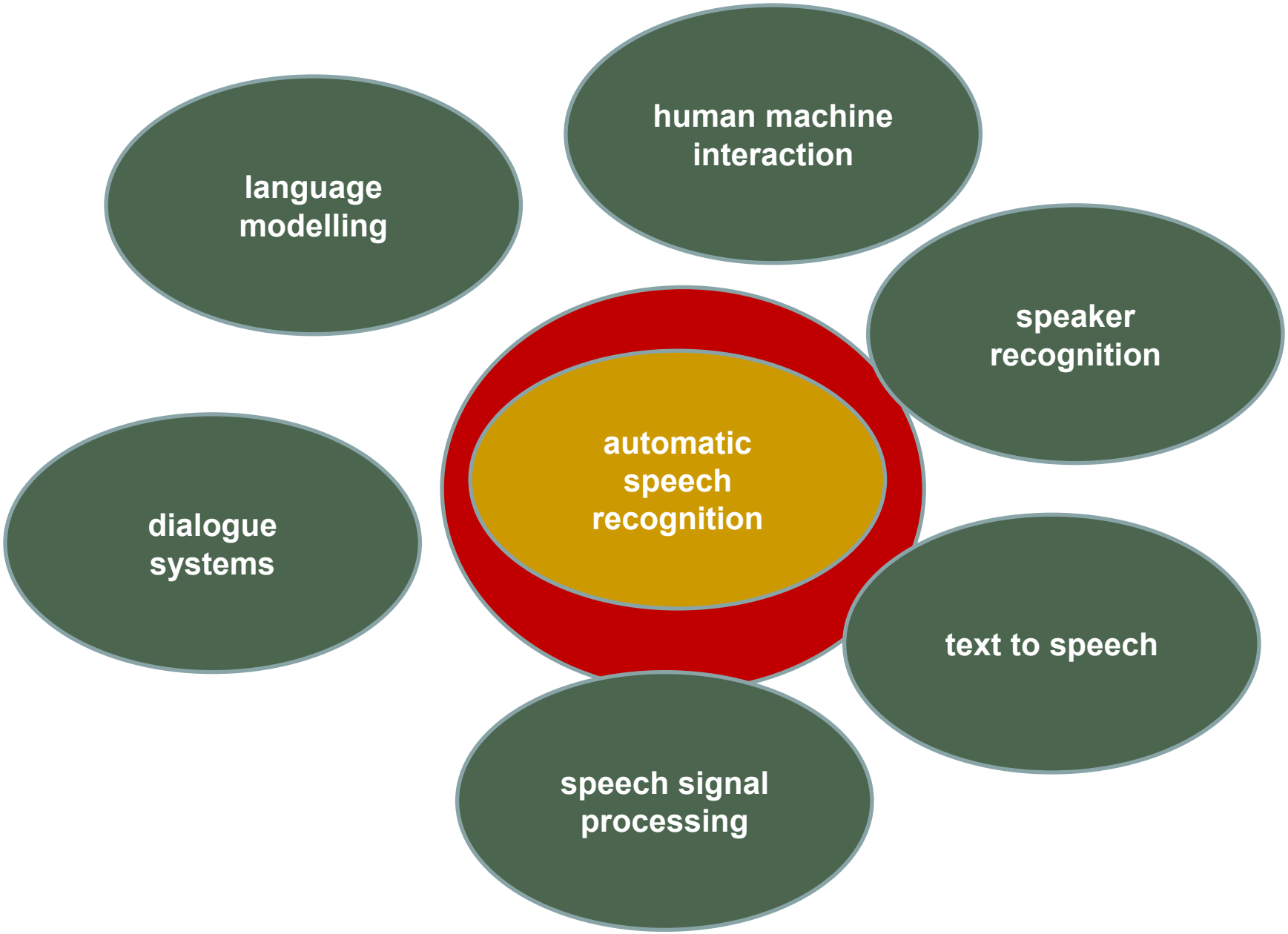
Architectures ASR
2025-6

Radboud University, Nijmegen

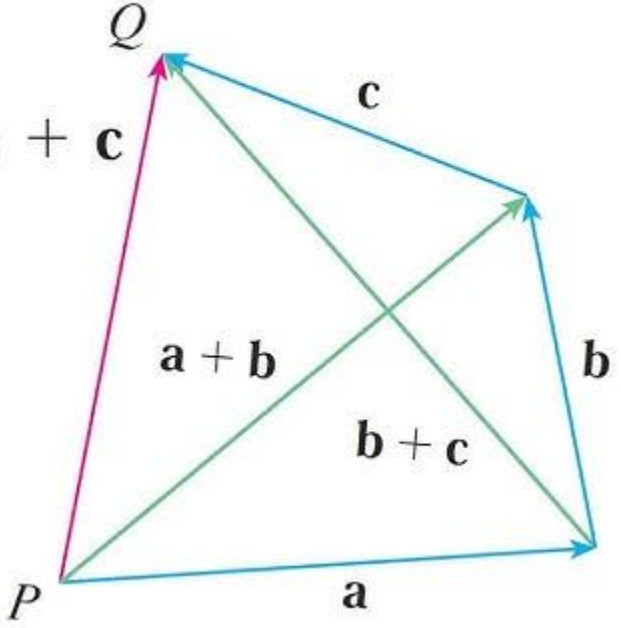
Louis ten Bosch


language modelling
human machine interaction
speaker recognition
automatic speech recognition
dialogue systems
text to speech
speech signal processing

Big picture: to vectors

Properties of Vectors

1. $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$
2. $\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c}$
3. $\mathbf{a} + \mathbf{0} = \mathbf{a}$
4. $\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$
5. $c(\mathbf{a} + \mathbf{b}) = c\mathbf{a} + c\mathbf{b}$
6. $(c + d)\mathbf{a} = c\mathbf{a} + d\mathbf{a}$
7. $(cd)\mathbf{a} = c(d\mathbf{a})$
8. $1\mathbf{a} = \mathbf{a}$



Big picture

Vectorization is a very important step in current approaches (in chatGPT, in end-to-end ASR, in reasoning models, in NLP, in chemistry, ...)

1970: audio → feature vectors (MFCC)
2013: words → vec: word2vec (context independent)
2015 and later: word dependent word embeddings (bank ≠ bank)
2017: attention mechanism → (very) long contexts
2020: wav2vec2.0 (mapping audio to probability vectors on tokens in a dictionary)
2023: whisper (encoder + decoder architecture)

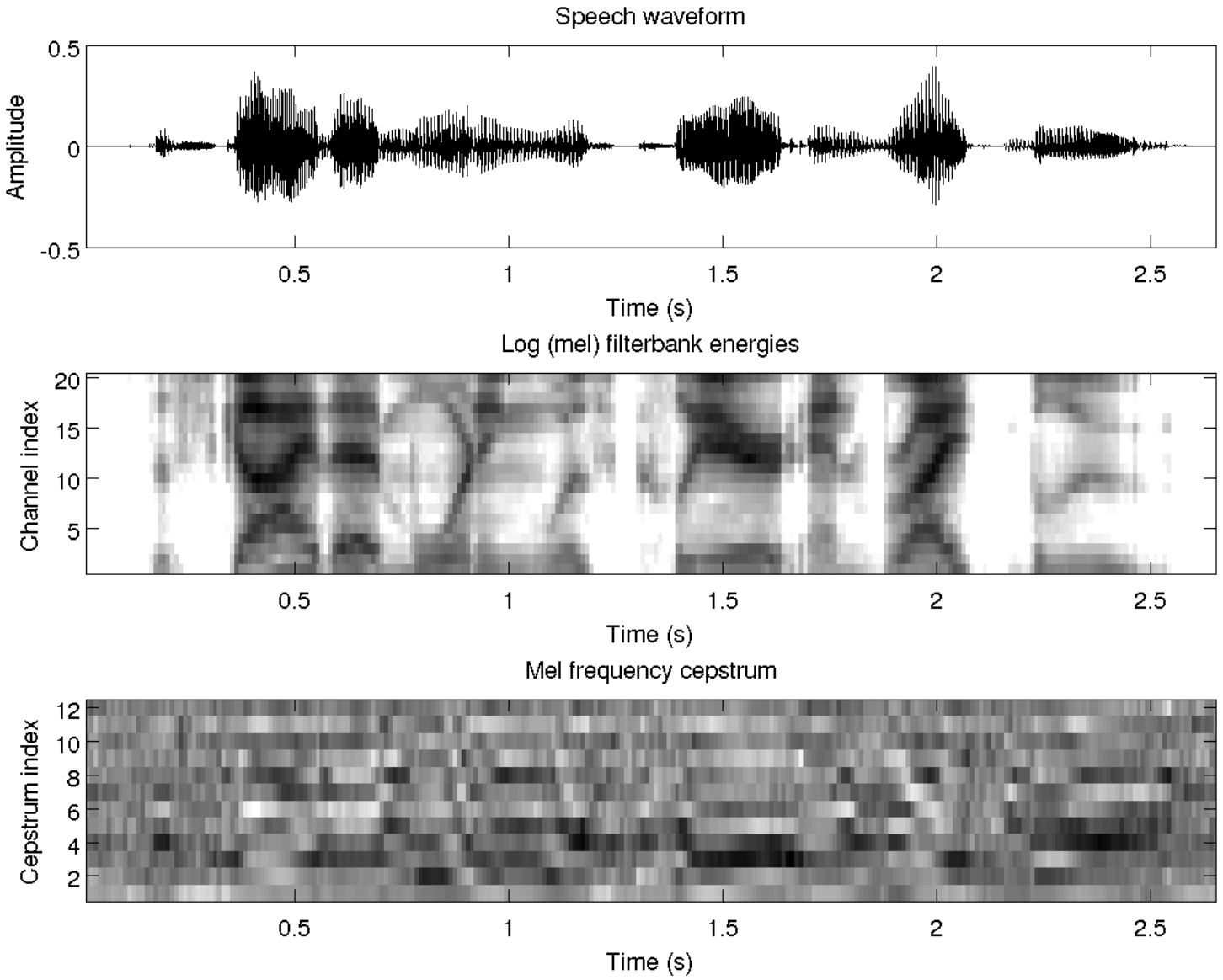
Vectorization → ideal for deep learning and neural networks

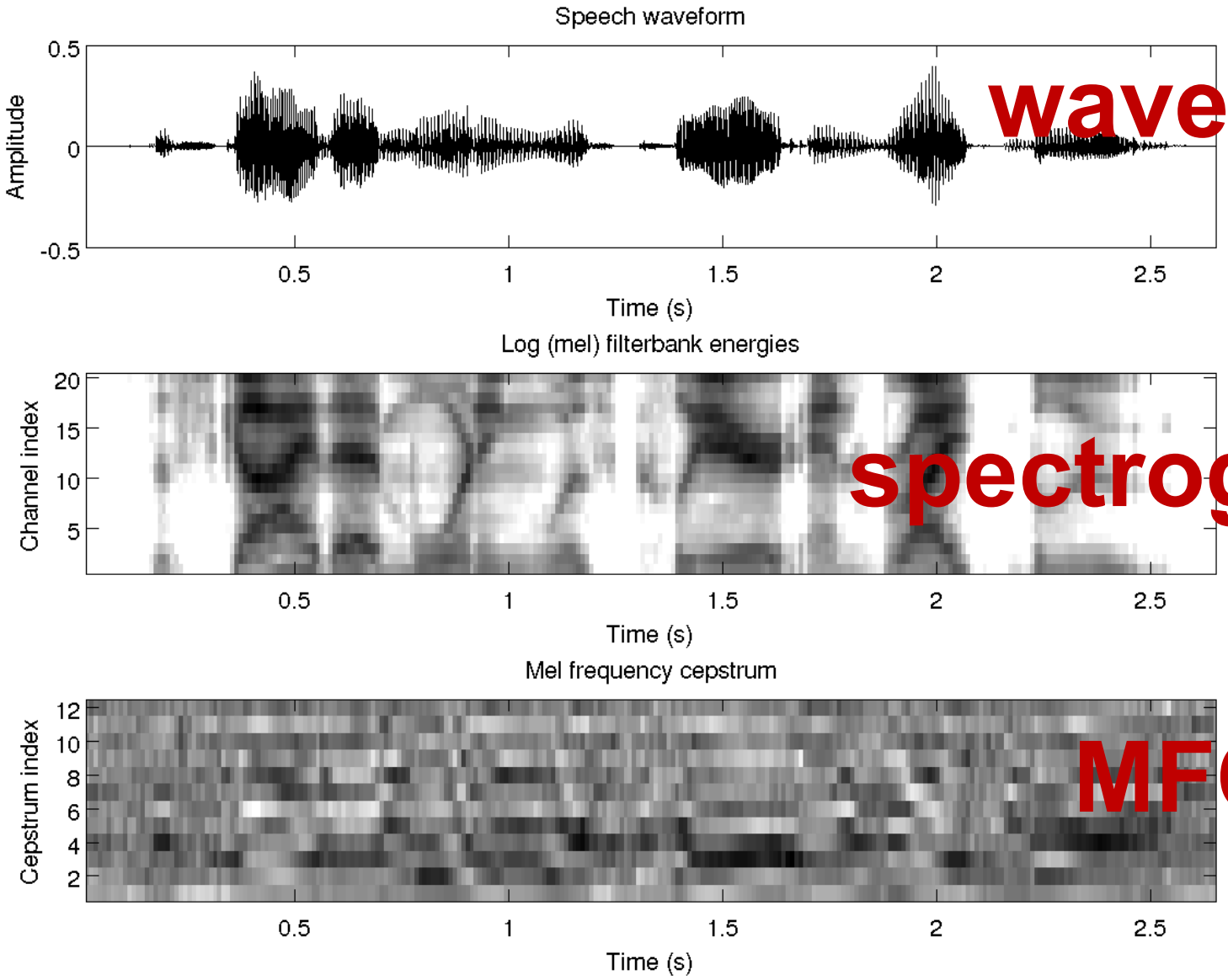
From audio to features

features are input for all downstream modules

+

what's not in the features cannot be classified later


Speech waveform
Amplitude
0.5
0
-0.5
0.5
1
1.5
2
2.5
Time (s)
Log (mel) filterbank energies
Channel index
20
15
10
5
0.5
1
1.5
2
2.5
Time (s)
Mel frequency cepstrum
Cepstrum index
12
10
8
6
4
2
0.5
1
1.5
2
2.5
Time (s)



waveform

spectrogram

MFCC

MFCC vectors

MFCC = Mel Frequency Cepstral Coefficients

Very many websites show information about MFCCs, often with useful Python function calls

- https://www.kaggle.com/ilyamich/mfcc-implementation-and-tutorial
- https://pypi.org/project/python_speech_features/
- Librosa python library https://librosa.org/doc/latest/index.html

From audio to MFCC

- audio (analog signal) → digital signal
  - AD conversion
- digital signal → MFCC feature vectors
  - in 5 steps

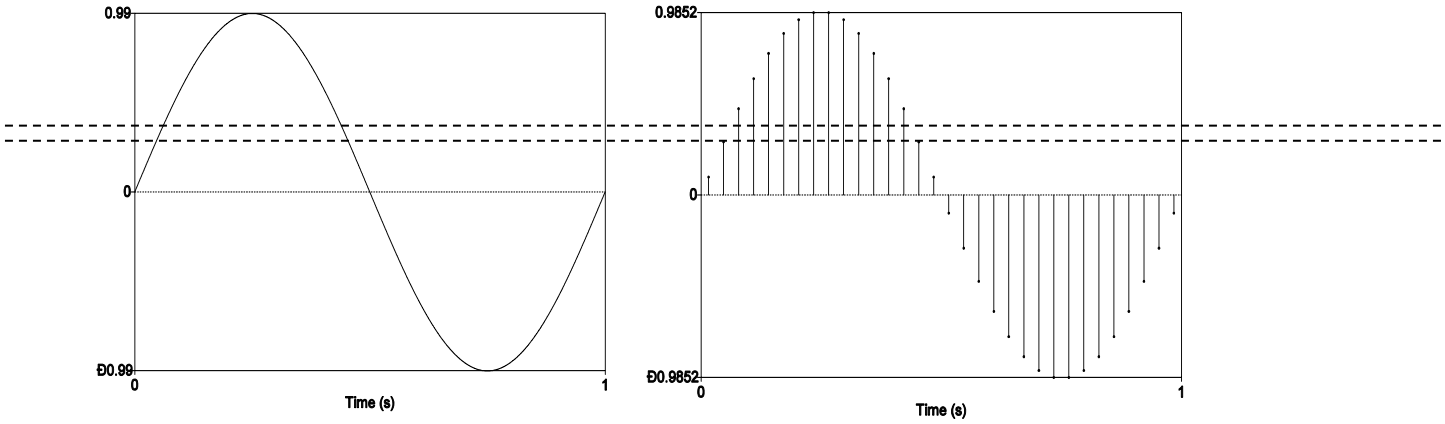
From audio to MFCC

From audio to MFCCs:

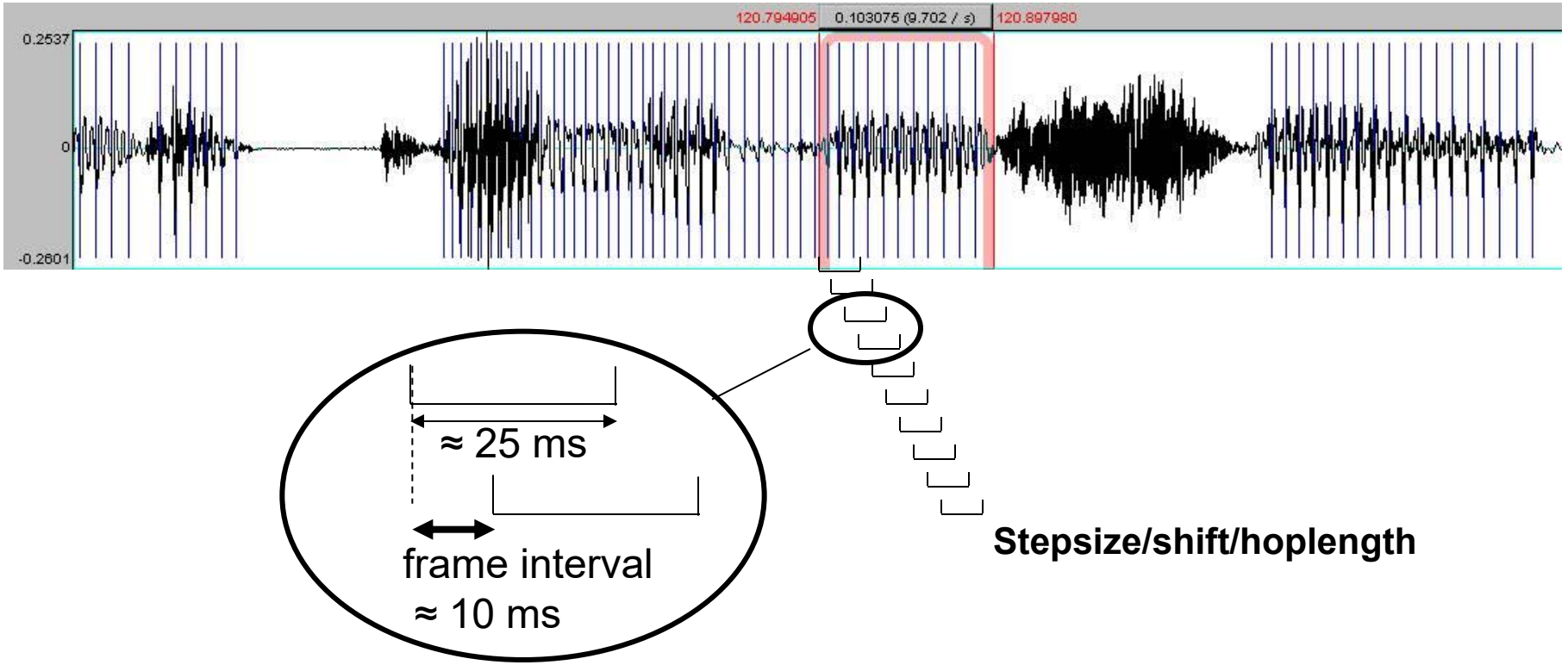
0. AD
1. Segmentation
2. Smoothing
3. *Fast Fourier Transform (FFT):* Conversion from the time domain to the frequency domain
4. Apply perceptual weighting based on human auditory processing.
5. Decorrelation

0 analog-digital (AD) conversion

- Discretisation in time
  - sampling frequency or sampling rate (samples/sec, Hz) determines the highest frequency that can be represented. **Nyquist**. (10 kHz-44 kHz)
- Discretisation in amplitude
  - Number of possible amplitude values is determined by bytes/sample, e.g.
    - 8 bits (1 byte): 2^8 (256) possible values
    - 16 bits (2 bytes): 2^{16} (65536) possible value



1 segmentation



25ms: analysis window length

10ms: frame shift/stepsize/hoplength

If sample freq = 16kHz, 25ms corresponds to 0.025*16000 = 400 samples.

1 what matters in segmentation?

what is being said ←→ shape of the vocal tract

shape of the vocal tract ←→ the energy envelope of the spectrum

What is a reasonable analysis duration ?

The average duration of a speech sound is 70ms.

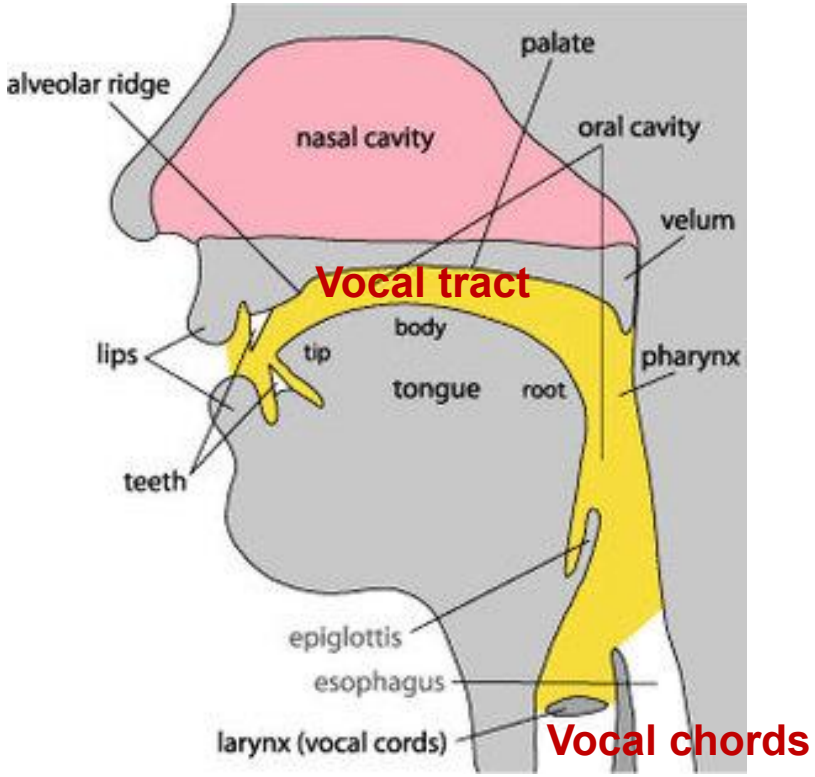
A defendable analysis frame duration is 25 ms.

What about the shift?

To accurately describe the changes in vocal tract shape over time, the number of analyses per second must be at least twice as high as the highest frequency with which the vocal tract changes (Nyquist criterion).

100 times/second (i.e. every 10 ms) is enough

1 segmentation: vocal tract



Articulation is relatively slow

About 12-14 speech sounds per second, i.e. 70 ms. per phone, on average

Articulations move synchronously/in parallel → assimilation of properties of neighboring sounds

2 smoothing/windowing

Hard boundaries give audible artefacts. These artefacts can be avoided by proper windowing: taper off the beginning and end of the signal.

For a well-chosen window, the spectrum is nearly identical to a signal of which the core part is repeated indefinitely.

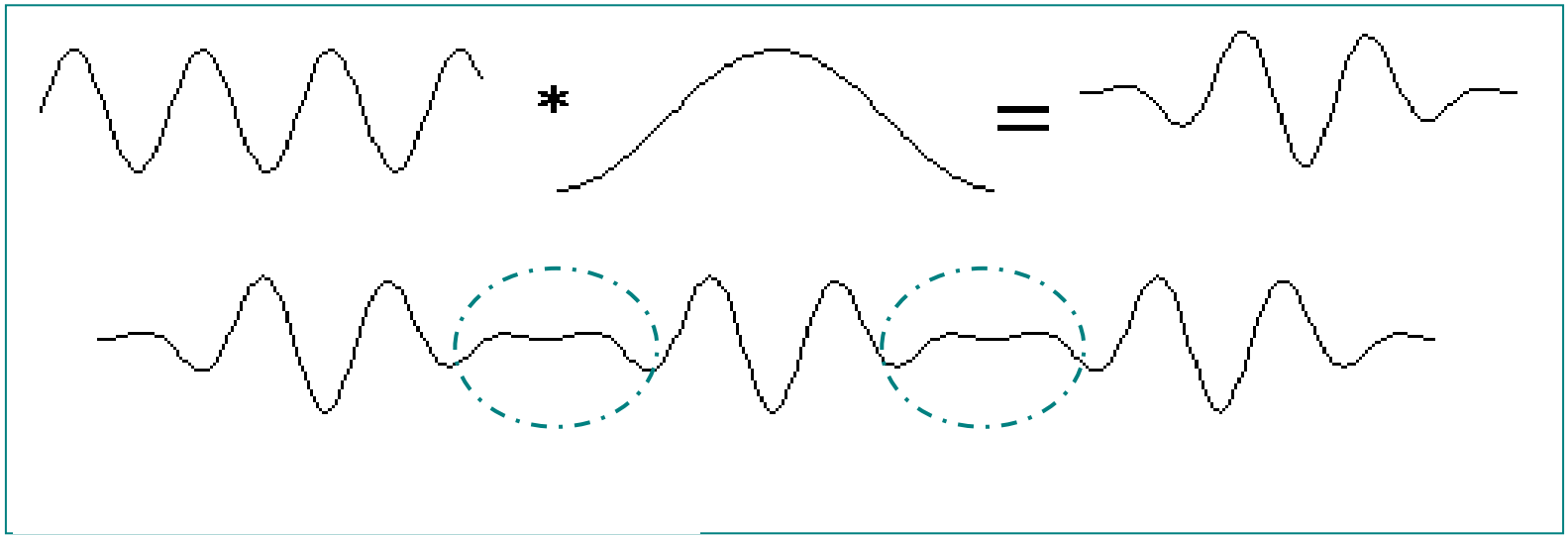
Often used windows are Hamming and Hanning windows. See e.g. https://www.youtube.com/watch?v=YsqGQzJ_2V0

2 smoothing/windowing

segmented waveform

window

windowed waveform



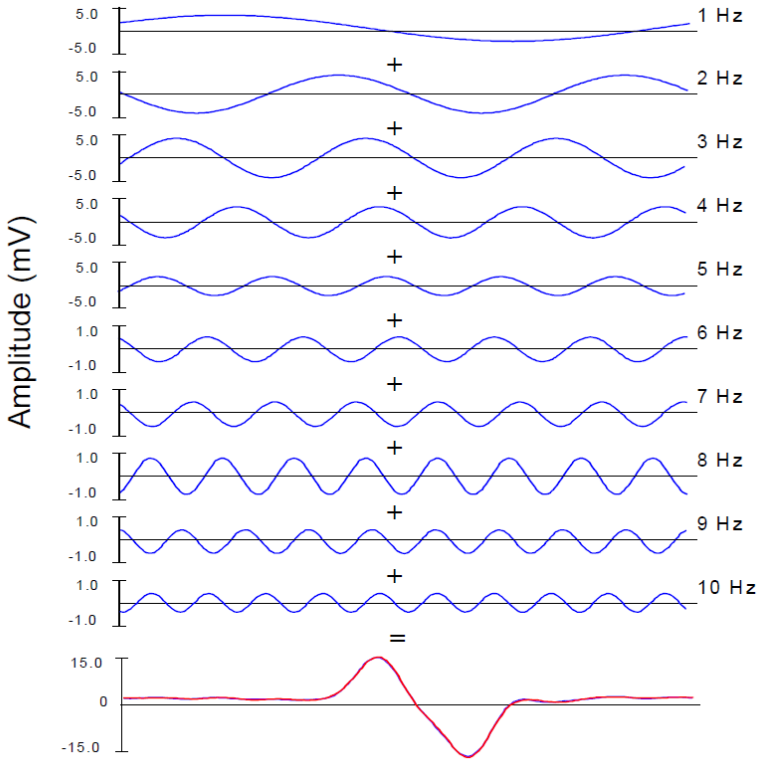
Source figures: http://www.bores.com/courses/intro/freq/3_window.htm

3 FFT

Fast Fourier transform (FFT): maps time domain to frequency domain

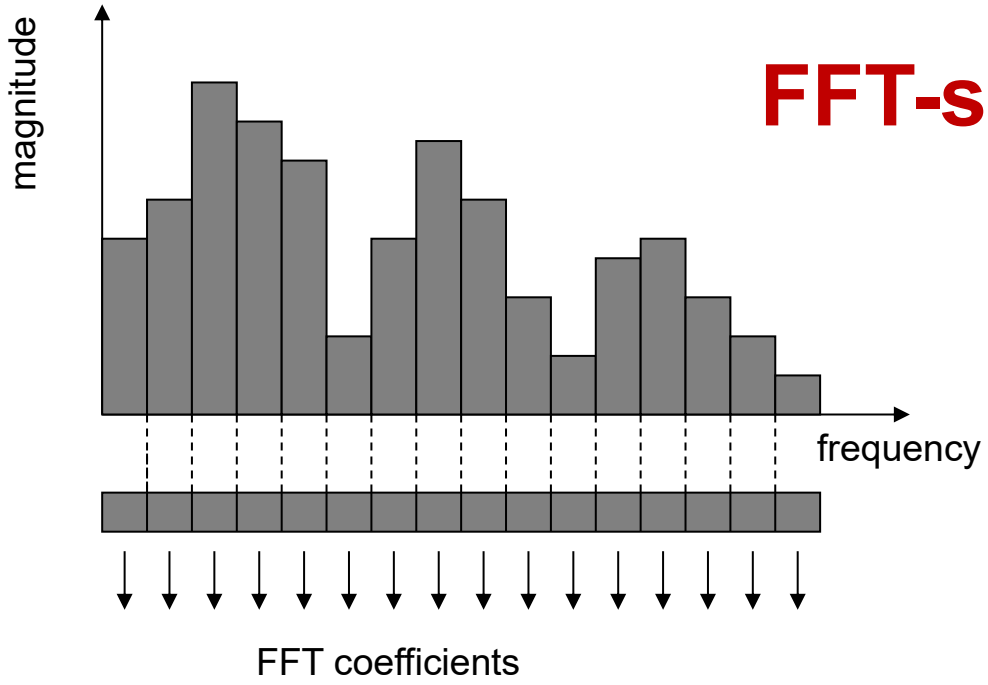
Jean-Baptiste Fourier: Every waveform is the sum of sine waves with a certain magnitude and phase

The signal in red is decomposed in terms of a weighted sum of sine waves with frequencies 1, 2, 3, 4 …



3 FFT

Output:



FFT-spectrum

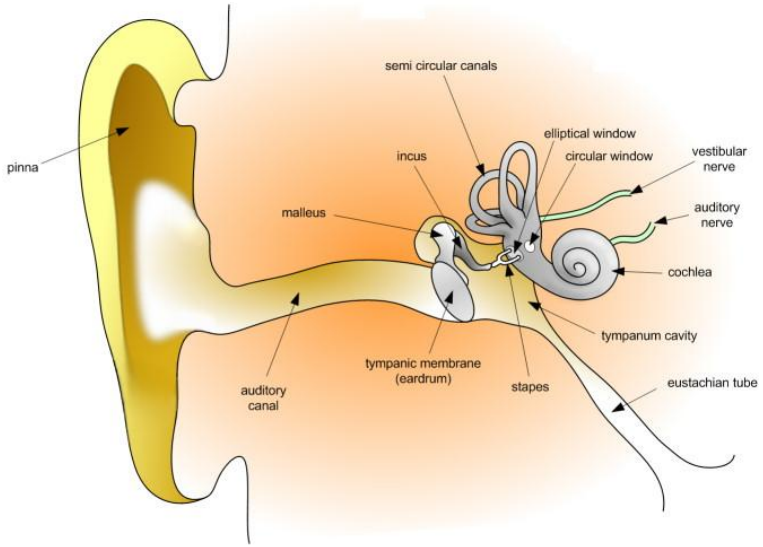
3 FFT

The resulting FFT coefficients are written in one vector

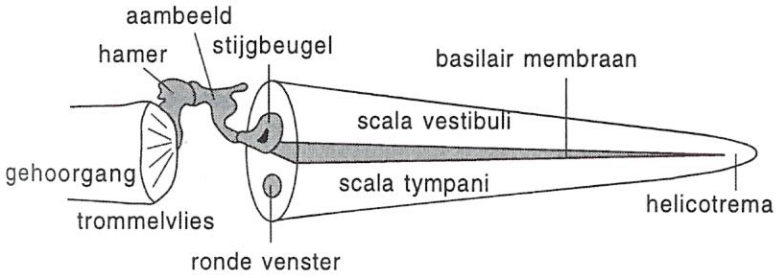
- the more coefficients, the more accurate the description
- but: higher-order coefficients may be noisy

4 perceptual weighting

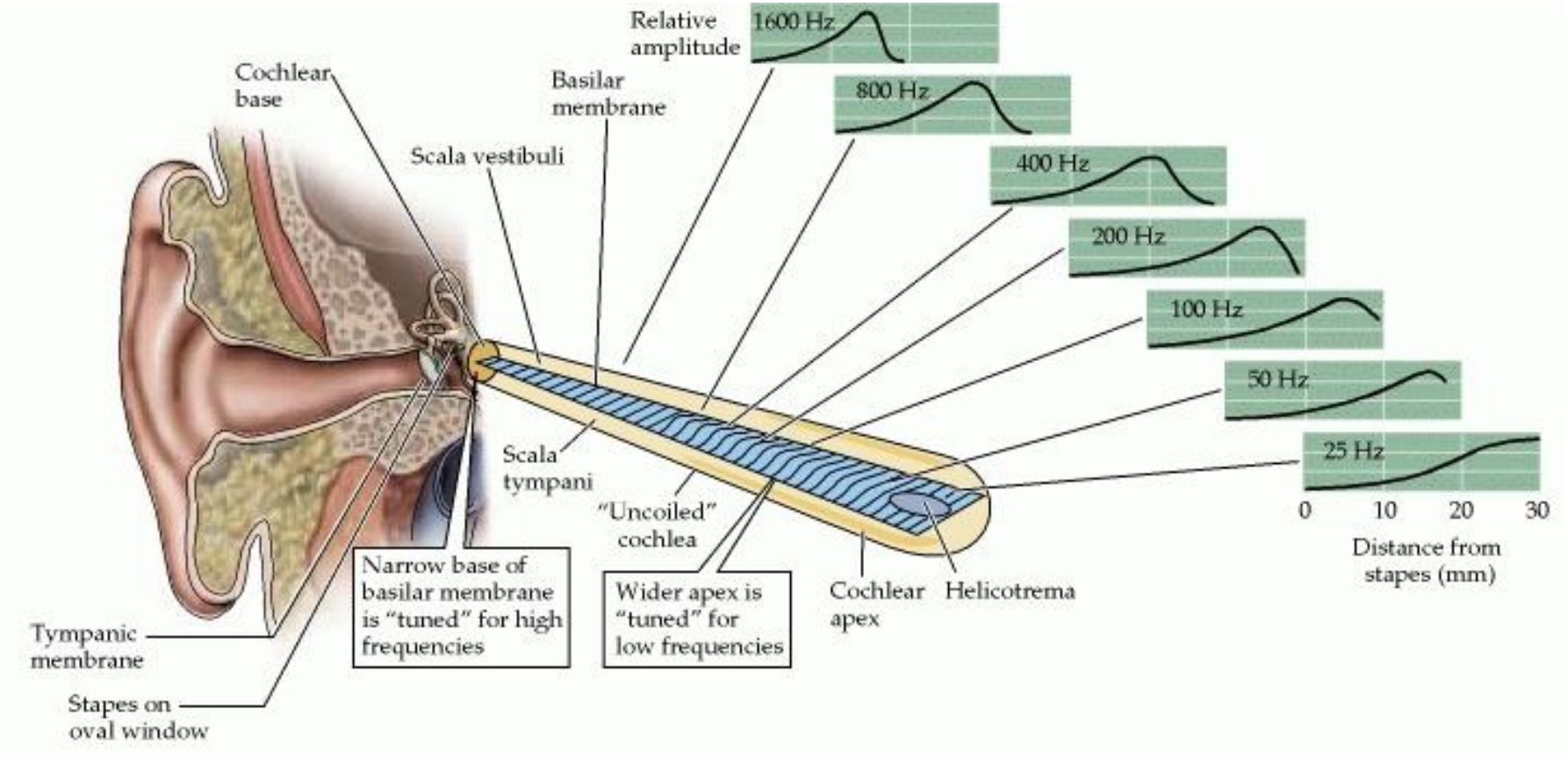
Convert FFT coefficients using perceptual properties of the human auditory system.



malleus, incus, stapes



Cochlea in normal (left, wikipedia) en unrolled form



Weber's law https://www.youtube.com/watch?v=hHG8io5qIU8

Physics	**Perception**
Energy as function of frequency	**log(E) as function of log(f)**

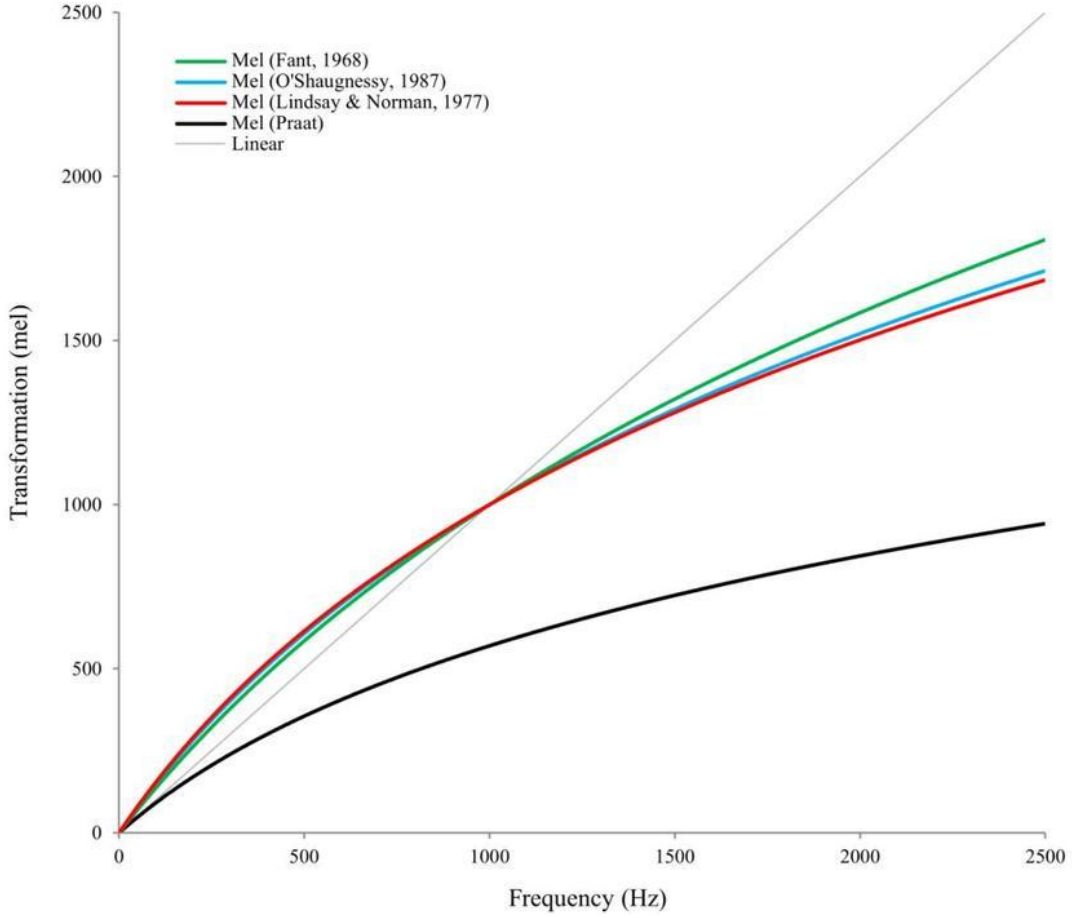
4 perceptual weighting

- The human ear is not sensitive to frequency along a linear scale
- Psycho-acoustical frequency scales often used to approximate the human non-linear sensitivity
- Examples: the *Mel scale, Bark scale*

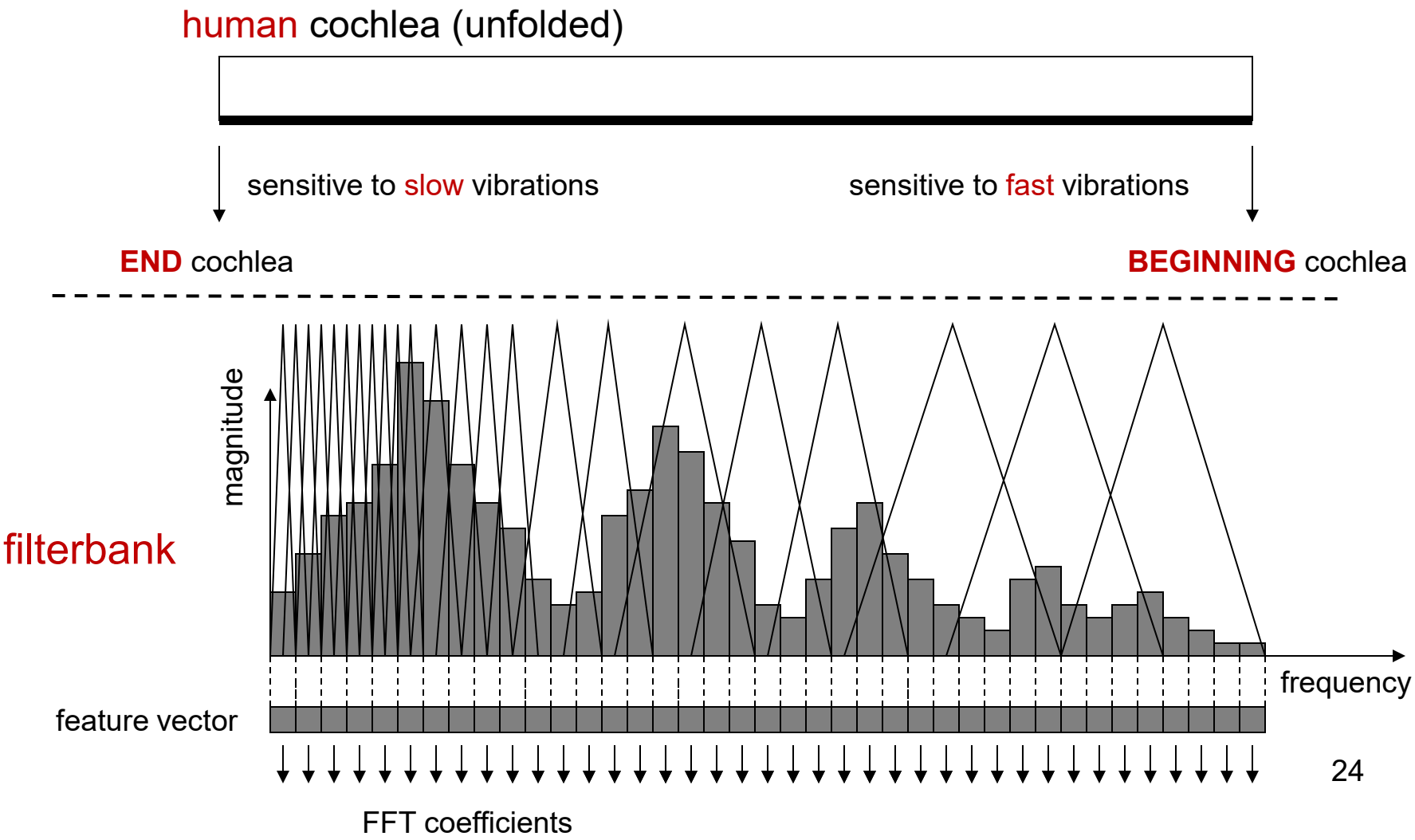
4 frequency to mel: f→ mel(f)

mel(f) =
1125 log(1+f/700)

There are several other transformations, all log-like.



4 mel-filterbank



4 log() all energy values in each filter

- E → log(E)

$$\Delta Percept = \frac{\Delta PhysicalQuantity}{PhysicalQuantity}$$

- **Weber's law**
- **https://www.youtube.com/watch?v=hHG8io5qlU8**
  - Energy → loudness
  - Fundamental frequency → pitch (piano)
  - Perception of physical phenomena
  - Estimation of physical quantities
  - Duration, length, pressure, …

5 decorrelation

The amount of energy in neighbouring filters is strongly correlated. In order to reduce this correlation, a Discrete Cosine Transform (DCT) is performed

We obtain the **Mel Frequency Cepstral Coefficients (MFCCs).**

These MFCCs are approximately statistically independent. The first 12 coefficients $c_1..c_{12}$ suffice to describe the relevant details of the spectrum (for that analysis window).

See e.g.

https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html for comments on the DCT step

Audio to MFCC: summary

Summary

		output	typical size
0.	A/D conversion	digital signal	16000/sec
1.	segmentaton	analysis stretch	400 samples/10ms
2.	smoothing	windowed signal	400 samples/10ms
3.	FFT	spectrum (A(f))	400 magnitudes/10ms
4.	filterbank	feature vector	20-40 energies/10ms
5.	decorrelation	feature vector	12 features/10ms

assuming 16kHz, 25 ms analysis frame

Alternative techniques to extract features from the speech signal

Techniques to extract features from speech signal	
Principal Component Analysis (PCA)	Linear map, fast, eigenvector-based, Traditional, eigenvector base method, OK for Gaussian data
Linear Discriminate Analysis (LDA)	Supervised linear map; fast, eigenvector-based Better than PCA for classification
Independent Component Analysis (ICA)	Linear map, iterative non-Gaussian, blind course separation, used for de-mixing non-Gaussian distributed sources
Linear Predictive Coding	Aiming at dim reduction, 10 to 16 coefficients
Cepstral Analysis	Represents shape of spectral envelope in power domain
Filter bank analysis	Uses filters tuned to specific frequencies
Mel-frequency cepstral coeff (MFCCs)	**Fourier Analysis, filter bank, human auditory pathway**
Kernel based feature extraction	Dimensionality reduction, reduces redundancy in features
Wavelet	It replaces the fixed bandwidth of Fourier transform with one proportional to frequency

Newer features

- MFCC (Mel-Frequency Cepstral Coefficients)
- TECC (Teager-Energy Cepstral Coefficients)
- TEMFCC (Teager-based Mel-Frequency Cepstral Coefficients)
- Features via Deep Denoising AutoEncoders (DDAE)
  - E.g. for generating whisper-robust cepstral features

ASR: architecture 1980-2019

Basic principle

HMM *models* of speech units (words, syllables, phones, context-dependent phones)

Signal preprocessing (front end) 1

Pronunciation model (lexicon) 2

Acoustic models 4

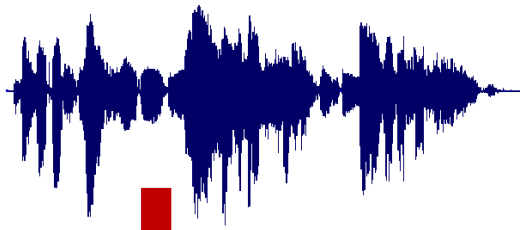
Speech recognition (search algorithm) 5

Language model 3

Words or phones + alignment

N-Best, etc.

ASR components in the modular approach



Feature vectors v1, v2, v3, …

Speech sounds E p @ l …

Words apple

Sentence I eat an …

Front-end (audio → vectors)

Acoustic model ([p] = pdf(vectors)

Lexicon (“apple” [E p @ l])

Language model P(“apple” | “I eat an”)

ASR components in the modular approach

The front-end: extracting features from audio

The lexicon specifies per word the corresponding sequence of speech units. There may be pronunciation variants (more variants per word)

The language model specifies the probability of the next word given N-1 previous words word sequences. (N-gram in the classical aproach)

The acoustic models describe for each speech unit how it sounds i.e. the probability density function of the corresponding feature vectors

The search algorithm looks for the most likely word sequence given the audio (using acoustic model, lexicon and language model)

Lexicon/dictionary (example)

word form	phonemic transcription
bang	**b A N**
bankbiljet	**b A N b I l j E t**
barbaren	**b A r b a r @**
barbecue	**b A r b @ k j u w**
onmiddellijk	**O m I d @ l @ k**
yesterday	**j E s t @ r d e**

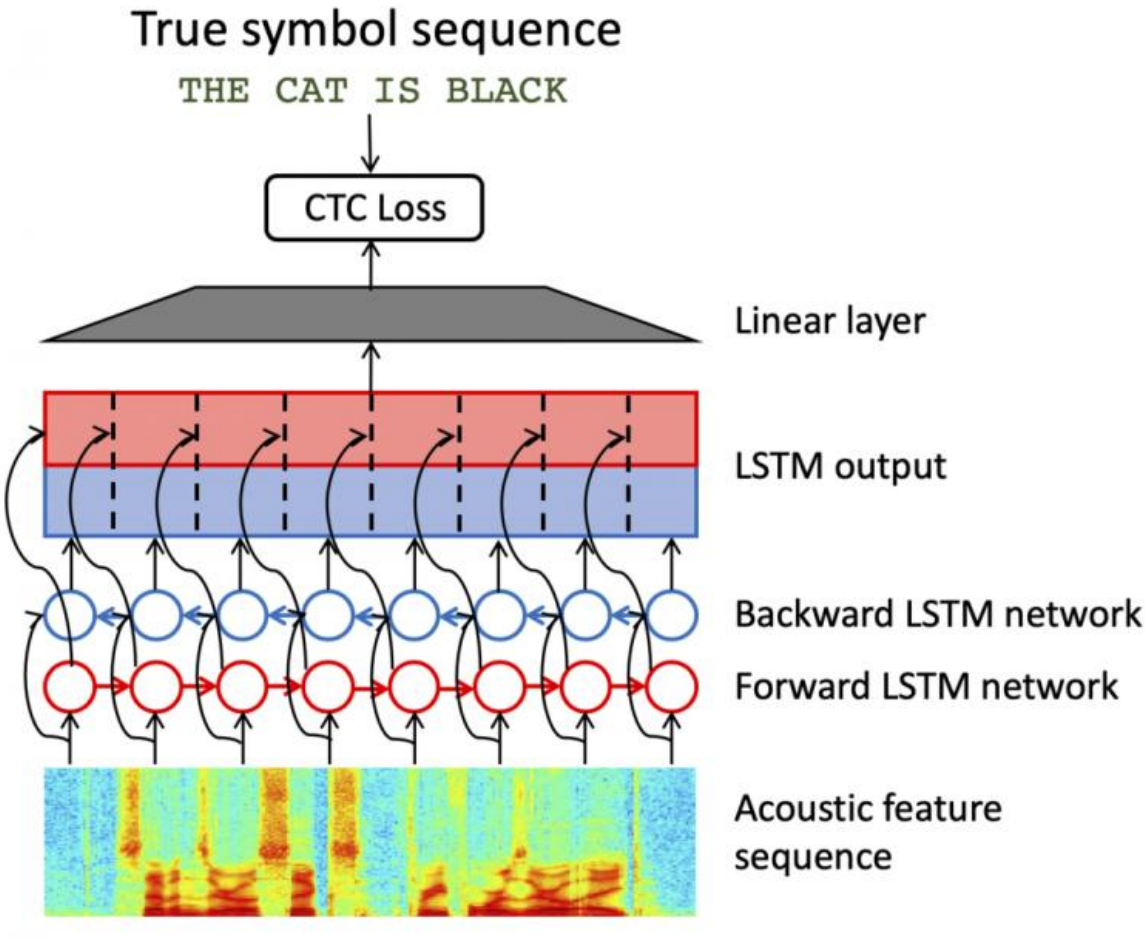
For a well studied language, lexicons are often available;
not always a clean source of information.
Compounding? (*Rechtsschutzversicherungsgesellschaften*)

Modular ASR

- Acoustic model (AM)
- Language model (LM)
- AM and LM are statistical models

- Given “training data” from the target language, we train statistical models
  - Audio + transcriptions → AM
  - Text → LM

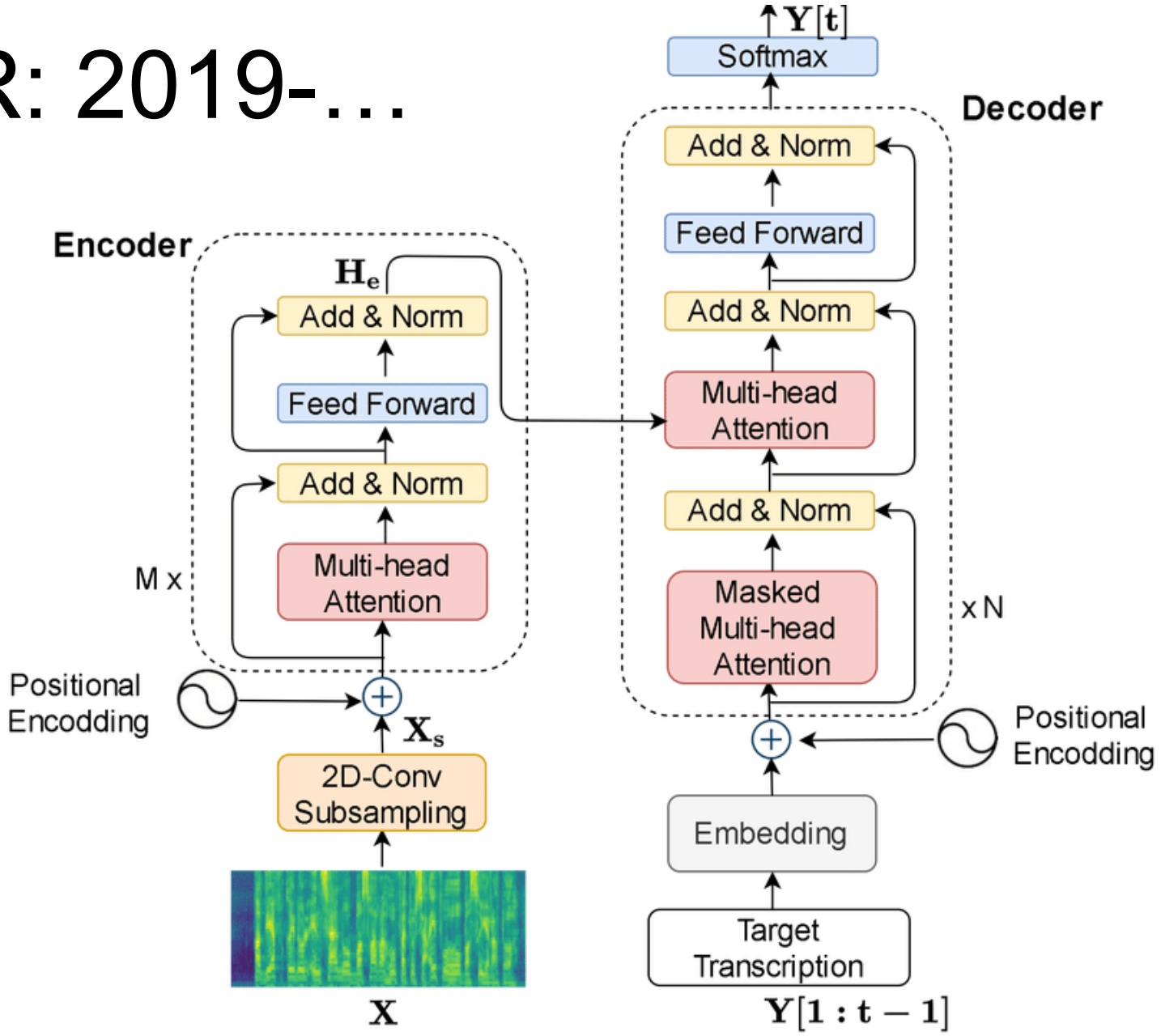
ASR: architecture 2013-…



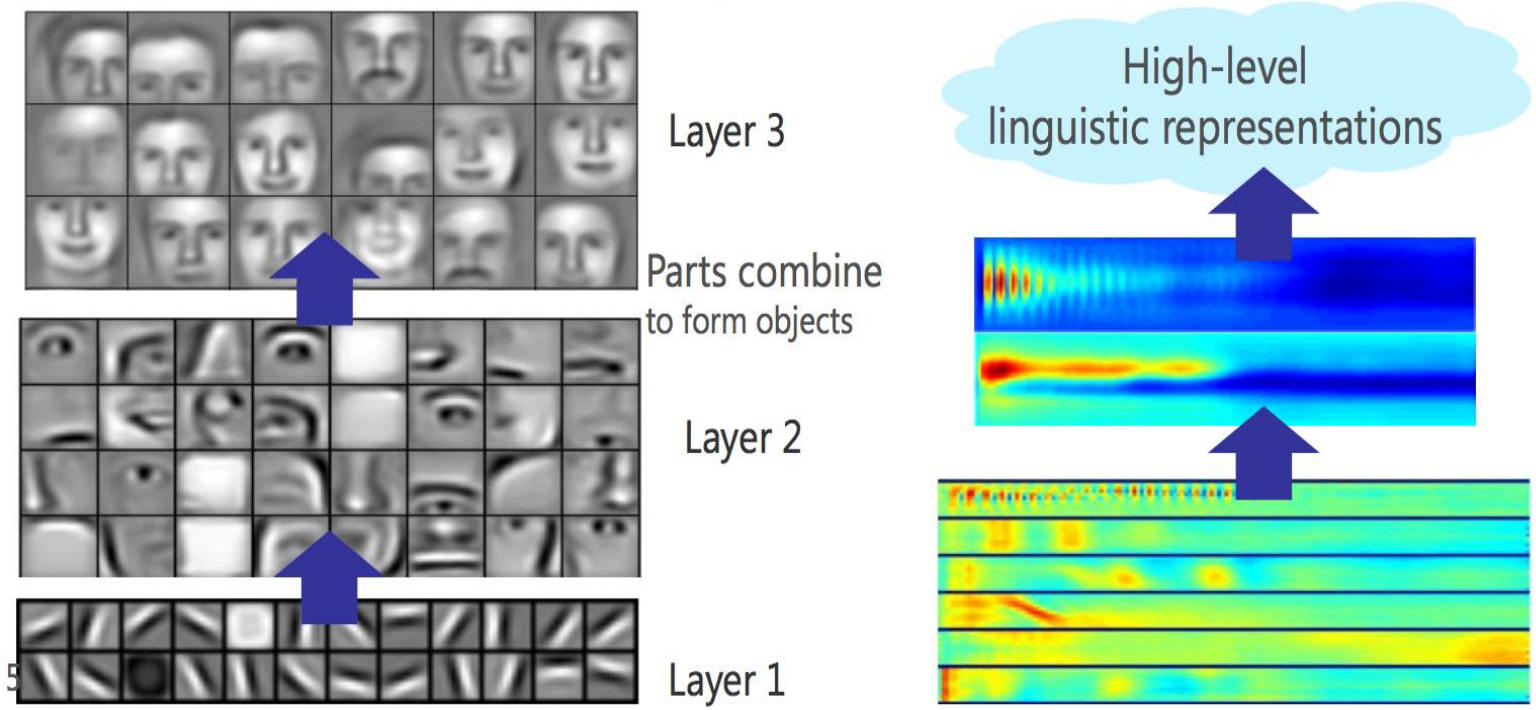
Since 2013, ASR exploits deep learning (RNN, LSTM, biLSTM, Transformers)

Figure from paper by IBM team, Kurata et al. (2019)
One method quickly superseded by another

ASR: 2019-…


Y[t]
Softmax
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-head Attention
Add & Norm
Masked Multi-head Attention
x N
Positional Encodding
Embedding
Target Transcription
Y[1 : t − 1]
Encoder
H_e
Add & Norm
Feed Forward
Add & Norm
Multi-head Attention
M x
Positional Encodding
X_s
2D-Conv Subsampling
X

Successive model layers learn deeper intermediate representations



Prior: underlying factors & concepts compactly expressed w/ multiple levels of abstraction

Lee, Largman, Pham & Ng, NIPS 2009

From conventional approaches to deep learning in ASR

- It is very useful to know about the conventional approach before diving into E2E-based approaches

- Nowadays many (highly specialized) papers on end-to-end architectures
  - e.g. about Wav2Vec2.0, Whisper, w2v-bert-2.0… (see Huggingface, OpenAI, other sources)
  - w2v-bert-2.0:: trained on 500 years = 4.3 Mh

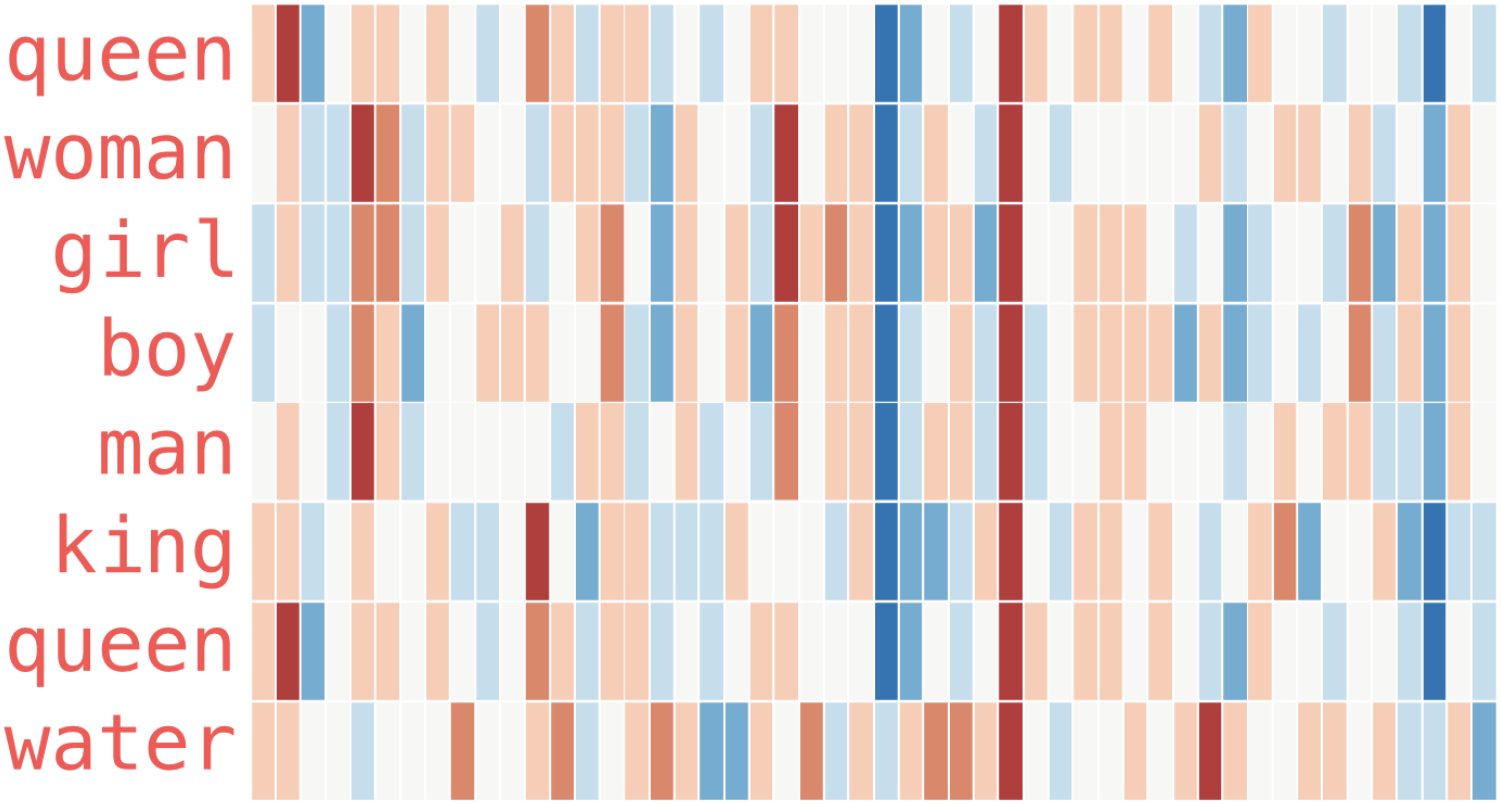
	pro	**con**
Classical approach	Insight-based	Lower performance
Recent deep learning approaches	Higher performance (relative 30-60% reduction error rates)	What do ***we*** learn? Explainable AI Open AI Responsible AI

Deep Neural Networks for ASR

General idea: **Representations → vectors, processes → networks**

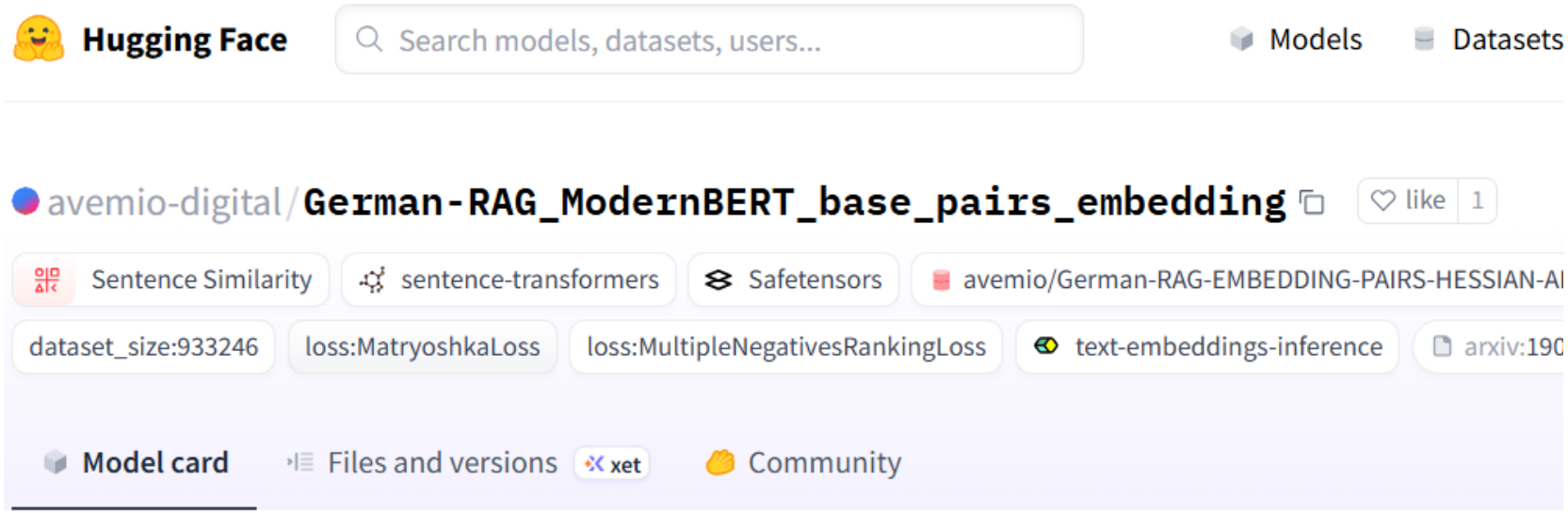
1. Starting from audio, choose preprocessing, get features for each n-millisecond chunk of audio
2. Pass these features through a neural net model (**e.g., citrinet, wav2vec2.0, w2v-bert-v2)** and obtain per-timestep **a logit matrix**
3. This matrix provides character (token) probabilities for each time slice (column)

Words: embeddings, word2vec



From: lajammar.github.io (retrieved sept. 2020)
See e.g. https://radimrehurek.com/gensim/models/word2vec.html

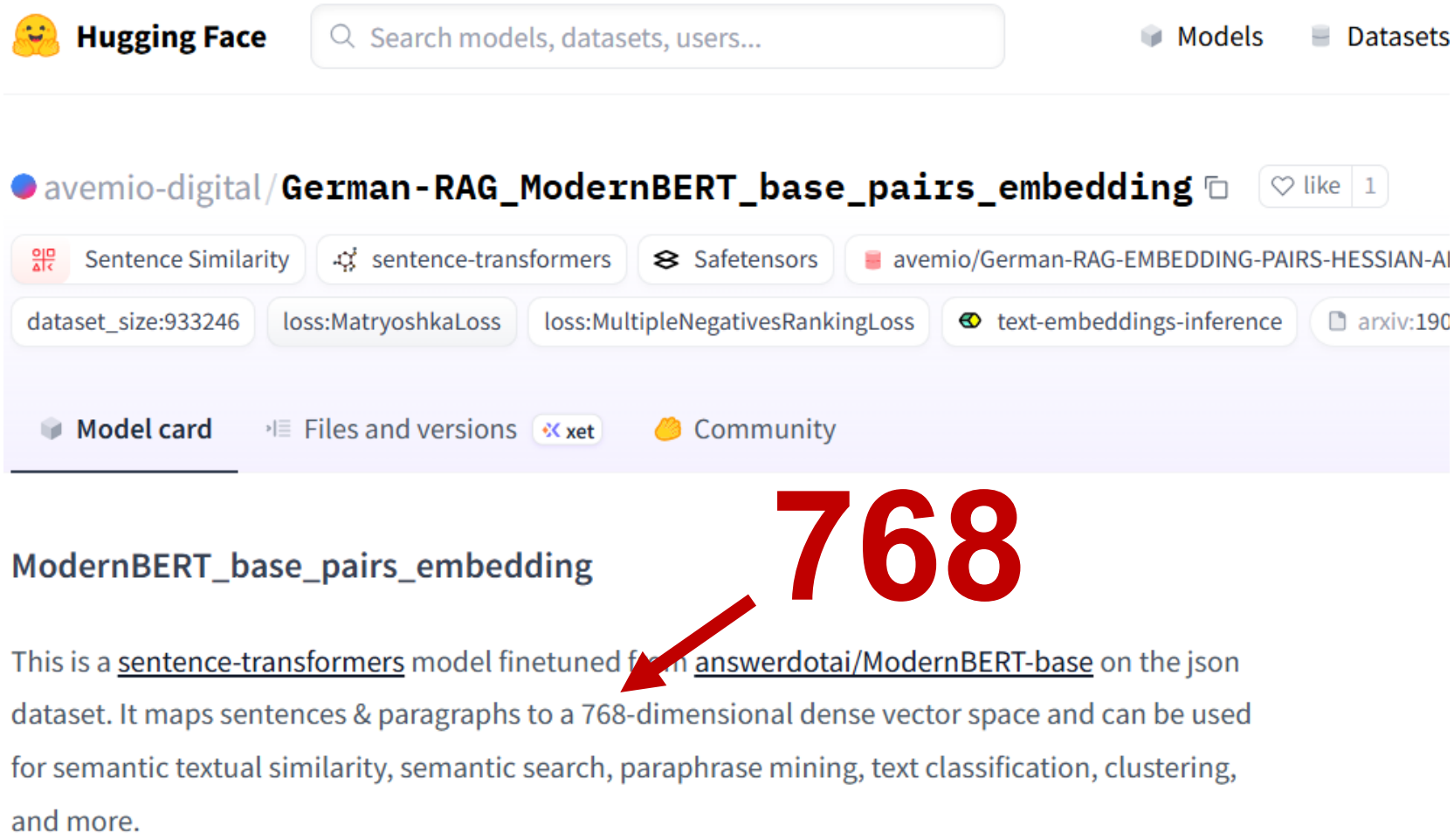
Embedding datasets are available for many languages



ModernBERT_base_pairs_embedding

This is a sentence-transformers model finetuned from answerdotai/ModernBERT-base on the json dataset. It maps sentences & paragraphs to a 768-dimensional dense vector space and can be used for semantic textual similarity, semantic search, paraphrase mining, text classification, clustering, and more.

Embedding datasets are available for many languages



word2vec and semantic relations

- **king**
- **[('king', 2.1073424255447017e-08), ('prince', 0.764360429535011 3), ('queen', 0.80283043754882 8), ('crown', 0.91192980048772 02), ('iii', 0.92171756478424 33), ('princess', 0.94665289642478 78), ('kings', 0.95712388081205 2), ('iv', 0.96017867848173 95), ('reign', 0.96304529634213 07)]**

- **blue**
- **[('blue', 1.4901161193847656e-08), ('white', 0.6977357395612396), ('red', 0.727167071580242 5), ('black', 0.75881830175367 06), ('yellow', 0.77042207269591 91), ('green', 0.79641604303120 23), ('purple', 0.80037167157437 4), ('pink', 0.83721368737399 38), ('orange', 0.84186821514725 97)]**

Distances: cosine distance

The following slides discuss Wav2vec2.0

as one of the open examples of recent end-to-end ASR architectures

Wav2vec2.0

- Wav2vec 2.0's architecture is based on the Transformer's encoder, with a training objective like BERT's masked language modeling objective, adapted for speech.

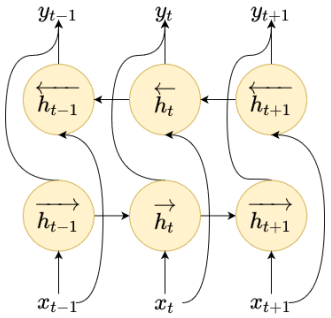
- This allows for efficient semi-supervised training
  - first, pre-train the model on a large quantity of unlabeled speech, then
  - fine-tune on a smaller labeled dataset.

- In wav2vec 2.0's original paper (Baevski et al, 2020), the authors showed that fine-tuning the model on only **one hour of labeled speech data** could beat the previous state-of-the-art systems trained on 100 times more labeled data.

Deep Neural Networks for ASR (S2T)

General idea: **Representations → vectors, processes → networks**

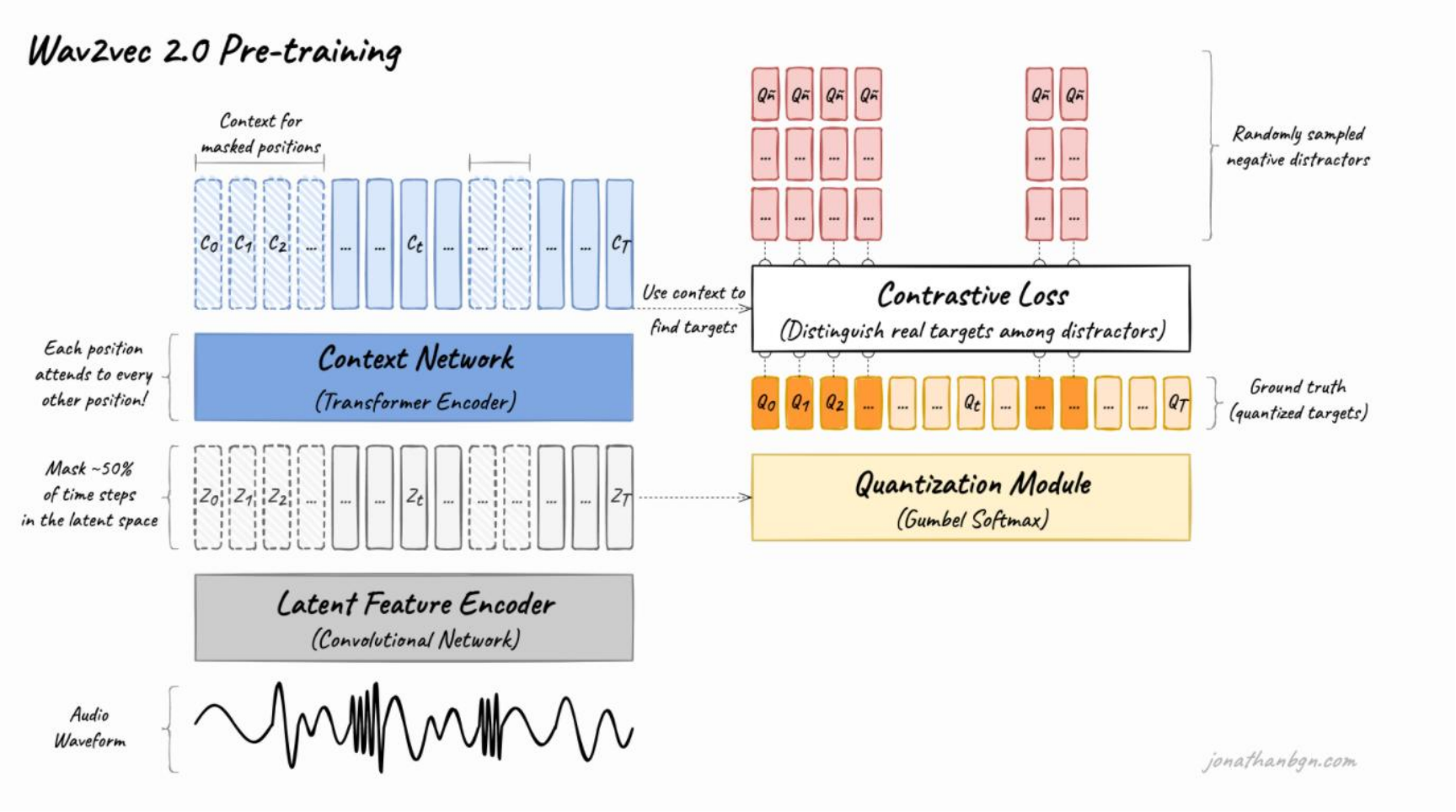
1. Starting from audio, choose preprocessing, get features for each n-millisecond chunk of audio
2. Pass these features through a neural net model (**e.g. wav2vec, citrinet, wav2vec2.0, …) and obtain per-timestep a logit matrix**
3. **This gives softmax logits predicting character probabilities for each time slice**

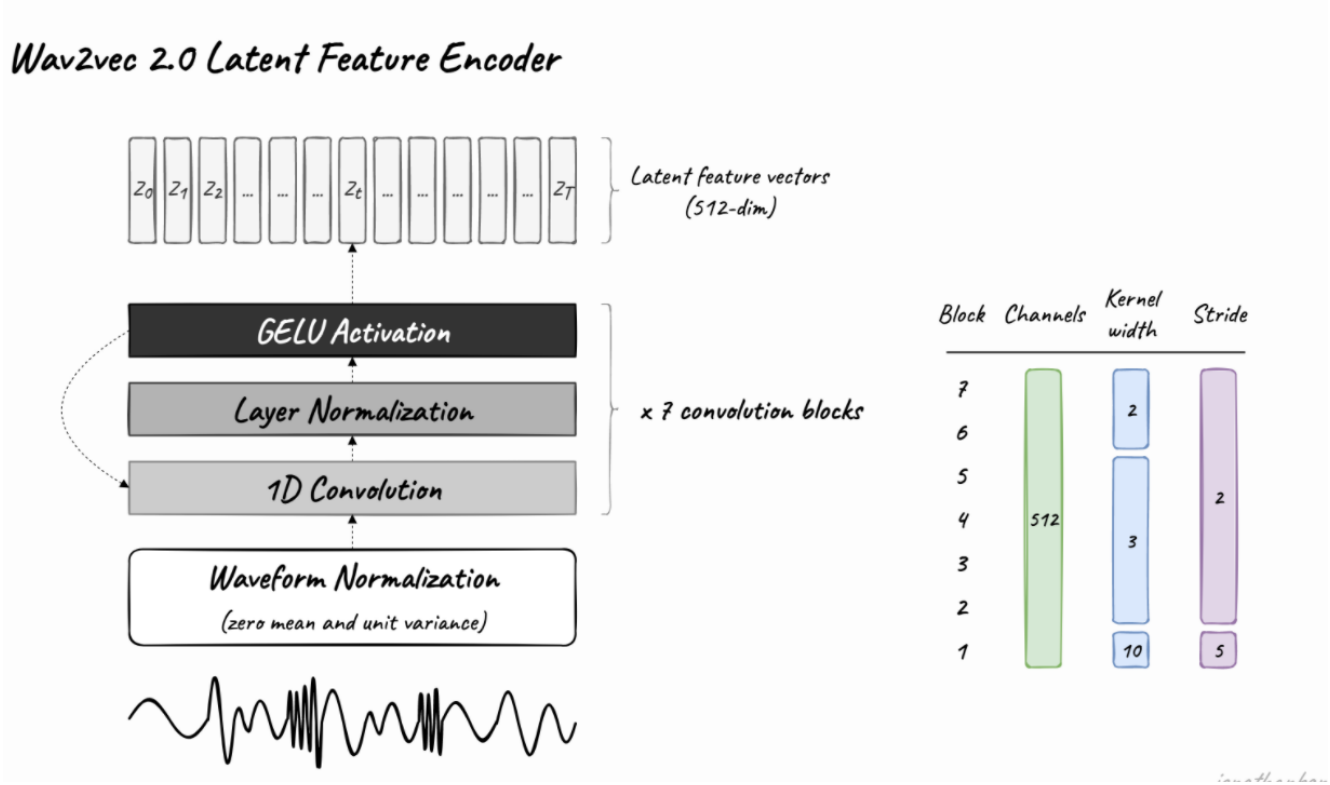


convolutional neural networks (CNNs)
recurrent neural networks (RNNs**), (b)LSTM, BERT, …**

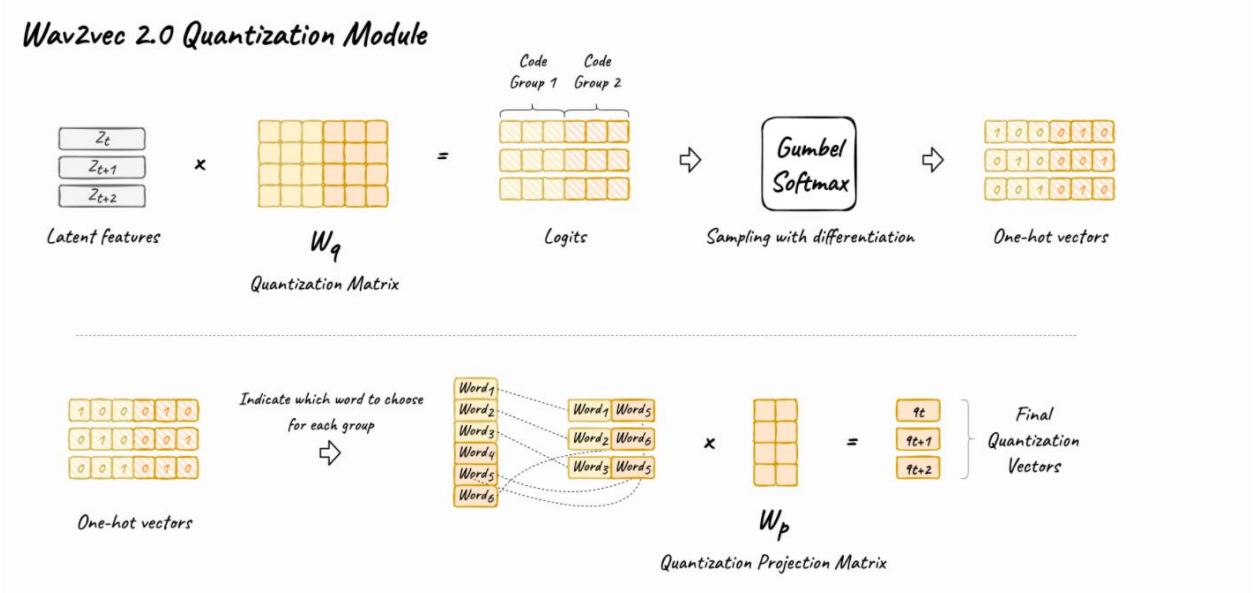
current hidden state is dependent on all the previous hidden states
long-term and short-term dependencies between different time-steps of the input
input signal → hidden sequence → output sequences

Global architecture
Wav2vec2.0





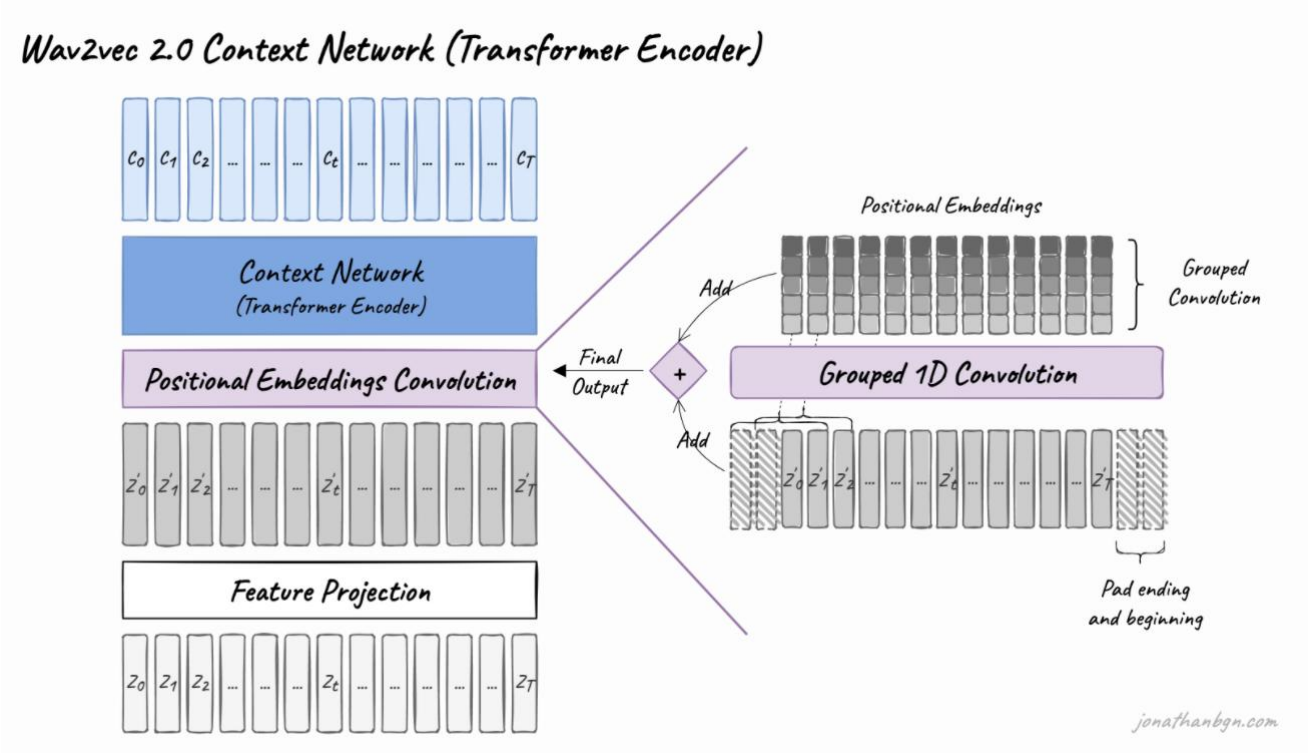
The feature encoder's job is to reduce the dimensionality of the audio data, converting the raw waveform into a sequence of feature vectors Z_0, Z_1, Z_2, ..., Z_T each 20 milliseconds. Its architecture is simple: a 7-layer convolutional neural network (single-dimensional) with 512 channels at each layer.



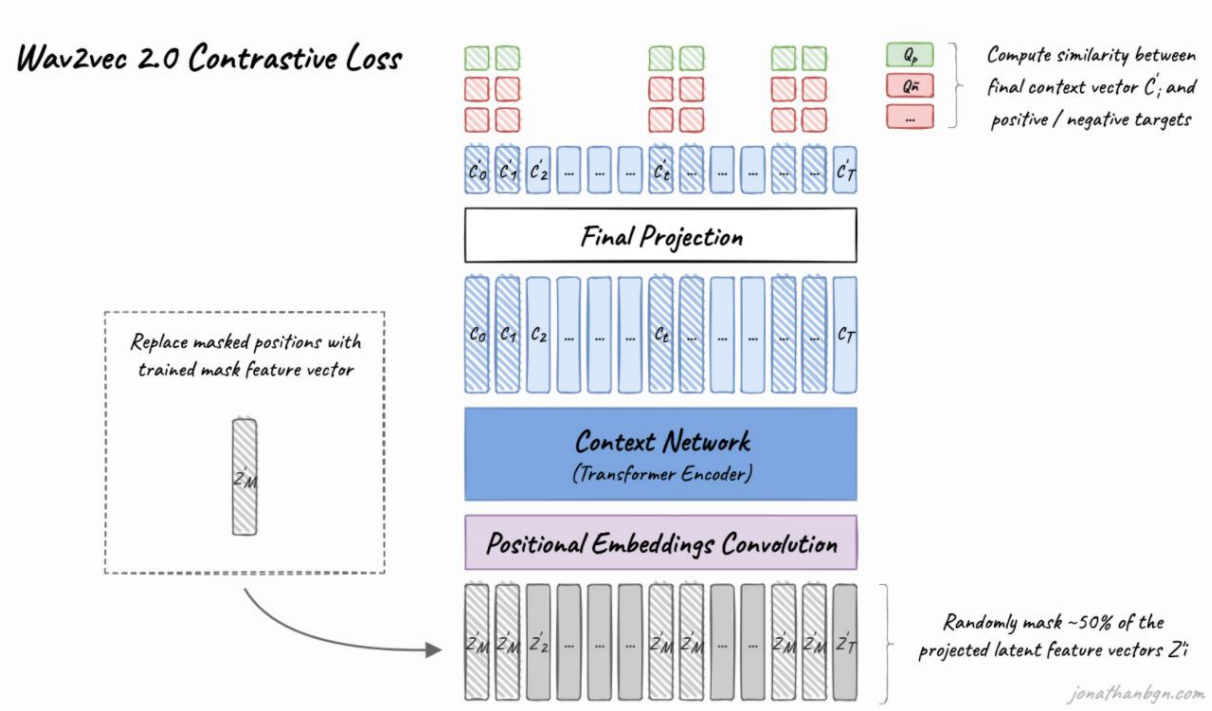
Wav2vec 2.0 proposes to **automatically learn discrete speech units**, by sampling from the **Gumbel-Softmax distribution**. Possible units are made of *codewords* sampled from *codebooks* (groups). *Codewords* are then concatenated to form the final speech unit.

Wav2vec uses 2 groups with 320 possible code words in each group, hence a theoretical maximum of 320 x 320 = 102,400 speech units.

https://jonathanbgn.com/2021/09/30/illustrated-wav2vec-2.html



The core of wav2vec 2.0 is its Transformer encoder, which takes as input the latent feature vectors and processes it through 12 Transformer blocks for the *BASE* version of the model, or 24 blocks for the *LARGE* version. To match the inner dimension of the Transformer encoder, the input sequence first needs to go through a feature projection layer to increase the dimension from 512 (output of the CNN) to 768 for *BASE* or 1,024 for *LARGE*.



The final context vectors then go through the last projection layer to match the dimension of the quantized speech units Q_t.

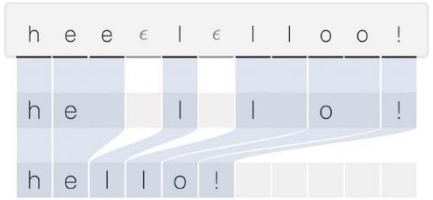
For each masked position, **negative distractors are uniformly sampled from other positions in the same sentence.**

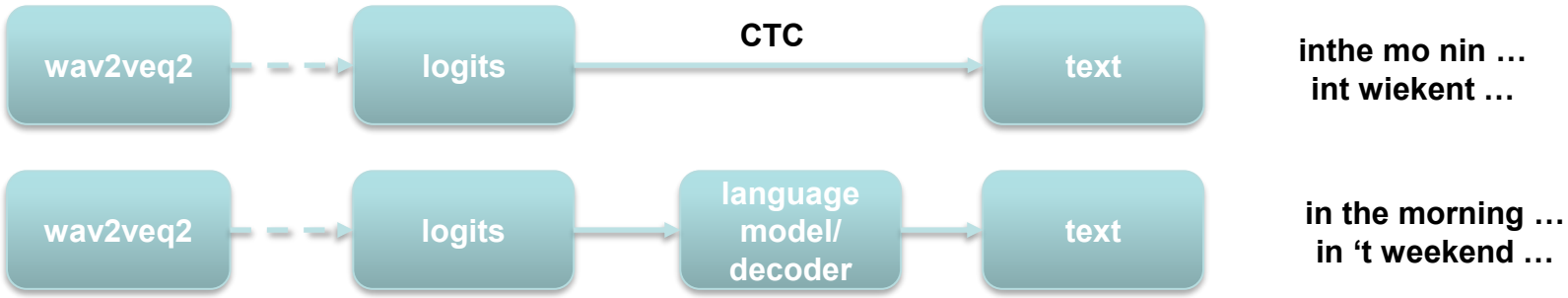
The model then compares the similarity (**cosine similarity**) between the projected context vector C'_t and the true positive target Q_p along with all negative distractors $Q_{\tilde{n}}$.

The contrastive loss then encourages high similarity with the true positive target and penalizes high similarity scores with negative distractors.

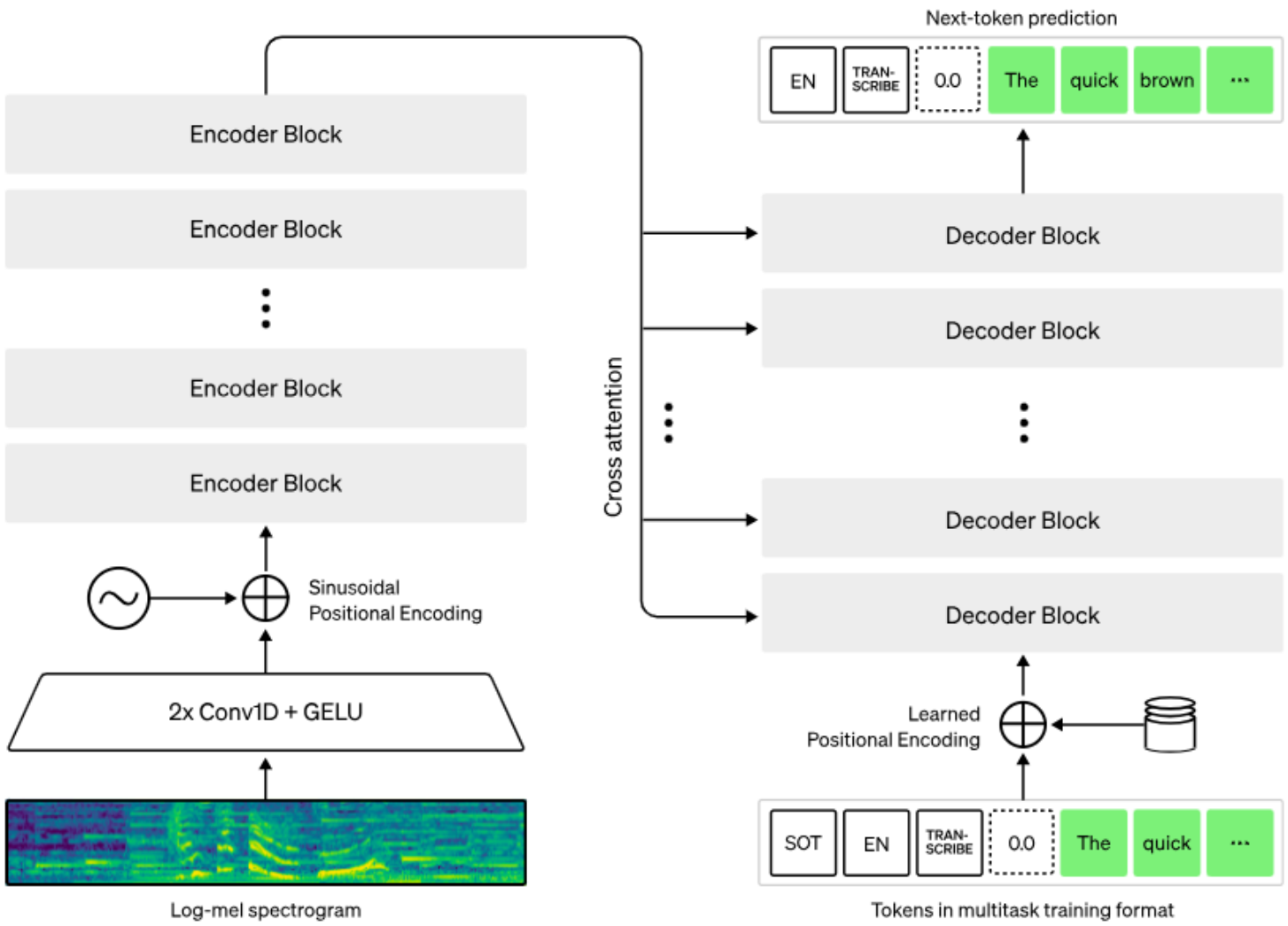
Logit matrix

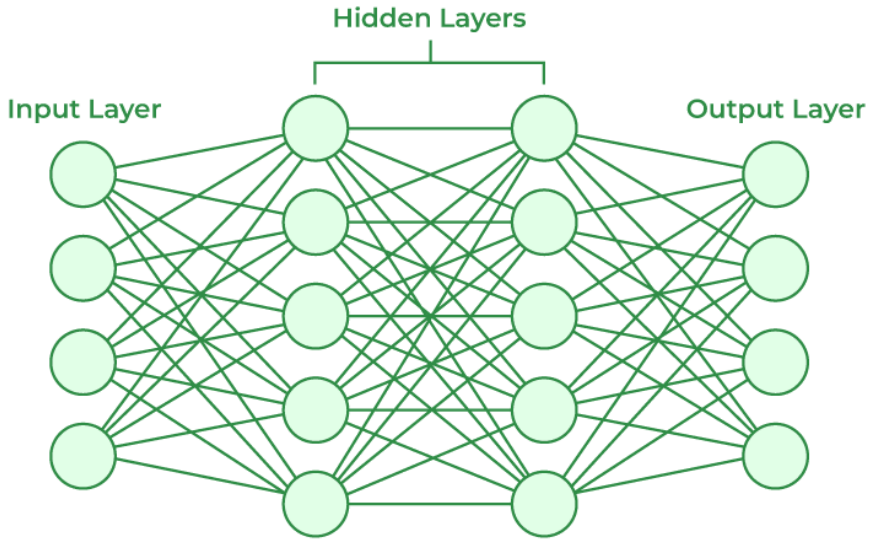
- Size of the logit matrix depends on the audio length
- Pass the generated per-timestep logits to a language model/decoder
- This gives actual text
- The decoder is comparing different paths through the logit matrix

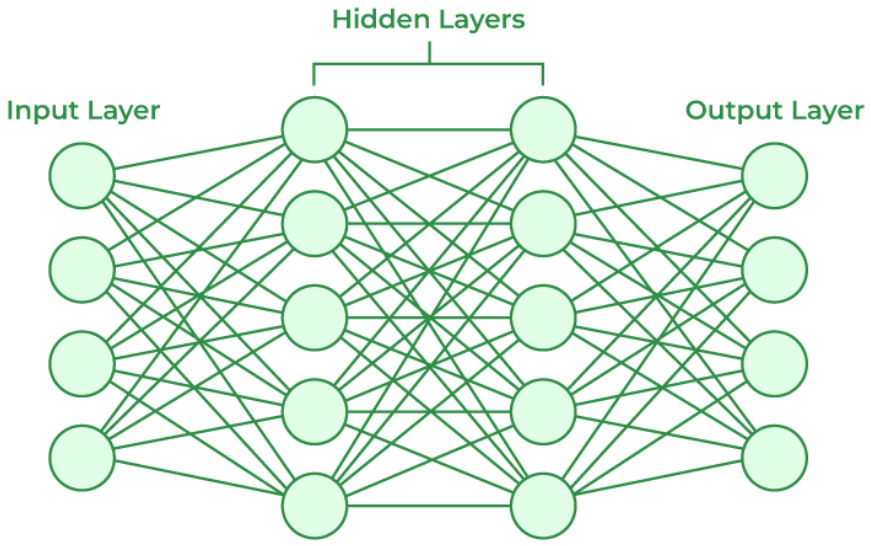


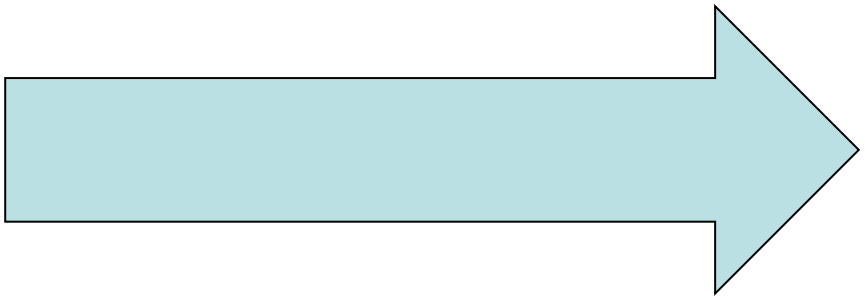


Whisper: encoder-decoder

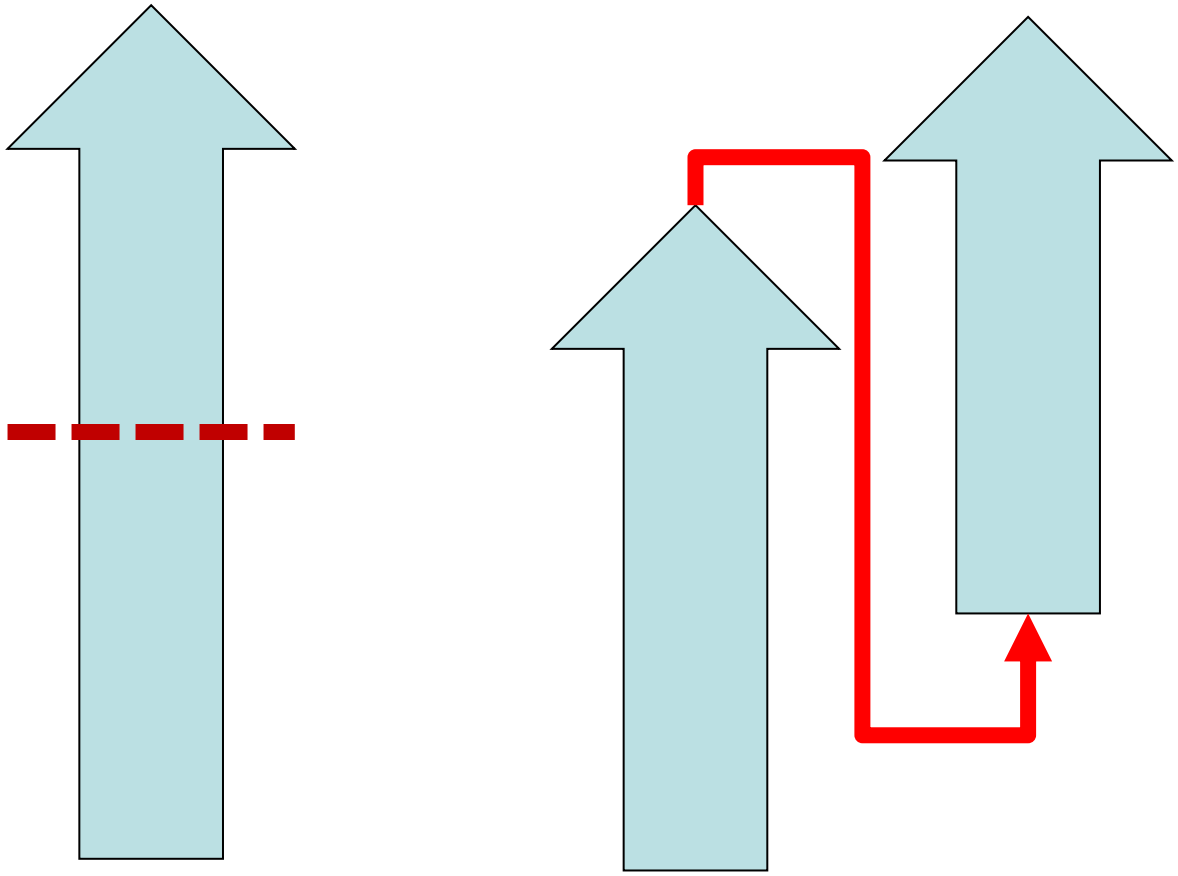


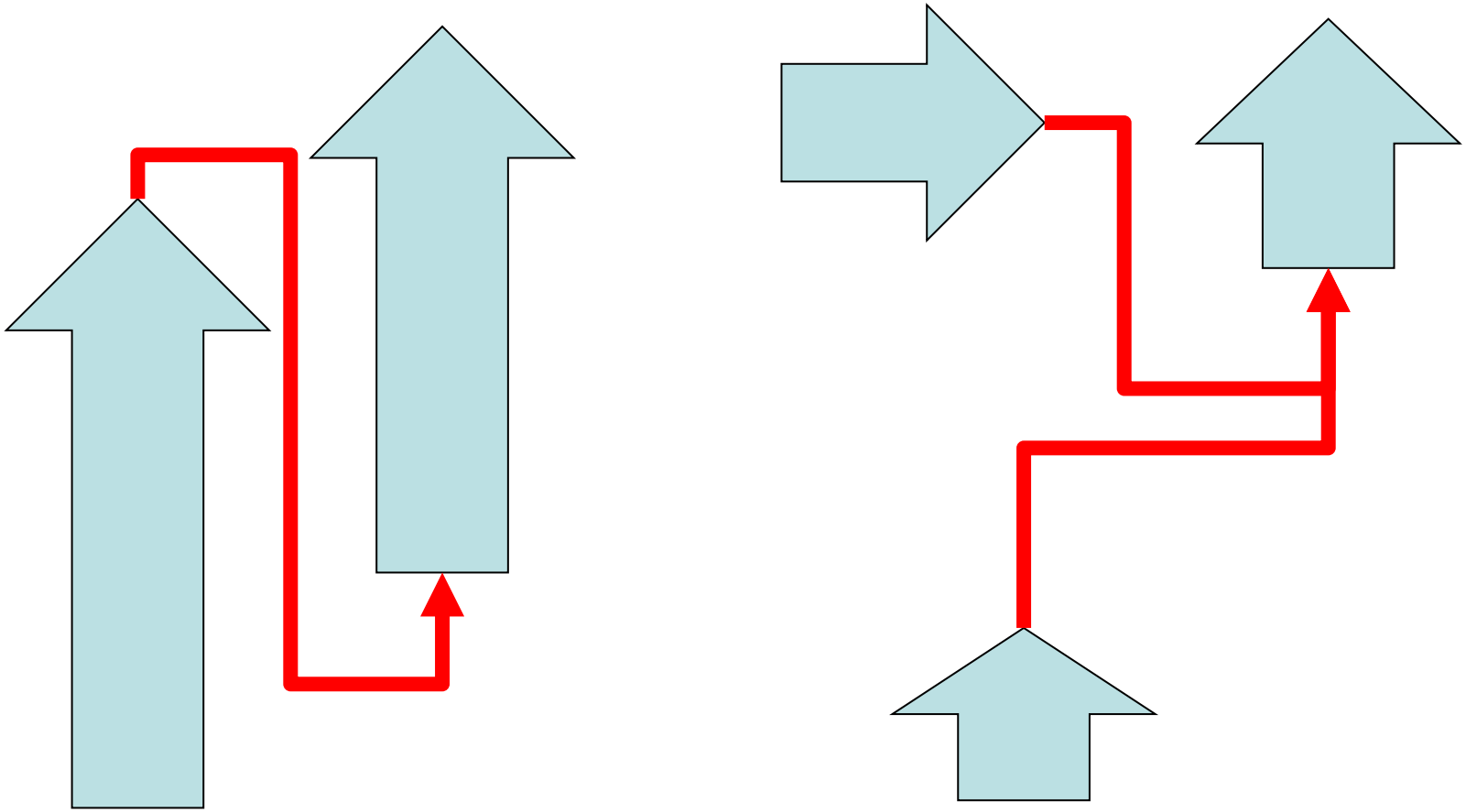

Hidden Layers
Input Layer
Output Layer



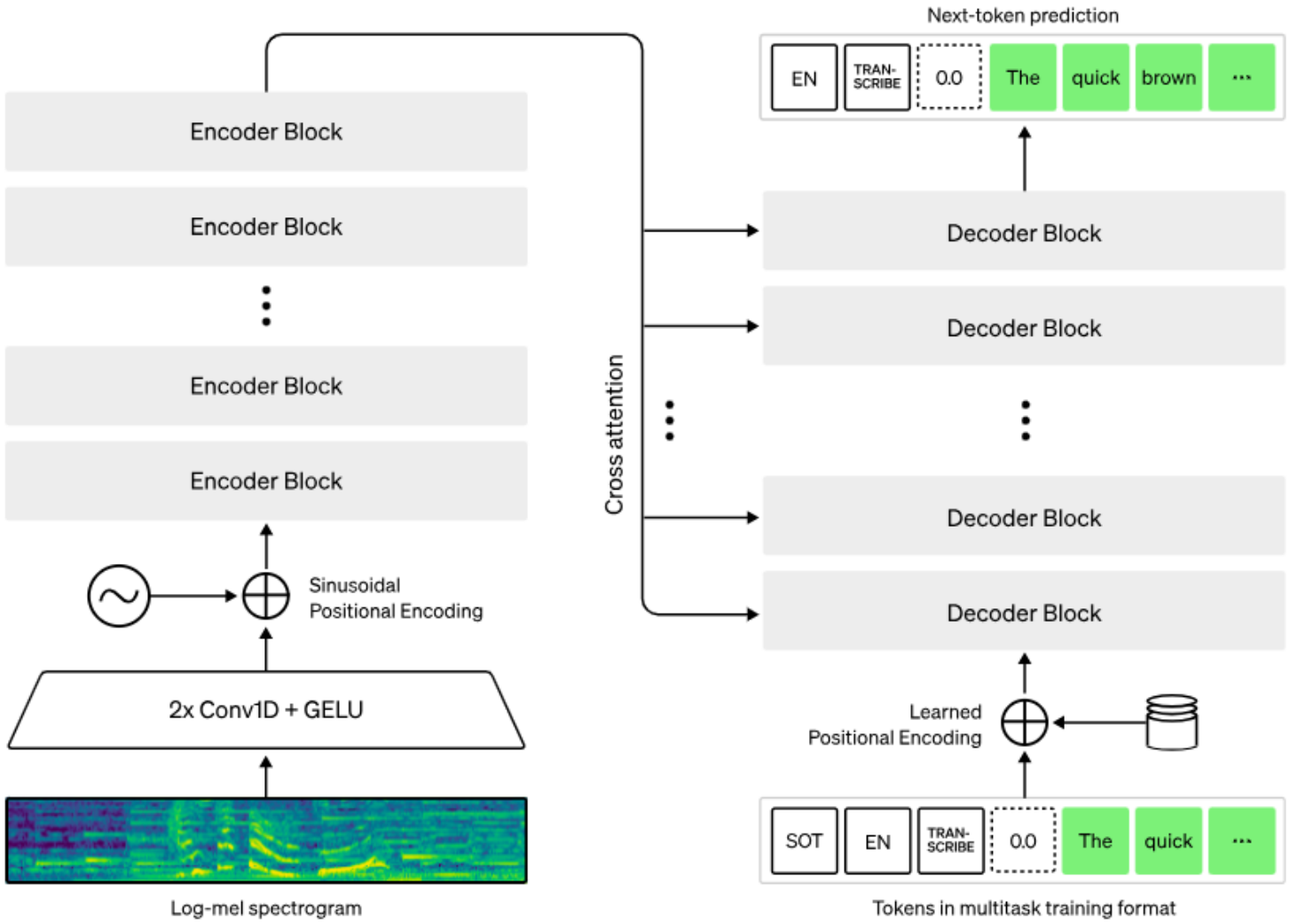


Information flow



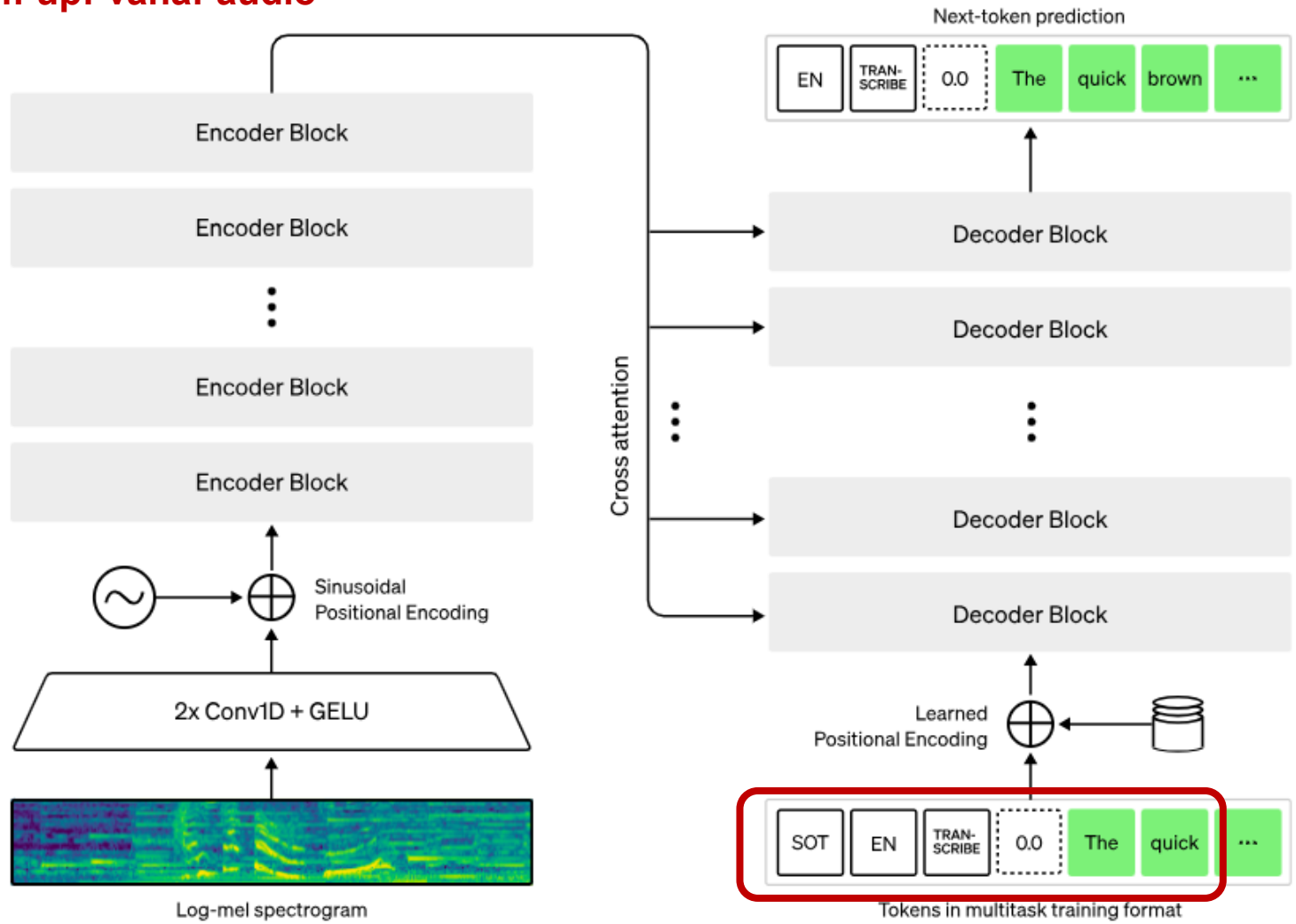


Whisper: encoder-decoder



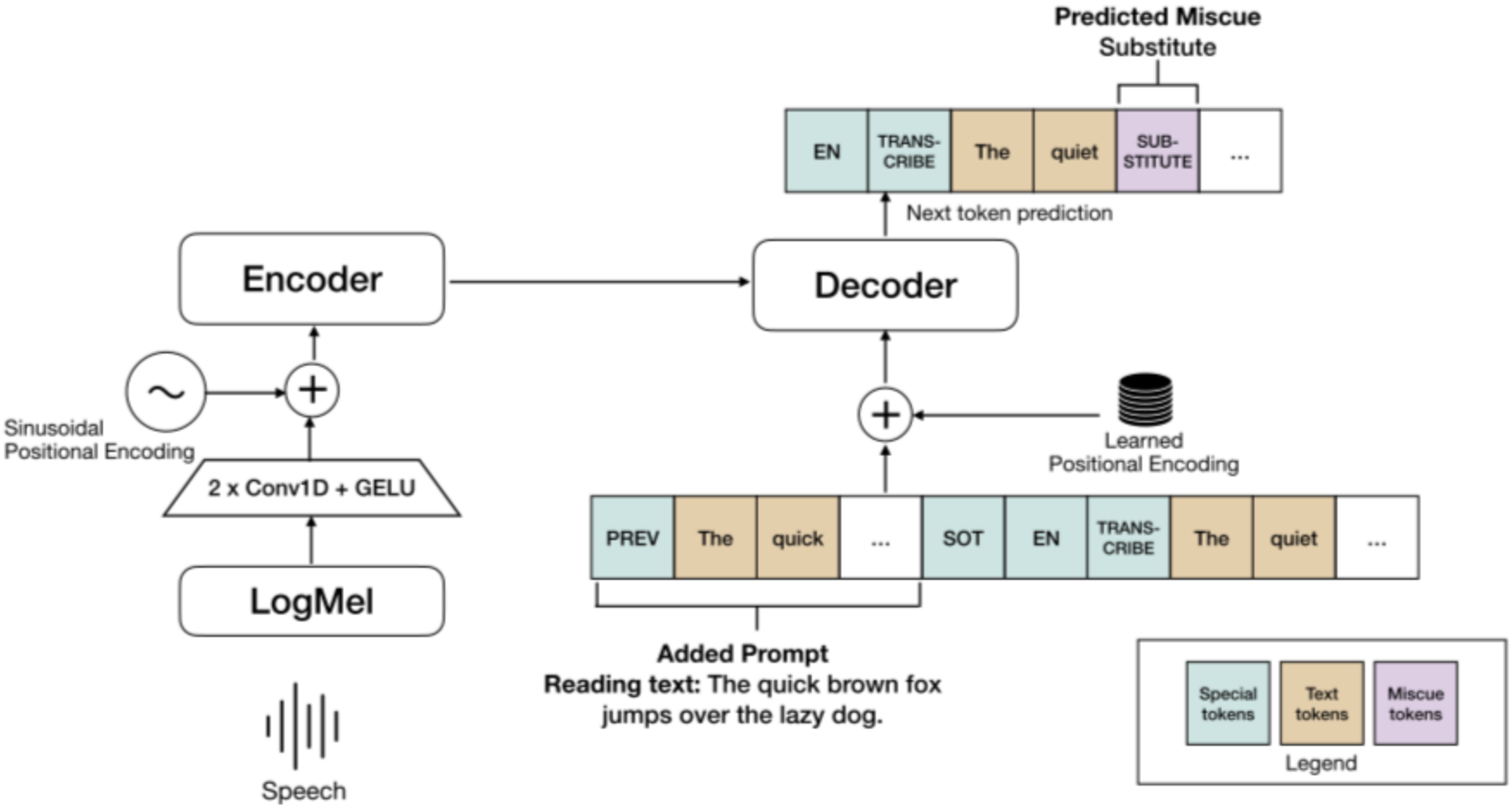
Whisper: encoder-decoder

Bottom-up: vanaf audio

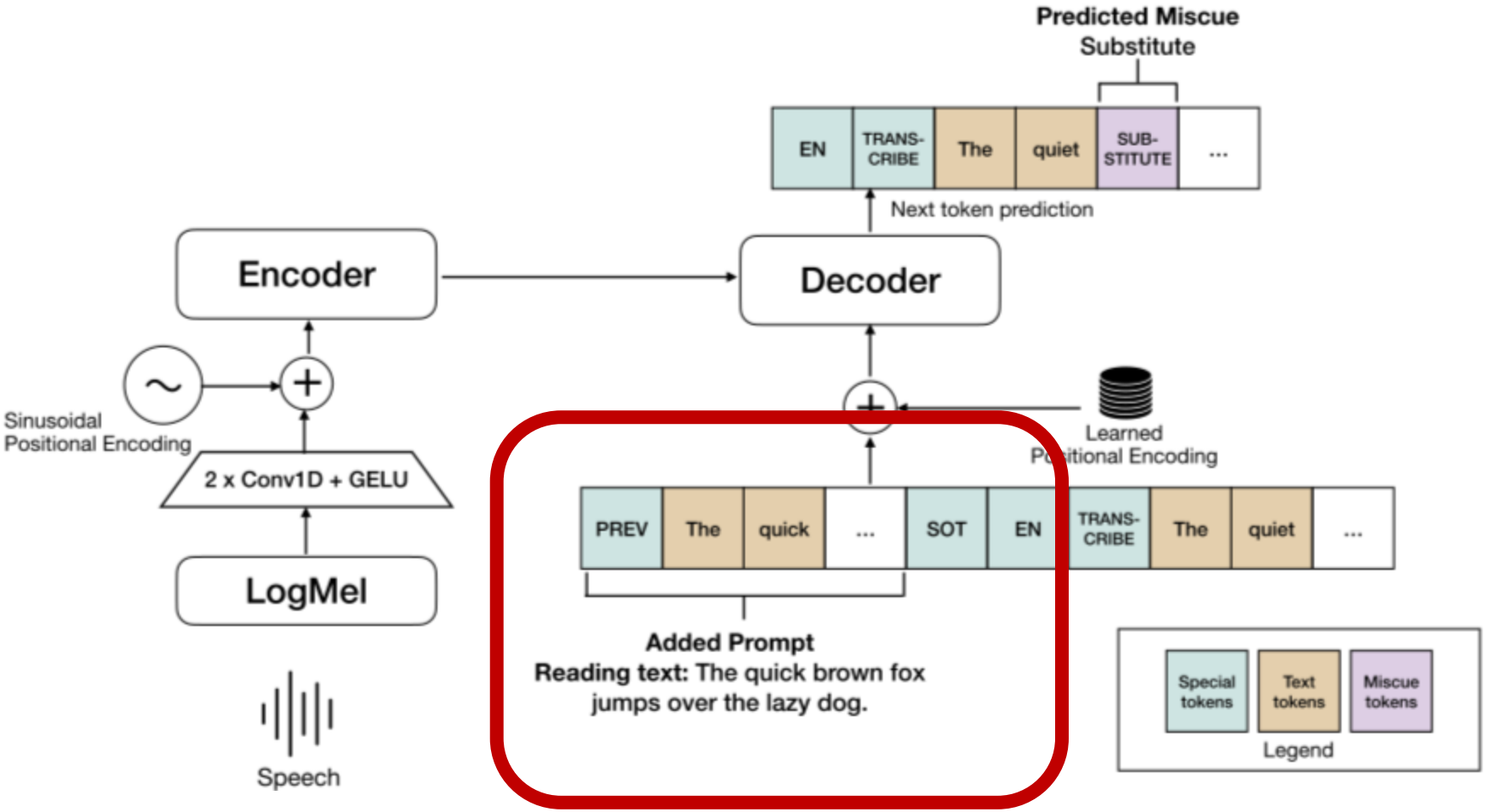


Precontext

https://www.themoonlight.io/en/review/prompting-whisper-for-improved-verbatim-transcription-and-end-to-end-miscue-detection



https://www.themoonlight.io/en/review/prompting-whisper-for-improved-verbatim-transcription-and-end-to-end-miscue-detection



Text prompt (more later)

Extraction of information

- How to extract information from the network?
- Explanability: understand I/O mapping
- Interpretability: understand process between I and O

Extract information from network

Speech representations can be extracted from various layers in the pipeline:

- in latent feature encoder
  - after encoding (512 dimensions, Baevski et al., 2020),
  - after quantization (512 dimensions)

- in transformer encoder
  - after each of the 12 (or 24) transformer layers (768 dimensions per layer).

Is latent information always informative?

- the final layers do not tend to be the most informative layers for downstream tasks
- informative abstractions/generalizations in hidden layers might be discarded in favor of actual target output

Knowledge distillation (Hinton et al., 2015)

Given a complex classification/recognition problem

1. Train many different classifiers/recognition systems, or one large model
2. Try to condense the better ones (the large model) into a small model

In Hinton's terms:

- The cumbersome model could be an ensemble of separately trained models or a single very large model trained with a very strong regularizer such as dropout.
- Once the cumbersome model has been trained, we can then use a different kind of training, which we call "distillation", to transfer the knowledge from the cumbersome model to a small model that is more suitable for deployment.

Example: (1) the 'cumbersome' model uses softmax with temperature *T*: $q = \frac{\exp(\frac{z}{T})}{\sum \exp(\frac{z_i}{T})}$

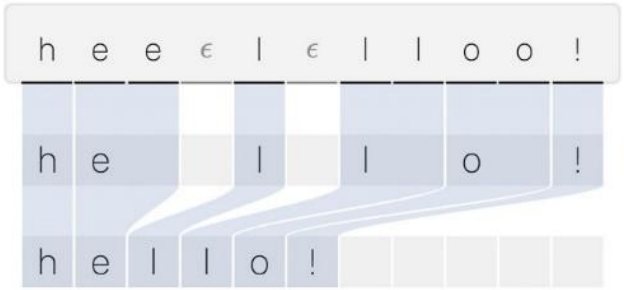
(2) in small model: freeze *T*

The following slides present CTC

as frequently used in the end-to-end ASR training architectures

Connectionist Temporal Classification (CTC)

- **CTC computes the alignment between the input speech signal and the output sequence of the words**
- CTC uses a blank label – that represents a pause or a transition between words or phonemes.
- For a given transcription sequence, there are several possible alignments
  - (a, −, b, c, −, −) (*a*, −, *b*, *c*, −, −) and (−, a, −, b, −, c, −, −) (*a*, *b*, *c*)
    both correspond to the character sequence
    (a, b, c) (*a*, *b*, *c*)



- CTC kicks out doubles

- **CTC aims to maximize the total probability of the correct alignments in order to get the correct output word sequence**
  - it doesn't require prior segmentation or alignment of the data

End2end approaches

- Connectionist Temporal Classification (CTC)
  - Solves the problem of unaligned input and output sequences by marginalizing the conditional likelihood of the output sequence given the input over all possible alignments
- Attention Modeling
  - Simultaneously optimize alignment and grapheme (or word) decoding using attention weights (linear combination of hidden states) to influence the generated output

Modern ASR papers are like *cookbooks*. Example (wav2vec2.0):

Our model is composed of a multi-layer convolutional feature encoder f : X → Z which takes as input raw audio X and outputs latent speech representations z1, …, zT for T time-steps. They are then fed to a Transformer g : Z →C to build representations c1, …, cT capturing information from the entire sequence. The output of the feature encoder is discretized to qt with a quantization module Z → Q to represent the targets in the self-supervised objective. Compared to vq-wav2vec, our model builds context representations over continuous speech representations and self-attention captures dependencies over the entire sequence of latent representations end-to-end.

The encoder consists of several blocks containing a temporal convolution followed by layer normalization and a GELU activation function. The raw waveform input to the encoder is normalized to zero mean and unit variance. The total stride of the encoder determines the number of time-steps T which are input to the Transformer. The output of the feature encoder is fed to a context network which follows the Transformer architecture. Instead of fixed positional embeddings which encode absolute positional information, we use a convolutional layer which acts as relative positional embedding. We add the output of the convolution followed by a GELU to the inputs and then apply layer normalization. (…)

Many pretrained models available on HuggingFace

HuggingFace is a general repository. MANY (Sept 2025, over 2 million) models, e.g.

- A Wav2Vec2 model that is pretrained on relatively clean LibriSpeech data – Wav2Vec2 model that is pretrained and fine-tuned on noisy + clean data
- An XLS-R model, which is known for great multilingual and multi-accent performance -- pretrained on the CommonVoice corpus
- A huBERT model, improves or matches Wav2Vec2 performance and extends the Wav2Vec2 architecture (Hsu et al., 2021), pretrained on LibriSpeech data
- Whisper (OpenAI) and variants: robust, open-source Automatic Speech Recognition (ASR) system developed by OpenAI and released in September 2022, trained on 680,000 hours of diverse, multilingual internet data

HuBERT, one of the successors of wav2vec2.0

HuBERT looks very similar to wav2vec 2.0: both models use the same convolutional network followed by a transformer encoder.

However, their training processes are very different:

- HuBERT uses the **cross-entropy loss**, instead of the more complex combination of contrastive loss + diversity loss used by wav2vec 2.0. This makes training easier and more stable
- HuBERT builds targets via a **separate clustering process**, while wav2vec 2.0 learns its targets simultaneously while training the model (via a quantization process using Gumbel-softmax).
- HuBERT **uses embeddings from the BERT encoder** to improve targets, while wav2vec 2.0 only uses the output of the convolutional network for quantization.
- https://jonathanbgn.com/2021/10/30/hubert-visually-explained.html

Impact of LM in deep networks

- Transformer layers in deep learning
  - a stack of identical encoder and decoder layers,
  - each with an attention mechanism (self-attention and, for decoders, encoder-decoder attention) and a feedforward network.
  - Transformers use attention mechanisms which can flexibly deal with context, much better than LSTMs or biLSTMs or RNNs can do.
  - Thanks to the attention layer, the transformer layers can pay attention to (extremely large) contexts.
- Depending on the architecture, an LM (LLM) can be used to modulate the impact of bottom-up information.

Impact of LM in deep networks

- by weaving the LM states into the decoder

  $$P(\text{output} \mid \text{audio}) \approx P(\text{acoustics}) \times P(\text{linguistic context})$$

- by shallow fusion in the beam search

  $$\log P(y|x) \approx \log P_{ASR}(y|x) + \lambda \log P_{LM}(y)$$

- By deep fusion

- By a two-pass system
  - first pass creates the hypothesis lattice, the second pass uses LM scores to rescore

prompt tuning

- https://arxiv.org/pdf/2312.08079

- Target-speaker automatic speech recognition aims to transcribe the speech of a target speaker from multi-talker speech.
- Most of the existing target-speaker ASR (TSASR) methods involve either training from scratch or fully finetuning a pre-trained model
- This paper leverages prompt tuning, a parameter-efficient fine-tuning approach, to extend Whisper, a large-scale single-talker ASR model, to TS-ASR.
- Results show that prompt tuning can achieve performance comparable to state-of-the-art full training approaches while only requiring about 1% of task-specific model parameters.

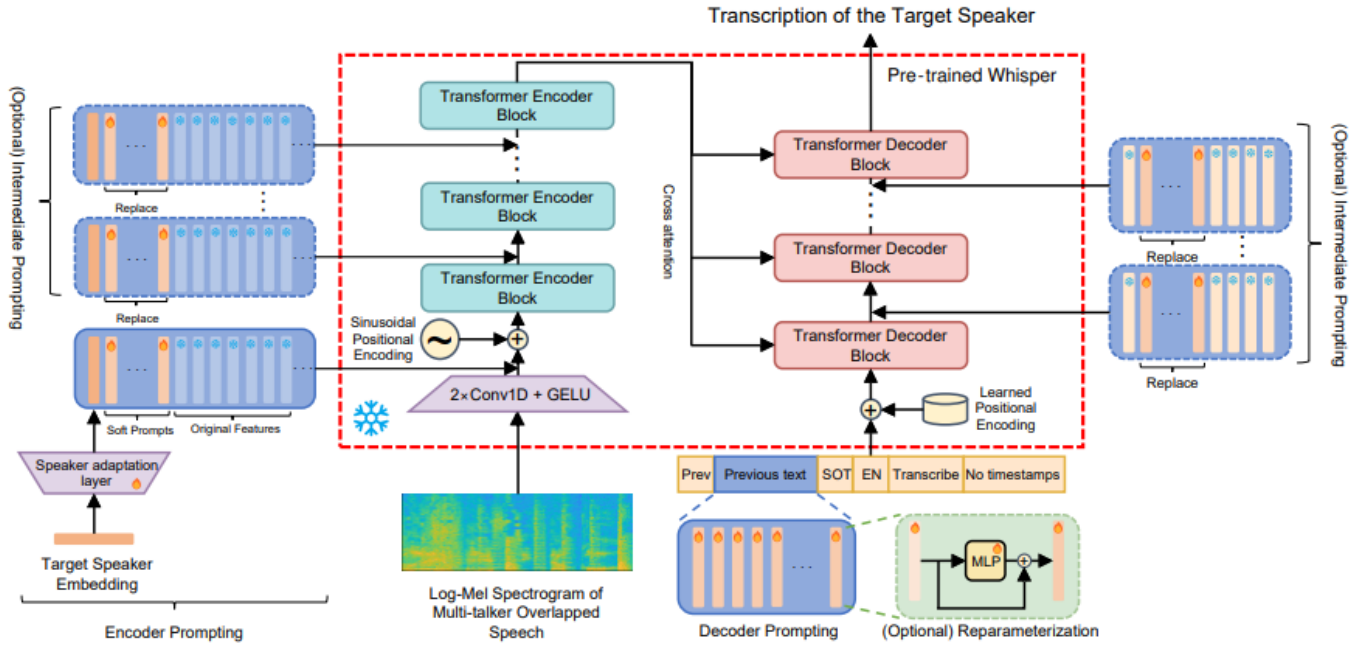


Fig. 1. *Overview of proposed prompting framework. Modules with solid-line borders are included in the baseline configuration, while modules with dashed-line borders are optional.*