Natural Language and Speech Processing

Lecture 11: Speech Features and End-to-End Speech Recognition

Tanel Alumäe

Contents

- What is speech recognition (and what it isn't)
- Feature extraction
  - Useful for other speech processing tasks, e.g. speaker recognition, language recognition
- Two end-to-end speech recognition architectures

Speech recognition

- Speech recognition is a technology that converts speech to text
  - It doesn’t handle anything else, like speech (text) understanding
  - Common misconception is that speech recognition = speech understanding
- Applications:
  - Dictation
  - Transcribing meetings, lectures, videos, telephone calls
  - A component in human-computer interaction systems
    - Google Assistant, Siri, etc
    - Amazon Echo, Google Home, etc
  - A component in speech-to-speech translation systems

Why is speech recognition difficult?

- There are many different words (many are very similar), especially in morphologically complex languages (like Estonian)
  - *kast, kass, kasti, kas, kastid*
- The vocabulary is higly dependent on the domain
  - E.g., Estonian radiology domain: *Nimmelordoos on säilinud; täheldatav on vähene sinistroskolioos; spondülolisteesi ei ilmne.*
- Most importantly, there is a lot of variery in speech

Sources of variability

Müra
- Teised kõnelejad
- Taustamüra
- Ruumiakustika

Moonutused
Müra
Kaja
Infokadu

KANAL

Kõne-tuvastus-süsteem

Mikrofon
- Moonutused
- Elektriline müra
- Suunaomadused

Kõneleja
- Häälekvaliteet
- Põhitoon
- Sugu
- Keel/dialekt

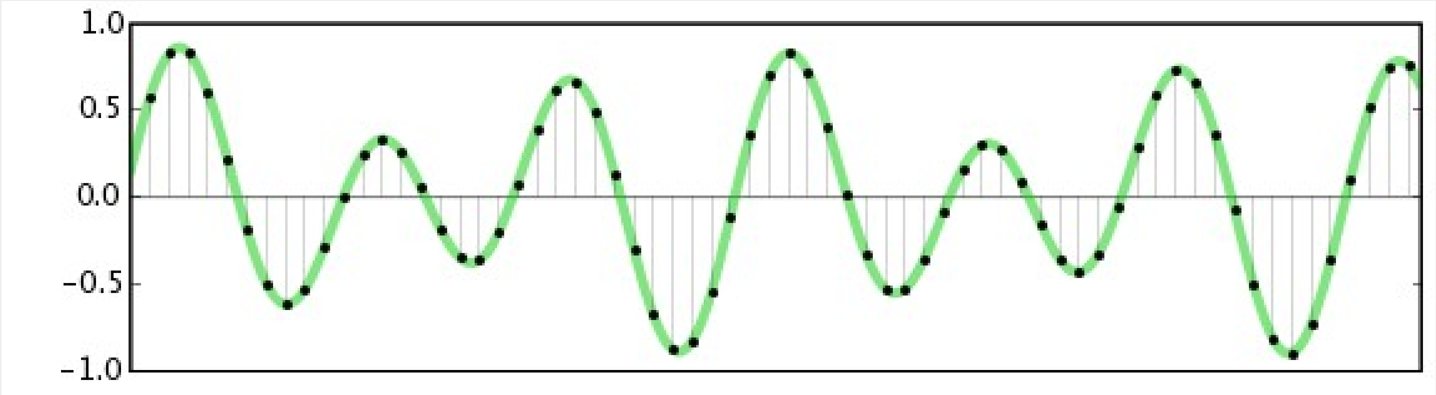
Kõnestiil
- Emotsioonid
- Kõnetempo
- Lombard'i efekt

Ülesanne
- Inimene-masin dialoog
- Dikteerimine
- Suhtlus
- Intervjuu

Foneetiline/ prosoodiline kontekst

Speech signal

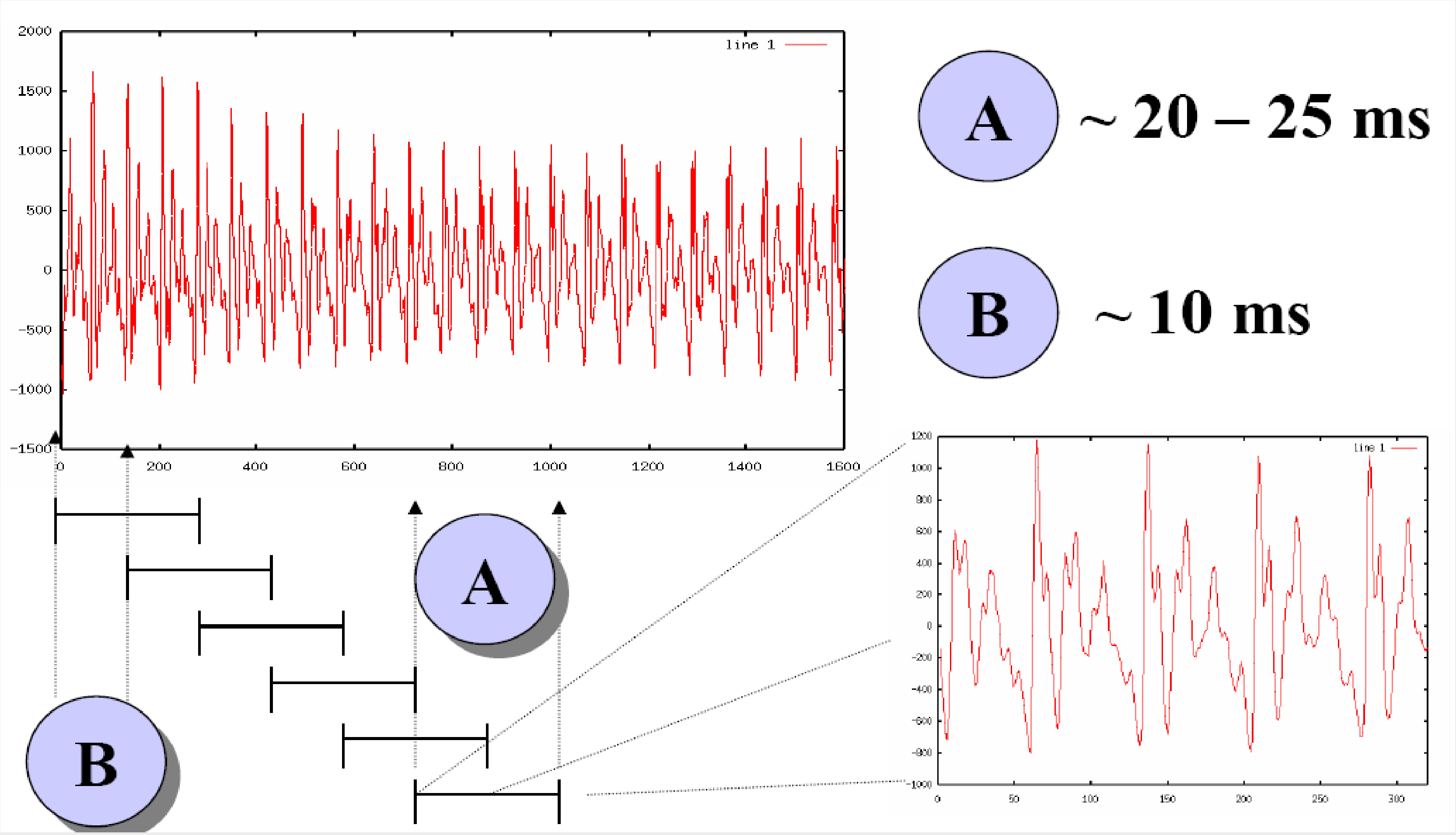
- Sound is a waveform of changing air pressure
- Speech is a sound produced by speech organs
- Microphone converts air pressure modulation to voltage modulation
- Analog-to-digital converter converts the continuous signal to digital signal, by **sampling** the value of the signal after perioding intervals
- Sampling frequency (samples per second):
  - Telephone: 8000 Hz
  - CD: 44100 Hz
  - For speech, 16000 Hz is usually sufficient



Feature extraction

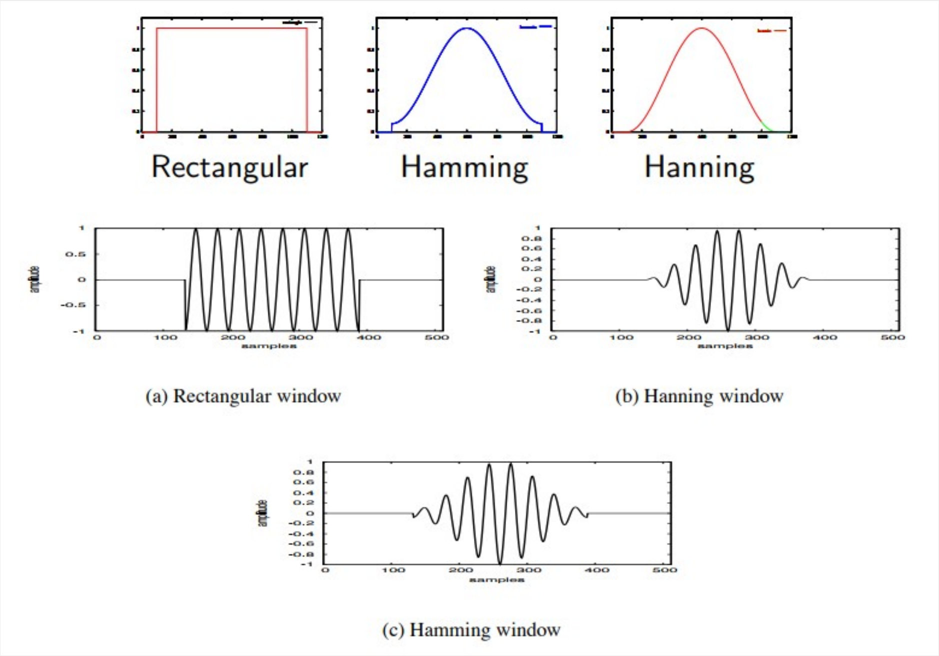
- Raw audio signal contains a lot of information (typically 16000 values every second)
- The goal of feature extraction:
  - Reduce the amount of information
  - Extract information that is important for distinguishing between speech sounds
  - Be robust agains noise, channel distortions, speaker variation

Splitting the signal into frames



Windowing

- After slicing the signal into frames, we apply a window function such as the **Hamming window** to each frame
- Why?
  - To counteract the assumption made by the FFT that the data is infinite and to reduce *spectral leakage*



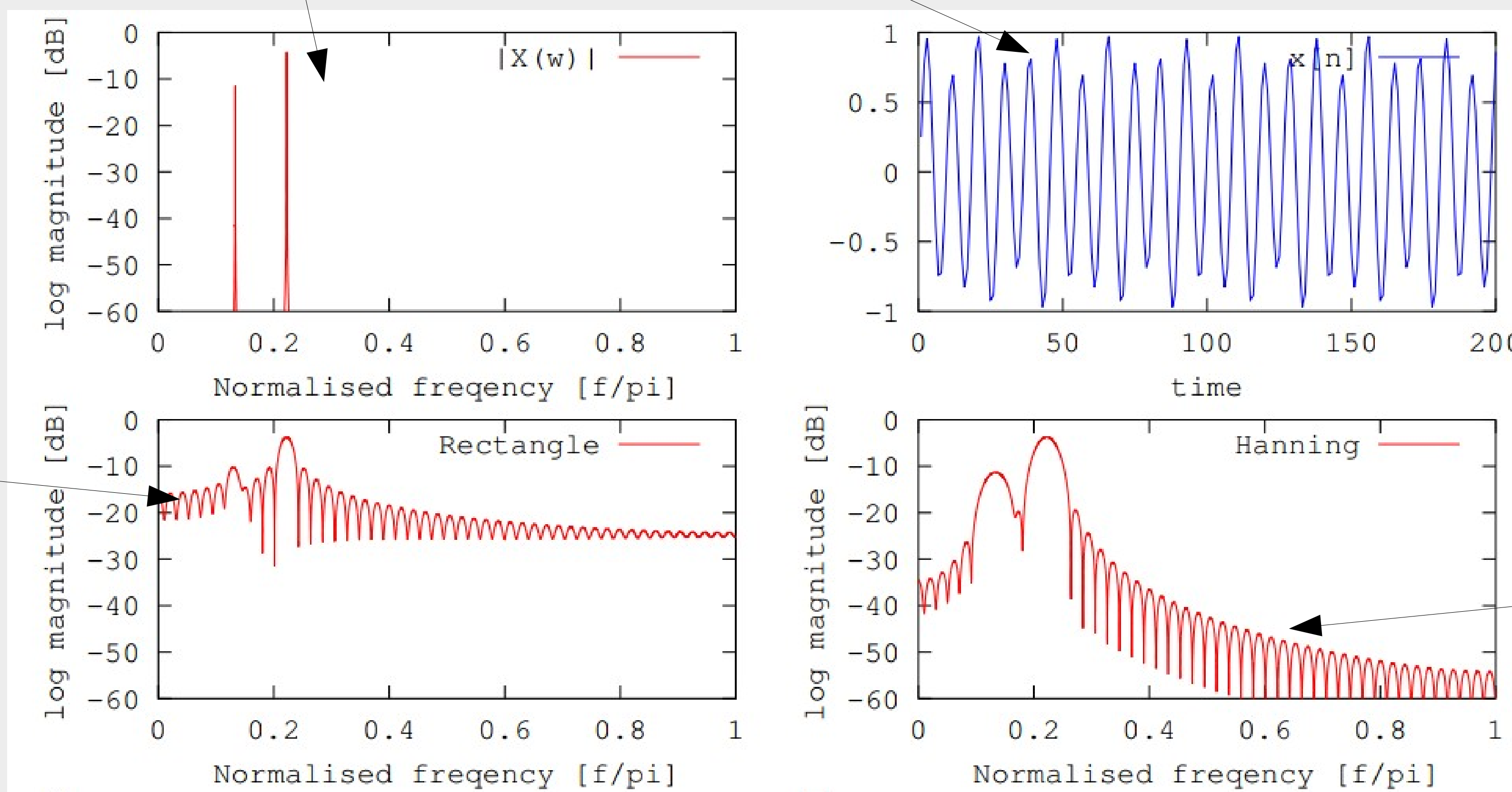
(a) Rectangular window
(b) Hanning window
(c) Hamming window

Effect of windowing

Source signal is generated using two frequencies

Bad

Better

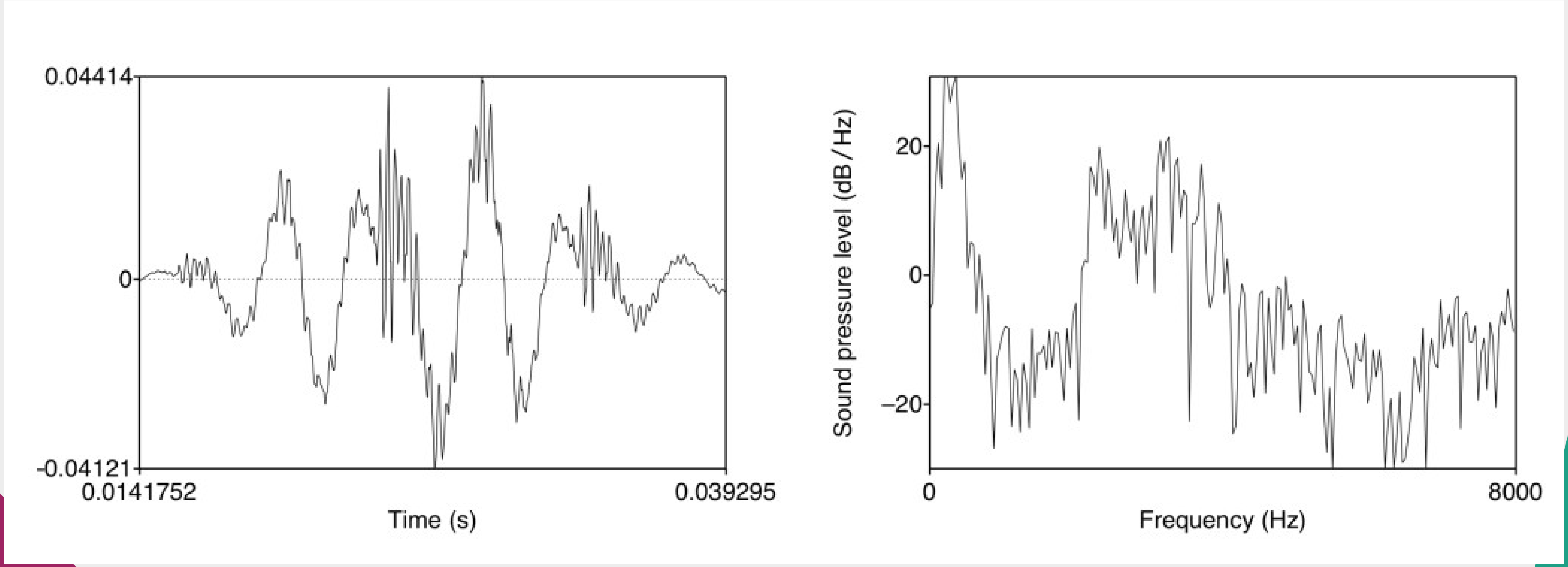


Discrete Fourier Transform

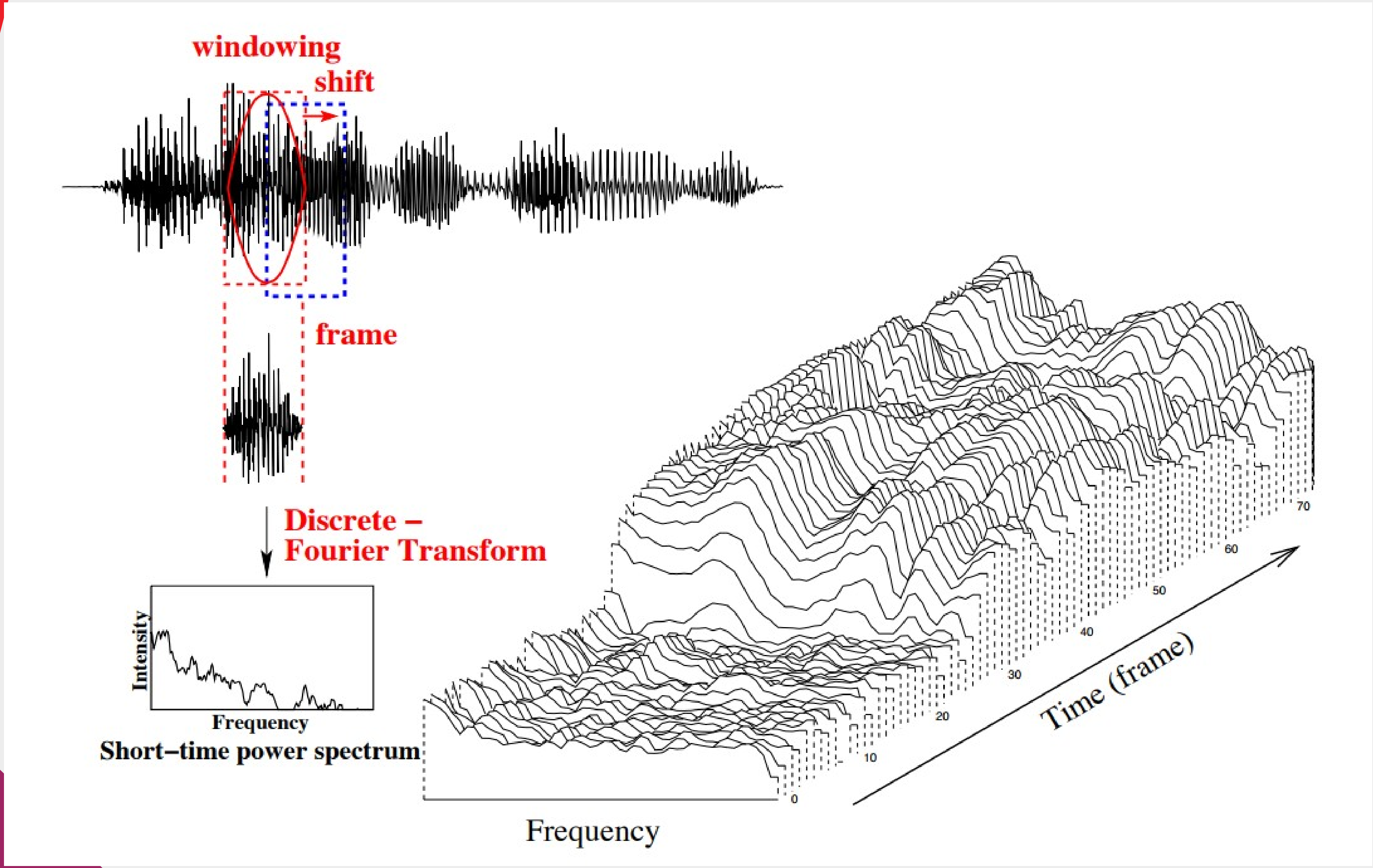
- Purpose: extracts spectral information from a windowed signal (i.e. how much energy at each frequency band)
- This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds.
- Input: windowed signal $x[0], \ldots, x[L-1]$ (time domain)
- Output: a complex number $X[k]$ for each of N frequency bands representing magnitude and phase for the k^{th} frequency component (frequency domain)

DFT Spectrum

- 25ms Hamming window of vowel /iy/ and its spectrum computed by DFT

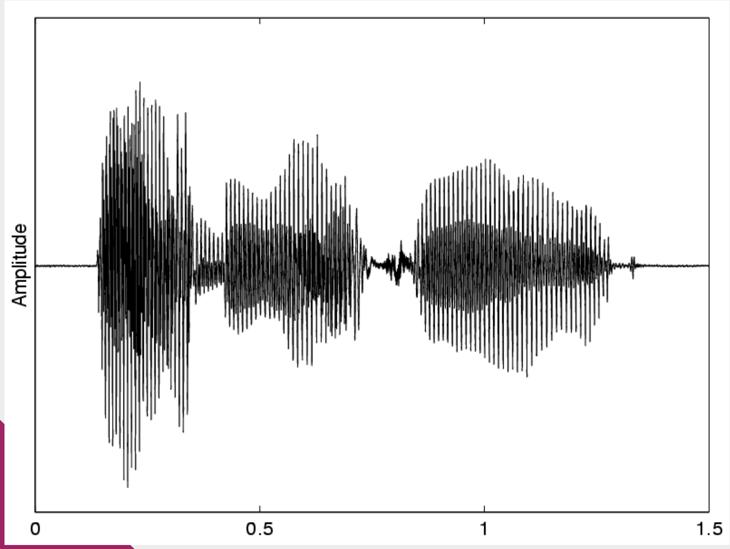


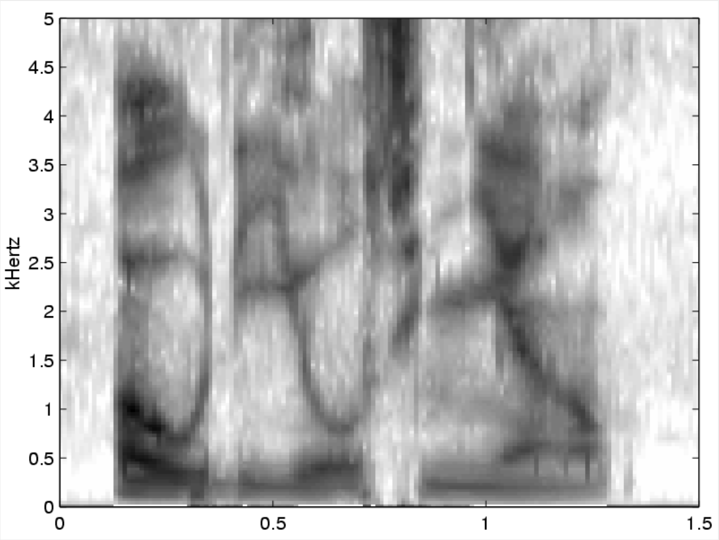
Short-time spectral analysis



Spectrogram

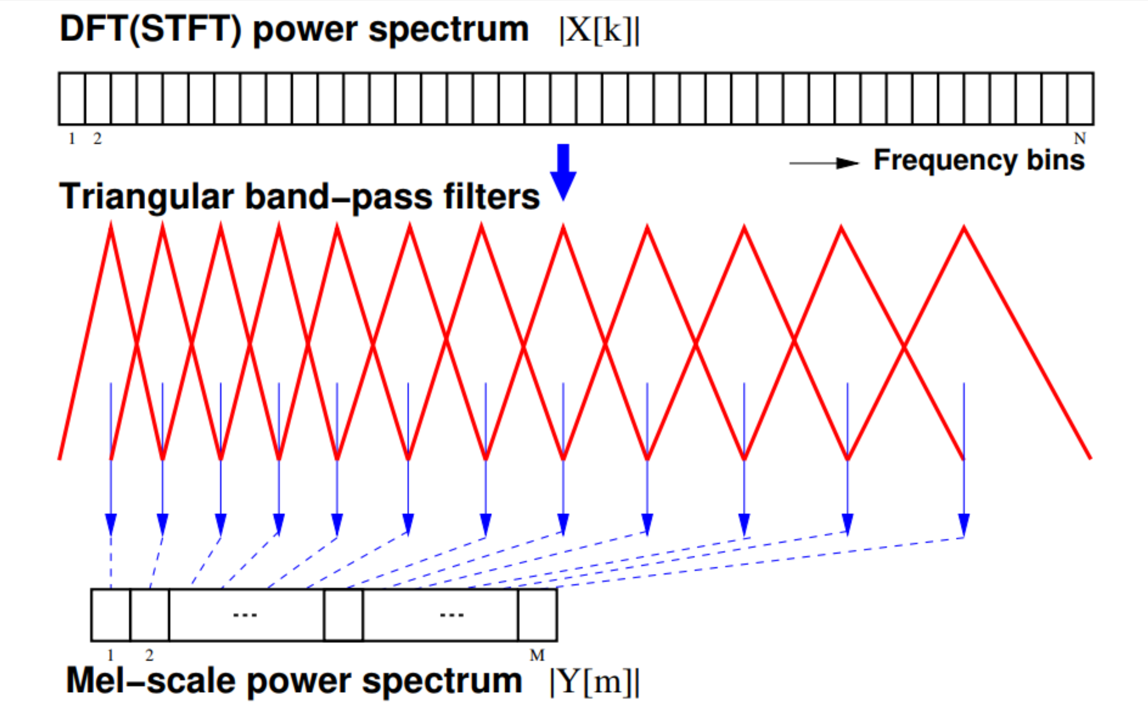
- If we turn the spectrogram 90 degrees up,
- represent the values using grayscale,
- and concatenate the spectrums of all frames
- Then we will get a **spectrogram**
- There are people (phoneticians) who can „read” the spectrogram, i.e., roughly understand what was said





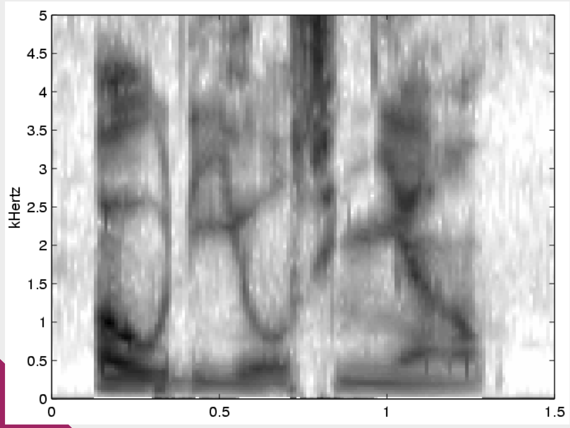
Filterbanks

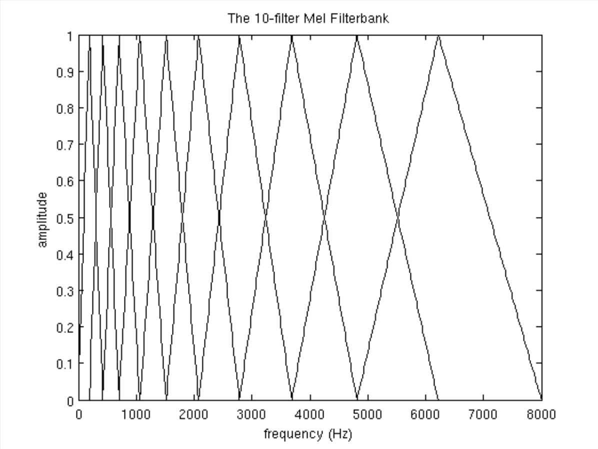
- Human hearing is less sensitive to higher frequencies — thus human perception of frequency is nonlinear
- Also, spectrogram contains still too much information
- That’s why, mel-scale filterbanks are used to discretize the spectorgram
- Mel-scale: use narrower bands in lower frequencies and wider bands in upper frequencies
- Each filter collects energy from a number of frequency bands in the DFT

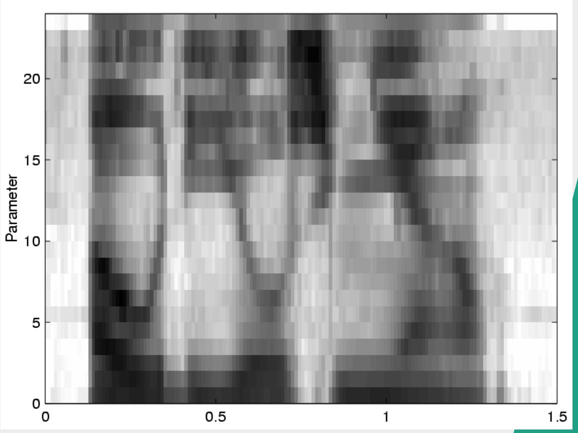


Log Mel Power Spectrum

- Compute the log magnitude squared of each mel-filter bank output: $log\ |Y\ [m]|^2$
  - Taking the log compresses the dynamic range
  - Human sensitivity to signal energy is logarithmic — i.e. humans are less sensitive to small changes in energy at high energy than small changes at low energy
  - Log makes features less variable to acoustic coupling variations







Feature normalization

- **Basic idea**: transform the features to reduce mismatch between training and test
- Cepstral Mean Normalisation (CMN): subtract the average feature value from each feature, so each feature has a mean value of 0. Makes features robust to some linear filtering of the signal (channel variation)

Resulting features

- Filterbank features
  - Short-time DFT analysis
  - Mel-filter bank
  - Log magnitude squared
  - Widely used for all kind of speech processing tasks
- Result: a speech signal (e.g., for an utterance) is transformed into a sequence of fixed dimensional feature vectors
  - The length of the sequence varies (based on the length of the input utterance)
- This representation is similar to document representation using word embeddings!
- Many familiar models (convolutional neural nets, LSTMs) can be also applied for speech!
- For old-school systems, the filterbank features are further processed using inverse discrete cosine transform that decorrelates the filterbank features
  - This results in Mel-frequency cepstral Coefficients (MFCCs)
  - Its not necessary for neural networks

Speech recognition: metrics

- The most common quality metrics for speech recognition is word error rate (WER)
- Distinguishes between 3 types of errors:
  - Insertion (I)
  - Deletion (D)
  - Substitution (S)

  To find the errors, reference and hypthesis texts are aligned, using a technique called dymanic programming
- ***WER=(S+D+I)/N*** , where ***N*** is the number of words in the reference

Word error rate, example

- Example (** is just a placeholder):

```
Scores: (#C #S #D #I) 11 1 1 0
REF:  aga ma olen aru saanud ET sel nädalal lugedes
siin SISEMINISTRI intervjuusid ja
HYP:  aga ma olen aru saanud ** sel nädalal lugedes
siin SISEMIST     intervjuusid ja
```

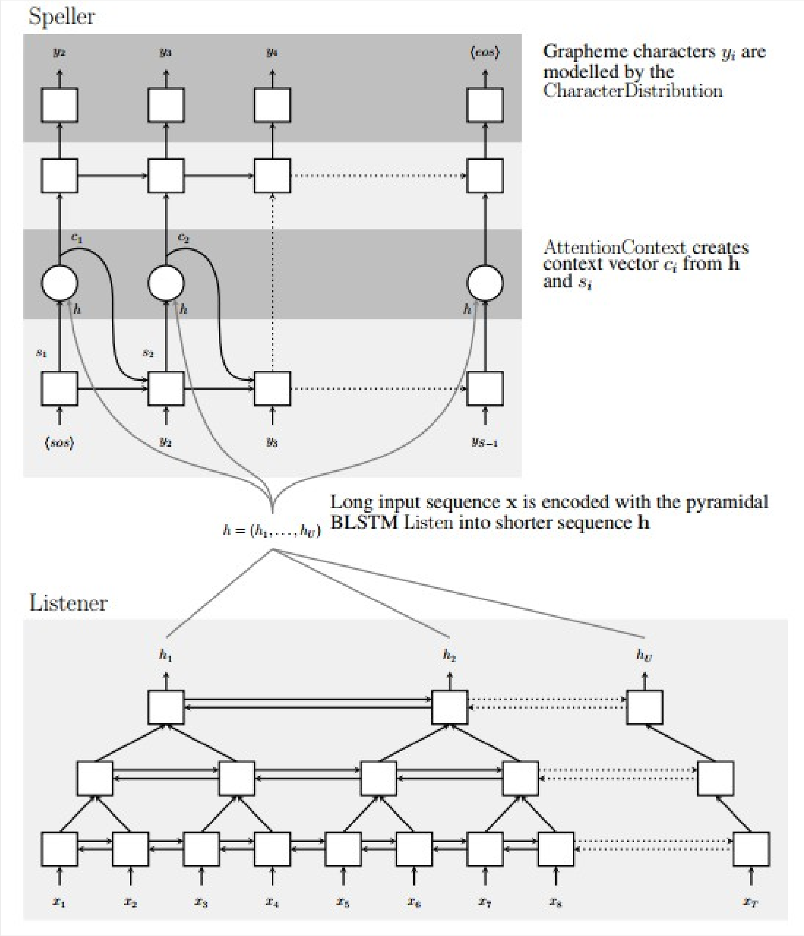
- **S=1, D=1, I=0, N=13**
- **WER=(1+1+0)/13=15.3%**

End-to-end speech recognition

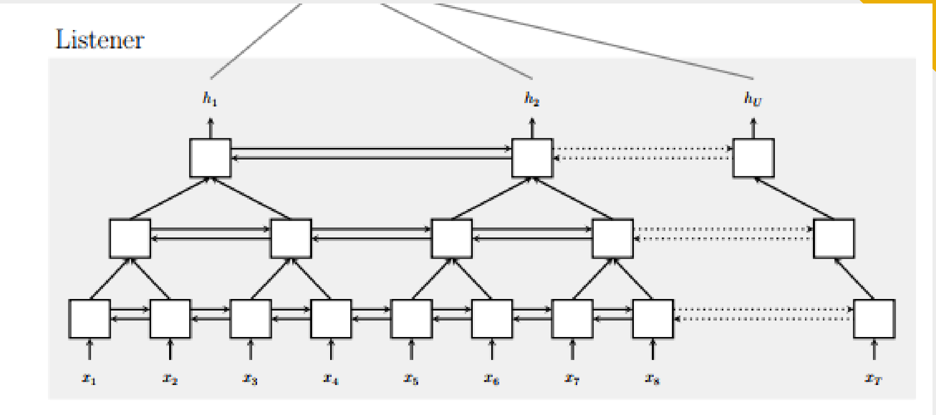
- There are two dominant models for end-to-end speech recognition
  - Listen-attend-and-spell (LAS) – very similar to neural speech synthesis, just opposite
  - Connectionist temporal classification (CTC)

Listen, Attend and Spell (LAS)

- LAS transcribes speech utterances directly into characters
- Consists of two submodules: the listener and the speller
  - The listener is an acoustic model encoder
  - The speller is an attention-based character decoder



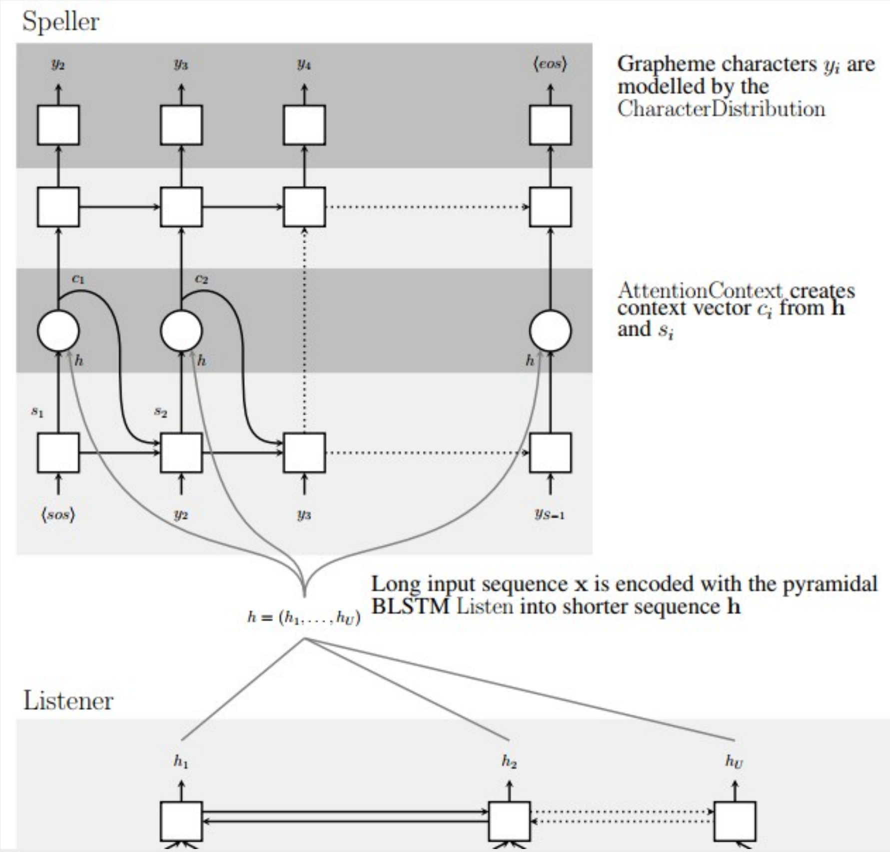
LAS: Listener



- Input to the Listener is a sequence of filterbank features
- The Listen model uses many layers of bidirectional LSTMs with a pyramidial structure
  - So that the model converges faster
- In each layer, two consecutive outputs of the lower layer are concatenated and processed by the biBLSTM
- The pyramidial structure also reduces computational cost

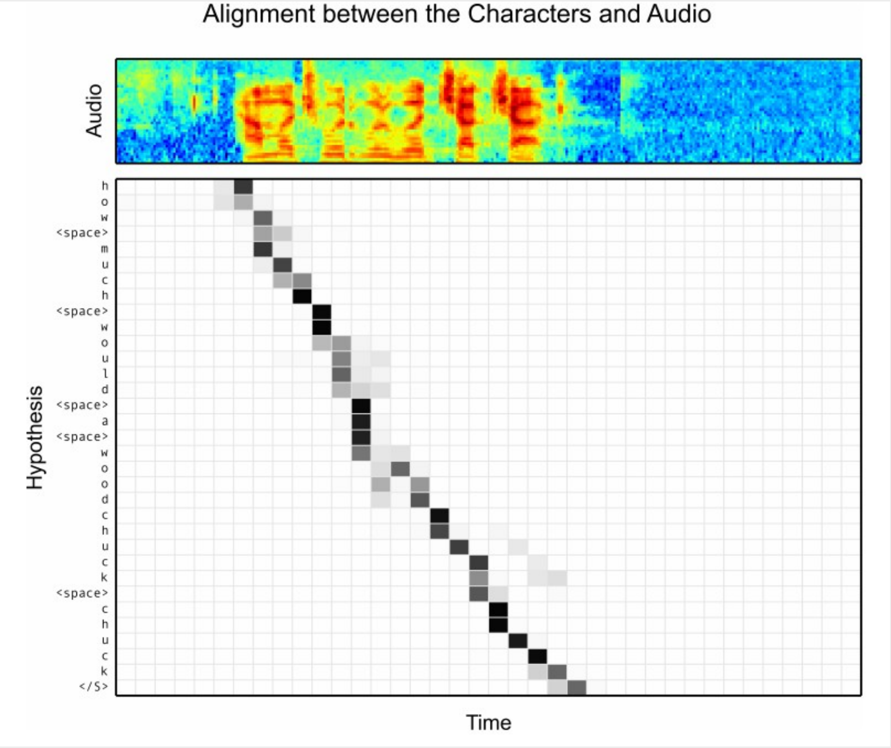
LAS: Attend and Spell

- The distribution over output characters y_i is a function of the decoder state s_i and context c_i
- The decoder state s_i is a function of the previous state s_{i-1}, the previously emitted character $\mathbf{y}_{i-1}$, an the previous context c_{i-1}
- The context vector c_i is produced by and attention mechanism from listener's states $h_{1..v}$



Attention visualization

- On the right: Alignments between character outputs and audio signal produced by the 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



Sampling outputs during training

- Using ground truth $\boldsymbol{y}_{i-1}$ as the previous output (i.e., method called „teacher forcing”) during training creates a mismatch between training and testing (i.e., applying the model for recognition)
  - During training the model is conditioned on the correct previous characters but during testing mistakes made by the model corrupt future predictions
- Improvment: train a first model using teacher forcing, then train a second model, where
  - Usually (90% of the time), ground truth $\boldsymbol{y}_{i-1}$ is used
  - But sometimes, use $\boldsymbol{y}_{i-1}$ generated by the first model

LAS: language model rescoring

- LAS model is trained using a limited amount of transcribed speech data
- But in addition, we have vast amounts of **text-only data** (newspapers, books, websites, Wikipedia)
- **Language model rescoring**: during decoding (i.e., generation from the Speller), prefer character sequences that have a high language model probability:

$$s(y|x) = \log P_{LAS}(y|x) + \lambda \log P_{LM}(y)$$

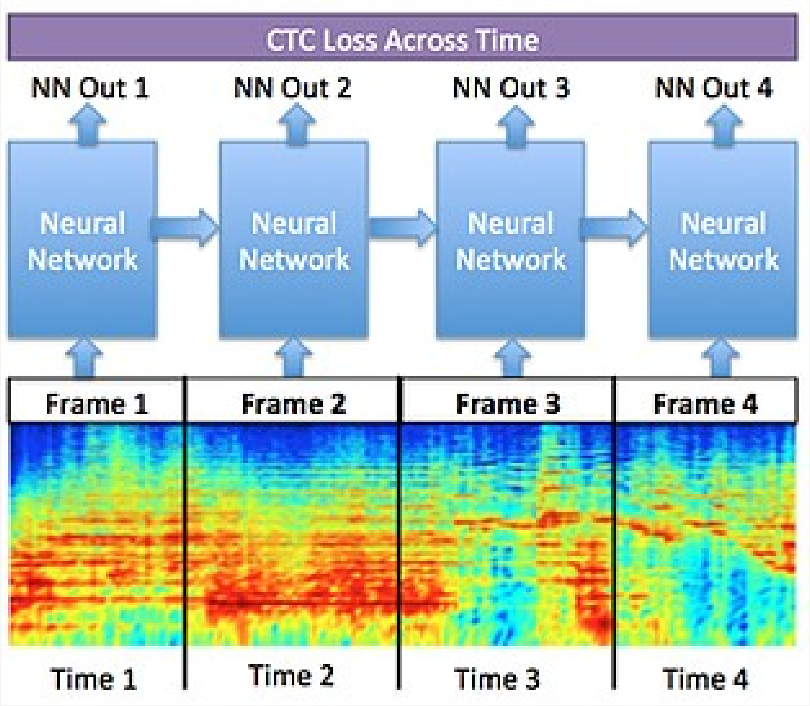
LAS: results

- LAS performs worse than a conventional system (using HMMs and DNNs)
- Sampling and LM rescoring improve results a lot

Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

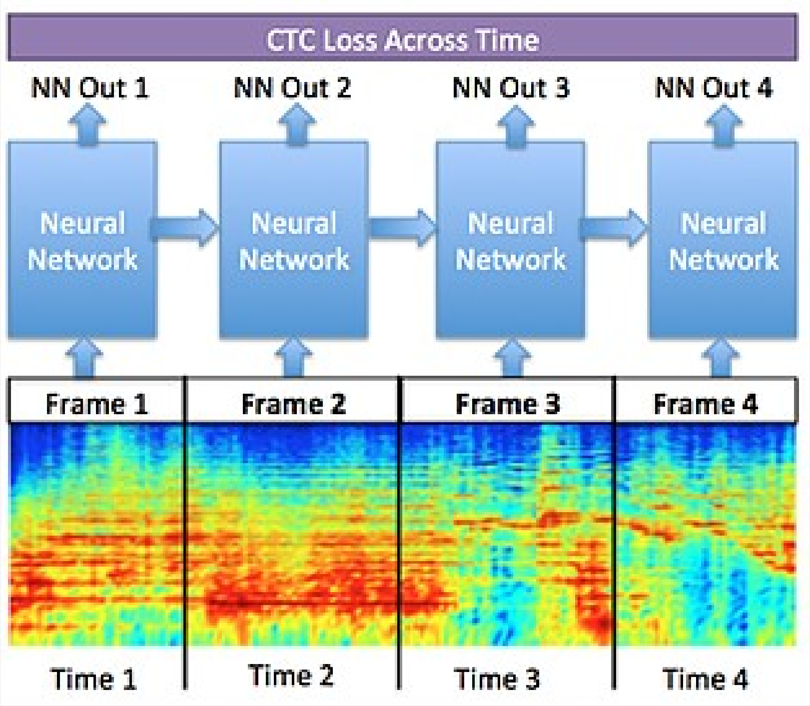
Connectionist Temporal Classification

- CTC allows to use a monolithic recurrent model, it doesn’t require seperate encoder and decoder



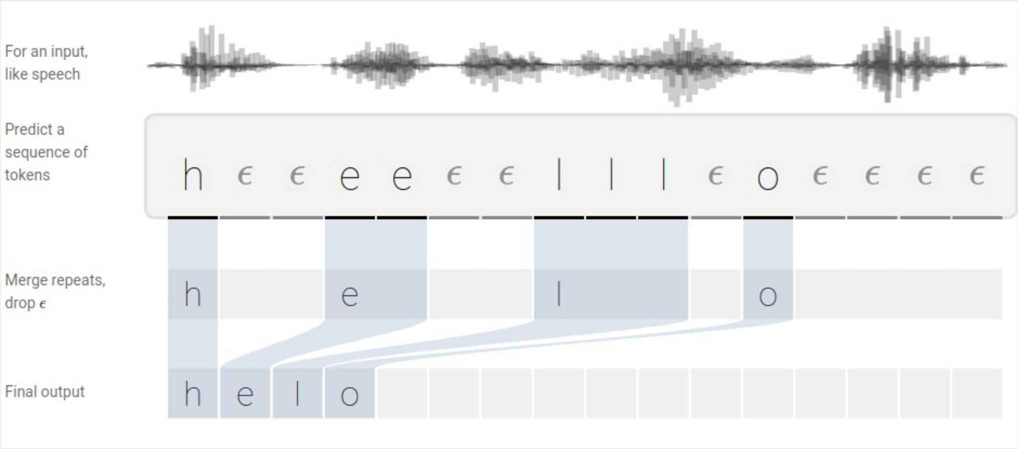
CTC: the model

- CTC is a recurrent model that for each input (filterbank) feature, outputs a character
- One extra character (blank, or ε (epsilon) is used
- The model itself typically combines several layers of convolutions and LSTMs



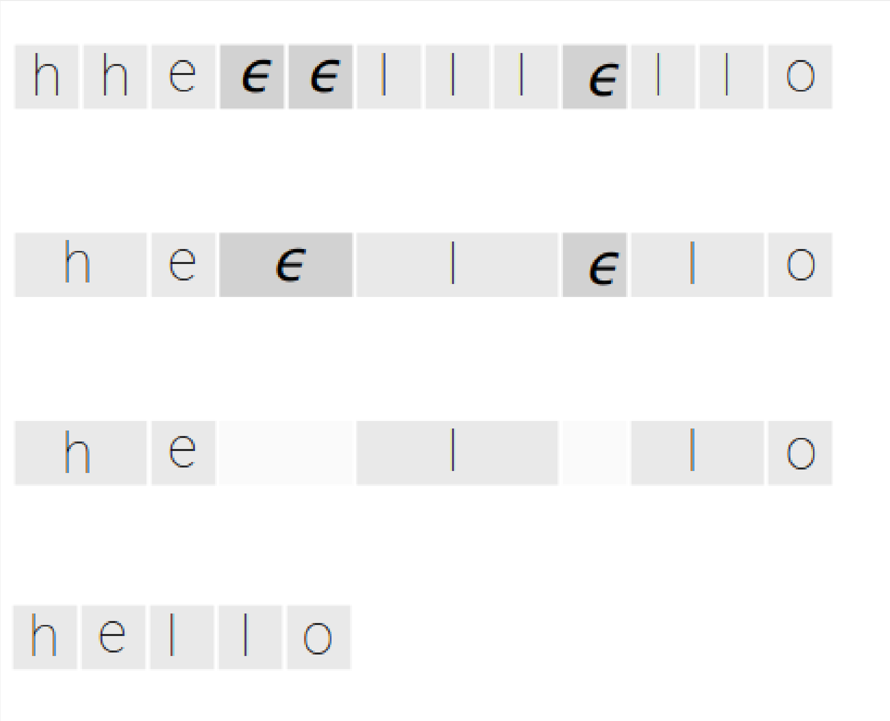
From CTC outputs to words

- CTC model outputs one character for each frame
- During post-processing, repeated characters are merged and epsilons are deleted



Postprocessing CTC outputs, again

1) Merge repeating characters
2) Remove epsilon tokens
3) Output the result



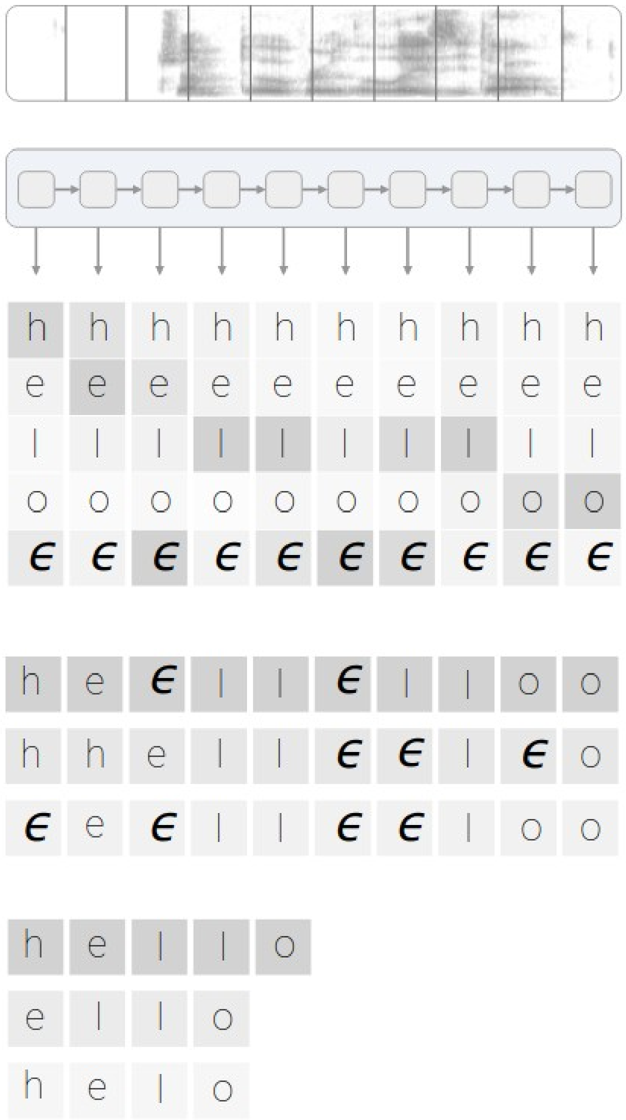
CTC loss function

- But how to train a CTC model?
  - The **exact ground-truth label** for each frame in the training data is not known
  - We only know the ground truth character sequence for the utterances
- Solution: **CTC loss function**
  - Marginalizes (sums) over all set of alignments **that produce the same character sequence**

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability**	**marginalizes** over the set of valid alignments	computing the **probability** for a single alignment step-by-step.

CTC loss, summarized



We start with an input sequence, like a spectrogram of audio.

The input is fed into an RNN, for example.

The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

With the per time-step output distribution, we compute the probability of different sequences

By marginalizing over alignments, we get a distribution over outputs.

CTC performance

- CTC has similar performance as LAS: not quite good as traditional HMM-DNN based models
- Both LAS and CTC require a lot of training data
  - The more data, the closer the performance to DNN-HMM
  - Thousands of hours is good
  - Estonian has only about 250h