# Architectures ASR
# 2025-6

Radboud University, Nijmegen

Louis ten Bosch

# 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)

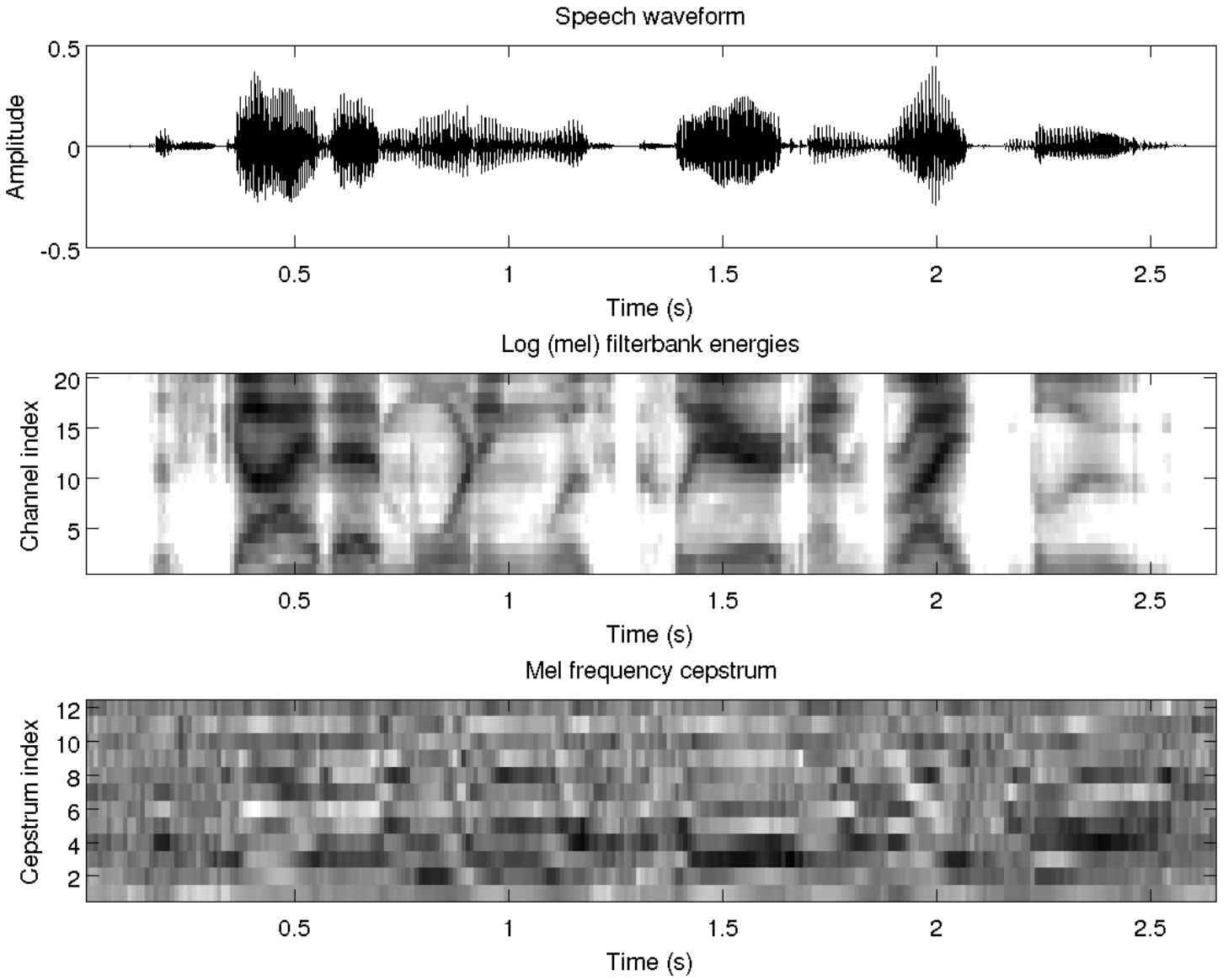
Vectorization → ideal for deep learning and neural networks

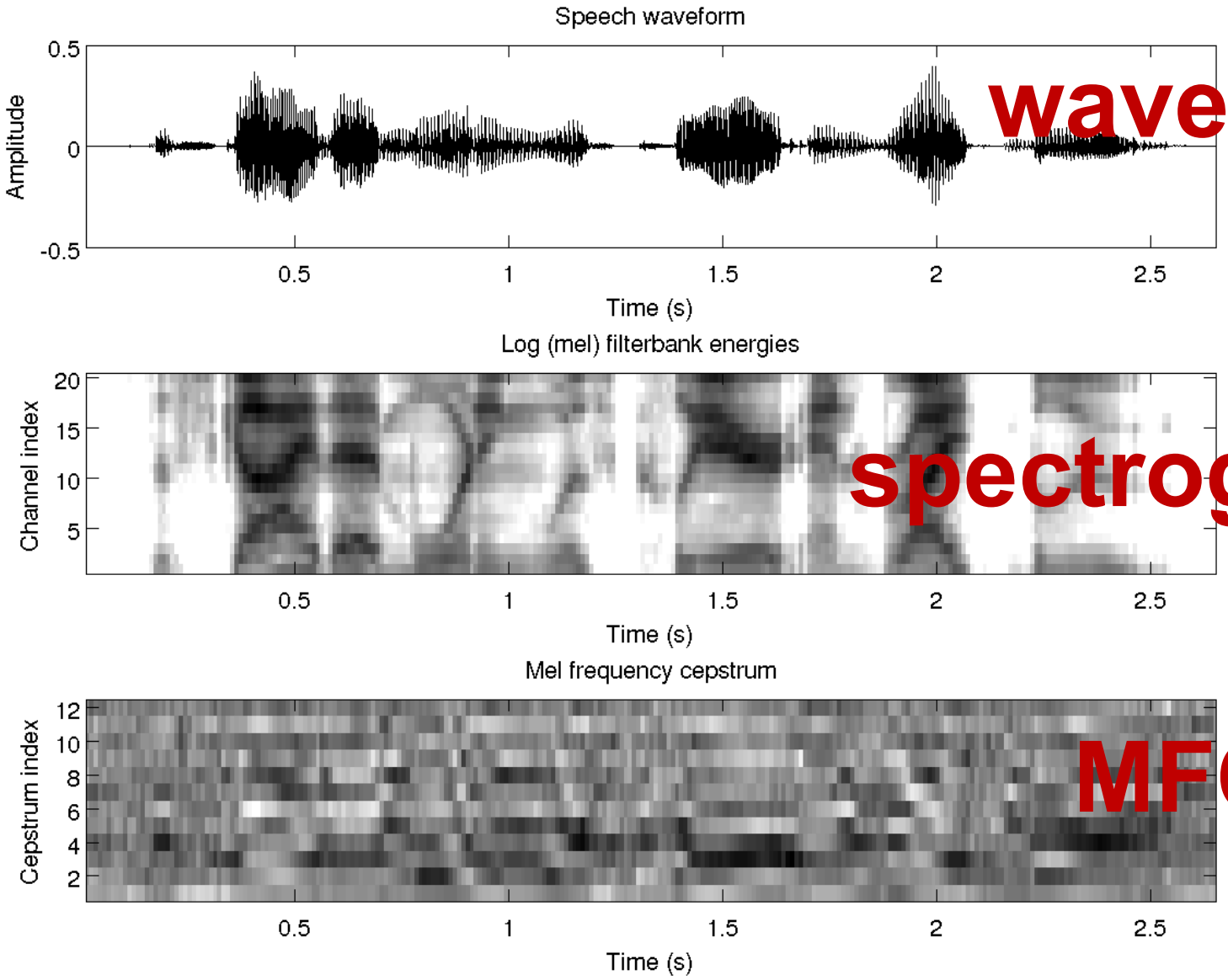
# From audio to features

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**

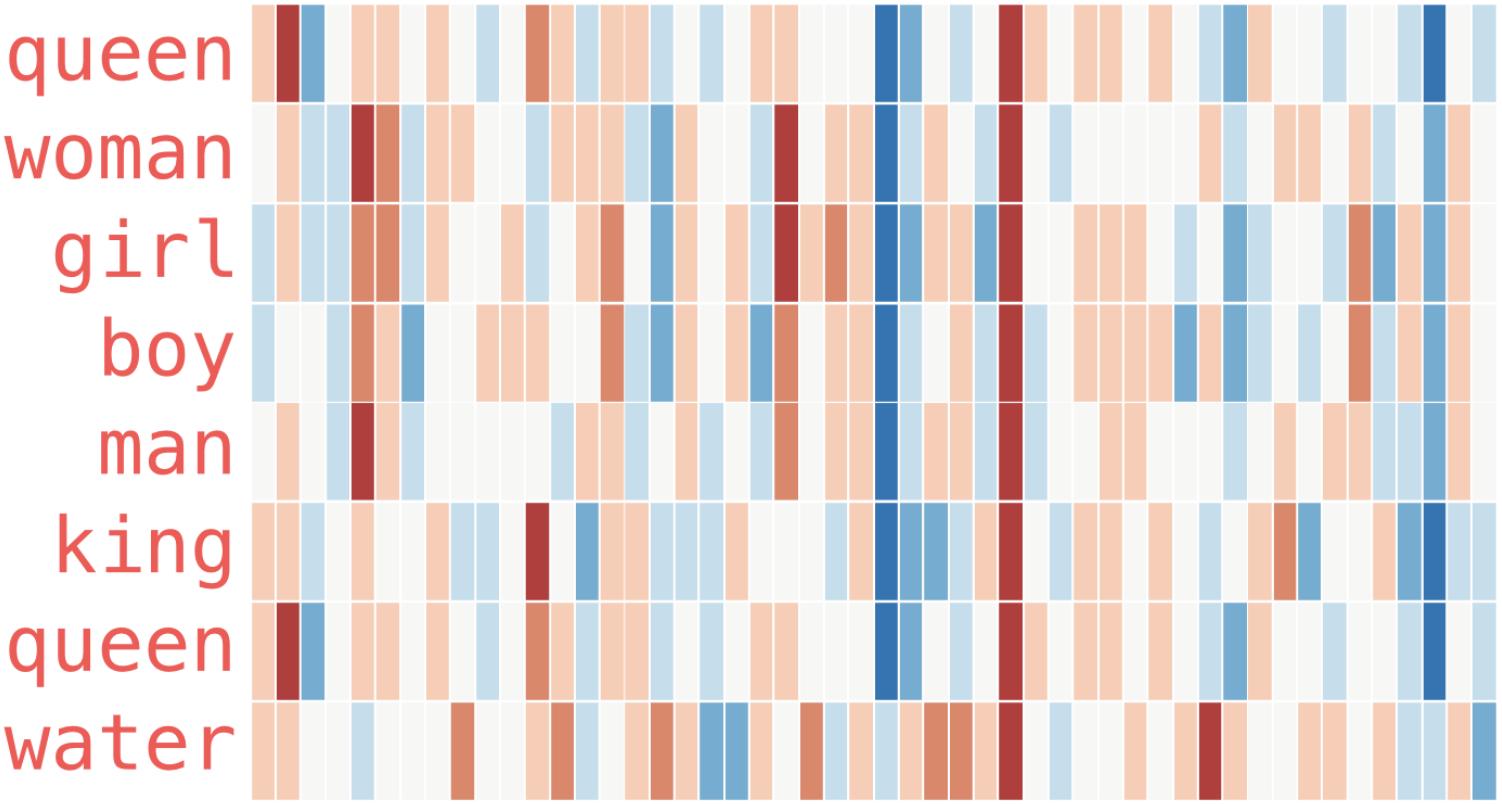
# MFCC vectors

MFCC = Mel Frequency Cepstral Coefficients

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

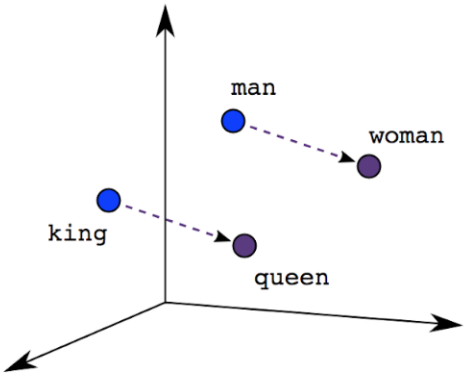
- https://www.kaggle.com/ilyamich/mfcc-implementation-and-tutorial
- https://pypi.org/project/python_speech_features/
- Librosa python library https://librosa.org/doc/latest/index.html

# Words: embeddings, word2vec

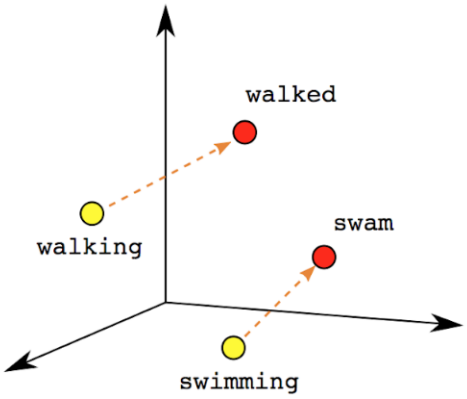



**From: lajammar.github.io (retrieved sept. 2020)**
**See e.g. https://radimrehurek.com/gensim/models/word2vec.html**

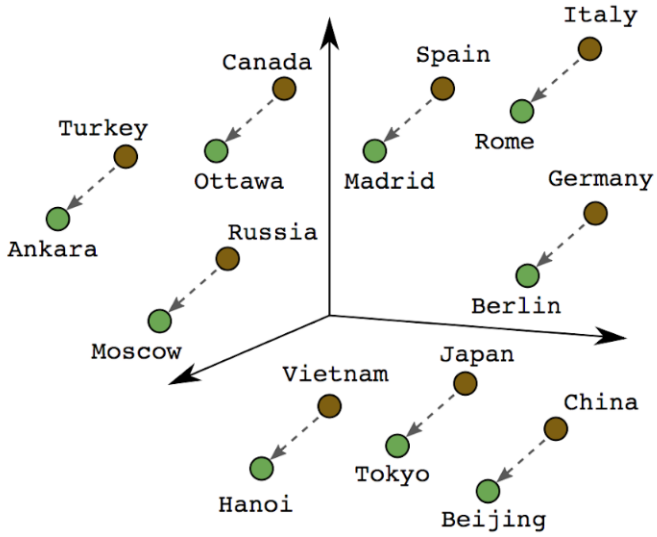
# WORDS AS VECTORS (WORD EMBEDDINGS)




Male-Female



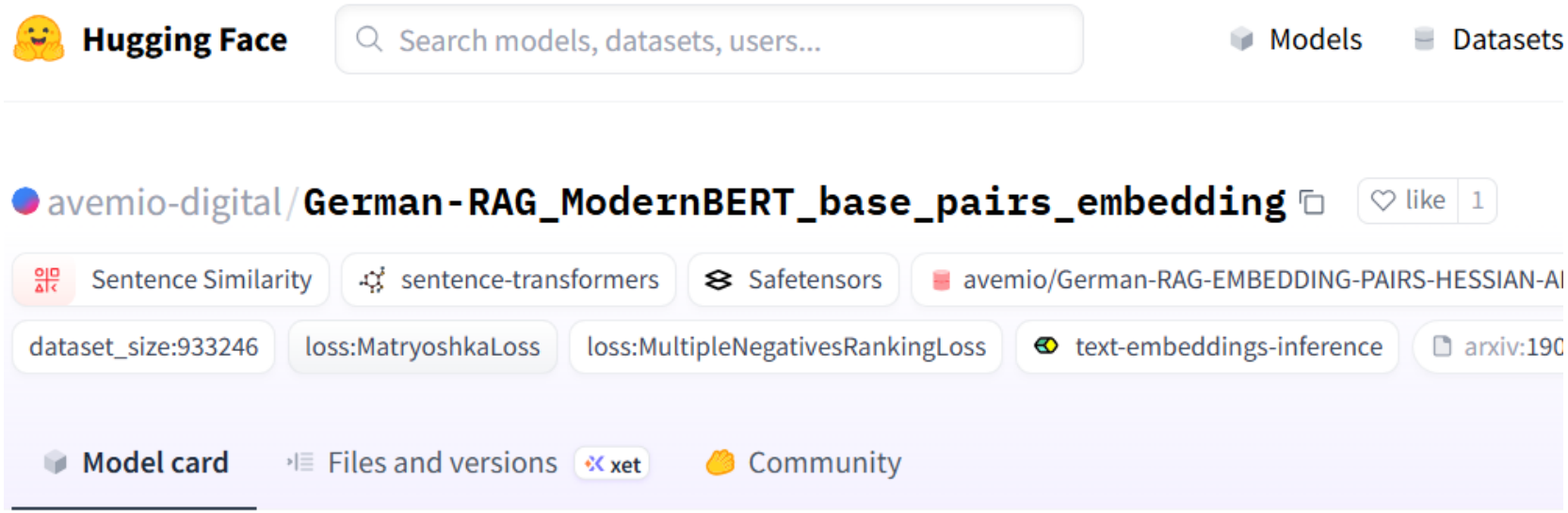

Verb Tense




Country-Capital

**Dimension??**
**100, 300 (word2vec)**
**768 (BERT, …)**
**1024 (Wav2vec2.0)**

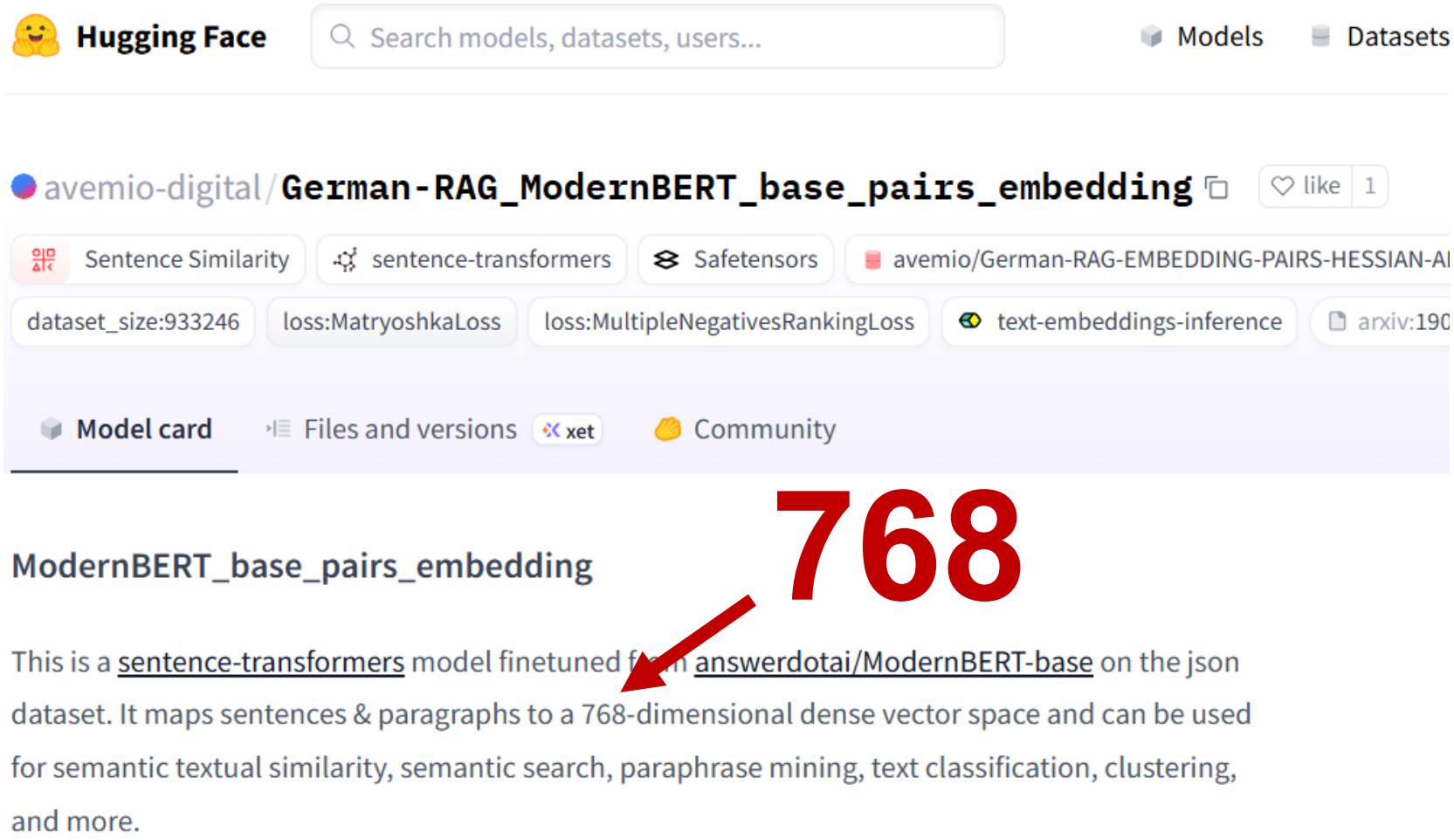
# Embedding datasets are available for many languages




## 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**

# ASR: architecture 1980-2019

## Basic principle

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

Signal preprocessing (front end) 1

Speech recognition (search algorithm) 5

Acoustic models 4

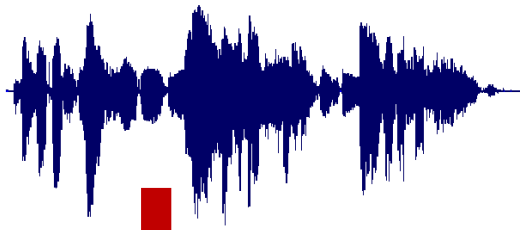
Pronunciation model (lexicon) 2

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

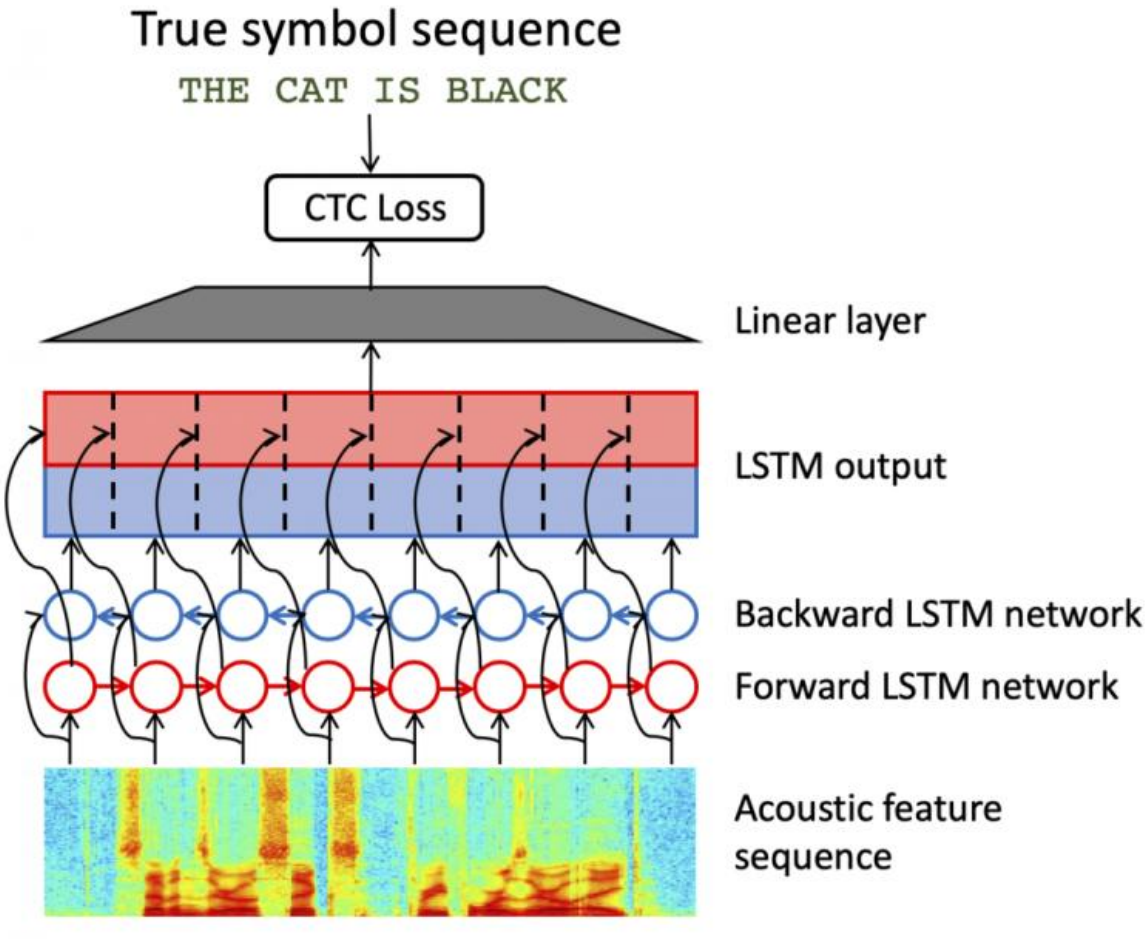
The lexicon specifies per word the corresponding sequence of speech units. There may be pronunciation variants (more variants per word)

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)

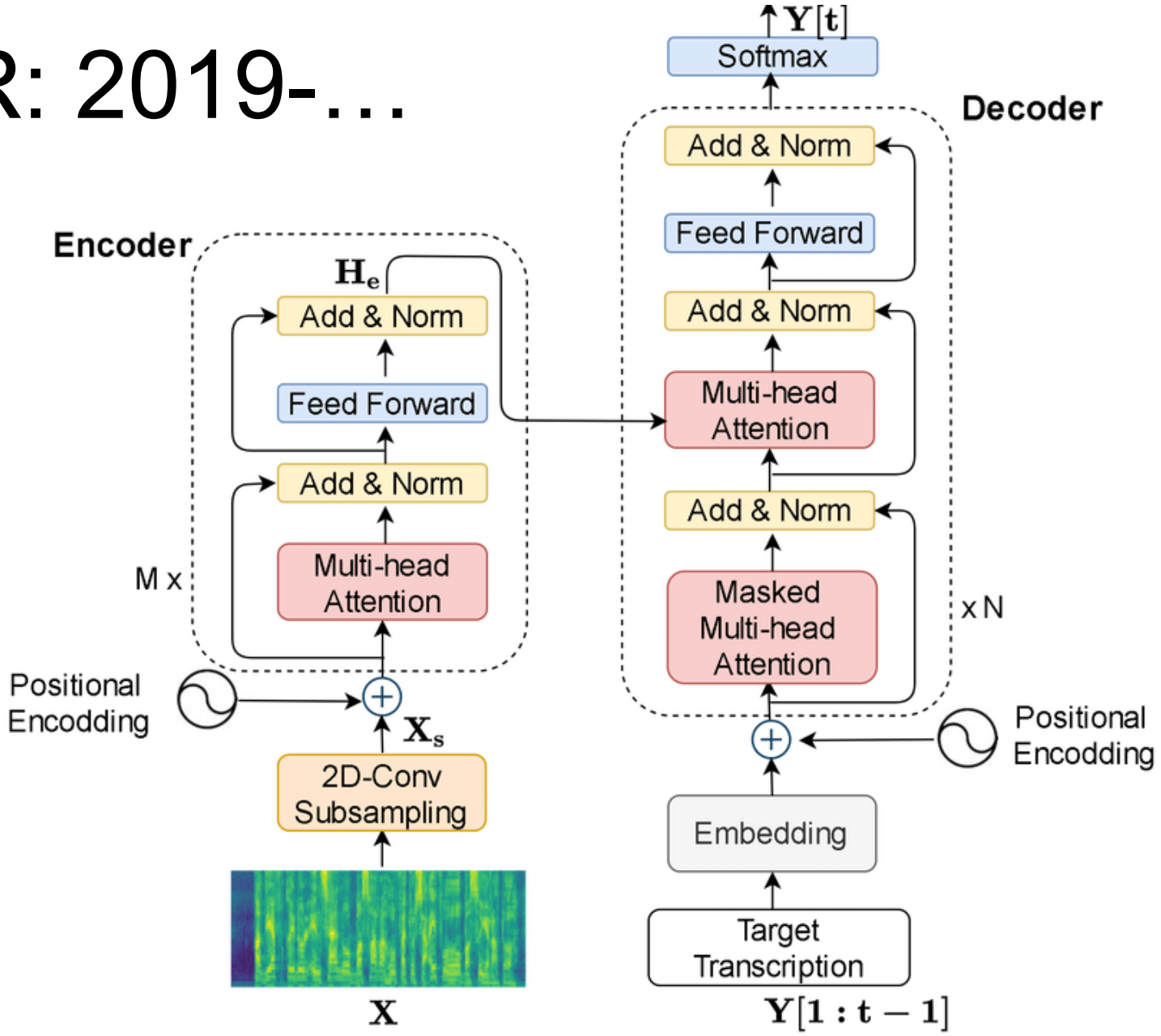
# ASR: architecture 2013-…



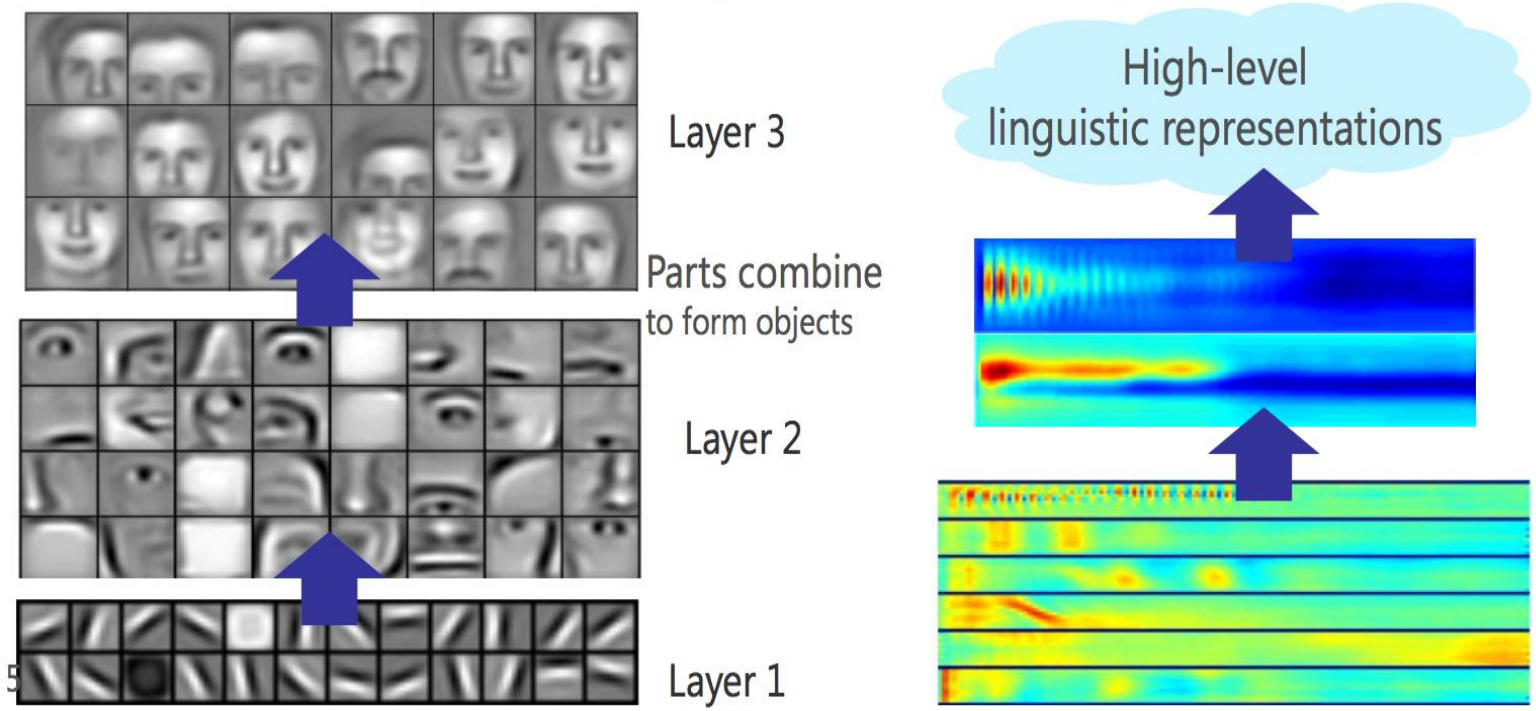

**Since 2013, ASR exploits deep learning (RNN, LSTM, biLSTM, Transformers)**

**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

# 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.

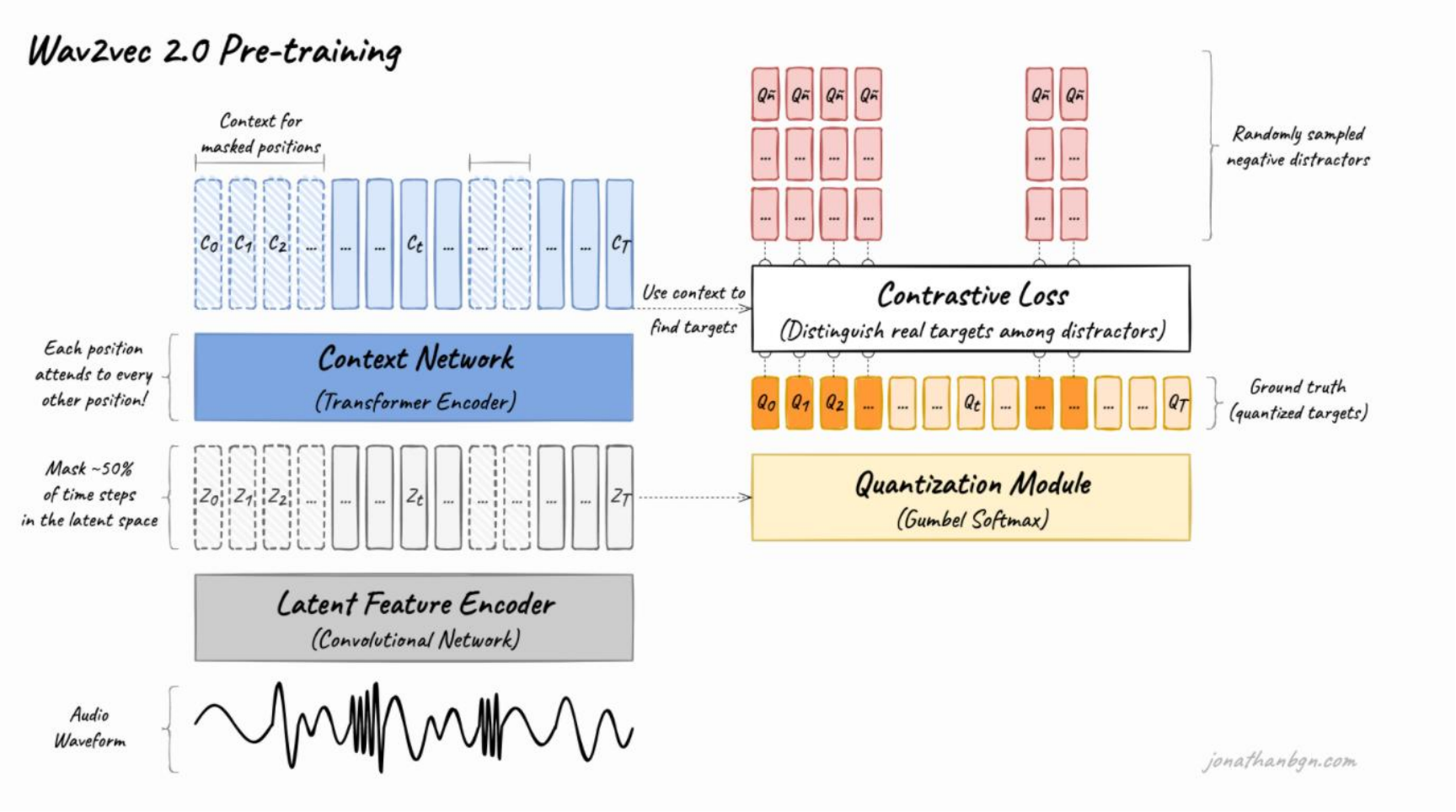
- In wav2vec 2.0's original paper (Baevski et al, 2020), the authors showed that fine-tuning the model on only **one hour of labeled speech data** could beat the previous state-of-the-art systems trained on 100 times more labeled data.

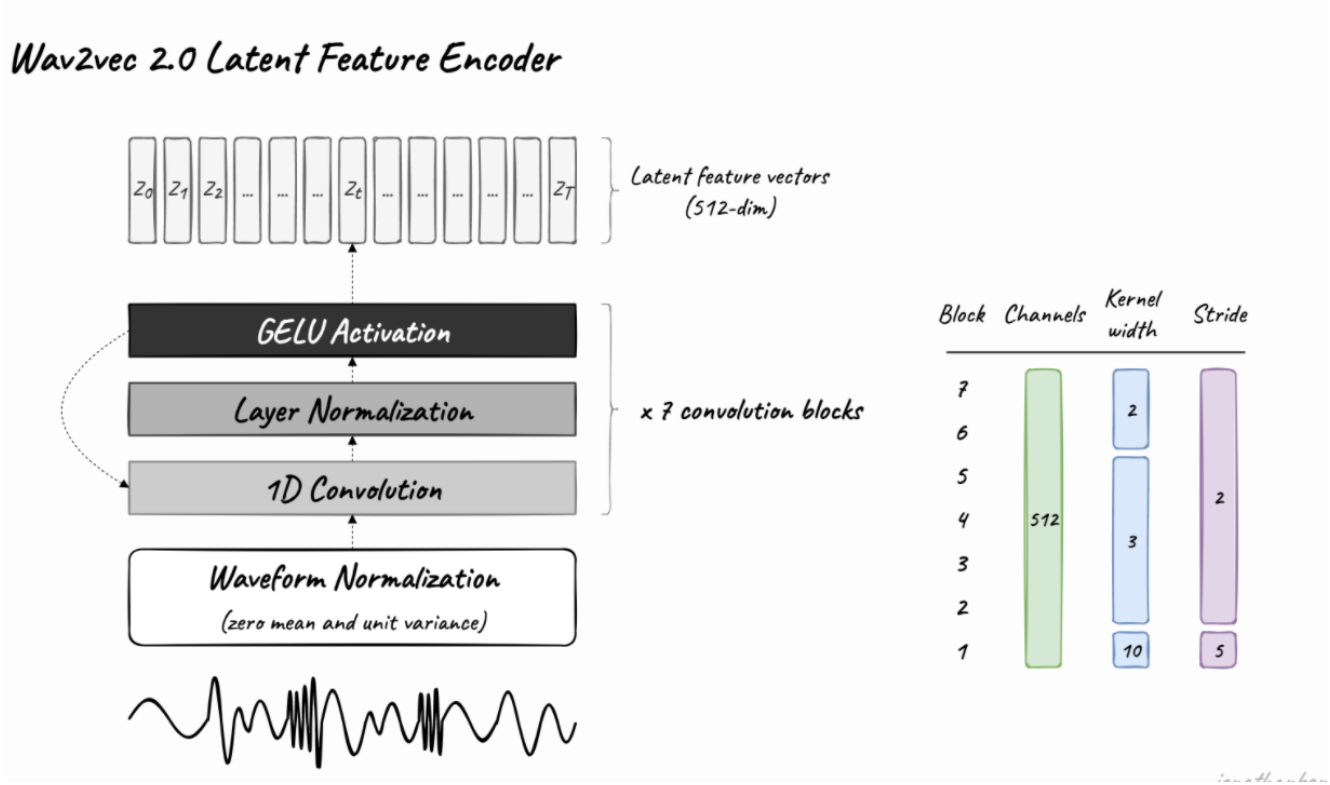
# Deep Neural Networks for ASR (S2T)

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**

# Global architecture
# Wav2vec2.0

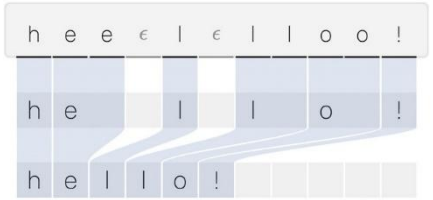


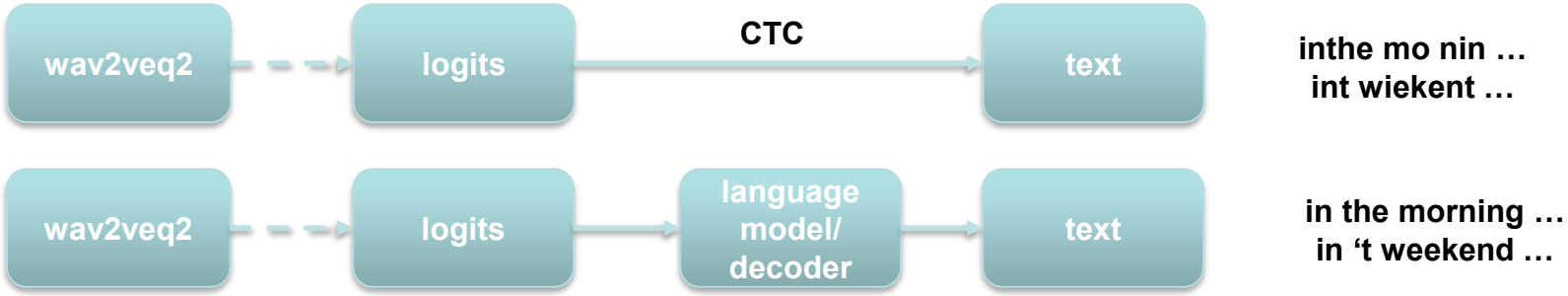



The feature encoder's job is to reduce the dimensionality of the audio data, converting the raw waveform into a sequence of feature vectors $Z_0$, $Z_1$, $Z_2$, ..., $Z_T$ each 20 milliseconds. Its architecture is simple: a 7-layer convolutional neural network (single-dimensional) with 512 channels at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

# Logit matrix/tensor

- 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

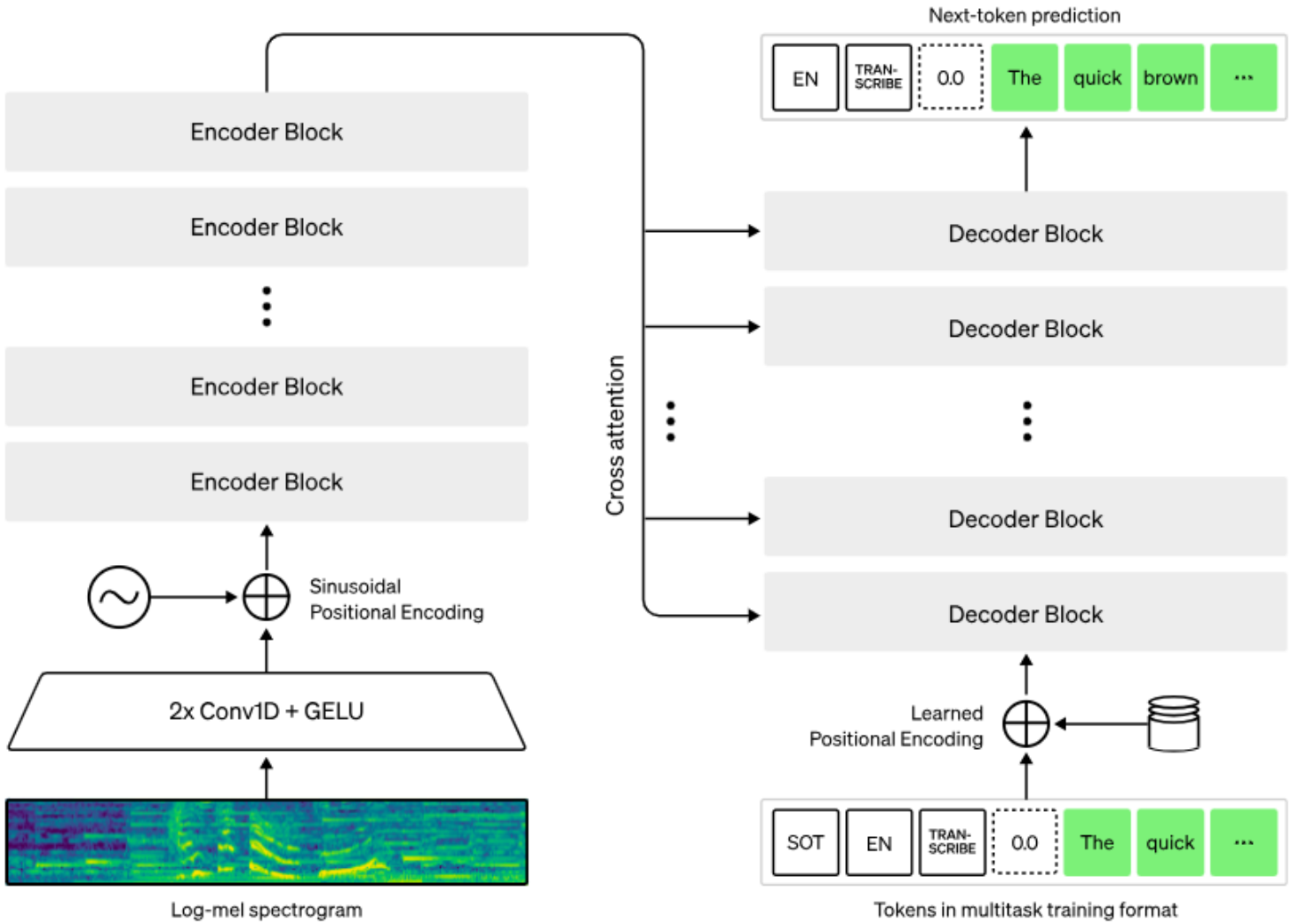


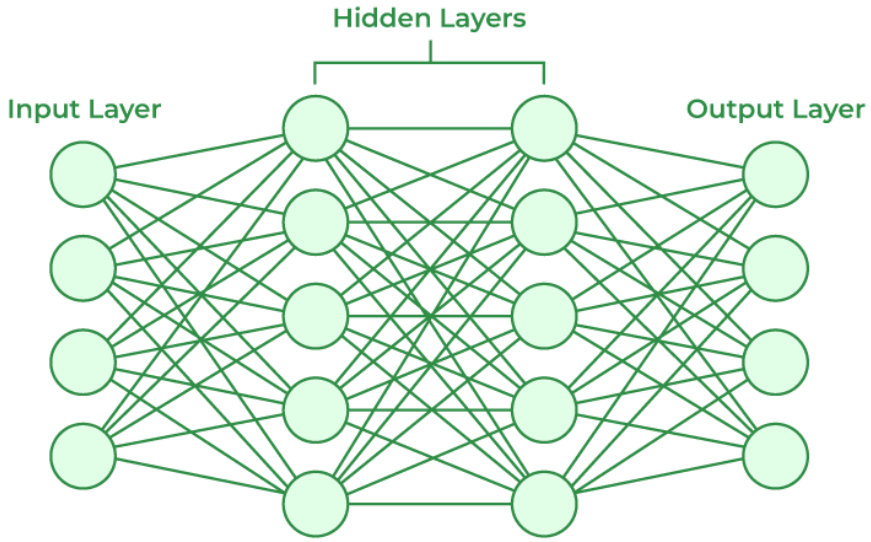



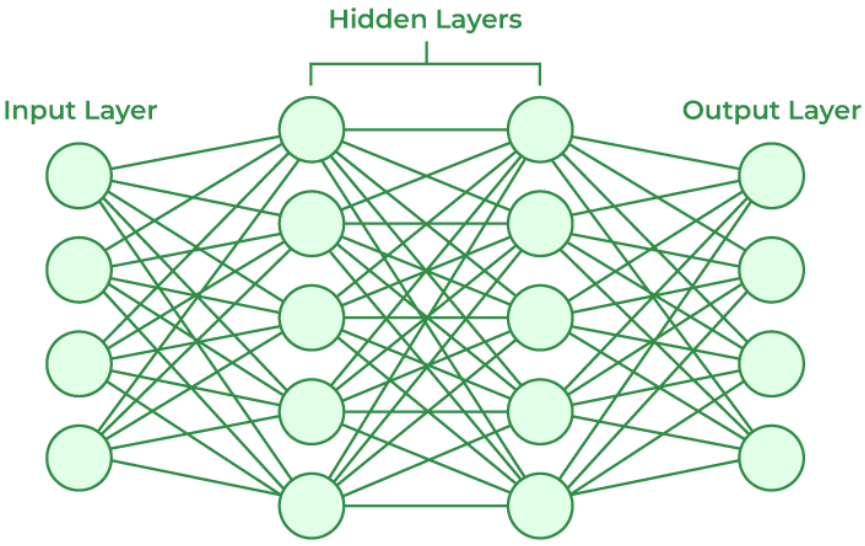
# Prompting in Whisper
# &
# the impact of a language model in deep networks

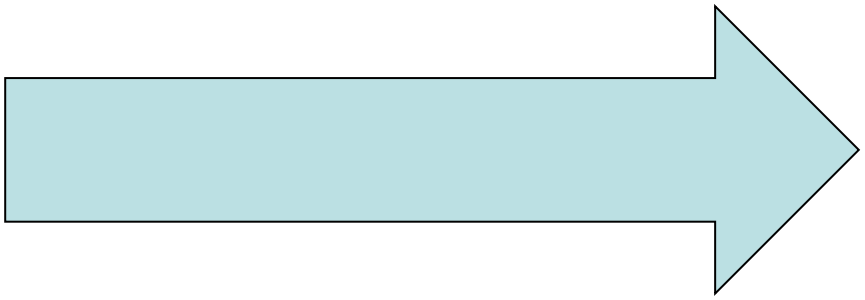
Louis ten Bosch
Graz, Nov – Dec 2025

# Prompting in Whisper: encoder-decoder

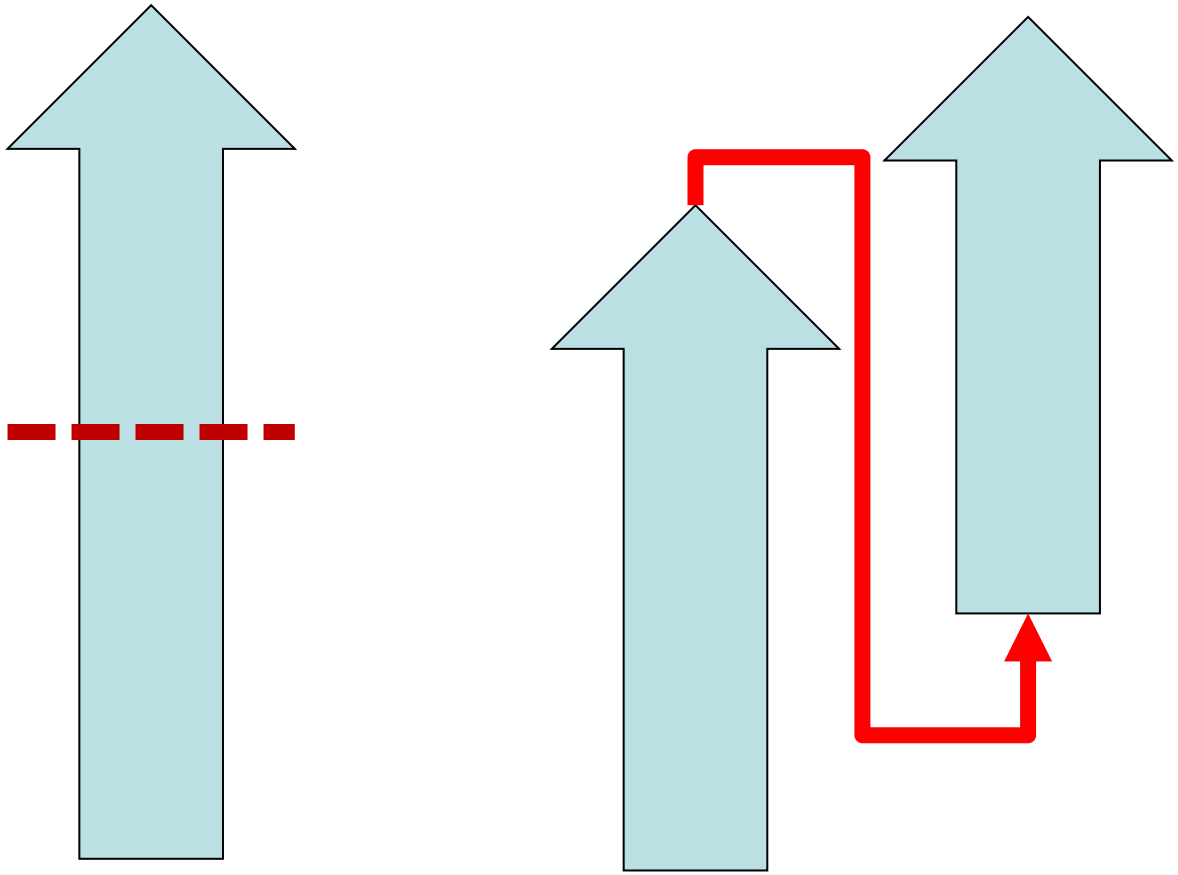


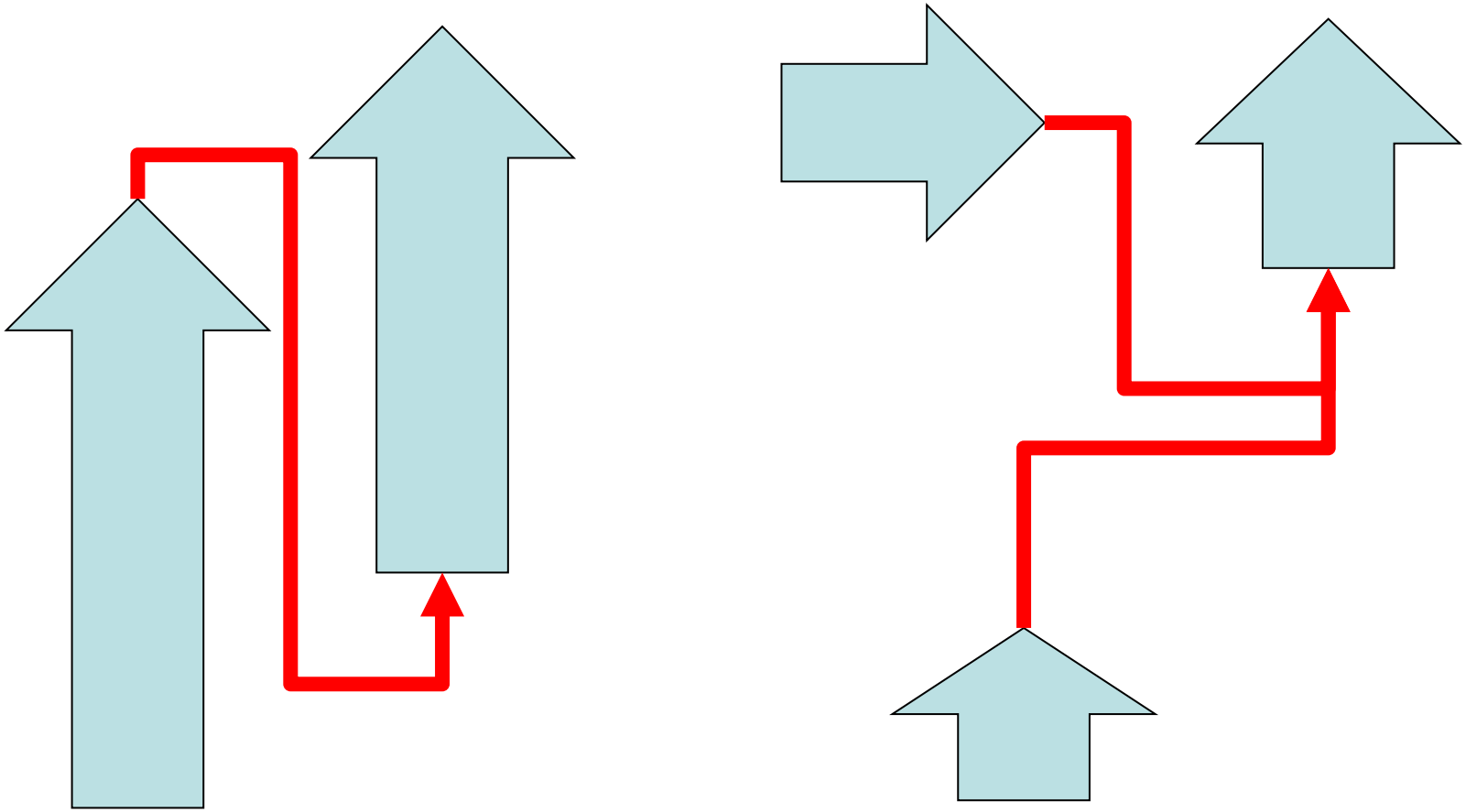

Hidden Layers
Input Layer
Output Layer



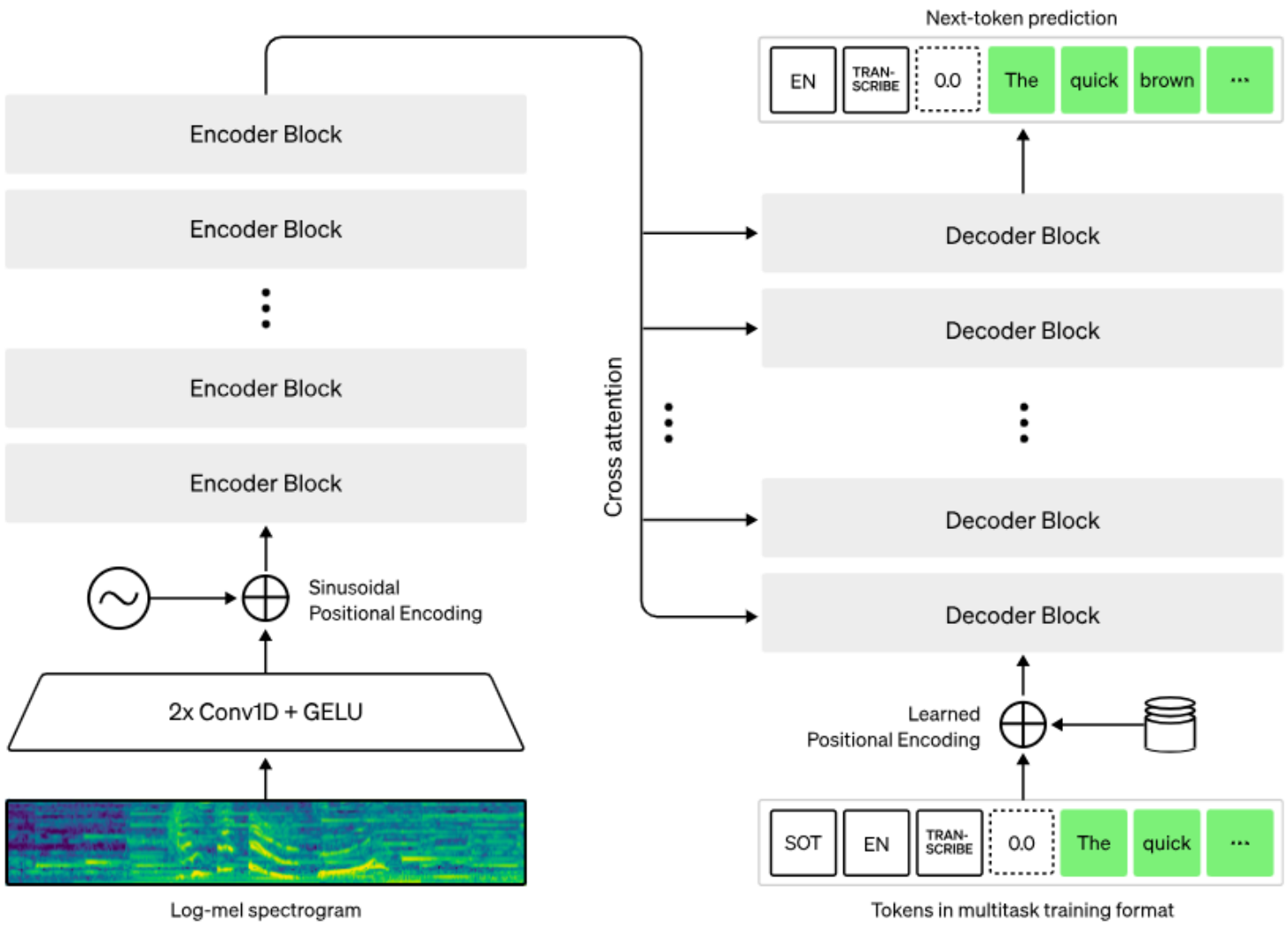



**Information flow**



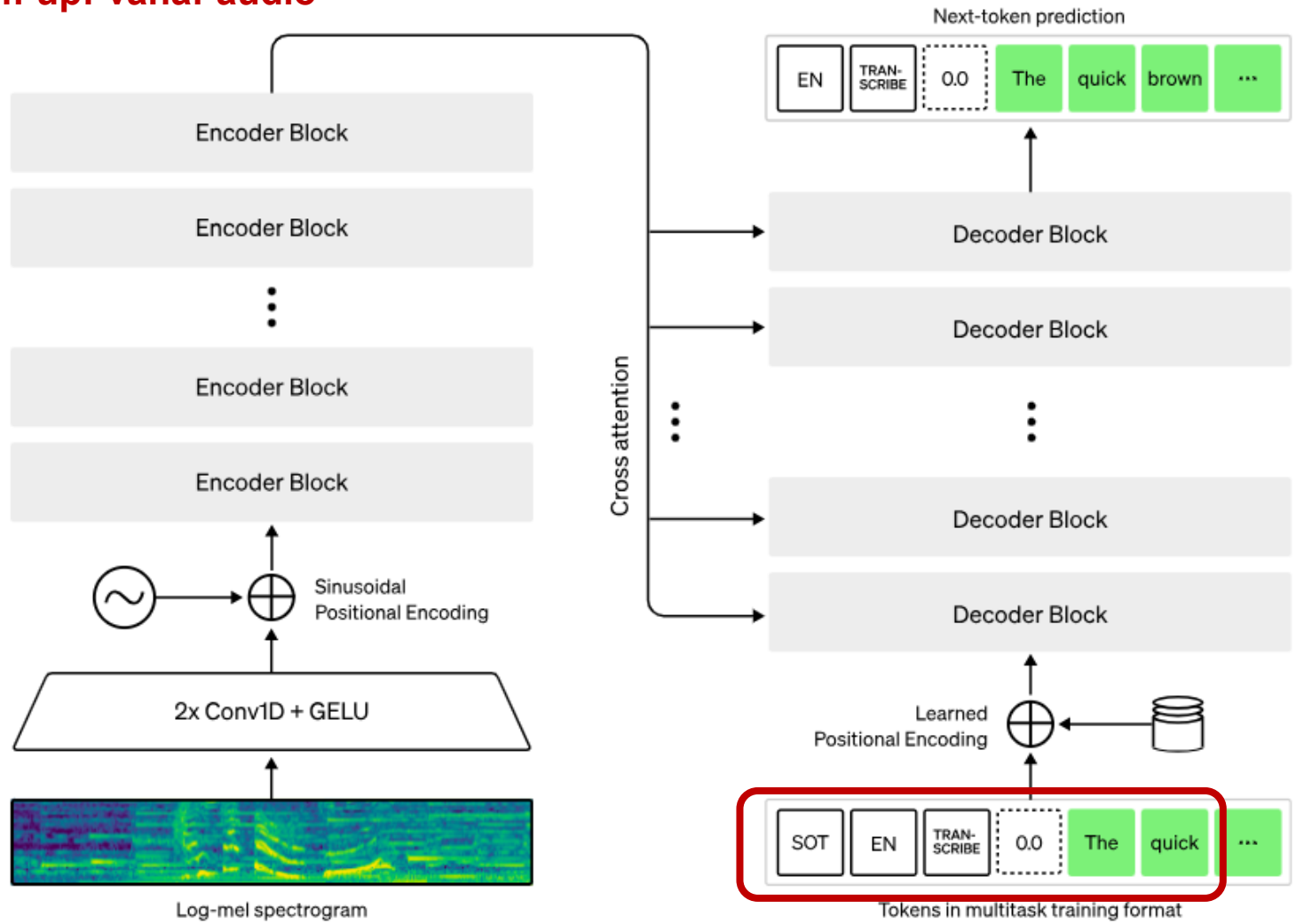


# Whisper: encoder-decoder



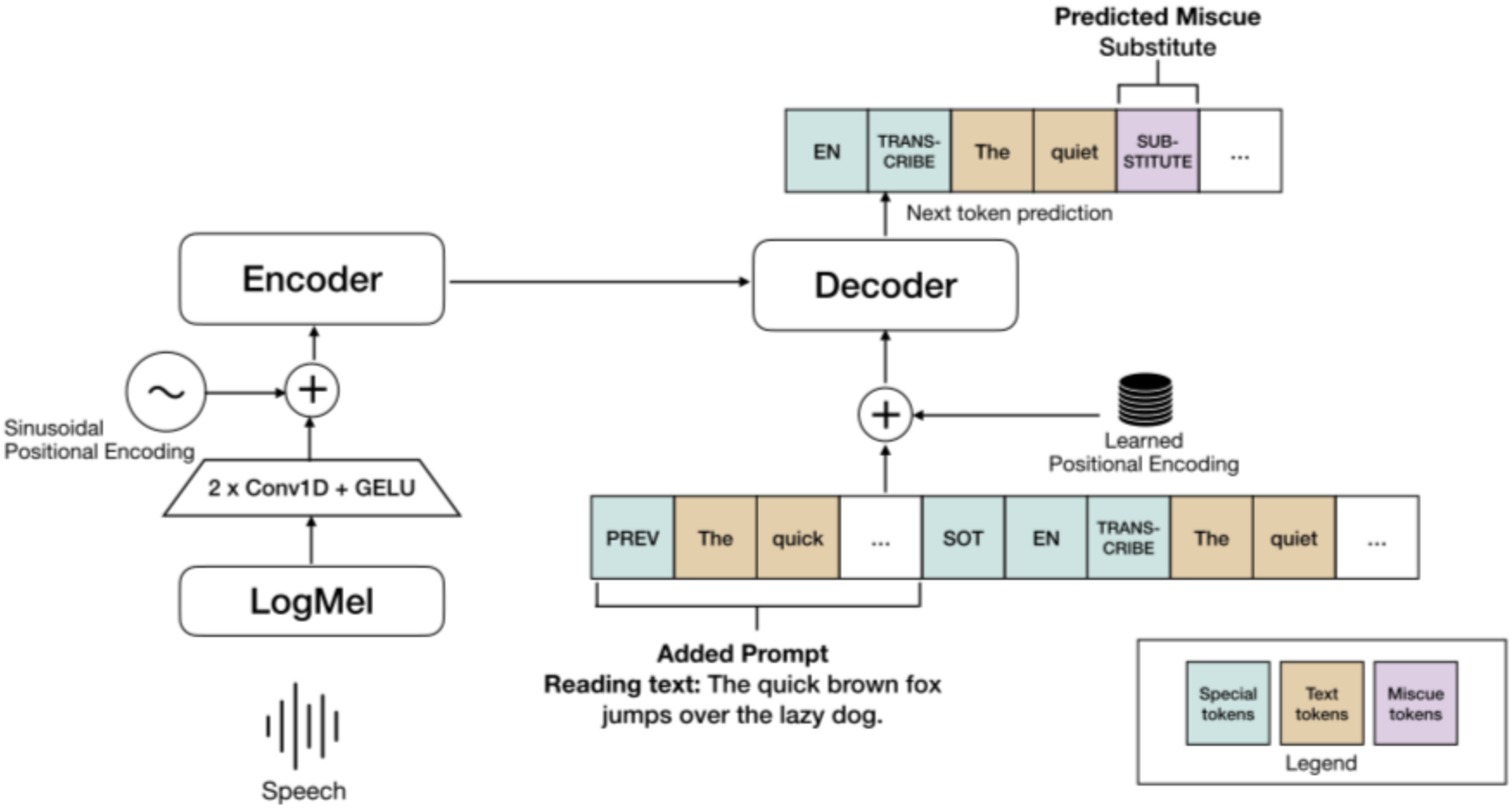
# Whisper: encoder-decoder

**Bottom-up: vanaf audio**

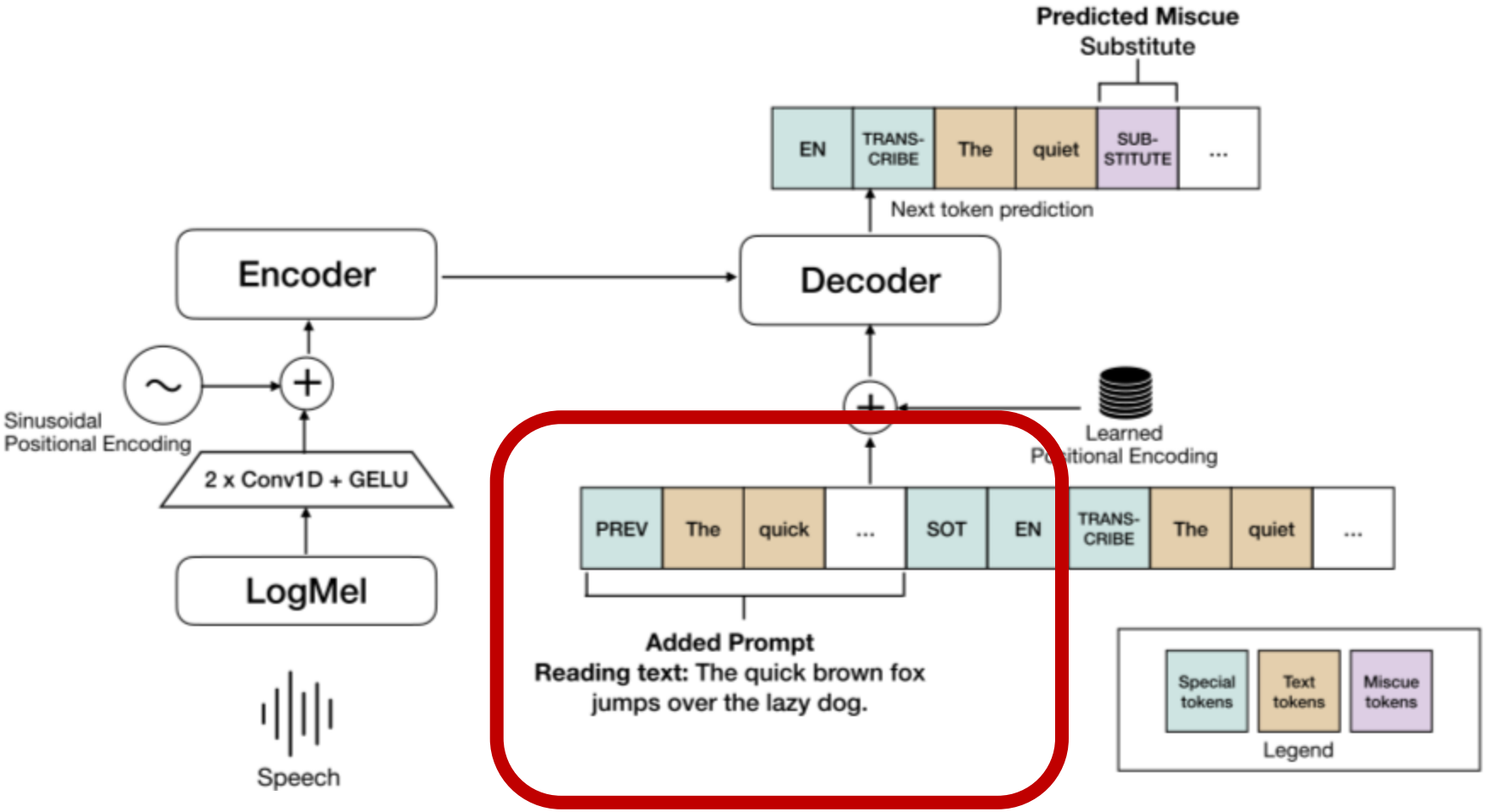



**Precontext**

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)**

# Prompt tuning

- Many examples. See e.g. the Verdini paper in Github

- 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.

# 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