

- 지속적으로 학습 가능한 딥러닝 문제
- 지리정보+딥러닝 응용문제

# 01 시작하며

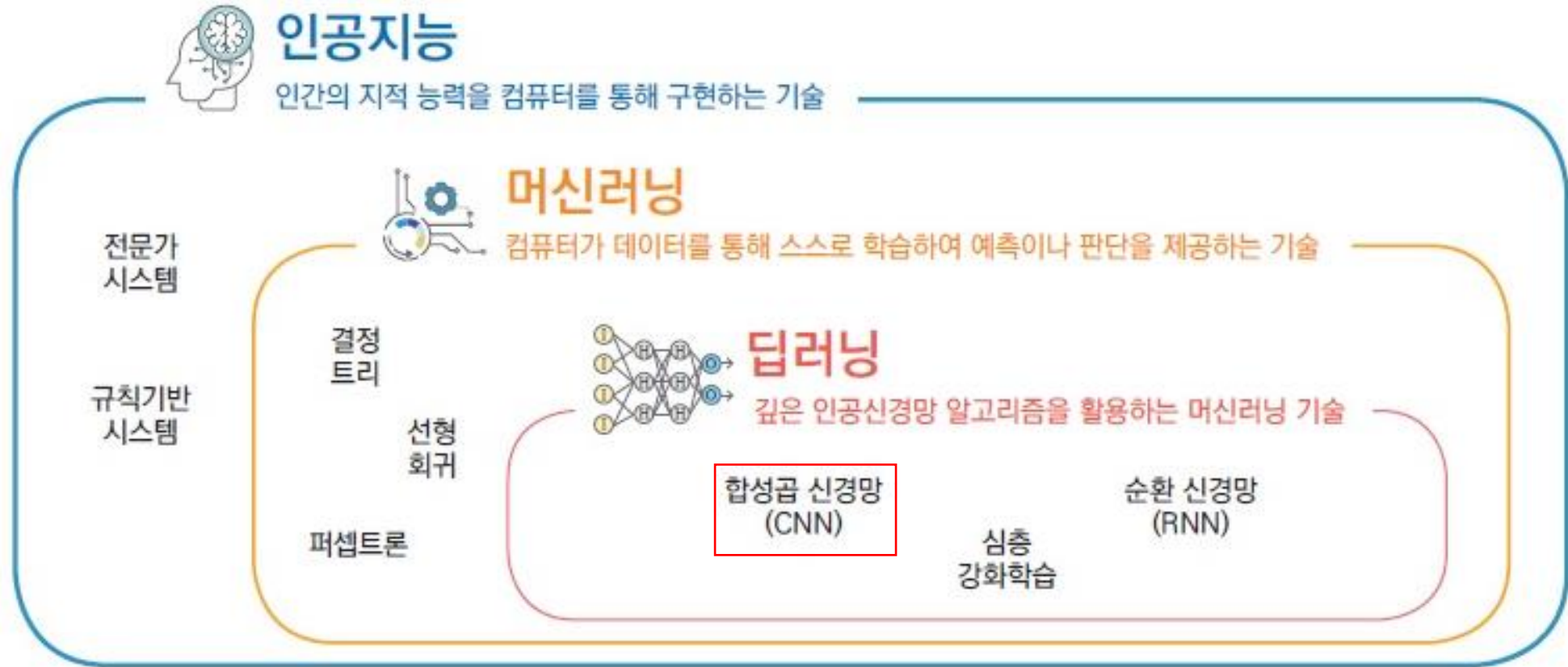
- 일시** : 2022년 2월 7일(월) ~ 11일(금) 일주일간, 오후 10시부터 12시까지  
**준비물** : 개인 노트북 (Colab시 맥, 윈도우 관계 없음)  
           개인 PC 환경은 윈도우 환경에서 진행(GPU 환경이 없을 시 CPU로 구동)  
**강의 환경** : 윈도우 환경 + COLAB

일정		내용	비고
1일차	10:00~10:50	딥러닝 프레임워크 및 Pytorch와 Pytorch 환경 설정	Colab환경
	11:00~11:50	Mnist 및 여러 데이터 활용 Classification(분류) 문제 해결	Colab환경
2일차	10:00~10:50	Object detection (물체 검출) 문제 해결하기 (1)	Colab환경
	11:00~11:50	Object detection (물체 검출) 문제 해결하기 (2)	Colab환경
3일차	10:00~10:50	생성적 적대적 모델(GAN)을 활용한 응용 (3)	Colab환경
	11:00~11:50	생성적 적대적 모델(GAN)을 활용한 응용 (4)	Colab환경
4일차	10:00~10:50	개인 PC 설정 및 환경 설정 (Anaconda+Pytorch)	개인 PC
	11:00~11:50	Git-hub와 여러가지 딥러닝 코드 맛보기	개인 PC
5일차	10:00~10:50	Git-hub에서 받은 코드 다루기 (1)	개인 PC
	11:00~11:50	Git-hub에서 받은 코드 다루기 (2)	개인 PC

- 딥러닝의 입문
- 딥러닝 공부할 때 고민되는 것들
  - 프레임워크
  - 학습 환경 : 컴퓨터 사양, GPU, 메모리
- 초심자로서의 딥러닝 : 어떻게 시작해야 하는지?
- 연구자로서의 딥러닝 : 무엇을 새로 만들 수 있을지?
- 실무자로서의 딥러닝 : 현장 시스템에 어떻게 적용할지

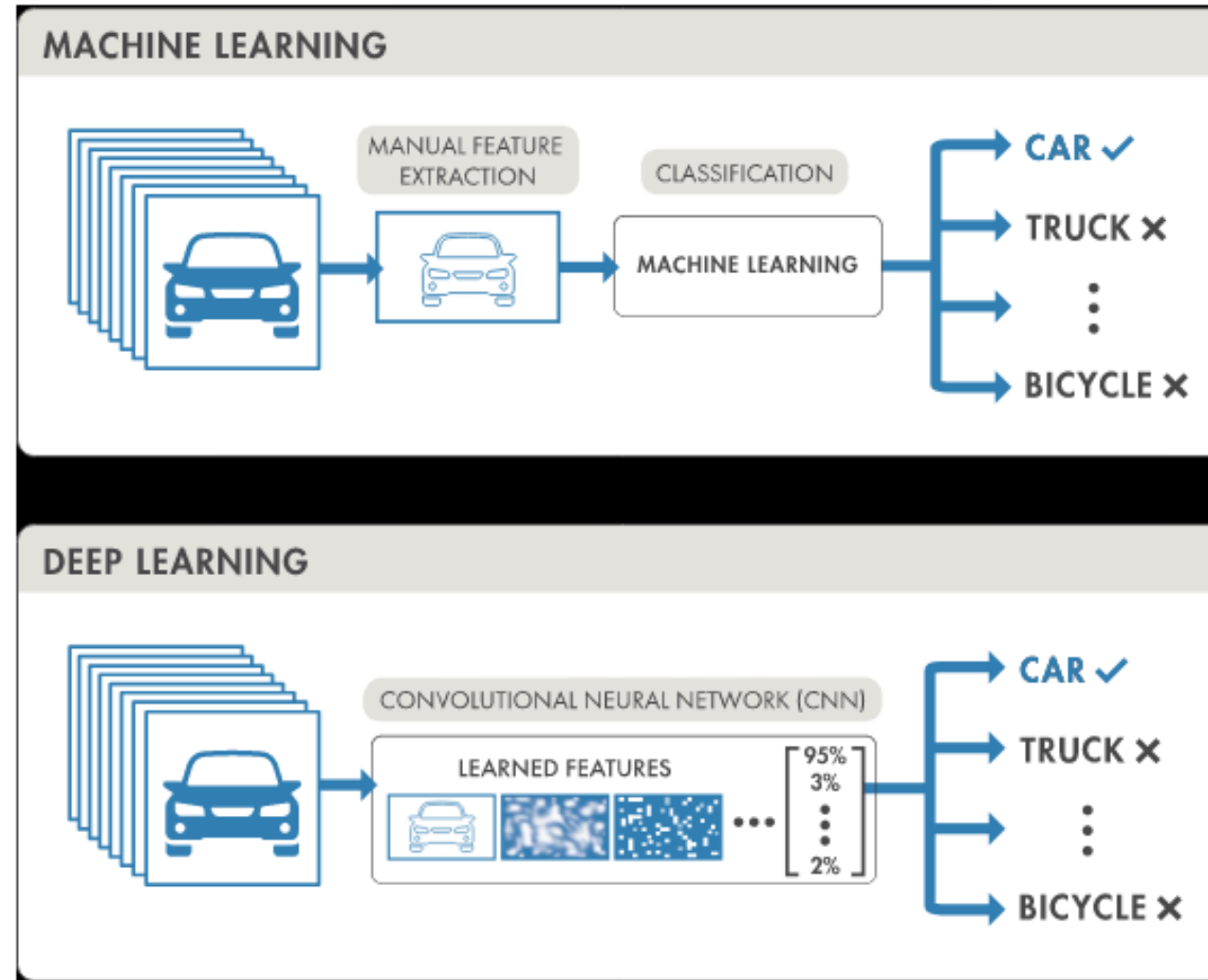
## 02 딥러닝이란?

### 인공지능, 머신러닝, 딥러닝의 관계



## 02 딥러닝이란?

### 기계학습과 딥러닝의 학습 방법 차이

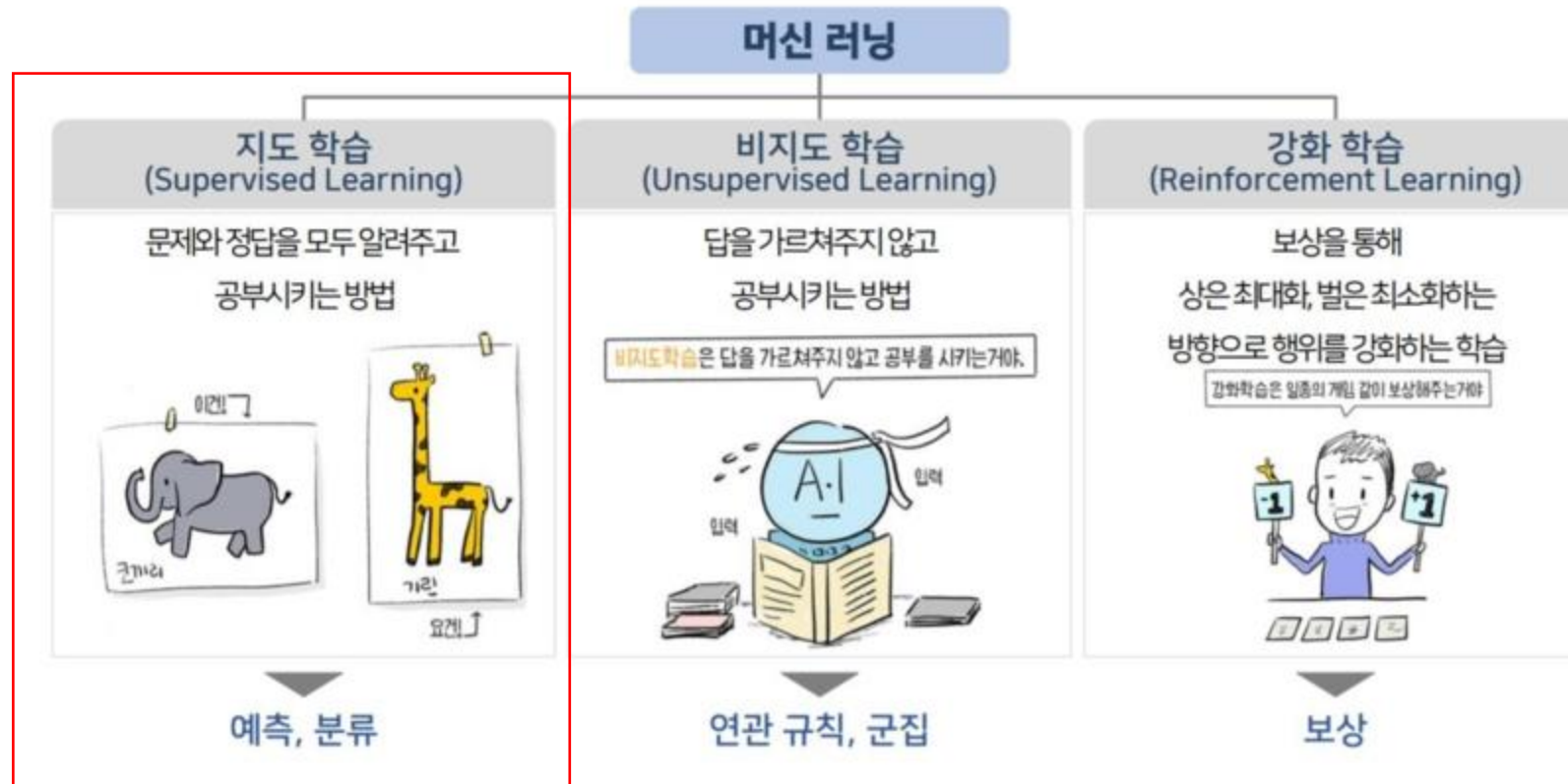


출처 : <https://arxiv.org/abs/1704.06857>



## 02 딥러닝이란?

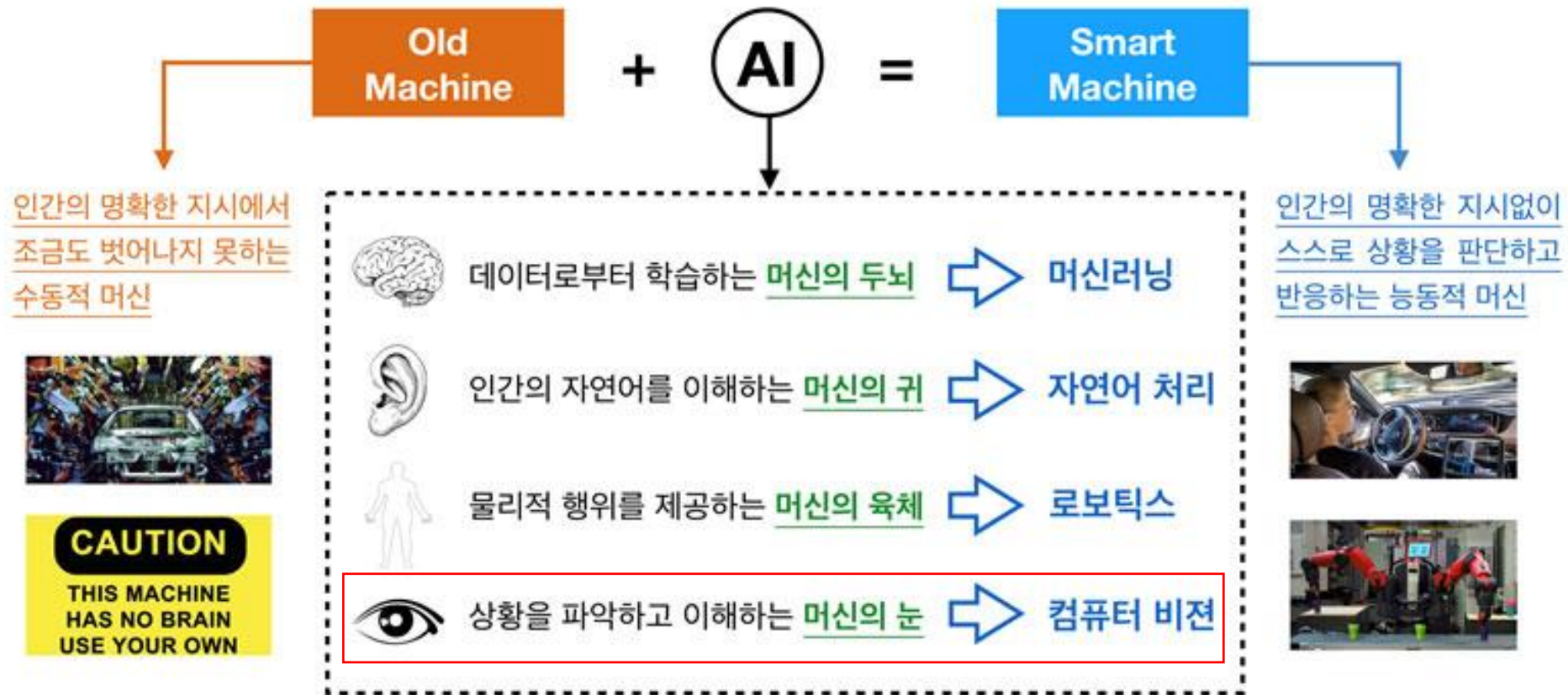
### 머신러닝의 여러가지 분야



출처 : <https://m.blog.naver.com/k0sm0s1/221863569856>

## 02 딥러닝이란?

### 머신러닝의 여러가지 응용분야



## 02 딥러닝이란?

### 이미지를 사용한 기계학습

**Semantic Segmentation**



CAT GRASS  
TREE

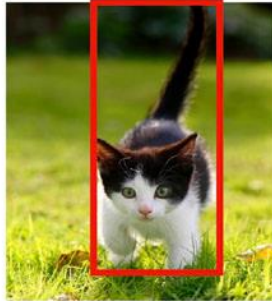
No object  
Just pixels

**Classification**



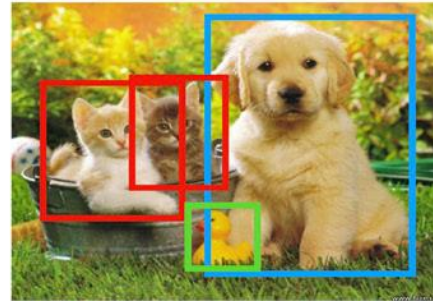
CAT

**Classification + localization**



CAT

**Object detection**



CAT DOG DUCK

**Instance segmentation**



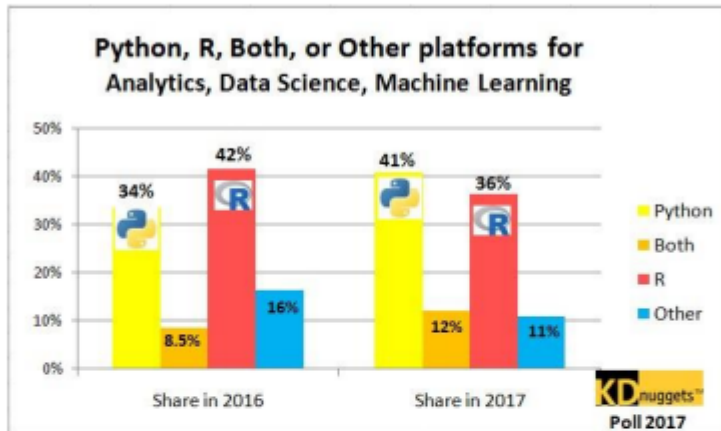
CAT CAT DOG DUCK

Single object

Multiple objects

출처 : [https://www.slideshare.net/darian\\_f/introduction-to-theartificial-intelligence-and-computer-vision-revolution](https://www.slideshare.net/darian_f/introduction-to-theartificial-intelligence-and-computer-vision-revolution)

# Programming language for Machine Learning



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



## ■ Python

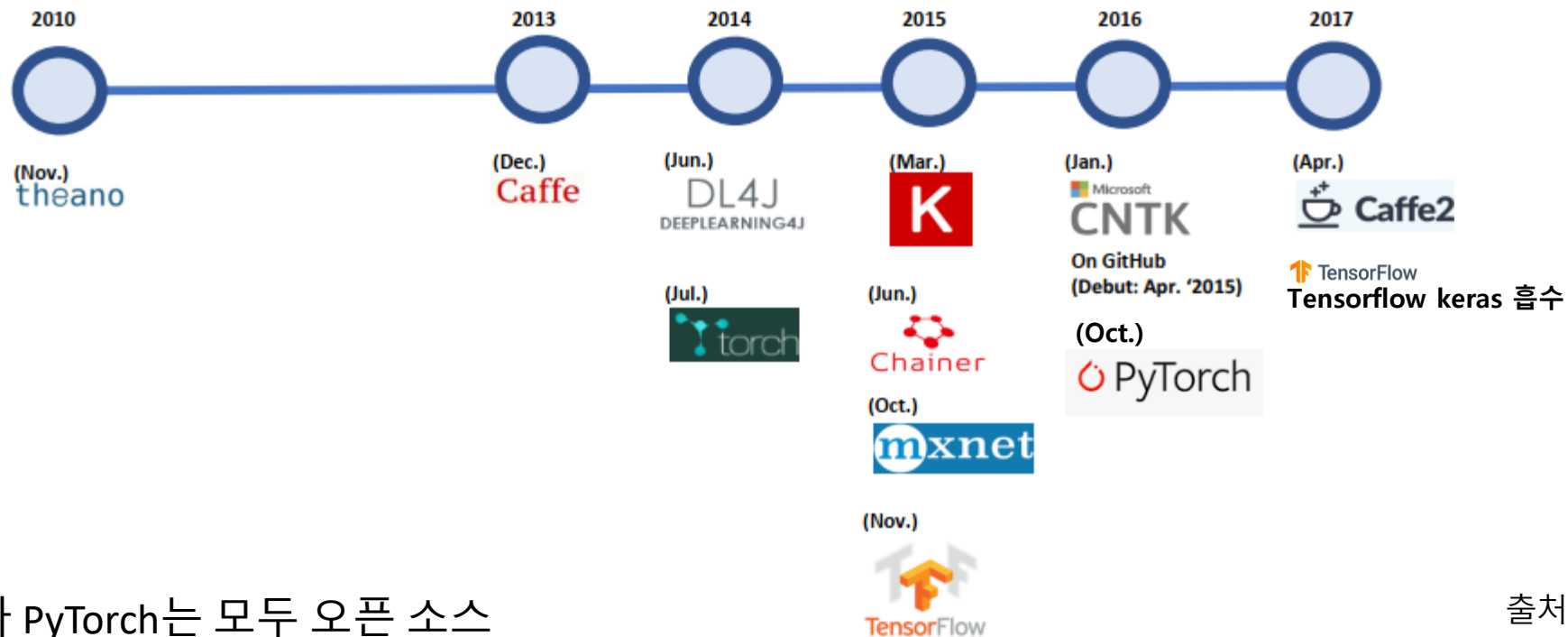
- ML 연구 분야에 있어서 대체하기 어려운 프로그래밍 언어
- Tensorflow, Pytorch 등의 딥러닝 프레임워크
- Numpy, Jupyter Notebook, Matplotlib, Pandas, ...



출처 : <https://bi.snu.ac.kr/Courses/ML2019/ML2019.html>

## 03 딥러닝과 프레임워크

### ◆ 시간순으로 본 딥러닝 프레임워크



- Tensorflow와 PyTorch는 모두 오픈 소스
- Tensorflow는 Theano를 기반, Google에서 개발
- PyTorch는 Torch를 기반으로 하고 Facebook에서 개발
- 중국 연구는 최근 바이두에서 만든 paddle을 사용

출처 :



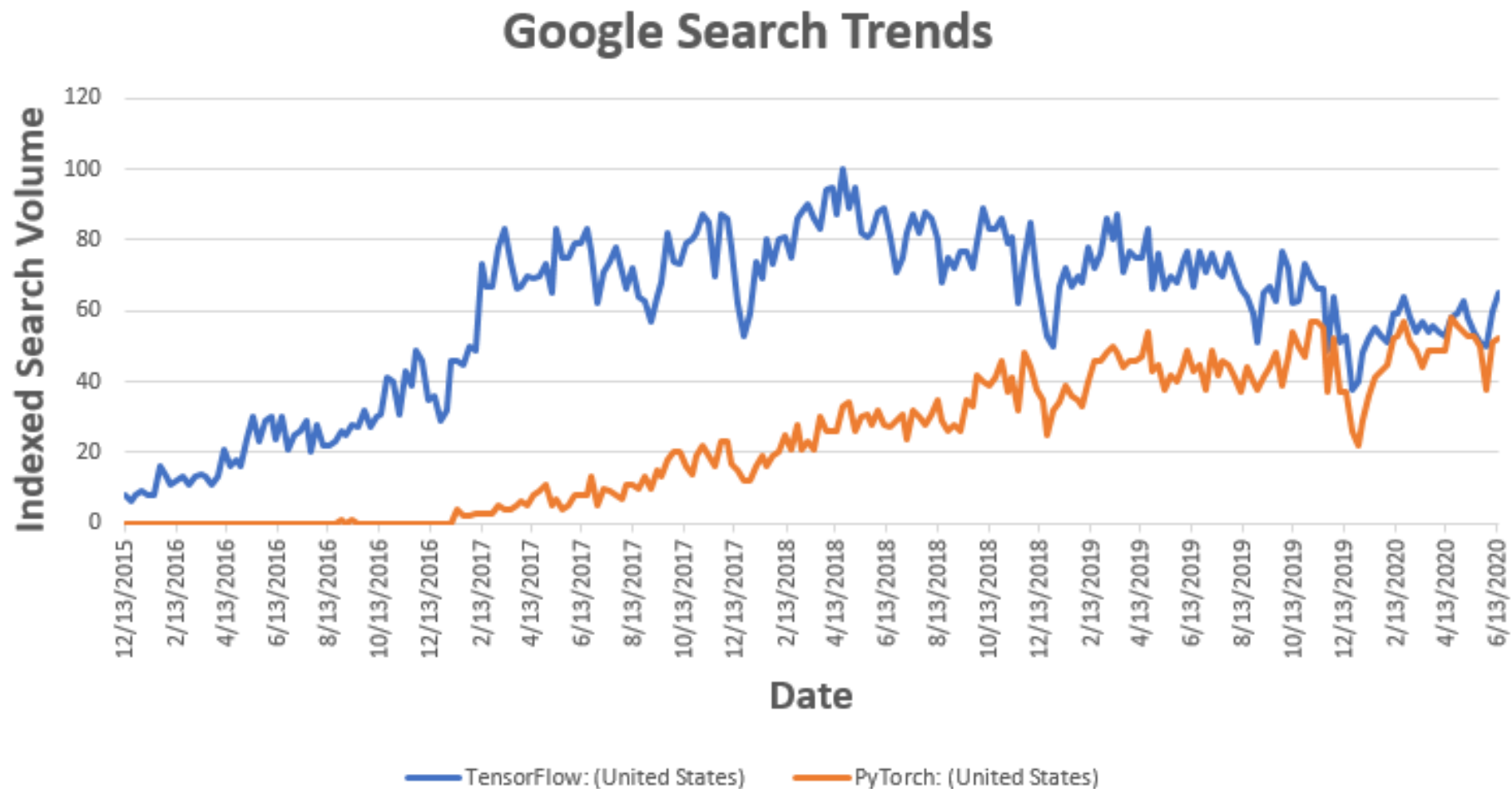
K-ICT 딥러닝 개요

송준이  
㈜ 아이먼티라이

Identify

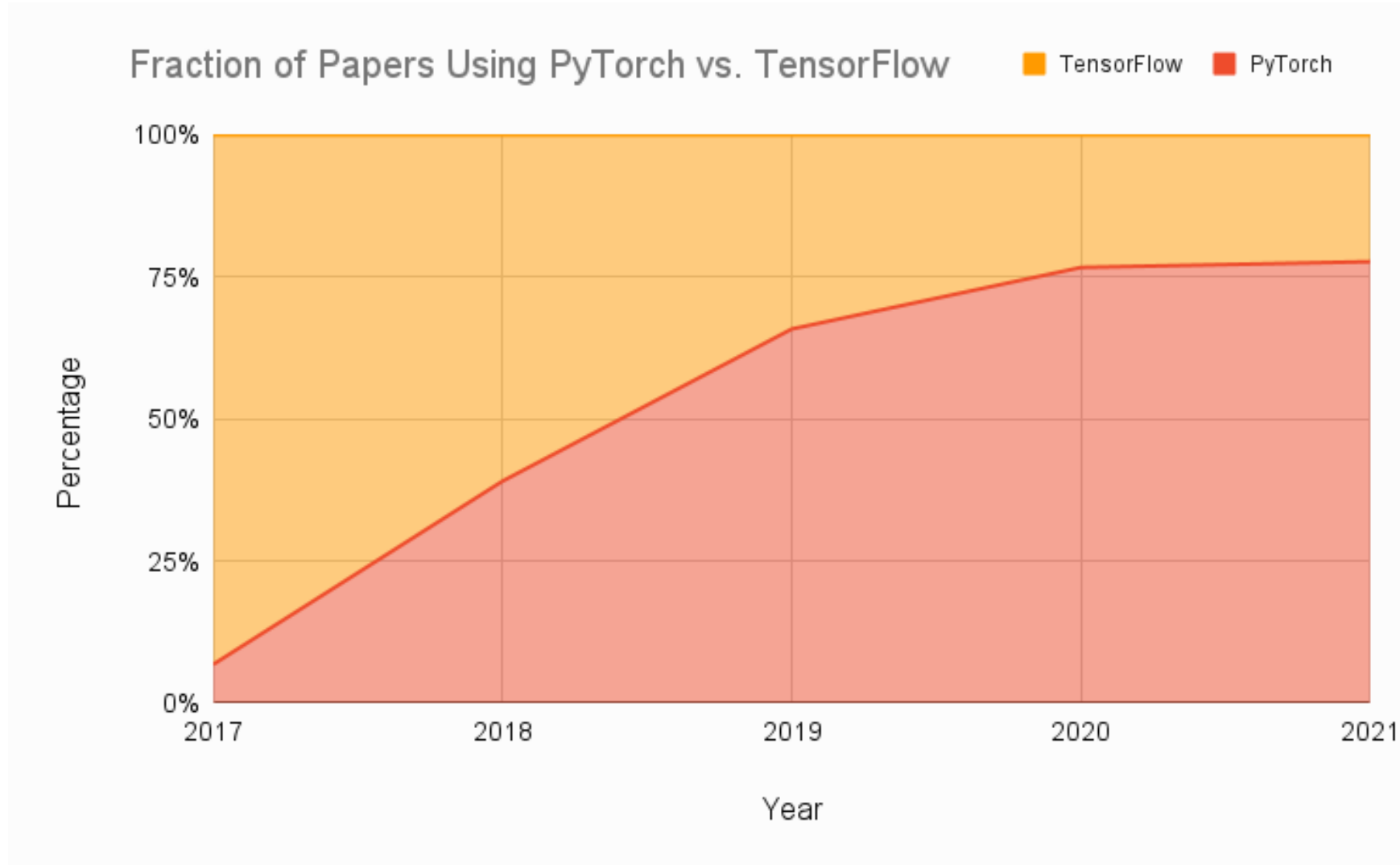


# 03 딥러닝과 프레임워크



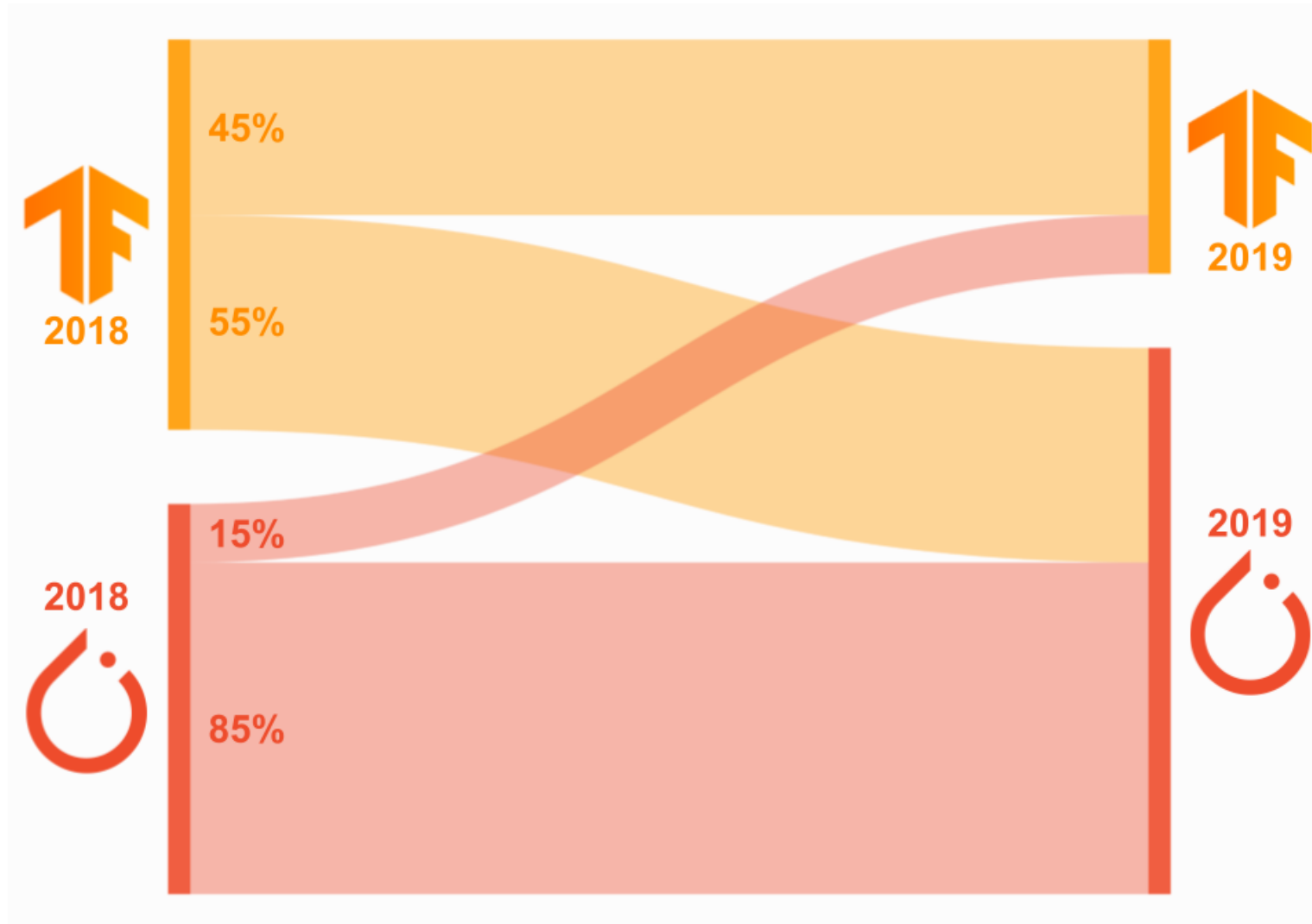
출처 : <https://wikidocs.net/156950>

1. **모델 가용성:** 딥 러닝의 영역이 매년 확장되고 모델이 차례로 커지면서 처음부터 최첨단 (SOTA, State-of-the-Art) 모델을 훈련하는 것은 더 이상 실현 가능하지 않습니다. 다행히 공개적으로 사용할 수 있는 SOTA 모델이 많이 있으며 가능한 한 이를 활용하는 것이 중요합니다.
2. **배포 인프라:** 성능이 좋은 모델을 훈련하는 것은 사용할 수 없다면 무의미합니다. 특히 마이크로서비스 비즈니스 모델의 인기가 높아짐에 따라 배포 시간을 줄이는 것이 무엇보다 중요합니다. 효율적인 배포는 기계 학습을 중심으로 하는 많은 비즈니스를 성패할 수 있는 잠재력을 가지고 있습니다.
3. **생태계:** 딥 러닝은 더 이상 고도로 통제된 환경의 특정 사용 사례에 국한되지 않습니다. AI는 수많은 산업에 새로운 힘을 불어넣고 있으므로 모바일, 로컬 및 서버 애플리케이션 개발을 용이하게 하는 더 큰 생태계 내에 있는 프레임워크가 중요합니다. 또한 Google의 **Edge TPU**와 같은 특수 기계 학습 하드웨어의 등장으로 성공적인 실무자는 이 하드웨어와 잘 통합될 수 있는 프레임워크로 작업해야 합니다. .

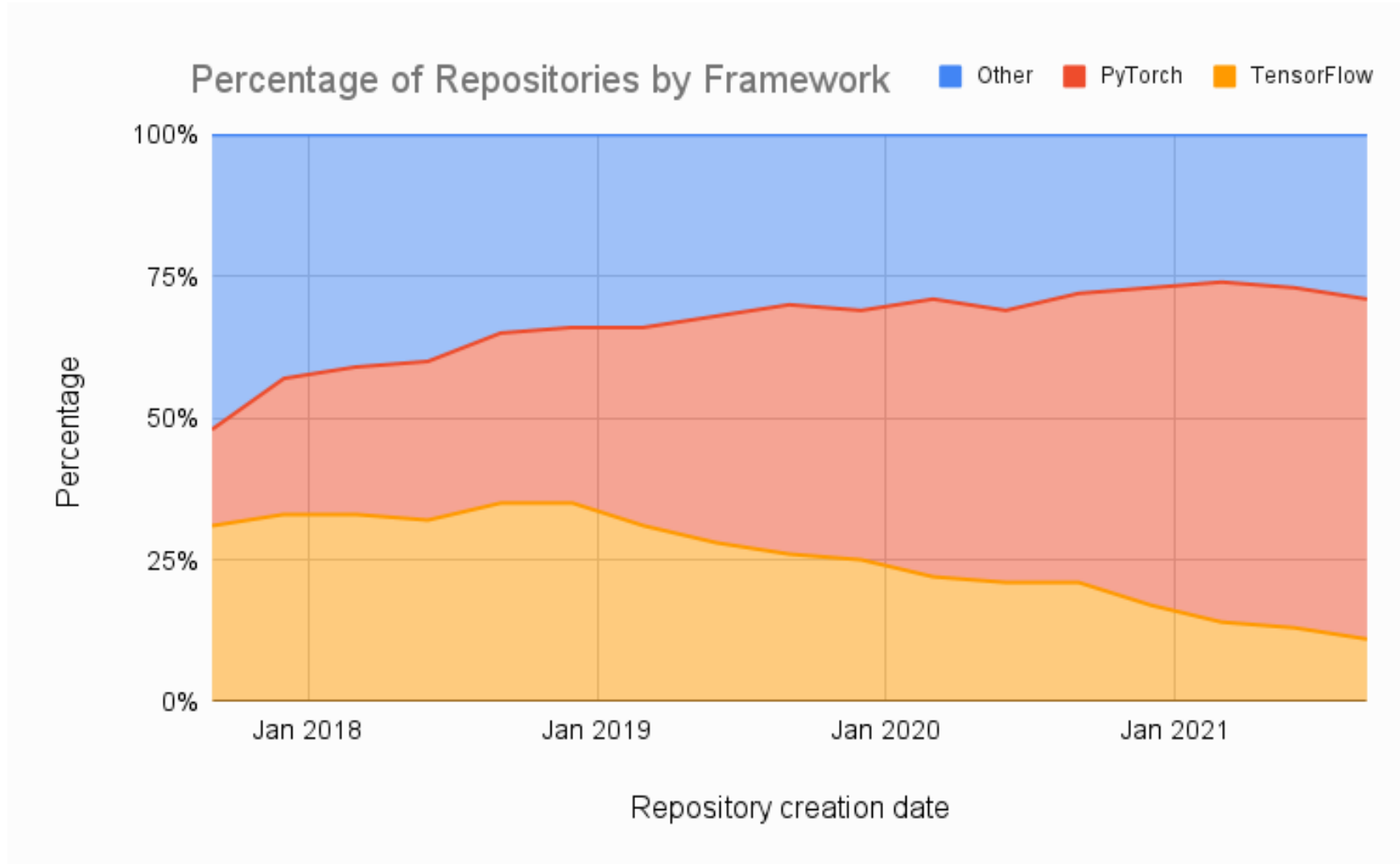


출처 : <https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022>

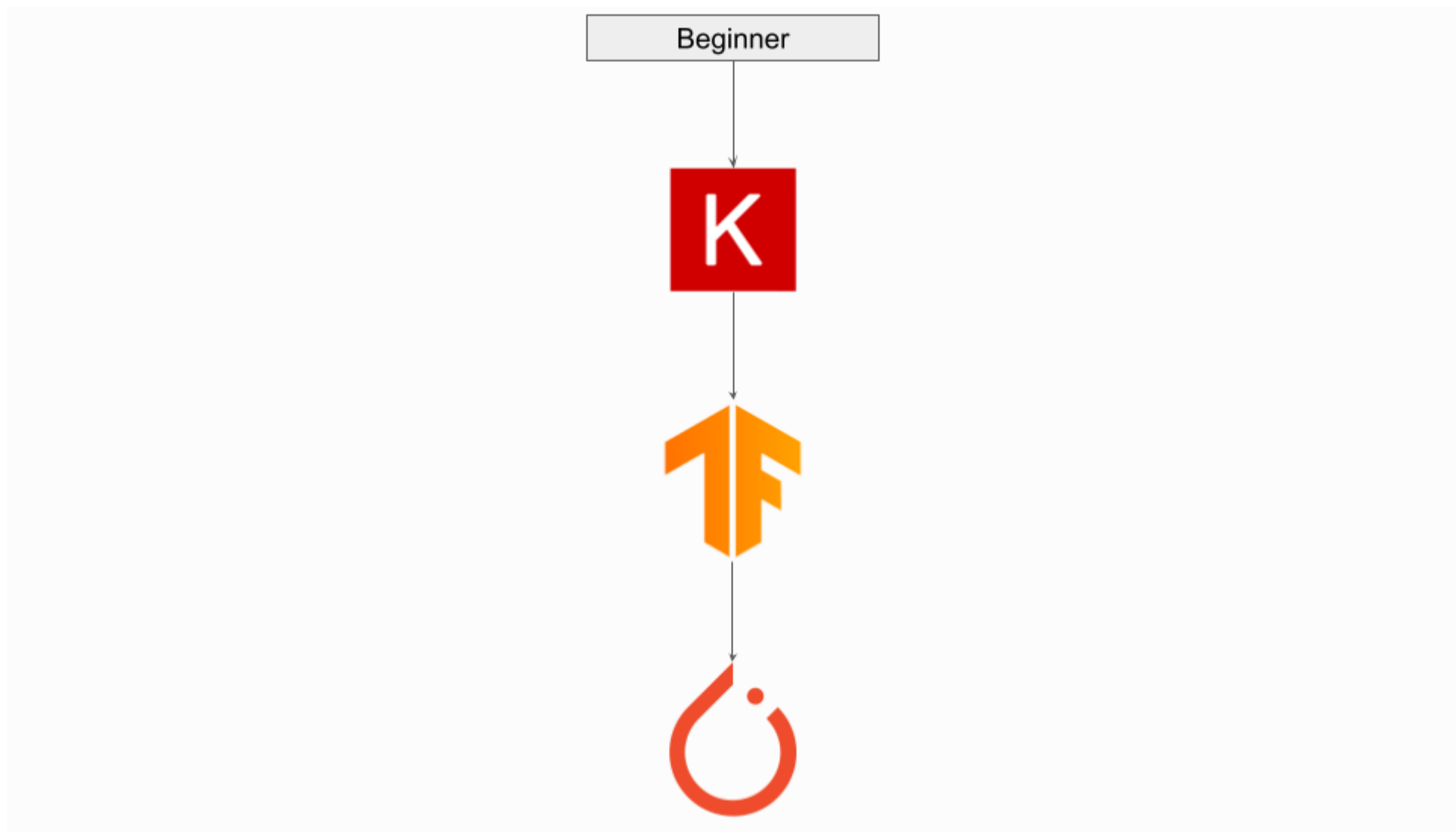




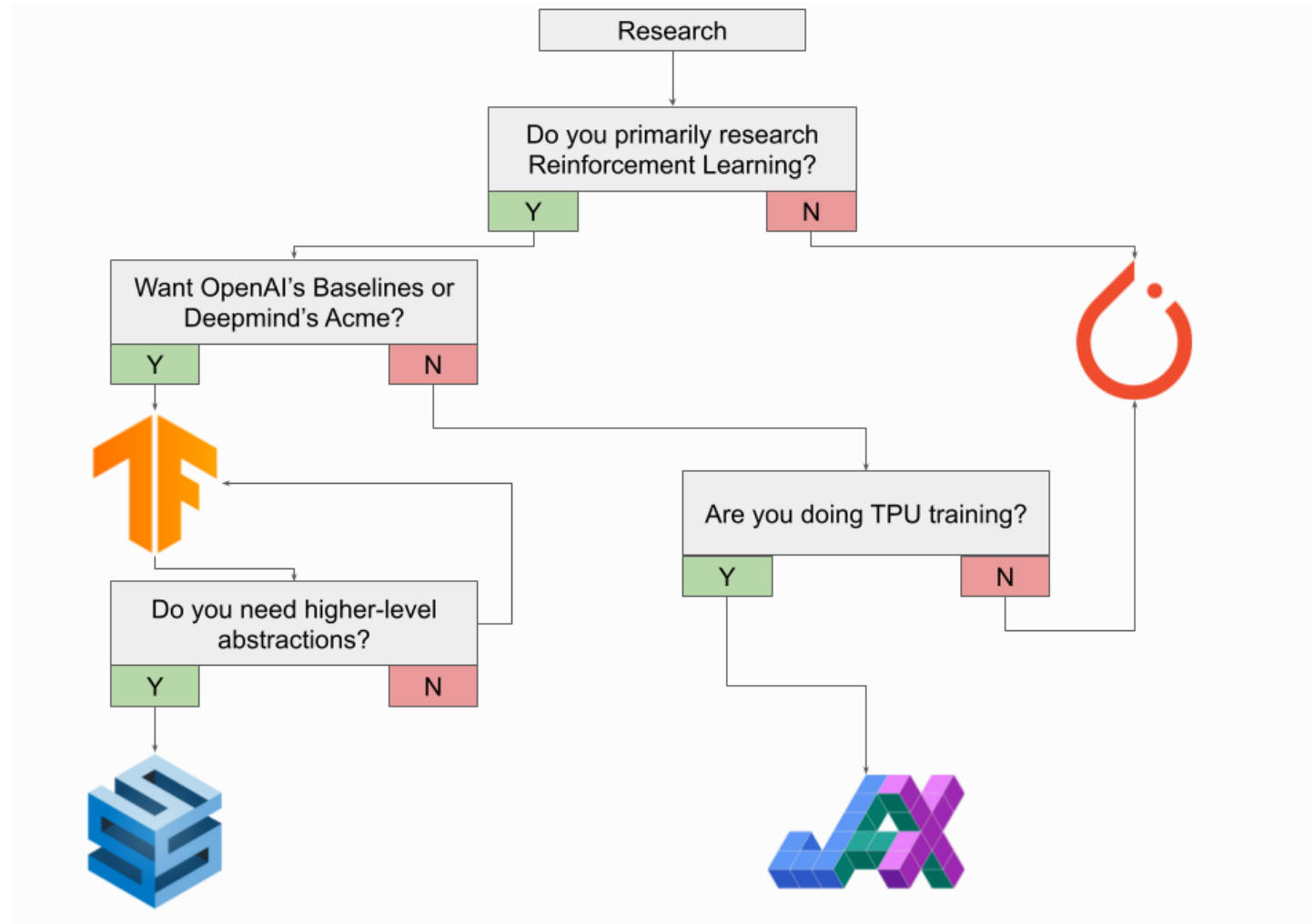
출처 : <https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022>

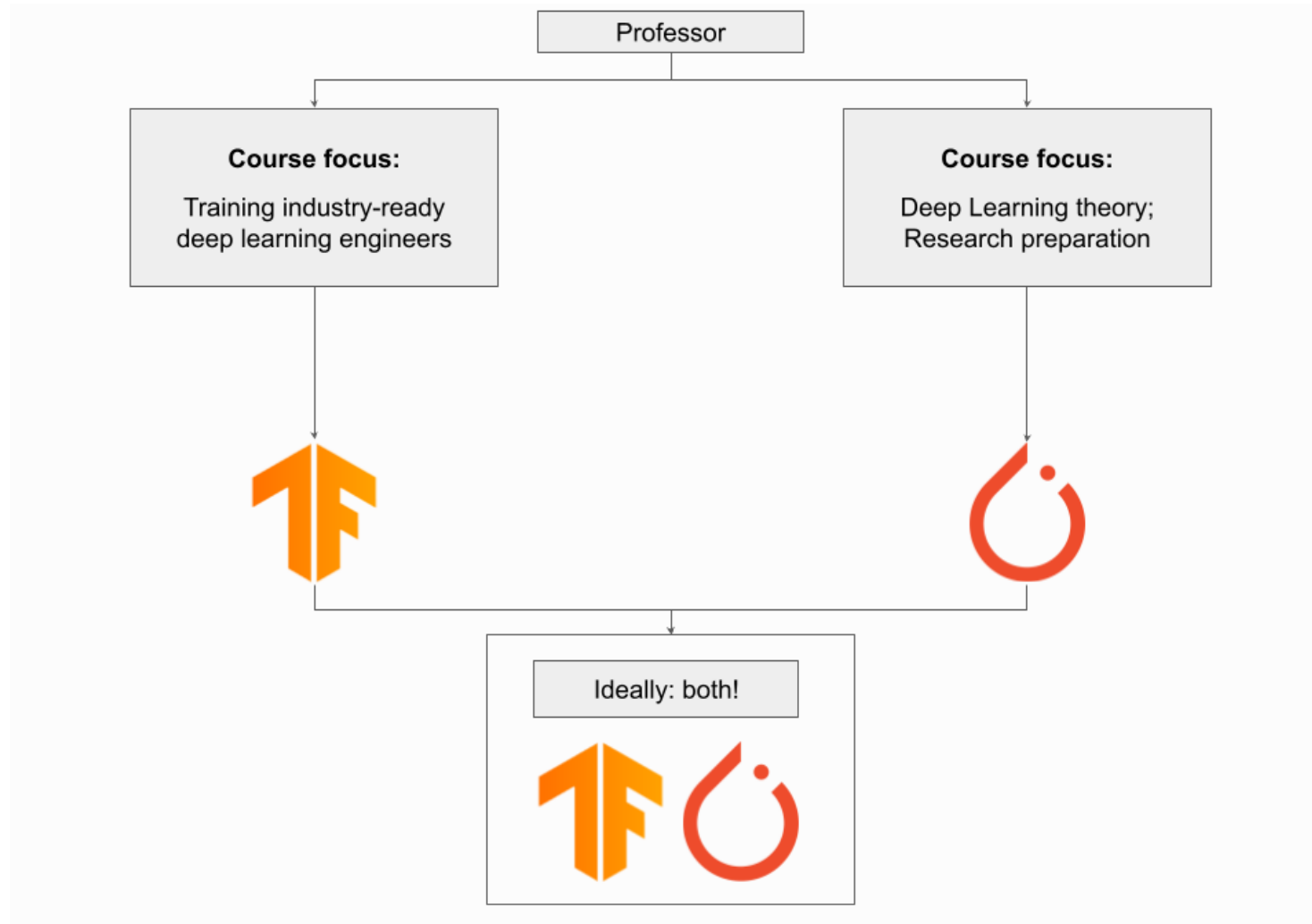


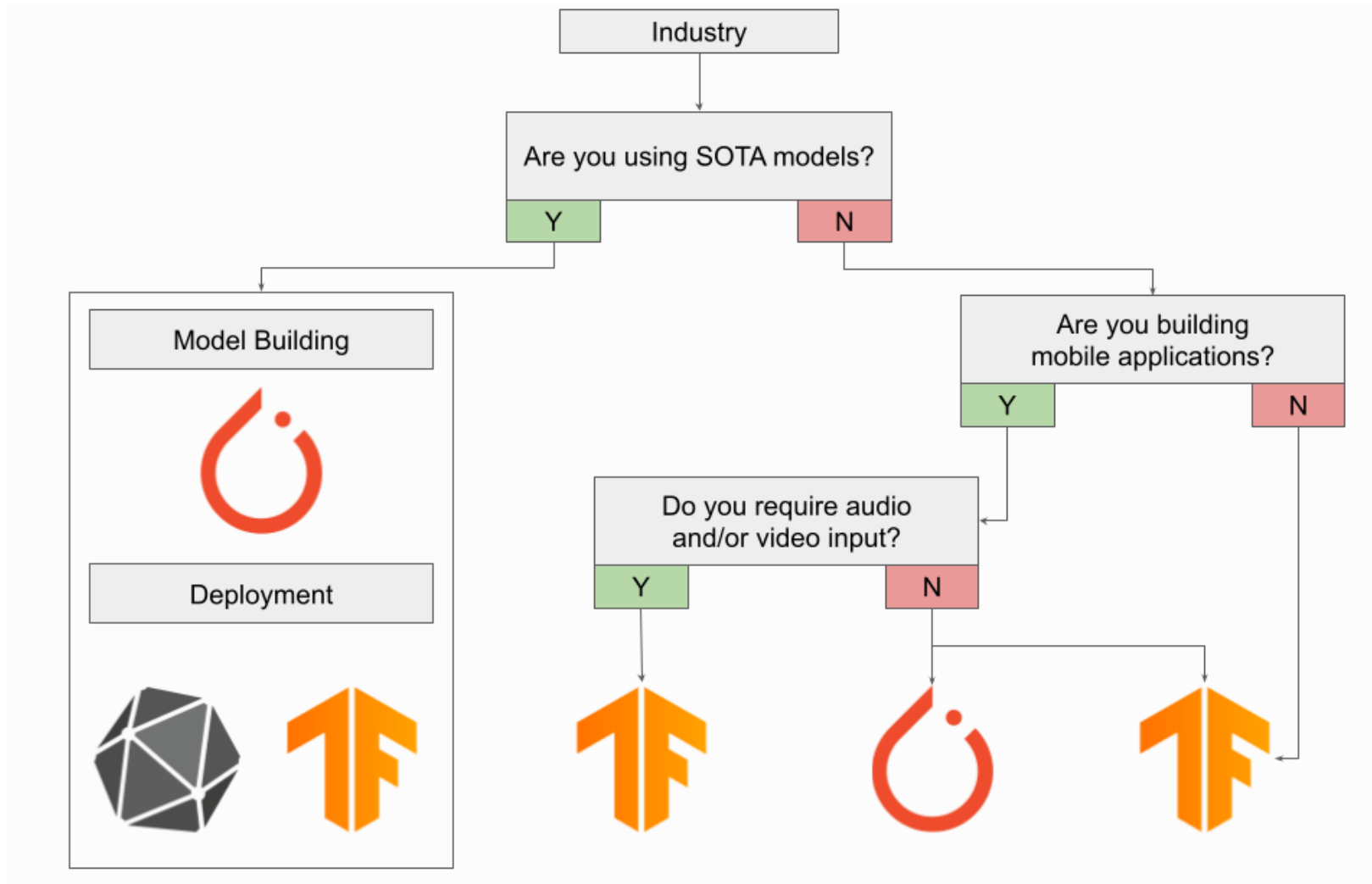
출처 : <https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022>



출처 : <https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022>

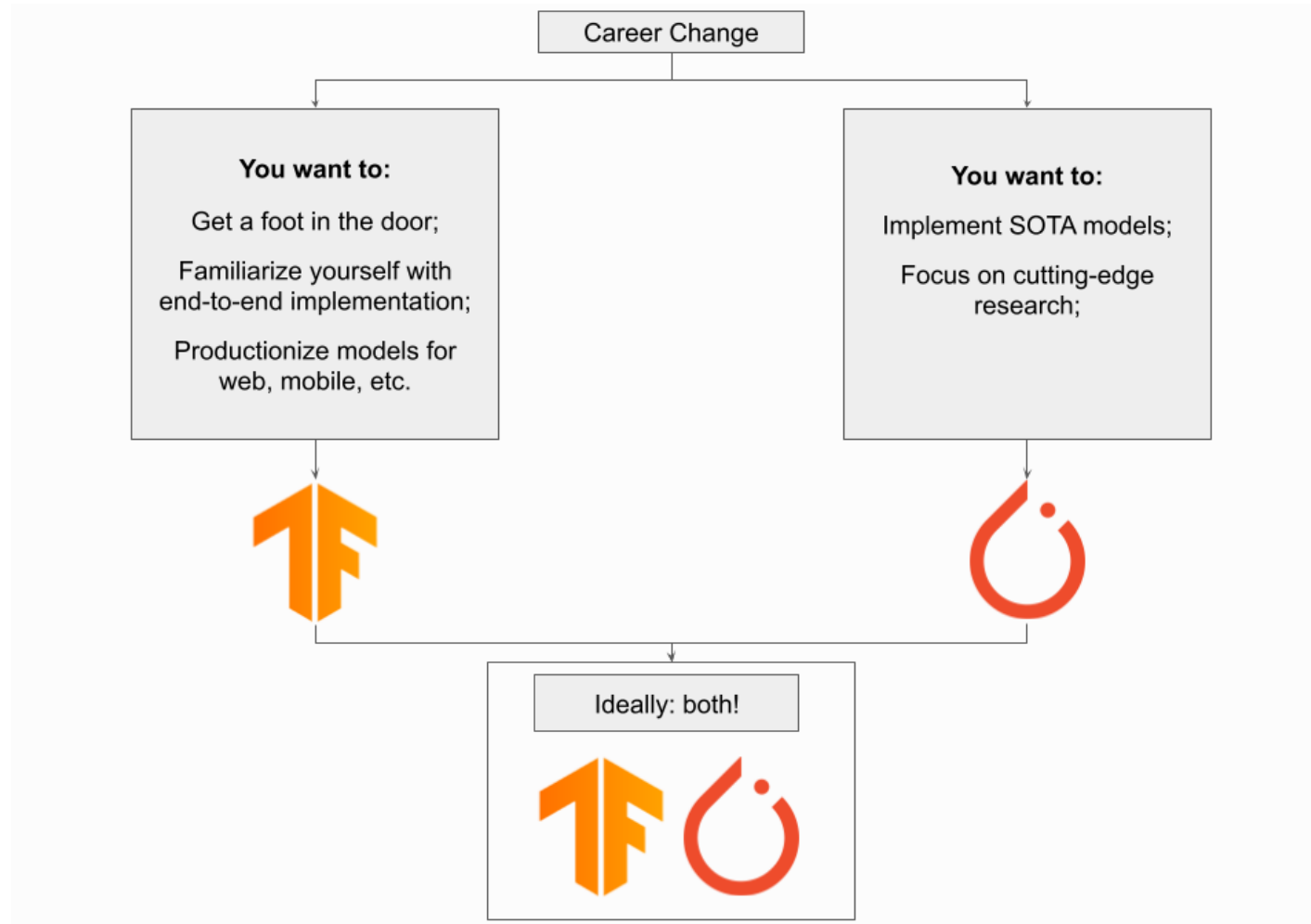






출처 : <https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022>

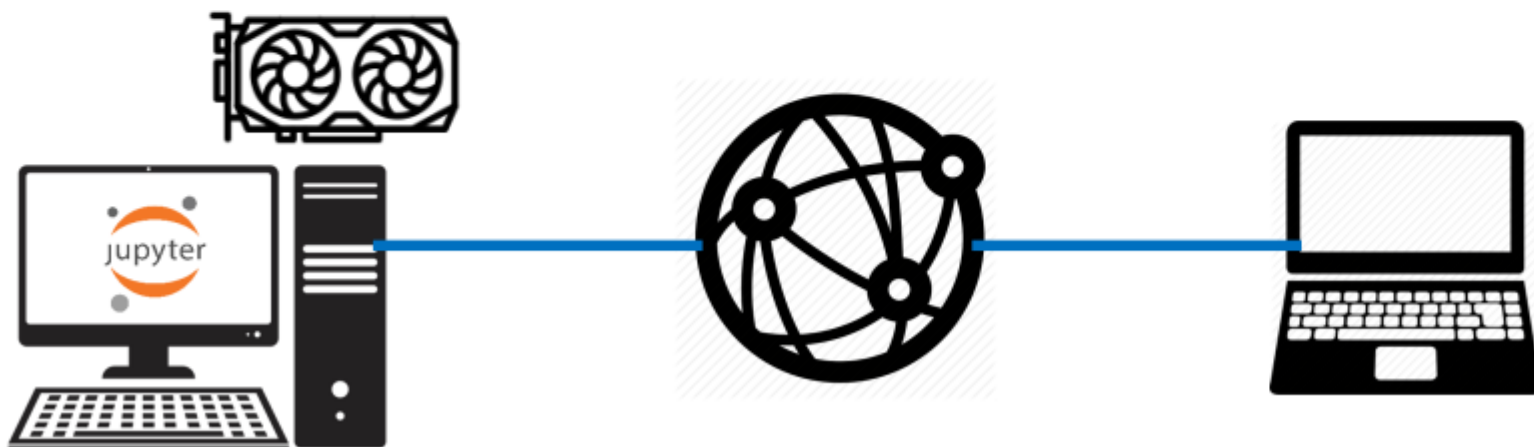
## 경력 변경을 원하고 있다면?



출처 : <https://velog.io/@freejack/PyTorch-vs-TensorFlow-in-2022>

# Jupyter Notebook?

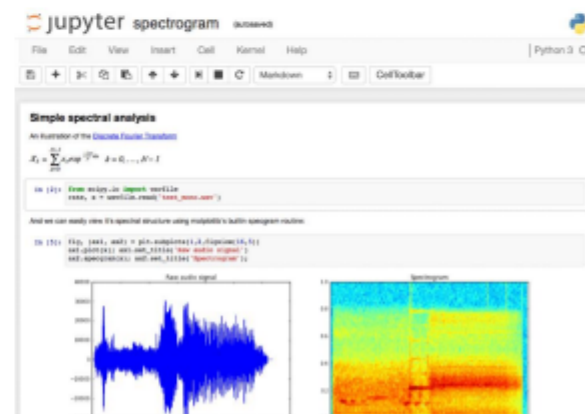
- 웹 브라우저에서 파이썬 코드를 작성하고 실행해 볼 수 있는 개발 도구
  - 원격 코딩 가능
  - 코드 블록 단위로 실행 / 디버깅
  - Text block을 이용한 문서화
  - Figure plotting 등 GUI



147.46.123.123

Pytorch를 활용한 딥러닝 학습 환경 구축 및 실습

147.46.123.123:8888

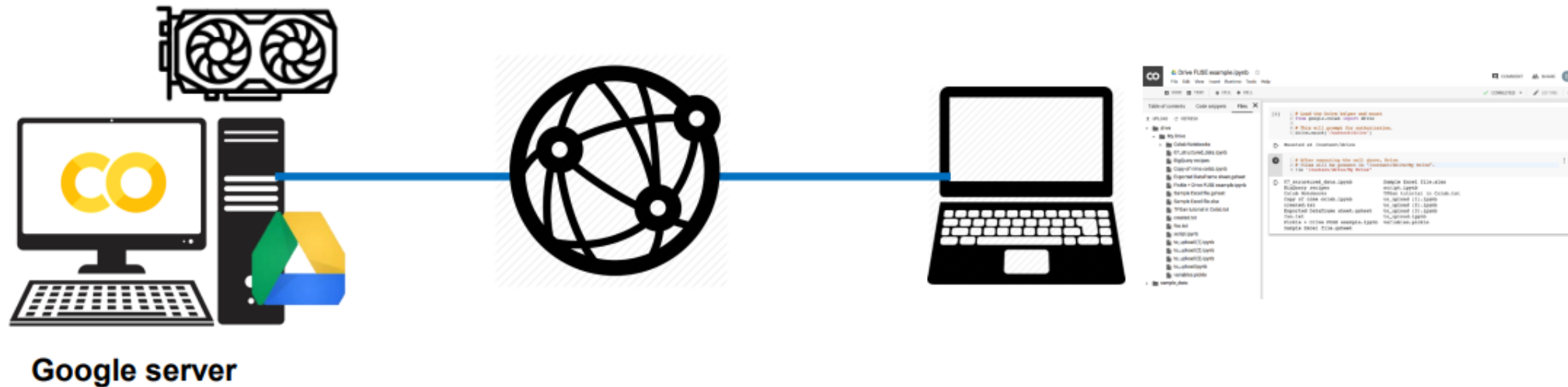


출처 : <https://bi.snu.ac.kr/Courses/ML2019/ML2019.html>



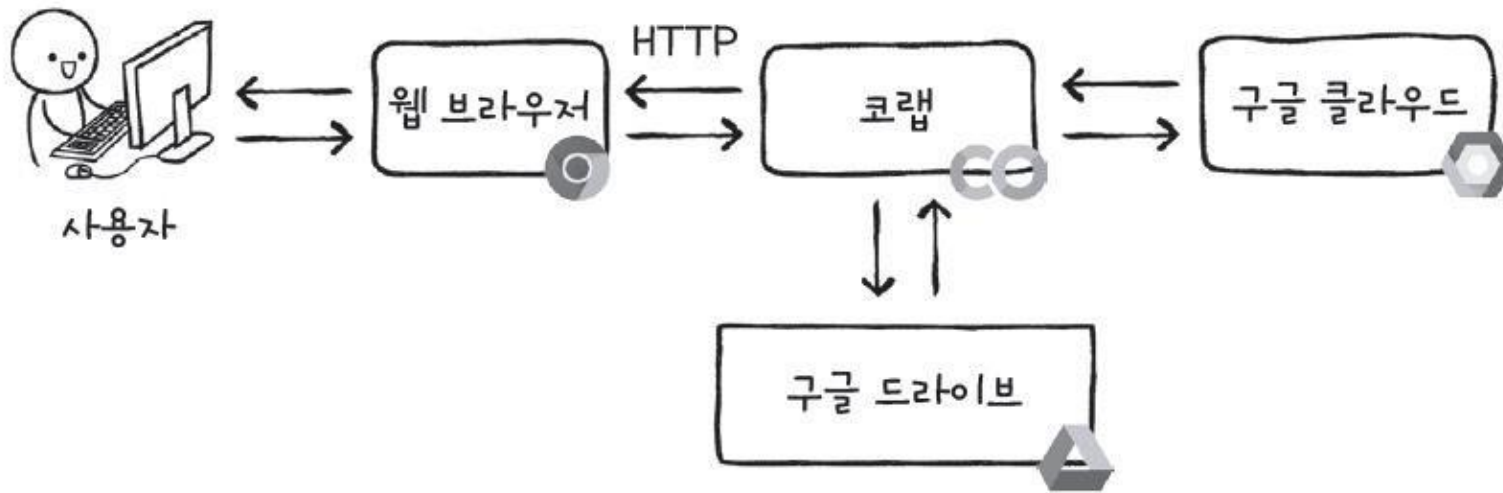
# Google Colab?

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



출처 : <https://bi.snu.ac.kr/Courses/ML2019/ML2019.html>

# colab



출처 : [https://colab.research.google.com/?utm\\_source=scs-index#scrollTo=Nma\\_JWh-W-IF](https://colab.research.google.com/?utm_source=scs-index#scrollTo=Nma_JWh-W-IF)

## 04 개발 환경 설정

### Colab에서 사용할 수 있는 GPU 유형은 무엇인가요?

NVIDIA GeForce RTX 3060

Colab에서 사용할 수 있는 GPU 유형은 시간에 따라 달라집니다. 이러한 방식은 Colab에서 리소스를 무료로 제공하는 데 필요합니다. Colab에서 사용할 수 있는 GPU로는 보통 Nvidia K80, T4, P100이 있습니다. Colab에서 가장 빠른 GPU에 더욱 안정적으로 액세스하는 데 관심이 있다면 [Colab Pro](#) 및 [Pro+](#)가 적합할 수 있습니다. Colab에서 특정 하드웨어를 사용하고 싶다면 [Colab GCP Marketplace VM](#)을 확인하세요.

Colab을 암호화폐 채굴에 사용하는 것은 전면 금지되어 있으며 사용할 경우 Colab을 사용할 수 없도록 계정이 완전히 제한될 수 있습니다.

### Colab에서 노트북을 얼마나 오래 실행할 수 있나요?

노트북은 최대 수명이 12시간인 가상 머신에 연결되어 실행됩니다. 유휴 상태가 너무 오래 지속되면 노트북의 VM 연결이 해제됩니다. 최대 VM 수명 및 유휴 시간 제한 동작은 시간 또는 사용량에 따라 달라질 수 있습니다. 이러한 방식은 Colab에서 컴퓨팅 리소스를 무료로 제공하는 데 필요합니다. 시간에 따라 크게 달라지지 않는 더 긴 VM 수명이나 더 관대한 유휴 시간 제한 동작에 관심이 있다면 [Colab Pro](#) 및 [Pro+](#)가 적합할 수 있습니다.

Colab VM의 전체 기간을 관리하고 싶다면 원하는 대로 관리할 수 있는 지속적인 환경을 제공하는 [Colab GCP Marketplace VM](#)을 확인하세요.

### Colab을 최대한 활용하려면 어떻게 해야 하나요?

한정된 리소스를 소수의 사용자가 독점하지 않도록 하기 위해 Colab의 리소스는 최근에 리소스를 상대적으로 적게 사용한 사용자에게 우선으로 할당됩니다. Colab을 최대한 활용하려면 작업이 끝난 후 Colab 탭을 닫고, 작업에 필요하지 않은 GPU를 선택하지 않는 것이 좋습니다. 이렇게 하면 Colab 사용 중에 사용량 한도에 걸릴 가능성이 낮아집니다. Colab 무료 버전의 리소스 한도 이상을 사용하는 데 관심이 있다면 [Colab Pro](#) 및 [Pro+](#)가 적합할 수 있습니다.

### Time out 조건

- 90분 이상 아무 인터랙션이 없는 경우
- 1일 12시간 이상 세션이 동작한 경우

출처 : <https://research.google.com/colaboratory/faq.html>

# 04 개발 환경 설정

## <1> NVIDIA GeForce RTX 3060

```
Epoch 27/100
313/313 [=====] - 4s 13ms/step - loss: 0.4121 - accuracy: 0.8161 - val_loss: 0.4364 - val_accuracy: 0.8018
0:02:13
```

Epochs = 27

실행시간 = 2m 13s (133s)

1 Epoch 당 실행 시간 = 4.9259s

## <2> 코랩 GPU (NVIDIA Tesla K80 GPU)

```
Epoch 36/100
313/313 [=====] - 6s 20ms/step - loss: 0.4088 - accuracy: 0.8166 - val_loss: 0.4322 - val_accuracy: 0.7984
0:03:54
```

Epochs = 36

실행시간 = 3m 54s (234s)

1 Epoch 당 실행 시간 = 6.5s

## <1> NVIDIA GeForce RTX 3060

```
Epoch 39/100
313/313 [=====] - 2s 8ms/step - loss: 0.4111 - accuracy: 0.8156 - val_loss: 0.4384 - val_accuracy: 0.7950
Epoch 40/100
313/313 [=====] - 3s 8ms/step - loss: 0.4106 - accuracy: 0.8169 - val_loss: 0.4385 - val_accuracy: 0.7946
Epoch 41/100
313/313 [=====] - 3s 11ms/step - loss: 0.4104 - accuracy: 0.8167 - val_loss: 0.4444 - val_accuracy: 0.7934
Epoch 42/100
313/313 [=====] - 3s 10ms/step - loss: 0.4102 - accuracy: 0.8156 - val_loss: 0.4412 - val_accuracy: 0.7942
0:02:32
```

Epochs = 42

실행시간 = 2m 32s (152s)

1 Epoch 당 실행 시간 = 3.619s

## <2> 코랩 GPU (NVIDIA Tesla K80 GPU)

```
Epoch 38/100
313/313 [=====] - 6s 20ms/step - loss: 0.4188 - accuracy: 0.8153 - val_loss: 0.4484 - val_accuracy: 0.7930
Epoch 39/100
313/313 [=====] - 6s 20ms/step - loss: 0.4185 - accuracy: 0.8148 - val_loss: 0.4464 - val_accuracy: 0.7912
Epoch 40/100
313/313 [=====] - 6s 20ms/step - loss: 0.4179 - accuracy: 0.8162 - val_loss: 0.4504 - val_accuracy: 0.7860
Epoch 41/100
313/313 [=====] - 6s 20ms/step - loss: 0.4168 - accuracy: 0.8157 - val_loss: 0.4474 - val_accuracy: 0.7936
Epoch 42/100
313/313 [=====] - 6s 20ms/step - loss: 0.4168 - accuracy: 0.8159 - val_loss: 0.4484 - val_accuracy: 0.7866
0:04:30
```

Epochs = 42

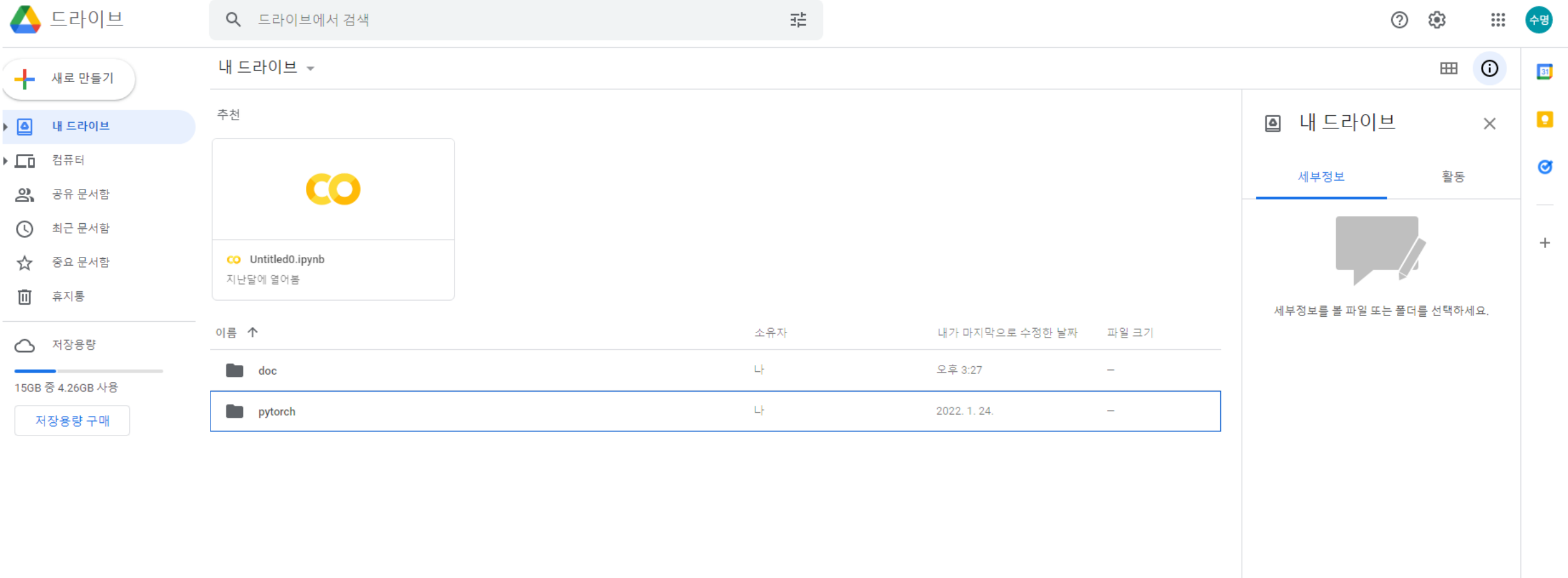
실행시간 = 4m 30s (270s)

1 Epoch 당 실행 시간 = 6.4286s

## Google Colab - 사용법

- 개인 구글 계정 필요
- Colab과 Jupyter Notebook 사용 방법은 유사한 부분이 많음
  - 실습 수업에서는 Colab 위주로 설명
  - GPU가 내장된 서버를 사용할 수 있을 시 로컬에서 작업을 권장

## 04개발 환경 설정



# 04 개발 환경 설정

+

새로 만들기

▶

내 드라이브

▶

컴퓨터

▶

공유 문서함

▶

최근 문서함

▶

중요 문서함

▶

휴지통

클라우드 저장용량

15GB 중 4.26GB 사용

저장용량 구매

🔍

드라이브에서 검색

🔍

내 드라이브 > pytorch ▾

이름 ↑	소유자	내가 마지막으로
🔗 Untitled0.ipynb	나	오후 2:52

📁

새 폴더

📄

파일 업로드

📁

폴더 업로드

📄

Google 문서

>

📄

Google 스프레드시트

>

📄

Google 프레젠테이션

>

📄

Google 설문지

>

⋮

더보기

>

📄

Google 드로잉

📍

Google 내 지도

📄

Google 사이트 도구

📄

Google Apps Script

🔗

Google Colaboratory

📄

Google Jamboard

+


연결할 앱 더보기

## 04 개발 환경 설정

Google Workspace Marketplace

colab

검색결과: colab



✓ 설치됨

Colaboratory

colab-team

Colab is a Jupyter notebook environment that runs in the browser using Google Cloud.

★ 4.7 • 10,000,000+



## 04 개발 환경 설정

Google Workspace Marketplace

?
⚙️
🔗
✕

# Colaboratory

Colab is a Jupyter notebook environment that runs in the browser using Google Cloud.

개발자: [colab-team](#)

제거

호환 기기:

★★★★★ 3,470

↓ 10,000,000+

개요

사용 권한

리뷰

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

+ Code + Text

Copy to Drive

Connect

Editing

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

Pytorch를 활용한 딥러닝 학습 환경 구축 및 실습

- 33 -

# 04개발 환경 설정

구분	무료	유료	의견
GPU	K80, T4	T4, P100 등 무료보다는 좋은 사양에 할당 TPU 우선 할당	상당히 애매하게 작성해놓음 무료보다는 조금 더 좋은 사양에 할당이 된다 정도인데 사용시간에 따라 다르다라고 언급 무료도 보통 T4로 할당 됨.
유지 시간	12시간	24시간 (단, 완전보장 못함)	24시간이라고 쓰여있지만, 끊길 수도 있고, 24시간 보다 줄어들수도 있다고 언급
RAM	12.72 GB	고용량 : 25.51 GB 표준 : 12.72 GB	런타임 유형을 고용량 RAM 으로 직접 변경해야 커짐
CPU	Intel(R) Xeon(R) CPU @ 2.20GHz / 2.30GHz	Intel(R) Xeon(R) CPU @ 2.30GHz	무료도 2.30 할당 되기도 함 사양 거의 동일

출처 : <https://limitsinx.tistory.com/135?category=905034>

# 04 개발 환경 설정

드라이브

+

새로 만들기

내 드라이브

Colab Notebooks

doc

pytorch

chapter0

chapter1

chapter2

chapter3

chapter4

chapter5

chapter6

github

컴퓨터

🔍

드라이브에서 검색

🔑

?

⚙️

☰

내 드라이브 > pytorch

📁 ⓘ

이름 ↑	소유자	내가 마지막으로 수정한 날짜	파일 크기
chapter0	나	2022. 2. 2.	—
chapter1	나	2022. 2. 1.	—
chapter2	나	2022. 2. 2.	—
chapter3	나	2022. 2. 2.	—
chapter4	나	2022. 2. 2.	—
chapter5	나	2022. 2. 2.	—
chapter6	나	2022. 2. 3.	—
github	나	2022. 2. 2.	—

## 폴더 구조 체크

## 04 개발 환경 설정



배포용 자료 (일 별로 자료 업로드 예정)

<https://url.kr/grs716>

내 드라이브 > KMU\_Pytorch\_특강 ▾ 👤



이름 ↑

소유자

내가 마지막으로 수정한 날짜

파일 크기



1일차 참고자료

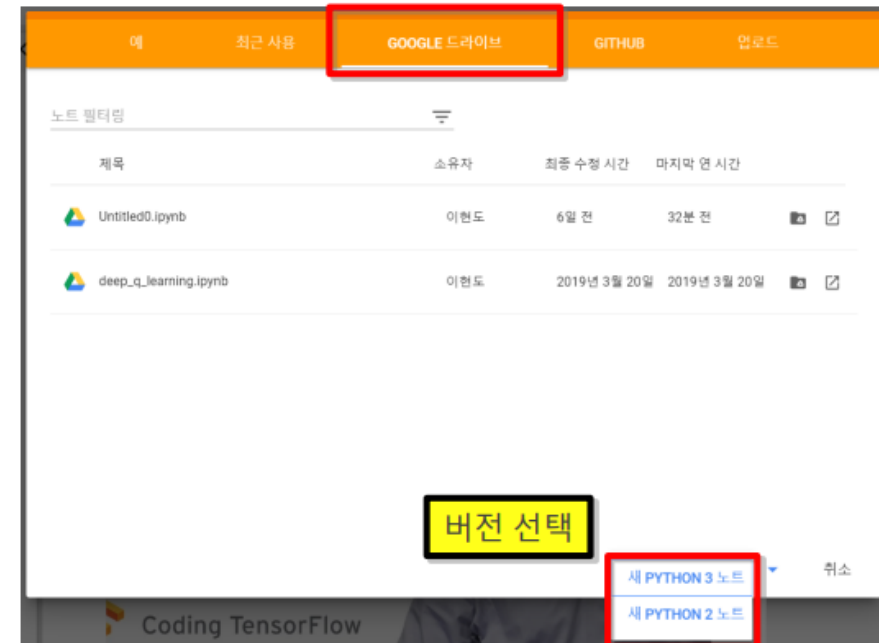
나

오전 12:08

—

## 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



출처 : <https://bi.snu.ac.kr/Courses/ML2019/ML2019.html>

# 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

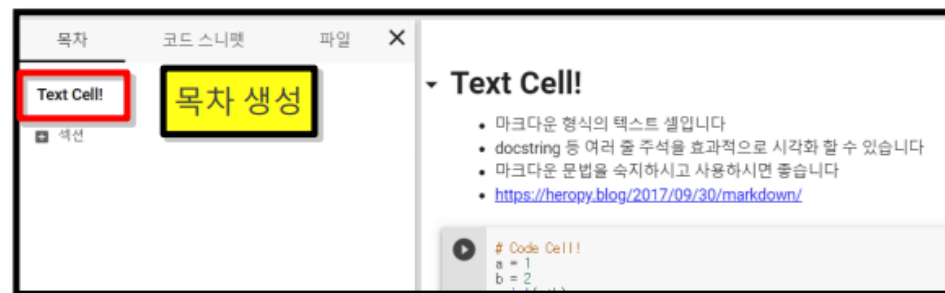
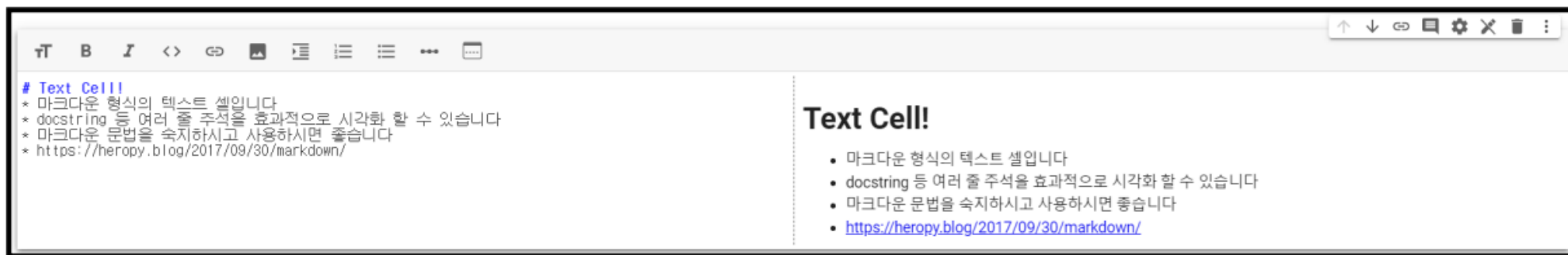
- 상단 메뉴의 런타임
  - 실행 중인 셀 중단
  - 런타임 다시 시작

The image shows the Google Colab interface with two screenshots. The top screenshot shows a code cell with the following code: `# Code Cell!  
a = 1  
b = 2  
print(a+b)  
  
# Ctrl+Enter 로 해당 코드 셀 실행`. The execution number [1] is highlighted. Below the code, the output is 3. A yellow box labeled '실행 번호' points to the execution number. A yellow box labeled '실행 (Ctrl + Enter)' points to the play button icon. A yellow box labeled '실행 결과' points to the output 3. The bottom screenshot shows the '런타임' (Runtime) menu with options: 모두 실행 (Ctrl+F9), 이전 셀 실행 (Ctrl+F8), 초점이 맞춰진 셀 실행 (Ctrl+Enter), 선택항목 실행 (Ctrl+Shift+Enter), 이후 셀 실행 (Ctrl+F10), 실행 중단 (Ctrl+M), 런타임 다시 시작... (Ctrl+M), 다시 시작 및 모두 실행..., and 모든 런타임 재설정... The '실행 중단' and '런타임 다시 시작...' options are highlighted with a red box. A yellow box labeled '실행 중단 런타임 재시작' points to these options.

# Google Colab - 사용법

## ■ Text cell

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





# Google Colab - 사용법

## ■ 단축키

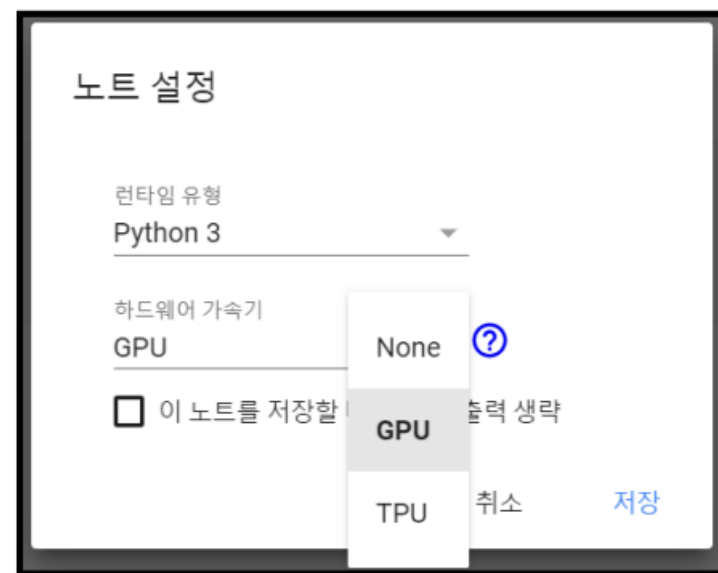
- 대부분의 작업은 단축키로 실행 가능
- 단축키 설정 가능
- 단축키 설정화면 – Ctrl+M H
- 유용한 단축키
  - 코드 셀 생성 – Ctrl+M A(B)
  - 코드 셀 실행 – Ctrl+Enter
  - 셀 삭제 – Ctrl+M D
  - 실행중인 셀 중단 – Ctrl+M I
  - 런타임 다시 시작 – Ctrl+M .
  - 코드(텍스트) 셀로 변환 – Ctrl+M Y(M)
  - 마지막 셀 작업 실행취소 – Ctrl+Shift+Z



## 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 Usage
GPU	PID	Type	Process name		
No running processes found					

## Google Colab - 사용법

### ■ 구글 드라이브 연동

- 간단한 인증 절차 이후 구글 드라이브의 파일을 Colab에서 접근 가능

The diagram illustrates the process of connecting Google Drive to Google Colab. It consists of several components:

- Code Cell:** A code cell in Colab showing the initial setup:
 

```
from google.colab import drive
drive.mount('/content/drive')
```
- Google Login:** A Google login screen with the text "로그인" (Login). It instructs the user to copy the code and paste it into the application. The authorization code is:
 

```
4/rAE5J7Ung6WcINc8TjgZ6VALX2tUPa7y6-  
gbV0q0jp8LkC1IozBsWV1
```
- File Explorer:** A file explorer window showing the directory structure. The 'content' folder is expanded, showing 'drive' and 'My Drive' subfolders. The 'My Drive' folder is highlighted with a red box.
- Terminal Window:** A terminal window showing the execution of the code. It displays the URL to go to in a browser, the authorization code, and the resulting file paths:
 

```
[3] from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth
```

```
Enter your authorization code:
Mounted at /content/drive
```

```
[5] !ls 'drive/My Drive/Colab Notebooks'
```

```
deep_q_learning.ipynb Lab1_1.ipynb Untitled0.ipynb
```

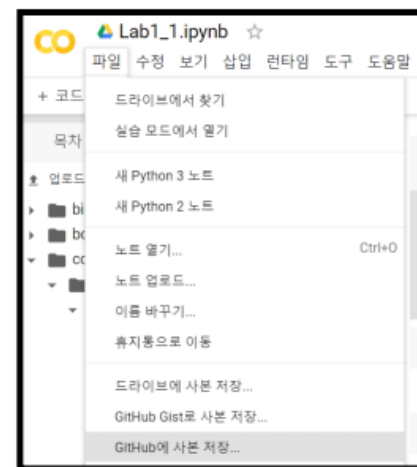
15

출처 : <https://bi.snu.ac.kr/Courses/ML2019/ML2019.html>

## 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



GitHub으로 복사

저장소:  브랜치:

파일 경로  
Lab1\_1.ipynb

변경사항 설명 메시지  
Colaboratory를 통해 생성됨

☒ Colaboratory 링크 추가

취소 확인

# 감사합니다

