


AI 모델 환경 설치 가이드

모델 학습 및 검증 방법



AI 모델 환경설치 가이드

※ 데이터 학습 및 검증 과정은 **docker** 기반으로 동작함.

학습 환경 구축

1. 학습 환경 및 모델

- 제한 사항: cuda 11.0 버전 이상 지원 가능한 GPU, 15G 이상의 memory, 60GB 의 HDD 또는 SSD
- 학습 환경
 - CPU : 4vCPU
 - Storage : GCS 220GB
 - Memory : 15GB
 - OS : Debian X86_64 GNU/Linux
 - GPU : Nvidia Tesla V100
- 학습 모델 : Unet + Resnet
 - 이미지의 context 를 추출 하는 contracting path 와, 픽셀의 semantic 한 정보를 추출하는 expanding path 로 이루어짐
 - U-net 모델은 biomedical 영역에서 좋은 성능을 보인것 뿐 아니라. 다른 task 에도 적용되기가 쉽고, 최신 모델에 비해 연산량이 적으면서 좋은 성능을 보이는 모델

AI 모델 환경설치 가이드

※ 데이터 학습 및 검증 과정은 **docker** 기반으로 동작함.

학습 환경 구축

2. docker 설치

- Linux, ubuntu 터미널 환경에서 설치

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo curl -fsSL [https://get.docker.com/](https://get.docker.com/) | sudo sh
$ sudo usermod -a -G docker ${USER}
$ docker --version.
```
- Windows, Mac OS GUI 기반 설치
<https://docs.docker.com/get-docker>

AI 모델 환경설치 가이드

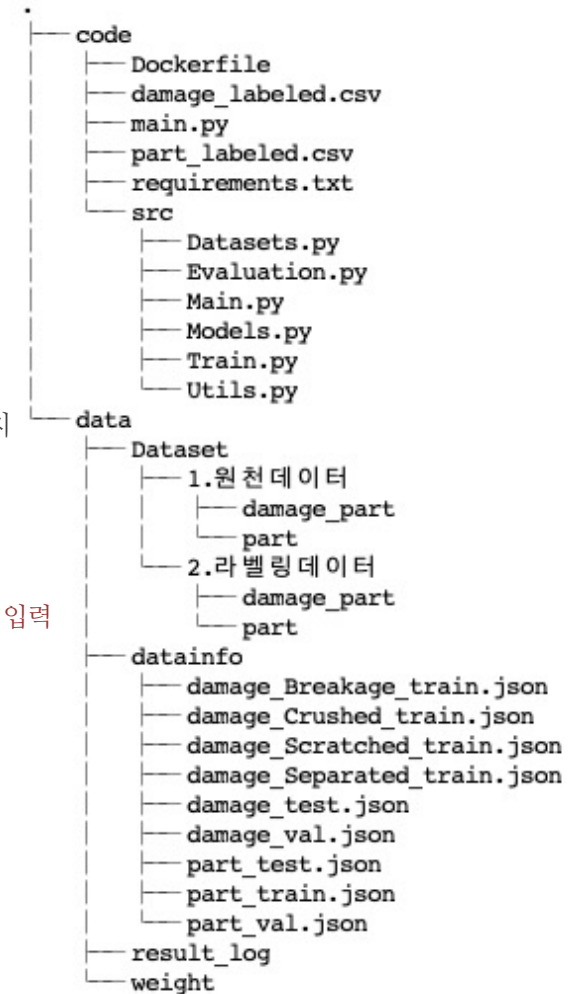
학습 환경 구축

3. 디렉토리 구축 및 설명

- code
 - 모델학습, 도커와 관련된 파일이 저장
 - 해당 파일은 AI hub의 “03.AI 모델/1.AI모델 소스코드”에서 받아야함
- data/Dataset
 - 학습에 필요한 파일(이미지, json)이 저장되어 있음
 - 해당 파일은 AI hub, “01.데이터 ”에서 받아야하며
 - Train, test, validation dataset 모두 동일한 폴더상에 위치
- data/datainfo
 - 이미지, 라벨링 파일들을 cocoformat으로 재구성
 - 해당 파일들을 code, dataset이 다운 완료된 후
 - code, data 상위 폴더로 이동하여 터미널에서 아래와 같이 입력
`$ python code/src/Utils.py --make_cocoformat 1 --task all`

❖ 중요 파일 설명

- code/main.py
 - 실행 파일. 해당 파일을 통해서 훈련 및 검증 수행
- code/part_labeled.csv, code/damage_labeled.csv
 - 이미지별 클래스 갯수 및
train, test, validation dataset 을 분류한 파일



도커 구동 매뉴얼

학습 환경 구축

4. Docker 이미지 생성

- “1.도커 설치, 2. 시연환경 디렉토리 구축” 이 완료된 상태여야 함.
 - 터미널 에서 code 에 접근한뒤 (Dockerfile 이 있는 디렉토리) 아래의 명령어 입력 (10~15분 소요)
`$ docker build -t damage_part_modeling .`
 - Conda 가상환경 기반으로 환경이 설정되며 code에 있는 파일들을 docker Container에 저장
- ❖ 실행중 permission denied error 가 발생시 아래의 명령어 입력 후 다시 이미지 생성
- `$ sudo chmod 666 /var/run/docker.sock`

도커 이미지 생성 완료 예시

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflictir
ger. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnin
Removing intermediate container 7379df3cf375
--> 710a53e1e509
Step 12/16 : ENV LC_ALL C.UTF-8
--> Running in f86e5dd86c3d
Removing intermediate container f86e5dd86c3d
--> 438bed3c42b2
Step 13/16 : ENV LANG C.UTF-8
--> Running in 415b1593edd3
Removing intermediate container 415b1593edd3
--> 0747302943e1
Step 14/16 : ENV NVIDIA_VISIBLE_DEVICES all
--> Running in belce6bf92af
Removing intermediate container belce6bf92af
--> bb5fe40e3763
Step 15/16 : ENV NVIDIA_DRIVER_CAPABILITIES compute,utility
--> Running in 5950066119cc
Removing intermediate container 5950066119cc
--> d21834e40bb7
Step 16/16 : CMD [ "/bin/bash" ]
--> Running in 0074e9389525
Removing intermediate container 0074e9389525
--> 99ed221f20db
Successfully built 99ed221f20db
Successfully tagged test_v1:latest
(base) jupyter@ai-instance-bruno-notebook:~/docker_test/code$
```

이미지 생성여부 확인

`$ docker images`

```
(base) jupyter@ai-instance-bruno-notebook:~/docker_test/code$ d
REPOSITORY          TAG
damage_part_modeling latest
test_v2             latest
<none>              <none>
nvidia/cuda         11.0.3-cudnn8-runtime-ubuntu18.0
gcr.io/inverting-proxy/agent <none>
nvidia/cuda         11.0-base
```

도커 구동 매뉴얼

학습 환경 구축

5. Docker image 가상화 및 터미널 환경 접근

- Docker 실행
 - 가상화된 이미지 'damage_part_modeling' 을 실행하면서, 로컬 저장소와 연결
 - 로컬 저장소의 data/ 안의 이미지를 load하여 학습 및 추론을 하기 위함
 - “:” 앞에 반드시 '3. 디렉토리 구축' 에서 저장한 data 의 절대경로를 입력해야 함
- 명령어 예시
- 결과

```
(base) jupyter@ai-instance-bruno-notebook:~/docker_test/code$ docker run --gpus all -it -v /home/jupyter/docker_test/data:/home/test/data damage_part_modeling
```

```
(base) root@e93d7237bbdd:/home/test#
```

모델 훈련

- “4. Docker image 가상화 및 터미널 환경 접근” 완료 후에 docker 터미널에서 아래의 명령어 수행

- ```
$ python main.py --train y --task part --cls 16
```

- ```
$ python main.py --train y --task damage --label all
```

부위 모델 훈련 예시

```
(base) root@82a0efc4049c:/home/test# python3 main.py --train part --task part --cls 16
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /root/.cache/torch/hub/checkpoints/resnet34-333f7ec4.pth
100%|██████████████████████████████████████████████████████████████████████████████| 83.3M/83.3M [00:01<00:00, 47.7MB/s]
loading annotations into memory...
Done (t=0.01s)
creating index...
index created!
loading annotations into memory...
Done (t=0.00s)
creating index...
index created!
--- start-training ---
1
/home/test/src/Train.py:102: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider converting the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at ../torch/csrc/utils/tensor_new.cpp:201.)
images = torch.tensor(images).float().to(self.device)
0 : 3.192164897918701
Start validation # epoch 0 # step 4
<class 'list'>
Validation #0 #0 Average Loss: 3.2212, mIoU: 0.0002, background IoU : 0.0015, target 1IoU : 0.0002, target 2IoU : 0.0000, target 3IoU : 0.0000, target 3IoU : 0.0000
2
0 : 3.0224900245666504
3
0 : 2.9282312393188477
```

모델학습 및 검증 방법

학습 결과 확인

7. 학습 로그 확인

- “5. 모델 훈련” 과정에서 에폭당 결과가 `data/result_log` 에 json 형태로 저장
[damage_label2]train_log.json, [part]train_log.json

훈련 결과 로그 예시 (part)

```
▼ root:
  comand: "python main.py"
  start_at_kst: "2022-02-11 18:32:11 KST+0900"
  end_at_kst: "2022-02-13 16:44:18 KST+0900"
▼ train_log: [] 26 items
  ► 0:
  ► 1:
  ▼ 2:
    epoch: 4
    ► train_loss: [] 13 items
  ▼ eval:
    ► img: [] 10049 items
  ▼ summary:
    mIoU: 0.27201680166816405
    average Loss: 1.2622319459915161
    background IoU: 0.8732628003764021
    ► target IoU: [] 15 items
    end_at_kst: "2022-02-11 23:07:48 KST+0900"
  ► 3:
  ► 4:
  ► 5:
```


모델학습 및 검증 방법

모델 검증

8. 모델 검증 수행

- “4. Docker image 가상화 및 터미널 환경 접근” 완료 후에 docker 터미널에서 아래의 명령어 수행
- 부위
`$ python main.py --eval y --task part --dataset [val, test] --weight_file [**weigh파일 이름]`
- 파손
`$ python main.py --eval y --task damage --dataset [val, test]`
- 검증 결과는 data/result_log 에 json 형태로 저장 (part_evaluation.json)
- ** data/weight/ 에 저장된 weight 파일 이름 작성 (부위) , 미입력시 검증에 사용된 weight file 사용

검증 결과 예시 (damage)

```
▼ root:
  comand: "python main.py --evaluation"
  start_at_kst: "2022-01-25 23:21:55 KST+0900"
  end_at_kst: "2022-01-25 23:22:06 KST+0900"
▼ evaluation: [ ] 4 items
▼ 0:
  label: 2
  ▼ eval:
    ► img: [ ] 8 items
    ▼ summary:
      mIoU: 0.5988457387725874
      average Loss: 0.16141125559806824
      background IoU: 0.943145095955046
      tartget IoU: 0.2545463815901288
      end_at_kst: "2022-01-25 23:21:59 KST+0900"
    ► 1:
```