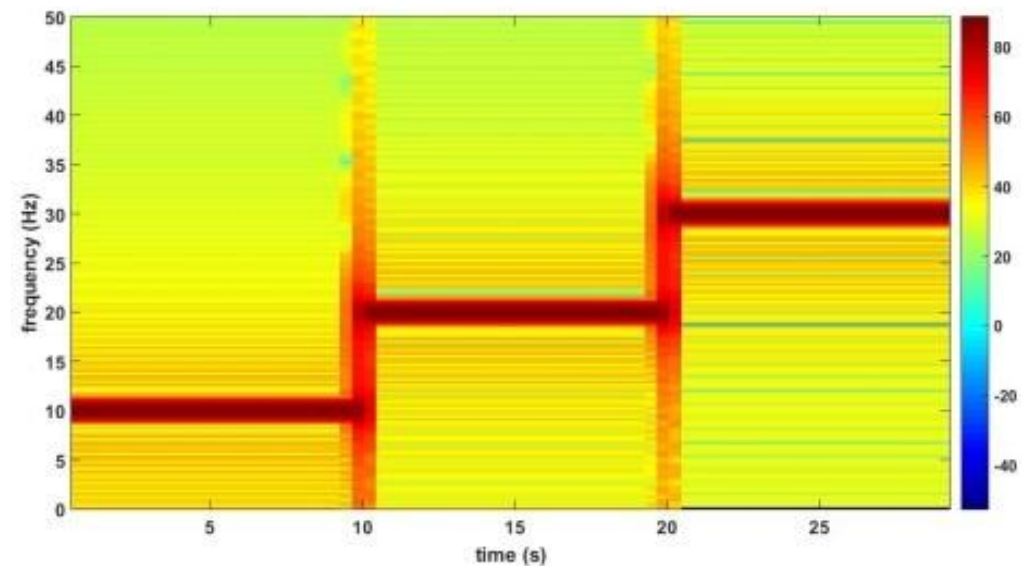


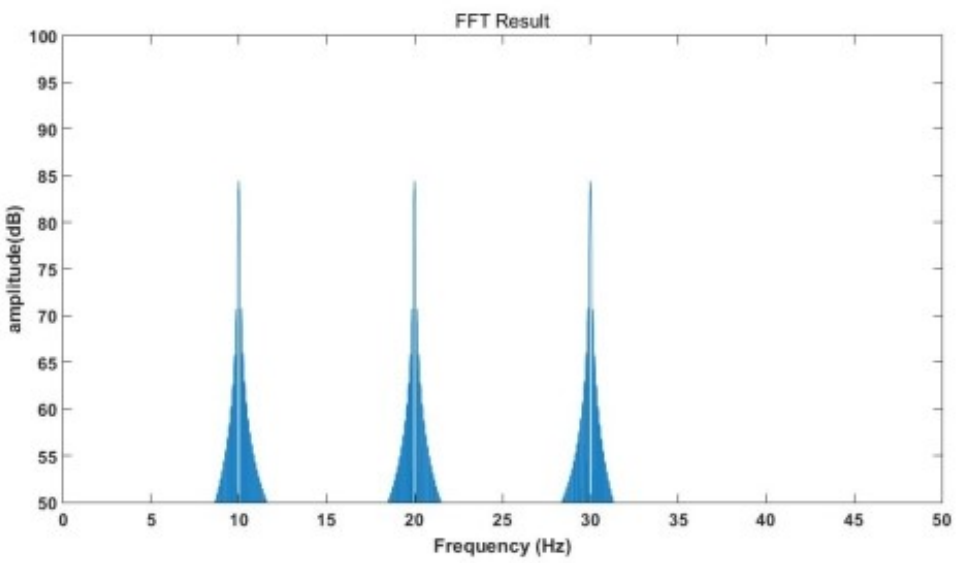
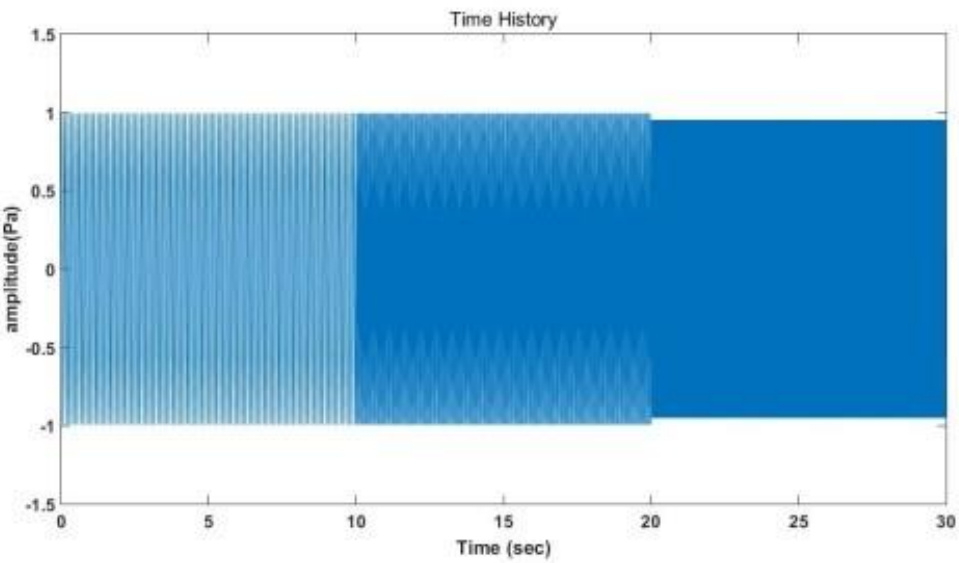
Chapter 09. 시계열을 활용한 딥러닝 (Time Sequence Processing)

STFT, MFCC, MelSpectrogram



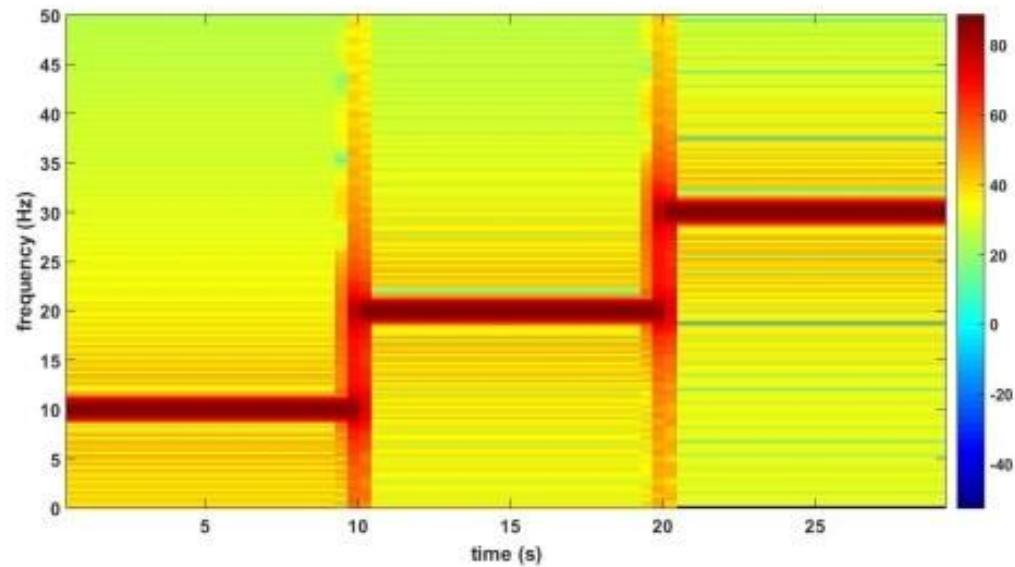
STFT

<http://blog.naver.com/PostView.nhn?blogId=vmv-tech&logNo=220936084562&parentCategoryNo=&categoryNo=17&viewDate=&isShowPopularPosts=false&from=postView>



어떤 주파수를 얼마나 가지고 있는지 명백히 보여주지만, 시간의 흐름에 따라서 어느시간대에 주파수가 얼마나 변했는지 알기 어려운 단점

STFT

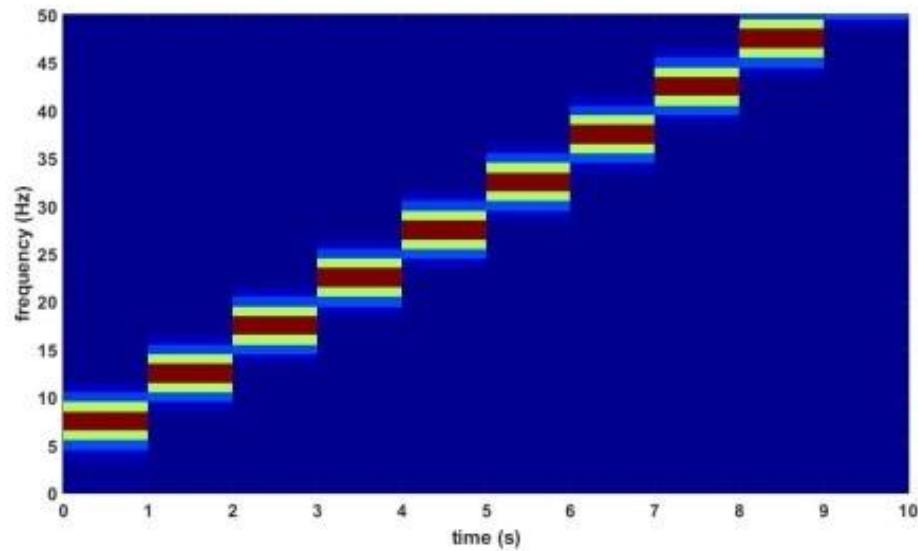


– 시간과 주파수 2개의 축

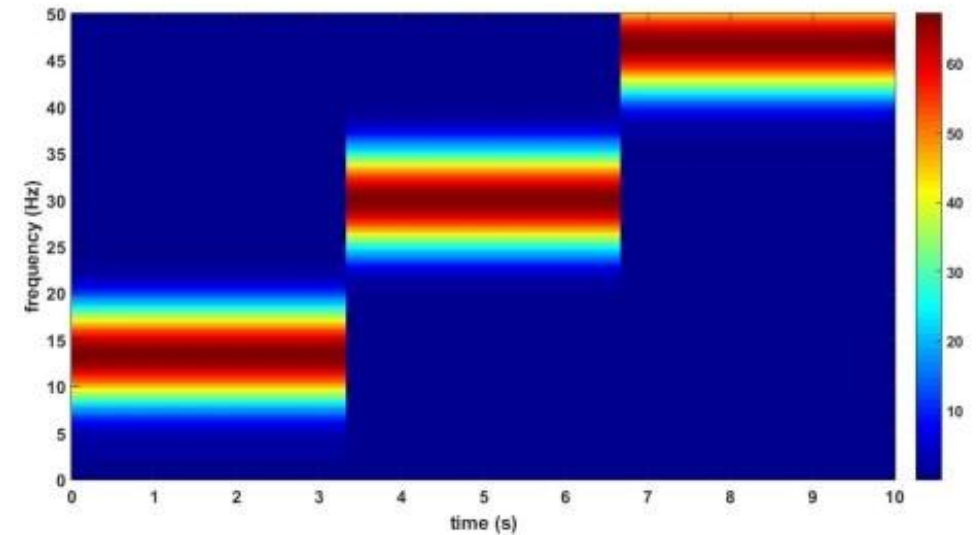
– STFT를 하게 되면 신호에 대해 자신이 알고 싶은 시점에서의 주파수 성분을 알 수 있습니다

STFT

<http://blog.naver.com/PostView.nhn?blogId=vmv-tech&logNo=220941821088&parentCategoryNo=&categoryNo=17&viewDate=&isShowPopularPosts=false&from=postView>

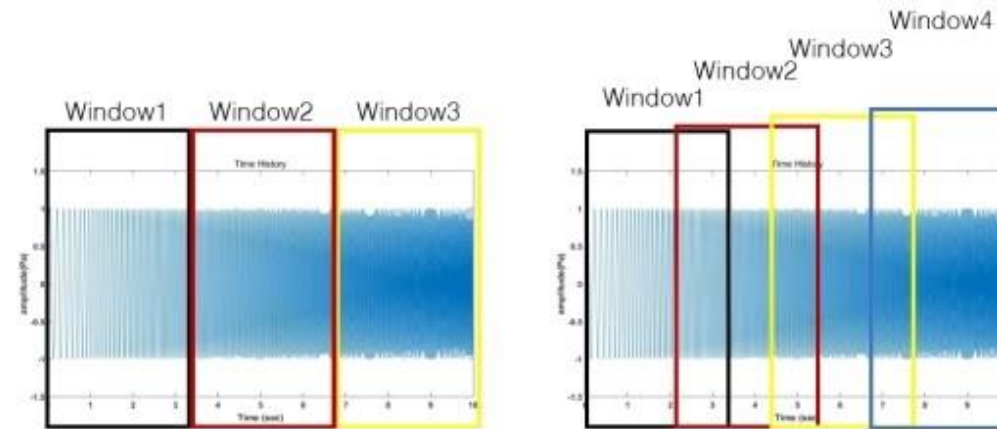


Window length=1 초

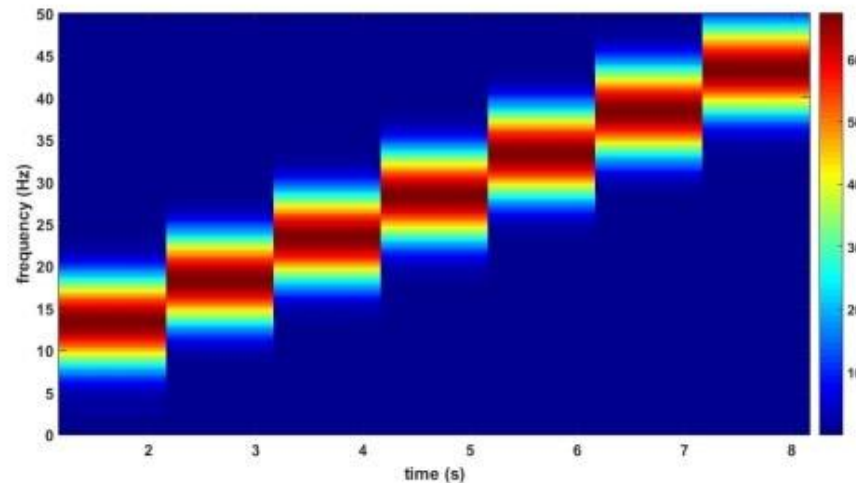


Window length=3.3 초

STFT



No overlap(왼쪽), Overlap 적용(오른쪽)



Overlap 70%(Window가 70% 겹쳐짐) 시간에 대한 Resolution이 좋아짐

MFCC(Mel Frequency Cepstral Coefficient)

사람의 귀가 주파수 변화에 반응하게 되는 양상이 선형적이지 않고 로그스케일과 비슷한 멜스케일을 따르는 청각적 특성을 반영한 켈프스트럼 계수 추출 방법

멜 스케일에 따르면 낮은 주파수에서는 작은 변화에도 민감하게 반응하지만, 높은 주파수로 갈수록 민감도가 작아지므로 특징 추출시에 주파수 분석 빈도를 이와 같은 특성에 맞추는 방식이며, 음성 신호의 특징벡터인 선형예측계수를 화자의 변동에 따른 변화와 무관하게 강인한 인식을 유지하는데 도움을 주기 위한 방법이기 때문에 선형예측계수보다 잡음에 훨씬 강인하다.

MFCC

<http://soyoon.github.io/asr/2017/05/04/Voice-Signal-Processing.html>

1. 입력 시간 도메인의 소리 신호를 작은 크기 프레임으로 자른다

-> 짧은 시간에 대해서는 신호가 많이 변하지 않는다.

2. 각 프레임에 대하여 Power Spectrum의

Periodogram estimate (estimate of the spectral density of a signal)를 계산한다.

-> 인간의 들어오는 소리의 주파수에 따라 다른 부분이 진동하는 cochlea (귀에 있는 기관)을 보고 고안

3. 2번에서 구한 Power Spectrum 에 Mel Filter bank를 적용하고, 각 필터에 에너지를 합한다.

-> 두개의 가까운 주파수의 차이를 구별하지 못하므로 다양한 주파수의 영역에 얼마나 에너지가 있는지 알기 위해 사용

MFCC

4. 3번에서 구한 모든 필터 बैं크 에너지의 Log를 취한다.

-> 인간의 듣기(우리는 선형 스케일로 소리의 강도를 듣지 않습니다.)에 영향을 받았습니다

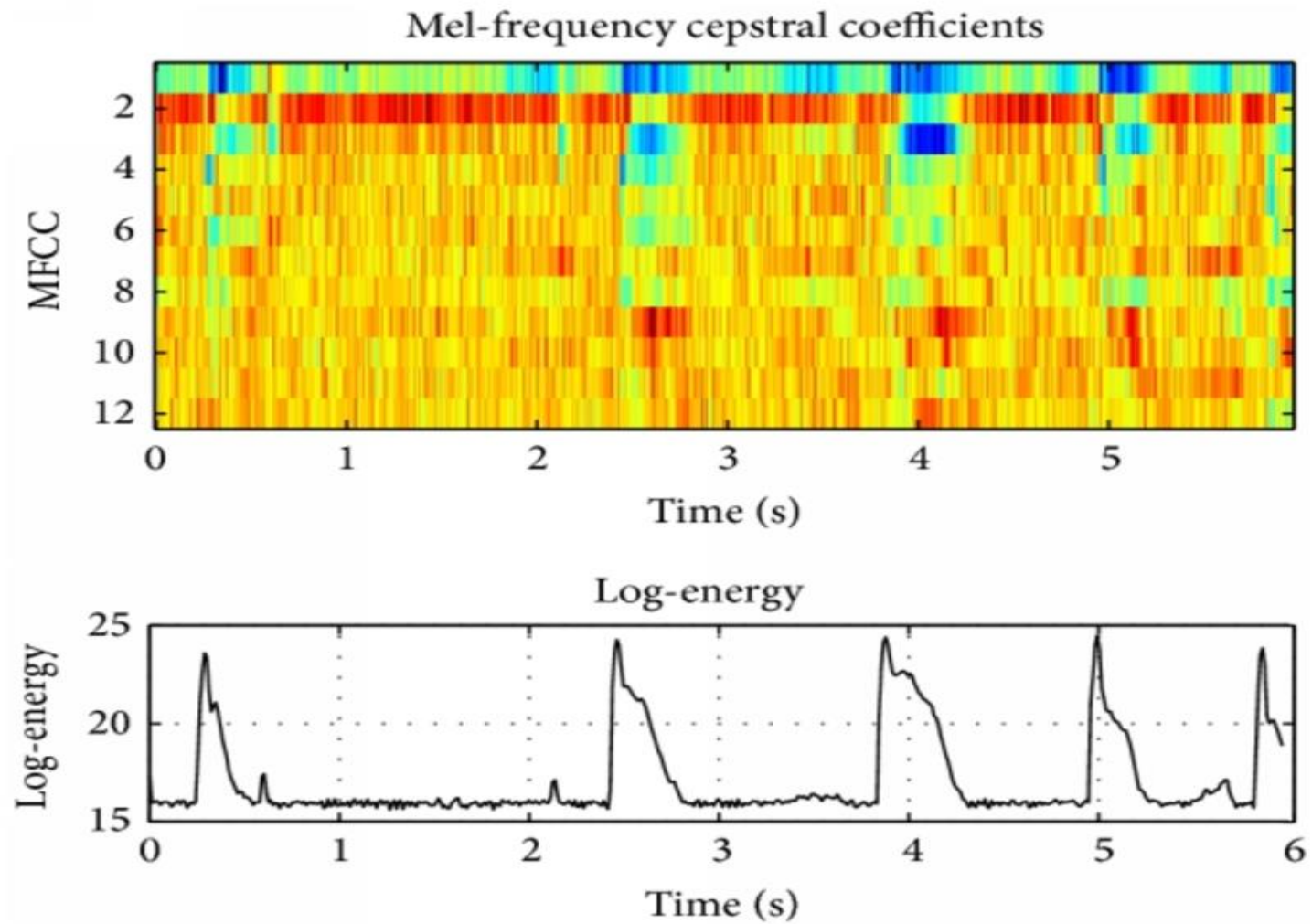
5. 4번 값에 DCT를 취한다.

-> filterbank는 모두 겹치기 때문에, 필터뱅크의 에너지는 서로 꽤나 상관관계가 있습니다. DCT 상관관계를 에너지들의 해제합니다

6. DCT를 취한 값에 Coefficients 2~13 만 남기고 나머지는 버린다.

-> DCT 계수가 높을수록 필터뱅크 에너지의 빠른 변화를 나타내고 이것은 ASR의 성능의 하락을 나타내기 때문입니다

MFCC



Melspectrogram

Melscale $m = 2595 \log_{10}(1 + f/700)$ 이 y 축인 스펙트로그램

1. Separate to windows: Sample the input with windows of size $n_{\text{fft}}=2048$, making hops of size $\text{hop_length}=512$ each time to sample the next window.

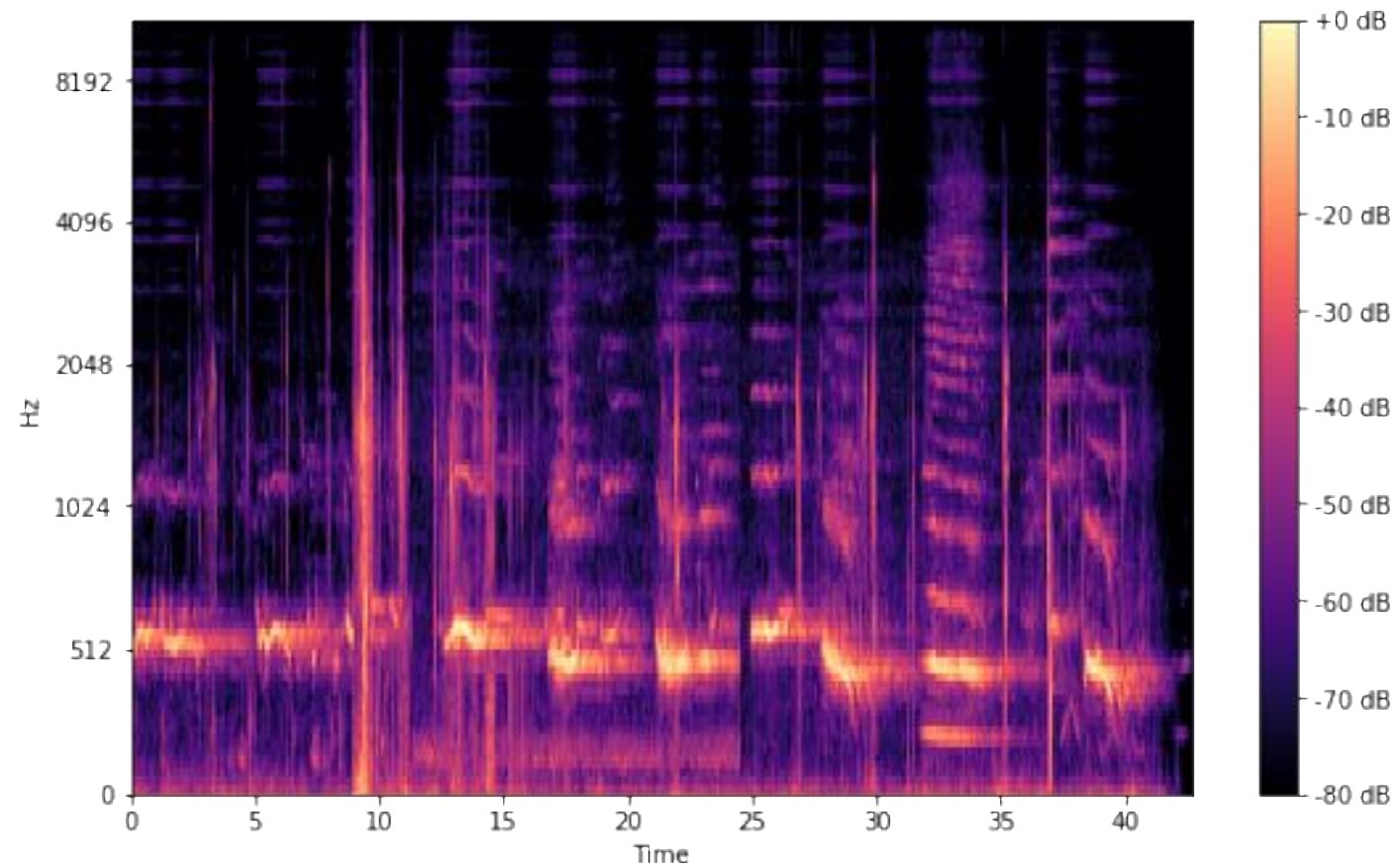
2. Compute FFT (Fast Fourier Transform) for each window to transform from time domain to frequency domain.

3. Generate a Mel scale: Take the entire frequency spectrum, and separate it into $n_{\text{mels}}=128$ evenly spaced frequencies.

And what do we mean by evenly spaced? not by distance on the frequency dimension, but distance as it is heard by the human ear.

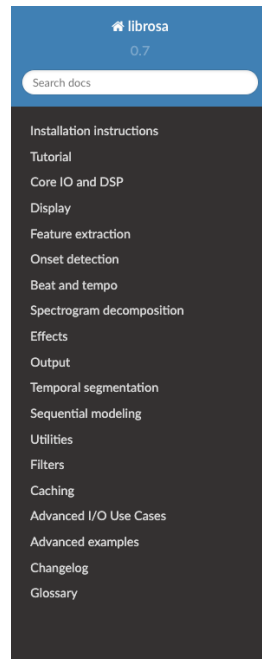
4. Generate Spectrogram: For each window, decompose the magnitude of the signal into its components, corresponding to the frequencies in the mel scale.

Melspectrogram



Code & Library

<https://datascienceschool.net/view-notebook/691326b7f88644f79ec7ddc9f27f84ec/>



Docs » LibROSA

[View page source](#)

LibROSA

LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

For a quick introduction to using librosa, please refer to the [Tutorial](#). For a more advanced introduction which describes the package design principles, please refer to the [librosa paper](#) at [SciPy 2015](#).

Getting started

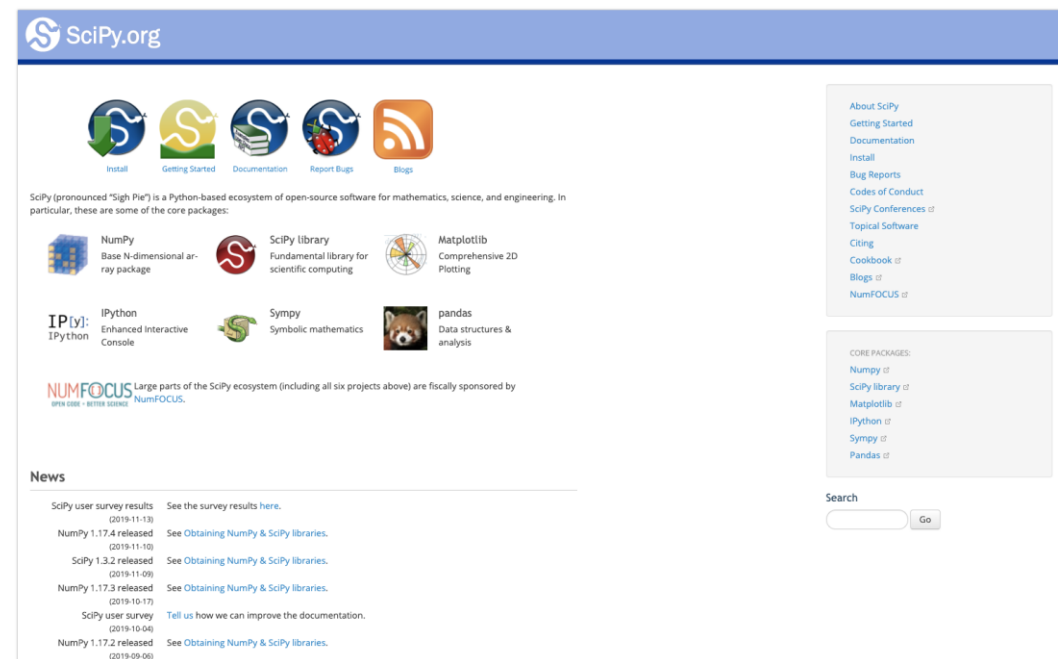
- [Installation instructions](#)
- [Tutorial](#)

Troubleshooting

If you have questions about how to use librosa, please consult the [discussion forum](#). For bug reports and other, more technical issues, consult the [github issues](#).

API documentation

- [Core IO and DSP](#)
- [Display](#)
- [Feature extraction](#)
- [Onset detection](#)
- [Beat and tempo](#)



- *Thank you*