# Speech Features

Fundamental representations of spoken audio

DL-NLP RG

# Motivation

- Most ASR: non-raw audio features as input to ML models
- Lots of these features (MFCC, log-mel-spectrogram, filterbank etc.)
- Features matter for performance
- Creating these features can be mysterious
  - What does the ‘log’ in log-mel-spectrogram refer to?
- Few centralized sources of knowledge

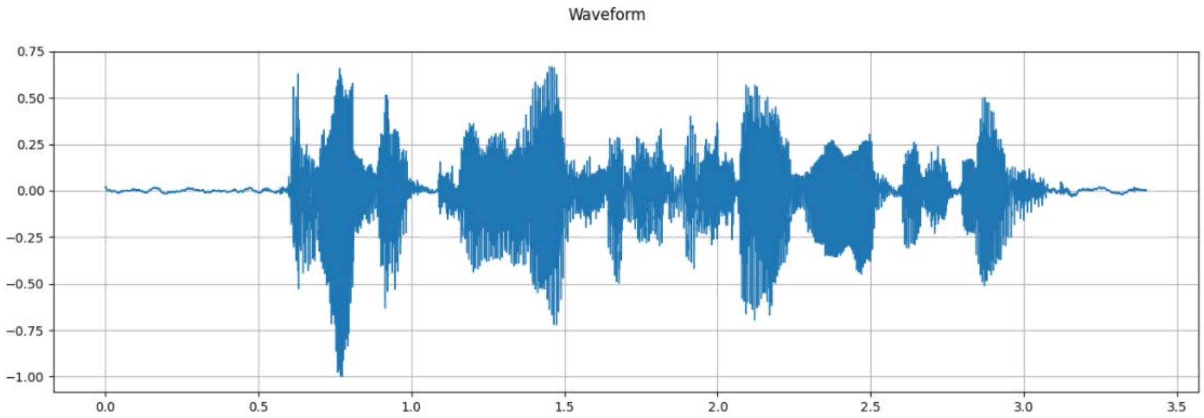
Goal: Clarify this ‘pipeline’ with hands on examples to act as introduction for future work in speech processing.

# Overview

- Waveforms
- Fourier Transforms
  - Discrete Fourier Transform
  - Short-time Fourier Transform
- Spectrograms
- Mel-scale Mel-filterbank
- Mel-spectrogram and log-mel-spectrogram
- Mel-feature Cepstral Coefficients
- GriffinLim

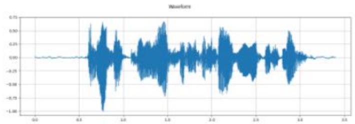
# Waveforms

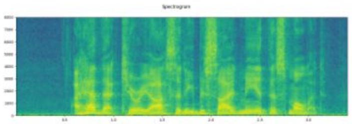
- A sound wave is a change in pressure (disturbance of a medium) that travels through the atmosphere.
- If we wish to record this sound wave, all we have to do is measure and record the air pressure in the atmosphere over time.
- This is captured by microphones, which transform sound waves' mechanical energy into electric energy.
- **Waveform** is the measurement of the air pressure over time.
- **Sampling rate**
    - The frequency at which we capture these electric voltages (amplitudes/pressures). In other words, number of electric voltage values noted down in one second.



# Waveforms (cont)

```
waveform, sample_rate = torchaudio.load(SAMPLE_WAV_SPEECH_PATH)

print_stats(waveform, sample_rate=sample_rate)
plot_waveform(waveform, sample_rate)
plot_specgram(waveform, sample_rate)
play_audio(waveform, sample_rate)
```



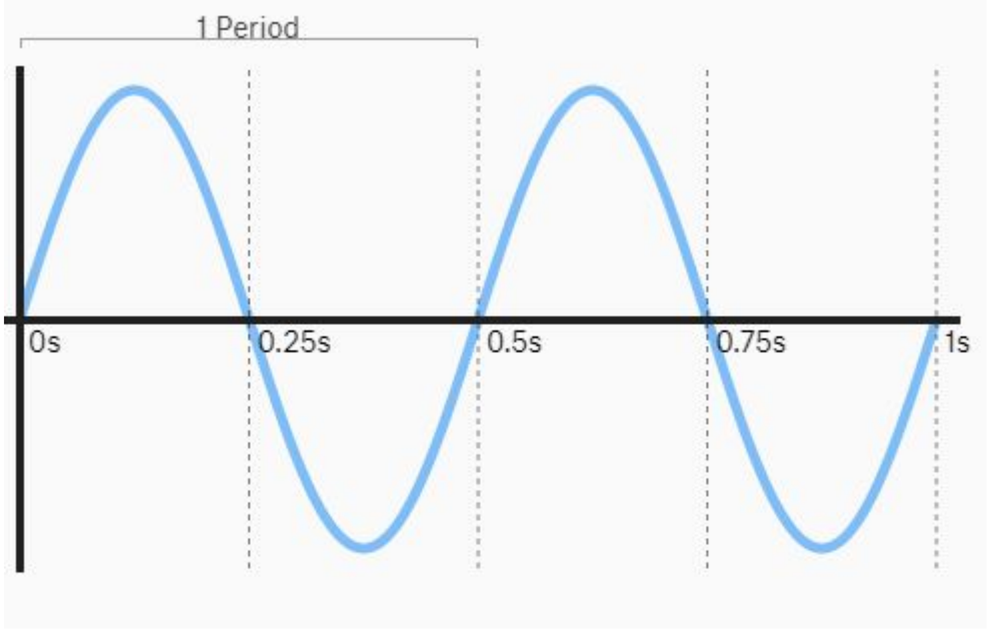


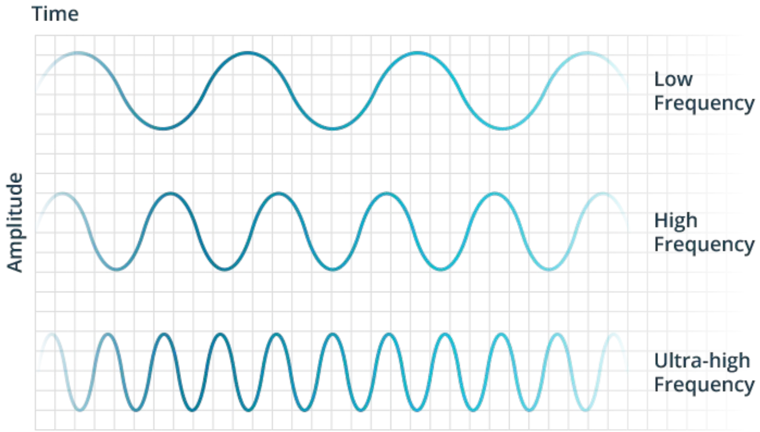
Out:

```
Sample Rate: 16000
Shape: (1, 54400)
Dtype: torch.float32
 - Max:      0.668
 - Min:     -1.000
 - Mean:     0.000
 - Std Dev:  0.122

tensor([[0.0183, 0.0180, 0.0180,  ..., 0.0018, 0.0019, 0.0032]])

<IPython.lib.display.Audio object>
```

3.4 seconds audio * 16000 = 54400

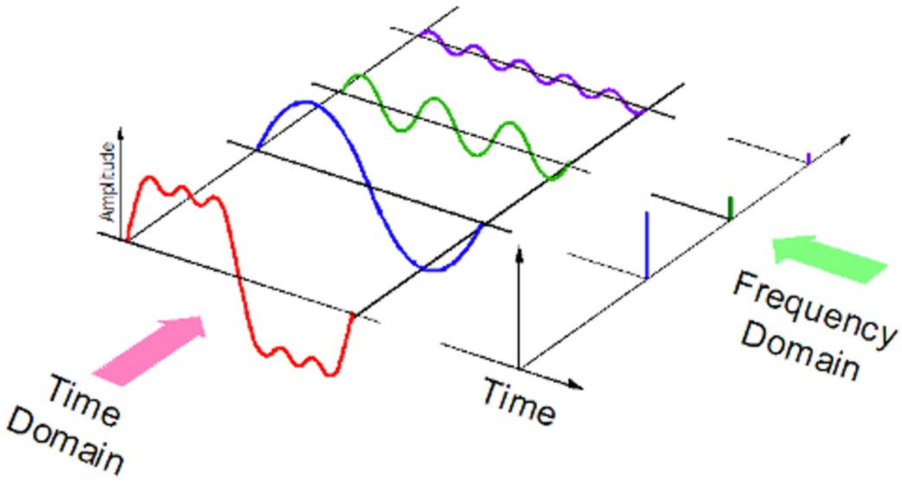
# Frequency


1 Period
0s
0.25s
0.5s
0.75s
1s


Time
Amplitude
Low Frequency
High Frequency
Ultra-high Frequency

# Fourier Transform

- Any digital signal can be expressed by a combination of proper set of sine waves - **Fourier theory**
- **Fourier transform converts time domain to frequency domain.**

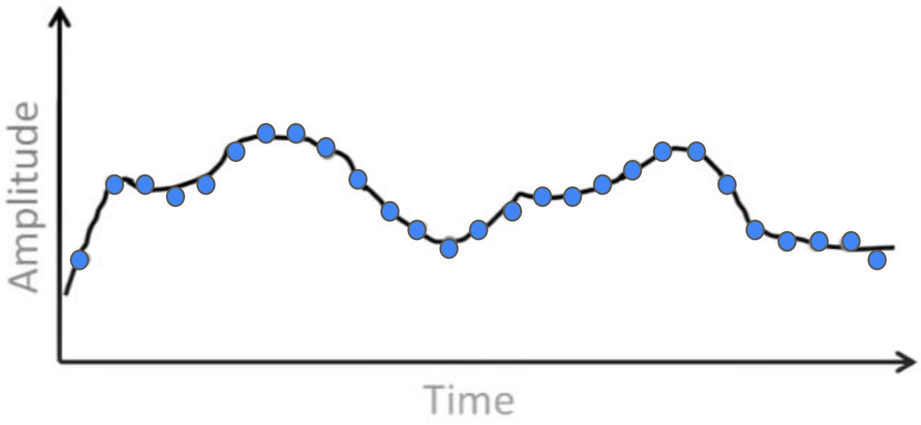



# Discrete Fourier Transform (DFT)

- **Discrete**: signals get sampled at a sample rate.

continuous $$X(F) = \int_{-\infty}^{\infty} x(t)\ e^{-j2\pi Ft} dt$$

discrete $$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}}$$



# Discrete Fourier Transform (DFT)

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}}$$

$b_n$

"kth" frequency bin

$$X_k = x_0 e^{-b_0 j} + x_1 e^{-b_1 j} + \ \ldots + x_n e^{-b_{N-1} j}$$

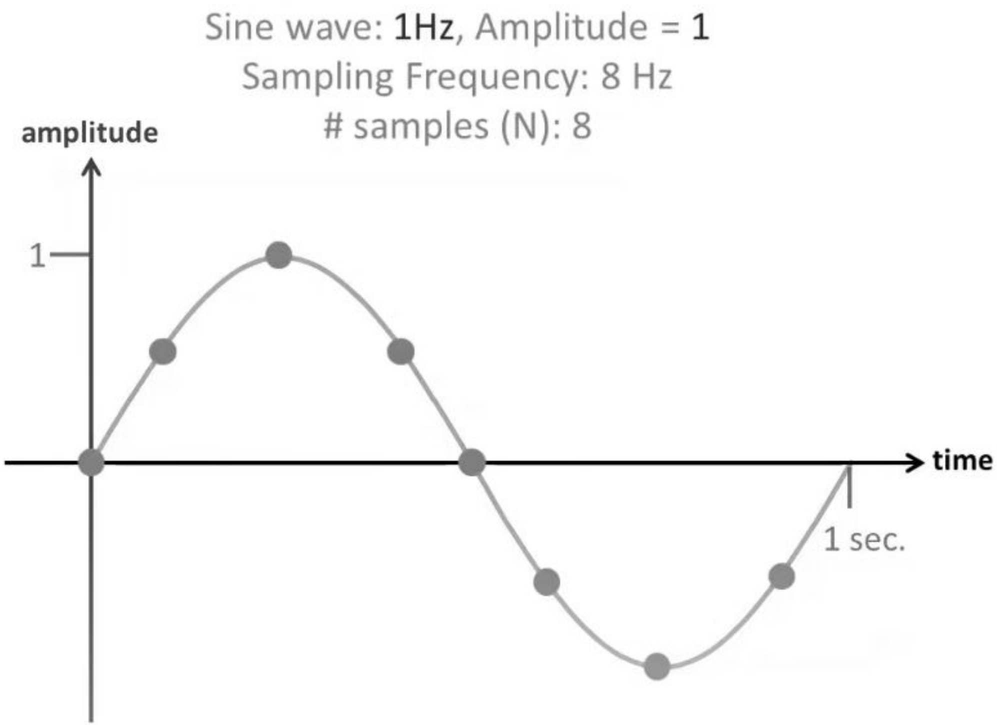
"nth" sample value

Euler's Formula:

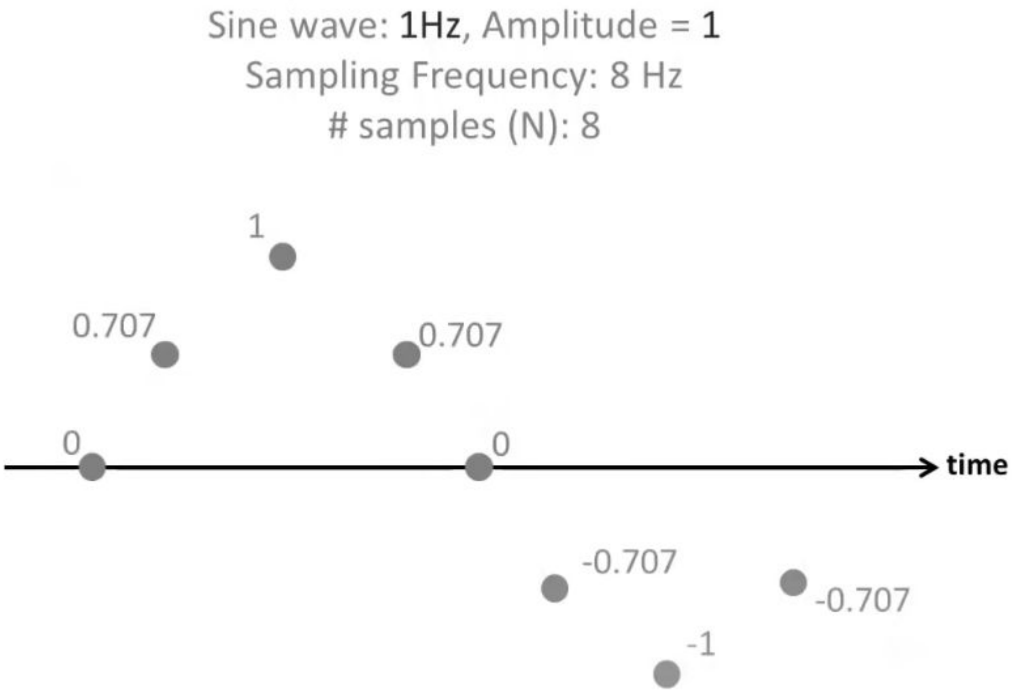
$$e^{jx} = cos\ x + j\ sin\ x$$

$$X_k = x_0[\cos(-b_0) + jsin(-b_0)] + \cdots$$

# Discrete Fourier Transform (DFT)



# Discrete Fourier Transform (DFT)



# Discrete Fourier Transform (DFT)

Sine wave: 1Hz, Amplitude = 1
Sampling Frequency: 8 Hz
# samples (N): 8

"kth" frequency bin

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j\,2\pi k n}{N}}$$

$x_0 = 0$
$x_1 = 0.707$
$x_2 = 1$
$x_3 = 0.707$
$x_4 = 0$
$x_5 = -0.707$
$x_6 = -1$
$x_7 = -0.707$

$$X_1 = 0 \cdot e^{-\frac{j\,2\pi(1)(0)}{8}} + 0.707 \cdot e^{-\frac{j\,2\pi(1)(1)}{8}} + 1 \cdot e^{-\frac{j\,2\pi(1)(2)}{8}} + \cdots$$

$$X_1 = 0 + 0.707\left[cos\left(-\frac{\pi}{4}\right) + jsin\left(-\frac{\pi}{4}\right)\right] + 1\left[cos\left(-\frac{\pi}{2}\right) + jsin(-\frac{\pi}{2})\right] + \ldots$$

$$X_1 = 0 + (0.5 - 0.5j) + (\text{-}j) + (-0.5 - 0.5j) + (0.5 - 0.5j) + (-j) + (-0.5 - 0.5j)$$

$$X_1 = -4j$$

# Discrete Fourier Transform (DFT)

Sine wave: 1Hz, Amplitude = 1
Sampling Frequency: 8 Hz
# samples (N): 8

Fourier Coefficients:

$$X_0 = 0$$
$$X_1 = 0 - 4j$$
$$X_2 = 0$$
$$X_3 = 0$$
$$X_4 = 0$$
$$X_5 = 0$$
$$X_6 = 0$$
$$X_7 = 0 + 4j$$

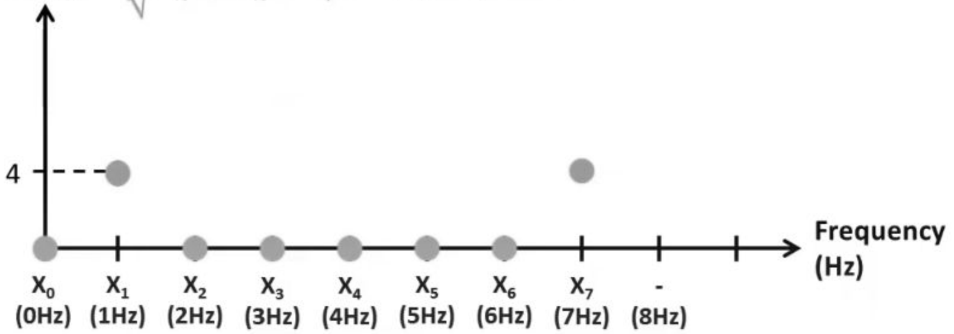
# Discrete Fourier Transform (DFT)

Sine wave: 1Hz, Amplitude = 1
Sampling Frequency: 8 Hz
# samples (N): 8

Fourier Coefficients:

$$
\begin{aligned}
X_0 &= 0 \\
X_1 &= 0 - 4j \\
X_2 &= 0 \\
X_3 &= 0 \\
X_4 &= 0 \\
X_5 &= 0 \\
X_6 &= 0 \\
X_7 &= 0 + 4j
\end{aligned}
$$

$$Mag = \sqrt{A_k^2 + B_k^2} = \sqrt{0^2 + (-4)^2} = 4$$



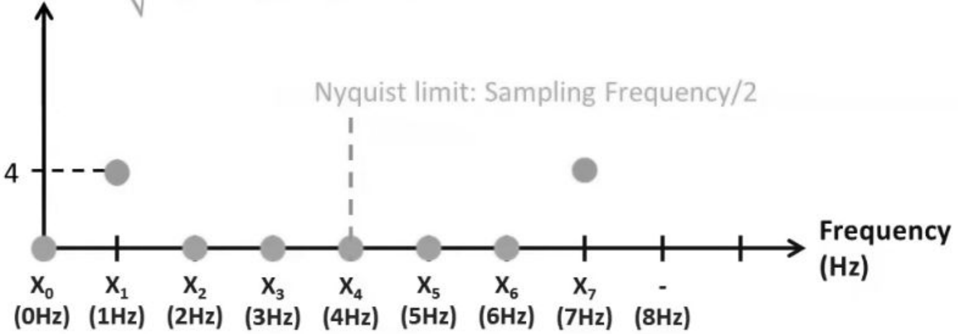
# Discrete Fourier Transform (DFT)

Sine wave: 1Hz, Amplitude = 1
Sampling Frequency: 8 Hz
# samples (N): 8

Fourier Coefficients:

$$
\begin{aligned}
X_0 &= 0 \\
X_1 &= 0 - 4j \\
X_2 &= 0 \\
X_3 &= 0 \\
X_4 &= 0 \\
X_5 &= 0 \\
X_6 &= 0 \\
X_7 &= 0 + 4j
\end{aligned}
$$

$$Mag = \sqrt{A_k^2 + B_k^2} = \sqrt{0^2 + (-4)^2} = 4$$



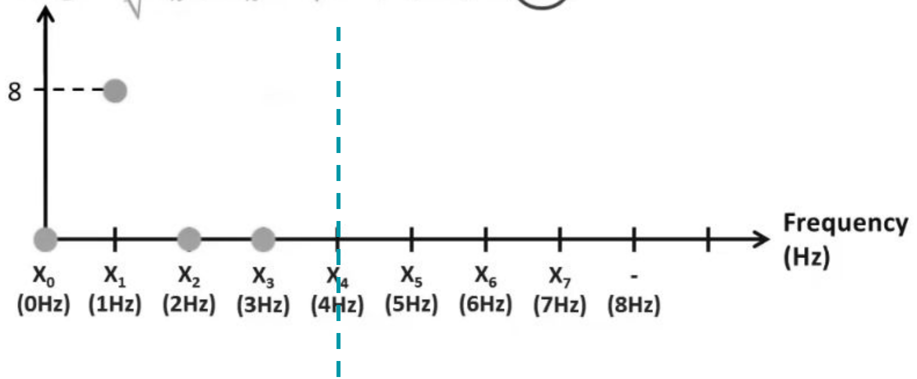
# Discrete Fourier Transform (DFT)

Sine wave: 1Hz, Amplitude = 1
Sampling Frequency: 8 Hz
# samples (N): 8

Single Sided
Fourier Coefficients:

$$X_0 = 0$$
$$X_1 = 0 - 8j$$
$$X_2 = 0$$
$$X_3 = 0$$

$$Mag = \sqrt{A_k^2 + B_k^2} = \sqrt{0^2 + (-8)^2} = 8$$



# Application of Fourier Transform in Text Classification

[FNet: Mixing Tokens with Fourier Transforms]

*Sep 2021*

FNets achieve 92 and 97% of their respective BERT-Base and BERT-Large counterparts' accurac on the GLUE benchmark, but train **80% faster on GPUs and 70% faster on TPUs.**

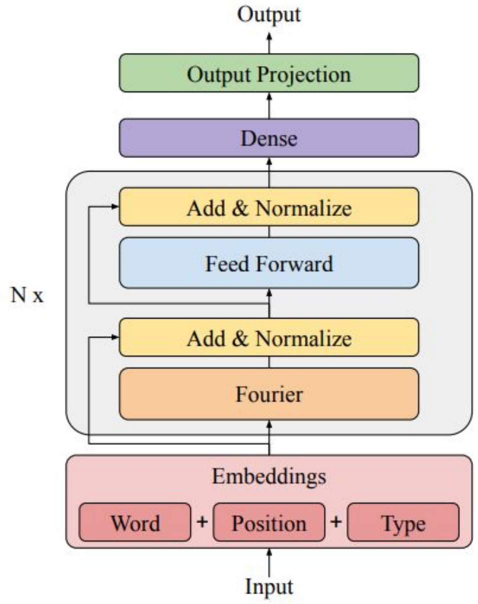


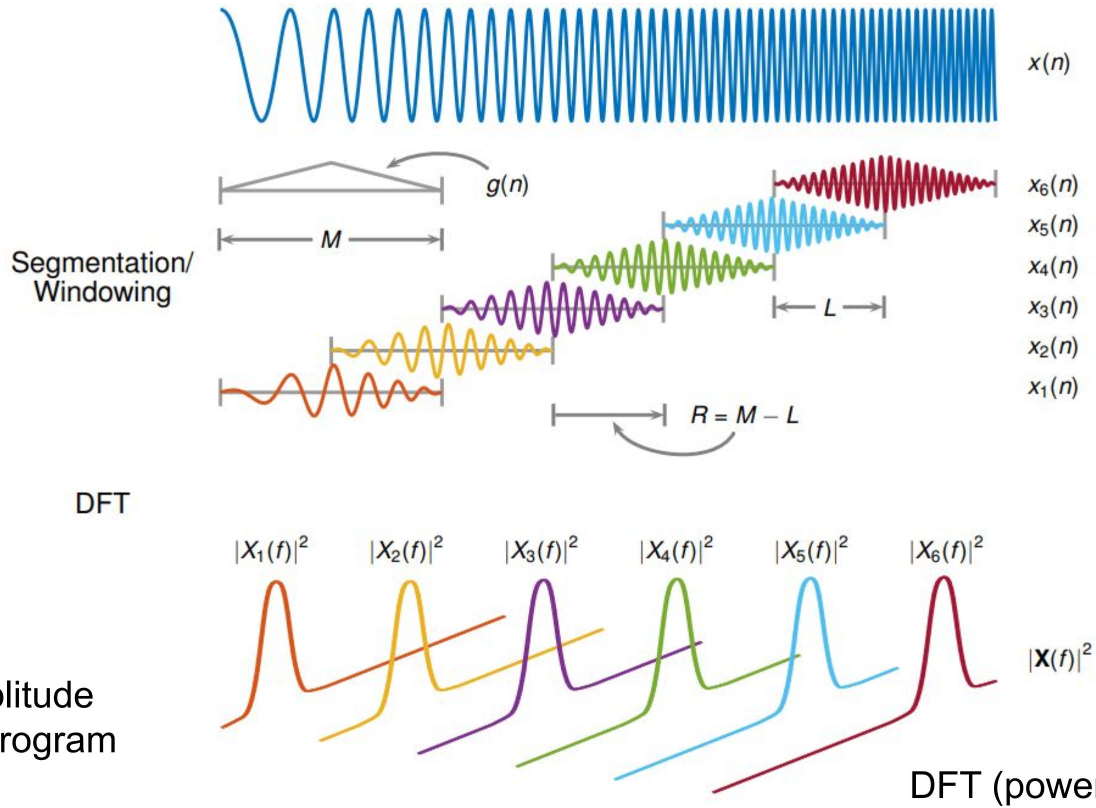

Figure 1: FNet encoder architecture with $N$ encoder blocks.

# Short-Time Fourier Transforms (STFT)

STFT: Take the DFT at each time step using a sliding window

Why? Allows us to deal with non-stationary signals.

L: overlap
M: window length
R: hop length
x: signal
X: DFT of x
g(n): window function




Taking the power of the amplitude gives us our standard Spectrogram components

DFT (power)

# Short-Time Fourier Transforms (STFT)

We calculate a matrix, where each row is the DFT at t=mR:

$$X_m(f) = \sum_{n=-\infty}^{\infty} x(n)g(n - mR)e^{-j2\pi fn}$$

Nx: length of signal
L: overlap
M: window length
Ndft: sample points used for DFT

f: frequency term
x: signal
X: DFT of x at time=mR
m: bin of STFT matrix
g(n): window function (Hann)
R: hop length

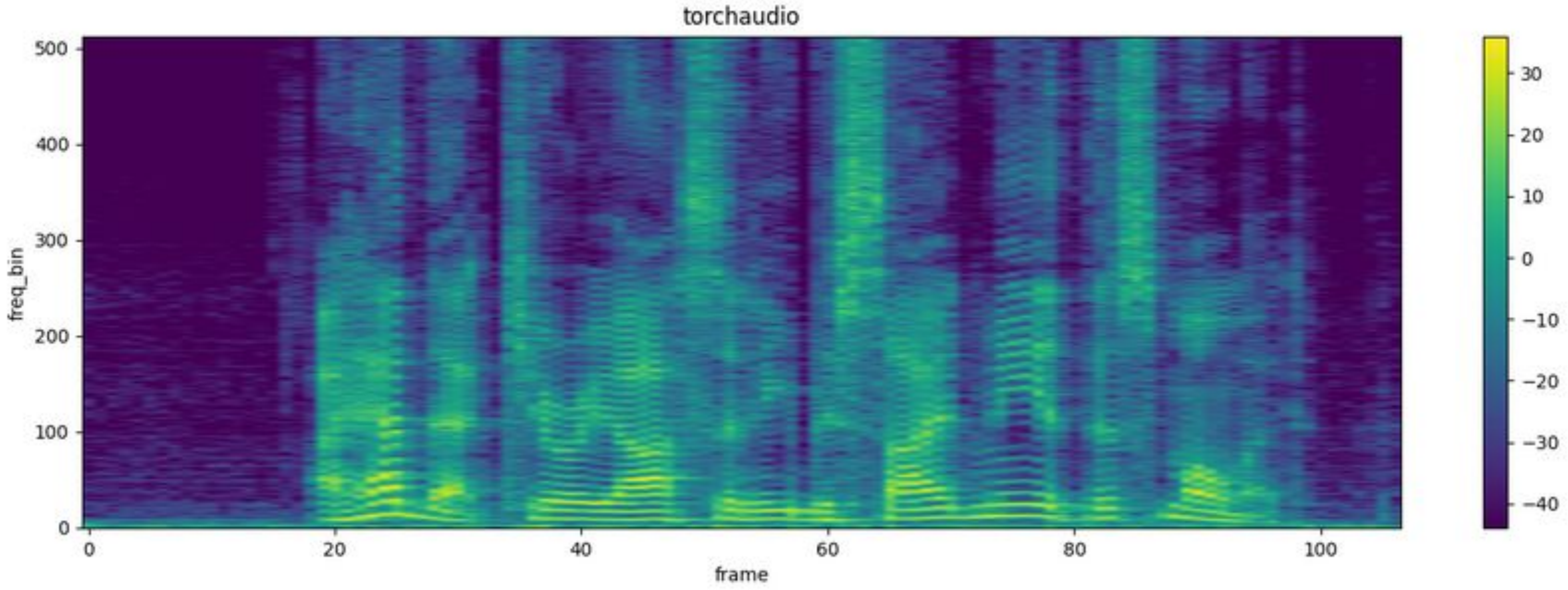
Dimensions of STFT matrix:

$$\left\lfloor \frac{N_x - L}{M - L} \right\rfloor \times \lfloor N_{\mathrm{DFT}}/2 \rfloor + 1$$

time bins freq bins

# Spectrograms

Square STFT values => Spectrogram:




This is a db scale spectrogram, but you could also have units in terms of power

# Mel-scale(s)

“Human-perception” based scale(s) of pitch, which converts from frequency (f) to even pitch units called ‘mels’

One scale: Hidden Markov Toolkit conversion formula:

mel = 2595.0 * np.log10(1.0 + f / 700.0).

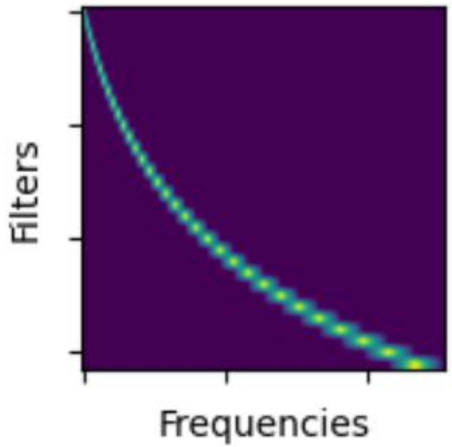
[Note: not a unique conversion formula]

Mel-scale is thus arbitrary, but empirically effective for speech processing.

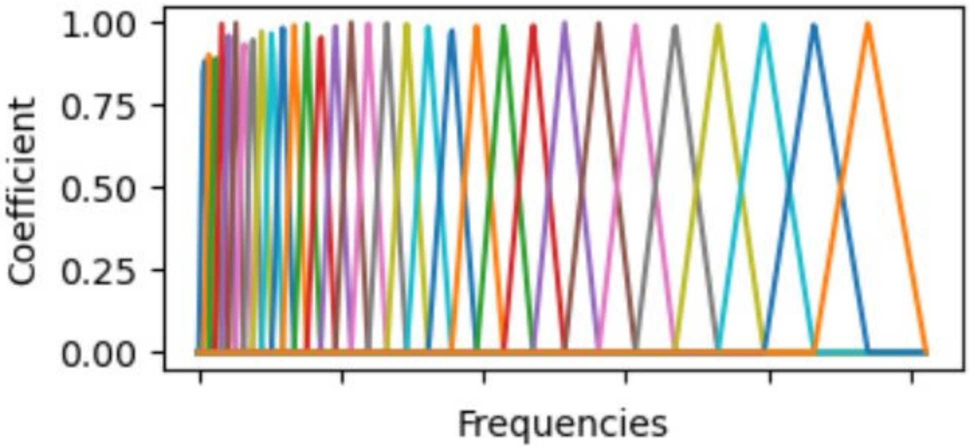
# Filterbanks

A filterbank gives the frequency response of a set of filters for an input signal. → used to convert a signal into a particular frequency scale (e.g. mel)

Frequency to mel map

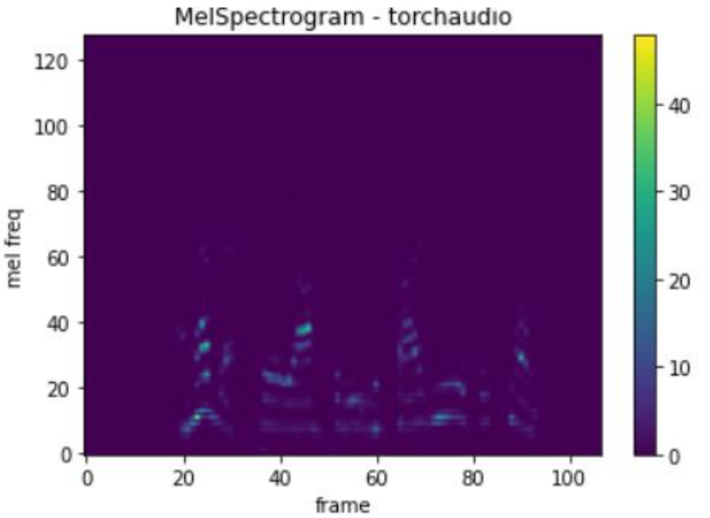



Mel filterbank (librosa and torchaudio)



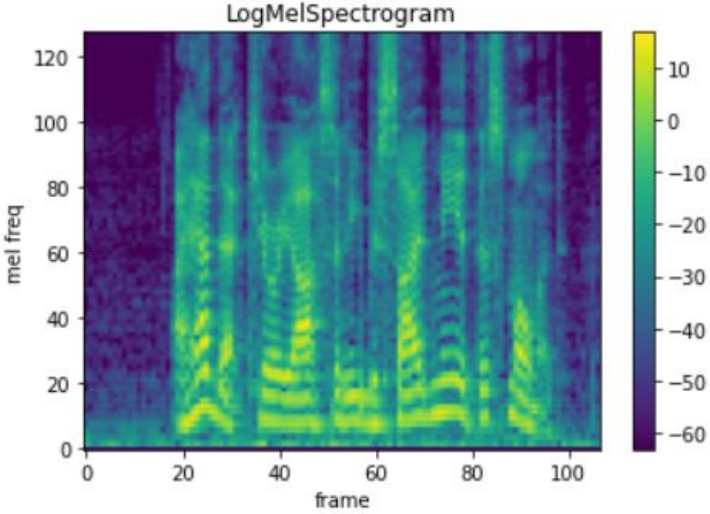
# Mel-scale Spectrogram

Instead of Hz use mels on
y axis



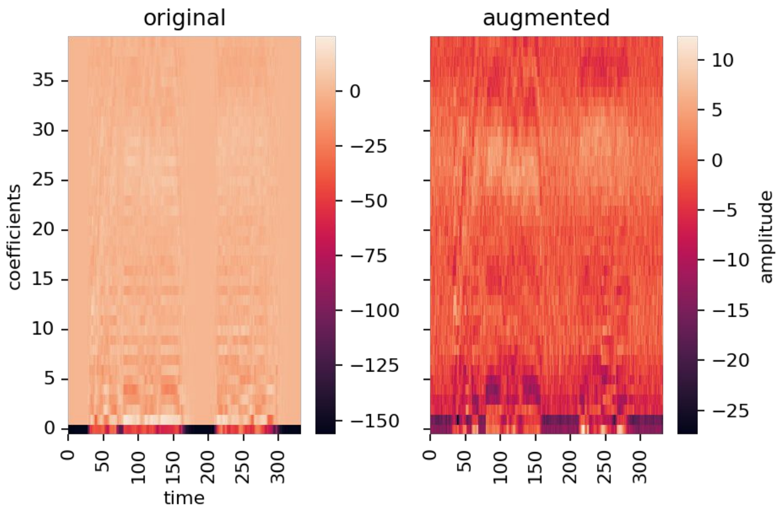

LogMel Spectrogram:

Use decibels instead of
power for amplitude



# Mel-frequency cepstrum

- In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.
- Modification of Mel Filterbank Coefficients
- Apply linear cosine transform
- Remove redundant components




Example of a MFCC Graph

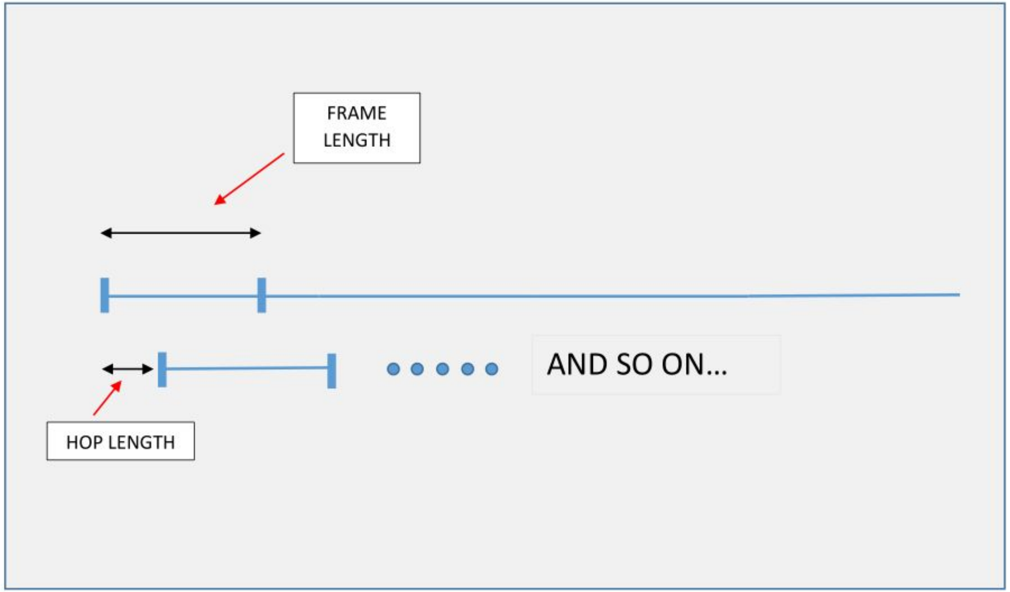
# MFCC

- MFCC coefficients contain information about the rate changes in the different spectrum bands.

```
array([[ -5.229e+02,  -4.944e+02, ...,  -5.229e+02,  -5.229e+02],
       [  7.105e-15,   3.787e+01, ...,  -7.105e-15,  -7.105e-15],
       ...,
       [  1.066e-14,  -7.500e+00, ...,   1.421e-14,   1.421e-14],
       [  3.109e-14,  -5.058e+00, ...,   2.931e-14,   2.931e-14]])
```

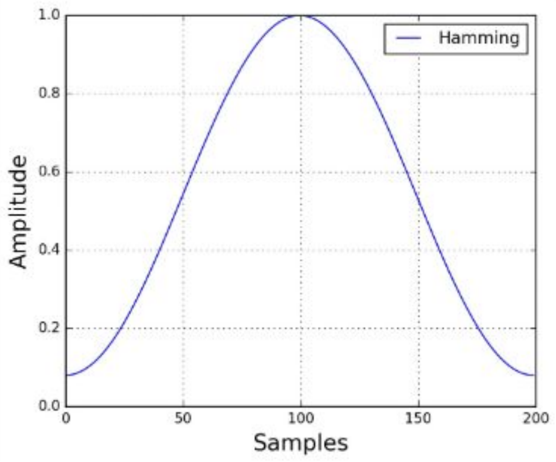
# MFCC Process

- Frame the signal into short frames



# MFCC Process

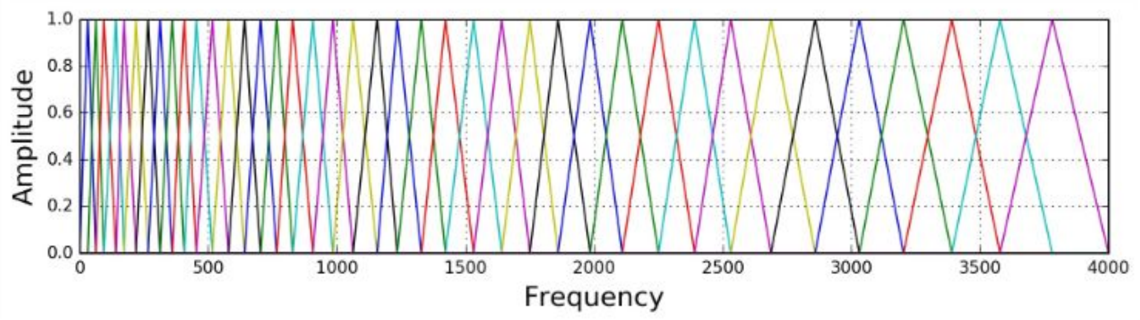
- Windowing is done: Windowing is essentially applied to notably counteract the assumption made by the Fast Fourier Transform that the data is infinite and to reduce spectral leakage.



*Hamming Window*

# MFCC Process

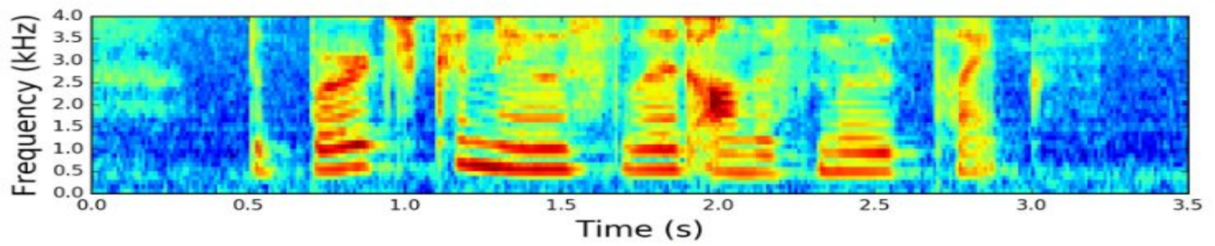
- Calculation of the Discrete Fourier Transform: We can now do an NN-point FFT on each frame to calculate the frequency spectrum, which is also called Short-Time Fourier-Transform (STFT), where NN is typically 256 or 512, NFFT = 512 and then compute the power spectrum
- Applying Filter Banks:




*Filter bank on a Mel-Scale*

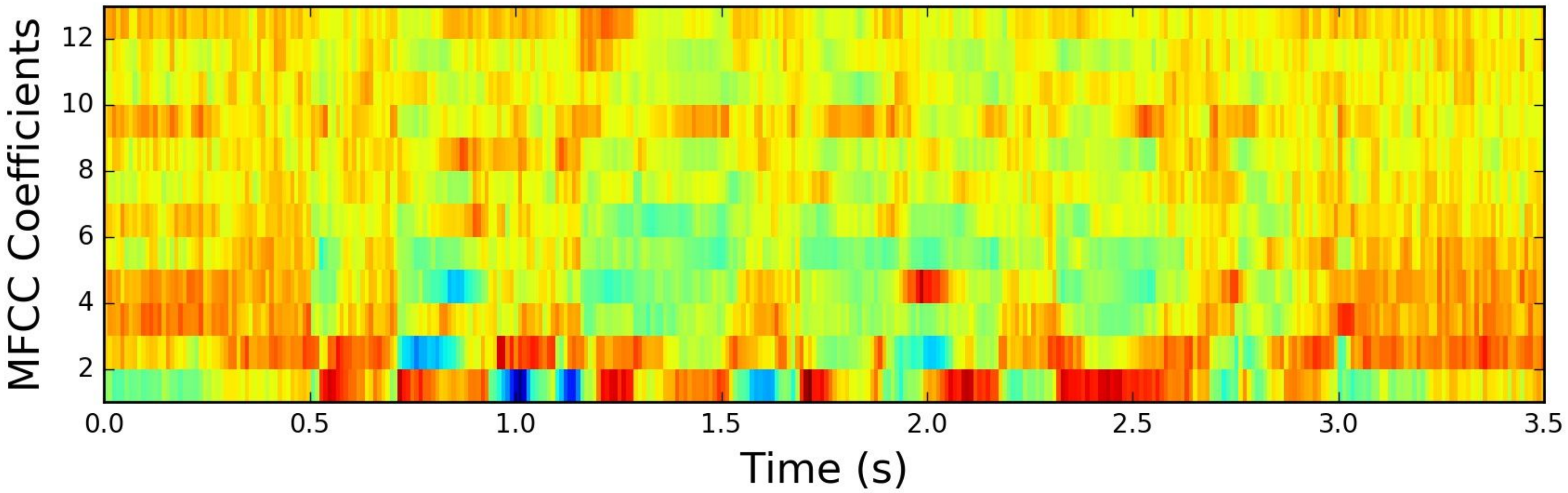
# MFCC Process

- After applying the Filter Banks we are left with the following spectrogram.
- We now apply the log of these spectrogram values to get the log filterbank energies.
- DCT (Discrete cosine transform) is also applied




*Spectrogram of the Signal*

# MFCC



# GriffinLim

Generates reconstructed audio signal from FFT matrix compared with Inverse STFT.