Audio Processing in NLP

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

July 21, 2025

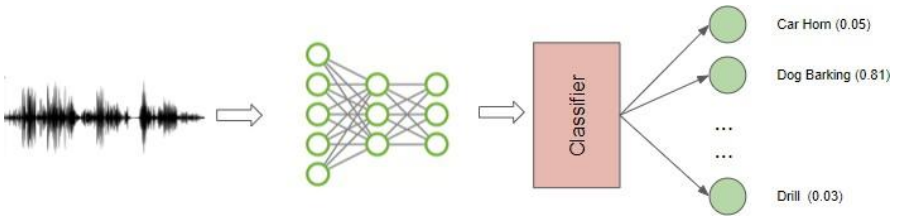

Classifier
Car Horn (0.05)
Dog Barking (0.81)
...
...
Drill (0.03)

Table of Contents

Table of Contents (cont.)

Motivation

- Audio data is abundant and diverse, encompassing speech, music, and environmental sounds.
- Traditional NLP methods focus on text, but audio data requires specialized techniques for processing.
- The rise of voice assistants and audio-based applications highlights the need for effective audio processing in NLP.
- Audio signals contain rich information that can enhance understanding and interaction in various applications.

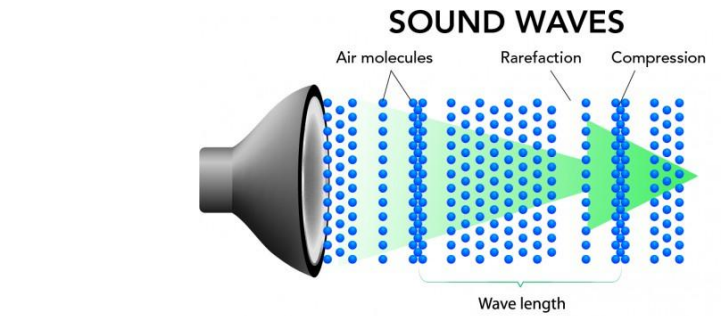
Learning Outcomes

By the end of this session, you will be able to:

- Understand the fundamentals of audio processing in NLP.
- Learn how to extract features from raw audio data.
- Explore various architectures for Automatic Speech Recognition (ASR).
- Gain insights into self-supervised learning models for audio.
- Implement Text-to-Speech (TTS) systems using modern techniques.
- Address advanced audio tasks such as speaker diarization and emotion recognition.

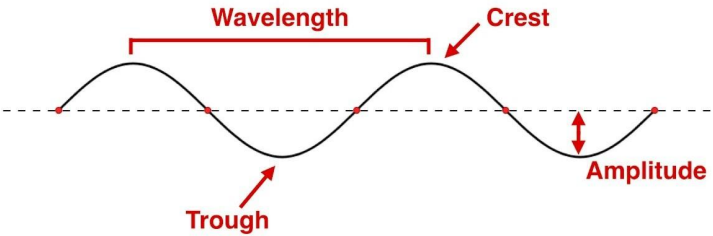
Fundamentals of Sound

What is Sound?



- Sound is a type of energy produced by vibrating objects.
- It travels through air (or other media) as waves.
- Sound waves are longitudinal waves, meaning the particle displacement is parallel to the direction of wave propagation.

Fundamentals of Sound: Physics Concepts



- **Waveform**: The shape of the sound wave, representing how air pressure changes over time.
- **Frequency**: Number of cycles (oscillations) per second, measured in Hertz (Hz). Determines the pitch of the sound.
- **Amplitude**: The height of the wave, representing the loudness or intensity of the sound.

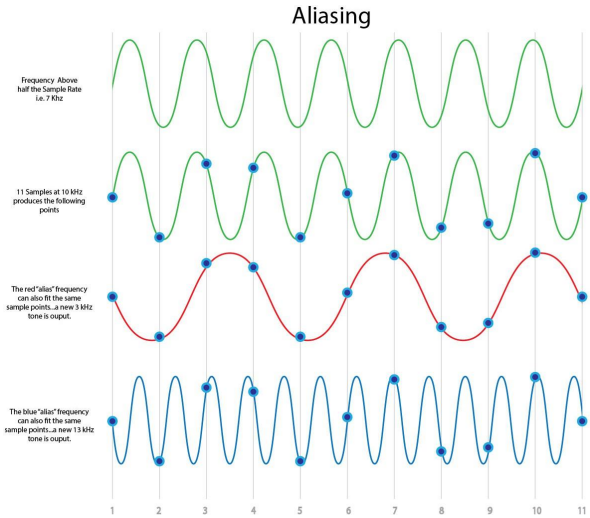
Fundamentals of Sound: Physics Concepts (cont.)

- **Sampling Theorem (Nyquist)**: To accurately digitize a sound, the sampling rate (f_s) must be greater than twice the highest frequency present in the signal (f_{max}):

$$f_s > 2f_{max}$$

- **Aliasing**: If the sampling rate is too low, higher frequencies are misrepresented as lower frequencies, causing distortion.

Fundamentals of Sound: Physics Concepts (cont.)



Pre-processing Pipeline

Pre-processing Pipeline Details

1. Pre-emphasis Filter

- Enhances high-frequency components of the audio signal.
- Equation:

$$y[n] = x[n] - \alpha x[n-1]$$

where $x[n]$ is the input signal, $y[n]$ is the output, and α is typically between
0.95 and 0.97.

- Helps balance the frequency spectrum and improves feature extraction.

2. Framing

- Audio signals are divided into short frames (e.g., 25 ms) to capture local temporal features

3. Windowing

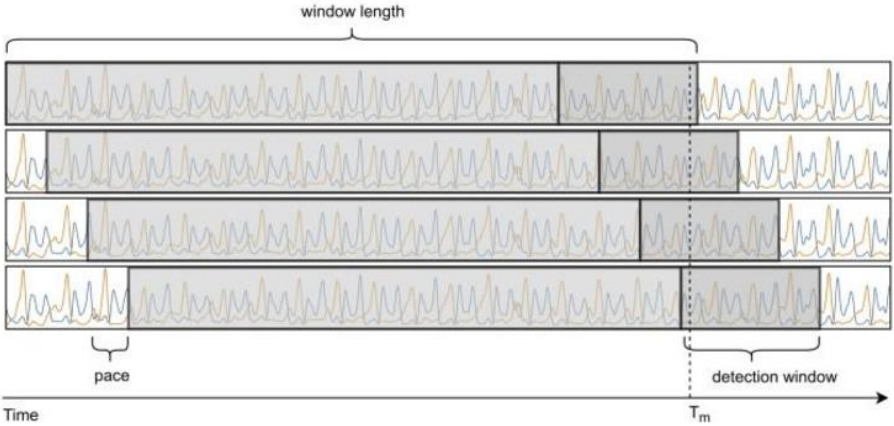
- Each frame is multiplied by a window function (commonly Hamming window) to reduce spectral leakage.
- Hamming window equation:

$$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right)$$

where N is the frame length.

Diagram: Waveform → Framed Windows

Pre-processing Pipeline Details (cont.)



Waveform segmented into overlapping frames and windowed (source: ResearchGate)

Pre-processing Pipeline Details (cont.)

Summary of Pre-processing Steps

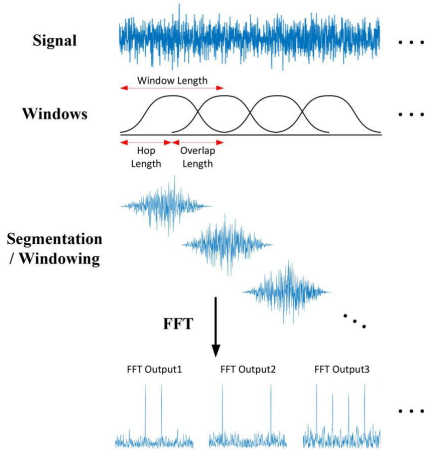
1. **Pre-emphasis:** Boosts high frequencies.
2. **Framing:** Splits signal into short, overlapping segments.
3. **Windowing:** Applies Hamming window to each frame.

STFT & Spectrograms

Short-Time Fourier Transform (STFT)

- STFT is a method to analyze the frequency content of a signal over time.
- It involves applying the Fourier Transform to short overlapping segments (frames) of the signal.
- The STFT provides a time-frequency representation of the signal.

Short-Time Fourier Transform (STFT) (cont.)



STFT process: Signal divided into frames, each transformed into frequency domain.

Short-Time Fourier Transform (STFT) (cont.)

- The STFT is computed as:

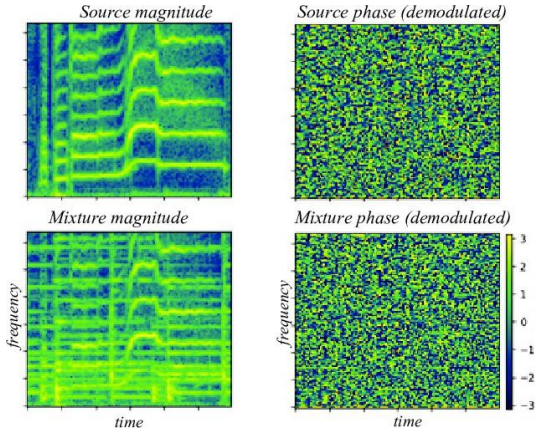
$$STFT(x[n]) = \sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{-j2\pi fm}$$

where $w[n-m]$ is a window function applied to each frame.

Short-Time Fourier Transform (STFT) (cont.)

- The result is a complex-valued matrix, where each column represents the frequency content of a frame.
- The magnitude of the STFT gives the amplitude of each frequency at each time step.
- The phase of the STFT provides information about the timing of the frequency components.

Short-Time Fourier Transform (STFT) (cont.)

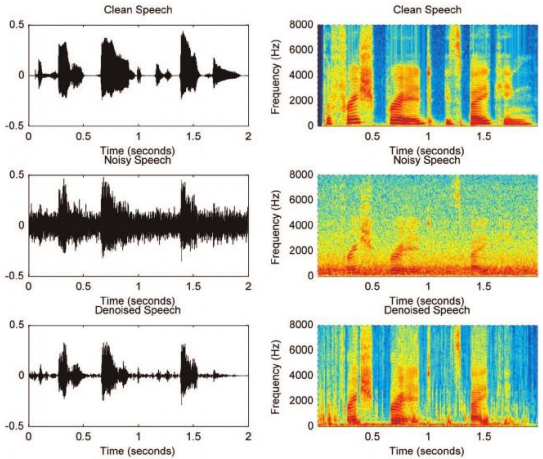


Magnitude and phase of STFT: Magnitude shows amplitude, phase shows timing of frequencies.

Spectrograms

- A spectrogram is a visual representation of the STFT.
- It displays time on the x-axis, frequency on the y-axis, and color intensity represents amplitude.
- Spectrograms are widely used in audio analysis, speech recognition, and music processing.

Spectrograms (cont.)

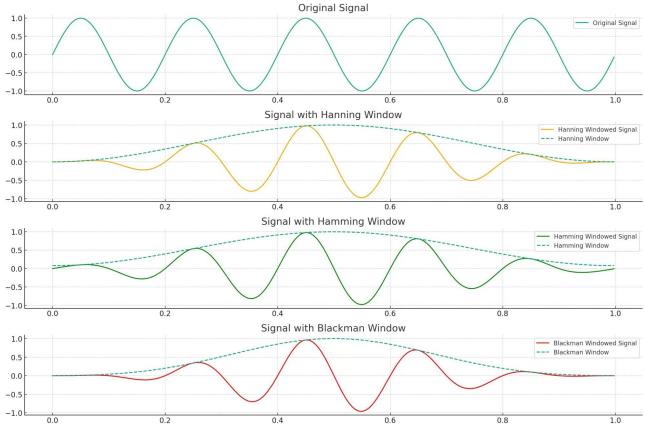


Example of a spectrogram: Time vs Frequency representation of an audio signal.

Spectrograms (cont.)

- Spectrograms can be computed using various window functions (e.g., Hamming, Hanning) and different frame sizes.
- They provide insights into how the frequency content of a signal changes over time.

Spectrograms (cont.)



Spectrogram variations: Different window functions and frame sizes affect the appearance of the spectrogram.

Mel-Scale & Filterbanks

Mel-Scale:

- The Mel scale is a perceptual scale that reflects how humans perceive pitch differences.
- It maps actual frequency (Hz) to Mel frequency, aligning more closely with human auditory perception.
- Equal distances on the Mel scale correspond to perceived equal pitch differences.

The transformation from frequency f (in Hz) to Mel scale m is:

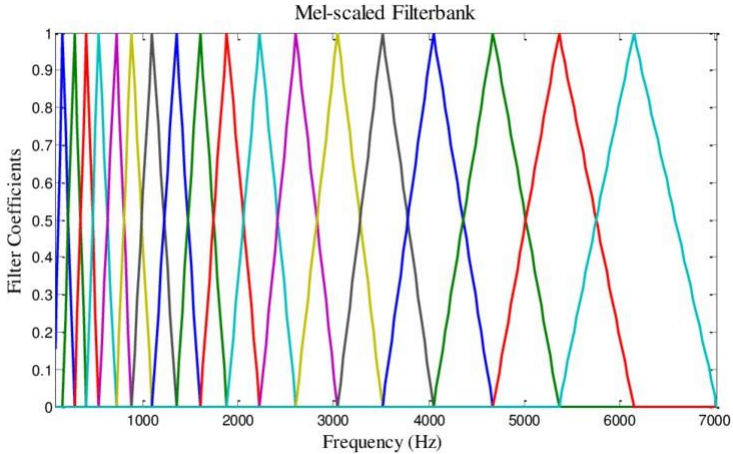
$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{1}$$

Mel-Scale & Filterbanks: Details (cont.)

Mel Filterbanks:

- The frequency axis is warped to the Mel scale.
- Typically, N triangular filters (e.g., $N = 40$) are created, spaced evenly in the Mel domain.
- Each filter collects energy from a range of frequencies, emphasizing perceptually important bands.
- The output is a set of Mel filterbank energies, which are used as features for audio and speech processing tasks.

Mel-Scale & Filterbanks: Details (cont.)



Mel-Scale & Filterbanks: Details (cont.)

Filterbank Construction Steps:

1. Convert the lower and upper frequency bounds to Mel scale.
2. Linearly space $N + 2$ points between these bounds in Mel scale.
3. Convert these points back to Hz to get filter edges.
4. For each filter k, define a triangular response:

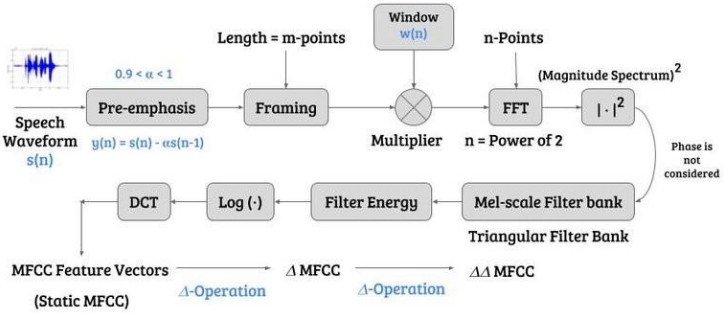
$$H_k(f) = \begin{cases} 0, & f < f_{k-1} \\ \frac{f - f_{k-1}}{f_k - f_{k-1}}, & f_{k-1} \leq f < f_k \\ \frac{f_{k+1} - f}{f_{k+1} - f_k}, & f_k \leq f < f_{k+1} \\ 0, & f \geq f_{k+1} \end{cases} \quad (2)$$

5. Multiply the power spectrum by each filter and sum to get filterbank energies.

MFCC Computation

- MFCCs are widely used in speech and audio processing.
- They capture the timbral texture of audio signals.
- Typically, only the first 13 coefficients are used for feature extraction.

MFCC Computation (cont.)



A block diagram for MFCC computation: From audio signal to MFCC features.

MFCC Computation (cont.)

Pipeline:

1. Compute log-Mel spectrogram
2. Apply Discrete Cosine Transform (DCT)
3. Keep first 13 coefficients (MFCCs)

MFCC Equation:

$$c_k = \sum_{n=1}^{N} \log(E_n) \cos\left[\frac{\pi k}{N}\left(n - \frac{1}{2}\right)\right] \qquad (3)$$

where:

- c_k is the k-th MFCC coefficient
- E_n is the energy in the n-th Mel filter
- N is the total number of Mel filters

Alternative Features

Linear Predictive Coding (LPC):

- Models the speech signal as a linear combination of past samples.
- Useful for speech compression and speaker recognition.
- Captures the spectral envelope efficiently.

Perceptual Linear Prediction (PLP):

- Incorporates psychoacoustic models to mimic human hearing.
- Reduces spectral information to perceptually relevant features.
- Often used in robust speech recognition systems.

Pitch:

-

Alternative Features (cont.)

- Extracted using autocorrelation or cepstral methods.

Formants:

- Resonant frequencies of the vocal tract.
- Key for phoneme classification and speech synthesis.
- Estimated using LPC or spectral analysis.

Spectrogram-based Learned Features:

- Deep learning models (CNNs, RNNs) extract features directly from spectrograms.
- Capture complex patterns and temporal dependencies.

Alternative Features (cont.)

When to Choose What:

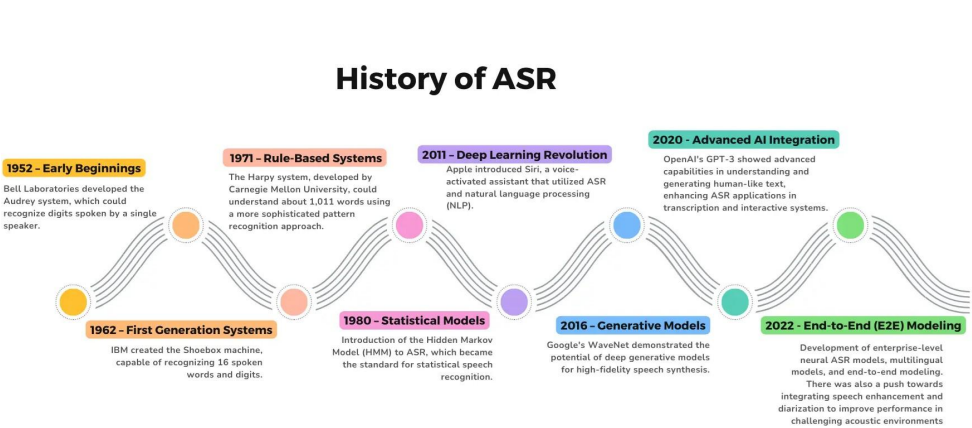
- **LPC/PLP:** When computational efficiency and interpretability are important; suitable for traditional speech tasks.
- **Pitch/Formants:** For tasks involving prosody, emotion, or phoneme-level analysis.
- **Spectrogram-based Learned Features:** For large-scale, data-driven applications where deep learning models can leverage raw audio representations.
- Consider the task requirements, available data, and computational resources when selecting features.

ASR Evolution

Automatic Speech Recognition (ASR) has undergone significant evolution over the past decades. The progression of core technologies is summarized
below:

- **GMM-HMM (Gaussian Mixture Model - Hidden Markov Model):**
  - Early ASR systems relied on statistical models.
  - GMMs modeled acoustic features.
  - HMMs captured temporal dynamics.
  - *Key paper: Rabiner, 1989.*
- **DNN-HMM (Deep Neural Network - Hidden Markov Model):**
  - Deep learning enabled DNNs to replace GMMs for acoustic modeling.
  - Significant improvement in recognition accuracy.

ASR Evolution (cont.)



A block diagram for MFCC computation: From audio signal to MFCC features.

Demonstration (Japanese, Google voice search)

Sometimes working, sometimes failed

Reference)
I want to go to the CMU campus

Recognition result)

I want to go to the gym you can

Sometimes working, sometimes failed

Reference)
CMU□学のキャンパスに□きたいです

Recognition result)

CMU洋楽のキャンパスに□きたいです

Evaluation metric 1

- Sentence error rate
  - An entire sentence (utterance) is correct or not (100% error in the case below)

```
Reference)
I want to go to the CMU
campus Recognition result)
I want to go to the gym you can
```

  - Too strict, and needs to consider some local correctness

Evaluation metric 2

- Word error rate (WER)
  - Using edit distance word-by-word:

```
Reference)
I want to go to the CMU
campus Recognition result)
I want to go to the gym you can
```

- # inser(on errors = 1, # subs(tu(on errors = 2, # of dele(on errors = 0 ➡ Edit distance = 3
- Word error rate (%): Edit distance (=3) / # reference words (=8) * 100 = 37.5%
- How to compute WERs for languages that do not have word boundaries?
  - Chunking or using character error rate

Evaluation metric 3

- Character error rate (CER)
  - Using edit distance character-by-character:

Reference)
CMU□学のキャンパスに□きたいです

Recognition result)

CMU洋楽のキャンパスに□きたいです

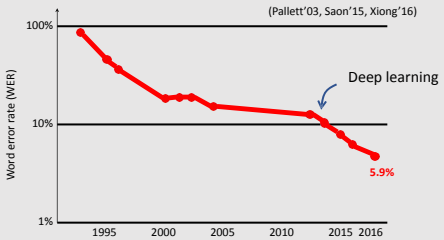
- # inser(on errors = 0, # subs(tu(on errors = 2, # of dele(on errors = 0
  - ➡ Edit distance = 2
- Character error rate (%): Edit distance (=2) / # reference words (=18) * 100 = 11.1%

Evaluation metric

- Other metrics
  - Phoneme error rate (need a pronunciation dictionary)
  - Frame error rate (need an alignment)
- NIST Speech Recognition Scoring Toolkit (SCTK)
  - ftp://jaguar.ncsl.nist.gov/pub/sctk-2.4.10-20151007-1312Z.tar.bz2
- WER can be > 100%, insertion case, deletion case

Speech recognition research is easy (?)

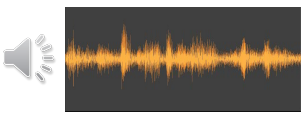
- We have WER or CER
- Objec(ve, easy to obtain, very applica(on-specific, single objec(ve
- This can show the clear progress of technologies
- This would be one reason that the effec(veness of deep learning was first shown in speech



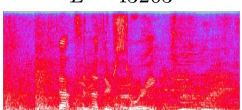
- Speech recognition demo & Evaluation metrics
- Standard speech recognition pipeline
- (a bit) mathematical formulation of speech recognition

Speech Recognition

- Mapping ***physical signal sequence*** to ***linguistic symbol sequence***



$$X = \{x_l \in \mathbb{Z} | l = 1, \ldots, L\}$$
$$L = 43263$$

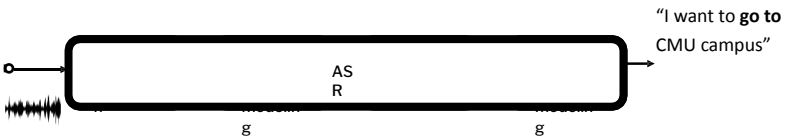


$$X = \{\mathbf{x}_t \in \mathbb{C}^D | t = 1, \ldots, T\}$$
$$T = 268$$

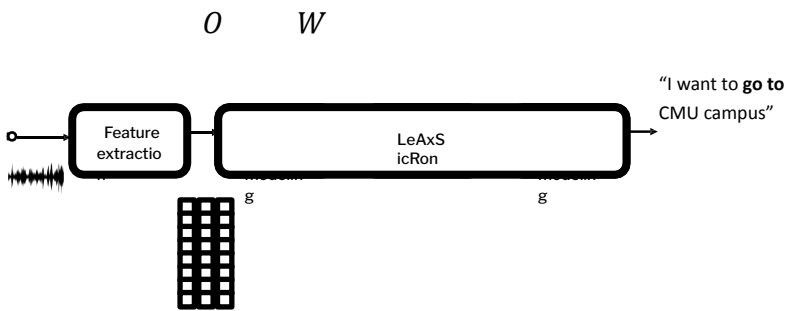
“That’s another story”

$$W = \{w_n \in \mathcal{V} | n = 1, \ldots, N\}$$
$$N = 3$$

Automatic speech recognition



Automatic speech recognition

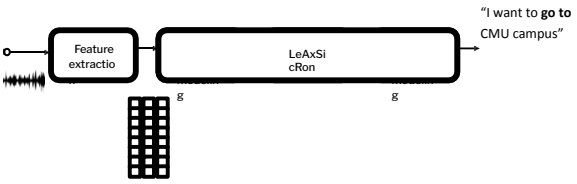


– Instead of star(ng from the waveform, we will oRen start from **speech features** (MFCC, etc.) through the **feature extrac-on** module
– Let's think of the conversion from speech feature O to text W

Speech recognition with a probabilistic formulation

- **MAP decision theory**: EsImate the most probable word sequence $\hat{W}$ among all possible word sequences W (I'll omit the domain someImes)

$$\hat{W} = \underset{W \in \mathcal{W}}{\operatorname{argmax}} p(W \mid O)$$



Probabilistic rules

- **Product rule**

$$p(x|y)p(y) = p(x, y)$$

- **Sum rule**

$$p(y) = \sum_{x} p(x, y)$$

- **Condi0onal independence assump0on**

$$p(x|y, z) = p(x|z) \qquad p(x, y|z) = p(x|z)p(y|z)$$

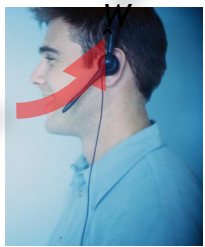
How to obtain the posterior $p(W \mid O)$

- Noisy channel model
  - Regarding O as a probabilistic variable (noisy observation)
  - Use the product rule

$$\underset{W}{\operatorname{argmax}}\, p(W \mid O) = \underset{W}{\operatorname{argmax}}\, \frac{p(O \mid W)p(W)}{p(O)}$$

$$= \underset{W}{\operatorname{argmax}}\, p(O \mid W)p(W)$$

Likelihood Prior



How to obtain the posterior $p(W \mid O)$

- Noisy channel model

$$\underset{W}{\operatorname{argmax}}\, p(W \mid O) = \underset{W}{\operatorname{argmax}}\, \frac{p(O \mid W)p(W)}{p(O)}$$

$$= \underset{W}{\operatorname{argmax}}\, p(O \mid W)p(W)$$

- Solving generaRng process of noisy observaRons!!
- SRll difficult to deal with them….

Speech <-> Text

Speech O:

Text W: I want to go to the CMU campus

Speech <-> Phoneme <-> Text

Speech O:

Phoneme L: AY W AA N T T UW G OW T UW DH AH S IY EH M Y UW K AE M P AH S

Text W: I want to go to the CMU campus

How to obtain the posterior $p(W|O)$

- Further factorize the model with **phoneme**
  - Let $L = (l_i \in \{/\text{AA}/, /\text{AE}/, \cdots\} | i = 1, \cdots, J)$ be a phoneme sequence

$$\arg\max_W p(W|O)$$

How to obtain the posterior $p(W|O)$

- Further factorize the model with **phoneme**
  - Let $L = (l_i \in \{/\text{AA}/, /\text{AE}/, \cdots\} | i = 1, \cdots, J)$ be a phoneme sequence

$$\arg\max_W p(W|O) = \arg\max_W \sum_L p(W, L|O)$$

Sum rule

How to obtain the posterior $p(W|O)$

- Further factorize the model with **phoneme**
  - Let $L = (l_i \in \{/\text{AA}/, /\text{AE}/, \cdots\} | i = 1, \cdots, J)$ be a phoneme sequence

$$\arg\max_W p(W|O) = \arg\max_W \sum_L p(W, L|O) \quad \text{Sum rule}$$

$$= \arg\max_W \sum_L \frac{p(O|W, L)p(L|W)p(W)}{p(O)} \quad \text{Product rule}$$

How to obtain the posterior $p(W|O)$

- Further factorize the model with **phoneme**
  - Let $L = (l_i \in \{/\text{AA}/, /\text{AE}/, \cdots\} | i = 1, \cdots, J)$ be a phoneme sequence

$$\arg\max_W p(W|O) = \arg\max_W \sum_L p(W, L|O) \quad \text{Sum rule}$$

$$= \arg\max_W \sum_L \frac{p(O|W, L)p(L|W)p(W)}{p(O)} \quad \text{Product rule}$$

$$= \arg\max_W \sum_L p(O|W, L)p(L|W)p(W)$$

Ignore $p(O)$ as it does not depend on W

How to obtain the posterior $p(W|O)$

- Further factorize the model with **phoneme**
  - Let $L = (l_i \in \{/\text{AA}/, /\text{AE}/, \cdots\} | i = 1, \cdots, J)$ be a phoneme sequence

$$\arg\max_W p(W|O) = \arg\max_W \sum_L p(W, L|O) \quad \text{Sum rule}$$

$$= \arg\max_W \sum_L \frac{p(O|W, L)p(L|W)p(W)}{p(O)} \quad \text{Product rule}$$

$$= \arg\max_W \sum_L p(O|W, L)p(L|W)p(W) \quad \text{Ignore } p(O) \text{ as it does not depend on } W$$

$$= \arg\max_W \sum_L p(O|L)p(L|W)p(W) \quad \text{Conditional independence assumption}$$

Noisy channel

$$\arg\max_{W} p(W|O) = \arg\max_{W} p(O|W)\, p(W)$$

$$\approx \arg\max_{W} \sum_{L} p(O|L)\, p(L|W)\, p(W)$$

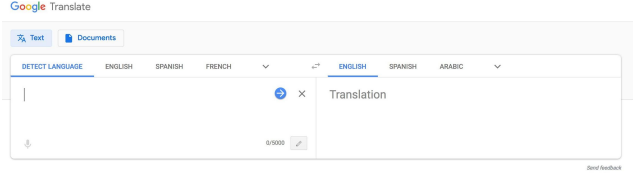
- **Speech recogni4on**
  - $p(O|L)$ AcousRc model (Hidden Markov model)
  - $p(L|W)$ Lexicon
  - $p(W)$ Language model (n-gram)

Noisy channel model

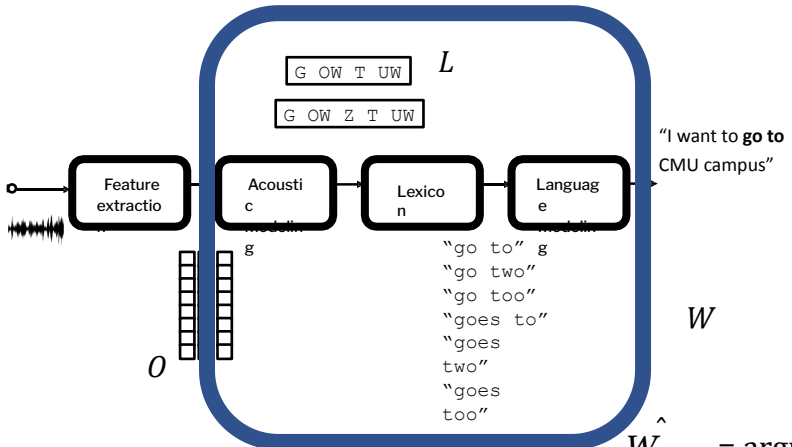
W: Target language text
Y: Source language text

$$\arg\max_{\#} p(W|Y) = \arg\max_{\#} p(Y|W)\ p(W)$$

- **Machine transla4on**
  - $p(\ |\ Y)W$: TranslaRon model
  - $p(W)$: Language model



Speech recognition pipeline



$$\hat{W} = \text{argmax}_{W \in W} p(W|O)$$

Speech recognition pipeline

$$\underset{W}{\operatorname{argmax}} \sum_{L} p(O|L)p(L|W)p(W)$$

$p(O|L)$

$p(L|W)$

$p(W)$

Please remember the noisy channel model

- **Factorization**
- **Conditional independence (Markov) assumptions**

We can elegantly factorize the speech recognition problem with a reasonable subproblem

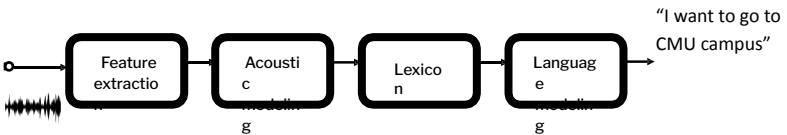
In the rest of courses, I'll introduce

- A bit further details of acoustic, lexicon, and language models
  - More and more factorization, conditional independence assumptions
- I'll skip these parts. If you want to know more about details, please check some other materials or take 11-751 "Speech Recognition and Understanding"
- For example, In 11-751 "Speech Recognition and Understanding", I'll spend 30% of entire semester for this problem

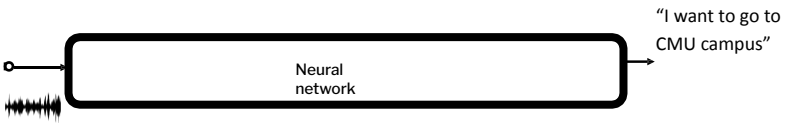
the

Table of Contents

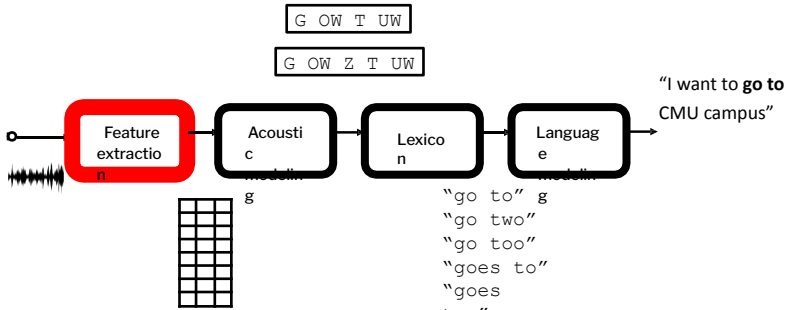
Main blocks


Feature extraction
Acoustic modeling
Lexicon
Language modeling
“I want to go to CMU campus”

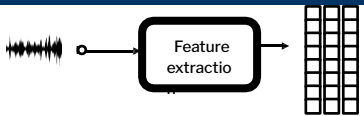
Main blocks (end-to-end ASR)



Speech recognition pipeline



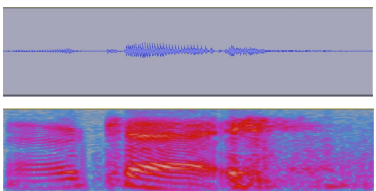
Waveform to speech feature



- Performed by so-called **feature extraction** module
  - Mel-frequency cepstral coefficient (MFCC), Perceptual Linear Prediction (PLP) used for **Gaussian mixture model (GMM)**
  - Log Mel filterbank used for **deep neural network (DNN)**
- Time scale
  - 0.0625 milliseconds (16kHz) to 10 milliseconds
- Type of values
  - Scalar (or discrete) to 12-dimensional vector
- **Mostly language-independent process**
  - Some languages use special features, e.g., pitch in Mandarin

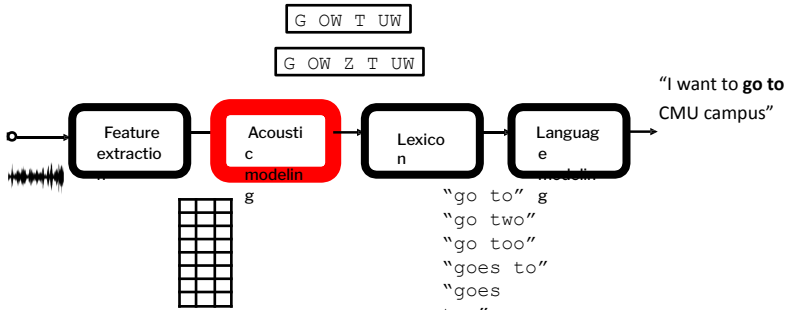
What kind of representation are desired?

- Need to preserve phoneRc/linguisRc informationon
- While suppressing irrelevant informaRon (speakers and noises)
- Befer to consider the compaRbility with backend modules

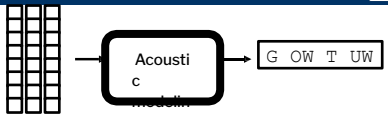


- Perceptual Linear PredicRon (PLP) or mulRlayer perceptron tandem (MLP-Tandem)
- Learnable frontend (CNN)
- Self-supervised learning (HuBERT)

Speech recognition pipeline



Speech feature to phoneme



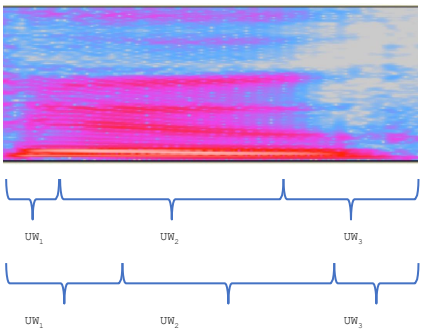
- Performed by so-called **acoustic modeling** module
  - Hidden Markov model (HMM) with **GMM** as an emission probability function
  - Hidden Markov model (HMM) with **DNN** as an emission probability function
- Time scale
  - 10 milliseconds to ~100 milliseconds (depending on a phoneme)
- Type of values
  - 12-dimensional continuous vector to 50 categorical value (~6bit)
- **Mostly language independent**
  - Map of the speech feature (language independent) to phoneme
- It can be a probability of possible phoneme sequences, e.g.,

G OW T UW or G OW with some scores

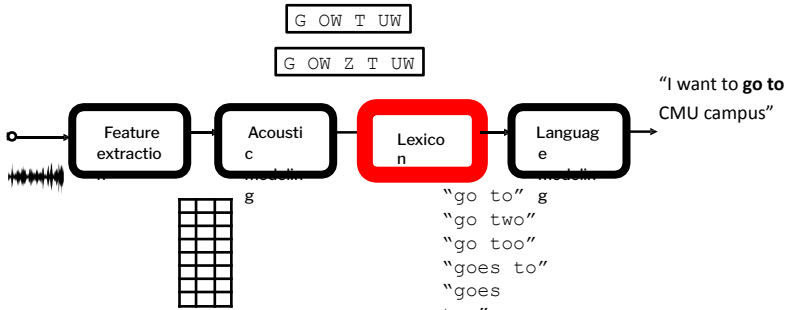
AcousBc model $p(O|L)$

- O and L are **different lengths**
- **Align** speech features and phoneme sequences by using HMM

- Provide $p(\quad | \quad) O \; L$ based on this alignment and model
- The most important problem in speech recognition



Speech recognition pipeline



Phoneme to word

- Performed by **lexicon** module
  - American English: CMU dictionary
- Time scale
  - 100 milliseconds (depending on a phoneme) to 1 second (depending on a word and also language)
- Type of values
  - 50 categorical value (~6bit) to 100K categorical value (~2Byte)
- **Language dependent**
- It can be **multiple** word sequences (one to many)

Lexicon $p(L|W)$

- Basically use a pronunciation dictionary, and map a word to the corresponding phoneme sequence
  - with the probability = 1.0 when single pronunciation
  - with the probability = $1.0/J$ when multiple (J) pronunciations

$$p(L \mid W) = p(/\mathrm{T}/, /\mathrm{OW}/ \mid \text{"two"}) = 1.0$$

What is phone and phoneme???

GO TO: "g oʊ t u" or "G OW T UW"

- Phone: g oʊ t u
  - Devised by International Phonetic Association
  - Physical categorization of speech sound
  - Not applicable to all languages, needs special characters, too many variations

- Phoneme: one of the units that distinguish one word from another in a **particular language**
  - /r/ and /l/ are degenerated in some languages (e.g., "rice" and "lice" sounds same for me!). Then, we don't have to distinguish them.
  - ARPAbet: `G OW T UW`
    - Proposed by ARPA for the development of speech recognition of only "American English"
    - Represented by ASCII characters

Pronunciation dictionary

- CMU dictionary
  - http://www.speech.cs.cmu.edu/cgi-bin/cmudict

”I want to go to the CMU campus”

AY W AA N T T UW G OW T UW DH AH S IY EH M Y UW K AE M P AH S

- Powerful, but limited
- Out of vocabulary issue, especially new word

  → Grapheme2Phoneme mapping based on machine learning

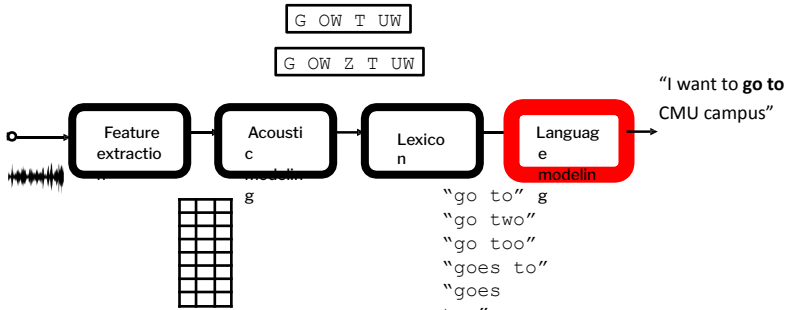
Let’s play the CMU dicBonary!

- Access: hXp://www.speech.cs.cmu.edu/cgi-bin/cmudict
- Find some in-vocabulary words
- Find five out-of-vocabulary words

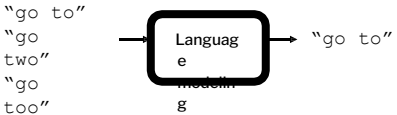
Multilingual phone dictionary

- hXps://en.wikIonary.org/wiki/WikIonary:Main_Page

Speech recognition pipeline

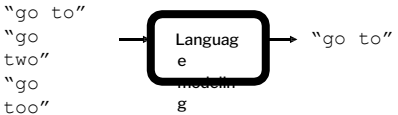


- **Performed by language modeling module $p(W)$**
  - N-gram
  - Recurrent neural network language model (RNNLM)
- **From training data, we can basically find how possibly "to", "two", and "too" will be appeared a_er "go"**
  - Part of WSJ training data, 37,416 uferances
    - "go to": **51** (mes
    - "go two":
    - "go too":

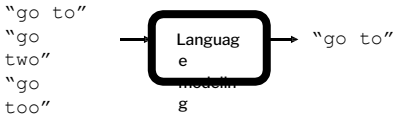


THE WALL STREET JOURNAL.

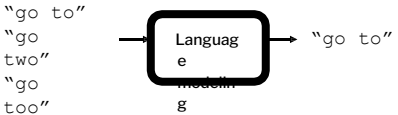
- **Performed by language modeling module $p(W)$**
  - N-gram
  - Recurrent neural network language model (RNNLM)
- **From training data, we can basically find how possibly "to", "two", and "too" will be appeared after "go"**
  - Part of WSJ training data, 37,416 utterances
    - "go to": **51** times
    - "go two": **0** times
    - "go too": **0** times



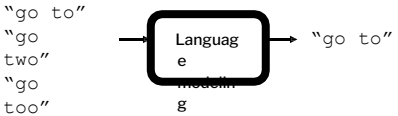
- **Performed by language modeling module $p(W)$**
  - N-gram
  - Recurrent neural network language model (RNNLM)
- **From training data, we can basically find how possibly "to", "two", and "too" will be appeared after "go"**
  - WSJ all text data, 6,375,622 sentences
    - "go to": **2710** times
    - "go two":
    - "go too":



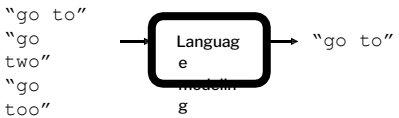
- **Performed by language modeling module $p(W)$**
  - N-gram
  - Recurrent neural network language model (RNNLM)
- **From training data, we can basically find how possibly "to", "two", and "too" will be appeared a_er "go"**
  - WSJ all text data, 6,375,622 sentences
    - "go to": **2710** (mes
    - "go two": **2** (mes, e.g., "those serving shore plants o7en go two hundred miles or more"
    - "go too":



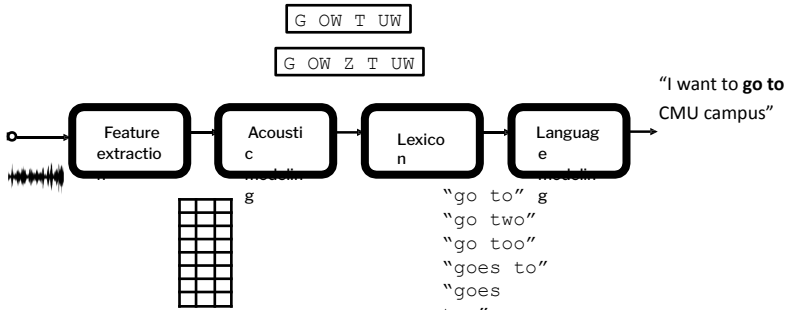
- **Performed by language modeling module $p(W)$**
  - N-gram
  - Recurrent neural network language model (RNNLM)
- **From training data, we can basically find how possibly "to", "two", and "too" will be appeared after "go"**
  - WSJ all text data, 6,375,622 sentences
    - "go to": **2710** times
    - "go two": **2** times, e.g., "those serving shore plants often go two hundred miles or more"
    - "go too": **41** times, e.g., "he could go too far"



- **Performed by language modeling module $p(W)$**
  - N-gram
  - Recurrent neural network language model (RNNLM)
- **From training data, we can basically find how possibly "to", "two", and "too" will be appeared a_er "go"**
  - WSJ all text data, 6,375,622 sentences
    - "go to": **2710** (mes
    - "go two": **2** (mes, e.g., "those serving shore plants o7en go two hundred miles or more"
    - "go too": **41** (mes, e.g., "he could go too far"

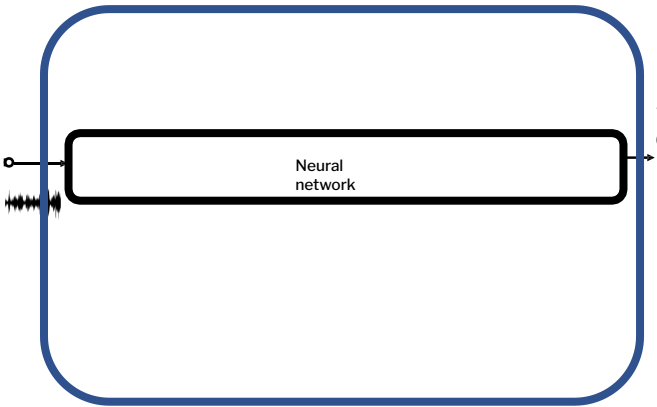


Building speech recognition is really difficult...



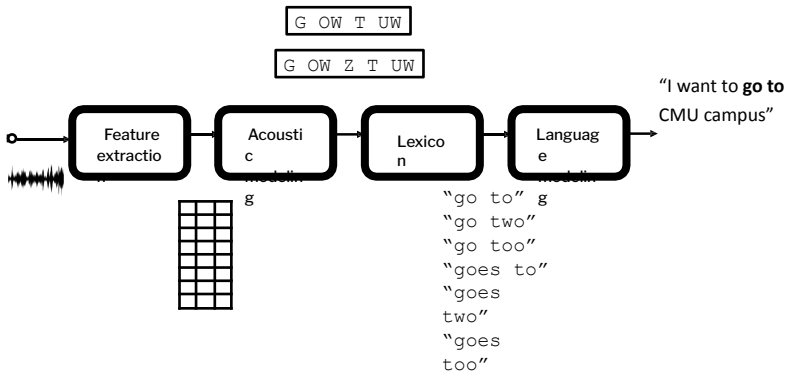
- We need to develop all components
- Each component requires a lot of background knowledge
- We need to tune hyper-parameters in each module

End-to-end speech recogniBon

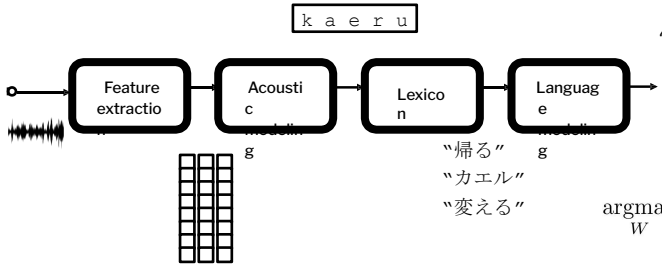


Simple, but not perfect soluAon for mulAlingual ASR

How to apply it to the other language?



How to apply it to the other language?



- We can just change the lexicon and language models (dic\onary and text only data)
- Not easy for end-to-end ASR (we need parallel data)

Summary

- Speech recognilon
  - Well defined problem (input: sound, output: text, evaluaRon metric)
  - The problem is well factorized with 1) feature extracRon, 2) acousRc model, 3) lexicon, and 4) language model
  - 1 and 2 are mostly language intendent while 3 and 4 are language dependent

ASR Evolution (cont.)

Summary:

- ASR systems have shifted from statistical models to deep learning-based approaches.
- Each stage in the evolution brought significant improvements in accuracy and robustness.
- End-to-End models simplify the pipeline and enable direct optimization for transcription tasks.

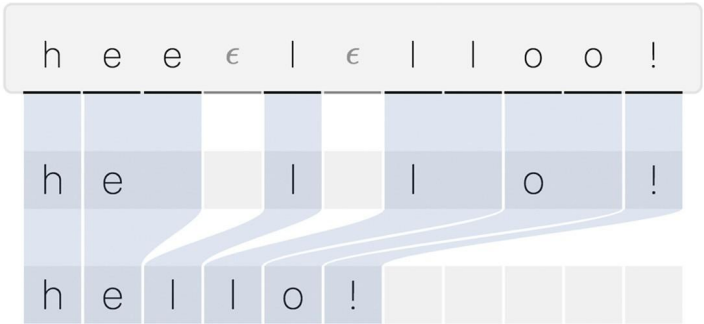
CTC: Solving the Alignment Problem in Sequence-to-Sequence Tasks

Many sequence tasks (e.g., speech-to-text) face the **alignment problem**: input and output sequences are of different lengths and not aligned. CTC enables training without explicit alignment.

- **Input:** Sequence $\mathbf{x} = (x_1, x_2, \ldots, x_T)$
- **Output:** Sequence $\mathbf{y} = (y_1, y_2, \ldots, y_U)$
- **Challenge:** $T \not= U$, unknown alignment

CTC introduces a special blank token (∅) and allows repeated labels. The final output is obtained by collapsing repeats and removing blanks.

Connectionist Temporal Classification (CTC) (cont.)



CTC: Blank token insertion and collapsing.

Connectionist Temporal Classification (CTC) (cont.)

CTC Loss Function:

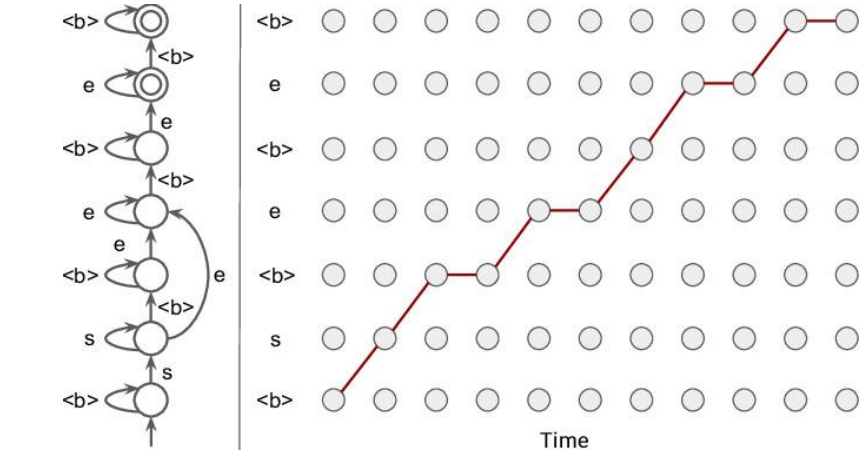
$$L_{CTC} = -\ln \sum_{\pi \in B^{-1}(\mathbf{y})} \prod_{t=1}^{T} p(\pi_t|\mathbf{x}) \quad (4)$$

- π: a possible alignment (path) with blanks and repeats
- $B^{-1}(\mathbf{y})$: set of all paths that collapse to $\mathbf{y}$
- $p(\pi_t|\mathbf{x})$: probability of label π_t at time t

CTC Decoding:

1. Predict a sequence of labels (including blanks) for each time step.
2. Collapse repeated labels and remove blanks to get the final output

Connectionist Temporal Classification (CTC) (cont.)



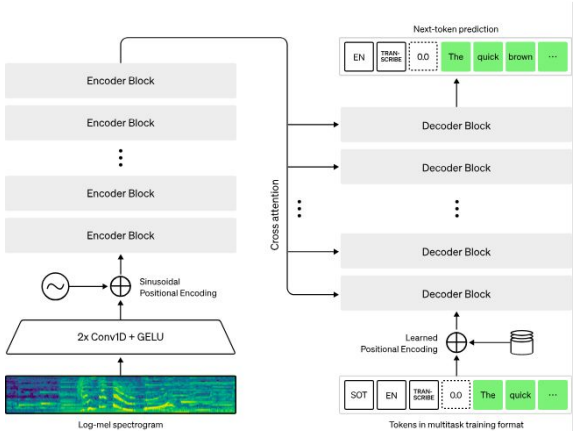
CTC alignment: mapping input frames to output tokens via blank and repeat insertion.

Seq2Seq ASR with Attention

Encoder–Decoder Overview:

- **Encoder:** Processes the input acoustic features (e.g., Mel-spectrograms) and encodes them into a sequence of hidden representations.
- **Decoder:** Generates the output transcription, one token at a time, conditioned on the encoder outputs and previous decoder states.
- **Attention Mechanism:** Allows the decoder to focus on relevant parts of the input sequence at each decoding step, improving alignment and performance.

Seq2Seq ASR with Attention (cont.)



Seq2Seq ASR with Attention: Encoder-Decoder Architecture

Attention Alignment:

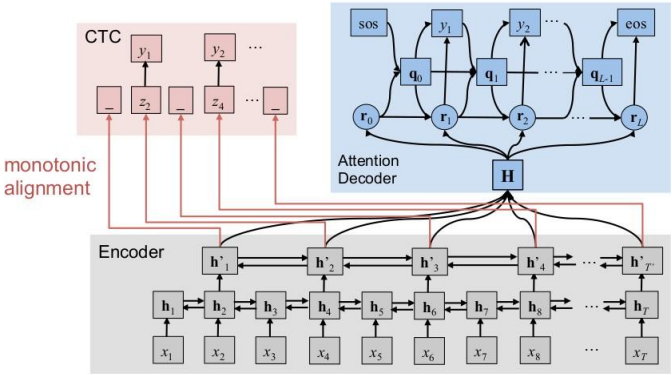
At each decoder time step t, the attention mechanism computes a context vector as a weighted sum of encoder outputs. The weights $a_{t,s}$ represent the alignment between decoder step t and encoder position s:

$$a_{t,s} = \frac{\exp(e_{t,s})}{\sum_{s'} \exp(e_{t,s'})} \tag{5}$$

where $e_{t,s}$ is the attention score (e.g., computed via a feedforward network) between decoder state at time t and encoder output at position s.

Seq2Seq ASR with Attention (cont.)

Multitask learning: $\mathcal{L}_{\text{MTL}} = \lambda \mathcal{L}_{\text{CTC}} + (1-\lambda)\mathcal{L}_{\text{Attention}}$



CTC guides attention alignment to be monotonic

Visualization of Attention Alignment

Seq2Seq ASR with Attention (cont.)

Example: Listen, Attend and Spell (LAS) on "hello world"

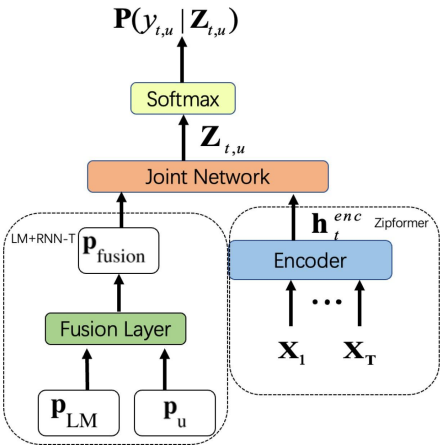
- The encoder processes the audio features for the phrase "hello world".
- At each decoding step, the attention mechanism aligns the decoder to the relevant audio frames.
- The decoder outputs the transcription character by character (or subword by subword), e.g., "h", "e", "l", "l", "o", " ", "w", "o", "r", "l", "d".

RNN-Transducer (RNN-T): Overview

RNN-Transducer (RNN-T) is a sequence-to-sequence model widely used for end-to-end speech recognition. It extends the CTC (Connectionist Temporal Classification) approach by modeling both input and output dependencies:

- **Sequence-to-sequence model:** Maps input sequences (e.g., acoustic features) directly to output sequences (e.g., transcriptions).
- **End-to-end speech recognition:** Eliminates the need for separate components (like acoustic, pronunciation, and language models).
- **Extension of CTC:**
  - CTC models only input dependencies and assumes output tokens are conditionally independent.
  - RNN-T models both input and output dependencies, allowing the

RNN-Transducer (RNN-T): Overview (cont.)



RNN-T Architecture: Encoder, Prediction Network, and Joint Network.

RNN-Transducer (RNN-T): Overview (cont.)

Architecture Components:

- **Encoder:** Processes the input acoustic features and produces high-level representations.
- **Prediction Network:** Acts like a language model, conditioning on previous non-blank output tokens.
- **Joint Network:** Combines encoder and prediction network outputs to produce logits over the output vocabulary (including the blank symbol).

$$\mathbf{h}_t = \text{Encoder}(\mathbf{x}_{1:t}) \tag{6}$$

$$\mathbf{g}_u = \text{PredictionNetwork}(y_{1:u-1}) \tag{7}$$

$$\mathbf{z}_{t,u} = \text{JointNetwork}(\mathbf{h}_t, \mathbf{g}_u) \tag{8}$$

RNN-Transducer (RNN-T): Overview (cont.)

Loss Overview:

- RNN-T loss marginalizes over all possible alignments between input and output sequences.
- Similar to CTC, but allows for output dependencies.
- Computed efficiently using dynamic programming.

$$L_{\text{RNN-T}} = -\log \sum_{\pi \in A(\mathbf{y}, T)} P(\pi|\mathbf{x}) \tag{9}$$

where $A(\mathbf{y}, T)$ is the set of all valid alignments.

RNN-Transducer (RNN-T): Overview (cont.)

Pros vs. CTC:

- Models output dependencies (better language modeling).
- No need for external language model during inference.
- More flexible alignments.

Cons vs. CTC:

- More complex and computationally intensive.
- Harder to train and tune.
- Decoding is slower due to output dependencies.

Transformer & Conformer in ASR

Transformer & Conformer in ASR

- **Transformers** and **Conformers** are state-of-the-art architectures for Automatic Speech Recognition (ASR).
- Both leverage self-attention to model long-range dependencies in audio sequences.
- Conformer extends Transformer by integrating convolutional modules for local feature extraction.
- Key components include:
  - Multi-head self-attention
  - Feed-forward networks
  - Residual connections and

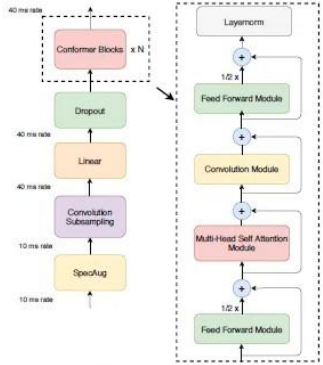


Figure 1: ***Conformer encoder model architecture.*** *Conformer comprises of two macaron-like feed-forward layers with half-step residual connections sandwiching the multi-headed self-attention and convolution modules. This is followed by a post layernorm.*

Transformer Encoder: Core Components

- **Multi-Head Self-Attention:**
  - Captures global context by attending to all positions in the input sequence.
  - Enables parallel processing of sequence data.
- **Feed-Forward Network (FFN):**
  - Applies non-linearity and projection to each position independently.
  - Typical form:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

- **Residual Connections & Layer Normalization:**
  - Stabilize training and improve gradient flow.

Conformer Block (Gulati et al., 2020)

- **Conformer** combines self-attention and convolution for effective speech modeling.
- **Block Structure:**
  - Feed-Forward → Multi-Head Self-Attention + Convolution → Feed-Forward
- **Key Features:**
  - **Convolution Module:**
    - Gated Linear Units (GLU)
    - Depthwise convolution for efficient local feature extraction
    - Batch normalization for stable training
  - Residual connections and layer normalization throughout

Self-Supervised Learning: Wav2Vec 2.0

wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations

Alexei Baevski **Henry Zhou** **Abdelrahman Mohamed** **Michael Auli**

{abaevski,henryzhou7,abdo,michaelauli}@fb.com

Facebook AI

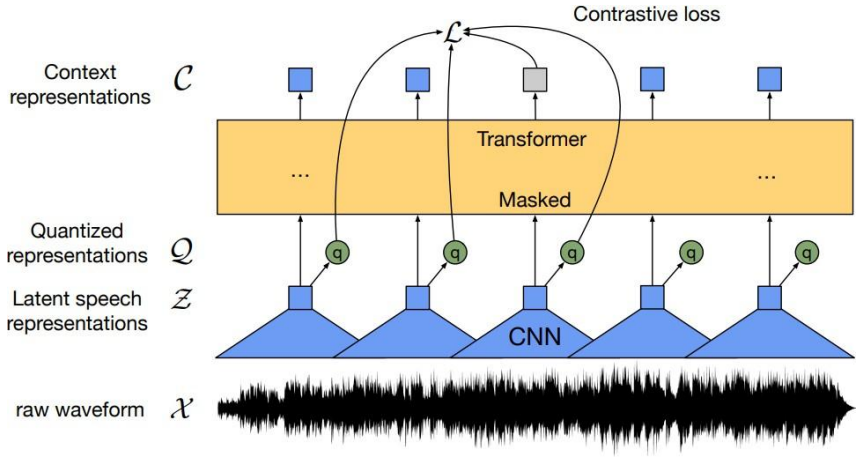
Abstract

We show for the first time that learning powerful representations from speech audio alone followed by fine-tuning on transcribed speech can outperform the best semi-supervised methods while being conceptually simpler. wav2vec 2.0 masks the speech input in the latent space and solves a contrastive task defined over a quantization of the latent representations which are jointly learned. Experiments using all labeled data of Librispeech achieve 1.8/3.3 WER on the clean/other test sets. When lowering the amount of labeled data to one hour, wav2vec 2.0 outperforms the previous state of the art on the 100 hour subset while using 100 times less labeled data. Using just ten minutes of labeled data and pre-training on 53k hours of unlabeled data still achieves 4.8/8.2 WER. This demonstrates the feasibility of speech recognition with limited amounts of labeled data.[1]

Wav2Vec 2.0: Overview (cont.)

- **Proposed by Baevski et al. (2020)**
- **Key Idea:** Self-supervised learning for speech representation.
- **Architecture:**
  - CNN encoder for raw waveform → latent representations z_t
  - Transformer for context representations c_t
  - Quantization module for discrete representations q_t

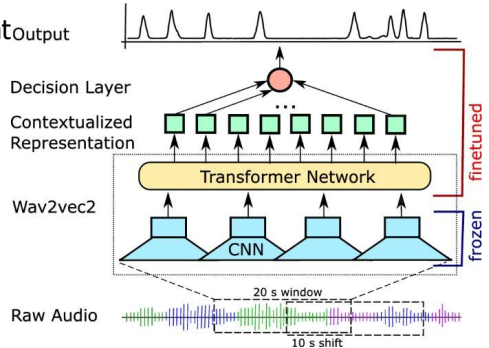
Wav2Vec 2.0: Overview (cont.)



Wav2Vec 2.0 architecture overview.

Wav2Vec 2.0: Feature Extraction

- **Step 1:** Raw audio waveform is input to a multi-layer CNN encoder.
- **Output:** Latent$_{Output}$



Wav2Vec 2.0: Contextualization and Masking

- **Step 2:** Random spans of z_t are masked.
- **Step 3:** Transformer network produces contextualized representations c_t.

$$z_t \xrightarrow{\text{mask spans}} \xrightarrow{\text{Transformer}} c_t$$

Wav2Vec 2.0: Quantization

- **Step 4:** Quantize z_t to discrete representations q_t.
- **Quantization:** Product of G codebooks, each of size V

.

$$z_t \xrightarrow{\text{Quantization}} q_t$$

Wav2Vec 2.0: Contrastive and Diversity Losses

- **Contrastive Loss:** Distinguish true quantized vector q^+ from negatives q^-.

$$L_t = -\log \frac{\exp \text{sim}(c_t, q^+)}{\sum_{q^-} \exp(\text{sim}(c_t, q^-))}$$

- **Diversity Loss:** Encourages usage of all codebook entries.

Wav2Vec 2.0: Performance

- **State-of-the-art results:**
  - ~1.8% WER with 960h labeled data
  - ~4.8% WER with only 10 minutes labeled data
- **Impact:** Enables efficient use of unlabeled speech

TIMIT phoneme recognition accuracy in terms of phoneme error rate (PER).

	dev PER	test PER
CNN + TD-filterbanks [59]	15.6	18.0
PASE+ [47]	-	17.2
Li-GRU + fMLLR [46]	–	14.9
wav2vec [49]	12.9	14.7
vq-wav2vec [5]	9.6	11.6
This work (no LM)		
LARGE (LS-960)	7.4	8.3

Fine-tuning for ASR with Wav2Vec 2.0

Fine-tuning Overview

- **Self-Supervised Learning (SSL):** Pre-train Wav2Vec 2.0 on large unlabeled audio.
- **Fine-tuning:** Use small labeled dataset for ASR.
- **Loss Function:** Connectionist Temporal Classification (CTC) Loss.
- **Goal:** Map audio features to text transcriptions.

Fine-tuning Results

- **Small labeled data:** Only 10 minutes of transcribed speech.
- **Performance:**
  - Word Error Rate (WER): 4.8% (clean) / 8.2% (other)
- **Efficient:** High accuracy with minimal supervision.

Fine-tuning Results (cont.)

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
10 min labeled						
Discrete BERT [4]	LS-960	4-gram	15.7	24.1	16.3	25.2
BASE	LS-960	4-gram	8.9	15.7	9.1	15.6
		Transf.	6.6	13.2	6.9	12.9
LARGE	LS-960	Transf.	6.6	10.6	6.8	10.8
	LV-60k	Transf.	4.6	7.9	4.8	8.2
1h labeled						
Discrete BERT [4]	LS-960	4-gram	8.5	16.4	9.0	17.6
BASE	LS-960	4-gram	5.0	10.8	5.5	11.3
		Transf.	3.8	9.0	4.0	9.3
LARGE	LS-960	Transf.	3.8	7.1	3.9	7.6
	LV-60k	Transf.	2.9	5.4	2.9	5.8
10h labeled						
Discrete BERT [4]	LS-960	4-gram	5.3	13.2	5.9	14.1
Iter. pseudo-labeling [58]	LS-960	4-gram+Transf.	23.51	25.48	24.37	26.02
	LV-60k	4-gram+Transf.	17.00	19.34	18.03	19.92
BASE	LS-960	4-gram	3.8	9.1	4.3	9.5
		Transf.	2.9	7.4	3.2	7.8
LARGE	LS-960	Transf.	2.9	5.7	3.2	6.1
	LV-60k	Transf.	2.4	4.8	2.6	4.9

Wav2Vec 2.0 + CTC Loss (PyTorch)

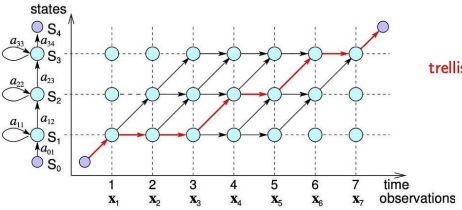
```
import  torch import
torchaudio
from  transformers  import  Wav2 Vec2 ForCTC ,                Wav 2 Vec2 Processor

processor  =  Wav 2 Vec2 Processor . from_pretrained (" facebook / wav2vec2 -
      base -960 h")
model  =  Wav 2 Vec2 ForCTC . from_pretrained (" facebook / wav2vec2 - base -960
      h")

input_audio , _ =  torchaudio . load (" audio . wav ")
inputs  =  processor( input_audio . squeeze (),  sampling_rate
      =16000 ,  return_tensors =" pt") with
torch . no_grad ():
       logits  =  model (** inputs). logits pred_ids  =  torch
. argmax ( logits ,  dim = -1)
transcription  =  processor. decode ( pred_ids [0]) print(
transcription )
```

Decoding: Beam Search / Viterbi

- **CTC Decoding:**
  - Greedy decoding: Select most probable token at each timestep.
  - Beam search: Explore multiple hypotheses for better accuracy.
  - Viterbi algorithm: Find most likely sequence.
- **Equation:**



$$p(\mathbf{X}, \text{path}_\ell \,|\, \boldsymbol{\lambda}) = p(\mathbf{X} \,|\, \text{path}_\ell, \boldsymbol{\lambda}) P(\text{path}_\ell \,|\, \boldsymbol{\lambda})$$

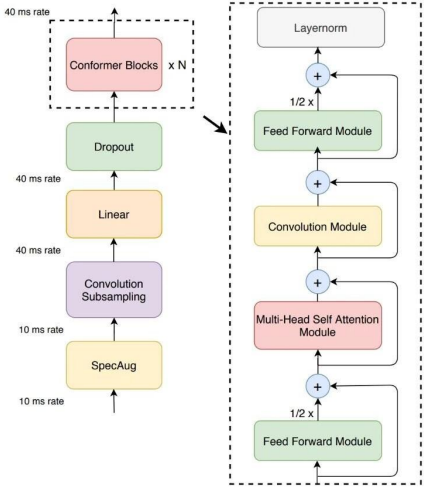
likelihood: $\sum_{\{\text{path}_\ell\}} p(\mathbf{X}, \text{path}_\ell \,|\, \boldsymbol{\lambda})$

decode: $\max_{\text{path}_\ell} p(\mathbf{X}, \text{path}_\ell \,|\, \boldsymbol{\lambda})$

Conformer-CTC: Non-autoregressive ASR

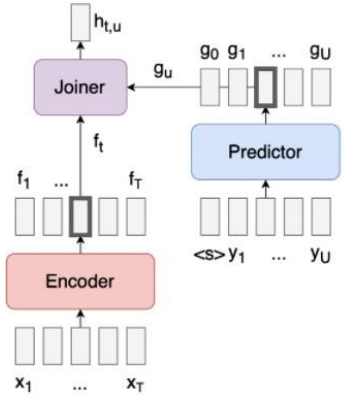
Conformer-CTC Overview

- **Conformer:** Combines convolutional and transformer layers.
- **CTC Loss:** Connectionist Temporal Classification for sequence alignment.
- **Non-autoregressive:** Processes entire input sequence in parallel.



CTC vs Transducer Models

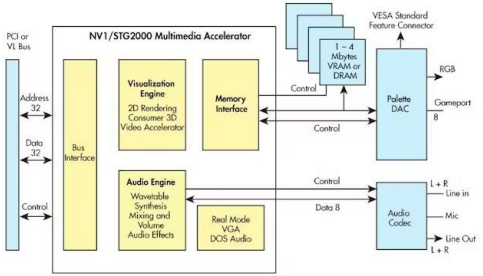
- **CTC (Connectionist Temporal Classification):**
  - Aligns input and output sequences without explicit segmentation.
  - Suitable for non-autoregressive models.
  - Decodes output in parallel.
  - Simpler and faster inference.
- **Transducer Models:**
  - Jointly model alignment and output prediction.
  - Typically autoregressive.
  - Better for streaming and online decoding.
  - More complex, higher accuracy in some cases.



An Overview of Transducer Models for ASR.

Conformer-CTC: NVIDIA Implementations

- **NVIDIA Riva**: Production-grade ASR toolkit.
- **Parakeet**: Efficient Conformer-CTC variant.
- **Citrinet**: Convolutional CTC model for fast inference.
- Optimized for GPU acceleration.
- Supports real-time and batch processing.



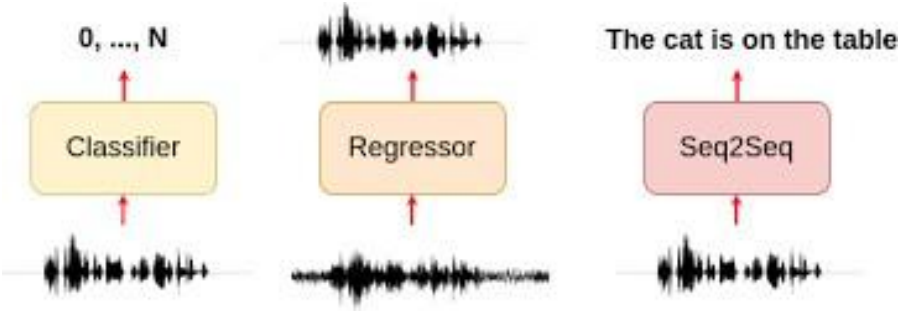
Source: NVIDIA Docs, arXiv

Tech Science: Conformer-CTC Advances

- **Conformer:** Integrates self-attention and convolution.
- **CTC Loss:** Enables parallel decoding.
- **Variants:** Parakeet, Citrinet, and others.
- **Performance:** State-of-the-art on LibriSpeech, CommonVoice.
- **References:**
  - Conformer Paper (arXiv)

Streaming ASR with SpeechBrain

- **SpeechBrain:** Open-source toolkit for speech processing.
- Supports streaming ASR with Conformer-CTC.



Summary: Non-autoregressive ASR

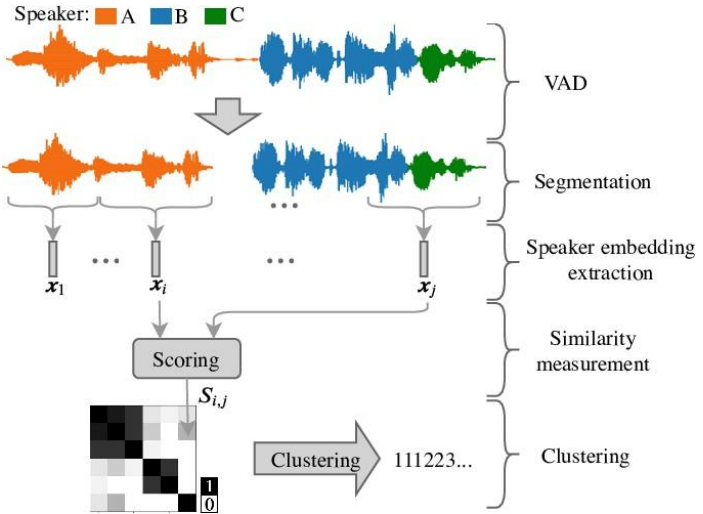
- Conformer-CTC enables fast, parallel ASR.
- CTC models are simpler and efficient.
- NVIDIA and SpeechBrain provide robust implementations.
- Streaming support is available for real-time applications.

Advanced Tasks: Diarization & Emotion Recognition

Speaker Diarization

- **Goal:** Identify ”who spoke when” in audio streams
- **Applications:**
  - Meeting transcription
  - Podcast analysis
  - Call center monitoring
- **Pipeline:**
  1. **Segmentation:** Detect speaker change points
  2. **Embedding:** Extract speaker features
  3. **Clustering:** Group segments by speaker

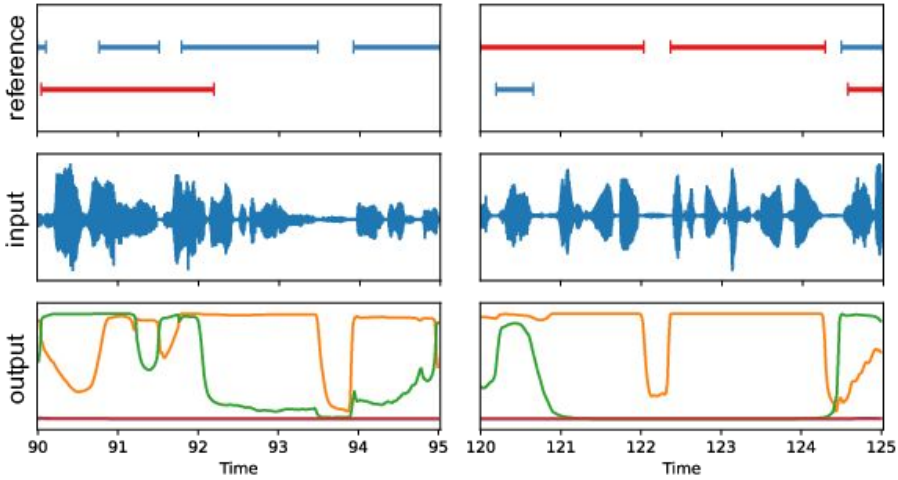
Speaker Diarization (cont.)



Speaker Diarization Tools

- **Popular Tool:** pyannote-audio
- **Features:**
  - Pre-trained diarization models
  - Easy integration with Python
  - Supports custom pipelines

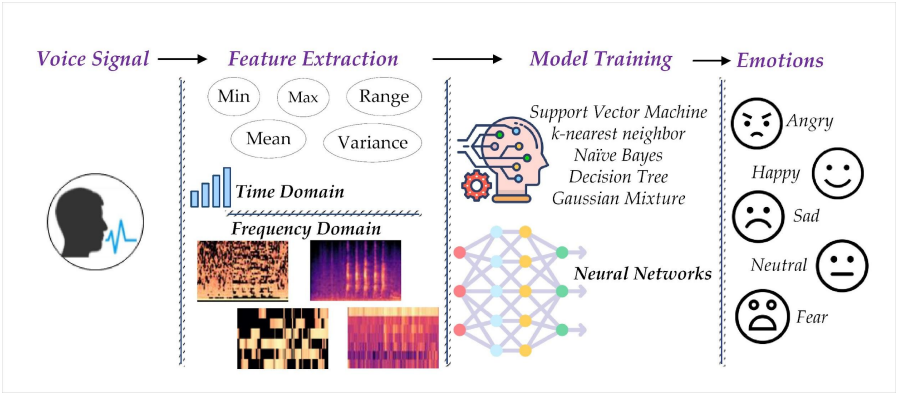
Speaker Diarization Tools (cont.)



Emotion Recognition

- **Goal:** Detect emotions from speech signals
- **Applications:**
  - Human-computer interaction
  - Mental health monitoring
  - Customer service analytics
- **Datasets:** IEMOCAP, YouTube, arXiv, PLOS

Emotion Recognition (cont.)



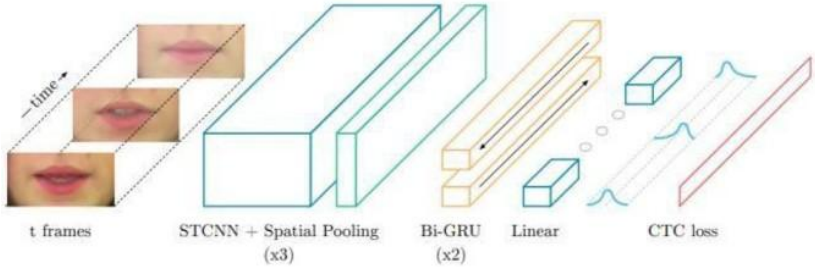
Emotion Recognition: Models

- **Wav2Vec 2.0:**
  - Self-supervised speech representation
  - Robust to noise and speaker variation
- **Neural CDEs:**
  - Continuous-time neural networks
  - Capture temporal dynamics in speech
- **Performance:**
  - Achieves ~70% accuracy on IEMOCAP

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Samples}}$$

Audio-Visual Speech Recognition (LipNet)

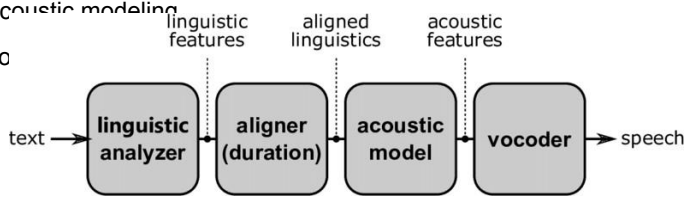
- **LipNet:**
  - Integrates lip movement and audio
  - Robust speech decoding in noisy environments
- **Benefits:**
  - Improved accuracy



Text-to-Speech (TTS) & Vocoders

Text-to-Speech (TTS) Overview

- Converts text input into human-like speech
- Applications: virtual assistants, accessibility, entertainment
- Key components:
  - Text analysis
  - Acoustic modeling
  - Vo



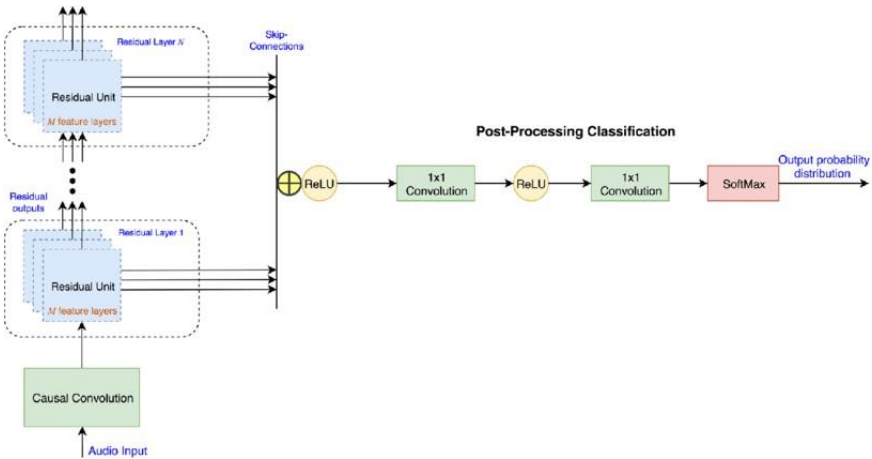
WaveNet (2016)

- Deep generative model for raw audio
- Autoregressive: predicts next sample given previous samples
- High-quality, natural-sounding speech
- Equation:

$$p(x) = \prod_{t=1} p(x_t | x_1, \ldots, x_{t-1})$$

- Computationally expensive for real-time use

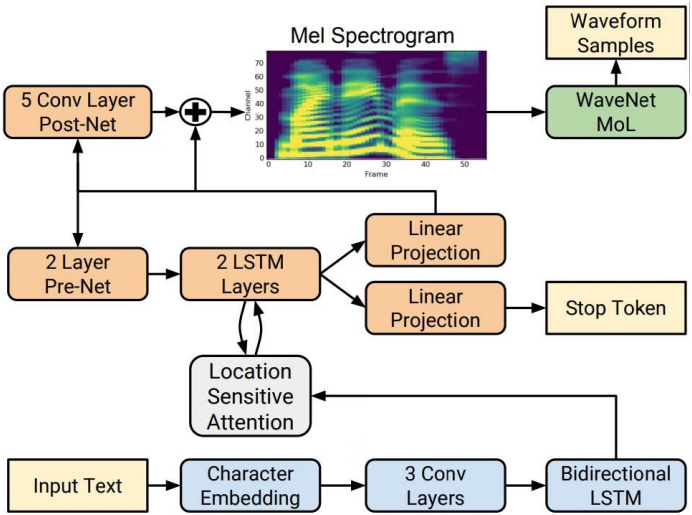
WaveNet (2016) (cont.)



Tacotron 2

- End-to-end TTS system
- Pipeline:
  - Text → Mel-spectrogram (sequence-to-sequence model)
  - Mel-spectrogram → WaveNet/GAN vocoder (audio synthesis)
- Improved prosody and pronunciation
- Flexible for different voices and languages

Tacotron 2 (cont.)



FastSpeech, HiFi-GAN, Glow-TTS, Diffusion Models

- Non-autoregressive models: faster inference

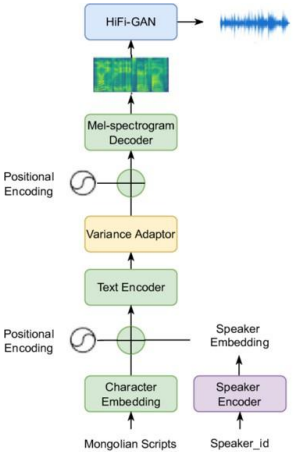
- FastSpeech:
  - Parallel generation of spectrograms
  - Duration prediction for alignment

- HiFi-GAN:
  - GAN-based vocoder
  - High-fidelity, real-time synthesis

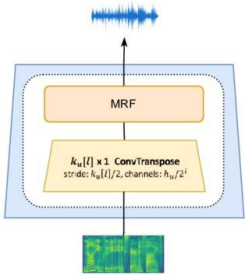
FastSpeech, HiFi-GAN, Glow-TTS, Diffusion Models (cont.)

- Glow-TTS:
  - Flow-based generative model
  - Efficient and expressive speech synthesis
- Diffusion models:
  - Iterative refinement of audio
  - State-of-the-art naturalness

FastSpeech, HiFi-GAN, Glow-TTS, Diffusion Models (cont.)



(a) Fastspeech2



(b) HiFi-GAN

Challenges & Current Research

Challenges in Audio NLP

- **Noise robustness and domain mismatch**
  Models often struggle with noisy environments and data from unseen domains.

- **SSL models need massive compute**
  Self-supervised learning approaches require significant computational resources.

- **Interpretability & data biases**
  Understanding model decisions and mitigating biases in datasets remain open problems.

- **Cross-modal fusion is still hard**

Key Takeaways

- Robust audio processing demands both signal-level and deep representation learning.
- ASR now thrives on SSL + Transformer architectures.
- Conformer combines locality and global context.
- Advanced tasks highlight the breadth of audio-NLP.
- Emerging avenues: TTS, diffusion, multimodal AI, efficient edge solutions

References

References

1 Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*.

2 Gulati, A., et al. (2020). Conformer: Convolution-augmented Transformer for Speech Recognition. *arXiv preprint arXiv:2005.08100*.

3 Van den Oord, A., Dieleman, S., Zen, H., et al. (2016). WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

4 Kim, Y., Kong, J., & Son, J. (2020). FastSpeech: Fast, Robust and Controllable Text to Speech. *arXiv preprint arXiv:2006.04558*.

5 Lopez-Moreno, I., et al. (2021). DiffWave: A Versatile Diffusion Model for Audio Synthesis. *arXiv preprint arXiv:2009.09761*.

6 SpeechBrain Documentation. https://speechbrain.readthedocs.io

7 torchaudio Documentation. https://pytorch.org/audio/stable/index.html

References (cont.)

8 NVIDIA NeMo Documentation. https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/

9 NVIDIA Riva Documentation. https://docs.nvidia.com/riva/

10 Tripathi, S., et al. (2022). Emotion Recognition from Speech using Wav2Vec2 and CDE. *arXiv preprint arXiv:2209.12345*.

11 Whisper: Multilingual Speech Recognition. https://github.com/openai/whisper

12 Jont Allen's Speech Page. http://jontalle.web.engr.illinois.edu

Credits

Dr. Prashant Aparajeya
Computer Vision Scientist — Director(AISimply Ltd)
p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.