

17기 정규세션

ToBig's 16기 강의자

김송민

Git & Framework

Computer Science

Unit 01 | Git 기초 - 로컬 저장소

Unit 02 | Git 협업 - 원격 저장소

Unit 03 | Framework

Unit 03-1 | Tensorflow

Unit 03-2 | Keras

Unit 03-3 | Pytorch

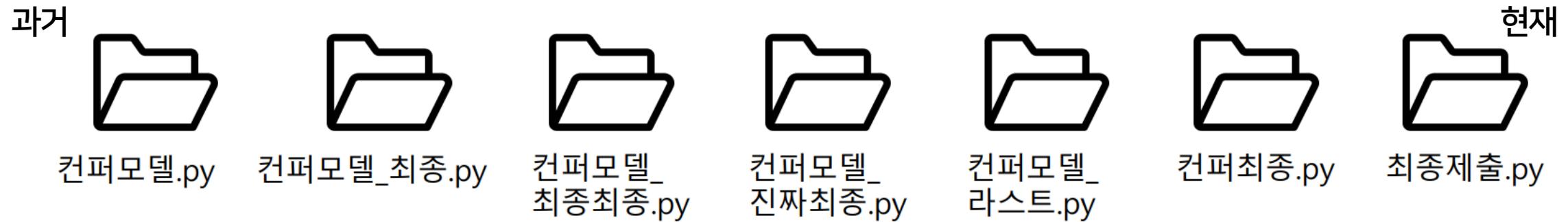
Unit 01 | Git 기초 - 로컬 저장소



깃(Git /git/)은 컴퓨터 파일의 변경사항을 추적하고
여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템이다

Unit 01 | Git 기초 - 로컬 저장소

- 버전 관리 시스템 (VCS)



수많은 버전의 파일 중…

Unit 01 | Git 기초 - 로컬 저장소

- 버전 관리 시스템 (VCS)



헷갈리지 않도록 최종 파일만 보여줍니다

Unit 01 | Git 기초 - 로컬 저장소

- 버전 관리 시스템 (VCS)



원하는 버전으로 돌아갈 수도 있어요!

Unit 01 | Git 기초 - 로컬 저장소

GOALS

- 1) 정규세션 과제 제출
- 2) 컨퍼런스 협업

Unit 01 | Git 기초 - 로컬 저장소

Git을 GUI로 사용할 수 있는
다양한 응용프로그램들이 있지만

- GitHub Desktop
- SourceTree
- GitKraken
- Tortoise Git

.

.

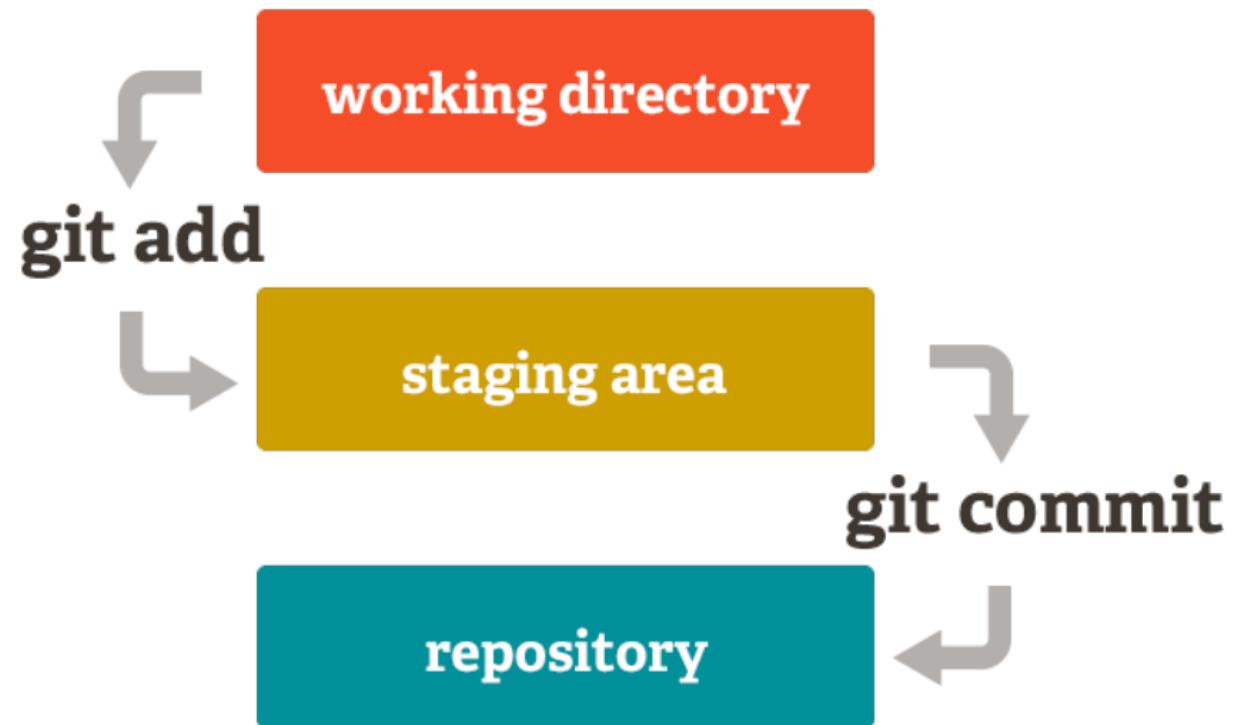
.

GUI가 더 단순하기 때문에
CLI를 사용할 줄 알면
GUI도 사용 가능

-> 각 환경에 맞게
명령 프롬프트(cmd) 창 사용하기!

깃 설치에 대한 내용은 강의에 포함되어 있지 않으니,
설치 돼 있지 않다면 설치 & github 계정도 만들기!

Unit 01 | Git 기초 - 로컬 저장소



로컬 저장소(local git repository) 만들고 파일 올리며 관리하기

Unit 01 | Git 기초 - 로컬 저장소

- **로컬 저장소와 원격 저장소**

- **로컬 저장소(Local Repository)**:

:내 PC에 파일이 저장되는 개인 전용 저장소입니다.

- **원격 저장소(Remote Repository)**

:파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유하기 위한 저장소입니다.

⇒ 로컬에 파일이 올라가도 다른 사람은 아직 볼 수 없습니다

Unit 01 | Git 기초 - 로컬 저장소

로컬 저장소에 파일 업로드 하는 프로세스



Unit 01 | Git 기초 - 로컬 저장소

• 로컬 저장소 만들기

1) 저장소 새로 만들기

git init

저장소 만들 디렉토리 위치에서 실행

2) 원격 저장소를 로컬로 복사해오기

git clone [원격 저장소 주소]

실행한 디렉토리에 원격 저장소 폴더가 복사됨

```
Anaconda Prompt (Anaconda3)
(base) C:\Users\sso>cd desktop
(base) C:\Users\sso\Desktop>cd 17th_tobigs
(base) C:\Users\sso\Desktop\17th_tobigs>git init
Initialized empty Git repository in C:/Users/sso/Desktop/17th_tobigs/.git/
(base) C:\Users\sso\Desktop\17th_tobigs> cd ..
(base) C:\Users\sso\Desktop>git clone https://github.com/SongMin358/random.git
Cloning into 'random'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
(base) C:\Users\sso\Desktop>
```

Unit 01 | Git 기초 - 로컬 저장소

• 깃 상태 확인 git status

- On branch master : 현재 master 브랜치에 있다는 뜻.
- Noting to commit : 커밋할 파일이 없다.
- working tree clean : 워킹 트리도 깨끗하다. (변경 사항이 없다)

```
(base) C:\Users\ssot\Desktop\random>git status
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean
```

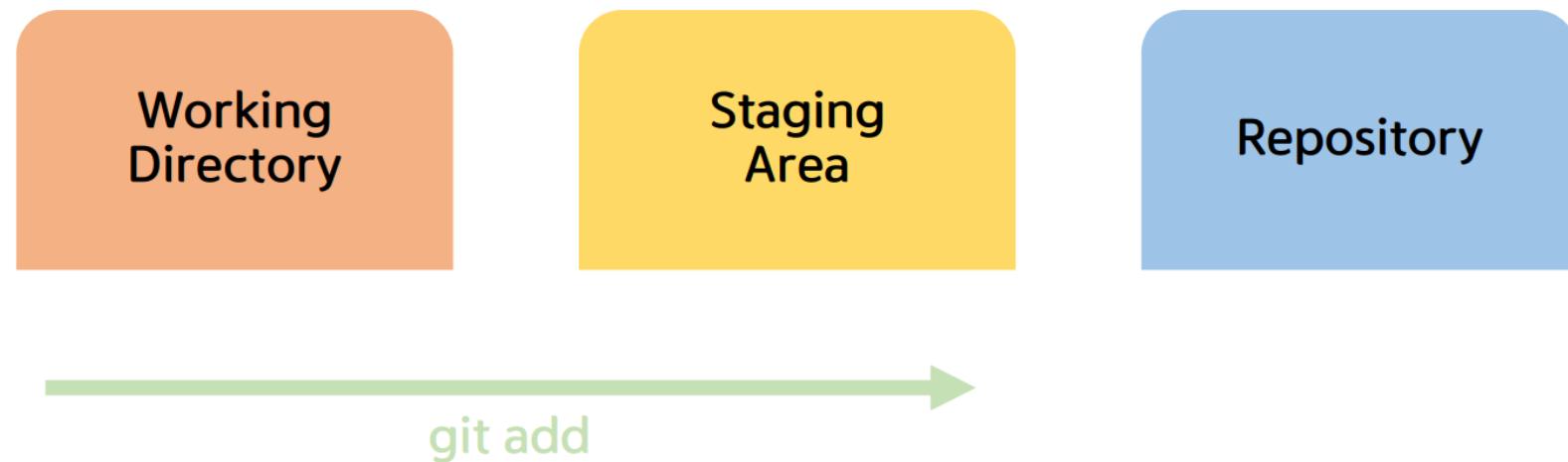
- 변경사항 있을 경우 확인 가능

```
(base) C:\Users\ssot\Desktop\random>git status
On branch master
Your branch is up to date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test2.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Unit 01 | Git 기초 - 로컬 저장소

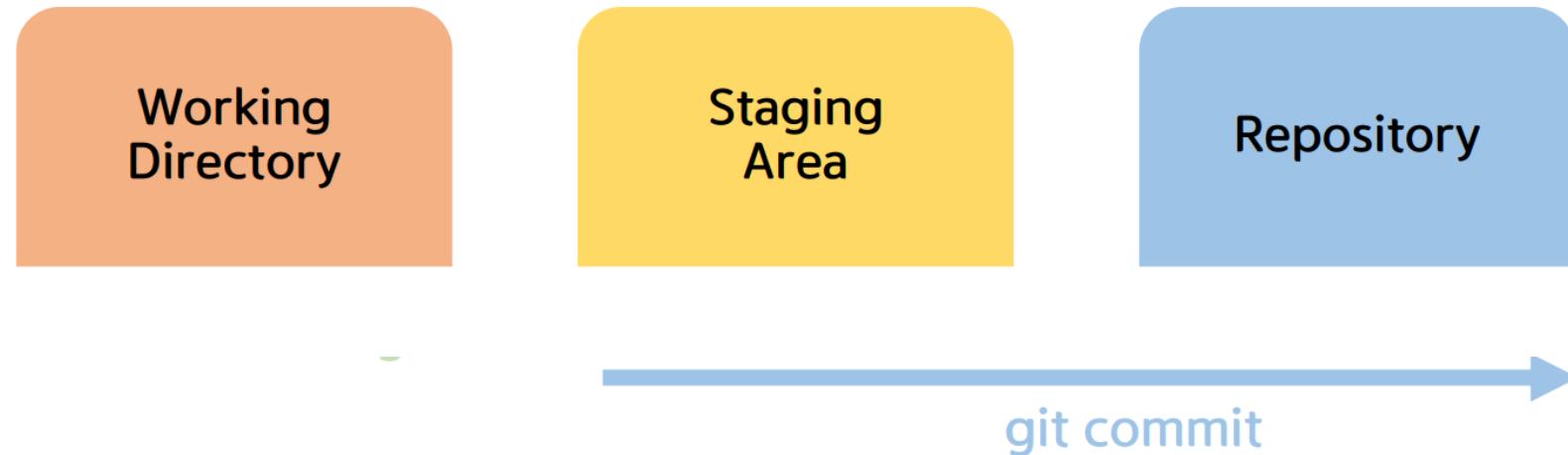
- Working directory → staging area
- 처음 코드 작성 시 working directory에 위치하는 파일들, 로컬에 올리기 전 대기시키기 (staging)



- **git add [파일명]** 실행 시 staging area로 이동함. 원하는 파일들만 인덱스에 등록해 커밋 가능
- **git add .** 사용하면 모든 파일 전체 이동

Unit 01 | Git 기초 - 로컬 저장소

- Staging area → local repository
- 로컬 저장소(local repository)에 올리기, 기록하기



- **git commit -m “[커밋 메시지]”** 실행 시 로컬 저장소로 커밋
- 이전 커밋 상태부터 현재 상태까지의 변경 이력이 시간순으로 기록된 커밋(혹은 리비전)이 만들어집니다.
- 커밋은 이력을 남기는 중요한 작업, 커밋 메시지는 필수!
 - 명료하고 이해하기 쉽게 남겨야 본인 뿐만 아니라 다른 사람이 커밋 이력을 확인하기 쉽습니다.

Unit 02 | Git 협업 - 원격 저장소

- 커밋 메시지 잘 남기는 법

좋은 git 커밋 메시지를 작성하기 위한 7가지 약속

- 제목과 본문을 한 줄 띄워 분리하기
- 제목은 영문 기준 50자 이내로
- 제목 첫글자를 대문자로
- 제목 끝에 . 금지
- 제목은 명령조로
- 본문은 영문 기준 72자마다 줄 바꾸기
- 본문은 어떻게보다 무엇을, 왜에 맞춰 작성하기

Unit 01 | Git 기초 - 로컬 저장소

- 만들어진 커밋들 기록 확인

```
(base) C:\Users\ssor\Desktop\random>git log
commit 188a88d42a73c61954c1691d975700445885163 (HEAD -> master, origin/master, origin/HEAD)
Author: SongMin358 <ssorange38@gmail.com>
Date:   Tue Jan 18 11:02:55 2022 +0900

    test2 commit

commit b1f95c14d8a59c32657834ed2b547deeab83ae24
Author: SongMin358 <ssorange38@gmail.com>
Date:   Thu Jul 22 17:22:32 2021 +0900

    test committing
```

- **git log** 실행 시 만들어진 커밋들 기록 확인 가능 (커밋마다 고유 이름이 붙습니다.)
- **git log --oneline** 보다 간략하게 확인할 수 있습니다. (이름이 보다 간단해집니다)

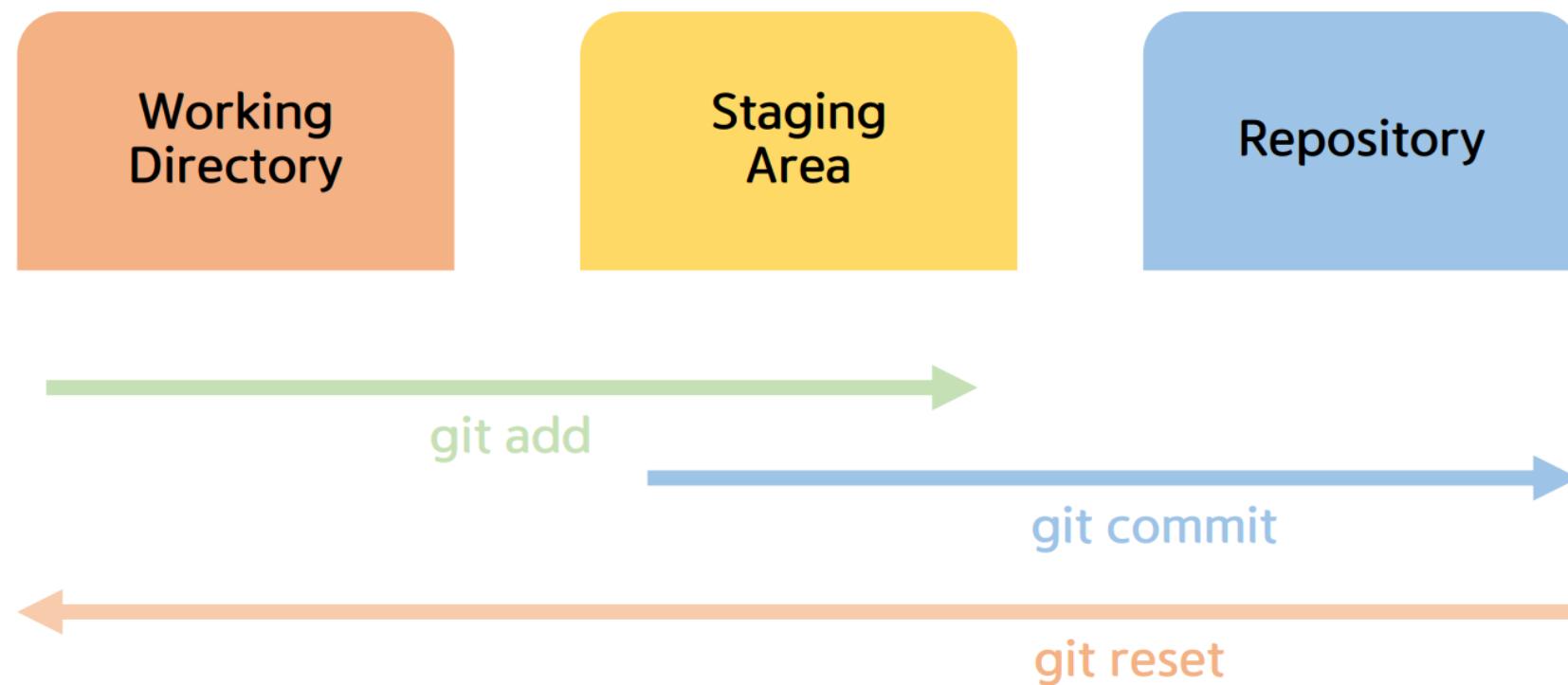
Unit 01 | Git 기초 - 로컬 저장소

- local repository → working directory



- **git reset** 실행 시 과거 버전으로 되돌아감
- **git reset —hard** 보다 강제적으로 되돌리기
- **git reset [고유이름]** or **git reset --hard [고유이름]**
git log에서 확인한 commit 고유 이름으로, 특정 버전을 지정하여 이동할 수 있습니다.

Unit 01 | Git 기초 - 로컬 저장소



Computer Science

Unit 01 | Git 기초 - 로컬 저장소

Unit 02 | Git 협업 - 원격 저장소

Unit 03 | Framework

Unit 03-1 | Keras

Unit 03-2 | Tensorflow

Unit 03-3 | Pytorch

Unit 02 | Git 협업 - 원격 저장소

깃허브를 이용해 원격 저장소를 관리해봅시다!

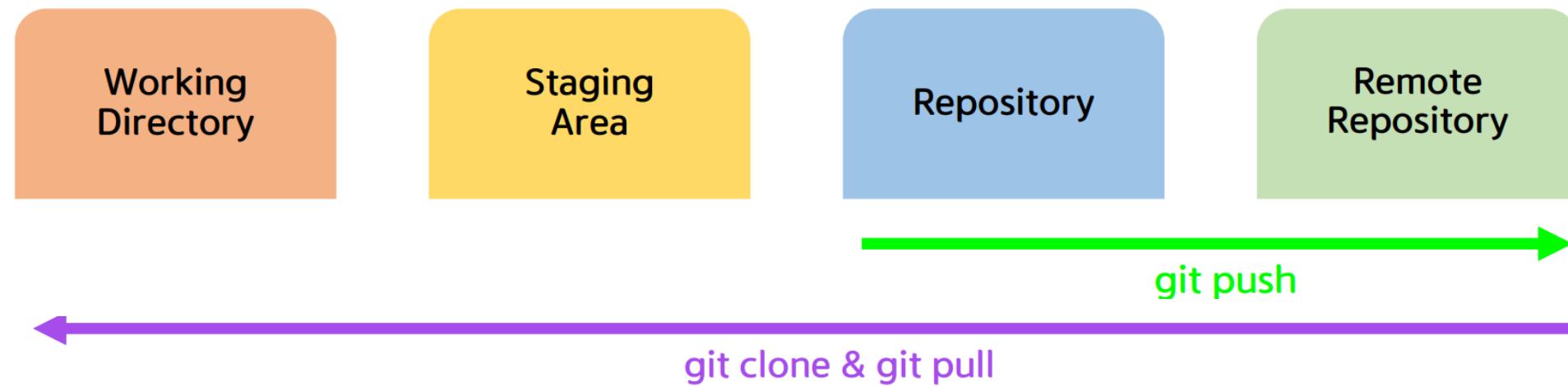


깃허브 (GitHub)는
깃(Git)을 사용하는 프로젝트를 지원하는 웹호스팅 서비스,
다시 말해 Git 저장소 웹사이트이며,
개발자들의 버전 제어 및 공동 작업을 위한 플랫폼이다.

비슷하게 Bitbucket, Gitlab 등 다른 플랫폼도 있습니다

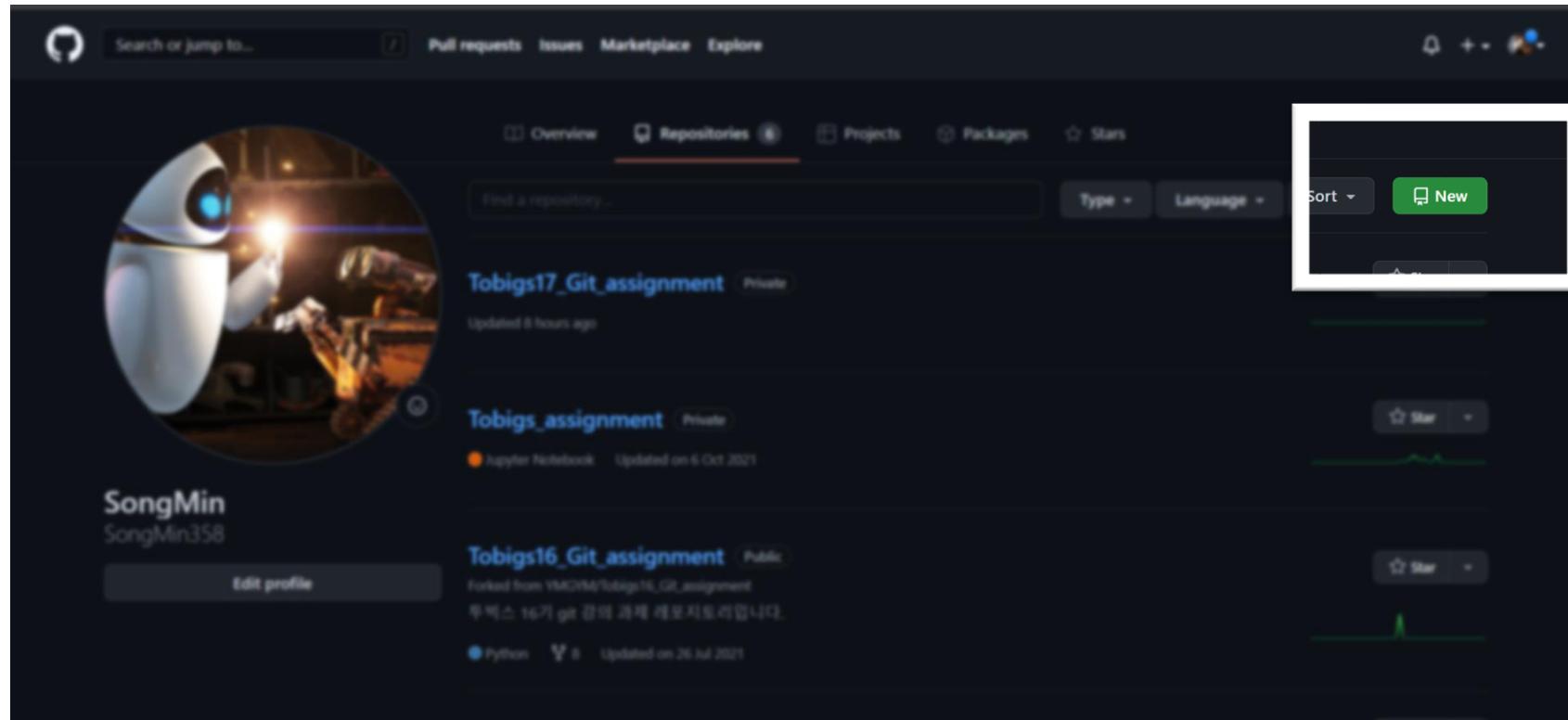
Unit 02 | Git 협업 - 원격 저장소

- 원격 저장소에 로컬 파일 올리기, 내려 받기, 충돌 해결하기
- local repository → remote repository
- remote repository → working directory



Unit 02 | Git 협업 - 원격 저장소

- 원격저장소 만들기



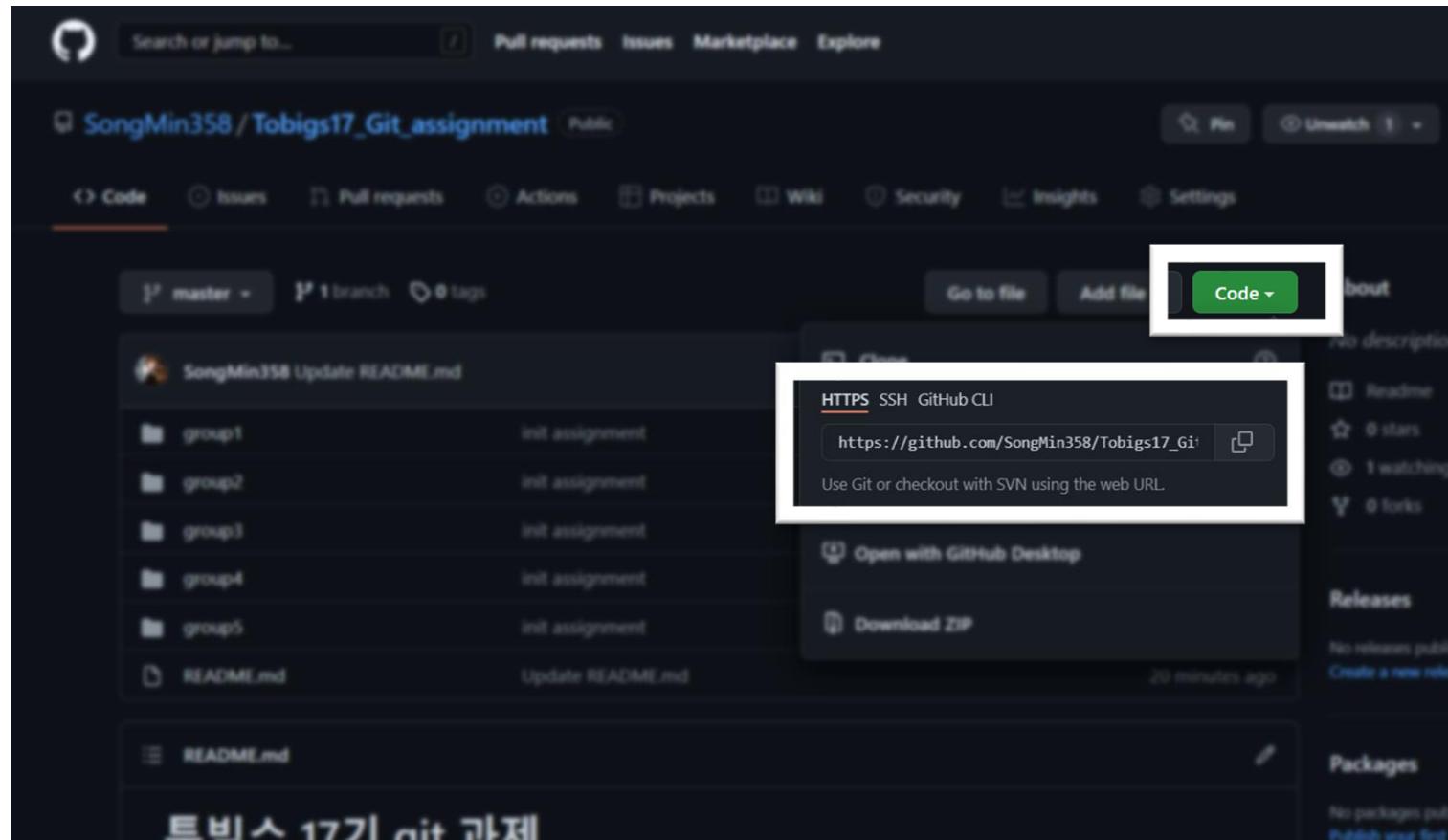
자신의 깃허브 페이지의
Repository 탭에서
'new' 클릭!

'create a new repository' 창 뜨면

Repository name 적어서
저장소명 설정 후
Create repository 버튼 클릭!

Unit 02 | Git 협업 - 원격 저장소

- 내 작업 환경에 원격저장소 등록하기



Repository 탭에 있는 저장소 목록에서
새로 만든 저장소로 들어가기

'code' 클릭해서 나오는
Git 주소 복사

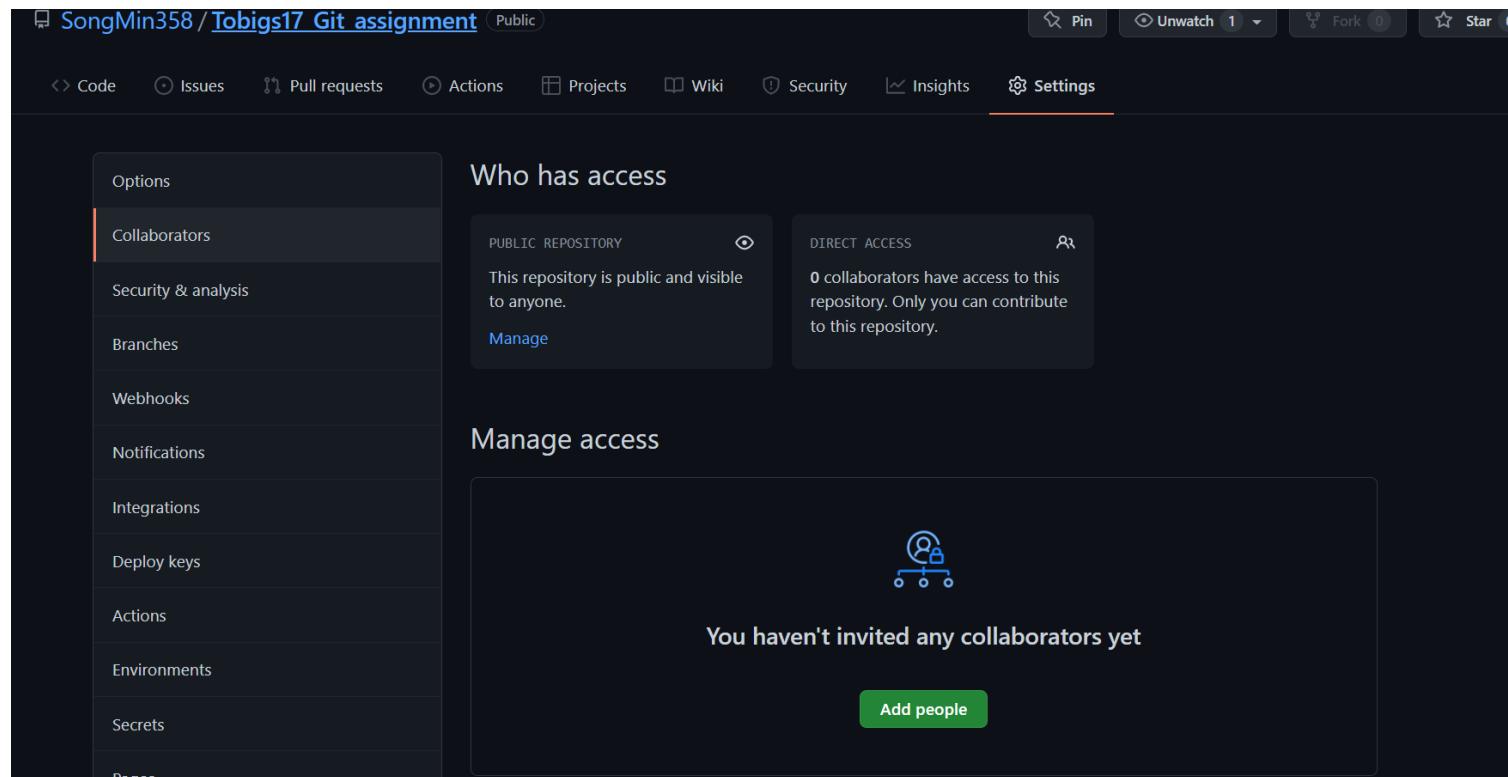
CMD 창에서 working directory로 이동한 뒤,

git remote add origin [원격 저장소 주소]

실행해서 내 작업 환경에 github 주소 등록

Unit 02 | Git 협업 - 원격 저장소

- 해당 저장소 같이 사용할 Collaborator 등록하기



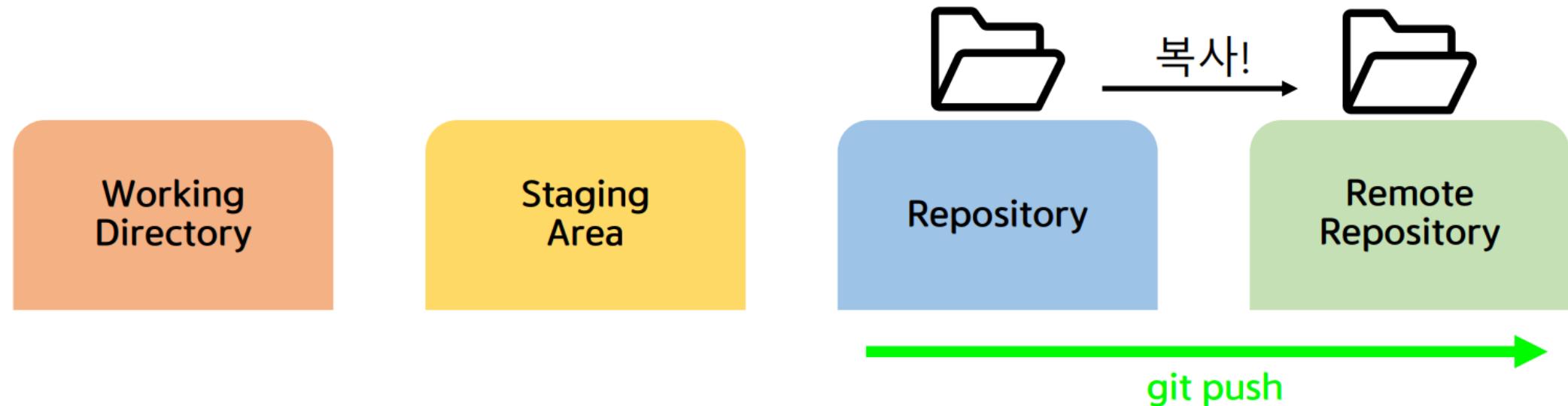
Setting의 collaborators에서

해당 원격저장소를 같이 사용할 사람 등록 가능!

'Add people' 버튼 클릭하고
Username, fullname, email 검색해서 등록

Unit 02 | Git 협업 - 원격 저장소

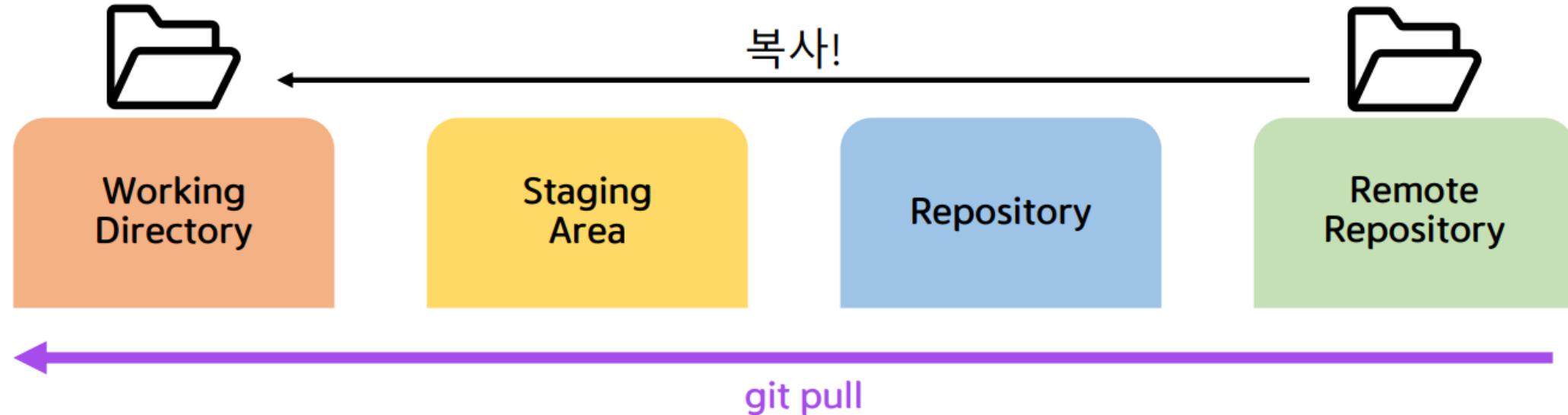
- local repository → remote repository
- 내 로컬 저장소에 있는 내용을 원격 저장소 서버로 올리기



- `git push [Remote명] [Branch명]` 실행 시 원격 저장소로 푸시
- `git push origin master` 가 기본입니다.

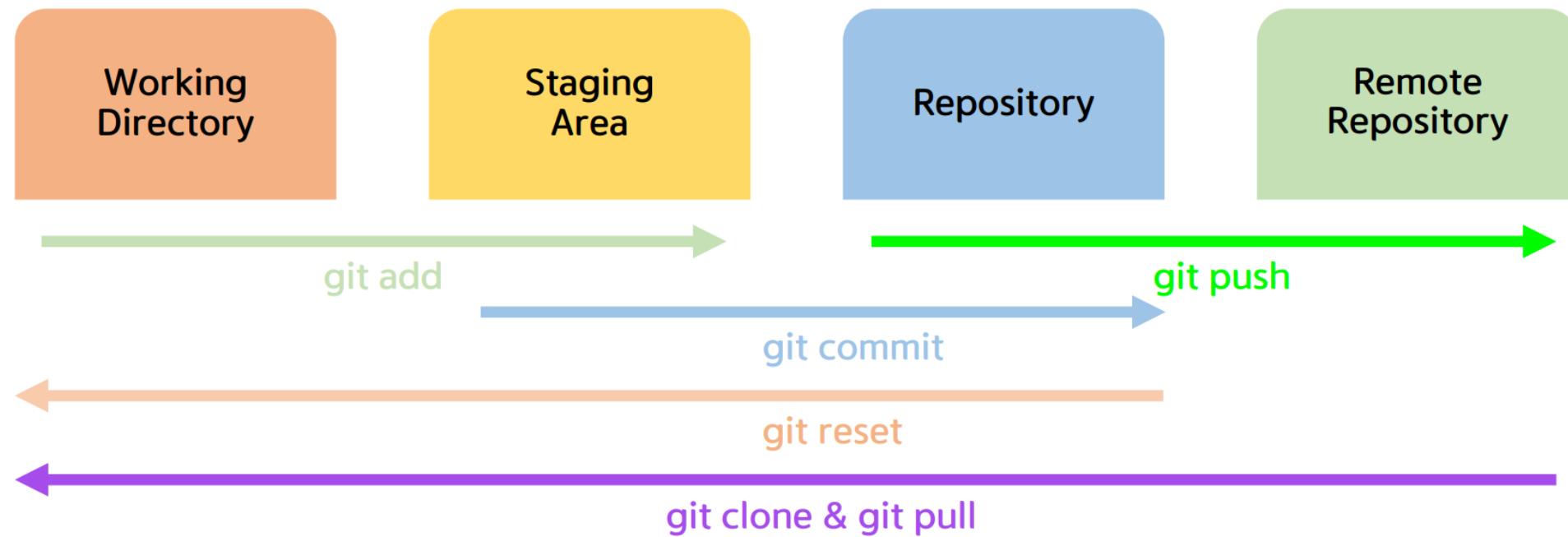
Unit 02 | Git 협업 - 원격 저장소

- remote repository → working directory
- 원격저장소에서 최신 내용 가져와서 내 WD 업데이트



- **git pull [Remote명] [Branch명]** 실행 시 원격 저장소에서 최신 내용 받아오기
- **git pull origin master** 가 기본입니다.

Unit 02 | Git 협업 - 원격 저장소

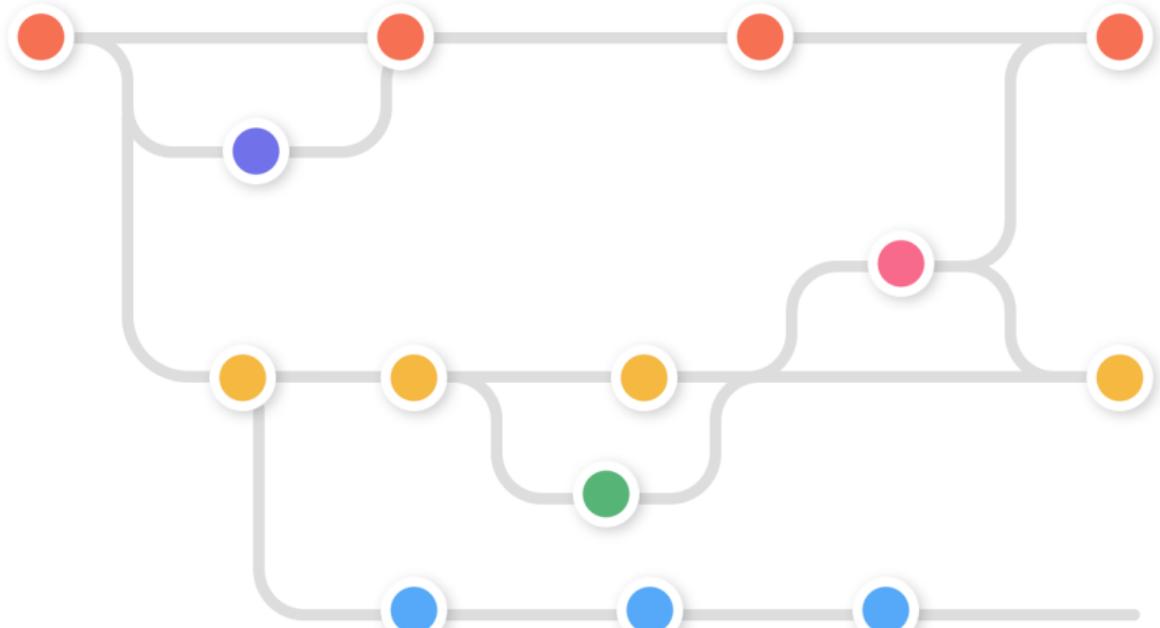


Unit 02 | Git 협업 - 원격 저장소

+ 심화 : 브랜치

- 현재 대표 코드(마스터)는 유지한 채로,
 - 가지 쳐서 편집하고
 - 마스터에 합칠 수도 있는
 - '실험실' 개념
-
- **git branch [브랜치명]**
 - 브랜치 생성
 - **git checkout [브랜치명]**
 - 사용할 브랜치로 전환
 - **git merge [합칠 브랜치명]**
 - 지정한 브랜치를
 - 현재 사용하고 있는 브랜치에 합치기

MASTER
HOTFIX
RELEASE
DEVELOP
FEATURE
FEATURE



Unit 02 | Git 협업 - 원격 저장소

- **merge conflict**
- commit & push 하려고 하는데, 내가 수정한 부분이 원격 저장소에서도 동시에 수정됐을 때 발생하는 오류
 - 해당 원격 저장소를 다시 pull 해서 업데이트 해보면,
 - 충돌이 발생한 파일에 세부적인 내용이 표시됨
- 겹치는 부분 중 어느 부분 남길지 선택, (내 수정본/새로운 수정본/둘 다…)
- 수동으로 수정하여 원하는 부분만 남긴 후 다시 commit & push!

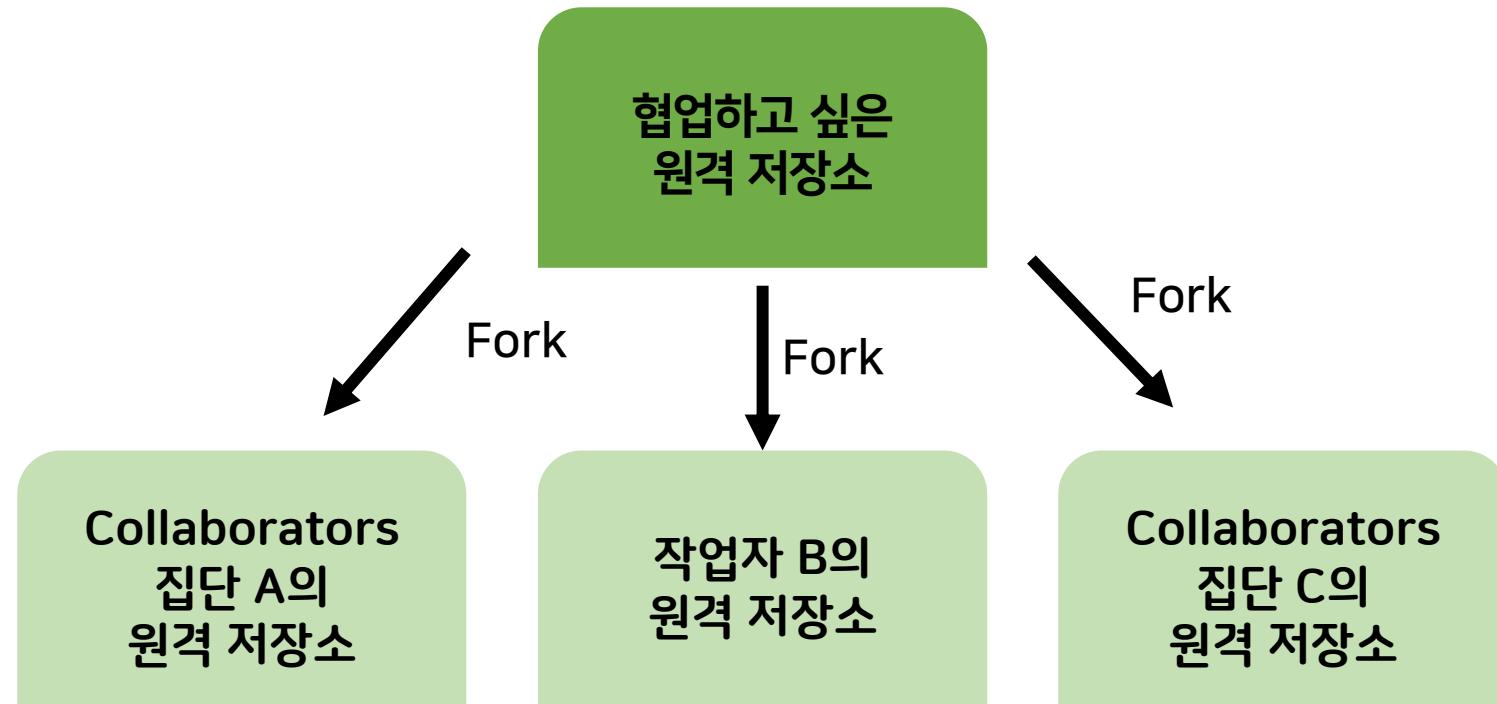
```
colors.txt
src > colors.txt
1 red
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
2 <<<<< HEAD (Current Change)
3 green
4 =====
5 white
6 >>>>> his-branch (Incoming Change)
7 blue

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
react-app-demo my-branch git merge his-branch
Auto-merging src/colors.txt
CONFLICT (content): Merge conflict in src/colors.txt
Automatic merge failed; fix conflicts and then commit the result.
x react-app-demo my-branch >M<
```

Unit 02 | Git 협업 - 원격 저장소

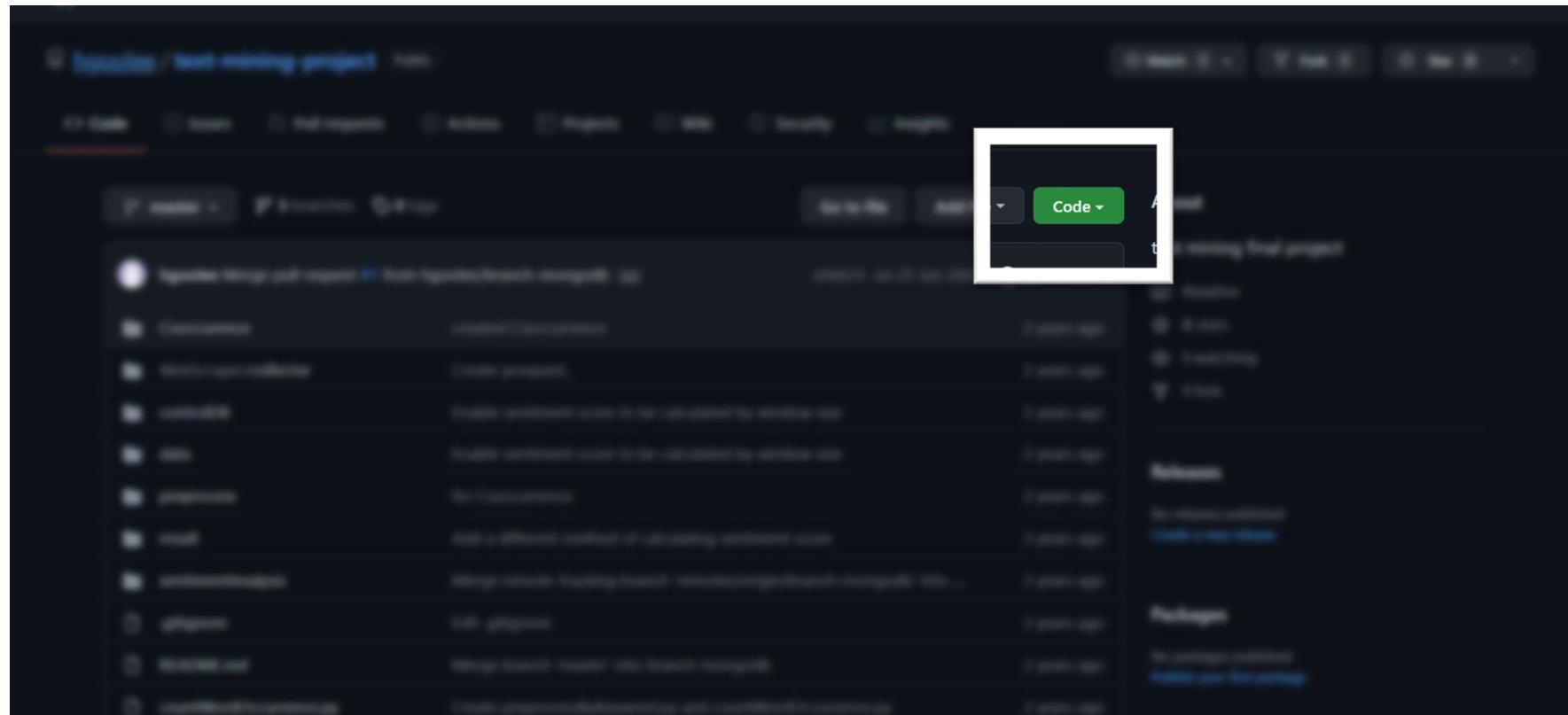
+ 심화 : Collaborator 이외의 원격 저장소에 대한 협업

- 협업하고 싶은 원격 저장소를
- **Fork** 해서 복제한 후 작업,
- 나와 내 collaborators가 작업한 내용을 해당 저장소에 반영시킬 수 있습니다.
 - 합쳐 달라고 (merge해달라고)
 - 요청하면 (pull request하면)
- 해당 저장소의 주인이
- conflict 확인 후 merge 시키고,
- 그러면 해당 저장소에 반영됩니다.



Unit 02 | Git 협업 - 원격 저장소

+ 심화 : Collaborator 이외의 원격 저장소에 대한 협업



눌러서 fork 한 후

Fork 된 원격 저장소를
로컬 저장소로 가져와 (git pull)
작업하면 됩니다

Unit 02 | Git 협업 - 원격 저장소

+ 실제 과정 예시

```
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git init
Reinitialized existing Git repository in C:/Users/ss0/Desktop/Tobigs17_Git_assignment/.git/
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git init
Initialized empty Git repository in C:/Users/ss0/Desktop/Tobigs17_Git_assignment/.git/
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git remote add origin https://github.com/SongMin358/Tobigs17_Git_assignment.git
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    group1/
    group2/
    group3/
    group4/
    group5/

nothing added to commit but untracked files present (use "git add" to track)

(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git add .

(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git commit -m "init assignment"
[master (root-commit) 8522c2e] init assignment
 25 files changed, 475 insertions(+)
 create mode 100644 group1/main.py
 create mode 100644 group1/player1/profile.json
 create mode 100644 group1/player2/profile.json
 create mode 100644 group1/player3/profile.json
 create mode 100644 group1/player4/profile.json
 create mode 100644 group2/main.py
(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git push origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.90 KiB | 389.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/SongMin358/Tobigs17_Git_assignment/pull/new/master
remote:
To https://github.com/SongMin358/Tobigs17_Git_assignment.git
 * [new branch]      master -> master

(base) C:\Users\ss0\Desktop\Tobigs17_Git_assignment>git log
commit 8522c2e446a96bc15441fc93e63fb4d0fb8ce1a3 (HEAD -> master, origin/master)
Author: SongMin358 <ssorange38@gmail.com>
Date:   Tue Jan 18 12:15:42 2022 +0900

  init assignment
```

여기서 잠깐! 과제 제출하는 방법

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



YMGYM ▾

Repository name *



tobigs16



Great repository names are short and memorable. Need inspiration? How about [turbo-octo-giggle](#)?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

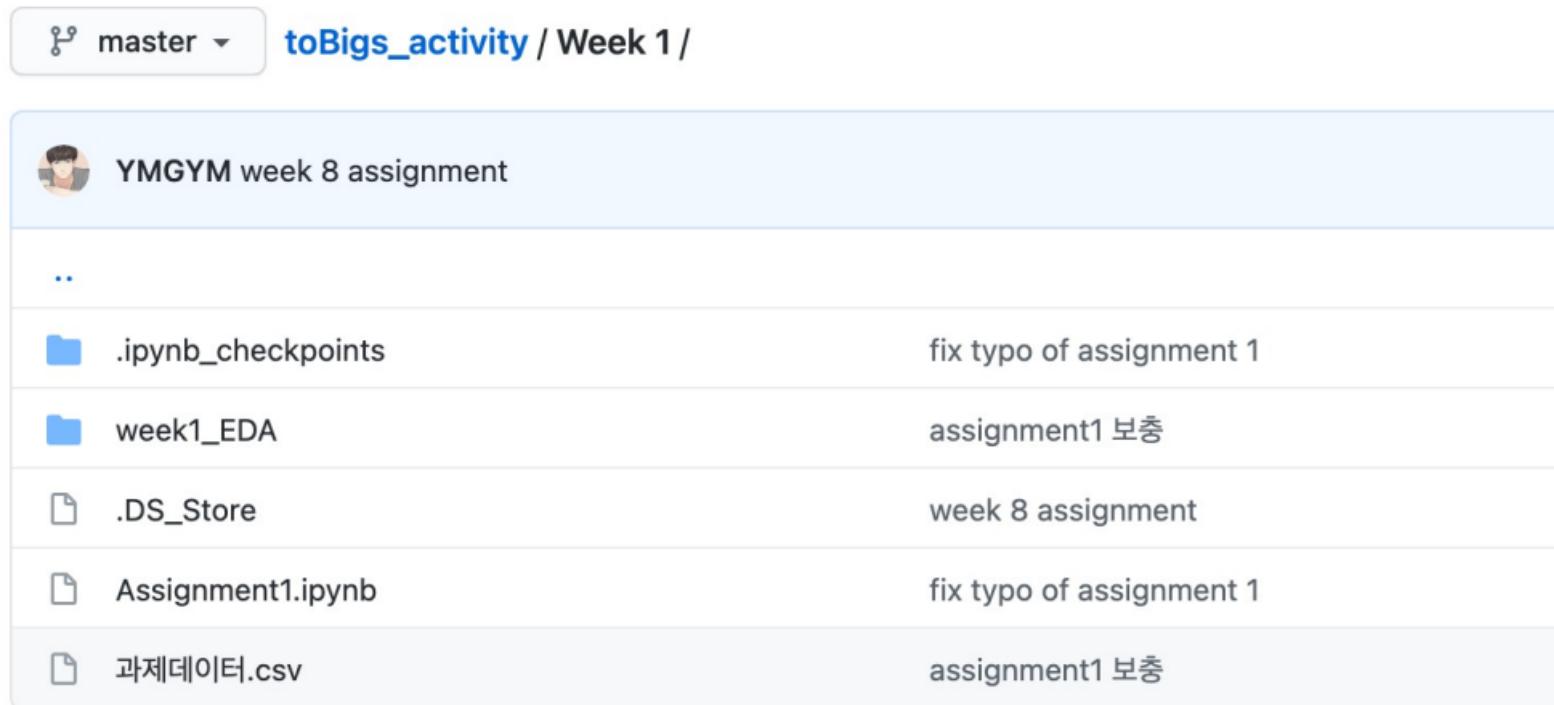
과제용 private 레포지토리 생성

여기서 잠깐! 과제 제출하는 방법

The screenshot shows a GitHub repository settings page for 'YMGYM/toBigs16Git_lecture'. The 'Settings' tab is selected. On the left, a sidebar lists various options: Options, Manage access (highlighted with a red box), Security & analysis, Webhooks, Notifications, Integrations, Deploy keys, Autolink references, Actions, Environments, Secrets, Pages, and Moderation settings. The main area is titled 'Who has access' and shows that the repository is a 'PUBLIC REPOSITORY'. It states that anyone can contribute to this public repository. Below this, the 'DIRECT ACCESS' section indicates that no collaborators have access yet. A modal window is open, titled 'Manage access', with the sub-section 'Invite a collaborator to toBigs16Git_lecture'. It contains a search bar with the text 'tobigs-datamarket' and a result card for 'Tobigs' (tobigs-datamarket) with a 'Invite collaborator' button. The entire 'Manage access' section and the modal are highlighted with a large red box.

Collaborator@ tobigs공식계정 등록

여기서 잠깐! 과제 제출하는 방법

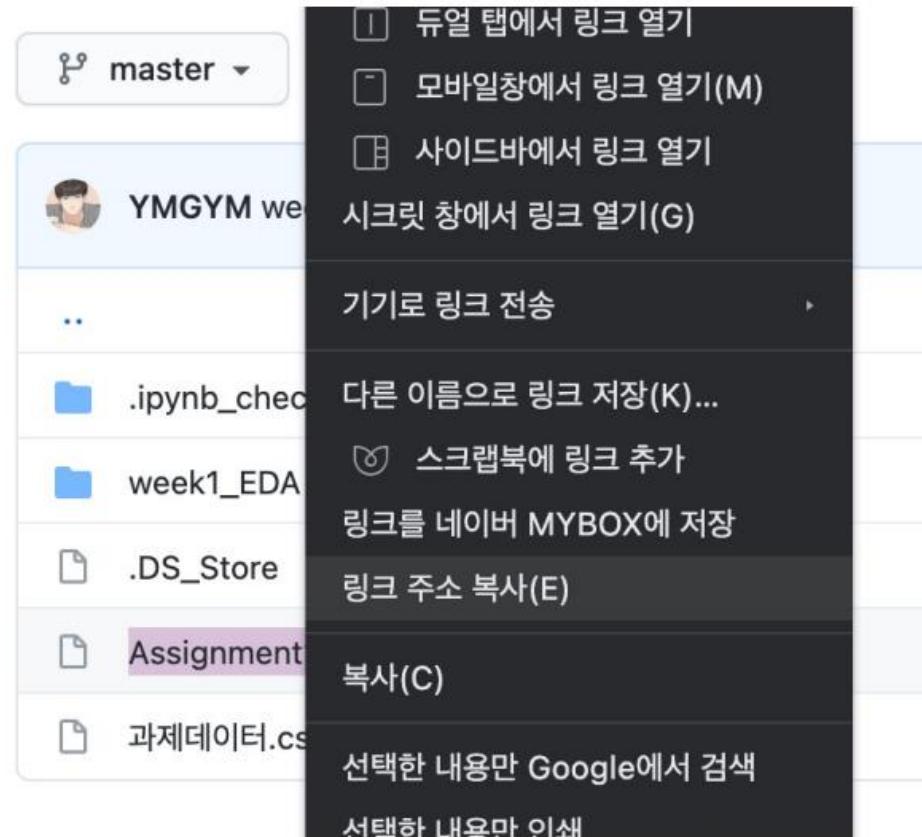


A screenshot of a GitHub repository interface. The repository name is 'toBigs_activity / Week 1 /'. The user profile picture is a cartoon character. The repository title is 'YMGYM week 8 assignment'. The file list includes: .., .ipynb_checkpoints (fix typo of assignment 1), week1_EDA (assignment1 보충), .DS_Store (week 8 assignment), Assignment1.ipynb (fix typo of assignment 1), and 과제데이터.csv (assignment1 보충).

File	Description
..	
.ipynb_checkpoints	fix typo of assignment 1
week1_EDA	assignment1 보충
.DS_Store	week 8 assignment
Assignment1.ipynb	fix typo of assignment 1
과제데이터.csv	assignment1 보충

과제를 push!

여기서 잠깐! 과제 제출하는 방법



과제 링크를 복사

여기서 잠깐! 과제 제출하는 방법

The screenshot shows a forum interface with a navigation bar at the top. The 'Assignment Post' section is highlighted with a red box. Below the navigation bar, there's a sidebar on the left with categories like '공지사항', '강의자료', '과제게시판' (which is highlighted in green), and '스터디 게시판'. The main content area displays a list of assignments with columns for '번호', '제목', '글쓴이', '날짜', and '조회 수'. Each assignment row has a red number indicating the post count.

번호	제목	글쓴이	날짜	조회 수
2935	이수민(15기) - 10주차 과제 (추천 시스템) 1	이수민	2021.04.07	7
2934	권오현(15기) - 10주차 과제 (추천 시스템) 1	권오현	2021.04.07	4
2933	오주영(15기) - 10주차과제(추천시스템) 1	오주영	2021.04.07	6
2932	오주영(15기) - 10주차과제(비지도생성모델) 1	오주영	2021.04.07	3
2931	권오현(15기) - 10주차 과제 (비지도 생성모델) 1	권오현	2021.04.06	2
2930	이수민(15기) - 10주차 과제 (비지도 생성모델) 1	이수민	2021.04.06	2
2929	강지우(15기) - 10주차 과제 (추천 시스템) 1	강지우	2021.04.06	6
2928	이윤정(15기) - 10주차 과제 (추천 시스템) 1	윤정	2021.04.06	3

데이터마켓 로그인 후 멤버 게시판 -> 과제 게시판

여기서 잠깐! 과제 제출하는 방법

The screenshot shows a user interface for assignment submission. On the left, there's a sidebar with navigation links: '멤버 게시판' (Member Board), '공지사항' (Announcements), '강의자료' (Lecture Materials), '과제게시판' (Assignment Board) which is highlighted in green, '스터디 게시판' (Study Board), and 'sas스터디'. The main area shows a breadcrumb navigation: Home > 멤버 게시판 > 과제게시판. Below this is a rich text editor toolbar with various formatting options like bold, italic, underline, and alignment. A red box highlights a specific section of the editor. Inside this red box, the title '000(17기) – n주차 과제 [강의명]' is displayed above a text input field containing the link '링크 : https://github.com/YMGYM/toBigs_activity/blob/master/Week%201/Assignment1.ipynb'. There are also buttons for '불러오기' (Import) and '확장 컴포넌트' (Advanced Component). The top right of the page has buttons for '본문확장' (Expand Content), '높이축소' (Reduce Height), and '넓이확장' (Expand Width).

Computer Science

Unit 01 | Git 기초 - 로컬 저장소

Unit 02 | Git 협업 - 원격 저장소

Unit 03 | **Framework**

Unit 03-1 | Tensorflow

Unit 03-2 | Keras

Unit 03-3 | Pytorch

Unit 03 | Framework

응용 프로그램을 개발하기 위한 여러 라이브러리나 모듈 등
하나로 묶은 일종의 패키지

검증된 알고리즘을 사용할 때마다 구현하는 것은 비효율적,
빠르게 핵심만 구현하도록 도와주는 역할

Unit 03 | Framework

프레임워크의 장점은?

1. 복잡한 모델의 구조를 몰라도 정해진 틀에 따라 사용해볼 수 있습니다.
2. GPU 활용하여 빠르게 처리할 수 있습니다.

Unit 03 | Framework

Frame(틀) + Work(일하다)

= Framework(일정하게 짜여진 틀, 뼈대를 가지고 일하다)



Unit 03 | Framework



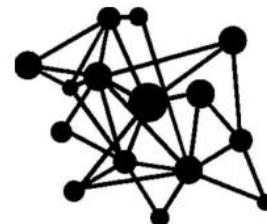
TensorFlow



PyTorch



Keras



DL4J



Caffe

Unit 03 | Framework



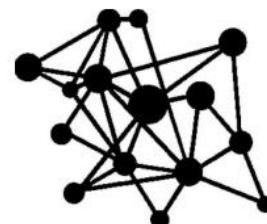
TensorFlow



PyTorch



Keras



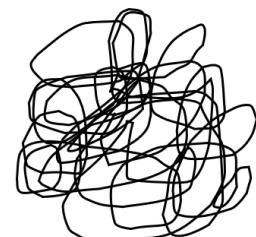
DL4J



Caffe

Unit 03 | Framework

- 이 프레임워크들의 기본적인 data structure Tensor 란?

7	$\begin{bmatrix} 15 \\ -3 \\ \vdots \\ 7 \end{bmatrix}$	$\begin{bmatrix} 15 & 1 \dots 11 \\ -3 & 8 \dots 9 \\ \vdots & \vdots \ddots \vdots \\ 7 & 9 \dots 5 \end{bmatrix}$	$\left[\begin{array}{cccc} 15 & 1 \dots 11 \\ -3 & 8 \dots 9 \\ \vdots & \vdots \ddots \vdots \\ 7 & 9 \dots 5 \end{array} \right]$	
스칼라	벡터	행렬	3차원 텐서	N차원 텐서
0차원	1차원	2차원	3차원	$x[1, 3, \dots, 1] = 7$ <small>N개 인덱스</small>

넘파이 배열과 유사
다차원 배열(multidimensional array) 형태라고 이해해볼 수 있음

Unit 03 | Framework

Tensorflow란?

TensorFlow를 사용해야 하는 이유

TensorFlow는 머신러닝을 위한 엔드 투 엔드 오픈소스 플랫폼입니다. 도구, 라이브러리, 커뮤니티 리소스로 구성된 포괄적이고 유연한 생태계를 통해 연구원들은 ML에서 첨단 기술을 구현할 수 있고 개발자들은 ML이 접목된 애플리케이션을 손쉽게 빌드 및 배포할 수 있습니다.

정보 →



손쉬운 모델 빌드

즉각적인 모델 반복 및 손쉬운 디버깅을 가능하게 하는 즉시 실행 기능이 포함된 Keras와 같은 높은 수준의 직관적인 API를 사용하여 ML 모델을 쉽게 빌드하고 학습시키세요.



어디서든 강력한 ML 제작

사용하는 언어에 상관없이 클라우드, 온프레, 브라우저 또는 기기에서 모델을 손쉽게 학습시키고 배포하세요.



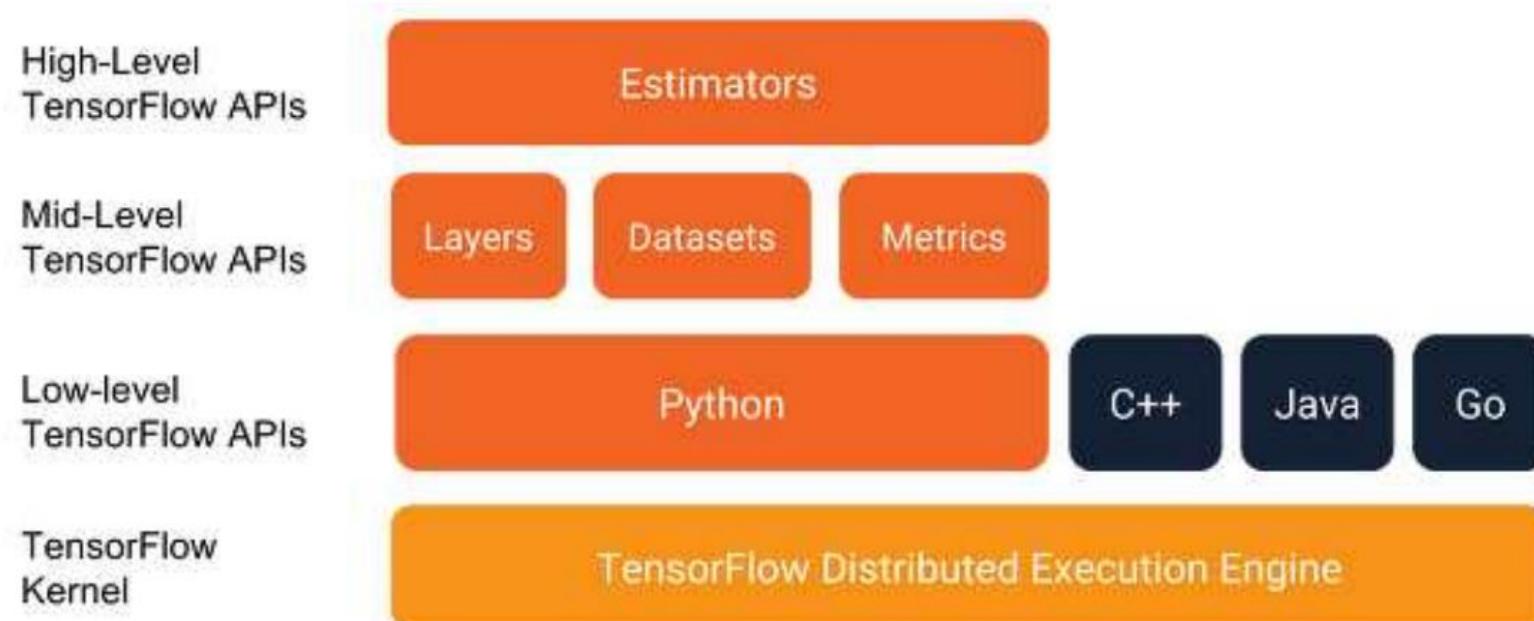
연구를 위한 강력한 실험

새로운 아이디어를 개념부터 코드, 최첨단 모델, 계시에 이르기까지 더 빠르게 발전시킬 수 있는 간단하고 유연한 아키텍처

딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 머신러닝 라이브러리로, 구글에서 만들었음.

Unit 03 | Framework

Tensorflow란?



파이썬을 최우선으로 지원하여 대다수 편의 기능이 **파이썬 라이브로리로만 구현되어있음.**

Unit 03 | Framework

왜 Tensorflow인가?



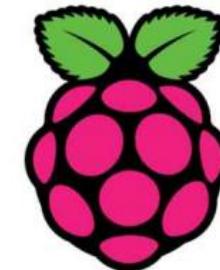
Mac OS



Linux



android

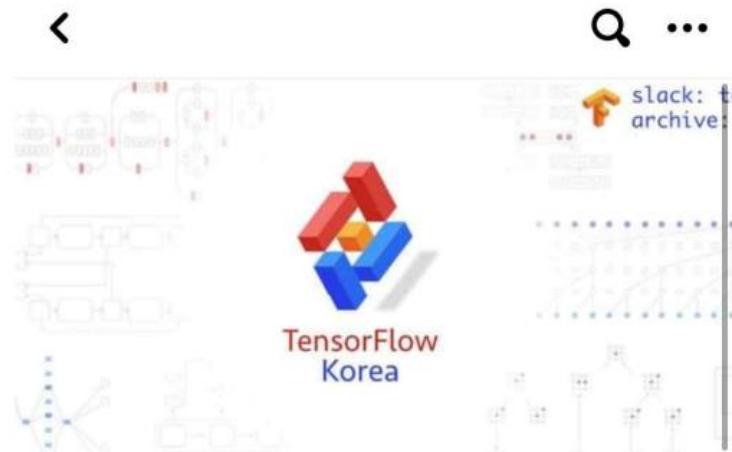


Raspberry Pi

윈도우, 맥, 리눅스 등의 다양한 시스템에서 쉽게 사용할 수 있
도록 지원

Unit 03 | Framework

왜 Tensorflow인가?



TensorFlow KR >

전체 공개 그룹 · 멤버 5.5만명

현존하는 머신러닝 라이브러리 중 커뮤니티가 가장 활발함

Unit 03 | Framework

Keras란?

케라스: 파이썬 딥러닝 라이브러리

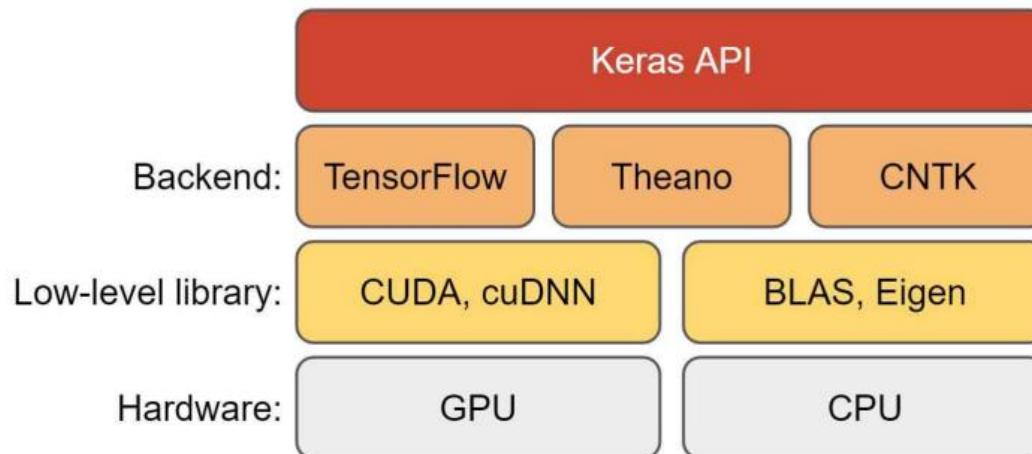


케라스에 오신걸 환영합니다.

케라스는 파이썬으로 작성된 고수준 신경망 API로 [TensorFlow](#), [CNTK](#), 혹은 [Theano](#)와 함께 사용하실 수 있습니다. 빠른 실험에 특히 중점을 두고 있습니다. 아이디어를 결과물로 최대한 빠르게 구현하는 것은 훌륭한 연구의 핵심입니다.

Unit 03 | Framework

Q. Tensorflow와 Keras 이 둘은 무슨 사이인가요?



Tensorflow와 마찬가지로 Keras 역시 라이브러리인데, Keras는 **Tensorflow(혹은 Theano나 CNTK)** 위에서 동작하는 라이브러리입니다.

Unit 03 | Framework

Q. Tensorflow가 있는데 왜 그 위에서 동작하는 Keras가 필요하나요?

Tensorflow는 처음 머신러닝을 접하는 사람이라면 아직 사용하기에는 어려운 부분이 많음. 반면, Keras는 사용자 친화적으로 개발된 라이브러리라 매우 사용이 편하기 때문에 **머신러닝 초보자가 공부하기에 안성맞춤!**

Q. 그럼 Keras만 사용해도 충분할까요?

단순한 신경망을 구현하는 정도라면 Keras만으로 충분하겠지만, **Tensorflow를 사용하면 더 디테일한 조작이 가능.**
따라서 사용 목적에 따라서 선택하는 것이 좋을 것!

Unit 03 | Framework

Pytorch란?



Deep Learning with PyTorch

PyTorch 소개

PyTorch(파이토치)는 연구용 프로토타입부터 상용 제품까지 빠르게 만들 수 있는 오픈 소스 머신러닝 프레임워크입니다.

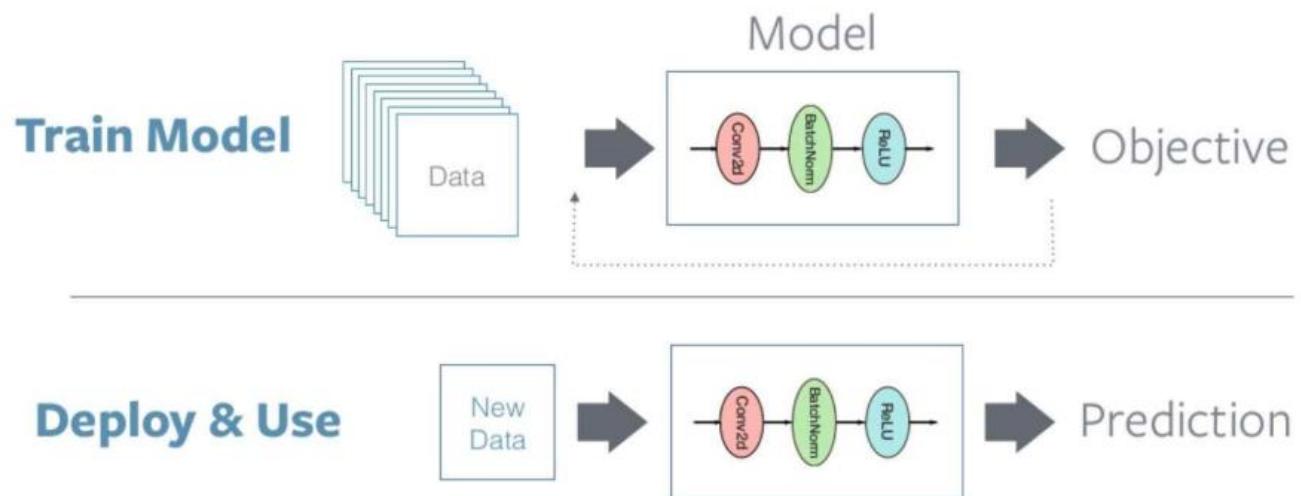
PyTorch는 사용자 친화적인 프론트엔드(front-end)와 분산 학습, 다양한 도구와 라이브러리를 통해 빠르고 유연한 실험 및 효과적인 상용화를 가능하게 합니다.

PyTorch에 대한 더 자세한 소개는 [공식 홈페이지](#) 또는 [공식 저장소](#)에서 확인하실 수 있습니다.

2016년에 발표된 딥러닝 구현을 위한 파이썬 기반의 오픈소스 머신러닝 라이브러리로 facebook 인공지능 연구팀에 의해 개발되었음

Unit 03 | Framework

Pytorch란?



현재의 딥러닝 트렌드에 맞춰 제작된 다른 프레임워크들과 다르게
pytorch는 미래의 연구 트렌드에도 대응할 수 있도록 제작됨

Unit 03 | Framework

Pytorch란?

< Q ...



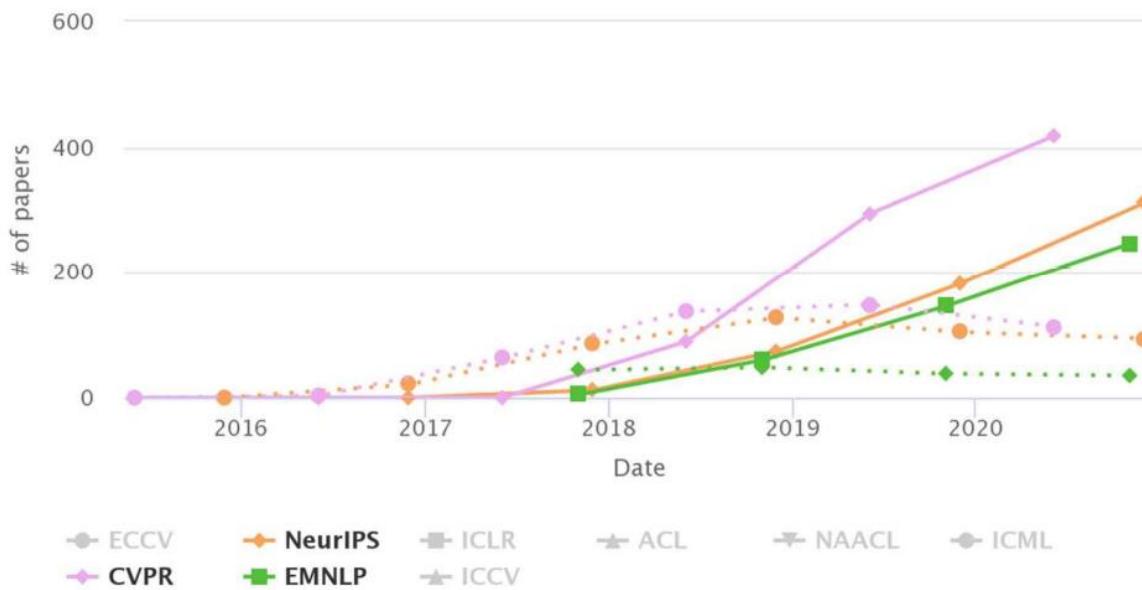
PyTorch KR >

전체 공개 그룹 · 멤버 1.1만명

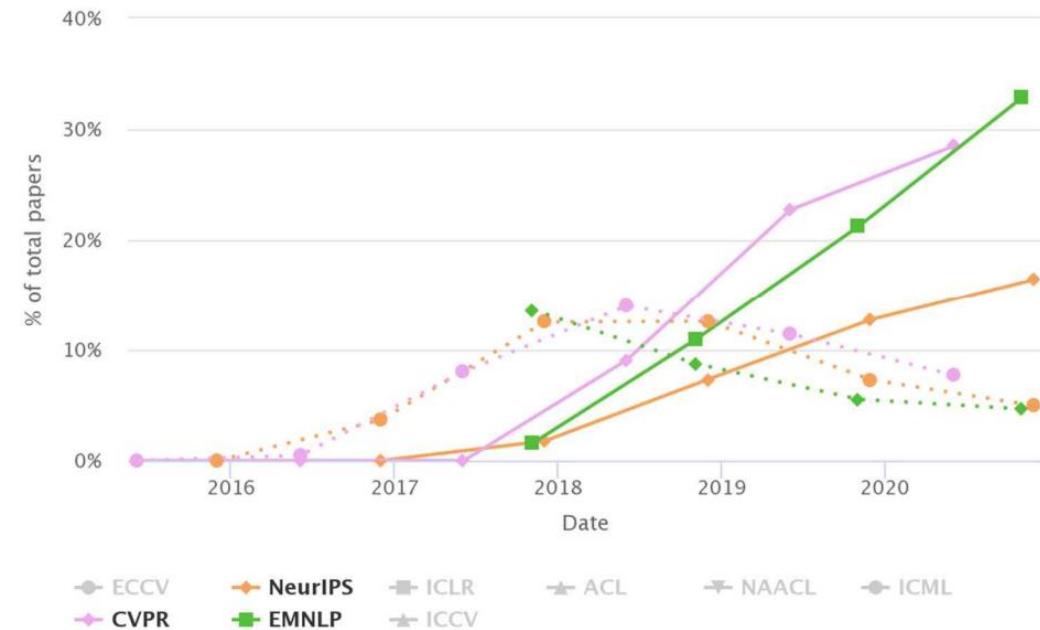
Tensorflow 못지 않게 커뮤니티가 활발함

Unit 03 | Framework

PyTorch (Solid) vs TensorFlow (Dotted) Raw Counts



PyTorch (Solid) vs TensorFlow (Dotted) % of Total Papers

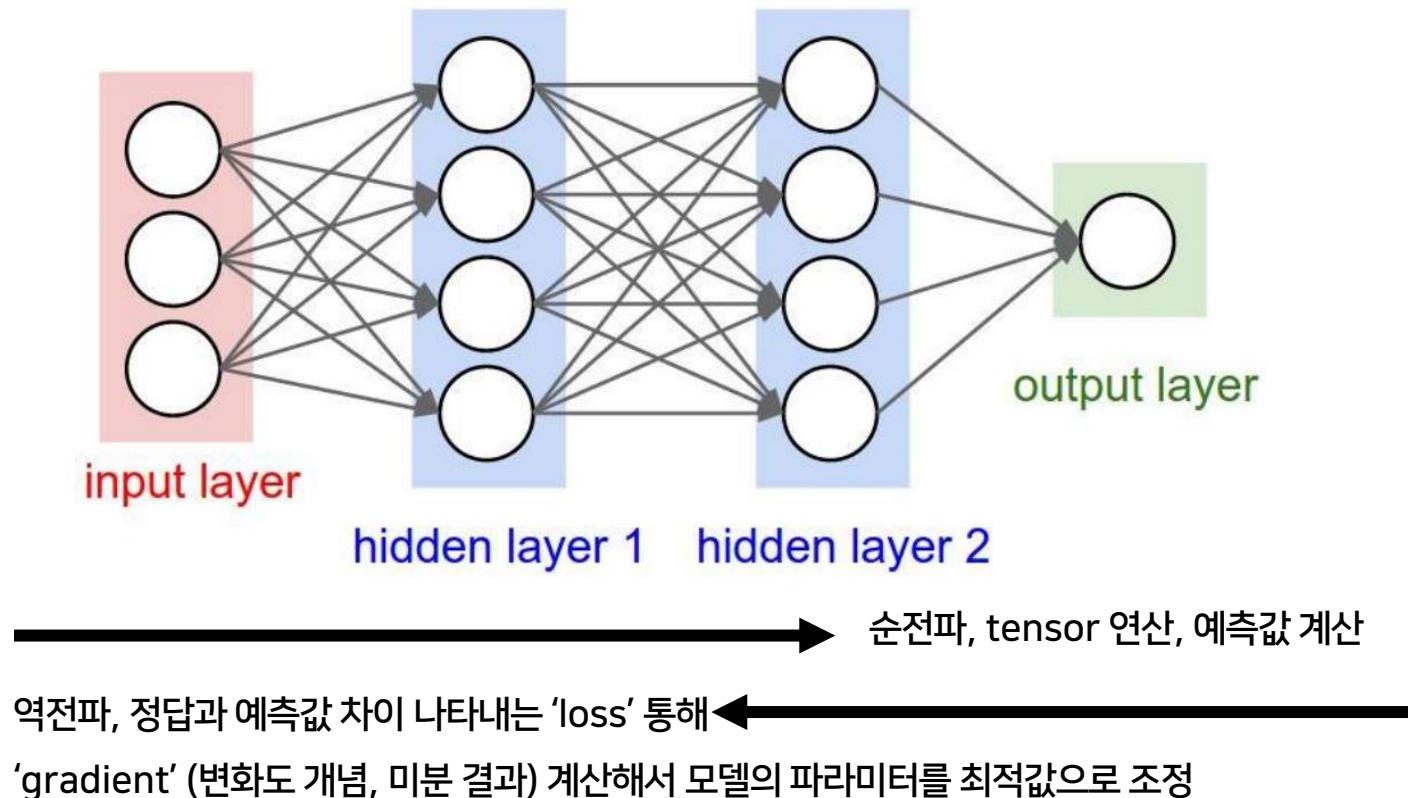


Pytorch로 쓰이는 논문 많아지고 있는 경향

Unit 03 | Framework

- MNIST 샘플 데이터로 Deep Neural Network 구현하는 예제 _ 프레임워크별로 살펴보기

OVERVIEW



- 데이터셋 불러오고 전처리
- 데이터 로더 설정
- 모델 선언
- 손실함수, 옵티마이저 선언
- 모델 학습
- 모델 평가

Unit 03 | Framework

- 데이터 로더
 - 데이터셋을 한 번에 모두 램에 올리지 않고,
 - 각 미니배치 만큼만 램에 올리도록 하여 효율적인 연산 가능하게 함
- 모델 선언
 - 텐서플로, 파이토치에서 모델 선언 시 모델 관련 추상 클래스를 상속받아
 - `__init__` 메소드로 각 레이어를 생성하고
 - 각각 `call`, `forward` 메소드를 이용해 데이터가 각 레이어를 통과하도록 함

+ 추상클래스는 메서드의 구현은 없이 이름만 존재하도록 정의해 둔 클래스를 의미한다.
프레임워크의 추상클래스를 모델 선언시에 상속 받아, 구현이 요구되는 특정 메서드들을 구현하면 되는 상황
- 손실함수와 옵티마이저
 - 태스크에 맞는 손실함수를 선언 손실함수 정리 <https://bskyvision.com/822>
 - 손실함수를 통해 계산된 값을 이용해 가중치를 최적화하는 옵티마이저 선언

Computer Science

Unit 01 | Git 기초 - 로컬 저장소

Unit 02 | Git 협업 - 원격 저장소

Unit 03 | Framework

Unit 03-1 | **Tensorflow**

Unit 03-2 | Keras

Unit 03-3 | Pytorch

Unit 03-1 | Tensorflow

- 1) 데이터 로더 생성
- 2) 모델 선언
- 3) 손실함수 및 옵티마이저 선언
- 4) 학습 함수 생성
- 5) 모델 학습

Unit 03-1 | Tensorflow

1) 데이터 로더 생성

Tensorflow, 모델에서 사용할 레이어, Model 모듈
기본적으로 가져오기

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
```

데이터 불러오기 (예시)

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

데이터 로더 생성,
불러온 데이터를 미니배치(덩어리)로 나누어
각 미니배치만큼만 램에 올려서 계산 효율

```
train_ds = tf.data.Dataset.from_tensor_slices(
    (x_train, y_train)).shuffle(10000).batch(32)
test_ds = tf.data.Dataset.from_tensor_slices((x_test, y_test)).batch(32)
```

Unit 03-1 | Tensorflow

2) 모델 선언

- Keras api 이용해 각 레이어 선언
- 추상클래스 Model 상속받아서
 - `__init__`으로 각 레이어 생성
 - `call`로 입력 데이터가 모델의 각 레이어 통과하도록 함

- `super()`

자식 클래스에서 부모클래스의 내용을 사용하고 싶을 경우

- `Flatten()`

다차원 배열을 1차원으로 변환해서 평탄화

```
class MyModel(Model): # Model 추상 클래스 상속받은 MyModel class
    # __init__ 메소드로 모델의 각 레이어 생성
    def __init__(self):
        super(MyModel, self).__init__()
        self.flatten = Flatten()
        self.d1 = Dense(256, activation = 'relu')
        self.d2 = Dense(256, activation = 'relu')
        self.d3 = Dense(10)

    # call 메소드로 입력 데이터가 모델의 각 레이어 통과하도록 함
    def call(self, x):
        x = self.flatten(x)
        x = self.d1(x)
        x = self.d2(x)
        return self.d3(x)

# Create an instance of the model
model = MyModel()
```

Unit 03-1 | Tensorflow

3) 손실함수 및 옵티마이저 선언

```
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

optimizer = tf.keras.optimizers.Adam()
```

- 여기서는 라벨이 정수 형태로 제공돼서 sparse categorical crossentropy 사용,
- 옵티마이저는 Adam 사용

4) 학습 함수 생성

- 프로세스

예측(predictions) -> 손실함수 계산(loss) ->
Gradient 계산 -> 최적화(optimizer)

- GradientTape

Gradient 계산 시 역전파 (거슬러 올라갈) 필요
GradientTape 통해서 순전파 시의 계산 그래프
기록하여 사용할 수 있게 됨

```
@tf.function
def train_step(images, labels):
    with tf.GradientTape() as tape:
        # training=True is only needed if there are layers with different
        # behavior during training versus inference (e.g. Dropout).
        predictions = model(images, training=True)
        loss = loss_object(labels, predictions)
        gradients = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients(zip(gradients, model.trainable_variables))

    train_loss(loss)
    train_accuracy(labels, predictions)
```

Unit 03-1 | Tensorflow

5) 모델 학습

- For문으로 에포크 수 만큼 반복시키기
 - 전체 에포크에 대해 학습 및 평가 함수를 반복하면서 학습과정을 기록하도록 코드 작성
- 이 과정에서 early stopping 등의 방법 이용 가능
 - Early stopping: 성능이 더 나아지지 않을 경우 반복 멈추도록 함

```
train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.Accuracy(name='train_accuracy')
test_loss = tf.keras.metrics.Mean(name='test_loss')
test_accuracy = tf.keras.metrics.Accuracy(name='test_accuracy')
```

```
EPOCHS = 5

for epoch in range(EPOCHS):
    # Reset the metrics at the start of the next epoch
    train_loss.reset_states()
    train_accuracy.reset_states()
    test_loss.reset_states()
    test_accuracy.reset_states()

    for images, labels in train_ds:
        train_step(images, labels)

    for test_images, test_labels in test_ds:
        test_step(test_images, test_labels)

    print(
        f'Epoch {epoch + 1}, '
        f'Loss: {train_loss.result()}, '
        f'Accuracy: {train_accuracy.result() * 100}, '
        f'Test Loss: {test_loss.result()}, '
        f'Test Accuracy: {test_accuracy.result() * 100}'
    )
```

Computer Science

Unit 01 | Git 기초 - 로컬 저장소

Unit 02 | Git 협업 - 원격 저장소

Unit 03 | Framework

Unit 03-1 | Tensorflow

Unit 03-2 | Keras

Unit 03-3 | Pytorch

Unit 03-2 | Keras

- 1) 모델 선언
- 2) 컴파일(손실함수, 옵티마이저, 평가지표 설정)
- 3) 모델 학습

Unit 03-2 | Keras

1) 모델 선언

- Sequential: 모델이 하나의 입력과 출력 가지는 경우 간단하게 각 레이어를 하나의 모델로 묶어주는 생성자
- model을 즉각적으로 인스턴스로 생성하며, 인스턴스 안에 model의 모든 정보 담겨야 함

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

2) 컴파일(손실함수, 옵티마이저, 평가지표 설정)

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

미리 인스턴스로 생성해서 설정할 수도 있습니다

```
optimizer = keras.optimizers.SGD(lr=0.01, momentum=0.5)
model.compile(optimizer=optimizer,
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Unit 03-2 | Keras

3) 모델 학습

- model 인스턴스의 fit 메소드 통해 진행
- Early stopping 등의 callback 함수, validation 반복 등에 대한 설정을 파라미터로 추가
- 들어갈 데이터, 에포크, 배치 사이즈도 여기서 설정

```
callbacks = [
    keras.callbacks.EarlyStopping(
        monitor="val_loss",
        min_delta=1e-2,
        patience=2,
        verbose=1,
    )
]
model.fit([
    x_train,
    y_train,
    epochs=20,
    batch_size=64,
    callbacks=callbacks,
    validation_split=0.2,
])
```

Computer Science

Unit 01 | Git 기초 - 로컬 저장소

Unit 02 | Git 협업 - 원격 저장소

Unit 03 | Framework

Unit 03-1 | Tensorflow

Unit 03-2 | Keras

Unit 03-3 | Pytorch

Unit 03-3 | Pytorch

- 1) 하이퍼 파라미터 및 device 설정
- 2) 데이터 및 데이터 로더 설정
- 3) 모델 선언
- 4) 모델, 손실함수 및 옵티마이저 설정
- 5) 학습 함수 설정
- 6) 평가 함수 설정
- 7) 모델 학습

Unit 03-3 | Pytorch

1) 하이퍼 파라미터 및 device 설정

- 파이토치는 특히 직접 GPU를 할당해야 GPU 이용 가능
 - 사용하고 있는 device를 print하여 확인할 것 (예시=Colab)
 - ‘using cuda device’ 출력되어야 다음 진행
 - ‘using cpu device’일 경우 런타임 유형 GPU로 변경

```
1 learning_rate = 1e-3
2 batch_size = 64
3 epochs = 5
4
5 device = 'cuda' if torch.cuda.is_available() else 'cpu'
6 print('Using {} device'.format(device))
7
8 # 런타임 -> 런타임 유형 변경 -> None을 GPU로 변경
```

2) 데이터 및 데이터 로더 설정

- `torch.utils.data.DataLoader()`
 - train: train 데이터 셋인지, test 데이터 셋인지 여부
 - download: 로컬에 다운받을지 여부
 - transform: 이미지 데이터 증강 (augmentation) 함수
 - 이미지 회전, 확대, 왜곡, 반전 등 무작위의 (그러나 사실적인) 변환을 적용하여 훈련 세트의 다양성을 증가시키는 함수

Unit 03-3 | Pytorch

3) 모델 선언

- 추상 클래스 nn.Module 상속받아
 - `__init__`으로 모델의 레이어 설정
 - `forward`로 순전파 진행하여 통과시키기

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 256),
            nn.ReLU(),
            nn.Linear(256, 256),
            nn.ReLU(),
            nn.Linear(256, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

4) 모델, 손실함수 및 옵티마이저 설정

- 적절한 손실함수, 옵티마이저 선택하여 선언
- GPU 사용 위해서는 **필수적으로 모델을 GPU에 올려야**, 즉 GPU 환경으로 옮겨주어야 함
 - `.to(device)`

```
model = NeuralNetwork().to(device)
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

Unit 03-3 | Pytorch

5) 학습 함수 설정

- For문으로 에포크 수 만큼 반복시키기
- 학습에 사용될 데이터 역시 GPU에 올려야 GPU 사용 가능
- 함수 살펴보기
 - optimizer.zero_grad(): 모델 각 파라미터의 gradient 0으로 초기화
 - loss.backward(): 기록해 둔 계산 그래프 따라 gradient 역전파
 - optimizer.step(): 역전파된 그래디언트 이용해 모델 파라미터 업데이트
- 필요 시 early stopping 등의 기능 추가 가능

```
def train_loop(dataloader, model, loss_fn, optimizer):  
    model.train()  
    size = len(dataloader.dataset)  
    for batch, (X, y) in enumerate(dataloader):  
        X, y = X.to(device), y.to(device)  
        # Compute prediction and loss  
        pred = model(X)  
        loss = loss_fn(pred, y)  
  
        # Backpropagation  
        optimizer.zero_grad()  
        loss.backward()  
        optimizer.step()  
  
        if batch % 100 == 0:  
            loss, current = loss.item(), batch * len(X)  
            print(f"loss: {loss:.7f} [{current:.5d}/{size:.5d}]")
```

Unit 03-3 | Pytorch

6) 평가 함수 설정

보통 아래 두 함수 같이 쓰임

- `model.eval()`: 모델 각 레이어를 상황에 맞게끔 설정
 - dropout layer, batchnorm layer 등 train할 때와 다르게 동작해야 하는 경우가 있으면 알아서 off 시키도록 함
 - 다시 train 하려면 `.train()` 함수 실행해서 모델 변경)
- `torch.no_grad()`: 모델이 계산 그래프 생성하지 않도록 설정, 연산량과 램 효율 극대화

```
def test_loop(dataloader, model, loss_fn):  
    model.eval()  
    size = len(dataloader.dataset)  
    num_batches = len(dataloader)  
    test_loss, correct = 0, 0  
  
    with torch.no_grad():  
        for X, y in dataloader:  
            X, y = X.to(device), y.to(device)  
            pred = model(X)  
            test_loss += loss_fn(pred, y).item()  
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()  
  
    test_loss /= num_batches  
    correct /= size
```

7) 모델 학습

- For문으로 에포크 수 만큼 반복시키기
 - 전체 에포크에 대해 학습 및 평가 함수를 반복
- 모델 저장, early stopping, 시각화 등의 방법 추가 가능

```
epochs = 5  
for t in range(epochs):  
    print(f"Epoch {t+1}-----")  
    train(train_dataloader, model, loss_fn, optimizer)  
    test(test_dataloader, model, loss_fn)  
print("Done!")
```

과제 링크

https://github.com/SongMin358/Tobigs17_Git_assignment

출처

- 투빅스 15기 안민준님 강의 자료
- 깃 홈페이지 [About - Git \(git-scm.com\)](#)
- 깃 Reference sheet [Git: Reference Sheet – NeSI Support](#)
- Merge conflict tutorial [Git merge conflict tutorial – Ihatetomatoes](#)
- https://backlog.com/git-tutorial/kr/intro/intro4_3.html
- <https://velog.io/@joygoround/Git-git-github-%EA%B8%B0%EC%B4%88-%EC%97%B0%EC%8A%B5>

- 투빅스 12기 이승현님 강의 자료
- 투빅스 15기 김재희님 강의 자료
- 투빅스 15기 조효원님 강의 자료
- <https://www.tensorflow.org/tutorials/quickstart/beginner?hl=ko>
- <https://tutorials.pytorch.kr/>
- https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/api_docs/python/
- <골빈해커의 3분 딥러닝,텐서플로맛>(김진중 저/한빛미디어)
- <코딩 셰프의 3분 딥러닝, 케라스맛> (김성진 저/한빛미디어)
- <펭귄브로의 3분 딥러닝, 파이토치맛> (김건우, 엄상준 저/한빛미디어)

출처

- <https://engkimbs.tistory.com/673>
- <https://backgomc.tistory.com/78>
- <https://www.slideshare.net/SessionsEvents/soumith-chintala-ai-research-engineer-facebook-at-mlconf-nyc-2017>
- <https://tensorflow.blog/케라스-딥러닝/3-2-케라스-소개/>
- <https://m.blog.naver.com/PostView.nhn?blogId=os2dr&logNo=221565409684&proxyReferer=https%3A%2Fwww.google.com%2F>
- [http://horace.io/pytorch-vs-tensorflow/?fbclid=IwAR0fC_bML92TWMHUiVAWeYrIPYfc4PFwHsGmz6ibfxDZ9zWIT1v3qx7zhCs\]\(http://horace.io/pytorch-vs-tensorflow/?fbclid=IwAR0fC_bML92TWMHUiVAWeYrIPYfc4PFwHsGmz6ibfxDZ9zWIT1v3qx7zhCs\)](http://horace.io/pytorch-vs-tensorflow/?fbclid=IwAR0fC_bML92TWMHUiVAWeYrIPYfc4PFwHsGmz6ibfxDZ9zWIT1v3qx7zhCs](http://horace.io/pytorch-vs-tensorflow/?fbclid=IwAR0fC_bML92TWMHUiVAWeYrIPYfc4PFwHsGmz6ibfxDZ9zWIT1v3qx7zhCs)
- [https://github.com/aymericdamien\]\(https://github.com/aymericdamien](https://github.com/aymericdamien](https://github.com/aymericdamien)
- <https://somjang.tistory.com/entry/TF20-MNIST-ValueError-Shapes-32-10-and-32-1-are-incompatible-%ED%95%B4%EA%B2%BO-%EB%B0%A9%EB%B2%95>
- <https://rfriend.tistory.com/m/556?category=711317>
- http://www.smartdesignlab.org/DL/AutoEncoder_Tensor2_0.html
- <https://eclipse360.tistory.com/40>
- https://www.youtube.com/watch?v=D_ws0YyHAM8&list=PLQ28Nx3M4Jrguyuwg4xe9d9t2XE639e5C&index=6
- <https://hwiyoung.tistory.com/335>
- <https://www.tensorflow.org/guide/migrate?hl=ko>
- <https://bskyvision.com/822>
- https://tykimos.github.io/2017/01/27/MLP_Layer_Talk
- <https://green-late7.tistory.com/48>

Q & A

들어주셔서 감사합니다.