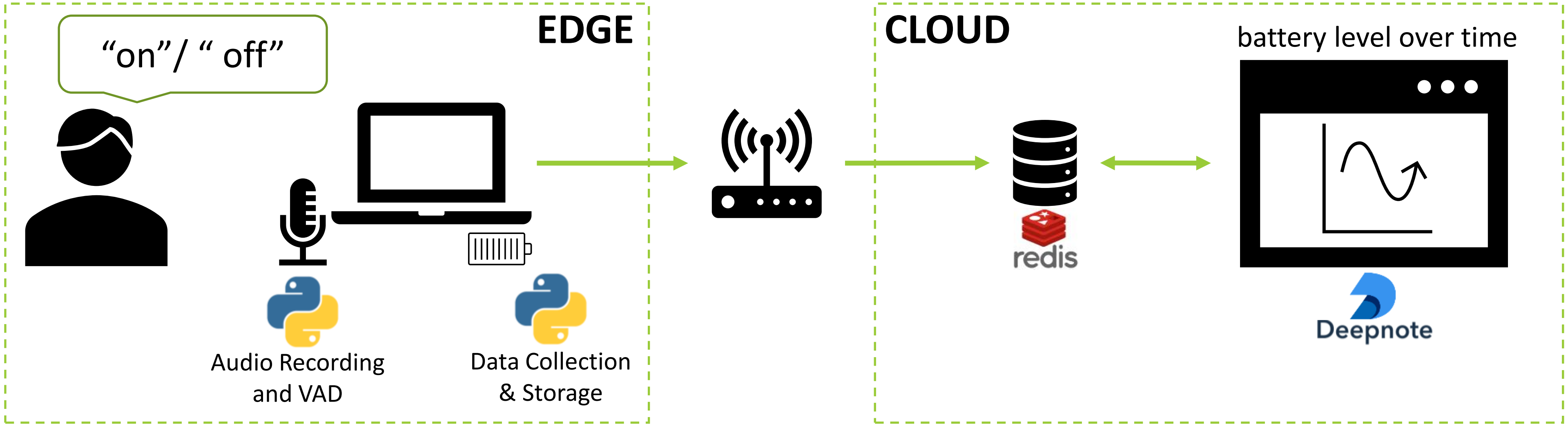
Machine Learning for IoT

LAB2: Pre-processing

LAB1-2: Smart Battery Monitoring (Simplified)



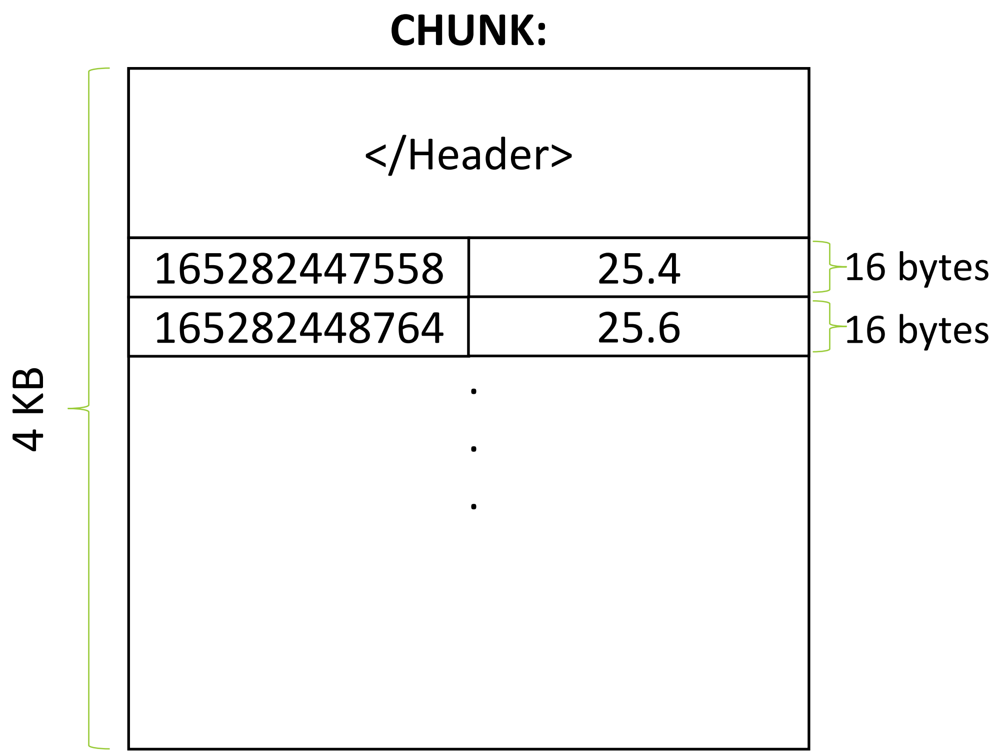
LAB2 Content

- Timeseries Processing:
  - Compression
  - Retention
  - Aggregation
- Audio Processing:
  - Resampling
  - Discrete Fourier Transform
  - Short-Time Fourier Transform
  - Mel Spectrogram
  - Mel-Frequency Cepstral Coefficients

Timeseries Processing

Redis TimeSeries Memory Model

- A Redis TimeSeries consists of a list of linked chunks
- Each chunk contains
  - Header
    - Information needed by Redis to manage the data
  - A set of Records
    - Each record consists of:
      - Timestamp: 64-bit (8 bytes)
      - Value: 64-bit (8 bytes)
- Chunk size is set when creating the TimeSeries
  - Default: 4 KB
  - Smaller → Less Memory, Slower Read/Write
  - Larger → More Memory, Faster Read/Write



TimeSeries Compression

- Lossless compression
  - Gorilla algorithm

Timestamp Compression:

Time	Δ	ΔΔ
--- 5000		
	5000	
-- 10000		0
	5000	
-- 15000		0
	5000	
-- 20000		

Value Compression:

t (°C)	double	XOR
25	0x41c80000	
		0x00000000
25	0x41c80000	
		0x00040000
25.5	0x41cc0000	
		0x00190000
26.625	0x41d50000	
		0x00043333
26.14	0x41d11eb8	

TimeSeries Compression

- Lossless compression
  - Gorilla algorithm
- Memory savings:
  - Depends on data
    - Best-case: 98.4%
    - Worst-case: 113.3%
      - Memory increases! But rare.
    - Average-case: 90.0%
- Compression improves performance due to a lower number of memory accesses

- **Note:** Compression is active by default

Example

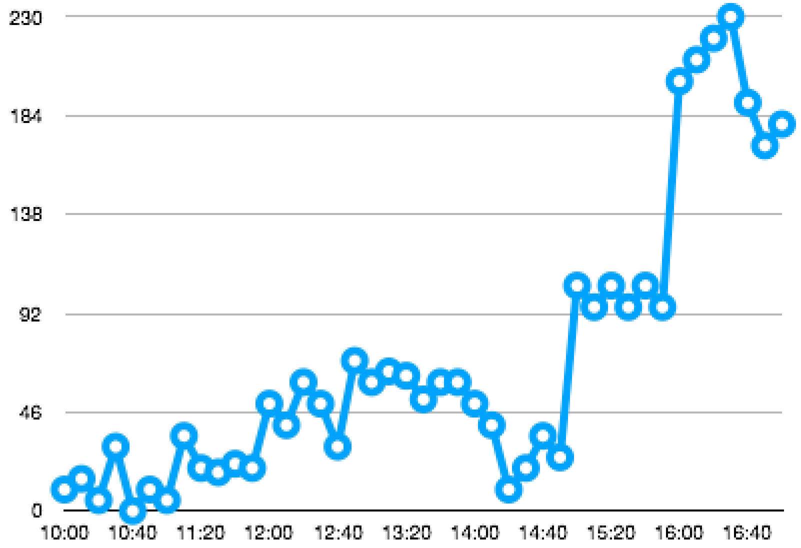
- Which is the memory usage to store temperature every 5 seconds after 1 month?
  - 1 month = 30 days * 24 hours * 60 minutes * 60 seconds = 2592000 seconds
  - # of records = 2592000 / 5 = 518400
  - Uncompressed Memory ≈ 518400 * 16 bytes = 8294400 bytes = 7.910 MB
  - Compressed Memory ≈ 7.910 MB - 90% = 0.791 MB

  - Approximations:
    - We neglected the header size
    - We neglected that the memory usage is always a multiple of the chunk size
    - We considered the average compression ratio

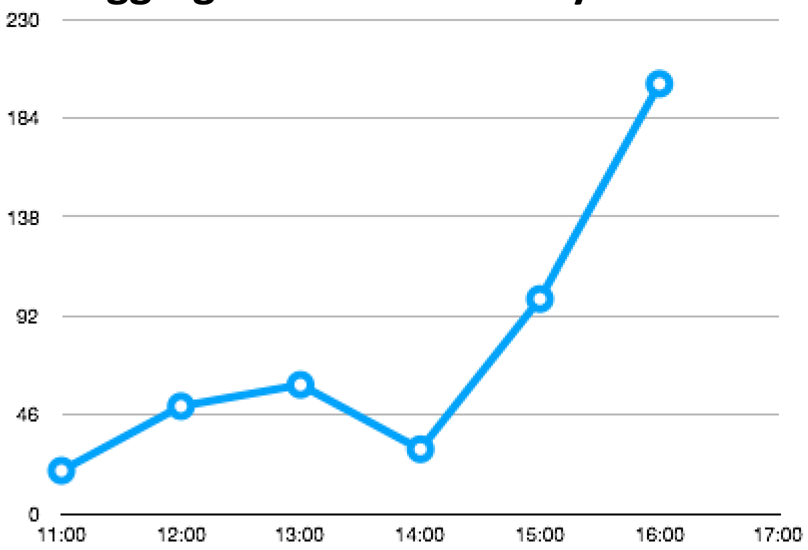
TimeSeries Aggregation

- Lossy Compression
- Aggregation Parameters:
  - Bucket Duration
  - Aggregation type: avg, sum, min, max, range, count, first, last.
- **<u>Note:</u>** Aggregation never changes the original timeseries but creates a new one

Original: 1 value every 8 minutes



Aggregated: 1 value every 1 hour



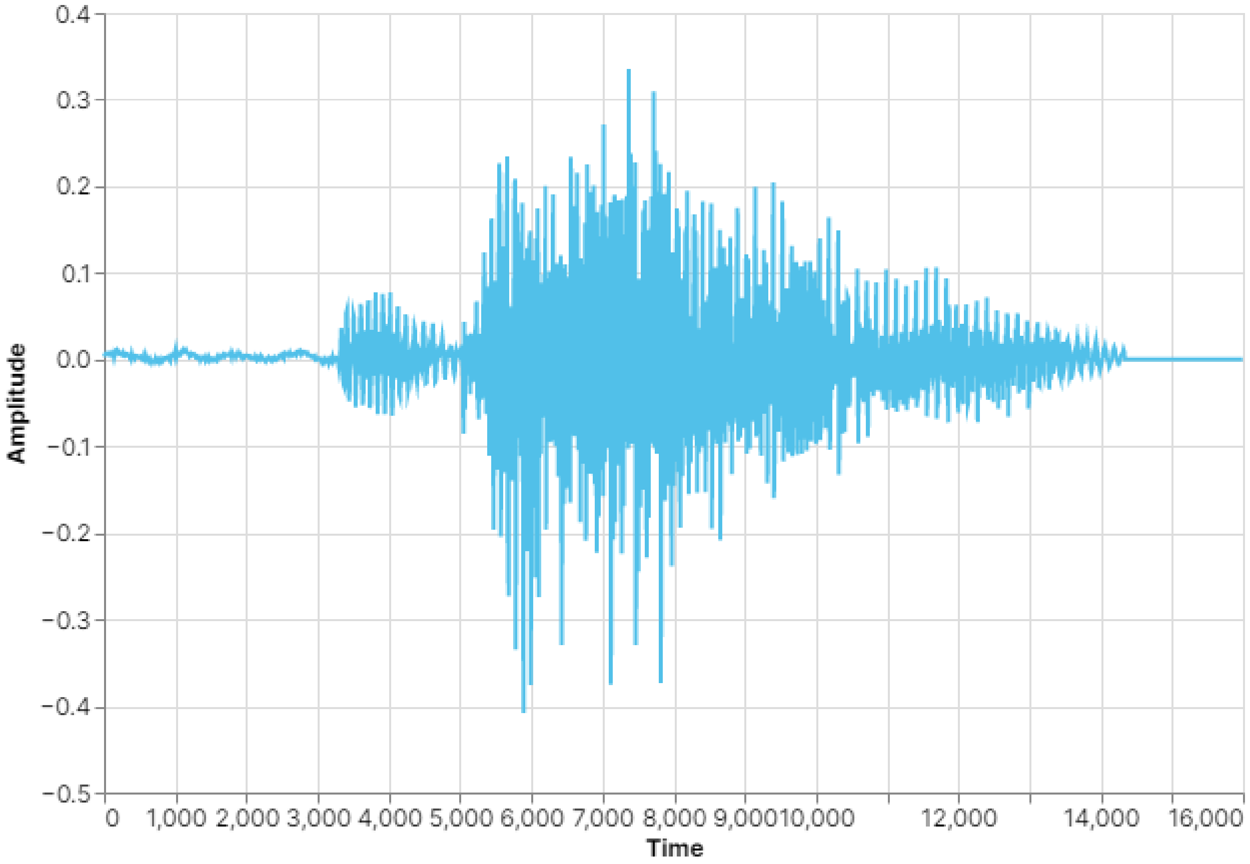
TimeSeries Retention

- You can prevent your timeseries growing indefinitely by setting a maximum age for samples compared to the last event time (in milliseconds).
- By default, retention is 0
  - i.e., the timeseries will be never trimmed

Audio Processing

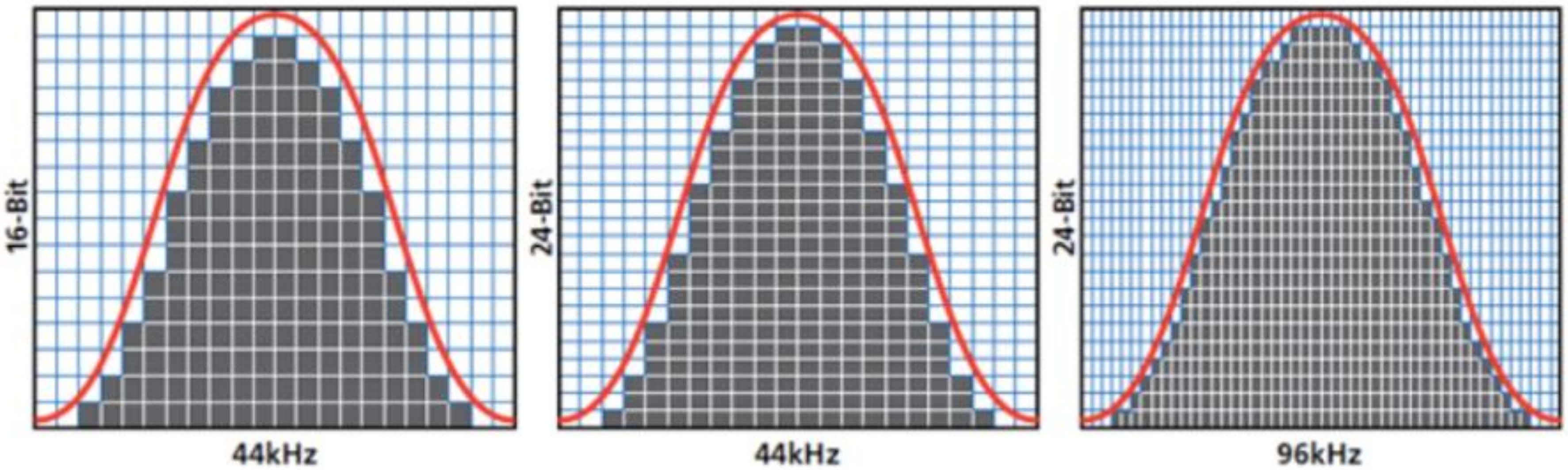
Waveform

- Speech signals are defined as pressure variations travelling through the air
- The waveform represents how the relative air pressure varies over time



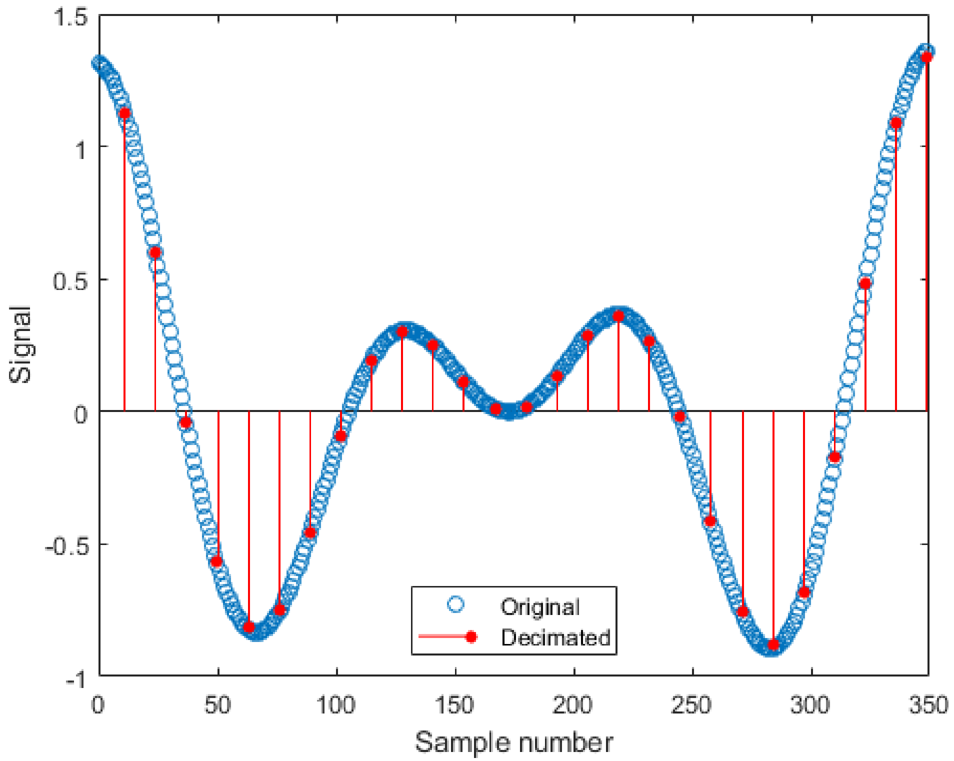
Waveform

- The "quality" of the waveform depends on:
  - Resolution
    - E.g., int16 (2 bytes), int24 (3 bytes), int32 (4 bytes)
  - Sampling Frequency
    - E.g., 48 kHz, 44.1 kHz, 16 kHz, …



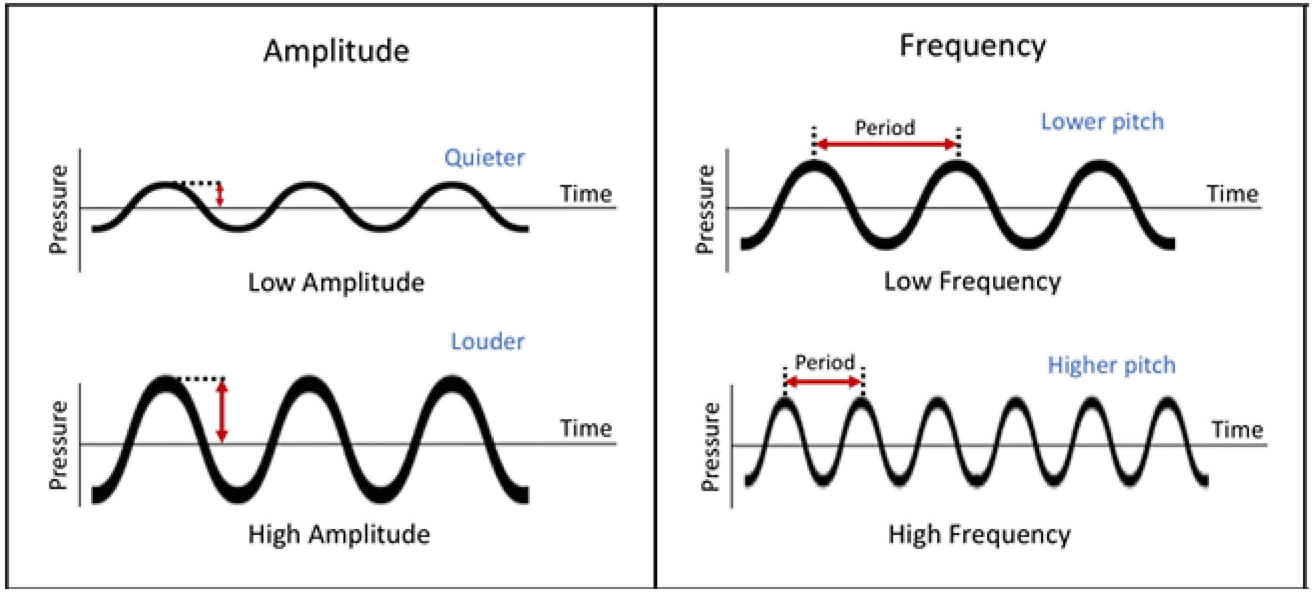
Resampling

- Downsample the signal from higher frequency to lower frequency



Waveform Properties

- Volume
  - Amplitude over time
  - Higher volume → louder
- Pitch
  - Related to frequency
  - Higher frequency → Higher sound



Audio Features

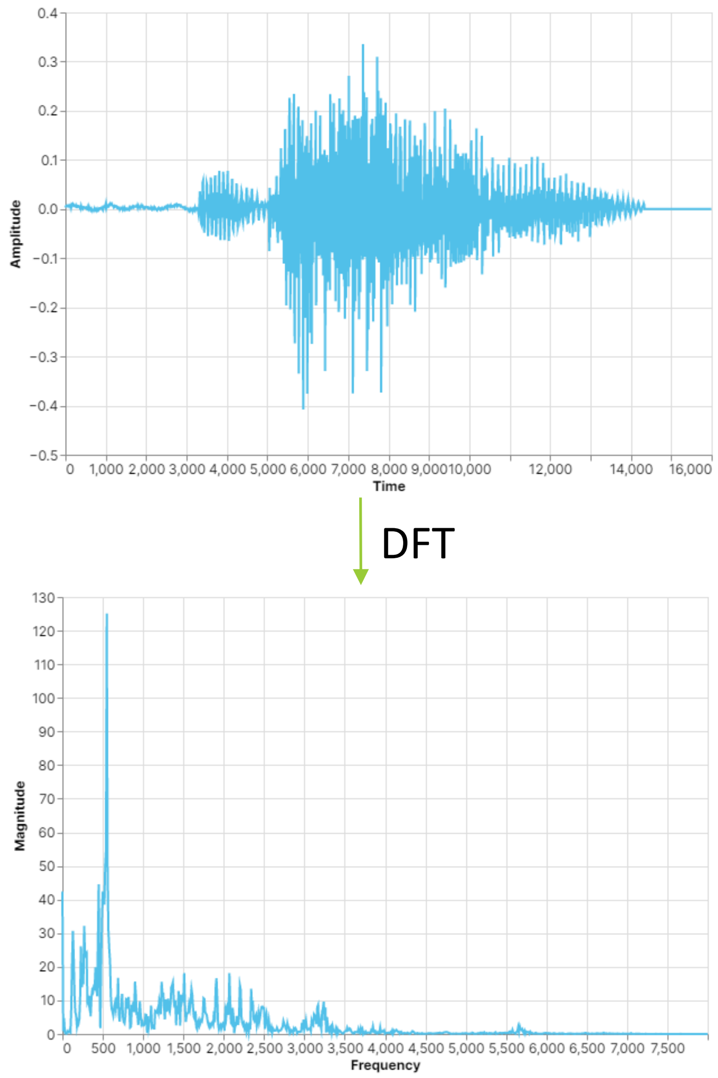
- Time domain
- Frequency domain
- Time-Frequency domain

Discrete Fourier Transform

- Time Domain x_n → Frequency Domain X_k
  - Compute this transformation for finite # frequencies
- DFT of a signal *x* of length *N*:

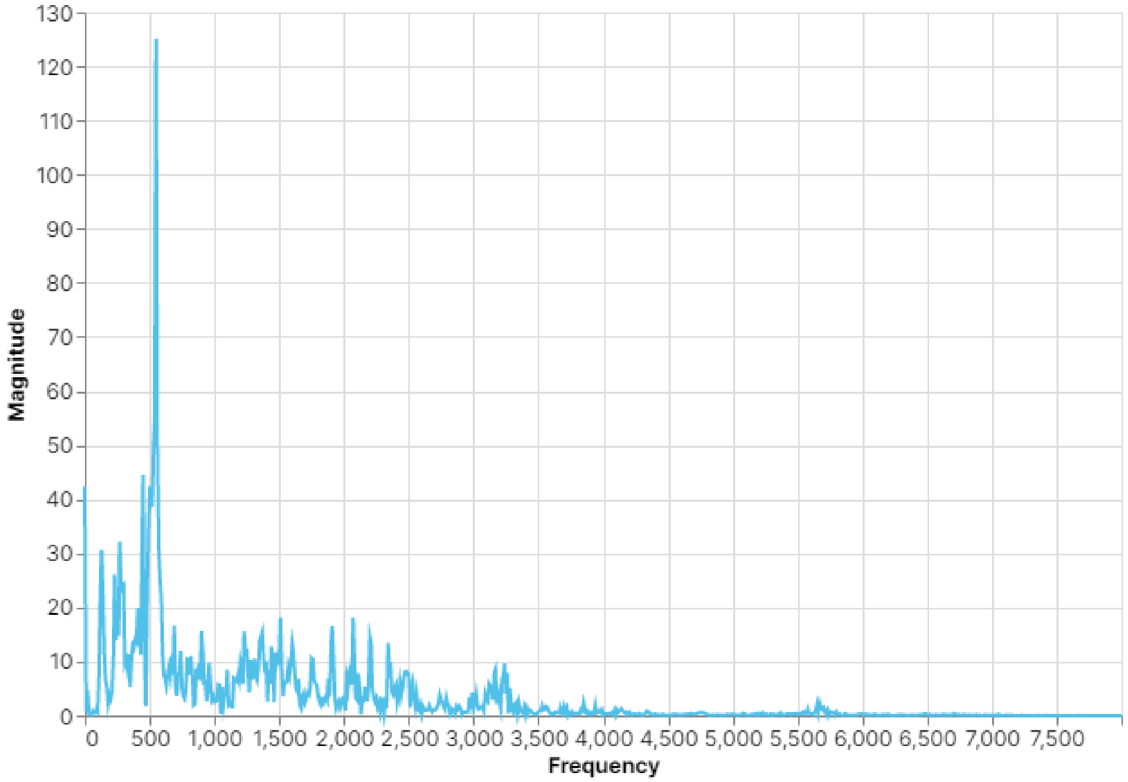
$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-i2\pi\frac{kn}{N}}$$

- The DFT of a real signal is symmetric
- Frequency range: [0, SR/2]
  - SR is the sampling rate
- Frequency resolution: SR/N



Redundancy in the DFT

- Output:
  - Array of shape: (N/2 + 1)
- Example:
  - 1s at 16 kHz → # samples = 16000
  - Output shape: (8001)



Computational efficiency: DFT vs FFT

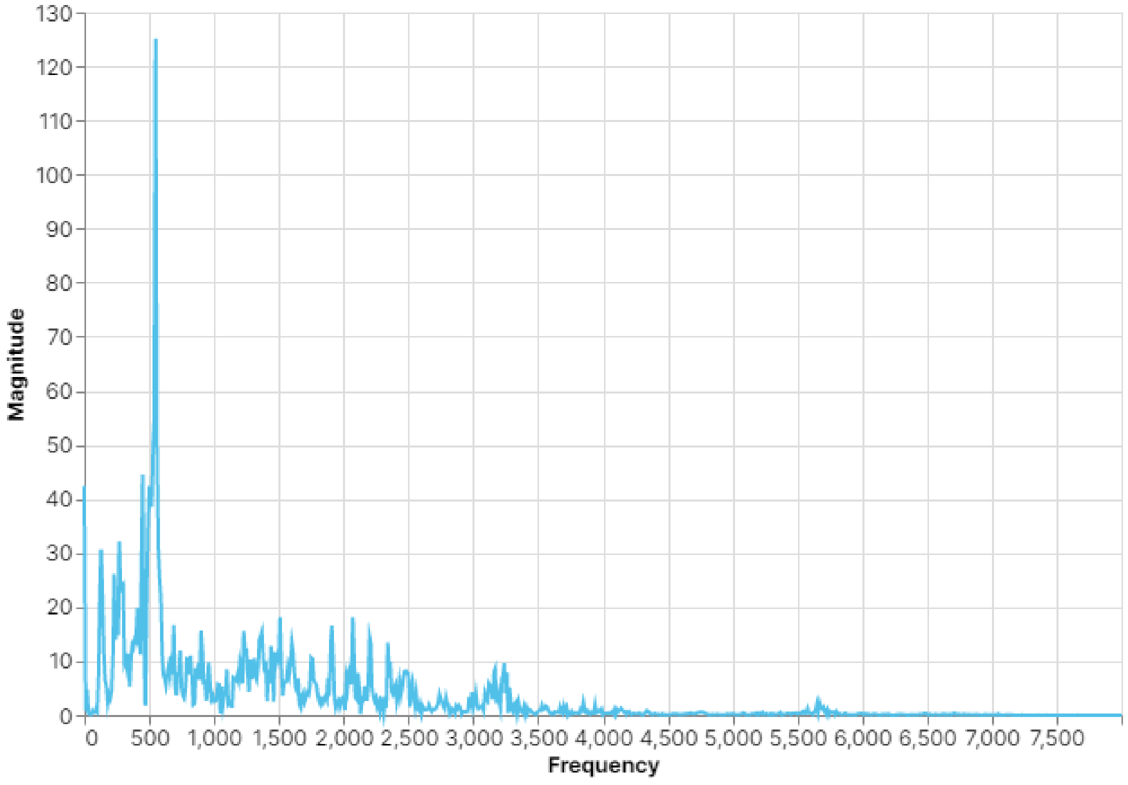
- DFT is computationally expensive (N^2, where N is the number of input samples)
  - *N can be any positive integer value*
- FFT is a more efficient implementation of DFT ($Nlog_2N$)
  - N must be a power of 2

N	# of DFT ops.	# of FFT ops.	FFT Efficiency
256	65 536	1 024	64 : 1
512	262 144	2 304	114 : 1
1024	1 048 576	5 120	205 : 1
2048	4 194 304	11 264	372 : 1
4096	16 777 216	24 576	683 : 1

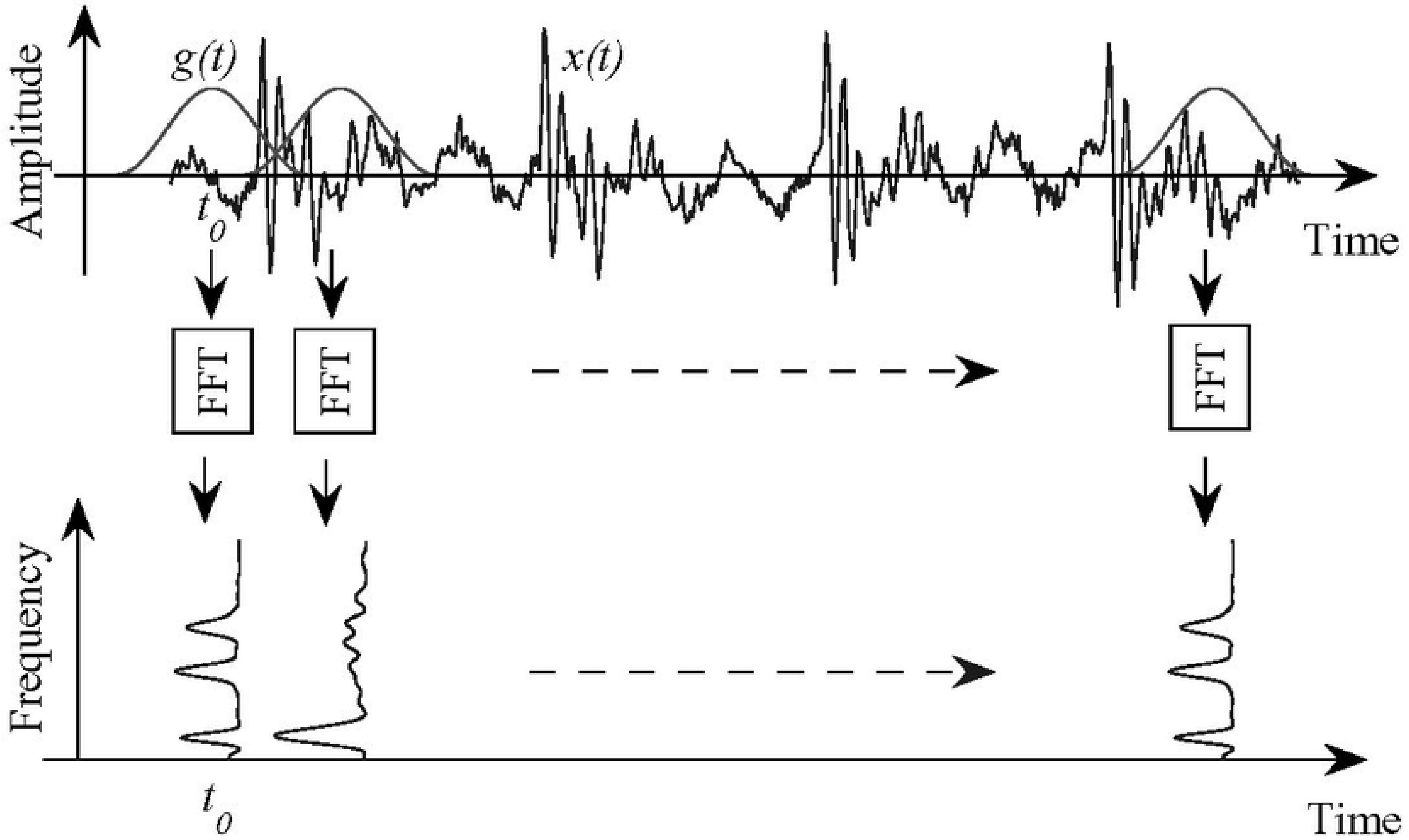
DFT Limitation in speech processing

- Speech signals are not stationary
  - The DFT is the average of all the phonemes in a word
  - NO time information
- Problem: we need to understand
  - WHICH phonemes are in the word → frequency
  - WHEN phonemes appear in the word → time

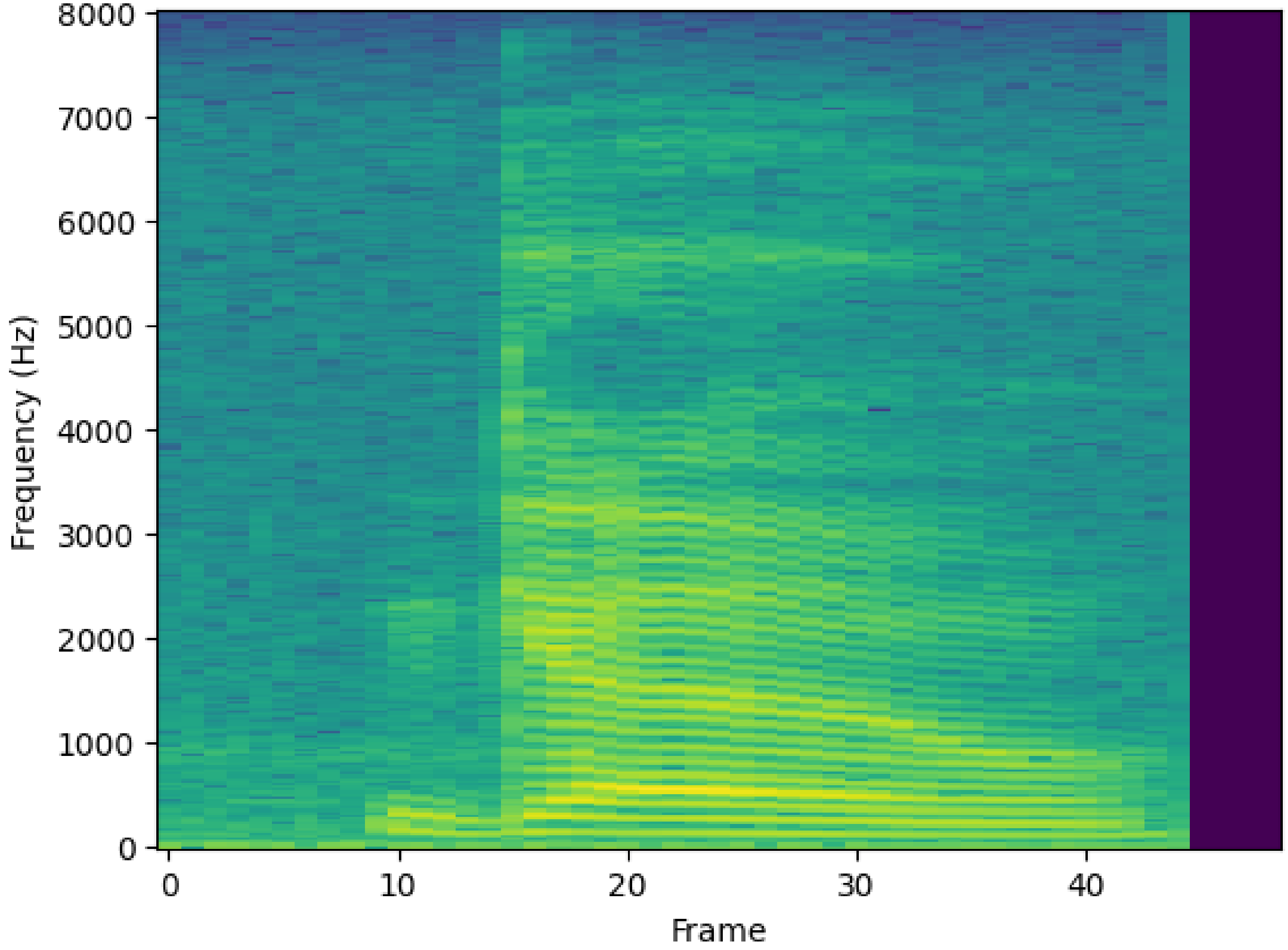
→Time-Frequency Domain



Short-Time Fourier Transform (STFT)

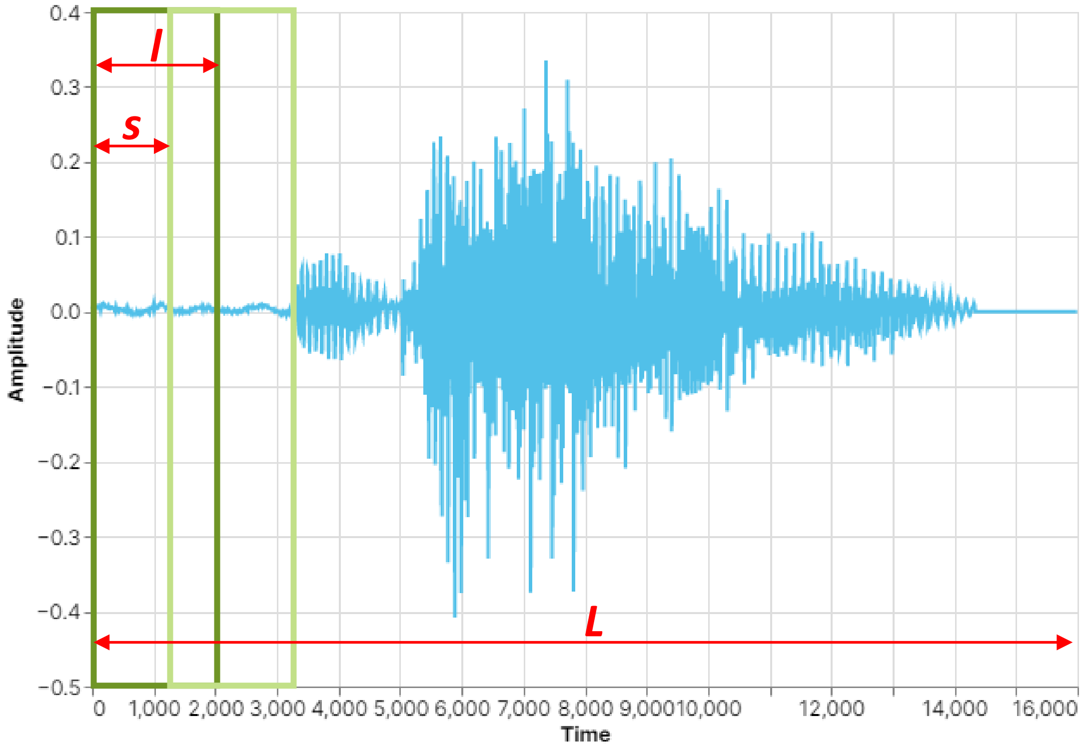


STFT Visualization



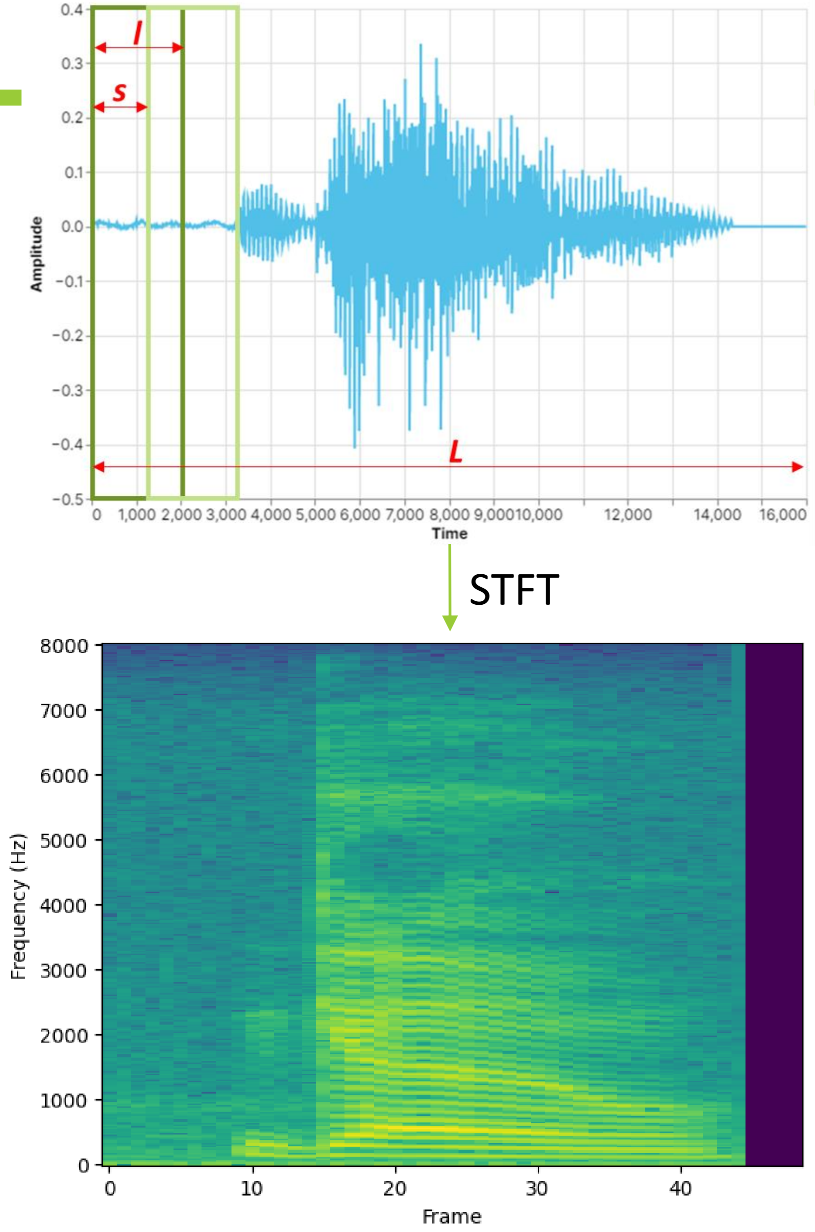
STFT parameters

- Frame Length *l*
- Frame step *s*
  - Also defined as percentage of overlap between two consecutive frames
- FFT Length
  - Commonly set equal to *l*



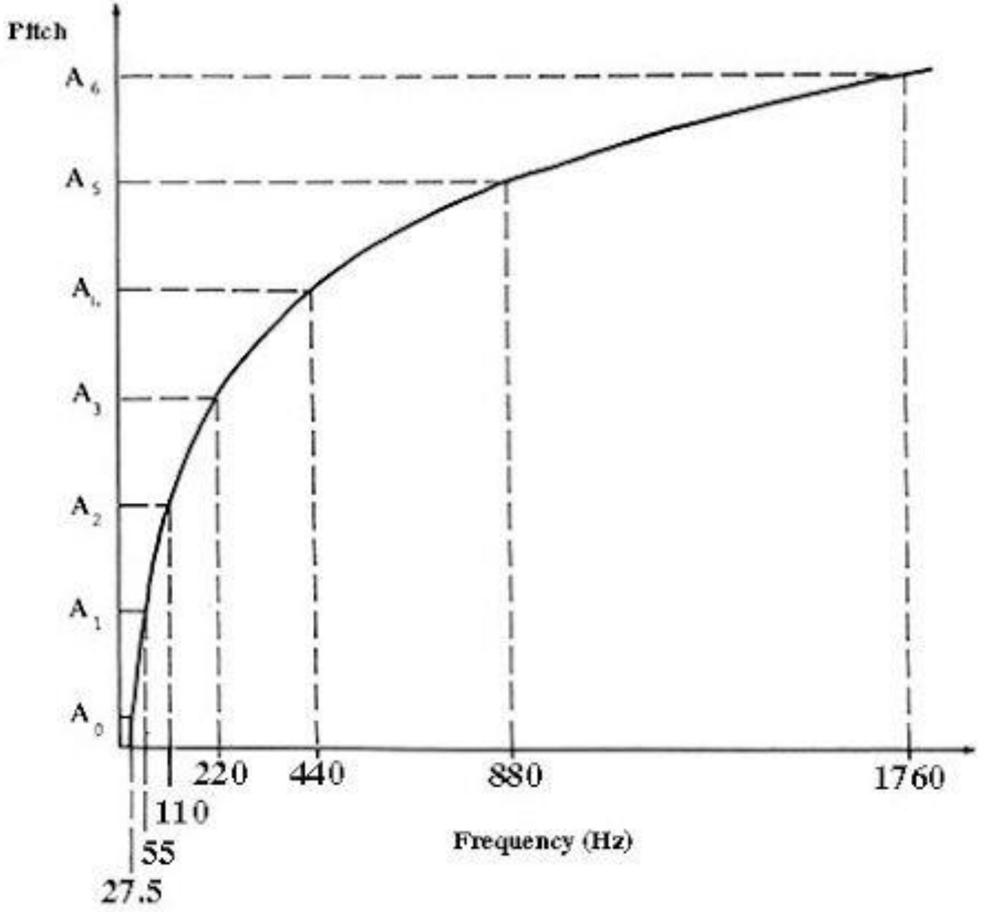
STFT Output: Spectrogram

- 2D Matrix of shape:
  - (# frequency bins, # frames) = (*l* / 2 + 1, (*L* – *l*) / *s* + 1)
- Example:
  - L = 1 s (@16 kHz), *l* = 40 ms, *s* = 20 ms
  - # Frequency bins:
    (40 ms * 16 kHz) / 2 + 1 = 321
  - # Frames:
    - (16000 – 40 ms * 16 kHz) / (20 ms * 16 kHz) + 1 = 49



STFT Limitation

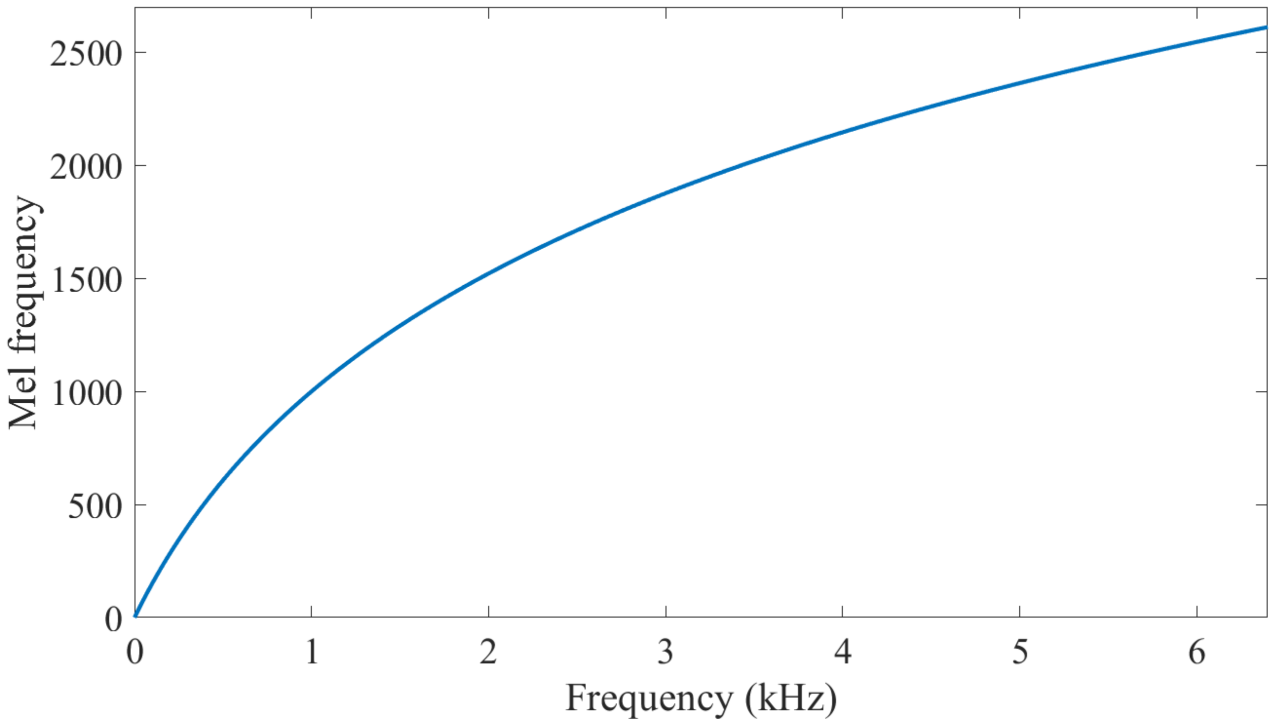
- Pitch:
  - 2 frequencies are perceived similarly if they differ by a power of 2



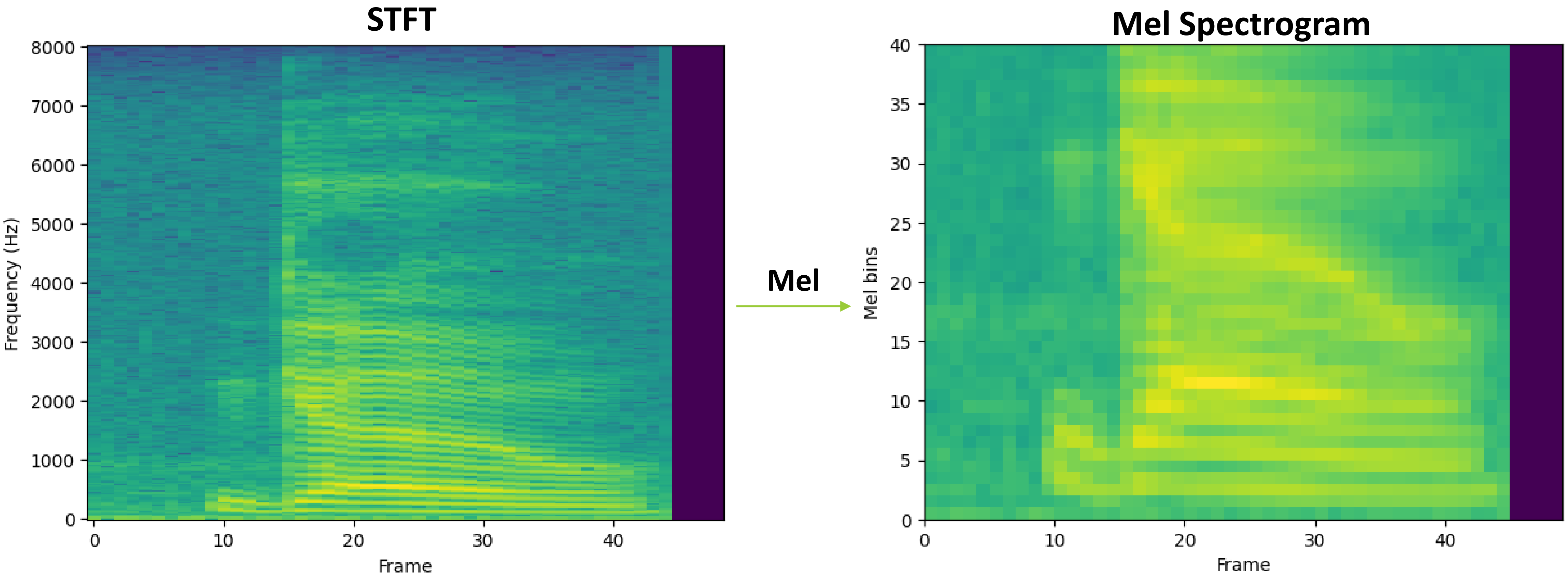
Mel scale

- Frequency-to-Mel transform:

$$m = 2595\ \log_{10}\left(1 + \frac{f}{700}\right)$$

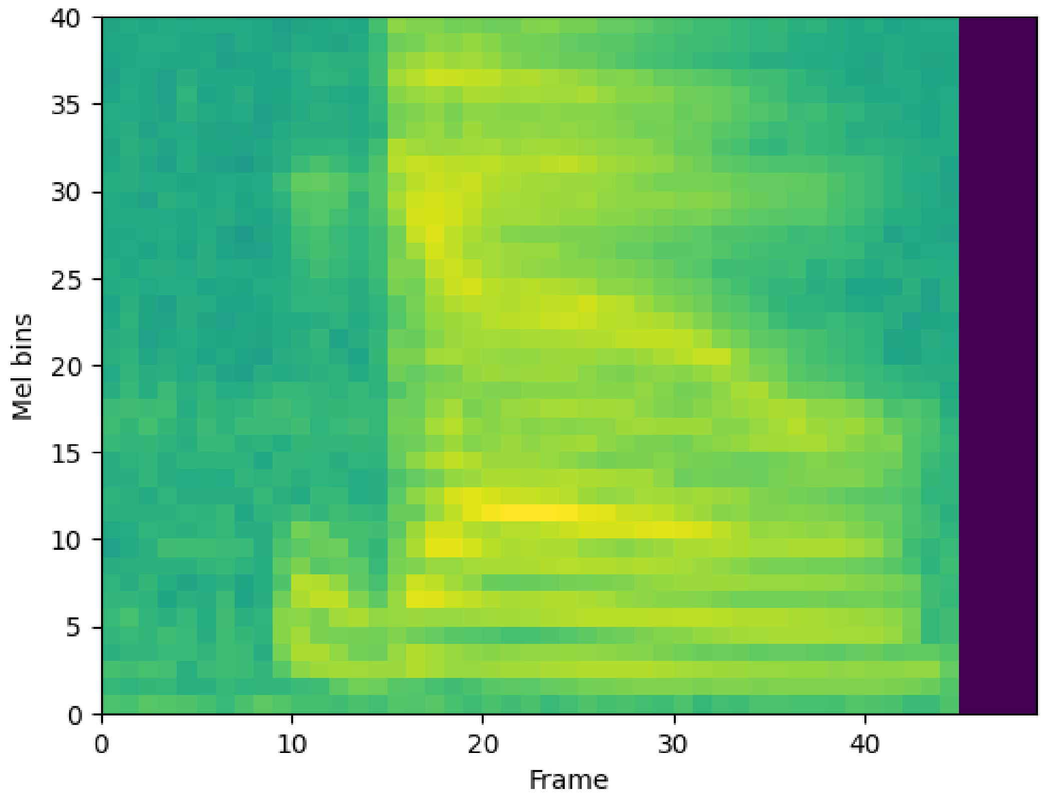


Mel Spectrogram



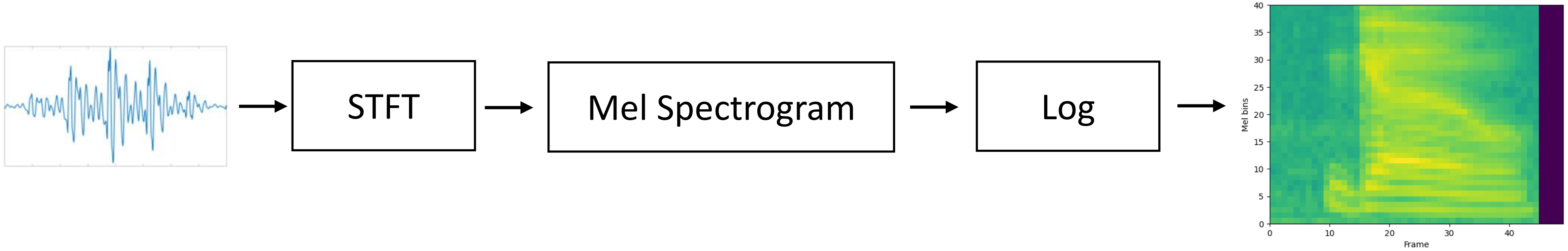
Mel Spectrogram

- Parameters:
  - Lower Frequency (in Hz)
  - Upper Frequency (in Hz)
  - # of Mel Frequency Bins
- Output:
  - 2D Matrix of shape:
    - (# Mel frequency bins, # frames)
- Example:
  - Lower Frequency: 20 Hz
  - Upper Frequency: 4000 Hz
  - # of Mel Frequency Bins: 40

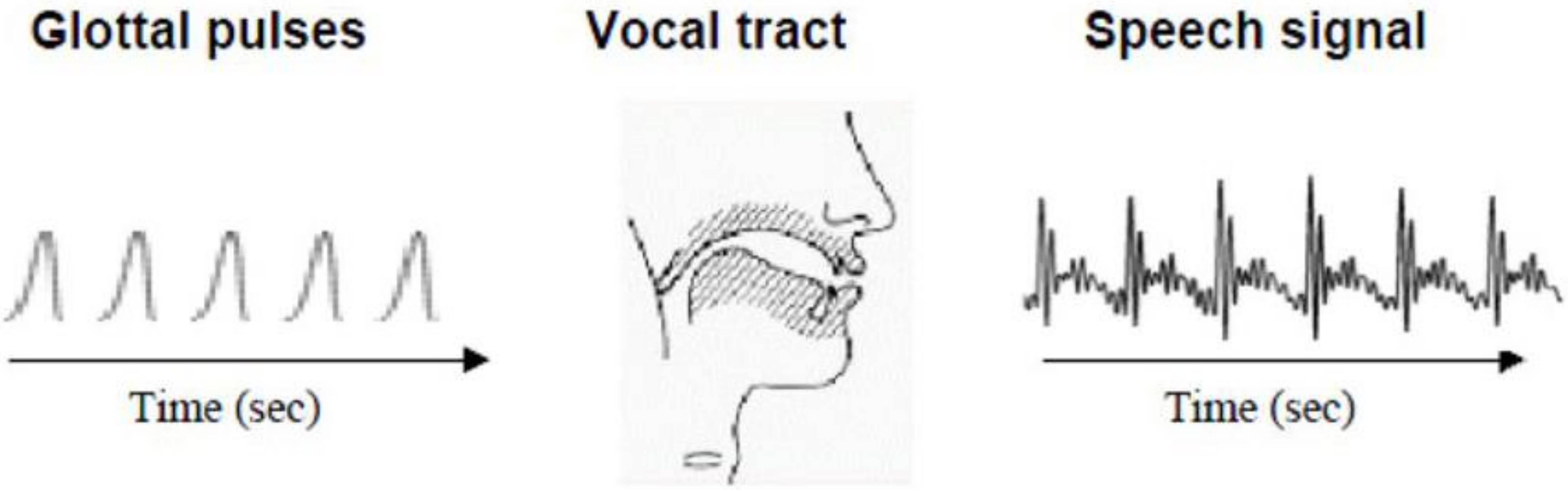


Log-Amplitude Mel Spectrogram

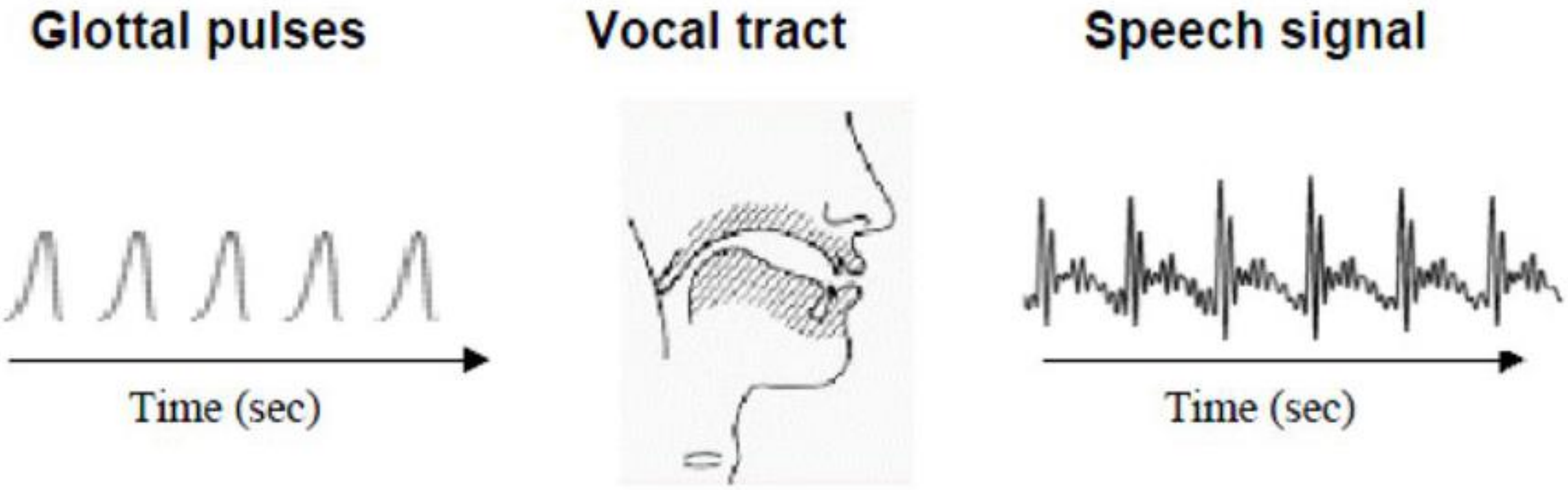
- Our perception of loudness is logarithmic
  - Apply logarithm on the amplitude of the spectrum



Speech generation

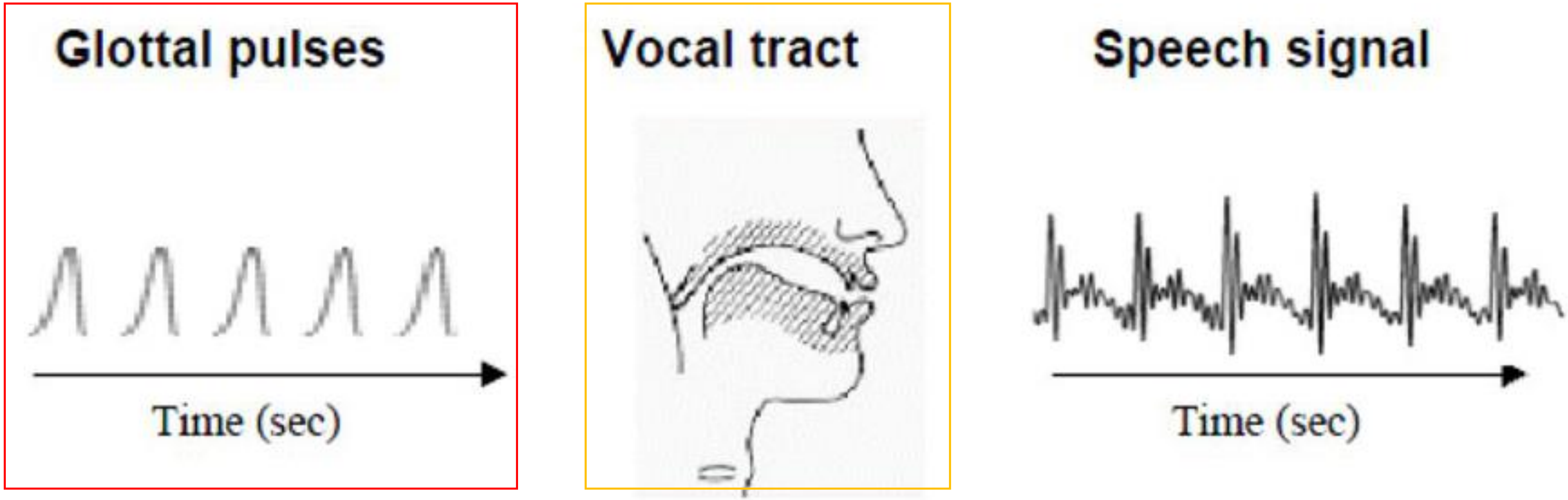


Speech generation



$$X(t) = E(t) + H(t)$$

Speech generation

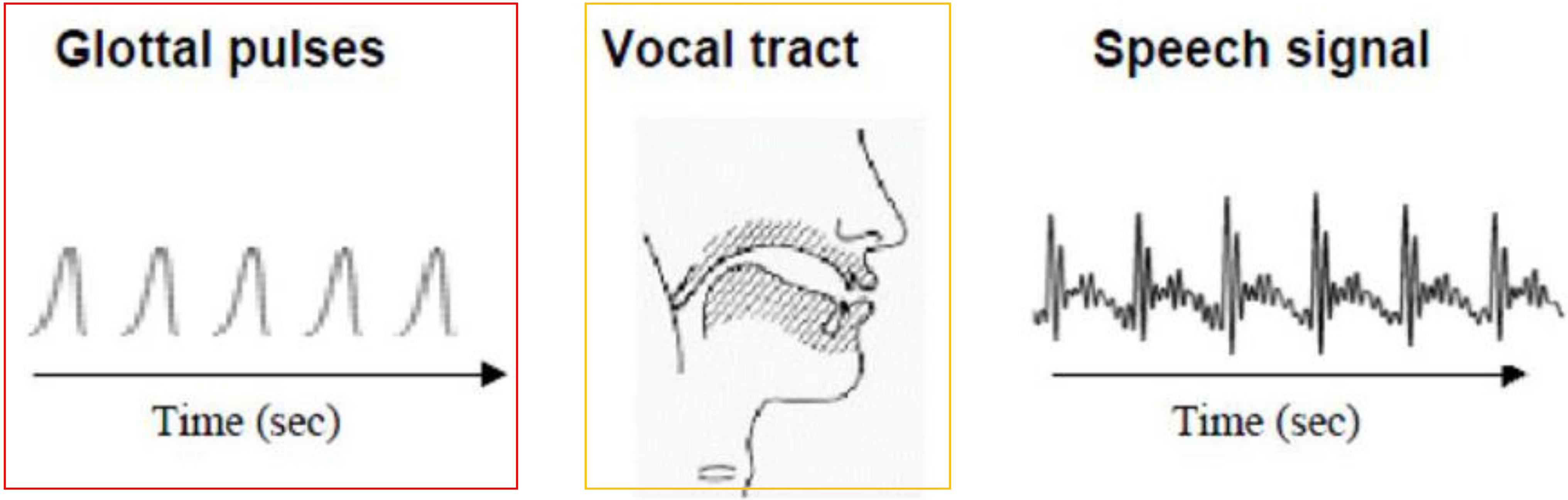


$$X(t) = \boxed{E(t)} + \boxed{H(t)}$$

Glottal Pulses

Spectral
Envelope

Speech generation



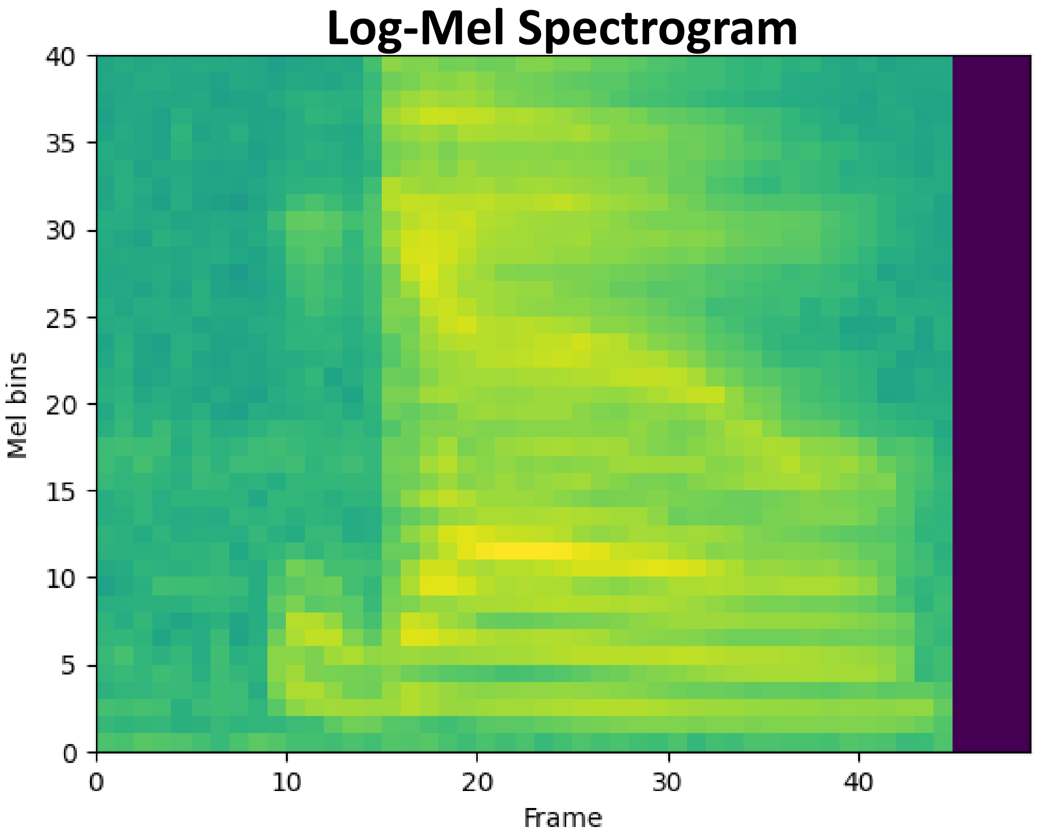
$$\boxed{X(t)} = \boxed{E(t)} + \boxed{H(t)}$$

Glottal Pulses

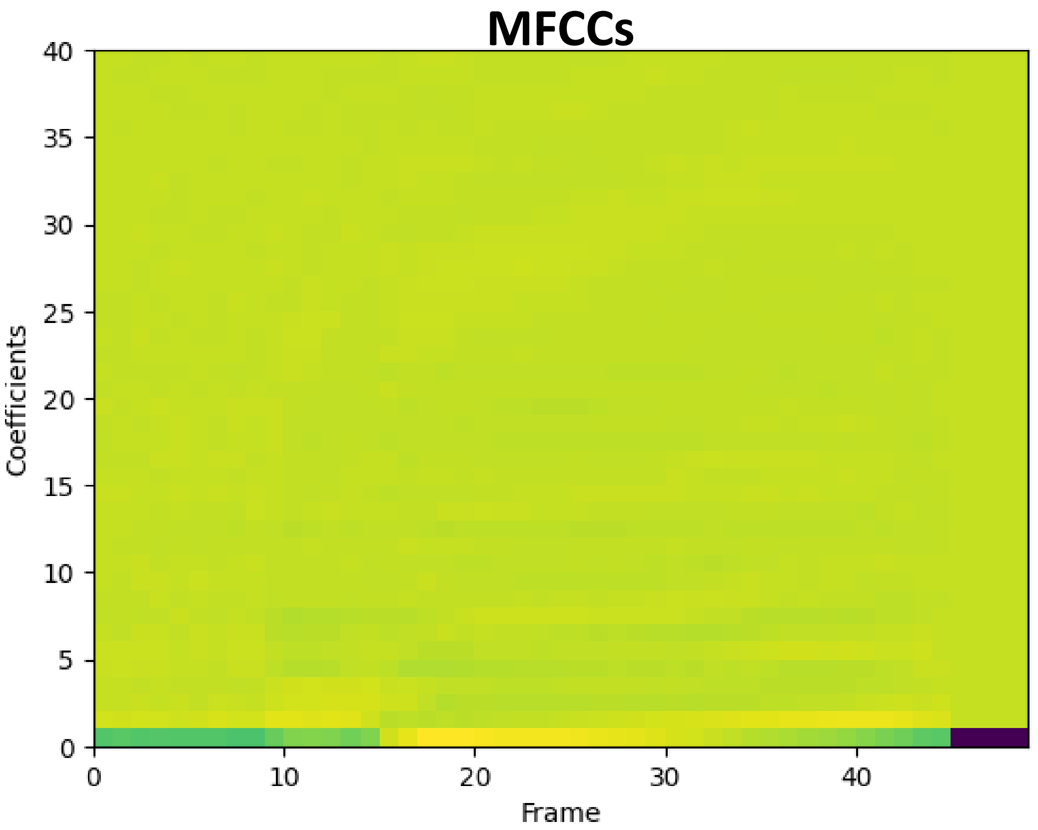
Cepstrum

Spectral Envelope

Mel Frequency Cepstral Coefficients (MFCCs)

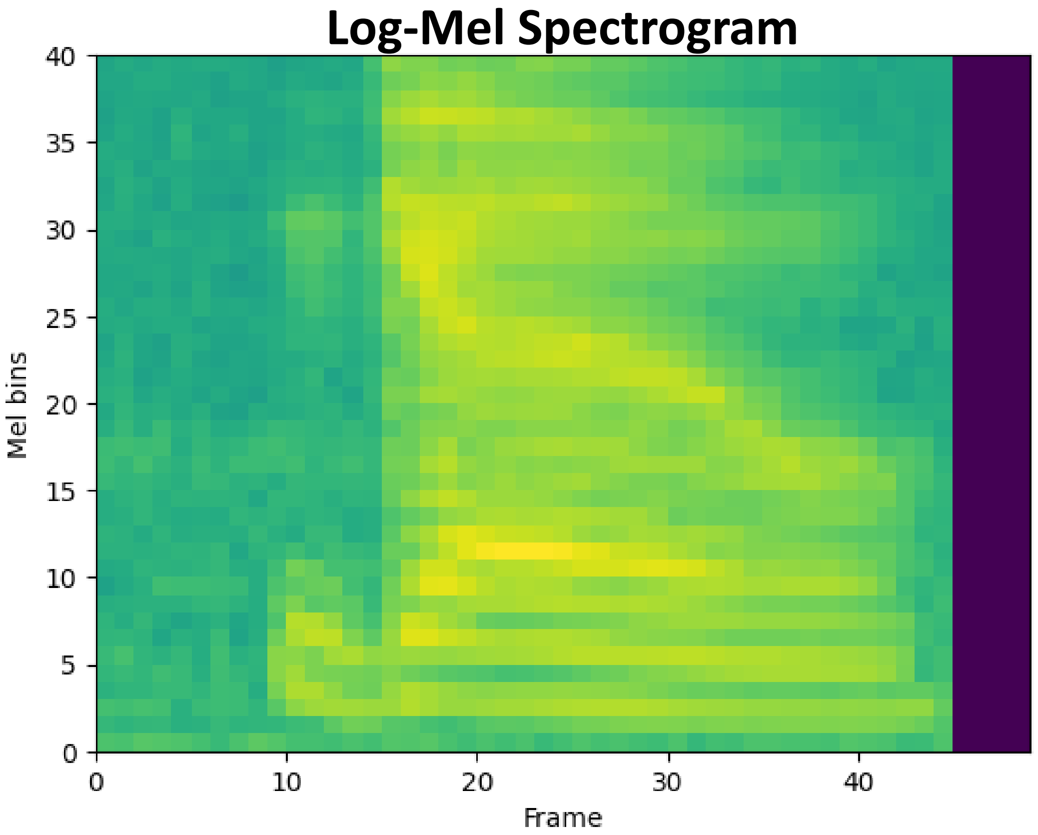


IDFT

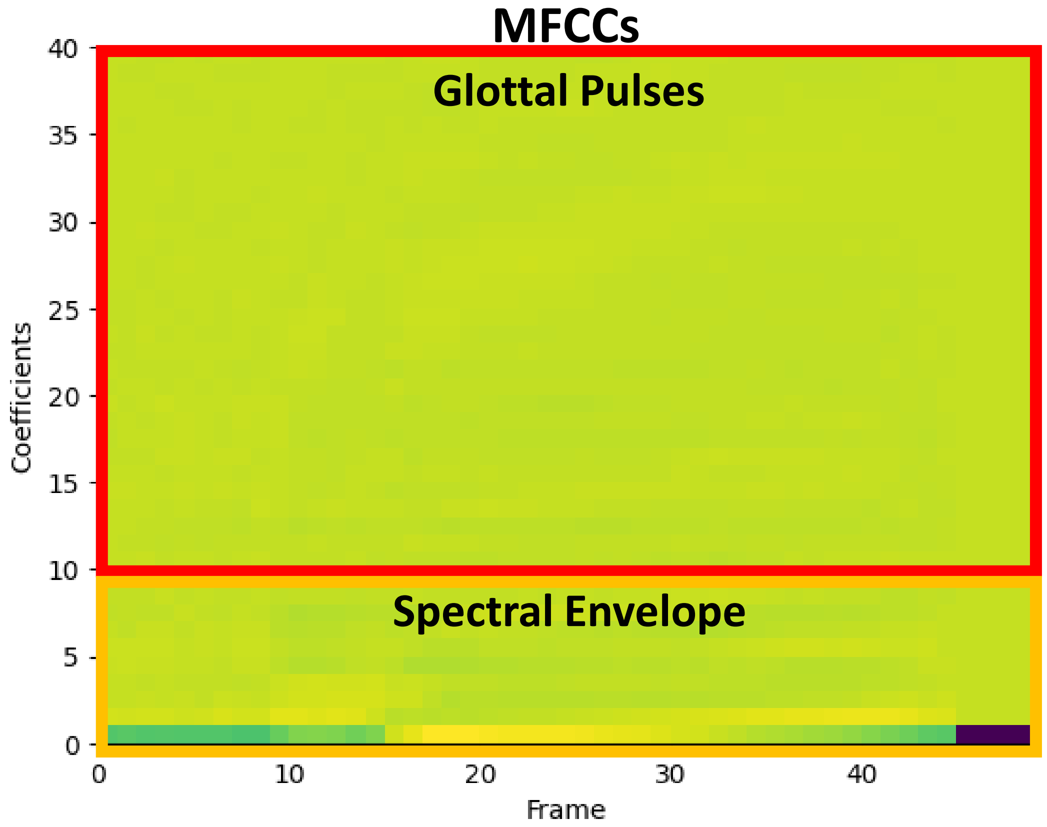


Spectrum → Cepstrum

Mel Frequency Cepstral Coefficients (MFCCs)



IDFT



Supplemental Material

- Redis TimeSeries
  - [Online] Redis TimeSeries Documentation
  - [Online] Redis TimeSeries Python SDK
  - [Online] Redis TimeSeries Python Examples
- Audio Processing
  - [Online] TensorFlow Signal Processing Operations
  - [Online] Introduction to Speech Processing (Tom Bäckström et al.)
  - [Book] Theory and Applications of Digital Speech Processing (Lawrence Rabiner, Ronald Schafer)
  - [Book] Multimedia signal processing theory and applications in speech, music and communications (Saeed Vaseghi)