모델 환경 설치 가이드

1. 데이터 정제를 위한 폴더 생성

- 이미지와 라벨링 데이터를 한 곳에 모으기 위해 데이터셋이 있는 경로에서 images_T 와 labels_T 폴더를 생성한다.

- 전체 폴더 구조 : 폴더 생성 명령어 → mkdir (폴더명)

workspace/
- **dataset/**
  - **images_T/** **최초 도커 실행 시 생성 필요**
  - **labels_T/** **최초 도커 실행 시 생성 필요**
- **data_utils/**
- **license_det/**
- **license_ocr/**

- 이미지 데이터는 images_T 폴더 안으로 라벨링 데이터는 labels_T 폴더 안으로 전부 옮겨야 합니다.
- 옮기는 작업을 직접하시거나 아래의 코드를 활용하여 진행하여도 됩니다.

Move to other path : 데이터양에 따라 다소 오래걸릴 수 있습니다.
(코드 위치 : 55_license_plate/license_det/move2other_path.py)
: 이미지 데이터를 images_T 폴더에 옮기는 작업을 수행한다.

→ **python move2other_path.py** **--move_file '(이미지가 있는 절대 경로)'**
--output_path '(옮길 폴더의 절대 경로)'
--mode 'jpg'

ex) 이미지 데이터가 있는 절대 경로 = /home/usr/원천데이터 ← '/'가 필요 없음
옮길 폴더의 절대 경로 = /home/usr/images_T/ ← '/'가 필요

: 라벨링 데이터를 labels_T 폴더에 옮기는 작업을 수행한다.

→ **python move2other_path.py** **--move_file '(이미지가 있는 절대 경로)'**
--output_path '(옮길 폴더의 절대 경로)'
--mode 'json'

ex) 라벨링 데이터가 있는 절대 경로 = /home/usr/라벨링데이터 ← '/'가 필요 없음
옮길 폴더의 절대 경로 = /home/usr/labels_T/ ← '/'가 필요

2. 도커 이미지 로드

- 모델 학습 및 검증에 필요한 환경과 라이브러리가 설치되어 있는 도커 이미지를 로드한다.
- 아래의 명령어로 도커 이미지를 로드한다.

→ **docker load -i (도커 이미지 경로/파일명)**

- 아래의 명령어를 통해 torch_test:latest 라는 이미지가 확인되면 정상적으로 로드된 것 입니다.

→ **docker images -a**

3. 도커 실행 및 경로 설정

- 학습 및 검증할 데이터셋 경로를 도커 실행 단계에서 같이 마운트한다.

→ **docker run -it --gpus all -v [샘플 데이터셋 경로]:/workspace/dataset torch_test:lastest**

- 도커 실행 이후 nvidia-smi 명령어를 통해 GPU 목록이 출력되는지 확인한 뒤 코드 실행이 가능합니다.

코드 사용 가이드

 # 학습용 데이터 정제 방법

1. 학습용 데이터 정제 코드 – data_util.py (코드 위치 : 55_license_plate/data_util.py)
- 이미지 데이터와 라벨링 데이터를 차량 번호 인식, 차량 번호판 탐지 모델을 학습 시키기 위한 데이터 정제 코드입니다. 데이터양에 따라 시간이 소요될 수 있습니다.

- 우선, images_T 와 labels_T 폴더를 생성하거나 학습용 데이터의 폴더 명을 앞에 언급한 폴더 명으로 변경하고 이미지 데이터는 images_T 에 라벨링 데이터는 labels_T 에 위치 시킵니다.

- 하단의 코드를 입력하여 위 정제 과정을 수행하면 됩니다.
ex) **python data_util.py --func 'json_split' --mode 'T' --output_path './dataset/'** ← **맨 뒤에'/' 필수**
(docker 에서 연결한 학습 데이터 경로)

- output_path 경로 및 폴더 구조

dataset/
images_T/
xxx.jpg
labels_T/
xxx.json

- 데이터 정제 과정은 아래 번호 순으로 진행하면 됩니다.
1. Split json train/valid/test (func : json_split, mode : T, output_path : ./dataset/)
: labels_train, labels_valid, labels_test 폴더 3 개가 생성되며, 각 폴더에 라벨링 데이터가 8:1:1 비율로 분배된다.
→ **python data_util.py --func 'json_split' --mode 'T' --output_path './dataset/'**

2. Crop images for OCR (func : json2img_crop, mode : T)
: crop_images 폴더가 생성되며, 폴더 내부에 차량번호인식을 위한 이미지 데이터가 저장된다.
→ **python data_util.py --func 'json2img_crop' --mode 'T' --output_path './dataset/'**

3. Transform OCR data on json (func : json2ocr, mode : train / valid / test)
: # 1 과정에서 생성된 폴더를 하나씩 수행하여, 차량번호인식을 위한 이미지 경로와 ground truth 를 매칭시켜주며, labels_train.txt, labels_valid.txt, labels_test.txt 3 가지 텍스트 파일이 생성된다.
→ **python data_util.py --func 'json2ocr' --mode 'train' --output_path './dataset/'**
→ **python data_util.py --func 'json2ocr' --mode 'valid' --output_path './dataset/'**
→ **python data_util.py --func 'json2ocr' --mode 'test' --output_path './dataset/'**

4-1. Transform YOLO data on json (func : json2yolo, mode : T)
: labels_T 폴더 내부에 있는 json 파일을 불러와 차량 번호판 탐지를 위한 라벨링 데이터로 정제가 되며, yolo_gt 폴더가 생성되고 그 폴더 내부에 각 이미지에 대한 텍스트 파일 형태로 저장된다.
→ **python data_util.py --func 'json2yolo' --mode 'T' --output_path './dataset/'**

4-2. Split YOLO dataset path

(코드 위치 : 55_license_plate/license_det/split_dataset_path_edit.py)

: yolo_gt 파일 내부에 있는 라벨링 데이터를 8:1:1 비율로 train_labels.txt, valid_labels.txt, test_labels.txt 가 생성된다.

→ **python split_dataset_path_edit.py --dataset_path '(yolo_gt 폴더가 있는 절대 경로의 상위 폴더)'**

ex) yolo_gt 폴더 위치 = /home/usr/yolo_gt
yolo_gt 폴더가 있는 절대 경로의 상위 폴더 = /home/usr/

4-3. Move to other path

(코드 위치 : 55_license_plate/license_det/move2other_path.py)

: yolo_gt 파일 내부에 있는 라벨링 데이터를 images_T 폴더에 옮기는 작업을 수행한다.

→ **python move2other_path.py --move_file '(yolo_gt 폴더의 절대 경로)'**
--output_path '(imaegs_T 폴더의 절대 경로)'
--mode 'txt'

ex) yolo_gt 폴더의 절대 경로 = /home/usr/yolo_gt ← '/'가 필요 없음
imaegs_T 폴더의 절대 경로 = /home/usr/images_T/ ← '/'가 필요

- [# 1 ~ # 3]는 차량 번호 인식 모델에 사용할 데이터를 정제하는 과정입니다.
- [# 4-1 ~ # 4-3]는 차량 번호판 탐지 모델에 사용할 데이터를 정제하는 과정입니다.

#특정 위치에서 실행이 필요한 코드는 코드 위치를 표시했습니다.

- 데이터 정제 이후 dataset 의 폴더 구조는 아래와 같습니다.

dataset/
images_T/
xxx.jpg
labels_T/
xxx.json
labels_train/
labels_valid/
labels_test/
crop_images/
yolo_gt/

 # 차량 번호 인식 모델

- licenseplate_ocr 폴더

1. 학습용 데이터 lmdb 형태 변환 코드 – create_lmdb.py

- 학습하기 이전에 필수적으로 거쳐야하는 작업으로 학습의 용이성을 위해 이미지-라벨링 정보를 lmdb 형태로 변환시켜주는 코드입니다.

- 앞서 생성한 labels_train.txt, labels_valid.txt, labels_test.txt 에 대해 각각 수행해야합니다.

→ python create_lmdb.py '(mode)' '(labels_train.txt 가 있는 폴더의 절대 경로)'

ex) mode : train / valid / test 3 가지를 한번씩 수행
labels_train.txt 파일 위치 = /home/usr/labels_train.txt
labels_train.txt 가 있는 폴더의 절대 경로 = /home/usr

- 위 코드를 실행한 후, input 폴더 위치에 lmdb_train, lmdb_valid, lmdb_test 파일이 생성됩니다.

2. 학습 코드 – train.py

- 차량번호인식을 위한 학습 코드입니다.

→ python train.py --train_data ./input/lmdb_train
--valid_data ./input/lmdb_valid
--batch_size 192
--manualseed 2000
--Transformation 'TPS'
--FeatureExtraction 'ResNet'
--SequenceModeling 'BiLSTM'
--Prediction 'Attn'

- 위 코드를 통해 학습이 완료되면, ./saved_models/TPS-ResNet-BiLSTM-Attn-Seed2000-Renewed 위치에 학습된 모델 생성되고 학습 관련된 log 또한 포함되어 있습니다.

- 학습 시 GPU 메모리 문제로 이루어지지않을 경우, batch_size 를 낮추어 수행하면 됩니다.

3. 테스트 코드 – test.py

- 학습이 완료된 모델을 통해 테스트셋으로 성능을 확인하는 코드입니다.

→ python test.py --eval_data ./input/lmdb_test
--batch_size 192
--saved_model './saved_models/TPS-ResNet-BiLSTM-Attn-Seed2000-Renewed/best_accuracy.pth'
--Transformation 'TPS'
--FeatureExtraction 'ResNet'
--SequenceModeling 'BiLSTM'
--Prediction 'Attn'

- 테스트 코드가 실행 완료되면, 테스트셋에 대한 학습된 모델 성능을 Accuracy 지표로 표시됩니다.
- result 폴더에 eval_log.csv 를 통해 그 결과를 확인할 수 있습니다.

 # 차량 번호판 탐지 모델

- licenseplate_det 폴더

1. 학습 코드 – train.py
- 차량 번호판 탐지를 위한 학습 코드입니다.
- 데이터 정제 과정에서 생성한 train_labels.txt, valid_labels.txt, test_labels.txt 파일을 data/car_plate 폴더 위치에 옮긴 후 실행해야 합니다.

→ **python train.py --batch_size 16**
--epochs 500
--project '/runs/train'
--name 'license_plate'

- project 는 학습 결과 및 모델을 저장하는 위치이며, name 은 저장되는 폴더 명입니다.

- 학습 시 GPU 메모리 문제로 이루어지지않을 경우, batch_size 를 낮추어 수행하면 됩니다.

2. 테스트 코드 – test.py
- 학습이 완료된 모델을 통해 테스트셋으로 성능을 확인하는 코드입니다.

→ **python test.py --batch_size 16**
--weights '/runs/train/license_plate/best_overall.pt'
--project '/runs/test'
--name 'license_plate'

- weights 는 학습이 완료된 모델 위치를 지정하느 것으로 폴더 내 모델을 선택하면 됩니다.
- project 는 테스트 결과를 저장하는 위치이며, name 은 저장되는 폴더 명입니다.

- result 폴더에 eval_log_det.csv 를 통해 그 결과를 확인할 수 있습니다.