Pytorch_Audio

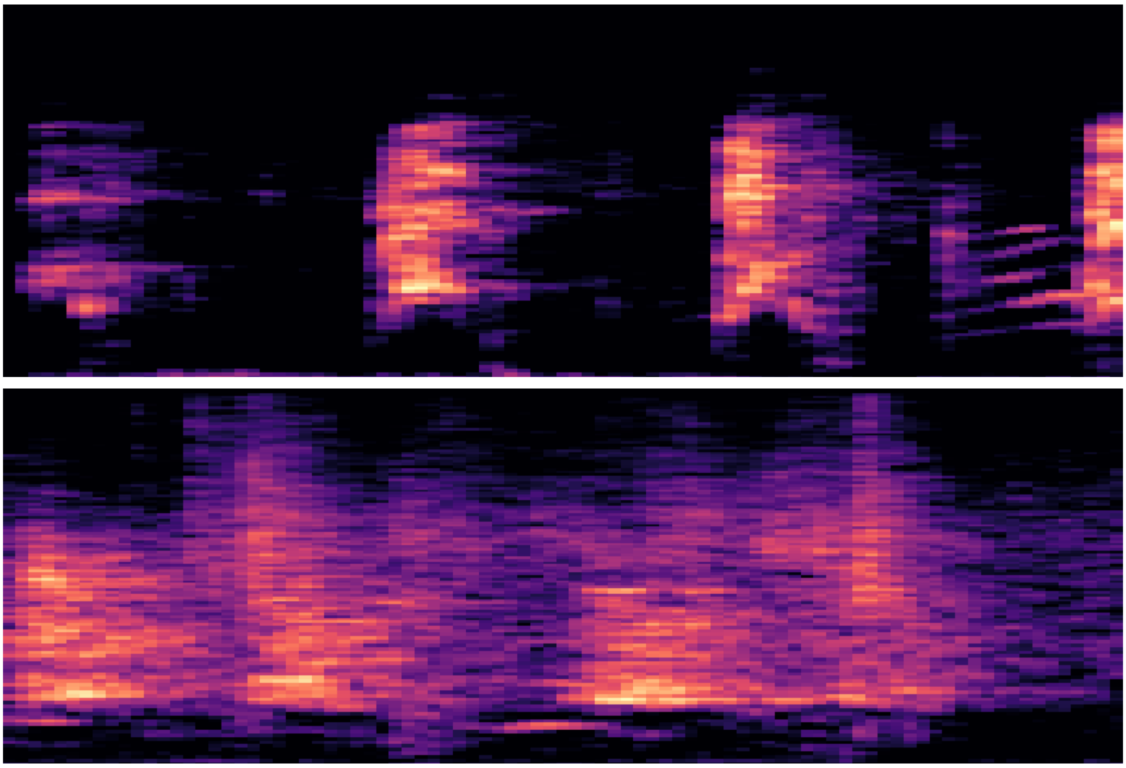
`Pytorch`, `Python=3.6`, `Windows10`, `CNN`, `Librosa`, `PyAudio`

0. Index

1. 개요

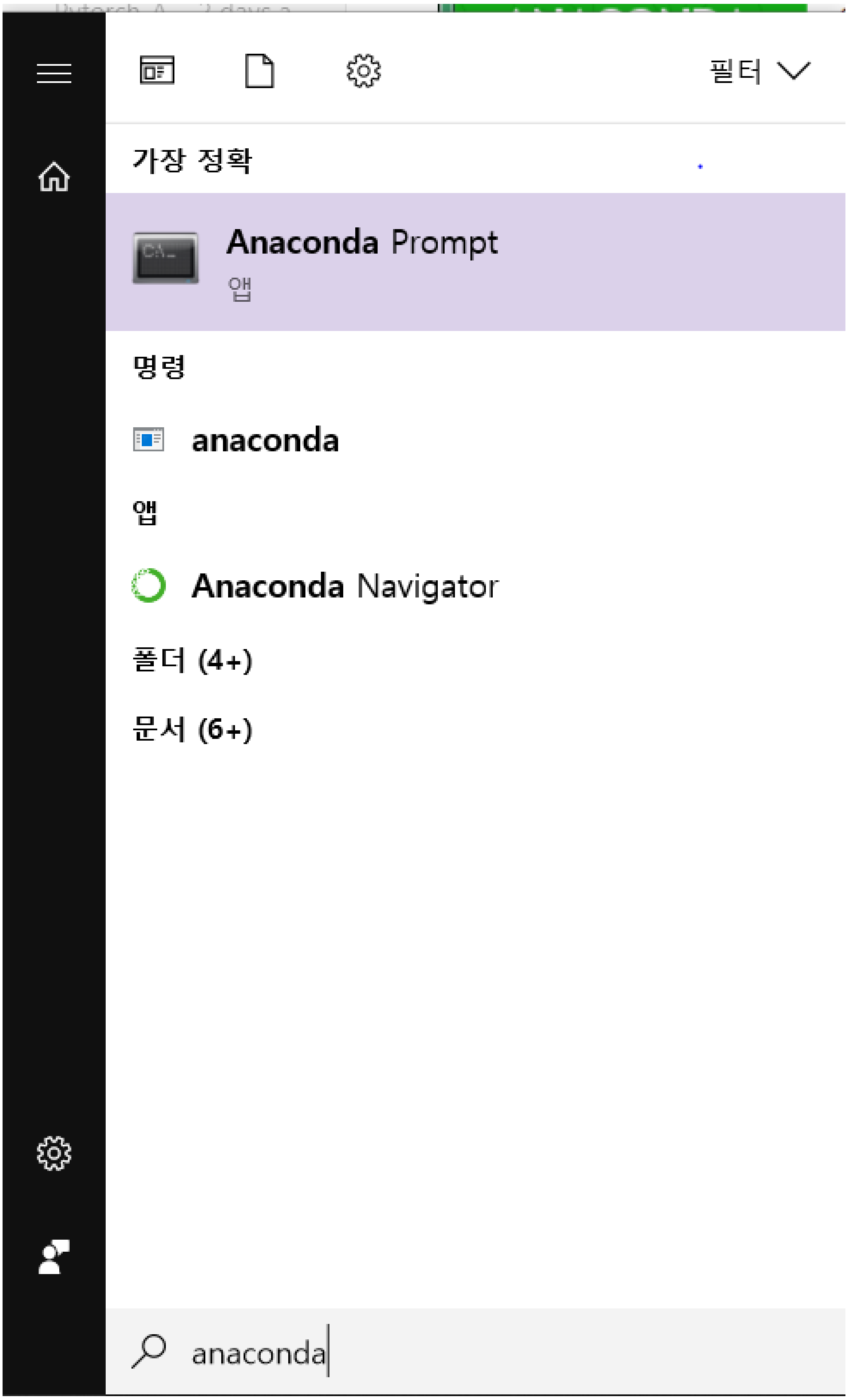
- **Pytorch**의 **CNN**, **Librosa**와 **PyAudio**를 통한 **Audio_Classification**(소리 구별)
- **Librosa**와 **PyAudio**를 이용해 **소리파일**(`.wav`, `.flac`)들을 **시각화**(스펙트로그램)이미지로 변환하여 **CNN**(Convolutional Neural Netowrk)를 통해 구별하도록 구현(**소리 구별 중심**)

[Spectrogram Example]



2. 라이브러리 설치

- **설치해야할 라이브러리**
    - **Pytorch**(이미 상위 폴더의 README에서 설치함)
    - **Librosa**(Python에서 많이 쓰이는 음성 파일 분석 프로그램)
    - **Pydub**(audio file의 channel이 mono가 아닌 stereo일 경우, mono로 변경하기 위함)
    - **PyAudio**(Audio 녹음 및 재생)
- **PIP를 이용한 라이브러리 설치**
    - > **pip**는 파이썬으로 작성된 패키지 소프트웨어를 설치 · 관리하는 패키지 관리 시스템이다.
    - **Anaconda Prompt를 실행한다**



- 가상환경 활성화

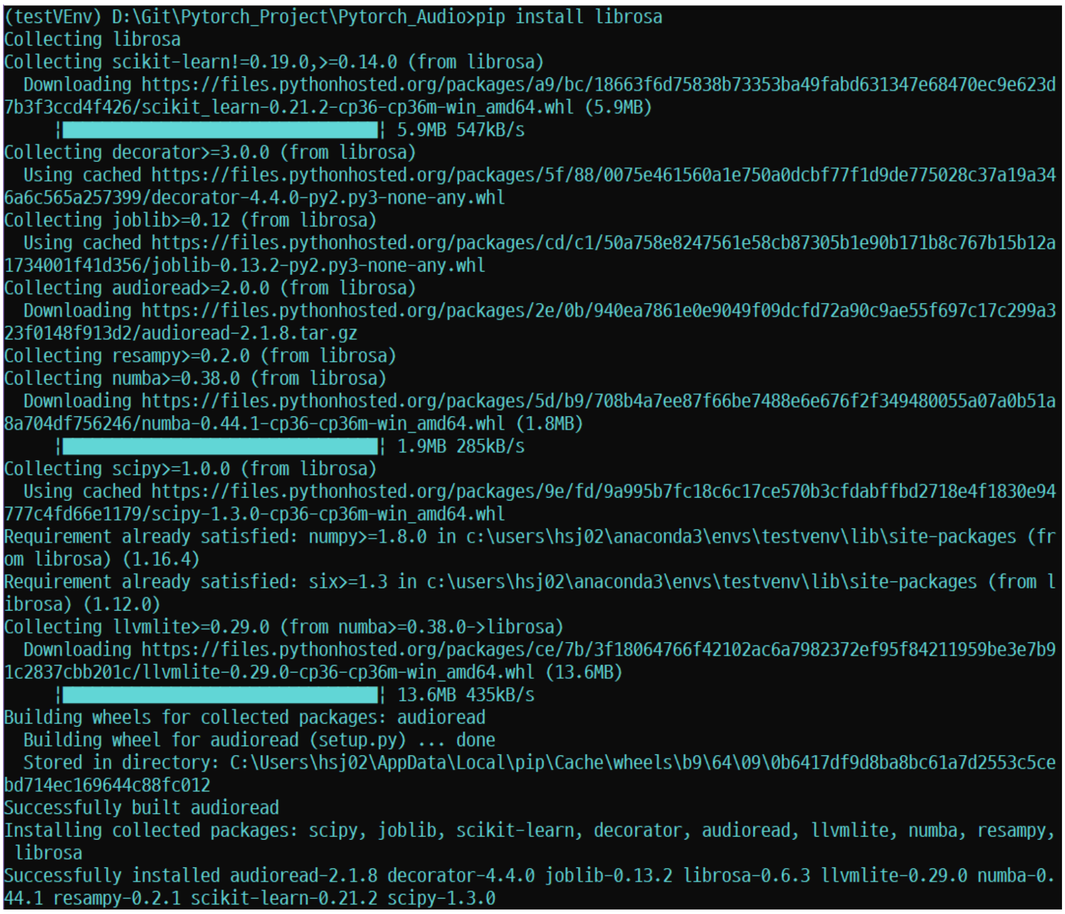
```
(base)C:\Users\(username)> conda activate (가상환경 이름)
```

`conda activate` 를 통해 설치 되어있는 Pytorch가 설치된 가상환경에 접속한다.

- **pip install**

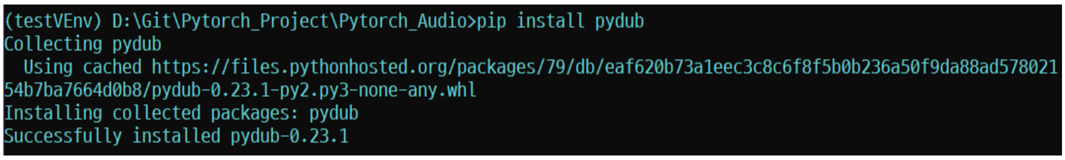
```
(testVenv)C:\Users\(username)> pip install Librosa Pydub PyAudio
```

[Librosa]



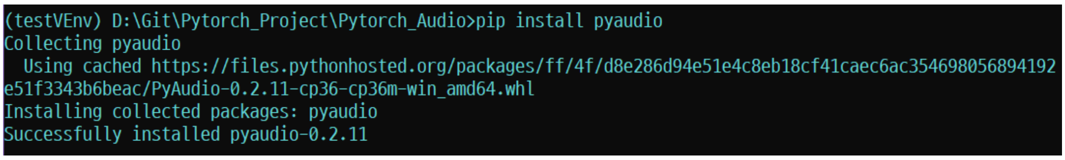
```
(testVEnv) D:\Git\Pytorch_Project\Pytorch_Audio>pip install librosa
Collecting librosa
Collecting scikit-learn!=0.19.0,>=0.14.0 (from librosa)
  Downloading https://files.pythonhosted.org/packages/a9/bc/18663f6d75838b73353ba49fabd631347e68470ec9e623d
7b3f3ccd4f426/scikit_learn-0.21.2-cp36-cp36m-win_amd64.whl (5.9MB)
     ¦████████████████████████████████¦ 5.9MB 547kB/s
Collecting decorator>=3.0.0 (from librosa)
  Using cached https://files.pythonhosted.org/packages/5f/88/0075e461560a1e750a0dcbf77f1d9de775028c37a19a34
6a6c565a257399/decorator-4.4.0-py2.py3-none-any.whl
Collecting joblib>=0.12 (from librosa)
  Using cached https://files.pythonhosted.org/packages/cd/c1/50a758e8247561e58cb87305b1e90b171b8c767b15b12a
1734001f41d356/joblib-0.13.2-py2.py3-none-any.whl
Collecting audioread>=2.0.0 (from librosa)
  Downloading https://files.pythonhosted.org/packages/2e/0b/940ea7861e0e9049f09dcfd72a90c9ae55f697c17c299a3
23f0148f913d2/audioread-2.1.8.tar.gz
Collecting resampy>=0.2.0 (from librosa)
Collecting numba>=0.38.0 (from librosa)
  Downloading https://files.pythonhosted.org/packages/5d/b9/708b4a7ee87f66be7488e6e676f2f349480055a07a0b51a
8a704df756246/numba-0.44.1-cp36-cp36m-win_amd64.whl (1.8MB)
     ¦████████████████████████████████¦ 1.9MB 285kB/s
Collecting scipy>=1.0.0 (from librosa)
  Using cached https://files.pythonhosted.org/packages/9e/fd/9a995b7fc18c6c17ce570b3cfdabffbd2718e4f1830e94
777c4fd66e1179/scipy-1.3.0-cp36-cp36m-win_amd64.whl
Requirement already satisfied: numpy>=1.8.0 in c:\users\hsj02\anaconda3\envs\testvenv\lib\site-packages (fr
om librosa) (1.16.4)
Requirement already satisfied: six>=1.3 in c:\users\hsj02\anaconda3\envs\testvenv\lib\site-packages (from l
ibrosa) (1.12.0)
Collecting llvmlite>=0.29.0 (from numba>=0.38.0->librosa)
  Downloading https://files.pythonhosted.org/packages/ce/7b/3f18064766f42102ac6a7982372ef95f84211959be3e7b9
1c2837cbb201c/llvmlite-0.29.0-cp36-cp36m-win_amd64.whl (13.6MB)
     ¦████████████████████████████████¦ 13.6MB 435kB/s
Building wheels for collected packages: audioread
  Building wheel for audioread (setup.py) ... done
  Stored in directory: C:\Users\hsj02\AppData\Local\pip\Cache\wheels\b9\64\09\0b6417df9d8ba8bc61a7d2553c5ce
bd714ec169644c88fc012
Successfully built audioread
Installing collected packages: scipy, joblib, scikit-learn, decorator, audioread, llvmlite, numba, resampy,
 librosa
Successfully installed audioread-2.1.8 decorator-4.4.0 joblib-0.13.2 librosa-0.6.3 llvmlite-0.29.0 numba-0.
44.1 resampy-0.2.1 scikit-learn-0.21.2 scipy-1.3.0
```

[Pydub]



```
(testVEnv) D:\Git\Pytorch_Project\Pytorch_Audio>pip install pydub
Collecting pydub
  Using cached https://files.pythonhosted.org/packages/79/db/eaf620b73a1eec3c8c6f8f5b0b236a50f9da88ad578021
54b7ba7664d0b8/pydub-0.23.1-py2.py3-none-any.whl
Installing collected packages: pydub
Successfully installed pydub-0.23.1
```

[PyAudio]

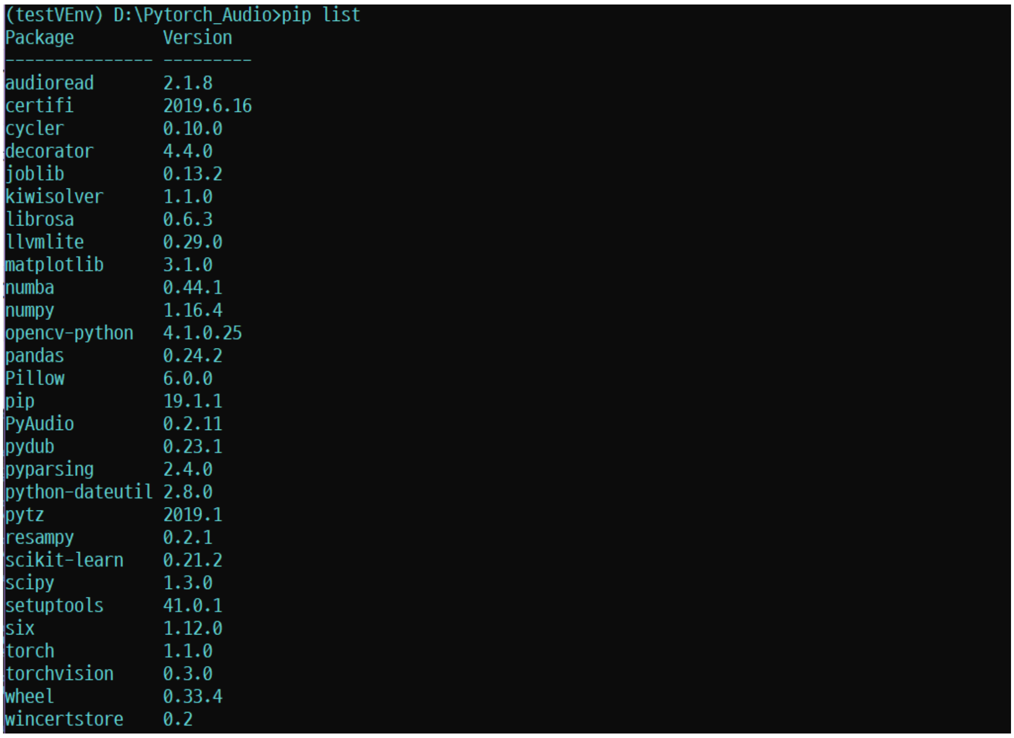


```
(testVEnv) D:\Git\Pytorch_Project\Pytorch_Audio>pip install pyaudio
Collecting pyaudio
  Using cached https://files.pythonhosted.org/packages/ff/4f/d8e286d94e51e4c8eb18cf41caec6ac354698056894192
e51f3343b6beac/PyAudio-0.2.11-cp36-cp36m-win_amd64.whl
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.11
```

- **설치확인**

```
(testVenv)d:\Pytorch_Audio> pip list
```

`pip list` 명령어를 통해 가상환경에 설치된 패키지를 확인할 수 있다.



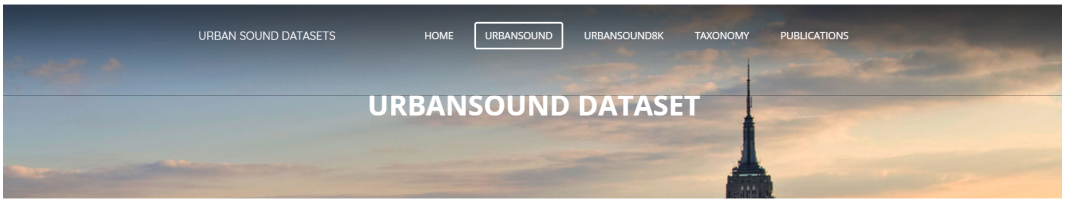
```
(testVEnv) D:\Pytorch_Audio>pip list
Package          Version
---------------- ---------
audioread        2.1.8
certifi          2019.6.16
cycler           0.10.0
decorator        4.4.0
joblib           0.13.2
kiwisolver       1.1.0
librosa          0.6.3
llvmlite         0.29.0
matplotlib       3.1.0
numba            0.44.1
numpy            1.16.4
opencv-python    4.1.0.25
pandas           0.24.2
Pillow           6.0.0
pip              19.1.1
PyAudio          0.2.11
pydub            0.23.1
pyparsing        2.4.0
python-dateutil  2.8.0
pytz             2019.1
resampy          0.2.1
scikit-learn     0.21.2
scipy            1.3.0
setuptools       41.0.1
six              1.12.0
torch            1.1.0
torchvision      0.3.0
wheel            0.33.4
wincertstore     0.2
```

필요한 라이브러리가 정상적으로 설치됨을 확인할 수 있다.

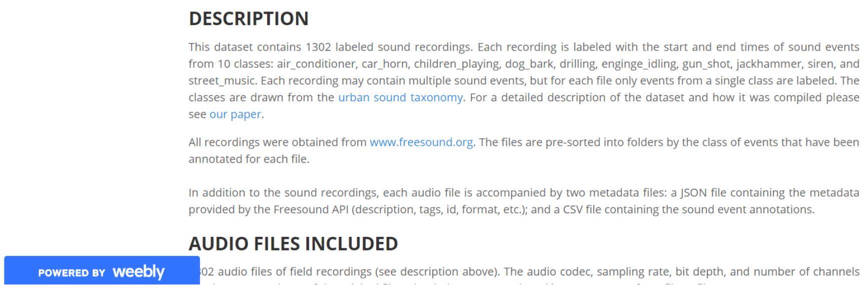
3. 데이터셋(UrbanSound) 다운로드

> 뉴욕 대학교 MARL(Music and Audio Research Lab)에서 2014 년에 공개한 UrbanSound는 10개의 클래스로 구성된 소리 데이터셋이다.

- **사이트 접속**
    - UrbanSound_DataSet



    -



DESCRIPTION

This dataset contains 1302 labeled sound recordings. Each recording is labeled with the start and end times of sound events from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music. Each recording may contain multiple sound events, but for each file only events from a single class are labeled. The classes are drawn from the urban sound taxonomy. For a detailed description of the dataset and how it was compiled please see our paper.

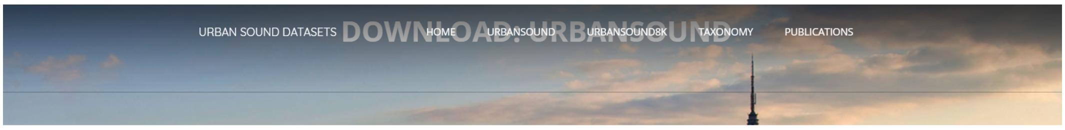
All recordings were obtained from www.freesound.org. The files are pre-sorted into folders by the class of events that have been annotated for each file.

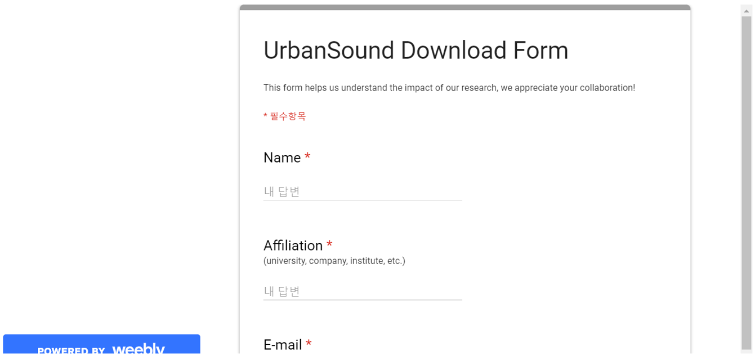
In addition to the sound recordings, each audio file is accompanied by two metadata files: a JSON file containing the metadata provided by the Freesound API (description, tags, id, format, etc.); and a CSV file containing the sound event annotations.

AUDIO FILES INCLUDED

02 audio files of field recordings (see description above). The audio codec, sampling rate, bit depth, and number of channels

- **신청서 작성**
    - **UrbanSound**를 선택한 뒤 **DownLoad**를 클릭한다.
    - 그러면 다운로드 폼이 뜨는데, 형식에 맞게 작성하면 된다.
    - 메일로 **다운로드 링크 파일이 전송**이 된다.



- 

  UrbanSound Download Form

  This form helps us understand the impact of our research, we appreciate your collaboration!

  * 필수항목

  Name *

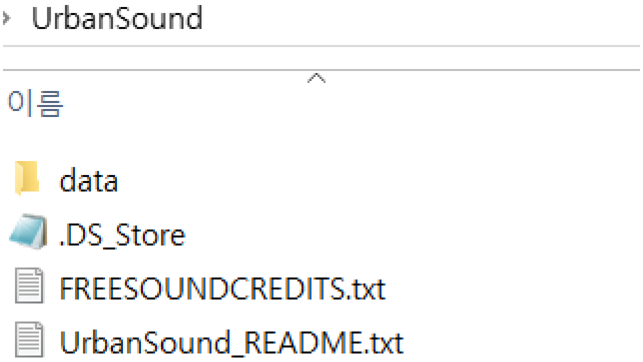
  내 답변

  Affiliation *
  (university, company, institute, etc.)

  내 답변

  E-mail *

- **다운로드**
  - 다운로드 후 **UrbanSound**폴더를 열어보면 다음과 같이 파일, 폴더가 구성 되어있다

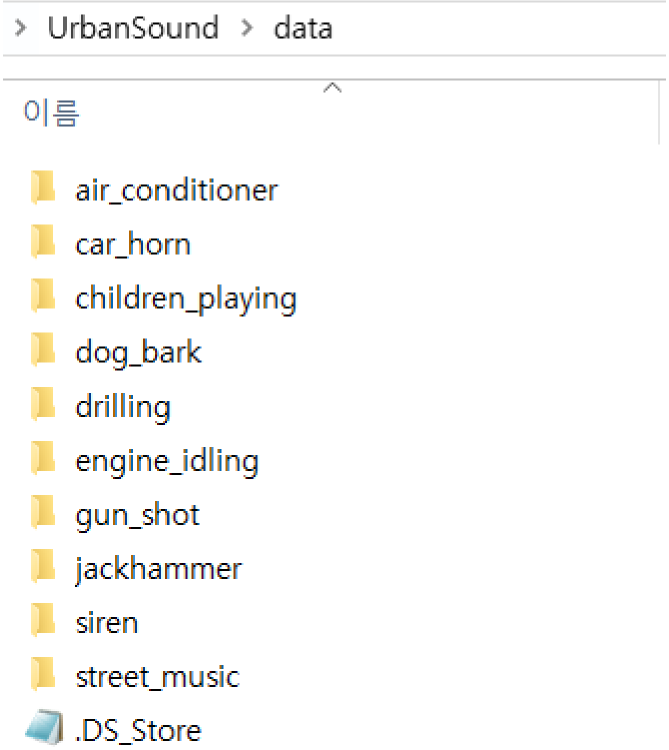
    

    UrbanSound

    이름

    data
    .DS_Store
    FREESOUNDCREDITS.txt
    UrbanSound_README.txt

  - 그 중 **data**폴더를 열어보면 다음과 같이 **Class**별로 구별되어 있다

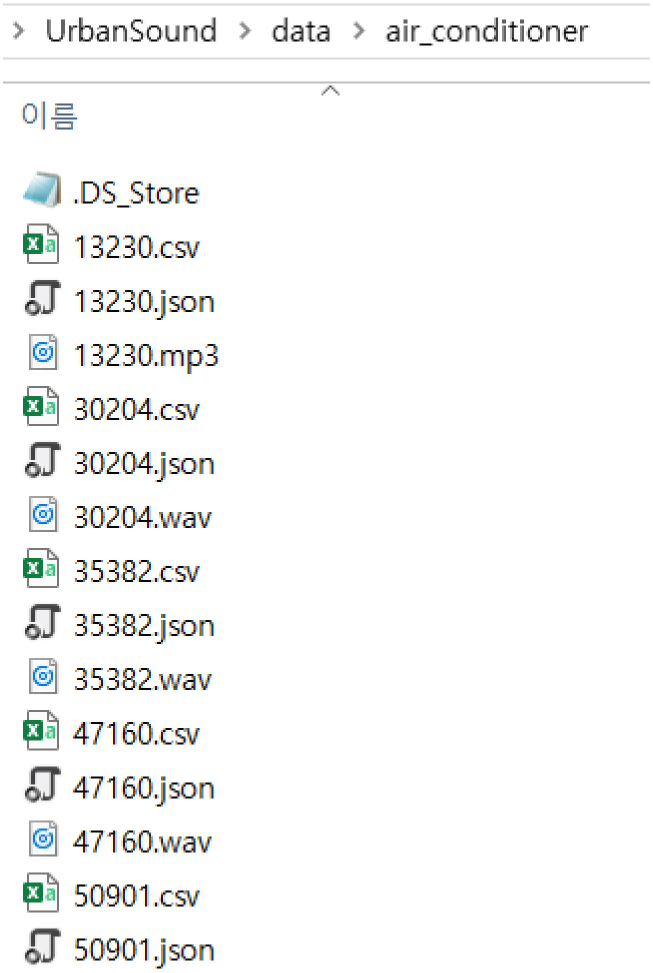
    

    UrbanSound > data

    이름

    air_conditioner
    car_horn
    children_playing
    dog_bark
    drilling
    engine_idling
    gun_shot
    jackhammer
    siren
    street_music
    .DS_Store

  - **air_conditioner**을 예시로 열어보니 다음과 같이 `.flac`, `.wav` 파일들이 담겨 있다.


> UrbanSound > data > air_conditioner

이름

.DS_Store
13230.csv
13230.json
13230.mp3
30204.csv
30204.json
30204.wav
35382.csv
35382.json
35382.wav
47160.csv
47160.json
47160.wav
50901.csv
50901.json

4. 전처리(1초 분할)

> 주어진 원데이터를 그대로 사용하면 불편하다. 그래서 원하는 형태로 변형해서 분석하는 경우가 많다. 따라서 분석에 용이하도록 데이터를 가공하는 작업을 **데이터 전처리**라고 한다.

- **Pydub**
- **소스 설명**
    - **전체 소스**(`audio_preprocess.py`)

```
import os
from pydub import AudioSegment
from pydub.generators import WhiteNoise

def main(args):
    urbansound_folder = args.urbansound_dir
    urbansound_dogbark_data_folder = urbansound_folder + os.sep + 'data/dog_bark'
    urbansound_graph_folder = urbansound_folder + os.sep + 'graph'
    urbansound_dogbark_graph_folder = urbansound_graph_folder + os.sep + 'positive'
    urbansound_other_graph_folder = urbansound_graph_folder + os.sep + 'negative'

    if not os.path.exists(urbansound_graph_folder):
        os.mkdir(urbansound_graph_folder)
    if not os.path.exists(urbansound_dogbark_graph_folder):
        os.mkdir(urbansound_dogbark_graph_folder)
    if not os.path.exists(urbansound_other_graph_folder):
        os.mkdir(urbansound_other_graph_folder)

    urbansound_other_data_folders = [urbansound_folder + os.sep + 'data/air_conditioner',
                                     urbansound_folder + os.sep + 'data/car_horn', \
                                     urbansound_folder + os.sep + 'data/children_playing',
                                     urbansound_folder + os.sep + 'data/drilling', \
                                     urbansound_folder + os.sep + 'data/engine_idling',
                                     urbansound_folder + os.sep + 'data/gun_shot', \
                                     urbansound_folder + os.sep + 'data/jackhammer',
                                     urbansound_folder + os.sep + 'data/siren', \
                                     urbansound_folder + os.sep + 'data/street_music']

    SECOND_MS = 500         #1000
    SEGMENT_MS = 500        #2000
    ASSIGNED_SAMPLERATE = 44100
    ESC50_AUDIO_START_POS = 500
    POSITIVE_SAMPLE_DB_TH = -40.0


    print('creating positive training set ..')
    idx = 0

    for file in os.listdir(urbansound_dogbark_data_folder):
        filename, extension = os.path.splitext(file)
        if extension == '.wav' or extension == '.ogg' or extension == '.mp3' or extension == '.flac' or extension == '.aif' or extension == '.aiff':
            # open sound file
            audiopath = urbansound_dogbark_data_folder + os.sep + file
            print(audiopath)
            audio = AudioSegment.from_file(audiopath).set_frame_rate(ASSIGNED_SAMPLERATE).set_channels(1).set_sample_width(2)[:]
            # open csv file
            csvpath = urbansound_dogbark_data_folder + os.sep + filename + '.csv'
            csv = open(csvpath, 'r')
            lines = csv.readlines()
            for line in lines:
                start = float(line.split(',')[0]) * SECOND_MS
                end = float(line.split(',')[1]) * SECOND_MS
                chunk1 = (end - start) / 10
                current = start
                while 1:
                    outfile = urbansound_dogbark_graph_folder + os.sep + str(idx) + '_dogbark.wav'
                    idx += 1
                    audioclip = audio[current:current + SEGMENT_MS]
                    if len(audioclip) != SEGMENT_MS:
                        lack = SEGMENT_MS - len(audioclip) + 100  # 100 for default crossfade
                        noiseclip = WhiteNoise().to_audio_segment(duration=lack, volume=-50)
                        lastclip = audioclip.append(noiseclip)
                        if lastclip.dBFS > POSITIVE_SAMPLE_DB_TH:
                            lastclip.export(outfile, format='wav')
                        break
                    else:
                        if audioclip.dBFS > POSITIVE_SAMPLE_DB_TH:
                            audioclip.export(outfile, format='wav')
                    current += SEGMENT_MS
                    chunk2 = end - current
                    if chunk2 < chunk1:
                        break
                # if current > end:
                # break
            csv.close()


    print ('creating negative training set ..')
    idx = 0
    for other_data_folder in urbansound_other_data_folders:
        for file in os.listdir(other_data_folder):
            filename, extension = os.path.splitext(file)
            if extension == '.wav' or extension == '.ogg' or extension == '.mp3' or extension == '.flac' or extension == '.aif' or extension == '.aiff':
                # open sound file
                audiopath = other_data_folder + os.sep + file
                print(audiopath)
                try:
                    audio = AudioSegment.from_file(audiopath).set_frame_rate(ASSIGNED_SAMPLERATE).set_channels(1).set_sample_width(2)[:]
                    num_segment = len(audio) // SEGMENT_MS
                    for i in range(0, num_segment):
                        if i % 4 == 0:  # less sample :)
                            outfile = urbansound_other_graph_folder + os.sep + str(idx) + '_other.wav'
                            idx += 1
                            audio[i * SEGMENT_MS: (i + 1) * SEGMENT_MS].export(outfile, format='wav')
                except:
                    print('failed to load this one ^^^^^')


if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser()
    parser.add_argument('--urbansound_dir', '-u', dest='urbansound_dir', required=True)
    args = parser.parse_args()
    main(args)

```

`audio_preprocess.py`은 크게 4개의 기능을 한다.

1. 폴더 생성
2. DogSound 1초 분할
3. OtherSound 1초 분할
4. argparse(명령행)

- **폴더 생성**

```
urbansound_folder = args.urbansound_dir
    urbansound_dogbark_data_folder = urbansound_folder + os.sep + 'data/dog_bark'
    urbansound_graph_folder = urbansound_folder + os.sep + 'graph'
    urbansound_dogbark_graph_folder = urbansound_graph_folder + os.sep + 'positive'
    urbansound_other_graph_folder = urbansound_graph_folder + os.sep + 'negative'

    if not os.path.exists(urbansound_graph_folder):
        os.mkdir(urbansound_graph_folder)
    if not os.path.exists(urbansound_dogbark_graph_folder):
        os.mkdir(urbansound_dogbark_graph_folder)
    if not os.path.exists(urbansound_other_graph_folder):
        os.mkdir(urbansound_other_graph_folder)

    urbansound_other_data_folders = [urbansound_folder + os.sep + 'data/air_conditioner',
                                     urbansound_folder + os.sep + 'data/car_horn', \
                                     urbansound_folder + os.sep + 'data/children_playing',
                                     urbansound_folder + os.sep + 'data/drilling', \
                                     urbansound_folder + os.sep + 'data/engine_idling',
                                     urbansound_folder + os.sep + 'data/gun_shot', \
                                     urbansound_folder + os.sep + 'data/jackhammer',
                                     urbansound_folder + os.sep + 'data/siren', \
                                     urbansound_folder + os.sep + 'data/street_music']
```

`os`라이브러리를 통해 저장소에 접근하고, `os.path.exist`를 사용하여 그 폴더의 존재 여부를 알 수 있다.

만약 폴더가 존재하지 않는다면 `os.mkdir`을 사용하여 폴더를 생성해준다.

- **DogSound**

```python
 print('creating positive training set ..')
    idx = 0

    for file in os.listdir(urbansound_dogbark_data_folder):
        filename, extension = os.path.splitext(file)
        if extension == '.wav' or extension == '.ogg' or extension == '.mp3' or extension == '.flac' or extension == '.aif' or extension == '.aiff':
            # open sound file
            audiopath = urbansound_dogbark_data_folder + os.sep + file
            print(audiopath)
            audio = AudioSegment.from_file(audiopath).set_frame_rate(ASSIGNED_SAMPLERATE).set_channels(1).set_sample_width(2)[:]
            # open csv file
            csvpath = urbansound_dogbark_data_folder + os.sep + filename + '.csv'
            csv = open(csvpath, 'r')
            lines = csv.readlines()
            for line in lines:
                start = float(line.split(',')[0]) * SECOND_MS
                end = float(line.split(',')[1]) * SECOND_MS
                chunk1 = (end - start) / 10
                current = start
                while 1:
                    outfile = urbansound_dogbark_graph_folder + os.sep + str(idx) + '_dogbark.wav'
                    idx += 1
                    audioclip = audio[current:current + SEGMENT_MS]
                    if len(audioclip) != SEGMENT_MS:
                        lack = SEGMENT_MS - len(audioclip) + 100  # 100 for default crossfade
                        noiseclip = WhiteNoise().to_audio_segment(duration=lack, volume=-50)
                        lastclip = audioclip.append(noiseclip)
                        if lastclip.dBFS > POSITIVE_SAMPLE_DB_TH:
                            lastclip.export(outfile, format='wav')
                        break
                    else:
                        if audioclip.dBFS > POSITIVE_SAMPLE_DB_TH:
                            audioclip.export(outfile, format='wav')
                    current += SEGMENT_MS
                    chunk2 = end - current
                    if chunk2 < chunk1:
                        break
                # if current > end:
                # break
            csv.close()
```

`UrbanSound` 폴더 중 `/Dog_bark` 폴더의 파일들에 접근한다.

소리파일과 소리 위치가 담긴 `csv` 파일을 통해 1초단위의 소리파일들을 생성한다.

그 파일들은 `urbansound/graph/positive` 에 저장된다.

- ### OtherSound

```
 print ('creating negative training set ..')
    idx = 0
    for other_data_folder in urbansound_other_data_folders:
        for file in os.listdir(other_data_folder):
            filename, extension = os.path.splitext(file)
            if extension == '.wav' or extension == '.ogg' or extension == '.mp3' or extension == '.flac' or extension == '.aif' or extension == '.aiff':
                # open sound file
                audiopath = other_data_folder + os.sep + file
                print(audiopath)
                try:
                    audio = AudioSegment.from_file(audiopath).set_frame_rate(ASSIGNED_SAMPLERATE).set_channels(1).set_sample_width(2)[:]
                    num_segment = len(audio) // SEGMENT_MS
                    for i in range(0, num_segment):
                        if i % 4 == 0:  # less sample :)
                            outfile = urbansound_other_graph_folder + os.sep + str(idx) + '_other.wav'
                            idx += 1
                            audio[i * SEGMENT_MS: (i + 1) * SEGMENT_MS].export(outfile, format='wav')
                except:
                    print('failed to load this one ^^^^^')
```

DogSound와 유사하지만 다른 class들을 모두 `OtherSound`로 분류한다는 것이 차이점이다. `csv`파일에 접근하지 않고 소리파일의 첫 1초만을 저장한다.

그 파일들은 `urbansound/graph/negative`에 저장된다.

- ### argparse

```
import argparse
    parser = argparse.ArgumentParser()
    parser.add_argument('--urbansound_dir', '-u', dest='urbansound_dir', required=True)
    args = parser.parse_args()
    main(args)
```

`argparse`는 명령줄 파싱 라이브러리이다.

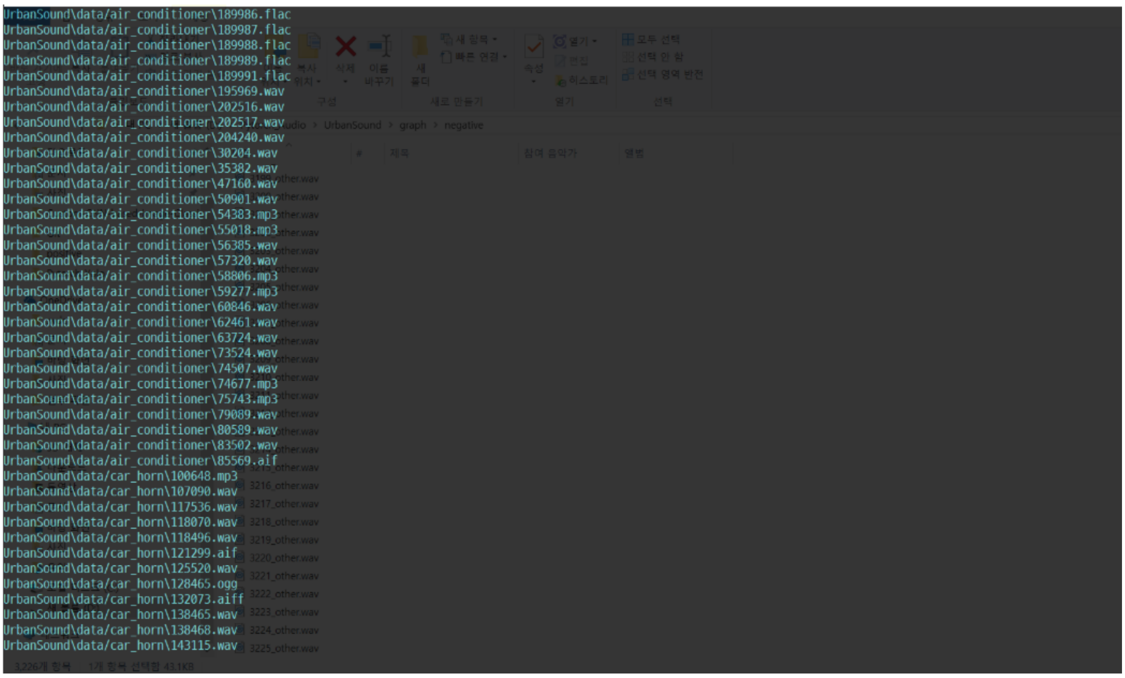
```
(testVenv)d:\Pytorch_Audio>python audio_preprocess --urbansound_dir UrbanSound
```

위와 같이 명령어를 입력하면, `--urbansound_dir`뒤의 문자열이 파싱되어 파일경로로 입력된다.

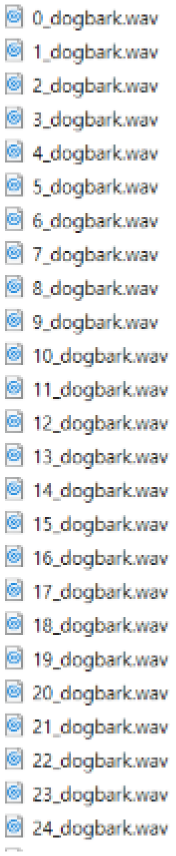
- ## 실행 결과

```
(testVenv)d:\Pytorch_Audio>python audio_preprocess.py --urbansound_dir
UrbanSound
```

위의 명령어를 입력하면 다음과 같이 전처리가 진행된다.



```
UrbanSound\data/air_conditioner\189986.flac
UrbanSound\data/air_conditioner\189987.flac
UrbanSound\data/air_conditioner\189988.flac
UrbanSound\data/air_conditioner\189989.flac
UrbanSound\data/air_conditioner\189991.flac
UrbanSound\data/air_conditioner\195969.wav
UrbanSound\data/air_conditioner\202516.wav
UrbanSound\data/air_conditioner\202517.wav
UrbanSound\data/air_conditioner\204240.wav
UrbanSound\data/air_conditioner\30204.wav
UrbanSound\data/air_conditioner\35382.wav
UrbanSound\data/air_conditioner\47160.wav
UrbanSound\data/air_conditioner\50901.wav
UrbanSound\data/air_conditioner\54383.mp3
UrbanSound\data/air_conditioner\55018.mp3
UrbanSound\data/air_conditioner\56385.wav
UrbanSound\data/air_conditioner\57320.wav
UrbanSound\data/air_conditioner\58806.mp3
UrbanSound\data/air_conditioner\59277.mp3
UrbanSound\data/air_conditioner\60846.wav
UrbanSound\data/air_conditioner\62461.wav
UrbanSound\data/air_conditioner\63724.wav
UrbanSound\data/air_conditioner\73524.wav
UrbanSound\data/air_conditioner\74507.wav
UrbanSound\data/air_conditioner\74677.mp3
UrbanSound\data/air_conditioner\75743.mp3
UrbanSound\data/air_conditioner\79089.wav
UrbanSound\data/air_conditioner\80589.wav
UrbanSound\data/air_conditioner\83502.wav
UrbanSound\data/air_conditioner\85569.aif
UrbanSound\data/car_horn\100648.mp3
UrbanSound\data/car_horn\107090.wav
UrbanSound\data/car_horn\117536.wav
UrbanSound\data/car_horn\118070.wav
UrbanSound\data/car_horn\118496.wav
UrbanSound\data/car_horn\121299.aif
UrbanSound\data/car_horn\125520.wav
UrbanSound\data/car_horn\128465.ogg
UrbanSound\data/car_horn\132073.aiff
UrbanSound\data/car_horn\138465.wav
UrbanSound\data/car_horn\138468.wav
UrbanSound\data/car_horn\143115.wav
```



0_dogbark.wav
1_dogbark.wav
2_dogbark.wav
3_dogbark.wav
4_dogbark.wav
5_dogbark.wav
6_dogbark.wav
7_dogbark.wav
8_dogbark.wav
9_dogbark.wav
10_dogbark.wav
11_dogbark.wav
12_dogbark.wav
13_dogbark.wav
14_dogbark.wav
15_dogbark.wav
16_dogbark.wav
17_dogbark.wav
18_dogbark.wav
19_dogbark.wav
20_dogbark.wav
21_dogbark.wav
22_dogbark.wav
23_dogbark.wav
24_dogbark.wav

5. 전처리(Mfcc)

- **Librosa**

  > **Librosa**는 음성 특징 추출 라이브러리로, 소리의 특징을 추출하는 것은 물론, 스펙트로그램 그래프로 변환할 수 있다.

- **Mfcc**

> **Mfcc**(Mel Frequency Cepstral Coefficient)는 **일정 구간**(Short time)식 나누어, 이 구간에 대한 **스펙트럼을 분석**하여 특징을 추출하는 기법이다.

- **소스 설명**
    - 전체 소스(`audio_mfcc.py`)

```python
import os
import librosa
import librosa.display
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab

def main(args):
    urbansound_folder = args.urbansound_dir
    urbansound_graph_folder = urbansound_folder + os.sep + 'graph'
    urbansound_graph_mfcc_folder = urbansound_folder + os.sep + 'graph_mfcc'
    urbansound_dogbark_graph_folder = urbansound_graph_folder + os.sep + 'positive'
    urbansound_other_graph_folder = urbansound_graph_folder + os.sep + 'negative'
    urbansound_dogbark_graph_mfcc_folder = urbansound_graph_mfcc_folder + os.sep + 'positive'
    urbansound_other_graph_mfcc_folder = urbansound_graph_mfcc_folder + os.sep + 'negative'

    if not os.path.exists(urbansound_graph_mfcc_folder):
        os.mkdir(urbansound_graph_mfcc_folder)
    if not os.path.exists(urbansound_dogbark_graph_mfcc_folder):
        os.mkdir(urbansound_dogbark_graph_mfcc_folder)
    if not os.path.exists(urbansound_other_graph_mfcc_folder):
        os.mkdir(urbansound_other_graph_mfcc_folder)


    for file in os.listdir(urbansound_dogbark_graph_folder):
        filename, extension = os.path.splitext(file)

        if extension == '.wav':
            # open sound file
            audiopath = urbansound_dogbark_graph_folder + os.sep + file
            print(audiopath)

            y, sr = librosa.load(audiopath)
            S = librosa.feature.melspectrogram(y, sr=sr, n_mels=128)

            log_S = librosa.amplitude_to_db(S, ref=np.max)
            fig = plt.figure(figsize=(12, 4))
            librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

            plt.title('mel power spectrogram')
            # plt.colorbar(format = '%+02.0f db')
            plt.tight_layout()

            plt.axis('off')
            plt.xticks([]), plt.yticks([])
            plt.subplots_adjust(left=0, bottom=0, right=1, top=1, hspace=0, wspace=0)

            plt.savefig(urbansound_dogbark_graph_mfcc_folder + '/' + filename + '.png')
            plt.close(fig)

    for file in os.listdir(urbansound_other_graph_folder):
        filename, extension = os.path.splitext(file)
        if extension == '.wav':
            # open sound file
            audiopath = urbansound_other_graph_folder + os.sep + file
            print(audiopath)

            y, sr = librosa.load(audiopath)
            S = librosa.feature.melspectrogram(y, sr=sr, n_mels=128)

            log_S = librosa.amplitude_to_db(S, ref=np.max)
            fig = plt.figure(figsize=(12, 4))
            librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

            plt.title('mel power spectrogram')
            # plt.colorbar(format = '%+02.0f db')
            plt.tight_layout()

            plt.axis('off')
            plt.xticks([]), plt.yticks([])
            plt.subplots_adjust(left=0, bottom=0, right=1, top=1, hspace=0, wspace=0)

            plt.savefig(urbansound_other_graph_mfcc_folder + '/' + filename + '.png')
            plt.close(fig)

if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser()
    parser.add_argument('--urbansound_dir', '-u', dest='urbansound_dir', required=True)
    args = parser.parse_args()
    main(args)


```

`audio_mfcc.py` 은 크게 4개의 기능을 한다.

1. 폴더 생성
2. DogSound 시각화 이미지 저장
3. OtherSound 시각화 이미지 저장

- **폴더 생성**

```python
urbansound_folder = args.urbansound_dir
    urbansound_graph_folder = urbansound_folder + os.sep + 'graph'
    urbansound_graph_mfcc_folder = urbansound_folder + os.sep + 'graph_mfcc'
    urbansound_dogbark_graph_folder = urbansound_graph_folder + os.sep + 'positive'
    urbansound_other_graph_folder = urbansound_graph_folder + os.sep + 'negative'
    urbansound_dogbark_graph_mfcc_folder = urbansound_graph_mfcc_folder + os.sep + 'positive'
    urbansound_other_graph_mfcc_folder = urbansound_graph_mfcc_folder + os.sep + 'negative'

    if not os.path.exists(urbansound_graph_mfcc_folder):
        os.mkdir(urbansound_graph_mfcc_folder)
    if not os.path.exists(urbansound_dogbark_graph_mfcc_folder):
        os.mkdir(urbansound_dogbark_graph_mfcc_folder)
    if not os.path.exists(urbansound_other_graph_mfcc_folder):
        os.mkdir(urbansound_other_graph_mfcc_folder)
```

`audio_preprocess.py`와 유사하다.

`os` 라이브러리를 통해 저장소에 접근하고, `os.path.exist`를 사용하여 그 폴더의 존재 여부를 알 수 있다.

만약 폴더가 존재하지 않는다면 `os.mkdir`을 사용하여 폴더를 생성해준다.

- **DogSound 시각화 이미지 저장**

```python
 for file in os.listdir(urbansound_dogbark_graph_folder):
        filename, extension = os.path.splitext(file)

        if extension == '.wav':
            # open sound file
            audiopath = urbansound_dogbark_graph_folder + os.sep + file
            print(audiopath)

            y, sr = librosa.load(audiopath)
            S = librosa.feature.melspectrogram(y, sr=sr, n_mels=128)

            log_S = librosa.amplitude_to_db(S, ref=np.max)
            fig = plt.figure(figsize=(12, 4))
            librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

            plt.title('mel power spectrogram')
            # plt.colorbar(format = '%+02.0f db')
            plt.tight_layout()

            plt.axis('off')
            plt.xticks([]), plt.yticks([])
            plt.subplots_adjust(left=0, bottom=0, right=1, top=1, hspace=0, wspace=0)

            plt.savefig(urbansound_dogbark_graph_mfcc_folder + '/' + filename + '.png')
            plt.close(fig)
```

1초 단위로 전처리된 `.wav` 파일들을 `librosa` 라이브러리를 이용해 분석한다.

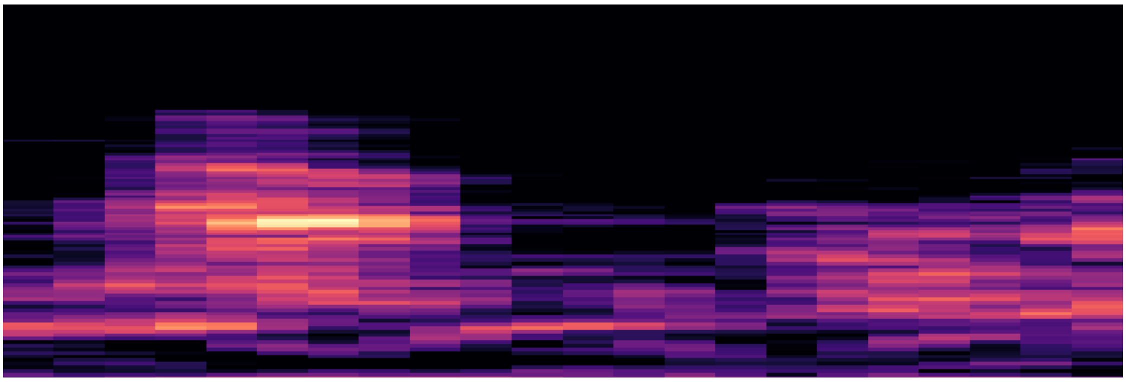
`librosa.feature.melspectrogram` 으로 그래프 변환하고,

`matplotlib` 를 사용해 시각화한다.

그 후 `.savefig` 를 통해 폴더에 `.png` 파일로 저장한다.

특히 `plt.subplots_adjust(left=0, bottom=0, right=1, top=1, hspace=0, wspace=0)` 부분은 이미지에서 불필요한 부분을 제거하여 오직 스펙트럼이미지만 나오도록 하는 것이다.

```
###### [Example Image]
```

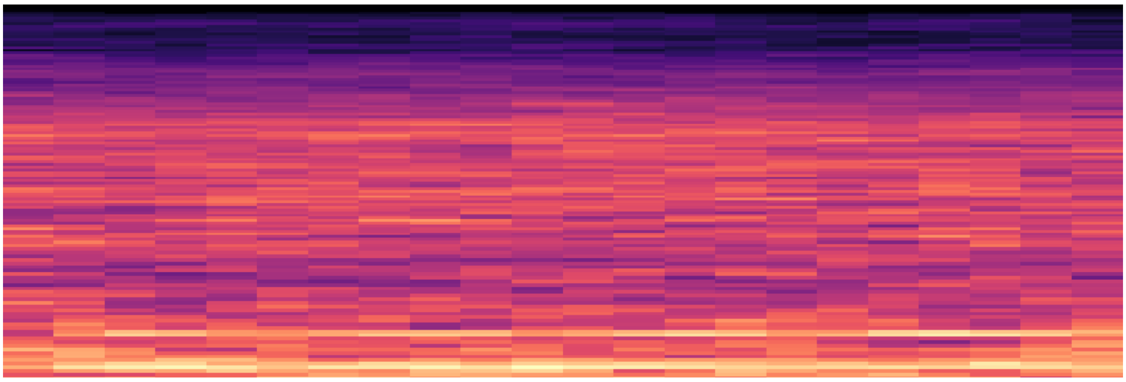


- **OtherSound 시각화 이미지 저장**

```
for file in os.listdir(urbansound_other_graph_folder):
        filename, extension = os.path.splitext(file)
        if extension == '.wav':
            # open sound file
            audiopath = urbansound_other_graph_folder + os.sep + file
            print(audiopath)

            y, sr = librosa.load(audiopath)
            S = librosa.feature.melspectrogram(y, sr=sr, n_mels=128)

            log_S = librosa.amplitude_to_db(S, ref=np.max)
            fig = plt.figure(figsize=(12, 4))
            librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

            plt.title('mel power spectrogram')
            # plt.colorbar(format = '%+02.0f db')
            plt.tight_layout()

            plt.axis('off')
            plt.xticks([]), plt.yticks([])
            plt.subplots_adjust(left=0, bottom=0, right=1, top=1, hspace=0, wspace=0)

            plt.savefig(urbansound_other_graph_mfcc_folder + '/' + filename + '.png')
            plt.close(fig)
```

위의 `DogSound 시각화 이미지 저장`과 유사하다.

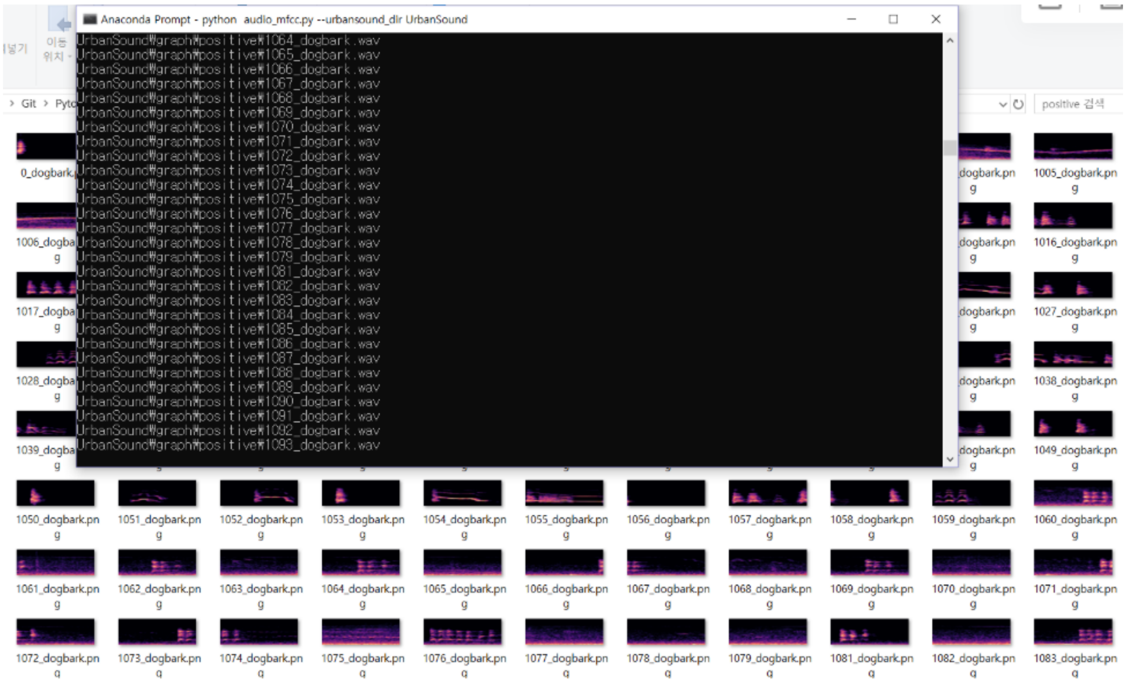
[Example Image]



- **실행 결과**

```
(testVenv)d:\Pytorch_Audio>python audio_mfcc.py --urbansound_dir UrbanSound
```

위의 명령어를 입력하면 다음과 같이 **전처리**가 동작된다.



Anaconda Prompt - python audio_mfcc.py --urbansound_dir UrbanSound

```
UrbanSound₩graph₩positive₩1064_dogbark.wav
UrbanSound₩graph₩positive₩1065_dogbark.wav
UrbanSound₩graph₩positive₩1066_dogbark.wav
UrbanSound₩graph₩positive₩1067_dogbark.wav
UrbanSound₩graph₩positive₩1068_dogbark.wav
UrbanSound₩graph₩positive₩1069_dogbark.wav
UrbanSound₩graph₩positive₩1070_dogbark.wav
UrbanSound₩graph₩positive₩1071_dogbark.wav
UrbanSound₩graph₩positive₩1072_dogbark.wav
UrbanSound₩graph₩positive₩1073_dogbark.wav
UrbanSound₩graph₩positive₩1074_dogbark.wav
UrbanSound₩graph₩positive₩1075_dogbark.wav
UrbanSound₩graph₩positive₩1076_dogbark.wav
UrbanSound₩graph₩positive₩1077_dogbark.wav
UrbanSound₩graph₩positive₩1078_dogbark.wav
UrbanSound₩graph₩positive₩1079_dogbark.wav
UrbanSound₩graph₩positive₩1081_dogbark.wav
UrbanSound₩graph₩positive₩1082_dogbark.wav
UrbanSound₩graph₩positive₩1083_dogbark.wav
UrbanSound₩graph₩positive₩1084_dogbark.wav
UrbanSound₩graph₩positive₩1085_dogbark.wav
UrbanSound₩graph₩positive₩1086_dogbark.wav
UrbanSound₩graph₩positive₩1087_dogbark.wav
UrbanSound₩graph₩positive₩1088_dogbark.wav
UrbanSound₩graph₩positive₩1089_dogbark.wav
UrbanSound₩graph₩positive₩1090_dogbark.wav
UrbanSound₩graph₩positive₩1091_dogbark.wav
UrbanSound₩graph₩positive₩1092_dogbark.wav
UrbanSound₩graph₩positive₩1093_dogbark.wav
```