

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)

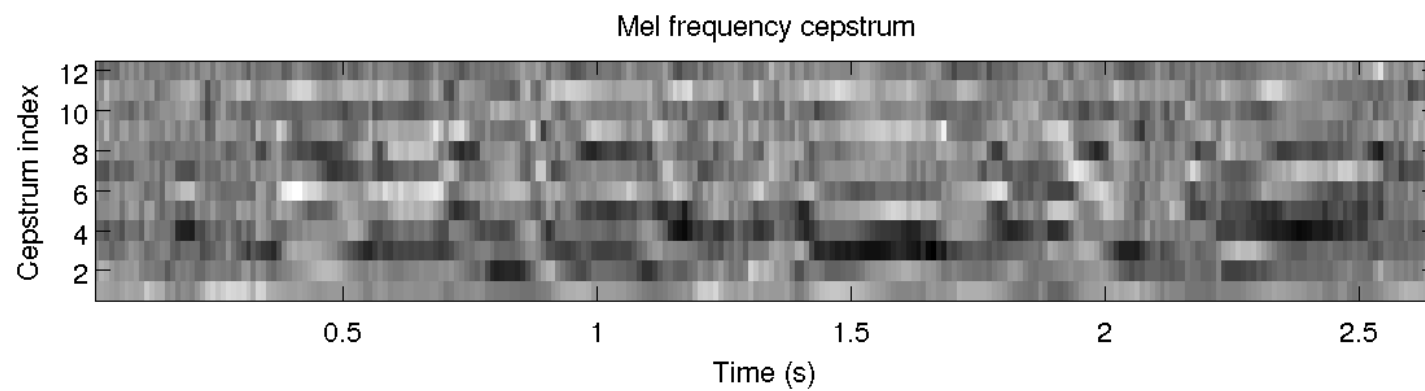
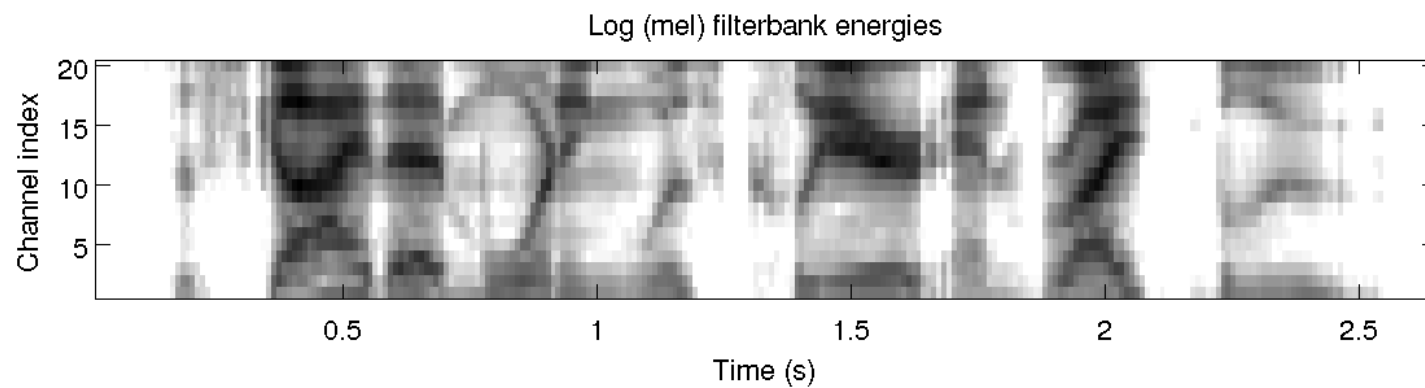
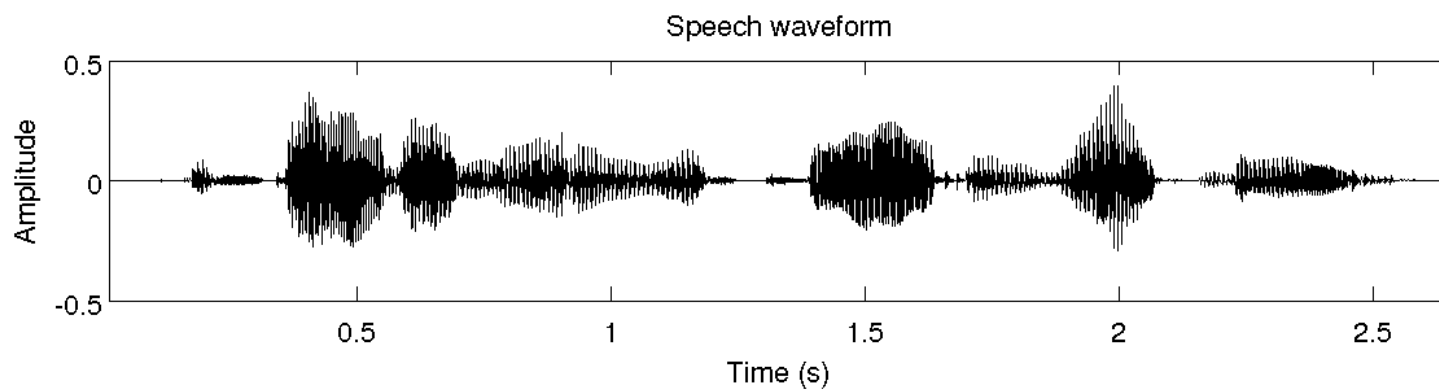
Vectorization → ideal for deep learning and neural networks

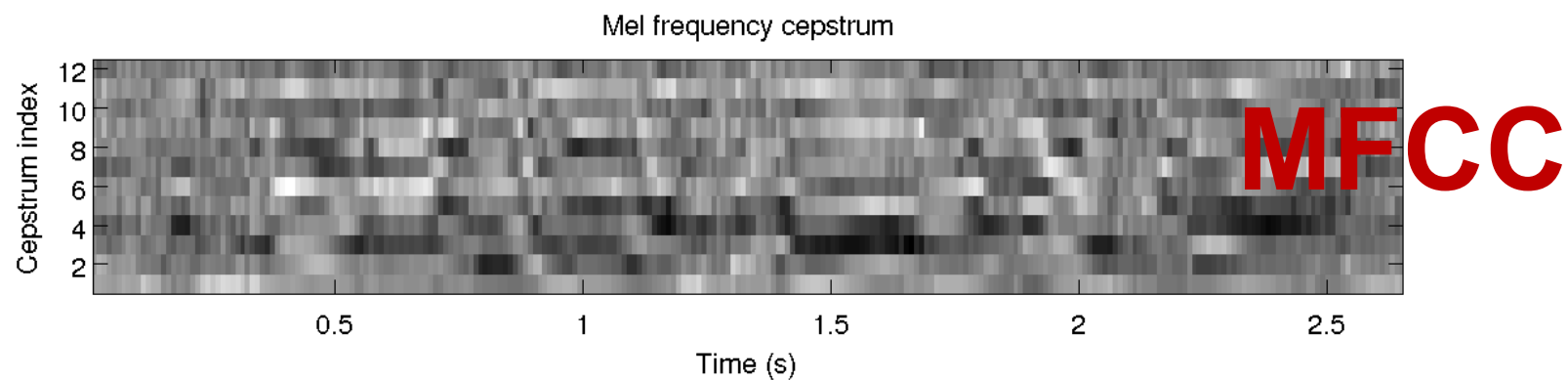
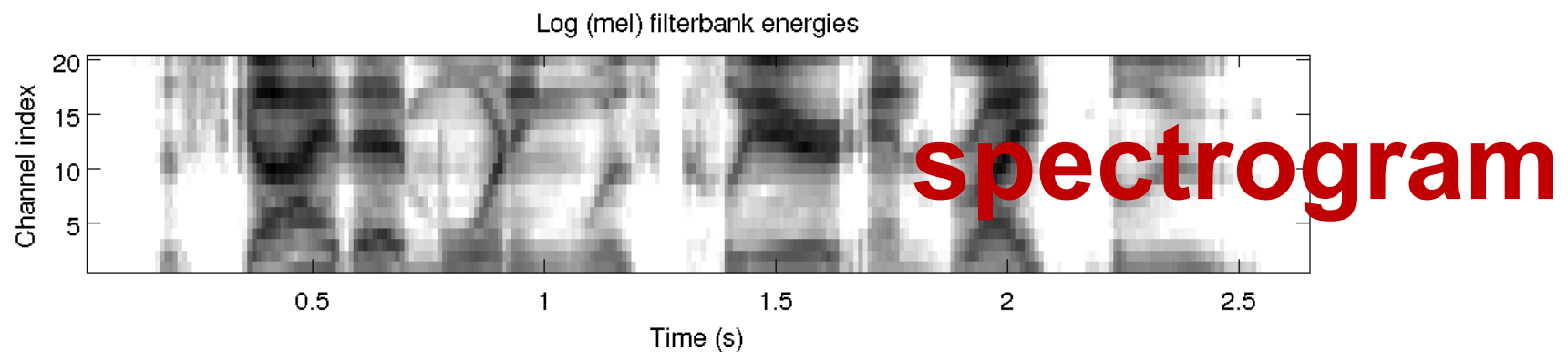
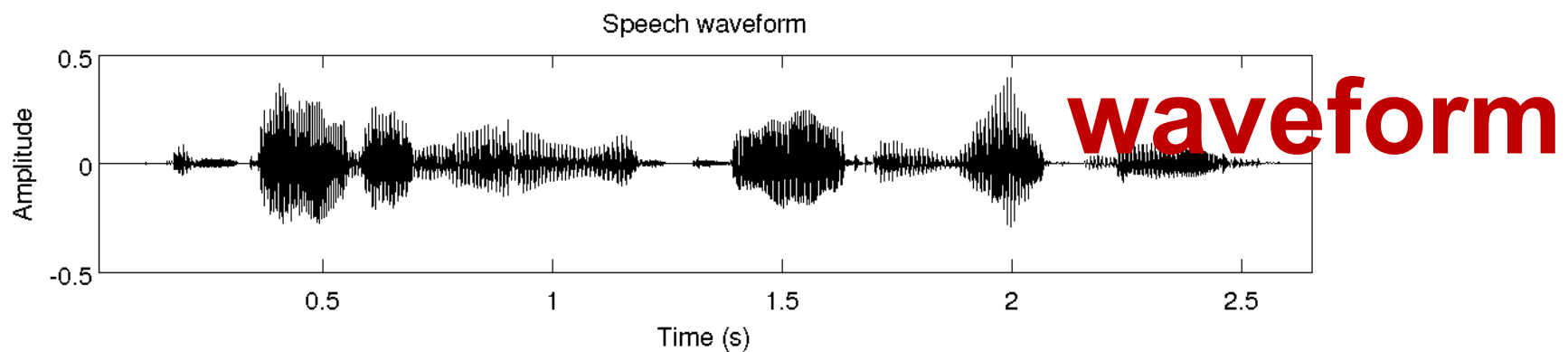
From audio to features

features are input for all
downstream modules

+

what's not in the features cannot
be classified later





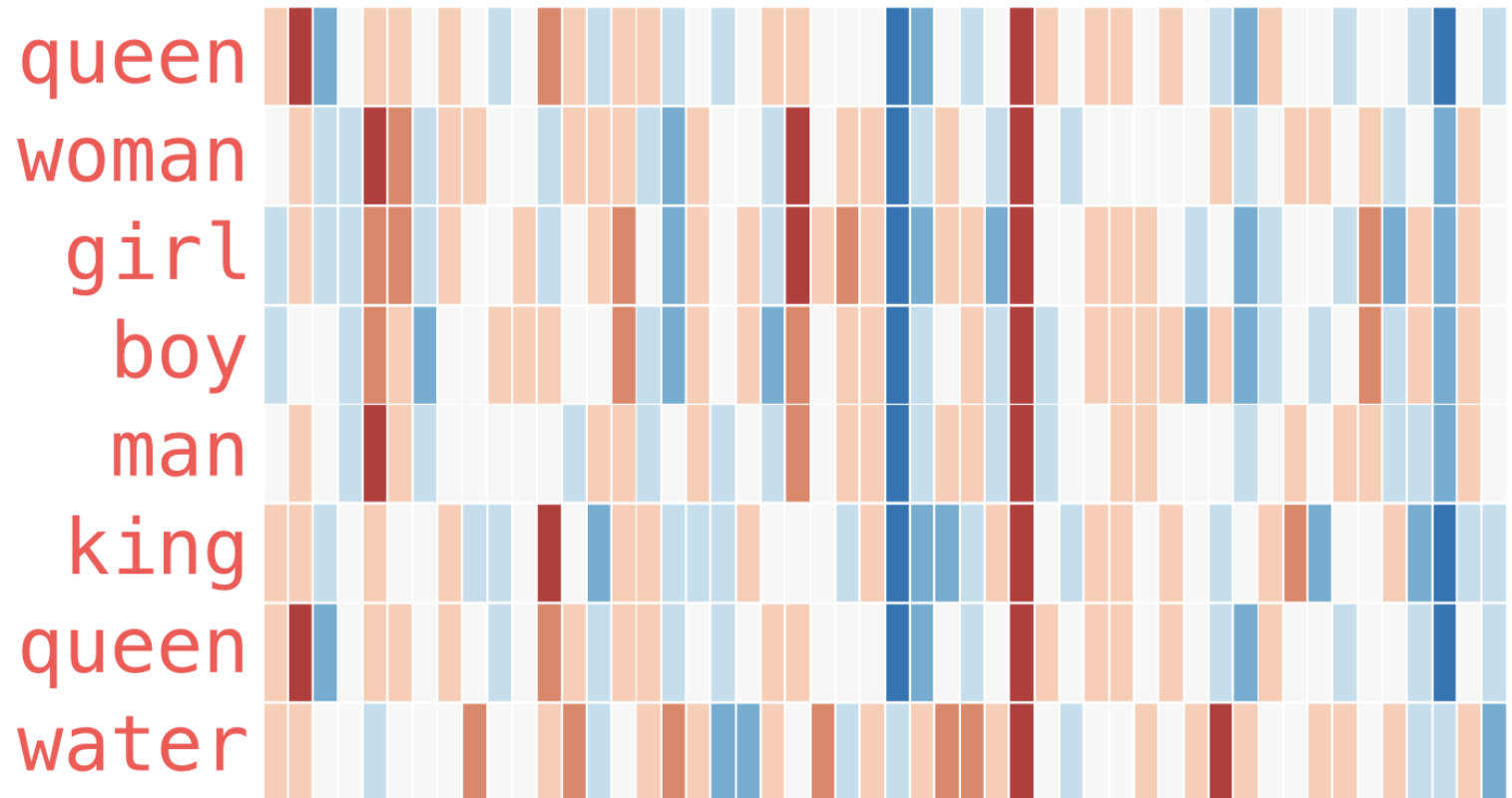
MFCC vectors

MFCC = Mel Frequency Cepstral Coefficients

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

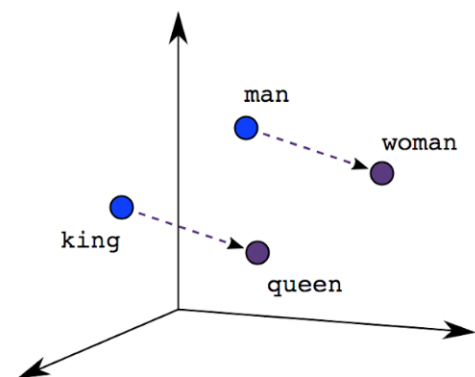
- <https://www.kaggle.com/ilyamich/mfcc-implementation-and-tutorial>
- https://pypi.org/project/python_speech_features/
- **Librosa python library**
<https://librosa.org/doc/latest/index.html>

Words: embeddings, word2vec

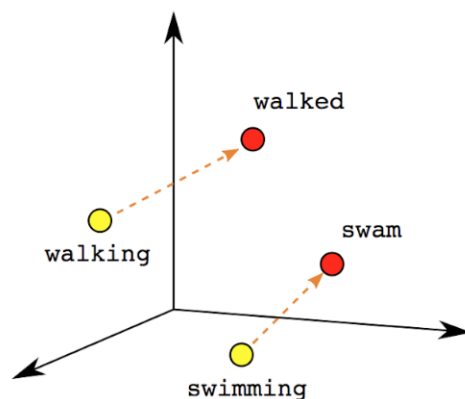


From: lajammar.github.io (retrieved sept. 2020)
See e.g. <https://radimrehurek.com/gensim/models/word2vec.html>

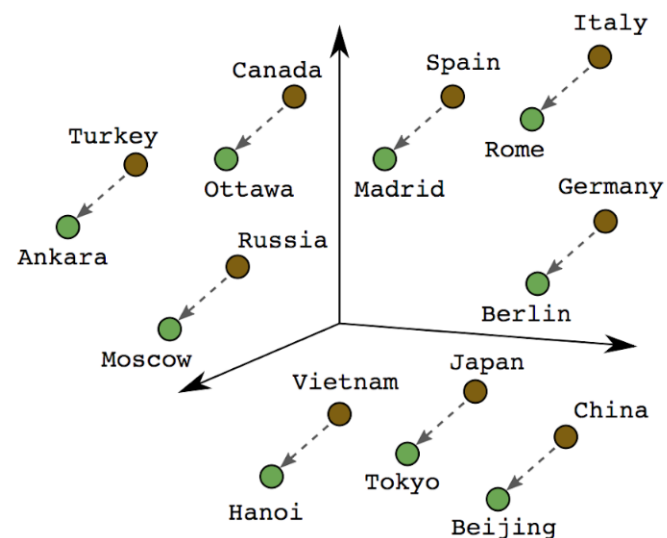
WORDS AS VECTORS (WORD EMBEDDINGS)



Male-Female



Verb Tense



Country-Capital

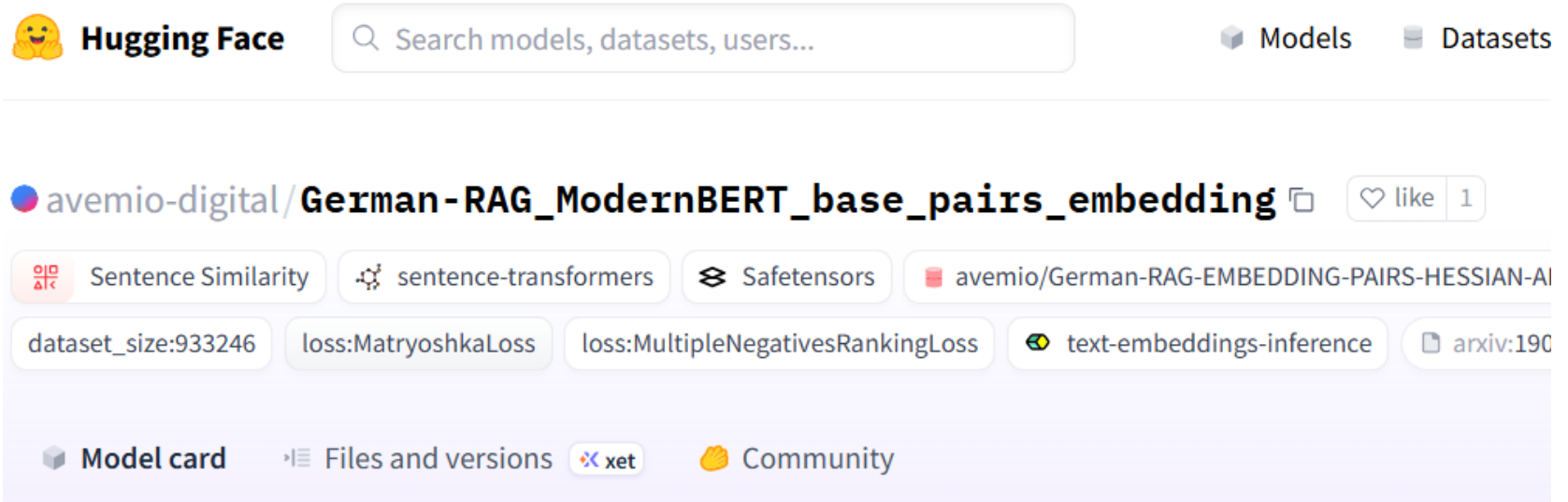
Dimension??

100, 300 (word2vec)

768 (BERT, ...)

1024 (Wav2vec2.0)

Embedding datasets are available for many languages

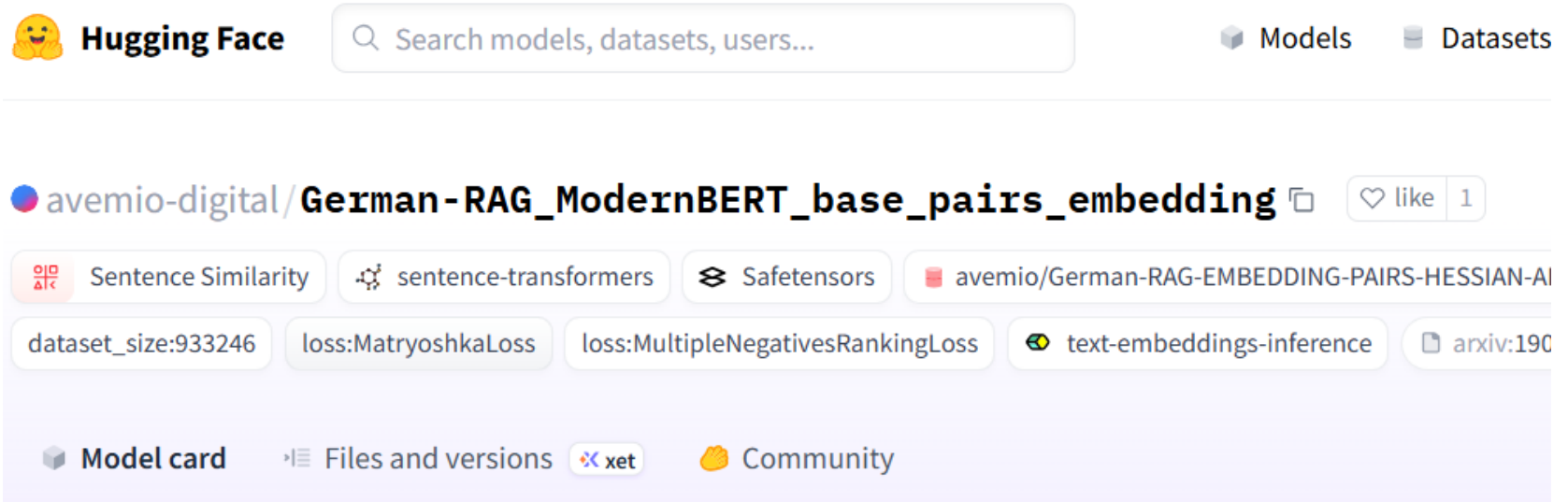


The screenshot shows the Hugging Face website interface. At the top, there is a search bar with the text "Search models, datasets, users...". To the left of the search bar is the Hugging Face logo and name. To the right are links for "Models" and "Datasets". Below the search bar, the dataset page for "avemio-digital/German-RAG_ModernBERT_base_pairs_embedding" is displayed. The page includes a "like" button with a count of 1. Below the dataset name, there are several tags: "Sentence Similarity", "sentence-transformers", "Safetensors", and "avemio/German-RAG-EMBEDDING-PAIRS-HESSIAN-AI". There are also metadata tags: "dataset_size:933246", "loss:MatryoshkaLoss", "loss:MultipleNegativesRankingLoss", "text-embeddings-inference", and "arxiv:190". At the bottom of the page, there are links for "Model card", "Files and versions", "xet", and "Community".

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



The screenshot shows the Hugging Face interface for a specific model. At the top, the 'Hugging Face' logo is on the left, and a search bar is in the center. To the right of the search bar are links for 'Models' and 'Datasets'. Below the header, the model name 'avemio-digital/German-RAG_ModernBERT_base_pairs_embedding' is displayed, followed by a 'like' button showing '1' like. A row of tags includes 'Sentence Similarity', 'sentence-transformers', 'Safetensors', and the repository name. Below these are more tags: 'dataset_size:933246', 'loss:MatryoshkaLoss', 'loss:MultipleNegativesRankingLoss', 'text-embeddings-inference', and 'arxiv:190'. At the bottom of the model card section, there are links for 'Model card', 'Files and versions', 'xet', and 'Community'.

ModernBERT_base_pairs_embedding

768

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.

word2vec and semantic relations

- king
- `[('king', 2.1073424255447017e-08), ('prince', 0.7643604295350113), ('queen', 0.802830437548828), ('crown', 0.9119298004877202), ('iii', 0.9217175647842433), ('princess', 0.9466528964247878), ('kings', 0.957123880812052), ('iv', 0.9601786784817395), ('reign', 0.9630452963421307)]`
- blue
- **Distances: cosine distance**
`[('blue', 1.4901161193847656e-08), ('white', 0.6977357395612396), ('red', 0.7271670715802425), ('black', 0.7588183017536706), ('yellow', 0.7704220726959191), ('green', 0.7964160430312023), ('purple', 0.800371671574374), ('pink', 0.8372136873739938), ('orange', 0.8418682151472597)]`

Architectures

Classical architecture

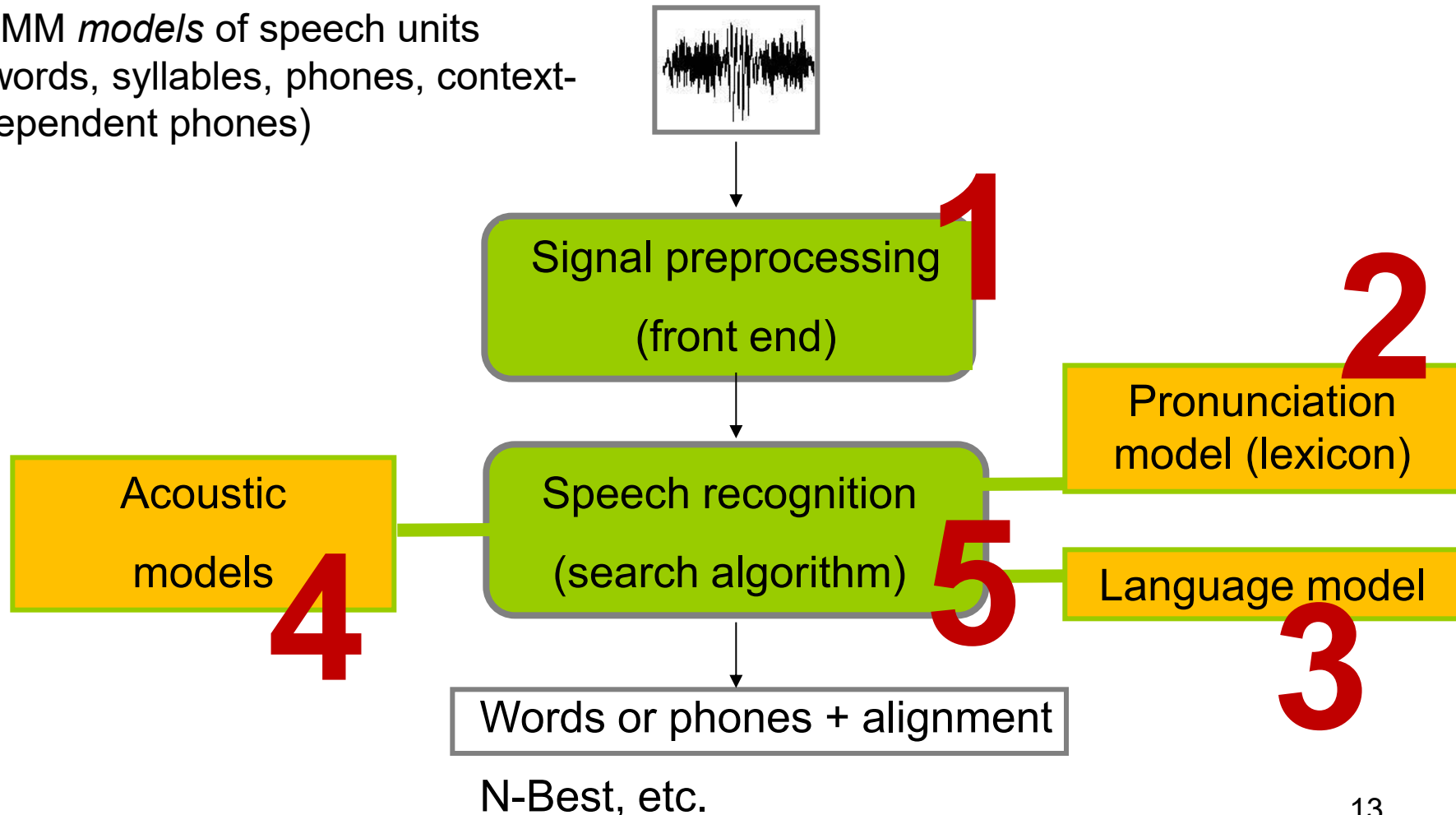
Wave2Vec2.0-like architecture

Whisper-like architecture

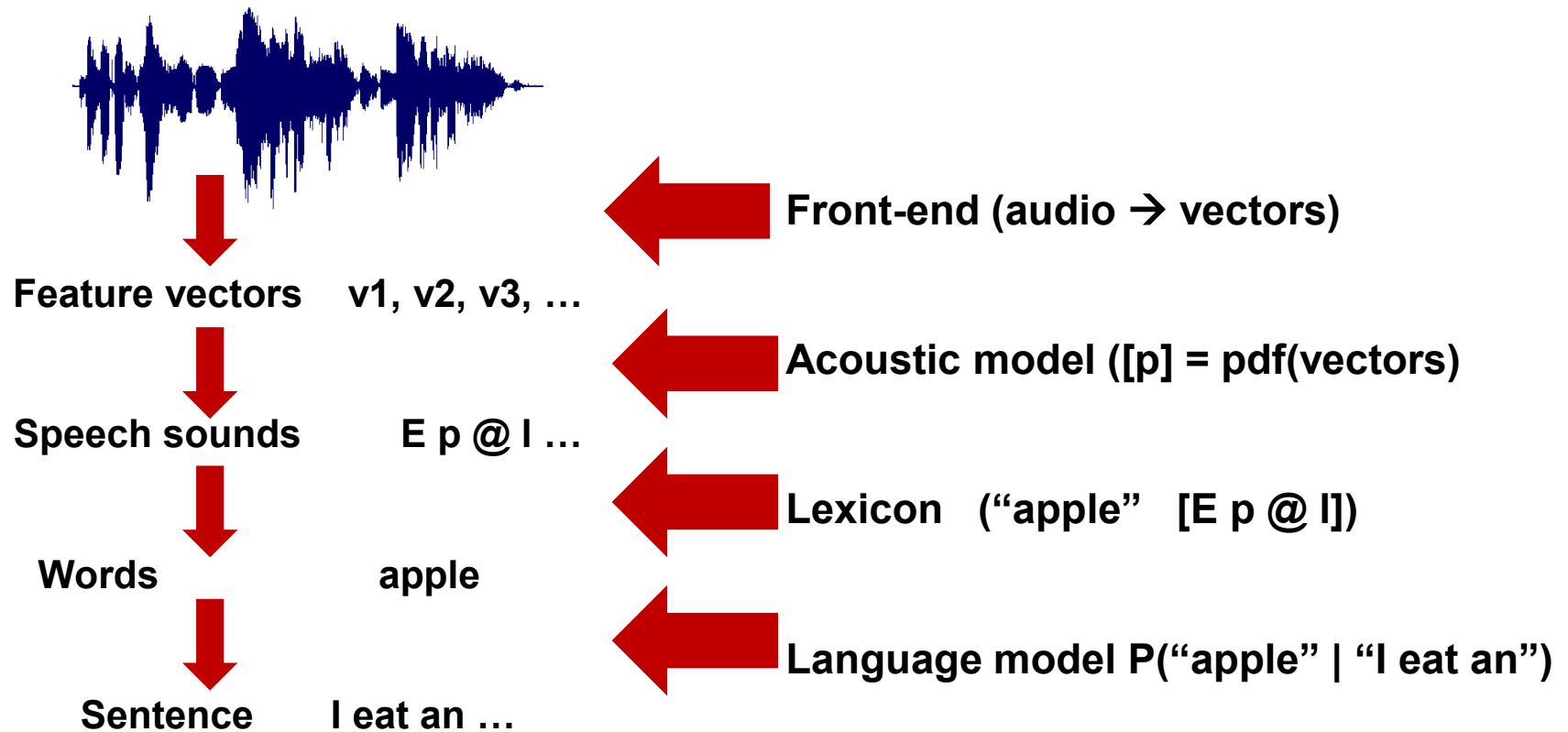
ASR: architecture 1980-2019

Basic principle

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



ASR components in the modular approach



ASR components in the modular approach

The **front-end**: extracting features from audio

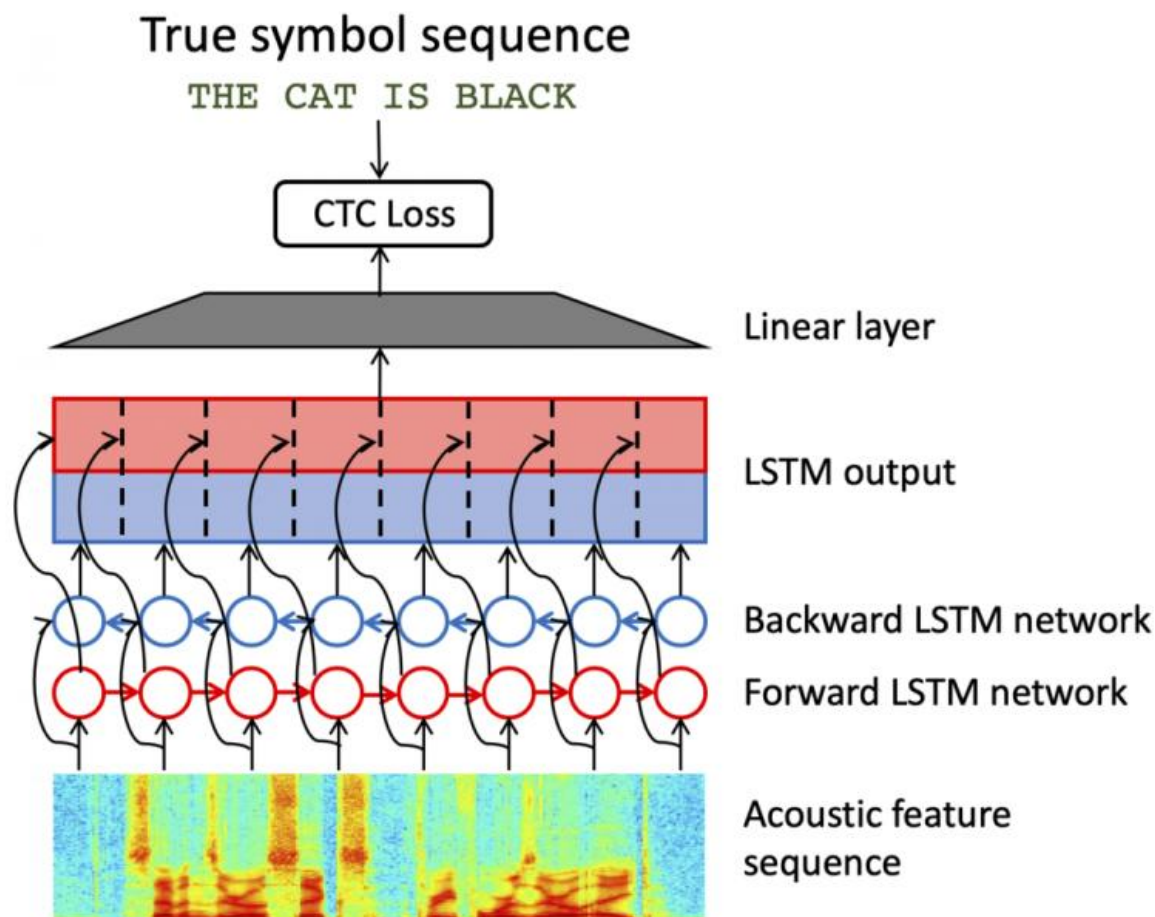
The **lexicon** specifies per word the corresponding sequence of speech units. There may be pronunciation variants (more variants per word)

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)

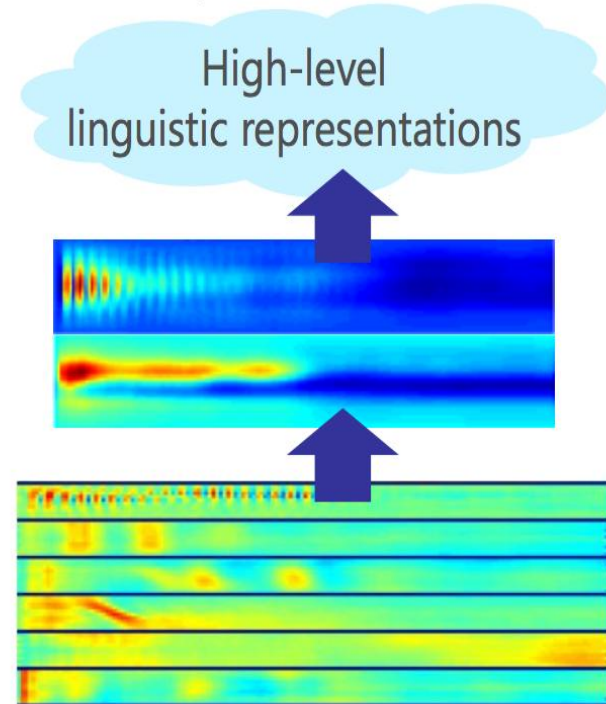
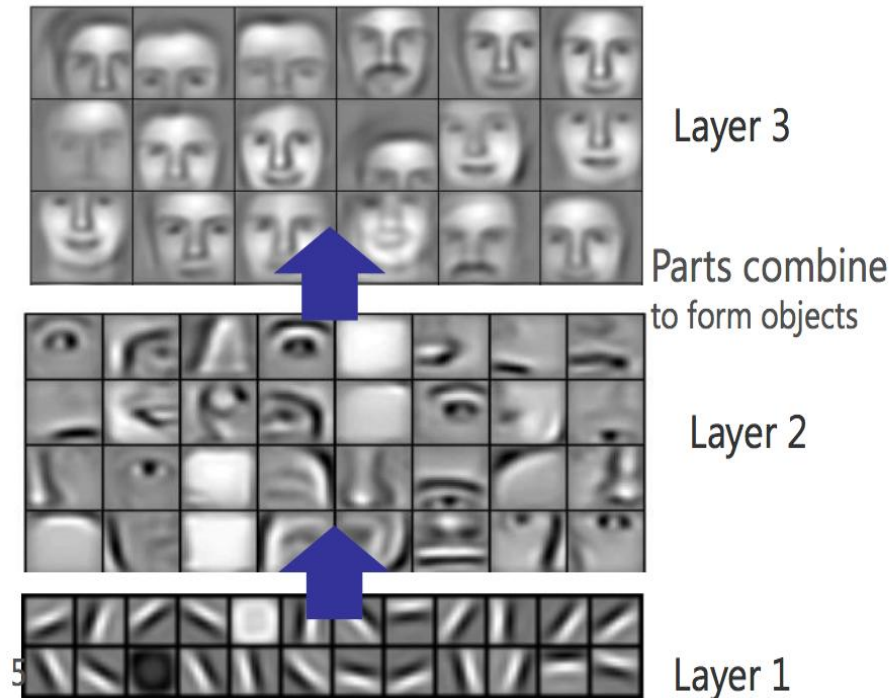
ASR: architectures 2013-...



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

Successive model layers learn deeper intermediate representations



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

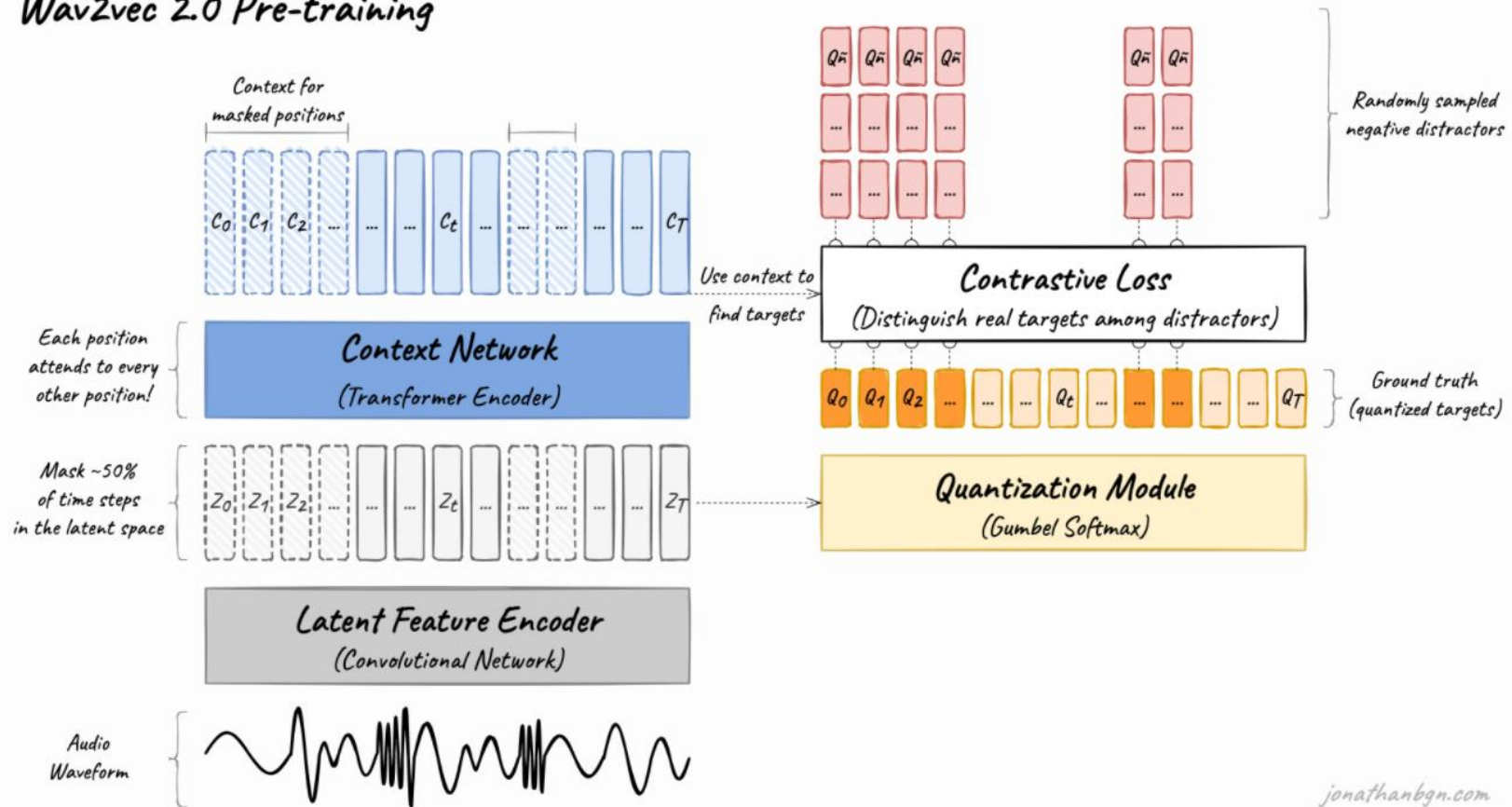
Lee, Largman, Pham & Ng, NIPS 2009

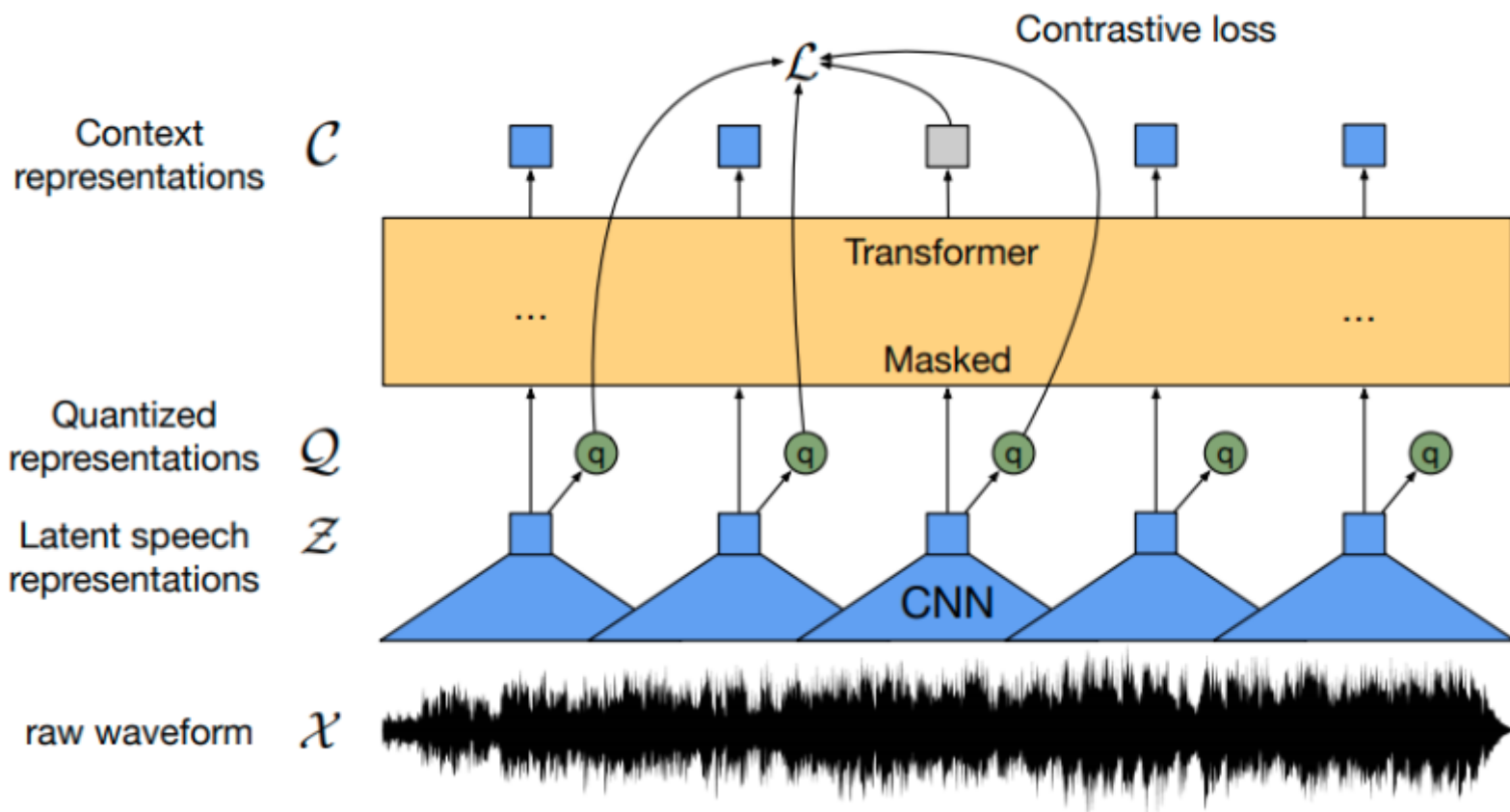
Wav2vec2.0

- 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.

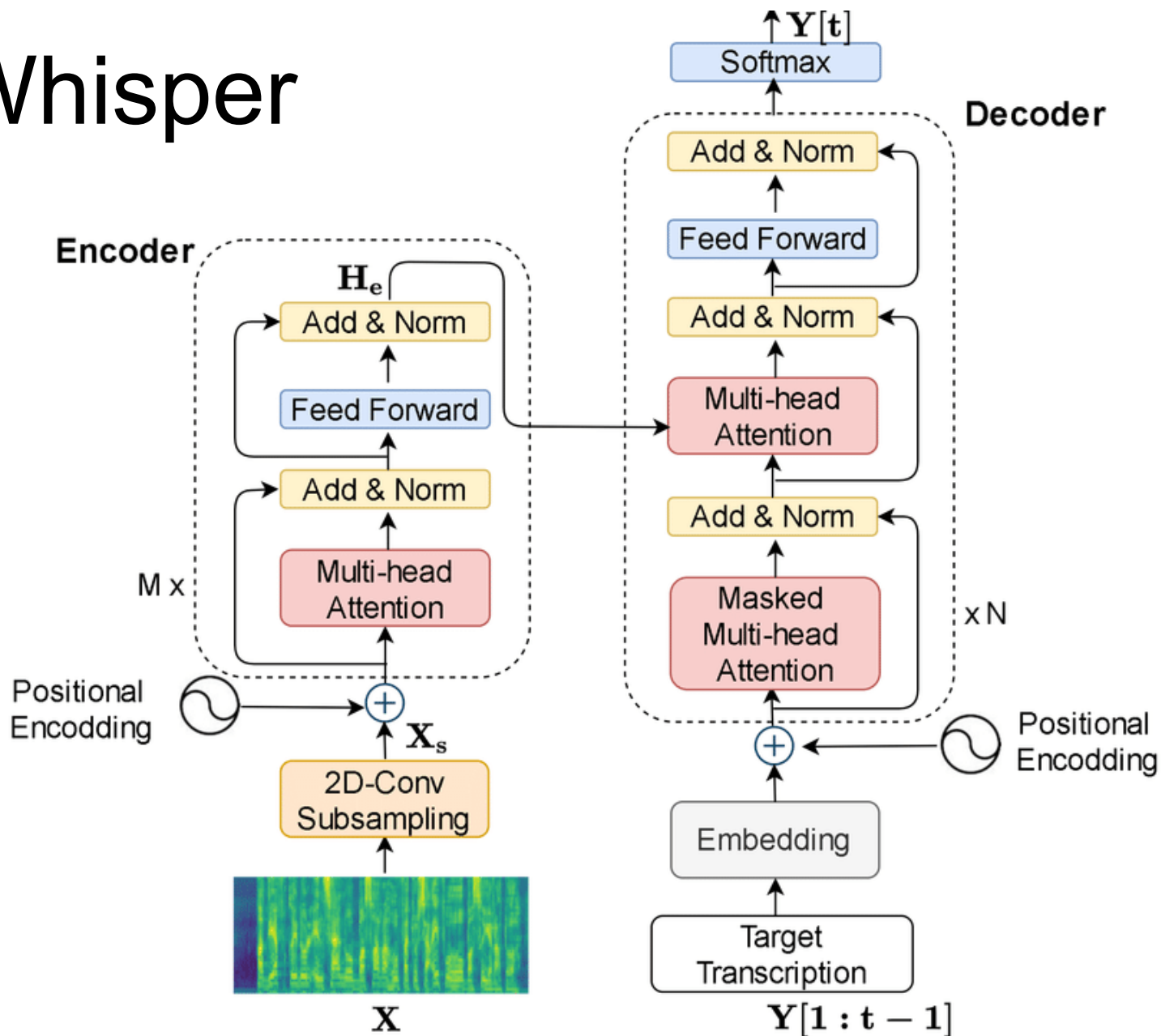
Wav2vec2.0

Wav2vec 2.0 Pre-training





Whisper



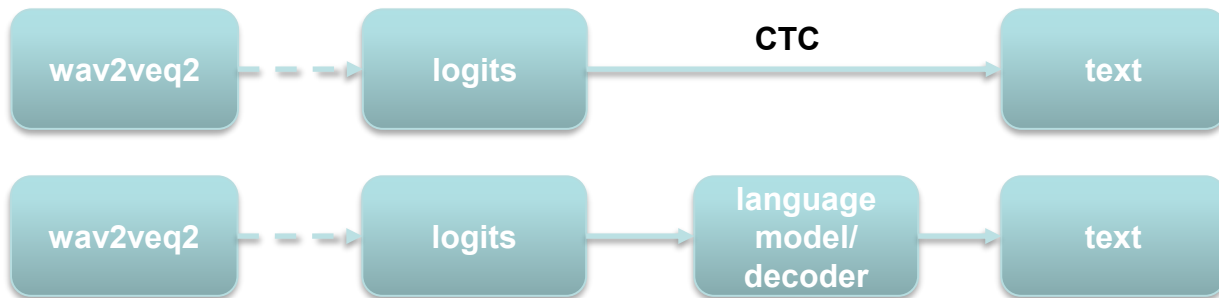
Deep Neural Networks for ASR (S2T)

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**

Logit matrix/tensor (in Wav2vec2)

- 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



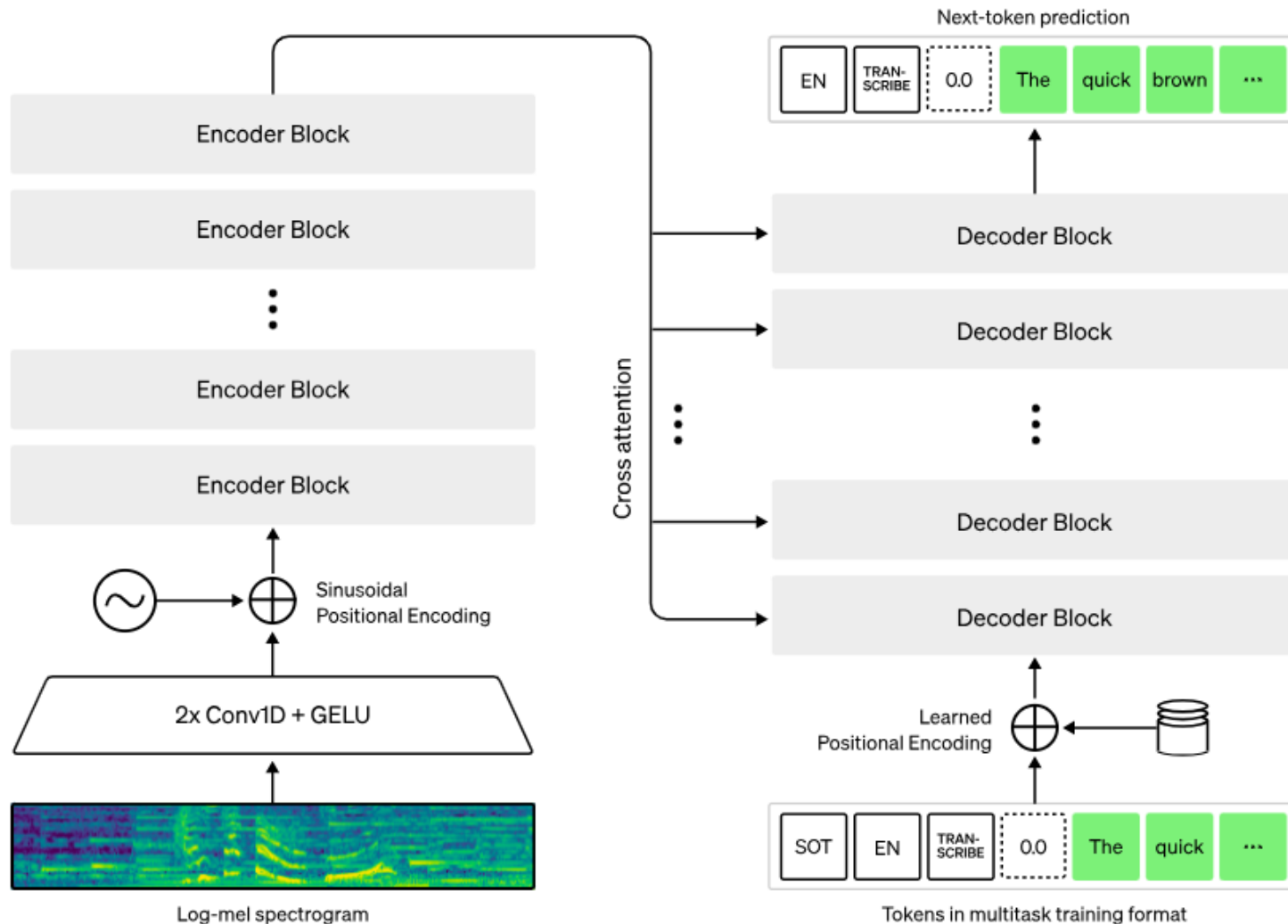
inthe mo nin ...
int wiekent ...

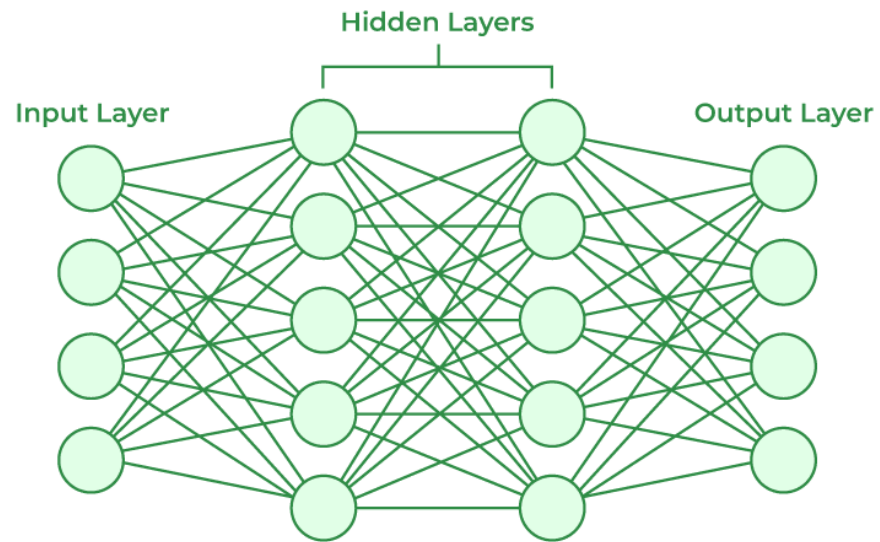
in the morning ...
in 't weekend ...

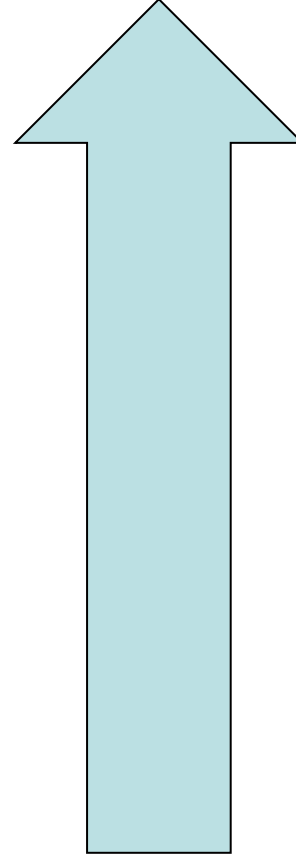
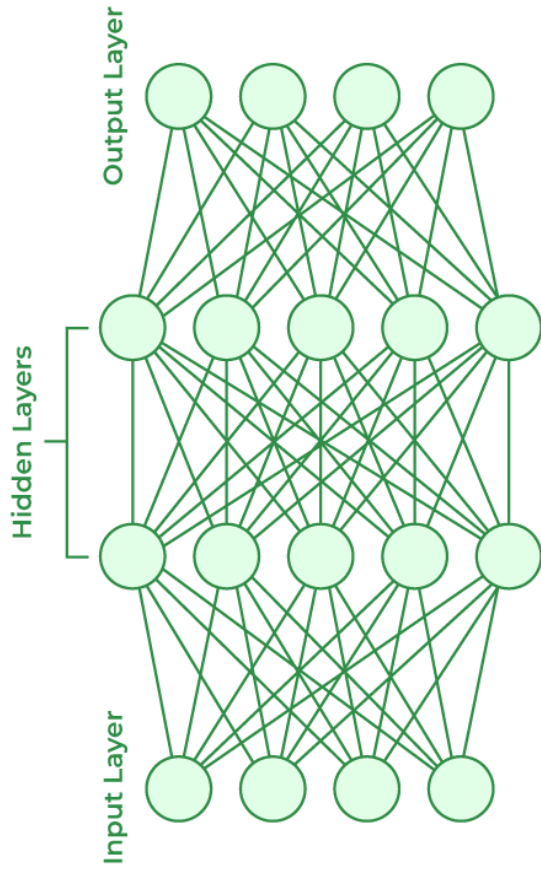
Prompting in Whisper & the impact of a language model in deep networks

Louis ten Bosch
Graz, Nov – Dec 2025

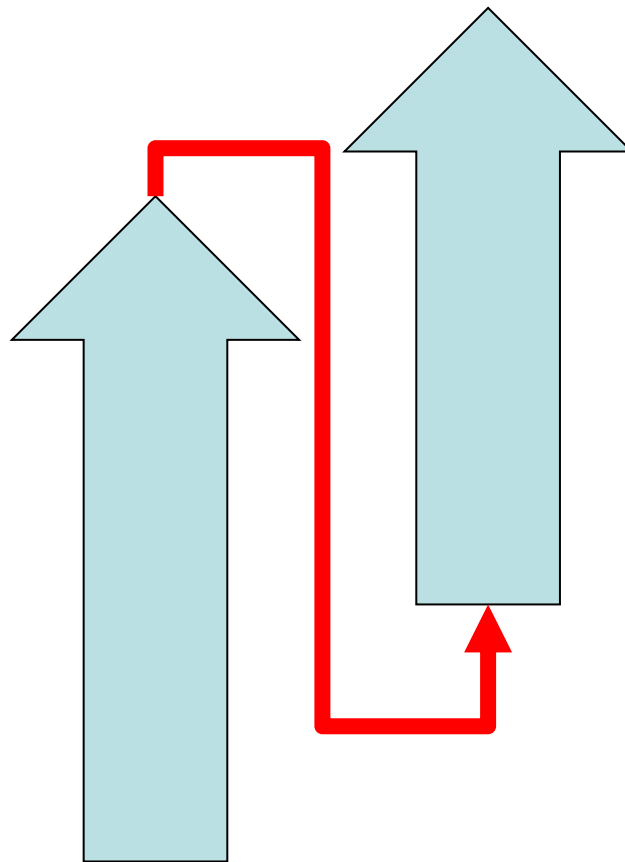
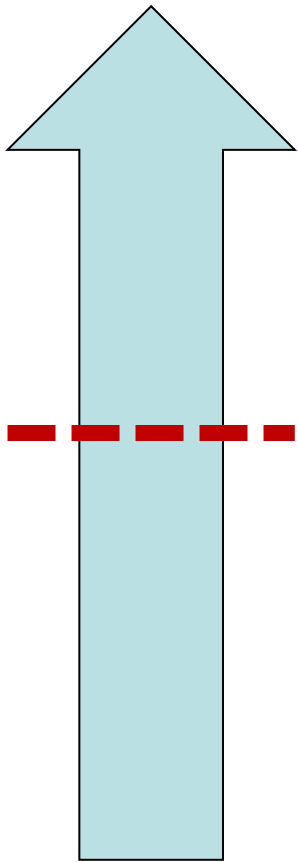
Prompting in Whisper: encoder-decoder

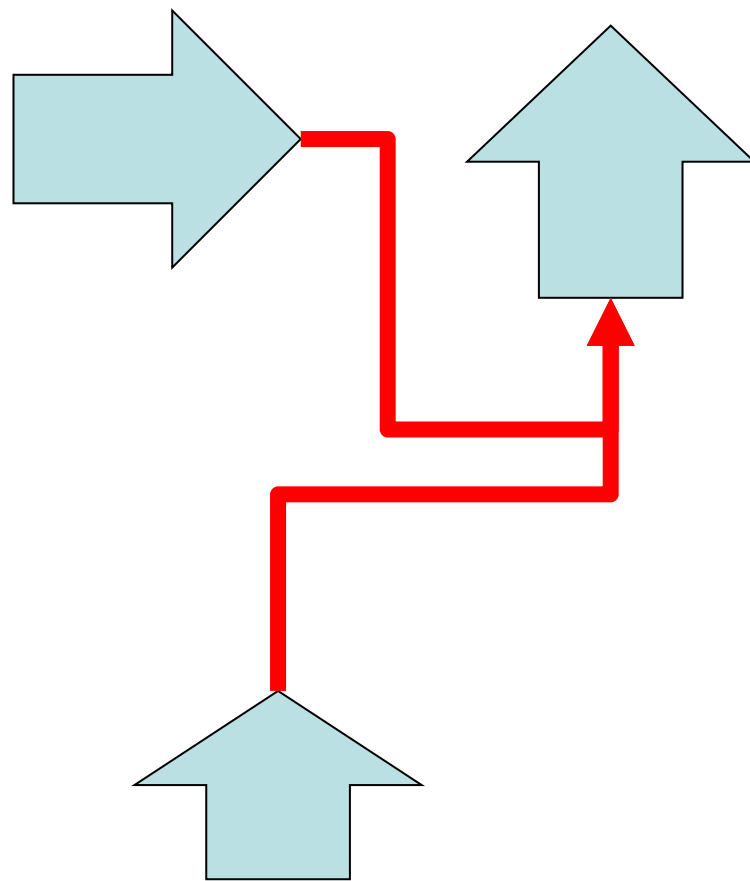
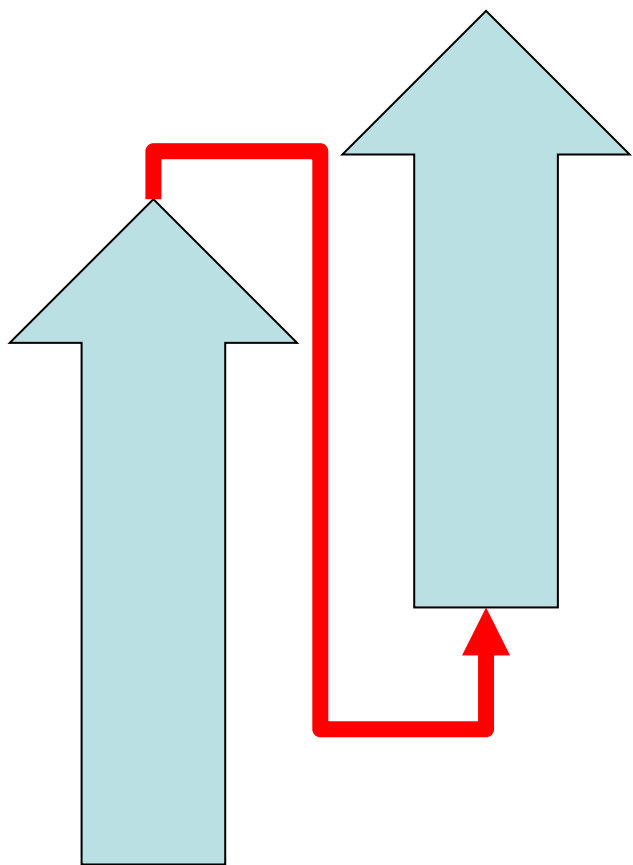






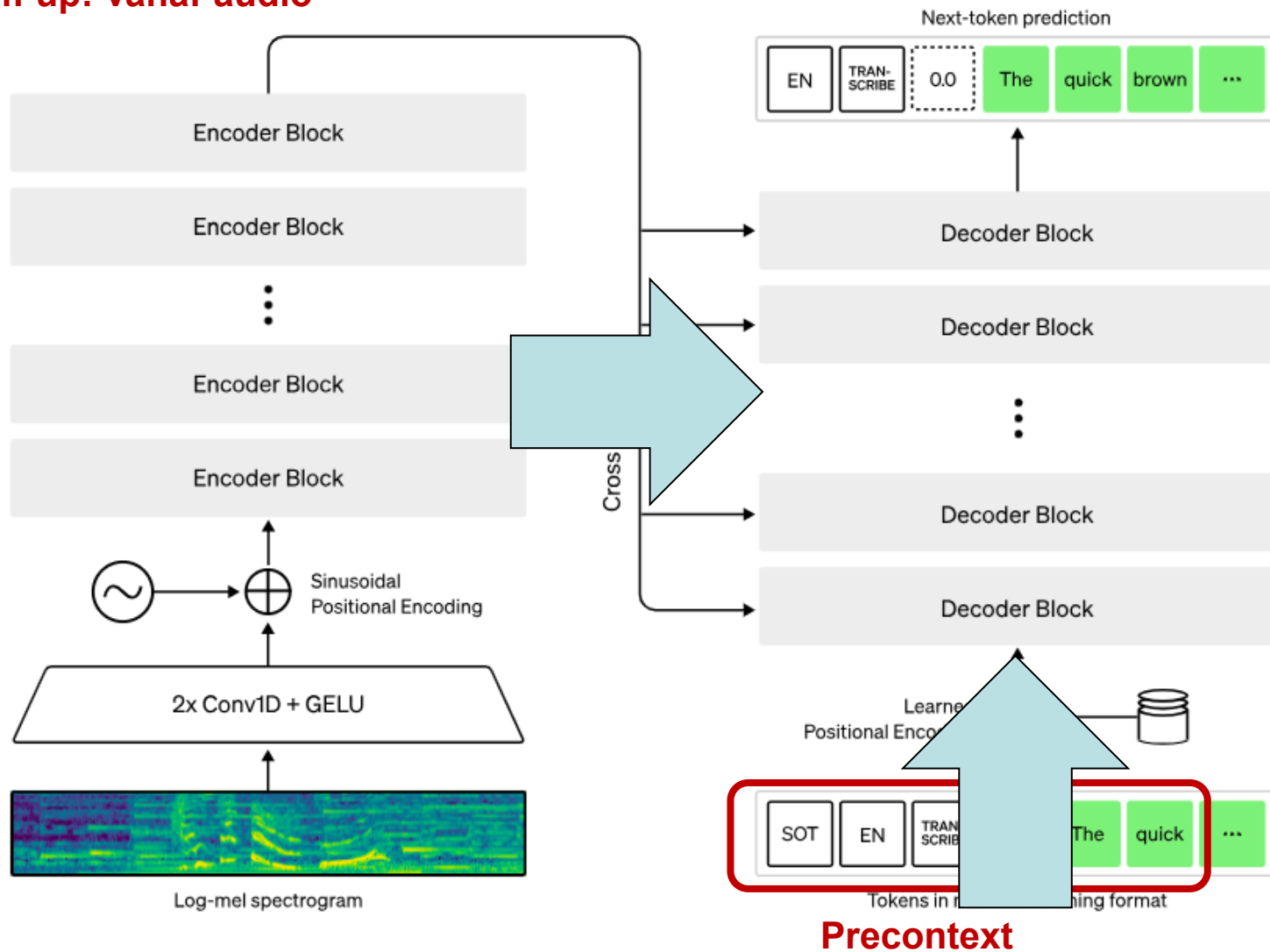
Information flow



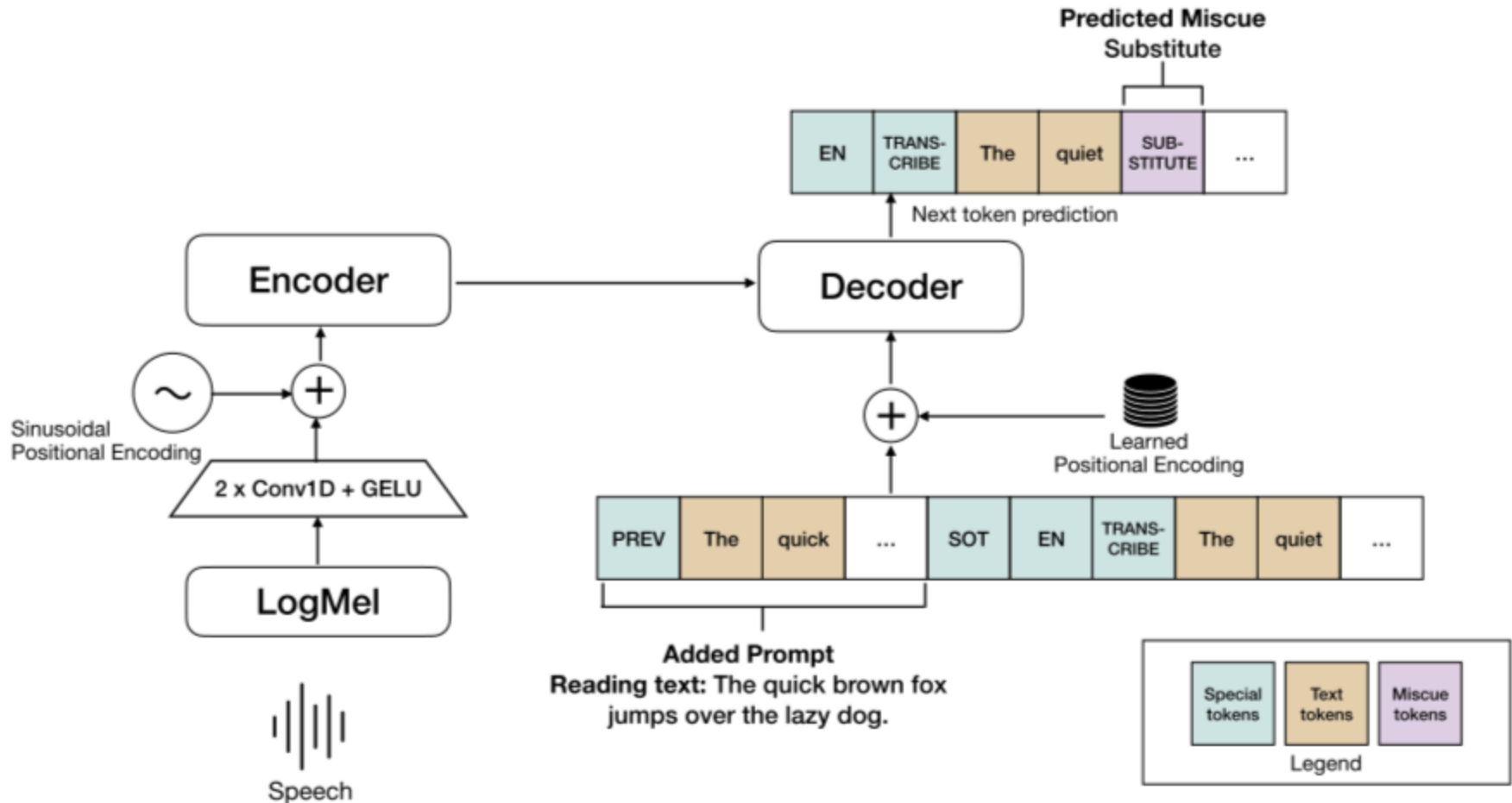


Whisper: encoder-decoder

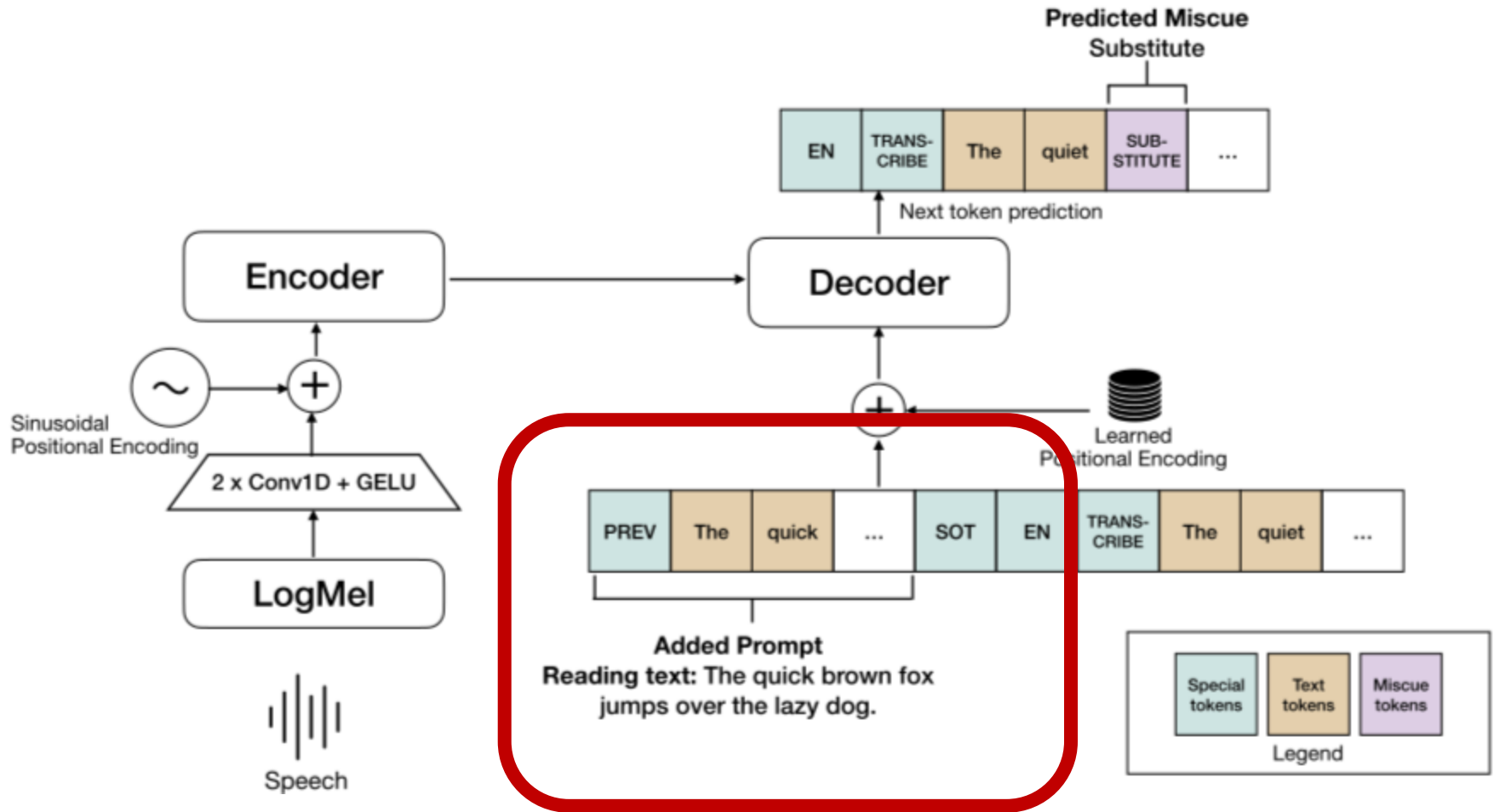
Bottom-up: vanaf audio



<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 in the litt. (see e.g. the Verdini paper in Github)
- https://cookbook.openai.com/examples/whisper_prompting_guide
- <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

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

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