"AI News That's Fit to Print"

The AI Newsletter

Bimonthly

Late Edition

K-water연구원 AI연구센터에서 정기적으로 발간하는 뉴스레터입니다. AI 분야의 뉴스 뿐 아니라 코딩실습과 기술정보 등 다양한 내용과 난이도로 담아냈습니다.

AI Newsletter No.9B

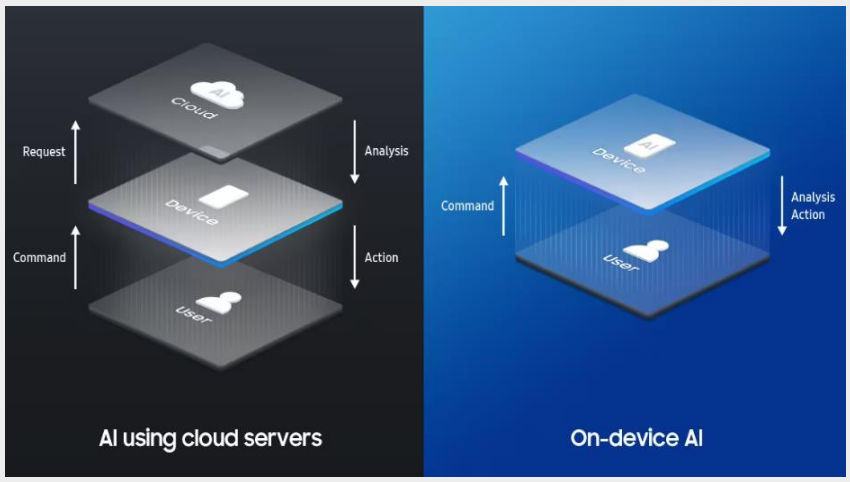
AI LAB, MONDAY, MAY 27, 2024

AI Tech

Tips

내 손안의 인공지능, 온디바이스 AI

최근 삼성전자가 갤럭시 s24를 출시하면서 애플·구글·화웨이 등 빅테크 경쟁사보다 먼저 온디바이스 (On-Device) AI 시대를 열었습니다. 바로 AI 서비스를 네트워크 없이 스마트폰을 통해 손안에서 활용 가능하도록 하였으며, 가전·PC 등 다양한 영역으로 확장하고 있습니다. 여기서 말하는 온디바이스 기술이란 말 그대로 '기기에 장착된' AI로 기존에 고성능 컴퓨팅 자원과 데이터가 필요했던 생성형 인공지능(Generative AI)과 같은 기술을 개인 기기에서 활용 가능할 수 있도록 한 것입니다. 이를 위해 AI 전용 프로세싱 칩인 NPU를 탑재하여 저전력 효율을 높여 사용자 경험을 극대화 했습니다. 이를 실현하기 위해서는 요구되는 다양한 연산을 처리하는 프로세스가 필요한데, 다양성을 갖춘 컴퓨팅 아키텍처를 통해 프로세스 강점을 최대화합니다.



클라우드 기반 AI(왼쪽)와 온디바이스 AI(오른쪽)의 차이점 (출처: 삼성반도체)

답답함 해결! 진행률을 표시하자! tqdm

코드 실행이 오래 걸리면 답답할 때가 있습니다. 제대로 진행되고 있는지, 얼마나 더 기다려야 하는지 궁금할 때가 있죠. tqdm은 진행 상태를 시각적으로 보여주는 라이브러리로, 특히 반복 작업(for, while 등)을 기다릴 때 유용합니다.

① tqdm 적용하기(basic)

```
from tqdm import tqdm
import time
for i in tqdm(range(15)):
    time.sleep(0.5)
```

```
20%|■    | 3/15 [00:01<00:06,  1.99it/s]   # 진행상태 표시(progress bar)
```

② tqdm 꾸미기
- desc : 진행중인 작업에 설명을 추가할 수 있습니다.
- ascii : True로 하면, bar모양을 네모에서 #으로 바꿉니다.
- ncols : bar의 길이를 조절할 수 있습니다.

```
for i in tqdm(range(15), desc='Processing', ascii=True, ncols=100):
    time.sleep(0.5)
```

```
Processing:    33%|###############        | 5/15 [00:02<00:05,  1.99it/s]
# 설명추가     # bar모양 변화(네모→#)     # bar길이 늘어남
```

Python 개발시 속도 개선법

대다수 프로그래밍 언어에서는 반복적인 연산을 수행하기 위해 반복문(for, while)을 활용하나, python에서는 더 빠른 속도의 코딩이 가능합니다.

QUICK TIPS Python에서 내장 함수 사용은 필수!

- C언어보다 Python의 연산 방식이 빠름
- 연산시간 복잡도가 O(n)<O(1) 밖에 소요되지 않음

① 코드 실행시간 측정과 python 내부 함수 활용을 위해 모듈 설치 및 선언

```
import timeit      # 일반적 속도 측정 모듈
import numpy       # python에서 가장 많이 사용하는 연산 함수 제공 모듈
```

② '1에서 1억 합계' 연산을 일반적 While문, For문, python 내장함수 사용, Numpy 함수 사용 비교

```
# Case1 (While문)
def while_loop(n=100_000_000):
    i = 0
    sum = 0
    while i < n:
        sum += i
        i += 1
return sum
```

```
# Case2 (For문)
def for_loop(n=100_000_000):
    sum = 0
    for i in range(n):
        sum += i
        i += 1
return sum
```

```
# Case3 (내장 함수 sum 사용)
def sum_range(n=100_000_000):
return sum(range(n))
```

```
# Case4 (Numpy 함수 sum 사용)
def sum_numpy(n=100_000_000):
return numpy.sum(numpy.range(n))
```

③ 연산 속도 비교 결과

```
print('while loop : ', timeit.timeit(while_loop, number=1))
print('for loop : ', timeit.timeit(for_loop, number=1))
print('sum_range : ', timeit.timeit(sum_range, number=1))
print('sum_numpy : ', timeit.timeit(sum_numpy, number=1))
```

```
While   loop   :  22.4999233
for     loop   :  13.172788400000002
sum_range      :  8.220602800000002
sum_numpy      :  0.2846335999999994
```

④ (결론) sum_numpy > sum_range > for > while 순으로 속도 차이가 있습니다. 파이썬 내장함수 사용이 가장 유리하며, 특히 숫자계산은 수치해석에 특화된 numpy 함수 사용이 유용합니다.

짧게, 더 짧게! Lambda 함수

lambda 함수는 코드를 간결하게 만드는 데 큰 도움이 됩니다. 특히 반복문을 사용하지 않아서 코드 길이를 줄이고 가독성을 높일 수 있습니다.
람다 함수는 단순히 보기만 좋아질 뿐만 아니라 실행 시간도 줄일 수 있습니다. 복잡한 코드일수록 람다 사용을 적극 추천합니다!

QUICK TIPS 람다 함수 사용법

```
lambda 매개변수 : 결과   # 무척 간단하죠?
```

① (Before) 아래 코드는 주어진 숫자를 제곱해서 보여주는 과정입니다.

```
def square(num):
return num * num
numbers = [1, 2, 3, 4, 5]
squared_numbers = []
for number in numbers:
squared_numbers.append(square(number))
print(squared_numbers)
```

```
> [1, 4, 9, 16, 25]   # 결과
```

② (After) Lambda 함수 적용 후 한층 간결해진 모습입니다.

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x *
    x, numbers))
print(squared_numbers)
```

```
[1, 4, 9, 16, 25]   # 결과
```

Hands-on AI Project

K-water Data Studio의 AI 개발 환경 활용해 보기

K-water 정보관리처에서는 디지털전환의 가속화와 AI적용에 대한 사내 보편화를 위해서 GPU 이용이 가능한 Jupyter 환경인 "Data Studio" 서비스를 제공하고 있습니다. 기존에는 AI 모델 개발을 위한 GPU 장착 등 별도의 하드웨어 환경 구축이 여의치 않은 경우 구글 코랩 등을 이용(무료 이용 시 성능제한 등 발생) 할 수 밖에 없었습니다.

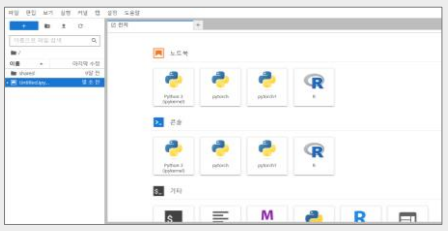
K-water Data Studio를 이용하면 비용부담과 성능제한이 없으며, 내부 자료에 대한 보안 우려도 전혀 없습니다. 이번 프로젝트에서는 직원들이 쉽게 Data Studio를 활용할 수 있도록 활용방법을 정리했습니다.

※ 아래의 내용은 AI연구센터 GitHub에서 열거나, Notebook 파일 (*.ipynb)을 다운로드 할 수 있습니다. 다운로드 한 Notebook자료를 Data Studio에 업로드하여 활용하실 수 있습니다. **AI연구센터 GitHub에서 열기(인터넷망)**

QUICK TIPS Data Studio 접속하기 (인터넷망)

K-water Data Studio에 접속(https://kds.kwater.or.kr)하셔서 오아시스와 같은 사번과 비밀번호로 로그인 하시면 됩니다.

<Data Studio 접속화면>


<Data Studio의 Jupyter Lab 환경>

① Data Studio의 Computing 환경 확인하기

1. Linux 환경 확인

1.1 Ubuntu OS Version Check in jupyter Lab of Data Studio

```
!cat /etc/*elease
```

1.2 이용가능한 Storage 공간 확인

```
!df -h
```

2. GPU 세팅에 대한 확인

2.1 NVIDIA GPU 드라이버 설치에 대한 확인

- GPU 사용을 위해서는 NVIDIA에서 제공하는 GPU Driver를 OS에 맞게 설치해야 하며 아래의 명령어를 통해서 설치된 GPU에 대한 정보를 얻을 수 있음
  1. 드라이버 버전: 현재 설치되어 사용하고 있는 NVIDIA GPU의 드라이버 버전
  2. CUDA 버전: 현재 사용하고 있는 드라이버와 호환이 잘 되는 CUDA 버전
  3. GPU: 현재 설치되어 있는 GPU의 번호로 이용가능한 GPU 개수를 확인할 수 있으며 0, 1, 2개의 GPU 이용 가능함
  4. Name: GPU의 model명으로 현재 Data Studio에는 Tesla V100 2대가 이용 가능함

```
!nvidia-smi
```

2.2 Runtime CUDA 버전의 확인

- 일반적으로 위의 CUDA Driver 버전 11.1와 Runtime CUDA 버전 11.2에서 Version 11.* 의 11버전은 맞춰줄 것을 추천하며, 현재 Data Studio에는 잘 설치가 되어 있어서 이에 맞게 Pytorch 등 딥러닝 모델 개발을 위한 Python libraries를 설치하면 됨

```
!nvcc --version
```

② Data Studio에서 Jupyter 환경의 세팅하기

1. New Conda 환경 만들기

- Python은 오픈소스 프로그래밍 언어로서 각 라이브러리들이 개별로 개발되므로 라이브러리별 버전 충돌이 자주 발생하게 됩니다. 이를 방지하기 위해 프로젝트별로 새로운 conda환경을 만들어서 python 코드를 개발함으로써 버전충돌을 방지할 수 있습니다.

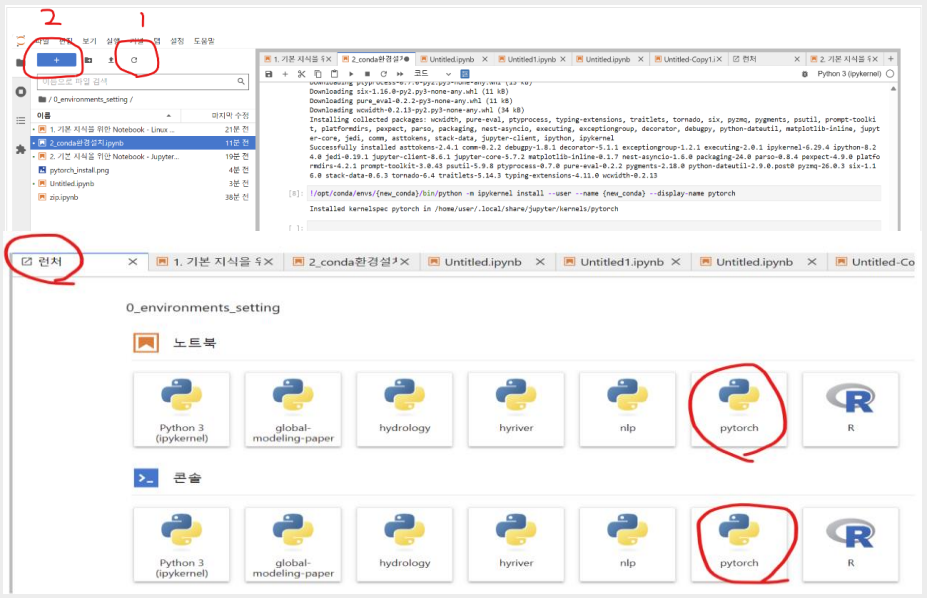
```
# 현재 나의 계정에 설치된 conda 환경 확인
!conda env list
```

```
#new conda 환경 이름 설정, 본 예시에서는 "pytorch"를 new conda 환경이름으로 설정
new_conda = "pytorch"
```

```
!conda create -n {new_conda} -y python=3.10
```

2. New Conda 환경에 Jupyter Kernel 설치

- 1) refresh 버튼을 클릭하고, 2) + 버튼을 클릭하면 두번째 그림과 같이 추가된 런처 탭에 나타난 Pytorch 커널을 확인할 수 있습니다.

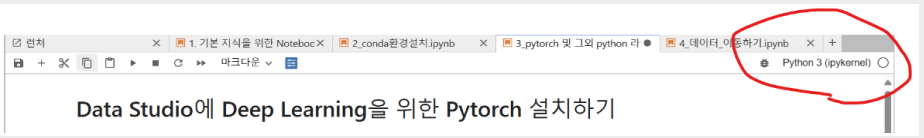



```
!/opt/conda/envs/{new_conda}/bin/pip install ipykernel
```

```
!/opt/conda/envs/{new_conda}/bin/python -m ipykernel install --user --name {new_conda} --display-name {new_conda}
```

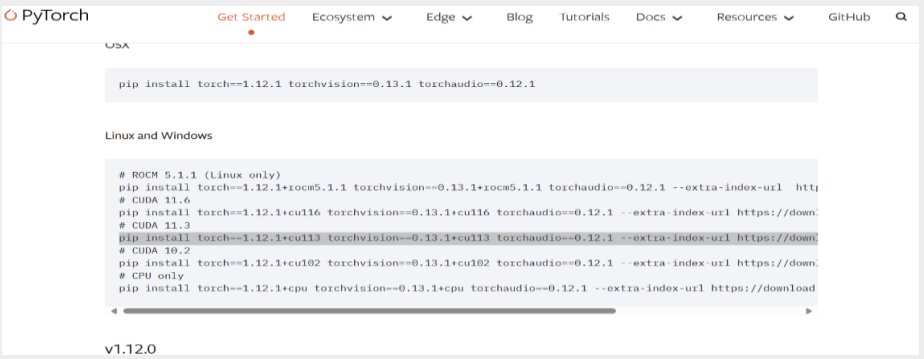
③ Data Studio에서 파이썬 라이브러리 설치하기 (Pytorch)

- 오른쪽 상단의 default로 지정된 Python 3 (ipykernel)을 클릭해서 ②에서 새롭게 생성한 conda 환경의 명칭을 지정하여 변경하고 본 절차를 진행하시기 바랍니다.



1. CUDA 버전 확인 및 버전에 맞는 Pytorch 설치

- ①에서 GPU 드라이버 CUDA 버전 및 Runtime CUDA 버전 확인결과 각각 11.1과 11.2이며, Pytorch install 사이트(https://pytorch.org/get-started/previous-versions)에서 호환되는 Pytorch 버전 확인결과 아래와 같이 CUDA 11.3버전이 설치 가능합니다.



Hands-on AI Project

```
# new conda 환경 이름 설정
new_conda = "pytorch"
```

```
!/opt/conda/envs/{new_conda}/bin/pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113
torchaudio==0.12.1 --extra-index-url https://download.pytorch.org/whl/cu113
```

2. pytorch가 잘 설치되었는지 확인

- 아래의 torch.cuda.is_available() 명령을 실행하여 "True"가 나오면 Pytorch가 GPU와 잘 연결되어서 사용 가능한 상태입니다.

```
import torch
```

```
torch.cuda.is_available()
```

3. 추가적으로 필요한 Python 라이브러리 설치하기

- 아래는 예시이므로 필요한 라이브러리만 설치하면 됩니다.

```
!/opt/conda/envs/{new_conda}/bin/pip install pandas
```

```
!/opt/conda/envs/{new_conda}/bin/pip install xgboost
```

```
!/opt/conda/envs/{new_conda}/bin/pip install scikit-learn
```

```
!/opt/conda/envs/{new_conda}/bin/pip install matplotlib
```

4. Python library Version 확인하기

```
import pandas as pd
print(pd.__version__)
```

④ Data Studio에서 데이터 이동을 위한 파일 압축 및 압축풀기

- 본 절차는 Data Studio에서 다양한 분석을 수행 후
  1. 분석한 데이터를 한번에 압축파일을 만들어서 다운받는 방법
  2. K-water 내부서버 등에서 분석을 위해 대용량의 데이터를 압축해서 Data Studio에 올린 후 압축 해제하기 위한 예시 입니다.

1. 폴더내 전체데이터를 압축하는 방법

- 현재 작업 중인 폴더명 확인

```
!pwd
```

- 압축할 폴더명 지정

```
import shutil, os
# 폴더별 지정
name = "0_environments_setting“   # data_studio_setting_examples
```

- 위에서 지정한 이름으로 폴더내 모든 데이터를 압축하기

```
zip_name = os.path.join("/home/user", name)
shutil.make_archive(name, 'zip', zip_name)
```

2. 외부에서 가져온 데이터 압축을 푸는 방법

- zip 폴더를 압축을 풀고자 하는 폴더로 이동하여 아래의 명령을 통해서 압축을 풀어서 데이터 분석에 활용하면 됩니다.

```
!unzip 0_environments_setting.zip
```