Lecture 09
Speech Processing & Recognition

CSCI E-89 Deep Learning

Fall 2024

Zoran B. Djordjević

References

- Part of the lecture on Signal Processing follows to a good measure Chapter 9, *Automatic Speech Recognition,* of the book:

  Speech and Language Processing, 2nd Edition, by Daniel Jurafsky and James Martin, 2008, Prentice Hall, and

  Chapter 16 Automatic Speech Recognition and Text to Speech in

  Speech and Language Processing, 3rd Edition, by Daniel Jurafsky and James Martin available online at

  https://web.stanford.edu/~jurafsky/slp3/
- Several code examples follow Kaggle's post on the subject:

https://www.kaggle.com/davids1992/speech-representation-and-data-exploration

- A brief and clear text on MFCC-s can be found at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs
- Supporting Python code can be found here: https://github.com/jameslyons/python_speech_features

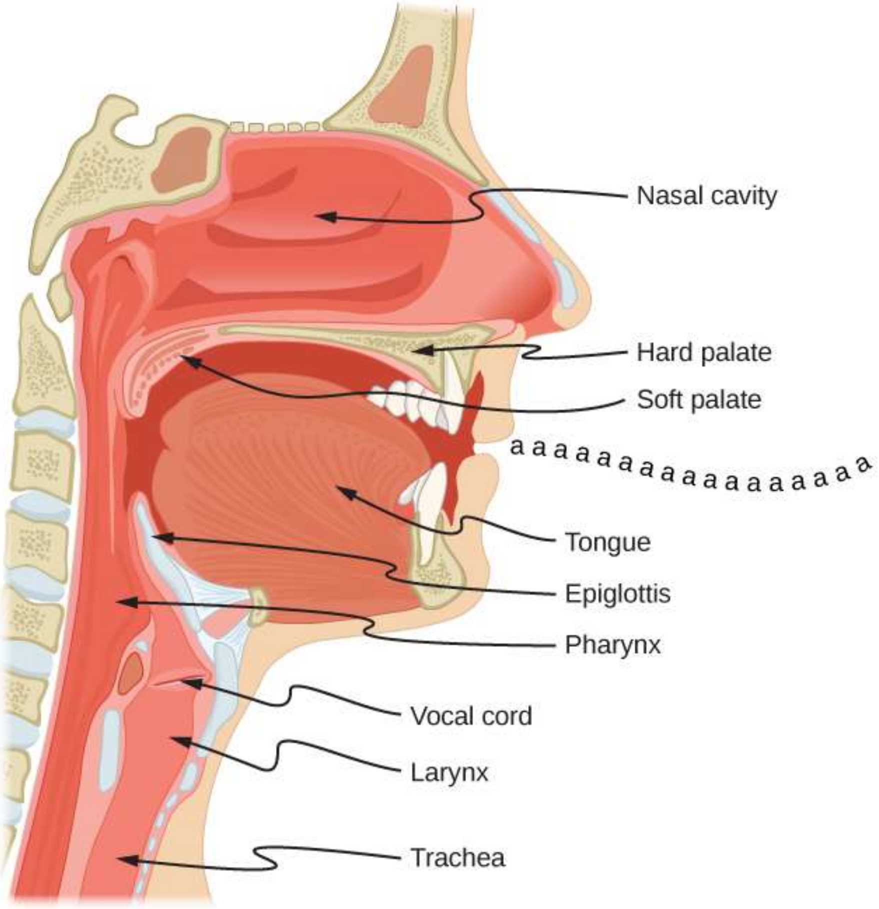
Speech Science & Signal Processing

- Speech has been a subject of great scientific interest and success for many decades.
- Major focus has been on “speech recognition” and transcription, meaning listening to a human speech and transcribing it into a written text.
- Successful scientific activity was in the area of speaker recognition. We hear a speech and identify the speaker.
- Big and successful effort was invested in transforming written text into audible speech, so-called text-to-speech (TTS).
- Recently, we are having great success in understanding speech in one language and transforming it into speech in another language.
- Most recently we are starting to have great success in producing speech that sounds as if spoken by a particular person.
- In the past, most of the Speech Science was done by medical physiologists, and electrical engineers. Some work was done by mathematicians and some by computer scientists. Most of the recent work, and in particular the work with Deep Neural Networks, is done by computer scientists.
- In group of slides we will introduce the basic concepts of a broad field referred to as the Signal Processing, which is very relevant for all speech related scientific and engineering activities.

How We Speak

Organs of Speech

- Sound is produced by the rapid movement of air. Humans produce most sounds in spoken languages by expelling air from the lungs through the windpipe (technically, the trachea) and then out the mouth or nose.
- As it passes through the trachea, the air passes through the larynx, commonly known as the Adam's apple or voice box. The larynx contains two small folds of muscle, the vocal folds (often referred to non-technically as the vocal cords), which can be moved together or apart.
- The space between these two folds is called the glottis. If the folds are close together (but not tightly closed), they will vibrate as air passes through them; if they are far apart, they won't vibrate.

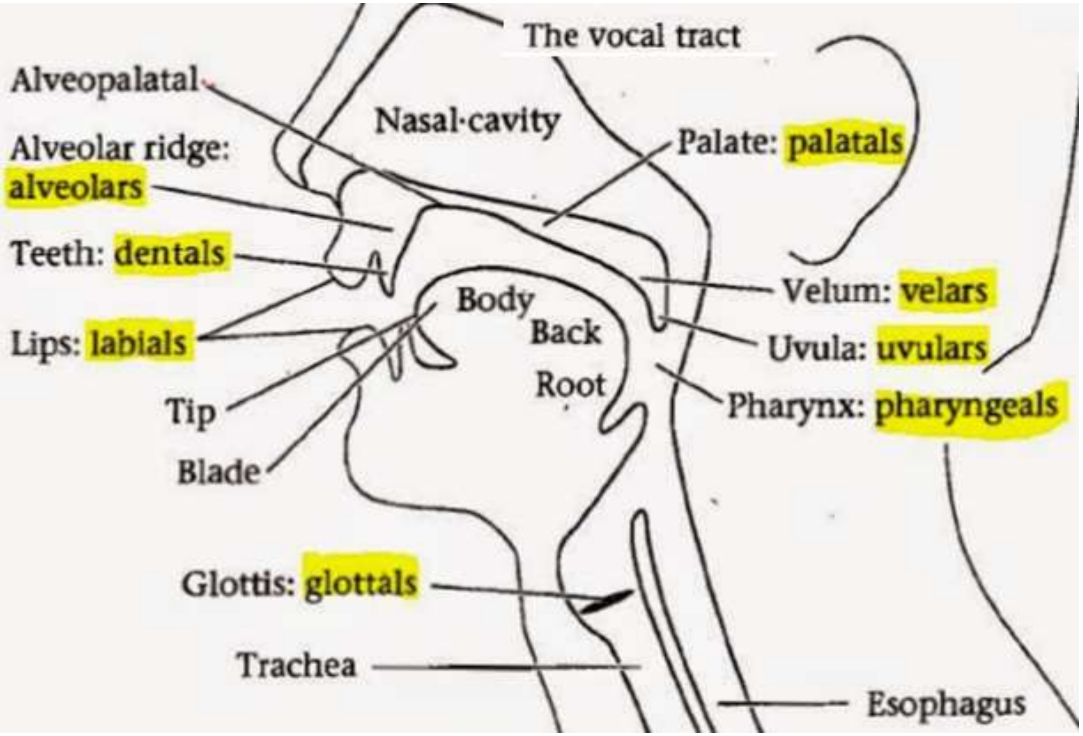


- Sounds made with the voiced sound vocal folds together and vibrating are called voiced; sounds made without this vocal unvoiced sound cord vibration are called unvoiced or voiceless. Voiced sounds include [b], [d], [g], [v], [z], and all the English vowels, among others. Unvoiced sounds include [p], [t], [k], [f], [s], and others.

Vocal Tract

- The area above the trachea is called the **vocal tract**; it consists of the **oral tract** and the **nasal tract**. After the air leaves the trachea, it can exit the body through the mouth or the nose. Most sounds are made by air passing through the mouth. Sounds made by air passing through the nose are called nasal sounds; nasal sounds (like English [m], [n], and [ng]) use both the oral and nasal tracts as resonating cavities.

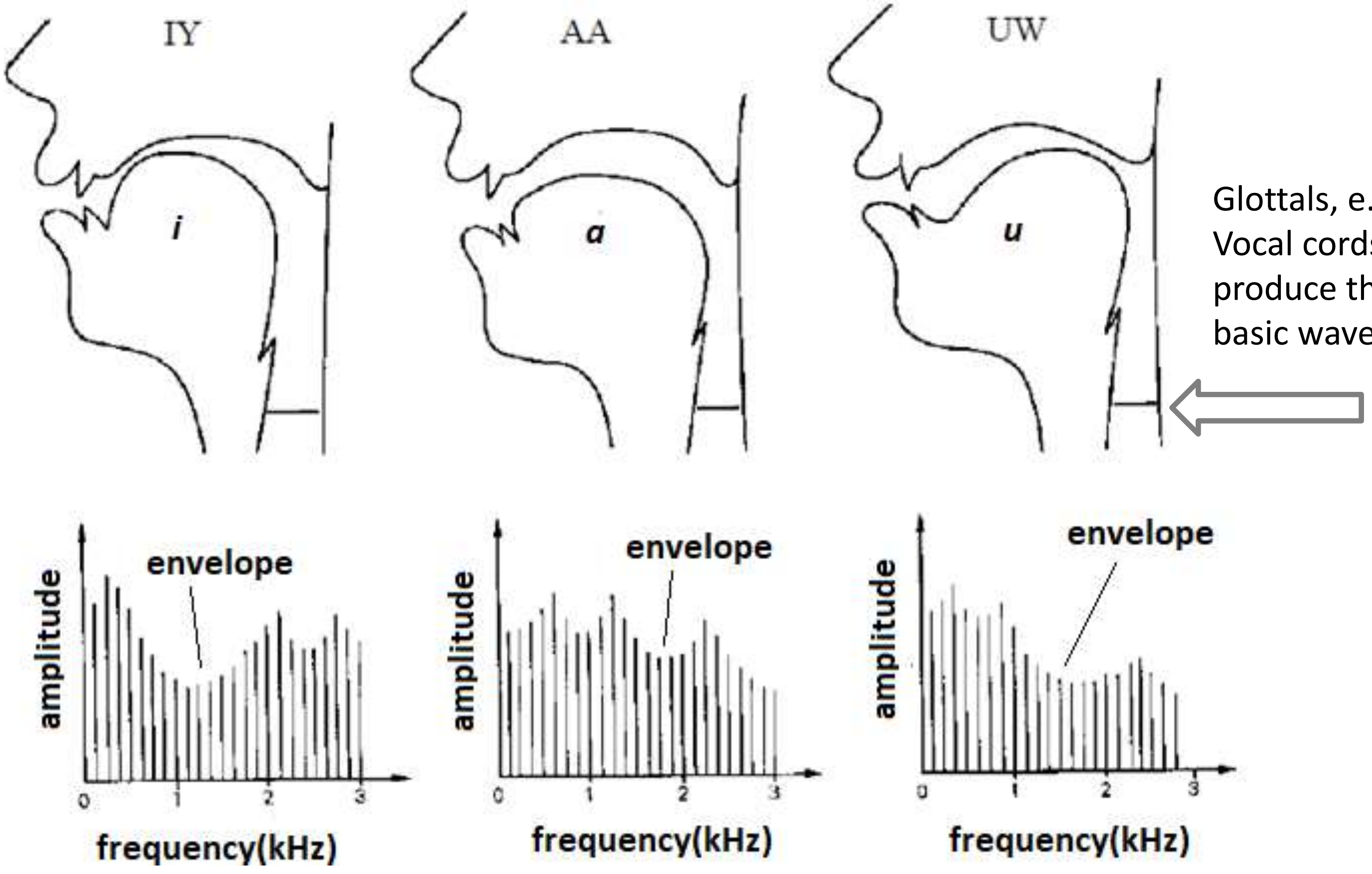
- Phones are divided into two main classes: **consonants** and **vowels**. Both kinds of sounds are formed by the motion of air through the mouth, throat or nose.
- **Consonants** are made by restriction or blocking of the airflow in some way, and can be voiced or unvoiced.
- **Vowels** have less obstruction, are usually voiced, and are generally louder and longer-lasting than consonants.
- The technical use of these terms is much like the common usage; [p], [b], [t], [d], [k], [g], [f], [v], [s], [z], [r], [l], etc., are consonants; [aa], [ae], [ao], [ih], [aw], [ow], [uw], etc., are vowels.



- **Semivowels** (such as [y] and [w]) have some of the properties of both; they are voiced like vowels, but they are short and less syllabic like consonants

Basic and Filtered Wave

- Speech is a time convolution of basic waves produced by glottals and modulation (filtering) by our mouth and nasal cavities.
- *In Frequency domain, convolution turns into a product of Fourier Images.*



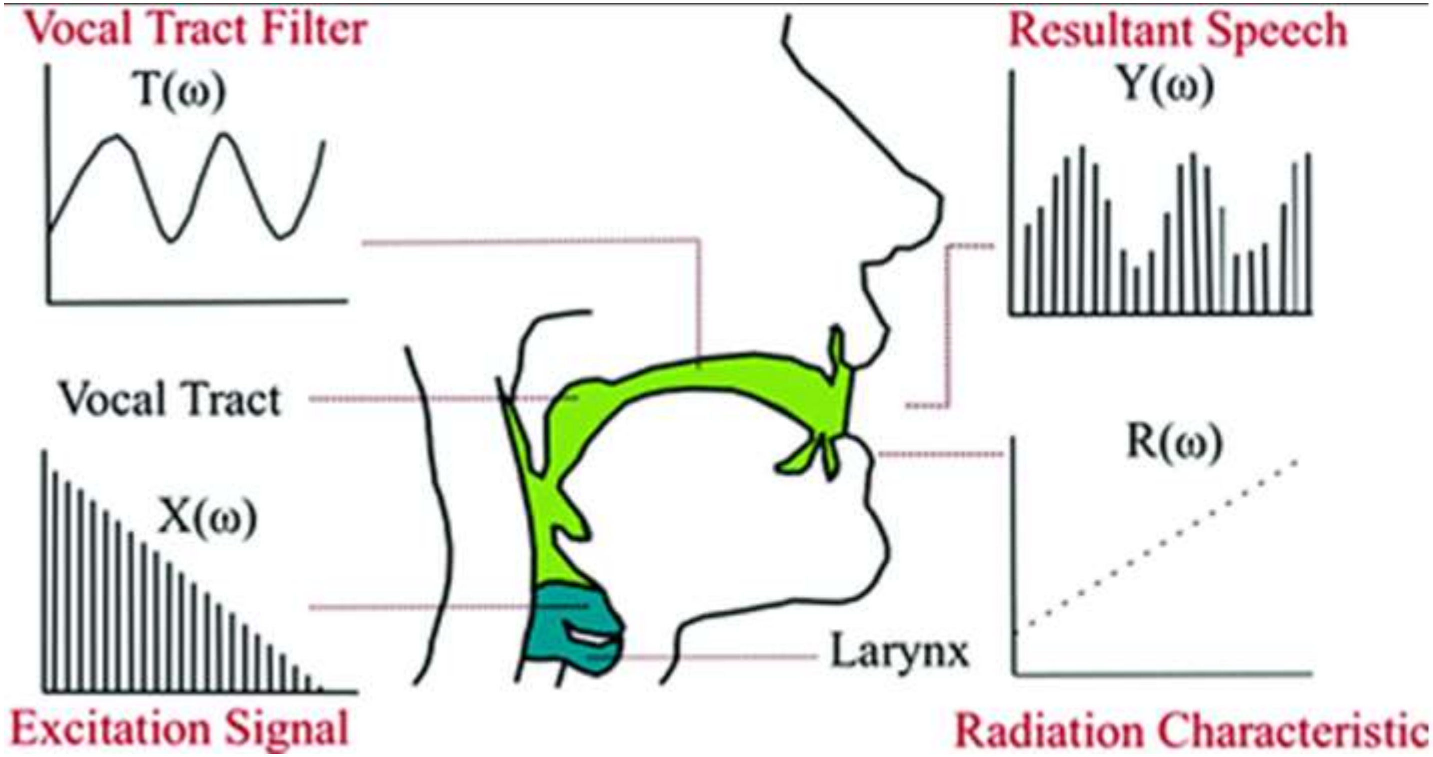
Glottals, e.i., Vocal cords produce the basic wave.

The Source-Filter Model

- Various sounds we produce are created by the resonant cavities of the mouth. The ***source-filter model*** is a way of explaining the acoustics of a sound by modeling how higher frequency pulses produced by the glottis (***the source***) are shaped by the vocal tract (***the filter***).
- Whenever we have a wave such as the vibration in air harmonic caused by the glottal pulse, the wave also has harmonics. A harmonic is another wave whose frequency is a multiple of the fundamental wave. In general, each of these waves will be weaker, that is, will have much less amplitude than the wave at the fundamental frequency.
- *The vocal tract acts as a kind of filter or amplifier*. Any cavity, such as a tube, causes waves of certain frequencies to be amplified and others to be damped.
- The amplification/modulation process is caused by the shape of the cavity. By changing the shape of the cavity, we can cause different frequencies to be amplified or attenuated.
- When we produce particular sounds, we are essentially changing the shape of the vocal tract cavity by placing the tongue and the other articulators in particular positions. The result is that different vowels cause different harmonics to be amplified or attenuated.
- Previous figure shows the vocal tract position for three vowels and a typical resulting spectrum. The formants are places in the spectrum where the vocal tract happens to amplify particular harmonic frequencies.

Voice Tract as a Filter

- Vocal cords impose on the air pushed out from the lungs a steady stream of unmodulated vibrations.
- Vocal Tract modulates those vibrations creating different envelopes that we recognize as different phoneme.



- In the literature the envelope is referred to as the “filter” or“filtering portion” and the excitation signal as the “basic wave” or the “source”

Speech as Interaction of Filter and Signal

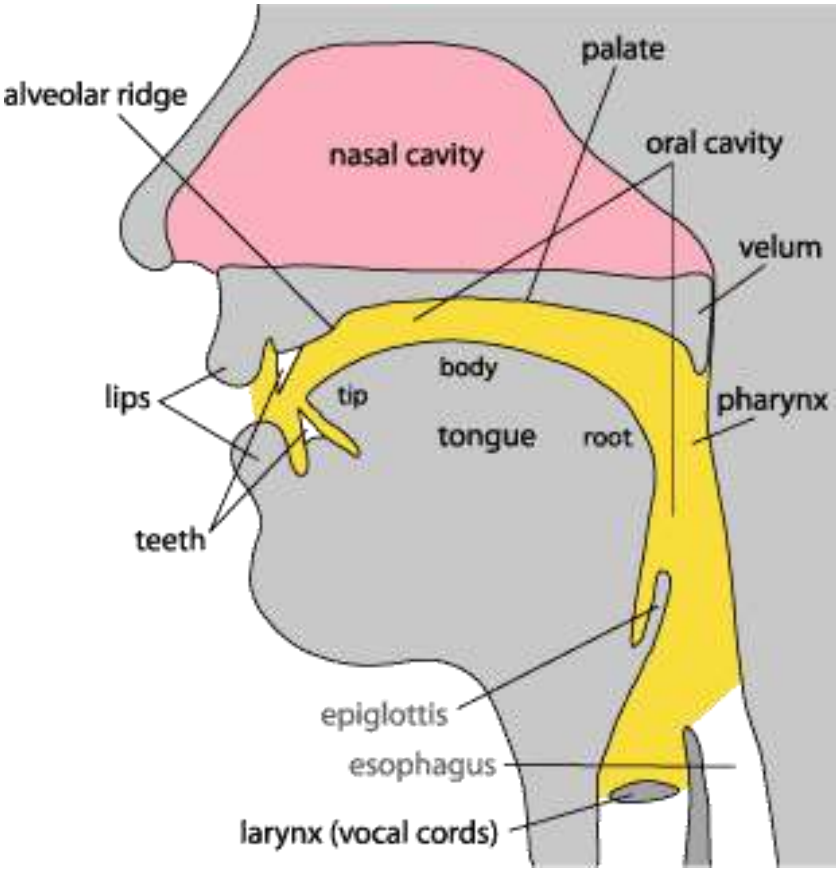
- Interaction of the filtering (modulating) influence of the vocal tract and the basic wave, signal, coming from the vocal cords results in speech signal.
- Time dependent speech $y(t)$ is a convolution of the modulating function $f(t)$ provided by the vocal cavity and the high frequency signal s(t) generated by the vocal cords.

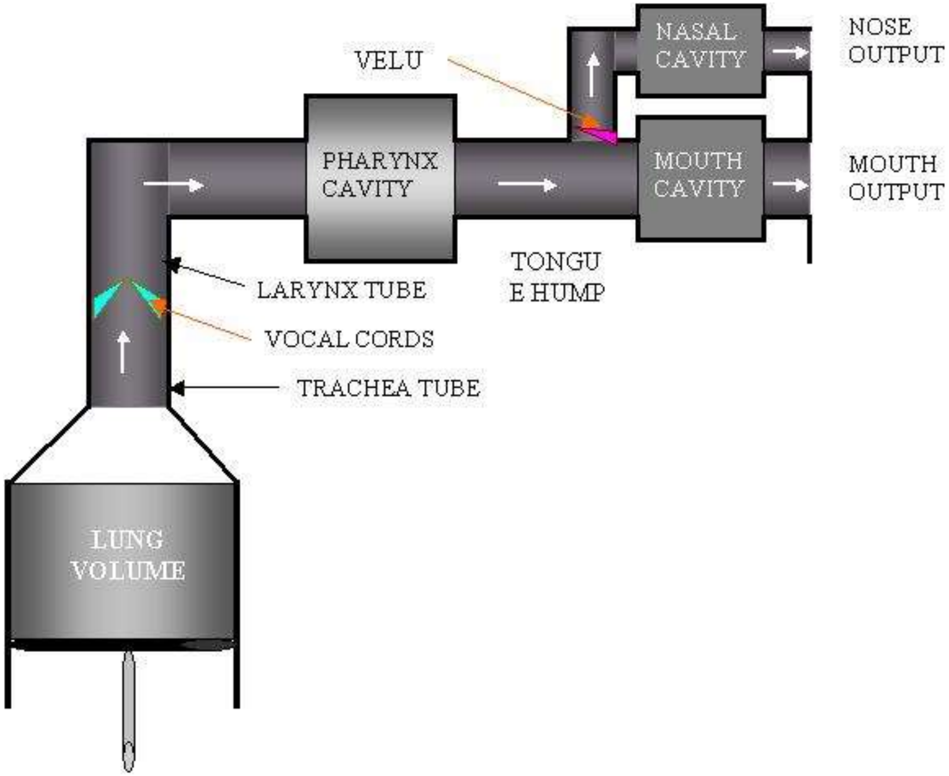
$$y(t) = \int f(t-\tau)s(\tau)dt$$

Sources of Different Sounds

- Vocal cord vibration
  - voiced speech (/aa/, /iy/, /m/, /oy/)
- Narrow constriction in mouth
  - fricatives (/s/, /f/)
- Airflow with no vocal-cord vibration, no constriction
  - aspiration (/h/)
- Release of built-up pressure
  - plosives (/p/, /t/, /k/)
- Combination of sources
  - voiced fricatives (/z/, /v/), affricates (/ch/)

Speech Production, Model of the Vocal Tract





- The shape of the vocal tract (or more accurately its area function) at any point in time is the most important determinant of the resonant frequencies of the cavity and the shape of the envelope modulating the vibrating air coming from the vocal cords.

Instruments that Make Speech

Attempts to construct mechanical devices producing sound and speech are quite old:

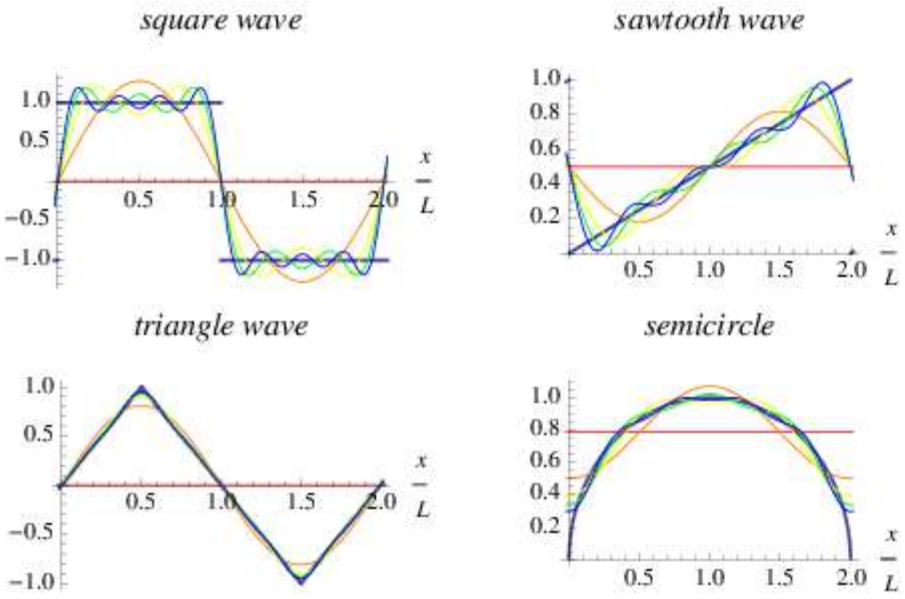
- Wolfgang von Kempelen, of the Mechanical Turk fame, between 1769 and 1790, built the first full-sentence speech synthesizer. His device consisted of a bellows to simulate the lungs, a rubber mouthpiece and a nose aperture, a reed to simulate the vocal folds, various whistles for the fricatives, and a small auxiliary bellows to provide the puff of air for plosives. By moving levers with both hands to open and close apertures, and adjusting the flexible leather "vocal tract", an operator could produce different consonants and vowels.

- Peruvian ceramic artists created vessels that could produce various animal sounds: Water whistling bottles - vasijas silbadoras de agua https://www.youtube.com/watch?v=BjVXWImbWnE

Signal Processing Concepts

Signal Processing and Fourier Series

- In many natural sciences we needed tools to represent any function into a combination of simpler functions. One such tool is the so-called Fourier Series.
- There is a mathematical theorem that states that any periodic function can be decomposed into a series of simple periodic functions (sines and cosines) of different frequencies.



- The ***Fourier series*** is named in honor of Jean-Baptiste Joseph Fourier (1768–1830).
- Fourier introduced the series for the purpose of solving the heat equation in metal plates, publishing his initial results in his 1807 *Mémoire sur la propagation de la chaleur dans les corps solides.*

- Each of the above functions could be expressed as a series (a long sum) of the form:

$$f(x') = \frac{1}{2} a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi x'}{L}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi x'}{L}\right).$$

Euler’s Formula

- Connection between simple periodic functions $\cos(x)$ and $\sin(x)$ and the exponential function is given by the Euler’s formula:

$$e^{ix} = \cos(x) + i\sin(x)$$

- We could rewrite that formula to express cosine and sine functions over the exponential function, as:

$$\cos(x) = \frac{1}{2}(e^{ix} + \ e^{-ix})$$

$$sin(x) \ = \frac{1}{2i}(e^{ix} - \ e^{-ix})$$

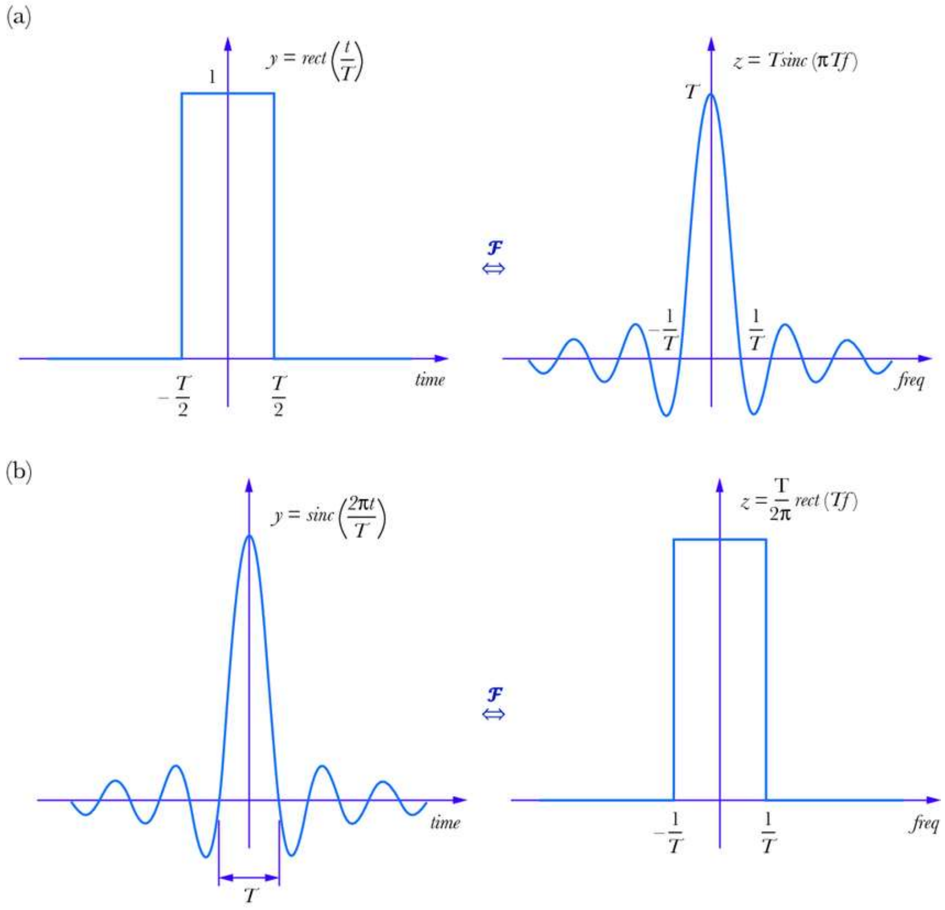
Fourier Transform

- In many applications we are dealing with functions which are non-periodic and yet, we would like to decompose them in sums of “simple” harmonic functions.
- In such cases, the Fourier sum is replaced by a Fourier Integral or Transform.
- A function $f(x)$ can be expressed as an integral over its Fourier Transform $F(k)$

- Two functions, $f(x)$ and $F(k)$ are related by the following equations:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk$$



- On the right, we see that the Fourier Transform of the simple rectangular function is $sinc(k) = \sin(k)/k$

Properties of Fourier Transform

- The **Similarity Theorem or time scaling property** of Fourier Analysis asserts that if a function becomes narrower in one domain by a factor a, it necessarily becomes broader by the same factor a in the other domain:

$$f(x) \rightarrow F(k) \quad \text{then } \; f(ax) \rightarrow \frac{1}{|a|} F(\frac{k}{a})$$

- **Time Shifting** Property states that if functions $f(x)$ and $F(k)$ are related by the Fourier transform:

$$f(x) \rightarrow F(k) \quad \text{then } \; f(x - x_o) \rightarrow \; e^{ikt_o} F(\mathrm{k})$$

- **The frequency shifting property** states that if we multiply the original function by a simple periodic function, its Fourier transform gets shifted in frequency

$$f(x) \rightarrow F(k) \quad \text{then } \; e^{ik_o x} f(x) \rightarrow \; F(k - k_0)$$

- The **Differentiation and Integration properties** state that

$$\frac{df(x)}{dx} \rightarrow ikF(k), \qquad \frac{d^n f(x)}{dx^n} \leftrightarrow \; (ik)^n F(k), \; \int f(x) dx \; \leftrightarrow \; \frac{1}{ik} F(k)$$

The Uncertainty Principle

- If we define the normalized variance, a spread, or the normalized second-moment of a function f(x) as:

$$(\Delta x)^2 = \frac{\int_{-\infty}^{+\infty} f(x) f^*(x)(x-\mu)^2 dx}{\int_{-\infty}^{+\infty} f(x) f^*(x) dx}$$

- And similarly, the normalized variance (spread) of its Fourier transform as:

$$(\Delta k)^2 = \frac{\int_{-\infty}^{+\infty} F(k) F^*(k)(k-k_0)^2 dk}{\int_{-\infty}^{+\infty} F(k) F^*(k) dk}$$

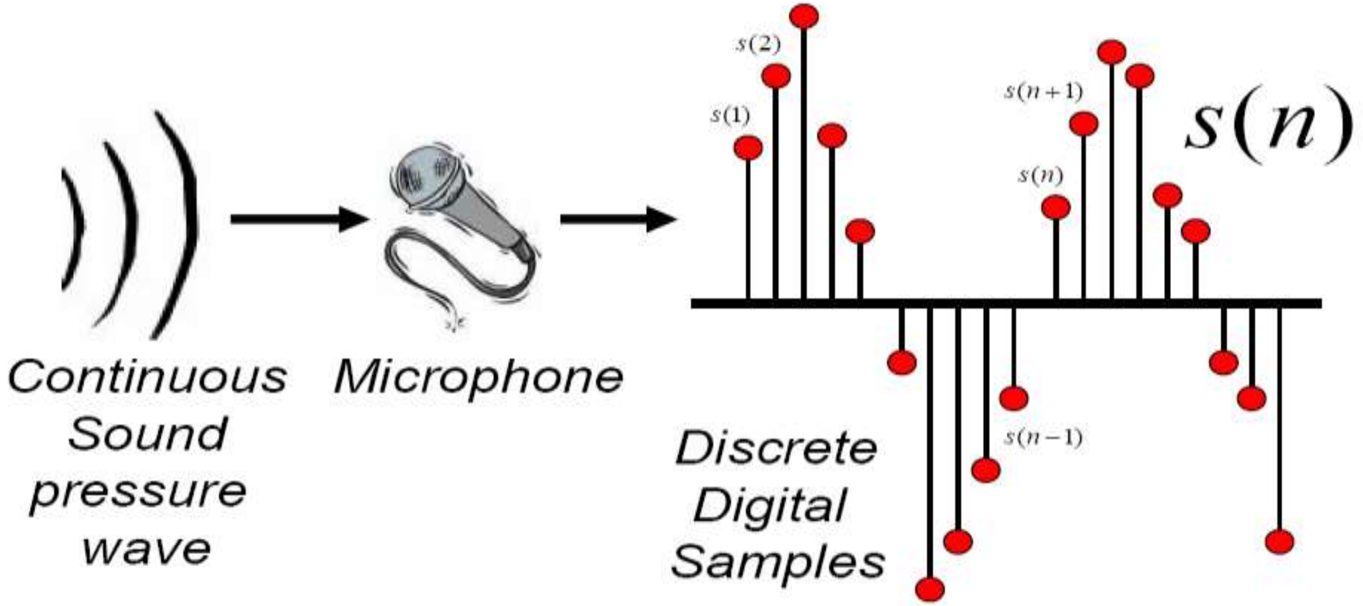
- It can be proven that there exists a fundamental lower bound on the product of these two spreads, regardless of the function f(x):

$$\boxed{(\Delta x)(\Delta k) \geq \frac{1}{4\pi}}$$

- This is the famous Gabor-Heisenberg Uncertainty Principle.

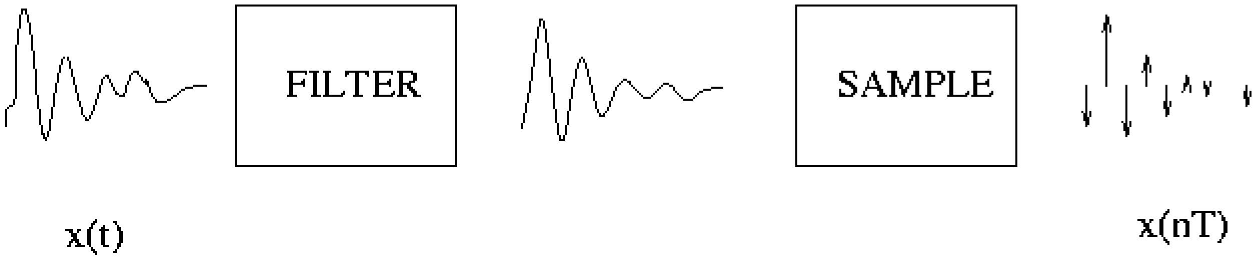
Discrete Representation of Signal

- Represent continuous signal into discrete form.



Sampling

- Sampling Theory



- Sampling Frequency
  - The maximum frequency that can be represented is half the sampling frequency
  - Telephone speech is sampled at 8 kHz. 16 kHz is generally regarded as sufficient for speech recognition and synthesis.

- Sampling Resolution
  - The "exactness" of each individual sample
  - 8-bit (-128 to 127) or 16-bit (-32768 to 32767)
  - 8 bits is considered enough for speech recognition

Sampling

- The first step in processing speech is to convert the analog representations (first air pressure, and then analog electric signals in a microphone), into a digital signal.
- This process of analog-to-digital conversion has two steps: sampling and quantization. A signal is sampled by measuring its amplitude at a particular time; the sampling rate is the number of samples taken per second. In order to accurately measure a wave, it is necessary to have at least two samples in each cycle: one measuring the positive part of the wave and one measuring the negative part.
- More than two samples per cycle increases the amplitude accuracy, but less than two samples will cause the frequency of the wave to be completely missed. Thus the maxi- mum frequency wave that can be measured is one whose frequency is half the sample rate (since every cycle needs two samples).
- This maximum frequency for a given sampling rate is called the **Nyquist frequency**. Most information in human speech is in frequencies below 10,000 Hz; thus a 20,000 Hz sampling rate would be necessary for complete accuracy.
- Telephone speech is filtered by the switching network, and only frequencies less than 4,000 Hz are transmitted by telephones. Thus an 8,000 Hz sampling rate is sufficient for telephone-bandwidth speech like the Switchboard corpus.
- A 16,000 Hz sampling rate (sometimes called wideband) is often used for microphone speech.

Digitizing the signal (A-D)

- Sampling:
  - measuring amplitude of signal at time t
  - 16,000 Hz (samples/sec) Microphone (“Wideband”):
  - 8,000 Hz (samples/sec) Telephone
  - Why?
    - Need at least 2 samples per cycle
    - max measurable frequency is half sampling rate
    - Human speech < 10,000 Hz, so need max 20K
    - Telephone conversations are filtered at 4K, so 8K is enough

Quantization

- 8,000 Hz sampling rate requires 8000 amplitude measurements for each second of speech. There are many amplitude measurements and it is important to store them efficiently.
- They are usually stored as integers, either 8-bit (values from -128–127) or 16 bit (values from -32768–32767).
- This process of representing real-valued numbers as integers is called ***quantization*** because there is a minimum granularity (the quantum size) and all values which are closer together than this quantum size are represented identically.
- We refer to each sample in the digitized quantized waveform as x[n], where n is an index over time.
- In speech recognition we rarely directly use the digitized, quantized representation of the waveform, we will rather extract so called **MFCC features.**

Nyquist-Shannon Sampling Theorem

- Sampling is a process of converting a signal (for example, a function of continuous time or space) into a sequence of values (a function of discrete time or space).

Shannon's version of the theorem states:

- If a function $x(t)$ contains no frequencies higher than B Hertz, it is completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart.

- A sufficient sample-rate is therefore anything larger than $2B$ samples per second. Equivalently, for a given sample rate f_s, perfect reconstruction is guaranteed possible for a bandlimit $B < f_s$
- When the bandlimit is too high (or there is no bandlimit), the reconstruction exhibits imperfections known as aliasing.
- Modern statements of the theorem are sometimes careful to explicitly state that $x(t)$ must contain no sinusoidal component at exactly frequency B, or that B must be strictly less than 1⁄2 the sample rate.
- The threshold 2B is called the Nyquist rate and is an attribute of the continuous-time input $x(t)$ to be sampled. The sample rate must exceed the Nyquist rate for the samples to suffice to represent $x(t)$. The threshold $f_s/2$ is called the Nyquist frequency and is an attribute of the sampling equipment.
- All meaningful frequency components of the properly sampled $x(t)$ exist below the Nyquist frequency.

Discrete Fourier Transform (DFT)

- In electronic, i.e. digital and computer systems we rarely perform “analog” Fourier Transform. We rather work with discrete signal $f(t) \rightarrow f(t_k)$ and discretized version of the Fourier Transform.
- Rather than using continuous integrals, we use sums over discrete samples. ***Discrete Fourier Transform*** (DFT) is defined as:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi ink/N}$$

- And its inverse $f_k = F_n^{-1}\left[\{F_n\}_{n=0}^{N-1}\right](k)$ as:

$$f_k = \frac{1}{N} \sum_{k=0}^{N-1} F_n e^{2\pi ink/N}$$

- Discrete Fourier transform (DFT) is extremely useful because it reveals periodicities in input data as well as the relative strengths of any periodic components.
- There are however a few subtleties in the interpretation of discrete Fourier transforms. In general, the discrete Fourier transform of a real sequence of numbers will be a sequence of complex numbers of the same length.

Discrete Fourier Transform

- Input:
  - Given a discrete signal $x_n$at N points: $0\ to\ N-1: \{x[0] \ldots x[N-1]\}$
  - For each of N discrete frequencies $\frac{2\pi}{N}k$ where $k\ \in \{0, \ldots, N-1\}$
  - A complex number X_k representing magnitude and phase of that frequency component in the original signal is called the Discrete Fourier Transform (DFT) and calculated as:

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

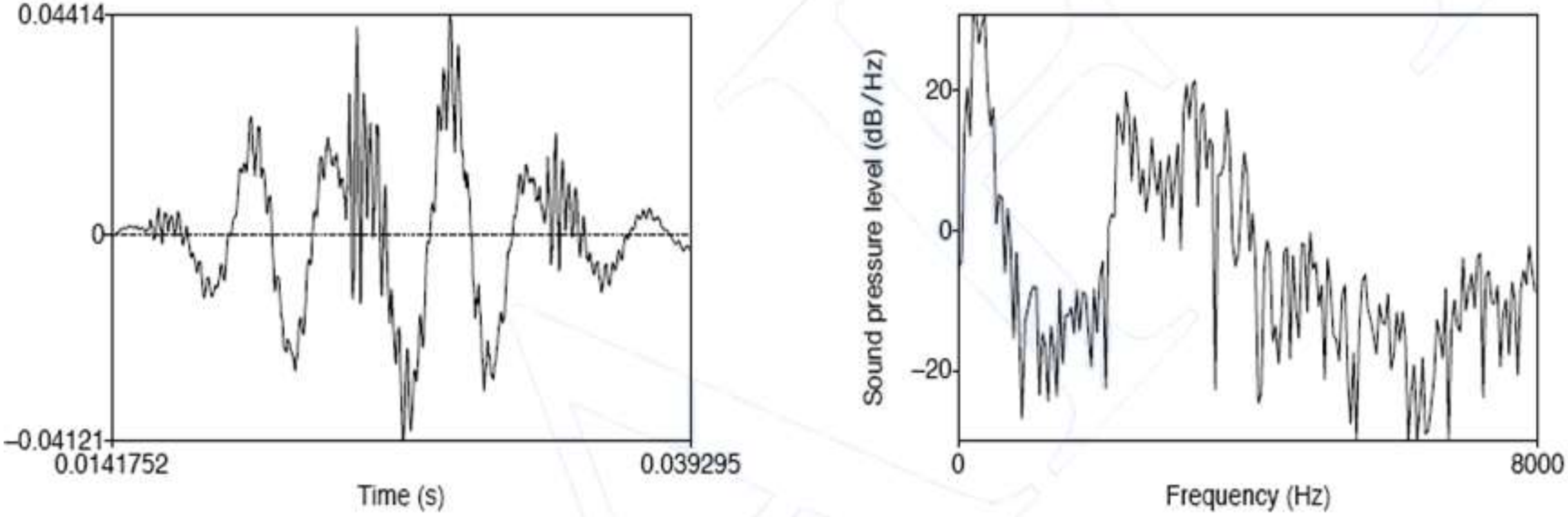
  - If we know $\{X_k\}$, we could calculate x_n as:

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

- This is the Inverse Discrete Fourier Transform, sometimes referred as IDFT.
- Standard algorithm for computing DFT is the Fast Fourier Transform (FFT):
  - Fast Fourier Transform (FFT) with complexity $N * \log(N)$
  - If we can, we choose N to be a power of 2, e.g., $N = 512\ or\ 1024$

Discrete Fourier Transform computes a spectrum

- Bellow we see the signal and the resulting Fast Fourier Transform.



Fast Fourier Transform (FFT)

- Fast Fourier Transform or FFT is a special algorithm for fast calculation of DFT. If your signal has N discrete equally spaced points, FFT calculates the Fourier transform of that signal in the time proportional to $Nlog(N)$. This is very fast.
- FFT is the main component of almost any signal processing toolkit.
- Many Python libraries come with an implementation of FFT, for example: https://docs.scipy.org/doc/scipy/reference/tutorial/fft.html
- https://numpy.org/doc/stable/reference/routines.fft.html

- This algorithm, including its recursive application, was invented around 1805 by Carl Friedrich Gauss, who used it to interpolate the trajectories of the asteroids Pallas and Juno, but his work was not widely recognized (being published only posthumously and in neo-Latin).
- Heideman, M. T., D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast Fourier transform," IEEE ASSP Magazine, 1, (4), 14–21 (1984)

- FFTs became popular after James Cooley of IBM and John Tukey of Princeton published a paper in 1965 reinventing the algorithm and describing how to perform it conveniently on a computer.
- Cooley, James W.; Tukey, John W. (1965). "An algorithm for the machine calculation of complex Fourier series". *Math. Comput.* **19** (90): 297–301.

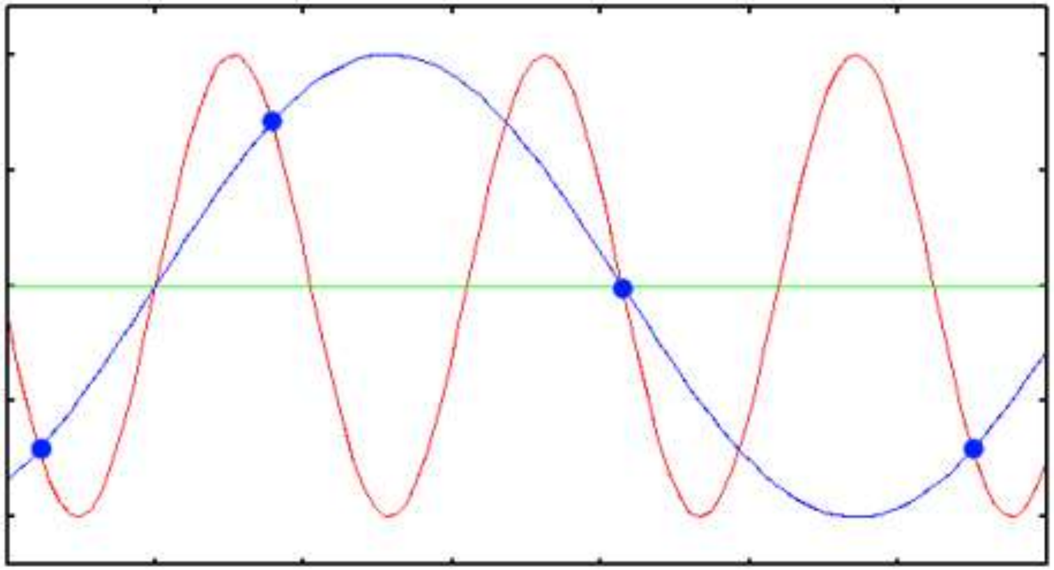
Aliasing

- Aliasing is term that you will encounter in many texts on signal processing. Aliasing refers to a phenomenon which occurs when you try to sample a periodic signal of high frequency at a very low sampling rate. Such sampling might result in an impression that your signal contains a low frequency component.

From Wikipedia:

- In signal processing and related disciplines, aliasing is an effect that causes different signals to become indistinguishable (or aliases of one another) when sampled. It also often refers to the distortion or artifact that results when a signal reconstructed from samples is different from the original continuous signal.
- Aliasing can occur in signals sampled in time, for instance digital audio, and is referred to as temporal aliasing. It can also occur in spatially sampled signals (e.g. moiré patterns in digital images); this type of aliasing is called spatial aliasing.
- Aliasing is generally avoided by applying low pass filters or anti-aliasing filters (AAF) to the input signal before sampling and when converting a signal from a higher to a lower sampling rate. Suitable reconstruction filtering should then be used when restoring the sampled signal to the continuous domain or converting a signal from a lower to a higher sampling rate.

Aliasing



- *The blue dots are samples measured with a frequency lower than the frequency of the red wave. The frequency isn't high enough to reconstruct the original signal, and the measurement produces a signal (blue curve) of wrong frequency*

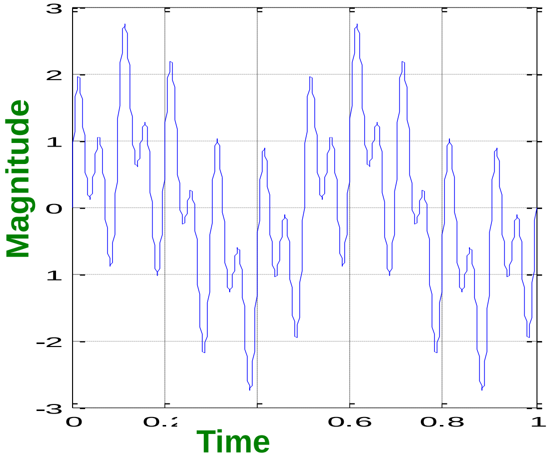
- **Aliasing** – from alias – is an effect that makes different signals indistinguishable when sampled. It also refers to the difference between a signal reconstructed from samples and the original continuous signal, when the resolution is too low. Basically, aliasing depends on the sampling rate and freqency content of the signal.

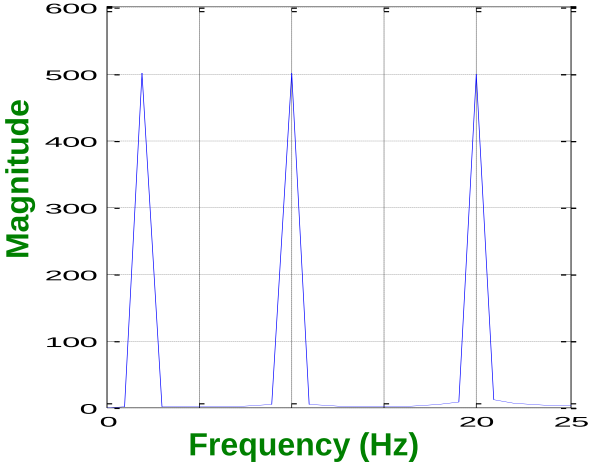
Short Time Fourier Transform
&
Spectrograms

Stationarity of Signal

2 Hz + 10 Hz + 20Hz

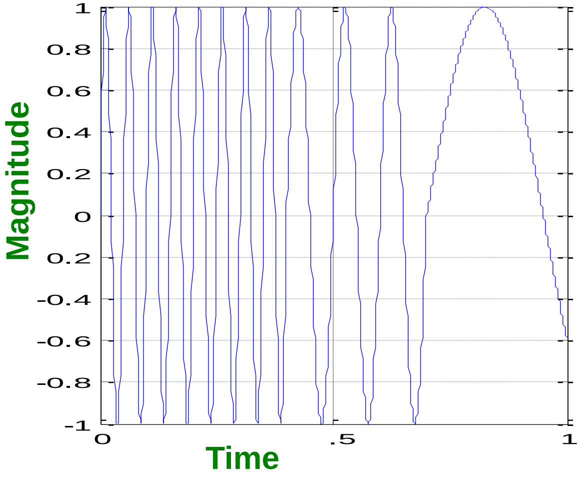
Stationary, i.e. repeatable, signal results in well defined, sharp features in frequency space

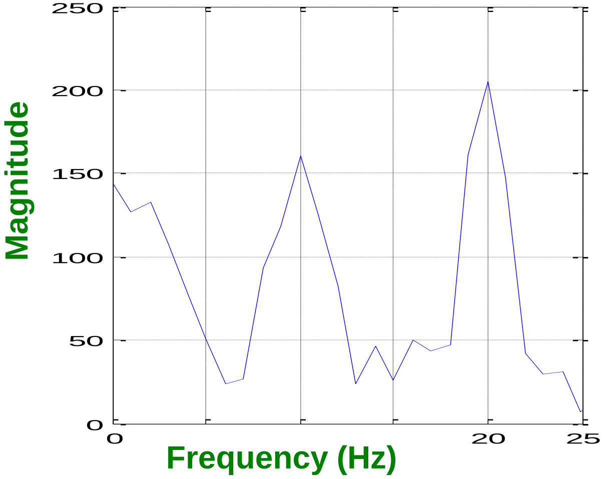




0.0-0.4: 2 Hz +
0.4-0.7: 10 Hz +
0.7-1.0: 20Hz

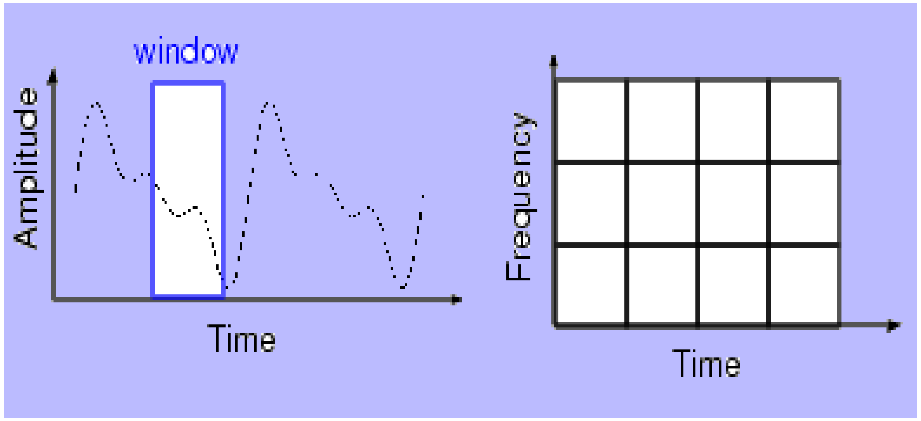
Non-Stationary, i.e. non- repeatable signal results in broad features in frequency space





Short Time Fourier Transform (STFT)

- Dennis Gabor, Nobel prize for Holography, used Short Time Fourier Transform (STFT) in 1946
  - To analyze only a small section of the signal at a time –
  - The technique is sometimes called *Windowing the Signal***.**
- The small segment of signal is assumed *stationary*
- A 2D transform



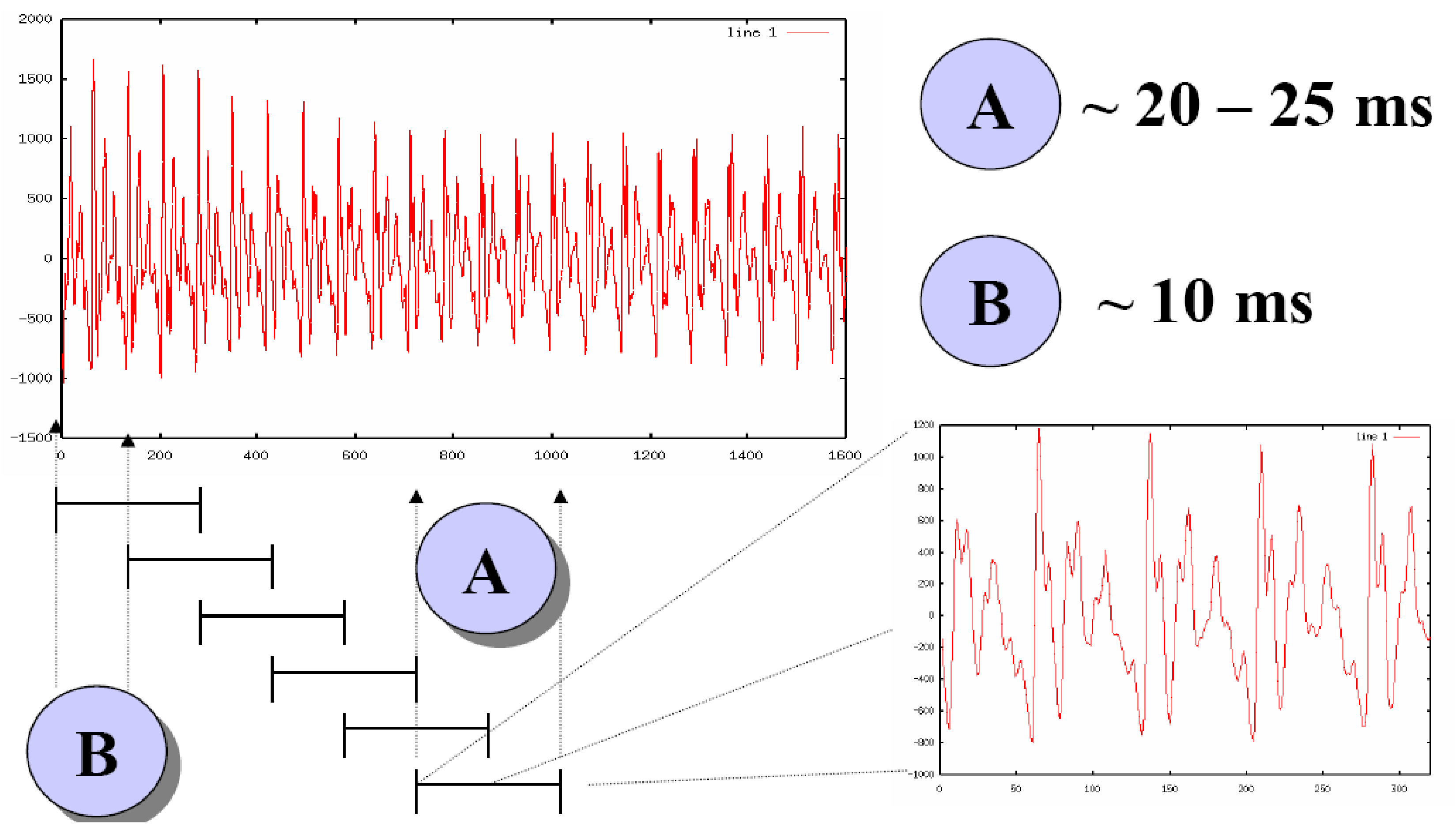
$$\mathrm{STFT}_X^{(\omega)}(t', f) = \int_t \left[x(t) \bullet \omega^*(t - t')\right] \bullet e^{-j2\pi ft} dt$$

$$\omega(t): \text{the window function}$$

A function of time and frequency

Windowing, Width and Offset

- Within the short window the signal is stationary (almost periodic), not simple periodic. Such signal could be readily Fourier analyzed to give us its spectral content



A ~ 20 – 25 ms

B ~ 10 ms

Slide from Bryan Pellom

Windowing

- Why divide speech signal into successive overlapping frames?
  - Speech is not a stationary signal; we want information about a small enough region that the spectral information is a useful cue.
- Frames
  - Frame size: typically, 10-25ms
  - Frame shift: the length of time between successive frames, typically, 5-10ms
  - Frames overlap.
- The extraction of the signal takes place by multiplying the value of the signal at time $n, s[n]$, with the value of the window at time $n, w[n]$:
  - $y[n] \ = \ w[n]s[n]$

Windowing

- Slides 32 and 33 suggest that these window shapes are rectangular, since the extracted windowed signal looks just like the original signal.
- Indeed the simplest window is the **rectangular window**. The rectangular window can cause problems, however, because it abruptly cuts the signal at its boundaries. These discontinuities create problems when we perform the Fourier transformation. Rectangular window generates a lot of high frequency noise.
- For this reason, a more common window used in MFCC extraction is the **Hamming window**, which shrinks the values of the signal toward zero at the window boundaries, avoiding discontinuities.
- The next slide shows both of these windows and the equations representing them (assuming a window is L frames long).

Common Window Shapes

- **Rectangular window**:
- $$w[n] = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & otherwise \end{cases}$$
- cause high frequency noise in FFT and DFT

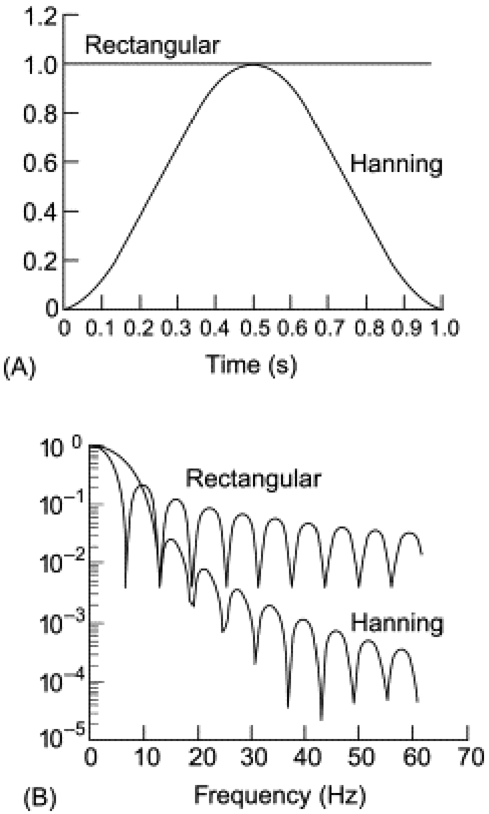
- **Hamming window:**

$$w[\mathrm{n}] = \begin{cases} \left[0.54 - 0.46\cos\left(2\pi\,\frac{n}{L-1}\right)\right], & 0 \le n \le L-1 \\ 0\ , & otherwise \end{cases}$$

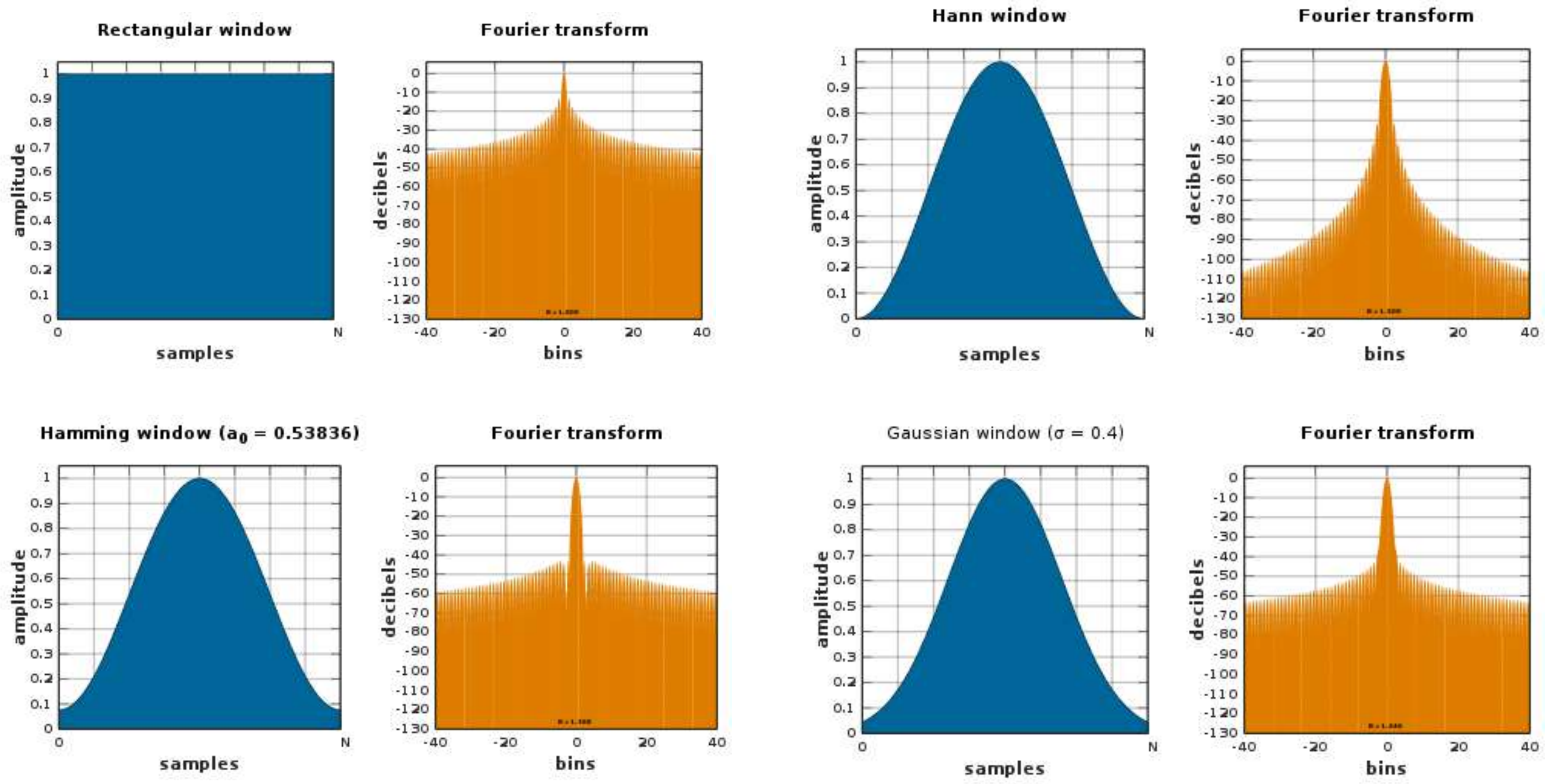
- avoids discontinuities and causes much less high-frequency noise.
- **Hann (Julius) window**:

$$w_H[\mathrm{n}] = \begin{cases} 0.5\left[1 - \cos\left(2\pi\,\frac{n}{L-1}\right)\right], & 0 \le n \le L-1 \\ 0\ , & otherwise \end{cases}$$

- Reduces high-frequency noise even more, though it reduces signal close to the edges to zero, what is sometimes undesirable.

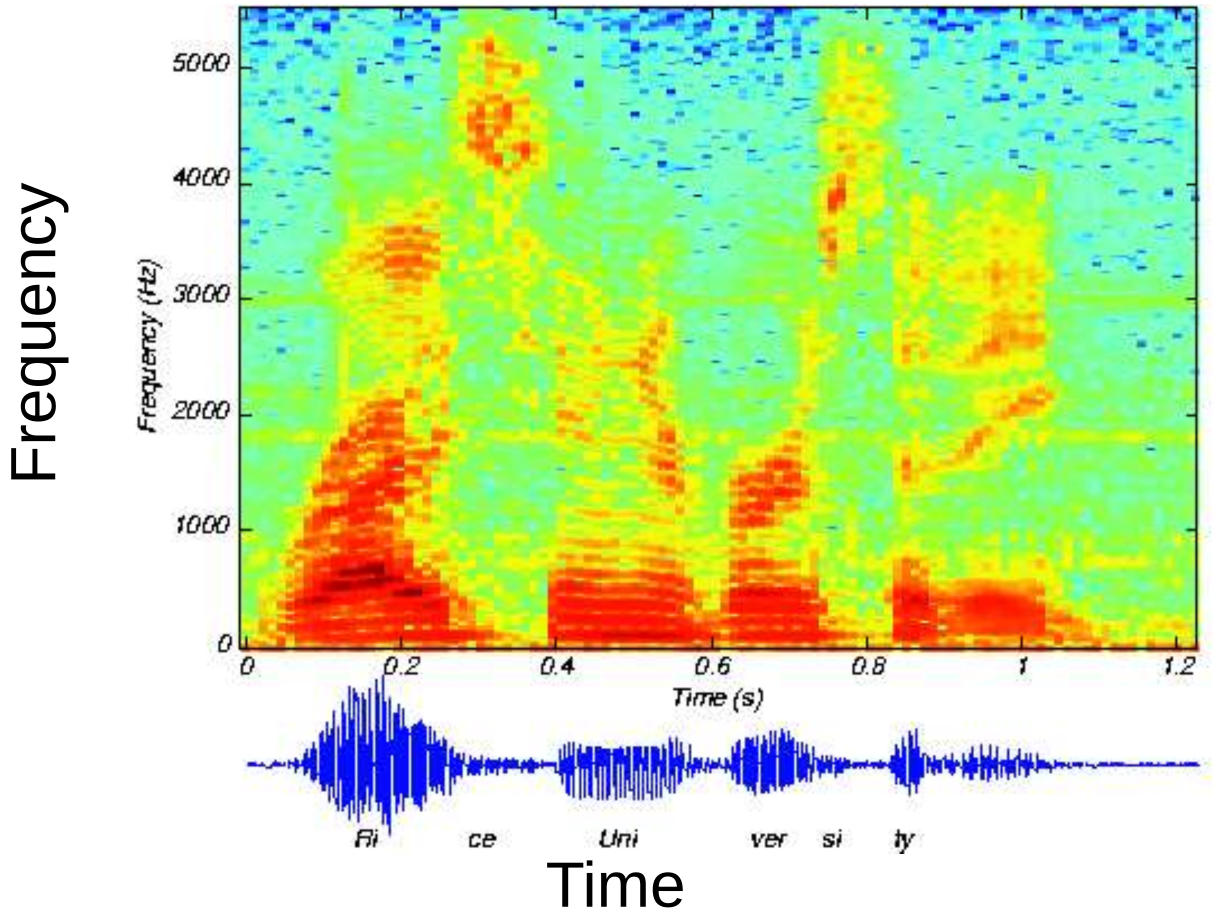


Various Windows and the effect in the Frequency Domain



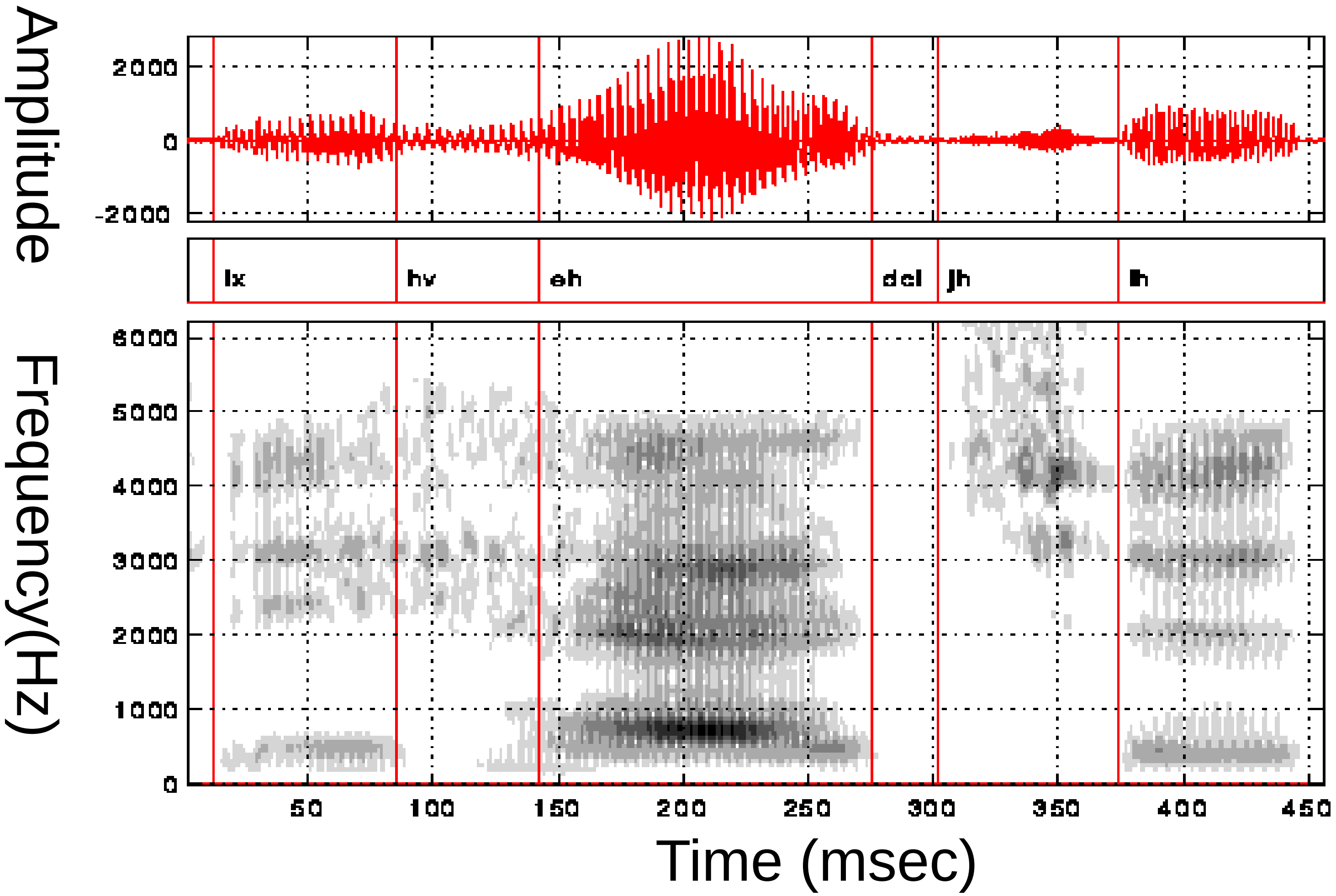
Spectrogram, Picture of your Voice

- Result of applying the DFT on every sliding window over the duration of a signal is called a Spectrogram.
- Spectrogram has *time* on the horizontal axis and the frequency along y axis. Every point is an amplitude of the Fourier transform (DFT) at given time and frequency.
- Say: “Rice University”. Strong, almost horizontal, bands are called **formants.**
- The strongest sounds are vowels. Each vowel has a set of distinct bands, the formants.



Spectrogram and Signal Strength

- Spectrogram compared to a time dependent signal strength



Seeing Formants: the spectrograms of vowels

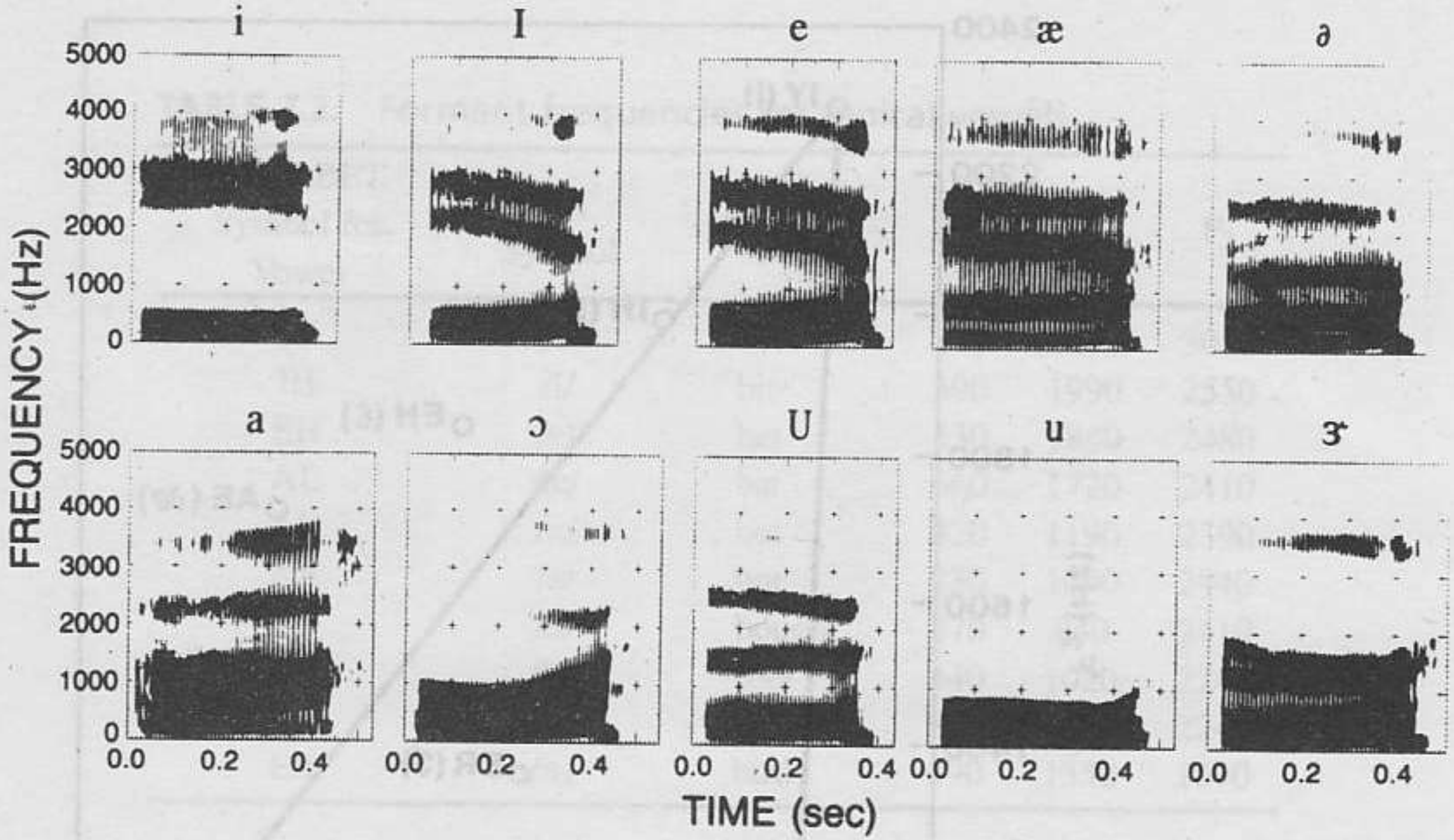
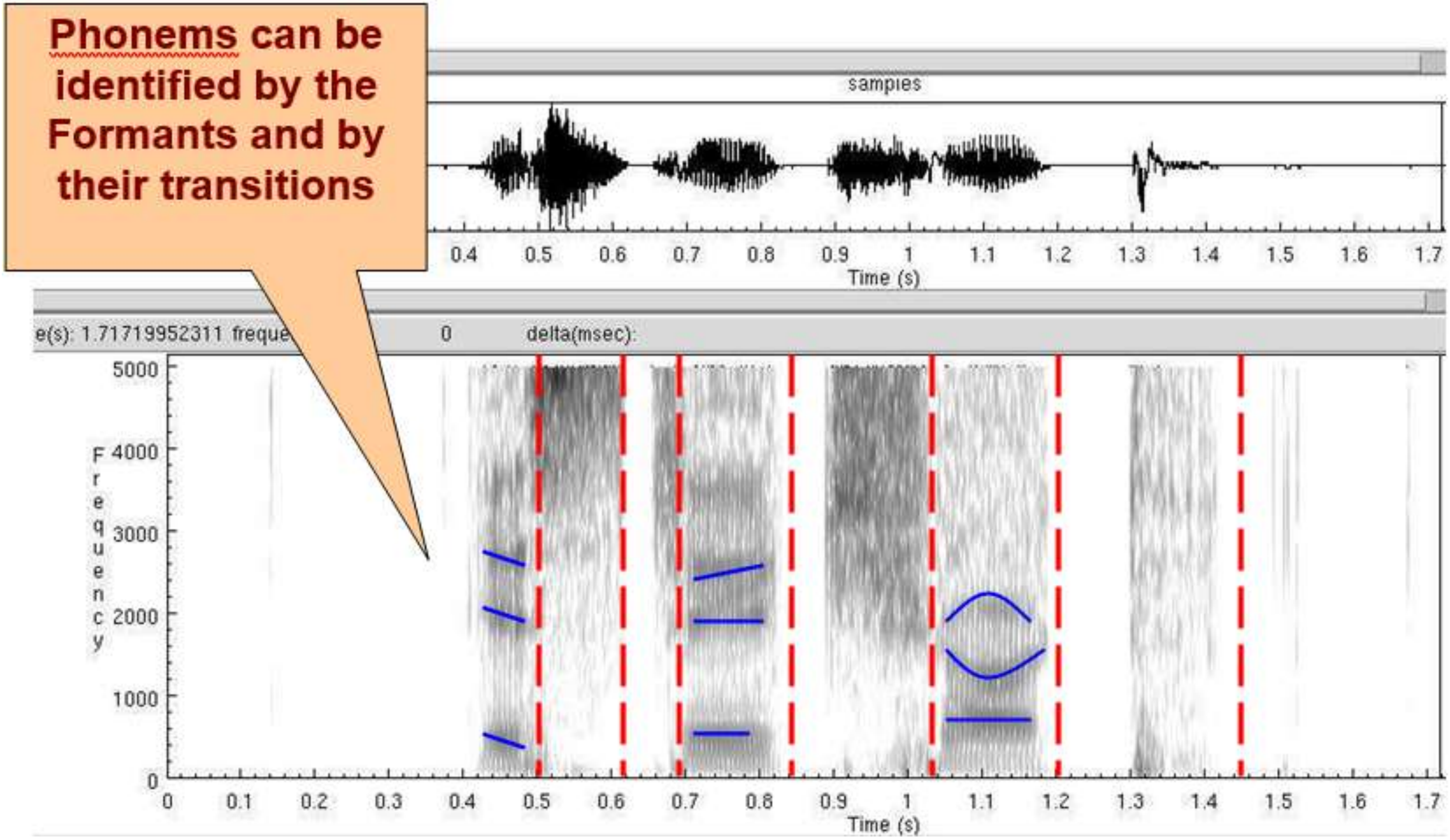


Figure 2.15 Spectrograms of the vowel sounds.

Why we bother with Formants

- Note that central frequencies F1, F2, etc. of the main formants are not constant in time but rather could raise, or fall, go up and then down and have other forms.



Phonems can be identified by the Formants and by their transitions

Recognizing Vowels: Extract Formants from Spectrogram

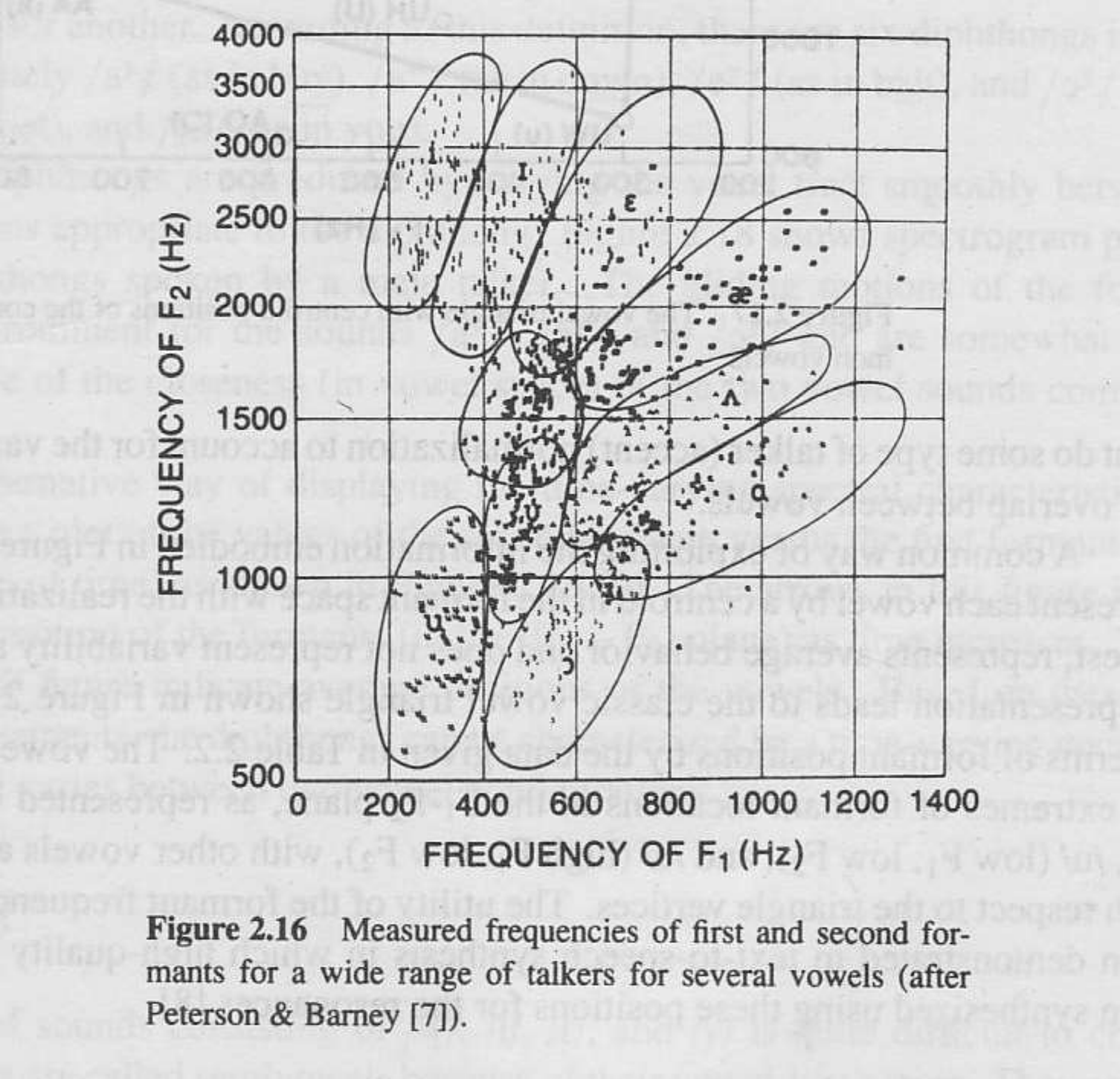
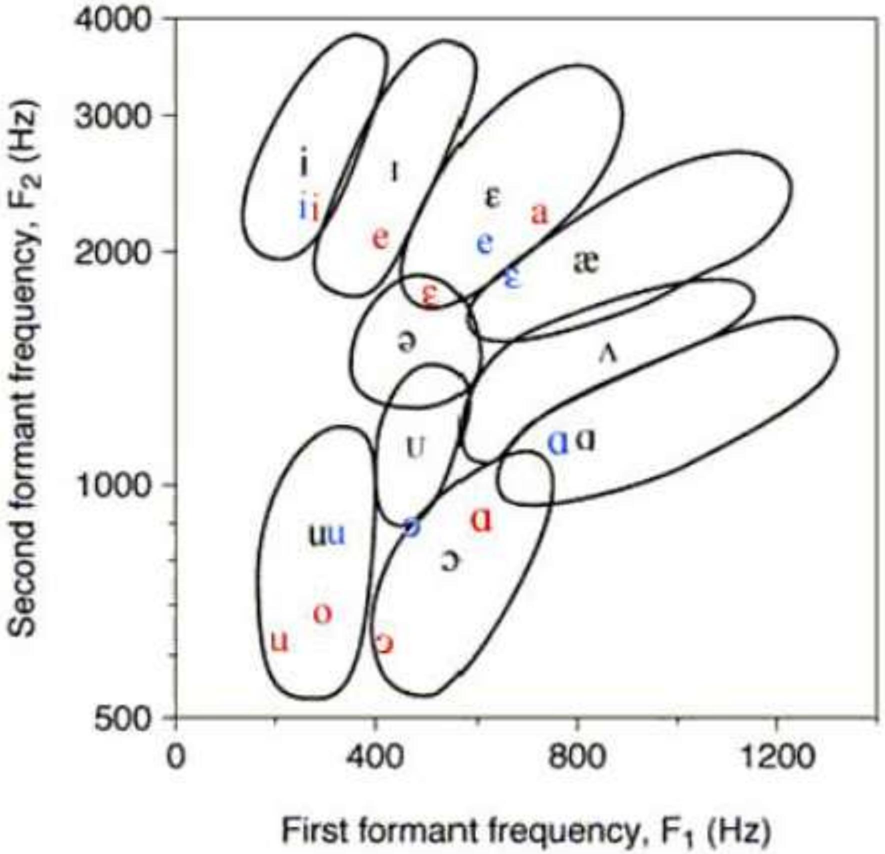


Figure 2.16 Measured frequencies of first and second formants for a wide range of talkers for several vowels (after Peterson & Barney [7]).

Vowels, F1 and F2 Frequencies

- This data combine two studies. Red points are from J.M. McCarty https://ccrma.stanford.edu/~jmccarty/formant.htm. Blue points are from *Louis Goldstein - Haskins Laboratories*



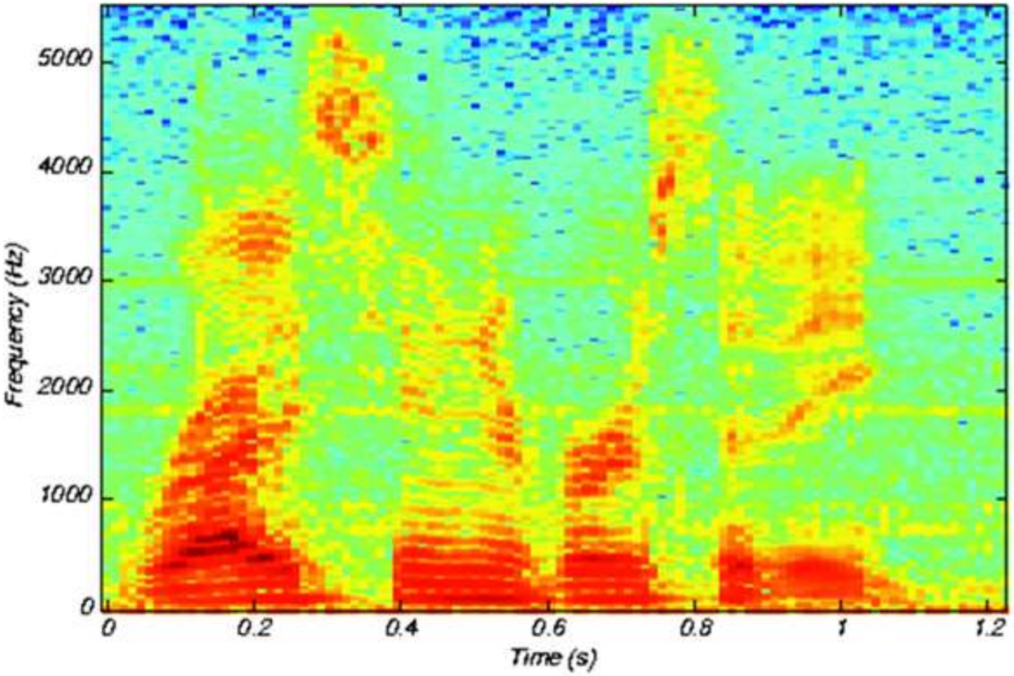
Spectrograms are Information Diagrams

Information Diagrams and Spectrograms

- Gabor took the Uncertainty Principle further with a great insight by noting that an Information Diagram representation of a signal in a plane defined by the axes of time and frequency is fundamentally quantized.
- There is an irreducible, minimal, area that any signal can possibly occupy in this plane. The uncertainty (or spread) in frequency, times the uncertainty (or duration) in time, has a lower bound.

$$\Delta t \cdot \Delta\omega \geq \frac{1}{4\pi}$$

- The spectrogram, like the one on slide 38, and reproduced here, is truly made of a large set of discrete rectangles (regions) of size $1/4\pi$.
- Please note, the rectangles you see on the image to the right are the consequence of digitalization and pixilation. Gabor's regions should be similar.



Uncertainty Principle, Quantization of Information Diagram

- This relationship is identical to the uncertainty relationship in quantum mechanics, where $f(x)$ is interpreted as the position of a particle, and $F(k)$ would be interpreted as its momentum. Momentum k is the inverse of $de\ Broglie$ wavelength $k = 2\Pi/\lambda$.
- Relationship $(\Delta x)(\Delta k) \geq \frac{1}{4\pi}$ is not just a property of quantum particles, but more generally a property of all functions and their Fourier Transforms.
- This relationship tells us that the information in continuous signals is quantized, since the minimal area they can occupy in the Information Diagram has an irreducible lower bound.
- Information Diagram has finite granularity.
- You cannot make grains of information, i.e. information quanta smaller than

$$(\Delta x)(\Delta k) \geq \frac{1}{4\pi}$$

- This also means that you cannot improve representation of a signal by adding more data points beyond certain upper bound.

Gabor's Logons

- Dennis Gabor named such minimal areas logons from the Greek word for information, or order: *logos*.

- Gabor established that the Information Diagram for any continuous signal can only contain a fixed number of information quanta.

- Each such quantum constitutes an independent datum (information value), and their total number within a region of the Information Diagram represents the number of independent degrees-of-freedom enjoyed by the signal.

- There is a unique family of signals that could actually be used to decompose an arbitrary signal into a decomposition with the lower bound in the Gabor-Heisenberg-Weyl Uncertainty Relation.

- Those signals are complex exponentials multiplied by Gaussians.
- They are sometimes referred to as *Gabor Wavelets*.

Mel Scale

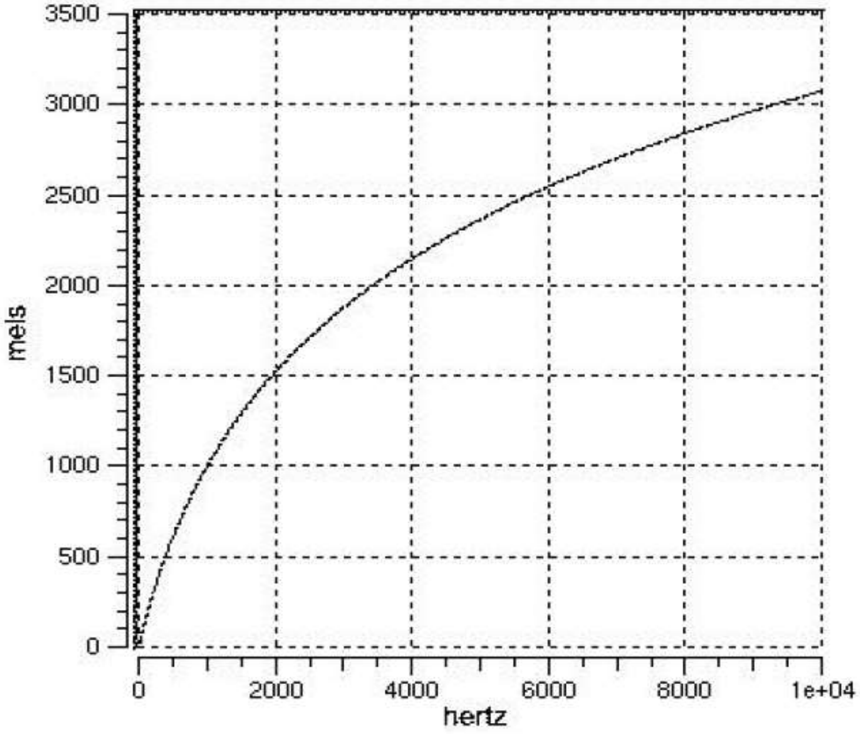
Mel Scale

- The **mel scale**, Stevens, Volkman and Newman (1937) is a perceptual scale of pitches judged by listeners to be equal in distance from one another.
- The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels.
- Above about 500 Hz, larger and larger intervals are judged by listeners to produce equal pitch increments. As a result, four octaves on the hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale.
- In music, an **octave** (Latin: *octavus*: eighth) or **perfect octave** is the interval between one musical pitch and another with half or double its frequency. It is defined by ANSI[2] as the unit of frequency level when the base of the logarithm is two. The octave relationship is a natural phenomenon that has been referred to as the "basic miracle of music", the use of which is "common in most musical systems".[3]
- The most important musical scales are typically written using eight notes, and the interval between the first and last notes is an octave.
- The name **mel** comes from the word **melody** to indicate that the scale is based on pitch comparisons.

Mel-scale

- Human hearing is not equally sensitive to all frequency bands
- Less sensitive at higher frequencies, roughly > 1000 Hz
- Human perception of frequency is non-linear.

The **mel scale**, Stevens, Volkman and Newman (1937) is a perceptual scale of pitches judged by listeners to be equal in distance from one another.



On-line Tone Generator

- You can verify that you can distinguish low frequencies easily but with great difficulty high frequencies using this on-line tone generator:

  https://www.szynalski.com/tone-generator/

Pitch, Mel-scale

- ***Pitch*** is a perceptual property of sounds that allows their ordering on a frequency-related scale, or more commonly, *pitch is the quality that makes it possible to judge sounds as "higher" and "lower" in the sense associated with musical melodies*.
- Pitch can be determined only in sounds that have a frequency that is clear and stable enough to distinguish from noise. Pitch is a major auditory attribute of musical tones, along with duration, loudness, and timbre.

- A **mel** is a unit of pitch
  - Definition:
    - Pairs of sounds perceptually equidistant in pitch
      - Are separated by an equal number of mels:

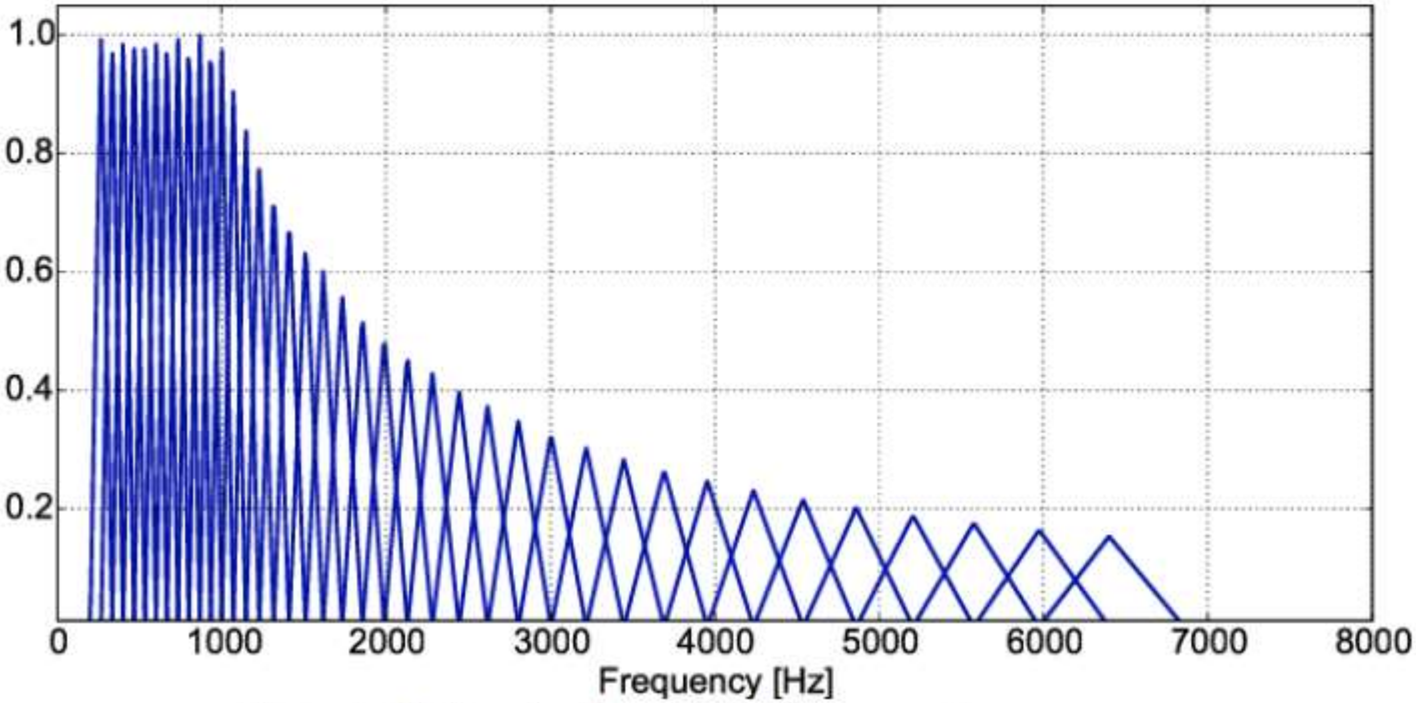
- Mel-scale is approximately linear below 1 kHz and logarithmic above 1 kHz

- Definition of Mel Frequency distribution

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

Non-linear Sampling

- Frequencies needed to reconstruction of sound should be sampled initially in a linear manner and then logarithmically. This is illustrated by the “filter bank”



Filter bank of 40 triangular filters in the Mel-frequency scale

Feature Extraction
Mel Frequency Cepstral Coefficients, MFCC

Cepstrum

- Noun “cepstrum” was derived by reversing the first four letters of "spectrum".
- A **cepstrum** (pronounced /kɛpstrəm/) is the result of taking the Fourier transform (DFT) of the log spectrum as if it were a signal.
- Log of the product of Fourier transform of the basic wave and the filter separates the two into a sum. That is why Cepstrum can distinguish between the filter (actual phonem) and the basic wave.
- The power cepstrum (of a signal) is the squared magnitude of the Fourier transform of the logarithm of the squared magnitude of the Fourier transform of a signal.

Power cepstrum of signal *f(t)*

$$= \left|\mathcal{F}\left\{\log(|\mathcal{F}\{f(t)\}|^2)\right\}\right|^2$$

MFCC, Mel-frequency cepstrums

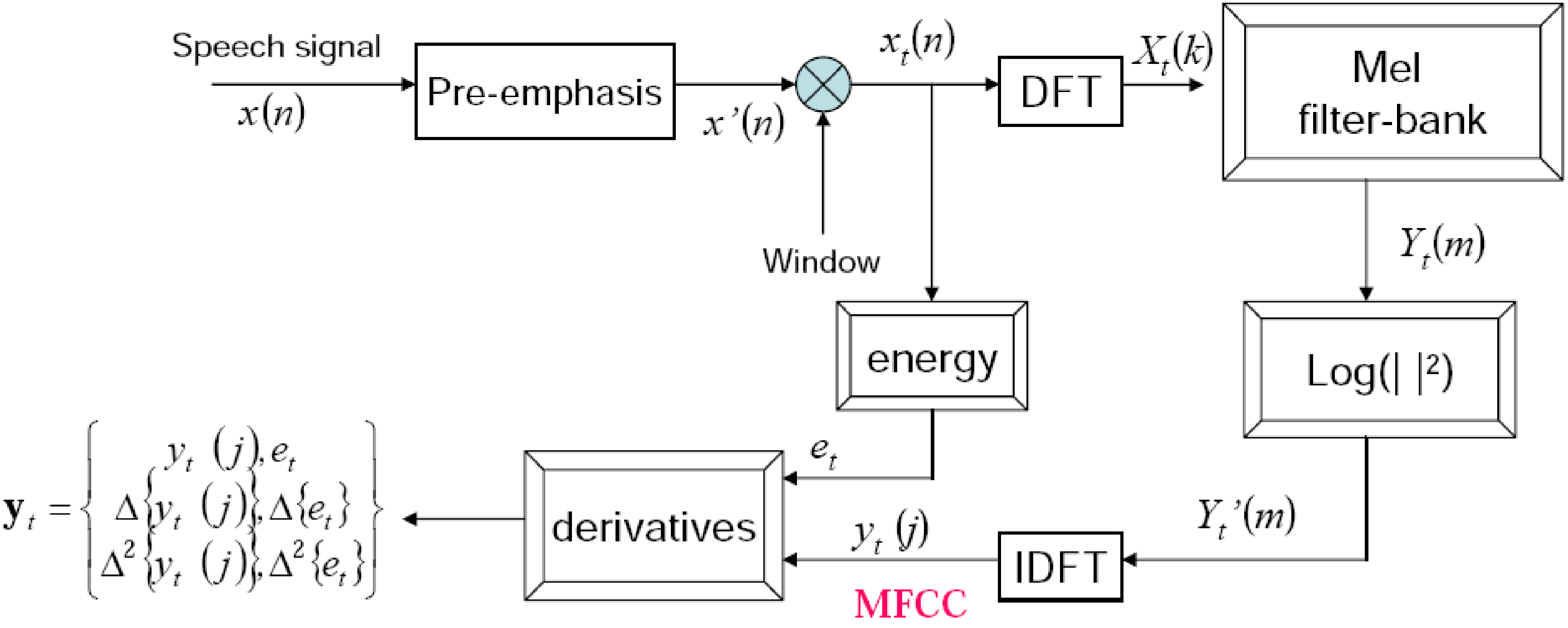
- In sound processing, the **mel-frequency cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

- **Mel-frequency cepstral coefficients** (**MFCCs**) are coefficients that collectively make up an MFC.
- The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal spectrum.

- MFCCs are commonly derived as follows:
  1. Take the Fourier transform of (a windowed excerpt of) a signal.
  2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
  3. Take the logs of the powers at each of the mel frequencies.
  4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
  5. The MFCCs are the amplitudes of the resulting spectrum.

Mel-Frequency Cepstral Coefficient (MFCC)

- MFCC is the most widely used spectral representation in ASR.
- Calculation of MFCC-s is a feature extraction process.
- Extraction of MFCC is a 7 or 8 Step Process

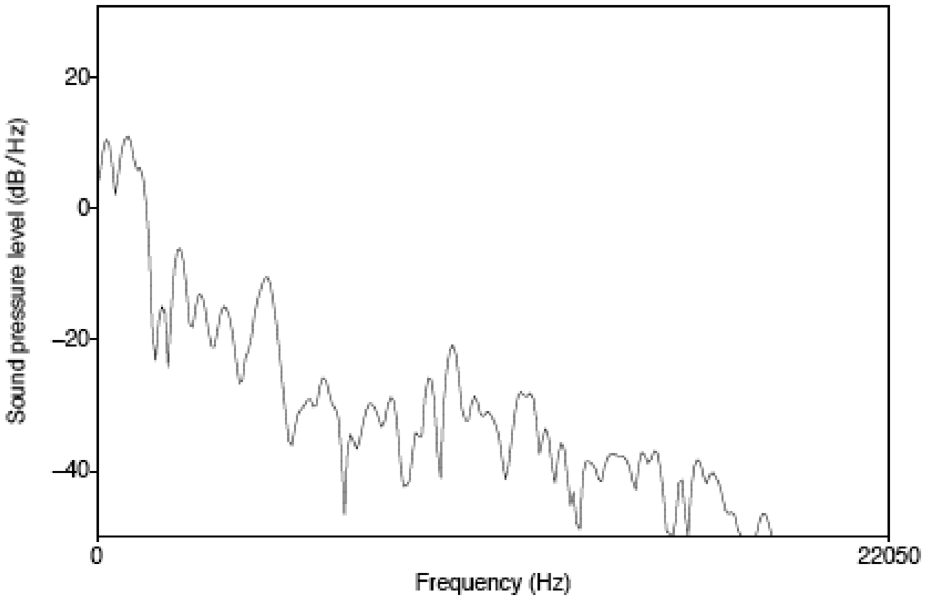


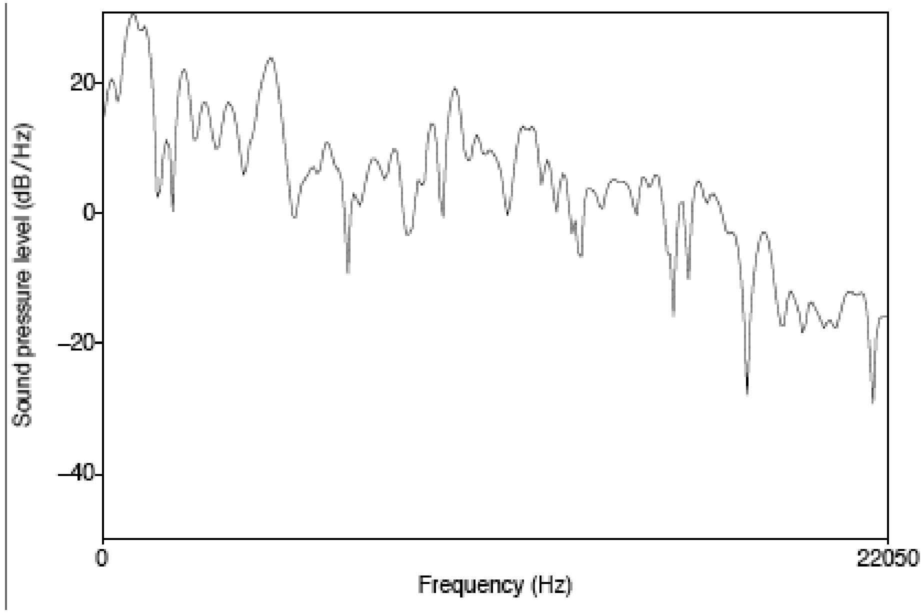
1. MFCC, Pre-Emphasis

- The first stage in MFCC feature extraction is to boost the amount of energy in the high frequencies. It turns out that if we look at the spectrum for voiced segments like vowels, there is more energy at the lower frequencies than the higher frequencies.
- This drop in energy across frequencies (which is called **spectral tilt**) is caused by the nature of the glottal pulse.
- Boosting the high frequency energy makes information from these higher formants more available to the acoustic model and improves phone detection accuracy.
- This *preemphasis* is done by using a filter. Next slide shows an example of a spectral slice from a pronunciation of the single vowel [aa] before and after *preemphasis*.

Example of pre-emphasis

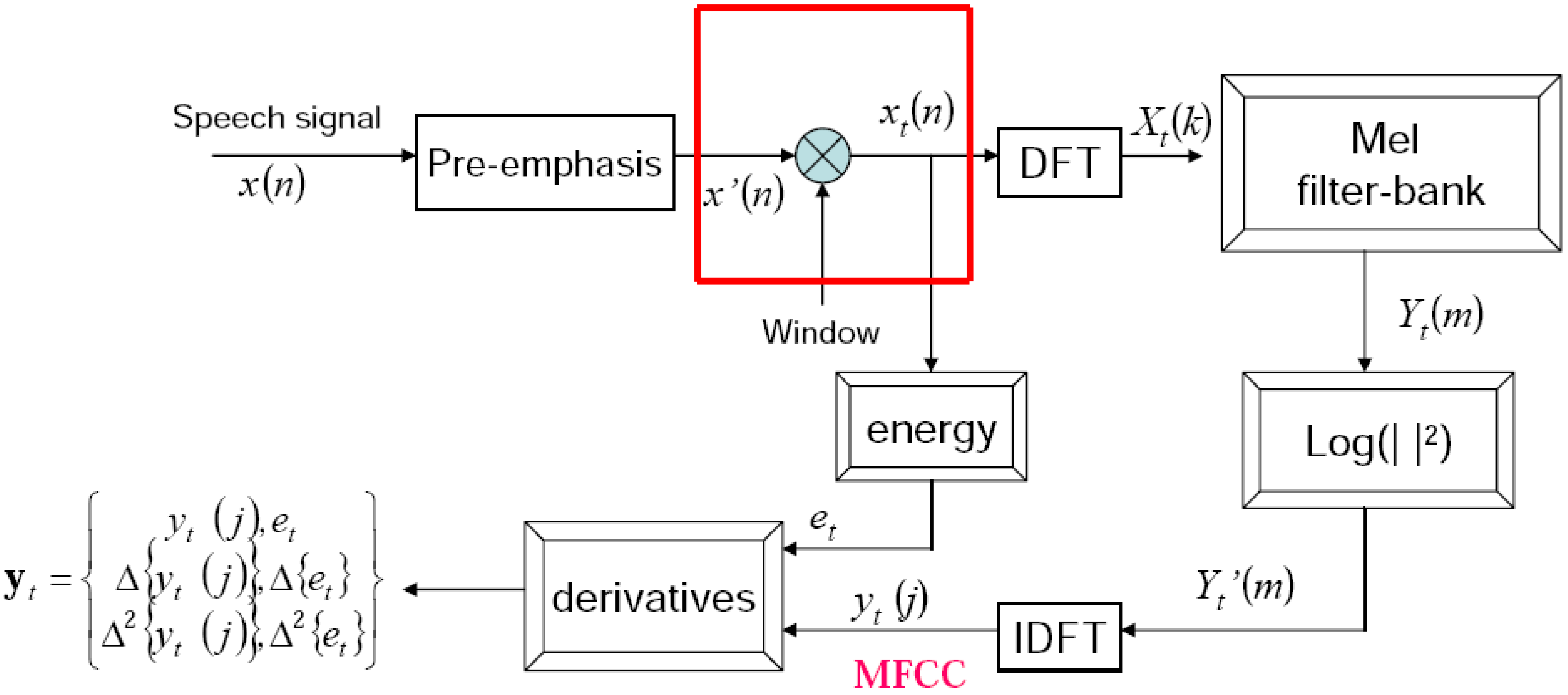
- Before and after pre-emphasis
  - Spectral slice from the vowel [aa]





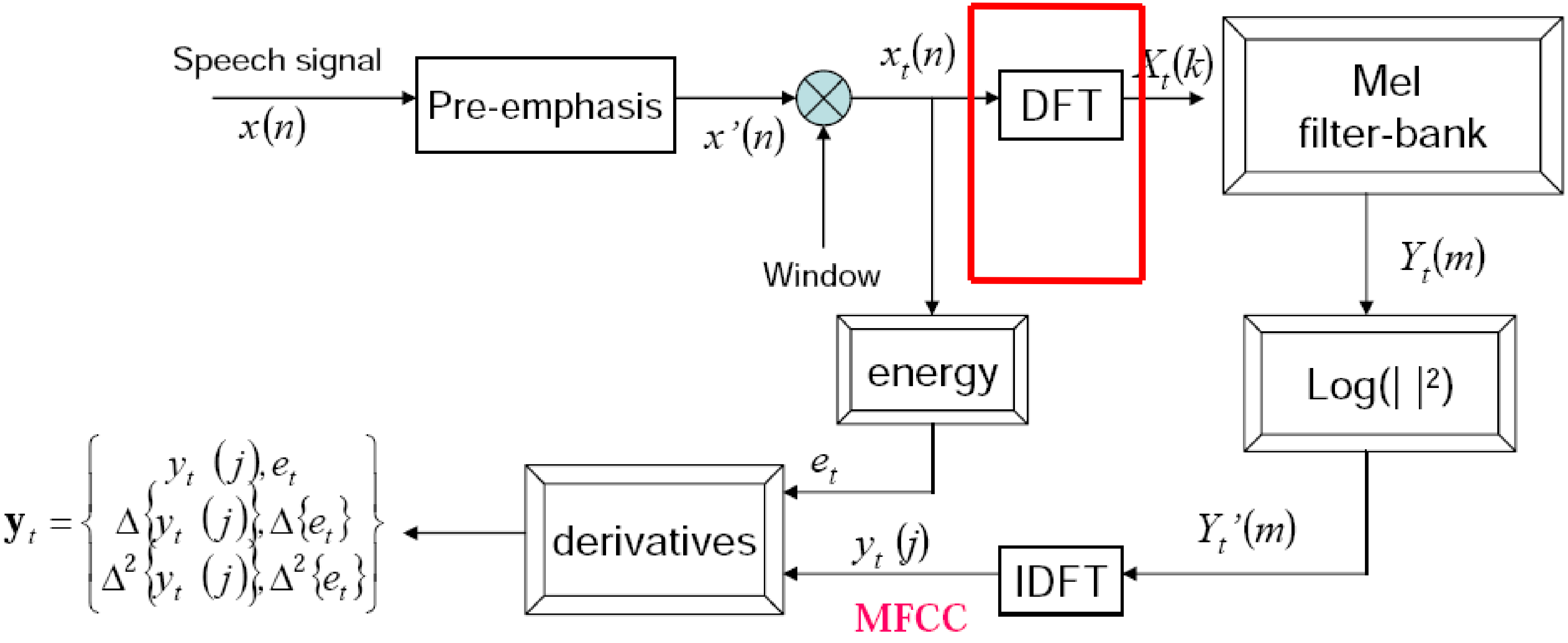
2. MFCC, Windowing

- The speech extracted from each window is called a frame. The windowing is characterized by three parameters: the window size or frame size of the window stride (its width in milliseconds), the frame stride, (also called shift or offset) between successive windows, and the shape of the window.
- To extract the signal we multiply the value of the signal at time n, s[n] by the value of the window at time n, w[n]:



3. MFCC, Frequency Analysis, DFT

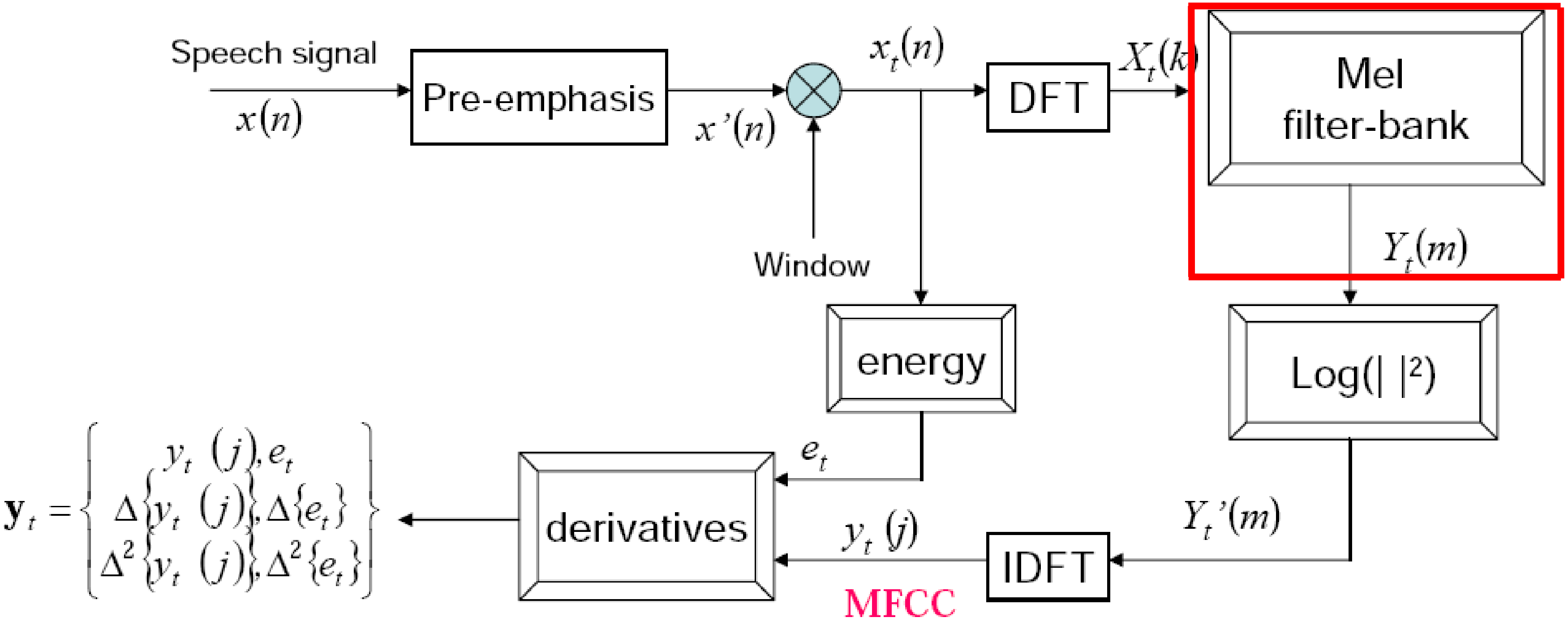
- We need to know how much energy the signal contains at different frequency bands. The tool for extracting spectral information for discrete frequency bands for a discrete-time (sampled) signal is the Discrete Fourier Transform or DFT.



Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
energy
Log(| |²)
e_t
derivatives
$y_t(j)$
IDFT
$Y_t'(m)$
MFCC
$\mathbf{y}_t = \left\{ \begin{array}{c} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{array} \right\}$

4. MFCC, MEL Filter Bank

- The results of the FFT tell us the energy at each frequency band. Human hearing, however, is not equally sensitive at all frequency bands; it is less sensitive at higher frequencies. This bias toward low frequencies helps human recognition, since information in low frequencies like formants is crucial for distinguishing values or nasals, while information in high frequencies like stop bursts or fricative noise is less crucial for successful recognition.
- Modeling this human perceptual property improves speech recognition performance in the same way.
- We implement this intuition by collecting energies, not equally at each frequency band, but according to the **mel** scale, an auditory frequency scale

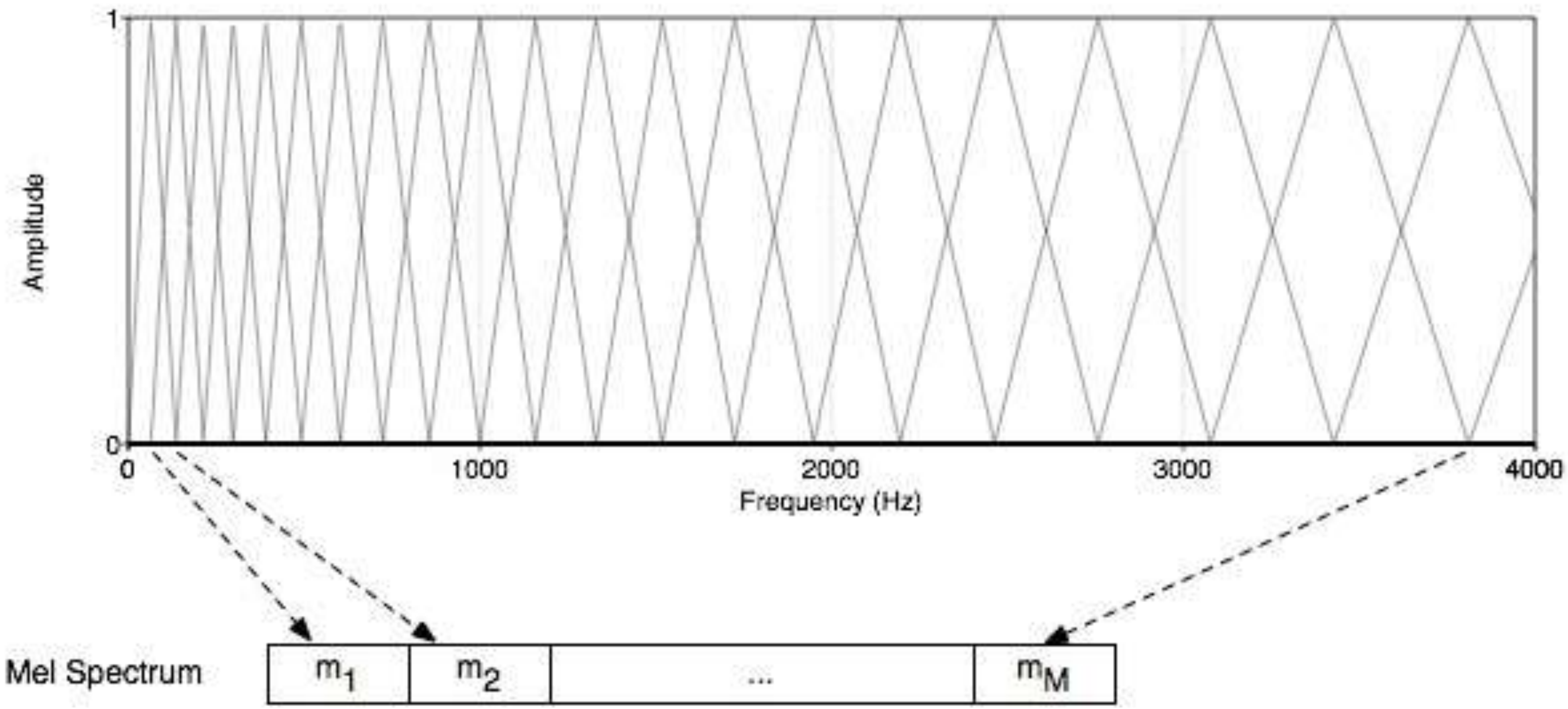


MEL Filter Bank Process

- During MFCC computation, this process is implemented by creating a bank of filters which collect energy from each frequency band, with 10 filters spaced linearly be- low 1000 Hz, and the remaining filters spread logarithmically above 1000 Hz.
- Figure on the next slideshows the bank of triangular filters that implement this idea.

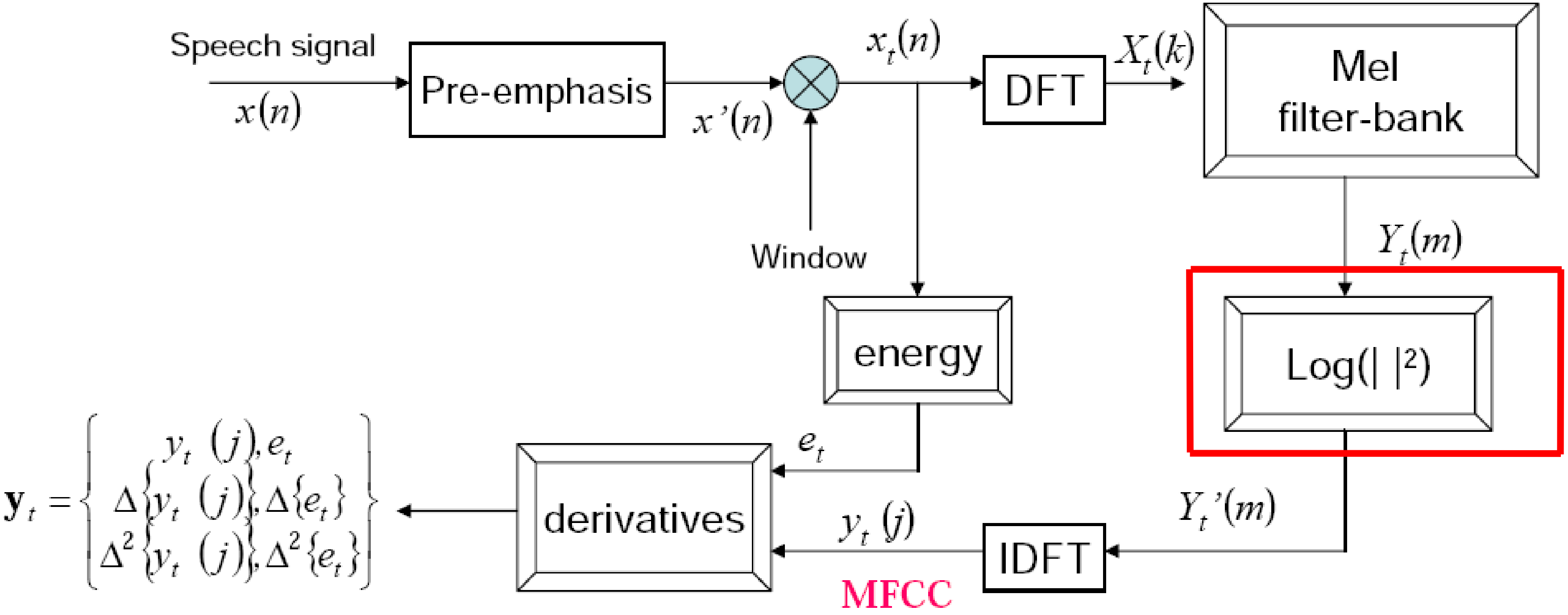
Mel Filter Bank Processing

- Mel Filter bank
  - Uniformly spaced before 1 kHz
  - Logarithmic scale after 1 kHz



5. MFCC, Log Energy Compute

- Finally, we take the log of each of the mel spectrum values. In general the human response to signal level is logarithmic; humans are less sensitive to slight differences in amplitude at high amplitudes than at low amplitudes. In addition, using a log makes the feature estimates less sensitive to variations in input (for example power variations due to the speaker's mouth moving closer or further from the microphone).

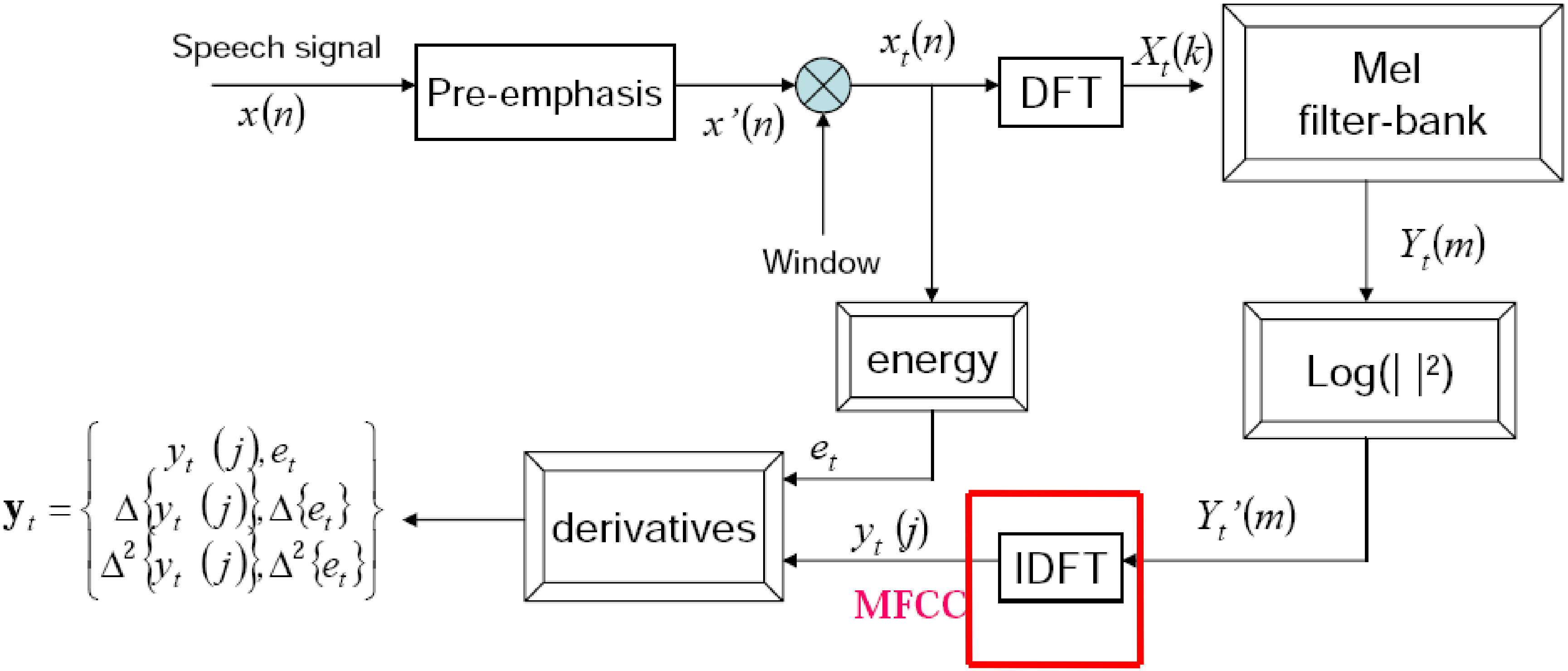

Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
energy
e_t
derivatives
$\mathbf{y}_t = \left\{ \begin{matrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{matrix} \right\}$

Log energy computation

- Why log energy?
- Logarithm compresses dynamic range of values
  - Human response to signal level is logarithmic
  - humans less sensitive to slight differences in amplitude at high amplitudes than low amplitudes
- Makes frequency estimates less sensitive to slight variations in input (power variation due to speaker’s mouth moving closer to mike)
- Phase information not helpful in speech

6. MFCC, Cepstrum, Inverse DFT

- It would be possible to use the **mel spectrum** by itself as a feature representation for phonem detection. The spectrum has some problems. It might be too noisy.
- For this reason the next step in MFCC feature extraction is the computation of the cepstrum. The cepstrum has a number of useful processing advantages and also significantly improves phonem recognition performance.

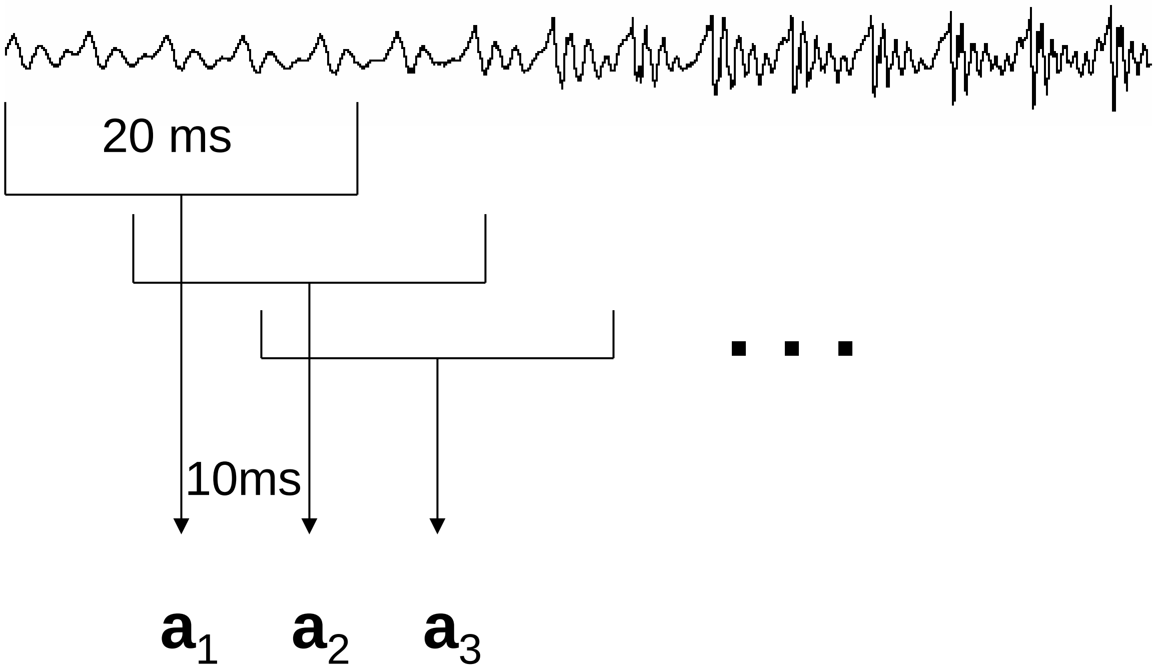

Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
energy
e_t
IDFT
$y_t(j)$
MFCC
derivatives
$\mathbf{y}_t = \left\{ \begin{matrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{matrix} \right\}$

Speech is a Product of Base signal and Filter

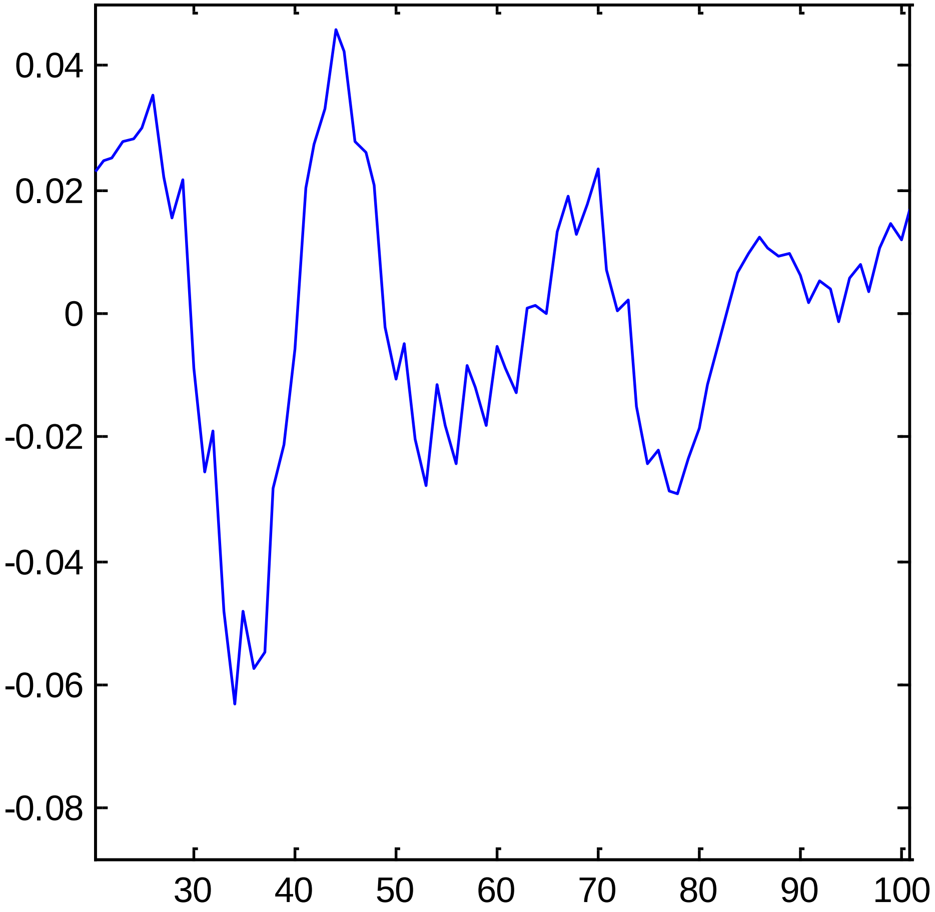
- Calculating cepstrums is as a useful way of separating the source and the filter.
- As described on slides 7 to 11 the speech waveform is created when a glottal source waveform of a particular fundamental frequency is passed through the vocal tract, which because of its shape has a particular filtering characteristic.
- But many characteristics of the glottal source (its fundamental frequency, the details of the glottal pulse, etc) are not important for distinguishing different phones. Instead, the most useful information for phone detection is the filter, i.e. the exact position of the vocal tract.
- If we knew the shape of the vocal tract, we would know which phone was being produced. This suggests that useful features for phone detection would find a way to deconvolve (separate) the source and filter and show us only the vocal tract filter. The cepstrum is one way to do this.

Speech recognition overviews

- Preprocessing
  - FFT to get spectral information
  - Frame extraction - 20 ms wide extracted every 10 ms



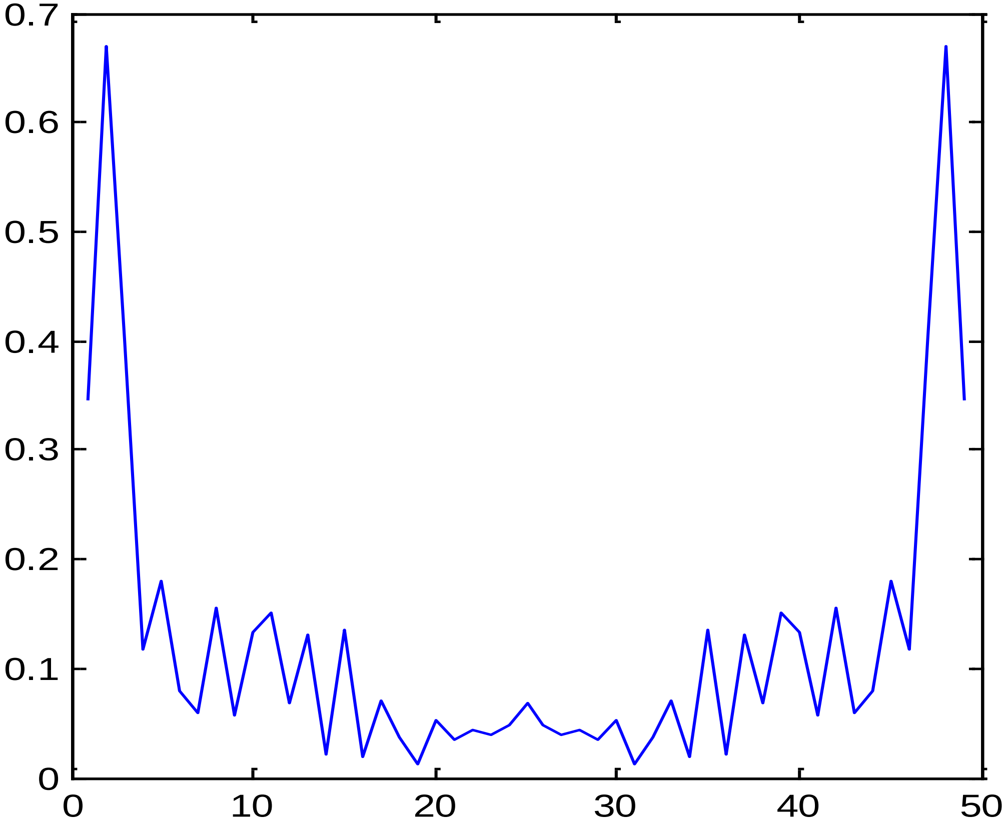
Each segment is sampled 256 times



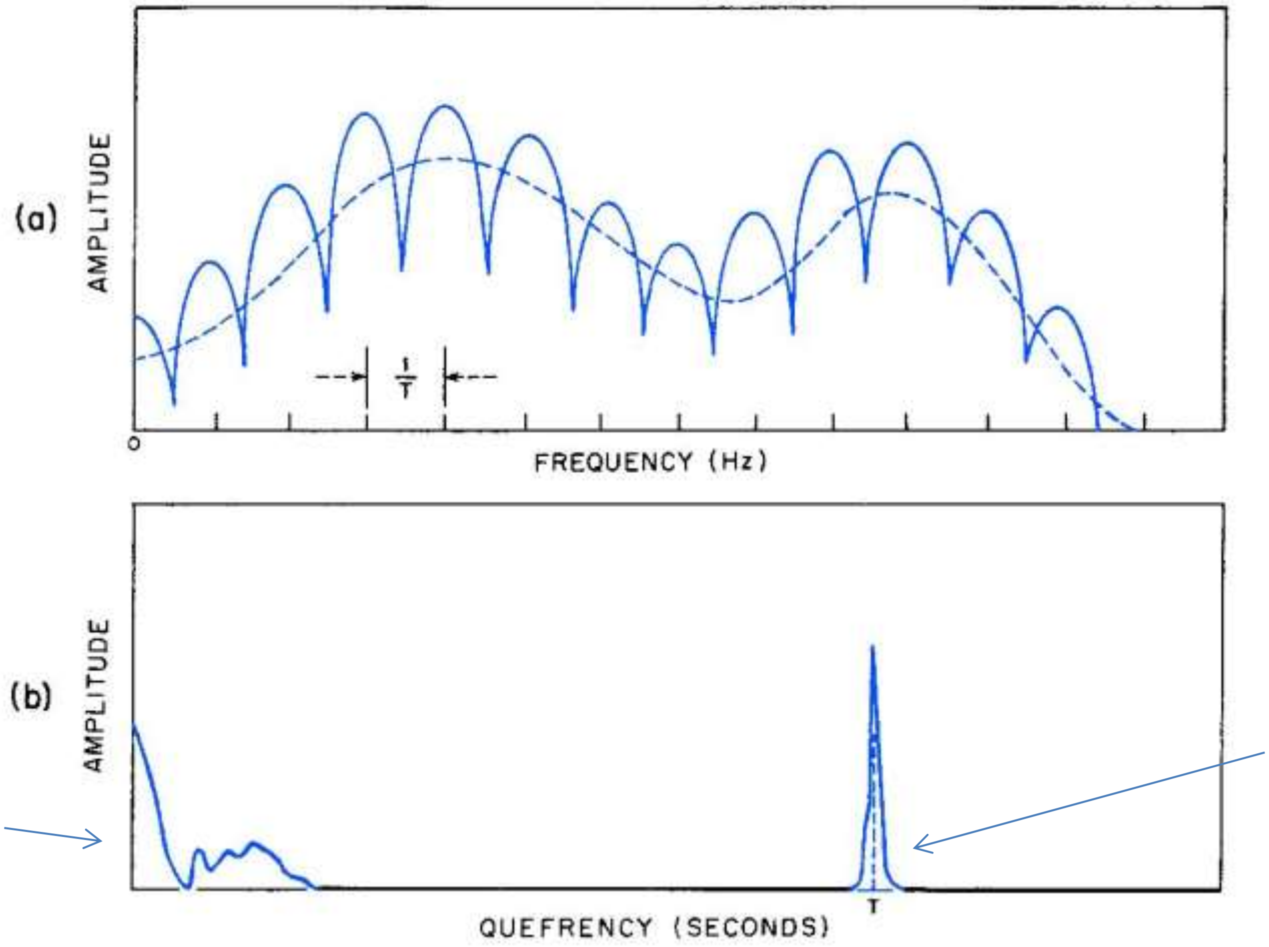
Fourier Transform is performed

Spectral Envelope is analyzed

- Spectral envelope of each short time segment has smoother characteristics.
- Identify characteristic of the envelope.



Cepstrum separates Basic (F0) Wave and the Filter



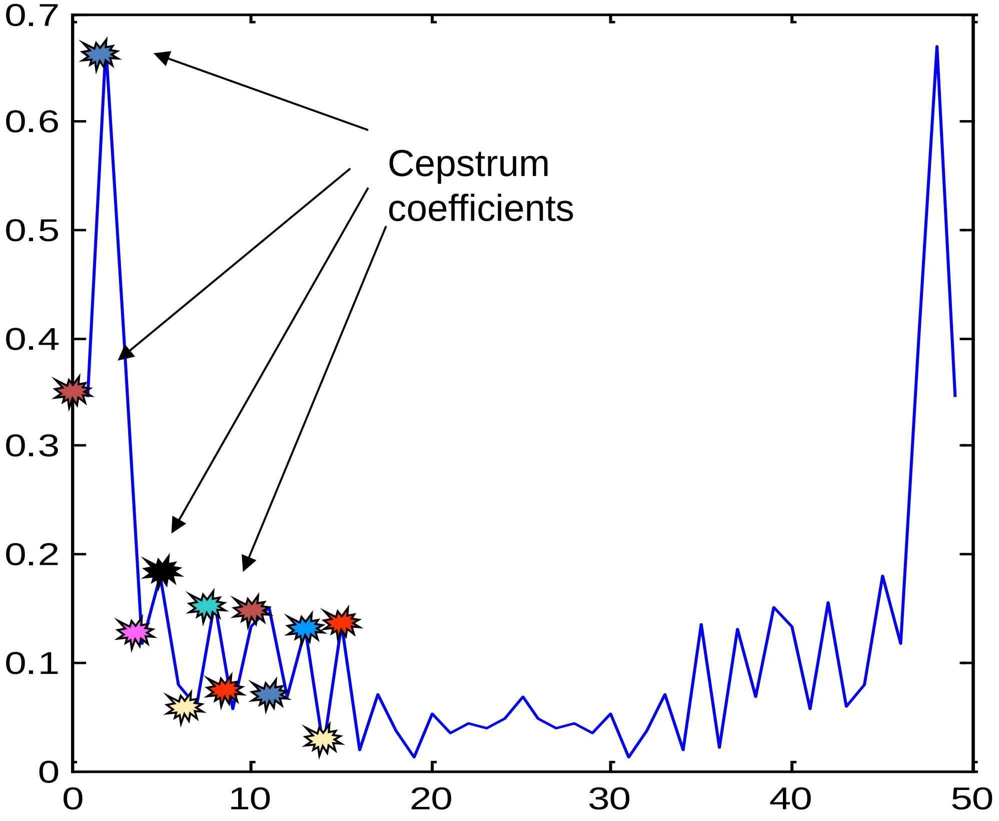
Filtering Portion

Basic frequency Portion

- Log of the product of Fourier transform of the basic wave and the filter separates the two into a sum. That is why Cepstrum can distinguish between the filter (actual phoneme) and the basic wave.

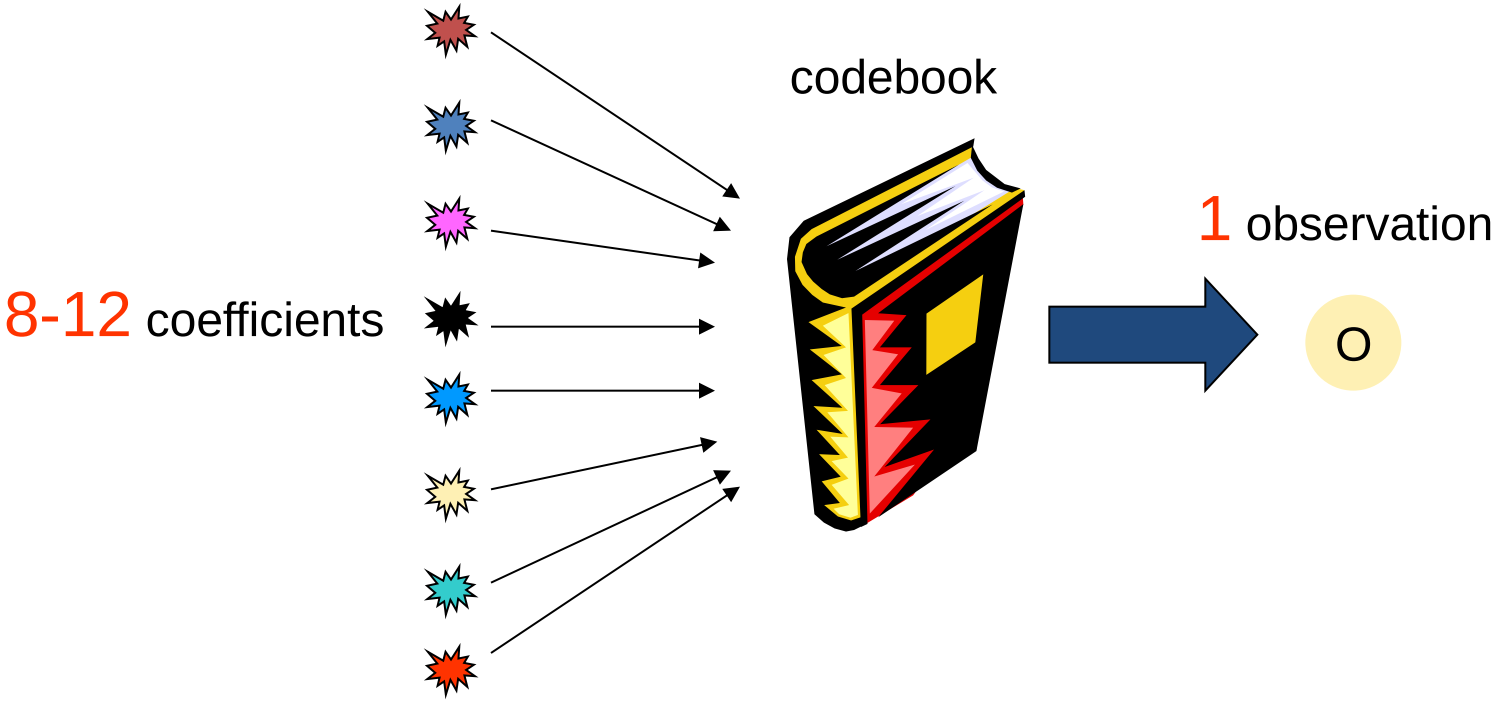
Cepstrum Coefficinets

- The first 8 to 12 characteristic points are selected.
- Those peaks and valleys represent cepstrum.



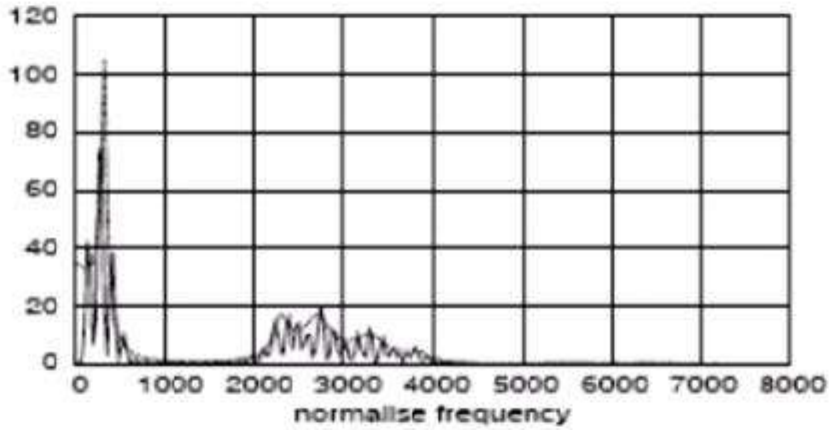
algorithmically: signal → FT → abs() → square → log → FT → abs() → square

Extracted Feature

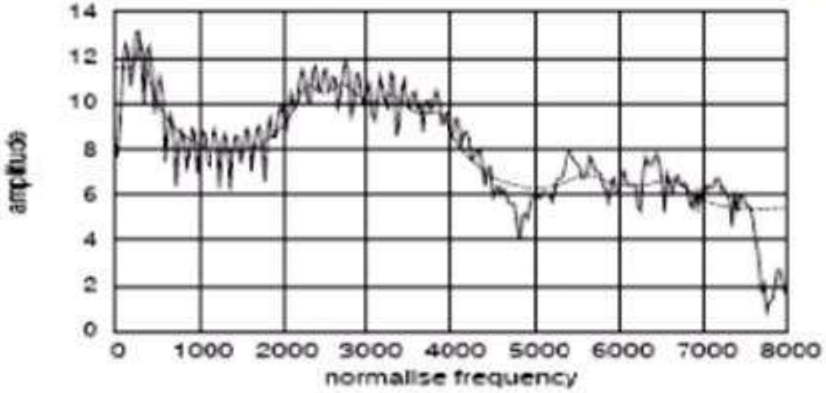


- Cepstrum-s are very efficient representation of speech waveforms.
- Referred to as MFCC, Mel-frequency cepstral coefficients

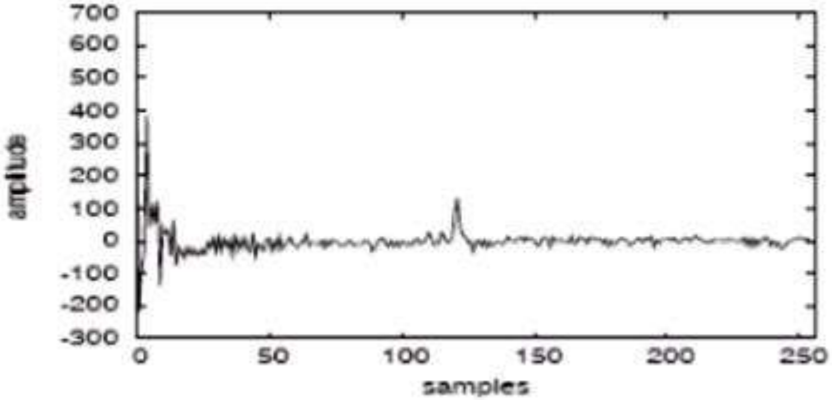
The Cepstrum



Spectrum



Log spectrum



Cepstrum
(samples/sec)

Spectrum of log spectrum

Mel Frequency cepstrum

- The cepstrum requires Fourier analysis
- But we're going from frequency space back to time
- So, we actually apply inverse DFT
- $$y_t[k] = \sum_{m=1}^{M} \log(|Y_t(m)|) \cos(k(m-0.5)\frac{\pi}{M}), \text{ k=0,...,J}$$

- Since the log power spectrum is real and symmetric, inverse DFT reduces to a Discrete Cosine Transform (DCT)

- DCT produces highly **uncorrelated** features

- In general, we use the first 8 to 12 cepstral coefficients (we don't want the later ones which have e.g. the F0 spike)

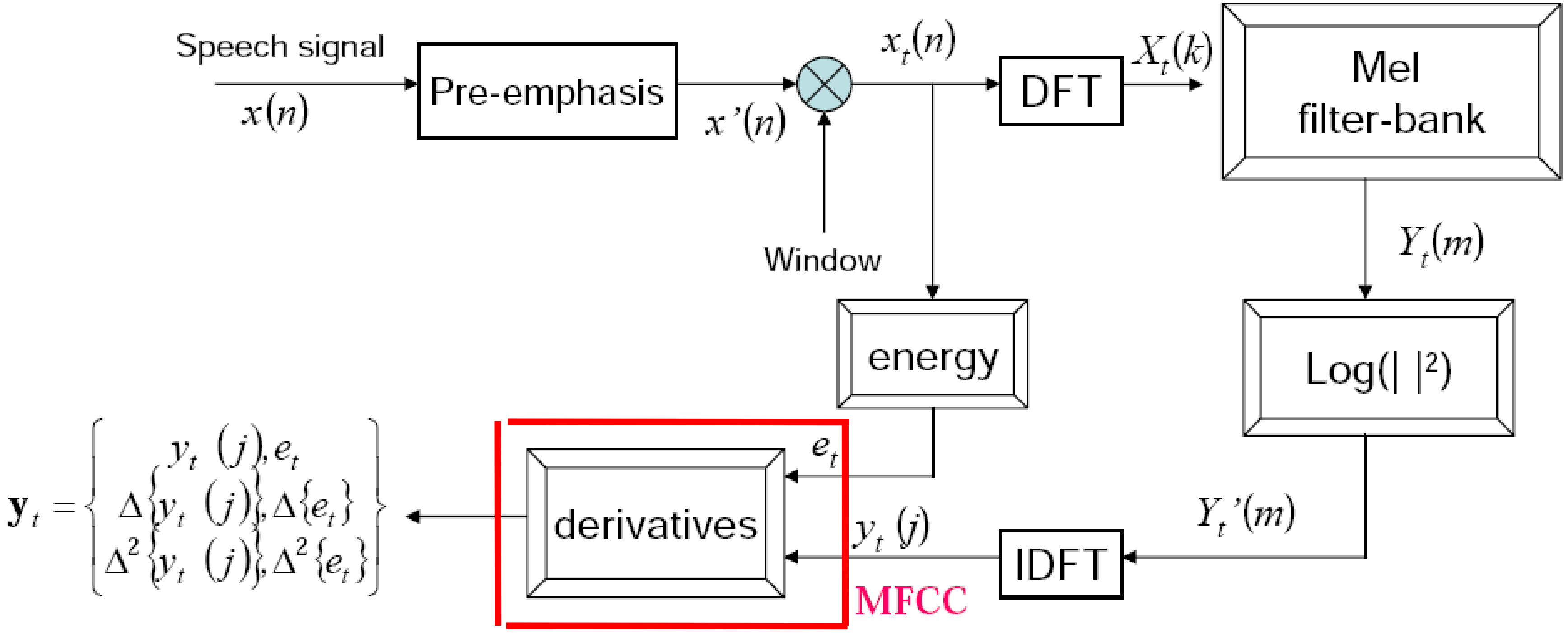
Energy

- The extraction of the cepstrum via the Inverse DFT from the previous section results in 12 cepstral coefficients for each frame. We next add a thirteenth feature: the energy from the frame.
- Energy correlates with phone identity and so is a useful cue for phone Energy detection (vowels and sibilants have more energy than stops, etc). The energy in a frame is the sum over time of the power of the samples in the frame; thus for a signal x in a window from time sample t1 to time sample t2, the energy is:

$$Energy = \sum_{t1}^{t2} x^2[t]$$

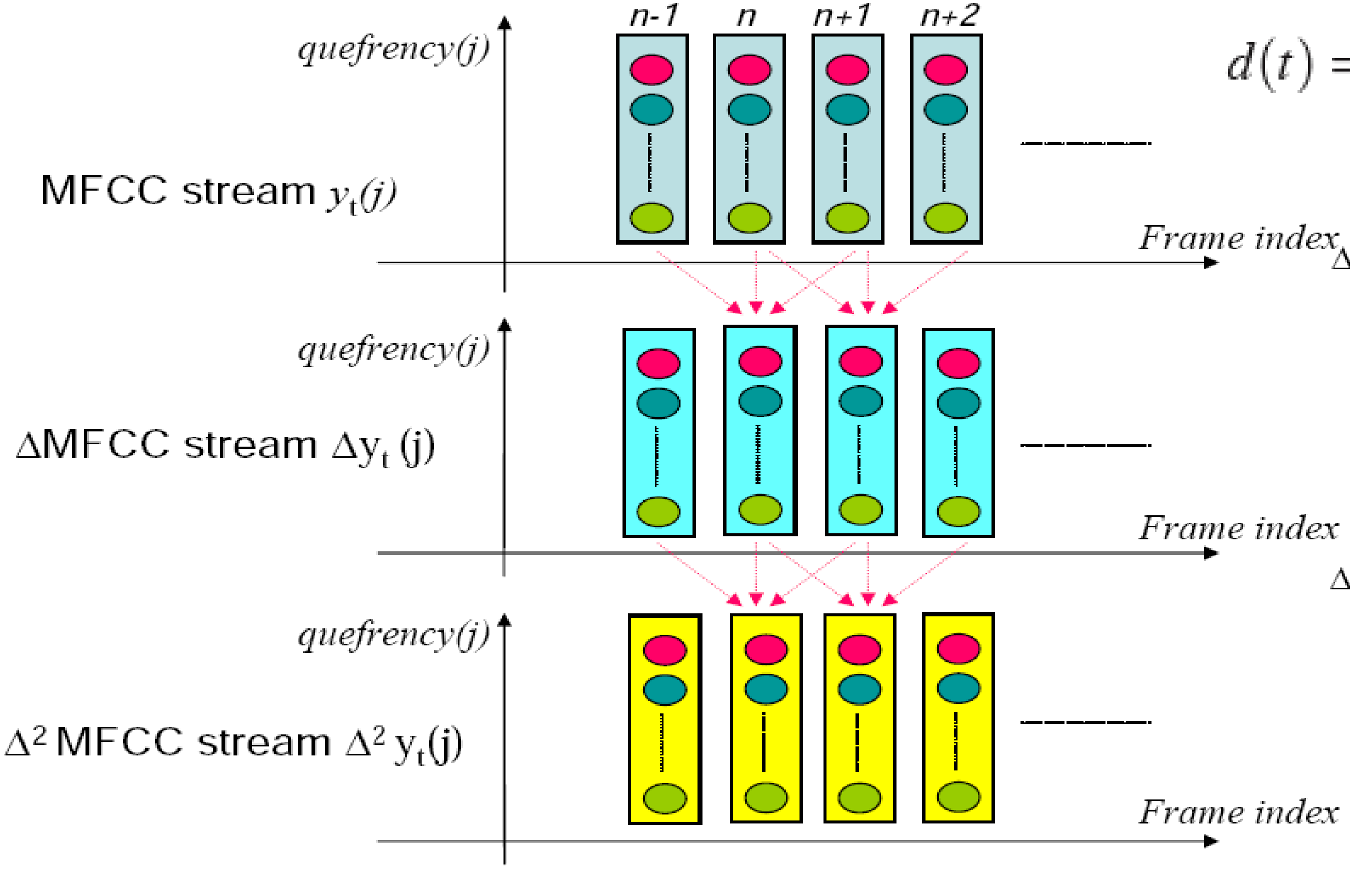
7. MFCC, Deltas and Energy

- Another important fact about the speech signal is that it is not constant from frame to frame. This change, such as the slope of a formant at its transitions, or the nature of the change from a stop closure to stop burst, can provide a useful cue for phonem identity. For this reason we also add features related to the change in cepstral features over time.
- We do this by adding for each of the 13 features (12 cepstral features plus energy feature), a delta or velocity feature, and a double delta or acceleration feature. Each of 13 delta features represents the change between frames in the corresponding cepstral energy feature, while each of the 13 double delta features represents the change between frames in the corresponding delta features.



Delta and double-delta

- Derivative: in order to keep the information of time variation of wave content, we need the first and the second derivative of MFCCs. We call those Deltas.



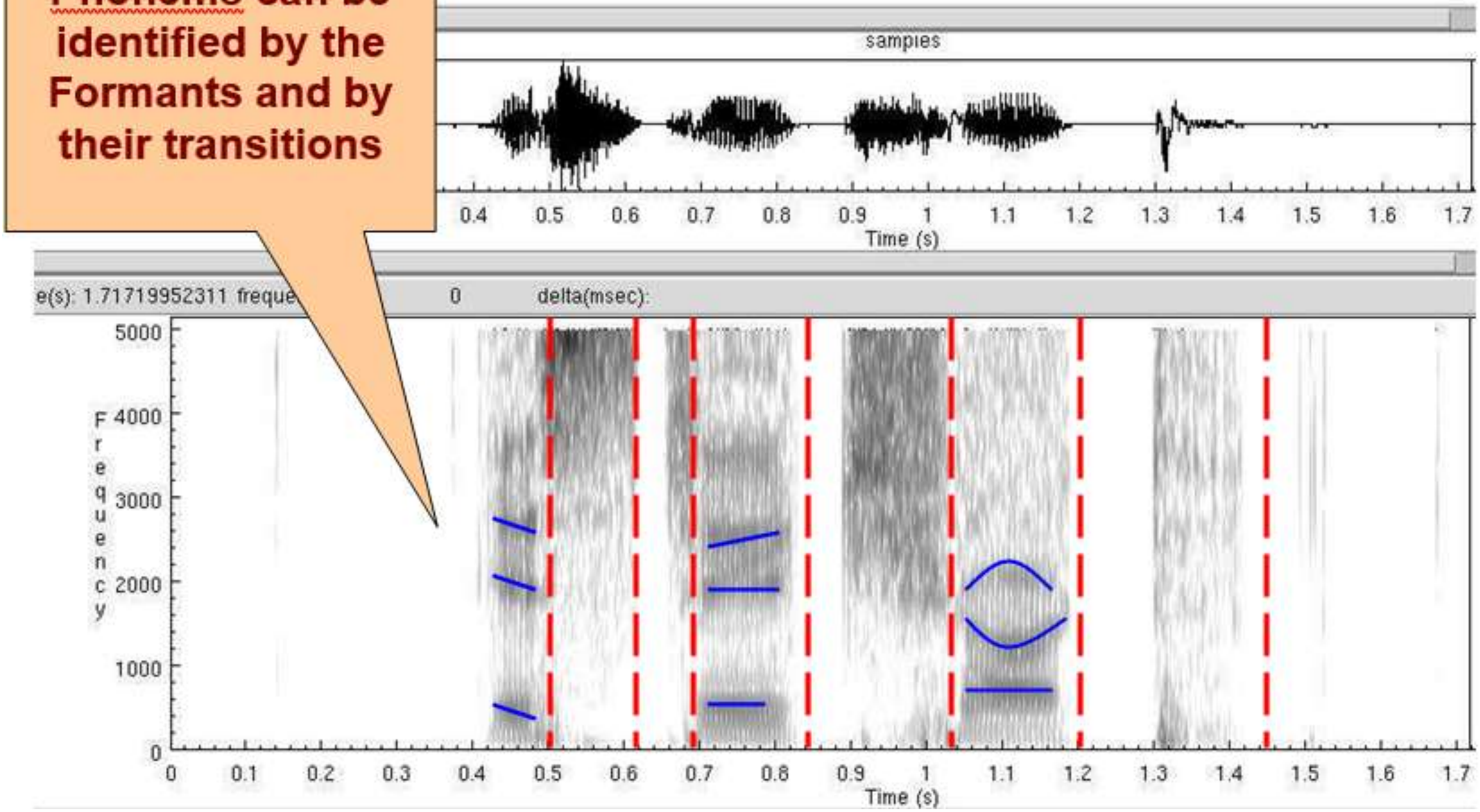
$$d(t) = \frac{c(t+1) - c(t-1)}{2}$$

$$\Delta y_t(j) = \frac{\sum_{m=-p}^{p} m \bullet y_{t-m}(j)}{\sum_{m=-p}^{p} m^2}$$

$$\Delta^2 y_t(j) = \frac{\sum_{m=-p}^{p} m \bullet \Delta y_{t-m}(j)}{\sum_{m=-p}^{p} m^2}$$

Why we bother with Formants

- Note that central frequencies F1, F2, etc. of the main formants are not constant in time but rather could raise, or fall, go up and then down and have other forms.



Phonems can be identified by the Formants and by their transitions

Typical MFCC features

- Window size: 20 ms
- Window shift: 10ms
- Pre-emphasis coefficient: 0.97
- MFCC:
  - 12 MFCC (mel frequency cepstral coefficients)
  - 1 energy feature
  - 12 delta MFCC features
  - 12 double-delta MFCC features
  - 1 delta energy feature
  - 1 double-delta energy feature
- **Total 39-dimensional features per frame**.
- We are generating a point in 39-dimensional vector space for every frame.

- In what we will do next in deep learning, we will use only the 12 MFC coefficients. We could use all 39 for a better result. We are always in hurry.

Inversion

- An MFCC can be approximately inverted to audio in four steps:
  a1) inverse DCT to obtain a mel log-power [dB] spectrogram,
  a2) mapping to power to obtain a mel power spectrogram,
  b1) rescaling to obtain short-time Fourier transform magnitudes, and finally
  b2) phase reconstruction and audio synthesis using Griffin-Lim algorithm
- Each step corresponds to one step in MFCC calculation.

- The **Griffin-Lim Algorithm (GLA)** is a phase reconstruction method based on the redundancy of the short-time Fourier transform. It promotes the consistency of a spectrogram by iterating two projections, where a spectrogram is said to be consistent when its inter-bin dependency owing to the redundancy of STFT is retained. GLA is based only on the consistency and does not take any prior knowledge about the target signal into account.

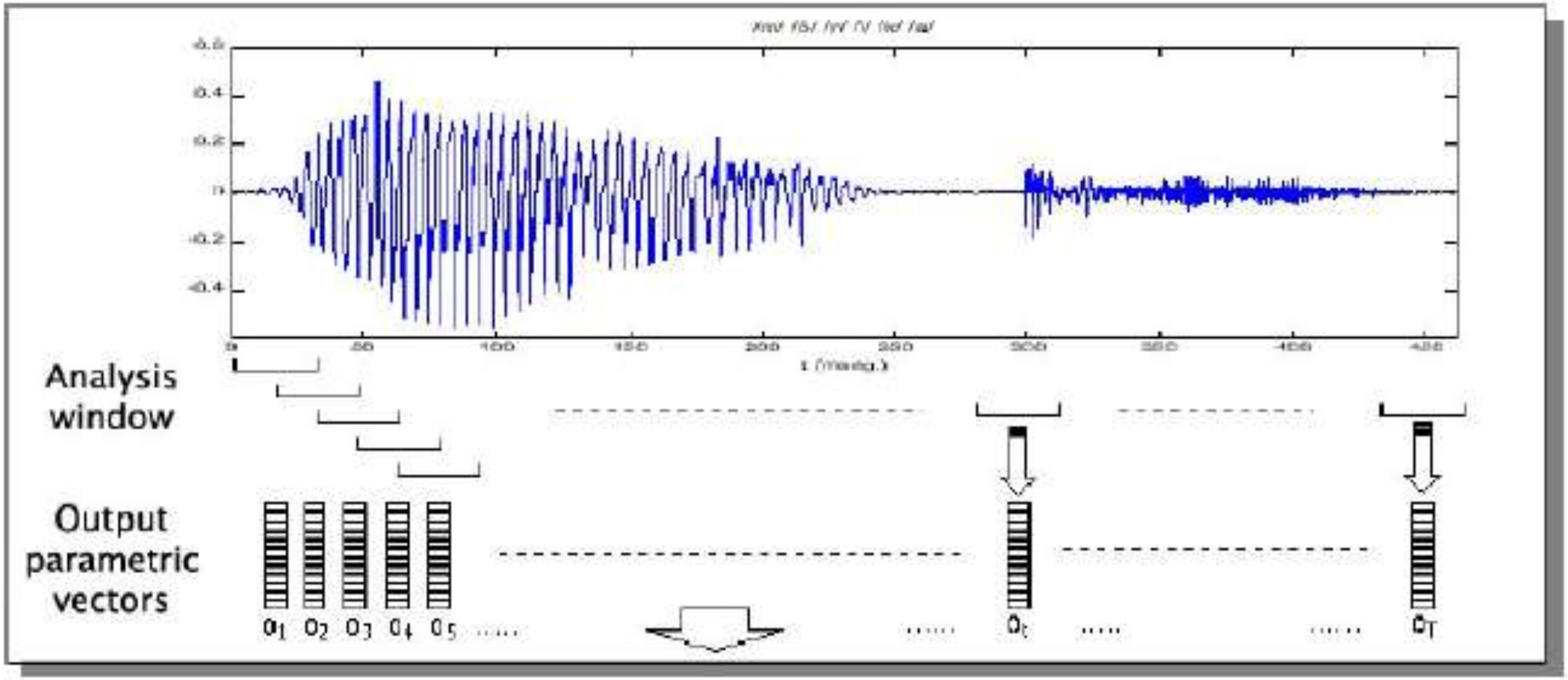
Noise sensitivity

- MFCC values are not very robust in the presence of additive noise, and so it is common to normalise their values in speech recognition systems to lessen the influence of noise. Some researchers propose modifications to the basic MFCC algorithm to improve robustness, such as by raising the log-mel-amplitudes to a suitable power (around 2 or 3) before taking the discrete cosine transform (DCT), which reduces the influence of low-energy components.[10]

Why is MFCC so popular?

- Efficient to compute
- Incorporates a perceptual Mel frequency scale
- Separates the source and filter

Output of Feature (MFCC) Extraction



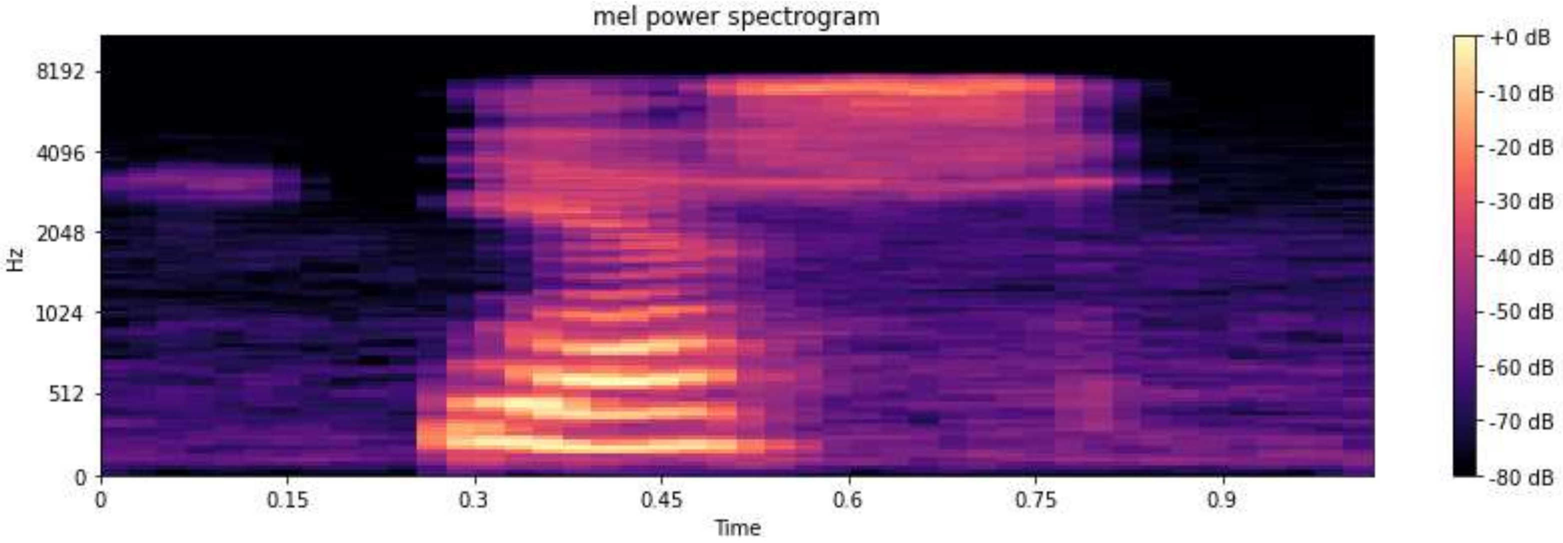
Speech Recognition

Sampled Sound Data Processing

- We now know how to generate an array of numbers which represent the sound wave amplitude at 1/16,000 th of a second intervals.
- We could feed these numbers directly into a neural network. Trying to recognize speech patterns by processing these samples is difficult. Instead, we can make the problem easier by pre-processing the audio data.
- We start by grouping sampled audio into 20-millisecond-long windows. Each such window has 400 samples. This recording is only 1/50th of a second long. Even this short recording is a complex mixture of different frequencies. All together, these different frequencies make up the complex sound of human speech.
- To make this data easier for a neural network to process, we break apart this complex sound wave into its component parts.
- We extract out contribution of different frequencies using MEL filters. We measure the energy is in each of those frequency bands what create a fingerprint of sorts for this audio snippet.
- All of this is done using Fourier transform, i.e. FFT.
- The result is a score of how important each frequency range is, from low pitch (i.e. bass notes) to high pitch. Each number below represents how much energy was in each 50hz band of our 20 millisecond audio clip. But this is a lot easier to see when you draw this as a chart. If we repeat this process on every 20 millisecond chunk of audio, we end up with a spectrogram (each column from left-to-right is one 20ms chunk).

How to represent Sound Recordings, Spectrums?

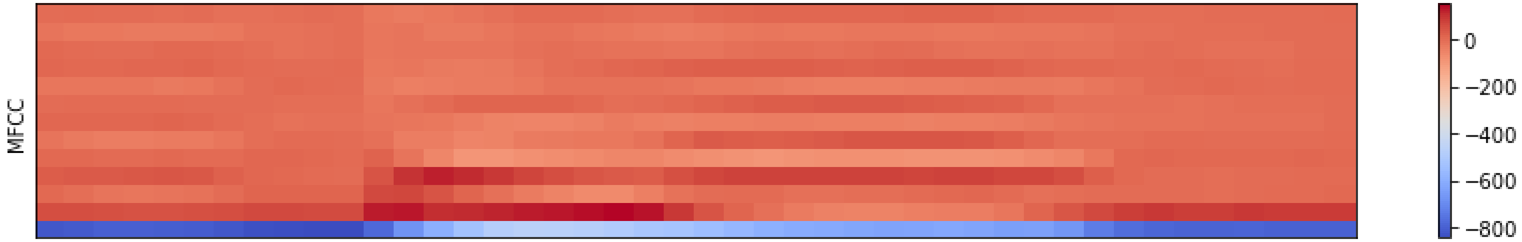
- For every audio file (wav file) we can calculate Spectrums (spectra) and could use those as representation of every word. Spectrum is a collection of FFTs of signal contained in 20 msecs long windows. Those windows slide with a step of 10 msecs.
- As illustrated below for word “yes”, time-frequency spectrum of that word is an image.



- We could tread those images in the same way we treated images of handwritten digits or any other images and use the machinery of the Convolutional Neural Network to create classification or recognition models.

Sound Recordings as Images, MFCC Arrays

- It was found to be more efficient to calculate MFCCs for every consecutive time window of 20 msec. Subsequently, we arrange those MFCCs as an array.
- Those arrays look like pictures. Below is an MFCC array for one utterance of “yes”



- The complexity of this image is lower than the complexity of the spectrogram for the same utterance.
- We could tread those MFCC images in the same way we treated images of handwritten digits or any other images and use the machinery of the Convolutional Neural Network to create classification or recognition models.
- This is what we will demonstrate in the following slides.

Keras Embeddings vs MFCCs

- What we are doing is very similar to Keras or TensorFlow embeddings for written words.
- Embedding is a mapping from discrete objects, such as words, to vectors of real numbers, e.g. https://www.tensorflow.org/programmers_guide/embedding
- We are preparing a fixed size vector for each audio file in the dataset for classification.
- We are dealing with a specific domain, audio embedding. MFCC encoding is one technique we could use to embed the audio files into a smaller dimensional vector space.
- We will compute MFCCs using `Librosa (0.10.1)` Python library.
- Different audio recordings differ in duration. CNNs can not handle sequence data of different length, so we need to prepare a fixed size vectors for all audio files.
- We do that by padding the output arrays with constant value such as "zero“.

- MFCCs are not the only possible embeddings for audio files. You might conduct experiments and discover that MFFCs are not the best embedding for every application. For example, MFCCs could be reversed into an understandable but poor-quality sound. If your application requires a better sounding voice

Objectives & Loading Audio, `*.wav` files

- Data file `data_speech_commands_v0.02.tar.gz` with some 2.3 GB of data is downloaded from https://storage.cloud.google.com/download.tensorflow.org/data/speech_commands_v0.02.tar.gz

- The directory into which we expand downloaded file has 35 collections of spoken words: `bird, dog, cat, four, happy`,..., etc. Words are recorded by many speakers. Each word could be spoken repeatedly, several times, by the same speaker.

- Our task is to classify audio files for seven classes of words:

`bed, bird, cat, five, happy, seven` and `yes`

- We are working with 7 classes to reduce processing time. This processing is not excessively long, and you can work with as many classes of words as you may wish.
- Please download data from Kaggle if you are willing to classify more than these seven words. If you do that you need to change the `num_classes` parameter which controls the number of neurons in the last classifying `Dense` layer and the output at the softmax layer.

- Each folder contains approximately 1700 audio files with utterances of specific word by different people.
- The name of the folder is the label of those audio files. You can play some audio files randomly to get an overall idea about the differences among them.
- We take audio input from one channel only. Files usually contain stereo sound with 2 channels.

Create Archives of MFCC Codes

- You MIGHT want to experiment with different architectures of your Deep Learning model. All those models would most probably, at least initially, rely of MFCC embedding of your audio files.
- Rather than recalculate those MFCC representation for every experiment, you can calculate them once and store in some convenient way.
- Python script `preprocess.py` which is attached to this lecture notes contains a function `save_data_to_array()` which separates audio files into train and test sets, creates numpy arrays made of MFCCs for every word in the dataset and then exports those into numpy archive files `*.npy`.
- Whenever you want to train a new model, you can import, load, those `*.npy` files and use them as direct inputs into your Keras or TensorFlow models.

- For the economy of processing, we will not take all possible 1000 MFFC vectors per 1 second of recording that could be generated with 10 msec slip, but would rather take 20 of those. Also, we will restrict ourselves to the first 11 MFFCs. Those restrictions make our dataset rather modes.
- Labels, i.e. words: `bird, cat, five`, etc. are encoded using one-hot encoding with the help of `tensorflow.keras.utils.to_categorical()` function.

Loading train set and test set

- The routine bellow looks into the current directory and select for loading those `*.npy` files which have corresponding directory word in the `./data` directory.

```
X_train, X_test, y_train, y_test = get_train_test()
# # Feature dimension
feature_dim_1 = 20; channel = 1
epochs = 1000; batch_size = 100
verbose = 1
num_classes = 6   # number of words used for training.
# Reshaping to perform 2D convolution
X_train = X_train.reshape(X_train.shape[0], feature_dim_1, feature_dim_2,
channel)
X_test = X_test.reshape(X_test.shape[0],feature_dim_1,feature_dim_2, channel)
y_train_hot = to_categorical(y_train)
y_test_hot = to_categorical(y_test)
print(X_train.shape, X_test.shape)
print(y_train.shape,y_test.shape)
(12154, 20, 11, 1) (8103, 20, 11, 1)
(12154,) (8103,)
```

- Each word is represented by 20 samples (feature_dim_1 = 20) along the time axis and 11 MFCC (the first 11) at each time point.

- Each word is represented as an image (matrix, array) of dimensions 50x11.

Definition of the Model

- We will build a sequential CNN using Conv2D layers with RLU (Rectified Linear Units) activation.

```
def get_model():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(2, 2), activation='relu',
input_shape=(feature_dim_1, feature_dim_2, channel)))
    model.add(Conv2D(48, kernel_size=(2, 2), activation='relu'))
    model.add(Conv2D(120, kernel_size=(2, 2), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.4))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])
    return model
```

Train the Model

- To train the model we invoke `fit()` method

```
epochs = 1000; batch_size = 100
model = get_model()
history = model.fit(X_train, y_train_hot, batch_size=batch_size,
epochs=epochs, verbose=verbose, validation_data=(X_test, y_test_hot))
```

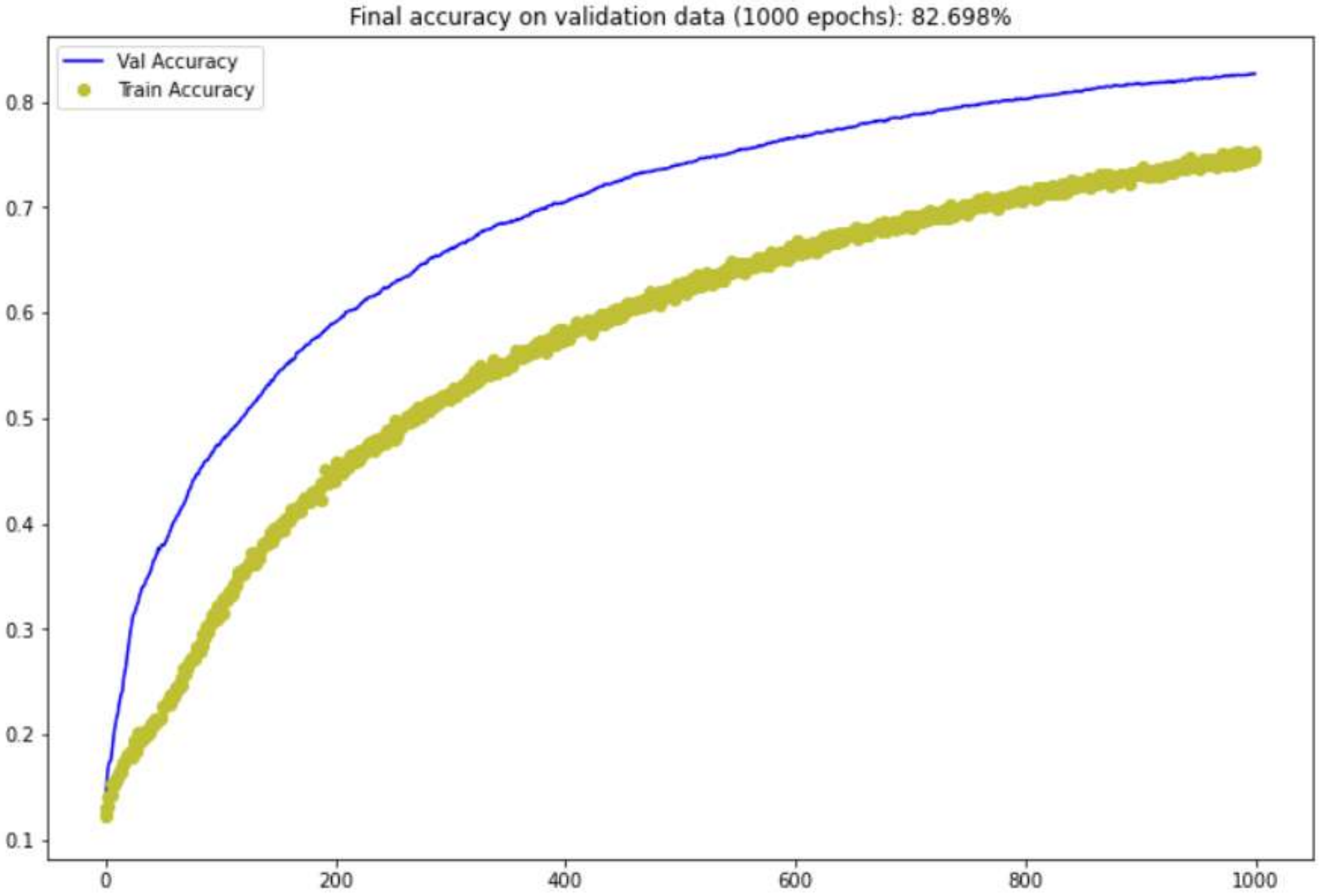
On my modest Nvidia card GTX 950, each epoch takes 3 to 4 seconds. The whole training takes around one hour. In a Google Colab, the training will be several times faster.

```
Epoch 1/1000
122/122 [==============================] - 4s 30ms/step - loss: 10.4575 - accuracy:
0.1223 - val_loss: 6.1507 - val_accuracy: 0.1172
Epoch 2/1000
122/122 [==============================] - 3s 25ms/step - loss: 8.3934 - accuracy:
0.1294 - val_loss: 4.3414 - val_accuracy: 0.1477
Epoch 3/1000

........
Epoch 1000/1000
122/122 [==============================] - 3s 25ms/step - loss: 0.7375 - accuracy:
0.7470 - val_loss: 0.5531 - val_accuracy: 0.8270
```

Validation Accuracy

- After 1000 epochs, the accuracy of the model reaches 82% but is far from saturation. There is no visible overfitting.



Testing the model with single words

- To examine the model with single spoken words, we will take an audio recording of a word that did not participate in training. We will map the audio signal into MFFCs and reduce those to 20 vectors of length 11.
- Then we will ask the model which class gave the largest output for such reshaped sample.
- The function to perform those operations is below.

```
# Predicts one sample
def predict(filepath, model):
  sample = wav2mfcc(filepath)
  sample_reshaped = sample.reshape(1,feature_dim_1,feature_dim_2, channel)
  return get_labels()[0][
      np.argmax(model.predict(sample_reshaped))
  ]
```

- Model will predict (classify) some audio files correctly and some it will not. If you would carry those experiments for a large number of samples our model will give correct classification some 72+% of the time.
- The following slide gives one positive and one negative example.

Examples of Word Recognition

- An accurate prediction:

```
train_audio_path = './data/'
filename = 'cat/6ac35824_nohash_0.wav'
new_sample_rate = 16000
samples,sample_rate = librosa.load(str(train_audio_path) + filename)
ipd.Audio(samples, rate=sample_rate)

print(predict('./data/cat/6ac35824_nohash_0.wav', model=model))
cat
```

- An inaccurate prediction:

```
train_audio_path = './data/'
#filename = 'yes/0a7c2a8d_nohash_0.wav'
filename = 'seven/b1114e4f_nohash_0.wav'
new_sample_rate = 16000
samples, sample_rate = librosa.load(str(train_audio_path) + filename)
ipd.Audio(samples, rate=sample_rate)

print(predict('./data/seven/b1114e4f_nohash_0.wav', model=model))
five
```

Script `preprocess.py`

- Script `preprocess.py` contains several very useful functions. Please examine the script very carefully. In the following slides we are listing those functions. Note that `DATA_PATH` is hard coded

```
import librosa
import os
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
import numpy as np
from tqdm import tqdm
DATA_PATH = "./data/"

# Input: Folder Path
# Output: Tuple (Label, Indices of the labels, one-hot encoded labels)
def get_labels(path=DATA_PATH):
    labels = os.listdir(path)
    label_indices = np.arange(0, len(labels))
    return labels, label_indices, to_categorical(label_indices)

# Handy function to convert wav2mfcc
def wav2mfcc(file_path, max_len=11):
    wave, sr = librosa.load(file_path, mono=True, sr=None)
    wave = wave[::3]
    mfcc = librosa.feature.mfcc(y=wave, sr=16000)

    # If maximum length exceeds mfcc lengths then pad the remaining ones
    if (max_len > mfcc.shape[1]):
        pad_width = max_len - mfcc.shape[1]
        mfcc = np.pad(mfcc, pad_width=((0, 0), (0, pad_width)), mode='constant')

    # Else cutoff the remaining parts
    else:
        mfcc = mfcc[:, :max_len]

    return mfcc
```

Script preprocess.py

```
def save_data_to_array(path=DATA_PATH, max_len=11):
    labels, _, _ = get_labels(path)

    for label in labels:
        # Init mfcc vectors
        mfcc_vectors = []

        wavfiles = [path + label + '/' + wavfile for wavfile in os.listdir(path + '/' + label)]
        for wavfile in tqdm(wavfiles, "Saving vectors of label - '{}'".format(label)):
            mfcc = wav2mfcc(wavfile, max_len=max_len)
            mfcc_vectors.append(mfcc)
        np.save(label + '.npy', mfcc_vectors)


def get_train_test(split_ratio=0.6, random_state=42):
    # Get available labels
    labels, indices, _ = get_labels(DATA_PATH)

    # Getting first arrays
    X = np.load(labels[0] + '.npy')
    y = np.zeros(X.shape[0])

    # Append all of the dataset into one single array, same goes for y
    for i, label in enumerate(labels[1:]):
        x = np.load(label + '.npy')
        X = np.vstack((X, x))
        y = np.append(y, np.full(x.shape[0], fill_value= (i + 1)))

    assert X.shape[0] == len(y)

    return train_test_split(X, y, test_size= (1 - split_ratio), random_state=random_state,
shuffle=True)
```

Script preprocess.py

```
def prepare_dataset(path=DATA_PATH):
    labels, _, _ = get_labels(path)
    data = {}
    for label in labels:
        data[label] = {}
data[label]['path'] =[path  + label + '/' + wavfile for wavfile in os.listdir(path + '/' +
label)]

        vectors = []
        for wavfile in data[label]['path']:
            wave, sr = librosa.load(wavfile, mono=True, sr=None)
            # Downsampling
            wave = wave[::3]
            mfcc = librosa.feature.mfcc(wave, sr=16000)
            vectors.append(mfcc)

        data[label]['mfcc'] = vectors

    return data

def load_dataset(path=DATA_PATH):
    data = prepare_dataset(path)
    dataset = []

    for key in data:
        for mfcc in data[key]['mfcc']:
            dataset.append((key, mfcc))

    return dataset[:100]


# print(prepare_dataset(DATA_PATH))
```