LAB-Audio-01: PyTorch Audio Processing 과제 보고서

개요

본 과제는 PyTorch Audio 라이브러리를 활용하여 sample audio dataset을 사용한 waveform과 spectrogram 출력, 그리고 Downsampling과 Upsampling을 수행하는 것을 목표로 한다.

과제 요구사항 달성

1. **sample audio dataset을 사용하여 waveform과 spectrogram 출력** ✓
2. **Downsampling and Upsampling 수행** ✓

데이터셋

- **파일 수**: 4개의 m4a 파일
- **파일명**: 09552039.m4a, 09552040.m4a, 09552041.m4a, 09552042.m4a
- **형식**: M4A (MPEG-4 Audio)

실습 과정 및 결과

1단계: 환경 설정 및 라이브러리 설치

필요한 라이브러리인 torchaudio, matplotlib, librosa를 설치하고 임포트를 완료했다.

```
!pip install torchaudio
!pip install matplotlib
!pip install librosa

import torch
import torchaudio
import matplotlib.pyplot as plt
import numpy as np
import os
import zipfile
```

2단계: 데이터셋 준비

구글 드라이브에서 압축된 sample audio dataset을 다운로드하고 압축을 해제했다. 총 4개의 m4a 파일이 성공적으로 추출되었다.

```
from google.colab import drive
drive.mount('/content/drive')

zip_path = '/content/drive/MyDrive/Colab Notebooks/audio_dataset.zip'
extract_path = '/content/audio_data'

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

m4a_files = [f for f in os.listdir(extract_path) if f.endswith('.m4a')]
```

결과: 4개 m4a 파일 성공적으로 로드됨

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
압축 해제된 파일들: ['095522039.m4a', '095522042.m4a', '095522041.m4a', '095522040.m4a']
m4a 파일들 (4개): ['095522039.m4a', '095522042.m4a', '095522041.m4a', '095522040.m4a']
```

3단계: 오디오 파일 정보 분석

각 오디오 파일의 기본 정보를 확인했다. `torchaudio.load()` 함수를 사용하여 파형 데이터와 샘플링 레이트를 추출했다.

```
for i, audio_file in enumerate(m4a_files, 1):
    file_path = os.path.join(extract_path, audio_file)
    waveform, sample_rate = torchaudio.load(file_path)
    print(f"{i}. 파일: {audio_file}")
    print(f"   파형 크기: {waveform.shape}")
    print(f"   샘플링 레이트: {sample_rate} Hz")
    print(f"   재생 시간: {waveform.shape[1] / sample_rate:.2f}초")
```

결과: 모든 파일의 메타데이터 성공적으로 추출됨

```
=== 오디오 파일 정보 ===
1. 파일: 095522039.m4a
   파형 크기: torch.Size([1, 177152])
   샘플링 레이트: 44100 Hz
   재생 시간: 4.02초
------------------------------------------------
2. 파일: 095522042.m4a
   파형 크기: torch.Size([1, 177152])
   샘플링 레이트: 44100 Hz
   재생 시간: 4.02초
------------------------------------------------
3. 파일: 095522041.m4a
   파형 크기: torch.Size([1, 177152])
   샘플링 레이트: 44100 Hz
   재생 시간: 4.02초
------------------------------------------------
4. 파일: 095522040.m4a
   파형 크기: torch.Size([1, 177152])
   샘플링 레이트: 44100 Hz
   재생 시간: 4.02초
------------------------------------------------
```

4단계: 파형 시각화 함수 정의

시간 도메인에서의 오디오 신호를 시각화하기 위한 함수를 구현했다.

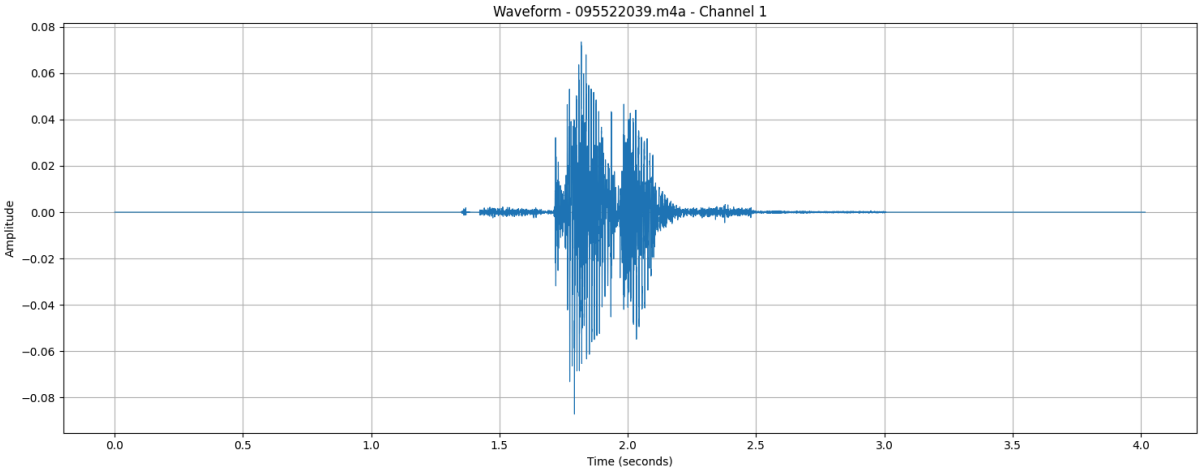
```
def plot_waveform(waveform, sample_rate, title="Waveform"):
    waveform = waveform.numpy()
    num_channels, num_frames = waveform.shape
    time_axis = torch.arange(0, num_frames) / sample_rate

    plt.figure(figsize=(15, 6))
    for c in range(num_channels):
        plt.subplot(num_channels, 1, c + 1)
        plt.plot(time_axis, waveform[c], linewidth=0.8)
        plt.title(f"{title} - Channel {c + 1}")
        plt.xlabel('Time (seconds)')
        plt.ylabel('Amplitude')
        plt.grid(True)
    plt.tight_layout()
    plt.show()
```

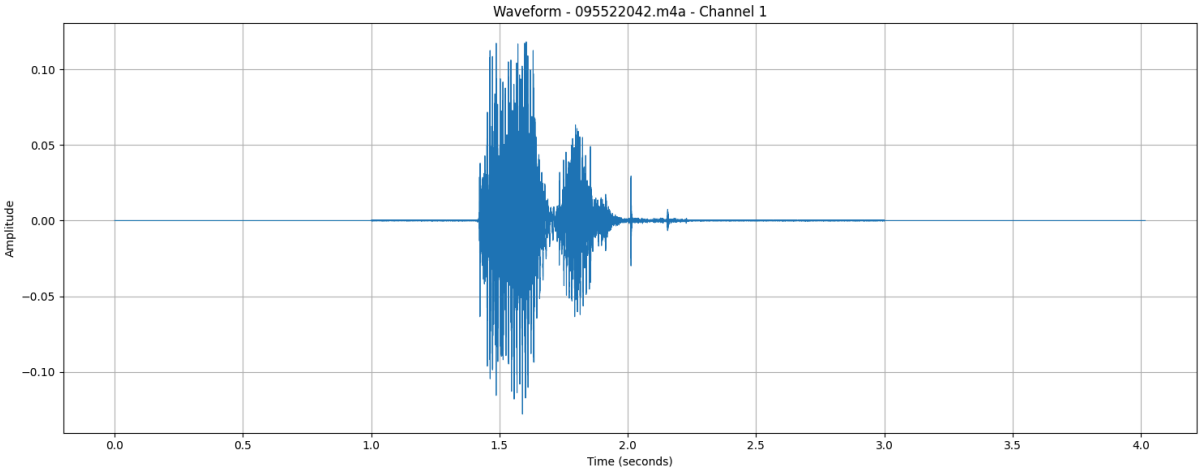
5단계: Waveform 출력 (요구사항 1 달성)

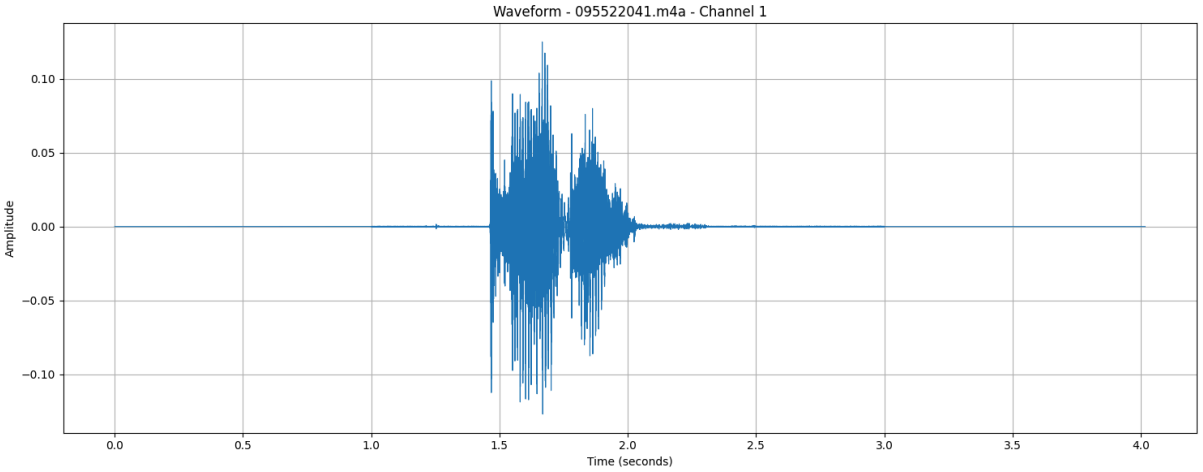
sample audio dataset의 각 파일에 대해 waveform을 성공적으로 출력했다.

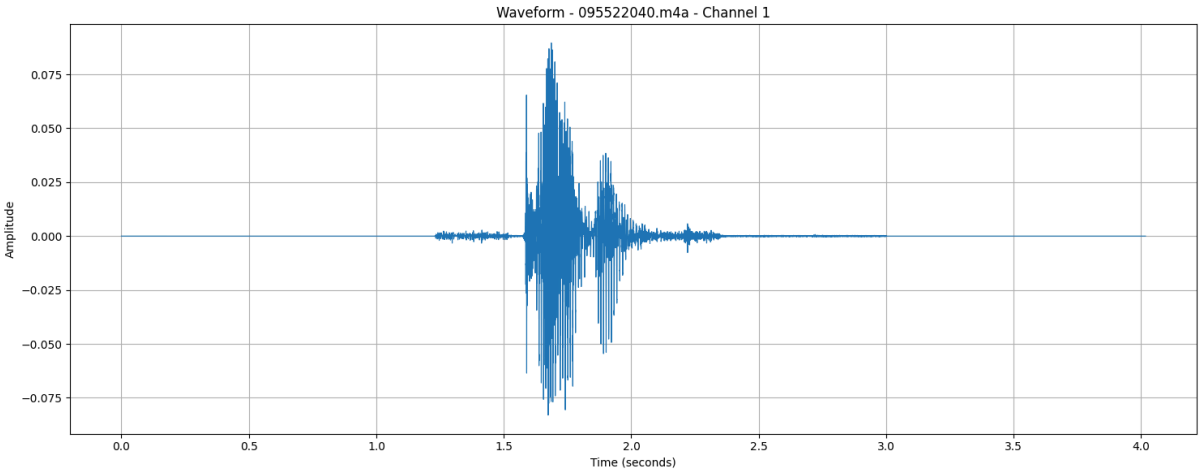
```
for audio_file in m4a_files:
    file_path = os.path.join(extract_path, audio_file)
    waveform, sample_rate = torchaudio.load(file_path)
    plot_waveform(waveform, sample_rate, f"Waveform - {audio_file}")
```

결과: 4개 파일의 waveform 시각화 완료









6단계: 스펙트로그램 시각화 함수 정의

주파수 도메인 분석을 위한 스펙트로그램 생성 함수를 구현했다.

```
def plot_spectrogram(waveform, sample_rate, title="Spectrogram"):
    waveform = waveform.numpy()

    if waveform.shape[0] > 1:
        waveform = waveform[0:1]

    spectrogram_transform = torchaudio.transforms.Spectrogram(
        n_fft=1024,
        hop_length=512,
        power=2.0
    )
    spectrogram = spectrogram_transform(torch.tensor(waveform))
```

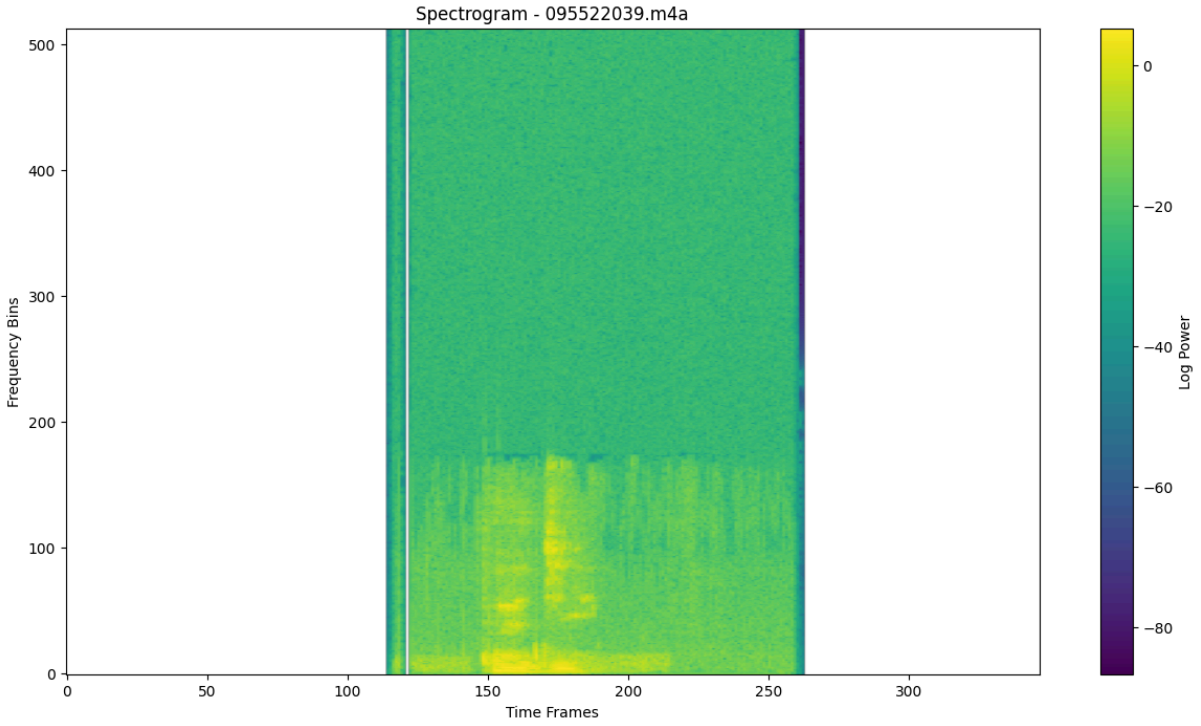
```
    plt.figure(figsize=(15, 8))
    plt.imshow(
        spectrogram[0].log2().numpy(),
        aspect='auto',
        origin='lower',
        cmap='viridis'
    )
    plt.title(title)
    plt.xlabel('Time Frames')
    plt.ylabel('Frequency Bins')
    plt.colorbar(label='Log Power')
    plt.show()
```

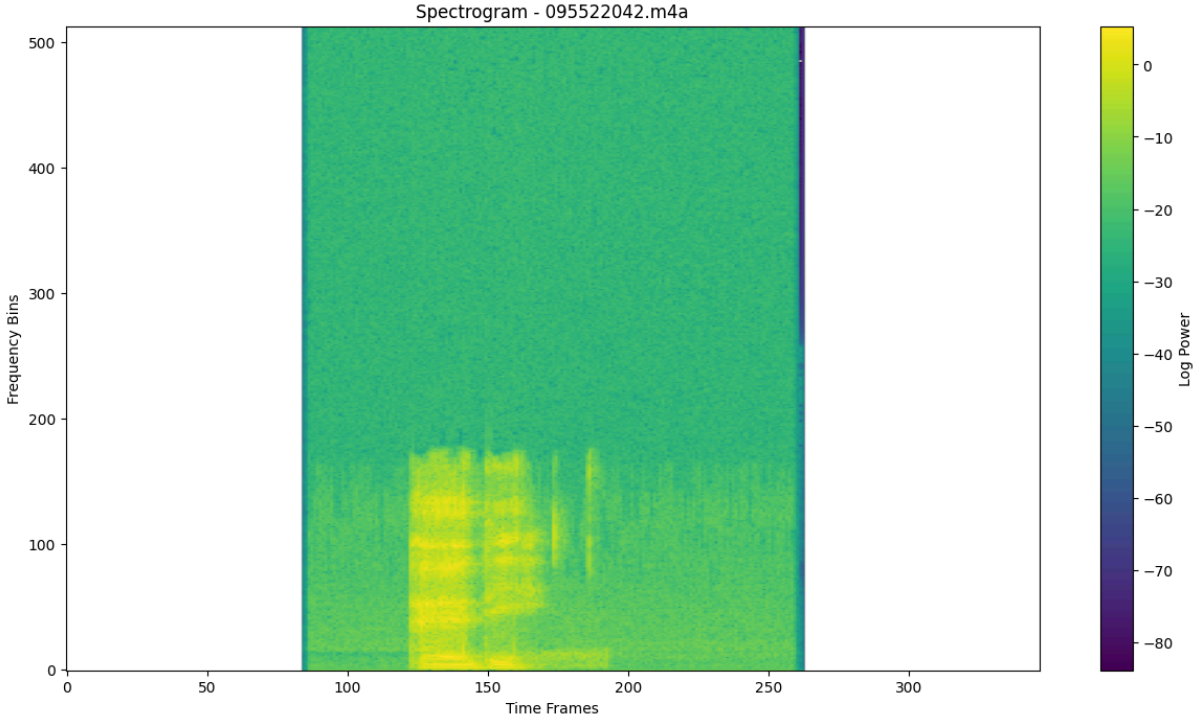
7단계: Spectrogram 출력 (요구사항 1 달성)

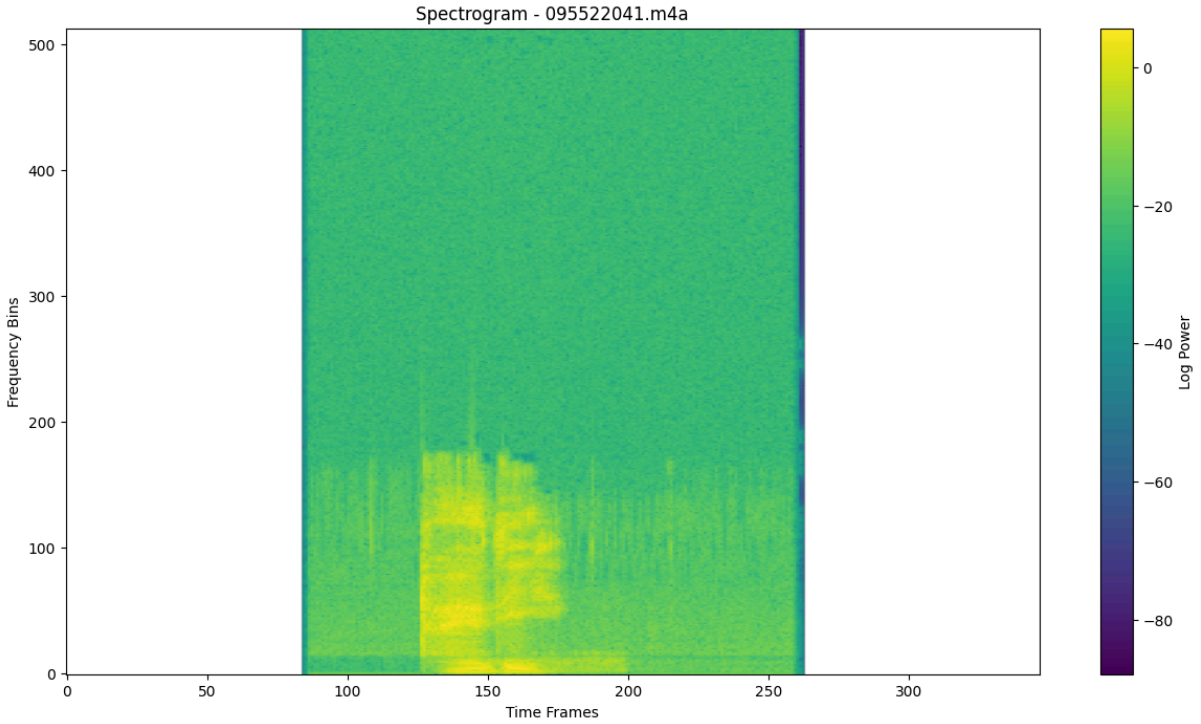
sample audio dataset의 각 파일에 대해 spectrogram을 성공적으로 출력했다.

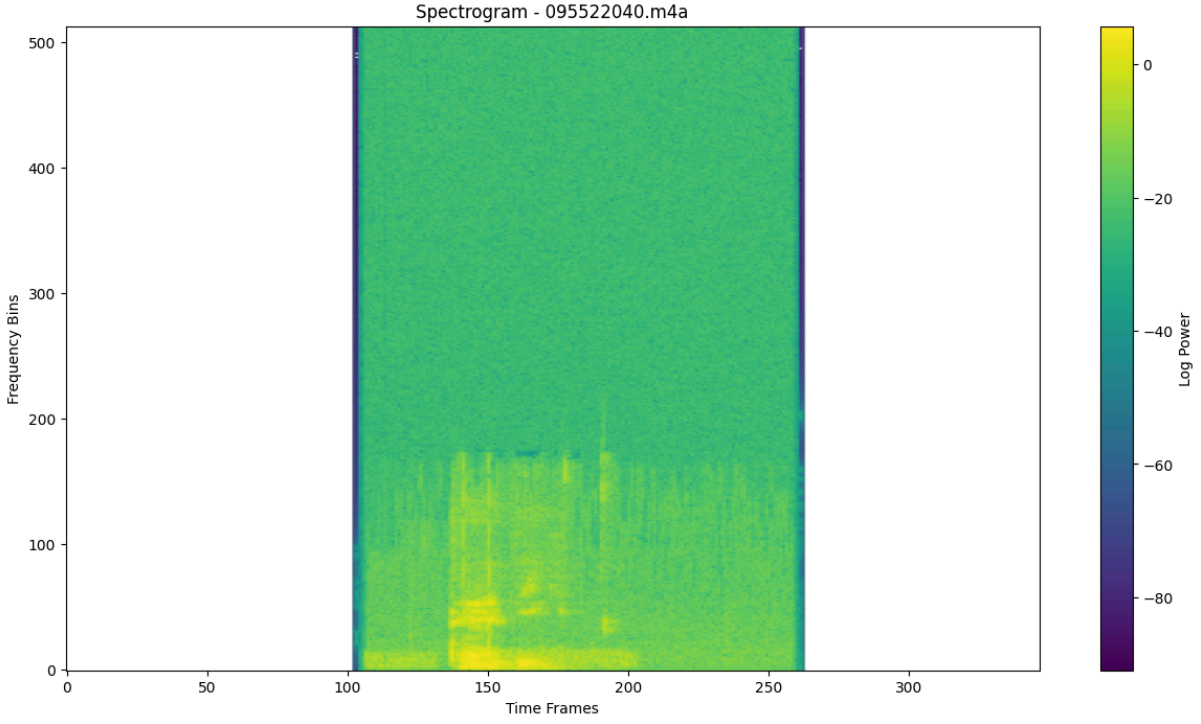
```
for audio_file in m4a_files:
    file_path = os.path.join(extract_path, audio_file)
    waveform, sample_rate = torchaudio.load(file_path)
    plot_spectrogram(waveform, sample_rate, f"Spectrogram - {audio_file}")
```

결과: 4개 파일의 spectrogram 시각화 완료


Spectrogram - 095522039.m4a
Frequency Bins
Time Frames
Log Power
0
50
100
150
200
250
300
0
100
200
300
400
500
0
−20
−40
−60
−80


Spectrogram - 095522042.m4a
Frequency Bins
Time Frames
Log Power
0
50
100
150
200
250
300
0
100
200
300
400
500
0
−10
−20
−30
−40
−50
−60
−70
−80





8단계: 리샘플링 함수 정의

Downsampling과 Upsampling을 위한 함수들을 구현했다.

```
def resample_audio(waveform, orig_freq, new_freq):
    resampler = torchaudio.transforms.Resample(orig_freq, new_freq)
```

```
    resampled_waveform = resampler(waveform)
    return resampled_waveform

def compare_resampling(original, downsampled, upsampled, orig_sr, down
_sr, up_sr, filename):
    fig, axes = plt.subplots(3, 1, figsize=(15, 12))

    # 원본, 다운샘플링, 업샘플링 파형을 3개 서브플롯으로 시각화
    time_orig = torch.arange(0, original.shape[1]) / orig_sr
    axes[0].plot(time_orig, original[0].numpy(), linewidth=0.8, color='blue')
    axes[0].set_title(f'Original - {filename} (SR: {orig_sr} Hz)')

    time_down = torch.arange(0, downsampled.shape[1]) / down_sr
    axes[1].plot(time_down, downsampled[0].numpy(), linewidth=0.8, color
='red')
    axes[1].set_title(f'Downsampled (SR: {down_sr} Hz)')

    time_up = torch.arange(0, upsampled.shape[1]) / up_sr
    axes[2].plot(time_up, upsampled[0].numpy(), linewidth=0.8, color='gree
n')
    axes[2].set_title(f'Upsampled (SR: {up_sr} Hz)')

    plt.tight_layout()
    plt.show()
```

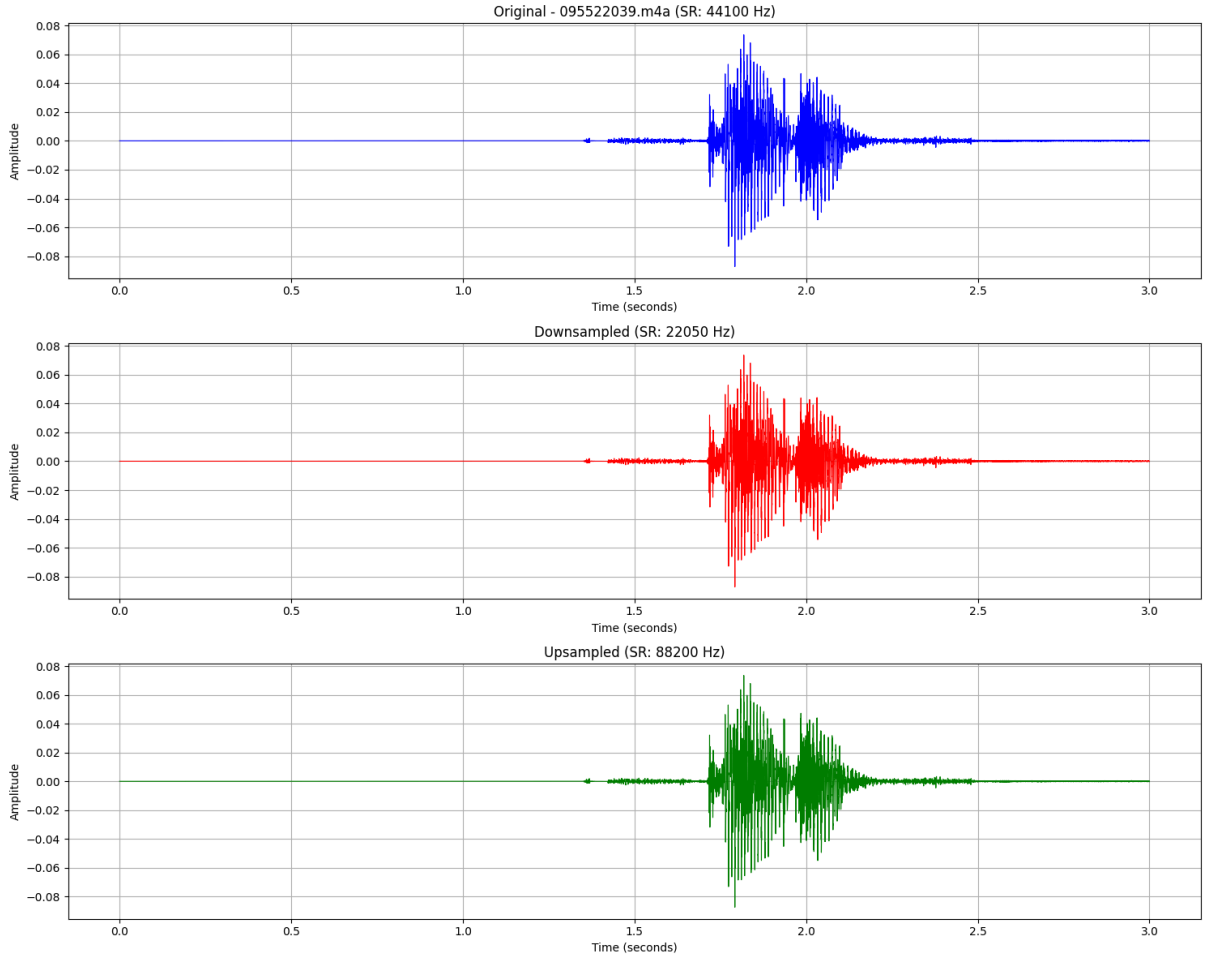
9단계: Downsampling과 Upsampling 수행 (요구사항 2 달성)

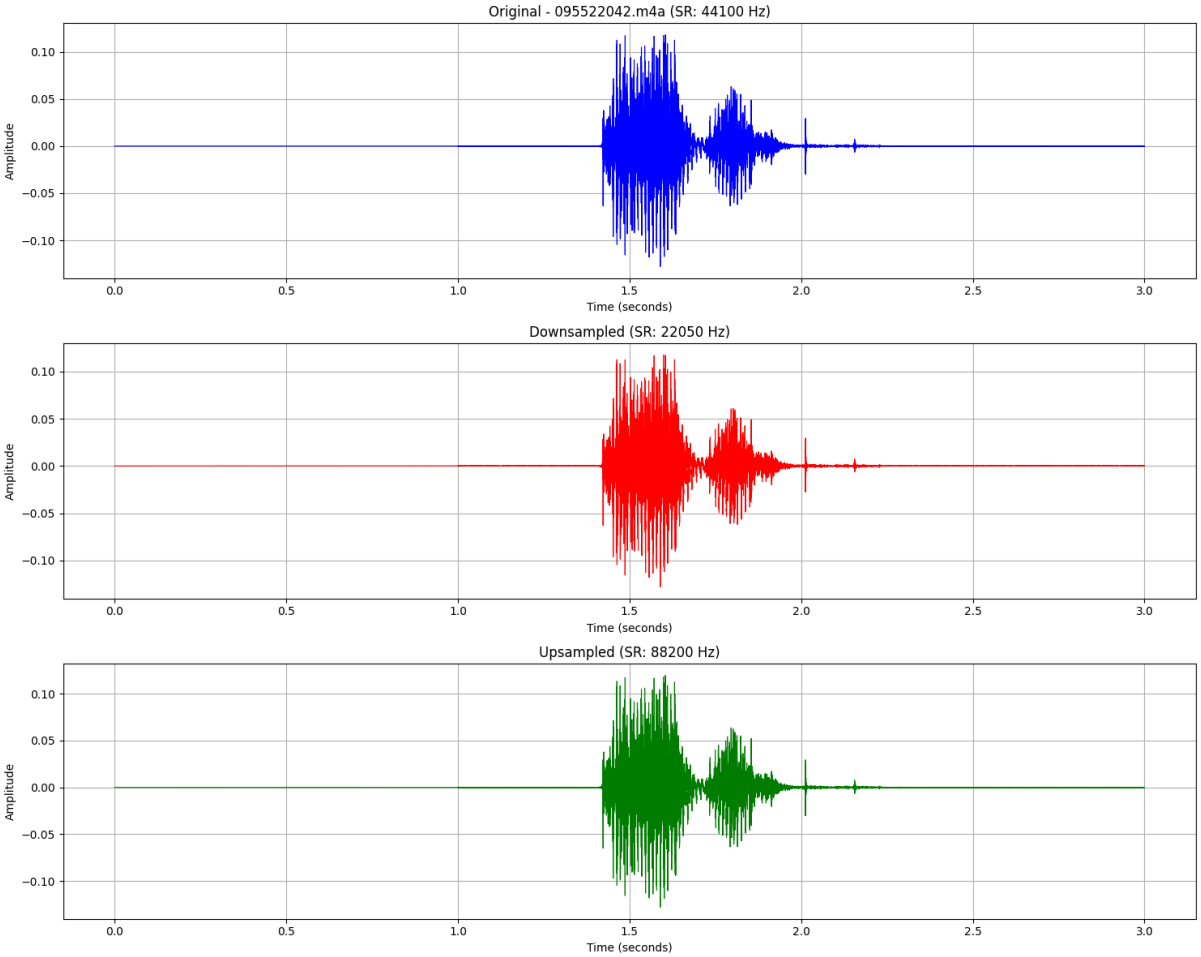
각 오디오 파일에 대해 다운샘플링(샘플링 레이트 절반)과 업샘플링(샘플링 레이트 2배)을 수행했다.

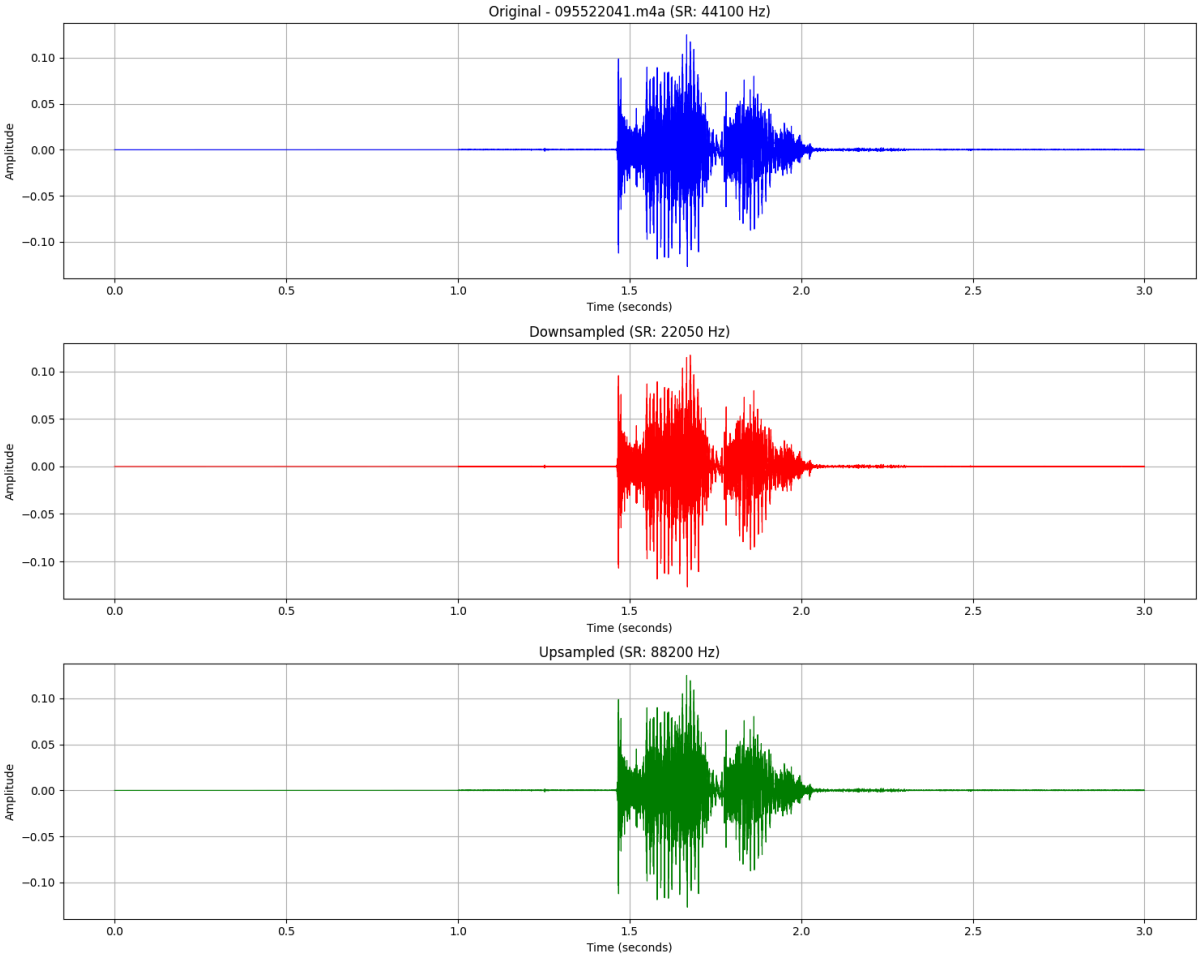
```
for audio_file in m4a_files:
    file_path = os.path.join(extract_path, audio_file)
    waveform, sample_rate = torchaudio.load(file_path)

    # 다운샘플링 (샘플링 레이트 절반으로 감소)
    down_sr = sample_rate // 2
    downsampled = resample_audio(waveform, sample_rate, down_sr)
```

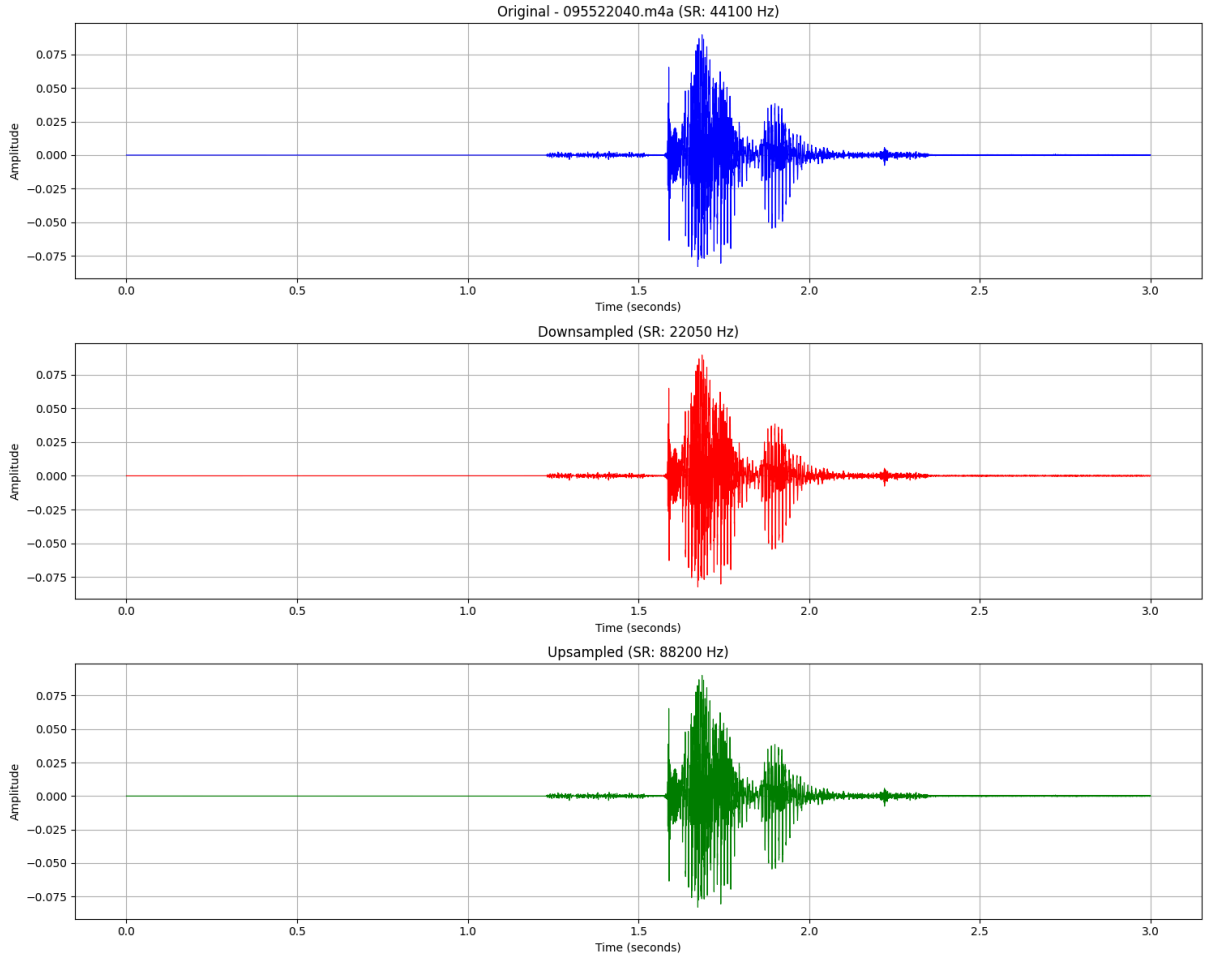
```
    # 업샘플링 (샘플링 레이트 2배로 증가)
    up_sr = sample_rate * 2
    upsampled = resample_audio(waveform, sample_rate, up_sr)

    print(f"파일: {audio_file}")
    print(f"원본: {waveform.shape} (SR: {sample_rate} Hz)")
    print(f"다운샘플링: {downsampled.shape} (SR: {down_sr} Hz)")
    print(f"업샘플링: {upsampled.shape} (SR: {up_sr} Hz)")

    # 비교 시각화 (첫 3초)
    max_samples_orig = min(waveform.shape[1], sample_rate * 3)
    max_samples_down = min(downsampled.shape[1], down_sr * 3)
    max_samples_up = min(upsampled.shape[1], up_sr * 3)

    compare_resampling(
        waveform[:, :max_samples_orig],
        downsampled[:, :max_samples_down],
        upsampled[:, :max_samples_up],
        sample_rate, down_sr, up_sr, audio_file
    )
```

결과: 4개 파일의 다운샘플링/업샘플링 성공적으로 수행됨


Original - 095522039.m4a (SR: 44100 Hz)
Amplitude
Time (seconds)
Downsampled (SR: 22050 Hz)
Amplitude
Time (seconds)
Upsampled (SR: 88200 Hz)
Amplitude
Time (seconds)


Original - 095522042.m4a (SR: 44100 Hz)
Amplitude
Time (seconds)
Downsampled (SR: 22050 Hz)
Amplitude
Time (seconds)
Upsampled (SR: 88200 Hz)
Amplitude
Time (seconds)


Original - 095522041.m4a (SR: 44100 Hz)
Downsampled (SR: 22050 Hz)
Upsampled (SR: 88200 Hz)
Amplitude
Time (seconds)
0.10
0.05
0.00
−0.05
−0.10
0.0
0.5
1.0
1.5
2.0
2.5
3.0



과제 완료 확인

요구사항 1: sample audio dataset을 사용하여 waveform과 spectrogram 출력

- **Waveform 출력**: 4개 파일 모두 완료 (5단계)
- **Spectrogram 출력**: 4개 파일 모두 완료 (7단계)

요구사항 2: Downsampling and Upsampling 수행

- **Downsampling**: 원본 샘플링 레이트의 50%로 감소 수행 (9단계)
- **Upsampling**: 원본 샘플링 레이트의 200%로 증가 수행 (9단계)

사용된 기술

- **PyTorch Audio**: 오디오 파일 로드 및 리샘플링
- **torchaudio.transforms.Spectrogram**: STFT 기반 스펙트로그램 생성

- **torchaudio.transforms.Resample**: 고품질 리샘플링 변환
- **Matplotlib**: 파형 및 스펙트로그램 시각화

결론

본 과제를 통해 PyTorch Audio 라이브러리의 핵심 기능들을 성공적으로 활용하여 오디오 신호 처리의 기본 작업들을 완료했다. 4개의 sample audio dataset에 대해 waveform과 spectrogram을 정확히 출력했으며, downsampling과 upsampling을 통한 샘플링 레이트 변환도 성공적으로 수행했다.