# Audio Data Collection and Analysis Lab Report

Ashok Enukonda

## 1 Introduction

This lab was about learning how to work with audio data. The main goal was to understand how to collect, check, label, and analyze sounds. We also learned how to make our own recording plan (protocol) for future data collection.

There were four main tasks:

- Collecting and checking the quality of audio files.
- Labeling files and creating spectrograms.
- Extracting useful features and cleaning the data.
- Writing a protocol for recording new audio data in real life.

## 2 Task 1 – Audio Data Collection and Quality Check

**Goal:** To collect audio files from the UrbanSound8K dataset and make sure all of them have the same quality and format.

### Steps I Followed

The dataset has many folders (`fold1` to `fold10`), each with `.wav` files. I selected four sound classes:

- Dog Bark
- Drilling
- Siren
- Street Music

I picked 20 audio files from each class (total = 80 files). I put them in new folders like this:

```
selected_4x20/
    dog_bark/
    drilling/
    siren/
    street_music/
```

Then I checked for possible problems:

- Too much background noise
- Very short clips
- Low volume (quiet sounds)
- Clipping or distortion

All files were converted to:

- Sampling rate: 44,100 Hz
- Channels: Mono
- Format: WAV (16-bit)

### Code Example

```
import librosa, os, numpy as np, pandas as pd
from pathlib import Path

# Path to main dataset
AUDIO_ROOT = Path(r"D:\DU-SEM2\Period 1\Data Collection\Lab 3 Audio Folder\
    archive")

# Check one file for quality
file_path = AUDIO_ROOT / "dog_bark_01.wav"
y, sr = librosa.load(file_path, sr=44100, mono=True)

# Calculate duration and RMS
duration = librosa.get_duration(y=y, sr=sr)
rms = np.mean(librosa.feature.rms(y=y))
print("Duration:", duration, "sec | RMS:", rms)
```

### Result

80 clean audio files ready. Sampling rate, bit depth, and format are all the same. Files with very low or noisy signals were noted in a CSV file (`quality_overview.csv`).

## 3 Task 2 – Data Labelling and Spectrograms

**Goal:** To label every audio file with its class name and create one spectrogram for each sound class.

### Steps I Followed

Each file got a label (class name) from its folder name. Example: `dog_bark_01.wav` → `label = dog_bark` The file names and labels were saved in `labeled_manifest.csv`. I created one spectrogram per class using Librosa.

### What is a Spectrogram?

A spectrogram is a picture that shows how sound changes over time.

- X-axis: Time (seconds)
- Y-axis: Frequency (Hz)
- Color: Loudness (dB)

### Code Example

```
import matplotlib.pyplot as plt
import librosa.display

y, sr = librosa.load("dog_bark_example.wav", sr=44100, mono=True)
S = np.abs(librosa.stft(y, n_fft=2048, hop_length=512))
D = librosa.amplitude_to_db(S, ref=np.max)

plt.figure(figsize=(10, 4))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='log')
plt.title('Spectrogram - Dog Bark')
plt.colorbar(format='%+2.0f dB')
plt.show()
```

### Result

I created one spectrogram for each class:

- Dog Bark
- Drilling
- Siren
- Street Music

These images clearly show how sound frequencies vary in each type of sound. (You can insert the spectrogram pictures in the report here.)

## 4 Task 3 – Feature Extraction and Data Cleaning

**Goal:** To get useful numerical features from each sound file and clean the final dataset.

## Steps I Followed

I loaded each audio file using Librosa and extracted:

- Zero Crossing Rate (ZCR): how often the signal crosses zero
- Root Mean Square (RMS): energy of the sound
- Spectral Centroid: average frequency
- Spectral Bandwidth: spread of frequencies
- Spectral Rolloff: point below which 85% and 95% of energy is contained
- MFCCs (Mel-Frequency Cepstral Coefficients): sound texture values

Each file's data was stored as one row in a CSV file (`audio_features_raw.csv`).

## Code Example

```python
import numpy as np, librosa, pandas as pd

def extract_features(file_path):
    y, sr = librosa.load(file_path, sr=44100, mono=True)
    features = {
        "zcr": np.mean(librosa.feature.zero_crossing_rate(y)),
        "rms": np.mean(librosa.feature.rms(y=y)),
        "spec_centroid": np.mean(librosa.feature.spectral_centroid(y=y, sr=
            sr)),
        "spec_bandwidth": np.mean(librosa.feature.spectral_bandwidth(y=y,
            sr=sr)),
    }
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    for i in range(13):
        features[f"mfcc{i+1}"] = np.mean(mfcc[i])
    return features
```

## Data Cleaning Steps

- Removed empty or duplicate columns.
- Verified that 80 rows (one per file) remained.
- Saved final data as `audio_features_cleaned.csv`.

## Result

Clean feature file with 80 rows × 30 columns, including features and the class label ready for machine learning.

## 5 Task 4 – Recording Protocol for Audio Data Collection

**Goal:** To describe how we can record audio data ourselves in a clean, safe, and ethical way.

### 1. Common Problems While Recording

- Background sounds (wind, talking, traffic)
- Different rooms sound different
- Poor microphones create hiss or uneven volume
- Movement of the recorder changes loudness
- Large files take up more storage

### 2. Equipment Setup

- Use a good condenser microphone or clip-on mic.
- Add a pop filter or foam cover to block wind noise.
- Keep the mic 15–30 cm away from the sound source.
- Record at 44.1 kHz, mono, 16-bit WAV.

### 3. Recording Environment and Settings

- Choose a quiet place without echo.
- Record at the same time of day for consistency.
- Use curtains or foam to reduce reflections.

### 4. Ethical Rules

- Ask people before recording their voices.
- Don’t record private or personal conversations.
- Store recordings securely and ethically.

### 5. Summary

Following these steps helps collect clean and consistent sound data that respects privacy and ethics. This makes the recordings ready for research or machine learning tasks.

## 6 Conclusion

In this lab, I learned how to prepare, clean, and analyze audio data.

- **Task 1:** Collected and checked the quality of 80 files (4 classes × 20 files).
- **Task 2:** Labeled files and created spectrograms.
- **Task 3:** Extracted sound features and cleaned the data.
- **Task 4:** Designed a clear and ethical protocol for recording audio data.

This project taught me the full process of handling real world sound from downloading files to building a dataset that is clean, labeled, and ready for analysis.