TSIA-206

Automatic Speech Recognition (ASR)

Geoffroy Peeters

contact: geoffroy.peeters@telecom-paris.fr
Télécom-Paris, IP-Paris, France

version: 2024-05-03-15-07

Various categories of audio content
Applications

	Speech	Music	Envirronmental Sounds
Description	**Speech to text,** aka Auto. Speech Reco. (ASR) Speaker recognition Speaker diarization	Audio ID (Shazam) Auto-tagging Recommendation Content-description (pitch, chord, tempo) Lyrics alignment/ recognition	Acoustic scene classificaton Sound event localization, detection
Transformation	Speech separation, enhancement Speech coding Speech transformation	Singing separation (Karaoke) Unmixing Style transfer	
Generation	**Text To Speech (TTS)**	Sound generation Music generation (OpenAI)	Sound texture synthesis
		MIR http://ismir.net	DCASE http://dcase.community

Speech
Applications

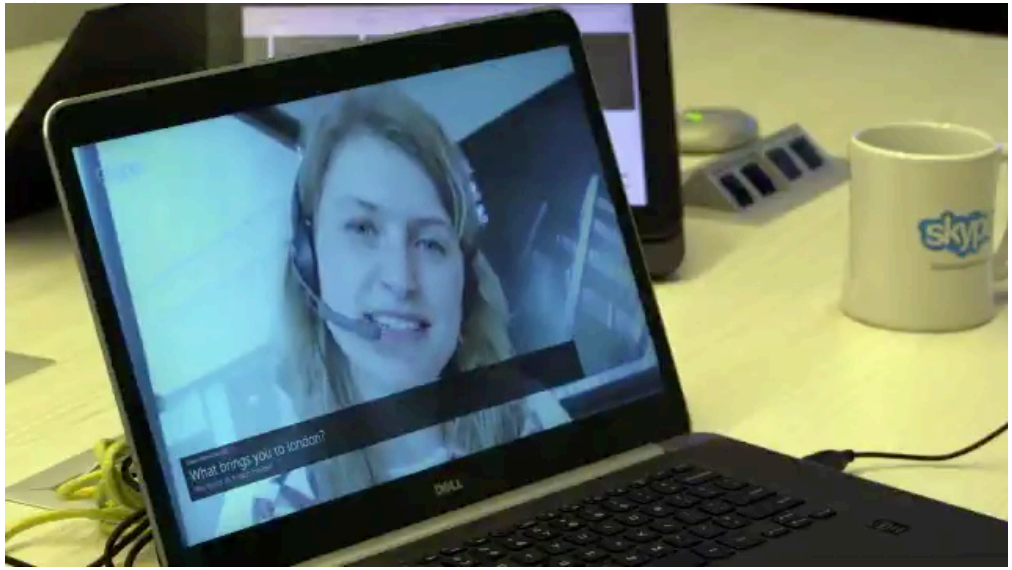
Automatic Speech Recognition (ASR) → Translation → Speech-To-Text (TTS)

Microsoft :

- https://www.youtube.com/watch?time_continue=544&v=Nu-nlQqFCKg

Skype :

- https://www.youtube.com/watch?v=JrlTzS7Fk6o



ASR: History

History of ASR

The first machines that "recognised" speech

https://www.youtube.com/watch?v=AdUi_St-BdM

- **1922**: Radio Rex
  - a toy!
  - pronounce vowel [eh] in “Rex” (First formant ≈ 500 Hz)
  - ⇒ spring released by 500 Hz acoustic energy
  - rex is pushed out his kennel

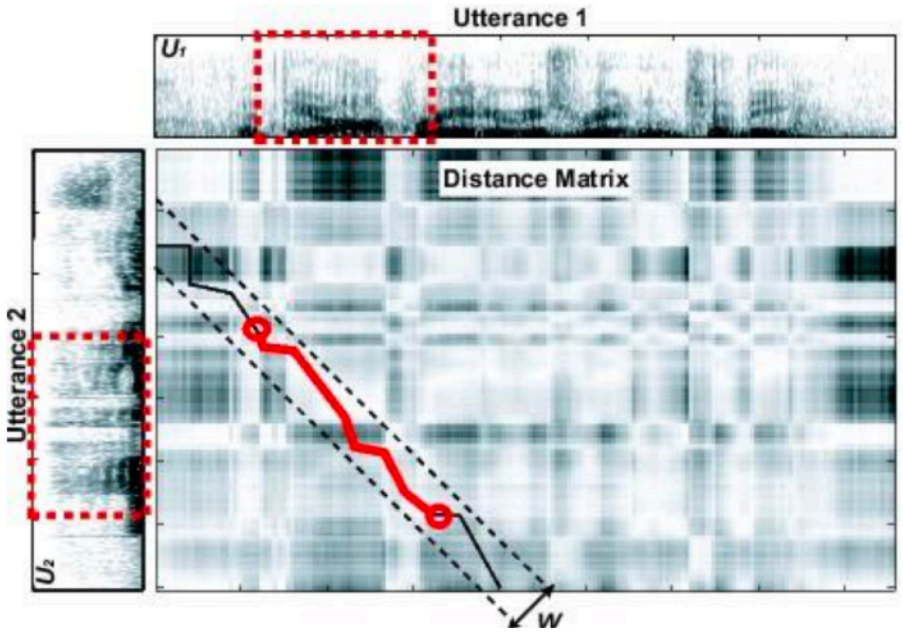
1952 Bell Labs Audrey. Not shown is six foot high rack of supporting electronics.

- **1952**: First system “Audrey” by Bell Labs
  - single-speaker digit recognition, formant localisation in the power spectrum
- **1961**: IBM Shoebox
  - recognise isolated words (including 0, 1, ... 9) LINK

https://computerhistory.org/blog/audrey-alexa-hal-and-more/

Dynamic Time Warping

- **1960s:**
  - First dynamic Time Warping (DTW) system to handle time variability
  - Distance measure for spectral variability



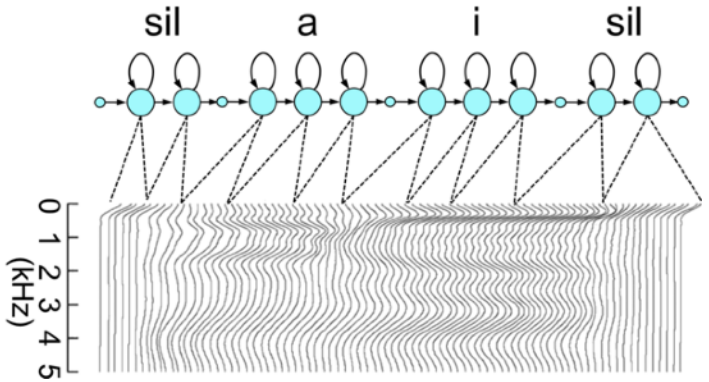
source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

https://computerhistory.org/blog/audrey-alexa-hal-and-more/

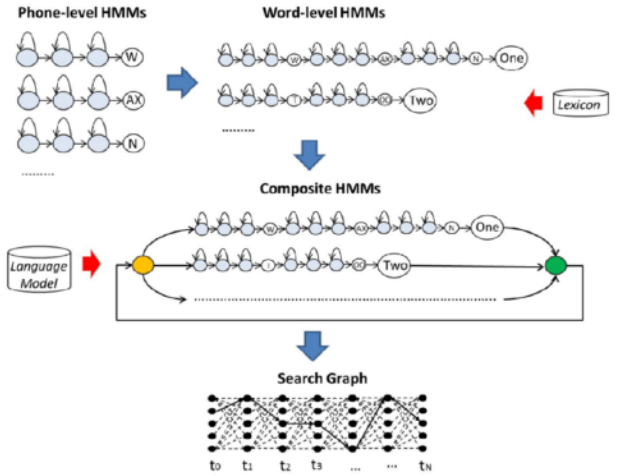
History of ASR

Hidden Markov Models / GMMs

- **1970s:**
  - Hidden Markov Models
  - Statistical models of spectral variations for discrete speech
- **1976**: CMU Harpy;
  - recognise 1000 **connected** words
- **1990s**
  - **Large vocabulary continuous dictation**
- **Mid 1980s**
  - HMMs become the dominant technique for all ASR
- **1986–1995**:
  - Speech recognition with **neural networks**
- **1995–2009**:
  - Superseded by **GMMs**
- **2000s**
  - Discriminative training (minimise word/phone error rate)



source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"



The Bakers' use of Hidden Markov Models.

https://computerhistory.org/blog/audrey-alexa-hal-and-more/

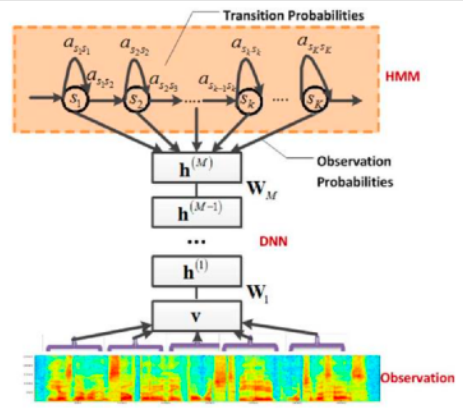
History of ASR

Neural Networks/ Deep Learning

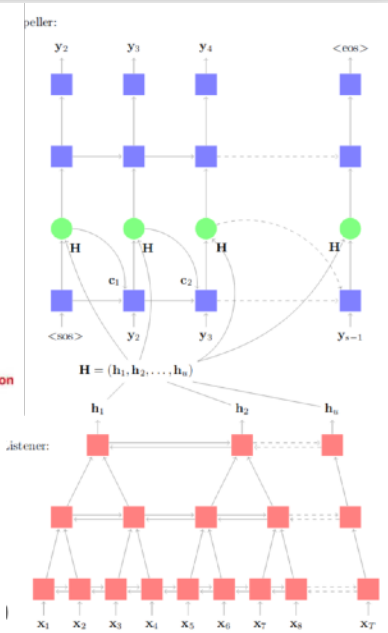
- **From 2009**: Deep feedforward (non-recurrent) networks
  - For acoustic modeling (Hinton, 2009)
  - Good results on TIMIT (Mohamed et al., 2009)
  - Results on large vocabulary systems 2010 (Dahl et al., 2011)
  - Google launches DNN ASR product 2011
  - Dominant paradigm for ASR 2012 (Hinton et al., 2012)

- **Since 2016:** End-to-End systems
  - Attention-based end-to-end approaches
    - ex: "Listen, Attend and Spell" (LAS)
  - CTC Connectionist Temporal Classification

- **Since 2020**: Self-Supervised Learning
  - 2020: wav2vec
  - 2021: wav2vec Unsupervised (wav2vec-U)



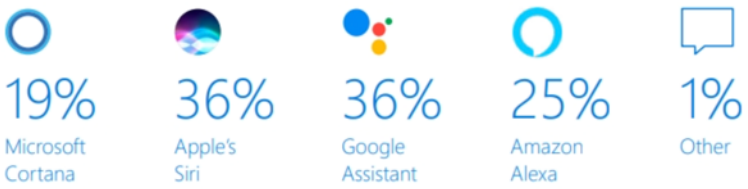
source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"



https://computerhistory.org/blog/audrey-alexa-hal-and-more/

Application-specific constraints

- From speaker-**dependent** and **small** vocabularies
  - Ex : Control/command applications (e.g. mobile phone...)
- To speaker-**independent** and **large** vocabularies
  - Ex : Siri, audio indexing



OpenAI Whisper

https://openai.com/research/whisper

ASR datasets
from **read** speech to **spontaneous** speech

- Even **bigger** datasets
  - DARPA-TIMIT Acoustic-Phonetic Continuous Speech Corpus LINK
    - 1988, Read speech - with hand-marked phone boundaries – 630 speakers × 10 utterances
  - Wall Street Journal (WSJ)
    - 1986 - Read speech. WSJ0 1991, 30k vocab
  - Broadcast News (BN)
    - 1996 104 hours
  - Switchboard (SWB) LINK
    - 1992. 2000 hours spontaneous telephone speech - 500 speakers
  - YouTube
    - 125,000 hours aligned captions (Soltau et al., 2016)
  - LibriSpeech (SLR12) LINK
    - 2015, Large-scale (1000 hours) corpus of read English speech
  - Mozilla Common Voice LINK
    - 2019 - ...
- To integrate **noise** environment
  - Google voice search
    - anonymised live traffic 3M utterances 2000 hours, hand-transcribed 4M vocabulary. Constantly refreshed, synthetic reverberation + additive noise
  - DeepSpeech
    - 5000h read (Lombard) speech + SWB with additive noise.

ASR software

-

Application name	Description	Open-source	License	Operating system	Programming language	Supported language, note	Offline or online
CMU Sphinx	HMM	Yes	BSD style	Cross-platform	Java	English, German, French, Mandarin, Russian	Offline
HTK	HMM neural net	No	HTK specific	Cross-platform	C	English; version 3.5 released December 2015	
Julius	HMM trigrams	Yes	BSD style, non-commercial	Cross-platform	C	Japanese, English; [2]	Offline
Kaldi	Neural net	Yes	Apache	Cross-platform	C++	English	
RWTH ASR	RWTH Aachen University	No	RWTH ASR, non-commercial use only	Linux, macOS	C++	English	
Whisper (speech recognition system)	Encoder/decoder transformer	Yes	MIT license	Cross-platform	Python (programming language)	Multilingual	Online (through API) and Offline

https://en.wikipedia.org/wiki/List_of_speech_recognition_software

- Also SpeechBrain https://speechbrain.github.io/

ASR performances
examples depending on the dataset

English Tasks	WER%
LibriSpeech audiobooks 960hour clean	1.4
LibriSpeech audiobooks 960hour other	2.6
Switchboard telephone conversations between strangers	5.8
CALLHOME telephone conversations between family	11.0
Sociolinguistic interviews, CORAAL (AAL)	27.0
CHiMe5 dinner parties with body-worn microphones	47.9
CHiMe5 dinner parties with distant microphones	81.3
Chinese (Mandarin) Tasks	**CER%**
AISHELL-1 Mandarin read speech corpus	6.7
HKUST Mandarin Chinese telephone conversations	23.5

Figure 26.1 Rough Word Error Rates (WER = % of words misrecognized) reported around 2020 for ASR on various American English recognition tasks, and character error rates (CER) for two Chinese recognition tasks.

Word Error Rate (WER)

Read speech

+/- spontaneous conversations

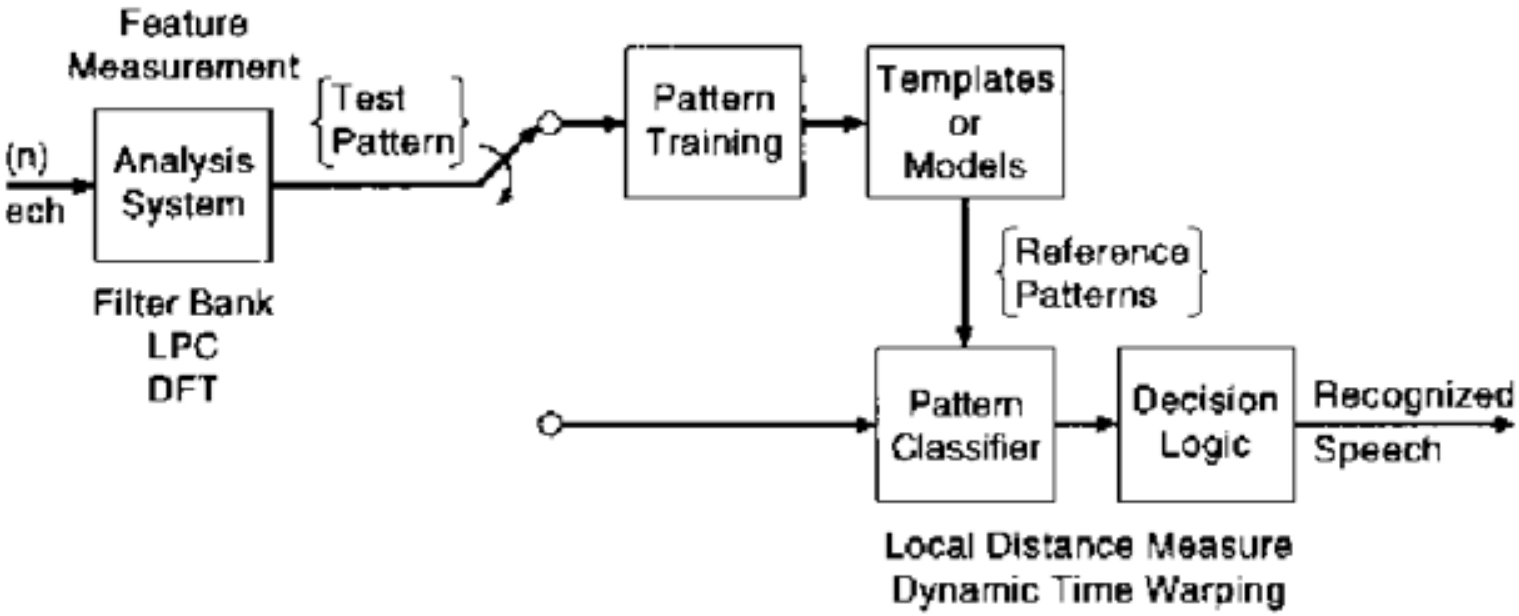
Group conversations

Character Error Rate (CER)

much lower for read Mandarin speech than for natural conversation.

ASR: recognition of isolated words

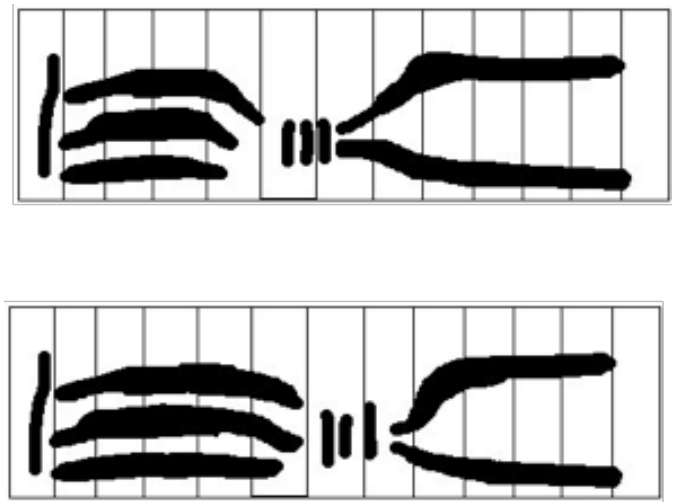
Statistical approaches : basic system



From L. Rabiner & B. Juang. Fundamentals of Speech recognition. Signal processing series. Prentice Hall,1993

Dynamic Time Warping
Temporal distorsions

- The same speaker cannot pronounce the same vocal sequence several times with exactly the same rhythm and the same total duration
  - ⇒ The time scales of two occurrences of the same word do not coincide.
  - ⇒ The vector sequences resulting from parameterisation cannot be compared with each other.



Two different pronunciations of the same word

Dynamic Time Warping Problem

- How to calculate a **measure of similarity between two sequences** that takes into account time distortions?
  - ⇒ Calculation of the distance/similarity matrix between two sequences
  - Use the DTW (Dynamic Time Warping) algorithm to calculate the **optimal path** for the similarity calculation

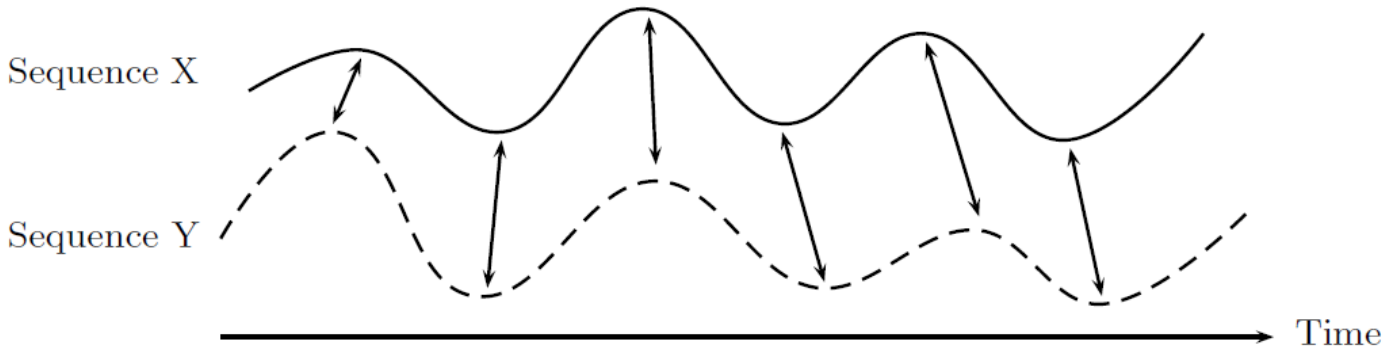
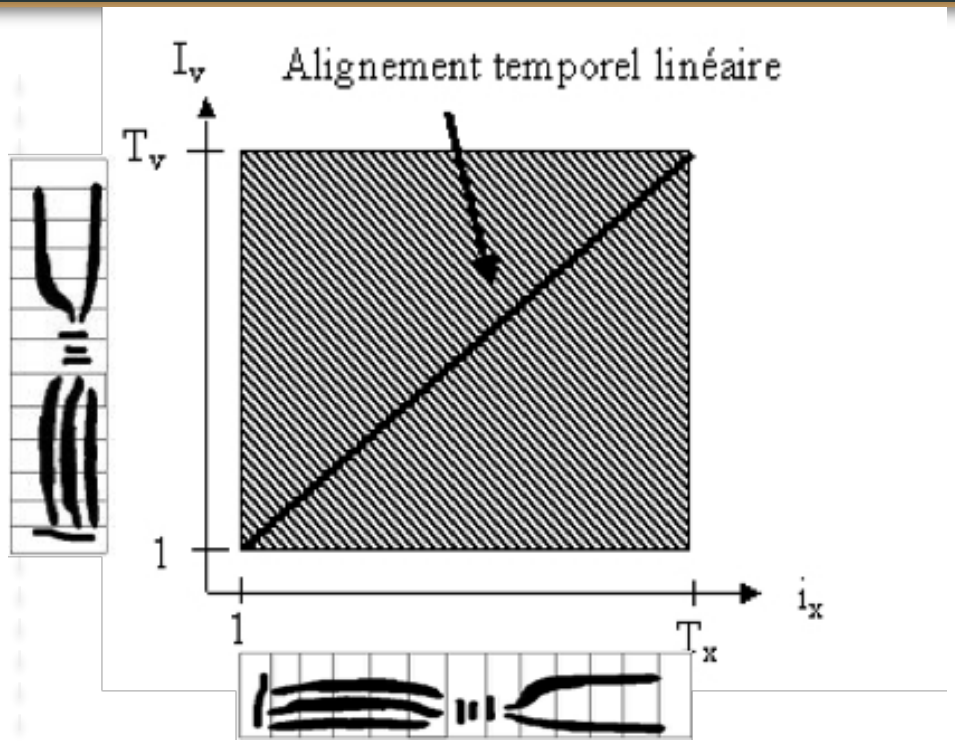


Fig. 4.1. Time alignment of two time-dependent sequences. Aligned points are indicated by the *arrows*

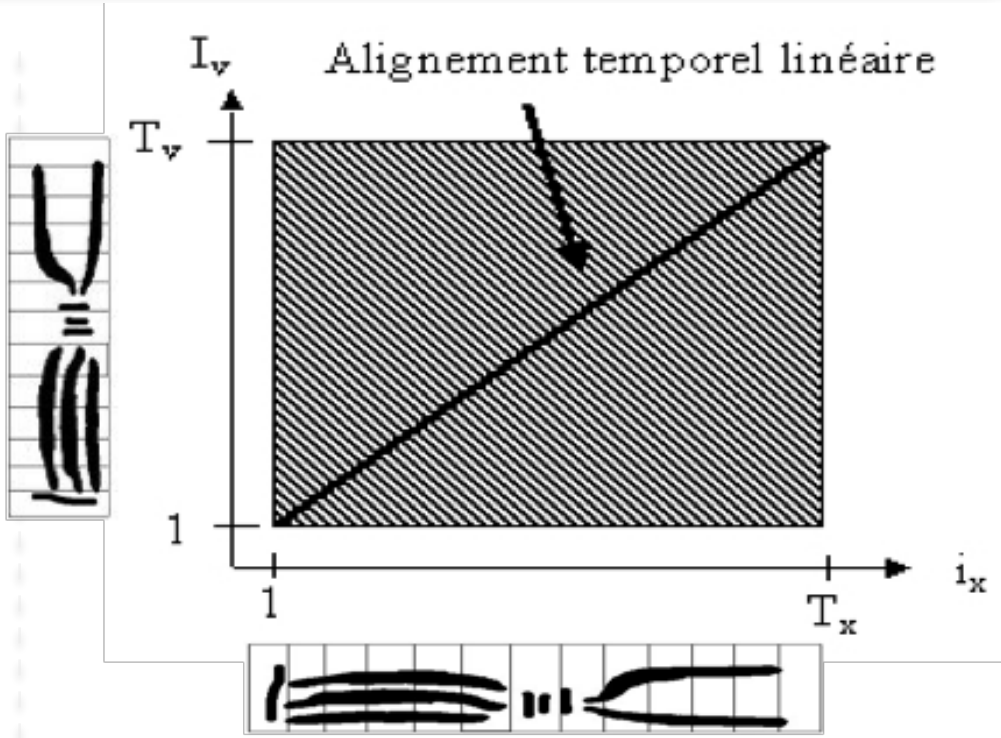
Dynamic Time Warping
Similarity matrix between sequences

- **Case 1 (unrealistic)**
  - both sequences are pronounced with exactly the same rhythm and duration ($T_y = T_x$)
  - If we have exactly the same sequences, the measure of similarity is maximum by summing the distances on the **diagonal**
  - X-axis and Y-axis = time
  - Figure = Matrix of similarities between 2 speech segments



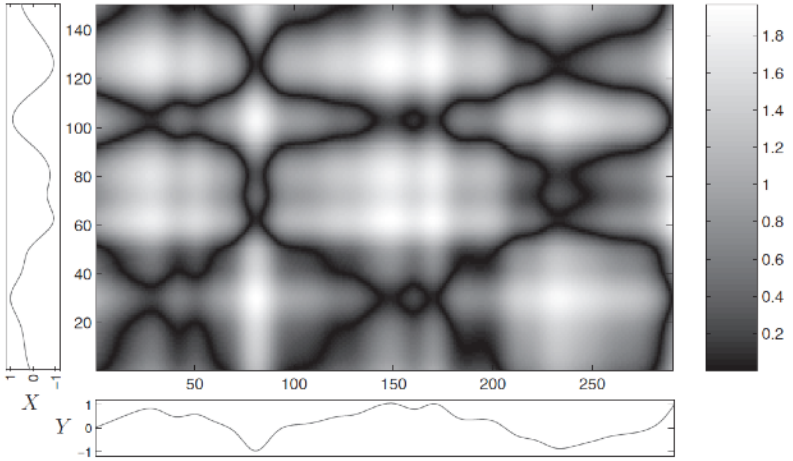
Dynamic Time Warping
Compute the similarity matrix between sequences

- **Case 2 (a little more realistic)**
  - the deformation is linear: a word and each of its segments is pronounced more quickly and in the same proportion (linear time alignment)



Dynamic Time Warping
Compute the similarity matrix between sequences

- **Cases 3 (much more realistic)**
  - the deformation between sequences is dynamic



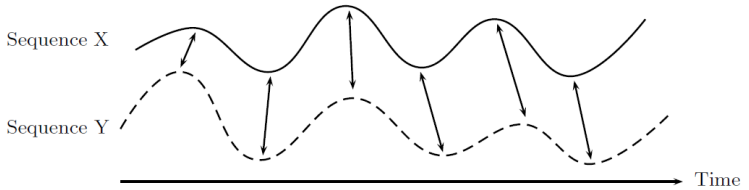


Fig. 4.1. Time alignment of two time-dependent sequences. Aligned points are indicated by the *arrows*

Dynamic Time Warping
Definition : Warping path/chemin de déformation

- A **warping path** $\pi_l = (n_l, m_l)$ defines an alignment between two sequences
  - Sequence $X = (x_1, \ldots, x_N)$
  - Sequence $Y = (y_1, \ldots, y_M)$
  - $\pi_l = (n_l, m_l)$ assigns the elements x_{n_l} of X to the elements y_{m_l} of Y.

- **Example**:
  - $\pi_1 = (n_1, m_1)$ indicates that x_{n_1} is best associated to y_{m_1}
  - $\pi_2 = (n_2, m_2)$ indicates that x_{n_2} is best associated to y_{m_2}
  - ...

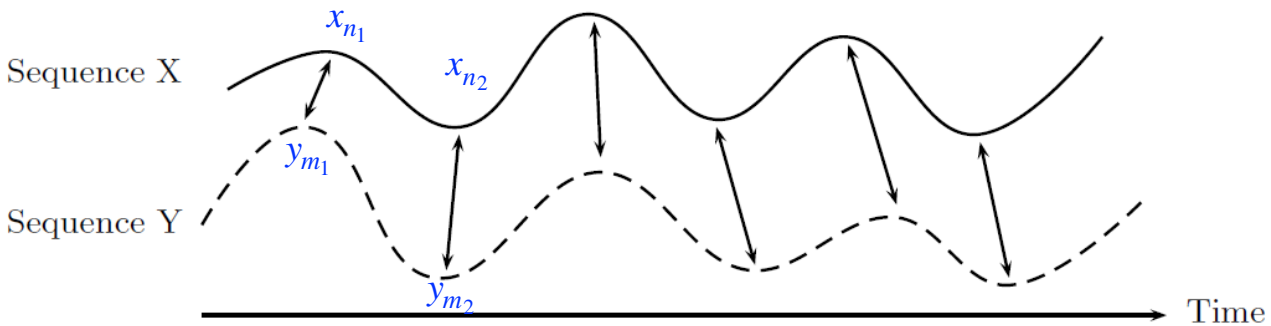
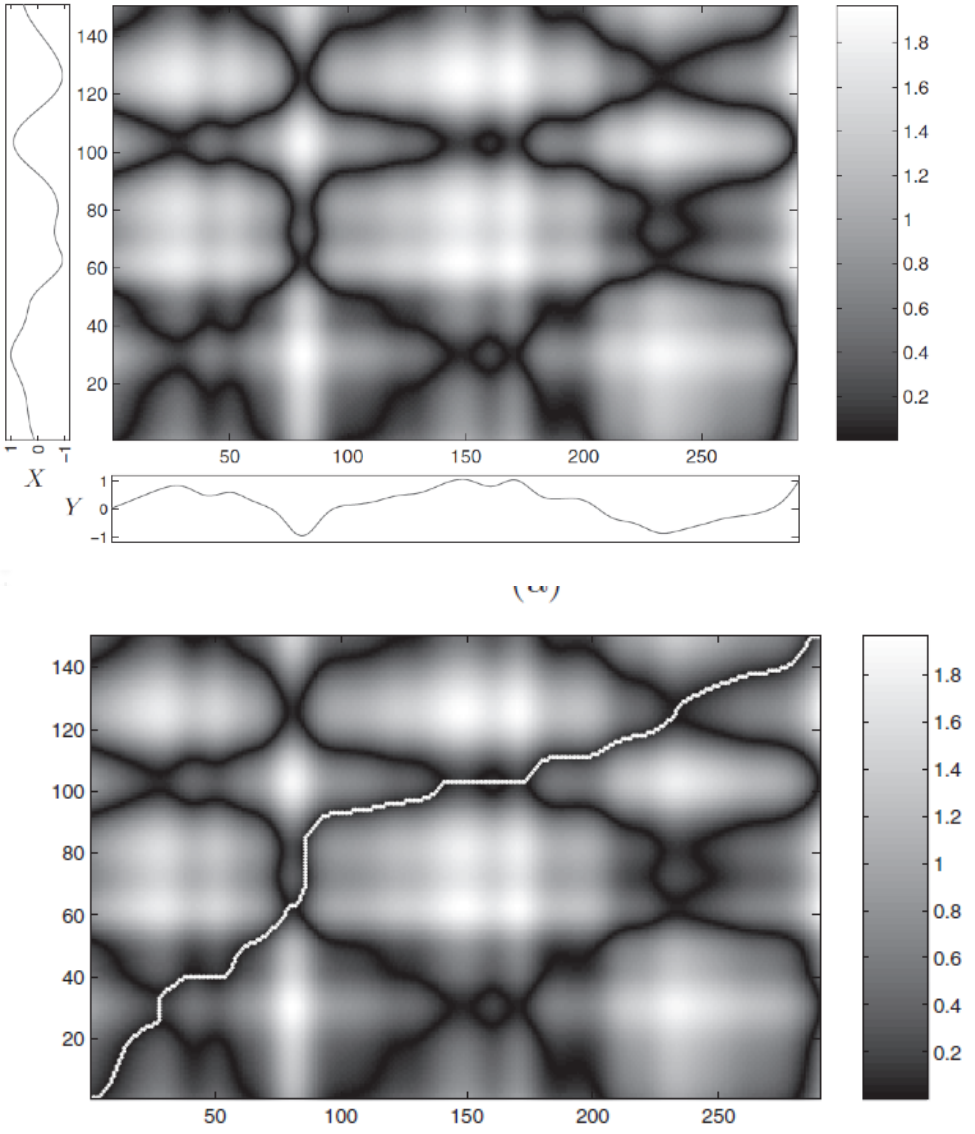


Fig. 4.1. Time alignment of two time-dependent sequences. Aligned points are indicated by the *arrows*

Dynamic Time Warping

- How to compute the **distance** between sequences?
  - **Sum** of the distances along the **best** warping path π^*
- DTW objective ?
  - We **look for the best warping path** π^*
  - i.e. the one that minimises the distance between the two sequences



Dynamic Time Warping

- First idea:
  - the distances associated with all warping paths (deformation) π are calculated and then the deformation path π^* for which the distance is minimal is chosen $\Rightarrow$ very expensive !!!

- Idea of the **DTW algorithm**:
  - we do not test all possible deformation paths
  - the optimal solution can be obtained from optimal **intermediate** solutions
  - the optimal intermediate solutions are obtained by an **iterative process**
    - calculates the minimum distance between each possible **sub**-sequence of X, and each possible **sub**-sequence of Y

Dynamic Time Warping

- We define the **local distance** as an Euclidean distance

$$c(\mathbf{x}_n, \mathbf{y}_m) = \sum_i \left(x_{n,i} - y_{m,i}\right)^2$$

- The optimal intermediate solutions are stored in a **cumulative distance** matrix

$$D(n,m) = \text{DTW}\left[X(1:n), Y(1:m)\right]$$

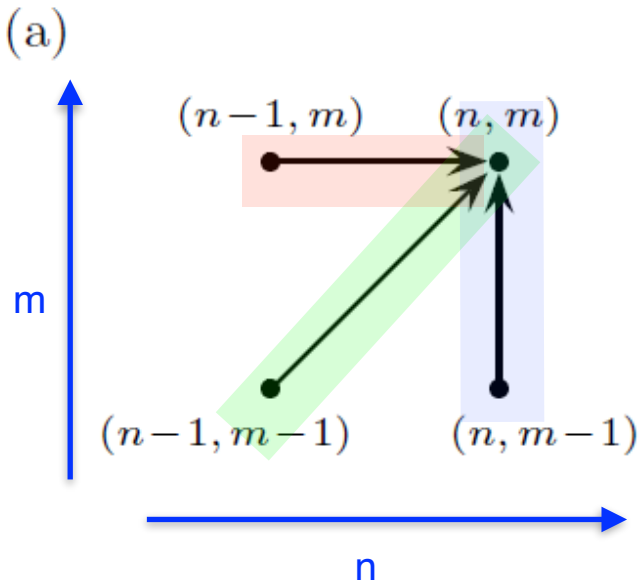
- $D(n,m)$ is calculated recursively by taking the minimum over three possible paths and adding local distance

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + c(x_n, y_m) \\ D(n-1,m) + c(x_n, y_m) \\ D(n,m-1)\} + c(x_n, y_m) \end{cases}$$

$$D(n,1) = \sum_{k=1}^{n} c(x_k, y_1)$$

$$D(1,m) = \sum_{k=1}^{m} c(x_1, y_k)$$

Initialisation: consider that the rows 0 and columns 0 have cost $= \infty$



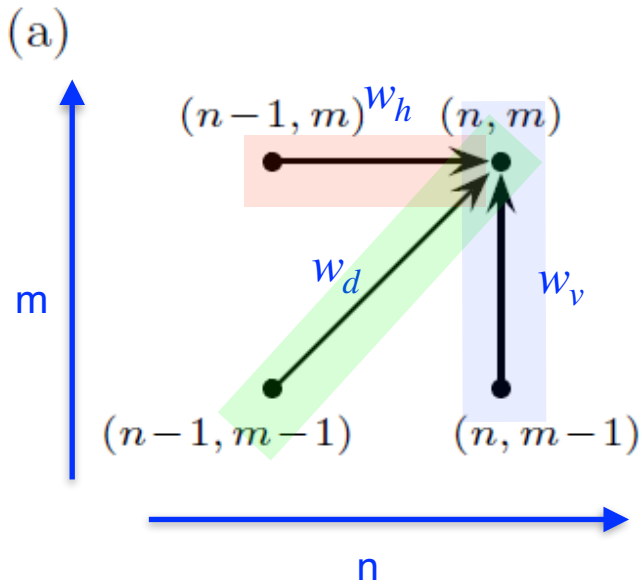
Dynamic Time Warping

- It is possible to add constraints on the calculation of cumulative distances
  - Ex: Predecessors limited to a few related elements, Paths only Left-Right

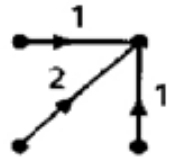
- Use of weights (penalties) according to the paths

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1)\} + w_v \cdot c(x_n, y_m) \end{cases}$$

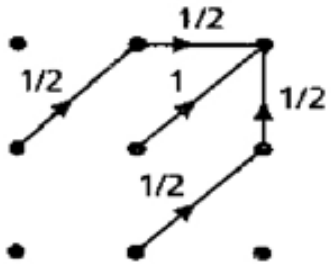
  - Objective of the weights ?
    - counterbalance the natural preference to the diagonal



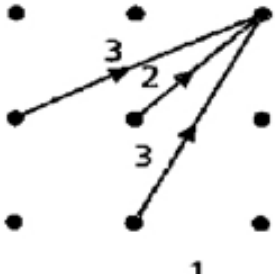
Dynamic Time Warping
Other possible paths



$$\min \left\{ \begin{array}{l} D(i_x - 1, i_y) + d(i_x, i_y), \\ D(i_x - 1, i_y - 1) + 2d(i_x, i_y), \\ D(i_x, i_y - 1) + d(i_x, i_y) \end{array} \right\}$$



$$\min \left\{ \begin{array}{l} D(i_x - 2, i_y - 1) + \frac{1}{2}[d(i_x - 1, i_y) + d(i_x, i_y)], \\ D(i_x - 1, i_y - 1) + d(i_x, i_y), \\ D(i_x - 1, i_y - 2) + \frac{1}{2}[d(i_x, i_y - 1) + d(i_x, i_y)] \end{array} \right\}$$



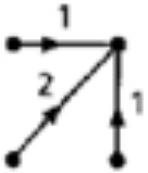
$$\min \left\{ \begin{array}{l} D(i_x - 2, i_y - 1) + 3d(i_x, i_y), \\ D(i_x - 1, i_y - 1) + 2d(i_x, i_y), \\ D(i_x - 1, i_y - 2) + 3d(i_x, i_y), \end{array} \right\}$$

Dynamic Time Warping
Practice

- A= (1 2 2 4 6) (test sequence)
- B= (2 3 4 5 6 7) (reference 1) or C= (1 2 4 4 6 6) (reference 2)

- Calculate the matrix of **local distances**
  - between A and B ?
  - between A and C ?
  - Use Euclidean distance $c(x_n, x_m) = \sum_i (x_{n,i} - y_{m,i})^2$

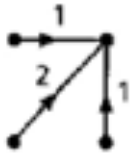
- Calculate the matrix of **cumulative distances/** constraint: $w_h = 1, w_v = 1, w_d = 2$



Dynamic Time Warping
Practice/ Solution

- A= (1 2 2 4 6) (test sequence)
- B= (2 3 4 5 6 7) (reference 1)
- Calculate the matrix of **cumulative distances/** constraint: $w_h = 1,\ w_v = 1, w_d = 2$

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1) + w_v \cdot c(x_n, y_m) \end{cases}$$



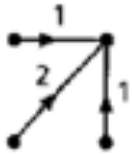
Distances locales

	7		36	25	25	9	1
	6		25	16	16	4	0
B	5		16	9	9	1	1
	4		9	4	4	0	4
	3		4	1	1	1	9
	2		1	0	0	4	16
			1	2	2	4	6
					A		

Dynamic Time Warping
Practice/ Solution

- A= (1 2 2 4 6) (test sequence)
- B= (2 3 4 5 6 7) (reference 1)
- Calculate the matrix of **cumulative distances/** constraint: $w_h = 1, w_v = 1, w_d = 2$

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1) + w_v \cdot c(x_n, y_m) \end{cases}$$



Distances locales

	7		36	25	25	9	1
	6		25	16	16	4	0
B	5		16	9	9	1	1
	4		9	4	4	0	4
	3		4	1	1	1	9
	2		1	0	0	4	16
			1	2	2	4	6
					A		

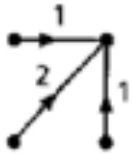
Distances cumulées

	7	100	91	56	56	16	4
	6	100	55	31	31	7	3
B	5	100	30	15	15	3	4
	4	100	14	6	6	2	6
	3	100	5	2	2	3	12
	2	100	1	1	1	5	21
		0	100	100	100	100	100
			1	2	2	4	6
					A		

Dynamic Time Warping
Practice/ Solution

- A= (1 2 2 4 6) (test sequence)
- B= (2 3 4 5 6 7) (reference 1)
- Calculate the matrix of **cumulative distances/** constraint: $w_h = 1,\ w_v = 1, w_d = 2$

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1) + w_v \cdot c(x_n, y_m) \end{cases}$$



Distances locales

	7		36	25	25	9	1
	6		25	16	16	4	0
B	5		16	9	9	1	1
	4		9	4	4	0	4
	3		4	1	1	1	9
	2		1	0	0	4	16
			1	2	2	4	6
					A		

Distances cumulées

	7	100	91	56	56	16	4
	6	100	55	31	31	7	3
B	5	100	30	15	15	3	4
	4	100	14	6	6	2	6
	3	100	5	2	2	3	12
	2	100	1	1	1	5	21
		0	100	100	100	100	100
			1	2	2	4	6
					A		

$2w_h + 2w_d + 3w_v = 2 + 4 + 3 = 9$

C = 4 / 9

Dynamic Time Warping
Backward algorithm

– finding the optimal path π^* can be obtained from optimal intermediate solutions

Algorithm: OPTIMALWARPINGPATH

Input: Accumulated cost matrix D.
Output: Optimal warping path p^*.

Procedure: The optimal path $p^* = (p_1, \ldots, p_L)$ is computed in reverse order of the indices starting with $p_L = (N, M)$. Suppose $p_\ell = (n, m)$ has been computed. In case $(n, m) = (1, 1)$, one must have $\ell = 1$ and we are finished. Otherwise,

$$p_{\ell-1} := \begin{cases} (1, m-1), & \text{if } n = 1 \\ (n-1, 1), & \text{if } m = 1 \\ \text{argmin}\{D(n-1, m-1), & \\ \qquad D(n-1, m), D(n, m-1)\}, & \text{otherwise,} \end{cases} \tag{4.6}$$

where we take the lexicographically smallest pair in case "argmin" is not unique.

Dynamic Time Warping
Final distance

- Calculation of the **final distance** ?
  - the distances on the optimal weighting path are **summed**
    - (take the high corner number of the cumulative distance)
  - **normalisation factor** used only at the end ?
    - the sum of the weights w_* along the path π^* from the start to the end point

Dynamic Time Warping
Practice/ Solution

- A= (1 2 2 4 6) (test sequence)
- C= (1 2 4 4 6 6) (reference 2)
- Calculate the matrix of **local distances**

Distances locales

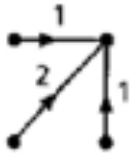
	6		25	16	16	4	0
	6		25	16	16	4	0
C:	4		9	4	4	0	4
	4		9	4	4	0	4
	2		1	0	0	4	16
	1		0	1	1	9	25
			1	2	2	4	6
					A		

merci à Antoine Marty

Dynamic Time Warping
Practice/ Solution

- A= (1 2 2 4 6) (test sequence)
- C= (1 2 4 4 6 6) (reference 2)
- Calculate the matrix of **cumulative distances/** constraint: $w_h = 1,\ w_v = 1, w_d = 2$

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1) + w_v \cdot c(x_n, y_m) \end{cases}$$



Distances locales

	6		25	16	16	4	0
	6		25	16	16	4	0
C:	4		9	4	4	0	4
	4		9	4	4	0	4
	2		1	0	0	4	16
	1		0	1	1	9	25
			1	2	2	4	6
					A		

Distances cumulées

	6	100	69	40	40	8	0
	6	100	44	24	24	4	0
C:	4	100	19	8	8	0	4
	4	100	10	4	4	0	4
	2	100	1	0	0	4	20
	1	100	0	1	2	11	36
		0	100	100	100	100	100
			1	2	2	4	6
					A		

$1w_h + 3w_d + 2w_v = 1 + 6 + 2 = 9$

C = 0 / 9 = 0

ASR: recognition of sentence

Sentence = concatenated words

Issues

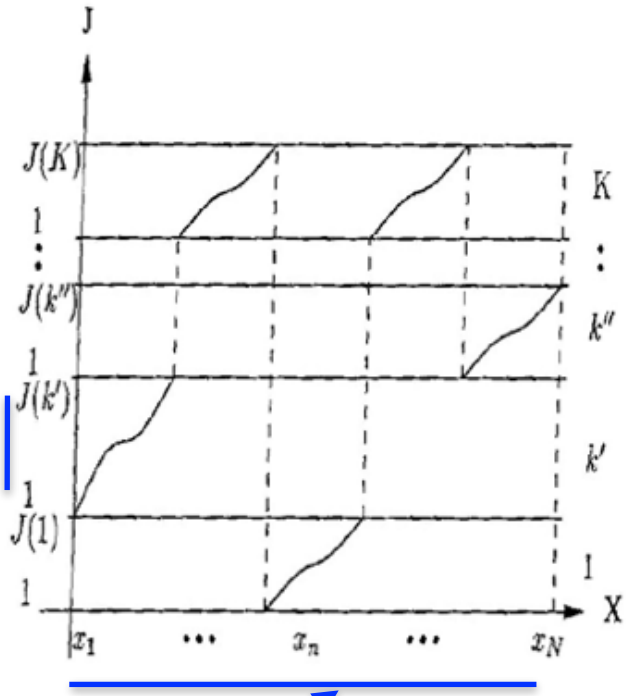
- 1/ **Combinatorial** explosion if all possible word sequences are tested as a reference

- 2/ **Coarticulation** between words
  - ex: table is not pronounced the same in “la tabl**e** rouge” vs. “la tabl**e e**t la chaise”

Sentence = concatenated words

Algorithm

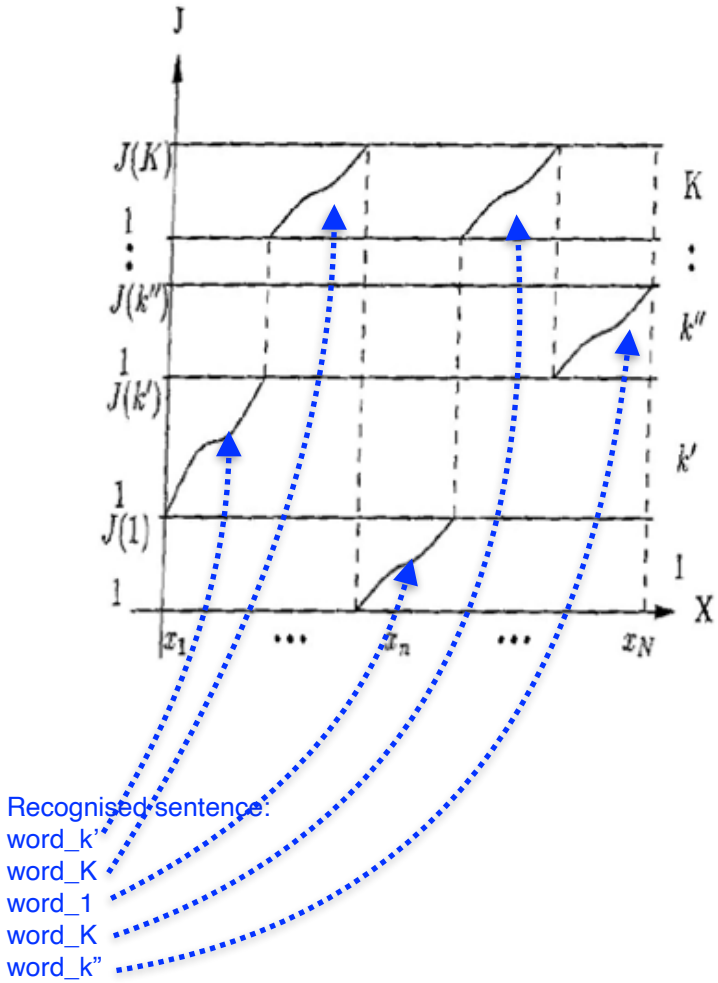
- 1) Construct a large matrix of local distances between :
  - **References:** each one of the $k \in K$ reference words in the vocabulary represented by the sequence of vectors $\{1, \ldots, J(k)\}$
  - **Observation**: the acoustic vectors of the test sentence seen as a sequence of time frames $(x_1, \ldots, x_N)$



Sentence = concatenated words

- 2) Dynamic programming on this matrix by setting **"jump" constraints**

- at the base of Viterbi used in Hidden Markov Models
  - (see https://web.stanford.edu/~jurafsky/slp3/A.pdf)



Sentence = concatenated words
Limits

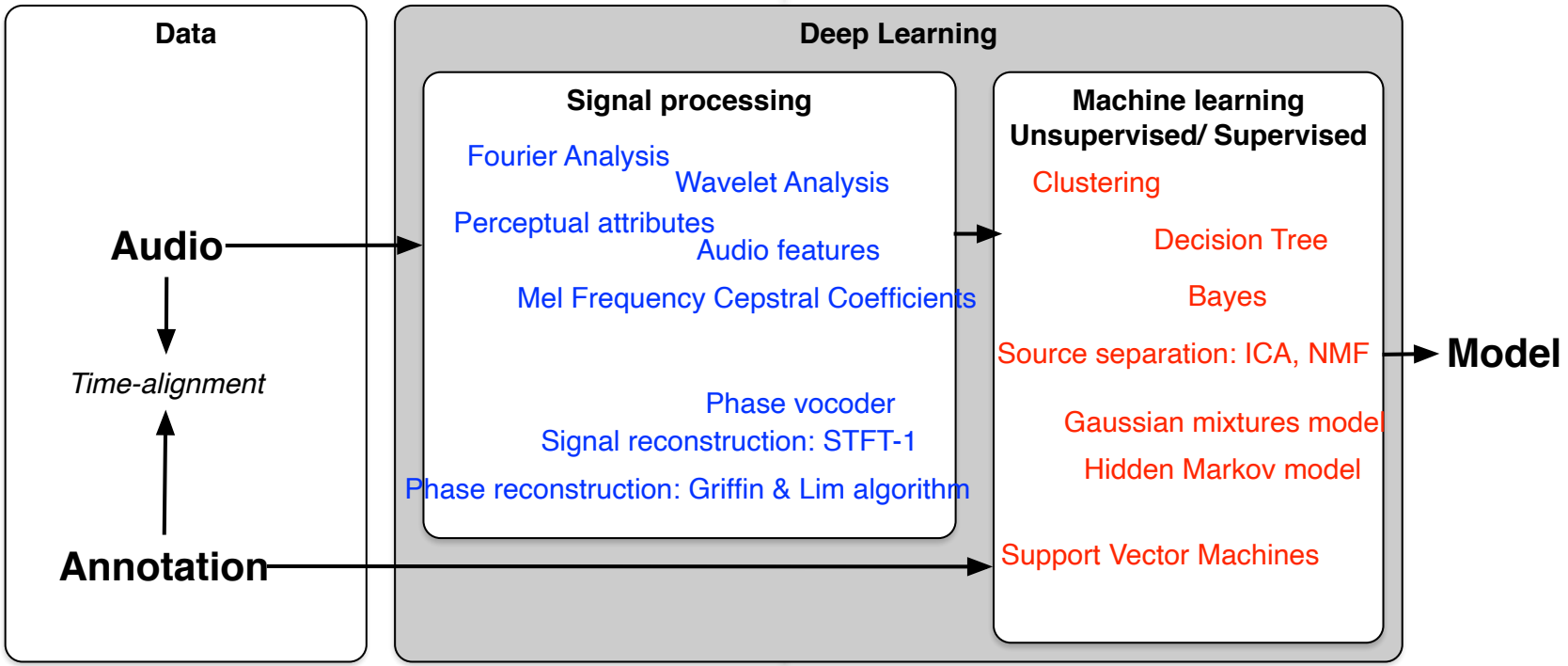
- Each word of the vocabulary is represented by **only one pronunciation**
  - Not OK for speaker-independent recognition

- A solution ?
  - use **acoustic models** based on Hidden Markov Models and language models
  - → the **traditional MFCC/GMM/HMM** approach

ASR: the traditional MFCC/GMM/HMM approach

ASR: the traditional MFCC/GMM/HMM approach

Audio features → Machine learning



ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM

- **Goal of Automatic Speech Recognition ?**
  - Find the most likely sentence (word sequence) W, which transcribes the speech audio A

$$\hat{W} = \arg\max_{W} P(W \mid A)$$
$$= \arg\max_{W} P(A \mid W) P(W)$$

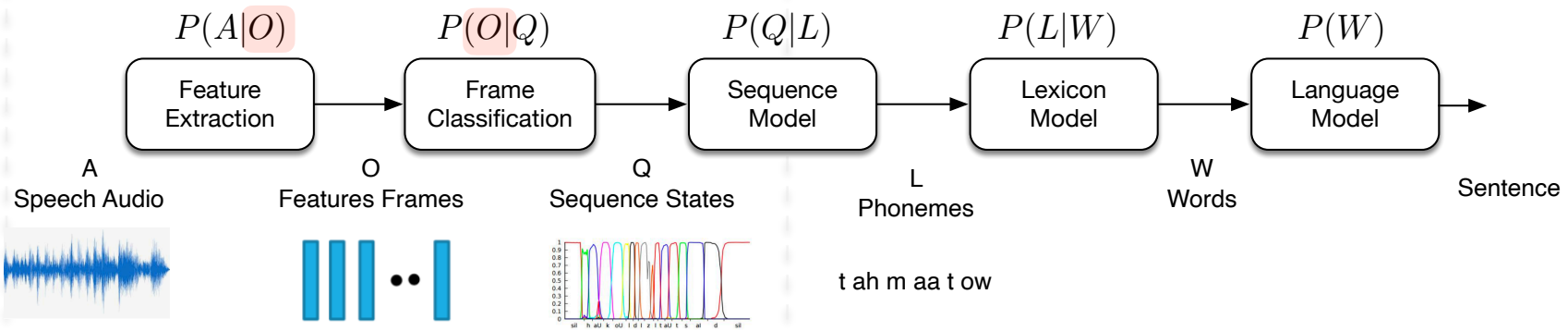
    - $P(A \mid W)$: the **acoustic** model
    - $P(W)$: the **language** model

- **Training ?**
  - Find the parameters of the acoustic and language models separately
    - Speech Corpus : speech waveform and human-annotated transcriptions
    - Language model : with extra data (prefer daily expressions corpus for spontaneous speech)

Traditional approach: GMM-HMM

– **Architecture of Speech Recognition**

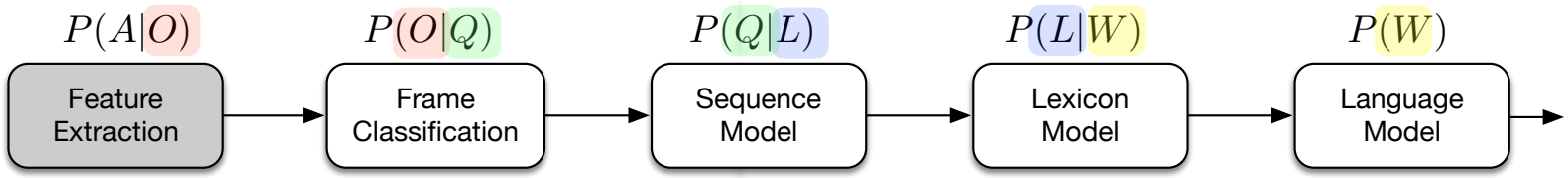
$$\hat{W} = \arg\max_W P(W|A)$$
$$= \arg\max_W P(A|W)P(W)$$
$$= \arg\max_W P(A|O)P(O|Q)P(Q|L)P(L|W)P(W)$$



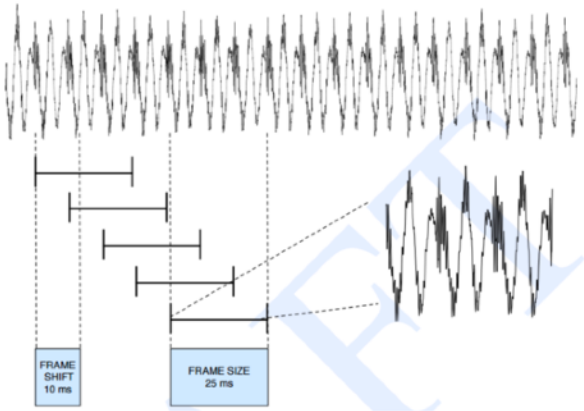
source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

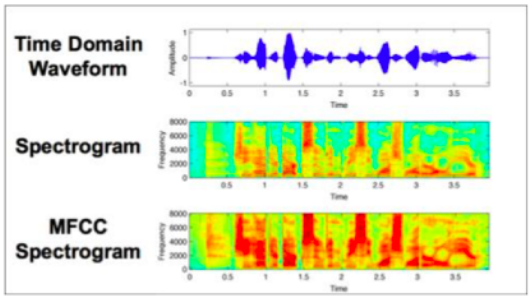
ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



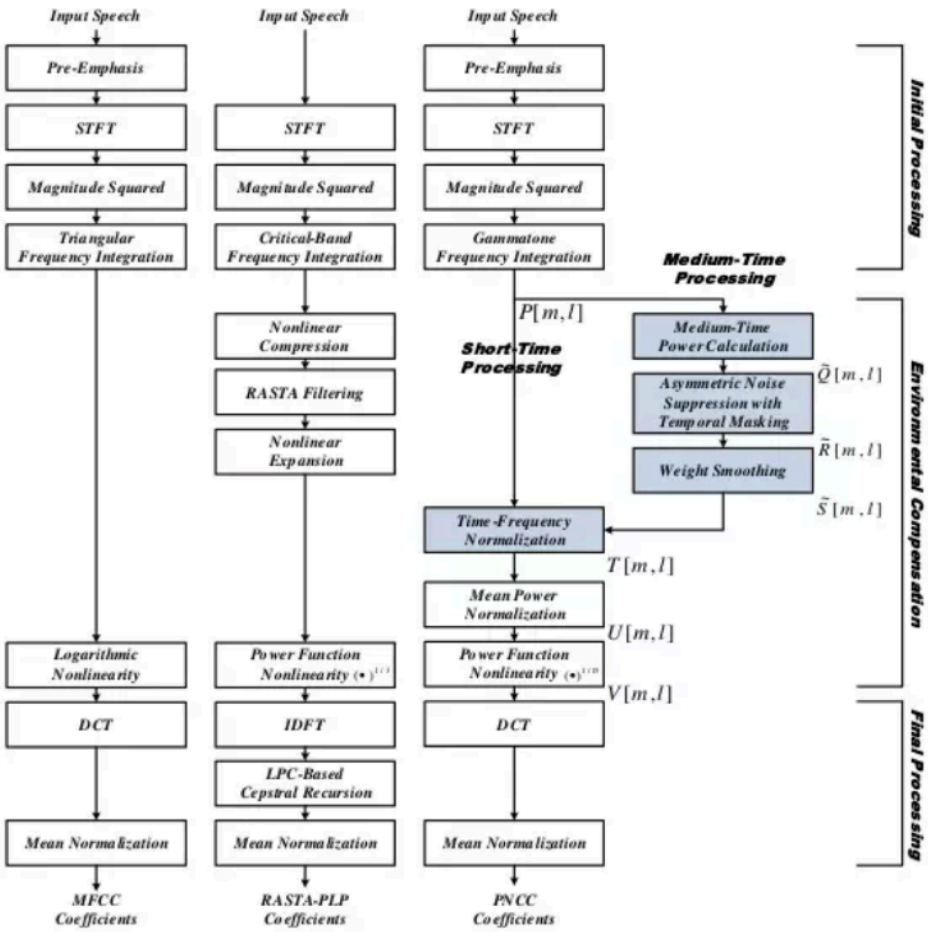
- $P(A \mid O) = P(Audio \mid Features)$
- **Feature extraction**
  - Raw waveforms are transformed into a sequence of feature vectors using signal processing approaches
  - Time domain to frequency domain
- Feature extraction is a deterministic process : $P(A \mid O) = \delta(A, \hat{A}(O))$
- Reduce information rate but keep useful information
  - Remove noise and other irrelevant information
- Extracted in 25ms windows and shifted with 10ms
- Still useful for deep models





source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

ASR: the traditional MFCC/GMM/HMM approach
Audio features



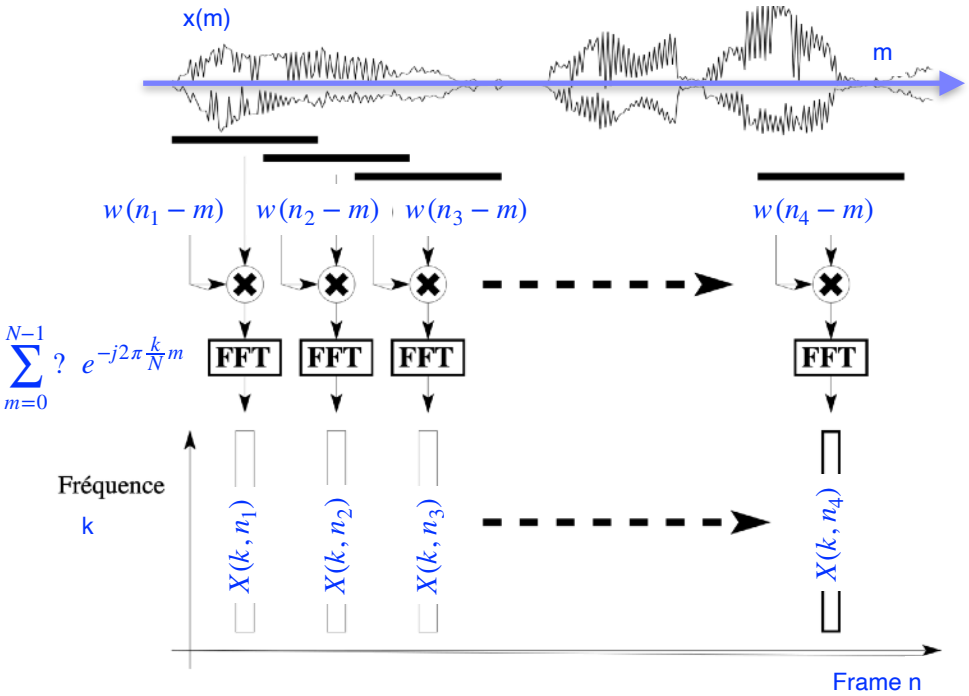
Comparison of feature extraction process for MFCC, Rasta-PLP and PNCC

Audio features

START →

Fourier Transform short time (STFT)

- $$X(k,n) = \sum_{m=-\infty}^{+\infty} \underline{w(n-m)}\, x(m)\, e^{-j2\pi\frac{k}{N}m} \quad \forall k \in [0, N-1]$$
  - Apply the DFT on successive segments (frames) of the audio signal,
    - each segment (frames) is centred around (start at) sample $n \in \mathbb{R}^+$
  - Why the STFT ?
    - Audio signal= non-stationary
      - its properties evolve over time
    - "Local" (in time) stationarity
      - over a time duration of $\pm 40\ ms$
    - STFT: a set of successive DFT over segments of $\pm 40\ ms$ duration
      - = Short-time analysis ("frames" in video)



Fourier Transform short time (STFT)

- $$X(k,n) = \sum_{m=-\infty}^{+\infty} \underline{w(n-m)}\, x(m)\, e^{-j2\pi \frac{k}{N} m} \quad \forall k \in [0, N-1]$$
  - Analysis window $w(t)$
  - $x(t) \cdot w(t) \rightleftarrows X(f) \circledast W(f)$
    - $w(t)$ is named "analysis window"
    - Parameters:
      - 1) **type** of the analysis window $w(t)$
      - 2) **time duration** L of the window $w(t)$
    - The type and time duration defines the spectral characteristics of the STFT
      - Width of the main lobe (at $-6dB_{20}$): $Bw = \frac{Cw}{L}$
      - Height of secondary lobes

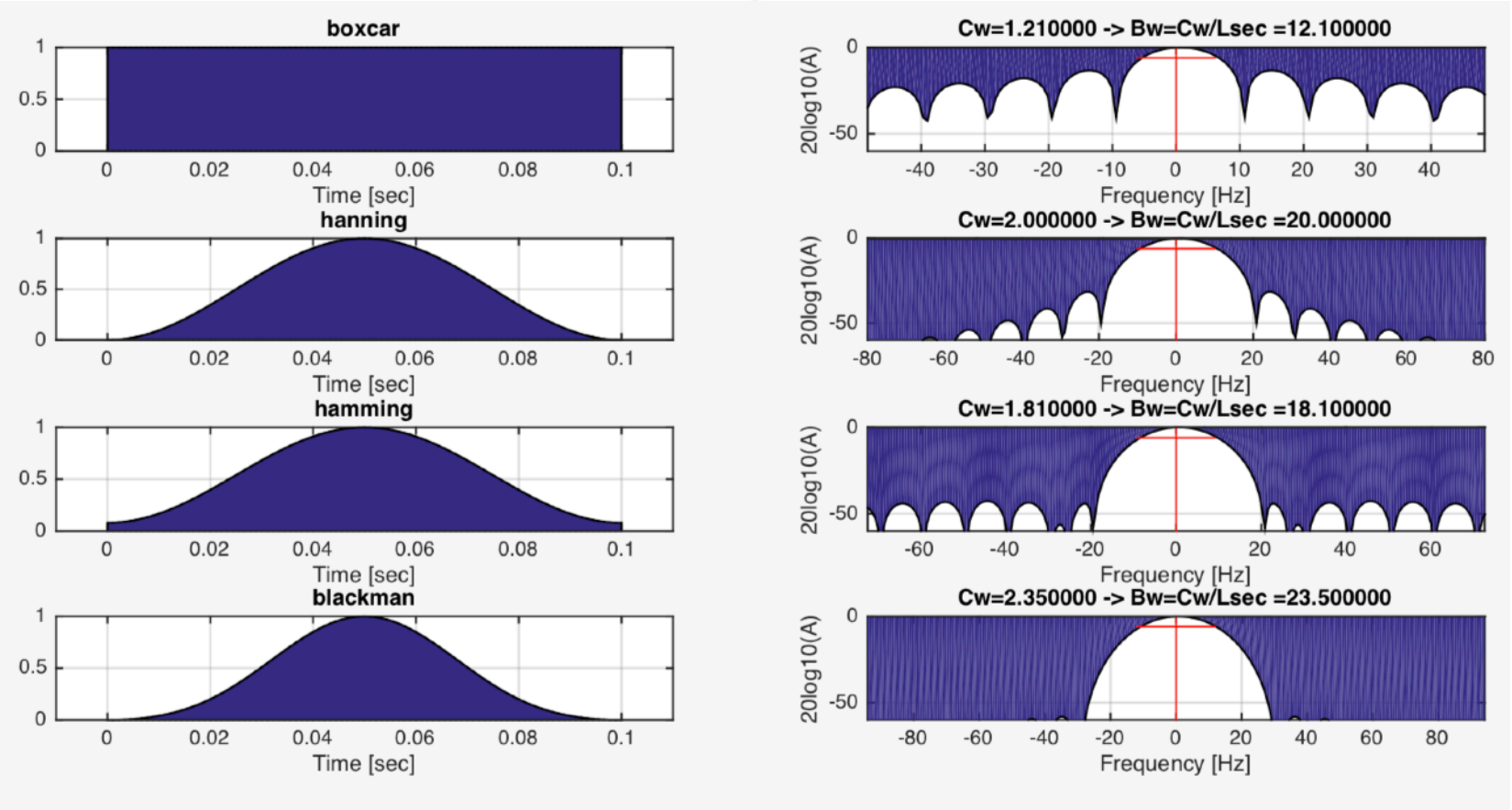
Propriétés	$x(t)$	$X(f)$
Similitude	$x(at)$	$\frac{1}{\lvert a \rvert} X(\frac{f}{a})$
Linéarité	$ax(t) + by(t)$	$aX(f) + bY(f)$
Translation	$x(t - t_0)$	$X(f) \exp(-j2\pi f t_0)$
Modulation	$x(t) \exp(j2\pi f_0 t)$	$X(f - f0)$
Convolution	$x(t) \circledast y(t)$	$X(f)Y(f)$
Produit	$x(t)y(t)$	$X(f) \circledast Y(f)$
Parité	réelle paire	réelle paire
	réelle impaire	imaginaire paire
	imaginaire paire	imaginaire paire
	imaginaire impaire	réelle impaire
	complexe paire	complexe paire
	complexe impaire	complexe impaire
	réelle	$X(f) = X^*(-f)$
		$\Re(X(f))$ est paire
		$\Im(X(f))$ est impaire
	$x^*(t)$	$X^*(f)$

Fourier Transform short time (STFT)

- 1) **Type** of the analysis window $w(t)$
  - Rectangular
    - $w(t) = 1$ si $t \in [0,T]$
    - $Cw = 1.21$
  - Hann
    - $w(t) = 0.5 - 0.5\cos\left(2\pi\frac{t}{L}\right)$ si $t \in [0,T]$
    - $Cw = 2$
  - Hamming
    - $w(t) = 0.54 - 0.46\cos\left(2\pi\frac{t}{L}\right)$ si $t \in [0,T]$
    - $Cw = 1.81$
  - Blackman
    - $w(t) = 0.42 - 0.5\cos\left(2\pi\frac{t}{L}\right) + 0.08\cos\left(4\pi\frac{t}{L}\right)$ si $t \in [0,T]$
    - $Cw = 2.35$

Fourier Transform short time (STFT)

- 1) **Type** of the analysis window $w(t)$

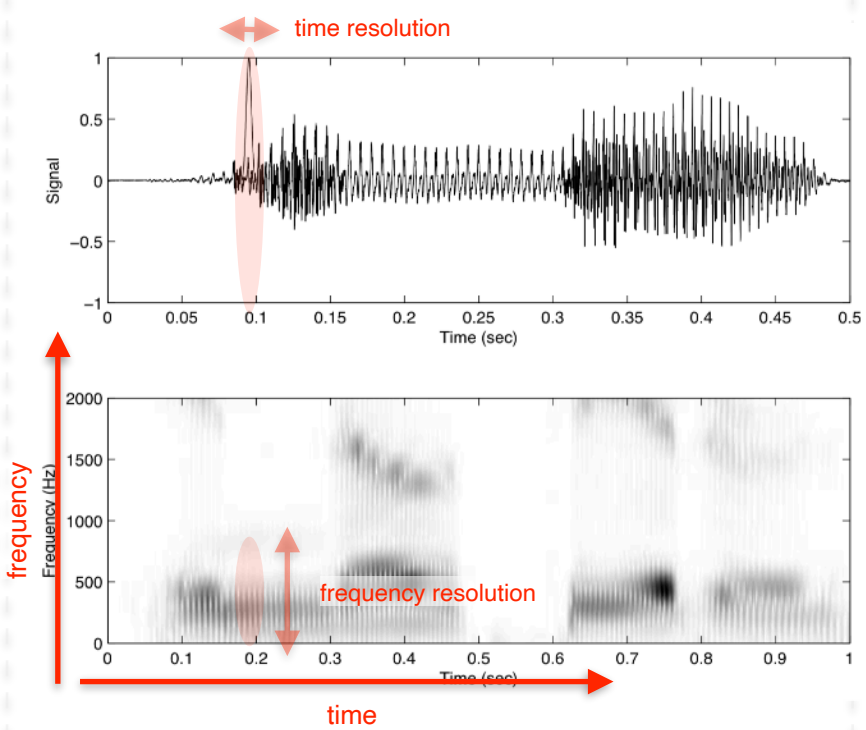


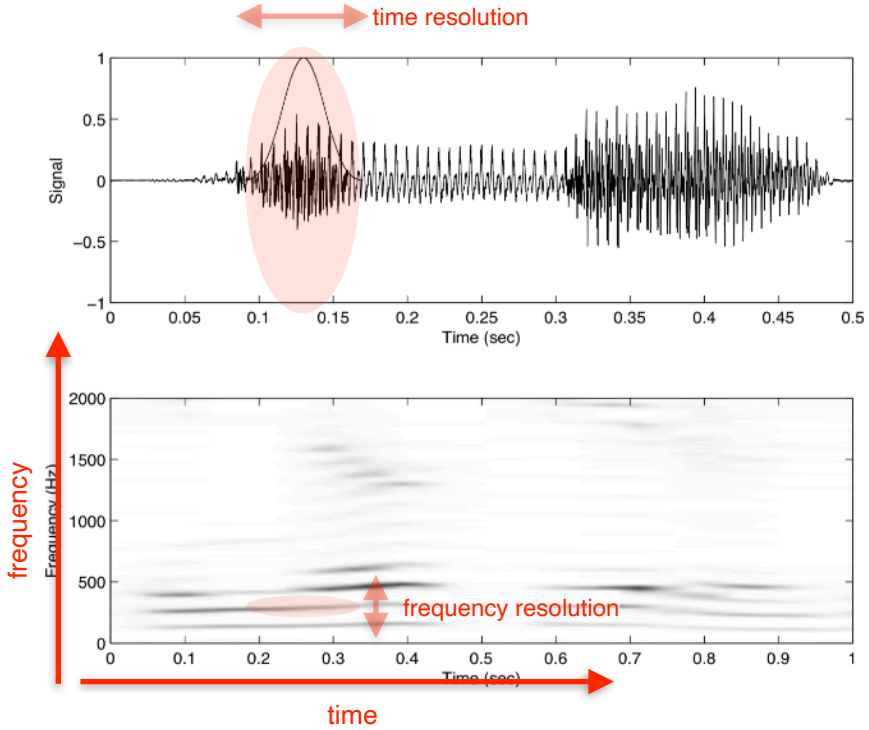
Fourier Transform short time (STFT)

- 2) **Time duration** L of the window $w(t)$
  - The shortest the window is ($L \ll$),
    - the largest temporal resolution we get (but the lowest spectral resolution)
  - The largest the window is ($L \gg$),
    - the larget spectral resolution we get (but the lowest temporal resolution)

Fourier Transform short time (STFT)

- **Time and frequency paradox**
  - It is not possible to observe the signal with a good time and frequency localisation simultaneously !



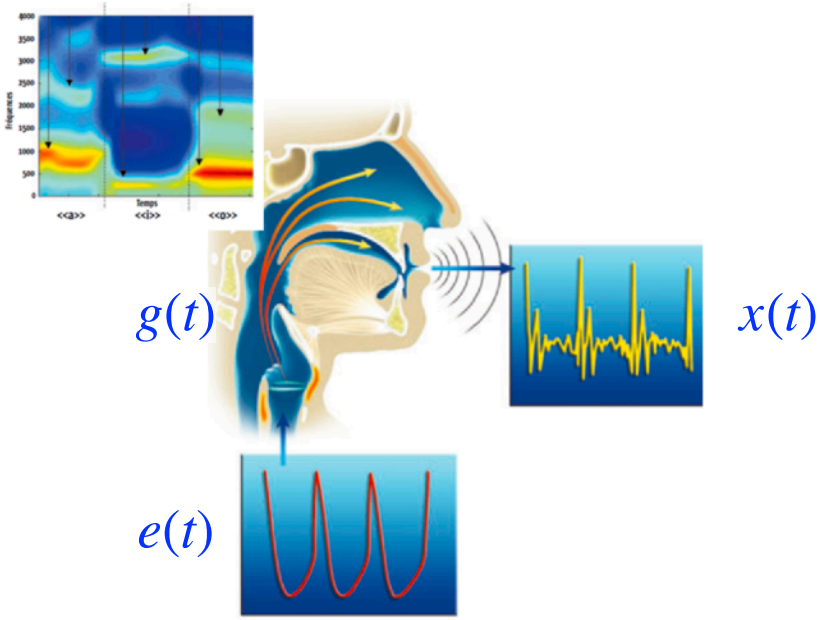


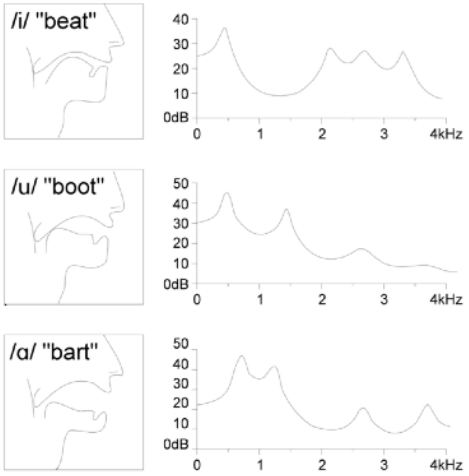
- **How to solve this issue ?**
  - Other transforms: wavelet, constant-Q-transform

Source/Filter model

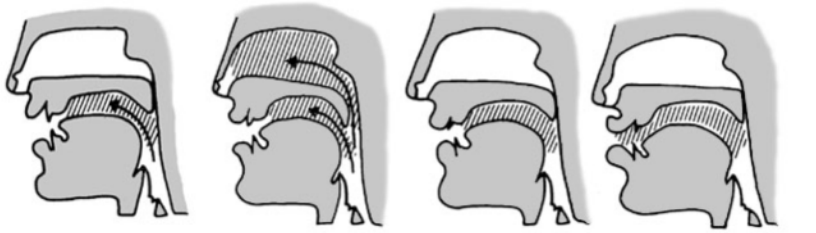
Model:

- represent the signal $x(t)$ as the results of a periodic pulse signal convolved with a resonant filters
- Example:
  - speech (voiced part)
    - vocal folds $e(t)$
    - mouth (resonance), nose (anti-resonance) $g(t)$
  - many musical instruments (trumpet)





source : Mike Brookes



source : outilsrecherche.over-blog.com

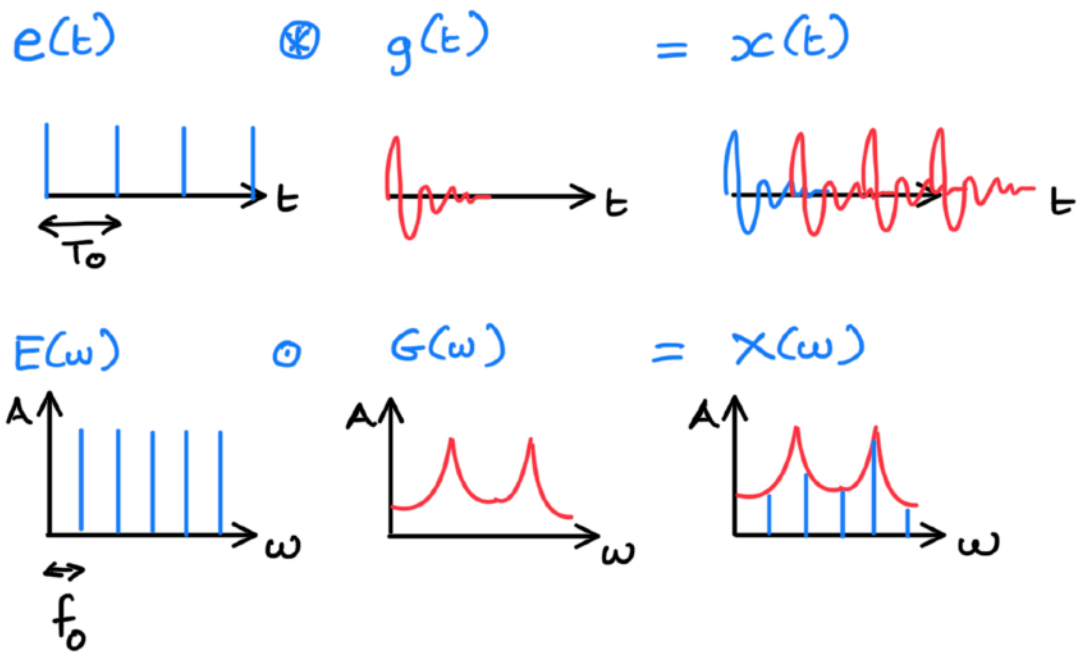
Source/Filter model

In time:

$$x(t) = e(t) \circledast g(t)$$

In frequency:

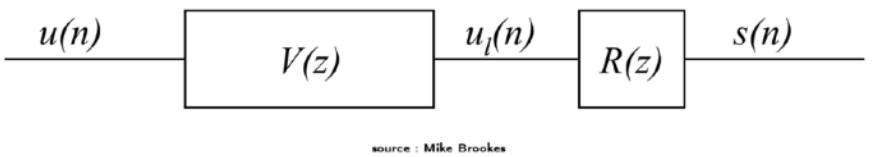
$$X(\omega) = E(\omega) \cdot G(\omega)$$



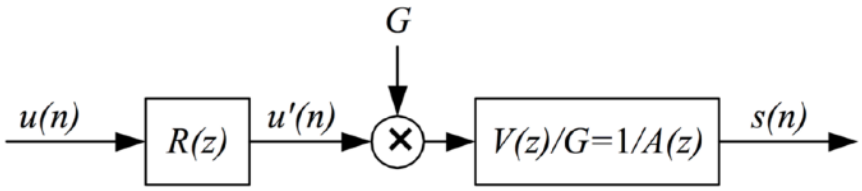
Source/Filter model

Resonant filter:

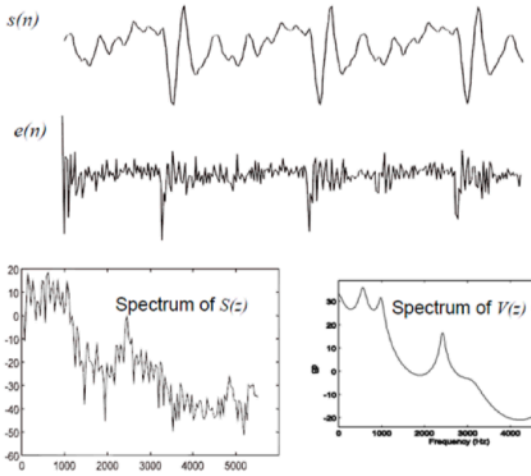
- $$V(z) = \frac{G}{1 - \sum_{j=1}^{P} a_j z^{-j}}$$
  - $$x(n) = Gu'(n) + \sum_{j=1}^{P} a_j x(n-j)$$
- If the gain of the resonances of the vocal fold are import, the second term dominates and
  - $$x(n) \simeq \sum_{j=1}^{P} a_j x(n-j)$$
  - This is an **auto-regressive model**



source : Mike Brookes



source : Mike Brookes



Complex cepstrum

– **Goal**

- describe the shape of the spectrum (the timbre) of a signal using a reduced set of coefficients

– **Complex cepstrum** $c(\tau)$

$$c(\tau) = TF^{-1}\left[\log(X(\omega))\right]$$
$$= \frac{1}{2\pi}\int_{\omega} \log[X(\omega)] \cdot e^{j\omega\tau} d\omega$$

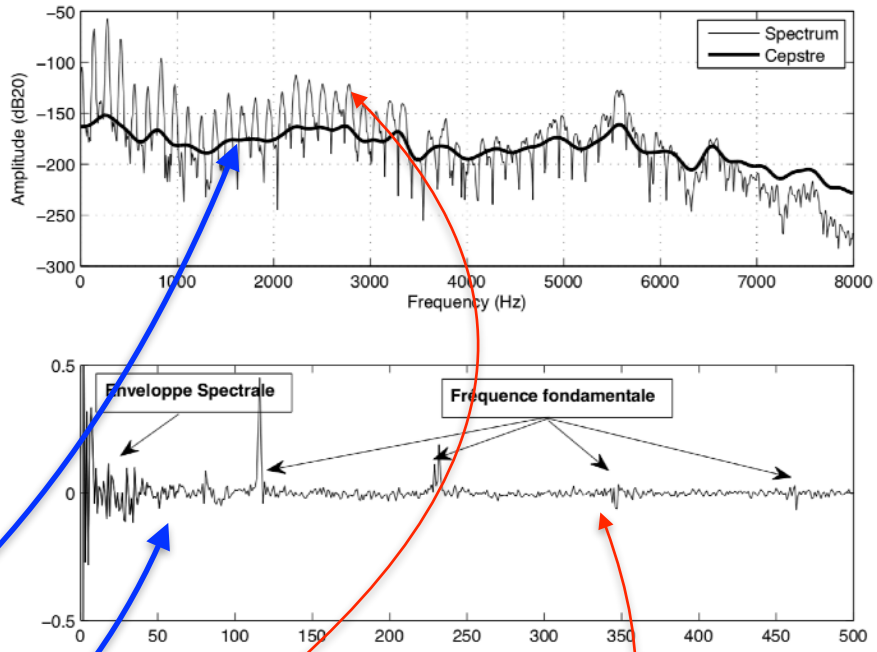
- τ is named "**quefrency**" (=frequency in reverse order)
- $x(t) \xrightarrow{TF} X(\omega) \xrightarrow{\log} \log(X(\omega)) \xrightarrow{TF^{-1}} c(\tau)$

Source/filter model

- Source $e(t)$: periodic signal
- Filter $g(t)$: resonant/ anti-resonant filter

$$x(t) = e(t) \circledast g(t)$$

$$\xrightarrow{TF} X(\omega) = E(\omega) \cdot G(\omega)$$



$$\xrightarrow{\log} \log(X(\omega)) = \underbrace{\log[G(\omega)]}_{\text{slow variations over } \omega} + \underbrace{\log[E(\omega)]}_{\text{fast variations over } \omega}$$

$$\xrightarrow{TF^{-1}} TF^{-1}\left[\log(X(\omega))\right] = \underbrace{TF^{-1}\left[\log[G(\omega)]\right]}_{\text{energy at quefrency } \tau<<} + \underbrace{TF^{-1}\left[\log[E(\omega)]\right]}_{\text{energy at quefrency } \tau>>}$$

Real cepstrum

– **Real ?** = cepstrum computed on the real part of the log-spectrum

$$X(\omega) = A(\omega) \cdot e^{j\phi(\omega)}$$
$$\log[X(\omega)] = \log[A(\omega)] + j\phi(\omega)$$
$$\Re\{\log[X(\omega)]\} = \log[A(\omega)]$$

$$\text{real cepstrum} = TF^{-1}\left[\Re\{\log[X(\omega)]\}\right]$$
$$= TF^{-1}\left[\log[A(\omega)]\right]$$
$$c(\tau) = \frac{1}{2\pi}\int_{\omega} \log[A(\omega)] \cdot e^{j\omega\tau} d\omega$$

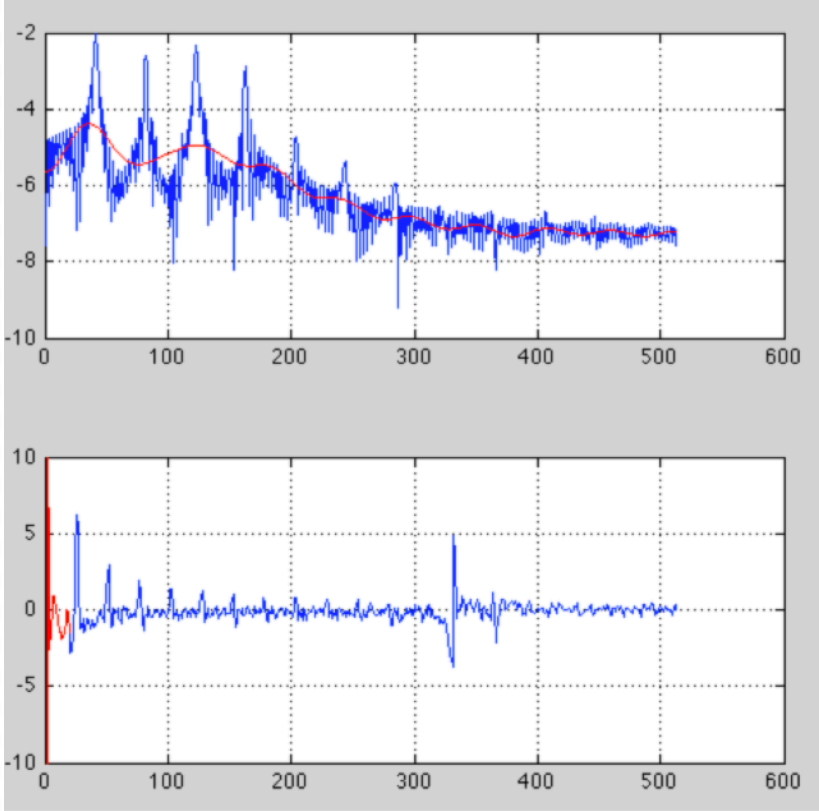
– The amplitude spectrum $A(\omega)$ is real and symmetric
  - its Fourier Transform reduces to the real part
    – reduces to the projection of $\log[A(\omega)]$ on a set of cosinus → Discrete Cosine Transform (DCT)

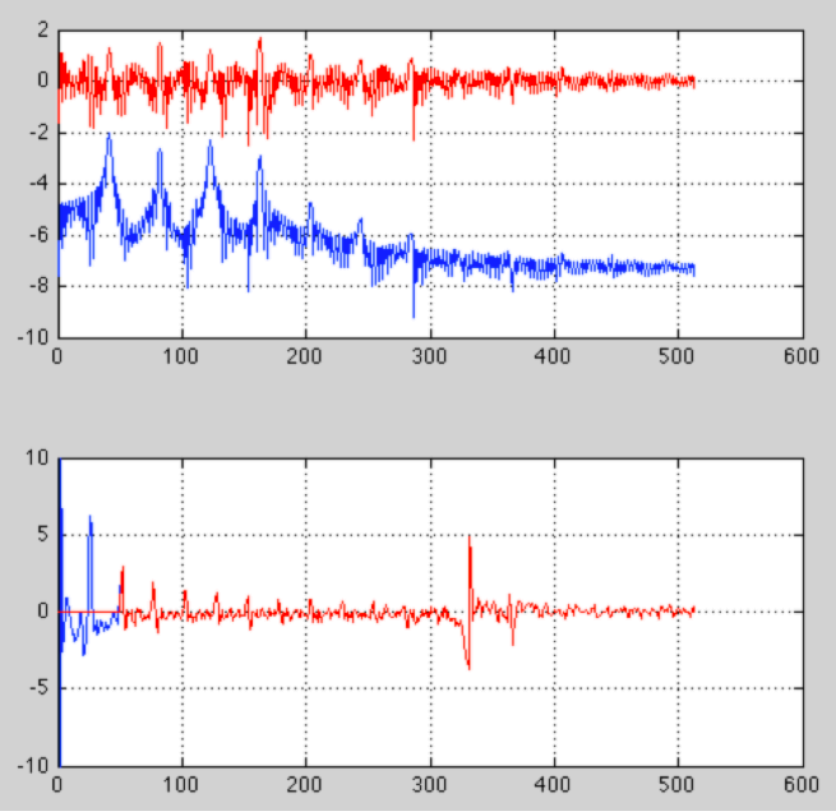
Audio features examples
Mel Frequency Cepstral Coefficients (4)

Real cepstrum

Spectral envelope:
- varies slowly over frequencies
- low quefrency (low frequency of FT^{-1}

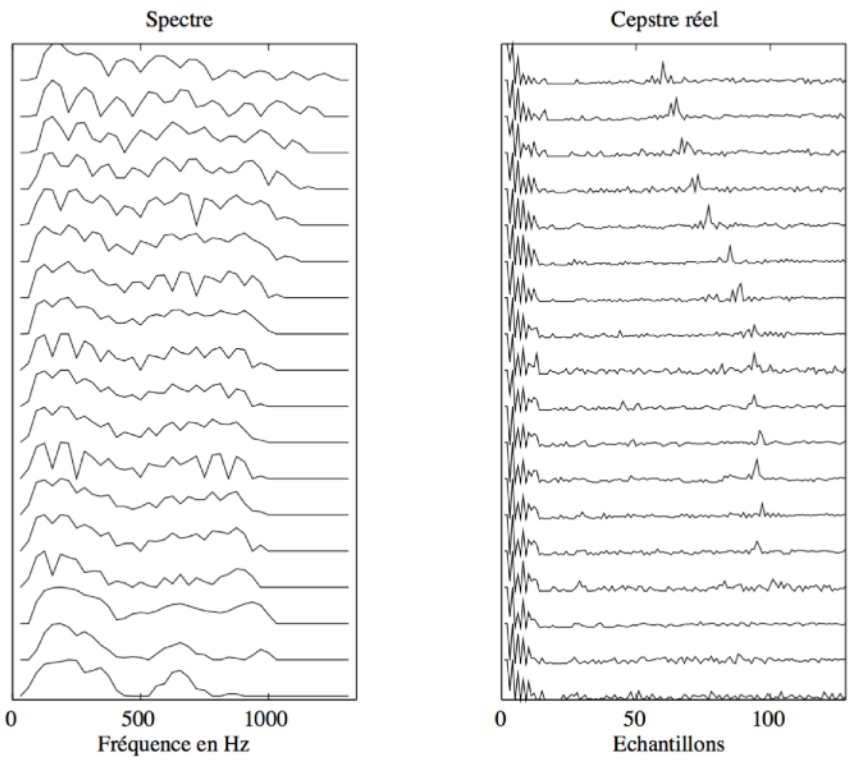
Fundamental frequency:
- varies quickly over frequencies
- high quefrency (high frequency of FT^{-1}



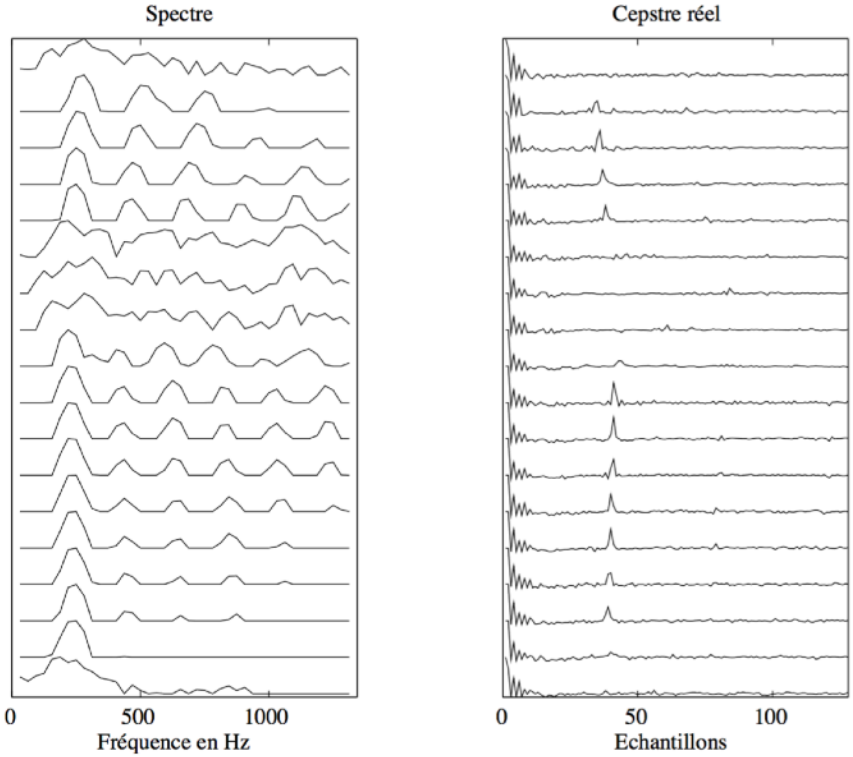


Audio features examples
Mel Frequency Cepstral Coefficients (5)

Real cepstrum



source : voix d'homme, Laroche, 1995



source : voix de femme, Laroche, 1995

Audio features examples
Mel Frequency Cepstral Coefficients (6)

Mel Frequency Cepstral Coefficients (MFCCs)

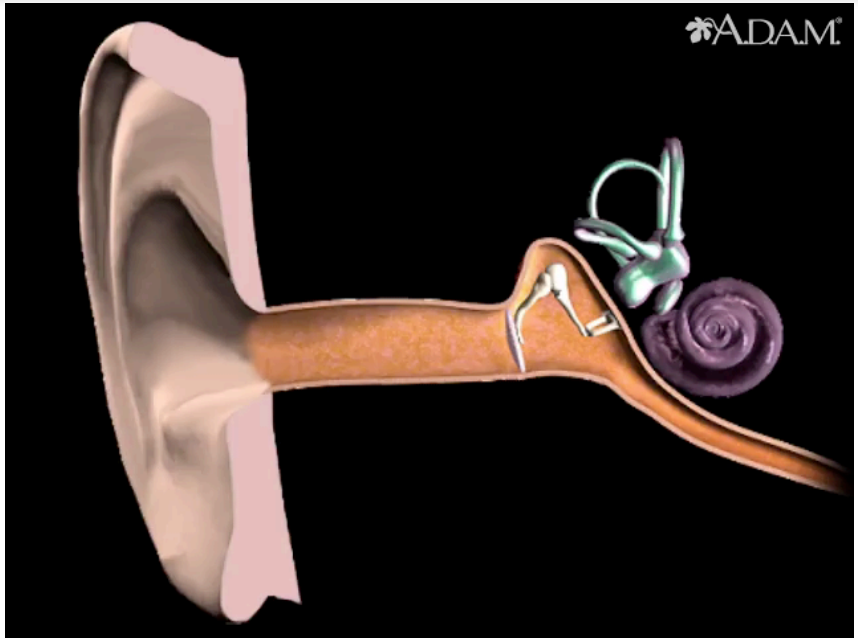
- MFCC ? = real cepstrum computed on the power spectrum $|X(\omega)|^2$ converted to the Mel scale (a perceptual scale)

- **Why perceptual scales ?**
  - Fourier Transform
    - decomposition on a set of sinusoidal components which frequencies are linearly spaced (f_k = 10Hz, 20Hz, 30Hz, ... Hz)
  - Human hearing:
    - decomposition on a set of filters which frequencies are logarithmically spaced (10, 20, 40, 80, ... Hz).
    - highest resolution in low-frequencies, lowest resolution in high frequencies
    - in speech, formants/resonances are closer together in low frequencies
  - MFCCs allows a more compact representation than the real cepstrum
- **How ?**
  - Use of perceptual scales: Mel-scale, Bark-scale, ERB-filters, Gamma-tone filters
- **Usage ?**
  - MFCCs are the most used features in audio: speech, music, environmental sounds recognition, ...

Audio features examples
Mel Frequency Cepstral Coefficients (7)

Human hearing

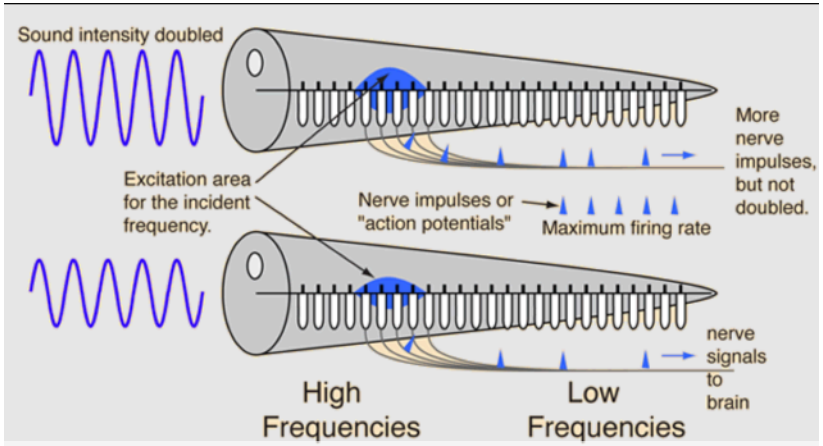
– Cochlea



https://medlineplus.gov/ency/anatomyvideos/000063.htm

– Critical bands
  - perception of two tones at f_1 and f_2
  - perception of a beating-tone at $\frac{f_1+f_2}{2}$

$$\cos f_1 + \cos f_2 = 2\cos\frac{f_1-f_2}{2}\cos\frac{f_1+f_2}{2}$$



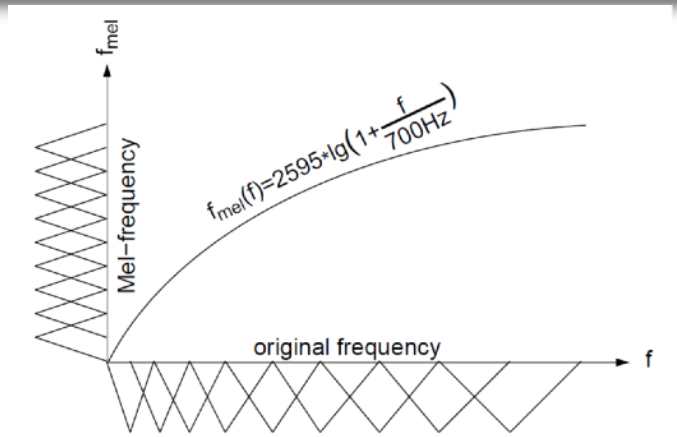
source: http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/loud.html

Audio features examples
Mel Frequency Cepstral Coefficients (8)

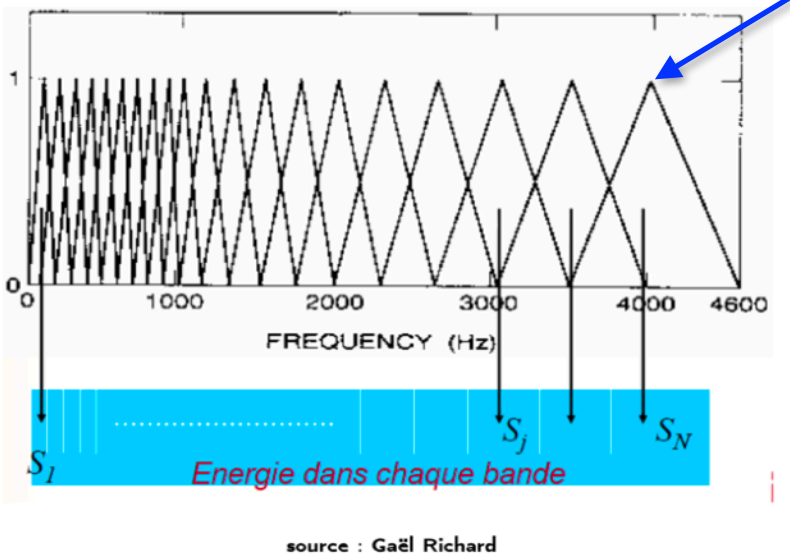
Mel scale ?

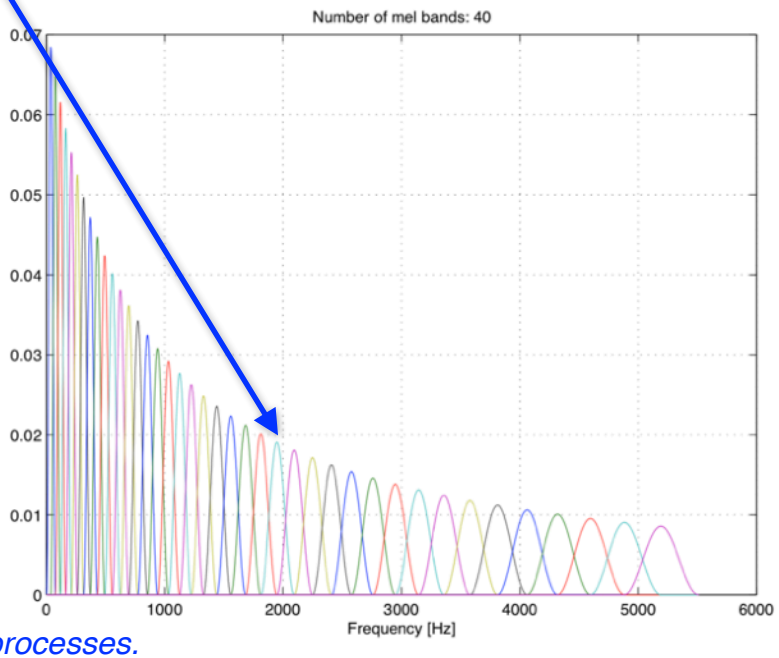
$$mel(f) = \frac{1000}{\ln 2} \ln\left(1 + \frac{f}{1000}\right)$$

- Remark: variations of the constant exist



different shapes for the filter: triangular, hanning, tanh

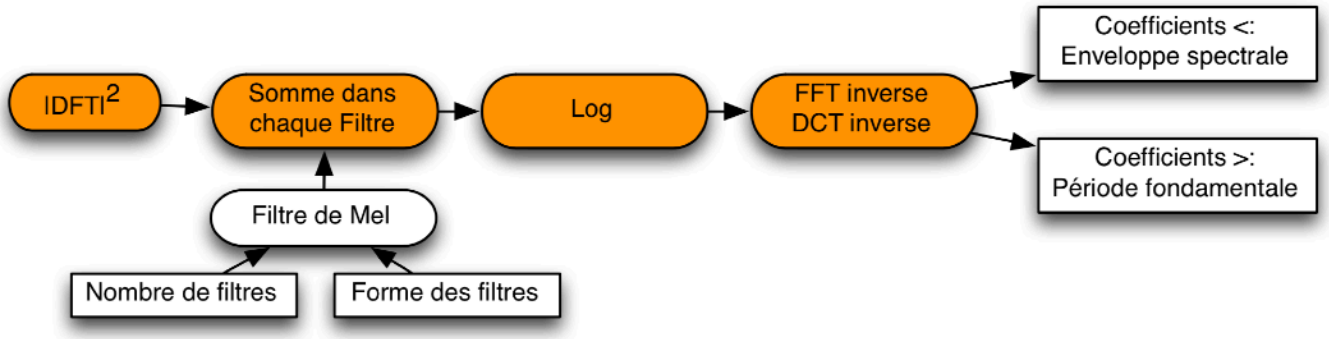




Fant, Gunnar. (1968). *Analysis and synthesis of speech processes.* In B. Malmberg (Ed.), *Manual of phonetics* (pp. 173-177). Amsterdam: North-Holland.

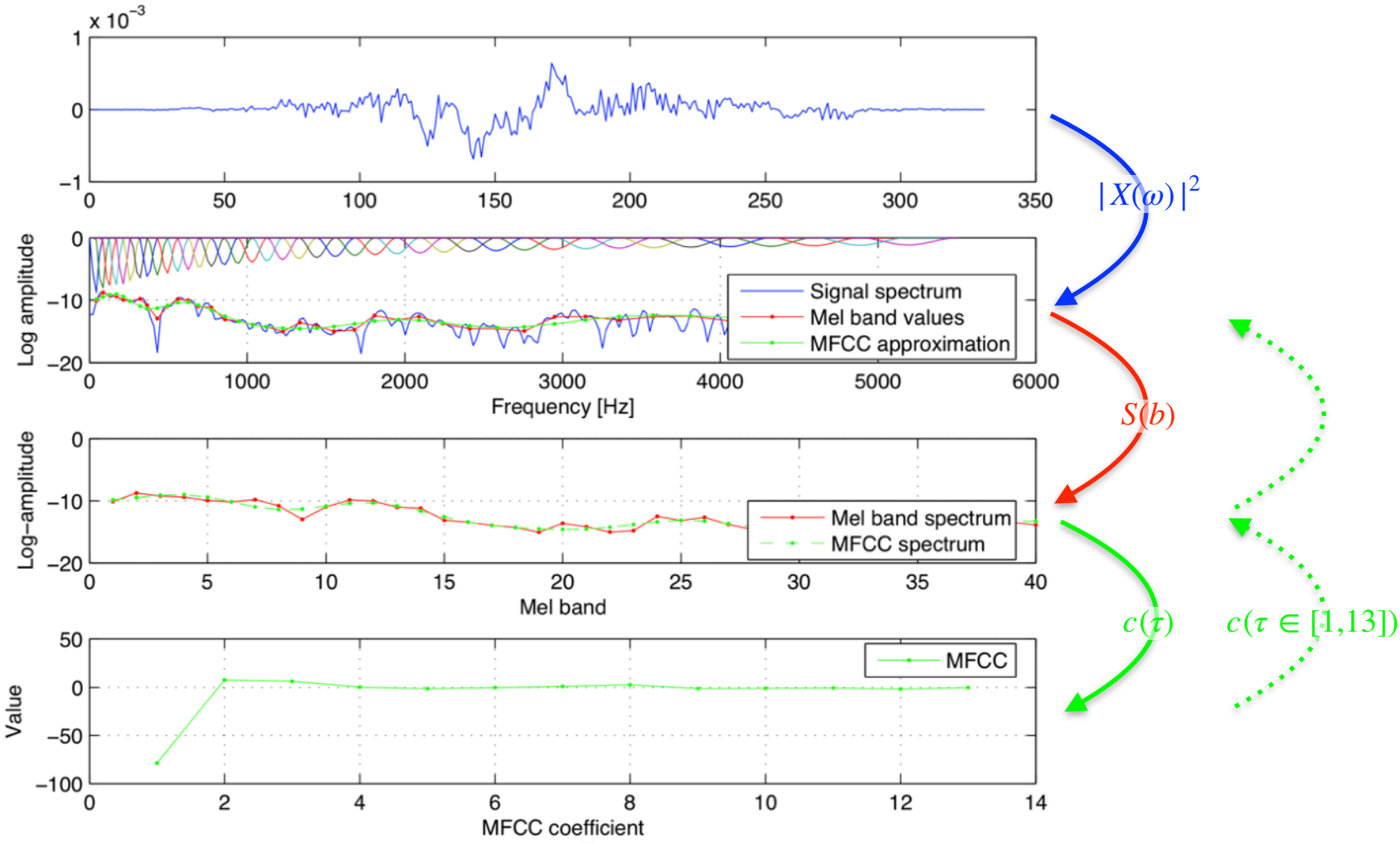
Computation steps for MFCCs

- Compute the power spectrum: $|X(\omega)|^2$
- Compute the Mel filters: $H_b(\omega)$ with $b \in [1,B]$
  - choice of the number of filters B: 40
  - choice of the shape of each filter: triangular, hanning, tanh, ...
- Convert the power spectrum to Mel bands: $S(b) = \sum_{\omega} |X(\omega)|^2 \cdot H_b(\omega)$
- Convert to logarithmic scale: $\log(S(b))$
- Compute the IFFT (or the IDCT): $c(\tau)$
- Select the first coefficients, close to 0 (usually the first 13 coefficients)
  - coefficients close to zero represent the decomposition of the Mel bands content on a set of cosinus with slow variations



Audio features examples
Mel Frequency Cepstral Coefficients (10)

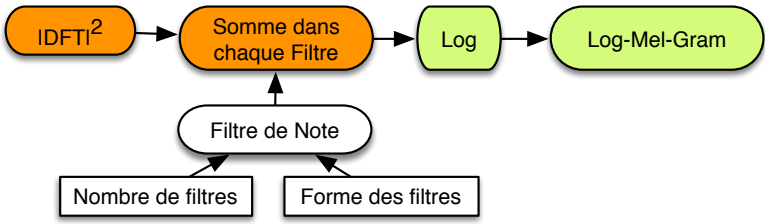
Example of the computation of MFCCs

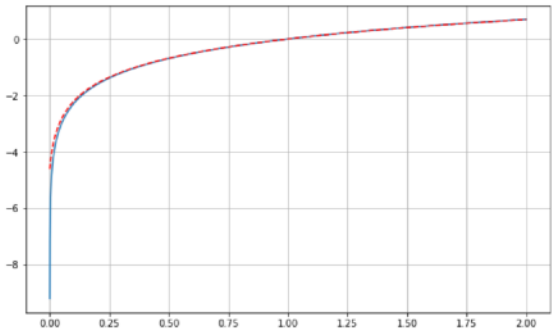


Audio features examples
Mel Frequency Cepstral Coefficients (11)

Variation for the case of DNN inputs: the Log-Mel-gram

- In the **Cepstrum**
  - the DCT is used to separate the contribution of the source and the filter
- In the **Mel Frequency Cepstral Coefficients**
  - the Mel-bands already mostly represent the filter contribution
  - the DCT is mostly used to de-correlated the dimensions
    - (latter used in GMM:diagonal covariance matrix Σ)
- **Log-Mel-Gram**
  - When using DNN, we don't need such a de-correlation of the inputs
  - we then bypass the DCT of the MFCC → Log-Mel-Gram
  - Tricks: to avoid singularity of $\log(x)$ → replace log by $\log(1+\gamma x)$ with $\gamma = 100$





ASR: Traditional approach
Audio features

- **MFCC** : Mel-Frequency Cepstral Coefficients
  - a representation of the real cepstral of a windowed shorttime signal derived from FFT
  - difference from the real cepstral is that a nonlinear frequency scale (Mel scale based on the human ear scale) is used, which approximates the behaviour of the auditory system
  - these coefficients are robust and reliable to variations according to speakers and recording conditions
- **PLP** : Perceptual Linear prediction
  - models the human speech based on the concept of psychophysics of hearing
  - discards irrelevant information of the speech and thus improves speech recognition rate
  - identical to LPC except that its spectral characteristics have been transformed to match characteristics of human auditory system.
- **RASTA** features
  - the rate of change of nonlinguistic component in speech often lies outside the typical rate of change of the vocal tract shape. The relative spectral (RASTA) technique takes advantage of this fact.
  - suppresses the spectral components that change more slowly or quickly than the typical range of change of speech.
- **PNCC** : Power-Normalized Cepstral Coefficients

RASTA-PLP : RelAtive SpecTrAl Perceptual Linear prediction

- **Critical band Integration**
  - Bark band
- **RASTA filtering**
  - applies a band-pass filter to the energy in each sub band frequency to smoothen the short-term noise variations and to remove any constant offset resulting from the static spectral coloration of the speech
- **Equal loudness pre-emphasis**
  - human hearing not equally sensitive at different frequencies; pre-emphasize equal-loudness relation $E_{pre}(\omega)$ is applied
- **Intensity-loudness power law**
  - simulates the nonlinear relation between the intensity of sound and its perceived loudness
  - applying cubic-root amplitude compression
- **Auto-regressive coefficients**
  - approximating the spectrum using all-pole
  - Inverse Fast Fourier Transform (IFFT) is applied. Then, cepstral coefficients are calculated from autoregressive coefficients

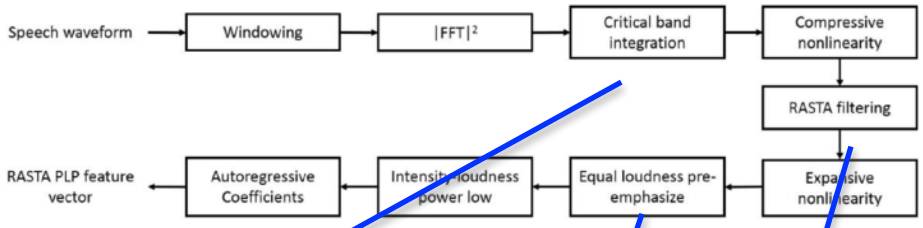
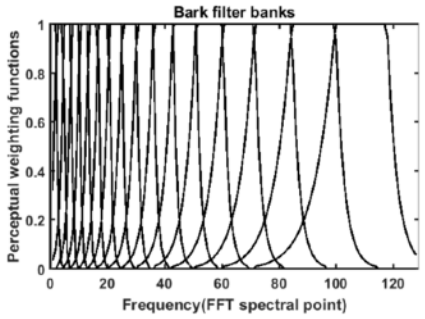


Figure 2-11: Block digram of RASTA-PLP system

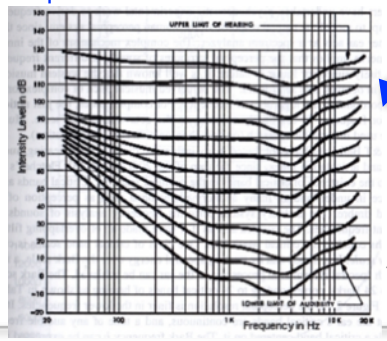
Critical bands



RASTA filtering

$$H(z) = 0.1z^4 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}}$$

Equal loudness contour



$$E_{pre}(\omega) = \frac{(\omega^2 + 56.8 \times 10^6)\omega^4}{(\omega^2 + 6.3 \times 10^6)^2 \times (\omega^2 + 0.38 \times 10^9)}$$

[H. Hermansky "**Perceptual linear predictive (PLP) analysis of speech**", In JAES, 1990] LINK

PNCC: Power-Normalized Cepstral Coefficients

- Gammatone filters
- Medium-Time Power Calculation
- Asymmetric Noise Suppression with Temporal Masking
- Time-Frequency normalization
- Mean power normalization
  - human = automatic gain control

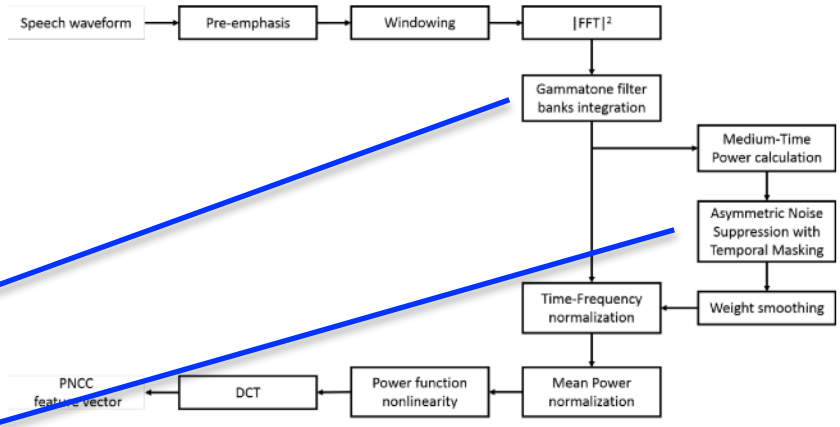


Figure 2-17: Block diagram of PNCC system

Gammatone

$$g(t) = \hat{a}t^{g-1}e^{-2\pi\hat{b}(f_c)t}\cos(2\pi f_c t + \phi)$$

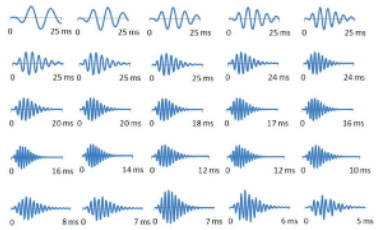


Figure 2-15: Set of 25 gammatone impulse responses with center frequencies from 100 Hz to 4 KHz

Temporal masking

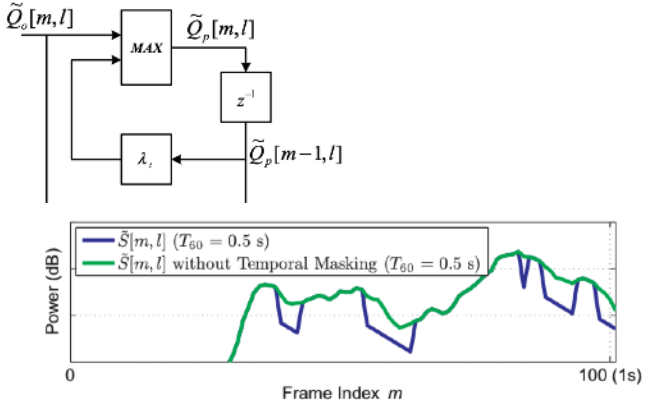


Fig. 5. Block diagram of the components that accomplish temporal masking

Mean Power nomalization

$$\mu[m] = \lambda_\mu \mu[m-1] + \frac{(1-\lambda_\mu)}{L}\sum_{l=0}^{L-1} T[m,l]$$

$$U[m,l] = k\frac{T[m,l]}{\mu[m]}$$

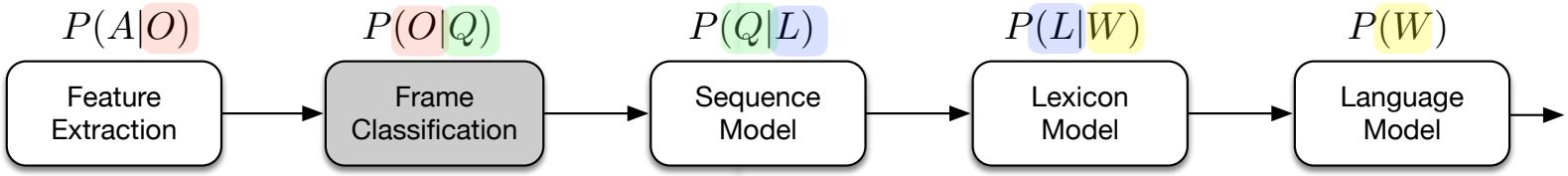
[Kim and Stern "**Power-Normalized Cepstral Coefficients (PNCC) for Speech Recognition**", In IEEE TASLP, 2016] LINK

Audio features

← END

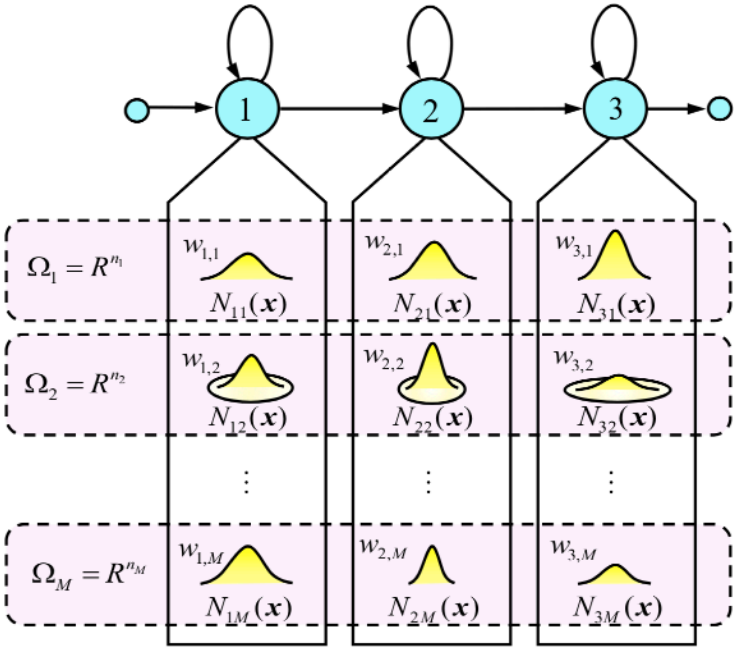
ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



- $P(O|Q) = P(Features|States)$
- **Frame Classification**
  - Gaussian Mixture Model (GMM)
    - Output probability is modeled by a mixture of Gaussian distributions

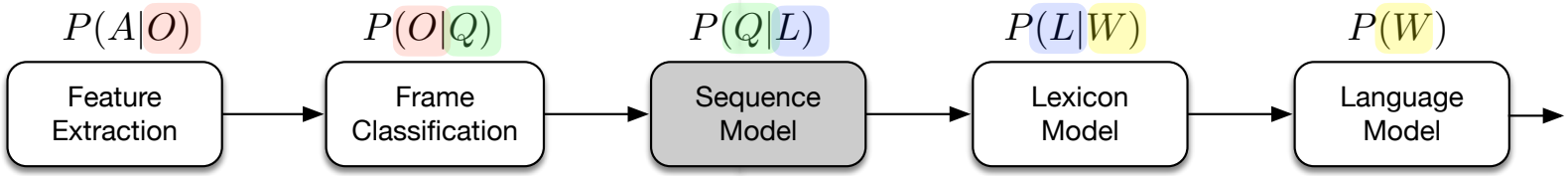
$$b_q(o_t) = b(o_t|q_t)$$
$$= \sum_{m=1}^{M} w_m \mathcal{N}(o_t|\mu_m, \Sigma_m)$$



source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

ASR: the traditional MFCC/GMM/HMM approach

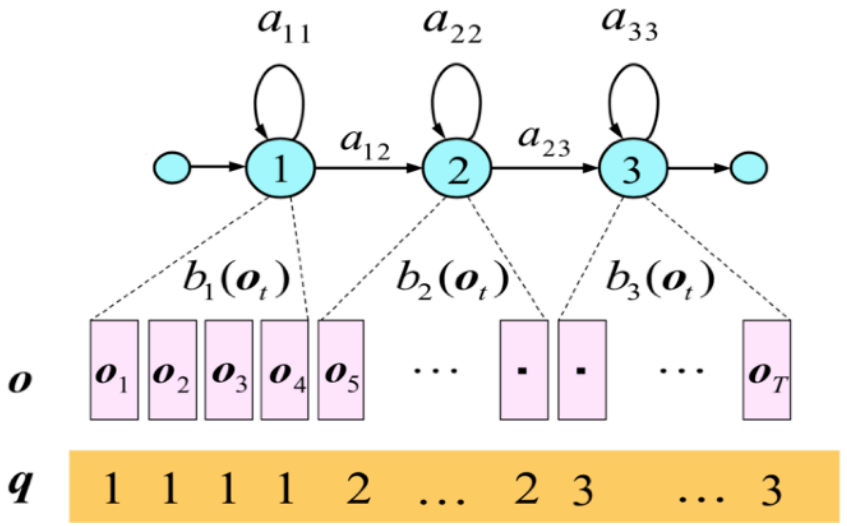
Traditional approach: GMM-HMM



- $P(Q|L) = P(States|Phonemes)$
- **Sequence model**
  - Hidden Markov Models (HMM)
    - The Markov chain whose state sequence is unknown

      a_{ij}: state transition probability

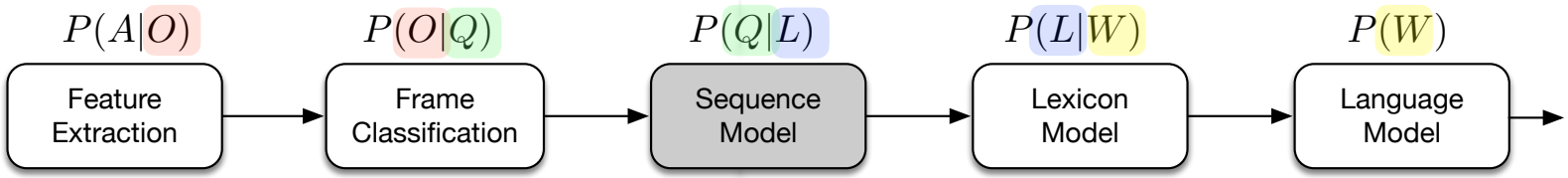
      $b_q(o_t)$: output probability



source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

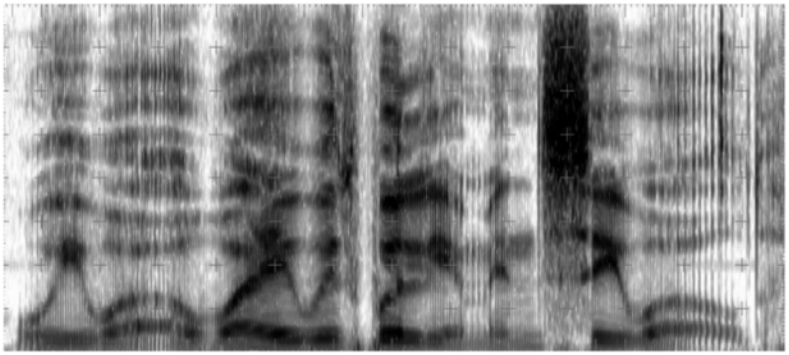
ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



- $P(Q|L) = P(States|Phonemes)$
- **Sequence model**
  - Hidden Markov Models (HMM)

- Context Dependent Model
  - "We **w**ere away with William in Sea **W**orld"

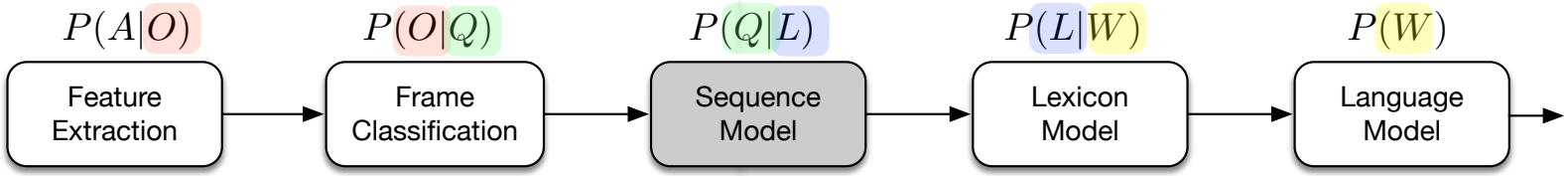


  - Realization of **w** varies but similar patterns occur in the similar context

source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

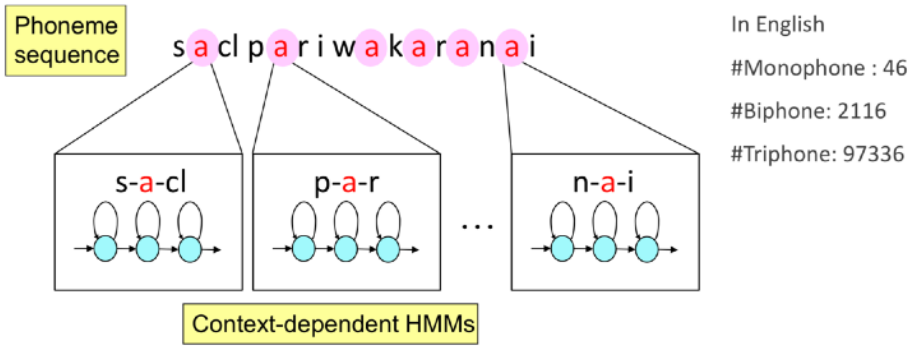
ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



- $P(Q|L) = P(States|Phonemes)$
- **Sequence model**
  - Hidden Markov Models (HMM)

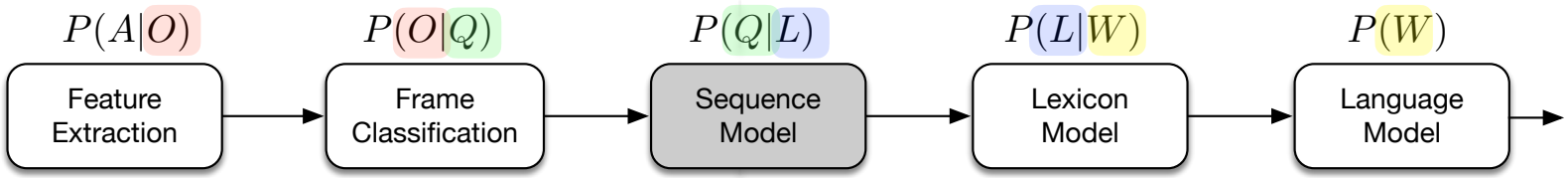
- Context Dependent Model



source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

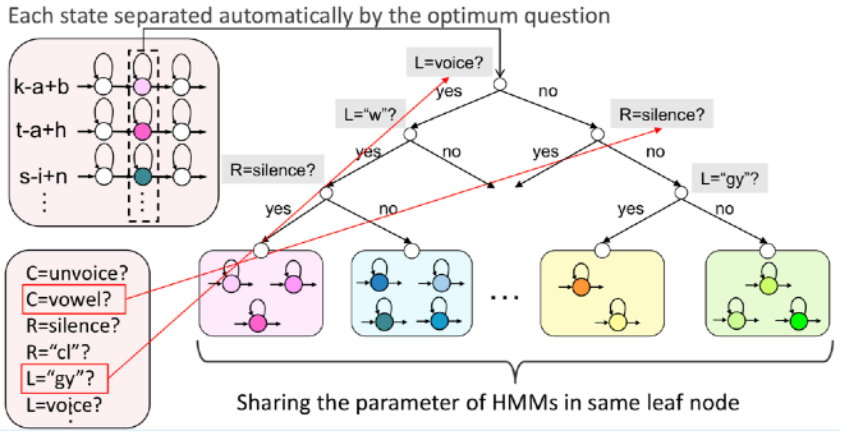
ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



- $P(Q|L) = P(States|Phonemes)$
- **Sequence model**
  - Hidden Markov Models (HMM)

- Decision Tree-based State Clustering



source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

Hidden Markov Model (HMM)

START →

Hidden Markov Model (HMM)

- Markov model
  - Andreï A. Markov (1856-1922): Russian mathematician

- Markov chain:
  - a stochastic process with discrete time t which can be in discrete states S_i $i \in \{1,\ldots,I\}$
  - q_t is the value of the state at time t

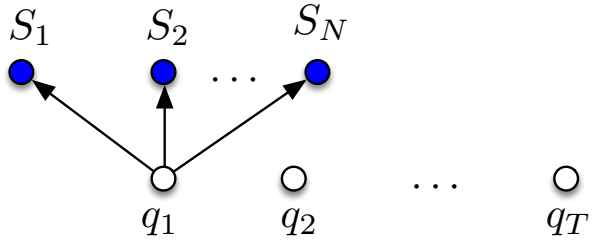
- First order Markov chain:
  - the prediction of the current state only depends on the previous time

$$p(q_t \mid q_{t-1}, q_{t-2} \cdots q_0) = p(q_t \mid q_{t-1})$$

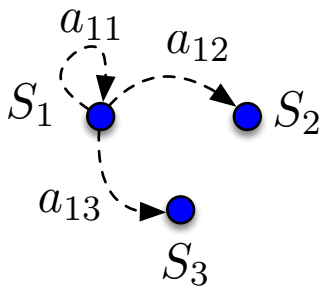
source: https://fr.wikipedia.org/wiki/Andre%C3%AF_Markov_(math%C3%A9maticien)

Observed Markov model

- The system is in one of a set of N distinct **states** $S_1, S_2, \ldots, S_N$
- We denote the **time** instants as $t = 1, 2, \ldots, T$
- We denote the actual state at time t as q_t
- We denote the sequence of actual state $Q = q_1, q_2, \ldots, q_T$



- Discrete first order Markov chain
  - $P(q_t = S_j \,|\, q_{t-1} = S_i, q_{t-2} = S_k, \ldots) = P(q_t = S_j \,|\, q_{t-1} = S_i)$
- **Transition probability** is independent of time
  - $a_{ij} = P(q_t = S_j \,|\, q_{t-1} = S_i) \quad 1 \leq i, j \leq N$
    - with $a_{ij} \geq 0$
    - $\sum_j a_{ij} = 1$



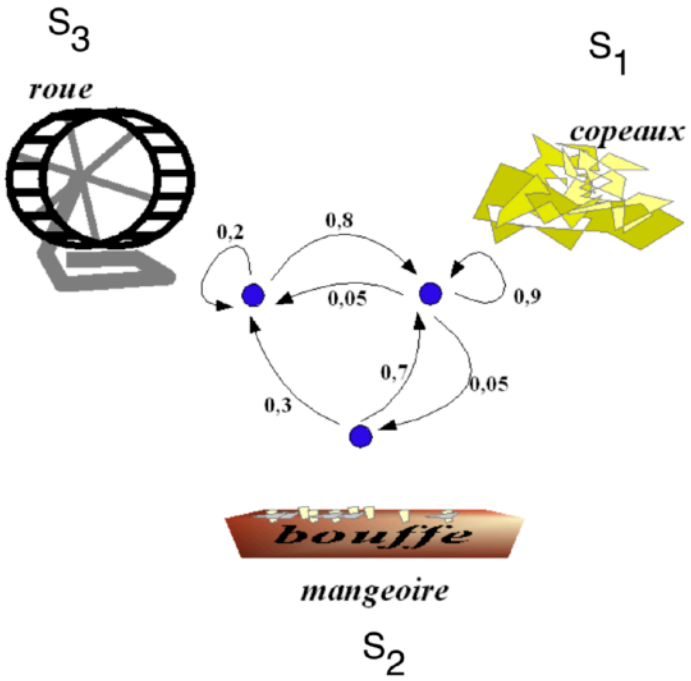
- **Initial state distribution** $\pi = \{\pi_i\}$ with
  - $\pi_i = P(q_1 = S_i)$

Observed Markov model example

- Doudou the hamster has 3 states during a day
  - he sleeps:
    - he is in state S_1 (sawdust shavings)
  - he eats:
    - he is in state S_2 (feeder)
  - he does exercice:
    - he is in state S_3 (hamster wheel)

- We can represent the succession of states a transition matrix $A = (a_{ij})$ between states

$$A = \begin{pmatrix} 0.9 & 0.05 & 0.05 \\ 0.7^{(i=2,j=1)} & 0 & 0.3 \\ 0.8 & 0 & 0.2 \end{pmatrix}$$

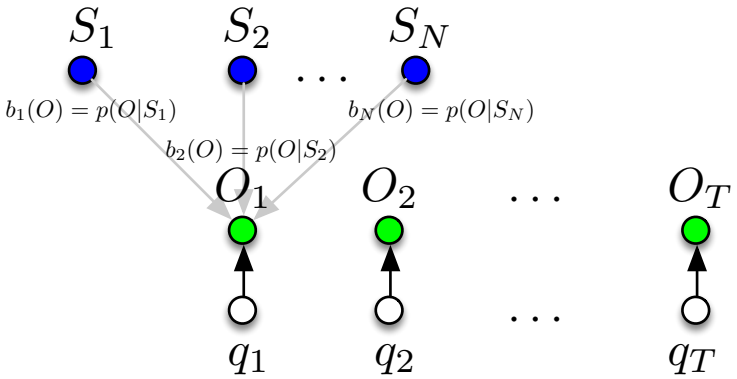


Modèle de Markov d'une journée de Doudou le hamster

source: https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markov

Hidden Markov model

– We do not observe directly the actual states $Q = q_1, q_2, \ldots, q_T$ but an emission of those
  - a sequence of **observations** $O = O_1, O_2, \ldots, O_T$



– Discrete Observations
  - **Set of symbols** v_k
  - $B = \{ b_j(k)\}$
    – with $b_j(k) = P(O_t = v_k \,|\, q_t = S_j)$
    the probability of observing symbol k in state j

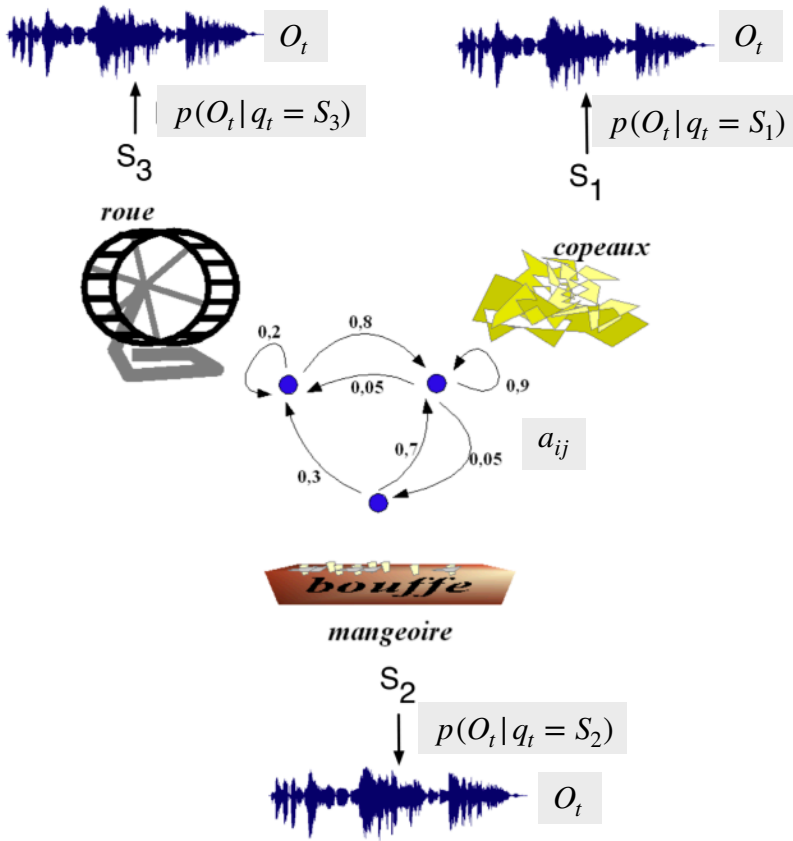
– Continuous Observation Densities
  - Gaussian Mixture Model
  - $b_j(O) = \sum_m c_{jm} \mathcal{N}(O, \mu_{jm}, \Sigma_{jm})$

– The observation is a probabilistic function of the state S_j

– We denote by $\lambda = \{A, B, \pi\}$ the set of elements defining an HMM

Hidden Markov model example

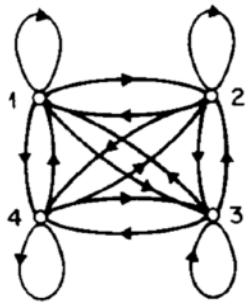
- In a hidden Markov model, we do not observe directly the states $Q = q_1, q_2, \ldots, Q_T$, they are "hidden"
- we observe an emission of the states q_t
  - example: we observe the sound $O = O_1, O_2, \ldots, O_T$ made by Doudou the hamster
- For each state, we can define
  - an emission probability given state S_j: $b_j(O_t) = p(O_t \mid q_t = S_j)$



Hidden Markov Model (HMM)

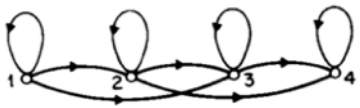
Various topologies of Markov models

- Ergodic/Fully Connected HMMs
  - a HMM allowing for transitions from any emitting state to any other emitting state



$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

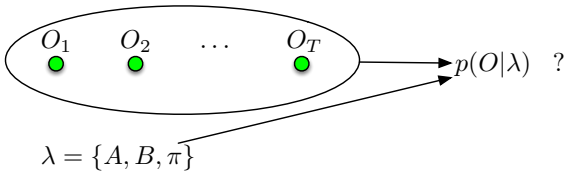
- Left-Right HMMs
  - an HMM where the transitions are not allowed to states which indices are lower than the current state: $a_{ij} = 0, \quad j < i$



$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

source: L. Rabiner, 1989.

Problem 1. Likelihood of a model (Forward algorithm)

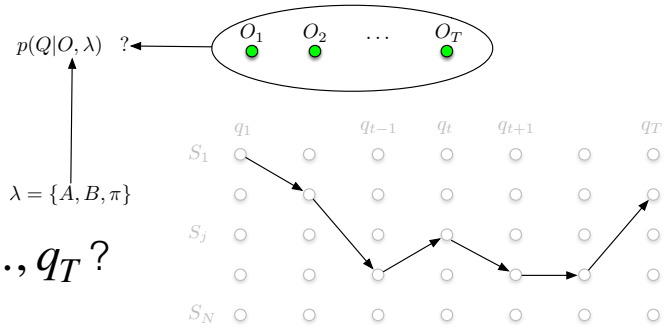


- Given
  - a sequence of observations $O = O_1, O_2, \dots, O_T$
  - a model $\lambda = \{A, B, \pi\}$
- What is the probability that the model has generated O ?
  - $P(O \mid \lambda)$?
- Application example:
  - does the observation sequence O corresponds to
    - the model λ_1 {sleep/eat/exercice} of Doudou the hamster ?
    - the model λ_2 {sleep/metro/work} of Bill the employee ?
- **Problem:**
  - We do not know through which states $Q = q_1, q_2, \dots, q_T$ the model has passed
    - We need to consider all possibilities

[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Problem 2. Decoding of the best sequence (Viterbi algorithm)

- Given
  - a sequence of observations $O = O_1, O_2, \ldots, O_T$
  - a model $\lambda = \{A, B, \pi\}$



- What is the corresponding sequence of states $Q = q_1, q_2, \ldots, q_T$?
  - $Q^\star = \arg\max_Q P(Q \mid O, \lambda)$?

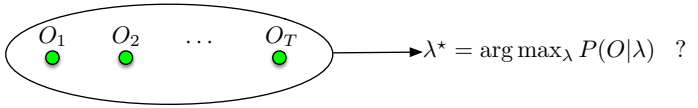
- Application example:
  - if we observe the sequence of sounds O produced by Doudou and given its model λ {sleep/eat/exercice}, what is the corresponding sequence of activities Q of Doudou ?

- **Problem**:
  - Find the most **optimal** sequence Q
    - Several possible definitions for **optimality**

[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Problem 3. Training a HMM model (Forward-Backward, Baum-Welch algorithm)

- Given
  - a sequence of observations $O = O_1, O_2, \ldots, O_T$



- What is the model $\lambda^\star$ that maximises the likelihood of the observations:
  - $\lambda^\star = \arg\max_\lambda P(O \mid \lambda)$
  - i.e. find the parameters $\lambda = \{A; B, \pi\}$ that maximises the likelihood of the observation's sequence O

- Application example:
  - find the parameters of the model λ of Doudou the hamster ?

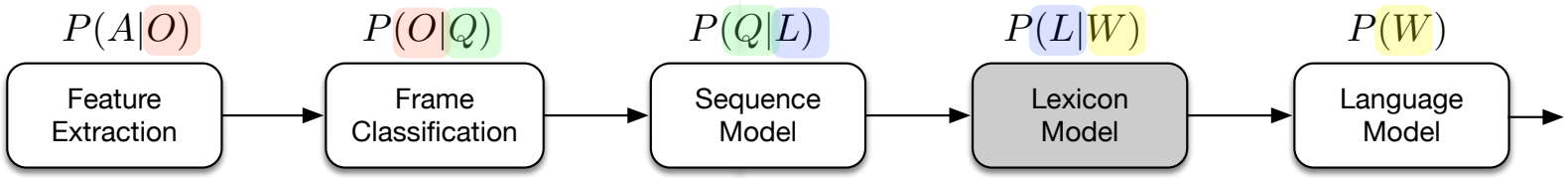
[L. Rabiner. "A tutorial on hidden markov model and selected applications in speech". Proceedings of the IEEE, 77(2):257–285, 1989.] LINK

Hidden Markov Model (HMM)

← END

ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



- $P(L|W) = P(Phonemes|Words)$
- **Lexicon model**
  - Lexical modelling forms the bridge between the acoustic and language models
  - Prior knowledge of language
  - Mapping between words and the acoustic units (phoneme is most common)

Deterministic

Word	Pronunciation
TOMATO	t ah m aa t ow
	t ah m ey t ow
COVERAGE	k ah v er ah jh
	k ah v r ah jh

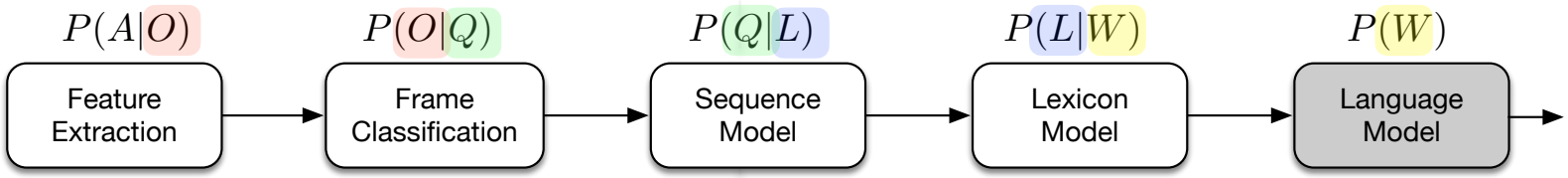
Probabilistic

Word	Pronunciation	Probability
TOMATO	t ah m aa t ow	0.45
	t ah m ey t ow	0.55
COVERAGE	k ah v er ah jh	0.65
	k ah v r ah jh	0.35

source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

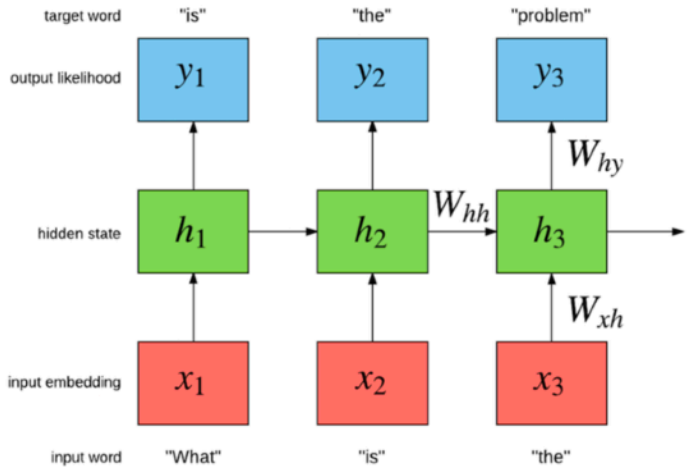
ASR: the traditional MFCC/GMM/HMM approach

Traditional approach: GMM-HMM



- $P(W) = P(Words)$
- **Language model**
  - Language model is a probabilistic model used to
    - Guide the search algorithm (predict next word given history)
    - Disambiguate between phrases which are acoustically similar - "Great wine" vs "Grey twine"
  - It assigns probability to a sequence of tokens to be finally recognized
  - N-gram model $P(W_N \mid w_1, w_2, \ldots, w_{N-1})$

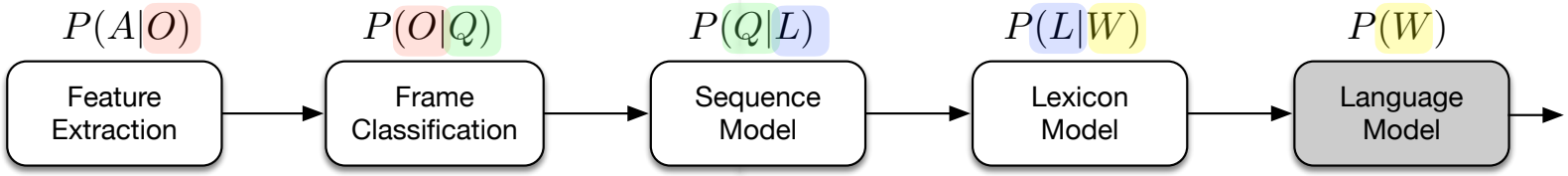
- using bi-grams, tri-grams
- using a Recurrent neural network



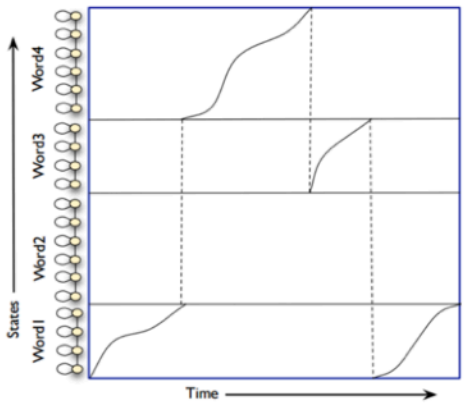
source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

ASR: the traditional MFCC/GMM/HMM approach

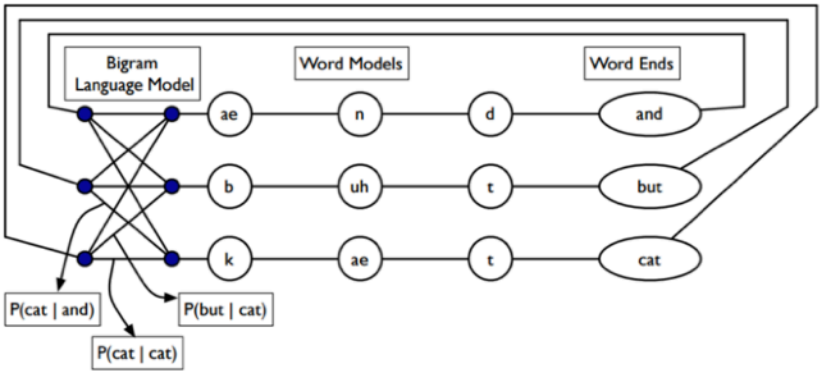
Traditional approach: GMM-HMM



- $P(W) = P(Words)$
- **Language model**
  - Decoding in Speech Recognition
    - Find most likely word sequence of audio input



Without language models



With language models

source : Yuchen Fan, Matt Potok, Christopher Shroba, "CS 598 LAZ : Cutting-Edge Trends in Deep Learning and Recognition"

ASR: Deep-learning approach

Deep Learning reminders

START →

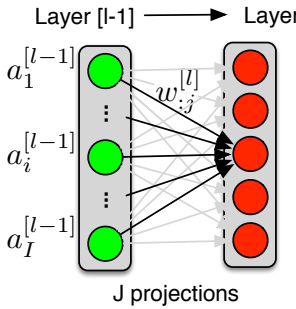
Architectures

Multi-Layer-Perceptron (MLP)

- Extension of the Perception

- Neurons are organized into Layers which are Fully-Connected
  - $a_j^{[l]}$ is connected to all neurons $a_i^{[l-1]}$
  - connection done through
    - multiplications by weights $w_{ij}^{[l]}$,
    - addition of a bias $b_j^{[l]}$,
    - passing through non-linear activation $g(.)$
  - Each $\mathbf{w}_j^{[l]}$ defines a specific projection of the previous layer

Fully Connected Neural Network



$$a_j^{[l]} = g(\sum_i a_i^{[l-1]} w_{ij}^{[l]} + b_j^{[l]})$$

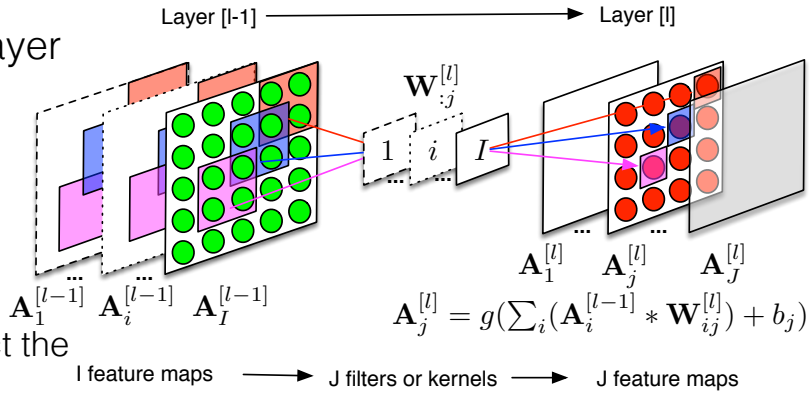
$$\mathbf{a}^{[l]} = g(\mathbf{a}^{[l-1]} \mathbf{W}^{[l]} + \mathbf{b}^{[l]})$$

[Frank Rosenblatt, "**The perceptron, a perceiving and recognizing automaton**" Project Para. Cornell Aeronautical Laboratory, 1957] LINK

Convolutional Neural Network (CNN)

- (1) Assume local connectivity of neurons of a given layer (vision: nearby pixels more correlated)
  - Each region (x, y) of $\mathbf{A}_i^{[l-1]}$ is projected individually
  - Results= feature map noted $\mathbf{A}_{i \to j}^{[l]}$ for the j^{th} projection
- (2) Add a **parameter sharing** property
  - for a given j, the same projection $\mathbf{W}_{ij}^{[l]}$ is used to project the different regions $(x, y) \rightarrow$ the weights are shared
  - apply the same projection to the various regions (x, y)
- (1)+(2) $\rightarrow$ convolution operator
- In practice
  - several input feature maps $\mathbf{A}_{1 \ldots i \ldots I}^{[l-1]}$, projection with a tensor $\mathbf{W}_{:j}^{[l]}$ (extends over I)
  - Results: feature map $\mathbf{A}_j^{[l]}$
  - Several projections: J different convolutions resulting in J output feature maps
- To reduce dimensionality, allows spatial invariance:
  - **max-pooling**

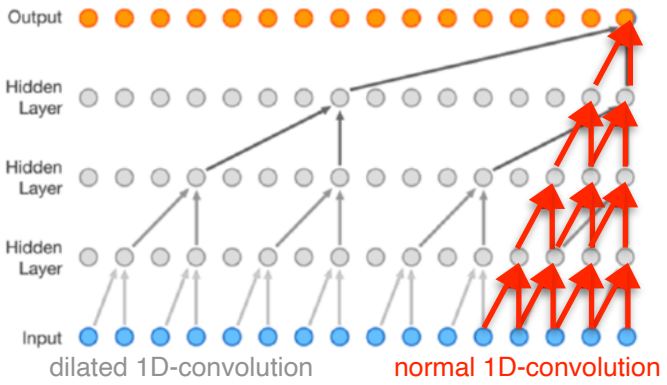
Convolutional Neural Network

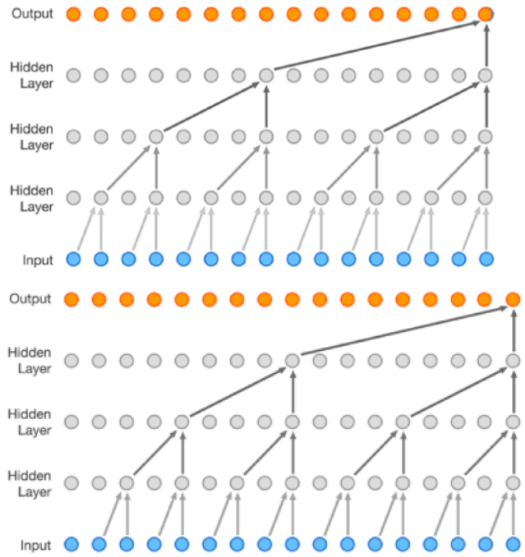


[K. Fukushima et al., "**Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition**". In Competition and cooperation in neural nets, 1982]
[Y. LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haner. "**Gradient- based learning applied to document recognition**". Proceedings of the IEEE, 86(11):2278–2324, 1998]

Temporal Convolutional Networks (TCN)

- Motivation: learn better projections than Fourier
- 1D-convolution applied on the raw audio waveform $x(n)$
  - filters $\mathbf{W}_j$ have only one dimension (time)
  - convolution is done over time
- Receptive field (RC)
  - portion of the input data to which a given neuron responds
  - images (256x256): only a few layers is necessary in CV to make the RC of a neuron cover the whole input image
  - audio (44100/sec): requires a huge number of layers
- 1D-Dilated- Convolutions
  - $(x \circledast_d w)(n) = \sum_{i=0}^{l-1} w(i)x(n-(d\,.\,i))$
  - the filters is convolved with the signal only considering one over d values





[A. van den Oord et al. "**Wavenet: A generative model for raw audio**", 2016] LINK

Temporal Convolutional Networks (TCN)

- 1D-Convolution
- + causality constraint
- + stacks two dilated-convolutions on top of each other (with weight-normalization, ReLu, DropOut)
- + with a parallel residual path

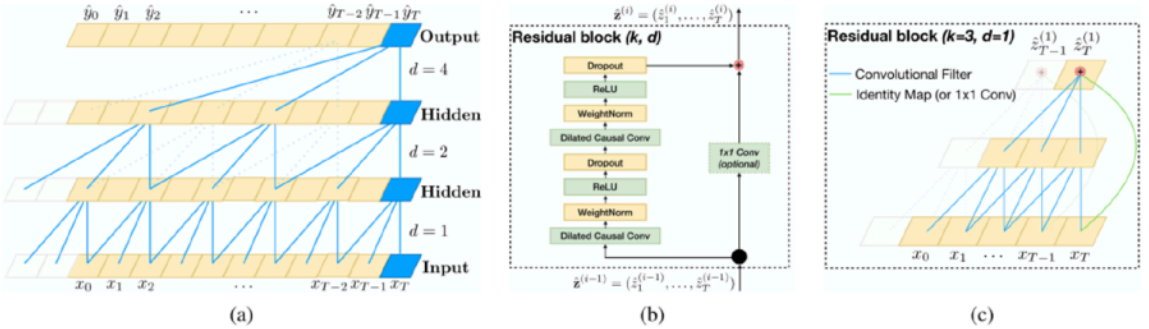


Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

[Shaojie Bai et al. "**An empirical evaluation of generic convolutional and recurrent networks for sequence modeling**", 2018] LINK

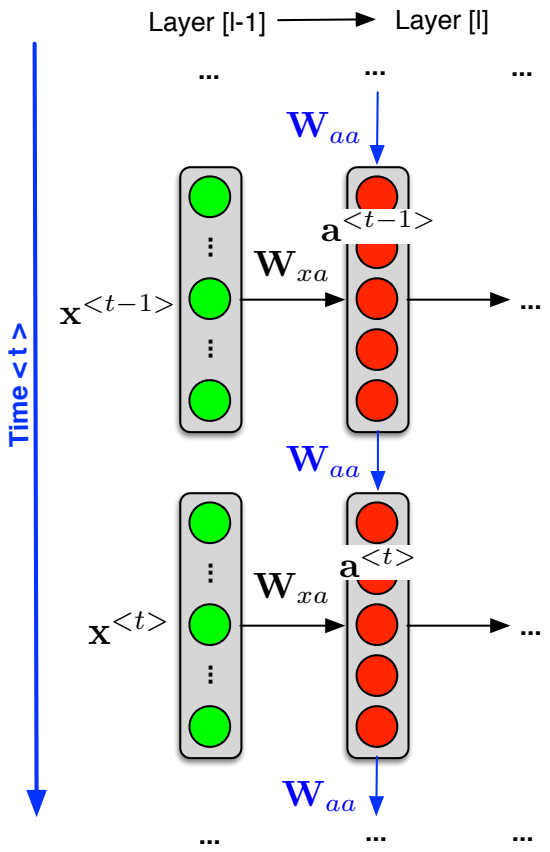
Recurrent Neural Network (RNN)

- RNNs take into account the sequential aspect of the data
  - RNNs have a memory that keeps track of previously proposed events of the sequence
  - Internal/hidden representation of the data at time t, $\mathbf{a}^{<t>}$ does not only depend on the input data $\mathbf{x}^{<t>}$ but also on the internal/ hidden representation at the previous time $\mathbf{a}^{<t-1>}$

- More sophisticated RNN cells
  - Long Short Term Memory (LSTM)
  - Gated Recurrent Units (GRU)

Recurrent Neural Network

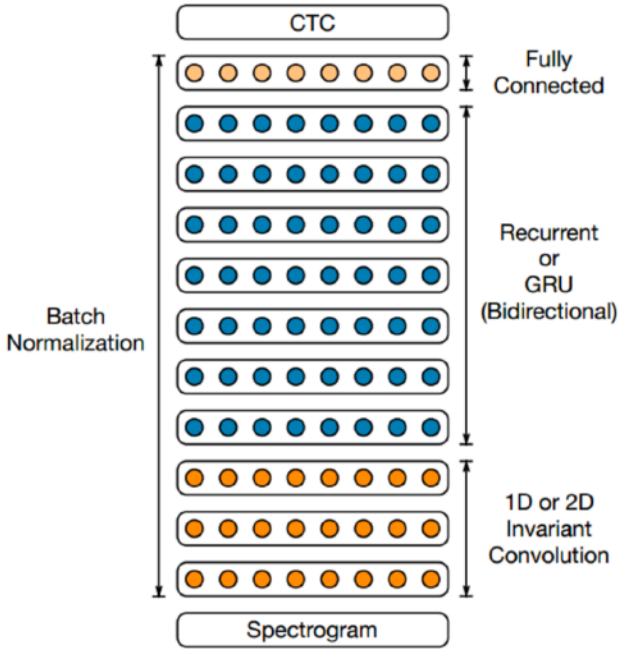
$$\mathbf{a}^{<t>} = g(\mathbf{x}^{<t>}\mathbf{W}_{xa} + \mathbf{a}^{<t-1>}\mathbf{W}_{aa} + \mathbf{b}_a)$$



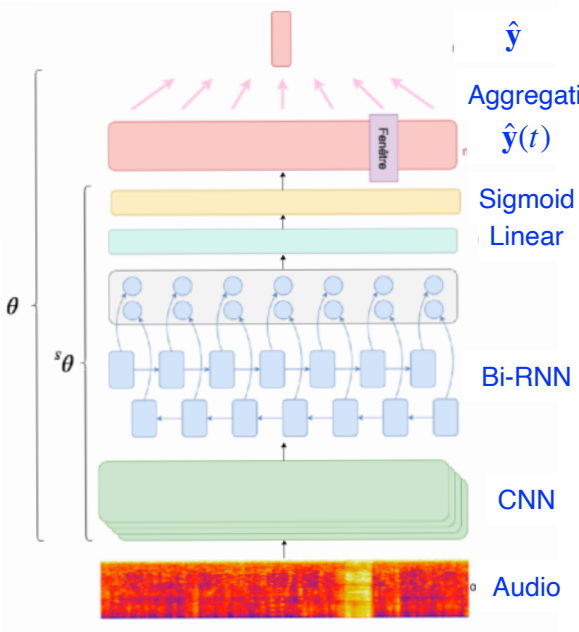
[David E Rumelhart et al. "**Learning representations by back-propagating errors**". Nature, 323(6088):533–536, 1986]
[Sepp Hochreiter and Jurgen Schmidhuber. "**Long short-term memory**". Neural computation, 9(8):1735–1780, 1997]
[Kyunghyun Cho et al. "**Learning phrase representations using RNN encoder-decoder for statistical machine translation**", 2014]

Typical audio architecture: RCNN = CNN followed by a RNN

- CNN (Conv1D or Conv2D) used for
  - feature extraction
- RNN used for
  - temporal smoothing
  - language model



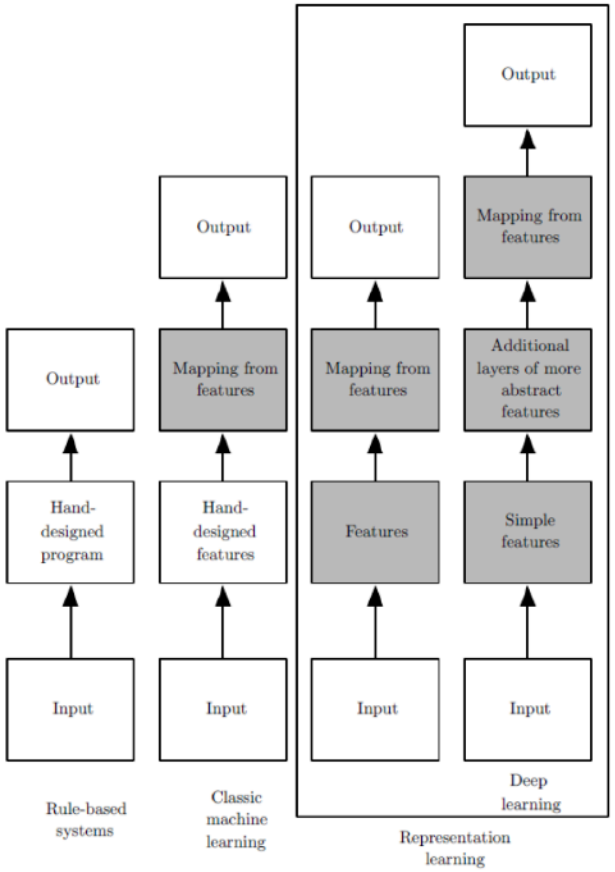
ASR Baidu Speech 2



DCASE Task 4, Turpault baseline

What is deep learning ?

Deep learning: learning hierarchical representations

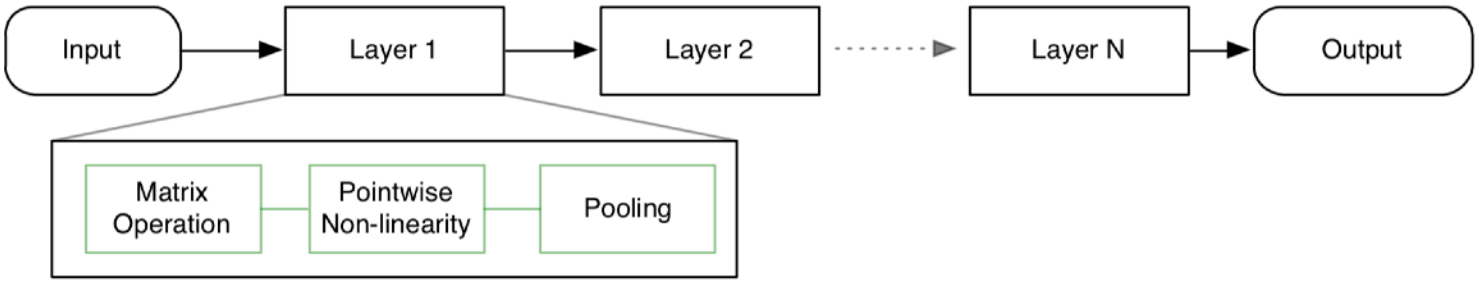


From Deep Learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

What is deep learning ?

Deep learning: learning hierarchical representations

- Deep learning is ...
  - Cascade of multiple layers, composed of a few simple operations
    - Linear algebra
    - Point-wise nonlinearities
    - Pooling



What is deep learning ?

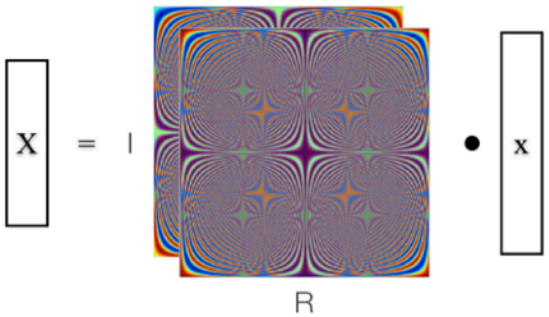
Deep learning: learning hierarchical representations

- Non-linearities enable complexity
  - Cascaded non-linearities allow for complex systems composed of simple, linear parts
    - The composite of two linear systems is just another linear system
    - The composite of two non-linear systems is an entirely different system

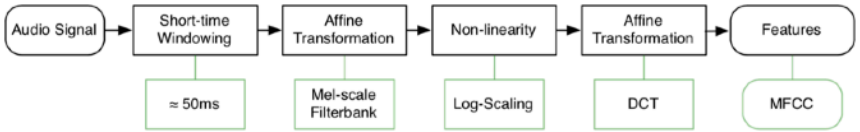
- DL can represent the algorithm of

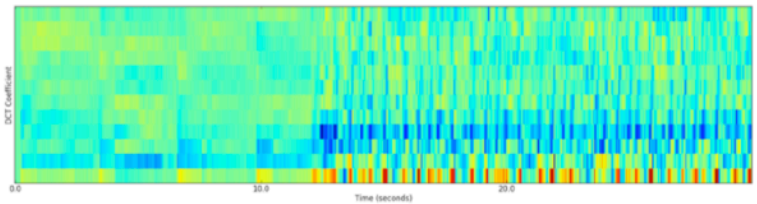
Fourier Transform

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}$$



Mel Frequency Cesptral Coefficients (MFCCs)





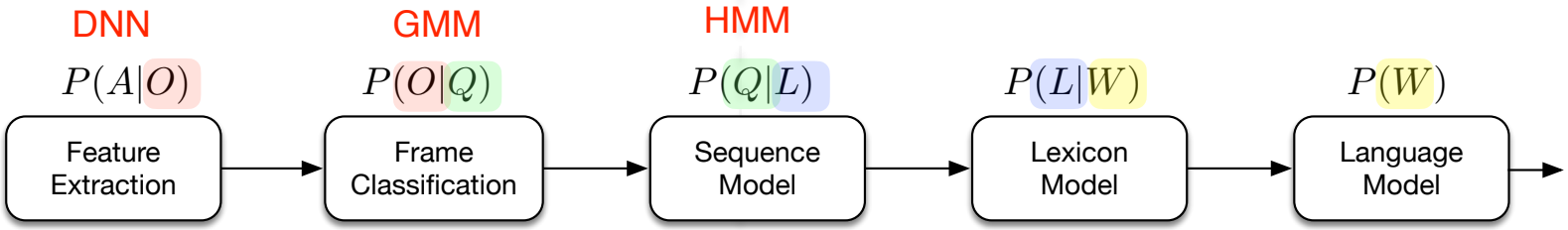
Deep Learning reminders

← END

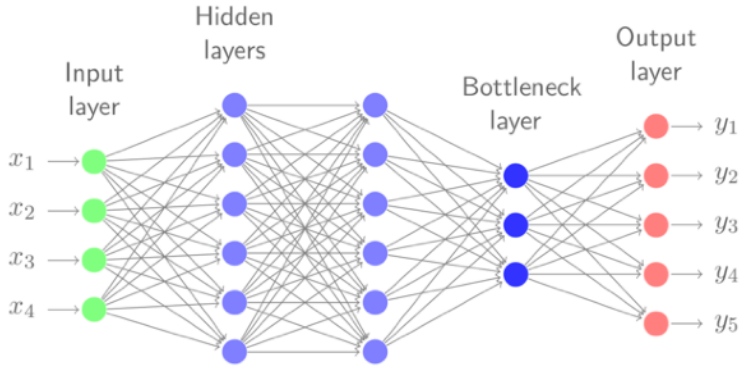
Architectures

ASR: Deep-learning approach

Deep learning approaches: 1) DNN ⇒ GMM ⇒ HMM



- **DNN / GMM / HMM**
  - Use neural networks to compute nonlinear feature representations
    - Trained audio features
    - Tandem, Bottleneck, DNN-derived features
    - Low-dimensional representation is modeled conventionally with GMMs
    - Allows all the GMM machinery and tricks to be exploited

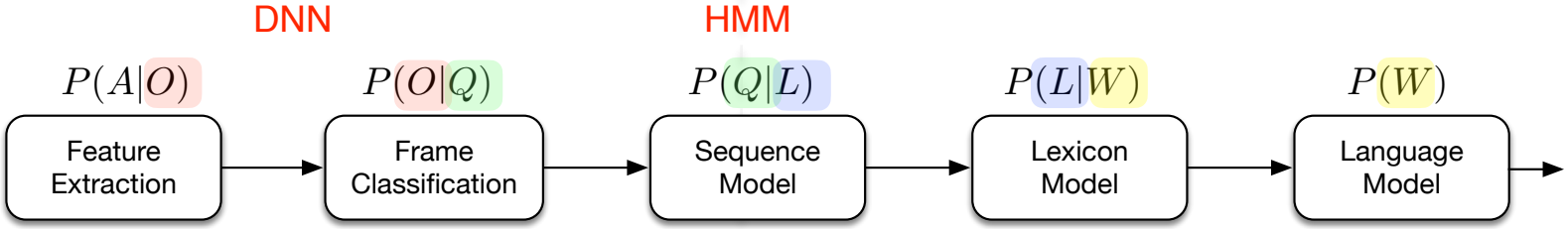


- **How?**
  - Train a neural network to discriminate classes.
  - Use output or a low-dimensional bottleneck layer representation as features.
- **TRAP :**
  - Concatenate PLP-HLDA features and NN features.

[H. Hermansky et al. "**Tandem connectionist feature extraction for conventional HMM systems**", 2000, IEEE ICASSP] LINK

ASR: Deep-learning approach

Deep learning approaches: 1) DNN ⇒ HMM



– **DNN / HMM**

- Use neural networks (DNN) instead of GMM to estimate phonetic unit probabilities
  - Perform frame classification using a DNN
  - Classify acoustic features for state labels
  - Take softmax output as a posterior $P(state \mid o_t) = p(q_t \mid o_t)$
  - Work as output probability in HMM

$$p(o_t \mid q_t) = \frac{p(q_t \mid o_t) p(o_t)}{p(q_t)}$$ where $p(q_t)$ is the prior probability for states

- Train the network as a classifier with a softmax across the phonetic units.
  - Train with cross-entropy
  - Softmax

$$y(i) = \frac{\exp(a_\theta(i))}{\sum_{j=1}^{N} \exp(a_\theta(j))}$$

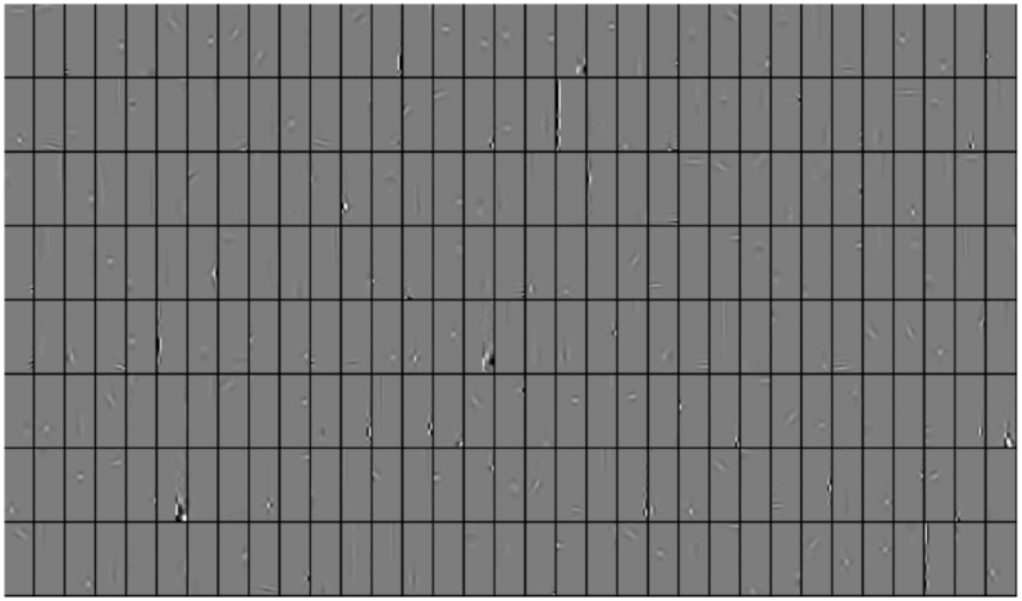
  - will converge to posterior across phonetic states $p(q_i \mid o_t)$

[G. Hinton, et al. "**Deep Neural Networks for Acoustic Modeling in Speech Recognition**". 2012, Signal Processing Magazine] LINK

Deep learning approaches: 1) DNN ⇒ HMM

– **Input features**

- Neural networks can handle high-dimensional features with correlated features
- Use 26 stacked filterbank inputs of 40-dimensional Mel-spaced each
- Example filters learned in the first layer of a fully-connected network :



(33 x 8 filters. Each subimage 40 frequency vs 26 time.)

ASR: Deep-learning approach

Deep learning approaches: 1) DNN ⇒ HMM

- **Neural network architectures**
  - Fully connected
  - Convolutional networks (CNNs)
  - Recurrent neural networks (RNNs)
    - LSTMs
    - GRUs

ASR: Deep-learning approach

Deep learning approaches: 1) DNN ⇒ HMM

- Pre-training
  - Unsupervised : stacked restricted Boltzmann machine (RBM)
  - Supervised : iteratively adding layers from shallow model
- Training
  - Maximum cross entropy for frames
- Fine-tuning
  - Maximum mutual information for sequences
- Comparison on different large datasets

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

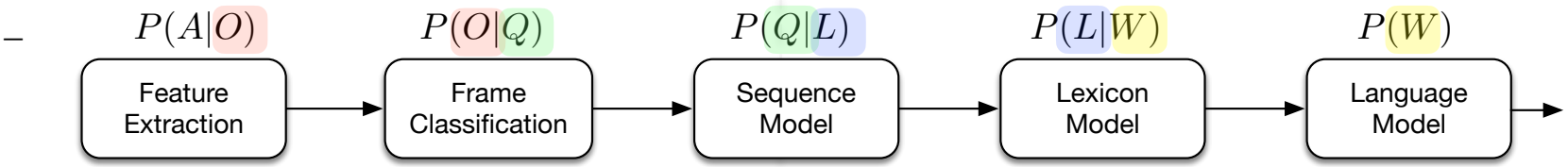
ASR: Deep-learning approach

Deep learning approaches: 1) DNN ⇒ HMM

- DNN-HMM vs. GMM-HMM
  - Deep models are more powerful
    - GMM assumes data is generated from single component of mixture model
    - GMM with diagonal variance matrix ignores correlation between dimensions
- Deep models take data more efficiently
  - GMM consists with many components and each learns from a small fraction of data
- Deep models can be further improved by recent advances in deep learning

Deep learning approaches: 3) End-to-End models: LSTM CTC

CNN / RNN / FC CTC

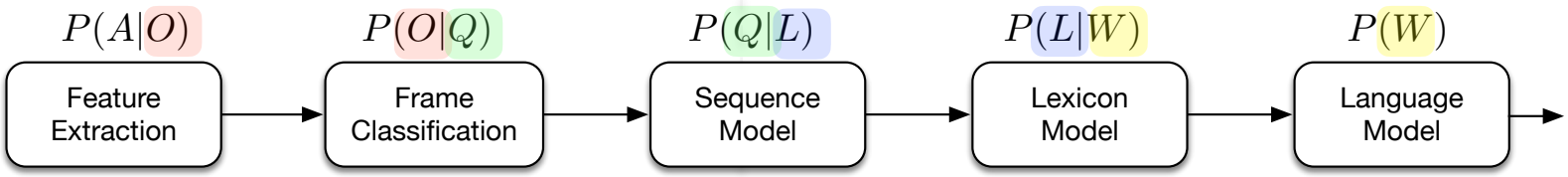


[Hannun et al. "**Deep Speech: Scaling up end-to-end speech recognition**", 2014] LINK

ASR: Deep-learning approach

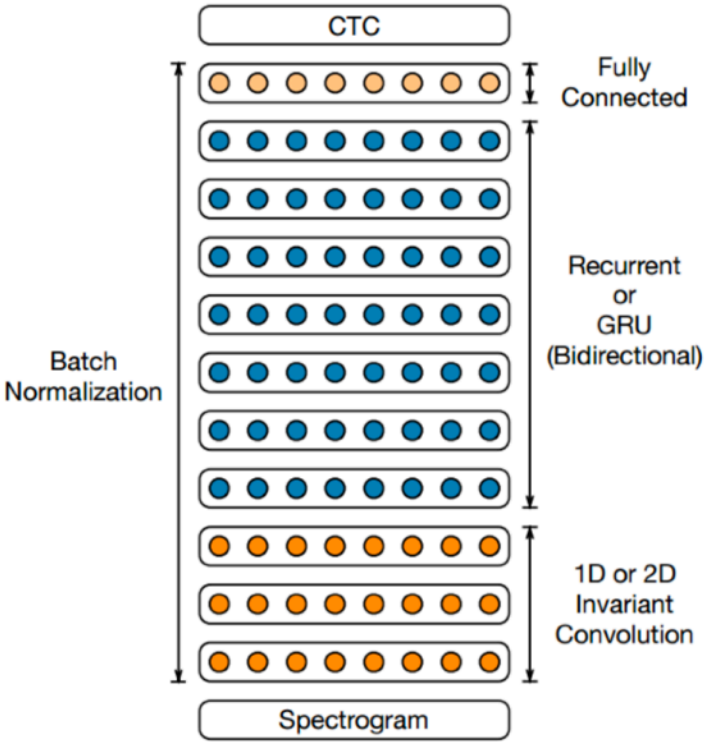
Deep learning approaches: 3) End-to-End models: Deep Speech 2 (Baidu)

CNN / RNN / FC CTC



- 3 layers of 2D-invariant convolution
- 7 layers of bidirectional simple recurrence 100M para
- Word error rates

Test set	Deep speech 2	Human
WSJ eval'92	3.60	5.03
WSJ eval'93	4.98	8.08
LibriSpeech test-clean	5.33	5.83
LibriSpeech test-other	13.25	12.69



[Hannun et al. "**Deep Speech: Scaling up end-to-end speech recognition**", 2014] LINK

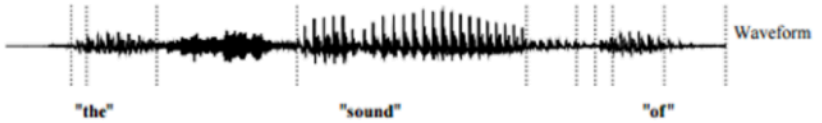
Connectionist Temporal Classification

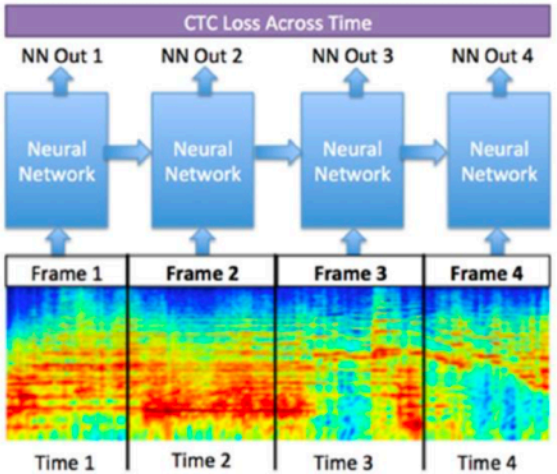
START →

Connectionist Temporal Classification

Goal

- end-to-end training of a neural network
  - FROM audio/spectrogram TO characters
- can be combined with a language model
- the neural network is usually a stack of bi-LSTM





Problem

- for training and decoding :
  - output Y (the characters) **need to be aligned with** input X (the audio/spectrogram)
- in practice input and output have different length ($T_y \neq T_x$)
- the manual alignment of X and Y is very costly (cannot at scale)

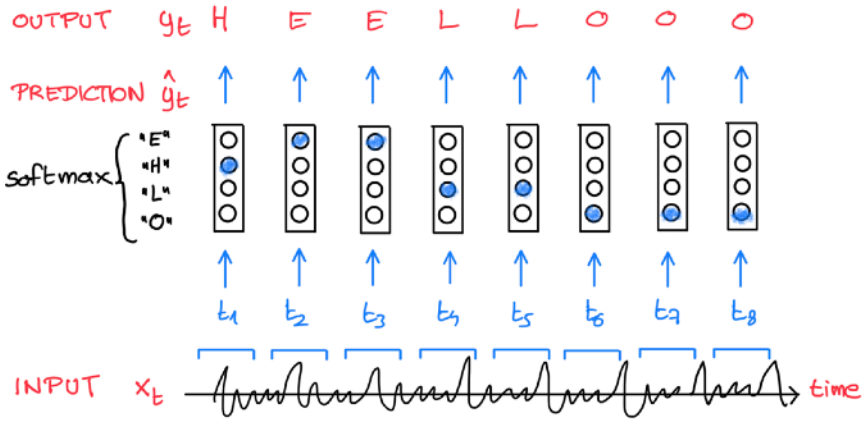
Solution

- Connectionist temporal classification
- Transform the output such that $T_y = T_x$

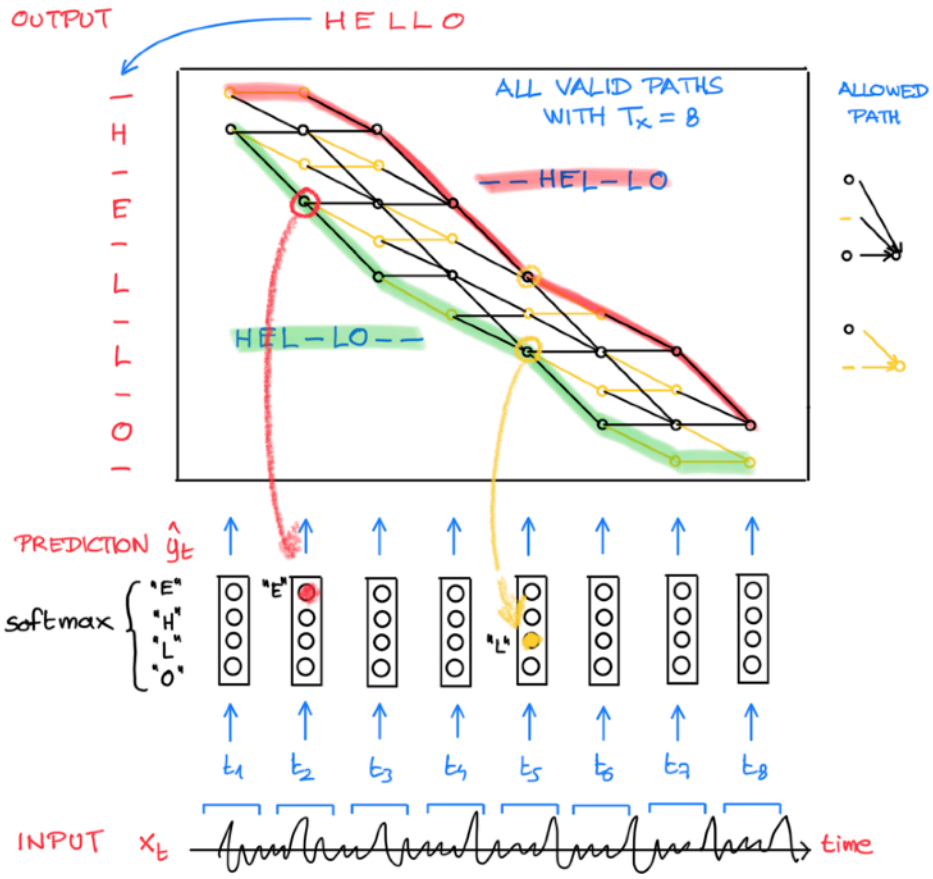
[A. Graves et al."Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks", ICML, 2006] Link
[W. Gharbieh, "Connectionist Temporal Classification, Labelling Unsegmented Sequence Data with RNN"] Link
[A. Hannun "Sequence Modeling With CTC"] https://distill.pub/2017/ctc/

Connectionist Temporal Classification

Strongly aligned



Weakly aligned data

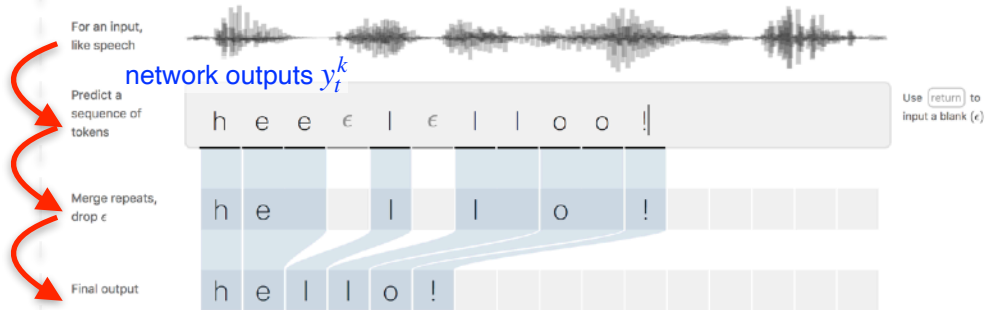


[A. Graves et al."Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks", ICML, 2006] Link

Connectionist Temporal Classification

Decoding

- **Neural network**
  - with input sequence $\mathbf{x} \in \mathbb{R}^m$ of length T and output sequence $\mathbf{y} \in \mathbb{R}^n$ of length T
  - $\mathbf{y} = f_\theta(\mathbf{x}) \quad (\mathbb{R}^m)^T \mapsto (\mathbb{R}^n)^T$,
- y_t^k: probability of observing output unit k at time t
  - y_t^k over the alphabet $L' = L \cup \{blank\}$
    - $L = \{a, b, \ldots, z\}$, the possible characters
    - L' = L+ a **blank** symbol ("ϵ" or "-")
- **Mapping function** $\mathcal{B} : \pi \in L'^T \mapsto l \in L^{\leq T}$
  - map a given **path** π of the sequence of network output to the final **labelling** l
  - how ? transforms the sequence by merging identical characters if not separated by "ϵ" or "-"
- **Examples** :
  - $\mathcal{B}$("hell-loo") = $\mathcal{B}$("hel-lo") = "hello"
  - $\mathcal{B}$("hellllоо") = "helo"
  - $\mathcal{B}$("cc-a-tt") = $\mathcal{B}$("c-a-t") = "cat"
  - $\mathcal{B}$("-c-a-t–") = $\mathcal{B}$("-c-a-t-") = "cat"
  - $\mathcal{B}$("c-aaa-at") = $\mathcal{B}$("c-a-at") = "caat"



Connectionist Temporal Classification

Decoding

– Get the most likely labelling l given an input sequence $\mathbf{x}$

$$\mathbf{l}^* = \arg\max_{\mathbf{l} \in L^{\leq T}} p(\mathbf{l}|\mathbf{x})$$

– Method-A : best path decoding

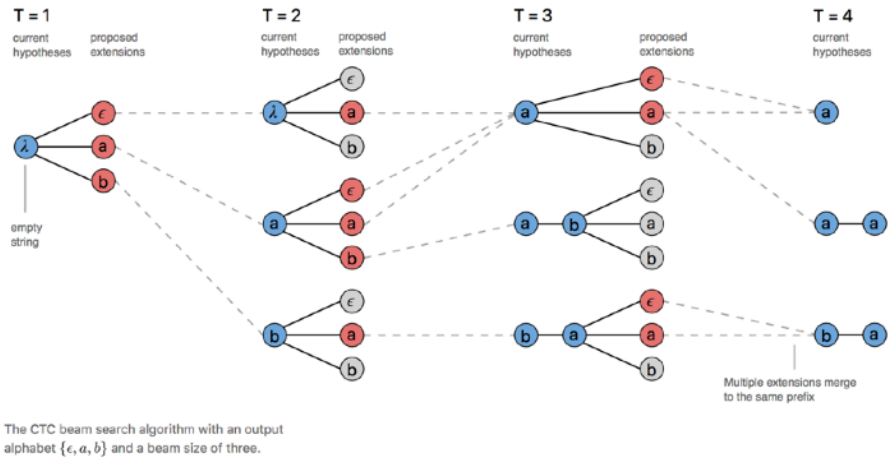
- assumption: the most probable path will correspond to the most probable labelling: $\mathbf{l} = \mathcal{B}(\pi^*)$
- π^* is just the concatenation of the most active outputs at every time-step

$$\pi^* = \arg\max_{\pi \in N^T} p(\pi|\mathbf{x})$$

- then collapse repeated characters and remove $blank$ to get $\mathbf{l}^* = \mathcal{B}(\pi^*)$

– Method-B : beam-search

- Problems of "best-path-decoding"
  - Assume the alignments [a,a,-] and [a,a,a] individually have lower probability than [b,b,b], but the sum of their probabilities is actually greater than that of [b,b,b]. The naive heuristic will incorrectly propose Y=[b] as the most likely hypothesis. It should have chosen Y=[a].
  - To fix this, the algorithm needs to account for the fact that [a,a,a] and [a,a,-] collapse to the same output.



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

Connectionist Temporal Classification

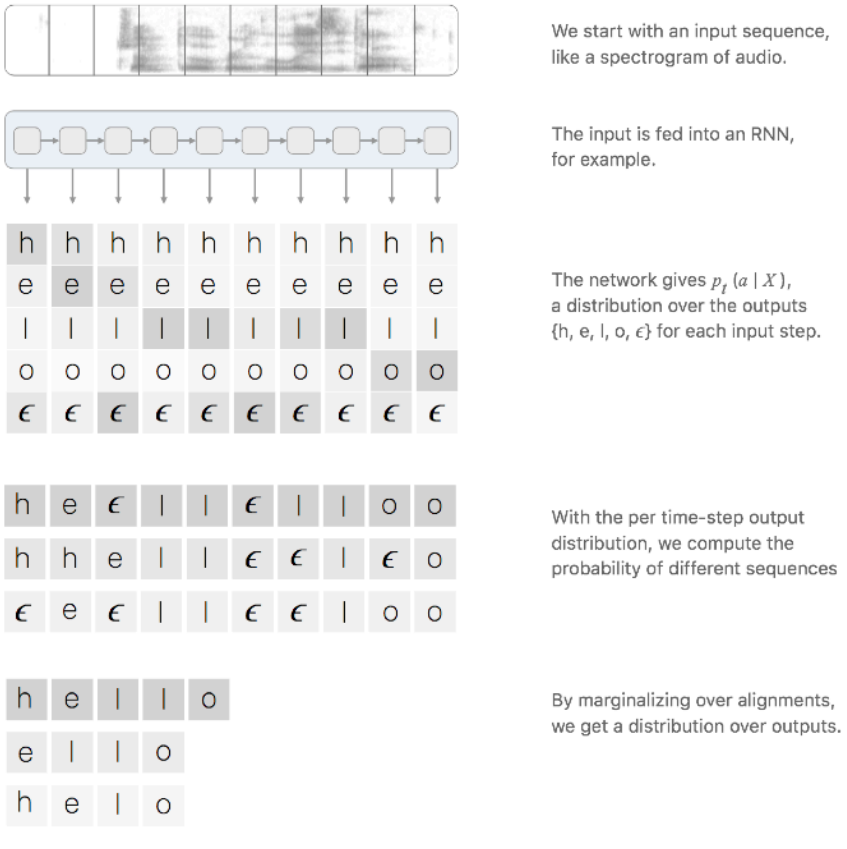
Training: maximise the probability $p(\mathbf{l}\,|\,\mathbf{x}) \Rightarrow$ minimise the CTC loss

- $p(\mathbf{l}\,|\,\mathbf{x})$?
  - we want to know how likely a given sentence $\mathbf{l}$ ="hello" is, given the input $\mathbf{x}$=spectrogram
  - T_x and T_y have different length
- **Alignements paths π**
  - we need to define the different possible π ("pronunciations") of $\mathbf{l}$="hello" such that it matches T_x
  - $\mathcal{B}^{-1}$("hello") for $T_x = 8$
    - "hel-looo", "hel-lloo", "hhhel-lo", "hheel-lo", . . .
- $p(\mathbf{l}\,|\,\mathbf{x})$
  - We sum up over all possible paths $\pi \in \mathcal{B}^{-1}(\mathbf{l})$

$$p(\mathbf{l}\,|\,\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi\,|\,\mathbf{x})$$

  - For a given path π

$$p(\pi\,|\,\mathbf{x}) = \Pi_{t=1}^{T} y_t^{\pi_t}$$



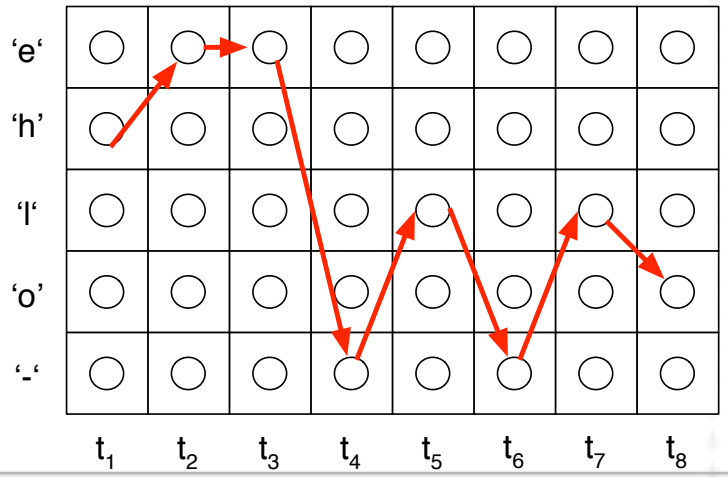
Training : minimise the CTC Loss

- CTCLoss = $-\log p(\mathbf{l})$
- **Example:**

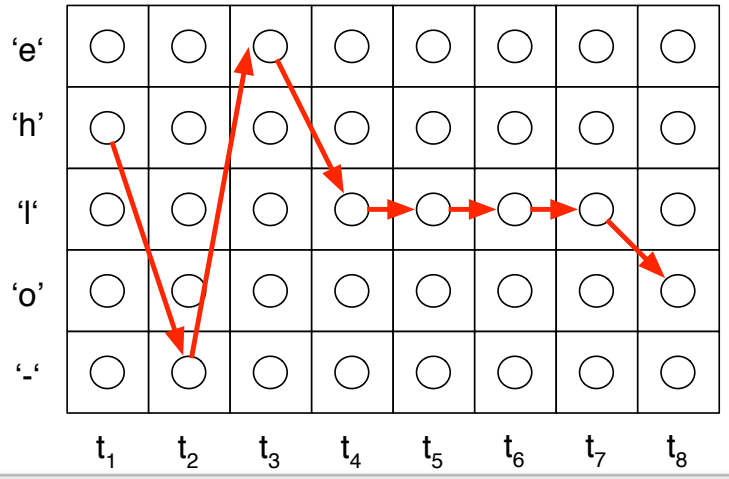
  for the world "hello", we wand to minimise $-\log p(\mathbf{l} = '' hello'' | \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l}=''hello'')} p(\pi | \mathbf{x})$

- **Problem:**
  - the number of alignement paths $\pi \in \mathcal{B}^{-1}(\mathbf{l})$ can quickly explode !
  - there are $5^8 = 390.625$ ways (possible paths) to go from t_1 to t_8 $\Rightarrow$ cannot be solved trivially

f('hee-l-lo')='hello'

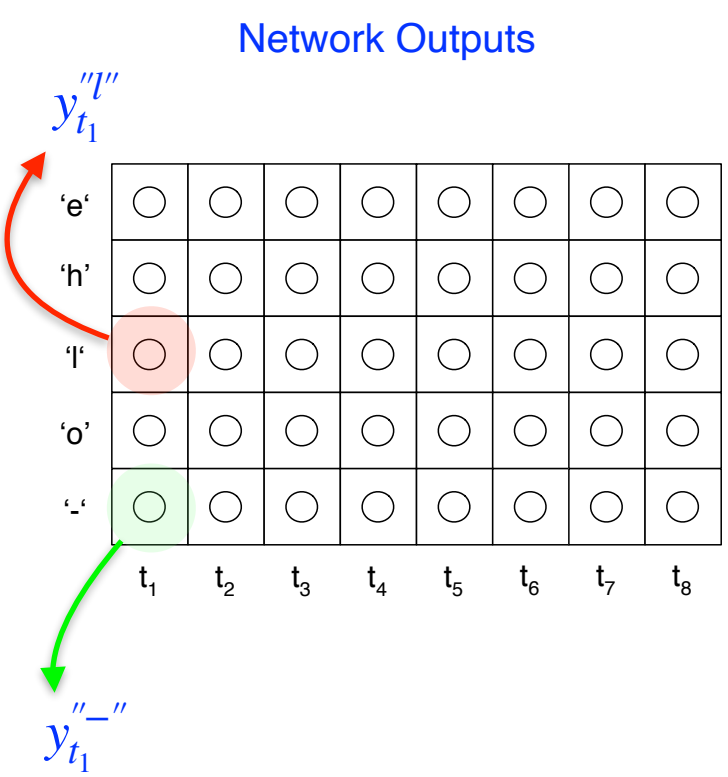


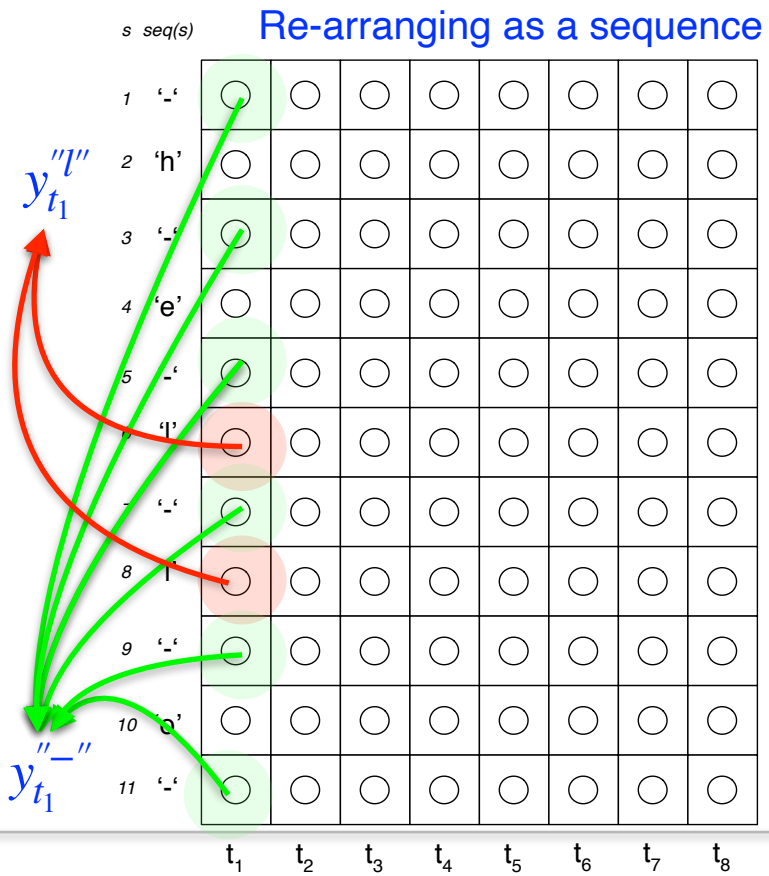
f('h-ellllo')='helo'



Training : the CTC Loss

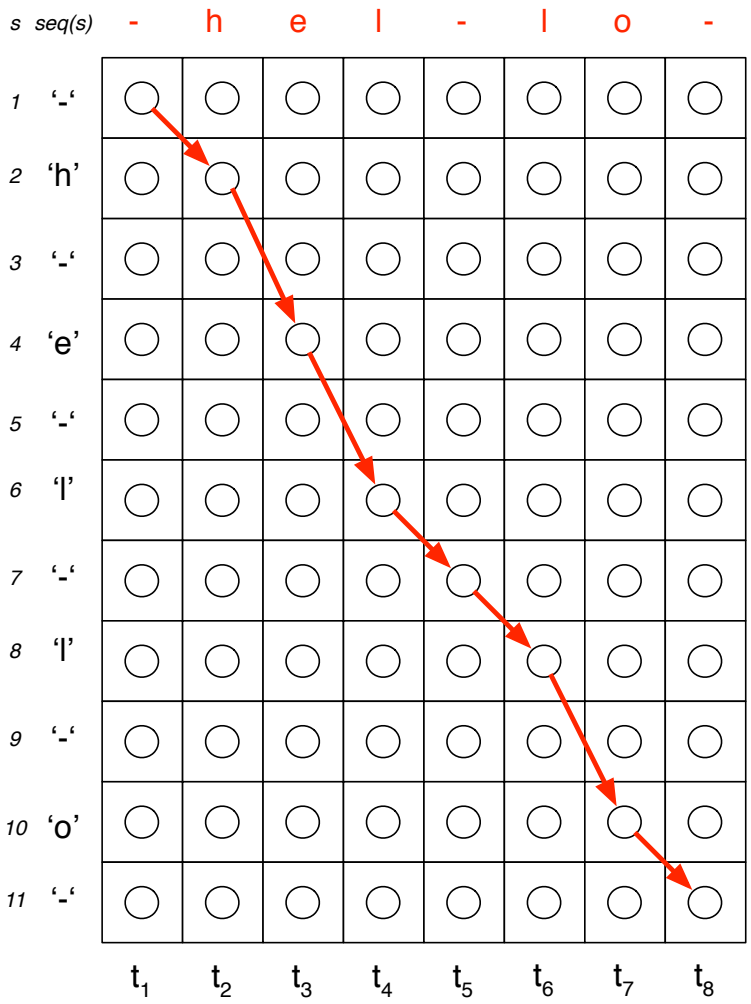
- What can be done then ? **Dynamic programming**
- Use a **New representation** : more easy to represent the correct paths in the graph





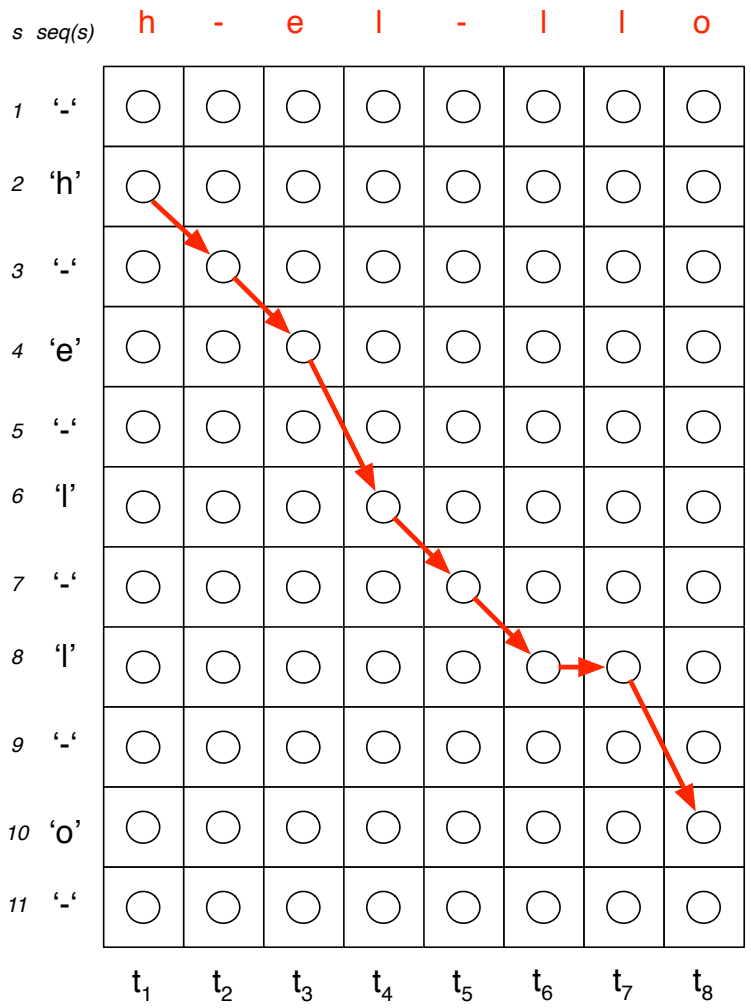
Training : the CTC Loss

- Alternative paths
  - $\mathcal{B}$**("-hel-lo-") = "hello" ,**
  - $\mathcal{B}$("-hel-llo") = "hello"
  - $\mathcal{B}$("h-el-llo") = "hello"
  - $\mathcal{B}$("h-el-lo-") = "hello"



Training : the CTC Loss

- Alternative paths
  - $\mathcal{B}$("-hel-lo-") = "hello"
  - $\mathcal{B}$("-hel-llo") = "hello"
  - **$\mathcal{B}$("h-el-llo") = "hello"**
  - $\mathcal{B}$("h-el-lo-") = "hello"

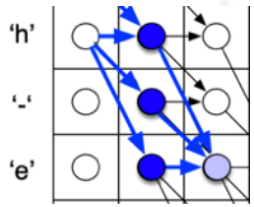


Connectionist Temporal Classification

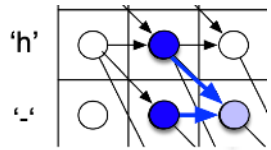
Training : the CTC Loss

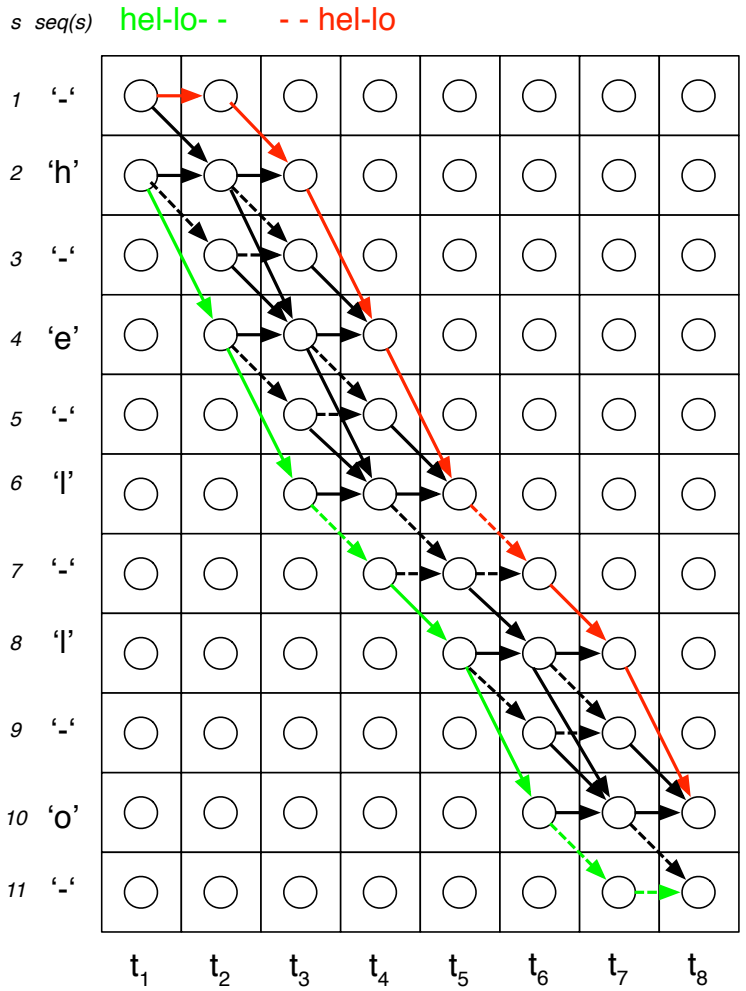
– All possible paths $\pi \in \mathcal{B}^{-1}(\mathbf{l})$
  - Capture variations in pronunciation from $\mathcal{B}$("hel-lo--") to f("--hel-lo")

– Allowed paths
  - rule: cannot skip a letter
  - to reach a letter
    – previous letter
    – previous blank
    – previous same letter



  - to reach a blank
    – previous same blank
    – previous letter





Training : the CTC Loss

- CTC Loss Components
  - **Forward variable** $\alpha_t(s)$
    - Calculates the total probability from the first time-step till time-step t and token s
  - **Backward variable** $\beta_t(s)$
    - Calculates the total probability from the time-step t and token s until last time-step

Connectionist Temporal Classification

Training : the CTC Loss

- **Forward Calculation Example 1**
  - Let's calculate $\alpha_{t=3}(s=4)$
  - There are 4 paths

$$p(''-he'') = y_{t_1}^{''-''} y_{t_2}^{''h''} \quad y_{t_3}^{''e''}$$

$$p(''hhe'') = y_{t_1}^{''h''} y_{t_2}^{''h''} \quad y_{t_3}^{''e''}$$

$$p(''h-e'') = y_{t_1}^{''h''} y_{t_2}^{''-''} \quad y_{t_3}^{''e''}$$

$$p(''hee'') = y_{t_1}^{''h''} y_{t_2}^{''e''} \quad y_{t_3}^{''e''}$$

  - Final probability is

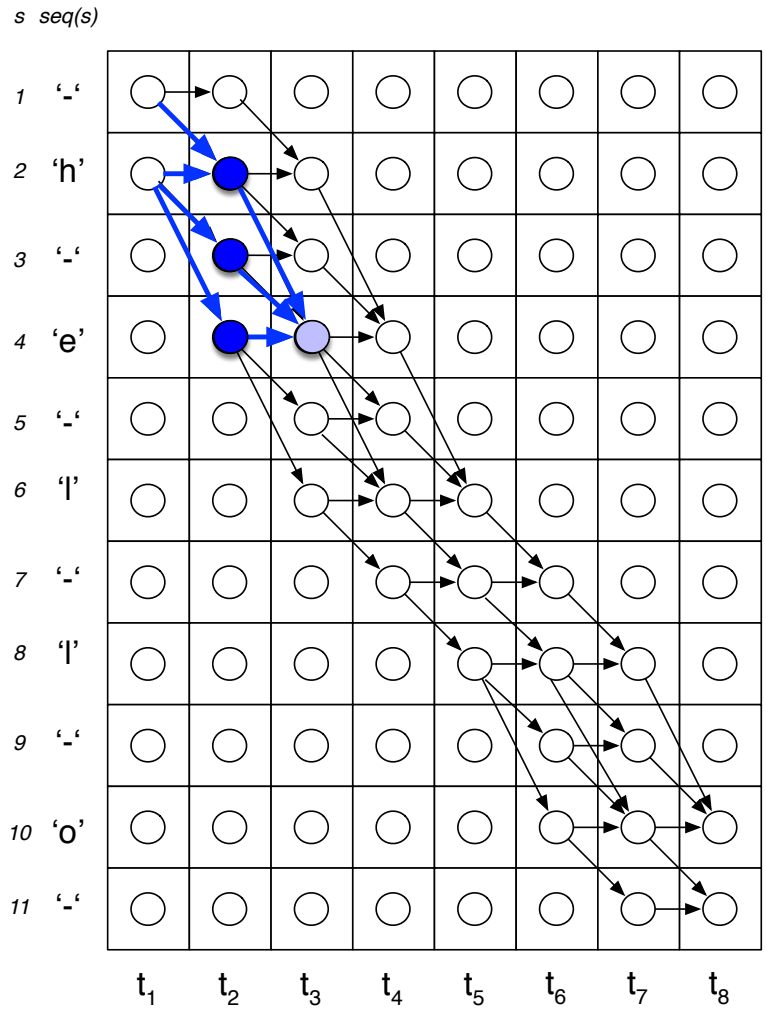
$$p(''-he'') + p(''hhe'') + p(''h-e'') + p(''hee'')$$

- **Using dynamic programming :**
  - this expression can be computed recursively using the three possible paths to $s = 4$

$$\alpha_{t_3}(4) = [\alpha_{t_2}(4) + \alpha_{t_2}(3) + \alpha_{t_2}(2)] \cdot y_{t_3}^{''e''}$$

  - or on general, noting $seq(s=4) = ''\,e''$

$$\alpha_t(s) = [\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2)] \cdot y_t^{seq(s)}$$



Training : the CTC Loss

- **Forward Calculation Example 2**
  - Let's calculate $\alpha_{t_3}(3)$
    - There are 2 paths (we cannot move from blank to blank)

$$\alpha_{t_3}(3) = [\alpha_{t_2}(3) + \alpha_{t_2}(2)] \cdot y_{t_3}^{''-''}$$

  - Let's calculate $\alpha_{t_6}(8)$
    - There are 2 paths (we cannot move from blank to blank)

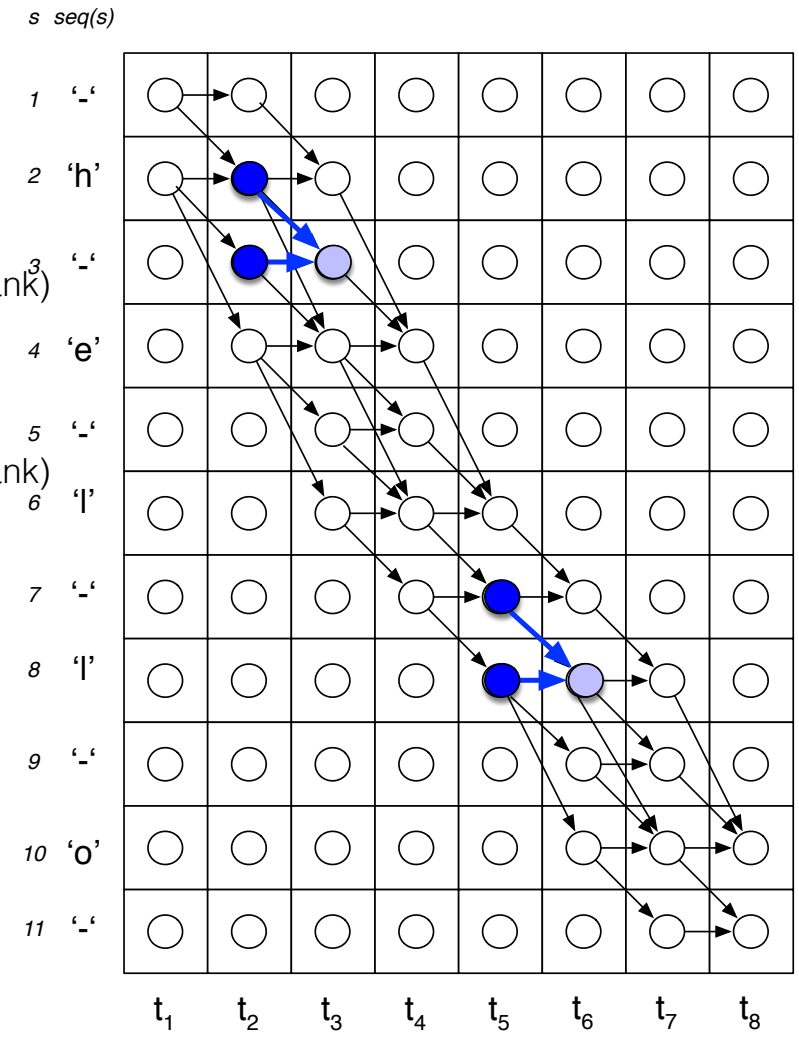
$$\alpha_{t_6}(8) = [\alpha_{t_5}(6) + \alpha_{t_5}(7)] \cdot y_{t_6}^{''-''}$$

- **Using dynamic programming:**
  - (for both):

$$\alpha_t(s) = [\alpha_{t-1}(s) + \alpha_{t-1}(s-1)] \cdot y_t^{seq(s)}$$

  - Note that:

$$p(''hello'') = \alpha_{t_8}(10) + \alpha_{t_8}(11)$$



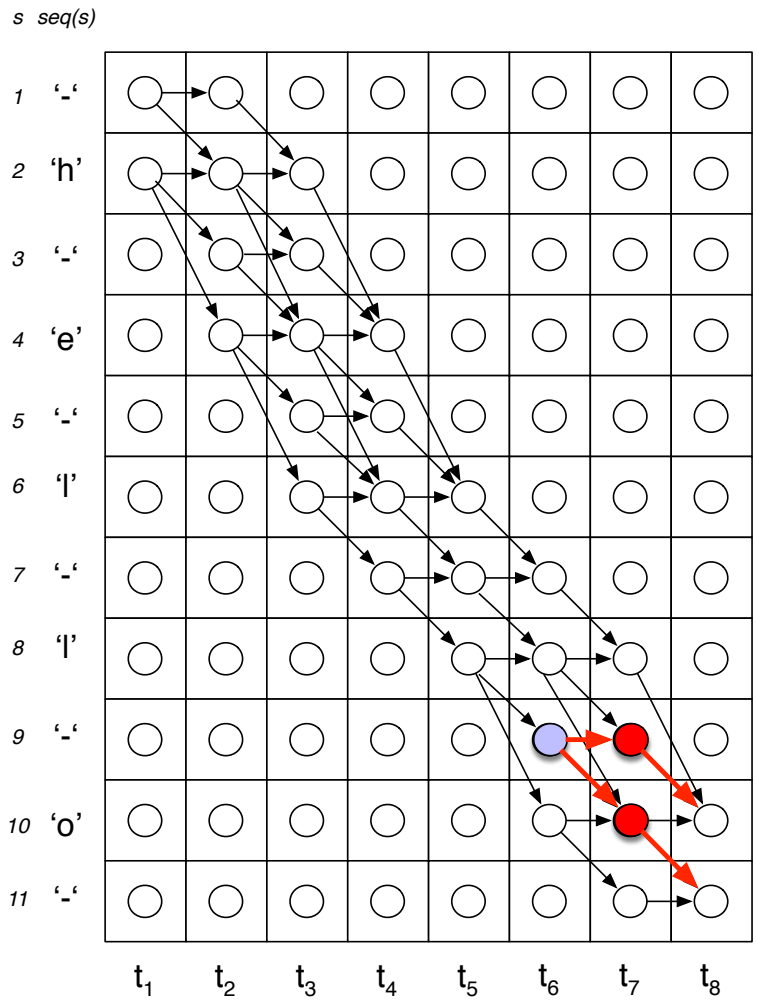
Connectionist Temporal Classification

Training : the CTC Loss

- **Backward calculation example**
  - this phase is just the opposite of the forward phase

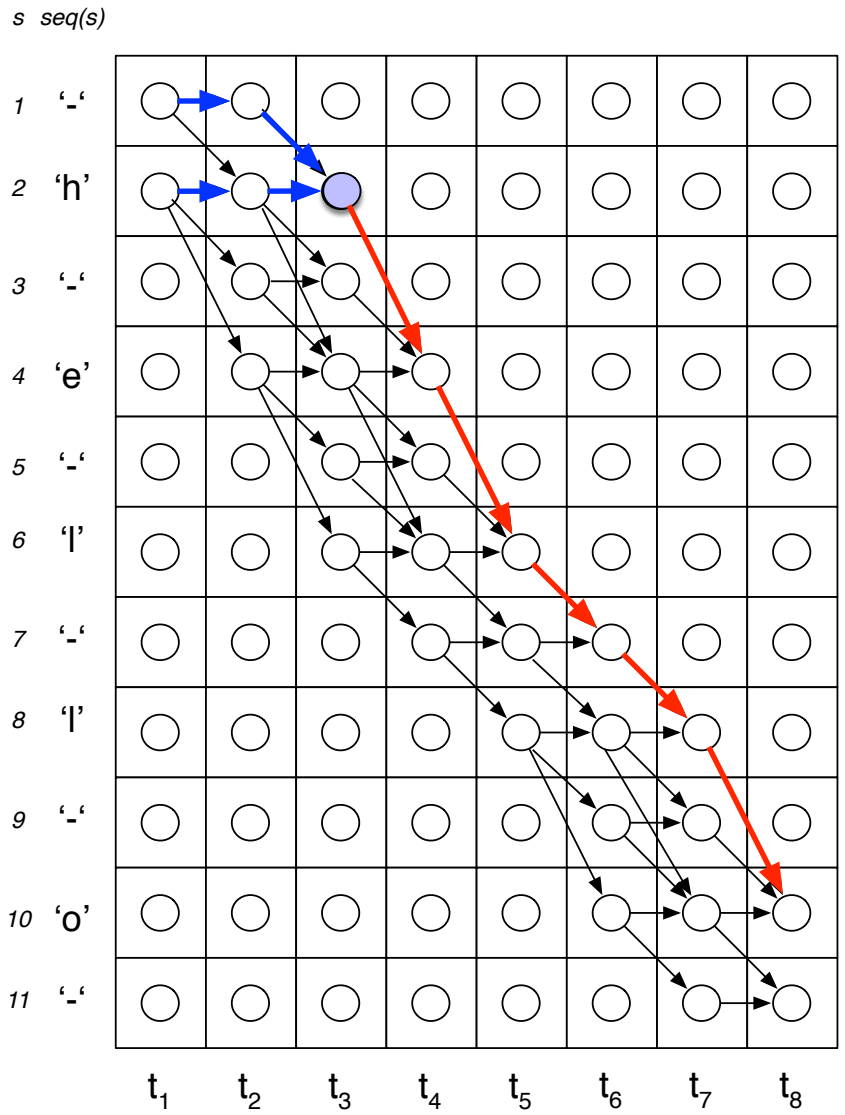
$$\beta_{t_6}(9) = p(''--o'') + p(''-oo'') + p(''-o-'')$$

- **Using dynamic programming**
  - the same logic here applies as the forward calculation

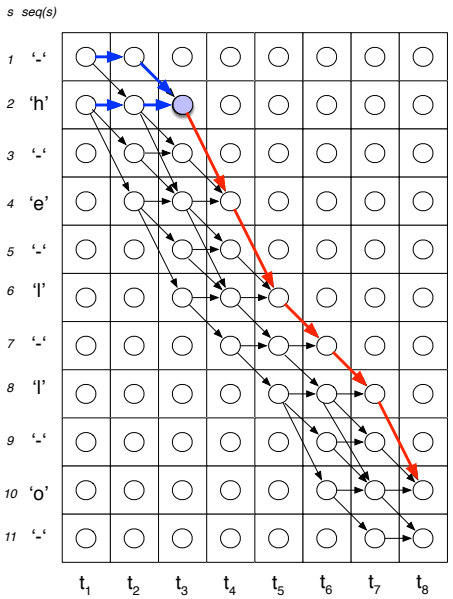


Training : the CTC Loss

– **Complete path calculation example**



Training : the CTC Loss



- **Complete path calculation example**

- The **forward pass** gives

$$\alpha_{t_3}(2) = p(''--h'') + p(''-hh'') + p(''hhh'')$$
$$= y_{t_1}^{''-''} y_{t_2}^{''-''} y_{t_3}^{''h''} + y_{t_1}^{''-''} y_{t_2}^{''h''} y_{t_3}^{''h''} + y_{t_1}^{''h''} y_{t_2}^{''h''} y_{t_3}^{''h''}$$

- The **backward pass** results in

$$\beta_{t_3}(2) = p(''hel - lo'')$$
$$= y_{t_3}^{''h''} y_{t_4}^{''e''} y_{t_5}^{''l''} y_{t_6}^{''-''} y_{t_7}^{''l''} y_{t_8}^{''o''}$$

- The **final results**

$$\alpha_{t_3}(2)\beta_{t_3}(2) = y_{t_1}^{''-''} y_{t_2}^{''-''} y_{t_3}^{''h''} \cdot y_{t_3}^{''h''} y_{t_4}^{''e''} y_{t_5}^{''l''} y_{t_6}^{''-''} y_{t_7}^{''l''} y_{t_8}^{''o''}$$
$$+ y_{t_1}^{''-''} y_{t_2}^{''h''} y_{t_3}^{''h''} \cdot y_{t_3}^{''h''} y_{t_4}^{''e''} y_{t_5}^{''l''} y_{t_6}^{''-''} y_{t_7}^{''l''} y_{t_8}^{''o''}$$
$$+ y_{t_1}^{''h''} y_{t_2}^{''h''} y_{t_3}^{''h''} \cdot y_{t_3}^{''h''} y_{t_4}^{''e''} y_{t_5}^{''l''} y_{t_6}^{''-''} y_{t_7}^{''l''} y_{t_8}^{''o''}$$
$$= \left[p(''--hel - lo'') + p(''-hhel - lo'') + p(''hhhel - lo'')\right] \cdot y_{t_3}^{''h''}$$

- **Total probability** of all paths going through $''h''$ at t_3

$$\frac{\alpha_{t_3}(2)\beta_{t_3}(2)}{y_{t_3}^{''h''}}$$

Training : the CTC Loss

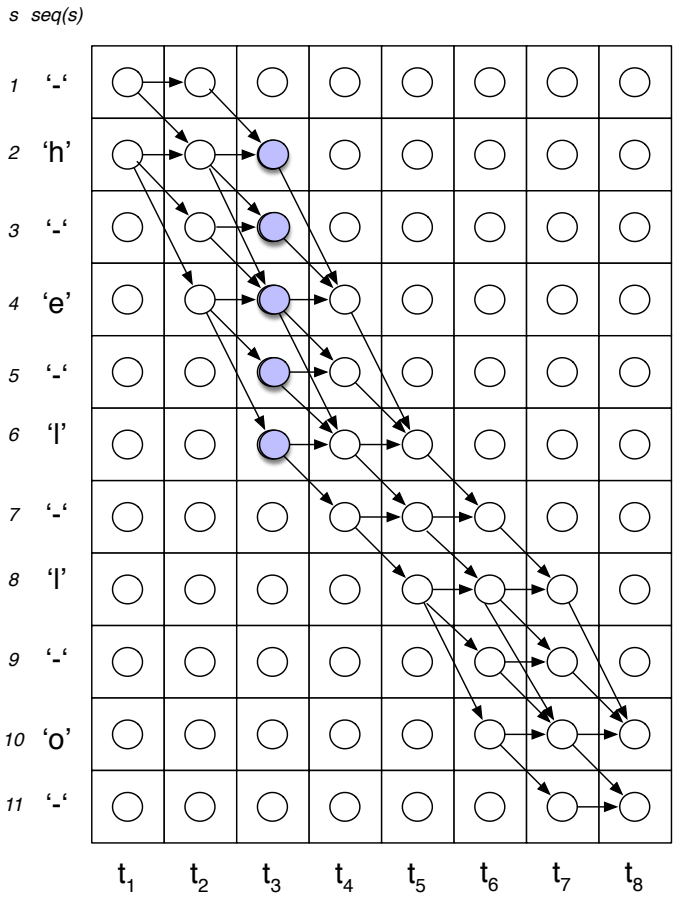
– **Calculating the Loss at a particular time-step example**

– $p(''hello'')$ at $t = 3$ is the sum of the probabilities of all paths through all the symbols

$$p(''hello'') = \sum_{s=2}^{6} \frac{\alpha_{t_3}(s)\beta_{t_3}(s)}{y_{t_3}^{seq(s)}}$$

– In general, the probability of the ground label is

$$p(''hello'') = \sum_{s=1}^{|seq|} \frac{\alpha_t(s)\beta_t(s)}{y_t^{seq(s)}}$$



Training : the CTC Loss

– **How to do back-propagation ?**

$$\frac{\delta - \log p(''hello'')}{\delta y_t^k} = \frac{-1}{p(''hello'')} \frac{\partial p(''hello'')}{\partial y_t^k}$$

– since

$$p(''hello'') = \sum_{s=1}^{|seq|} \frac{\alpha_t(s)\beta_t(s)}{y_t^{seq(s))}}$$

– we have

$$\frac{\partial p(''hello'')}{\partial y_t^k} = \frac{1}{(y_t^k)^2} \sum_{s:seq(s)=k} \alpha_t(s)\beta_t(s)$$

Training : the CTC Loss

– **Back-propagation example 1**

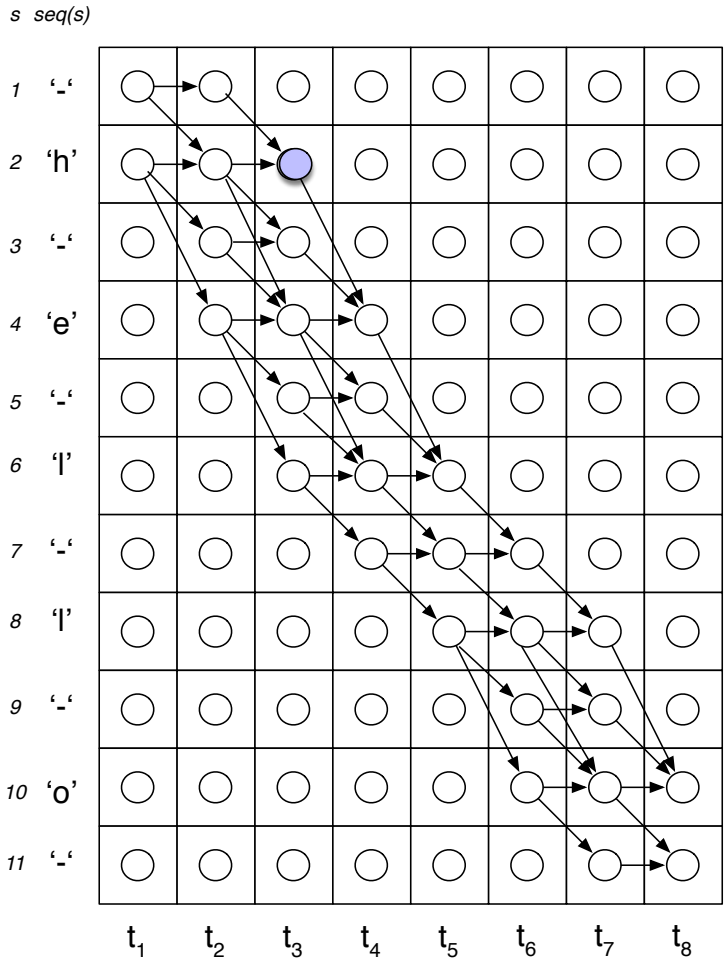
– Given

$$\frac{\partial p(''hello'')}{\partial y_t^k} = \frac{-1}{(y_t^k)^2} \sum_{s:seq(s)=k} \alpha_t(s)\beta_t(s)$$

– For $t = 3$ and $k = '' h''$

- h occurs at $s = 2$

$$\frac{\partial p(''hello'')}{\partial y_{t_3}^{''h''}} = \frac{-1}{(y_{t_3}^{''h''})^2} \alpha_{t_3}(2)\beta_{t_3}(2)$$



Training : the CTC Loss

– **Back-propagation example 2**

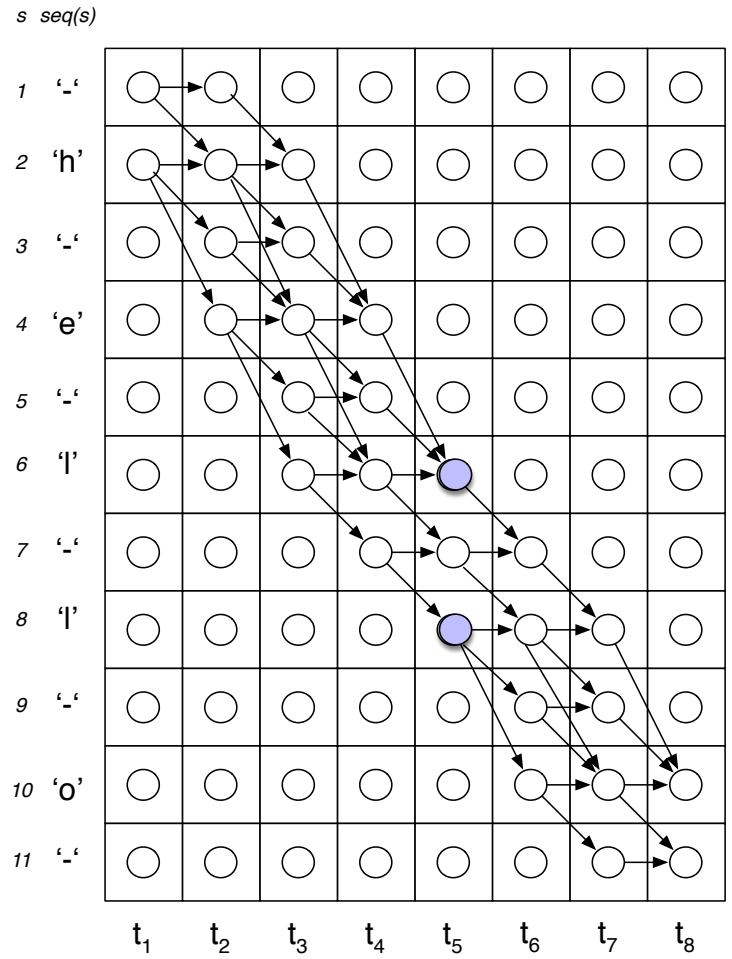
– Given

$$\frac{\partial p(''hello'')}{\partial y_t^k} = \frac{-1}{(y_t^k)^2} \sum_{s:seq(s)=k} \alpha_t(s)\beta_t(s)$$

– For $t = 5$ and $k = '' \, l''$

- l occurs at $s = 6$ and $s = 8$

$$\frac{\partial p(''hello'')}{\partial y_{t_5}^{''l''}} = \frac{-1}{(y_{t_5}^{''h''})^2}[\alpha_{t_5}(6)\beta_{t_5}(6) + \alpha_{t_5}(8)\beta_{t_5}(8)]$$



Implementation

```
# Target are to be padded
T = 50      # Input sequence length
C = 20      # Number of classes (including blank)
N = 16      # Batch size
S = 30      # Target sequence length of longest target in batch (padding length)
S_min = 10  # Minimum target length, for demonstration purposes

# Initialize random batch of input vectors, for *size = (T,N,C)
input = torch.randn(T, N, C).log_softmax(2).detach().requires_grad_()

# Initialize random batch of targets (0 = blank, 1:C = classes)
target = torch.randint(low=1, high=C, size=(N, S), dtype=torch.long)

input_lengths = torch.full(size=(N,), fill_value=T, dtype=torch.long)
target_lengths = torch.randint(low=S_min, high=S, size=(N,), dtype=torch.long)

ctc_loss = nn.CTCLoss()
loss = ctc_loss(input, target, input_lengths, target_lengths)
loss.backward()
```

CTC vs Framewise

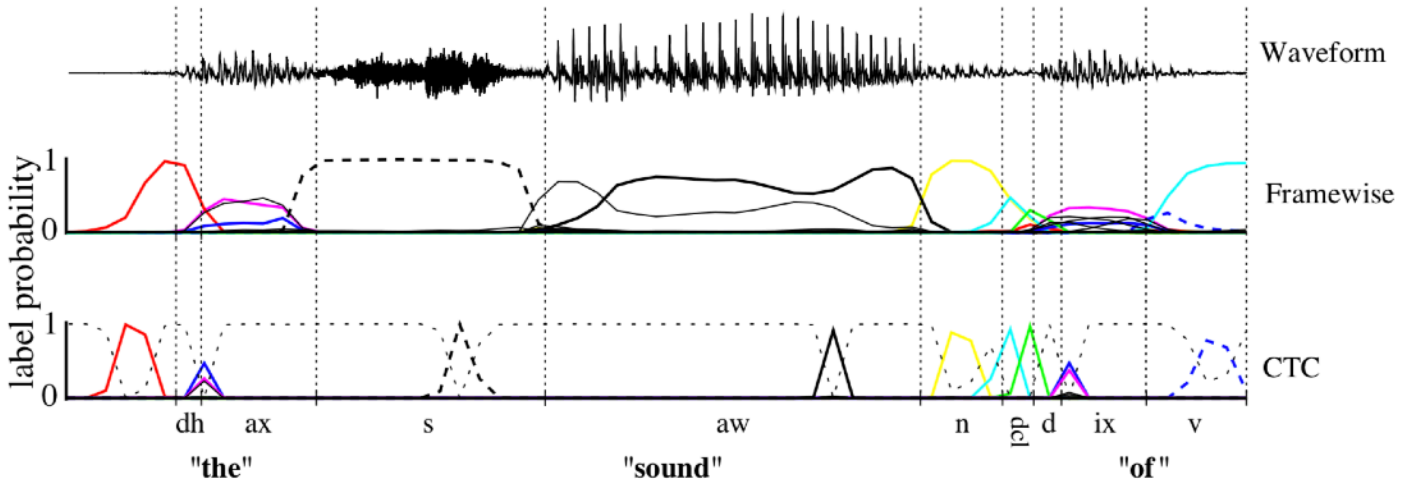
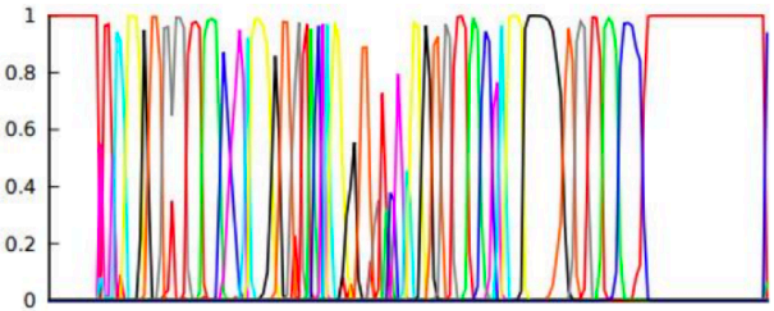


Figure 1. **Framewise and CTC networks classifying a speech signal**. The shaded lines are the output activations, corresponding to the probabilities of observing phonemes at particular times. The CTC network predicts only the sequence of phonemes (typically as a series of spikes, separated by 'blanks', or null predictions), while the framewise network attempts to align them with the manual segmentation (vertical lines). The framewise network receives an error for misaligning the segment boundaries, even if it predicts the correct phoneme (e.g. 'dh'). When one phoneme always occurs beside another (e.g. the closure 'dcl' with the stop 'd'), CTC tends to predict them together in a double spike. The choice of labelling can be read directly from the CTC outputs (follow the spikes), whereas the predictions of the framewise network must be post-processed before use.

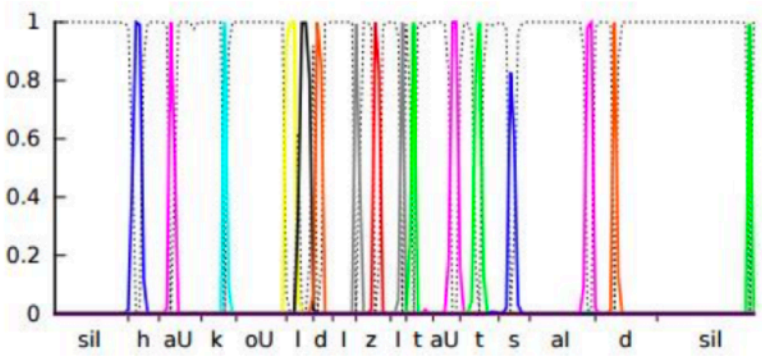
CTC vs HMM

– Output probability of LSTM

HMM



CTC



– CTC embeds history in continuous hidden space, more capable than HMM
– CTC has spiky predictions, more discriminable between states than HMM
– CTC with less states (40) is significantly faster for decoding than HMM (10k states)

Connectionist Temporal Classification

Example of results with CTC

AS OF APRIL FIRST ALL INTEREST INCOME WILL BE TAXED AT TWENTY PERCENT.

BLSTM-CTC transcribed grapheme string:

a s SP o f SP a i p r o l f i r s t SP a l SP i n t e r e s t SP i n c o m e SP w i l l SP b e SP t a x t SP i t SP t w e n t i n SP p e r c e n t SP

Final output after decoding:

AS OF APRIL FIRST AL **INTEREST INCOME WILL BE** TAX T. IT **TWENTY PERCENT.**

W SJ1 example sentence 1. Average recognition case. Manual annotation (top), BLSTM-CTC grapheme output-SP denotes short-pause, i.e. a word boundary (middle), BLSTM-CTC grapheme output after dictionary based (unweighted, no language model) decoding (bottom), see section III. Correctly recognised words are printed in bold-face.

Model	Prediction
Ground Truth	SILENCE
CTC	SILENCE
RNN-Transducer	SILENCE
Attention	i want to get to get to get to get to get to get to get to do that
Ground Truth	play the black eyed peas songs
CTC	
+ Greedy	lading to black **irpen** songs
+ Beam Search + LM	leading to black **european** songs
RNN-Transducer	
+ Greedy	play the black **eye piece** songs
+ Beam Search	play the black **eye piece** songs
+ LM rescore	play the black eyed peas songs
Attention	
+ Greedy	play the black **eyed pea** songs
+ Beam Search	play the black **eyed pea** songs
+ LM rescore	play the black eyed peas songs

Performances of CTC

– Word error rate compared with HMM based models

	Context	Error Rate (%)
LSTM-HMM	Uni	8.9
	Bi	9.1
LSTM-CTC	Uni	9.4
	Bi	8.5

– Bi-direction LSTM is more essential for CTC

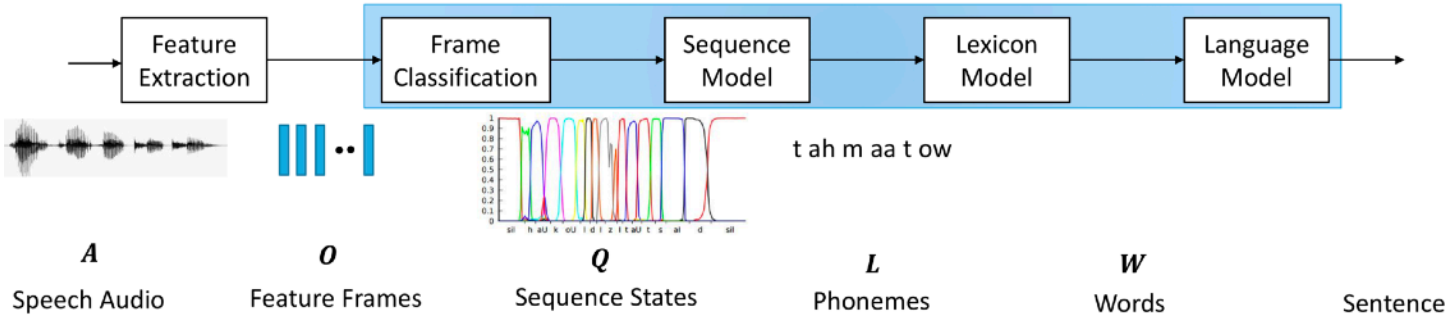
Table 1. **Label Error Rate (LER) on TIMIT**. CTC and hybrid results are means over 5 runs, ± standard error. All differences were significant ($p < 0.01$), except between weighted error BLSTM/HMM and CTC (best path).

System	LER
Context-independent HMM	38.85 %
Context-dependent HMM	35.21 %
BLSTM/HMM	33.84 ± 0.06 %
Weighted error BLSTM/HMM	31.57 ± 0.06 %
CTC (best path)	31.47 ± 0.21 %
CTC (prefix search)	30.51 ± 0.19 %

Connectionist Temporal Classification

← END

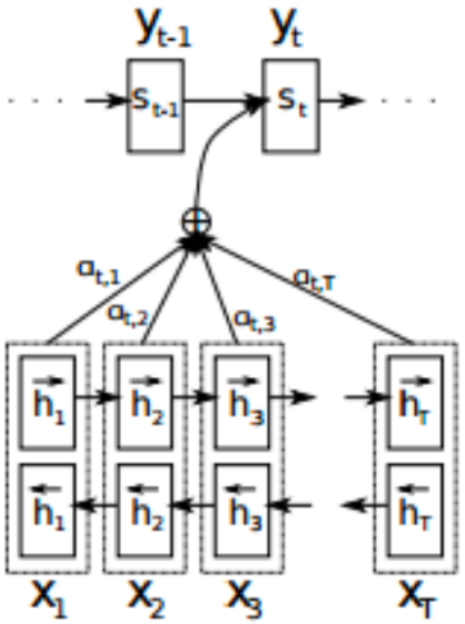
Deep learning approaches: 3) End-to-End models: Attention mechanisms



- **Predict character sequence directly**
- Attention mechanisms also use in
  - Image generation
  - Image text embedding
  - Question answering
  - Machine translation
- Weighted sum of previous history

$$\alpha_{t,j} = softmax(e_{t,j}) = \frac{\exp(e_{t,j})}{\sum_{k=1}^{T} \exp(e_{t,k})}$$

- Essential for modelling long sequences like speech



[Chan, Jaitly, Le, Vinyals, "**Listen, Attend and Spell**", IEEE ICASSP, 2016] LINK

ASR: Deep-learning approach

Deep learning approaches: 3) End-to-End models: Attention mechanisms

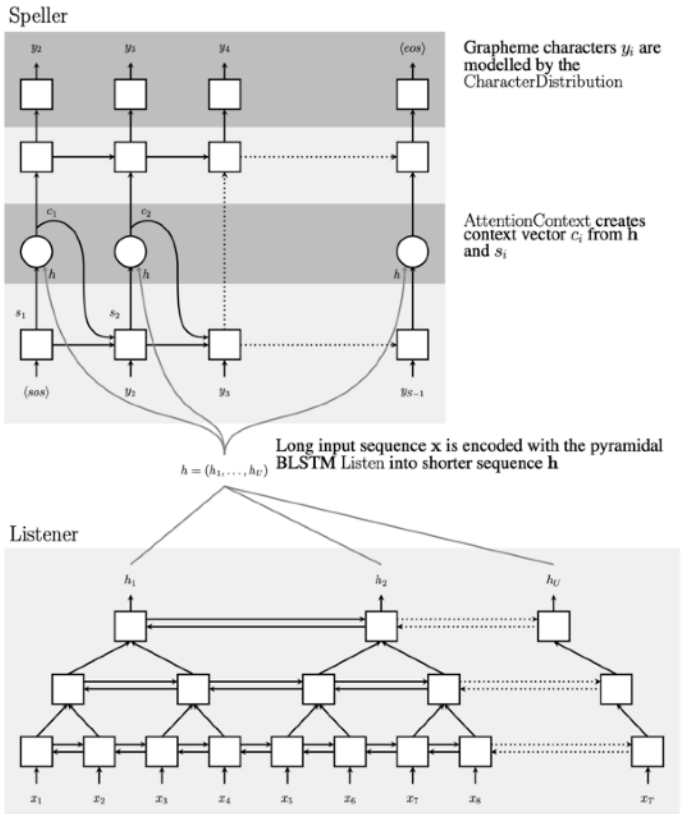


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence **x** into high level features **h**, the speller is an attention-based decoder generating the **y** characters from **h**.

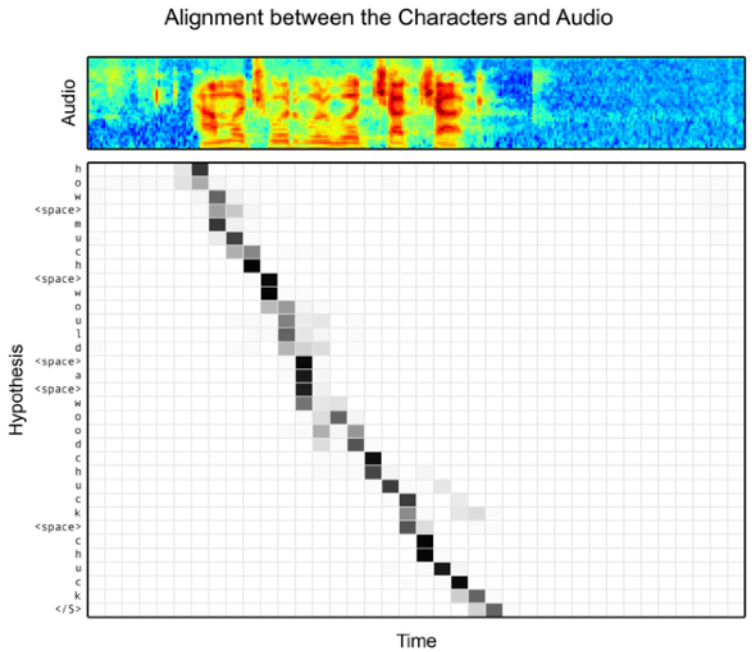


Figure 2: Alignments between character outputs and audio signal produced by the Listen, Attend and Spell (LAS) model for the utterance "how much would a woodchuck chuck". The content based attention mechanism was able to identify the start position in the audio sequence for the first character correctly. The alignment produced is generally monotonic without a need for any location based priors.

[Chan, Jaitly, Le, Vinyals, "**Listen, Attend and Spell**", IEEE ICASSP, 2016] LINK

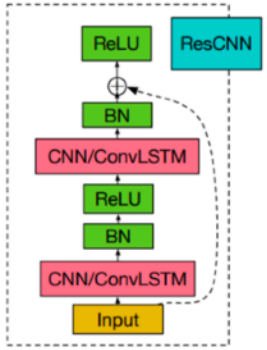
https://jonathan-hui.medium.com/speech-recognition-deep-speech-ctc-listen-attend-and-spell-d05e940e9ed1

Deep learning approaches: 3) End-to-End models: Attention mechanisms

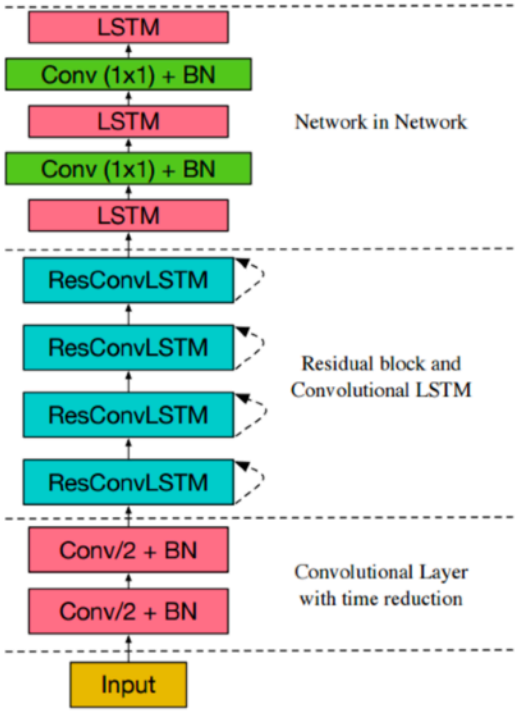
– **Comparisons**

- Word error rate w/o language models on Wall Sti

	WER
CTC	30.1
Proposed	10.53

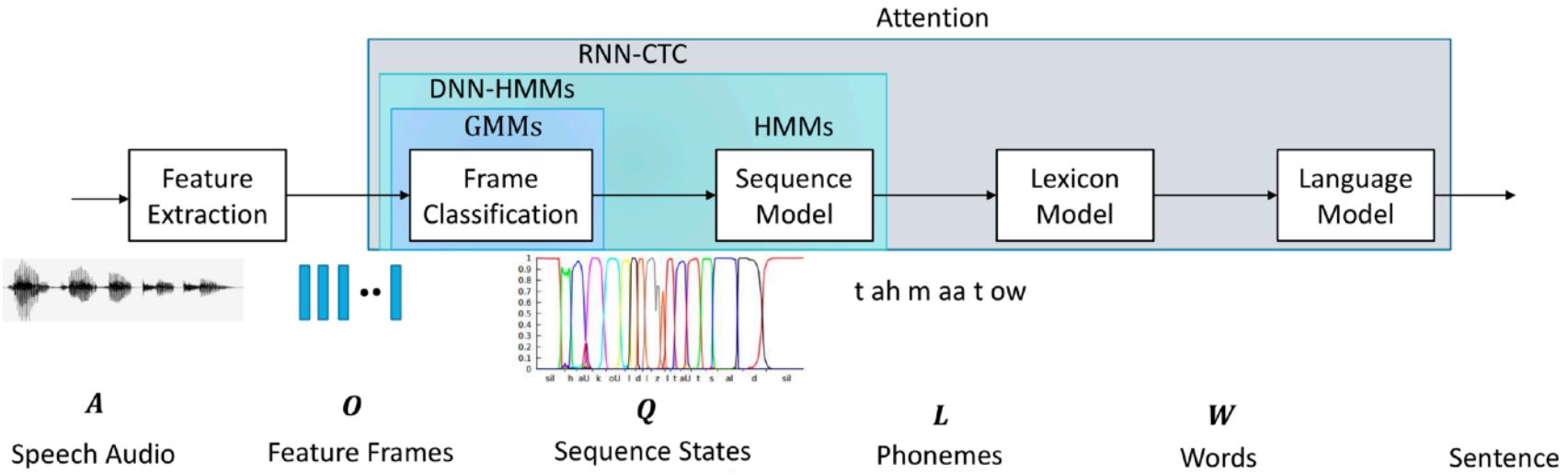


(a) ResCNN block



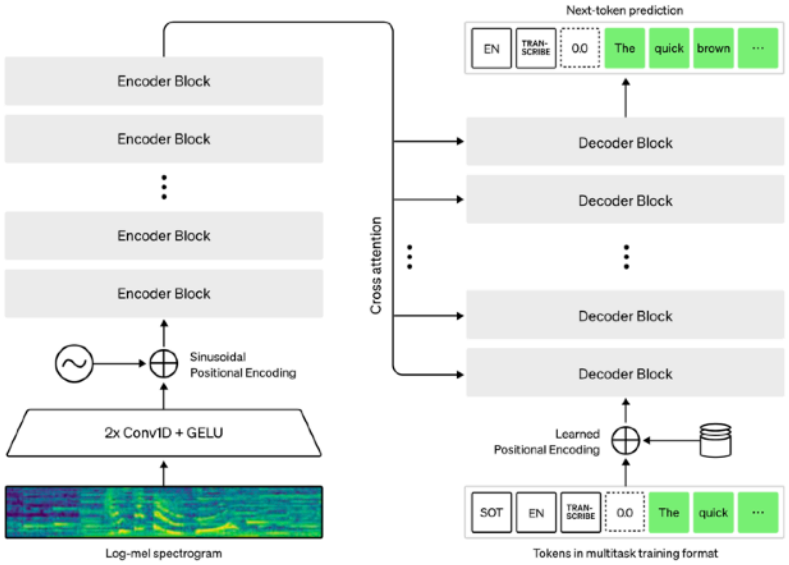
Encoder Network

ASR: Deep-learning approach



Deep learning approaches: 4) End-to-End models: Transformer

- **OpenAI Whisper**: Encode/Attention/Decoder but with **Transformer** architecture
  - trained on **680,000** hours of **multilingual** and **multitask** supervised data collected from the web
  - enables transcription in multiple languages + translation from those languages into English
  - 30-second chunks, converted into a log-Mel spectrogram



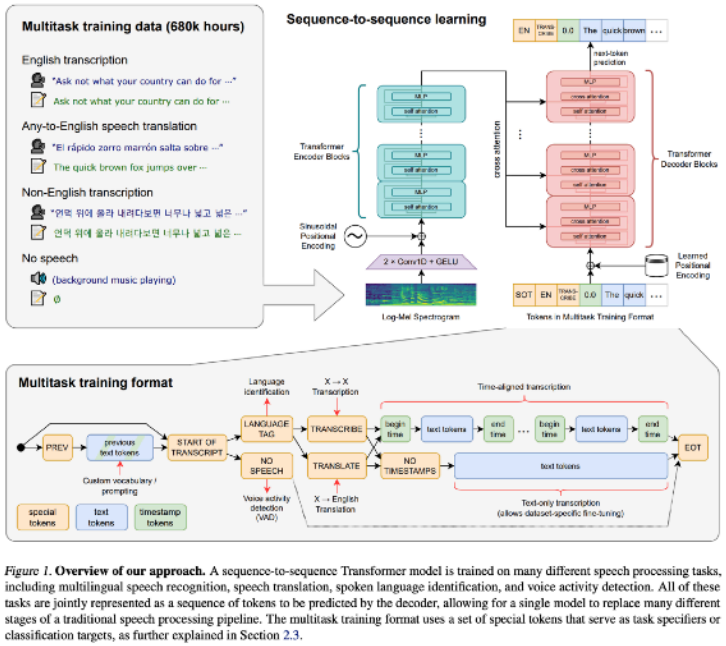


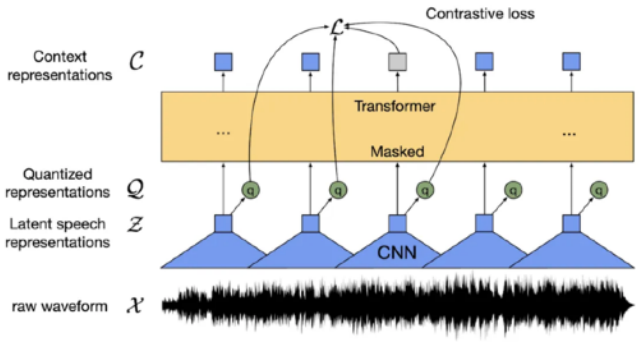
Figure 1. **Overview of our approach.** A sequence-to-sequence Transformer model is trained on many different speech processing tasks, including multilingual speech recognition, speech translation, spoken language identification, and voice activity detection. All of these tasks are jointly represented as a sequence of tokens to be predicted by the decoder, allowing for a single model to replace many different stages of a traditional speech processing pipeline. The multitask training format uses a set of special tokens that serve as task specifiers or classification targets, as further explained in Section 2.3.

[Radford et al. "**Robust Speech Recognition via Large-Scale Weak Supervision**", ICML, 2023] LINK
https://openai.com/research/whisper

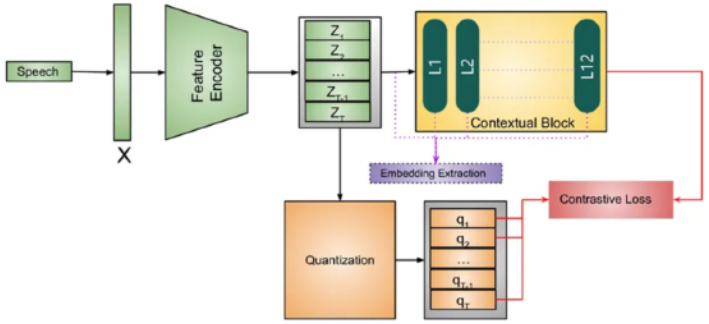
ASR: Deep-learning approach

Deep learning approaches: 5) ASR with pre-trained (SSL) models

– **Wav2Vec2**

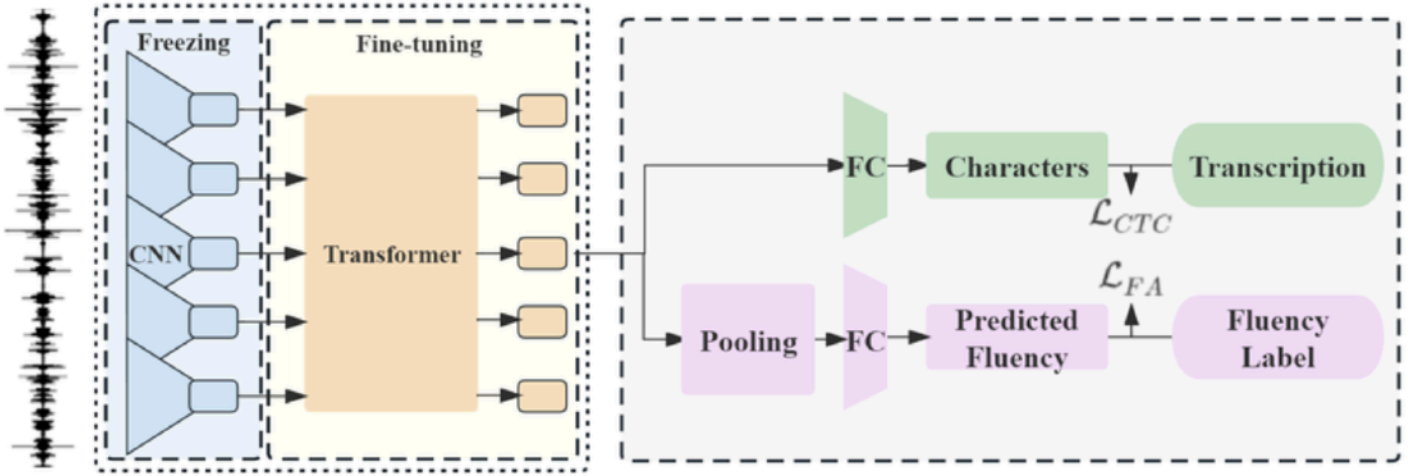


Components of Wav2Vec 2.0



Architecture of Wav2Vec 2.0

– **Fine-tuning**



[Baevski et al. "**wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations**", 2020] LINK