NPFL123 Dialogue Systems
11. Speech Recognition

https://ufal.cz/npfl123

Ondřej Dušek, Patrícia Schmidtová, Vojtěch Hudeček & Jan Cuřín

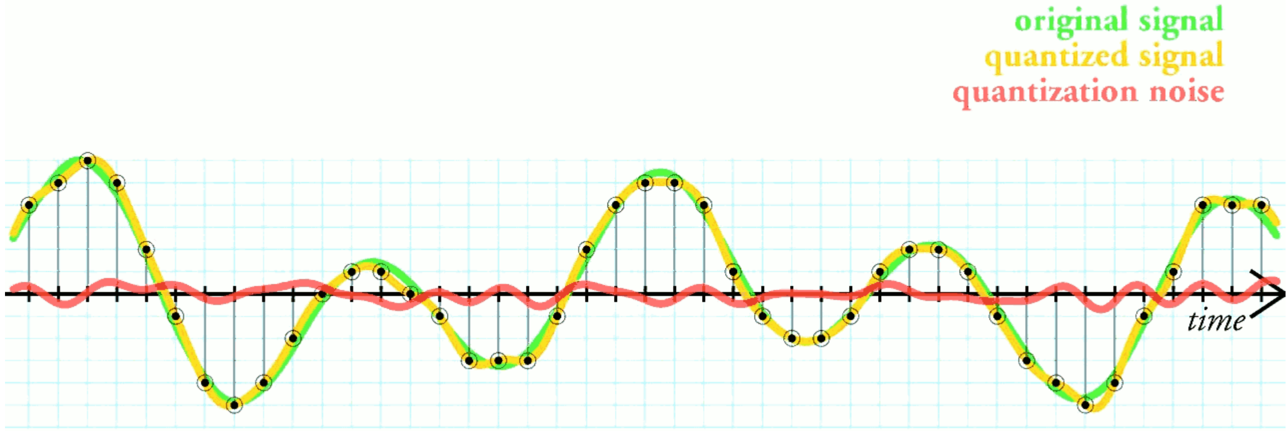
loosely based on earlier slides by Petr Fousek, Pavel Květoň, Michal Jůza

15. 5. 2023

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Speech recognition

- Task: convert audio (sound wave) → text
  - generally just words, no punctuation or capitalization used
- Audio: waveform
  - wave position in time (samples)
    - 8 kHz – 44 kHz frequency (telephone → CD quality)
    - 8-16 kHz mostly used for speech
  - quantized (=8-bit/16-bit number)
  - lot more than just words:
    - speaker identity (age, gender, dialect, speech defects), emotional state (pitch, loudness, health)
    - environment, noise (reverb, distance, channel effects)



https://en.wikipedia.org/wiki/Quantization_%28signal_processing%29

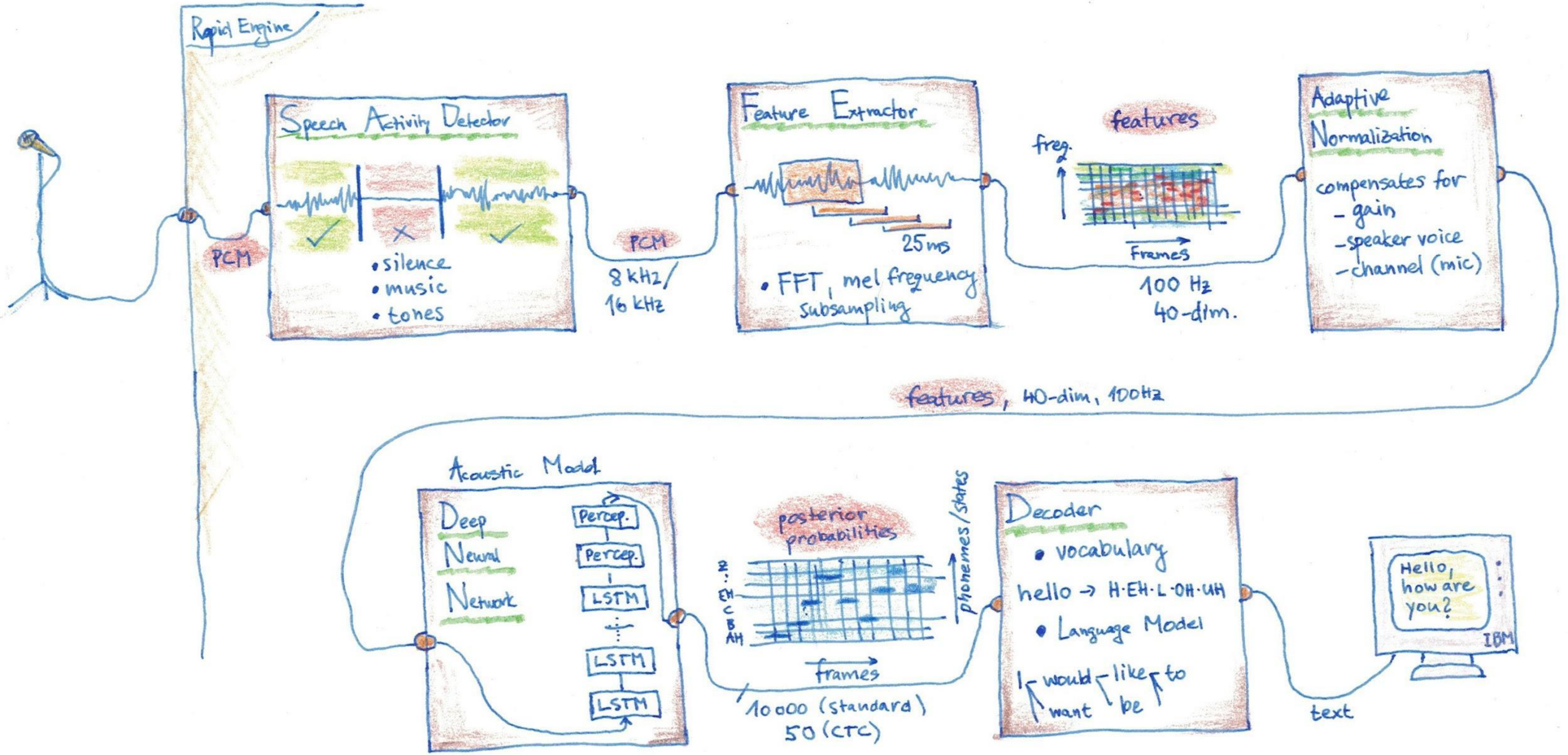
- ASR is basically very harsh lossy compression
  - from ~ 64 kbps (8 kHz, 8-bit) to ~ 50 bps (text)
  - for context, low-bitrate audio codecs are ~ 500 bps at least

ASR History

- First commercial success in “ASR”: Radio Rex (1920)
  - spring triggered by 500 Hz audio (~F1 formant of [ɛ] in “Rex”)
- 1950’-60’s – rule-based formant detection
  - digit recognition, isolated words
- 1970’s – first statistical modelling, HMMs
- 1980’s – larger models, adding language models
- 1990’s ~ first practically usable, large-vocab, continuous speech
- 2000’s – early neural approaches
- late 2010’s – fully neural, end-to-end ASR

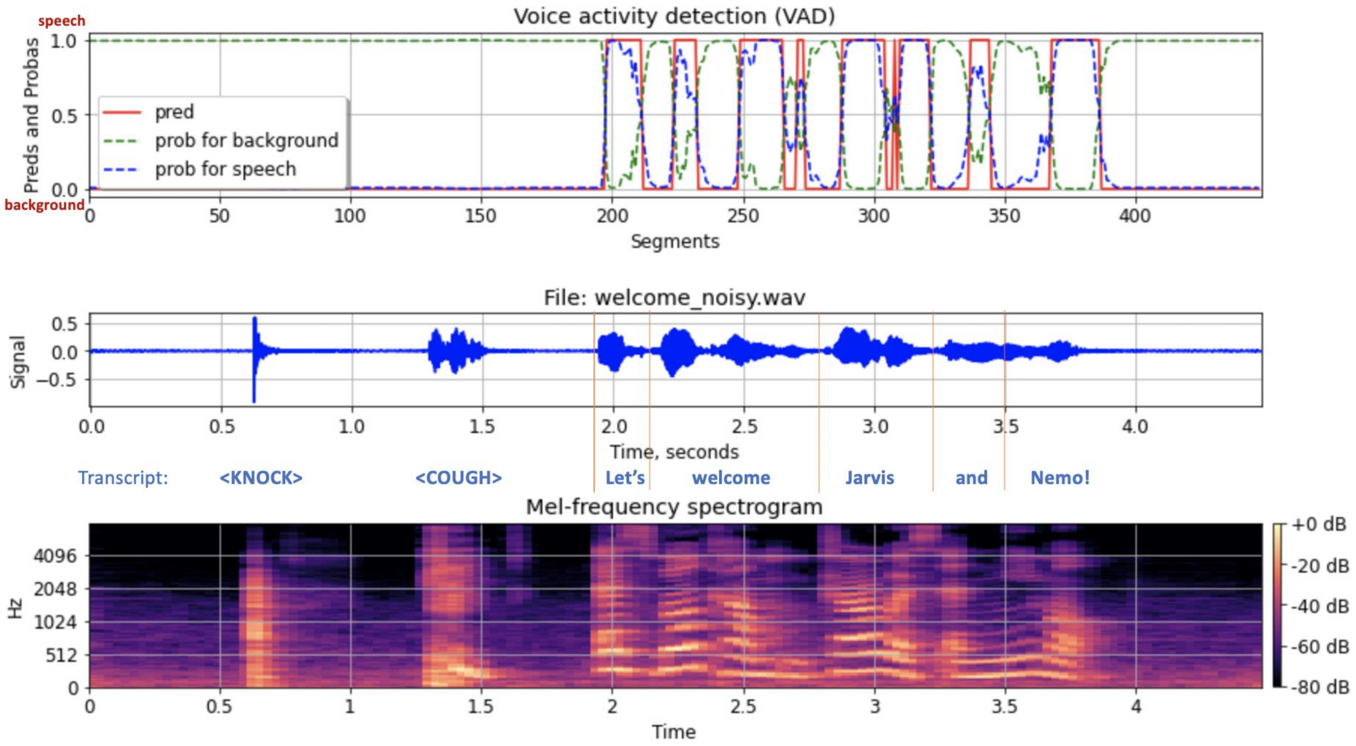
https://en.wikipedia.org/wiki/Speech_recognition

Conventional ASR



Speech Activity Detector

https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/zh/latest/voice_activity_detection/tutorial.html



Preprocessing step in ASR

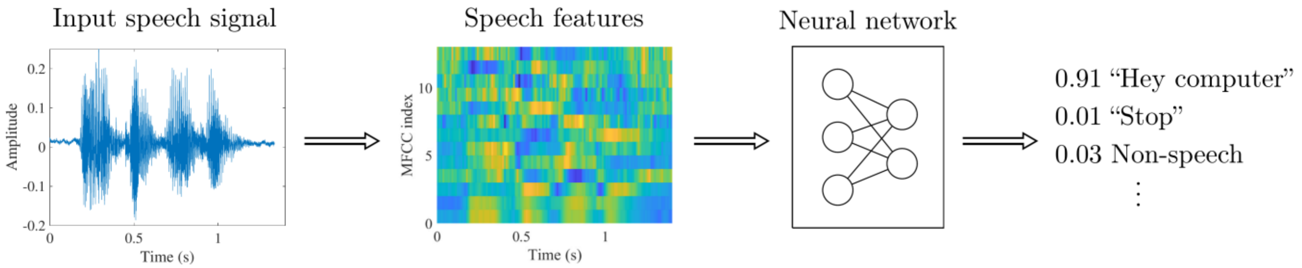
- Save CPU: run ASR only when there is speech
- Avoid confusing ASR with non-speech sounds

- Handcrafted (now obsolete)
  - Track signal amplitude contours
  - Simple, for low-resource tasks, assumes low noise
- Statistical / neural
  - Trained on large corpora to tell speech from other sounds – binary classifier
  - Input features same as ASR (→ →)
  - Accurate but more CPU-demanding
- basic smoothing needs to be applied

Wake words

- trigger to “start listening” (i.e. run full-scale ASR)
- simpler & more precise than VAD – detecting specific wake word
  - *OK Google, Alexa, Hey Siri*
  - simpler than to recognize that user is speaking to the system
  - simpler to distinguish from background noise
- basically a small-vocabulary ASR problem
  - ASR system running continuously
  - low-power, low-accuracy, but good enough for wake word

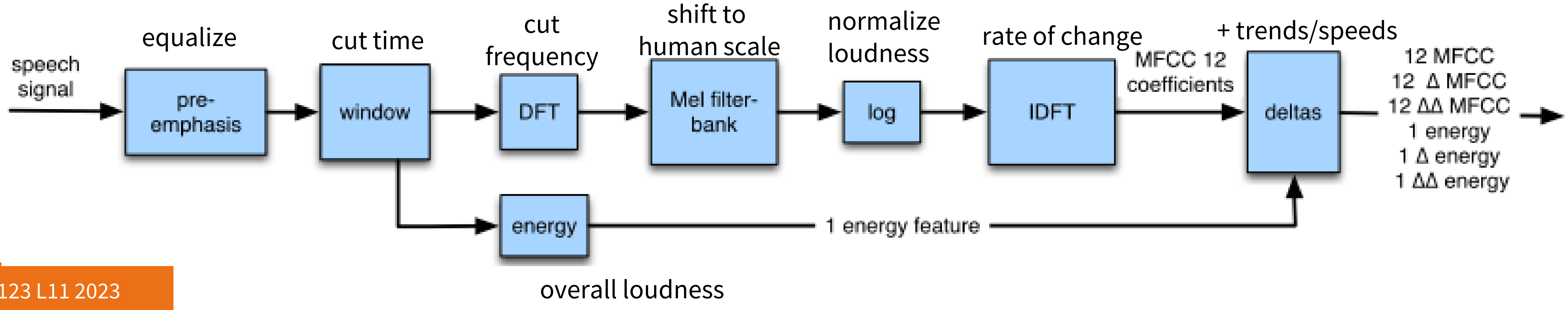
(small feed forward/convolutional)



https://speechprocessingbook.aalto.fi/Recognition/Wake-word_and_keyword_spotting.html

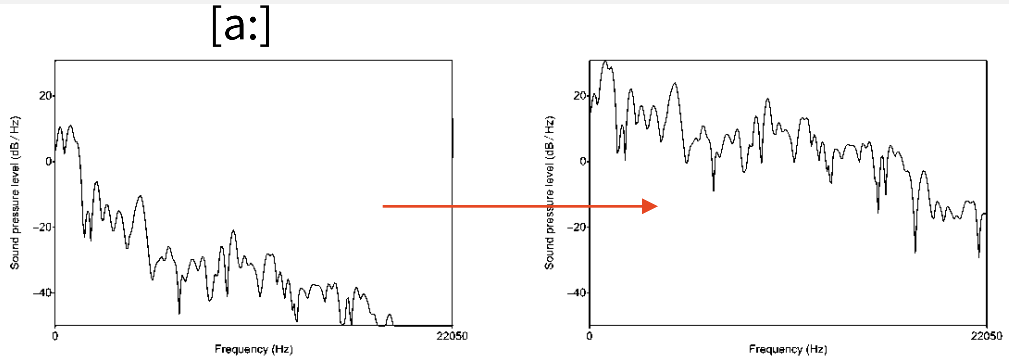
Features for ASR – Preprocessing

- In: Raw waveform ~ 1 number per 0.125 ms (8 kHz)
  - current pos. of the sound wave (~continuous) – sample, 8-bit/16-bit quantized
- Out: Mel Frequency Cepstral Coefficients ~ 40 features per 10 ms
  - step-wise (~discrete), dissected to frequency loudness & trends
- Inspired by humans:
  - information for 1 phone spans 250-400ms (coarticulation)
  - need to follow at least 4-7 freq. channels for intelligibility (10+ for better fidelity)
  - speech ~ 2-10 phones/sec (peak 4), auditory cortex reaction ~ 2-20 Hz

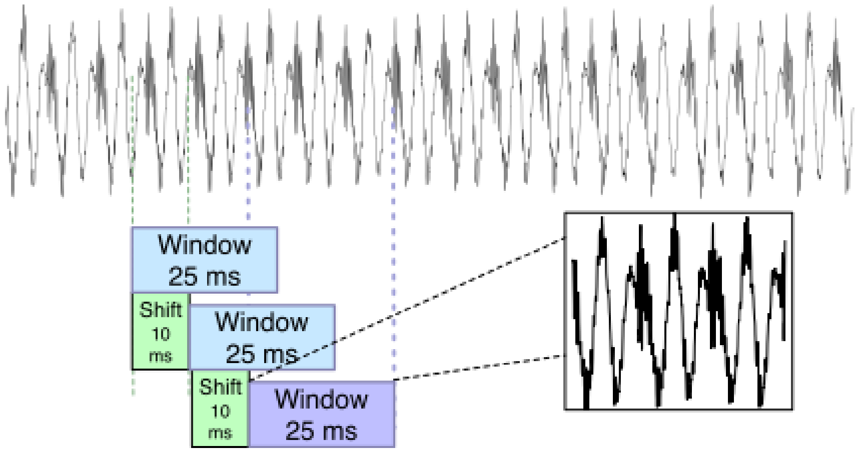


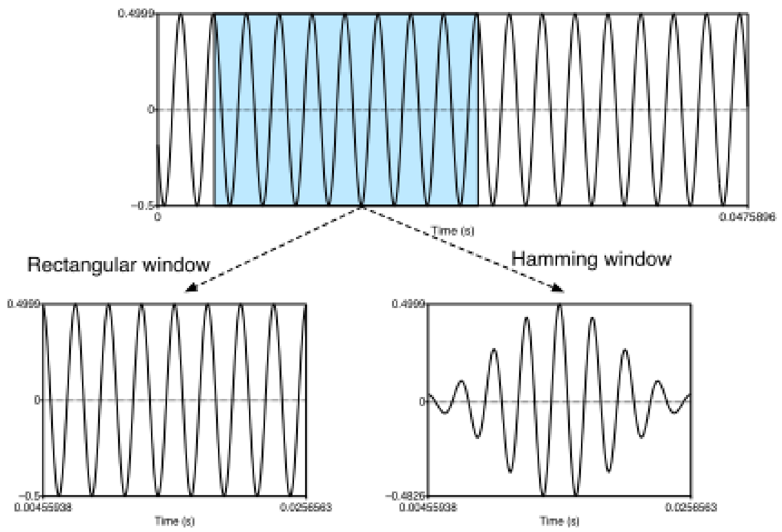
Features for ASR

- Preemphasis
  - boost higher frequencies (equalization)
- Windowing ~ frames
  - sliding: 25 ms / each 10 ms – overlapping
  - Hamming window – middle is emphasized
- Energy = overall loudness (+Δ, ΔΔ)



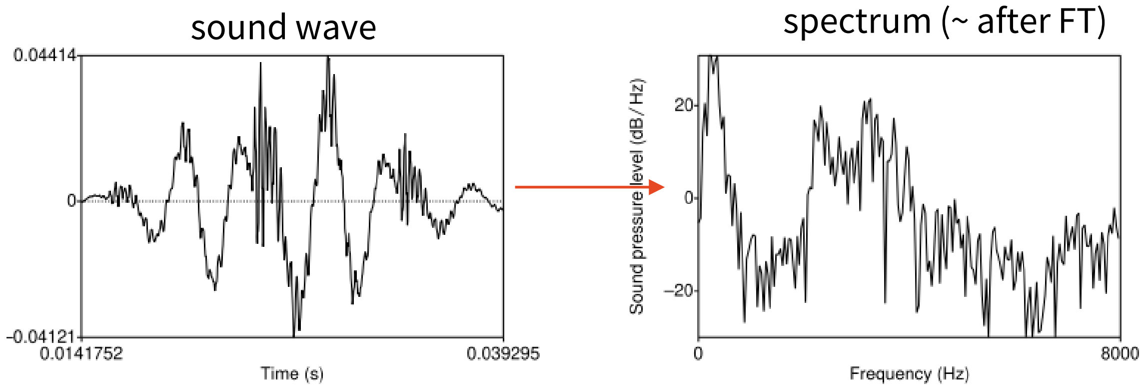
(Jurafsky & Martin, 2009)

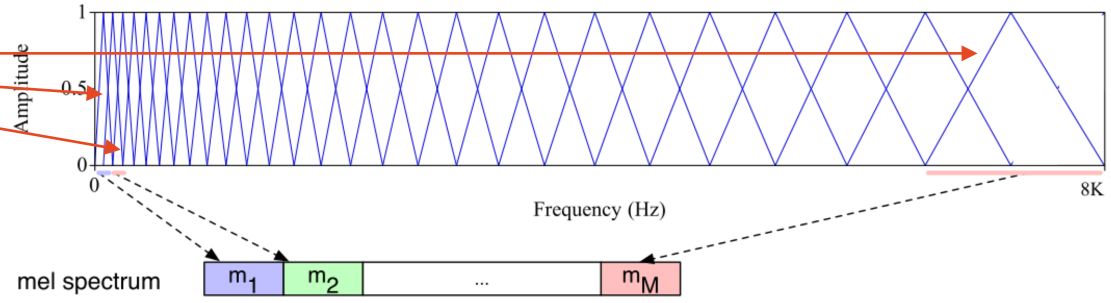


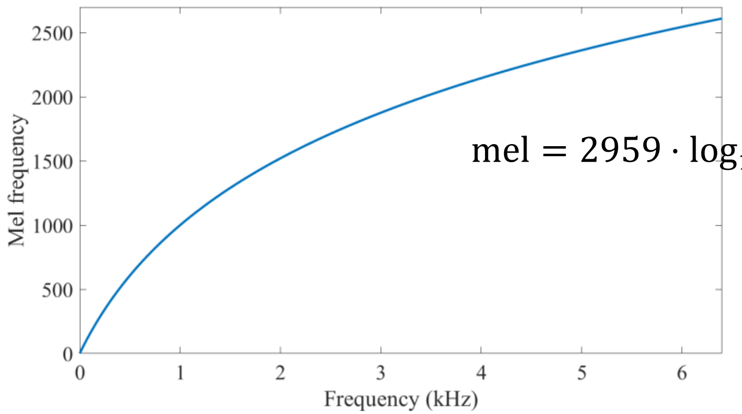


(Jurafsky & Martin, 2023)

- Spectrum – Fourier transform
  - loudness at different frequencies
- Mel bank filter
  - loudness at ~12-16 mel banks (i.e. frequency ranges)
    - using triangular frequency filters (sum everything within the filter)
    - ranges equal on mel scale (get wider in terms of normal frequency)
  - mel scale – logarithmic
    - corresponds to human perception of pitch







$$\text{mel} = 2959 \cdot \log_{10}\left(1 + \frac{\text{freq}}{700}\right)$$

Features for ASR

- Logarithmic volume
  - ~human-like, robust to loudness variation
- Cepstrum – another (inverse) Fourier transform
  - ~ "spectrum of log spectrum"
  - "rate of change in various spectral bands"
  - decorrelated (unlike filterbanks, which are overlapping)
  - slow changes – relevant to phones
    - ~ formants, other properties
    - usual speech: 2-10 phones per sec.
    - ~ only keep coeffs 2-13 (or thereabouts)
  - high range – harmonics (F0)
- Δ, ΔΔ: (× 3 features) – trends, speed of trends

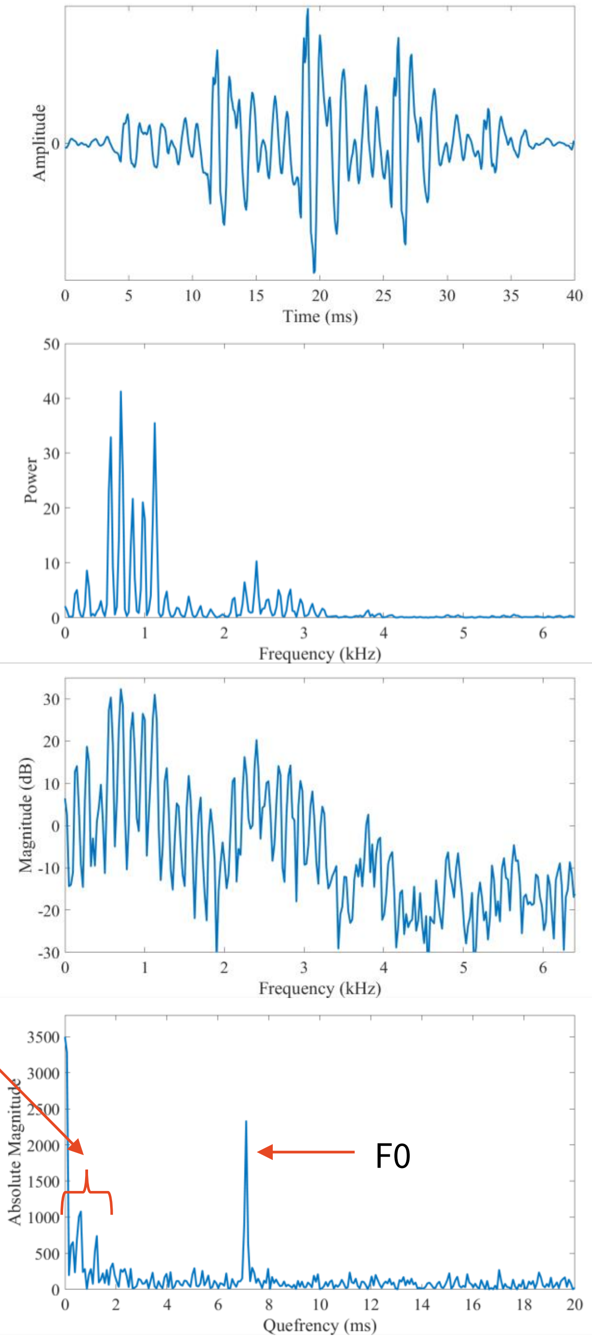
wave

spectrum

log spectrum

info about slowly changing
features of log spectrum
~ formants

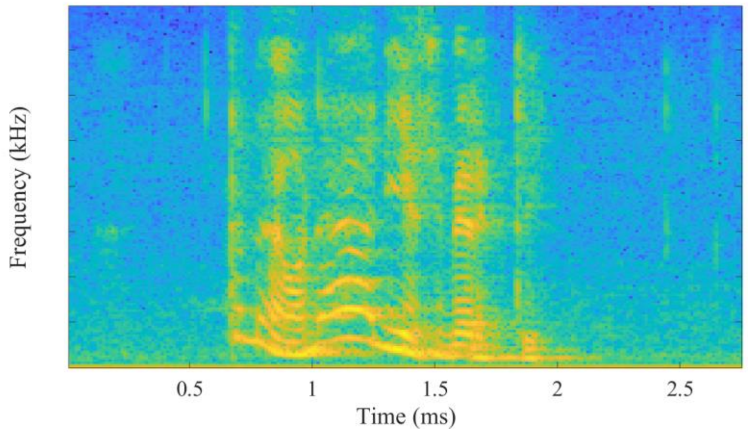
cepstrum



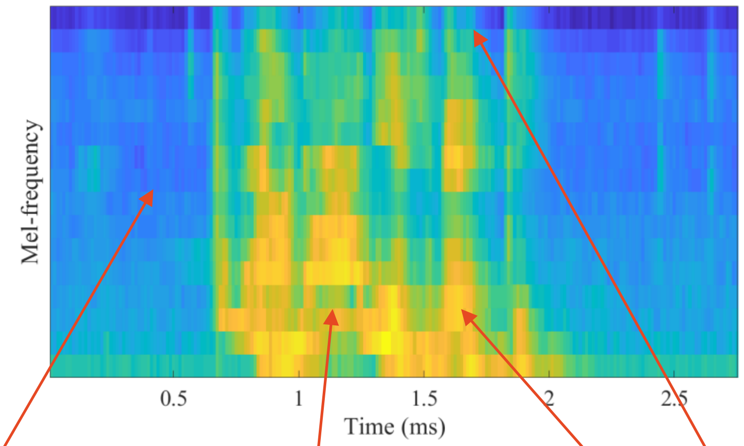
https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779
http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd
https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC

Features for ASR

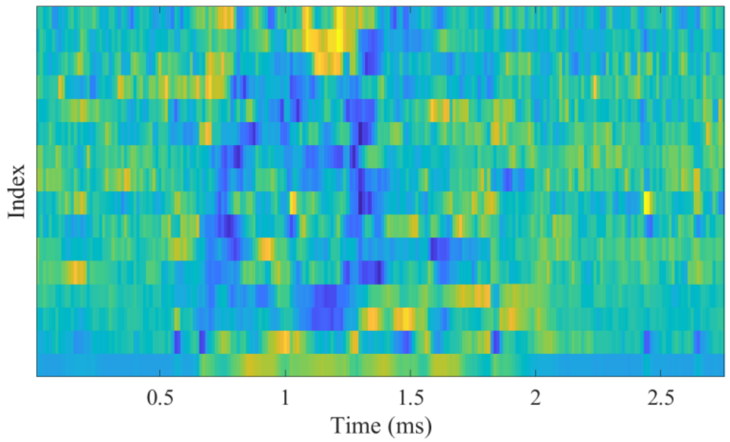
spectrogram



mel filtered spectrogram



MFCCs



less detail

general shape preserved

wider in low frequencies,
narrower in higher frequencies

hard to interpret,
uncorrelated

- MFCCs used mainly in older/low-resource systems
- newer: mel spectrograms (filterbank) / raw spectrograms / raw audio

https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC

Conventional ASR

- We want to model $P(\text{text}|\text{audio})$
- Can't model directly, so using Bayes:

$$P(\text{text}|\text{audio}) = \frac{P(\text{audio}|\text{text})P(\text{text})}{P(\text{audio})}$$

  - $P(\text{audio})$ is a constant, we're ignoring that
  - $P(\text{audio}|\text{text})$ ~ **acoustic model $\boldsymbol{P_A}$**
  - $P(\text{text})$ ~ **language model $\boldsymbol{P_T}$**
- **decoder** then combines information from both

Acoustic model

- $P_A = P(\text{audio}|\text{text})$, where
  - audio = ASR features, i.e. spectrograms
  - text = sequence of phone[me]s
- assuming independence between audio frames:

$$P(\text{audio}|\text{text}) = \prod_i P(a_i|t_i)$$

  - i – time (frame no.)
  - a_i – audio feature vector (~ spectrum)
  - t_i – acoustic class (~ phone[me], context-dependent phone)

Acoustic model

Spectrogram
(audio features)

true phones

silence

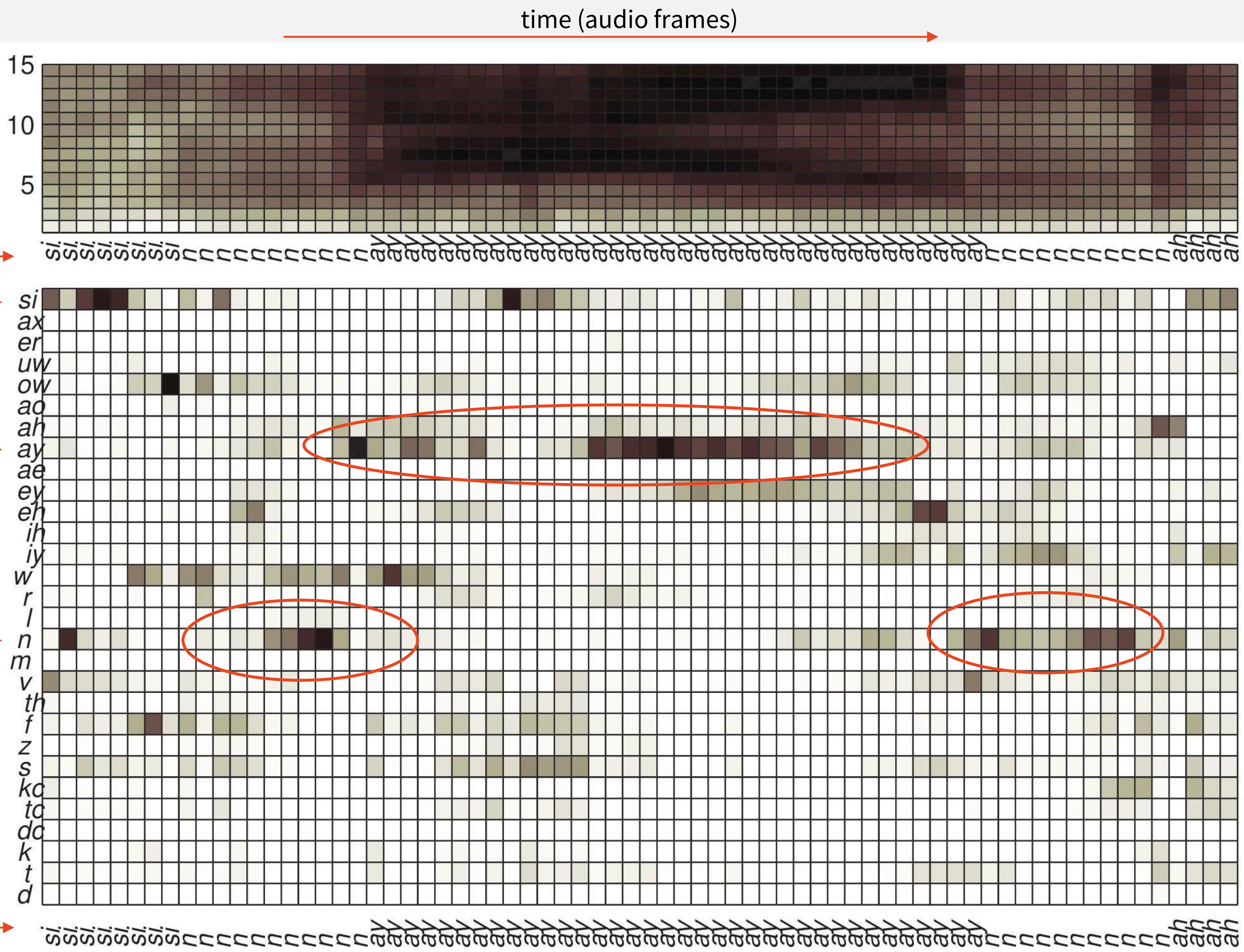
“nine”

[ai]

Acoustic model output
(phone posterior probs.)

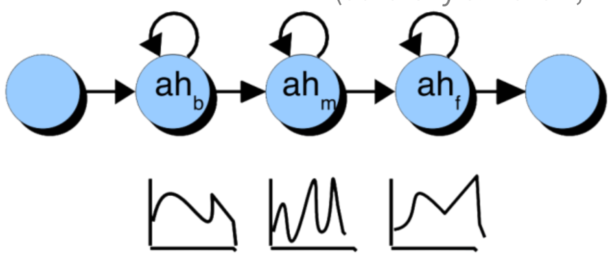
[n]

true phones

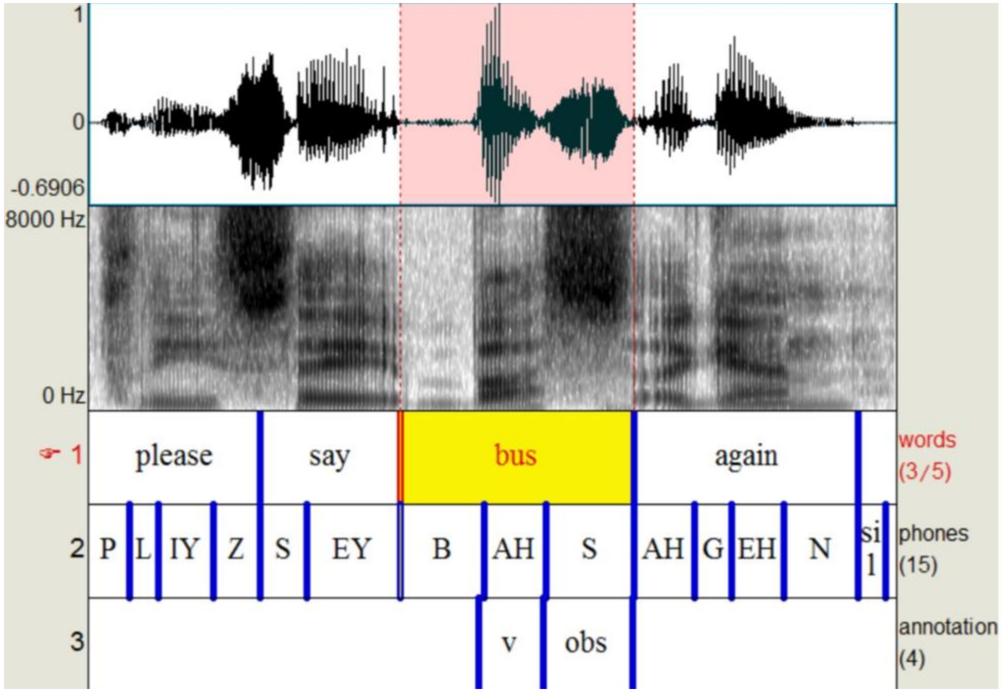


Acoustic model

(Jurafsky & Martin, 2009)



- Representing each phone by an HMM
  - start – mid – end, with loops (~ different lengths)
- Original: GMM – Gaussian mixtures
  - each HMM transition/emission is a multivariate Gaussian
  - clustering, as there are too many options
- Improvement: DNN
  (=feed forward neural net) instead of GMM
- Training – Baum Welch force-alignment
  - start from equal lengths of all phonemes, iteratively shift & increase likelihood
  - GMMs used to produce alignment to train DNN



Language model

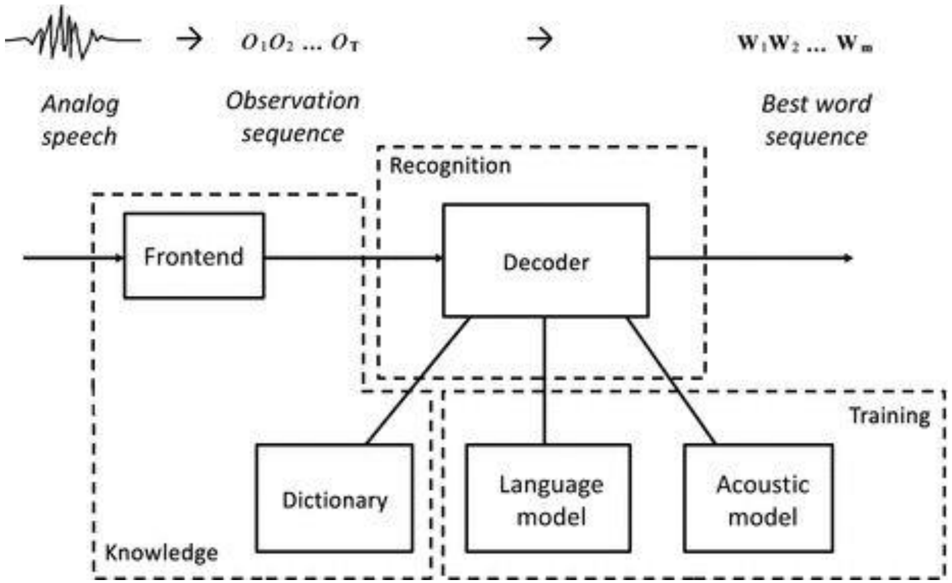
- $P(\text{text})$, where text ~ sentence, consisting of words $w_1, \ldots w_n$
- sequence probability modeled with a LM:

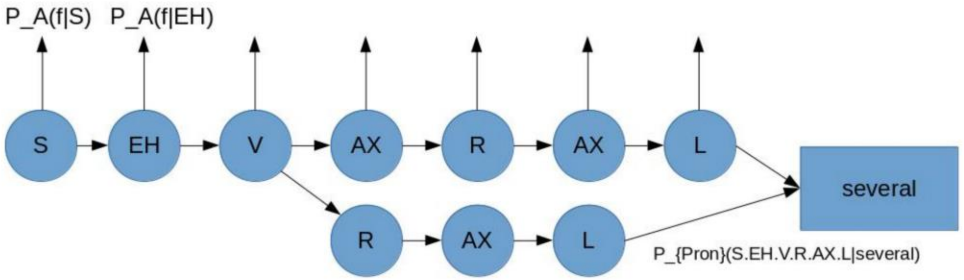
$$P_T(\text{text}) = \prod_i P(w_i | w_{i-1}, w_{i-2}, \ldots)$$

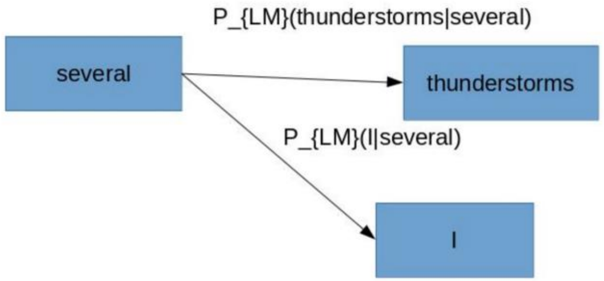
  - **words given preceding context**
  - traditionally n-gram LMs
  - more recently neural LMs
- Words w_i mapped to acoustic classes t_i using a pronouncing **dictionary**
  - or rules – essentially reverse of TTS's grapheme-to-phoneme conversion
  - multiple pronunciation variants considered
    e.g. S EH V AX R AX L ['sɛvəɹəl] vs. S EH V R AX L [‘sɛvɹəl]

Decoder

- Text *encoded* into acoustic signal / audio features → decoding back
- Hidden Markov Models
  - decoding word sequence from observed sequence of features
  - Dynamic programming (Viterbi)
  - Finding the best path through a finite state transducer
    composed of acoustic model & language model & pronouncing dictionary

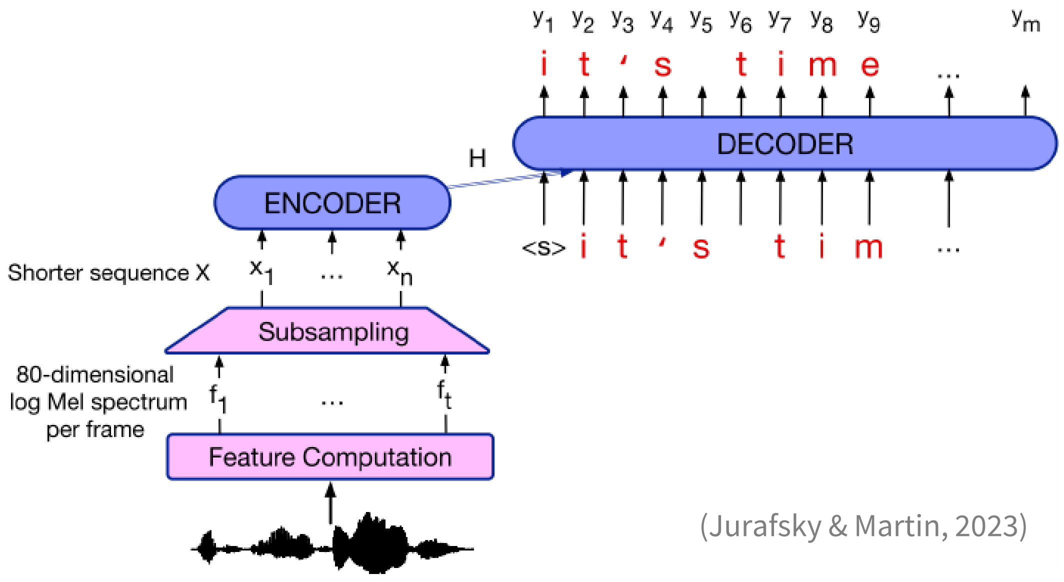






End-to-End ASR: Encoder-decoder

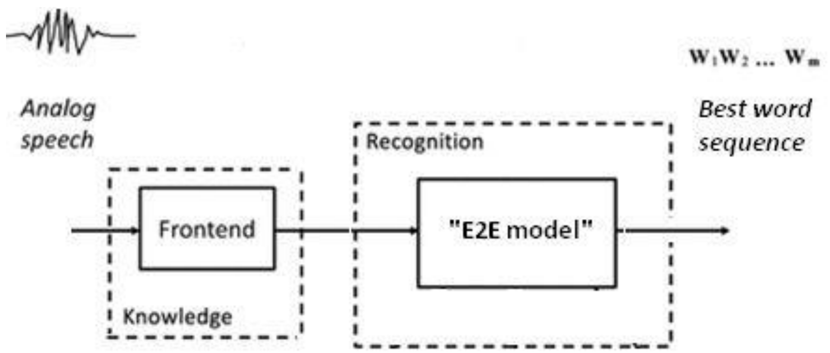
- Models $P(\text{text}|\text{audio})$ directly
- **Attention encoder-decoder** (AED) as in language tasks
  - a.k.a. listen-attend-spell (LAS)
  1. encode audio features
  2. decode text character-by-character
- RNN (LSTM) + attention / Transformer
- Audio is too fast/long → slowing it down ("low frame rate")
  - e.g. concatenate every 3 frames of audio ~ 40-dim → 120-dim at ⅓ speed
- Optional external language model: beam search & rerank



(Jurafsky & Martin, 2023)

Encoder-decoder ASR Pros & Cons

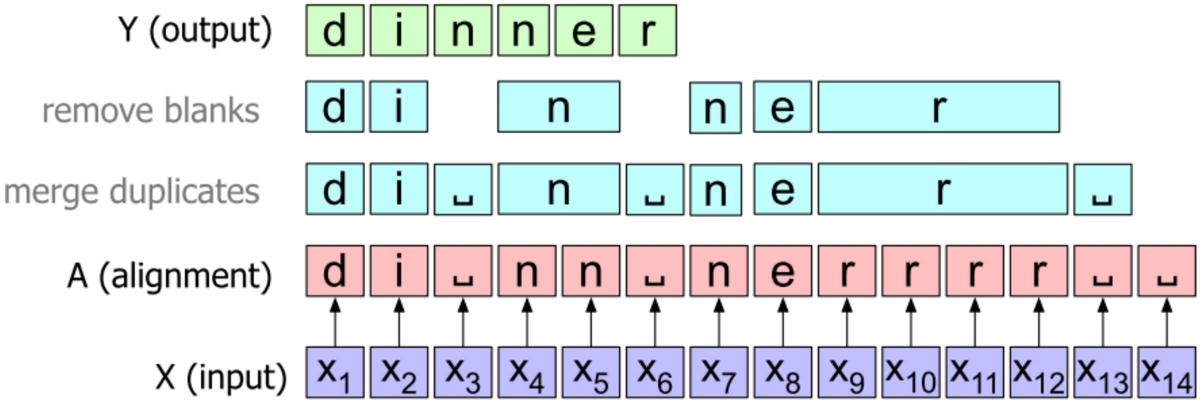
- Easier to train
  - pronunciation not modeled explicitly – direct audio to letter
  - no need to align phones & audio frames
  - audio & transcript is enough to train
- Easier to run – simpler decoder
- Inaccurate word/character timestamps
- Not low-latency
  - assuming whole sentence input → output
- Harder to customize: retrain everything
  - dictionary – unknown words may be guessed well as-is
  - language model – can use beam search & rescoring by an external LM



CTC (Connectionist Temporal Classification)

(Hannun, 2017)
https://distill.pub/2017/ctc/

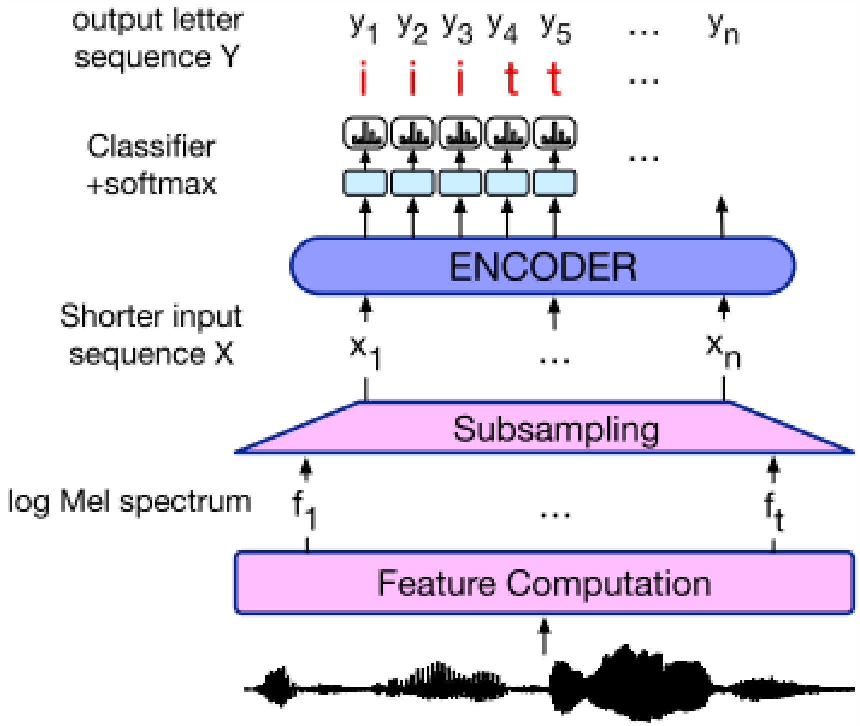
- Alt. idea: **predict something for every input frame**
  - **_/ε (“blank”)** for silence & double letters
  - collapse duplicates & remove blanks later
- Problem: many-to-one alignments
  - Many predicted sequences align to the same collapsed output
  - solution: clever summing
- training: minimizing **CTC loss**
  - sum over all possible alignments
  - computed by dynamic programming (forward-backward algorithm)
- inference: modified beam search
  - beam of output prefixes after collapsing



(Jurafsky & Martin, 2023)

CTC Model

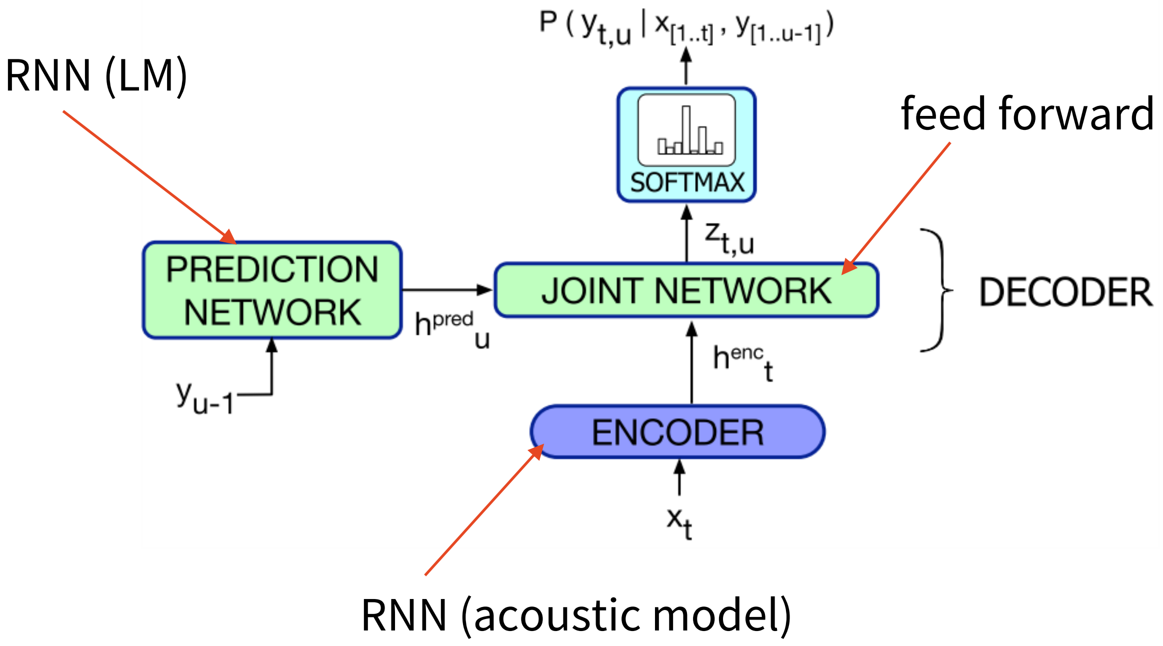
- Encoder + softmax classifier only
  - output something for every step
- Great for low latency
  - can work in parallel too
- Worse performance overall
  - strong assumption: outputs independent of each other (non-autoregressive)
- Can be combined with encoder-decoder
  - CTC as additional encoder loss
  - inference: combine probs. from both



(Jurafsky & Martin, 2023)

Transducers (RNN-T): Low-latency & accuracy

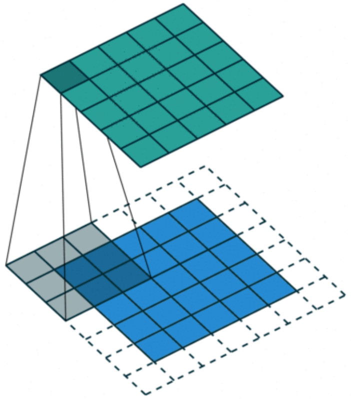
- Remove output independence
- Add RNN *prediction network* conditioned on prev. output
  - i.e. a language model component
- (RNN) acoustic model & RNN LM → joint (feed-forward) decoder
- Still predicts 1 output per frame
- All trained with CTC loss
  - You can retrain LM & keep acoustics
- Transformer variant (*s/RNN/Transformer/g*)



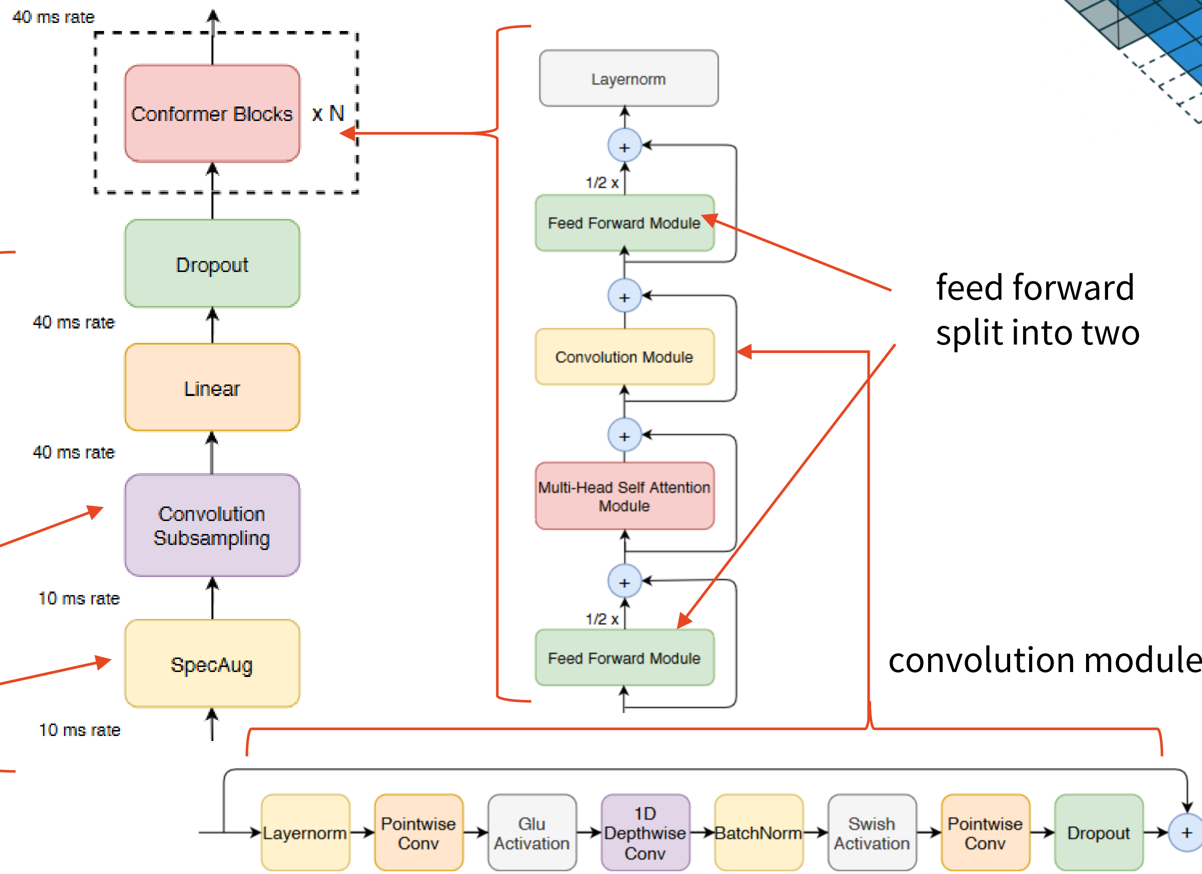
https://lorenlugosch.github.io/posts/2020/11/transducer/
(He et al., 2019) http://arxiv.org/abs/1811.06621
(Zhang et al., 2020) https://arxiv.org/abs/2002.02562

Conformer – better representation

https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d



- Transformer-like architecture, but with convolutions
  - CNN: applying same parameters (kernel) repeatedly over shifted inputs
- CNN for local interaction
- Transformer-like for global
- Used as acoustic model (encoder) in a Transducer



~embeddings

lower frame rate
by convolution (over time)

warping/masking
for robustness

feed forward
split into two

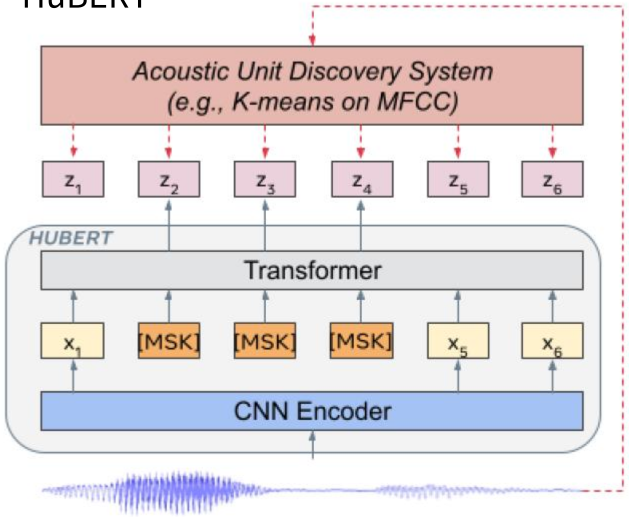
convolution module added

(Gulati et al., 2020) https://arxiv.org/abs/2005.08100

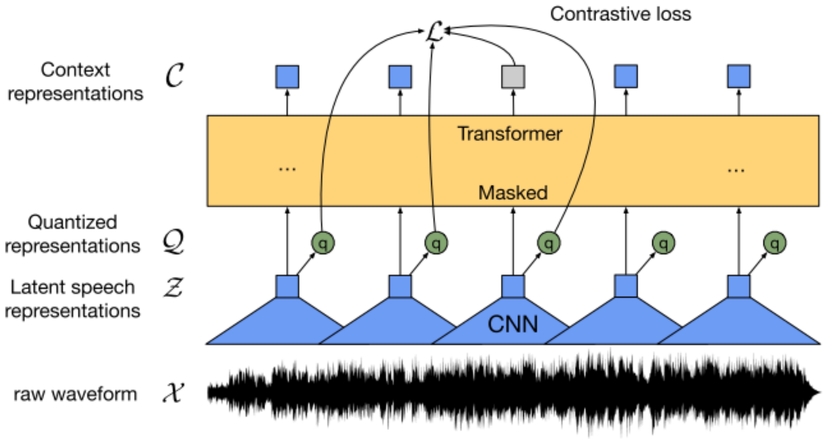
Self-supervised models

- Learning from large data without transcriptions
  - ~ 1000s of hours of audio
  - input: raw audio & convolutions
  - creating some inventory of pseudo-phonemes
    - HuBERT – clustering based on MFCC
    - Wav2Vec2 – jointly trained quantization
  - masking out some pseudo-phonemes
    & learning to predict them
- Finetuning on transcriptions (CTC loss)
  - works with ~ minutes of labeled data
- usable with Transducers / attention too

HuBERT



Wav2Vec2

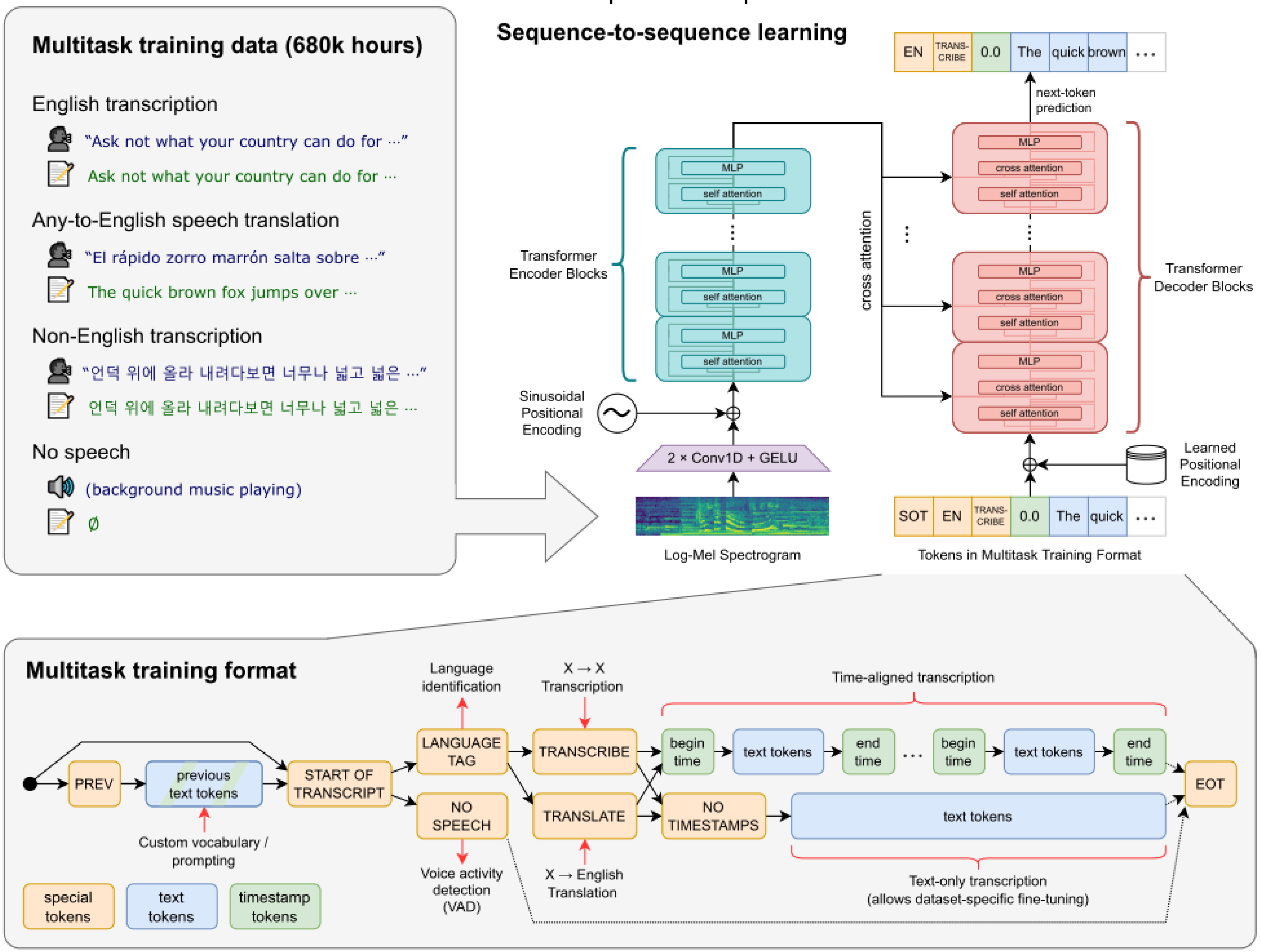


(Baevski et al., 2020) http://arxiv.org/abs/2006.11477
(Hsu et al., 2021) http://arxiv.org/abs/2106.07447

Weak supervision & multi-tasking

- Any transcriptions available
  - scraping the web (even low-quality)
- + speech translation
- + multiple languages
- aim: no finetuning

OpenAI Whisper



(Radford et al., 2022)
https://arxiv.org/abs/2212.04356

Challenges

- Human-human spontaneous speech harder than human-system
  - unscripted speech, disfluencies, repairs
  - stark topic shifts
- Specific domains
- Demographics
  - gender imbalances
  - non-native speech
- Language coverage
- Noise
- Latency/on-device

(Szymański et al., 2020) https://aclanthology.org/2020.findings-emnlp.295/

Summary

- VAD → features → ASR → text
- Features: MFCCs/filter banks/raw
- Traditional: separate acoustic & language models
- Neural:
  - Attention-based
  - CTC-based
  - Transducers
- Pretrained models
- Weak supervision

Thanks

Contact us:

https://ufaldsg.slack.com/
{odusek,schmidtova,hudecek}@ufal.mff.cuni.cz
Skype/Meet/Zoom (by agreement)

Labs in 10 mins

Get these slides here:

http://ufal.cz/npfl123

References/Inspiration/Further:

- Jurafsky & Martin's Speech & Language Processing (3rd ed., 2023): https://web.stanford.edu/~jurafsky/slp3/16.pdf
- Jurafsky & Martin's Speech & Language Processing (2nd ed., 2009)
- https://en.wikipedia.org/wiki/Speech_recognition
- https://speechprocessingbook.aalto.fi/Recognition_tasks_in_speech_processing.html
- https://wiki.aalto.fi/display/ITSP/Introduction+to+Speech+Processing