에러정리

비트캠프 KDT 5기 수강생 최건일

< 목차 >

---

1. CUDA, cuDNN, Pytorch, Tensorflow 환경맞추기

프로젝트에 시작하기에 앞서 원활한 진행을 위해서는 환경 세팅을 미리 준비하는 것이 중요하다. 프로젝트 진행 도중에 환경 버전을 바꾸는 것은 위험이 따르기 때문이다. 딥러닝 프로젝트에서 가장 베이스가 되는 환경은 CUDA, cuDNN, Pytorch, Tensorflow 가 될 것이다.

CUDA는 Nvidia에서 개발하고 있으며, 그래픽처리장치(GPU)에서 수행하는 병렬처리 알고리즘을 C프로그래밍 언어를 비롯한 산업 표준 언어를 사용하여 작성할 수 있도록 하는 기술이다. 따라서 Nvidia가 만든 cuda 코어가 장착된 GPU에서 작동된다. CUDA를 사용하는 이유는 매우 간단한데, 많은 양의 연산을 동시에 처리하는것이 목표인 딥러닝에서 유용하게 사용 할 수 있기 때문이다. 코어의 갯수만큼 병렬연산이 가능한데 CPU의 경우는 일반적으로 8~16개의 코어를 가지고 있는데 반해 GPU의 코어는 몇천 개 이상이다. 즉 딥러닝 연산을 빠르게 하기 위해 CUDA를 사용한다고 요약 할 수 있다.

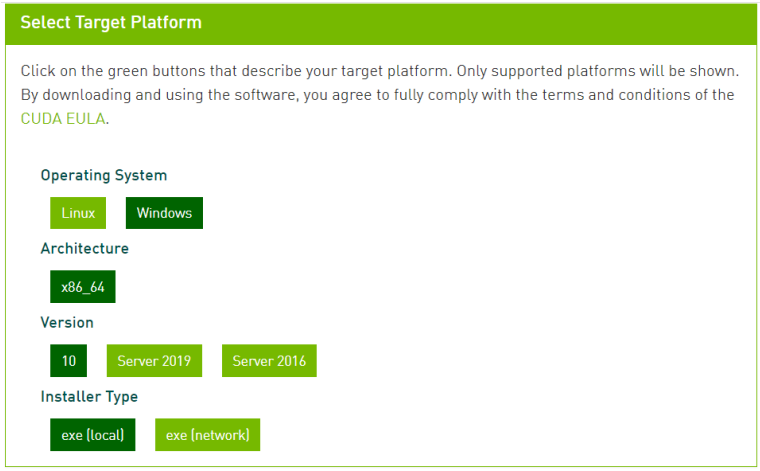
쿠다를 설치하기 전 본인 PC에서 쿠다를 몇 버전까지 사용 가능한지 확인해야 한다. Cmd 창에 아래 명령어를 입력 하면 본인 PC 그래픽카드가 사용 가능한 최대 쿠다버전을 확인 할 수 있다. 당연히 nvidia GPU여야 가능하다.

```
nvidia-smi
```

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 512.78       Driver Version: 512.78       CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   53C    P8     6W /  N/A |    613MiB /  4096MiB |     14%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

보이는 쿠다 버전이 GPU가 쿠다 몇 버전까지 허용하는지 알려준다. 작성자는 11.6버전까지 설치 할 수 있지만 호환성을 위해 11.2 버전을 설치했다. 설치는 아래 주소에서 다운받을 수 있다. (회원가입 필수)

https://developer.nvidia.com/cuda-11.2.0-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exelocal

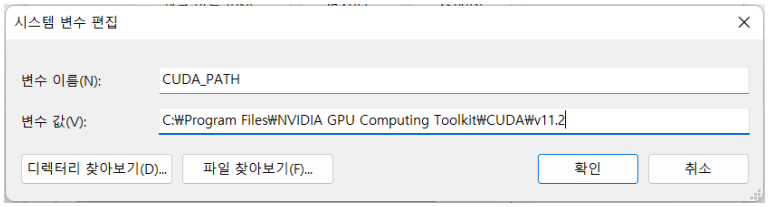


설치 이후 마찬가지로 cmd창에 아래 명령어를 입력하여 쿠다 버전을 확인한다.

```
nvcc -V
```

```
PS C:₩Users₩nengcham> nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Nov_30_19:15:10_Pacific_Standard_Time_2020
Cuda compilation tools, release 11.2, V11.2.67
Build cuda_11.2.r11.2/compiler.29373293_0
```

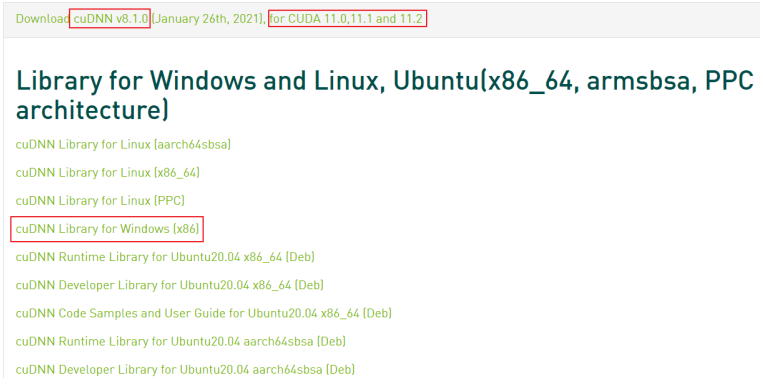
또한 설치시 환경변수에 자동으로 등록되지만 확인을 해보는 것이 좋다.



이후 설치한 쿠다 버전에 맞춰 cuDNN을 설치한다. cuDNN이란 엔디비아 CUDA 딥 뉴럴 네트워크 라이브러리 즉 딥 뉴럴 네트워크를 위한 GPU 가속화 라이브러리의 기초 요소로 컨볼루션(Convolution), 풀링(Pooling), 표준화 (Normalization), 활성화(Activation)와 같은 일반적인 루틴을 빠르게 이행할 수 있도록 하는 라이브러리이다.

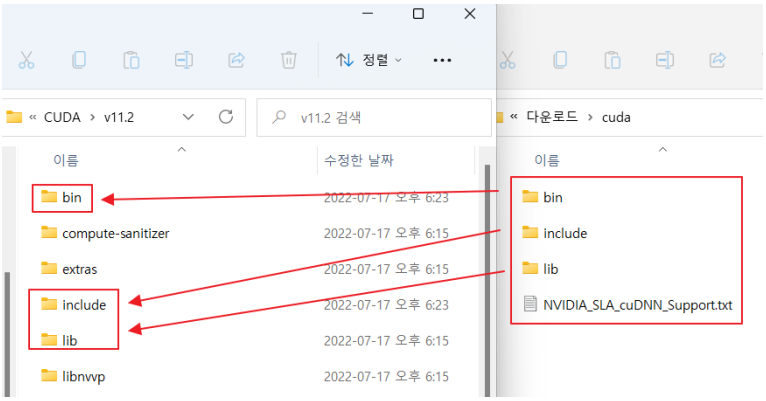
설치는 아래의 주소에서 다운로드 한다.

https://developer.nvidia.com/rdp/cudnn-archive



설치한 쿠다버전, os에 맞춰 다운로드 받는다. 다운받은 파일을 다음의 경로에 덮어쓰기를 한다.

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.2
```



다음은 torch 및 tensorflow 설치이다. 각 라이브러리들은 쿠다 및 cuDNN 버전에 맞춰 설치하면 된다. 버전 확인은 아래의 공식홈페이지에서 확인 가능하다.

- **PyTorch**

https://pytorch.org/get-started/previous-versions/

- **Tensorflow**

https://www.tensorflow.org/install/source_windows#tested_build_configurations

GPU

버전	Python 버전	컴파일러	빌드 도구	cuDNN	CUDA
tensorflow_gpu-2.7.0	3.7~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.6.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.5.0	3.6~3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.4.0	3.6-3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0
tensorflow_gpu-2.3.0	3.5-3.8	MSVC 2019	Bazel 3.1.0	7.6	10.1

작성자 기준으로 보면 CUDA 11.2, cuDNN 8.1 을 설치했기때문에 Conda 가상환경에서 python 3.8 버전에 pytorch 1.8.0, tensorflow 2.7.0을 설치했다.

다음은 Pytorch, Tensorflow가 GPU로 정상 작동하는지 확인하는 작업이다. 아래의 코드로 확인 가능하다.

```
import torch
USE_CUDA = torch.cuda.is_available()
print(USE_CUDA)

import tensorflow as tf
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
import torch
USE_CUDA = torch.cuda.is_available()
print(USE_CUDA)
✓ 1.9s
Python
True
```

```
import tensorflow as tf
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
✓ 6.5s
Python
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 2136091479059960753
xla_global_id: -1
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 1738407936
locality {
  bus_id: 1
  links {
  }
}
incarnation: 6008180427284465344
physical_device_desc: "device: 0, name: NVIDIA GeForce RTX 3050 Ti Laptop GPU, pci bus id: 0000:01:00.0, compute capability: 8.6"
xla_global_id: 416903419
]
```

2. Kobert 환경잡기(mxnet, gluonnlp)

Kobert 라이브러리 설치 명령어는 다음과 같다.

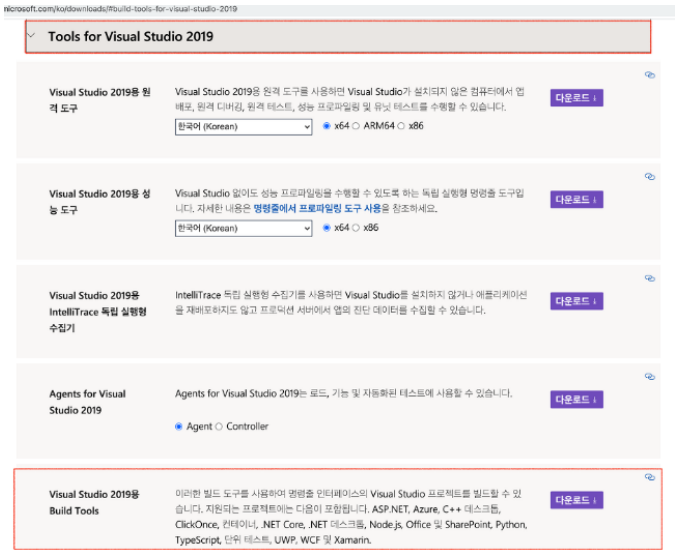
```
pip install --upgrade --no-deps --force-reinstall git+https://git@github.com/SKTBrain/KoBERT.git@master
```

기본 개발환경은 앞선 1번에서 잡았기때문에 필요한 라이브러리만 설치하고 프로젝트를 진행하면 된다. 하지만 pip install mxnet을 설치하려고 하면 에러가 발생한다. Mxnet은 공식홈페이지에 따라 python을 3.6버전으로 낮추면 해결 가능하지만 앞서 설명했듯이 호환성 문제로 바꾸는게 쉽지않다. 또한 gluonnlp 라이브러리를 설치할 때, 토크나이저 mecab을 설치할 때에도 오류가 발생 하는데 오류 코드는 다음과 같다.

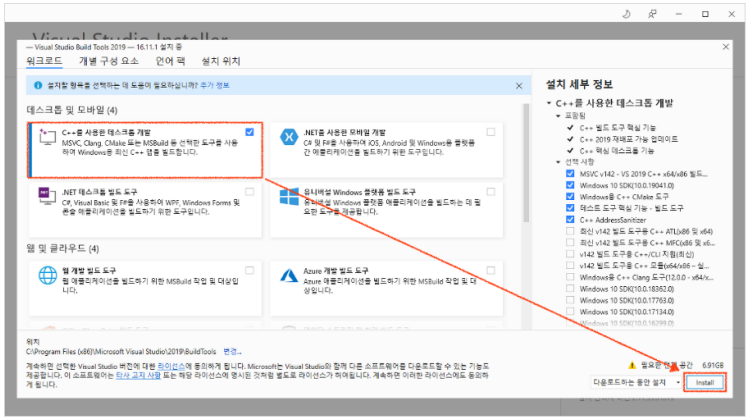
Microsoft Visual build tools가 설치되어 있지 않아 에러가 발생한 것이다. 위 오류 세가지는 visual studio를 설치하면

모두 해결 가능하다. 아래에 주소에서 2019버전을 다운 받는다

https://visualstudio.microsoft.com/ko/downloads/



C++를 사용한 데스크톱 개발을 선택 후 설치한다.



설치 완료 후 재부팅을 진행하고 라이브러리들을 설치하면 오류 없이 설치된다.

```
(kobert) C:\Users\bitcamp>pip install gluonnlp
Collecting gluonnlp
  Using cached gluonnlp-0.10.0.tar.gz (344 kB)
Requirement already satisfied: numpy>=1.16.0 in c:\programdata\anaconda3\envs\kobert\lib\site-packages (from gluonnlp) (1.16.6)
Requirement already satisfied: cython in c:\programdata\anaconda3\envs\kobert\lib\site-packages (from gluonnlp) (0.29.30)
Requirement already satisfied: packaging in c:\programdata\anaconda3\envs\kobert\lib\site-packages (from gluonnlp) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\programdata\anaconda3\envs\kobert\lib\site-packages (from packaging->gluonnlp) (3.0.9)
Building wheels for collected packages: gluonnlp
  Building wheel for gluonnlp (setup.py) ... done
  Created wheel for gluonnlp: filename=gluonnlp-0.10.0-cp36-cp36m-win_amd64.whl size=459853 sha256=222c92847a693c8aa0f3c2f71df17406cba95bd53bebc49e9d634cacd81602d
  Stored in directory: c:\Users\bitcamp\AppData\Local\pip\Cache\wheels\62\62\3a\53be069ac8c9dde533dacce0e716193a4a43e87b5d3f5008c
Successfully built gluonnlp
Installing collected packages: gluonnlp
Successfully installed gluonnlp-0.10.0
```

```
# Name                    Version           Build  Channel
_mutex_mxnet              0.0.40                mkl
backcall                  0.2.0             pypi_0    pypi
blas                      1.0                   mkl
boto3                     1.23.10           pypi_0    pypi
botocore                  1.26.10           pypi_0    pypi
ca-certificates           2022.4.26         haa95532_0
certifi                   2021.5.30     py36haa95532_0
cffi                      1.14.6        py36h2bbff1b_0
chardet                   3.0.4         py36haa95532_1003
charset-normalizer        2.0.12            pypi_0    pypi
click                     8.0.4             pypi_0    pypi
colorama                  0.4.5             pypi_0    pypi
cryptography              35.0.0        py36h71e12ea_0
cudatoolkit               11.0.221          h74a9793_0
cython                    0.29.30           pypi_0    pypi
dataclasses               0.8           pyh4f3eec9_6
decorator                 5.1.1             pypi_0    pypi
entrypoints               0.4               pypi_0    pypi
freetype                  2.10.4            hd328e21_0
gluonnlp                  0.10.0            pypi_0    pypi
```

3. Stylegan 환경잡기

Stylegan 모델을 훈련 및 실행하다보면 다음과 같은 오류를 만날 수 있다.

```
  File "c:/Users/nengcham/EandI/final-jupyter/dsg_oop.py", line 3, in <module>
    from util import load_image, visualize
  File "C:\ProgramData\Anaconda3\envs\stylegan\lib\site-packages\torch\utils\cpp_extension.py", line 1506, in _write_ninja_file_and_build_library
    verify_ninja_availability()
  File "C:\ProgramData\Anaconda3\envs\stylegan\lib\site-packages\torch\utils\cpp_extension.py", line 1562, in verify_ninja_availability
    raise RuntimeError("Ninja is required to load C++ extensions")
RuntimeError: Ninja is required to load C++ extensions
```

py 파일에서 C/C++ 컴파일, 빌드를 요구하고 이 때 원하는 컴파일 도구를 불러오지 못해 발생하는 오류이다. 앞선 2번 에서 VS2019를 설치했지만 환경변수에서 경로에 없어 못잡는 경우이다. 아래의 경로를 Path에 추가하면 해결된다.

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\bin\Hostx64\x64

C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE

개인 GPU 성능에 따라 다음 오류가 발생할 수도 있다

```
RuntimeError: CUDA out of memory.
```

위 오류는 GPU Vram 메모리가 부족하다는 뜻이다.

작성자 의 경우 3050ti laptop에서 위와 같은 오류를 겪었고 해당 그래픽카드는 vram 4GB 였다. GPU 2080에서 실행시 vram 부족현상은 없었고(8GB) 점유율은 아래와 같았다.

4. fastAPI "type_error.dict" 해결

fastAPI에서 post메소드 테스트중 아래와 같은 오류를 만날 수 있다.

```
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:50005 - "POST /test HTTP/1.1" 422 Unprocessable Entity
```

```
"detail": [
  {
    "loc": [
      "body"
    ],
    "msg": "value is not a valid dict",
    "type": "type_error.dict"
  }
 ]
}
```

이는 최신버전 fastAPI post 처리 때문이다. 데이터 전송시 JS에서 헤더에 다음과 같이 추가하여 전송하면 해결된다.

```
headers={"Content-Type": "application/json"}
```

혹은 fastAPI를 다운그레이드 하면 된다. 0.65.1 버전에서는 오류가 발생하지 않는다.

```
(stylegan) C:₩WINDOWS₩system32>pip uninstall fastapi
Found existing installation: fastapi 0.79.0
```

```
(stylegan) C:₩WINDOWS₩system32>pip install fastapi==0.65.1
Collecting fastapi==0.65.1
```

참조 : https://github.com/tiangolo/fastapi/issues/3373