2026.08.18 Workshop

AI 기반 식물표본 이미지 동정

AI based identification of Herbarium images

성신여자대학교 식물분자계통학실 문현지

중앙대학교 AutoML 연구실 한수정

실습 과정

1

딥러닝 분석 환경 구축

2

데이터 준비 및 전처리

- 이미지 크기 줄이기
- 비식물 표본 구성 요소 제거하기-HerbPAI 활용

3

식물표본 이미지 AI 분류 분석

- 데이터 분할
- ResNet-18 기반 분류 모델 구축
- 5-fold 교차검증
- 최종 모델 생성 및 성능 평가

4

결과 해석 및 모델 활용

- 분류 성능 확인
- Confusion matrix
- Grad-CAM
- Gradio를 이용한 분류 모델 활용

1 딥러닝 분석 환경 구축

1. 딥러닝 분석 환경 구축

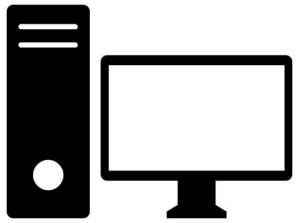
- **Conda (25.5.1)**
  : 딥러닝에 필요한 Python과 여러 라이브러리를 설치하고 관리하는 도구
- **Python (3.9.21)**
  : 딥러닝 코드를 작성하고 실행하는 프로그래밍 언어
- **PyTorch (2.3.1)**
  : 딥러닝 모델을 만들고 학습하는 데 사용하는 핵심 라이브러리
- **torchvision (0.18.1)**
  : 이미지 처리와 ResNet 등의 이미지 분류 모델을 제공하는 PyTorch용 라이브러리
- **CUDA (11.8)**
  : NVIDIA GPU를 이용하여 딥러닝 연산을 빠르게 수행할 수 있게 하는 환경
- **JupyterLab (4.4.1)**
  : 웹 브라우저에서 Python 코드를 작성·실행하고 결과를 확인하는 작업 공간

* 라이브러리(library): 특정 기능을 쉽게 사용할 수 있도록 미리 만들어 놓은 코드 모음

2. 서버 접속 및 개인 가상환경 생성

GPU 서버 원격 접속

<개인 PC>



SSH로 원격접속

*SSH: 개인 PC에서 **원격 서버에 안전하게 접속**하기 위한 방식

<Viola 서버>

NVIDIA RTX A6000 × 2
각 GPU당 VRAM 48 GB

*VRAM: GPU가 딥러닝 연산에 사용하는 전용 메모리

가상환경

프로젝트별로 Python과 라이브러리(분석에 필요한 프로그램) 버전을 **독립적으로 관리하는 작업 환경**

Q1) 왜 가상환경을 사용할까요?

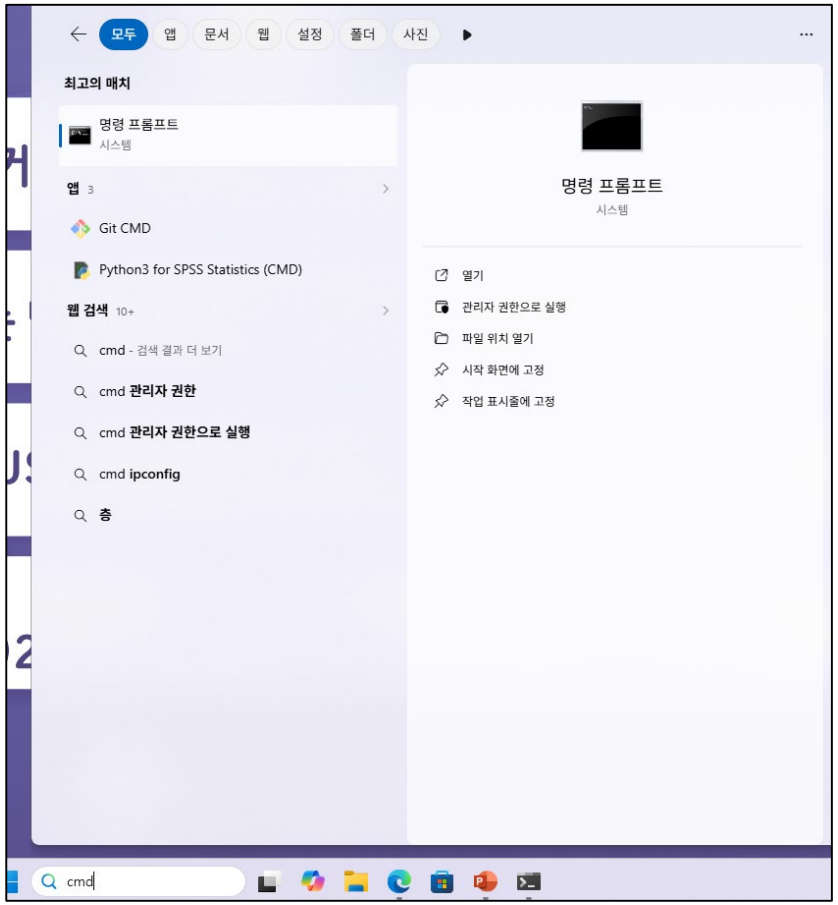
프로젝트마다 필요한 프로그램의 버전이 다를 수 있기 때문에,
서로 영향을 주지 않고 안정적으로 분석하기 위해 사용합니다.

2. 서버 접속 및 개인 가상환경 생성

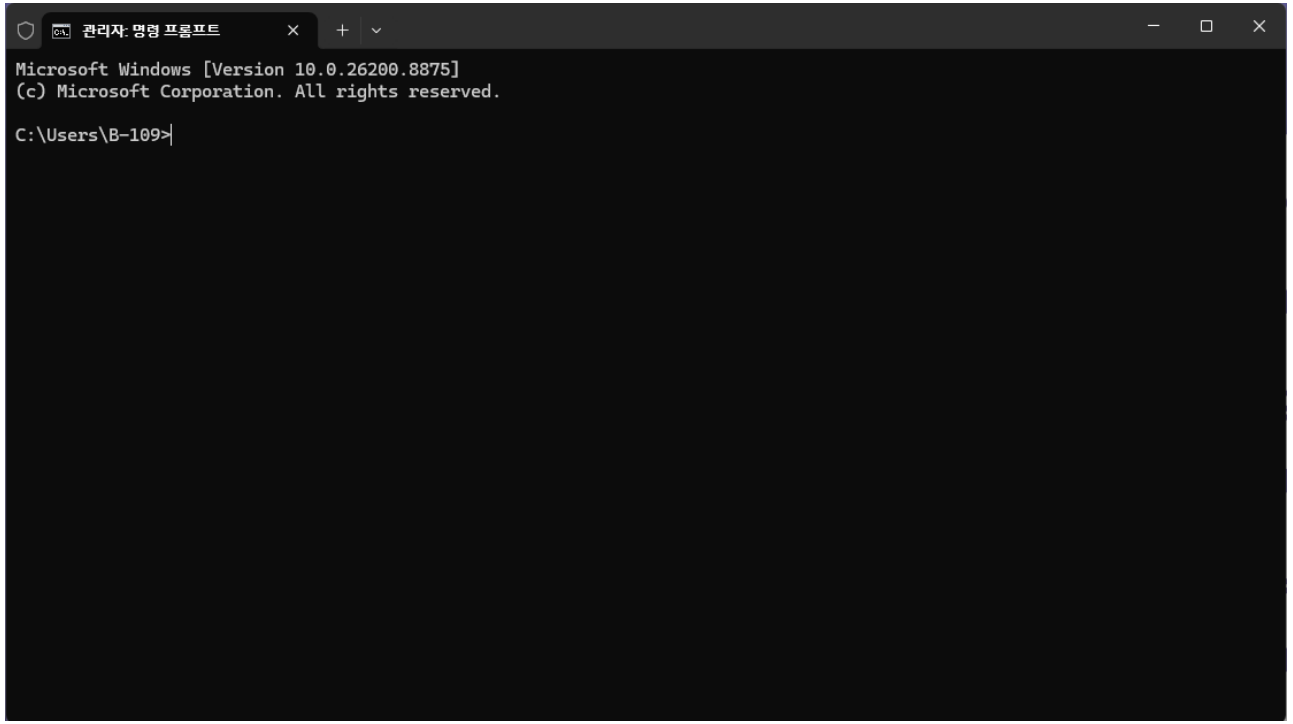
***터미널이란?**
마우스 대신 **명령어를 직접 입력하여 컴퓨터를 조작하는 창**

① GPU 서버 원격 접속

윈도우 검색창에서 **cmd**를 검색하여 '**명령 프롬프트**'를 실행합니다.



명령 프롬프트(cmd) 창



2. 서버 접속 및 개인 가상환경 생성

① GPU 서버 원격 접속

1) 표에서 본인에게 배정된 ID를 확인하여 서버에 접속합니다..

Ssh 아이디@113.198.1.215

- ssh : 원격 서버에 안전하게 접속
- 아이디 : 본인에게 배정된 ID
- 113.198.1.215 : GPU 서버 주소

```
C:\Users\B-109>ssh team20@113.198.1.215
```

2) 비밀번호를 입력합니다.

※ 비밀번호 입력 시
화면에는 아무것도 표시되지 않지만 정상적으로 입력되고 있습니다.

Passwod: workshop

3) 아래와 같이 표시되면 서버 접속 완료!

```
team20@Viola:~$
```

*의미:
team20이라는 사용자가 Viola 서버에 접속했고,
현재 홈 폴더에서 명령어 입력을 기다리고 있다

- team20 → 현재 로그인한 **사용자 ID**
- @Viola → 접속한 **서버 이름**
- ~ → 현재 위치가 사용자의 **홈 폴더**라는 뜻
- $ → 이제 명령어를 입력할 수 있다는 표시

조장	ID	ID	설치 완료
곽도현	team1	team17	완료
송민재	team2	team18	완료
박준규	team3	team19	완료
조민준	team4	team20	완료
신수진	team5	team21	완료
심선희	team6	team22	완료
홍호준	team7	team23	완료
이재경	team8	team24	완료
박종수	team9	team25	
장영종	team10	team26	
김명숙	team11	team27	
황건하	team12	team28	
정서현	team13	team29	
김수장	team14	team30	
최인수	team15		
김승휘	team16		

2. 서버 접속 및 개인 가상환경 생성

② Miniconda 설치

사용자 홈 폴더로 이동
- cd = 폴더 이동
- ~ = 내 개인 홈 폴더

```
cd ~
```

Miniconda 설치 파일 다운로드

```
wget https://repo.anaconda.com/miniconda/Miniconda3-py39_25.5.1-0-Linux-x86_64.sh
```

- wget 주소 : **인터넷에서 파일 다운로드**

```
team20@Viola:~$ wget https://repo.anaconda.com/miniconda/Miniconda3-py39_25.5.1-0-Linux-x86_64.sh
```

다운로드한 Miniconda 설치 파일 실행

```
bash Miniconda3-py39_25.5.1-0-Linux-x86_64.sh
```

- bash: Linux 서버에서 명령어를 실행해 주는 프로그램

```
team20@Viola:~$ bash Miniconda3-py39_25.5.1-0-Linux-x86_64.sh
```

※※ 설치 중 입력하세요 ※※

- ENTER → 라이선스 확인
- yes → 라이선스 동의
- 설치 경로 → 기본값 사용 시 ENTER
- conda init 질문 → yes

변경된 터미널 설정을 현재 터미널에 적용

```
source ~/.bashrc
```

- source : **설정 파일의 내용을 현재 터미널에 바로 적용**
- ~ : **내 홈 폴더**
- .bashrc : Bash 터미널의 **환경 설정 파일**

Conda가 정상적으로 설치되었는지 버전 확인

```
conda --version
```

- conda : **Conda 프로그램 실행**
- --version : 프로그램의 **설치된 버전 확인**

```
team20@Viola:~$ source ~/.bashrc
(base) team20@Viola:~$ conda --version
conda 25.5.1
```

2. 서버 접속 및 개인 가상환경 생성

③ 워크숍용 가상환경 생성

1. Anaconda 기본 채널의 이용약관 동의

conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/main

2. Anaconda R 채널의 이용약관 동의

conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/r

- conda tos accept : Anaconda 채널의 **이용약관(ToS)에 동의**
- --channel : 이용할 **패키지 저장소(채널)를 지정**
- pkgs/main : Python 패키지가 제공되는 **Anaconda 기본 채널**
- pkgs/r : R 관련 패키지가 제공되는 **Anaconda R 채널**

* **채널(Channel)** = 프로그램이나 라이브러리를 내려받는 **온라인 저장소**

3. 워크숍용 독립적인 Python 환경 생성 (Python 3.9.21)

conda create -n herbpai python=3.10 -y

#혹시 에러가 발생하면 "-y"만 터미널에서 직접 입력해주세요

- conda create : **새로운 가상환경 생성**
- -n workshop : 가상환경 이름을 **workshop**으로 지정
- python=3.9.21 : Python **3.9.21 버전 설치**
- -y : 설치 중 확인 질문에 자동으로 **Yes**

```
(base) team20@Viola:~$ conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/main
accepted Terms of Service for https://repo.anaconda.com/pkgs/main
(base) team20@Viola:~$ conda tos accept --override-channels --channel https://repo.anaconda.com/pkgs/r
accepted Terms of Service for https://repo.anaconda.com/pkgs/r
(base) team20@Viola:~$ conda create -n workshop python=3.9.21 -y
```

2. 서버 접속 및 개인 가상환경 생성

③ 워크숍용 가상환경 생성

4. 가상환경 활성화
conda activate workshop

- conda activate : 지정한 **가상환경으로 전환**
- workshop : 사용할 가상환경 이름

```
(base) team20@Viola:~$ conda activate workshop
(workshop) team20@Viola:~$
```

활성화되면 터미널 앞부분이 바뀝니다.

#Python 버전 확인:
python --version

- python : 현재 환경의 Python 실행
- --version : 설치된 **Python 버전 확인**

```
(workshop) team20@Viola:~$ python --version
Python 3.9.21
```

3. 딥러닝 패키지 설치

① PyTorch + torchvision + CUDA 11.8

pip install torch==2.3.1 torchvision==0.18.1 --index-url https://download.pytorch.org/whl/cu118

```
(workshop) team20@Viola:~$ pip install torch==2.3.1 torchvision==0.18.1 \
--index-url https://download.pytorch.org/whl/cu118
Looking in indexes: https://download.pytorch.org/whl/cu118
Collecting torch==2.3.1
  Downloading https://download-r2.pytorch.org/whl/cu118/torch-2.3.1%2Bcu118-cp39-cp39-linux_x86_64.whl (839.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 839.4/839.6 MB 100.3 MB/s eta 0:00:01
```

② 분석에 필요한 기본 패키지

pip install numpy pandas matplotlib pillow scikit-learn

```
(workshop) team20@Viola:~$ pip install numpy pandas matplotlib pillow scikit-learn
Requirement already satisfied: numpy in ./miniconda3/envs/workshop/lib/python3.9/site-packages (2.0.2)
Collecting pandas
  Downloading pandas-2.3.3-cp39-cp39-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (91 kB)
Collecting matplotlib
  Downloading matplotlib-3.9.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Requirement already satisfied: pillow in ./miniconda3/envs/workshop/lib/python3.9/site-packages (11.3.0)
Collecting scikit-learn
  Downloading scikit_learn-1.6.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Collecting python-dateutil>=2.8.2 (from pandas)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
```

주요 용도:

- **NumPy** → 수치 계산
- **Pandas** → 분석 결과 정리
- **Matplotlib** → 학습 결과 그래프
- **Pillow** → 이미지 읽기 및 처리
- **Scikit-learn** → 성능평가, confusion matrix 등

Q. AI는 이미지를 어떻게 이해할까?

한 픽셀은 R, G, B의
세 값으로 표현됩니다

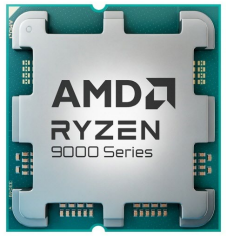
(128, 72, 137)

AI 모델은 이미지를 픽셀의 RGB 값으로 이루어진 숫자 배열로 입력 받습니다.

Q. 딥러닝 연구에서는 왜 GPU 성능이 중요할까?

CPU vs. GPU

- **CPU (Central Processing Unit)**
  다양한 작업을 처리하는 **범용 연산 장치**



- **GPU (Graphics Processing Unit)**
  많은 계산을 **병렬로 동시에 처리**
  → 딥러닝의 대규모 행렬·텐서 계산에 적합

이미지 수 ↑ · 해상도 ↑ · 모델 크기 ↑

↓

계산량 ↑ · 학습시간 ↑

해상도 ↑ · Batch size ↑ · 모델 크기 ↑ → GPU 메모리 사용량 ↑

↓

고성능 GPU

- 높은 연산 성능 → 학습 시간 단축
- 큰 GPU 메모리 → 더 큰 이미지·Batch·모델 처리 가능

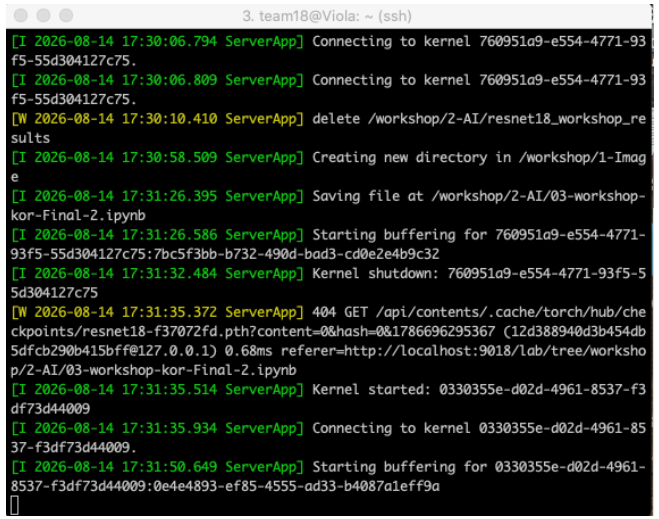
4. JupyterLab을 이용한 GPU 서버 원격 분석

Q) 내 PC에서 GPU 서버의 JupyterLab에 어떻게 접속할까요?

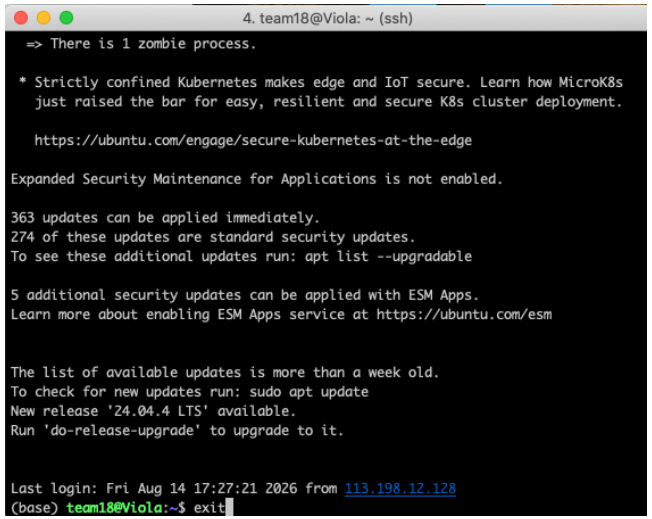
JupyterLab과 딥러닝 계산은 **GPU 서버에서 실행**되며,
우리는 **내 PC의 웹 브라우저에서 JupyterLab을 사용**합니다.

JupyterLab을 인터넷에 직접 공개하지 않고,
SSH 터널을 이용하여 내 PC와 GPU 서버를 안전하게 연결합니다.

서버에서 JupyterLab 실행

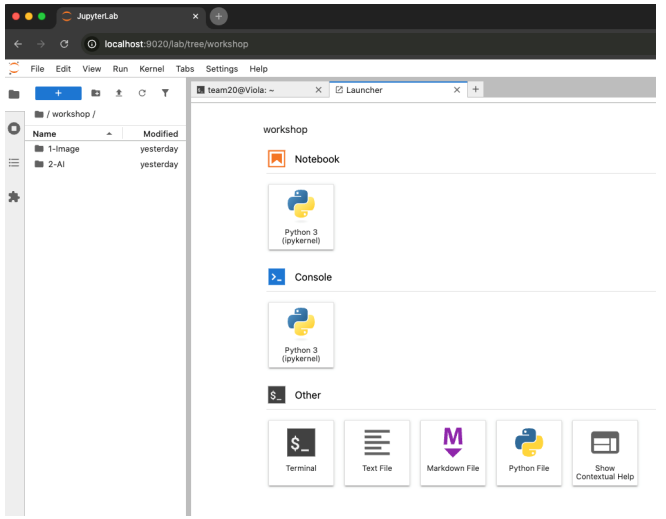


SSH로 안전한 통로 생성



SSH 터널

웹 브라우저에서 JupyterLab 접속



4. JupyterLab을 이용한 GPU 서버 원격 분석

① Jupyter 환경 설치

```
pip install ₩
  jupyter==1.1.1 ₩
  jupyterlab==4.4.1 ₩
  notebook==7.4.1 ₩
  ipykernel==6.29.5
```

- jupyter : Jupyter 기본 기능
- jupyterlab : 웹 기반 분석 환경
- notebook : Jupyter Notebook 실행 기능
- ipykernel : Python 코드를 실행하는 커널
- ==버전 : 설치할 패키지의 정확한 버전을 지정

```
(workshop) team20@Viola:~$ pip install \
  jupyter==1.1.1 \
  jupyterlab==4.4.1 \
  notebook==7.4.1 \
  ipykernel==6.29.5
Collecting jupyter==1.1.1
  Downloading jupyter-1.1.1-py2.py3-none-any.whl.metadata (2.0 kB)
Collecting jupyterlab==4.4.1
  Downloading jupyterlab-4.4.1-py3-none-any.whl.metadata (16 kB)
Collecting notebook==7.4.1
  Downloading notebook-7.4.1-py3-none-any.whl.metadata (10 kB)
```

4. JupyterLab을 이용한 GPU 서버 원격 분석

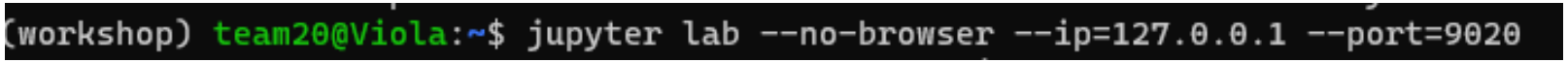
******: Port 번호**

② 서버에서 JupyterLab 실행

#SSH로 접속한 서버 명령어 창에서 JupyterLab을 실행합니다.

jupyter lab --no-browser --ip=127.0.0.1 --**port=******

- jupyter lab → JupyterLab 실행
- --no-browser → 서버에서 브라우저를 열지 않음
- --ip=127.0.0.1 → 서버 내부에서만 접속하도록 설정
- --port=**** → 사용할 포트 번호 지정



jupyter lab --no-browser --ip=127.0.0.1 --**port=9024**

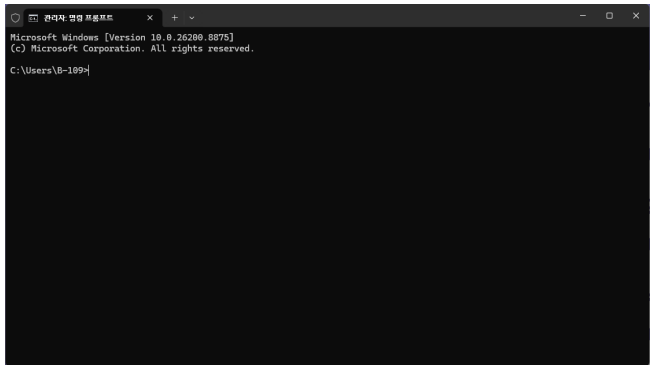
조장	ID	Port	ID	Port	설치 완료
곽도현	team1	9001	team17	9017	완료
송민재	team2	9002	team18	9018	완료
박준규	team3	9003	team19	9019	완료
조민준	team4	9004	team20	9020	완료
신수진	team5	9005	team21	9021	완료
심선희	team6	9006	team22	9022	완료
홍호준	team7	9007	team23	9023	완료
이재경	team8	9008	team24	9024	완료
박종수	team9	9009	team25	9025	
장영종	team10	9010	team26	9026	
김명숙	team11	9011	team27	9027	
황건하	team12	9012	team28	9028	
정서현	team13	9013	team29	9029	
김수장	team14	9014	team30	9030	
최인수	team15	9015			
김승휘	team16	9016			

4. JupyterLab을 이용한 GPU 서버 원격 분석

******: Port 번호**

③ 내 PC에서 SSH 터널 연결

1) 내 PC에서 새 터미널(cmd) 창을 열어주세요



※ 새 터미널 창을 열어주세요!

2) 서버와 안전하게 연결되는 통로를 만듭니다

ssh -L ****:127.0.0.1:**** 아이디@113.198.1.215

- ssh → 서버에 안전하게 연결
- -L → 내 PC의 포트와 서버의 포트를 연결
- 첫 번째 9021 → **내 PC에서 사용할 포트**
- 127.0.0.1:9021 → **서버에서 실행 중인 JupyterLab의 위치**
- 아이디@서버주소 → 접속할 서버

```
C:\Users\Sangtae Kim>ssh -L 9020:127.0.0.1:9020 team20@113.198.1.215
```

터미널 창 두개는 계속 그대로 유지해주세요.

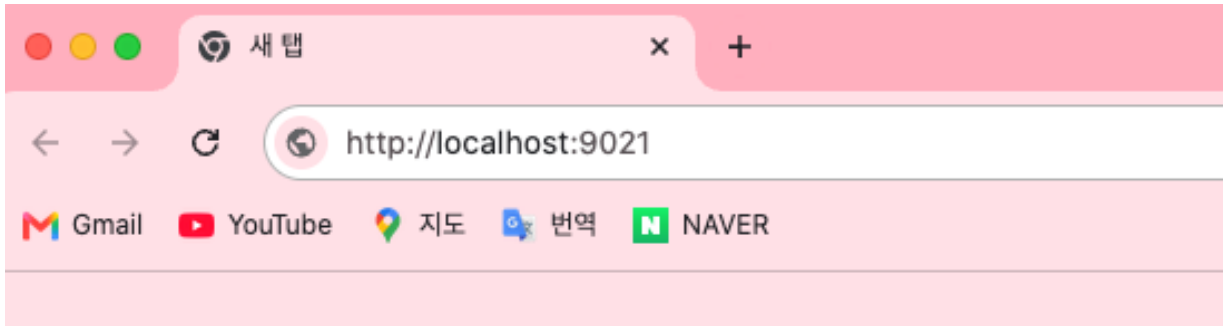
조장	ID	Port	ID	Port	설치 완료
곽도현	team1	9001	team17	9017	완료
송민재	team2	9002	team18	9018	완료
박준규	team3	9003	team19	9019	완료
조민준	team4	9004	team20	9020	완료
신수진	team5	9005	team21	9021	완료
심선희	team6	9006	team22	9022	완료
홍호준	team7	9007	team23	9023	완료
이재경	team8	9008	team24	9024	완료
박종수	team9	9009	team25	9025	
장영종	team10	9010	team26	9026	
김명숙	team11	9011	team27	9027	
황건하	team12	9012	team28	9028	
정서현	team13	9013	team29	9029	
김수장	team14	9014	team30	9030	
최인수	team15	9015			
김승휘	team16	9016			

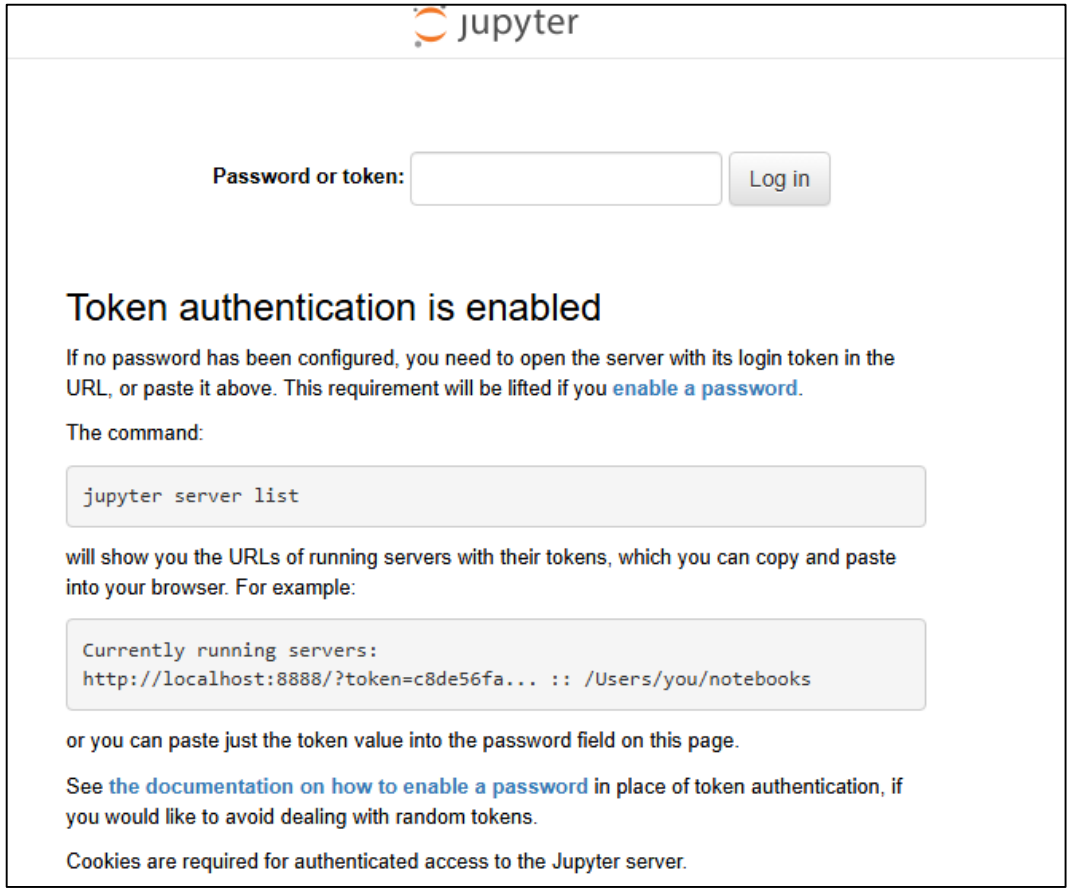
4. JupyterLab을 이용한 GPU 서버 원격 분석

④ 웹 브라우저에서 접속

#Chrome 등의 웹 브라우저 주소창에 아래 주소를 입력합니다.

http://localhost:****





JupyterLab 로그인 창
(Token 입력 화면)

4. JupyterLab을 이용한 GPU 서버 원격 분석

⑤ JupyterLab Token 확인 및 입력

1)두 번째 창에서 workshop 가상환경 활성화
#③에서 **SSH 터널 연결에 사용한 두 번째 터미널 창**에서 아래 명령어를 입력합니다.

conda activate workshop

- conda activate : **가상환경을 활성화**
- workshop : 사용할 **가상환경 이름**

2) JupyterLab Token 확인#
#현재 실행 중인 JupyterLab의 접속 정보를 확인하기 위해 아래 명령어를 입력합니다.

jupyter server list

- jupyter : Jupyter 관련 명령 실행
- server list : 현재 실행 중인 **Jupyter 서버 목록 확인**

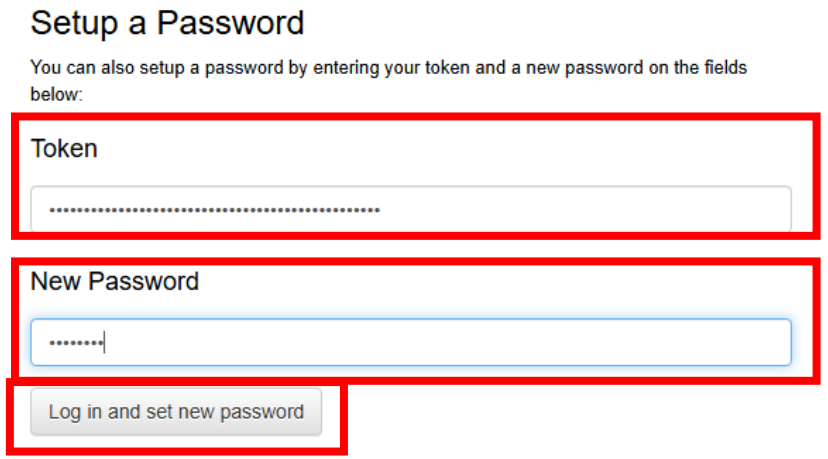
#출력된 주소에서 **token= 뒤에 있는 문자열을 복사**합니다.

```
(base) team20@Viola:~$ conda activate workshop
(workshop) team20@Viola:~$ jupyter server list
Currently running servers:
http://127.0.0.1:9020/?token=bc8f4dbdbd4f815d37db1687ae7ce70c94605483d3f5e25a :: /home/team20
```

Token 정보

3) Token 입력 및 비밀번호 설정
#JupyterLab 로그인 화면에서

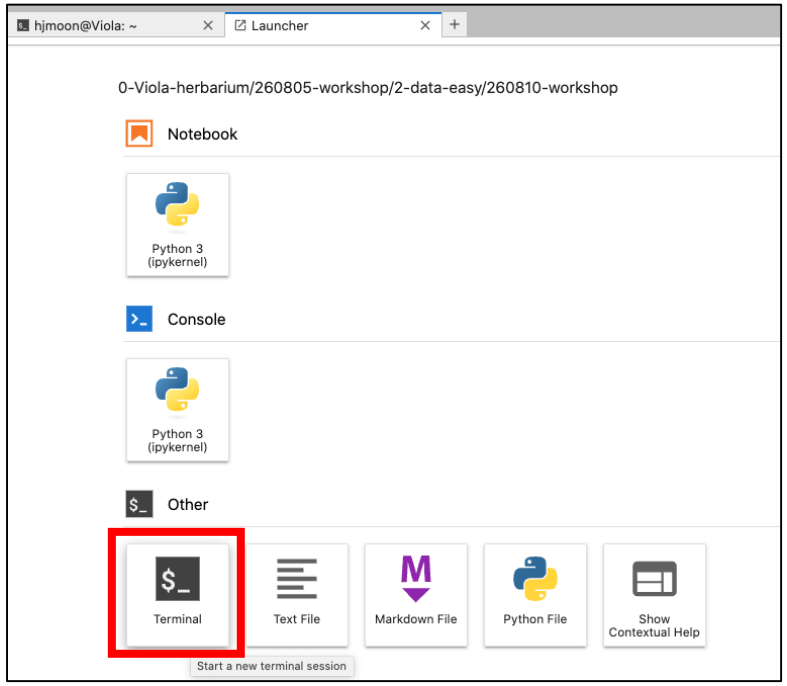
- **Token**: 복사한 Token 정보 입력
- **New Password**: Workshop 입력



5. Gradio 설치

학습한 AI 모델을 간단한 **웹 인터페이스**로 실행

 # 1. JupyterLab의 Launcher에서 Terminal을 엽니다.



 # 2. 앞서 만든 'workshop' 가상환경으로 전환

conda activate workshop

 # 3. 웹 인터페이스 제작에 사용할 Gradio 설치

pip install gradio==3.50.2

- pip install : Python 패키지 설치
- gradio : 간단한 웹 인터페이스를 만들 수 있는 라이브러리
- ==3.50.2 : Gradio의 **정확한 버전 지정**

 # 4. 설치된 Gradio의 버전 및 정보 확인

pip show gradio

- pip show : 설치된 패키지의 정보 확인
- gradio : 확인할 패키지 이름

```
(base) team20@Viola:~$ conda activate workshop
(workshop) team20@Viola:~$ pip install gradio==3.50.2
```

```
(workshop) team20@Viola:~$ pip show gradio
Name: gradio
Version: 3.50.2
```

2 데이터 준비 및 전처리

Q. AI는 이미지에서 무엇을 학습할까?

① 우리가 기대하는 특징

- 잎의 형태
- 꽃·열매
- 줄기
- 전체적인 식물체 형태(habit)

→ **종 구분에 유용할 수 있는 생물학적 특징**

② 의도하지 않은 단서도 학습할 수 있음

- 라벨 · 바코드
- 표본 대지 배경
- scale bar
- 촬영 방식
- 특정 기관에서 반복되는 표본 배치

→ **종과 우연히 함께 나타나는 패턴도 분류에 이용할 수 있음**

→ **Shortcut learning:** 생물학적 특징보다 정답과 쉽게 연결되는 단서를 이용하는 현상

AI는 종을 구분하는 데 도움이 되는 반복적인 시각 패턴을 학습합니다

워크숍에서 사용할 데이터 구성

① 모델 개발·평가용 데이터셋

→ GPU **서버**에 업로드

```
Scutellaria/
├──── 01_Scu_ind/            (50 images)
│     ├──── S_ind_01.jpg
│     ├──── ...
│     └──── S_ind_50.jpg
│
├──── 02_Scu_pek_tra/        (50 images)
├──── 03                     (50 images)
├──── 04                     (50 images)
└──── 05_Scu_pek_uss/        (50 images)
```

5종 × 50장 = 총 250장

- **용도:** 모델 학습 · 교차검정 · 최종 테스트

② Gradio 웹앱 시연용 이미지

→ 개인 PC **바탕화면**에 저장

Scutellaria_predict

- **용도:** 학습된 모델에 새로운 이미지를 입력하여 **종 예측 결과 확인**

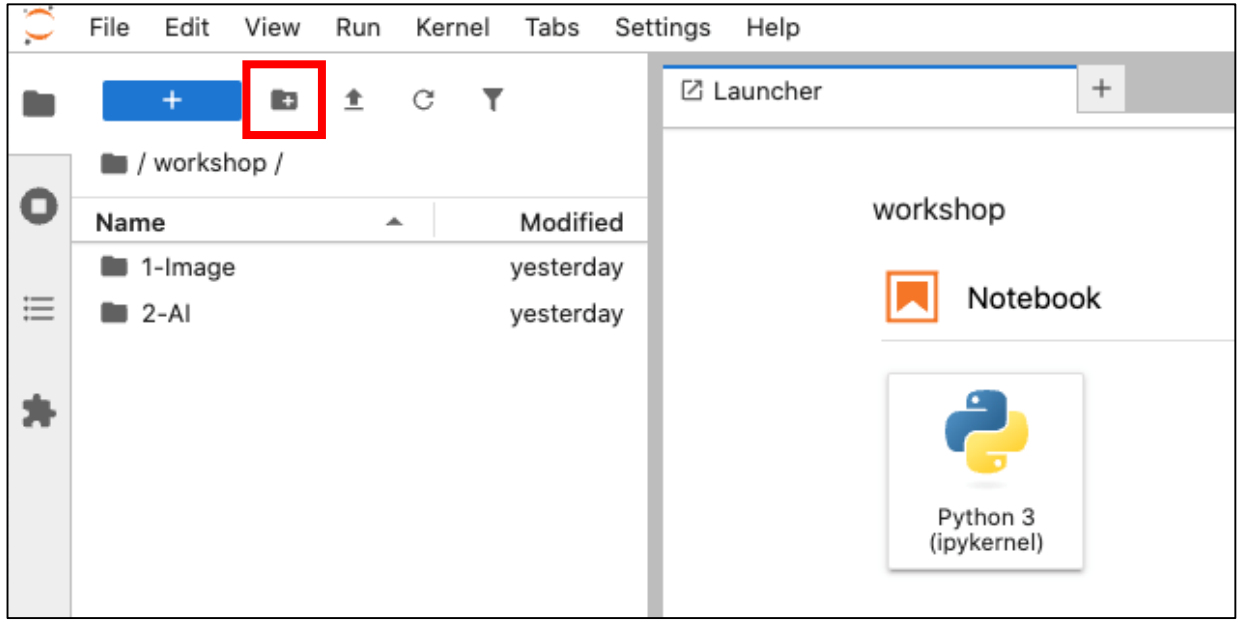
0. 워크숍 폴더 구조 만들기

/home/아이디/

- Workshop
  - 1-Image
    - 01_Image_resize.ipynb
    - 1-data
    - 2-resize
    - 3-herbpai
  - 2-AI
    - 02_AI_classification-kr.ipynb
    - 03_Gradio_webapp.py

1) JupyterLab에서 폴더 생성

폴더 생성 버튼

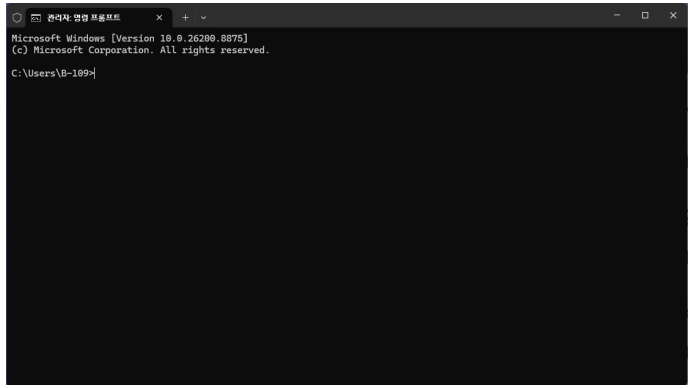


※ 폴더명은 **대·소문자를 정확히 입력해주세요**

2) 워크숍 파일 옮기기

1. 내 컴퓨터에서 서버로 데이터셋 옮기기

1) 새 명령 프롬프트(cmd) 창을 열어주세요.



2) 표본 데이터셋을 내 PC에서 서버로 옮기기

각자 준비해 오신 표본 데이터를 서버로 전송해주세요.

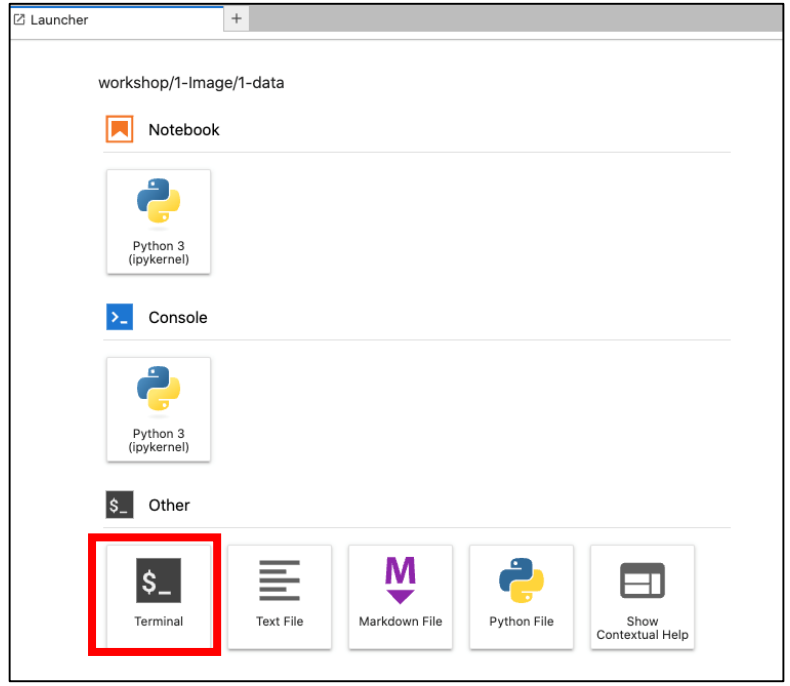
scp "C:₩Users₩B-109₩Desktop₩workshop₩Viola.zip" 아이디@113.198.1.215:~/Workshop/1-Image/1-data

- scp : Secure Copy의 약자. **내 PC의 파일을 원격 서버로 안전하게 전송하는 명령어**
- "C:₩...₩Viola.zip" : **내 PC의 파일 위치**
- 아이디@113.198.1.215 : **접속할 서버**
- ~/workshop/1-Image/Input : **서버에 저장할 위치**

```
C:\Users\B-109>scp "C:\Users\B-109\Desktop\workshop\00-Dataset-Viola.zip" team18@113.198.1.215:~/workshop/1-Image/ 1-data
team18@113.198.1.215's password:
00-Dataset-Viola.zip                                                    75% 1492MB 105.0MB/s   00:04 ET00-Dataset-Viola.zip
                                                        80% 1593MB 104.6MB/00-Dataset-Viola.zip
                                85% 1700MB 104.7MB/s   00:0200-Dataset-Viol00-Dataset-Viol00-Dataset-Viola.zip
                                     100% 1983MB 105.4MB/s   00:18
```

2. 서버에서 데이터셋 압축 풀기

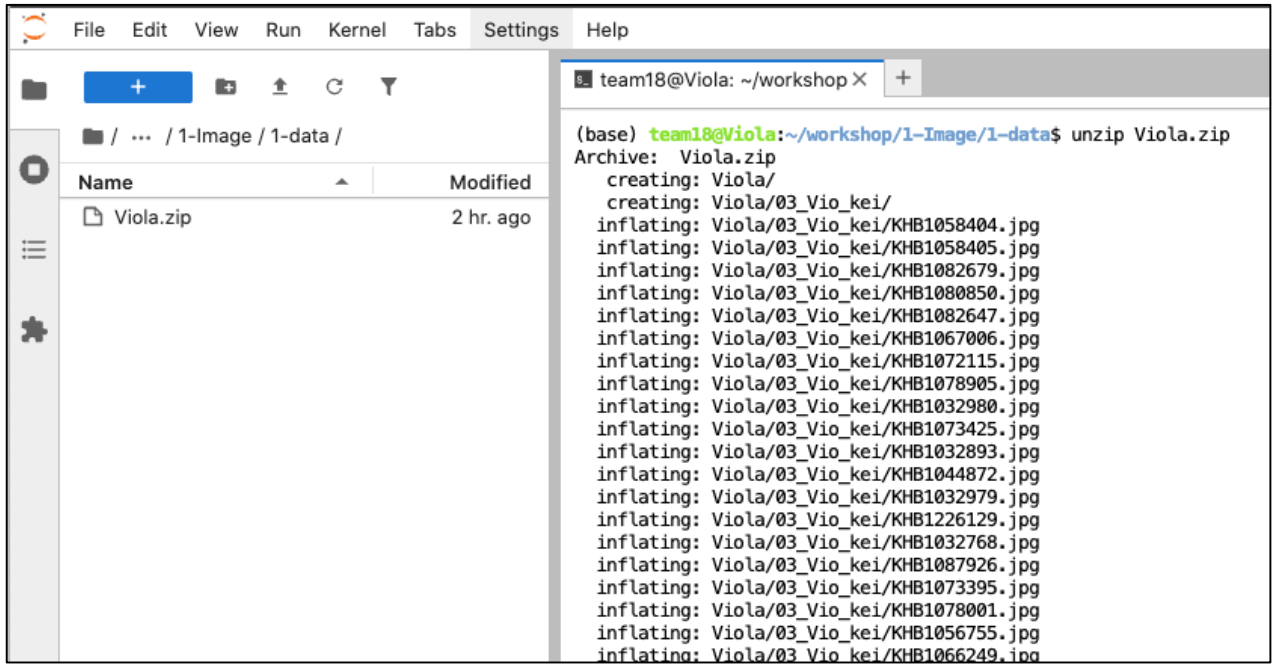
1) JupyterLab에서 Workshop/1-Image/1-data/ 폴더로 이동한 후
새 Terminal을 열어주세요



2) 데이터셋(.zip) 파일의 압축을 풀어주세요
unzip ***.zip

- unzip : **.zip 압축 파일의 압축을 해제하는 명령어**

* 파일명의 앞부분을 입력한 후 Tab 키
를 누르면 파일명이 자동 완성됩니다.
예) V + Tab → Viola.zip



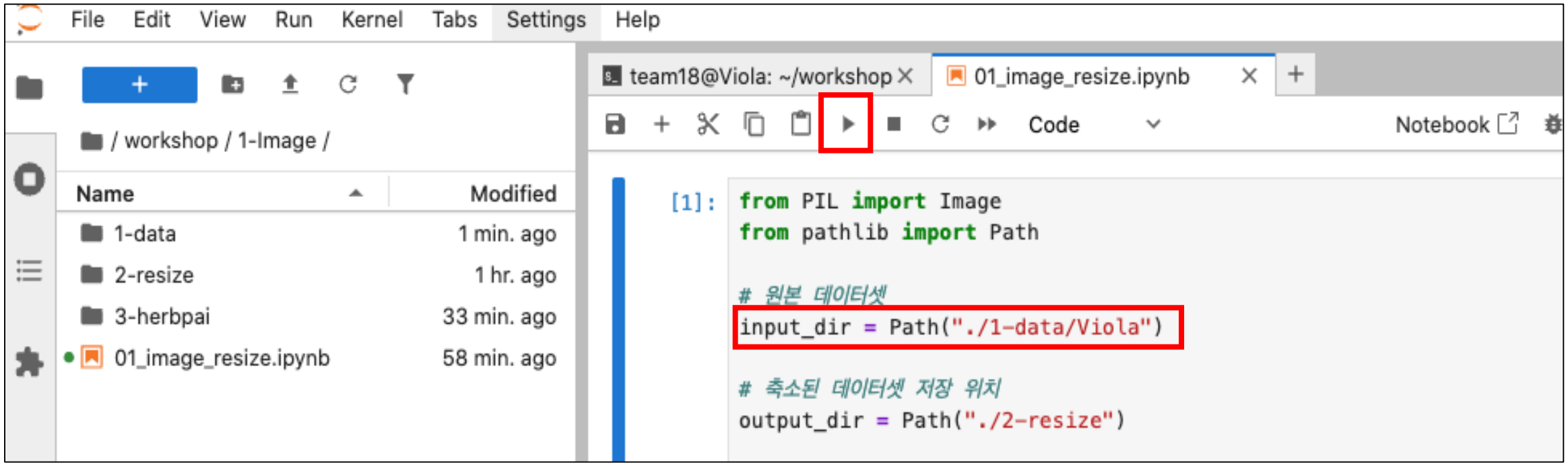
3. 이미지 크기 조정(Resize)

1) Workshop/1-Image/01_image_resize.ipynb 더블클릭하여 여세요

2) 입력 데이터셋 경로를 확인하세요

3) ▶ 실행 버튼을 눌러 이미지 크기 조정 코드를 실행하세요

4) Workshop/1-Image/2-resize 폴더에서 결과를 확인하세요



4. HerbPAI로 데이터 전처리하기

웹사이트

https://huggingface.co/spaces/SjSj1008/HerbPAI

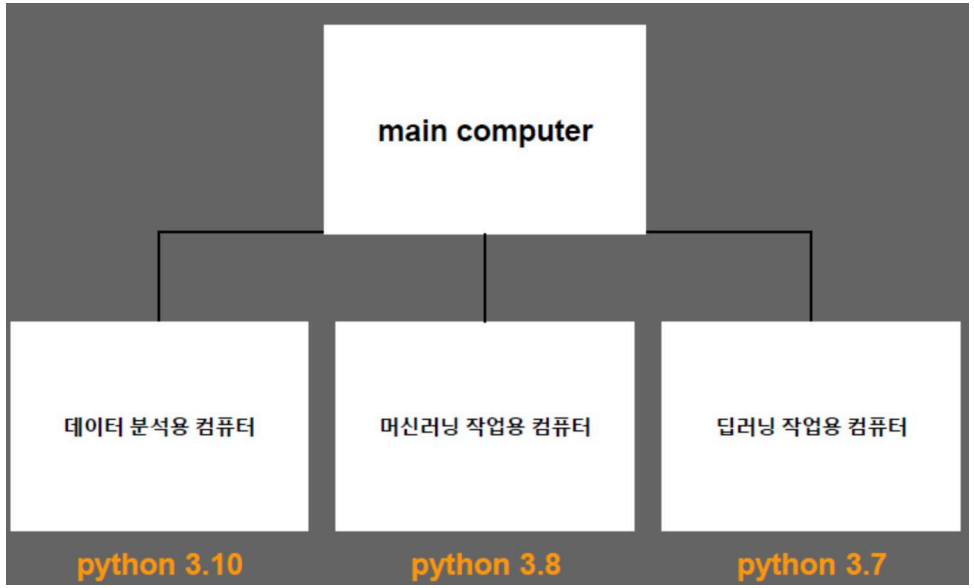
GitHub

https://github.com/Sujeong-Han/HerbPAI

4. HerbPAI로 데이터 전처리하기

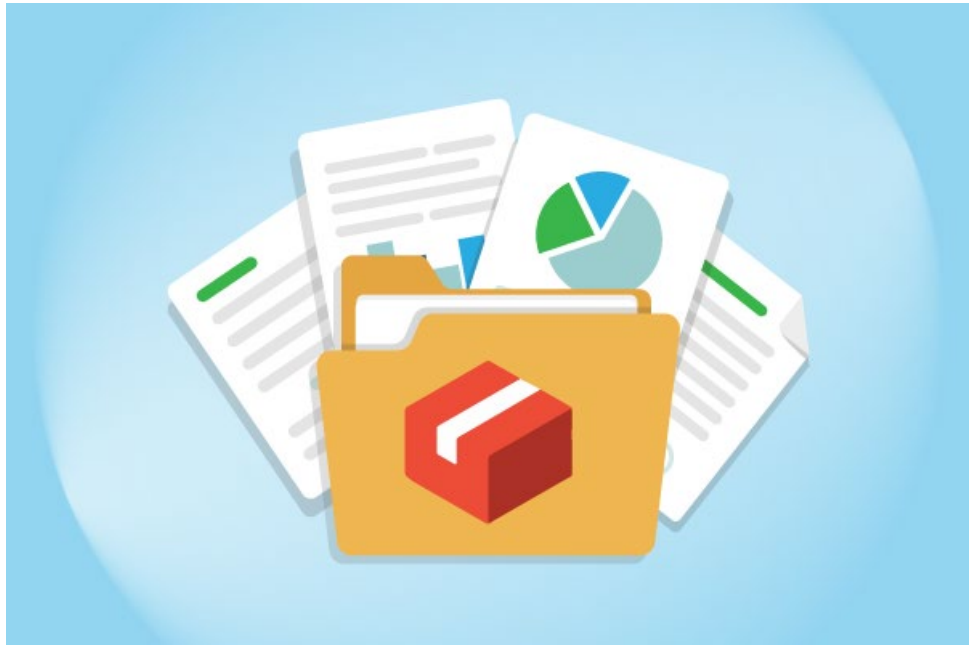
0) 서버 원격 접속

ssh 아이디@113.198.1.215



1) conda 환경 생성

conda create -n herbpai python=3.10 -y
conda activate herbpai



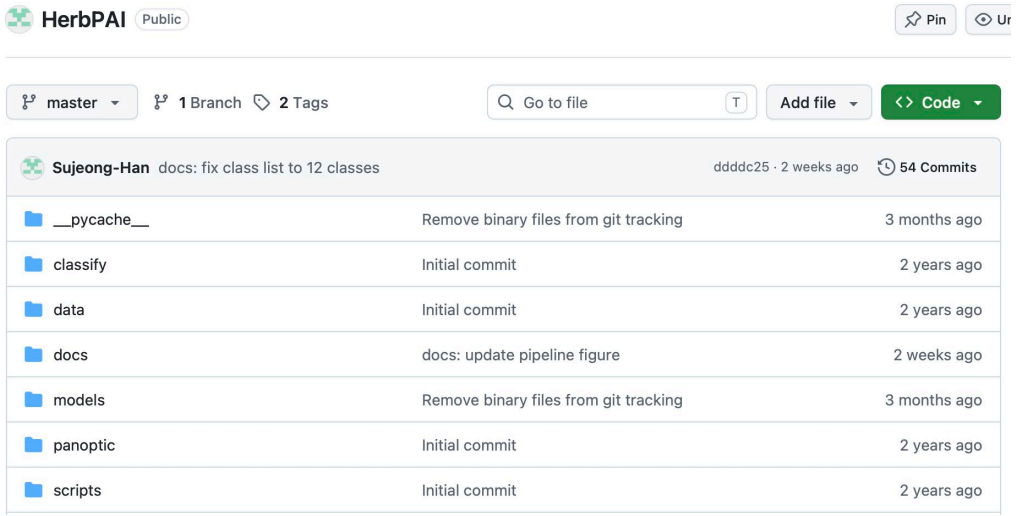
2) Git lfs 설치

conda install -c conda-forge git-lfs -y
git lfs install
git lfs version

4. HerbPAI로 데이터 전처리하기

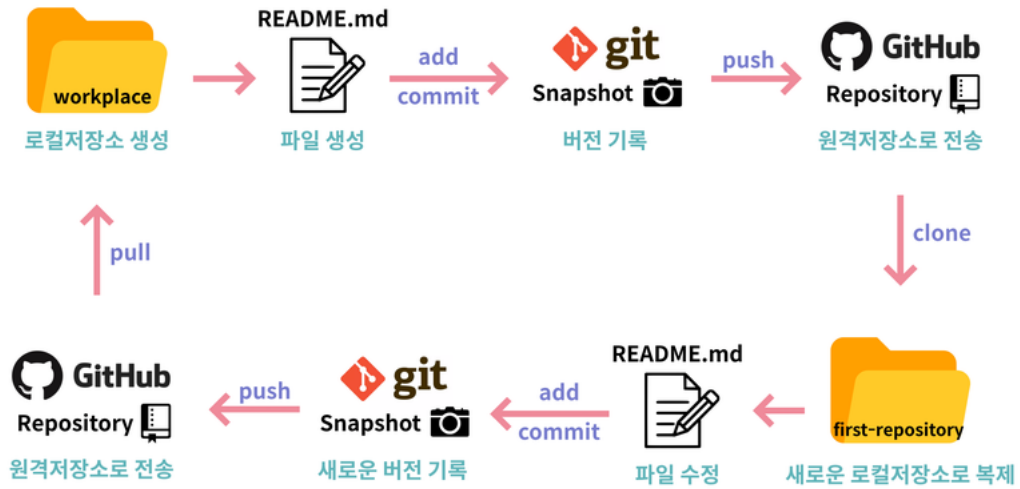
3) Github 코드 가져오기

```
cd ~
git clone https://github.com/Sujeong-Han/HerbPAI.git
cd HerbPAI
```



GitHub

코드를 저장하고, 수정 사항을 기록하며, 서로 협업할 수 있게 도와주는 세계 최대 규모의 웹 기반 온라인 플랫폼

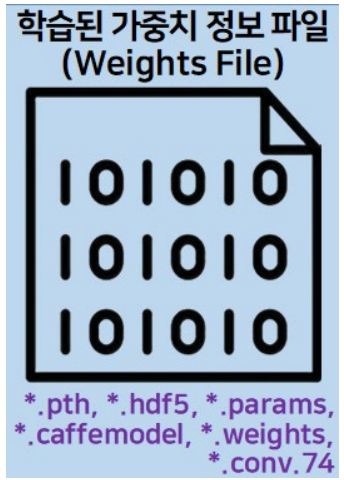


4. HerbPAI로 데이터 전처리하기

4) 파일 검증

ls -lh best.pt

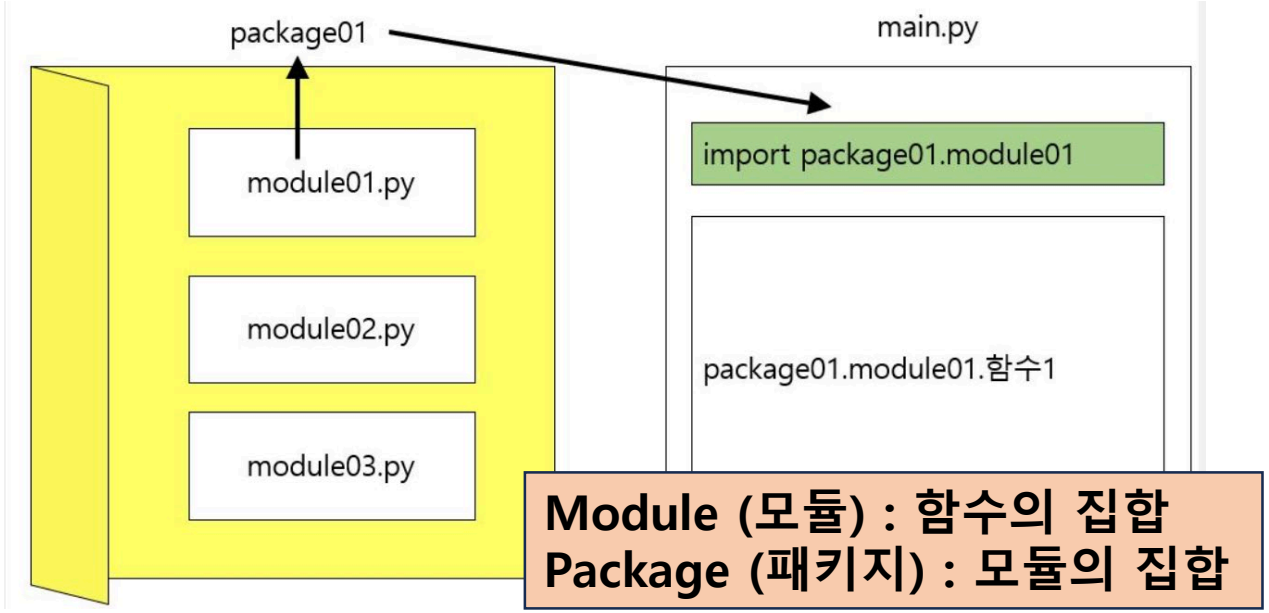
133mb ~ 134mb 가 나와야 정상



133mb ~ 134mb

5) 패키지 설치

pip install -r requirements.txt
pip install "setuptools<81"



4. HerbPAI로 데이터 전처리하기

6) 단일 이미지 Test

```
cd ~/HerbPAI
python detect_dual.py ₩
  --source "$HOME/workshop/1-Image/2-resize/01_Vio_alb/KHB1032938.jpg" ₩
  --weights best.pt ₩
  --keep-canvas ₩
  --project "$HOME/workshop/1-Image/3-herbpai" ₩
  --name test ₩
  --exist-ok
```

```
(herbpai) team18@Viola:~/HerbPAI$ cd ~/HerbPAI
python detect_dual.py \
  --source "$HOME/workshop/1-Image/2-resize/01_Vio_alb/KHB1032938.jpg" \
  --weights best.pt \
  --keep-canvas \
  --project "$HOME/workshop/1-Image/3-herbpai" \
  --name test \
  --exist-ok
/home/team18/HerbPAI/utils/general.py:29: UserWarning: pkg_resources is deprecated as an API. See https://setuptools.py
pa.io/en/latest/pkg_resources.html. The pkg_resources package is slated for removal as early as 2025-11-30. Refrain fro
m using this package or pin to Setuptools<81.
  import pkg_resources as pkg
detect_dual: weights=['best.pt'], source=/home/team18/workshop/1-Image/2-resize/01_Vio_alb/KHB1032938.jpg, data=data/co
co128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, s
ave_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update
=False, project=/home/team18/workshop/1-Image/3-herbpai, name=test, exist_ok=True, line_thickness=3, hide_labels=False,
 hide_conf=False, half=False, dnn=False, vid_stride=1, crop_margin=0, keep_canvas=True
YOLO 🚀 v2.1.0-2-gddddc25 Python-3.10.20 torch-2.13.0+cu130 CPU

Fusing layers...
yolov9-e summary: 839 layers, 68564776 parameters, 0 gradients
image 1/1 /home/team18/workshop/1-Image/2-resize/01_Vio_alb/KHB1032938.jpg: 640x448 1 Specimen, 1 Specimen_label, 1 Ins
titutional_stamp, 1 Barcode, 1 DB_stamp, 1 Ruler, 292.5ms
Speed: 0.2ms pre-process, 292.5ms inference, 0.8ms NMS per image at shape (1, 3, 640, 640)
Results saved to /home/team18/workshop/1-Image/3-herbpai/test
```

결과 확인 : ls "$HOME/workshop/1-Image/3-herbpai/test"

```
(herbpai) team18@Viola:~/HerbPAI$ ls "$HOME/workshop/1-Image/3-herbpai/test"
KHB1032938.jpg
```

4. HerbPAI로 데이터 전처리하기

7) 전체실행

```
ls "$HOME/workshop/1-Image/2-resize"
```

```
(herbpai) team18@Viola:~/HerbPAI$ ls "$HOME/workshop/1-Image/2-resize"
01_Vio_alb  02_Vio_col  03_Vio_kei  04_Vio_man  05_Vio_ros
```

```
cd ~/HerbPAI

for d in "$HOME/workshop/1-Image/2-resize"/*/; do
  python detect_dual.py ₩
    --source "$d" ₩
    --weights best.pt ₩
    --keep-canvas ₩
    --project "$HOME/workshop/1-Image/3-herbpai" ₩
    --name "$(basename "$d")" ₩
    --exist-ok
done
```

```
(herbpai) team18@Viola:~/HerbPAI$ cd ~/HerbPAI
(herbpai) team18@Viola:~/HerbPAI$ for d in "$HOME/workshop/1-Image/2-resize"/*/; do
  python detect_dual.py \
    --source "$d" \
    --weights best.pt \
    --keep-canvas \
    --project "$HOME/workshop/1-Image/3-herbpai" \
    --name "$(basename "$d")" \
    --exist-ok
done
```

3 식물표본 이미지 AI 분류 분석

0. 분석에 필요한 Python 라이브러리 불러오기

주요 라이브러리

도구	역할
Python	파일·폴더 경로 관리
NumPy / Pandas	수치 계산 및 데이터 정리
Matplotlib / Pillow (PIL)	그래프 출력 및 이미지 처리
PyTorch	**AI 모델 학습**
scikit-learn	데이터 분할 및 성능 평가
torchvision	이미지 전처리 및 ResNet-18 사용

import 문법

문법	설명
import A	A 모듈 전체 불러오기
from A import B	A에서 B 기능만 가져오기
import A as B	A를 불러와 B라는 이름으로 사용

1. 기본 설정: AI 실험의 조건을 정하는 곳

① 데이터와 결과 경로 설정

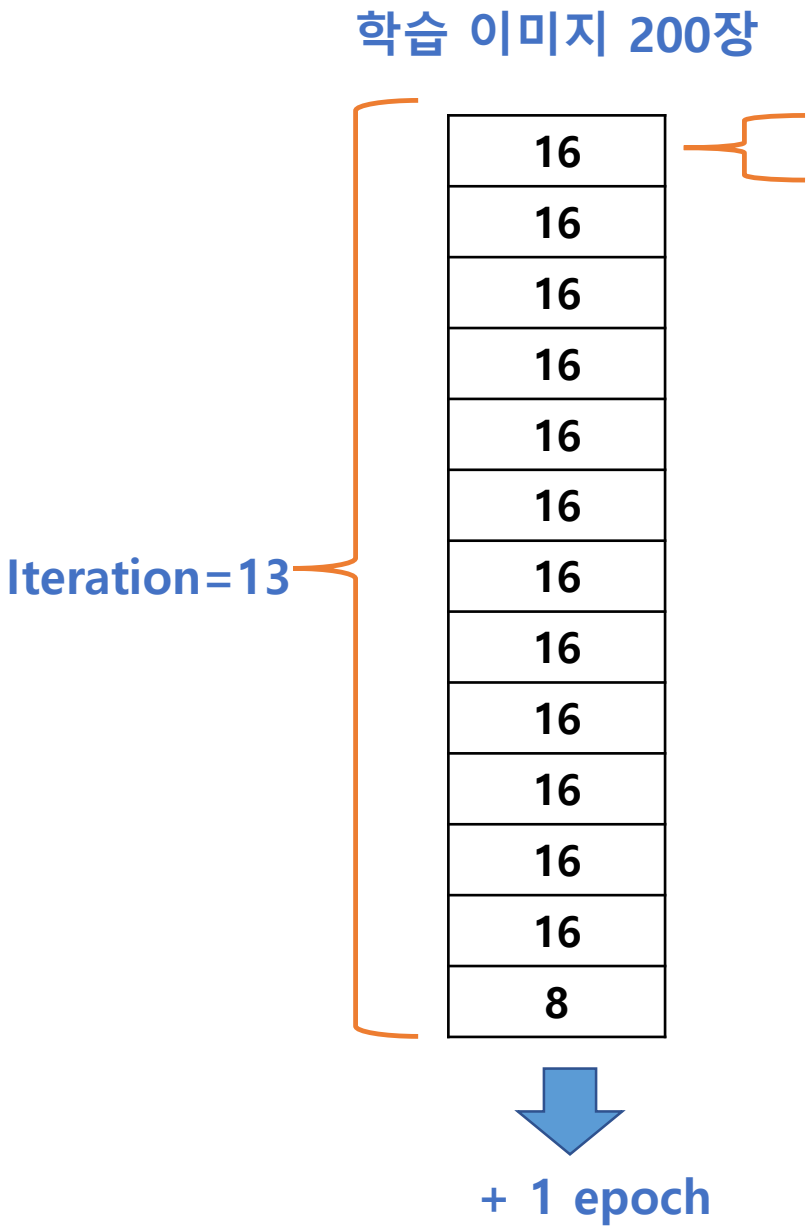
- **DATA_DIR**: 학습에 사용할 이미지가 있는 위치
- **OUTPUT_DIR**: 분석 결과를 저장할 위치

② 주요 설정값

설정값	의미	워크숍 설정
TEST_SIZE	최종 평가용 데이터 비율	20%
***NUM_EPOCHS**	전체 학습 데이터를 반복해서 학습하는 최대 횟수	30
***BATCH_SIZE**	한 번에 모델에 입력하는 이미지 수	16
N_SPLITS	교차검증에서 데이터를 나누는 수	5
***IMAGE_SIZE**	모델에 입력되는 이미지 크기	256 × 384
***LEARNING RATE**	모델이 한 번에 얼마나 크게 학습할지 결정	0.0001

- **Hyperparameter(*)**
  AI가 학습을 시작하기 전에 사람이 미리 정하는 설정값
- **Parameter**
  AI가 학습하면서 스스로 조정하는 값 (예: 가중치)

Q. AI는 이미지를 한 번에 몇 장씩, 몇 번 반복해서 학습할까?



NUM_EPOCHS = 30
BATCH_SIZE = 16

- **Epoch:** 전체 학습 데이터를 몇 번 반복해서 볼 것인가
- **Batch size:** 이미지를 한 번에 몇 장씩 학습할 것인가
  - GPU 메모리 한계 때문에 이미지를 **여러 묶음으로 나누어 학습**
  - **8, 16, 32** 등 주로 2의 거듭제곱 단위를 사용
  - 이번 실습에서는 **Batch size = 16**
- **Iteration:** 하나의 batch를한 번 학습하는 과정

2. 재현성 설정 및 GPU 확인

① 같은 조건에서 다시 실험하기-SEED

- 딥러닝 학습에는 **무작위 과정**이 포함됨
  - 데이터 섞기
  - 모델 가중치 초기화
  - 데이터 증강 등

- **SEED를 고정하면 같은 조건에서 실험을 반복하기 쉬움**

→ 실험 결과의 재현성과 비교 가능성 확보

```
SEED = 42
set_seed(SEED)
```

② 학습에 사용할 연산장치 선택

GPU 사용 가능 → GPU(CUDA)로 학습
GPU 사용 불가 → CPU로 학습

```
device = torch.device("cuda" if
torch.cuda.is_available() else "cpu")
```

3. 이미지 입력 전처리

① 비율 유지 + Padding

- 모든 이미지를 **256 × 384** 크기로 통일
- 원본의 **가로세로 비율은 유지**
- 남는 공간은 **검은색 padding**으로 채움

→ **표본 왜곡 방지**

663 × 949 pixel

256 X 384 pixel

Q1. 왜 이미지 크기를 줄여서 학습하나요?
→ 계산량과 GPU 메모리 사용을 줄여 더 빠르고 효율적으로 학습하기 위해서입니다.

Q. 이미지 크기를 줄이면 정보가 사라져 분류 성능이 떨어지는 것 아닌가요?
→ AI는 픽셀 값으로부터 색, 윤곽, 질감, 형태 등의 패턴을 학습하므로, 분류에 중요한 패턴이 유지되는 범위에서는 크기를 줄여도 반드시 성능이 떨어지는 것은 아닙니다.

Q3. 왜 이미지는 모두 같은 크기로 통일하나요?
→ AI는 이미지를 픽셀 값으로 이루어진 숫자 배열로 계산하므로, 여러 이미지를 동시에 계산하려면 배열의 가로·세로 크기를 같게 맞춰야 합니다.

Q4. 입력 이미지 크기는 마음대로 정해도 되나요?
→ 정해진 정답은 없으며, 사전학습된 모델이 처음 학습될 때 사용한 이미지 크기(예: ResNet 224×224)를 참고하여, 이미지 비율·필요한 세부정보·계산량을 고려하고 실제 성능을 비교해 결정합니다.

3. 이미지 입력 전처리

*과적합 (Overfitting): 학습 데이터에서는 성능이 높지만, 새로운 데이터에서는 성능이 떨어지는 현상
*일반화 성능 (Generalization): 학습하지 않은 새로운 데이터에서도 좋은 성능을 보이는 능력

② 데이터 증강(Data Augmentation)

데이터 증강이란?
- 기존 학습 이미지에 **무작위 변형을 적용하여, 학습할 때마다 조금씩 다른 모습으로 보여주는 방법**
- 실제 이미지 파일 수가 늘어나는 것은 아님

왜 필요한가?
- **제한된 학습 데이터의 다양성을 보완**
  → 같은 이미지에 위치·방향·밝기 등의 변화를 주어 더 다양한 사례를 학습
- **모델의 일반화 성능 향상**
  → 특정 학습 이미지에 과도하게 맞춰지는 **과적합을 완화**

Training 데이터에만 무작위 증강 적용
- Validation / Test에는 적용하지 않음
  → **항상 동일한 조건에서 모델의 성능을 평가하기 위해**

```
RandomHorizontalFlip(p=0.5)              #50%확률로 좌우 반전
RandomRotation(degrees=5)                # −5° ~ +5° 범위에서 무작위 회전
ColorJitter(brightness=0.2, contrast=0.2)  # 밝기·대비를 제한된 범위에서 무작위 조절
```

3. 이미지 입력 전처리

③ AI가 이미지를 읽을 수 있도록 변환하기

이미지 → Tensor 변환 → ImageNet 기준 정규화 → ResNet-18 입력

① Tensor 변환 (ToTensor)

- 컴퓨터는 이미지를 **픽셀의 숫자값**으로 처리
- 일반 이미지: RGB 픽셀값 **0–255**
- **ToTensor()** 적용

  → 픽셀값을 **0–1 범위로 변환**

  → 숫자를 **Tensor 형태로 정리**

- Tensor = AI가 계산할 수 있는 숫자 배열

```
transforms.ToTensor(),
```

예시
[255, 100, 0] → [1.00, 0.39, 0.00]

② 정규화 (Normalize)

- 이번에 사용하는 ResNet-18은 **ImageNet 이미지로 미리 학습된 모델**
- RGB 채널의 값을 **ImageNet 학습 때 사용한 기준에 맞게 조정**

  → 사전학습된 ResNet-18이 **익숙한 숫자 범위와 분포**로 변환

```
transforms.Normalize(
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225],)
```

4. AI 학습용 이미지 불러오기

이미지와 정답 정보를 AI 학습에 사용할 수 있도록 연결해 주는 단계

역할: AI가 사용할 이미지를 한 장씩 불러오고 필요한 정보를 함께 준비

- 식물표본 이미지 **불러오기**
- 이미지를 **RGB 형식으로 변환**
- 학습에 맞게 **전처리(transform) 적용**
- 이미지와 함께 **정답 종(label) 정보 반환**
- 이미지 번호와 파일 경로도 함께 반환하여 맞춘/틀린 표본 추적

반환 정보

- 이미지 + 정답 종 + 이미지 번호 + 파일 위치

```
def __init__(self, samples, indices, transform):    # Dataset을 처음 만들 때 실행되는 부분
def __len__(self):                                   # 이 Dataset에서 사용할 이미지가 총 몇 장인지 알려주는 부분
def __getitem__(self, position):                     # 요청된 이미지 1장을 불러오는 부분
```

5. 데이터 불러오기

이미지를 AI가 학습할 수 있도록
“이미지 + 정답(label)” 형태의 데이터셋으로 변환

① 폴더 이름 = 종(class)

```
1-data/
├── Species_A/
├── Species_B/
└── Species_C/
```

- **ImageFolder**
: 각 하위 폴더를 하나의 **종(class)**으로 자동 인식

② AI는 종 이름 대신 숫자 label로 학습

Species_A → 0
Species_B → 1
Species_C → 2

- 각 이미지에 해당 종의 **정답 번호(label)**가 자동으로 연결

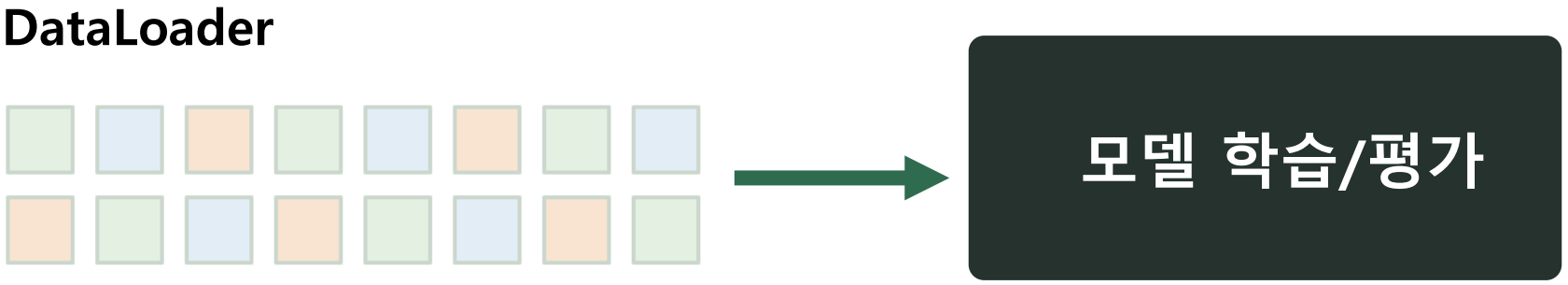
③ 데이터셋 구성 확인

Classes: 3
Total images: 150

Species_A: 50
Species_B: 50
Species_C: 50

- 종 수 · 전체 이미지 수 · 종별 이미지 수 확인

7. DataLoader: 이미지를 batch로 묶어 모델에 전달



BATCH_SIZE = 16

이미지 16장을 한 묶음으로 처리

Train loader
- shuffle=True
  → 매 epoch마다 이미지 순서를 섞어 학습

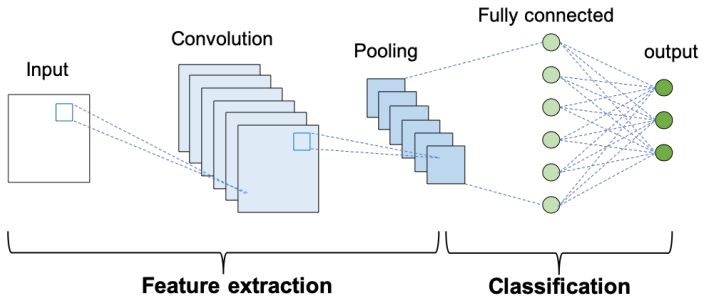
Validation loader
- shuffle=False
  → 이미지 순서를 섞지 않고 평가

8. ResNet-18: 전이학습(Transfer learning)

대규모 이미지에서 학습한 시각적 특징을 식물표본 분류에 활용

사전학습(Pre-training)

ImageNet (대규모 일반 이미지)



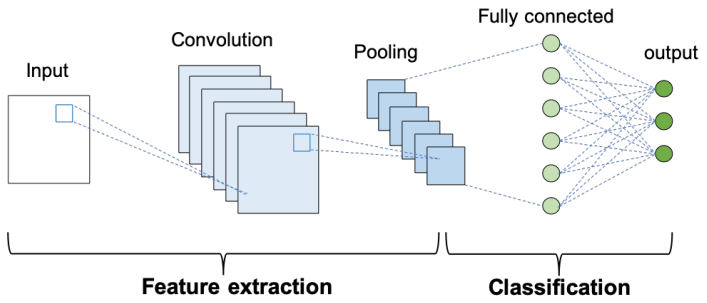
- Dog
- Horse
- Ship
- ...

경계 · 색 · 질감 · 형태 등 일반적인 시각 특징 학습

전이학습

미세 조정(Fine-tuning)

Herbarium dataset



- *Viola albida*
- *Viola biflora*
- *Viola collina*
- ...

사전 학습된 가중치를 바탕으로 식물표본 데이터에 맞게 미세조정

전이학습(Transfer Learning)이란?

- 대규모 이미지에서 학습한 **특징 추출 능력을 새로운 분류 문제에 활용**

Q1. 왜 사전학습 모델을 사용할까?

- 처음부터 학습하려면 **대량의 학습 데이터가 필요**
- 사전학습된 특징을 활용하면 **제한된 데이터에서도 안정적인 학습 가능**

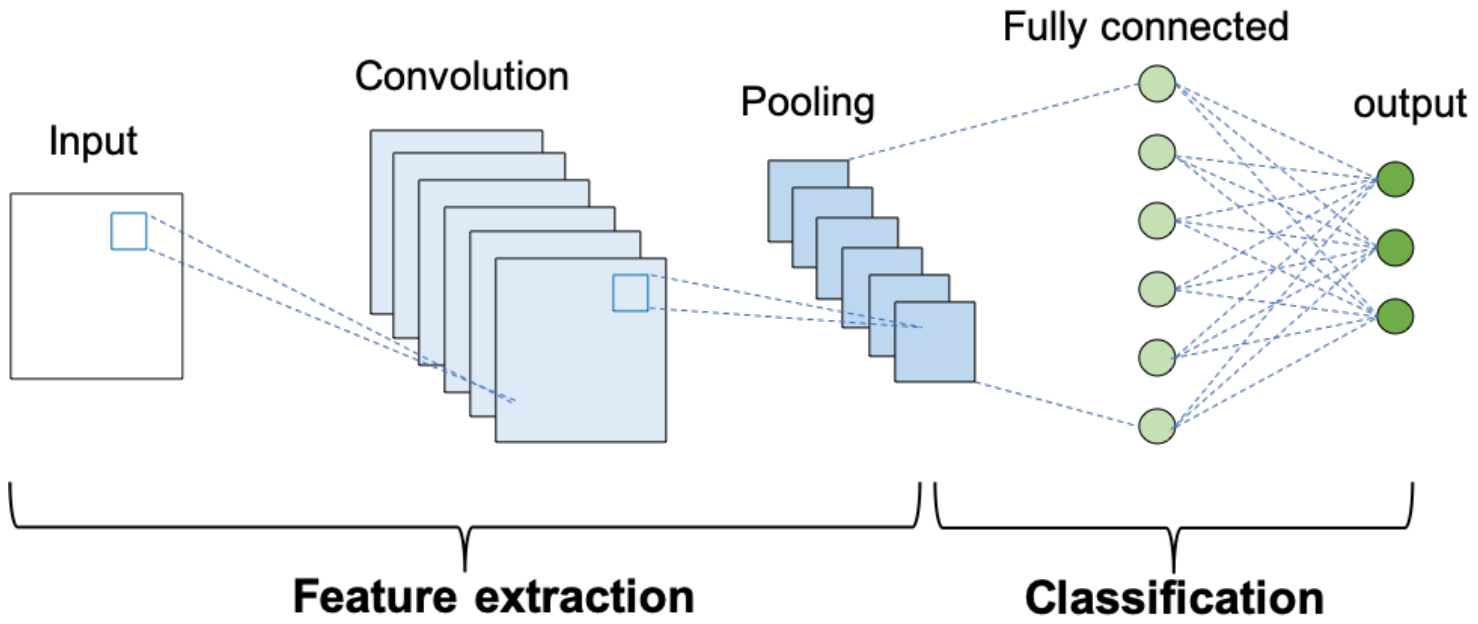
Q2. ImageNet에서 학습한 특징이 식물 분류에도 유효할까?

- CNN은 먼저 **윤곽·질감·형태와 같은 일반적인 시각 특징**을 학습
- 이를 기반으로 **식물표본의 종 구분에 필요한 특징을 추가로 학습**

8. ResNet-18: 합성곱신경망(CNN, Convolutional Neural Network)란?

이미지에서 특징을 자동으로 추출하여 분류하는 신경망

※ CNN의 기본 구조를 단순화한 개념도



① Convolution · 특징 추출

필터가 이미지의 여러 영역을 살펴보며 **선·윤곽·질감 등의 특징 패턴을 찾습니다**

② Pooling · 특징 요약

추출된 특징의 크기를 줄이면서 **중요한 정보를 요약합니다.**

③ Classification · 분류

학습된 특징을 바탕으로 **각 종의 예측 점수를 계산하여 최종 분류합니다**

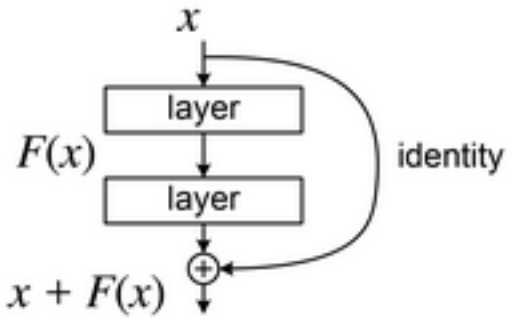
CNN은 사람이 분류 특징을 직접 지정하는 것이 아니라,

이미지 분류에 유용한 특징을 학습 과정에서 자동으로 학습합니다.

8. ResNet-18

① ResNet-18은 CNN 모델의 한 종류
18개의 주요 학습 층으로 구성된 비교적 가벼운 ResNet 모델

② ResNet의 핵심 — Skip Connection
앞의 입력 정보를 몇 개의 층을 건너뛰어 **뒤의 결과에 직접 더하는 연결 구조**
→ 신경망이 깊어져도 학습이 원활하게 이루어지도록 도움



https://en.wikipedia.org/wiki/Residual_neural_network

③ 왜 ResNet-18을 사용할까?
- **비교적 작은 모델 규모** → 학습 속도가 빠르고 계산 부담이 적음
- **Skip connection** → 안정적인 학습에 도움
- **ImageNet 사전학습 가중치 활용 가능** → 제한된 데이터에서도 전이학습 가능

9. 모델은 어떻게 학습할까? — 한 Epoch에서 일어나는 일

한 batch의 학습과정

① 이미지 입력
식물표본 이미지 + 정답 종(label)

↓

② 예측하기
ResNet-18이 각 종에 대한 예측 점수를 계산

↓

③ Loss 계산
예측 결과를 정답과 비교하여 Loss 계산

↓

④ 가중치 수정
Loss가 줄어드는 방향으로 가중치를 조금 수정

다음 batch에서 반복

- **Epoch란?**
  전체 학습 데이터를 한 번 모두 학습하는 것 = **1 Epoch**
- **손실(Loss)이란?**
  모델의 예측이 정답과 얼마나 다른지를 나타내는 수치
- **가중치(Weight)**
  모델의 예측 결과를 결정하는, 학습을 통해 조정되는 내부의 값

학습 (Train)	검증 (Validation)
예측 → Loss 계산 → **가중치 수정 O**	예측 → Loss · 성능 확인 → **가중치 수정 X**

모델은
"예측 → 틀린 정도 계산 → 조금 수정"을
반복하며 학습

9–10. 예측 결과를 이용한 성능 지표 산출

실제값(y_true)과 예측값(y_pred)을 비교하여 분류 성능을 평가

① Validation 예측 결과 수집

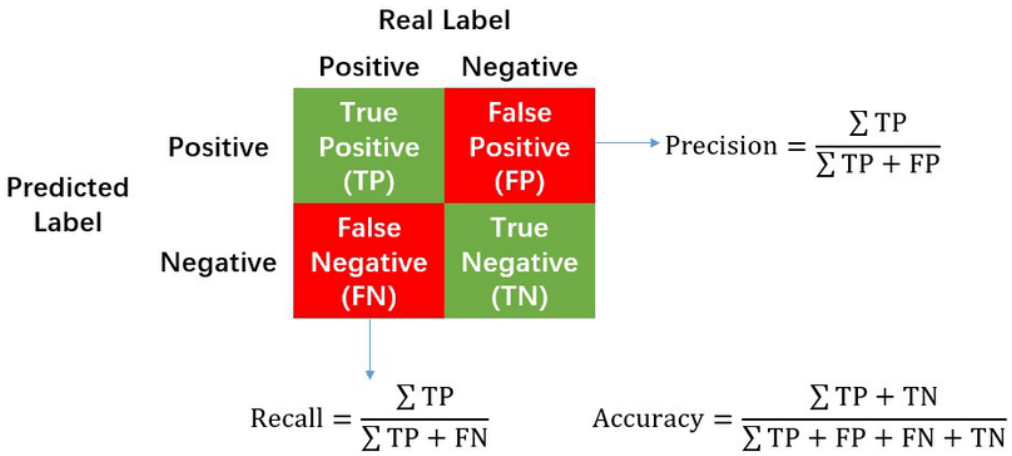
Validation 이미지 → 모델 예측
→ 실제값(y_true) · 예측값(y_pred) 수집

이미지	실제 종 (정답)	AI 예측
1.jpg	Species A	Species A
2.jpg	Species B	Species C
3.jpg	Species C	Species C
…	…	…

9–10. 예측 결과를 이용한 성능 지표 산출

Ma, J., Ding, Y., Cheng, J. C., Tan, Y., Gan, V. J., & Zhang, J. (2019). Analyzing the leading causes of traffic fatalities using XGBoost and grid-based analysis: a city management perspective. IEEE Access, 7, 148059-148072.

실제값(y_true)과 예측값(y_pred)을 비교하여 분류 성능을 평가



② 분류 성능 지표 산출

Accuracy (정확도)	전체적으로 얼마나 잘 맞혔는가?
Precision (정밀도)	그 종이라고 예측한 것들이 얼마나 정확했는가? 모델이 True라고 분류한 것 중 실제 True
Recall (재현율)	실제 그 종을 얼마나 놓치지 않고 찾아냈는가? 실제 True 중 모델이 True로 예측
F1-score (F1 점수)	Precision과 Recall을 함께 고려한 값

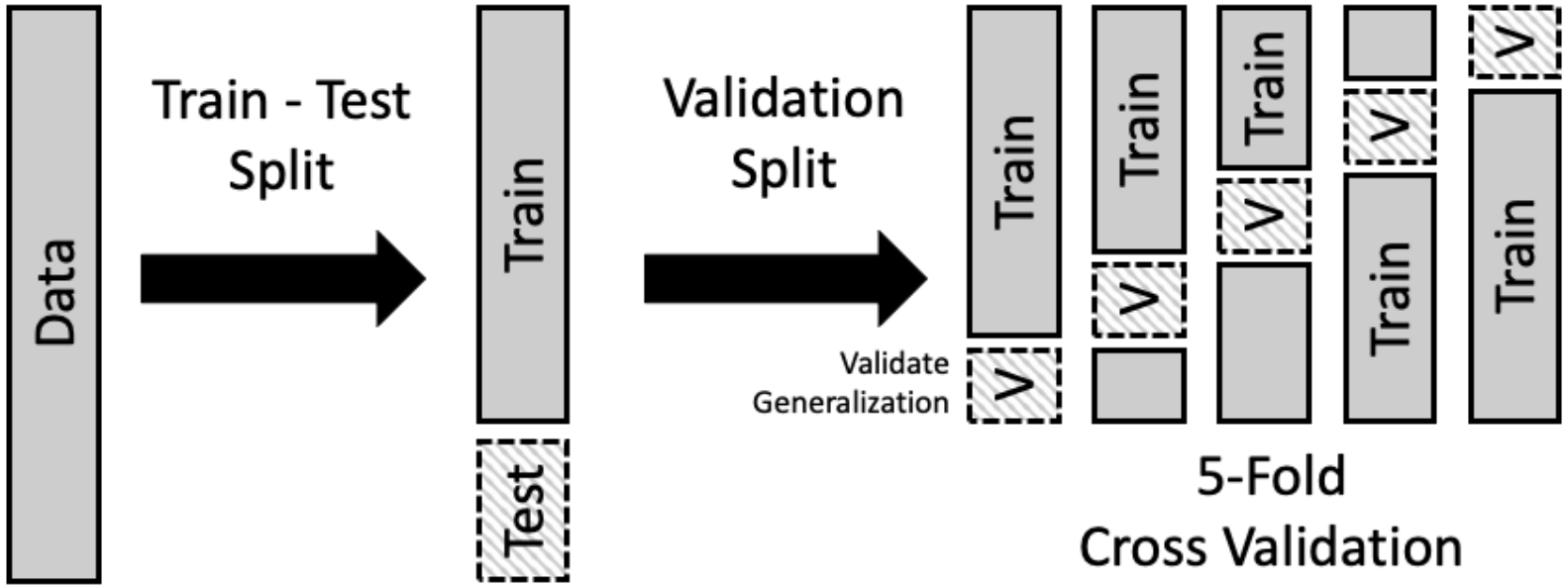
- **Macro 평균**
  각 종의 성능을 각각 계산한 뒤, 종별 점수의 평균을 계산
  → **모든 종이 동일한 비중**으로 반영

Q. 왜 Accuracy만 사용하지 않을까요?

- Accuracy만으로는 **종별 분류 성능의 차이**를 알기 어렵습니다.
- 따라서 **Precision, Recall, F1-score**를 함께 확인하여 모델의 성능을 여러 관점에서 평가합니다.

https://www.lightly.ai/blog/train-test-validation-split

6. 모델 학습을 위한 데이터셋 분할 — Train · Validation · Test



학습 셋(Train set)	검증 셋(Validation set)	시험셋(Test set)
문제집	모의고사	수능
모델이 실제로 학습하는 데이터	**학습 중 모델의 성능을 확인하는 데이터**	**모든 학습이 끝난 뒤 성능을 평가하는 데이터**
• 예측 • 틀린 정도(Loss) 계산 • **가중치를 수정**	• 얼마나 잘 분류하는지 확인 • **가중치는 수정하지 않음**	• 학습에 사용하지 않음 • **마지막에 최종 성능 확인**

6. 독립 Test set 먼저 분리

전체 250장

- 모델 학습 및 검증에 사용
- 5-fold 교차 검정
- 최종 Epoch 결정

- 학습 · 모델 선택에 사용하지 않음
- 마지막에 최종 평가에만 사용

학습에 사용하지 않은 Test set으로 모델의 최종 분류 성능을 평가

10. 교차검정으로 모델 성능 평가하기 — 5-Fold Stratified Cross-Validation

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
1회차	Validation	Train	Train	Train	Train
2회차	Train	Validation	Train	Train	Train
3회차	Train	Train	Validation	Train	Train
4회차	Train	Train	Train	Validation	Train
5회차	Train	Train	Train	Train	Validation

- Development set (교차검정에 사용하는 이미지): **200장**
- **각 회차:** Train 160장 / Validation 40장
- 종별: Train **32장** / Validation **8장**

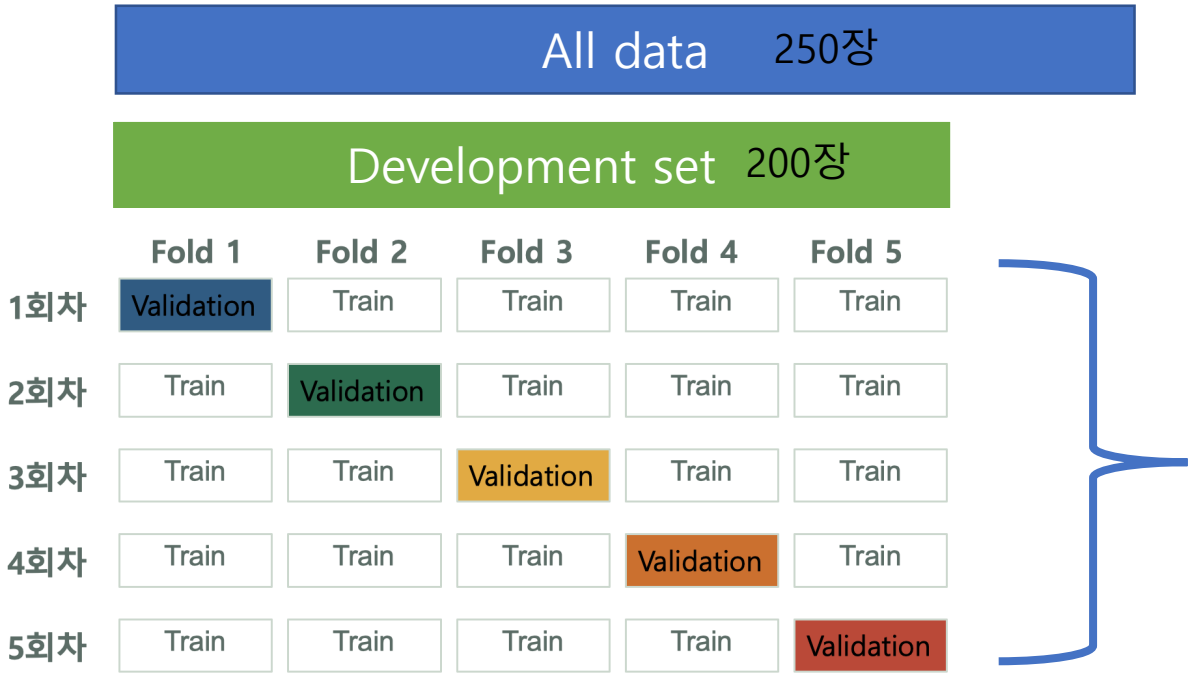
- **교차검정(Cross-validation)이란?**
  데이터를 여러 부분으로 나누어 **학습과 검증을 반복하는 평가 방법**
- **왜 사용할까?**
  한 번의 데이터 분할에 의존하지 않고 **성능을 더 안정적으로 평가하기 위해**
- **Stratified란?**
  각 Fold에 **종별 이미지 비율이 비슷하도록** 나누는 방법
- **Fold란?**
  교차검정을 위해 **데이터를 나눈 각각의 조각**

검증 데이터를 바꿔가며 학습과 검증을 반복하고,
모든 이미지가 한 번씩 검증 데이터로 사용됩니다.

Q. 5번 학습하면 같은 모델을 이어서 학습하나요?
아니요. **각 Fold마다 새로운 모델을 학습**합니다.

11-12. 교차검정 결과를 최종 학습에 어떻게 사용할까?

Epoch: 최종 모델을 몇 번 학습할 것인가?



Fold 별 best epoch

***Best epoch**
: Validation loss가 가장 낮았던 시점

중앙값 (median) 선택

최종 학습 epoch 수 결정

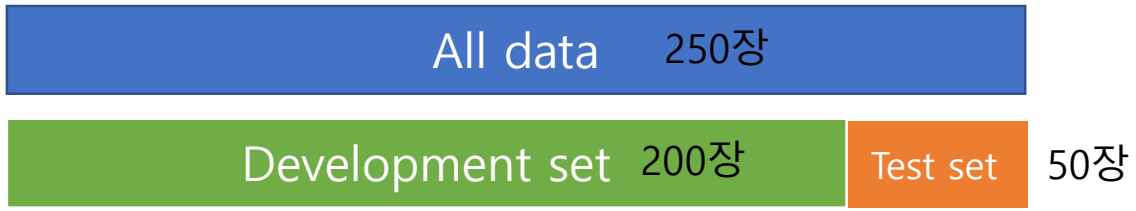
예) fold별 최고 epoch = 6, 8, 7, 12, 7
→ 최종 학습 epoch = 7

Q1. 왜 교차검정에서 얻은 가중치를 그대로 사용하지 않나요?
→ 각 Fold 모델은 **전체 데이터의 일부만 학습**했기 때문입니다. 최종 모델은 전체 학습 데이터를 사용해 새로 학습합니다.

Q2. 교차검정 결과는 최종 학습에 어떻게 활용하나요?
→ 각 Fold에서 확인한 **best epoch를 바탕으로 최종 모델의 학습 횟수를 결정**합니다.

Q. 왜 최종 학습 전에 epoch 수를 정해야 하나요?
→최종 학습에서는 별도의 Validation set을 두지 않으므로, 교차검정 결과를 이용해 학습 횟수를 미리 정합니다.

13–14. 최종 모델 학습과 독립 Test set 평가



교차검정에 사용한 전체 데이터로
최종 모델을 학습한 뒤,
독립적인 Test set으로 최종 성능 평가

사전학습된 ResNet-18 새 모델 준비 → **교차검정에 사용한 전체 데이터로 최종 모델 학습** → **Final model 저장** → **독립 Test set으로 최종 성능 평가**

- 교차검정에서 정한 epoch수

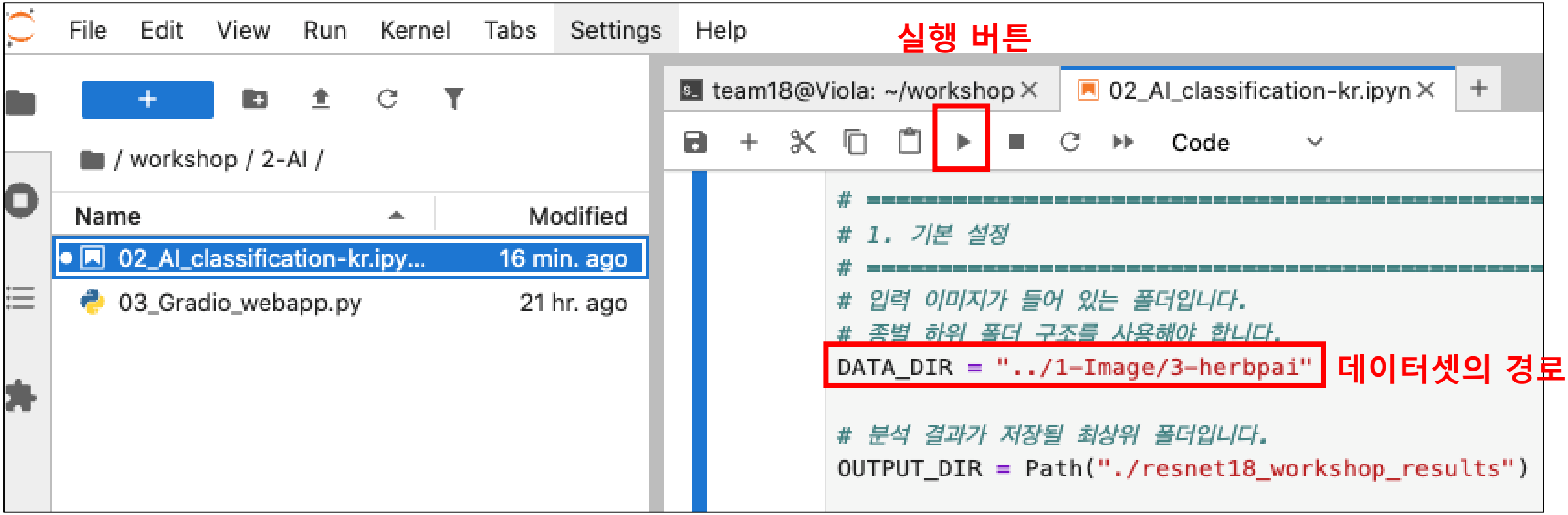
- 최종 분류 성능
- 어떤 종을 잘못 구분했는지
- 모델이 이미지의 어디를 보았는지

Q. 왜 Test set은 마지막에만 사용하나요?
→모델 개발에 영향을 주지 않은 데이터로 **최종 성능을 공정하게 평가**하기 위해서입니다.

AI 분류 코드 실행하기

1) Workshop/2-AI/02_AI_classification-kr.ipynb를 더블클릭하여 여세요.

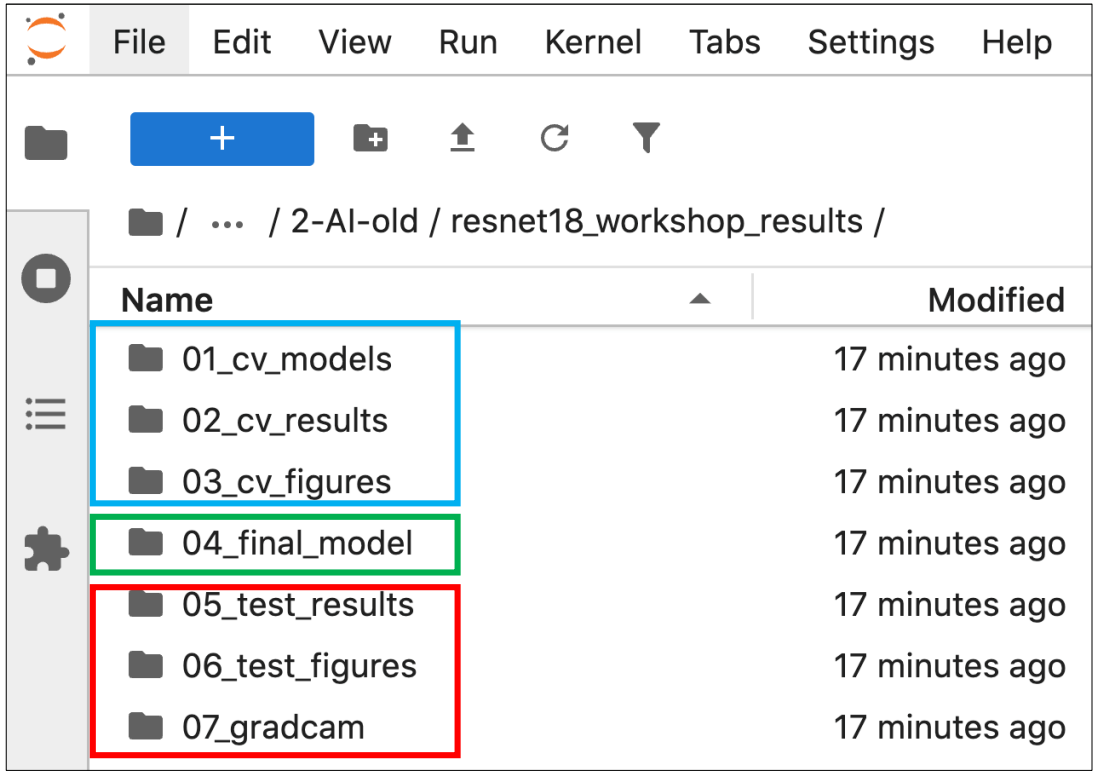
2) 모델에 입력할 데이터셋의 경로가 올바른지 확인하세요.



3) ▶ 실행 버튼을 눌러 분류 코드를 실행하세요.

4 결과 해석 및 모델 활용

AI 분류 결과 확인



- **01–03 교차검증**
  학습 과정 및 안정성 확인
- **04 최종 모델**
  학습이 완료된 AI 모델
- **05–07 독립 Test**
  최종 성능 및 예측 결과 해석

01_cv_models Fold별 최적 모델

02_cv_results 교차검증 성능 및 학습 결과

03_cv_figures 교차검증 학습 그래프

04_final_model 최종 학습 모델

05_test_results 독립 Test 성능 및 예측 결과

06_test_figures confusion matrix

07_gradcam Grad-CAM 결과

1. 모델의 성능은 안정적인가? — 5-fold Cross-validation

① 01_cv_models — Fold별 최적 모델

resnet18_fold1.pth ~ resnet18_fold5.pth
→ 각 Fold에서 **Validation loss가 가장 낮았던 모델** 저장
→ 결과 해석보다는 학습된 모델 파일

/ ...
/ resnet18_workshop_results / 01_cv_models /

Name	Modified
resnet18_fold1.pth	2 hr. ago
resnet18_fold2.pth	2 hr. ago
resnet18_fold3.pth	2 hr. ago
resnet18_fold4.pth	2 hr. ago
resnet18_fold5.pth	2 hr. ago

② 02_cv_results — 교차검증 수치 결과

- cross_validation_summary.csv
  → 5개 Fold의 평균 성능 ± 표준편차
- fold_metrics.csv
  → 각 Fold의 Validation Accuracy, Macro Precision, Macro Recall, Macro F1
- final_epoch_selection.csv
  → 교차검증 결과를 이용해 결정한 최종 학습 epoch
- training_history/
  → Fold별 epoch의 Loss / Accuracy 기록
- initial_development_test_split.csv
  → Development / Test에 포함된 이미지 목록

/ ... / resnet18_workshop_results / 02_cv_results /

Name	Modified
training_history	2 hours ago
cross_validation_summary.csv	2 hours ago
final_epoch_selection.csv	2 hours ago
fold_metrics.csv	2 hours ago
initial_development_test_split.csv	2 hours ago

③ 03_cv_figures —교차검증 학습곡선

- cross_validation_training_overview.png
  → 5개 Fold의 학습 곡선 비교
- fold1_training_curves.png ~ fold5_training_curves.png
  → Fold별 학습 곡선

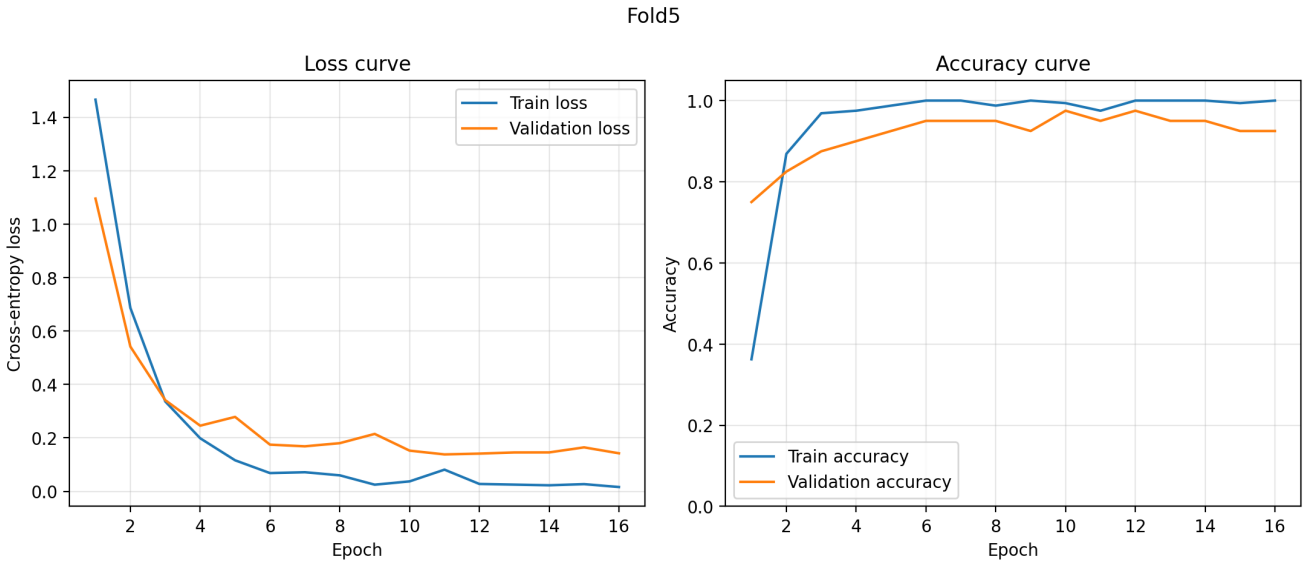
/ ... / resnet18_workshop_results / 03_cv_figures /

Name	Modified
cross_validation_training_overview....	2 hours ago
fold1_training_curves.png	2 hours ago
fold2_training_curves.png	2 hours ago
fold3_training_curves.png	2 hours ago
fold4_training_curves.png	2 hours ago
fold5_training_curves.png	2 hours ago

https://www.kaggle.com/code/ryanholbrook/overfitting-and-underfitting

2. AI는 제대로 학습하고 있을까? — 학습곡선(learning curve)

workshop/2-AI/resnet18_workshop_results/03_cv_figures/



- **Loss = 예측과 정답의 차이를 나타내는 값**
- 낮을수록 좋음
- Validation loss 최저점 = Best model

- **Accuracy = 정답을 맞힌 비율**
- 높을수록 좋음

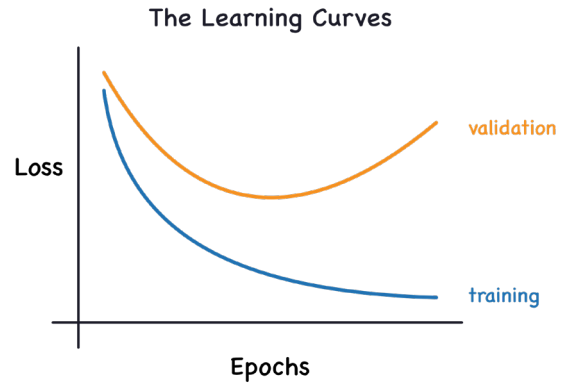
학습 곡선에서 확인할 3가지

① Train과 Validation Loss가 감소하는가?

② Train과 Validation Accuracy가 증가하는가?

③ 학습이 진행될수록 Train과 Validation의 차이가 커지는가?

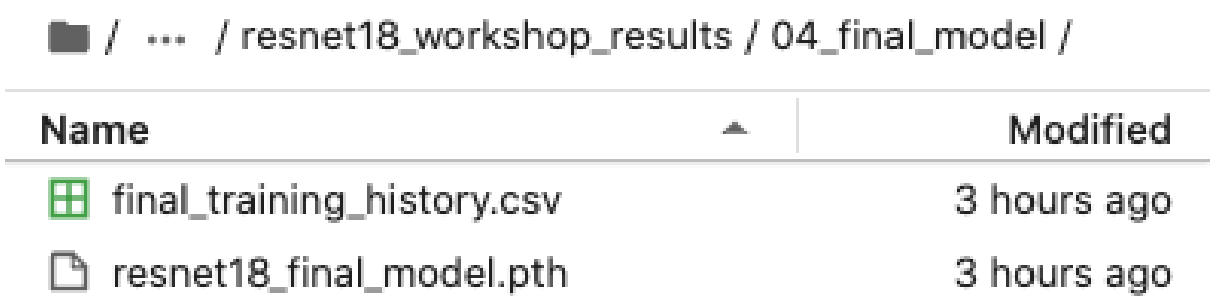
과적합의 전형적인 패턴



과적합(Overfitting)

- Train loss는 계속 감소하지만 Validation loss는 다시 증가하는 현상
- 학습 데이터에는 잘 맞지만 새로운 데이터에 대한 성능은 개선되지 않음

3. 최종 학습 결과는 무엇일까?



① **final_training_history.csv**

→ 최종 모델의 **학습 과정 기록**

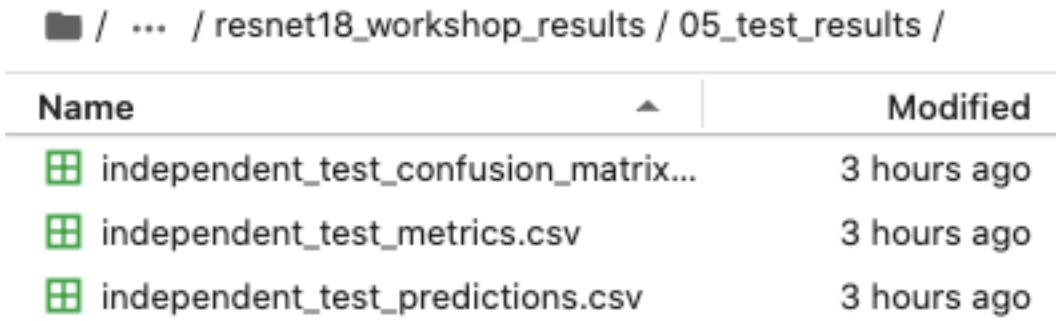
→ Epoch별 Loss, Accuracy 등 확인

② **resnet18_final_model.pth**

→ 최종적으로 학습된 **ResNet-18 모델**

→ **독립 Test 평가 및 실제 이미지 분류(Gradio)에 사용**

4. 처음 보는 표본도 잘 분류할까? — 분류성능지표



- independent_test_metrics.csv → **최종 성능 지표**
- independent_test_predictions.csv → **이미지별 실제 종 / 예측 종**
- independent_test_confusion_matrix.csv → **종별 분류 결과 원자료**

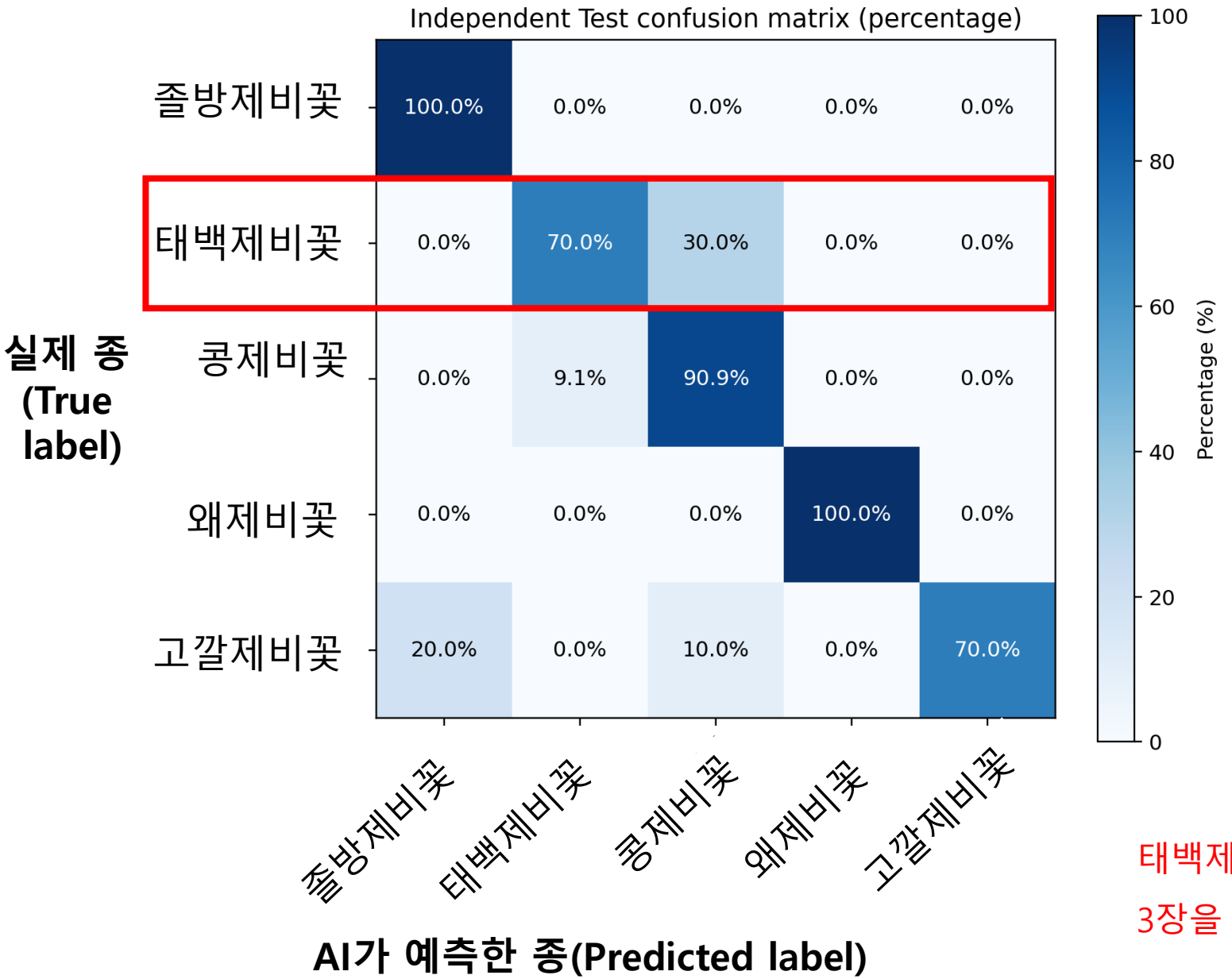
independent_test_metrics.csv

test_loss	test_accuracy	test_macro_precision	test_macro_recall	test_macro_f1	test_images	final_training_images	final_num_epochs
0.432	0.863	0.885	0.862	0.862	51	200	12

- **Test Accuracy = 86.3%**
  처음 보는 51장 중 **44장을 정확히 분류**
- **Macro F1 = 86.2%**
  각 종의 Precision과 Recall을 함께 고려한 F1-score의 평균

- **Macro Precision = 88.5%**
  각 종별 Precision을 동일한 비중으로 평균한 값
  AI가 특정 종이라고 예측했을 때 얼마나 정확했는가
- **Macro Recall = 86.2%**
  각 종별 Recall을 동일한 비중으로 평균한 값
  실제 각 종의 표본을 얼마나 잘 찾아냈는가
- **Test loss = 0.432**
  예측이 정답에서 얼마나 벗어났는지를 나타내는 값
  낮을수록 좋음

5. AI는 어떤 종을 어떤 종으로 헷갈렸을까? — 혼동행렬(Confusion matrix)



/ ··· / resnet18_workshop_results / 06_test_figures /

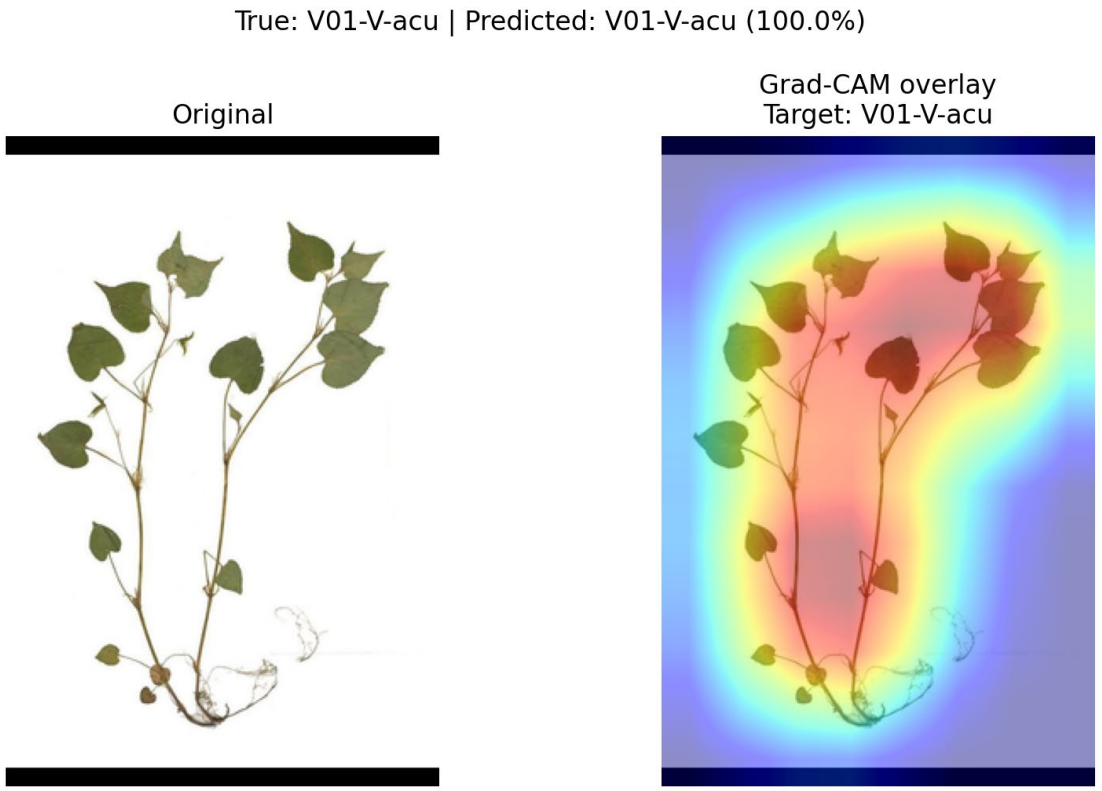
Name	Modified
independent_test_confusion_matrix...	19 seconds ago
independent_test_confusion_matrix...	19 seconds ago

읽는 순서

- **대각선** : 정확하게 분류한 이미지
- **대각선 밖** : 다른 종으로 오분류한 이미지
- **큰 숫자의 오분류**가 어디에서 나타나는지 확인
- 실제로 형태가 유사한 종인지 비교

태백제비꽃 전체 10장 중에
3장을 콩제비꽃으로 헷갈림(30%)

6. AI는 표본의 어느 영역에 주목했을까? — Grad-CAM



Grad-CAM 읽는 방법

- **빨간색·노란색**: AI가 더 강하게 반응한 영역
- **파란색**: 상대적으로 덜 반응한 영역

/ ··· / resnet18_workshop_results / 07_gradcam /

Name	Modified
correct	3 hours ago
incorrect	3 hours ago
gradcam_metadata.csv	3 hours ago

- correct → 맞게 분류한 표본
- incorrect → 잘못 분류한 표본
- gradcam_metadata.csv → 실제 종, 예측 종, confidence 및 Grad-CAM 파일 정보

식물분류학적으로 해석해보기

① 식물체에 반응하는가?

② 실제 분류형질이 있는 부위인가?

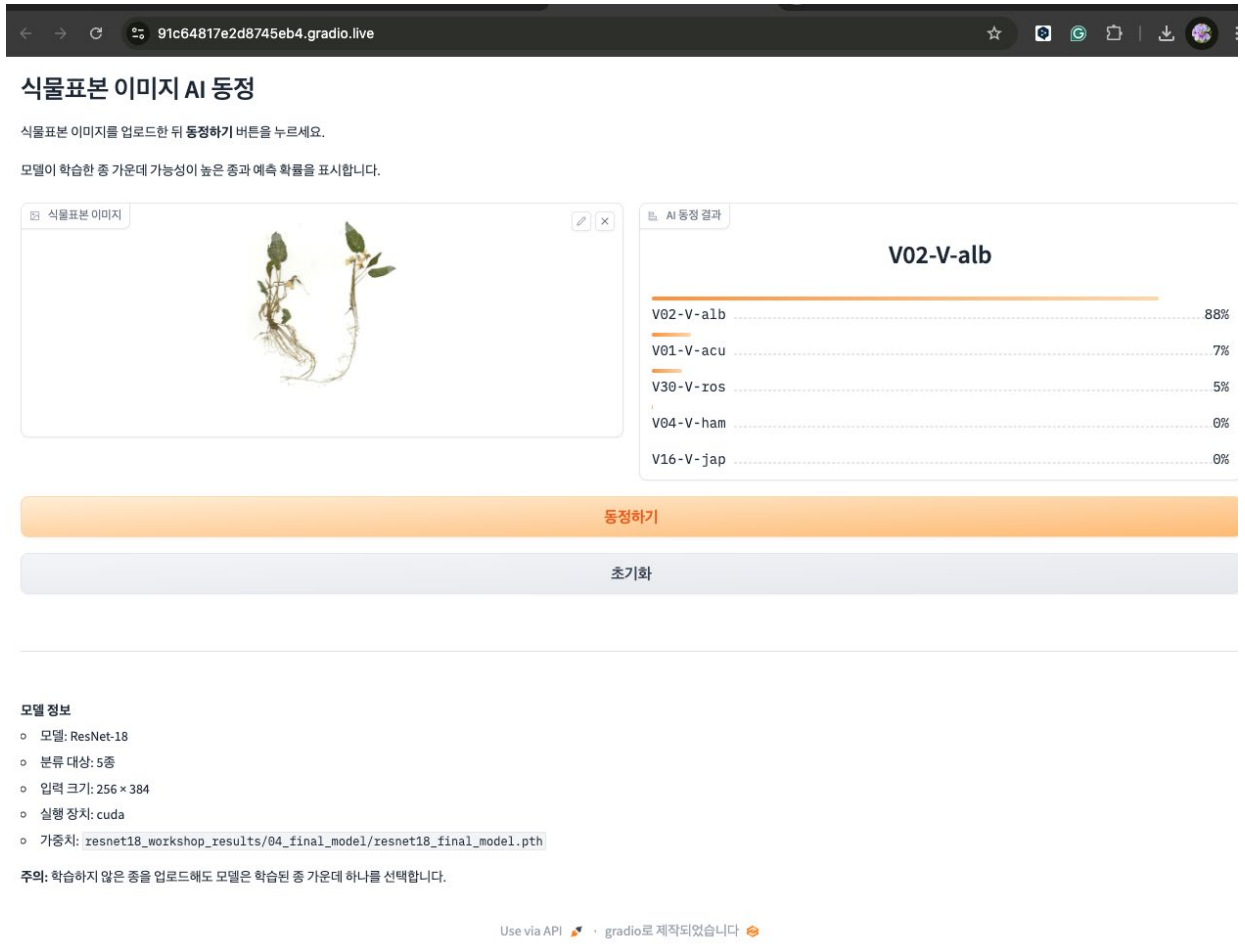
③ 라벨·배경 등에 반응하지 않는가?

④ 정답과 오답에서 차이가 있는가?

Grad-CAM 결과는 어떻게 얻을까요?

1. 이미지를 AI 모델에 입력
2. 예측한 종에 대한 **마지막 합성곱 블록의 특징과 기울기(gradient)를 계산**
3. 영향이 큰 영역을 **Heatmap**으로 만들어 원본 이미지에 겹쳐 표시

7. 학습한 AI 분류 모델 직접 사용해보기



학습에 사용하지 않은 표본 이미지로 테스트해보세요!

-예시 데이터: Viola_predict.zip

① 표본 이미지 업로드

→ 본인의 Test 이미지 1장을 선택

② "동정하기" 클릭

→ 학습한 ResNet-18 모델로 종 예측

③ 결과 확인

→ 가장 높은 예측값의 종과 각 종의 예측 확률 확인

AI 분류 웹앱 실행하기

1) Workshop/2-AI/ 위치에서 새 Terminal을 여세요.

2) workshop 가상환경을 활성화하세요.

conda activate workshop

3) Gradio 웹앱를 실행하세요.

python 03_Gradio_webapp.py

- python : Python 프로그램을 실행하는 명령어
- 02_Gradio_webapp.py : 실행할 Gradio 웹앱 Python 파일

4) 출력된 public URL을 웹 브라우저 주소창에 붙여넣으세요.

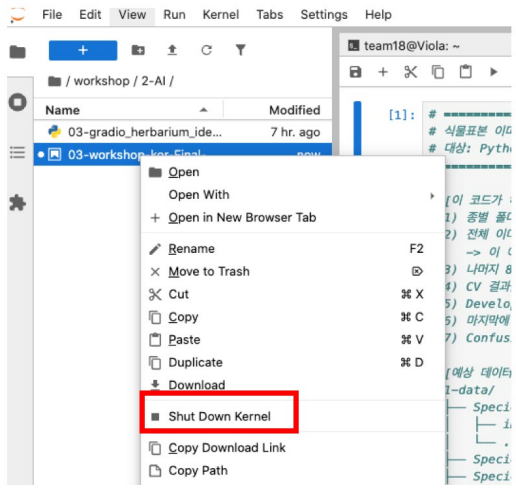
```
(base) team20@Viola:~/workshop/2-AI_species_classification$ conda activate workshop
(workshop) team20@Viola:~/workshop/2-AI_species_classification$ python 03-gradio_herbarium_identifier.py
============================================================
Gradio 식물표본 동정 웹앱
============================================================
Device: cuda
Model: /home/team20/workshop/2-AI_species_classification/resnet18_workshop_results/04_final_model/resnet18_final_model.pth
Classes: ['V01-V-acu', 'V02-V-alb', 'V04-V-ham', 'V16-V-jap', 'V30-V-ros']
공개 접속 링크를 만드는 중입니다.
Running on local URL:  http://127.0.0.1:7860
IMPORTANT: You are using gradio version 3.50.2, however version 4.44.1 is available, please upgrade.
--------
Running on public URL: https://9303f35c3a5a4aa707.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://hu
```

프로그램 안전하게 종료하기

1) 실행 중인 Jupyter Notebook Kernel을 종료하세요.

#JupyterLab에서 .ipynb 파일을 우클릭한 후:

Shut Down Kernel



2) Gradio 웹앱을 종료하세요.

#Gradio를 실행한 Terminal에서:

ctrl + c

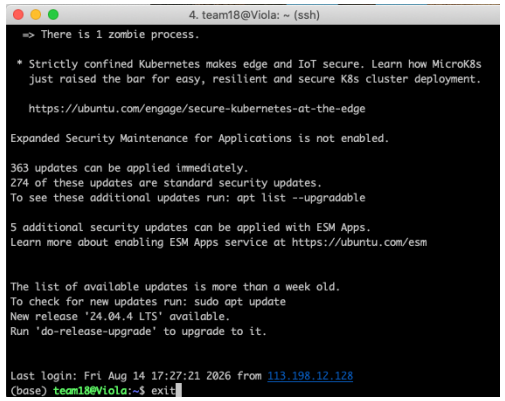
3) SSH 연결과 터미널을 종료하세요.

#서버 SSH 연결 종료

exit

#터미널 종료

exit



4) JupyterLab과 SSH 연결을 종료하세요.

#JupyterLab 서버 종료

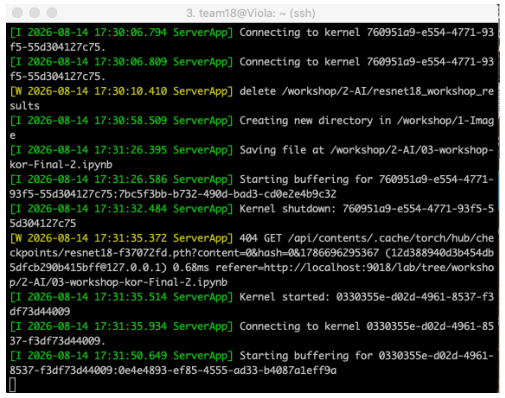
ctrl + c

#서버 SSH 연결 종료

exit

#터미널 종료

exit



※ 창 닫기 ≠ 프로그램 종료

- 창만 닫으면 서버에서 프로그램이 **계속 실행**될 수 있습니다.
- 작업이 끝난 후에는 Kernel과 실행 중인 프로그램을 종료하여 **서버 및 GPU 자원을 반환해 주세요.**