

생활 환경소음분류 AI모델 실행가이드

- 목차
 - 학습환경
 - 1. 코드 파일 설명
 - 2. 도커 파일 설명
 - 3. 실행방법
 - 4. 모델 평가 산출물 확인
 - 5. 샘플데이터

학습환경

1. 개발 시스템 환경

분류	정보
CPU정보	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz
메모리	512 GB
그래픽카드	NVIDIA A100 80GB * 2장
하드용량	7TB
운영체제	Ubuntu 22.04

2. 개발 언어 / 학습 프레임워크

- 개발 언어 : Python 3.8.16
- AI 프레임워크 및 Python 라이브러리 설치

```
pip install torch==1.13.1 torchaudio==0.13.1 torchvision==0.14.1
matplotlib==3.5.3 pandas==1.4.4 scikit-learn==1.1.2 scikit-
image==0.19.3 seaborn==0.12.1 openpyxl==3.0.10 numpy==1.24.2
```

1. 코드 파일 설명

- 코드 파일 구조 확인
 - docker-compose.yml 실행하여 컨테이너 내부에서 실행하는것을 기준으로 한다.

```
docker
├── data
│   └── 소음원별 시나리오 v1.0.xlsx
├── Dockerfile
├── docker-compose.yml
└── sample_handler.py
```

```
├─ custom_preprocessor.py
├─ f1score.py
├─ torch_main_copy.py
├─ run.py
└─ test_only.py
```

1.1 데이터셋 배치작업 및 전처리 파일

- sample_handler.py
 - 1.원천데이터의 .jpg, .wav과 같은 파일명을 가진 json파일을 2.라벨링데이터로 부터 함께 클래스별 폴더명으로 복사한다
 - 전처리 시에 사용됨
 - 클래스명(상위폴더명)으로 데이터를 묶어 주는 전처리 작업
 - 실행시 dataset 폴더가 생성
- custom_preprocessor.py
 - 1800*600 멜스펙토그램 이미지를 전처리 하는 파일
 - 멜스펙트로그램 외에 x, y축의 tick 값을 제외 하여 저장한다
 - 멜스펙트로그램 이미지 파일을 정사각형 모양으로 크롭하여 여러장으로 각각 저장
 - 실행시 dataset_augmented 폴더가 생성

1.2 학습 실행 파일

- data/소음원별 시나리오 v1.0.xlsx
 - 오차행렬 한글화 작업시 필요
- torch_main_copy.py
 - 전처리된 dataset_augmented폴더의 사진을 로드하여 학습에 이용
 - 학습 실행 후 산출물 파일들이 saved_model 폴더내에 생성
- run.py
 - 데이터 전처리 부터 학습까지 필요한 코드 한번에 실행하는 파일
 - 산출물이 saved_model 폴더내에 생성
- f1score.py
 - 학습 및 테스트 시에 사용됨

1.3 저장된 모델 실행 파일

- test_only.py
 - 학습 후 저장된 모델의 성능을 측정
 - 산출물이 saved_model 폴더내에 생성

2. 도커파일 설명

- docker-compose.yml파일을 참고한다

2.1 마운트할 train/val/test 및 sample 데이터셋의 로컬호스트내 경로 설명

- 자유롭게 docker-compose.yml 의 {로컬경로}를 수정한다
- {로컬경로}:{도커 컨테이너 내 경로}
- 마운트할 {로컬경로}에 압축 해제한 train, val, test, (sample) 폴더가 있기만 하면 된다.

- C:/다운로드/train
- C:/다운로드/val
- C:/다운로드/test
- 인 경우 {로컬경로}는 C:/다운로드 이다. 단, {도커 컨테이너 내 경로}는 수정금지
- 작성에서

```
# @docker-compose.yml
services:
  nia_noise:
    ...
  volumes:
    - C:/다운로드:/mnt/data
```

2.2 전처리 또는 학습 후 산출물 경로 설명

- 전처리된 {로컬경로}/train/1. 원천데이터는 {로컬경로}/train/dataset_augmented에서 확인가능
- 각종 산출물은 {로컬경로}/saved_model에서 확인 가능

2.3. 실행 명령어

- 이미지, 컨테이너 생성 및 접속하기

```
# docker-compose.yml 마운트경로 수정 후 이 파일이 있는 경로에서
USERID="$(id -u)" GROUPID="$(id -g)" docker-compose up -d --build
# 컨테이너 접속하기
docker exec -it nia_noise bash
```

- 컨테이너 내부에서

```
# 데이터 전처리
python custom_preprocessor.py
# 모델 학습 및 평가
python torch_main_copy.py
# 모델 평가
python test_only.py
# 샘플데이터 전처리/모델 학습 및 평가
python run.py
```

- 컨테이너 접속종료 및 내리기

```
exit
docker-compose down -v
```

3. 실행방법

3.1 데이터셋 전처리 방법

- 40분 정도 소요

1. 필요한 폴더 구조 확인

- train/val/test 별 폴더 구조 동일

```
{로컬경로} 예시: c:/다운로드
├── train
├── val
└── test
    ├── 1.원천데이터
    │   └── 2.라벨링데이터
    │       ├── A.충간소음
    │       ├── ...
    │       └── D.교통소음
    │           ├── 1.자동차
    │           ├── ...
    │           └── 4.기타
    │               └── a.심야에울리는횡단보도신호기소리
```

2. 실행 명령어

- 데이터셋 전처리 실행

```
$ python custom_preprocessor.py
```

- 실행 예시

3. 실행 후 폴더구조 확인

- train/val/test 별 폴더 구조 동일
- train/val/test 별 1.원천데이터 및 2.라벨링데이터로 전처리 된 파일이 복사 저장됨

```
{로컬경로} 예시: c:/다운로드
├── train
├── val
└── test
    ├── 1.원천데이터
    ├── 2.라벨링데이터
    │   ├── dataset(클래스명 폴더로 .jpg, .json 단순 분류)
    │   ├── dataset_augmented(time annotation적용하여 .jpg 데이터증강)
    │   └── dataset_augmented_balanced(클래스별 데이터 비율이 편향되어 정합성
코드 실행시)
```

```

├─ A1a
├─ A1b
├─ ...
└─ D4a

```

3.2 모델 학습 방법

- GeForce RTX 3060에서는 3시간 40분가량 소요

1. 필요한 폴더 구조 확인

- train/val/test 별 폴더 구조 동일
- 모델 학습 또는 테스트시에는 전처리 된 폴더를 사용함

```

{로컬경로} 예시: c:/다운로드
├─ train
│   └─ dataset_augmented
├─ val
│   └─ dataset_augmented
└─ test
    └─ dataset_augmented

```

2. 실행 명령어

- 데이터 학습 실행

```
$ python torch_main_copy.py
```

- 실행 예시

3. 실행 후 폴더구조 확인

- 데이터 학습후 산출물 확인
- 학습 종료시 모델 평가를 시행하여 산출물 발생

```

{로컬경로} 예시: c:/다운로드
├─ saved_model
│   ├── 20230104_97p@14epoch.pt
│   ├── ...
│   └─ 20230109 02-22-55 confusion summary.xlsx
├─ train
├─ val
└─ test

```

3.3 모델 평가 방법

- 모델 학습 종료 후 동일한 평가를 함
- 따로 평가만 할 때에는 평가할 모델 및 동일한 구조의 테스트 데이터셋이 필요

1. 필요한 폴더 구조 확인

- test데이터셋 및 테스트할 모델 경로 확인

```
{로컬경로} 예시: c:/다운로드
├── saved_model
│   └── 20230104_97p@14epoch.pt
├── test
│   ├── dataset_augmented
│   └── dataset_augmented_balanced(클래스별 데이터 비율을 맞춘 데이터로 실행시)
```

2. 실행 명령어

- 데이터 평가 실행
- train, validation 데이터로 학습된 모델 불러와서 test 데이터로 평가

```
$ python test_only.py
```

- 실행 예시

3. 실행 후 폴더구조 확인

- 모델 평가후 산출물 확인

```
{로컬경로} 예시: c:/다운로드
├── saved_model
│   ├── 20230104_97p@14epoch.pt
│   ├── ...
│   └── 20230109 02-22-55 confusion summary.xlsx
├── test
```

4. 모델 평가 산출물 확인

1. 산출물 폴더 구조 확인

- torch_main_copy.py, run.py 또는 test_only.py 실행시 saved_model에 산출물이 저장된다.

```
{로컬경로} 예시: c:/다운로드
├── saved_model
│   ├── 20230104_97p@14epoch.pt(학습시에만 생성)
│   └── 20230109 02-22-55 confusion.xlsx
```

```

├── 20230109 02-22-55 confusion for heatmap.xlsx
├── 20230109 02-22-55 confusion heatmap.png
├── 20230109 02-22-55 confusion heatmap2.png
├── 20230109 02-22-55 confusion result.xlsx
└── 20230109 02-22-55 confusion summary.xlsx

```

2. 파일 설명

1. confusion.xlsx

- 최초 생성되는 오차행렬 dataframe.
- sklearn.metrics.confusion_matrix()을 단순 저장한 것
- df.column이 예측값, df.index가 참값이다
- 이를 바탕으로 필요한 지표를 직접 계산 및 추가한 엑셀파일들을 생성한다.

2. confusion for heatmap.xlsx

- 컬럼과 행에 A1a~D4a으로 표시된 classID을 한글로 바꿔서 저장

3. confusion heatmap.png

- 오차행렬을 히트맵으로 변환
- 각각의 셀에 대하여 해당 행값을 해당 컬럼값으로 예측한 총 갯수를 표시

4. confusion heatmap2.png

- (각 셀값/행별총합) * 100 = TP/(TP+FN) * 100 = recall * 100

5. confusion result.xlsx

- precision, recall, accuracy, f1-score 등을 계산해놓은것

6. confusion summary.xlsx

- 오차행렬 없이 각종 지표만 표시되도록 요약하여 저장한 파일

5. 샘플데이터

1. 필요한 실행 파일 확인

- docker-compose.yml 으로 컨테이너 내부에서 실행하는것을 기준으로 한다.

```

docker
├── data
│   └── 소음원별 시나리오 v1.0.xlsx
├── Dockerfile
├── docker-compose.yml
└── run.py

```

2. 실행 명령어

- 컨테이너 내부에서 전처리/학습/평가 일괄 실행

```
python run.py
```

- 실행 예시

3. 필요한 폴더 구조 확인

- 데이터셋 다운로드 위치 및 폴더구조 확인
- train/val/test split 된 폴더 구조 없이 데이터 로드시에 자동으로 split 함

{로컬경로} 예시: c:/다운로드

```
└─ sample
  └─ 1.원천데이터
  └─ 2.라벨링데이터
    └─ A.충간소음
      └─ 1.중량충격음
        └─ b.아이들발걸음소리
      └─ 2.경량충격음
        └─ a.가구끄는소리
      └─ 3.생활소음
        └─ b.샤워할때물소리
      └─ 4.악기
        └─ b.피아노연주소리
      └─ 5.애완동물
        └─ b.고양이우는소리
    └─ B.공사장
      └─ 1.건설장비
        └─ b.항발기의파일뽑는소리
      └─ 2.차량
        └─ a.덤프트럭의엔진소리
    └─ C.사업장
      └─ 1.상가건물
        └─ a.옥외설치확성기의소음
      └─ 2.학원시설
        └─ a.등하원아이들떠드는소리
      └─ 3.골프연습장
        └─ a.골프연습장의타구음
    └─ D.교통소음
      └─ 1.자동차
        └─ a.심야에자동차가빠르게주행하는소리
      └─ 2.트럭
        └─ a.심야에배송트럭이빠르게주행하는소리
      └─ 3.이륜차
        └─ a.심야에이륜차가빠르게주행하는소리
      └─ 4.기타
        └─ a.심야에울리는횡단보도신호기소리
```

4. 실행 후 폴더구조 확인

◦ 전처리/학습/평가 산출물 확인

```
{로컬경로} 예시: c:/다운로드
└─ sample
   └─ 1.원천데이터
   └─ 2.라벨링데이터
   └─ dataset(전처리시 자동생성 - 클래스명 폴더로 .jpg, .json 단순
분류)
   └─ dataset_augmented(전처리시 - time annotation적용하여 .jpg 데
이터증강)
   └─ dataset_augmented_balanced(클래스별 비율이 편향되어 정합성 코
드 실행시)
   └─ test(test_only.py실행용으로 자동생성)
   └─ dataset_augmented_balanced
   └─ saved_model
   └─ 20230104_97p@14epoch.pt
   └─ ...
   └─ 20230109 02-22-55 confusion summary.xlsx
```