




Vishal Rajput

Speech Recognition

AI-Vision Engineer @SkyeBase | AI Writer | AI Researcher | Masters AI, KU Leuven



“ *Speech is dress of thoughts. Good looks can attract people towards you, but speech is what wins their heart.* ”

Agenda

- ❑ What is speech recognition
- ❑ Why do we need speech recognition
- ❑ What are the main problems
- ❑ Understand the basics: Traditional ASR
- ❑ State of Speech recognition
- ❑ Where does OpenAI comes in
- ❑ What's next





what is speech recognition



All



Images



Videos



Books



News



More

Tools

About 475.000.000 results (0,52 seconds)

Dictionary

Definitions from [Oxford Languages](#) · [Learn more](#)

Search for a word



speech recognition

noun

the process of enabling a computer to identify and respond to the sounds produced in human speech.
"speech recognition technologies"

[Feedback](#)

Translations and more definitions

Background

All of us are always communicating, and as we all know that our world is becoming more and more digital and advance, why not also talk to the machines that have become integral part of our society.



Why do we need such systems, how can they help us?

Firstly, because we can make such systems, so why not.

Secondly, to make our lives little more easier and lazier at the same time.

Thirdly, critical places where we are not in positions to use our hands to operate stuff.

Lastly, to make our work little faster.

Speech recognition

- Many exciting & valuable applications



Content captioning



Cars / Hands-free interfaces



Home devices



Mobile devices

Well, I can survive without a voice operated home theatre, but I would like to have one for my GPS, don't want to look at my screen when I'm driving.





Problems

Low signal-noise ratio

Speaker variability (accents)

Natural Conversational Speech (ahh, hunn, yaaa)

Parts of Speech Recognition



Speech Transcription



Word Spotting/ trigger word



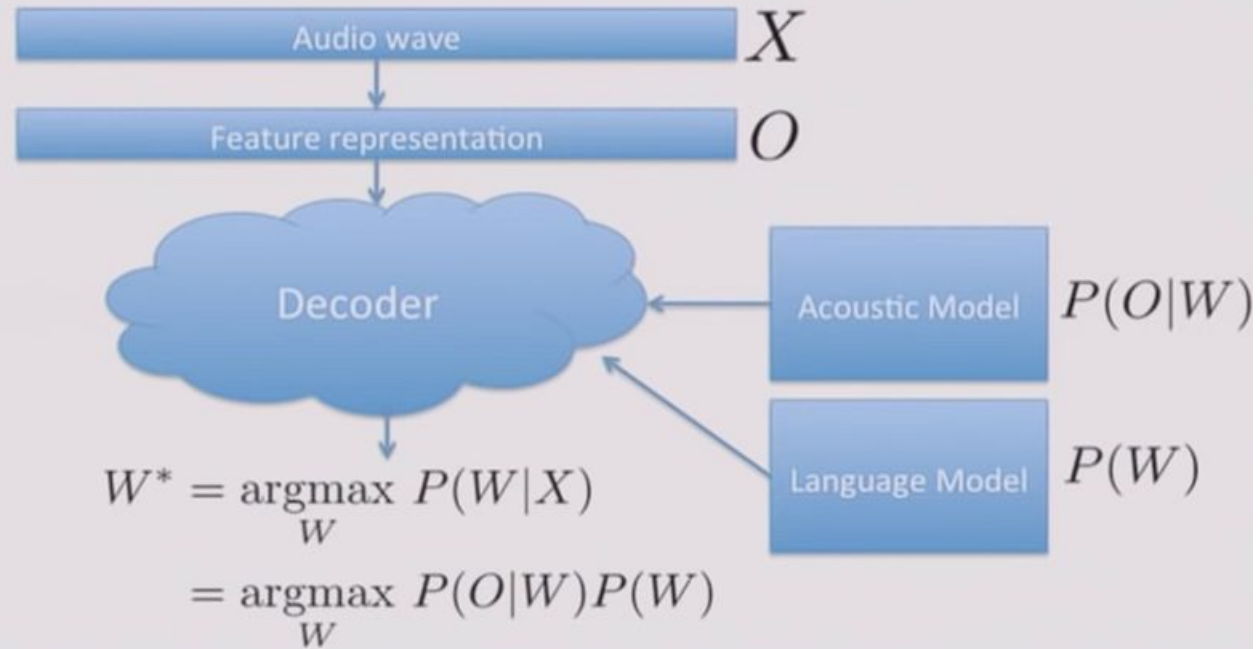
Speaker identification/verification

Speech Recognition -- the classical way

It was trained in a reverse way, first we made text-to-audio system and then used it to make speech recognition systems.

Traditional ASR pipeline

- Traditional systems break problem into several key components:



Traditional ASR pipeline

Traditional ASR pipeline

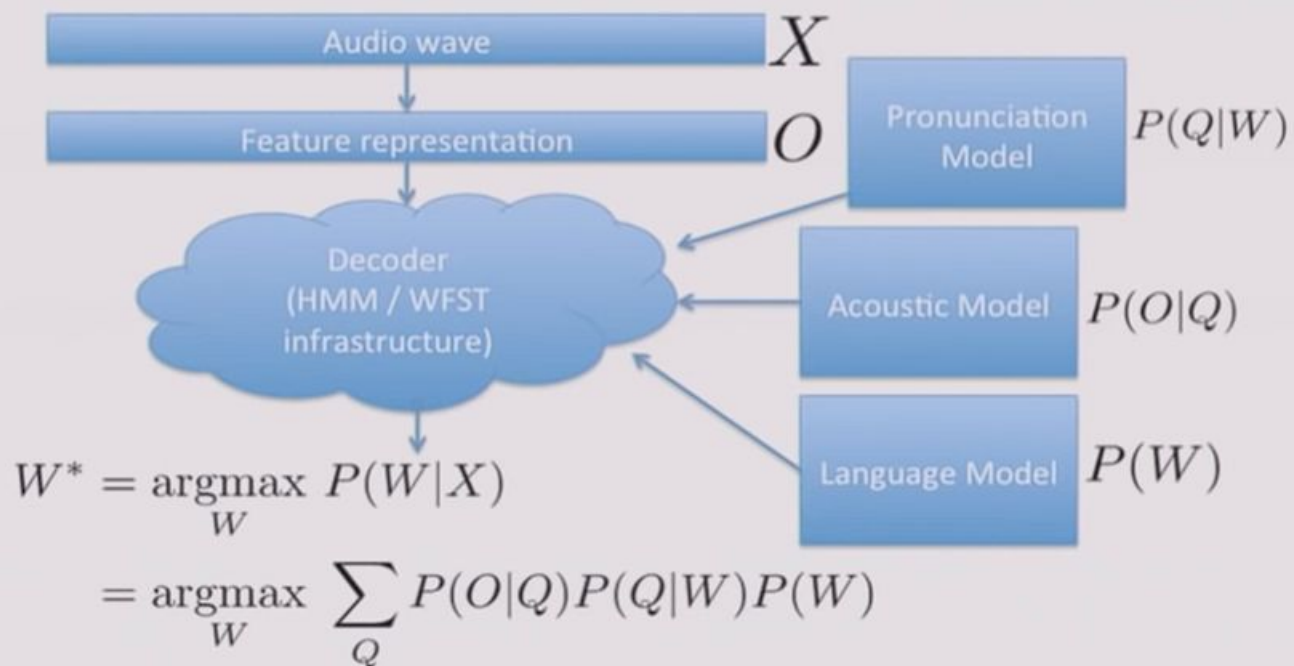
- Usually represent words as sequence of “phonemes”:

$$w_1 = \text{“hello”} = [\text{HH AH L OW}] = [q_1 q_2 q_3 q_4]$$

- Phonemes are the perceptually distinct units of sound that distinguish words.
 - Quite approximate... but sorta standardized-ish.
 - Some labeled corpora available (e.g., TIMIT)

Traditional ASR pipeline

- Traditional systems usually model phoneme sequences instead of words. This necessitates a dictionary or other model to translate.



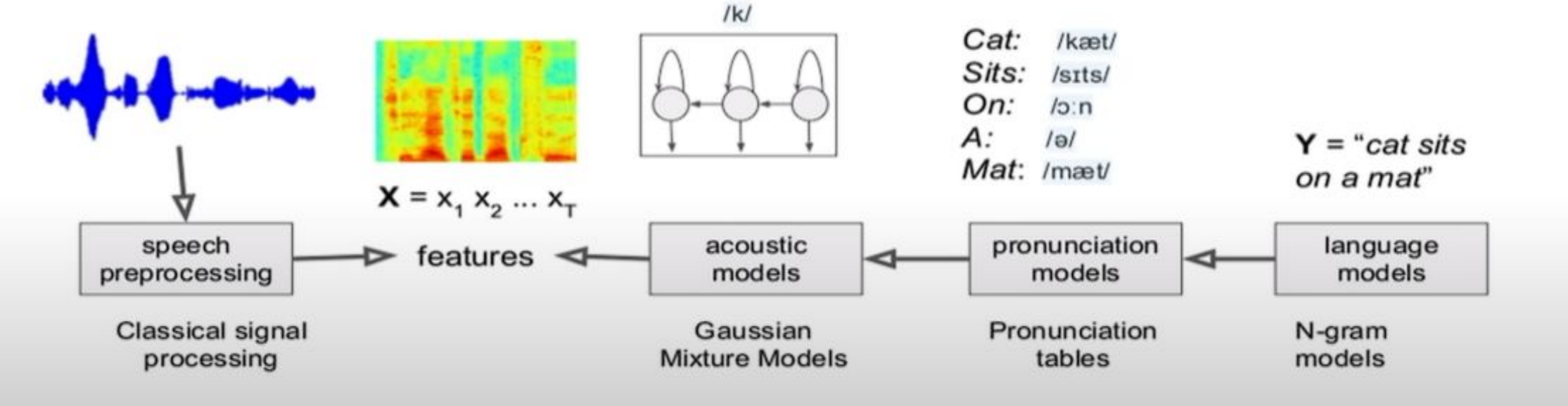
Advantages:

Can be tweaked for new pronunciation.

But still quite complex for new accent.

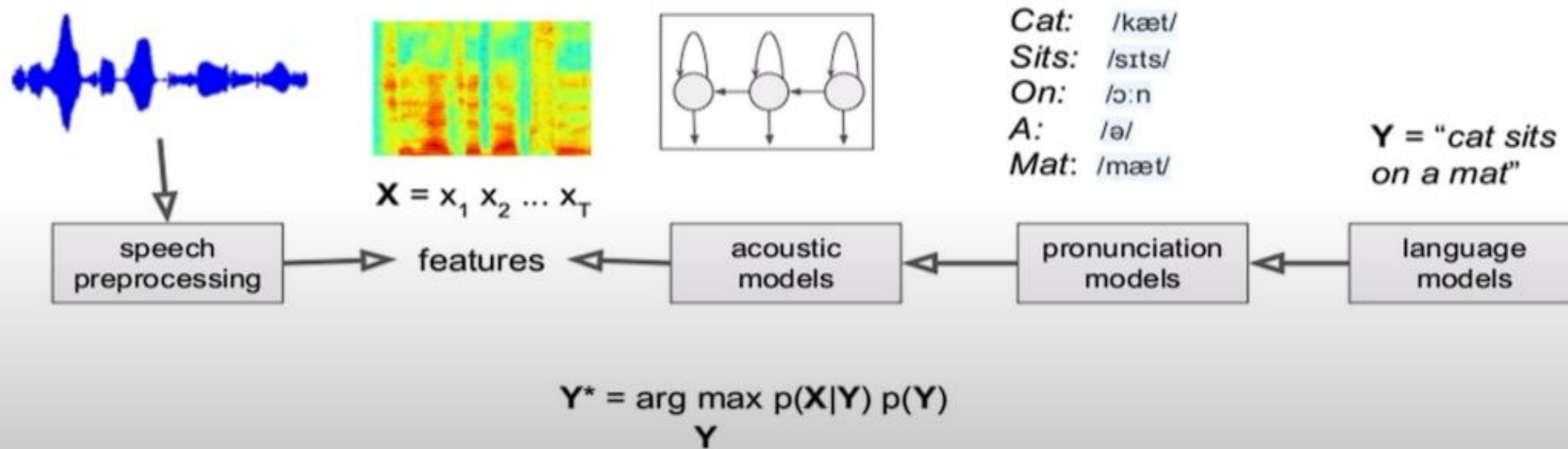
Speech Recognition -- the classical way

- Different statistical models used in different components



Speech Recognition -- the classical way

- Inference: Given audio features $\mathbf{X} = x_1 x_2 \dots x_T$ infer most likely text sequence $\mathbf{Y}^* = y_1 y_2 \dots y_L$ that caused the audio features

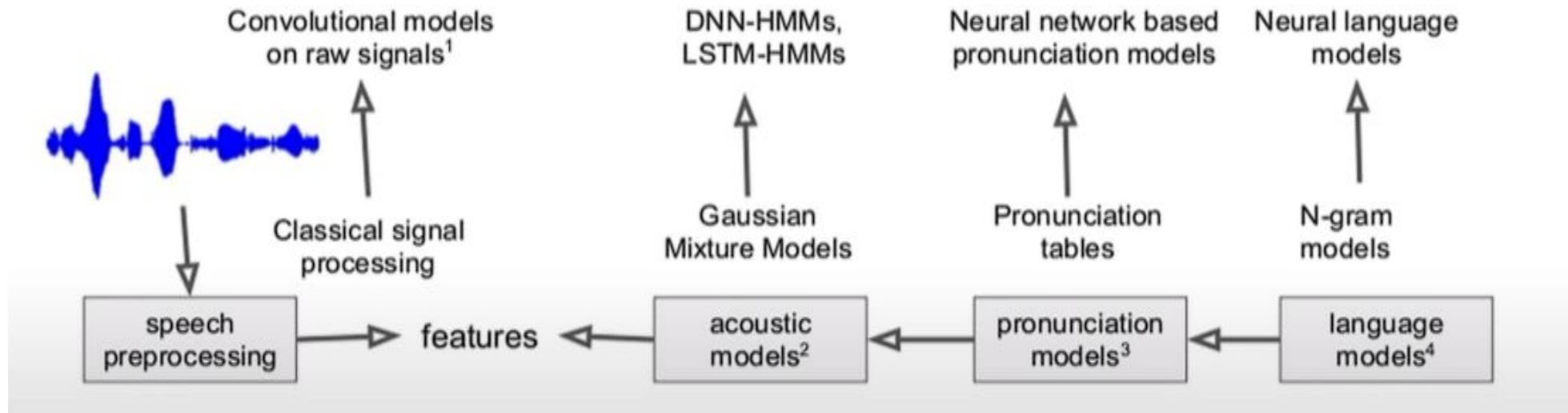




Introducing Deep learning
first in acoustic model

Speech Recognition -- the neural network invasion

- Each of the components seems to be better off with a neural network



And yet ...

- Each component is trained independently, with a different objective!
- Errors in one component may not behave well with errors in another component
- Instead, lets train models that encompass all of these components together (end-to-end models)
 - Connectionist Temporal Classification (CTC)
 - Sequence to sequence (Listen Attend and Spell)



How do we handle audio data

- Simple 1D signal:



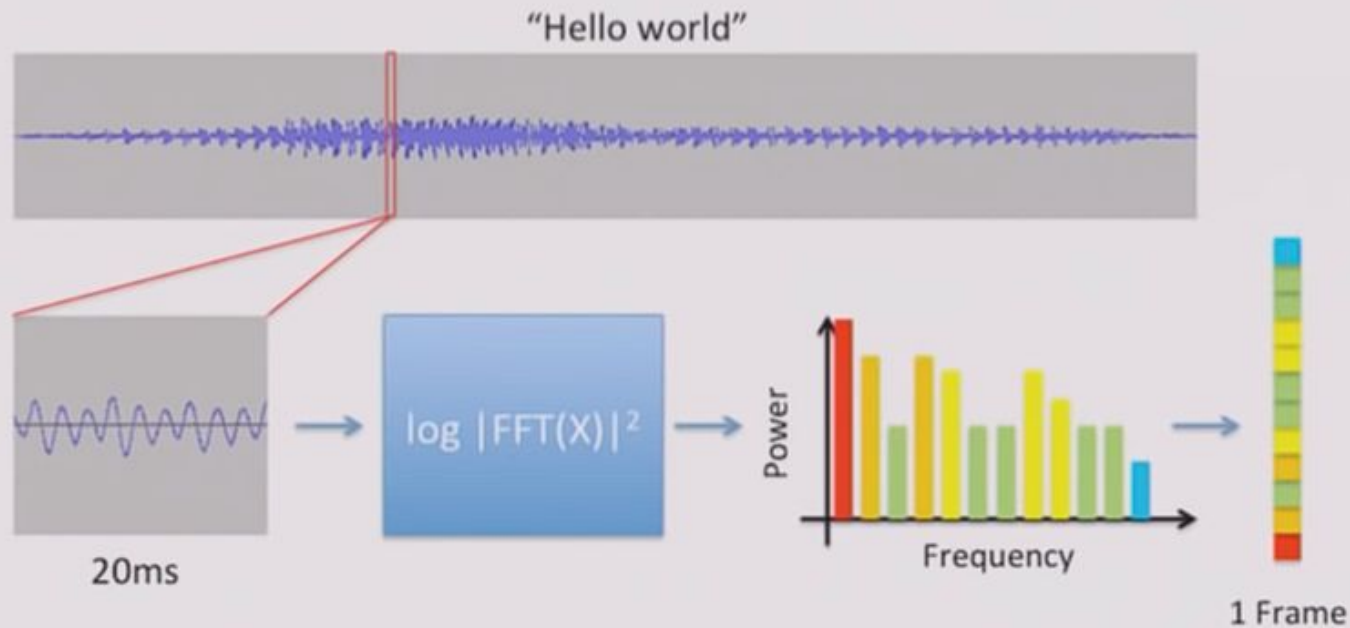
Typical sample rates for speech: 8KHz, 16KHz.
Each sample typically 8-bit or 16-bit.

- 1D vector: $X = [x_1 x_2 \dots]$

- Two ways to start:
 - Minimally pre-process (e.g., simple spectrogram).
 - We'll use this.
 - Train model from raw audio wave.
 - It works!
- See, e.g., Sainath et al., Interspeech 2015

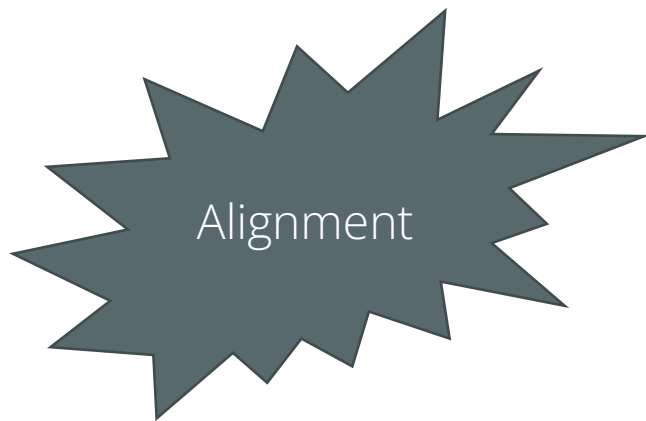
Spectrogram

- Take a small window (e.g., 20ms) of waveform.
 - Compute FFT and take magnitude. (i.e., power)
 - Describes frequency content in local window.



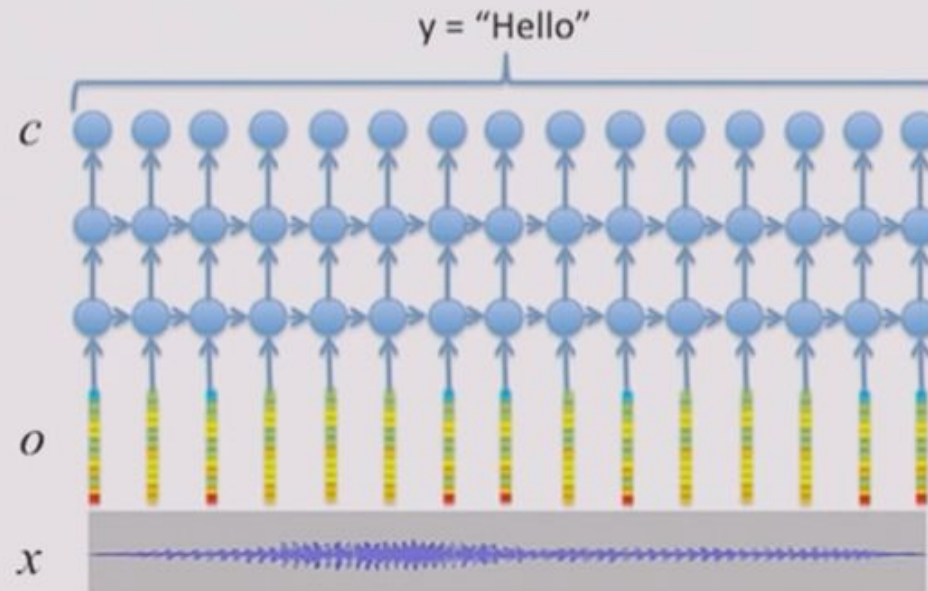
We use Mel Spectrogram
not spectrogram

Can you identify the problem
this?



Acoustic Model

- Goal: create a neural network (DNN/RNN) from which we can extract transcription, y .
 - Train from labeled pairs (x, y^*)

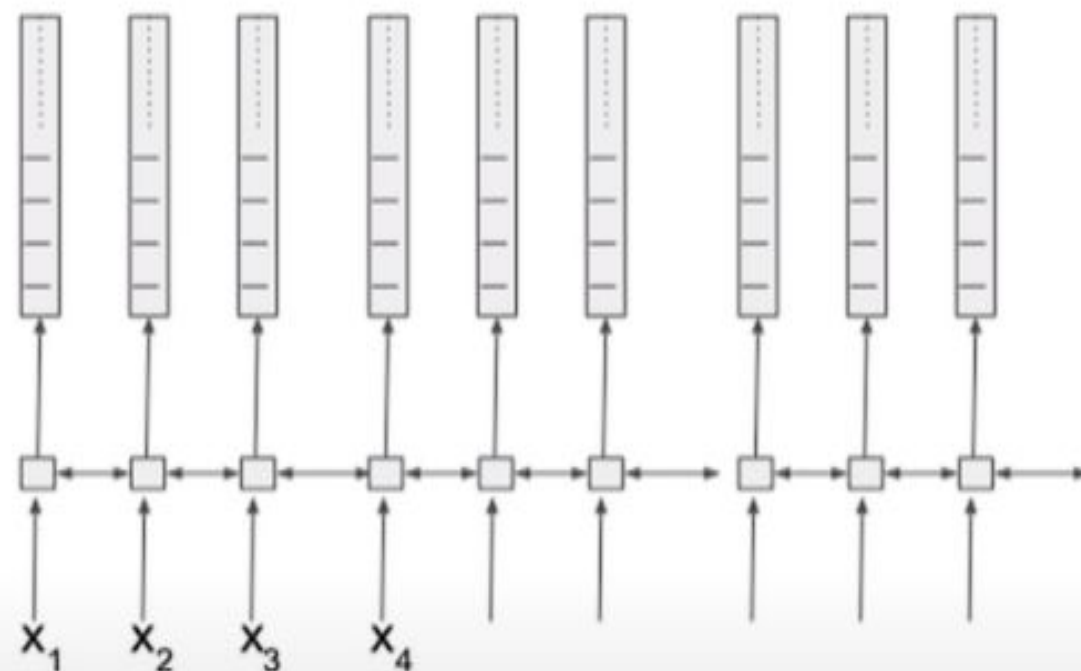




CTC (Connectionist Temporal Classification)

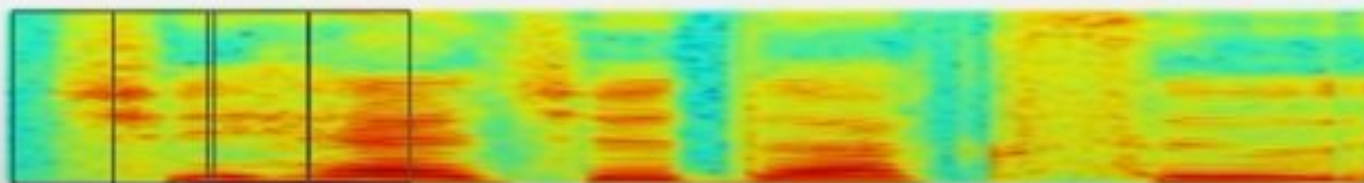
In CTC, the output of the neural network is a sequence of probability distributions over all possible labels, at each time step. CTC then evaluates the probability of the target transcript given the network's output sequence by considering all possible alignments between the network's predictions and the target sequence. The final loss is the negative log-likelihood of the target transcript, given the network's output.

Connectionist Temporal Classification



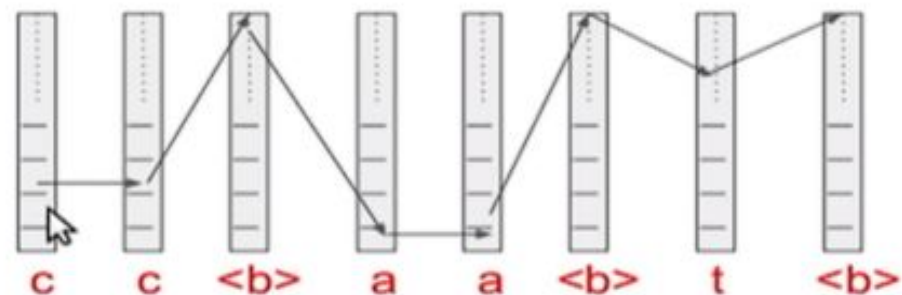
Softmax over vocabulary
 $\{a, b, c, d, e, f, \dots, z, ?, \dots, !, \dots\}$ and extra
token $\langle b \rangle$.

Softmax at step, t , gives a score $s(k,t)$
 $= \log \Pr(k, t | \mathbf{X})$ to category k in the
output at time t .



CTC - How frame predictions map to output sequences

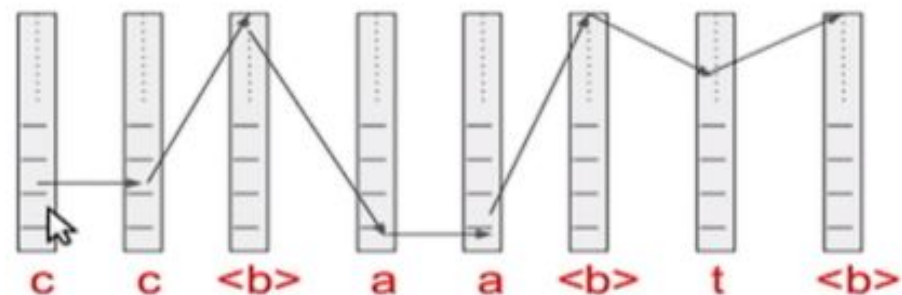
- Repeated tokens are deduplicated
 - **cc aa t **
- Any original transcript, maps to all possible paths in the duplicated space:
 - **ccaat** maps to cat
 - **ccat** maps to cat
 - **cccccaaaaattttt** maps to cat
 - **cccccaaaaattttt** maps to cat
- The score (log probability) of any path is the sum of the scores of individual categories at the different time steps
- The probability of any transcript is the sum of probabilities of all paths that correspond to that transcript



Because of dynamic programming, it is possible to compute both the log probability $p(\mathbf{Y}|\mathbf{X})$ and its gradient exactly! This gradient can be propagated to neural network whose parameters can then be adjusted by your favorite optimizer!

CTC - How frame predictions map to output sequences

- Repeated tokens are deduplicated
 - **cc aa t **
- Any original transcript, maps to all possible paths in the duplicated space:
 - **ccaat** maps to cat
 - **ccat** maps to cat
 - **cccccaaaaattttt** maps to cat
 - **cccccaaaaattttt** maps to cat
- The score (log probability) of any path is the sum of the scores of individual categories at the different time steps
- The probability of any transcript is the sum of probabilities of all paths that correspond to that transcript



Because of dynamic programming, it is possible to compute both the log probability $p(\mathbf{Y}|\mathbf{X})$ and its gradient exactly! This gradient can be propagated to neural network whose parameters can then be adjusted by your favorite optimizer!

LAS model

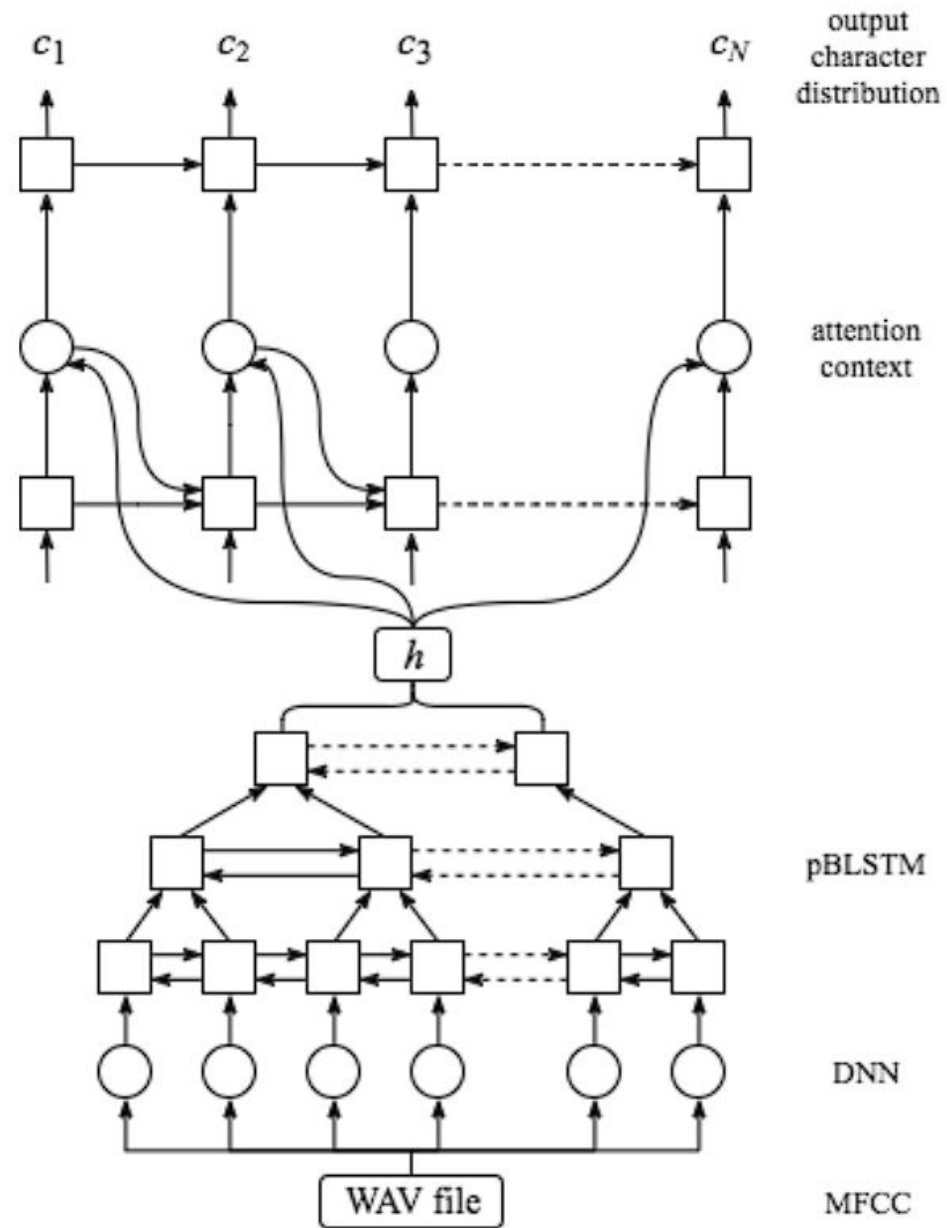
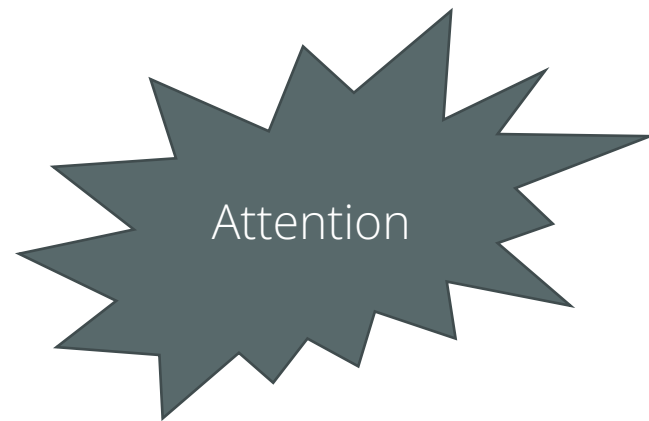


Figure 2: LAS model with a DNN layer.



Let's appreciate other papers before going into OpenAI's whisper

WAV2VEC: UNSUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Wav2vec mainly to create a better representation of audio data. It uses contrastive loss to do that. (Unsupervised)

Predicts future samples from the current sample.

They first encode raw speech samples into a representation of features at a lower temporal frequency. Then, they implicitly model a density ratio and try to predict the future based on this "latent space."

The model takes raw audio signals as input and then applies two networks. The encoder network embeds the audio signal in a latent space, and the context network combines multiple time steps of the encoder to obtain contextualized representations.

VQ-WAV2VEC: SELF-SUPERVISED LEARNING SPEECH REPRESENTATION

- VqWav2vec aims at learning discrete representation of speech via context prediction task.
- It uses Wav2vec first and then uses BERT for context prediction.

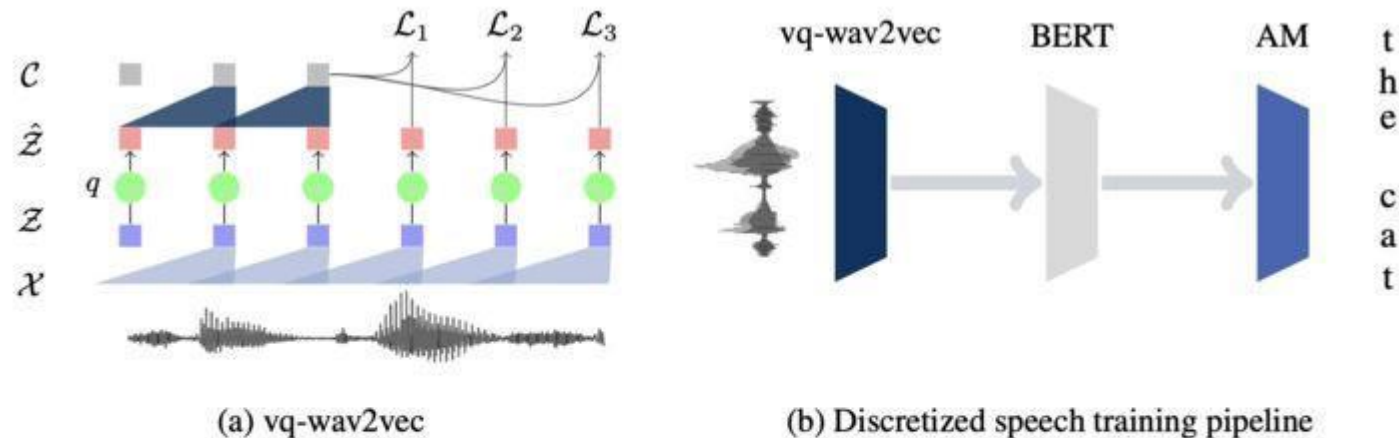


Figure 1: (a) The vq-wav2vec encoder maps raw audio (\mathcal{X}) to a dense representation (\mathcal{Z}) which is quantized (q) to $\hat{\mathcal{Z}}$ and aggregated into context representations (\mathcal{C}); training requires future time step prediction. (b) Acoustic models are trained by quantizing the raw audio with vq-wav2vec, then applying BERT to the discretized sequence and feeding the resulting representations into the acoustic model to output transcriptions.

Wav2vec2: A Framework for Self-Supervised Learning of Speech Representations

- Wav2vec2 is a framework for self-supervised learning of representations from raw audio data
- Encodes speech audio via a multi-layer convolutional neural network and then masks spans of the resulting latent speech representations, similar to masked language modelling
- The latent representations are fed to a Transformer network to build contextualized representations and the model is trained via a contrastive task where the true latent is to be distinguished from distractors.
- They learn discrete speech units to represent the latent representations in the contrastive task.
- After pre-training on unlabeled speech, the model is fine-tuned on labeled data with a Connectionist Temporal Classification (CTC) loss to be used for downstream speech

XLSR-wav2vec: UNSUPERVISED CROSS-LINGUAL REPRESENTATION LEARNING FOR SPEECH RECOGNITION

- XLSR proposes a framework to learn cross-lingual speech representation by pre-training a single model from the raw waveform.
- This model is based on wav2vec2.
- The resulting model is fine-tuned on labeled data, and experiments show that cross-lingual pre-training significantly outperforms monolingual pre-training.
- The latent representations are fed to a Transformer network to build contextualized representations and the model is trained via a contrastive task where the true latent is to be distinguished from distractors.

OpenAI's Whisper (Finally!!! The wait is over)





Data Preparation

There were two main pathways namely supervised and unsupervised in speech recognition. In the **unsupervised setting** (*1000000 hours of audio data is available*).

Pre-trained audio encoders learn high-quality representations of speech, but because they are purely unsupervised they lack an equivalently performant decoder mapping those representations to usable outputs, necessitating a finetuning stage in order to actually perform a task such as speech recognition.

In **supervised**, there is still only a moderate amount of this data available (*5000 hours*).

OpenAI used weak supervision to scale the data from 5000 hours to 680000 hours (scaled by a factor of 130).

Their initial inspection showed a large number of subpar transcripts in the raw dataset. To address this, they developed several automated filtering methods to improve transcript quality. They also developed many heuristics to detect and remove machine-generated transcripts from the training dataset. For ex., *an all-uppercase or all-lowercase transcript is very unlikely to be human generated*.

They also performed manual inspection of these data sources sorting.

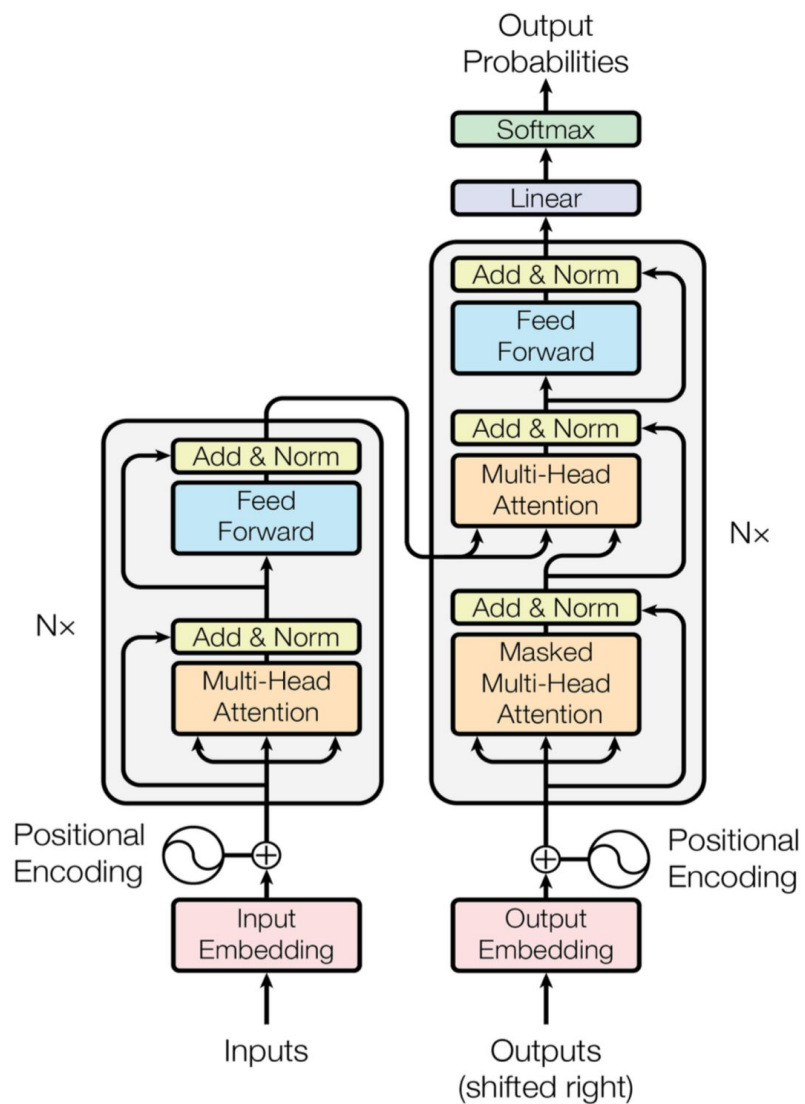
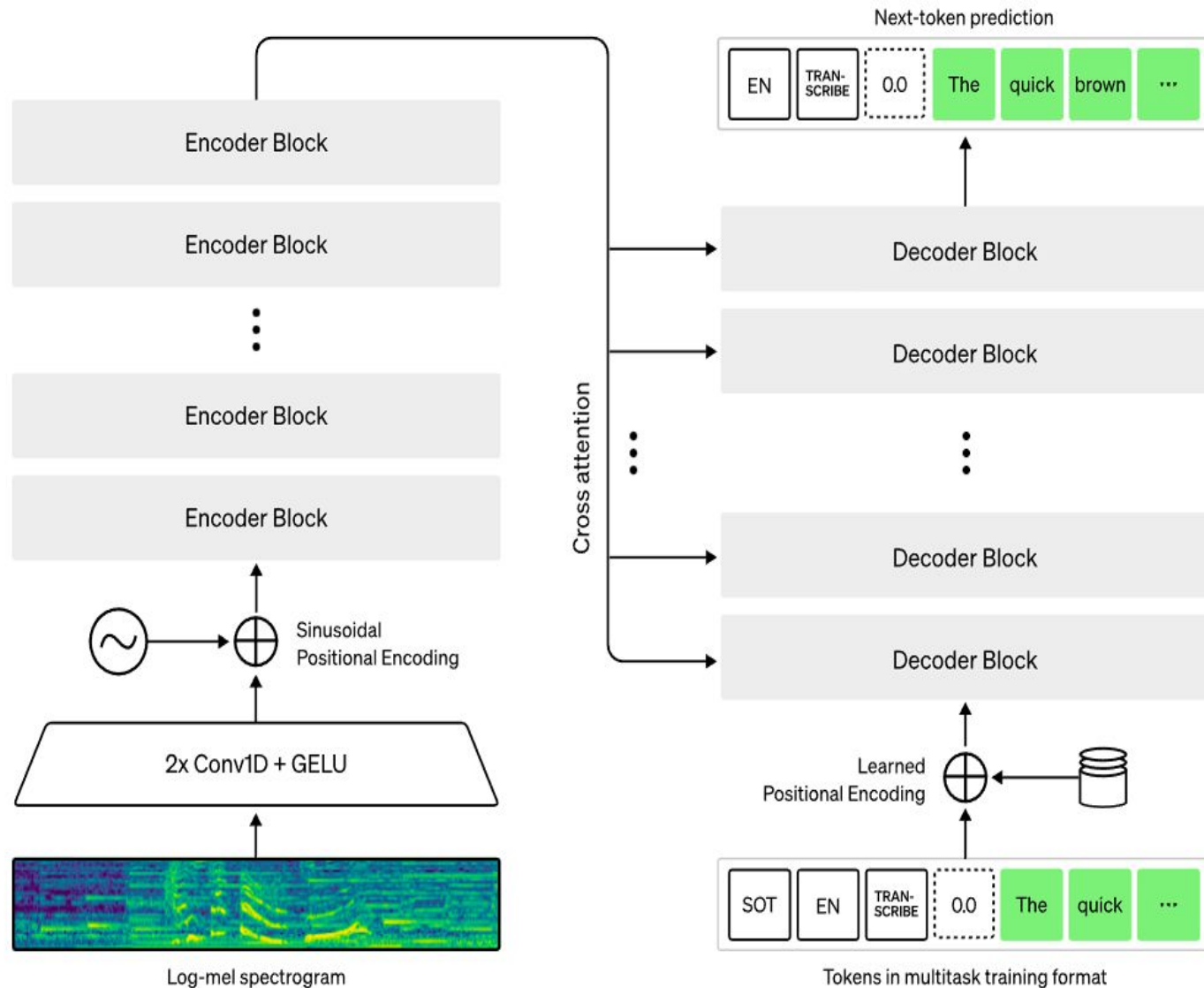
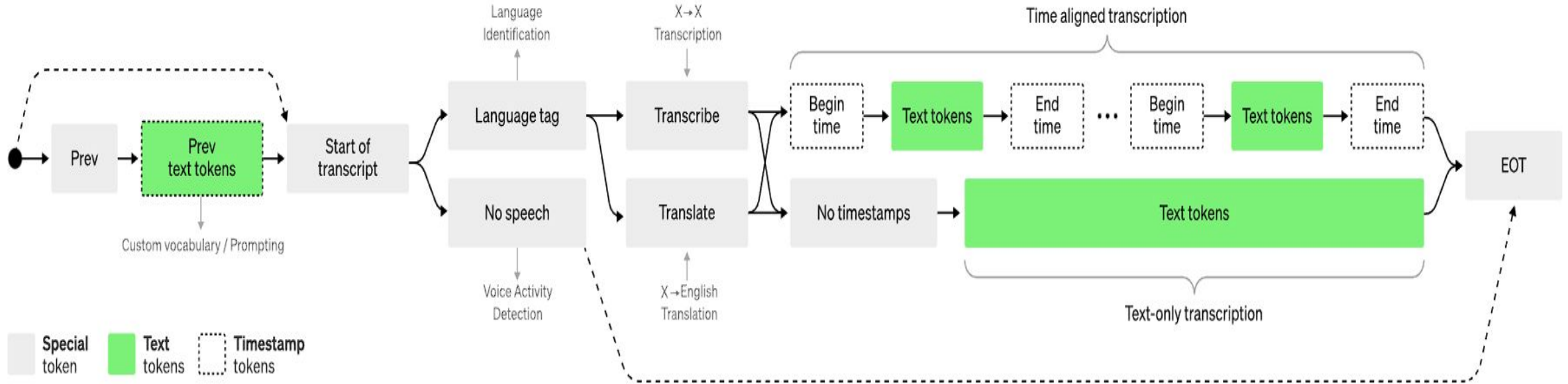


Figure 1: The Transformer - model architecture.





The flow of model prediction

- ❑ First, the model predicts the language being spoken which is represented by a unique token for each language in our training set (99 total).
- ❑ In the case where there is no speech in an audio segment, the model is trained to predict a **<|nospeech|>** token indicating this.
- ❑ The next token specifies the task (either transcription or translation) with an **<|transcribe|>** or **<|translate|>** token.
- ❑ After this, the model specifies whether to predict timestamps or not by including a **<|notimestamps|>** token for that case.
- ❑ At this point, the task and desired format are fully specified, and the output begins.

Other Details

For timestamp prediction, the model predicts time relative to the current audio segment, quantizing all times to the nearest 20 milliseconds which matches the native time resolution of Whisper models.

Earlier in development model had a tendency to transcribe plausibly but almost always incorrect guesses for the names of speakers. Later, the model is fine-tuned briefly on the subset of transcripts that do not include speaker annotations which removes this behaviour.

Speech recognition research typically evaluates and compares systems based on the word error rate (WER) metric. However, WER, which is based on string edit distance, penalizes all differences between the model's output and the reference transcript including innocuous differences in transcript style. Whisper opts to address this problem with extensive standardization of text before the WER calculation to minimize penalization of non-semantic differences.

```

31 def __init__(self, path):
32     self.file = None
33     self.fingerprints = set()
34     self.logdupes = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, "requests.log"),
39                          "a")
40         self.file.seek(0)
41         self.fingerprints.update(fingerprint(fp) for fp in self.fp)
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool("debug", True)
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)

```

#Transcription can also be performed within Python:
`import whisper`

```

model = whisper.load_model("base")
result = model.transcribe("audio.mp3")
print(result["text"])

```

#Example usage of `whisper.detect_language()` and `whisper.decode()` which provide lower-level access to the model.

```
model = whisper.load_model("base")
```

```

# load audio and pad/trim it to fit 30 seconds
audio = whisper.load_audio("audio.mp3")
audio = whisper.pad_or_trim(audio)

```

```

# make log-Mel spectrogram and move to the same device as the model
mel =
whisper.log_mel_spectrogram(audio).to(model.device)

```

```

# detect the spoken language
_, probs = model.detect_language(mel)
print(f"Detected language: {max(probs, key=probs.get)}")

```

```

# decode the audio
options = whisper.DecodingOptions()
result = whisper.decode(model, mel, options)

```

```

# print the recognized text
print(result.text)

```



What's next

What's next

Improved decoding strategies: Long-form transcription

Increase Training Data For Lower-Resource Languages

Studying fine-tuning

Studying the impact of Language Models on Robustness: It might be language model entirely

Whisper in full glory

speech recognition.

📄 READ PAPER

🔗 VIEW CODE

📩 VIEW MODEL CARD

Whisper examples:

Accent ▾



REVEAL TRANSCRIPT

Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web. We show that the use of such a large and diverse dataset leads to improved robustness to accents, background noise and technical language. Moreover, it enables transcription in multiple languages, as well as translation from those languages into English. We are open-sourcing models and inference code to serve as a foundation for building useful applications and for further research on robust speech processing.

< 00:00 🗑️ ⏸️ 🔴



Thank You!!!

<https://www.linkedin.com/in/vishal-rajput-999164122/>

<https://medium.com/aiguys>