음성인식 실습 1

Torchaudio를 이용한 음성파일 다루기

김지환

서강대학교 컴퓨터공학과

Table of contents

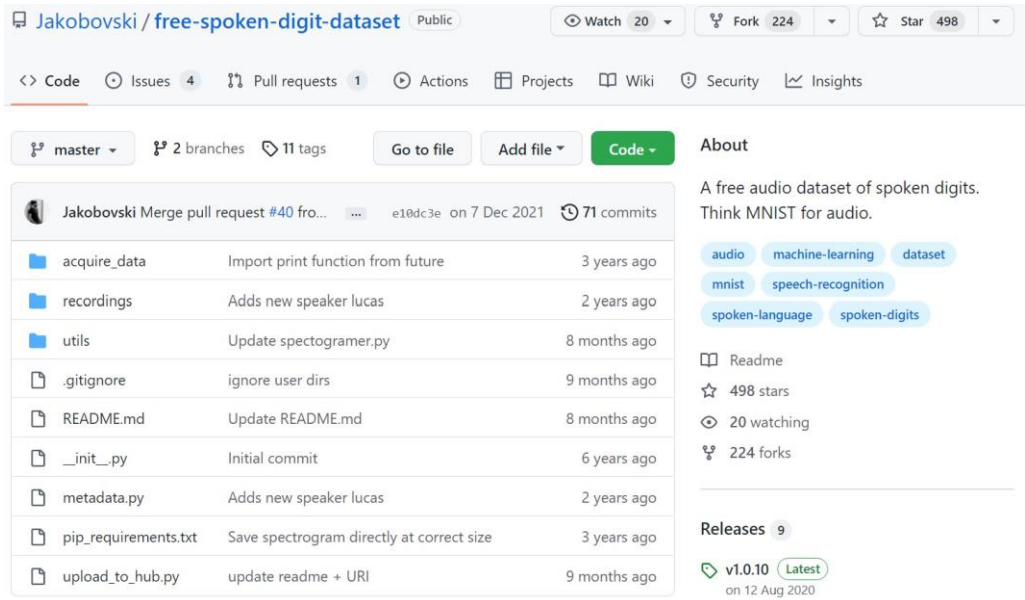
1.1 Google colab 소개

- Google Colaboratory(Google)
  - https://colab.research.google.com/
  - 웹 브라우저에서 파이썬을 작성하고 실행할 수 있는 서비스
  - 클라우드 기반의 주피터 노트북 개발환경
  - 기본적으로 파이썬 사용가능
    - Tensorflow, PyTorch, mataplotlib, scikit-learn, pandas 등의 ML/DL에 사용하는 라이브러리들을 기본적으로 지원
  - K80 GPU를 무료로 사용 가능
    - 사용량의 제한이 있으나 일반적으로 교육용으로 사용하기에는 문제 없음

	Colab Free	Colab Pro	Colab Pro +
Guarantee of resources	Low	High	Even Higher
GPU	K80	K80, T4 and P100	K80, T4 and P100
RAM	16 GB	32 GB	52 GB
Runtime	12 hours	24 hours	24 hours
Background execution	No	No	Yes
Costs	Free	9.99$ per month	49.99$ per month
Target group	Casual user	Regular user	Heavy user

1.2 음성 데이터 셋

- 오늘 실습에서 사용할 데이터 셋
  - Free-spoken-digit-dataset
    - 음성 버전의 MNIST dataset
    - https://github.com/Jakobovski/free-spoken-digit-dataset



- Free-spoken-digit-dataset 특징
  - 6 speakers
  - 3,000 recordings(50 of each digit per speaker)
  - English pronunciations (영어 발음)
  - Files are named in the following format: {digitLabel}_{speakerName}_{index}.wav
    - ✓ Example: 7_jackson_32.wav
  - Index 0-49 중의 10%인 0-4는 test 할 때 사용되고, 5-49는 training dataset으로 쓰임

1.2 음성 데이터 셋

- git clone을 통해 데이터 셋 받고 확인

```
!git clone https://github.com/Jakobovski/free-spoken-digit-dataset
```

```
Cloning into 'free-spoken-digit-dataset'...
remote: Enumerating objects: 4237, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 4237 (delta 12), reused 8 (delta 8), pack-reused 4212
Receiving objects: 100% (4237/4237), 30.37 MiB | 30.76 MiB/s, done.
Resolving deltas: 100% (116/116), done.
```

```
!ls
```

```
free-spoken-digit-dataset  sample_data
```

```
!ls free-spoken-digit-dataset
```

```
acquire_data  metadata.py          README.md   upload_to_hub.py
__init__.py   pip_requirements.txt  recordings  utils
```

- 모든 wav파일의 path를 리스트로 저장

```
from pathlib import Path
```

`recodingds` : 디렉토리에 음성 파일들이 위치

```
audio_dir = Path("./free-spoken-digit-dataset/recordings")
audios = list(audio_dir.rglob("*.wav"))
```

audio파일들의 Path를 확인

```
audios[:10]
```

```
[PosixPath('free-spoken-digit-dataset/recordings/2_george_4.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/4_yweweler_44.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/0_theo_49.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/5_lucas_29.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/2_lucas_3.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/0_jackson_46.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/9_yweweler_9.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/6_nicolas_15.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/2_lucas_35.wav'),
 PosixPath('free-spoken-digit-dataset/recordings/5_yweweler_10.wav')]
```

- ipd.Audio를 이용한 wav 파일 들어보기

```
import IPython.display as ipd
```

```
idx = 5
print(audios[idx])
ipd.Audio(str(audios[idx]))
```

```
free-spoken-digit-dataset/recordings/0_jackson_46.wav
```

1.2 음성 데이터 셋

- torchaudio를 이용하여 음성 불러오기
  - Loading audio data
    - torchaudio.load()를 사용해서 audio data load
    - 이 함수의 input은 audio data의 경로 또는 파일자체
    - 이 함수의 return은 waveform (Tensor) 와 sample rate (int)
    - By default, Tensor의 타입은 torch.float32 이고 [-1.0, 1.0] 범위를 갖음
    - **Waveform, sample_rate = torchaudio.load(SAMPLE_WAV)**

```
import torchaudio
```

```
[ ] print(audios[1])
    y,sr = torchaudio.load(audios[1])
```

```
free-spoken-digit-dataset/recordings/4_yweweler_44.wav
```

```
[ ] print(type(y))
```

```
<class 'torch.Tensor'>
```

```
[ ] y.shape, sr
```

```
(torch.Size([1, 2911]), 8000)
```

  - 마찬가지로 ipd.Audio를 이용해서도 torch.Tensor 타입의 변수를 읽고 들을 수 있음

```
import IPython.display as ipd
```

```
[ ] ipd.Audio(y, rate=sr)
```

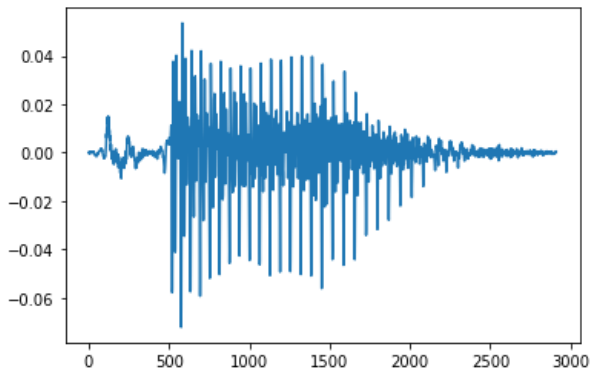
1.3 음성파일의 시각화

- torch.Tensor타입의 Waveform의 시각화
  - matplotlib.pyplot 을 이용하여 audio sample을 시각화 가능
  - Python의 slicing을 통해 특정구간을 확대하여 확인 가능

```
import matplotlib.pyplot as plt
```

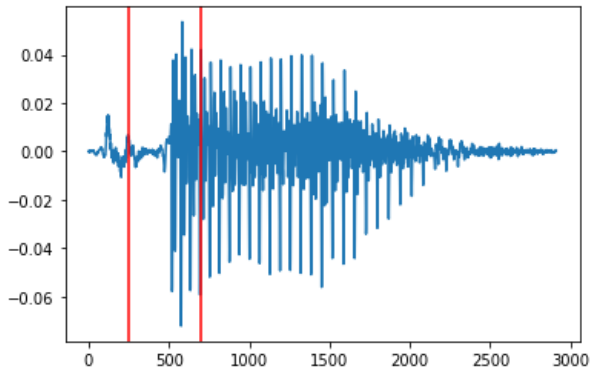
```
[ ] plt.plot(y[0])
```

```
[<matplotlib.lines.Line2D at 0x7fc501252190>]
```



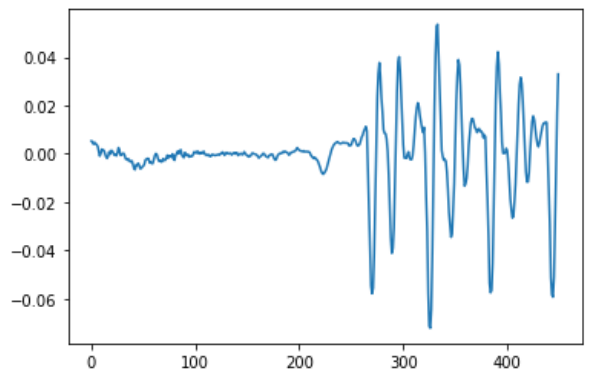

```
[ ] start,end = 250,700
    plt.plot(y[0])
    plt.axvline(start,color='r')
    plt.axvline(end,color='r')
```

```
<matplotlib.lines.Line2D at 0x7fc4fcd43c10>
```



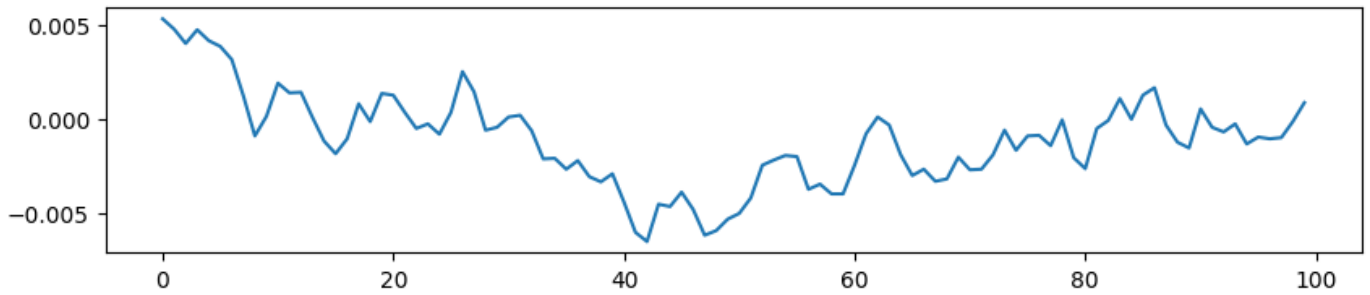

```
[ ] plt.plot(y[0][start:end])
```

```
[<matplotlib.lines.Line2D at 0x7fc4fcc3a610>]
```



1.3 음성파일의 시각화

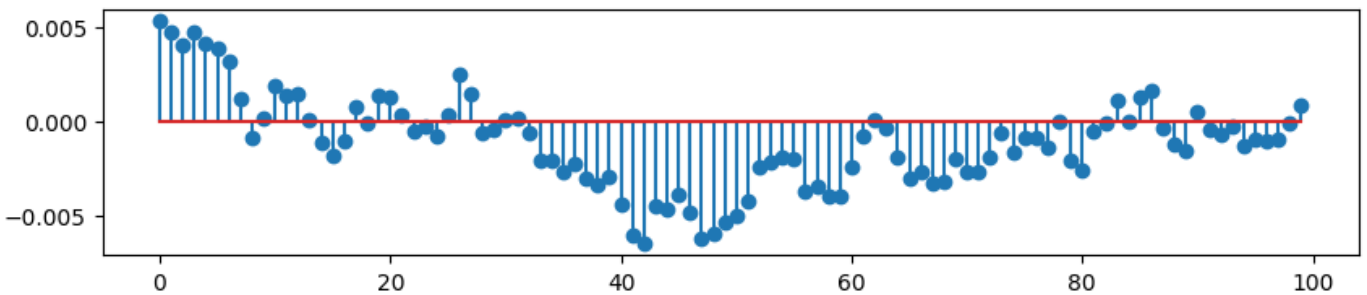
```
[ ]  start,dur = 250,100
     #plt.bar(range(dur),y[0][start:start+dur])
     plt.figure(figsize=(10,2),dpi=100)
     plt.plot(range(dur),y[0][start:start+dur])
     plt.show()
```



- matplotlib.pyplot.stem 을 이용하여 sample 확인

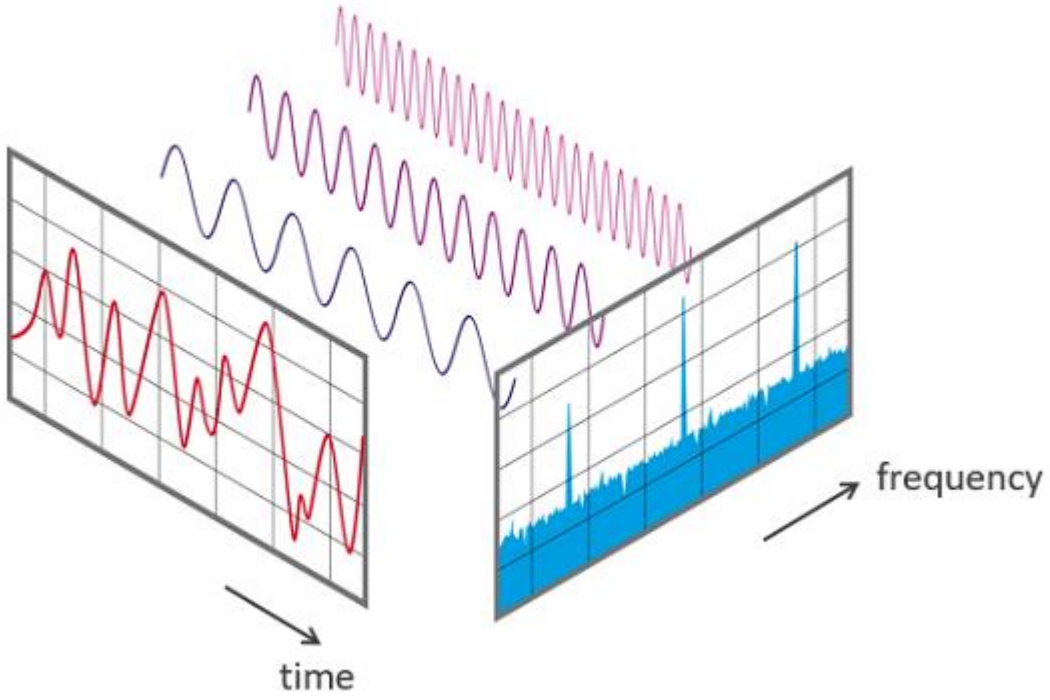
```
[ ]  plt.figure(figsize=(10,2),dpi=100)
     plt.stem(range(dur),y[0][start:start+dur], use_line_collection=True)
```

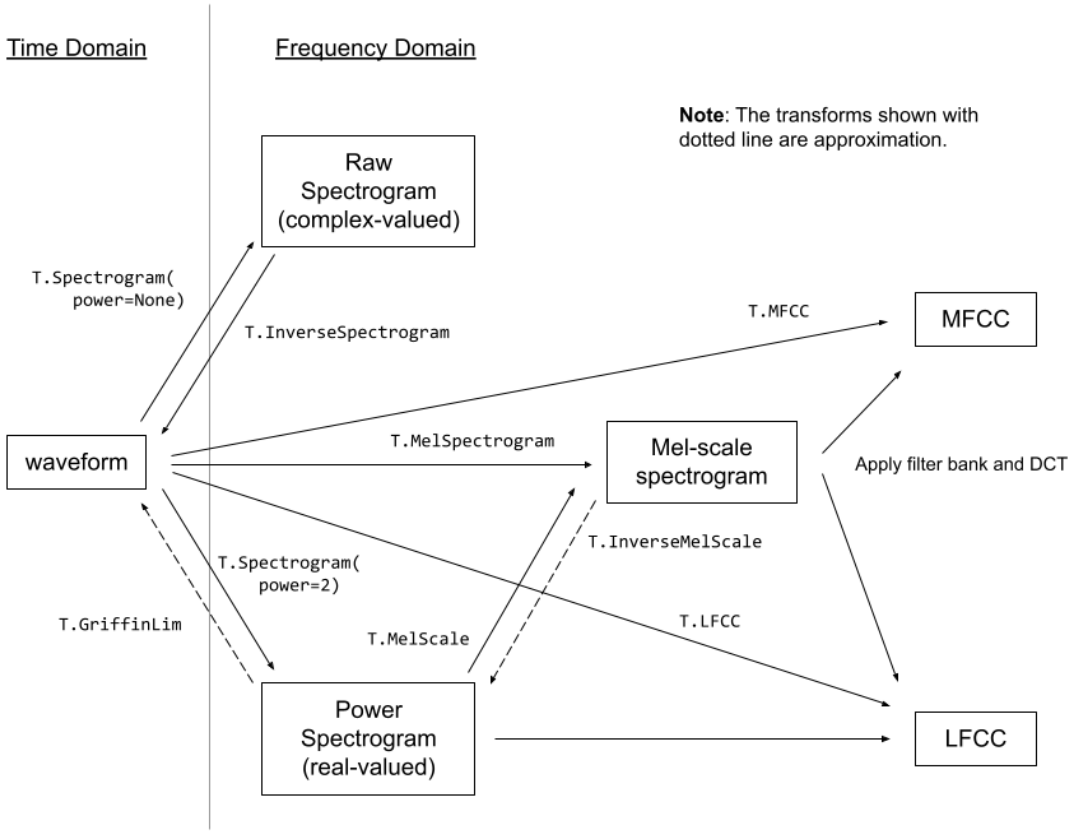
```
<StemContainer object of 3 artists>
```



1.4 Spectrogram

- Time과 frequency domain을 각 축으로 표현하고, amplitude를 색으로 표현한 것을 spectrogram이라 함





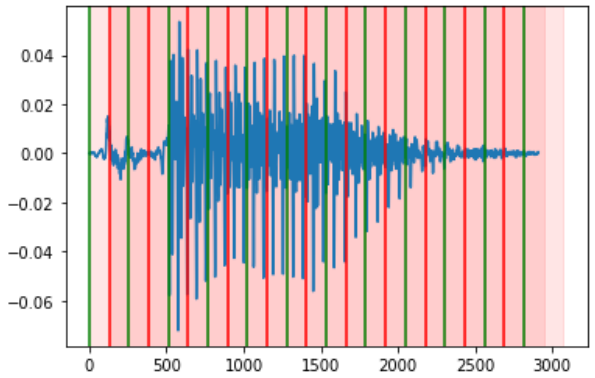
1.4 Spectrogram

◆ Raw Spectrogram

- torchaudio.transforms.Spectrogram class를 이용

```
import torchaudio.transforms as T
```

```
[ ]  n_fft=256
     win_length = n_fft
     hop_length=win_length//2
     start,dur = 0,2911
     plt.plot(y[0][start:start+dur])
     i=0
     for x in range(start,dur,hop_length):
       i+=1
       c='r' if i%2==0 else 'g'
       plt.axvline(x,color=c)
       plt.axvspan(x,x+win_length,color='r',alpha=0.1)
```




```
[ ]  spec_converter = T.Spectrogram(n_fft=n_fft,win_length=win_length,hop_length=hop_length)
     spec = spec_converter(y)
```

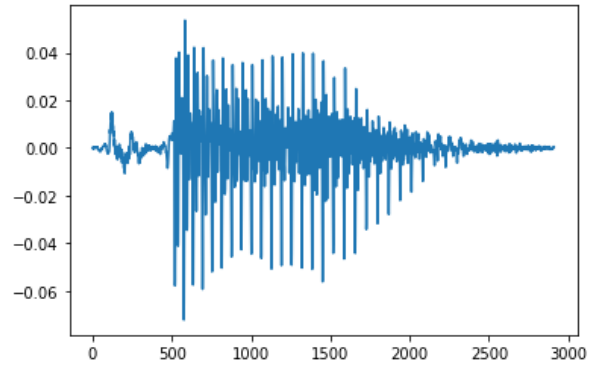
```
[ ]  spec.shape
```

```
torch.Size([1, 129, 23])
```

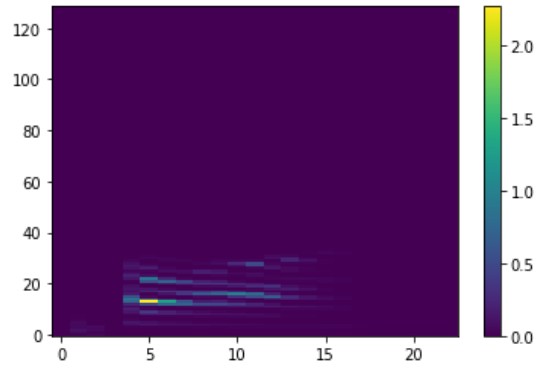
```
[ ]  print(len(y[0]))
     print(len(y[0])//hop_length+1)
```

```
2911
23
```

```
[ ]  plt.plot(y[0])
     plt.show()
     plt.imshow(spec[0],origin="below",aspect='auto',interpolation='nearest')
     plt.colorbar()
```




```
<matplotlib.colorbar.Colorbar at 0x7fc4f87945d0>
```



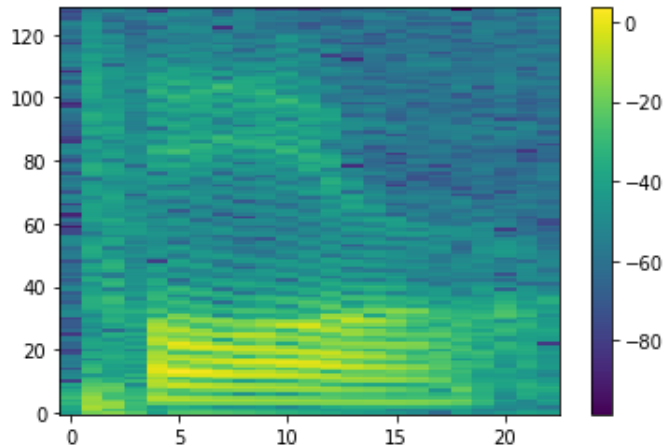
1.4 Spectrogram

- AmplitudeToDB
  - Power/amplitude scale에서 데시벨 scale로 바꿔줌
  - torchaudio.transforms.AmplitudeToDB(stype: str = 'power', top_db: Optional[float] = None)

```
[ ]  db_converter = T.AmplitudeToDB()
```

```
[ ]  db_spec = db_converter(spec)
     plt.imshow(db_spec[0],origin='lower',aspect='auto',interpolation='nearest')
     plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7fc4f8e9e290>
```




```
[ ]  spec.shape

     torch.Size([1, 129, 23])
```

1.4 Spectrogram

◆ Mel-Spectrogram

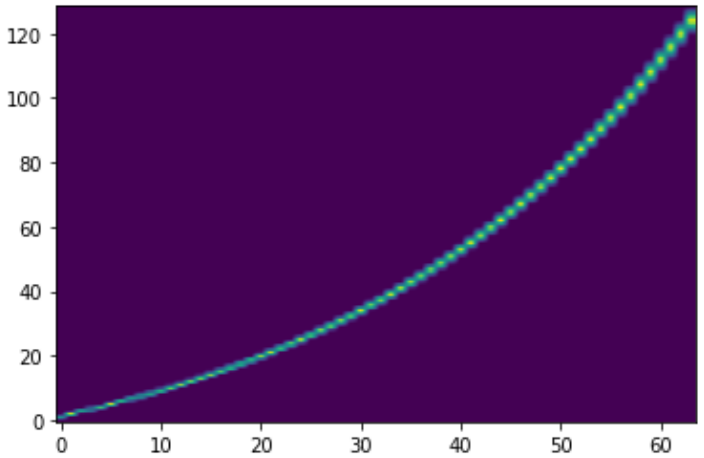
참고 : https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

- MelScale
  - 일반 STFT를 삼각형 필터 뱅크가 있는 멜 주파수 STFT로 변환합니다.

```
[ ]  mel_scale = T.MelScale(n_mels=64,sample_rate=8000,f_min=20,f_max=4000,n_stft=129)
```

```
[ ]  plt.imshow(mel_scale.fb,aspect='auto',origin='lower')
```

```
<matplotlib.image.AxesImage at 0x7fc4f785d690>
```



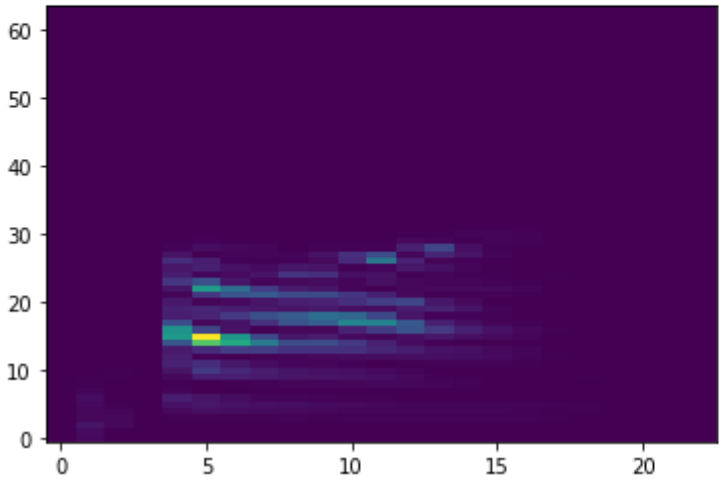
1.4 Spectrogram

- Mel-Spectrogram

```
[ ]  mel_converter = T.MelSpectrogram(sample_rate=8000,n_mels=64,n_fft=256,hop_length=n_fft//2)
```

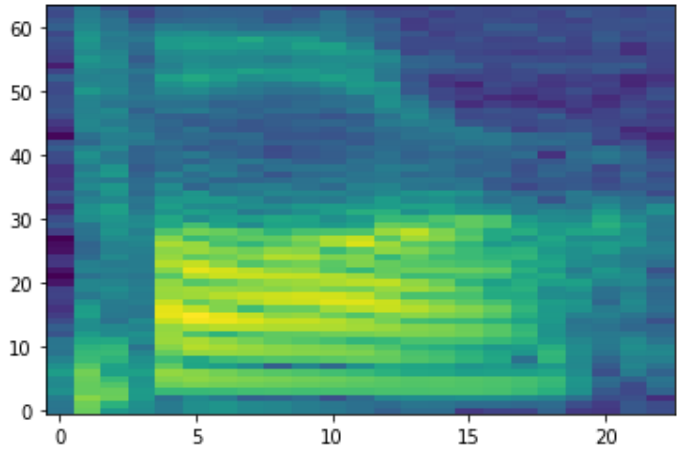
```
[ ]  mel_spec = mel_converter(y)
     plt.imshow(mel_spec[0],aspect='auto',interpolation='nearest',origin='lower')
```

```
<matplotlib.image.AxesImage at 0x7fc4f9aa2790>
```



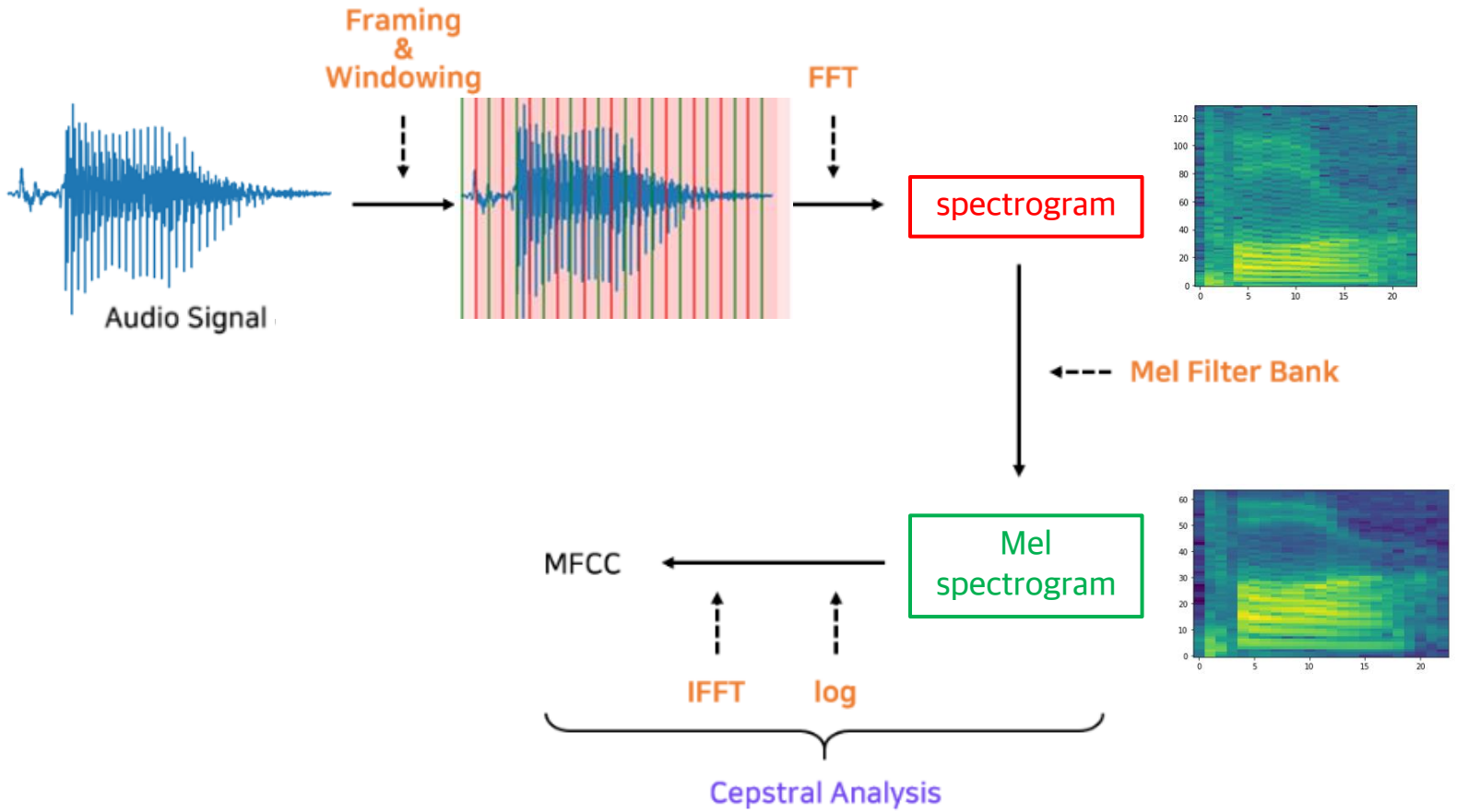

```
[ ]  mel_spec = db_converter(mel_spec)
     plt.imshow(mel_spec[0],aspect='auto',interpolation='nearest',origin='lower')
```

```
<matplotlib.image.AxesImage at 0x7fc4f9ad4390>
```



1.5 MFCC

- **Mel-Frequency Cepstral Coefficient**



Input Speech

↓

Preemphasis
Windowing

↓

FFT&Magnitude
Spectrum

↓

Mel-filterbank

↓

Log ( )

↓

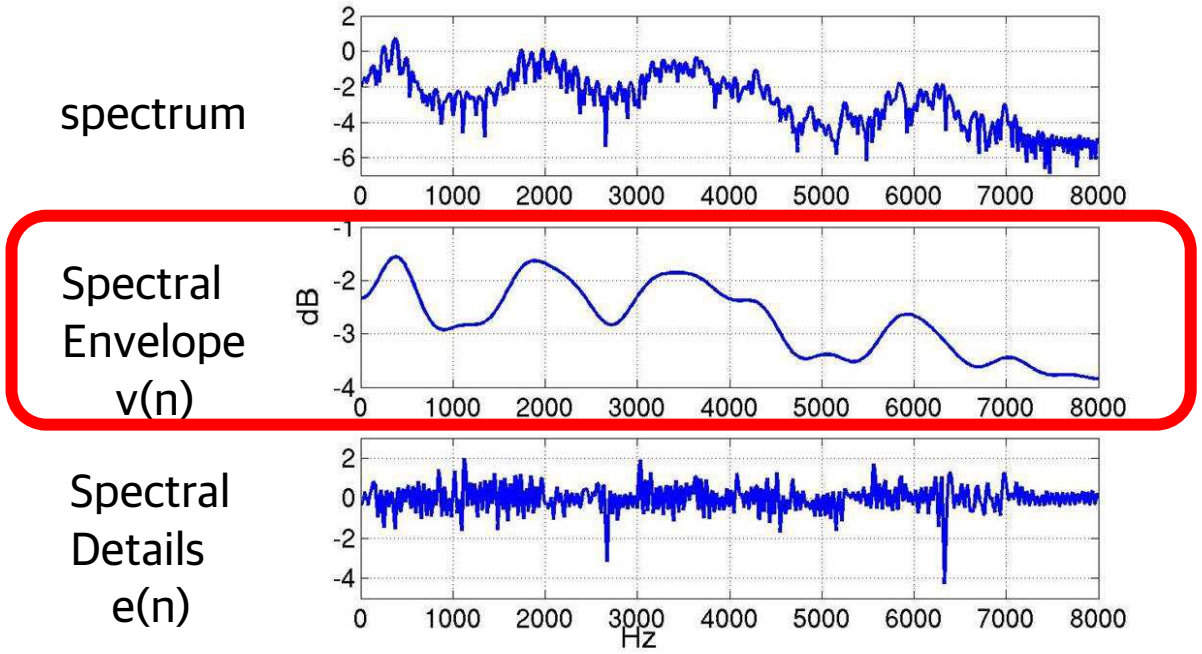
DCT

↓

$o_t=(c_t(0),\dots,c_t(12))$

1.5 MFCC

- **Cepstral Analysis**



1.5 MFCC

```
CLASS torchaudio.transforms.MFCC(
    sample_rate: int = 16000,
    n_mfcc: int = 40,
    dct_type: int = 2,
    norm: str = 'ortho',
    log_mels: bool = False,
    melkwargs: Optional[dict] = None)
```

```
[ ]  melkwargs={
            "n_fft":256,
            "n_mels": 64,
            "hop_length": 256//2,
            "mel_scale": "htk",
        }
     mfcc_converter = T.MFCC(sample_rate=8000,n_mfcc=40,melkwargs=melkwargs)
```

```
[ ]  mfcc = mfcc_converter(y)
     mfcc.shape
```

```
torch.Size([1, 40, 23])
```

```
[ ]  #mfcc = db_converter(mfcc)
     plt.imshow(mfcc[0],origin='lower',aspect='auto',interpolation='nearest')
```

```
<matplotlib.image.AxesImage at 0x7fc4f9699f90>
```