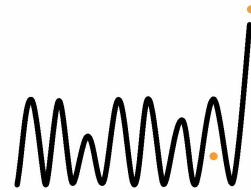


# Automatic Speech Recognition

Petr Fousek

thanks to: Pavel Kveton, Michal Juza



THE MAMA AI

# Outline

- Intro
- Conventional ASR
  - Front-end
  - Model
  - Decoder
- End-to-end ASR

# Introduction



First commercial success in ASR:  
Radio Rex (1920)

A celluloid dog released by a spring when triggered with a 500 Hz acoustic energy (roughly the first formant of the vowel [eh] in “Rex”).

"Humans evolved to optimally use the acoustic channel to communicate - why not to inspire by them?"

- Fletcher – temporal and frequency masking – 1950's
- Information about a phone spans 250 - 400 ms - coarticulation
- 4-7 frequency channels min. for intelligibility, >10 for fidelity
- modulations about 2-10Hz (peak at 4Hz ~ 250ms)
- auditory cortex response also in 2 - 20Hz range

# Information in audio

Audio contains lots of information irrelevant for ASR:

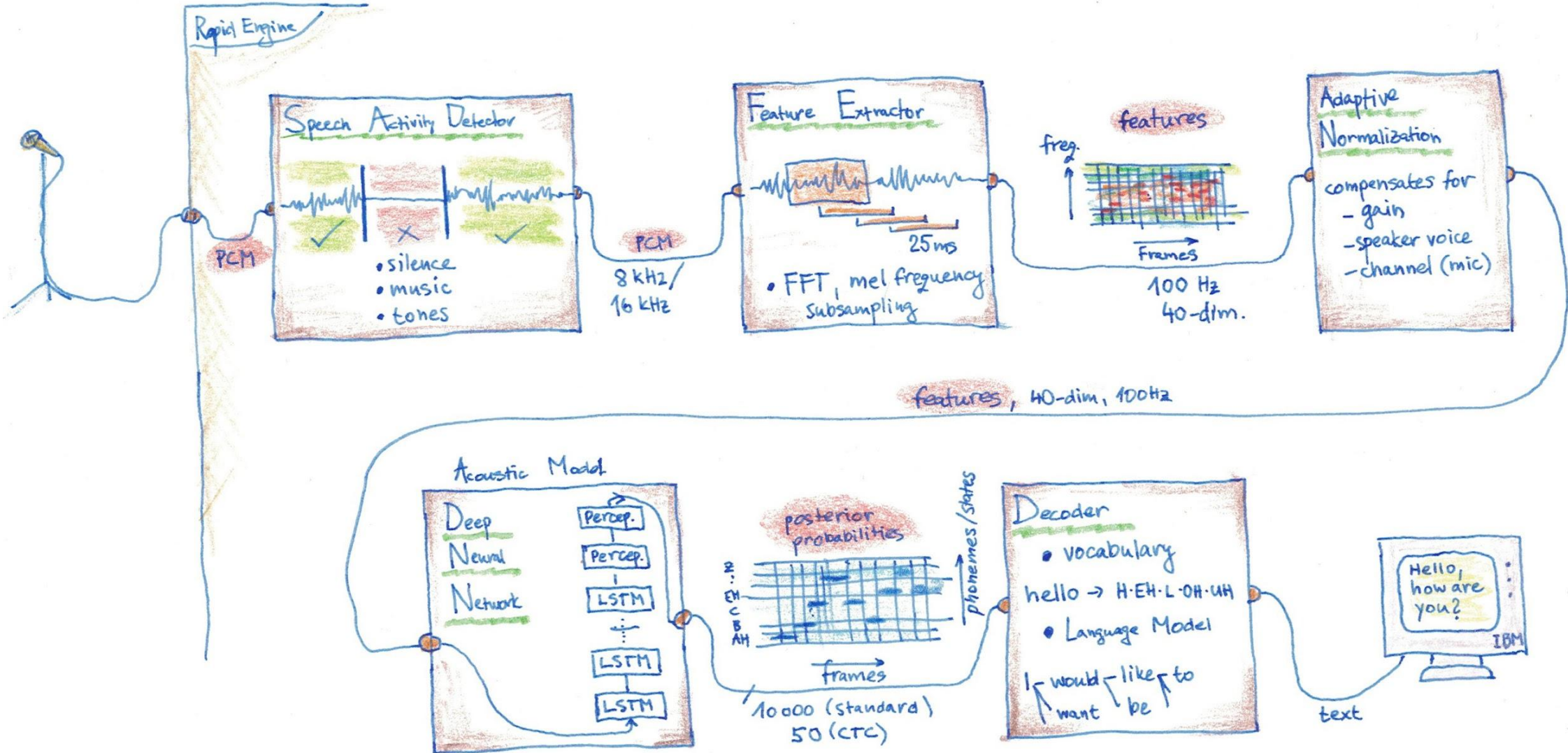
- Speaker identity (gender, age, speaking style, dialect)
- Speaker's state (emotions, health)
- Environment (ambient sounds, reverb/echo of the room)
- Distortions (noise, channel effects, distance)

How many bits per second we need:

- to store speech, we need about 64 kbps (8kHz, 8bits)
- low-bitrate audio codecs: down to 500 bps
- text: about 50 bps

... so ASR is an ultimate speech compressor.

# Conventional ASR



# Speech Activity Detector

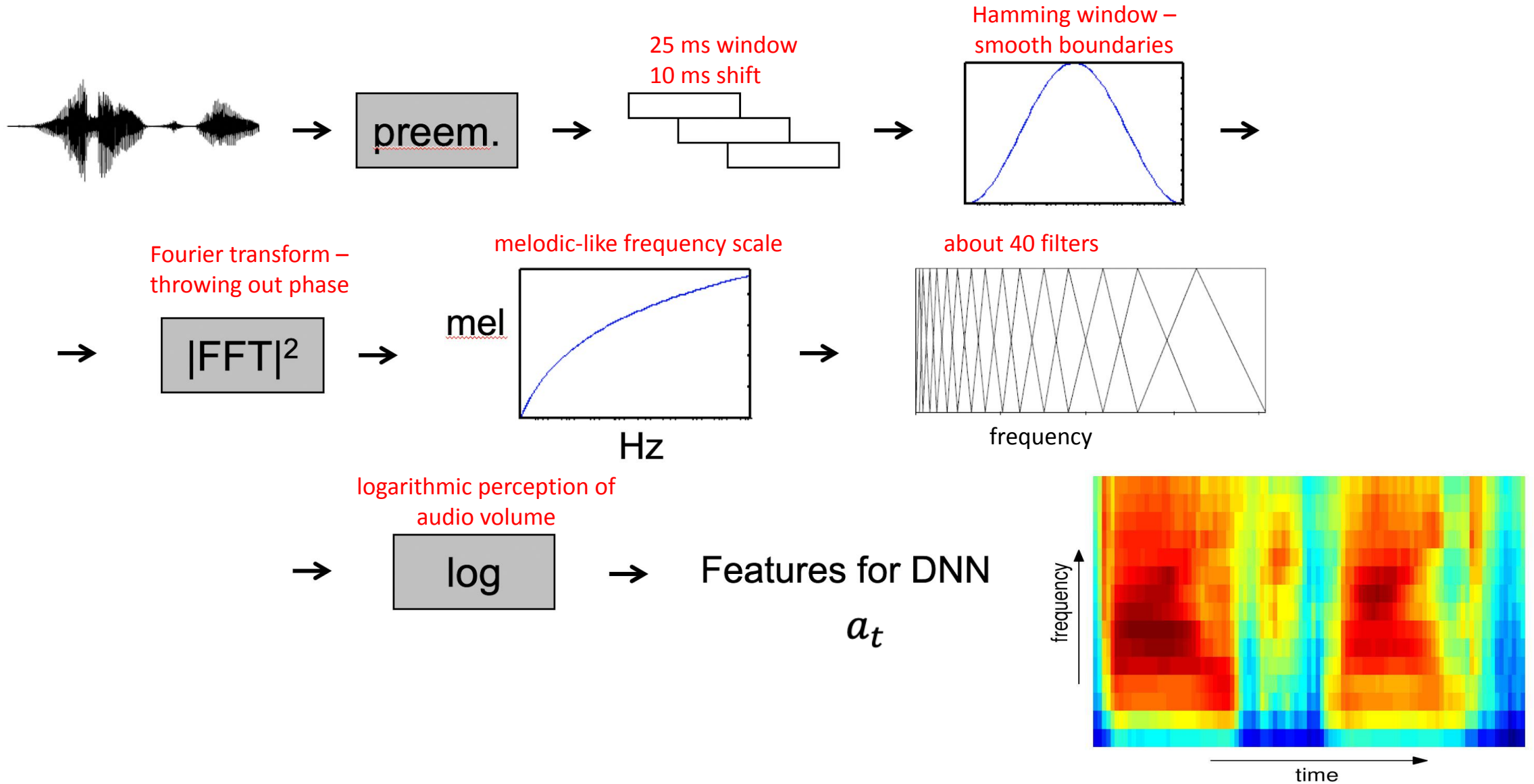
## Preprocessing step in ASR

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

## Technologies:

- Hand-crafted (now obsolete)
  - Track signal amplitude contours
  - Simple, for low-resource tasks
- NN Model-based
  - A model trained on large corpora to tell speech from other sounds
  - Input features often shared with ASR - log mels suffice
  - Accurate but more CPU-demanding

# Features for ASR





# Conventional speech model

$$P(T|A)$$

T ... text, A ... acoustics

- Unable to model  $P(T|A)$  directly, so using Bayes:

$$P(T|A) = \frac{P(A|T)P(T)}{P(A)}$$

- $P(A)$  constant, ignoring
- $P(A|T)$  ... acoustic model  $P_A$
- $P(T)$  ... language model  $P_T$

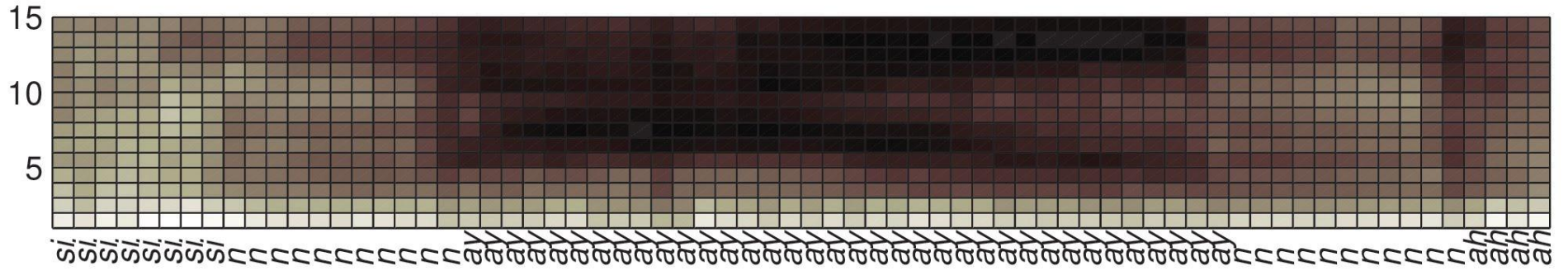
# Acoustic model $P_A(A|T)$

- $T$  ... hypothesized sequence of acoustic units
- we assume independence between frames so that we can write

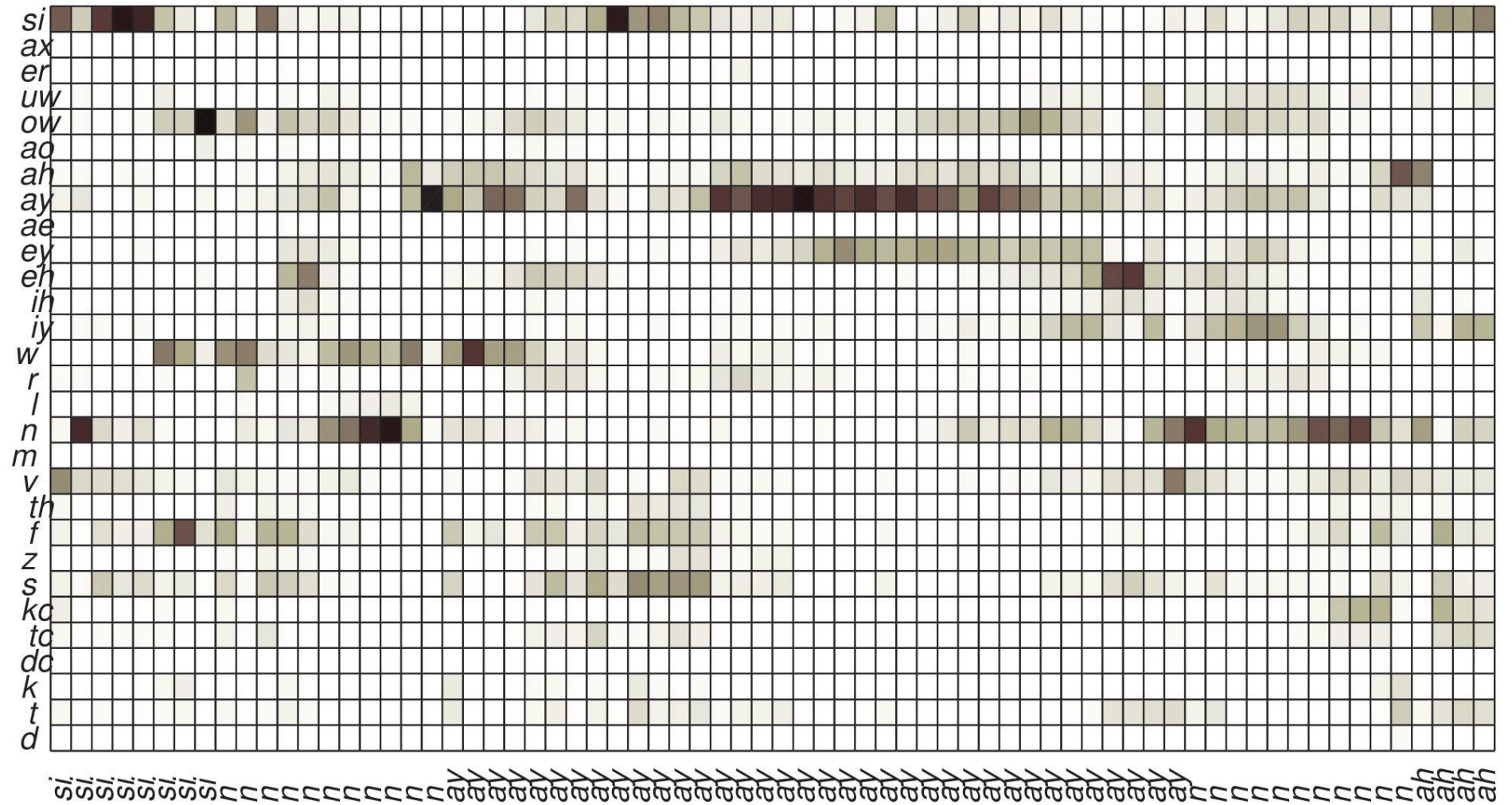
$$P(A|T) = \prod_i P_A(a_i | t_i) \quad i \dots \text{time}$$

- $i$  ... time
- $a_i$  ... feature vector
- $t_i$  ... acoustic class (a phone or context-dependent phone)

Spectrogram



features for the  
word "nine" –  
n-ay-n



# Language model $P_{LM}(T)$

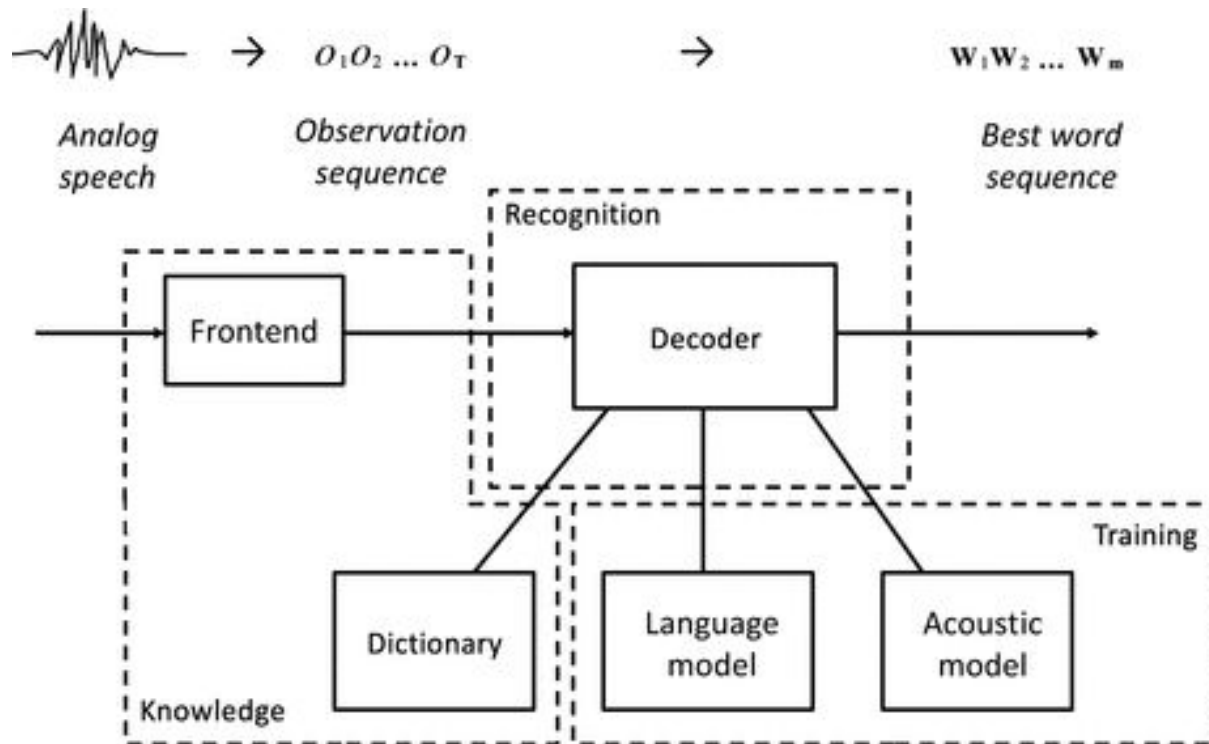
- $T$ ... sentence, consists of word sequence  $w_1 \dots w_N$
- sequence probability modelled with n-gram LM (or neural LMs)

$$P_{LM}(T) = \prod_i P(w_i | w_{i-1}, w_{i-2}, \dots)$$

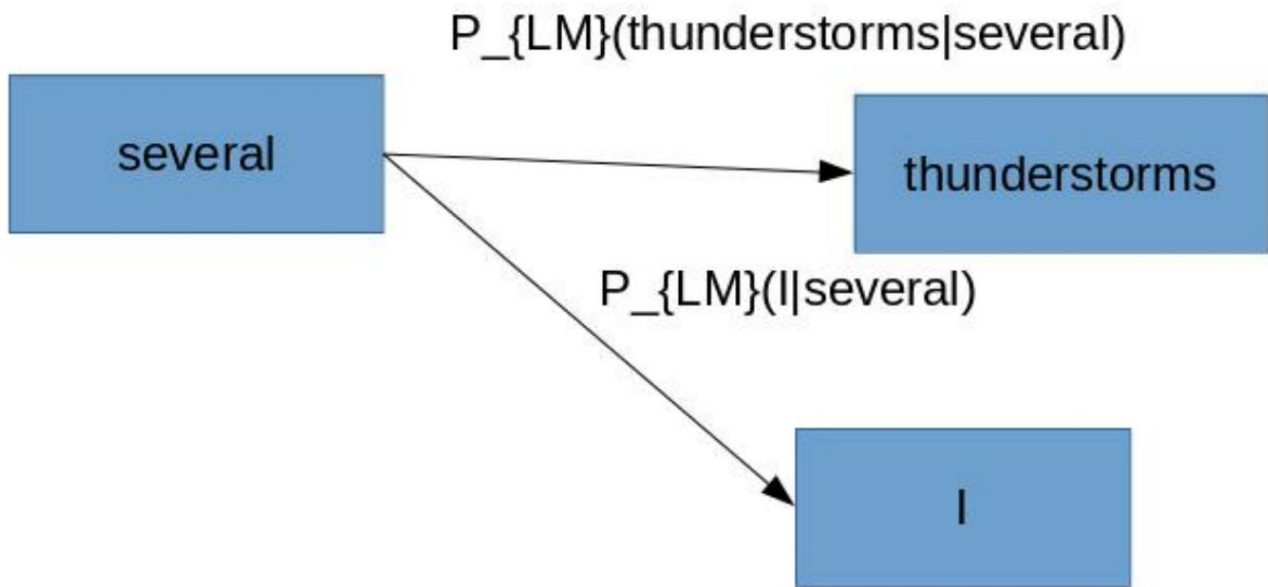
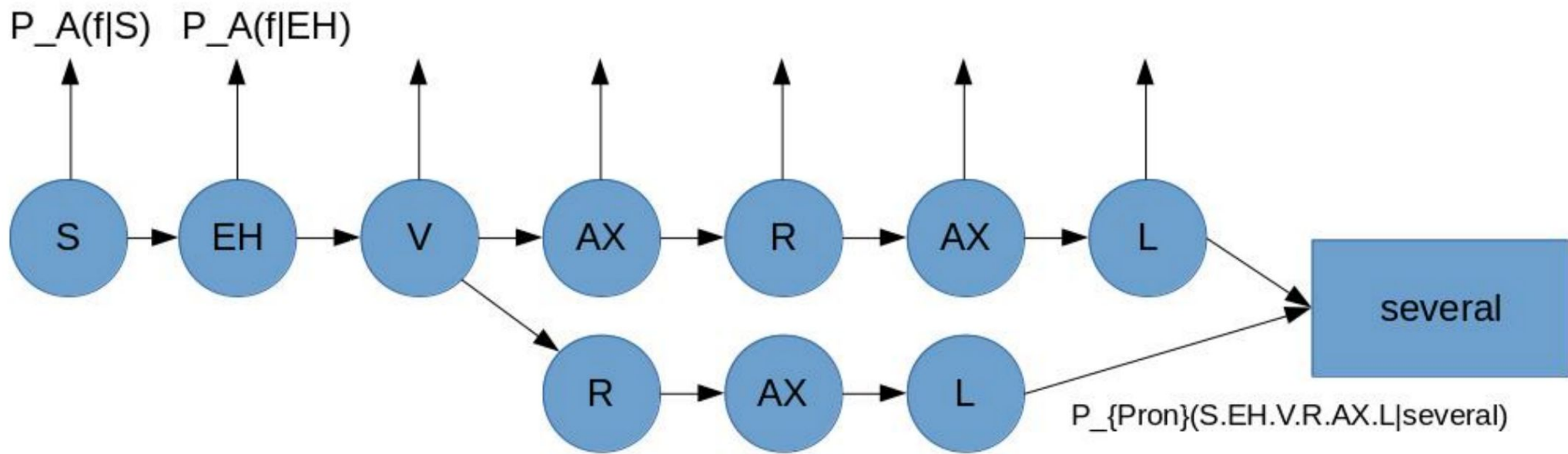
How to map words  $w_i$  to acoustic classes  $t_i$  ?  $\rightarrow$  need **Dictionary**  $P_{pron}(pron|w)$

- "several" S EH V AX R AX L
- "several" S EH V R AX L

# Decoder



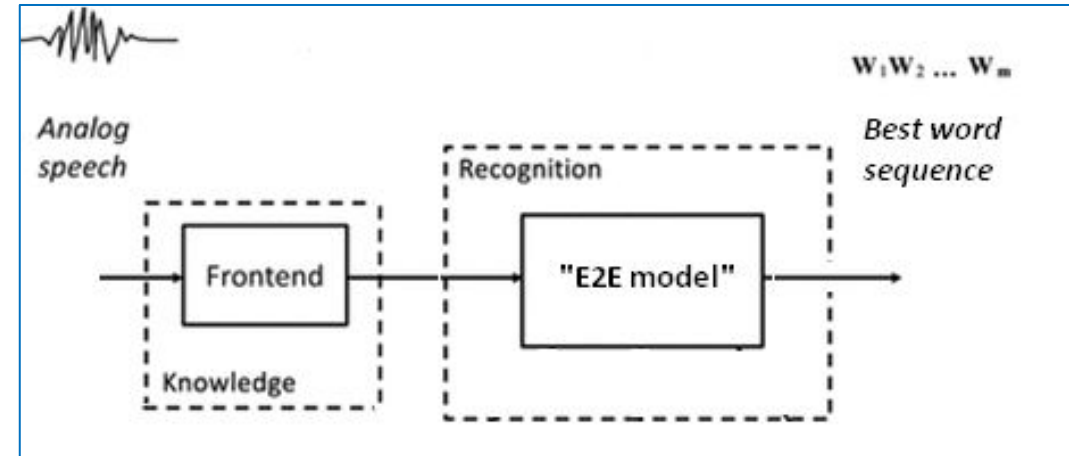
- Why "decoder"? - the text has been *encoded* into acoustic signal (or features), now we attempt to *decode* it back
- Hidden Markov Models - decoding hidden message (sentence) from the sequence of features
- Dynamic programming (Viterbi) to find the best path through the FST (composed from AM, LM, Dictionary)



# End-to-end (E2E) ASR

## E2E model

- Models  $P(T|A)$  directly
- Able to predict per-frame probability of characters, sub-words or even words
- No need for Dictionary (pronunciation is not modelled explicitly)
- No separate acoustic model → no need for alignment between symbols and audio
- Decoder can be much simpler
- All you need to build the model is audio and its transcript



# E2E model downsides

- **Inaccurate word/character time stamps**
  - because there is no explicit symbol alignment
- **Not streaming by nature**
  - Encoder-decoder scheme with attention blocks is great for parallelization but not for low-latency response
- **Customization needs workarounds:**
  - Hard to add new words (but e2e model often guesses them right)
  - Acoustic model customization is tough - joint model
  - Language model customization - external neural model or phone-level n-gram



# Towards streaming E2E model

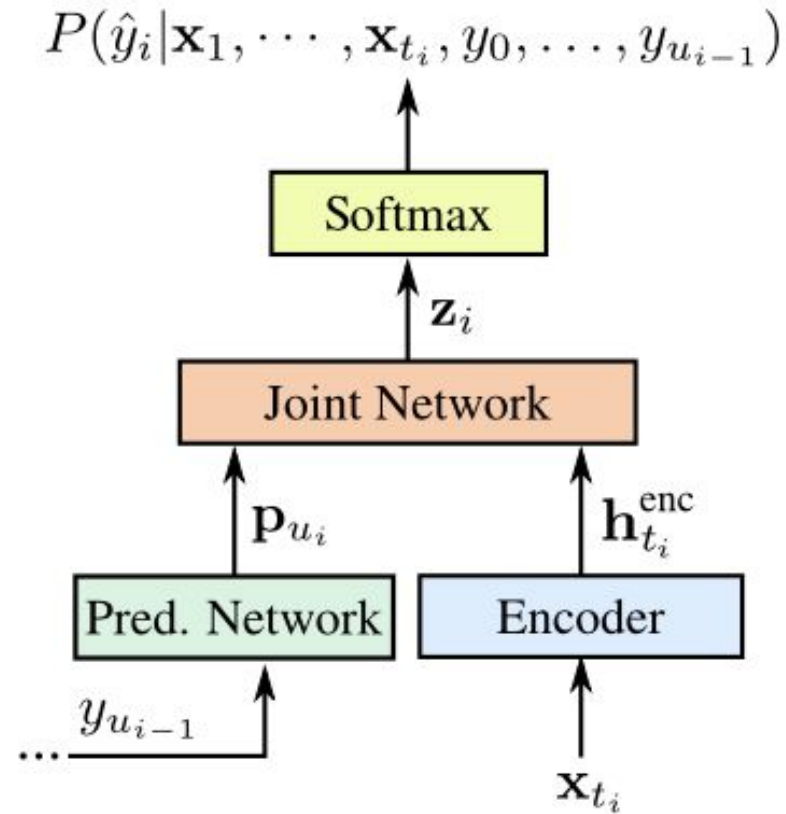
## CTC - Connectionist temporal classification

- training objective independent on alignment between acoustic input (dense) and labels (sparse)
- introduces “blank” symbol \_
- alignments “h \_ \_ e l \_ l \_ \_ o” and “h e l l \_ \_ o \_ \_ \_” are equivalent

## RNN-T

- CTC assumes independence of output on previous outputs
- RNN-T has explicit model of output sequence - prediction network
- allows training “language model” NN independently

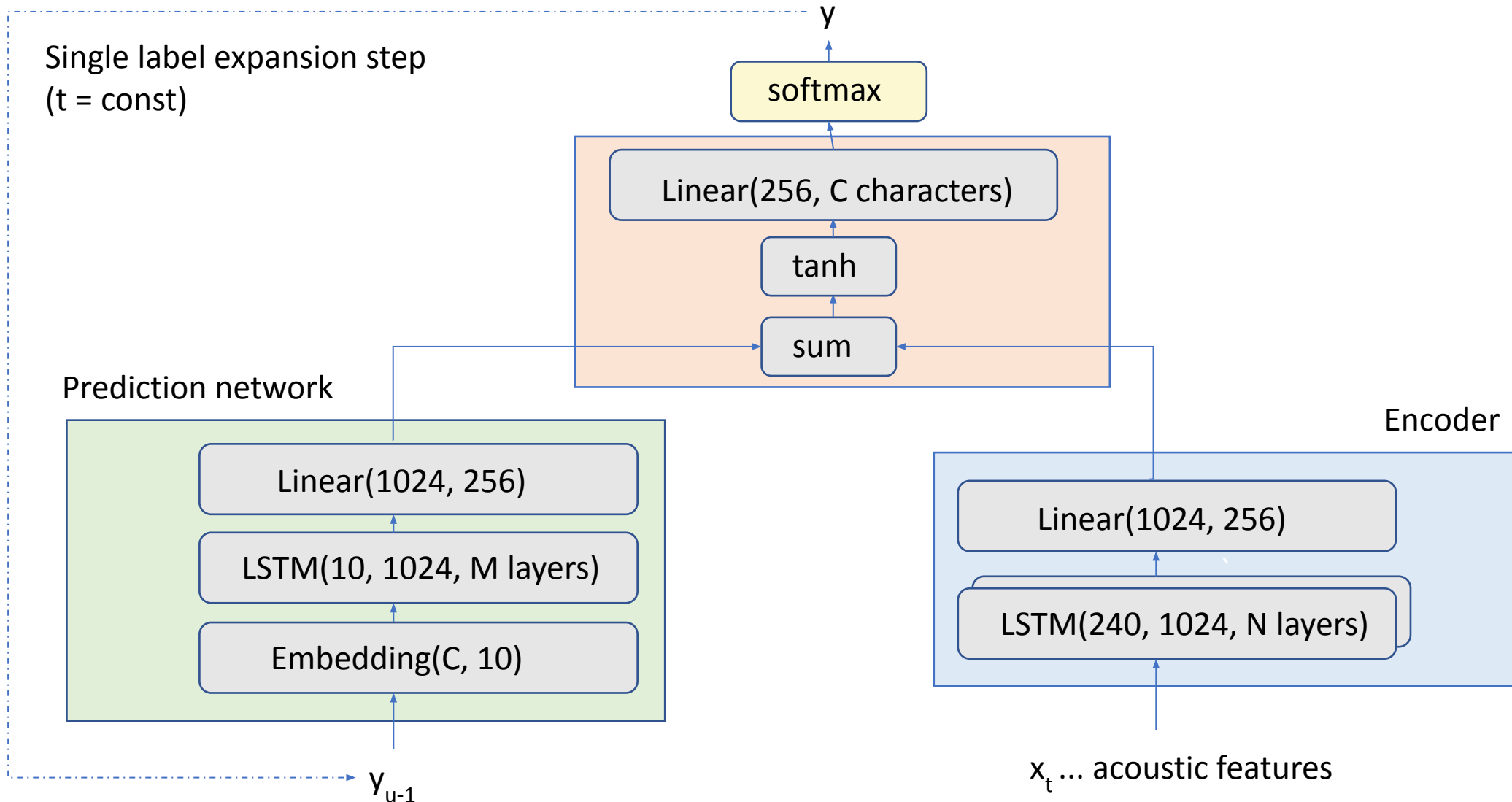
# RNN-Transducer model



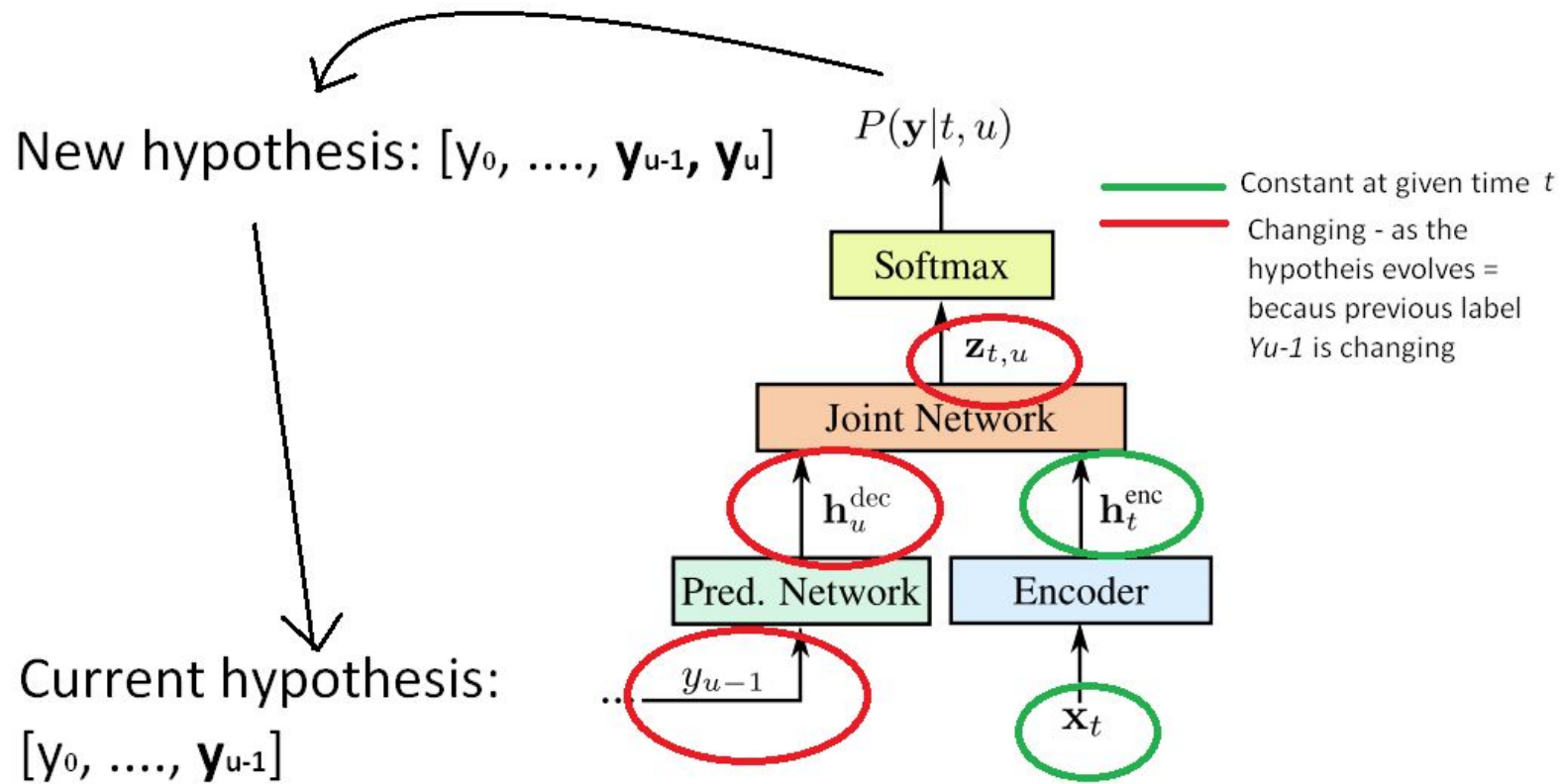
(b.) RNN-Transducer

\* "Streaming End-to-end Speech Recognition for Mobile Devices", ICASSP'19, Google

# RNN-T model architecture – an example



# RNN-T Decoding



- Beam search
- RNN-T gives  $P(y|x, y_-)$ , where  $y$  is the next character,  $x$  are the audio frames so far,  $y_-$  is the current hypothesis
- RNN-T does not always consume input (allows to decode multiple characters in a single frame)

# RNN-T loss

- For each frame, NN outputs probabilities of characters + blank symbol
- We have the correct transcript at train time
- An alignment is a sequence of characters + blank symbols
- A consistent alignment is one consistent with the correct transcript, e.g.
  - Let's say we have 8 acoustic frames and a transcript "hello"
  - an alignment "  he  llo    " is consistent
  - so is " h e  llo    "
  - by definition, blank symbol means go to next frame
- RNN-T optimizes the sum of probabilities across *all* consistent alignments

# Extensions

- Streaming with bi-directional RNNs - chunking model
- [Alignment-length-synchronous decoding](#)
- External LM - shallow fusion, n-gram LM, grammars
- Transformers [can also do streaming](#) - limited context and RNN-T loss

END