语音信息处理实验报告

课程名称：语音信息处理　　实验日期：2023/10/31

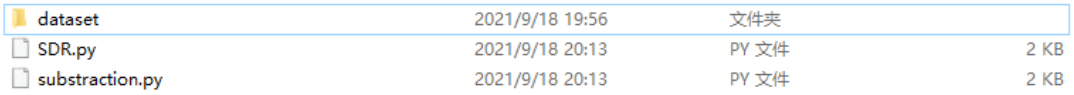
班级：人工智能 3 班　　姓名：什么玩意　　学号：我不会，我不会，啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊

实验一 基于谱减法的语音增强

一、实验目的

利用谱减法对语音进行降噪，了解谱减法的基本算法，分析不同种类的噪声对谱减法性能的影响，以及谱减法应用前后语音性能提升的效果。

二、实践内容：



已经提前给出数据和谱减法代码(substraction.py)

一般情况下，我们不能拿到噪声的音频，因此只能利用混合音频(mixture)估计噪声频谱。一般认为前五帧是只包含噪声不包含人声的静音段，可以用此来估计噪声。

利用谱减法给每条混合音频降噪。

并对降噪的语音进行分析，可以从听感，语谱图以及 SDR 等方面进行分析。

注：SDR 全称 signal-to-distortion ratio，可以对语音进行估计，数值越大代表音频越纯净。

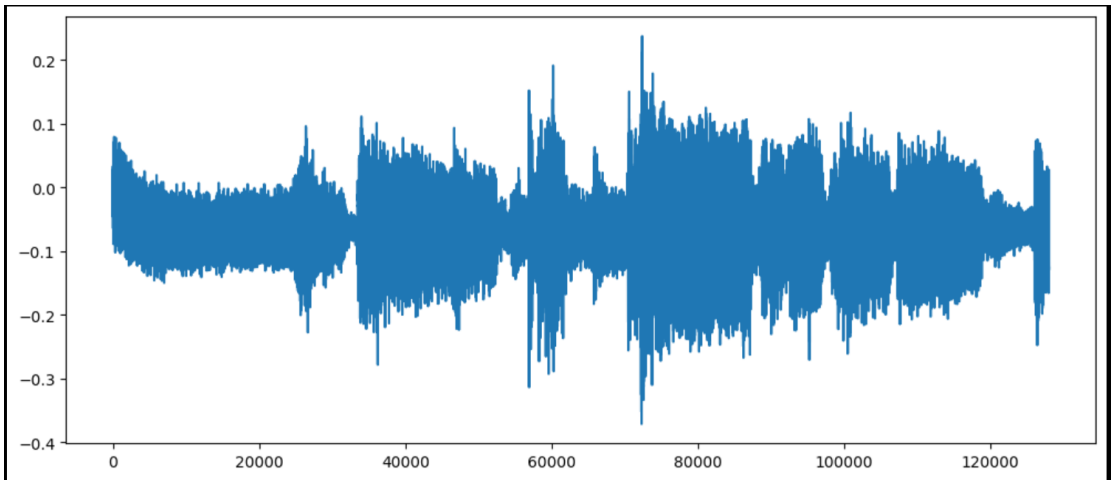
- 本实验提供的谱减法代码仅仅为基础版，谱减法也有很多变体。同学们也可参考文章自行修改编写其他种类的谱减法。

三、实践结果与分析

首先采样率 16000 加载音频信号，进行短时傅里叶变换

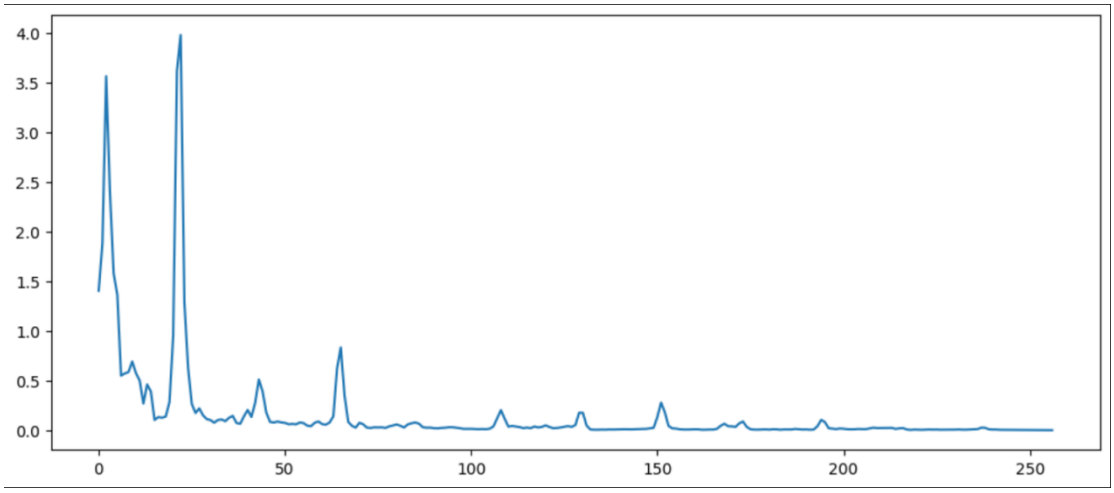
```
audio_data, sr = librosa.load(mixture_file, sr=16000)
print(audio_data.shape)
stft_audio = librosa.stft(audio_data, n_fft=512)  # 对音频信号进行短时傅里叶变换
print(stft_audio.shape)
plt.plot(audio_data)
```

打印图像观察



计算幅度值，估计噪音幅度

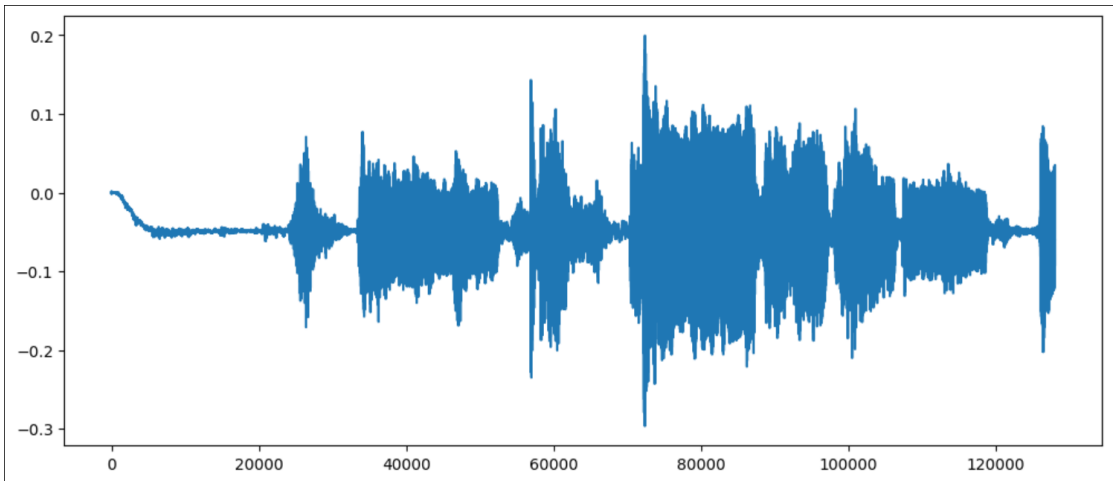
```
# 噪声幅度计算 假设前 5 帧为 silence(noise)  也可以采用其他方式估计噪声
mag_audio = np.abs(stft_audio)  # 幅度谱
print(mag_audio.shape)
noise_mean = np.zeros((mag_audio.shape[0],))
for i in range(0, 5):
    noise_mean += mag_audio[:,i]
noise_mean /= 5  # 取平均
print(noise_mean.shape)
plt.figure(figsize=(12,5))
plt.plot(noise_mean)
```



计算相位谱，去除均值噪声后进行逆傅里叶变换，实现去噪

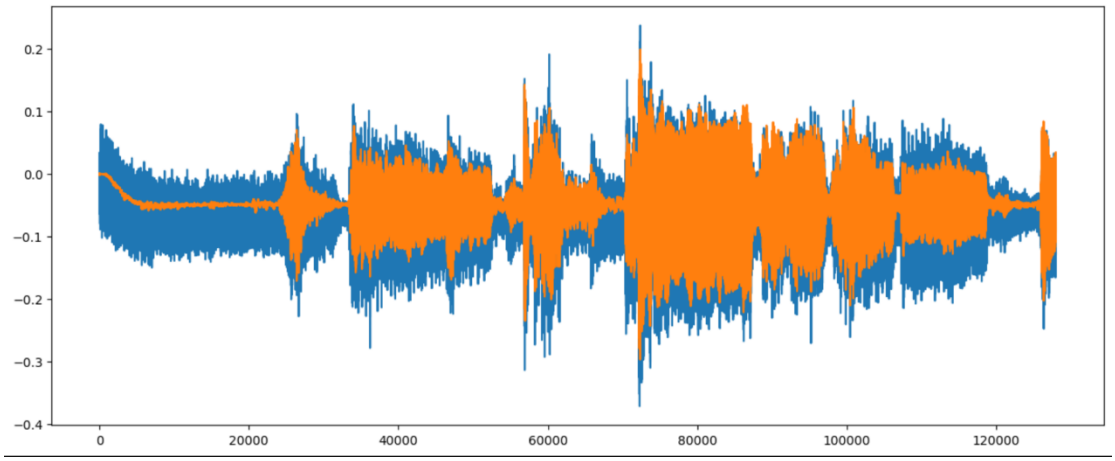
```
pha_audio = np.angle(stft_audio)  # 相位谱
print(pha_audio.shape)
for i in range(mag_audio.shape[1]):
    mag_audio[:,i] = mag_audio[:,i] - noise_mean
mag_audio_ = np.where(mag_audio > 0, mag_audio, 0)  # 大于 0 的部分保持不变 负数取 0
print(mag_audio_.shape)
stft_audio_ = mag_audio_ * np.exp(1.0j*pha_audio)  # 利用原始相位信息进行逆傅里叶变换变换
```

```
print(stft_audio_.shape)
wav_data = librosa.istft(stft_audio_)
print(wav_data.shape)
plt.figure(figsize=(12,5))
plt.plot(wav_data)
```




```
plt.figure(figsize=(15, 6))
# plt.subplot(121)
plt.plot(audio_data)
# plt.subplot(122)
plt.plot(wav_data)
```

打印观察前后关系，可见谱减法去噪效果

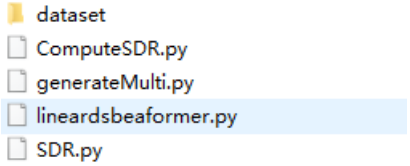


实验二 多通道语音增强

一. 实验目的：

应用波束形成算法对多通道音频降噪，了解麦克风阵列数据降噪的基本流程。

二．实践内容：



本实验已经提供数据 dataset。

Lineardsbeamformer.py: delay-and-sum 算法的具体实现，通过更改 doa 的数值可以增强对应角度的声音而抑制其他角度的声音。 一般来说当人声角度和噪声角度之间的差异越大，音频增强后的效果越好。如果两个角度相差不大则不容易区分。

ComputeSDR.py:用来计算 clean 和 mixture 以及 estimate 和 clean 之间的 SDR 大小。注意：多通道音频无法直接计算 SDR 一般提取第 0 个通道的音频（单通道）来计算 SDR。

GenerateMulti.py: 用来生成多通道的音频数据，需要指定声源的角度和声源的距离以及单通道的声音信号。由于本实验仅仅提供了 4 组音频，同学们可以自行生成其他角度的多通道音频信号。（选做）

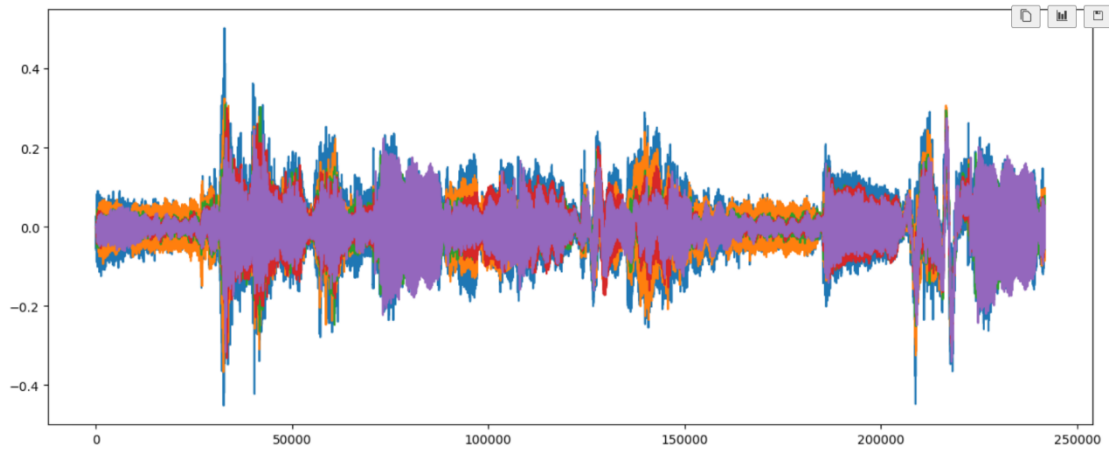
可以通过对比音频增强前后的 SDR、听感差异、语谱图差异等方面来分析 delay-and-sum 的效果。

四、实践结果与分析

加载语音资源，五个通道，定义声音角度

```
doa = 20  #声源传来的角度   clean 的语音的角度  #根据不同的音频 需要自己修改
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
wave_data, sr = sf.read('./dataset/mixture/clean_20_unsteadyNoise_10.wav')
if sr!=16000:
    wave_data = librosa.resample(wave_data, sr, 16000)
print(f'wave_data.shape {wave_data.shape}') # [T, C]
plt.figure(figsize=(15, 6))
plt.plot(wave_data)
```

观察声纹信号，确实不一

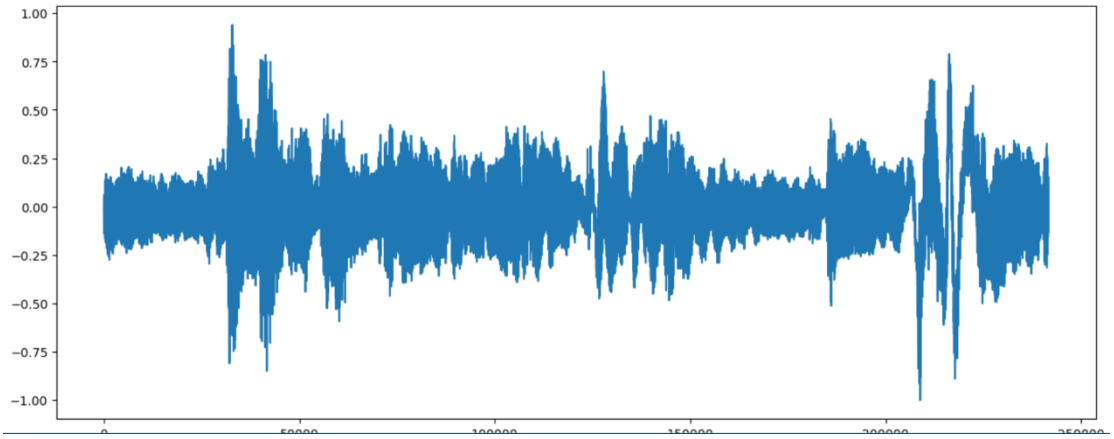


对每个声纹信号进行短时傅里叶变换

```
wav_stft = []
for i in range(wave_data.shape[1]):
    wav_stft.append(librosa.stft(wave_data[:,i]))
wav_stft = np.array(wav_stft)
print(wav_stft.shape)
```

```
enh_stft = run(doa, wav_stft)
enh_wav = librosa.istft(enh_stft)
enh_wav = enh_wav / np.max(np.abs(enh_wav))
print(enh_wav.shape)
plt.figure(figsize=(15, 6))
plt.plot(enh_wav)
```

经过处理之后，波形无法体现变化，肉耳听完感觉人声清澈了许多，噪音也清澈许多。



拆解 run 函数：

```
def run(doa, obs, c=340, sr=16000):
    num_bins = obs.shape[1]
    # 180 degree <---------> 0 degree
    dist = np.cos(doa * np.pi / 180) * topo
    omega = np.pi * np.arange(num_bins) * sr / (num_bins - 1)
    # temp_1 = distance/c
```

```
    steer_vector = np.exp(-1j * np.outer(omega, dist / c))
    # temp = np.abs(steer_vector)
    weight = steer_vector / num_mics #sv 对应的是 delay and sum beaformer 中的 weight
    obs = np.transpose(obs, (1, 0, 2))
    obs = np.einsum("...n,...nt->...t", weight.conj(), obs)
    return obs
```

doa:声源角度,范围为 0-180 度
obs:传感器采集的音频信号数组,形状为(通道数,样本数)
c:声速,默认为 340m/s
sr:采样率,默认 16000Hz
dist:根据 doa 和传感器数组结构计算各通道与声源的距离差，这个变量算了五个 mic 的距离
omega:频率向量,范围 0-采样率/2
steer_vector:时间延迟核,考虑每个通道与声源的距离差异产生的时延
weight:时间延迟核归一化后的权重向量
obs 转置后与 weight 作内积，np.einsum 是 numpy 内置的张量计算函数，实现延迟和相位差的增益,强调从特定方向来的声源，返回滤波后的信号,实现了基于固定角度的滤波功能