CÔNG TY CỔ PHẦN VCCORP

4TH WEEK REPORT

LEADER: MR. NGÔ VĂN VĨ

Speech Recognition

Lê Văn Hậu

Contents

Forth Week Report

FireFly

July 11, 2025

1 General View on Speech Recognition

Speech recognition, or Automatic Speech Recognition (ASR), is the process of converting spoken language into text. It's a complex interdisciplinary field that draws from computer science, computational linguistics, and signal processing.

1.1 Representation of Speech as a Digital Sequence

The first step in any ASR system is to convert the analog speech signal (the continuous sound waves our voices produce) into a digital format that computers can process. This involves:

- **Microphone:** Converts the acoustic sound waves into an electrical (analog) signal.
- **Analog-to-Digital Converter (ADC):** This crucial component takes the continuous analog electrical signal and transforms it into a discrete digital sequence. This is done through two main processes:
  - **Sampling**
  - **Quantization**

The result is a sequence of numbers (e.g., integers or floating-point numbers), representing the amplitude of the speech waveform at discrete time points. This is known as **Pulse Code Modulation (PCM)**.

1.2 Image of Frequency and Feature Extraction

Raw audio is transformed into "frequency images" or **acoustic features** for ASR, mimicking human speech perception. This **feature extraction** process involves:

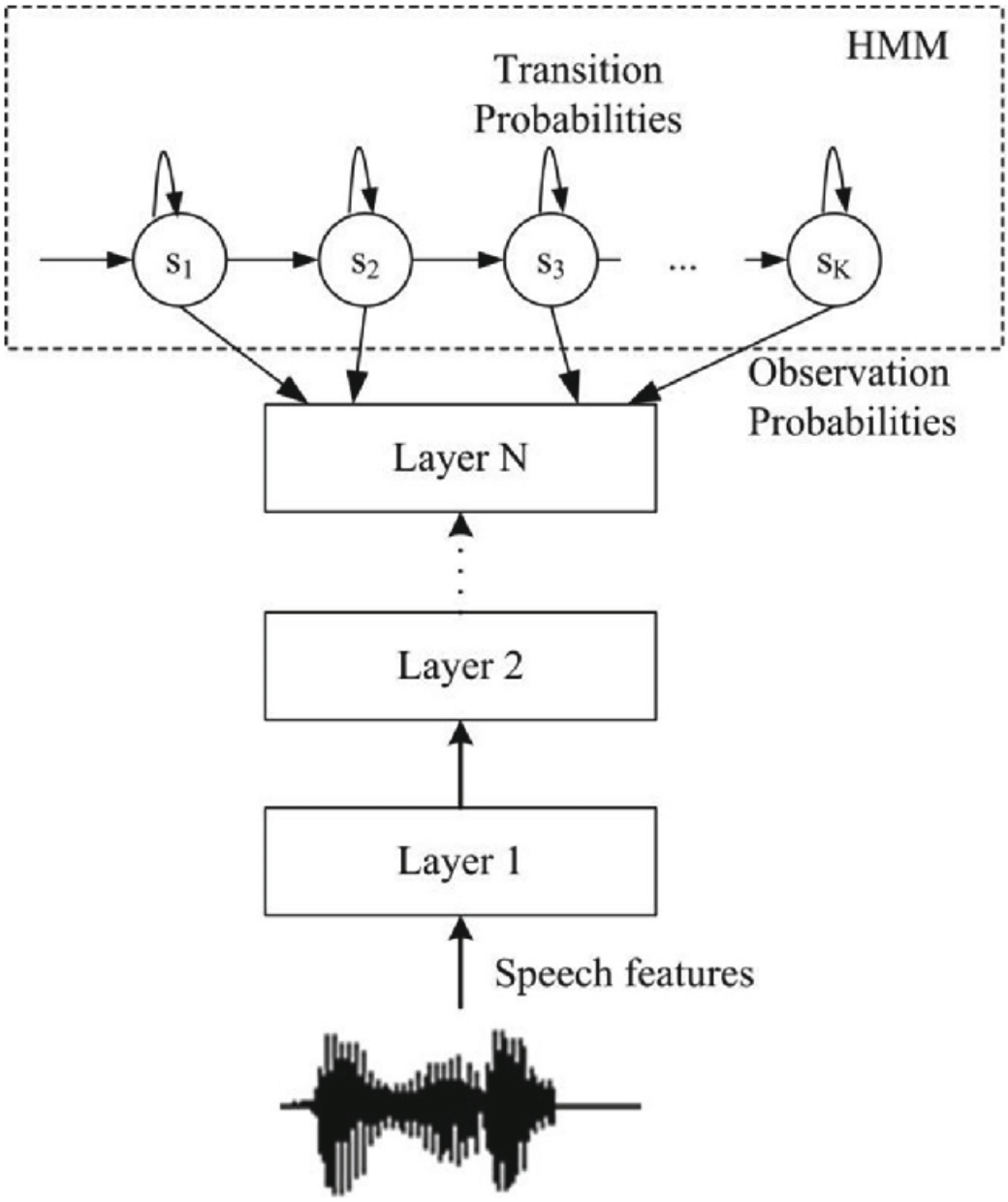
- **Framing:** Dividing the continuous signal into short, overlapping frames (e.g., 10-25ms with 50
- **Windowing:** Applying a window function (e.g., Hamming) to each frame to smooth edges and prevent Fourier transform artifacts.
- **Fourier Transform (STFT):** Computing a Fast Fourier Transform (FFT) for each frame to decompose it into constituent frequencies. Plotting these frames over time creates a **spectrogram**, a visual "image of frequency" showing how frequency content changes over time.

- **Mel-Frequency Cepstral Coefficients (MFCCs) / Log-Mel Filterbank Energies (FBANKs):** To compact spectrograms and align with human perception:
  - **Mel-scale filtering:** Frequencies are weighted by the non-linear Mel scale, applying triangular filters (Mel filterbank) to the power spectrum.
  - **Logarithmic compression:** Filter band energies are log-compressed, yielding **Log-Mel Filterbank Energies (FBANKs)**, often used directly in modern ASR.
  - **Discrete Cosine Transform (DCT):** For MFCCs, a DCT is applied to log-Mel energies to decorrelate features and compact information into fewer coefficients (typically 12-40 per frame).

The result is a sequence of **feature vectors**, each representing a short speech segment, which serves as input to the ASR model.

2 HMM-DNN Hybrid Systems

This was the state-of-the-art ASR architecture before the rise of end-to-end models. It combines the temporal modeling strengths of Hidden Markov Models (HMMs) with the powerful classification capabilities of Deep Neural Networks (DNNs).



- **How it works:** The system is composed of several distinct parts:
  - **Acoustic Model (AM):** A DNN takes acoustic features (like MFCCs) from short audio frames and predicts the probability of different phonetic states (e.g., the beginning, middle, or end of a phoneme like /t/).

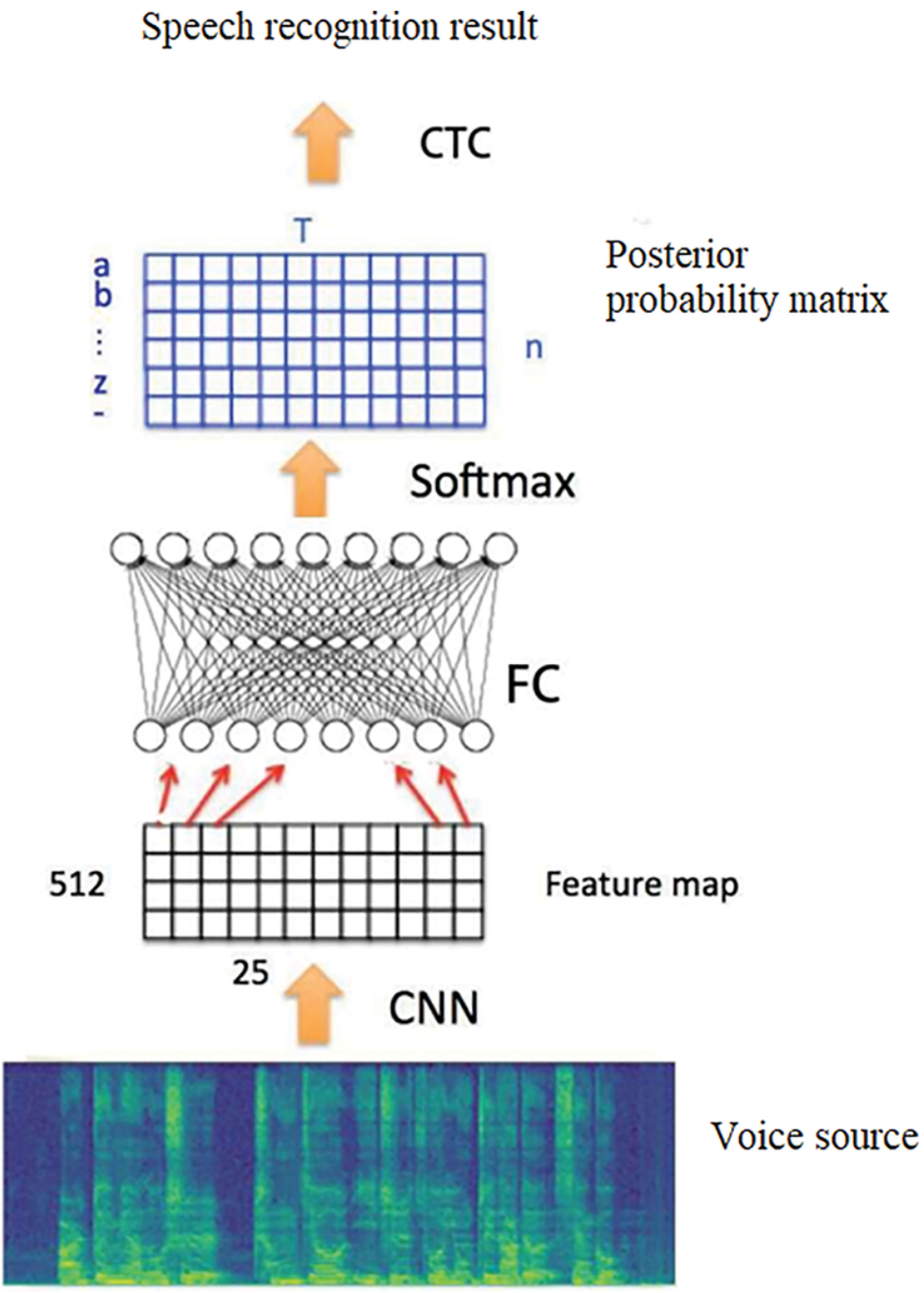
- **Hidden Markov Model (HMM):** The HMM models the temporal sequence of these phonetic states. The probabilities from the DNN serve as the HMM's "emission probabilities".
- **Pronunciation Model (Lexicon):** A dictionary that maps words to their corresponding sequences of phonemes (e.g., `CAT -> /k/ /æ/ /t/`).
- **Language Model (LM):** Typically an n-gram model that provides the probability of a sequence of words, helping to distinguish between acoustically similar but grammatically different phrases (e.g., "recognize speech" vs. "wreck a nice beach").

During decoding, a Viterbi search algorithm in probabilities and features from DNN finds the most likely sequence of words that could have generated the acoustic observations, given all the component models. Its core idea is dynamic programming on a tree.

3 Connectionist Temporal Classification (CTC)

Before CTC, speech recognition models required time-consuming and error-prone pre-alignment of audio segments to their corresponding phonemes or characters.

CTC revolutionized end-to-end Automatic Speech Recognition (ASR) by providing a loss function and decoding algorithm to train sequence-to-sequence models, like neural networks, without requiring pre-aligned datasets.


Speech recognition result
CTC
T
a
b
z
-
n
Posterior
probability matrix
Softmax
FC
512
Feature map
25
CNN
Voice source

3.1 Key Concepts

- **The Problem Solved:** CTC addresses the challenge of unknown alignments between audio frames and transcript characters, allowing the network to learn this mapping automatically.
- **Mechanism:**
  1. **Blank Token:** A special `<blank>` token is added to the vocabulary of possible outputs (e.g., `a-z`, `<blank>`).
  2. **Frame-wise Prediction:** The neural network outputs a probability distribution over all characters (including `<blank>`) for each audio frame.
  3. **Path Collapsing:** An output sequence is collapsed by first removing repeated characters and then all `<blank>` tokens. For example, `h_ee_ll<blank>l_o` collapses to `hello`.
  4. **Loss Calculation:** The CTC loss aggregates the probabilities of all possible output paths that collapse to the correct target transcript, enabling training without explicit alignments.

3.2 Example of Collapsible Paths

Consider the target transcript 'hello'. The following are some frame-by-frame output sequences (paths) that, when collapsed by the CTC rules, result in 'hello':

- `h h e l l o o` → `hello`
- `h <blank> e l <blank> l o` → `hello`
- `h e <blank> l l o o` → `hello`
- `h e l l o <blank> <blank>` → `hello`
- *and many more...*

The core idea of CTC is to reward the network if *any* of these valid paths have a high probability, as the "true" alignment is unknown.

3.3 Mathematical Formulation

For each frame t (from 1 to T), the neural network outputs a probability distribution over all possible labels in the vocabulary (including the blank token). Let $P(k_t)$ denote the probability of label k at time t.

The probability of a single, specific path $\pi = (\pi_1, \pi_2, \ldots, \pi_T)$ (where each π_t is a label from the vocabulary, including blank) is the product of the probabilities of each label at its respective time step, assuming conditional independence:

$$P(\pi) = \prod_{t=1}^{T} P(\pi_t|X)$$

Where $P(\pi_t|X)$ is the probability of the network outputting label π_t at time t given the input X.

Now, let B be the CTC "collapse" function. We seek all paths π such that $B(\pi) = Y$.

The probability of the true transcript Y given the input X is the sum of probabilities of all valid paths:

$$P(Y|X) = \sum_{\pi:B(\pi)=Y} P(\pi|X)$$

3.4 The Loss Function

The objective during training is to maximize $P(Y|X)$ for the correct transcript Y. In machine learning, this is typically achieved by minimizing a loss function. Therefore, the CTC loss is defined as the negative logarithm of this probability:

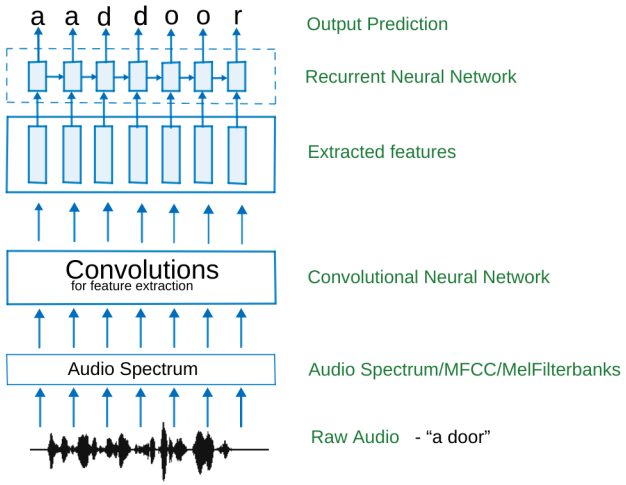
$$\mathscr{L}_{\text{CTC}}(Y|X) = -\log\left(\sum_{\pi:B(\pi)=Y} P(\pi|X)\right)$$

Minimizing this loss trains the network to increase the aggregate probability of all paths that correctly map to the target transcript.

4 End-to-End Neural ASR

End-to-End (E2E) models aim to simplify the traditional ASR pipeline by replacing the separate AM, HMM, and Pronunciation models with a single, deep neural

network. This network learns to directly map a sequence of audio features to a sequence of text (characters or words).



There are three main families of E2E models:

- **CTC-based Models:** These use CTC loss as described above. They are often simpler and faster to train. A common architecture is a stack of convolutional and/or recurrent layers followed by a softmax layer that makes the frame-wise predictions for CTC.
  - **Example Model:** Deep Speech 2.
  - **Reference:** An Intuitive Explanation of End-to-End Speech Recognition with Deep Speech
  - **Many state-of-the-art CTC-based ASR models:**
    - **Original:** Audio Features -> (Stack of CNNs/RNNs) -> Softmax Layer -> CTC Loss
    - **Transformer-based:** Audio Features -> (Transformer Encoder Blocks) -> Softmax Layer -> CTC Loss
- **Attention-based Encoder-Decoder Models:** These models (also called sequence-to-sequence models) first "listen" to the entire audio sequence with an **Encoder** network to create a rich representation. Then, a **Decoder** network generates the output text one character at a time, using an **Attention Mechanism** to focus on the most relevant parts of the encoded audio for each character it predicts.

  - The idea behind **Attention-based Encoder-Decoder** Models in **ASR** (like Listen, Attend and Spell) is very similar to how **Machine Translation Transformers** work
  - **Example Model:** Listen, Attend and Spell (LAS).
  - **Reference:** Attention and Augmented Recurrent Neural Networks (Distill.pub) (Explains the core attention concept)

- **RNN-Transducer (RNN-T) Models:** This architecture combines the strengths of both CTC and attention models. It processes the audio stream and generates output labels simultaneously, making it naturally suited for online/streaming recognition. It uses an encoder network (like attention models), a prediction network that looks at the previously generated text, and a "joiner" network that combines information from both to produce an output.
  - **Reference:** An Introduction to RNN-T