Speech Recognition Techniques: A Comparative Review

Mr. Vasu Jeevan Naik
Masters of Engineering,
Department of Information Technology, Goa College of Engineering, Ponda, Goa 403401
vasunaik1999@gmail.com

Dr. Nilesh B. Fal Dessai
Head of the Department,
Department of Information Technology, Goa College of Engineering, Ponda, Goa 403401
nfd@gec.ac.in

Keywords: Speech Recognition, Speech to Text, Hidden Markov Models, Recurrent Neural Networks, Connectionist Temporal Classification, Convolutional Neural Networks.

Abstract— Speech recognition, also known as speech-to-text or automatic speech recognition, is a technology that enables the transformation of spoken words into written or machine-readable text. This technology has numerous applications such as speech-to-text dictation, voice command, voice-enabled interfaces, automated call centers, hand-free computing, and translation. The working of a speech recognition system involves several steps such as data collection, data preprocessing, feature extraction, model selection, model training, model evaluation, and model deployment. There are various algorithms/methods used to build speech recognition systems, including Hidden Markov Models (HMMs), Recurrent Neural Networks (RNNs), Connectionist Temporal Classification (CTC), Gaussian Mixture Models (GMM), Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN). In this paper, we present an overview of speech recognition and four methods: Hidden Markov Models (HMMs), Recurrent Neural Networks (RNNs), Connectionist Temporal Classification (CTC), and Convolutional Neural Networks (CNN) used for building speech recognition system. Also, we have discussed how these methods can be evaluated using several metrics such as Word Error Rate (WER), Word Accuracy, Character Error Rate (CER), Precision, Recall, F1 Score, Perplexity, Latency, and Speaker Dependence/Independence.

I. INTRODUCTION

Speech recognition, also known as speech-to-text or automatic speech recognition, is a technology that transforms spoken words into written or machine-readable text. It has been a rapidly growing field in recent years due to the advancement in the field of machine learning and artificial intelligence. In recent years it is gaining more importance as it can improve accessibility, efficiency, user experience, and customer service, automate data entry and enable new applications.

Speech recognition technology is used in a wide range of applications, including:

Speech-to-text: This application allows users to dictate text into a device or software, which converts spoken words into text. This can be useful for individuals who have difficulties typing, such as those with disabilities or elderly individuals.

Voice Command: Voice command allows users to control various devices and software through voice commands. This includes voice assistants such as Apple's Siri, Amazon's Alexa, and Google Assistant.

Voice-enabled Interfaces: Voice-enabled interfaces are used in a variety of devices, including smartphones, smart home devices, and in-car entertainment systems. These interfaces allow users to interact with the devices using voice commands.

Automated Call Centers: Speech recognition technology is used in automated call centers to provide customers with a more natural and efficient way to interact with the system. Customers can use voice commands to complete tasks, such as checking account balances or making a payment.

Hand-free Computing: Speech recognition technology enables hands-free computing, making it easier for users to perform tasks such as dictating documents, sending emails, or searching the web.

Translation: Speech recognition technology can be used in translation applications, allowing users to speak in one language and have the spoken words translated into another language in real time.

Speech recognition can be done using different methods such as Hidden Markov Models (HMMs), Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs), Connectionist Temporal Classification (CTC), Convolutional Neural Networks (CNN), etc.

II. WORKING OF SPEECH RECOGNITION SYSTEM

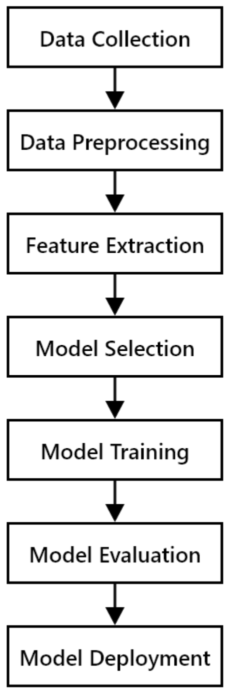

Fig 1: Working of Speech Recognition System

Building a speech recognition system involves a systematic approach, the steps to be followed are as follows:

Data Collection: The first step is to collect a large dataset of speech signals and their corresponding transcriptions. This dataset will be used to train the speech recognition model.

Data Preprocessing: The next step is to preprocess the collected data to remove any noise, reduce the variability of the speech signals, and convert the speech signals into a form that can be used by the model.

Feature Extraction: The next step is to extract a set of acoustic features from the speech signals. This includes extracting information about the sound of the speech, such as the pitch, energy, and spectral characteristics.

Model Selection: Choose a suitable model to represent the speech signals and the relationship between speech signals and text transcriptions. The choice of model will depend on the specific requirements of the application and the complexity of the speech signals.

Model Training: Train the chosen model on the preprocessed speech signals and their corresponding transcriptions. This involves optimizing the model parameters to minimize the error between the model's predictions and the true transcriptions.

Model Evaluation: Evaluate the performance of the trained model on a held-out validation set to ensure that it is accurate and generalizes well to new speech signals.

Model Deployment: Finally, deploy the trained model in a real-world application and integrate it with other components, such as a language model, to improve the overall performance of the speech recognition system.

III. METHODS OF SPEECH RECOGNITION

In this section, we discuss the various algorithms/methods which can be used to build a speech recognition system.

A. Hidden Markov Models (HMM)

Hidden Markov Model (HMM) is a statistical model that is widely used for speech recognition. It provides an efficient algorithm for state and parameter estimation [1]. It is based on the idea that the underlying speech signal can be viewed as a Markov process, where the state of the process is hidden and the observation or speech signal is visible to the observer. HMM is the dominant technology in the field of automatic speech recognition and it represents each unit of sound (phoneme) in the speech recognition process [1]. The foundation of modern HMM-based speech recognition systems was laid down by research groups at Carnegie Mellon, IBM, and Bell Labs in the 1970s [1]. HMMs consist of interconnected states which emit observable outputs and have two types of parameters: symbol emission probabilities and state transition probabilities [3].

The HMM has three basic problems, that is evaluation, decoding, and learning problem [2] [3].

Evaluation Problems: The evaluation problem of Hidden Markov Models (HMM) refers to the calculation of the likelihood of an observed sequence given a specific HMM model. This problem involves computing the probabilities of observing a specific sequence given the HMM parameters, such as the state transition probabilities, the observation probabilities, and the initial state probabilities. The evaluation problem can be solved using the forward algorithm, which computes the probability of observing the entire sequence from the start to the end.

Decoding Problem: The decoding problem of Hidden Markov Models (HMM) refers to finding the most likely sequence of hidden states given the observed sequence and the HMM parameters. This problem is also known as the Viterbi algorithm and involves finding the most probable sequence of hidden states that generate the observed sequence. The Viterbi algorithm can be used to determine the most likely speech recognition sequence or the most likely sequence of phonemes given the observed speech signal.

Learning Problem: The learning problem of Hidden Markov Models (HMM) refers to the estimation of the HMM parameters given a set of observed sequences. This problem involves finding the best set of HMM parameters that maximize the likelihood of the observed sequences. The learning problem can be solved using the maximum likelihood estimation or the Baum-Welch algorithm, which iteratively updates the HMM parameters until convergence. This learning problem is crucial for speech recognition systems, as it allows the system to adapt to different speakers and speech styles.

There are several algorithms used in Hidden Markov Models (HMMs) for speech-to-text conversion. Below are a few mentioned:

Forward Algorithm: The forward algorithm is used to calculate the probability of observing a sequence of speech signals given the HMM model parameters. It uses the following formula to compute the forward probability:

$$\alpha_t (i)= \Sigma_j P(q_t=i|q_{t-1}=j) * \alpha_{t-1}(j) * P(O_t|q_t=i)$$

where $\alpha_t(i)$ is the forward probability at time t and state i, $P(q_t=i|q_{t-1}=j)$ is the state transition probability, $\alpha_{t-1}(j)$ is the forward probability at time t-1 and state j, and $P(O_t|q_t=i)$ is the emission probability.

Backward Algorithm: The backward algorithm is used to calculate the probability of observing the future speech signals given the current state and the HMM model parameters. It uses the following formula to compute the backward probability:

$$\beta_t(i) = \Sigma_j \beta_{t+1}(j) * P(q_{t+1}=j|q_t=i) * P(O_{t+1} | q_{t+1}=j)$$

where $\beta_t(i)$ is the backward probability at time t and state i, $\beta_{t+1}(j)$ is the backward probability at time t+1 and state j, $P(q_{t+1}=j|q_t=i)$ is the state transition probability, and $P(O_{t+1}|q_{t+1}=j)$ is the emission probability.

Viterbi Algorithm: The Viterbi algorithm is used to find the most likely state sequence given an observation sequence and the HMM model parameters. It uses the following formula to compute the maximum probability at each time step:

$$\delta_t(i) = \max_j \delta_{t-1}(j) * P(q_t=i|q_{t-1}=j) * P(O_t|q_t=i)$$

where $\delta_t(i)$ is the maximum probability at time t and state i, $\delta_{t-1}(j)$ is the maximum probability at time t-1 and state j, $P(q_t=i|q_{t-1}=j)$ is the state transition probability, and $P(O_t|q_t=i)$ is the emission probability.

Baum-Welch Algorithm: The Baum-Welch algorithm is used to estimate the HMM parameters giving an observation sequence and the initial estimates of the parameters. The algorithm iteratively updates the estimates until convergence. The formula for updating the emission probability is as follows:

$$\pi_i = \Sigma_t \alpha_t(i) * \beta_t(i) / \Sigma_{i,j} \alpha_T(j) * \beta_T(j)$$

where π_i is the estimate of the emission probability for state i, $\alpha_t(i)$ is the forward probability at time t and state i, $\beta_t(i)$ is the backward probability at time t and state i, and T is the length of the observation sequence.

Pseudocode for HMM:

1. *Load the speech signal data*
2. *Preprocess the speech signal data (e.g., normalization, feature extraction, etc.)*
3. *Train an HMM on a set of speech signal data to learn the transition and emission probabilities*
4. *Use the trained HMM to transcribe a test set of speech signals*
5. *Decode the output of the HMM to obtain the transcribed text*
6. *Evaluate the accuracy of the transcribed text using metrics such as Word Error Rate (WER), Precision, Recall, F1 Score, and Accuracy.*

Advantages of HMM:

HMMs are flexible: HMMs can model a wide range of systems, including speech recognition, image processing, natural language processing, and many others.

HMMs can handle noisy data: HMMs can effectively deal with noisy or incomplete data, as they incorporate statistical probabilities to account for errors or uncertainties in the data.

HMMs are computationally efficient: HMMs use an algorithm called the Viterbi algorithm, which is a dynamic programming approach that makes them computationally efficient.

HMMs can handle sequential data: HMMs are particularly useful for modeling sequential data, as they can account for the temporal dependencies between observations in a sequence.

HMMs can perform well with limited training data: HMMs can perform well with limited training data because they use statistical modeling techniques that can learn from small datasets.

B. Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) are a type of neural network that is particularly well-suited to processing sequential data such as speech signals. The main idea behind RNNs is to use feedback connections that allow information to be passed from one step in the sequence to the next. This allows the network to model dynamic dependencies in the data. RNNs are inherently deep in time since their hidden state is a function of all previous hidden states [7].

RNN is a network that has a memory that decides future predictions. This is because as it predicts one letter it will affect the likelihood of the upcoming letter which it will predict too[8]. Hence having a memory of previous predictions boosts the network to make more accurate predictions going forward [8]. RNN has the capability to spot a specific word in any length of connected words, even if they are spoken by unknown speakers [9].

RNNs are designed to handle sequential information by maintaining a hidden state that is updated at each time step based on the current input and the previous hidden state. This allows RNNs to capture the context and dependencies between adjacent time steps in the speech signal. In speech recognition, an RNN can be trained to predict the phonemes or sub-word units in an audio signal based on the spectral features extracted from the speech signal. The extracted spectral features are passed through the RNN as input, and the hidden state of the RNN is updated at each time step based on the input and the previous hidden state. Finally, the output of the RNN is used to predict the transcription of the speech signal.

One popular type of RNN for speech recognition is the Long Short-Term Memory (LSTM) network, which is designed to address the vanishing gradient problem and enable long-term memory retention. Another type of RNN that is commonly used for speech recognition is the Gated Recurrent Unit (GRU), which is similar to LSTM but has fewer parameters.

The mathematical formula for a basic RNN can be represented as follows:

$$h_t = f(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = g(W_o y_t + U_o h_t + b_o)$$

where:

h_t is the hidden state at time t

x_t is the input at time t

y_t is the output at time t

W_h, W_o, U_h, and U_o are weight matrices

b_h and b_o are biased vectors

f and g are activation functions

The hidden state h_t is computed based on the current input x_t and the previously hidden state h_{t-1}. The output y_t is then computed based on the currently hidden state h_t. The weight matrices and bias vectors are learned during the training process to minimize the error between the predicted outputs and the actual outputs.

There are many variants of RNNs, including long short-term memory (LSTM) and gated recurrent units (GRUs), that have been developed to address some of the limitations of basic RNNs.

Pseudocode for RNN:

```
1. Load the speech signal data
2. Preprocess the speech signal data (e.g., normalization, feature extraction, etc.)
3. Define the RNN architecture (e.g., number of layers, type of RNN cells, activation functions, etc.)
4. Train the RNN on the preprocessed speech signal data
5. Test the trained RNN on a test set of speech signals
6. Decode the output of the RNN to obtain the transcribed text
7. Evaluate the accuracy of the transcribed text using metrics such as Word Error Rate (WER), Precision, Recall, F1 Score, and Accuracy.
```

Advantages of RNN:

RNNs can handle variable-length input: Unlike feedforward neural networks, which require a fixed input size, RNNs can handle inputs of varying lengths. This makes them ideal for processing data with temporal dependencies, such as time-series data or natural language text.

RNNs can remember previous inputs: RNNs have a "memory" component that allows them to remember previous inputs and use this information to make predictions about future inputs. This makes them particularly useful for tasks such as speech recognition, where context is important for accurate predictions.

RNNs can learn from long sequences: RNNs are designed to work with sequences of arbitrary length, which means they can learn from long sequences of data. This makes them ideal for applications such as language modeling, where the input sequence can be very long.

RNNs can perform both regression and classification: RNNs can be used for both regression and classification tasks. This means they can be applied to various applications, such as time-series prediction, text classification, and image captioning.

RNNs can be trained with backpropagation: RNNs can be trained using the backpropagation algorithm, which allows for efficient learning and optimization of the network weights.

C. Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a type of feedforward artificial neural network commonly used for analyzing visual imagery, and it is particularly well-suited for tasks such as image classification, object detection, and recognition [4]. CNN consists of three main components, including the convolutional layer for feature extraction, the sub-sampling (pooling) layer for dimensionality reduction, and the fully connected layers that map the features to the desired output [5] [6].

In terms of image classification, CNNs consist of interconnected nodes organized into convolutional and pooling layers that extract visual patterns from the input data, making them highly effective in visual recognition tasks [5]. The input data is organized as a number of feature maps. This can be thought of as a 2D array of pixel values for image processing, with RGB values as three different 2D feature maps[5]. In speech recognition, a CNN is a type of neural network that specifically processes speech signals represented as spectrograms and for speech feature vectors, the input can be loosely thought of as a spectrogram with static, delta, and delta-delta features[5]. The convolutional layers in the network perform feature extraction, learning to identify important spectral or temporal patterns in the speech signal. The final layers of the network can then be used for tasks such as speech classification, speaker identification, or phoneme recognition.

In comparison to traditional recurrent neural networks (RNNs) and hidden Markov models (HMMs), which have been widely used for speech recognition, CNNs have the advantage of parallel processing, allowing for faster and more efficient training and inference. Additionally, CNNs can be trained end-to-end, directly from the raw audio signal, without the need for manual feature extraction, which is often required in traditional speech recognition models.

Steps for building Convolutional Neural Networks (CNNs) for the speech-to-text:

Preprocessing: The raw speech signal is transformed into a spectrogram or mel-frequency cepstral coefficients (MFCCs) using a mathematical process. The resulting spectrogram or MFCCs can be represented as a matrix X, with dimensions T x F, where T is the number of time frames and F is the number of frequency bins.

Convolution: The first layer in the CNN is typically a convolutional layer, which performs a dot product between a set of filters and the input matrix X. The filters can be represented as a matrix W with dimensions F' x K, where F' is the number of filters and K is the filter length. The output of the convolutional layer is another matrix Y with dimensions (T-K+1) x F', which can be represented as:

$$Y = f(W * X + b)$$

where f is an activation function, such as ReLU, and b is a bias term.

Pooling: The next layer in the CNN is typically a pooling layer, which performs a down-sampling of the output from the previous layer. Common pooling methods include max pooling and average pooling. The pooling layer can be represented as:

$Z = \text{pool}(Y)$

where the pool is the pooling operation.

Repeat: The convolutional and pooling layers can be repeated multiple times to form a deep network, allowing the CNN to learn multiple levels of abstraction from the input spectrogram or MFCCs.

Fully connected layer: The final layer in the CNN is typically a fully connected layer, which performs a dot product between the output from the previous layer and a set of weights. The fully connected layer can be represented as:

$P = W' * Z + b'$

where W' is a matrix of weights and b' is a bias term.

Softmax: The final output from the CNN is often passed through a softmax function to produce a probability distribution over the possible transcriptions. The softmax function can be represented as:

$Q = \text{softmax}(P)$

where Q is the probability distribution over the possible transcriptions.

Training: The CNN can be trained using a labeled dataset by minimizing the cross-entropy loss between the predicted probabilities Q and the ground-truth transcriptions, using an optimization algorithm such as stochastic gradient descent or Adam. The loss function can be represented as:

$L = - \sum_i \log(Q_i)$

where i is the index of the ground-truth transcription.

Pseudocode for CNN

1. *Load the speech signal data*
2. *Preprocess the speech signal data (e.g., normalization, segmentation, etc.)*
3. *Define the CNN architecture (e.g., number of layers, number of filters, activation functions, etc.)*
4. *Train the CNN on the preprocessed speech signal data.*
5. *Test the trained CNN on a test set of speech signals.*
6. *Decode the output of the CNN to obtain the transcribed text.*
7. *Evaluate the accuracy of the transcribed text using metrics such as Word Error Rate (WER), Precision, Recall, F1 Score, and Accuracy.*

Advantages of CNN:

CNNs can automatically learn features: CNNs use a series of convolutional layers to automatically learn features from input images or videos. This allows CNNs to perform tasks such as object detection and image classification without the need for hand-engineered features.

CNNs can handle large inputs: CNNs can handle inputs of arbitrary size, which makes them ideal for processing large images or videos.

CNNs can be trained with backpropagation: CNNs can be trained using the backpropagation algorithm, which allows for efficient learning and optimization of the network weights.

CNNs can perform both classification and regression: CNNs can be used for both classification and regression tasks, which makes them applicable to a wide range of applications.

CNNs can be used for transfer learning: CNNs can be pre-trained on large datasets and then fine-tuned on smaller datasets for specific tasks. This allows for the efficient use of computational resources and can improve performance on smaller datasets.

D. Connectionist Temporal Classification (CTC)

The problem in speech recognition is mapping an audio waveform to a sequence of words or characters. This is typically done using a deep neural network (DNN) that transforms the input acoustic features into a sequence of output probabilities, which are then mapped to words or characters using a language model. However, the output sequence may have a different length than the input sequence, making it difficult to train the DNN. CTC provides a solution to this problem by allowing the DNN to output a sequence of probabilities that can include "blank" labels to indicate pauses or silences. These blank labels are used to collapse consecutive repeated labels and produce a final output sequence that matches the length of the input sequence.

The CTC algorithm works by summing over all possible alignments of the output sequence with the input sequence and then computing the gradient of the log-likelihood of the correct output sequence with respect to the DNN parameters. This gradient can be computed efficiently using the forward-backward algorithm and can be used to update the DNN parameters using stochastic gradient descent.

In speech recognition, CTC loss is often deployed for training deep Recurrent Neural Networks (RNNs) with Long Short Term Memory (LSTM) cells or Gated Recurrent Units (GRUs) [10]. The output units of a CTC-trained neural network can be characters (Chars), context-independent phonemes (CI-Phns), or context-dependent phonemes (CD-Phns) [11].

Connectionist Temporal Classification (CTC) is a widely used algorithm in speech recognition systems. It is an end-to-end deep learning method that models the relationship between input speech features and the corresponding transcription (text).

The algorithm can be mathematically represented as follows:

Let the speech feature sequence be represented as $X = \{x_1, x_2, x_3, ..., x_T\}$ where T is the length of the sequence and x_t is the feature vector at time step t.

The target sequence (text) is represented as $Y = \{y_1, y_2, y_3, ..., y_0\}$ where U is the length of the target sequence and y_u is the target label (e.g., a phoneme or word).

CTC introduces a new blank symbol "-" that is used to align the speech feature sequence with the target sequence. The algorithm creates an expanded sequence $Z = \{z_1, z_2, z_3, ..., z_S\}$ where S is the length of the expanded sequence and z_s is a symbol from the set $\{y_u, "-"\}$.

The goal of CTC is to find the optimal alignment between X and Y such that the expanded sequence Z has the maximum probability, P(Z|X). This can be represented as:

$P(Z|X) = max_Z (P(Z, X)) = max_Z (P(Z|X) * P(X))$

Where P(X) is the prior probability of X.

To compute P(Z|X), we use a neural network with an output layer that computes the probability of each symbol at each time step. This can be represented as:

$P(z_s|x) = \text{softmax}(f_s(x))$

Where $f_s(x)$ is the output of the neural network at time step s for input x.

Once the probabilities are computed, the optimal alignment between X and Y can be found using dynamic programming. The dynamic programming algorithm computes the probability of the target sequence given the input sequence by considering all possible paths through the expanded sequence. The final result is the sum of probabilities over all possible paths that yield the target sequence.

The dynamic programming algorithm can be represented as follows:

Let $A_{s,u}$ be the probability of getting to the end of the expanded sequence Z (S) and the end of the target sequence Y (U) with $z_s = y_u$.

$A_{s,u} = P(z_s = y_u, z_{s-1} \neq y_u, ..., z_1 \neq y_u)$

The base case is $A_0,0 = 1$ and $A_{s,u} = 0$ for all $s > 0$ and $u > 0$.

The recursive case is defined as:

$A_s,u = P(z_s = y_u | x) * (\sum_{k=0}\text{^}\{u-1\} A_{s-1,k})$

The final result is:

$P(Z|X) = \sum_{u=1}\text{^}U A_S,u$

This algorithm is used to train the neural network in an end-to-end manner, where the input speech features are directly mapped to the target transcription. The trained network can then transcribe speech in real time by computing the most likely target sequence given a speech feature sequence.

Pseudocode for CTC

1. *Load the speech signal data*

2. *Preprocess the speech signal data (e.g., normalization, feature extraction, etc.)*
3. *Define the neural network architecture (e.g., number of layers, type of activation functions, etc.)*
4. *Train the neural network using CTC loss on the preprocessed speech signal data*
5. *Test the trained neural network on a test set of speech signals*
6. *Decode the neural network's output using the CTC algorithm to obtain the transcribed text.*
7. *Evaluate the accuracy of the transcribed text using metrics such as Word Error Rate (WER), Precision, Recall, F1 Score, and Accuracy.*

Advantages of CTC:
CTC can handle variable-length input: CTC is designed to handle input sequences of varying lengths, which makes it ideal for processing sequential data such as speech or handwriting.

CTC can handle output sequences of different lengths: CTC allows for the prediction of output sequences of different lengths, which is useful in applications where the length of the output sequence is unknown or variable.

CTC is robust to alignment errors: CTC can handle misalignments between input and output sequences, which is useful in applications where the alignment is uncertain or noisy.

CTC can be trained with backpropagation: CTC can be trained using the backpropagation algorithm, which allows for efficient learning and optimization of the network weights.

CTC can be combined with other loss functions: CTC can be combined with other loss functions, such as cross-entropy, to improve performance on specific tasks.

IV. ANALYSIS OF TECHNIQUES

CNNs and RNNs are deep learning models commonly used in speech recognition where CNNs are often used for feature extraction from spectrograms or other representations of audio, while RNNs are used for modeling temporal dependencies in the data.

HMMs, on the other hand, are a traditional statistical model for speech recognition. They model speech as a sequence of hidden states, which emit observations that correspond to acoustic features.

CNNs are typically not used as the sole model for speech recognition tasks, as they are designed for image classification rather than sequence modeling. However, they can be used as a feature extractor for speech signals, where the output of a CNN is fed into another model such as an RNN or HMM for sequence modeling.
RNNs are widely used in speech recognition, as they are designed to handle sequential data and capture long-term dependencies between elements in the sequence. RNNs can be

trained end-to-end on raw speech signals, and they have achieved good performance on a variety of speech recognition tasks.

HMMs have been widely used in speech recognition for many years and they continue to be a popular choice for the task. HMMs are particularly well suited to the statistical structure of speech signals, and they can be trained using the Baum-Welch algorithm. HMMs can be combined with other models, such as deep neural networks, to improve performance.

CTC is a commonly used method for solving sequential labeling problems, including speech recognition. It allows the model to make multiple label predictions for a given input sequence and then computes a cost function to evaluate the quality of the predicted labels. CTC has been shown to work well for speech recognition, particularly in the context of end-to-end training with deep neural networks.

Overall, CNNs and RNNs are more flexible and powerful than HMMs, and CTC can help address the problem of variable-length outputs. However, HMMs are still commonly used in certain applications, and there is ongoing research exploring how to combine these different techniques for improved speech recognition performance.

V. EVALUATION PARAMETERS

There are several metrics used to evaluate speech recognition algorithms, including:

Word Error Rate (WER): WER measures the difference between the reference transcript and the recognized speech, expressed as a percentage of the number of words in the reference transcript.

Word Accuracy: Word accuracy measures the proportion of correctly recognized words, expressed as a percentage of the total number of words in the reference transcript.

Character Error Rate (CER): CER measures the difference between the reference transcript and the recognized speech, expressed as a percentage of the number of characters in the reference transcript.

Precision, Recall, and F1 Score: Precision, recall, and F1 score are commonly used in machine learning to evaluate the performance of a classification model. In the context of speech recognition, precision measures the proportion of recognized words that are correct, recall measures the proportion of reference words that are correctly recognized, and the F1 score is the harmonic mean of precision and recall.

Perplexity: Perplexity is a measure of how well a language model predicts the likelihood of a sequence of words. In the context of speech recognition, it measures the ability of the model to predict the words in the reference transcript.

Latency: Latency is the time between the start of the spoken input and the output of the recognition result. Low latency is important for real-time applications such as voice-activated systems.

Speaker Dependence/Independence: Speaker dependence measures the extent to which the performance of the speech recognition algorithm depends on the speaker, while speaker independence measures its ability to recognize speech from different speakers.

These metrics can be used to compare different speech recognition algorithms and determine which algorithm performs best for a specific task or application.

VI. CONCLUSION

In this paper we discussed four techniques to implement speech recognition system: Hidden Markov Models (HMMs), Recurrent Neural Networks (RNNs), Connectionist Temporal Classification (CTC), and Convolutional Neural Networks (CNN). Each of these techniques has its own strengths and weaknesses for speech recognition. RNNs and CTC have been shown to work well in speech recognition tasks and are popular choices for end-to-end training with deep neural networks. HMMs are still widely used and are well suited to the statistical structure of speech signals. The choice of technique will depend on the specific requirements of a given problem, and it may be necessary to use a combination of techniques to achieve the best performance.

REFERENCES

[1]. S. Boruah and S. Basishtha, "A study on HMM-based speech recognition system," 2013 IEEE International Conference on Computational Intelligence and Computing Research, Enathi, India, 2013, pp. 1-5, doi: 10.1109/ICCIC.2013.6724147.

[2] M. Aymen, A. Abdelaziz, S. Halim, and H. Maaref, "Hidden Markov Models for automatic speech recognition," 2011 International Conference on Communications, Computing and Control Applications (CCCA), Hammamet, Tunisia, 2011, pp. 1-6, doi: 10.1109/CCCA.2011.6031408.

[3] Liu, Yang, "A Study of Hidden Markov Model. " Master's Thesis, University of Tennessee, 2004. https://trace.tennessee.edu/utk_gradthes/2326

[4] X. Yang, H. Yu, and L. Jia, "Speech Recognition of Command Words Based on Convolutional Neural Network," 2020 International Conference on Computer Information and Big Data Applications (CIBDA), Guiyang, China, 2020, pp. 465-469, doi: 10.1109/CIBDA50819.2020.00110.

[5] O. Abdel-Hamid, A. -r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pp. 1533-1545, Oct. 2014, doi: 10.1109/TASLP.2014.2339736.

[6] Ayad Alsobhani et al, “Speech Recognition using Convolution Deep Neural Networks,” 2021 J. Phys.: Conf. Ser. 1973 012166

[7] A. Graves, A. -r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 2013, pp. 6645-6649, doi: 10.1109/ICASSP.2013.6638947.

[8] A. Amberkar, P. Awasarmol, G. Deshmukh and P. Dave, "Speech Recognition using Recurrent Neural Networks," 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), Coimbatore, India, 2018, pp. 1-4, doi: 10.1109/ICCTCT.2018.8551185.

[9] H. Hasegawa and M. Inazumi, "Speech recognition by dynamic recurrent neural networks," Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), Nagoya, Japan, 1993, pp. 2219-2222 vol.3, doi: 10.1109/IJCNN.1993.714167.

[10] E. Variani, T. Bagby, K. Lahouel, E. McDermott and M. Bacchiani, "Sampled Connectionist Temporal Classification," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 4959-4963, doi: 10.1109/ICASSP.2018.8461929.

[11] P. Wang, J. Li, and B. Xu, "Applying connectionist temporal classification objective function to Chinese Mandarin speech recognition," 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP), Tianjin, China, 2016, pp. 1-5, doi: 10.1109/ISCSLP.2016.7918372.