Automatic Speech Recognition

Peravali Harshavardhan Reddy, Umashankar Bashaboyina, Ketan Eswar .B, Ankalapu Shiva
Department of Electrical Engineering
Indian Institute of Technology Hyderabad
{ee21btech11040, ee21btech11008, ee21btech11010, ee21btech11003}@iith.ac.in

Ranveer Sahu
Department of Engineering Science
Indian Institute of Technology Hyderabad
es21btech11025@iith.ac.in

Abstract

Automatic Speech Recognition (ASR) has emerged as a pivotal technology in the field of human-computer interaction, enabling seamless communication between machines and users.ASR has witnessed significant advancements over the past decades, evolving from traditional pipeline-based systems to sophisticated deep learning models, such as Recurrent Neural Networks (RNNs), Transformers, Conformers and end-to-end models, which map audio inputs directly to text outputs, offer a streamlined approach to ASR. While these models offer significant improvements in performance, they come with increased computational complexities and the need for large datasets to train effectively. The Linda Johnson(LJ) Speech dataset, with its 13,100 high-quality audio recordings and corresponding transcriptions, is widely used due to its manageable size and clean, consistent data. This dataset provides an ideal foundation for training ASR systems on resource-limited platforms.The advancements in ASR make it possible for machines to understand and respond to spoken language, enhancing applications like virtual assistants, transcription services, and real-time translation. As technology continues to improve, its role in everyday life and industrial applications grows, making it a critical area of ongoing research and development. In this paper we implemented a Deep Speech 2 like model by adding BiLSTM, and just using Adam optimizer, We achieved a word error rate of 0.3099.

1. Introduction

Automatic Speech Recognition (ASR) has become a critical technology in various domains, enabling seamless interaction between humans and machines. Converting spoken language into text is essential in applications such as virtual assistants, transcription services, and language translation. ASR systems aim to improve recognition accuracy while managing the complexities of real-world environments. In ASR, noise, accents, and background sounds can significantly degrade performance, making effective noise handling and robust models critical. ASR models have evolved over time, ranging from traditional statistical models to sophisticated deep learning approaches. Every model to perform Speech Recognition has broadly same steps as in fig.1 with Deep Speech 2 as model; Audio preprocessing, extracting features from it and sending it to a model to find relation between text and audio information.

2. Related work

Traditional statistical ASR models, include stages like feature extraction, acoustic modeling, and language modeling. They have been foundational in speech recognition. Hidden Markov Models (HMMs)[10] are effective in modeling temporal sequences and are known for their robustness to noise and varying speech rates. Similarly, Gaussian Mixture Models (GMMs)[13], often used in conjunction with HMMs, offer a probabilistic approach to capturing acoustic feature variations. While these models have been instrumental in achieving reliable performance, they require extensive training and tuning, and can encounter difficulties when dealing with complex speech patterns, such as accents or long-range dependencies in speech.

Deep learning has significantly advanced ASR systems by addressing limitations of traditional statistical models. Deep Neural Networks (DNNs) enhance feature extraction by learning hierarchical representations, leading to improved accuracy and adaptability to large datasets.[11] implements DNN-hidden Markov model which replaces the GMMs that estimate the probability density functions

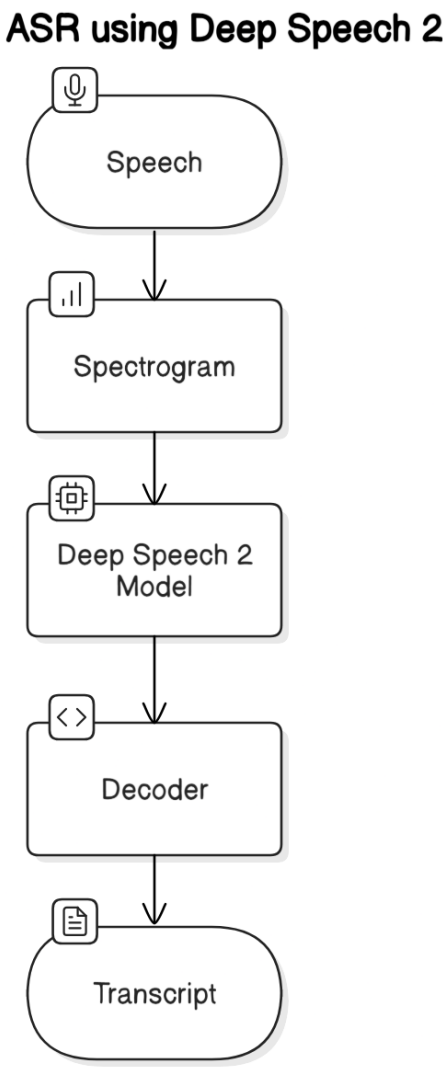

Figure 1. High level flow chart of Speech Recognition

by DNNs.[8] explains various DNN models that can be implemented for speech recognition. Unlike traditional DNN-HMM models, [5] learns all the components of a speech recognizer jointly through two components: a listener and a speller. However, they require substantial computational resources and large amounts of labeled data. Recurrent Neural Networks (RNNs)[1], particularly Long Short-Term Memory (LSTM) networks[9] further improved speech recognition by capturing temporal dependencies, making them ideal for sequential data like speech.

DeepSpeech [7] is an end-to-end automatic speech recognition (ASR) model inspired by Deep Learning. It uses Recurrent Neural Networks (RNNs) or similar architectures to transcribe speech directly into text, trained on large datasets with audio-transcription pairs. Key features include a simple architecture, ability to generalize well, and the use of Connectionist Temporal Classification (CTC) for aligning speech with text.

DeepSpeech 2 [2] is an improved version of the original DeepSpeech model. It expands on the architecture by incorporating Bidirectional Recurrent Neural Networks (BRNNs) and adding more layers for increased accuracy. It supports both English and Mandarin speech recognition, demonstrating robustness across languages. The model leverages GPUs for training, employs batch normalization, and uses CTC for end-to-end training. DeepSpeech 2 outperforms traditional ASR models by learning directly from raw audio without hand-engineered features.

Building on the advancements of deep learning, state-of-the-art ASR systems now leverage Transformer models, which utilize self-attention mechanisms to process entire sequences simultaneously, significantly improving context understanding and capturing long-range dependencies without encountering the vanishing gradient problem. This Transformer models can further be enhanced by incorporating several methods as mentioned in [4],reduces the number of parameters in the full-precision model and achieves a further 4x compression by fully quantizing to 8-bit fixed-point precision while retaining performance.The Conformer model further enhances performance by combining convolutional neural networks with Transformers, improving local feature extraction while retaining global context. This hybrid approach has demonstrated superior accuracy in various speech recognition benchmarks, as highlighted in [6].

End-to-end models have become a prominent trend in Automatic Speech Recognition (ASR) by directly mapping audio inputs to text outputs, eliminating the need for intermediate steps. This streamlined approach enhances efficiency and is favored in both industry and academia. Systems based on Conformer and Transformer architectures are particularly popular for their superior accuracy and adaptability to various speech datasets. A key example is [12], which introduced a self-supervised learning framework to capture meaningful audio representations, and its successor [3], showcases the effectiveness of self-supervised learning in ASR while achieving state-of-the-art performance with minimal labeled data. These models have significantly influenced recent advancements in end-to-end speech recognition technologies, reinforcing the shift towards more integrated and efficient ASR solutions.

3. Method

The development of the Automatic Speech Recognition (ASR) system leverages the DeepSpeech2 with BiLSTM layers. LJ Speech dataset is used to train and evaluate model. Block diagram of proposed model is as in fig.2

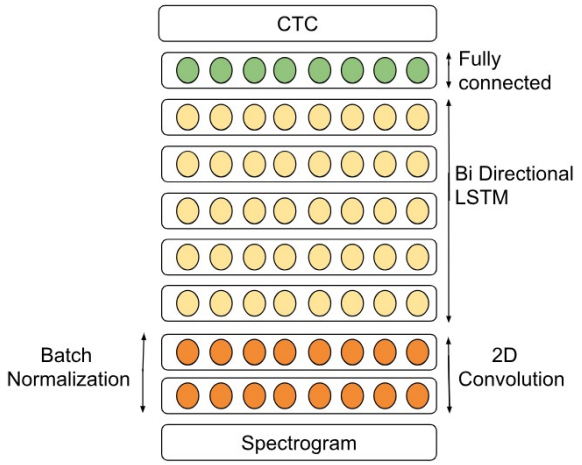

Figure 2. Block diagram of proposed model

3.1. Data Processing and Encoding

Feature extraction is a crucial step in the preprocessing phase of audio data analysis, particularly in tasks like Automatic Speech Recognition (ASR). It involves transforming raw audio signals into a set of meaningful features that reduce dimensionality and retain essential information. Common methods for feature extraction include Short-Time Fourier Transform (STFT), which analyzes audio signals in both time and frequency domains by breaking them into short overlapping segments; Mel-Frequency Cepstral Coefficients (MFCC), which mimic human auditory perception by using a Mel scale to capture the power spectrum; and spectrograms, which provide a visual representation of frequency variations over time. In our implementation, we focused on computing the spectrogram, applying STFT to derive the magnitude spectrogram, and subsequently normalizing the features.Text encoding is a vital part of the preprocessing pipeline, particularly for mapping transcriptions to a format understandable by the model. Here each character in the transcription is encoded into numerical indices, where each unique character (such as letters, digits, punctuation marks) is assigned a corresponding integer. This encoding, along with the spectrogram features, forms the input to the ASR model. The process of both spectrogram generation and text encoding allowed us to prepare the audio and text data effectively, ensuring that the model receives a consistent, noise-filtered, and well-structured input for training and prediction. As we work with only numbers, we took characters $l \in \{blank, a, b, .., z, ?, !, space\}$. Any special characters not in defined characters are mapped blank, in a way we can say they are not considered. Finally we map them back to characters to find word error rate and evaluate model.

4. Model Architecture

4.1. Input Layer

The model takes a spectrogram as input, represented as a 2D matrix with dimensions (time, frequency), denoted as $x = (x_1, x_2, \ldots, x_T)$, where each x_t is a feature vector.

4.2. Convolutional Layers

Conv2D Layer 1: Utilizes 32 filters of size $[11, 41]$ with strides $[2, 2]$, reducing dimensionality and extracting features.

$$h_t^{(1)} = \text{ReLU}\left(\text{BatchNorm}\left(\text{Conv2D}(x_t, W_1)\right)\right)$$

Conv2D Layer 2: Employs 32 filters of size $[11, 21]$ with strides $[1, 2]$, refining the extracted features further.

$$h_t^{(2)} = \text{ReLU}\left(\text{BatchNorm}\left(\text{Conv2D}(h_t^{(1)}, W_2)\right)\right)$$

Batch normalization is applied for regularization after each convolution, followed by ReLU activation for introducing non-linearity.

4.3. Recurrent Layers (BiLSTM)

A stack of 5 bidirectional LSTM layers, each with 512 hidden units, is used to model temporal dependencies within the data, allowing the model to capture both past and future context. Dropout is applied between recurrent layers to mitigate overfitting.

The forward LSTM for the k-th layer is given by:

$$h_t^{f,k} = \overrightarrow{\text{LSTM}}(h_t^{(k-1)}, h_{t-1}^{f,k})$$

$$h_t^{f,k} = o_t^f \odot \tanh(c_t^f)$$

The backward LSTM for the k-th layer is:

$$h_t^{b,k} = \overleftarrow{\text{LSTM}}(h_t^{(k-1)}, h_{t+1}^{b,k})$$

$$h_t^{b,k} = o_t^b \odot \tanh(c_t^b)$$

The LSTM equations are defined as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{(Forget Gate)}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)}$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad \text{(Candidate Cell State)}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \quad \text{(Cell State Update)}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)}$$
$$h_t = o_t \cdot \tanh(c_t) \quad \text{(Hidden State Update)}$$

The bidirectional hidden states are computed as:

$$\overrightarrow{h_t}, \overrightarrow{c_t} = \text{LSTM}(x_t, \overrightarrow{h_{t-1}}, \overrightarrow{c_{t-1}}) \quad \text{(Forward LSTM)}$$

$$\overleftarrow{h_t}, \overleftarrow{c_t} = \text{LSTM}(x_t, \overleftarrow{h_{t+1}}, \overleftarrow{c_{t+1}}) \quad \text{(Backward LSTM)}$$

The final hidden state is obtained by concatenating the forward and backward hidden states(as in fig. 3:

$$h_t^k = [h_t^{f,k}; h_t^{b,k}]$$

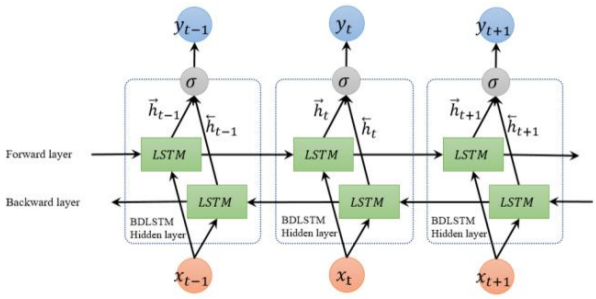

Figure 3. Bidirectional LSTM

4.4. Dense Layer

A fully connected dense layer processes the output from the final BiLSTM layer:

$$z_t = W_{\text{out}} h_t^5 + b_{\text{out}}$$

4.5. Batch Normalization

Batch Normalization is a technique used to normalize the inputs of each layer to reduce internal covariate shift, thereby stabilizing and accelerating training. It is applicable to both convolutional and fully connected layers, but not typically used directly in recurrent layers such as LSTMs.

Given an input x to a layer, Batch Normalization is defined as:

$$\text{BatchNorm}(x) = \gamma \frac{x - \mu_{\text{batch}}}{\sqrt{\sigma^2_{\text{batch}} + \epsilon}} + \beta$$

where: - $\mu_{\text{batch}} = \frac{1}{m}\sum_{i=1}^{m} x_i$ is the empirical mean over a minibatch of size m, - $\sigma^2_{\text{batch}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\text{batch}})^2$ is the empirical variance, - γ and β are learnable parameters that control scaling and shifting, respectively, - ϵ is a small constant added for numerical stability.

4.5.1 Application in Convolutional Layers

In convolutional layers, Batch Normalization is applied across all spatial dimensions and channels for each filter. The transformation is given by:

$$\text{BatchNorm}(h_t^{(k)}) = \gamma \frac{h_t^{(k)} - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}} + \beta$$

where μ_t and σ_t^2 are computed over the entire feature map output by a specific filter for the current minibatch.

4.5.2 Application in Fully Connected Layers

In fully connected (dense) layers, the affine transformation followed by Batch Normalization is expressed as:

$$z_t = W h_t + b, \quad \hat{z}_t = \text{BatchNorm}(z_t)$$

where W and b are the weights and bias of the layer, and $\hat{z}_t$ is the normalized output passed to the non-linearity.

4.5.3 Usage in Recurrent Layers (BiLSTM Considerations)

While Batch Normalization is not directly applied within recurrent layers like LSTMs due to temporal dependencies, Layer Normalization or Weight Normalization can serve as alternatives to stabilize training. In Layer Normalization:

$$\hat{h_t} = \frac{h_t - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}}, \quad \mu_t = \frac{1}{d}\sum_{j=1}^{d} h_t^j, \quad \sigma_t^2 = \frac{1}{d}\sum_{j=1}^{d}(h_t^j - \mu_t)^2$$

where normalization occurs over the hidden dimensions d instead of the batch.

4.6. Softmax Layer

The softmax layer produces a probability distribution over all possible characters or phonemes, including a blank token for CTC:

$$P(l_t|x) = \frac{\exp(z_t^l)}{\sum_{l' \in \mathcal{L}} \exp(z_t^{l'})}$$

5. Optimization

The model is trained using the Adam optimizer, chosen for its adaptive learning rate capabilities, allowing faster and more stable convergence during training. learning rate used is 10^{-4}

6. Loss Function

The model employs Connectionist Temporal Classification (CTC) loss, well-suited for sequence-to-sequence tasks with unknown alignments between input and output:

$$P(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} P(\pi|x)$$

The CTC loss is defined as:

$$\mathcal{L}_{\text{CTC}} = -\log P(l|x)$$

6.1. Training Objective

The overall training objective is to minimize the CTC loss over the entire dataset:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{CTC}}(l^{(i)}, x^{(i)})$$

7. Results

We performed optimization by using above defined training objective, with a batch size of 32 and 10 epochs, using P100 accelerator provided by Kaggle Platform, word error rate (WER) with each epoch are as in fig.4.

$$\text{WER} = \frac{S + D + I}{N}$$

where: S is the number of substitutions (incorrect words in the prediction), D is the number of deletions (missing words in the prediction), I is the number of insertions (extra words in the prediction), and N is the total number of words in the reference.

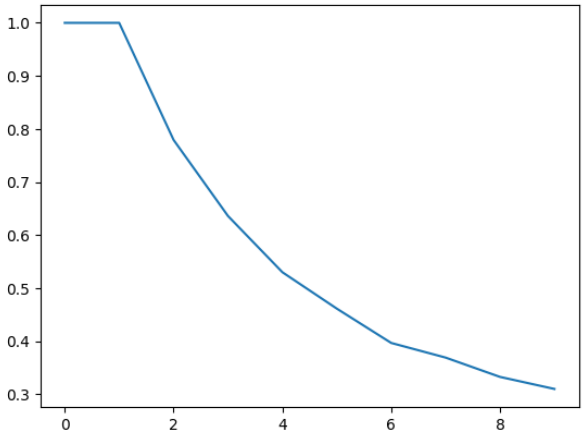

Figure 4. WER vs epochs

After 10 epochs we reached a WER of 0.3099. And few predictions after training are as follows:

Target: `but his despair must have been great as was evident from his attempt to strangle himself in the stationhouse`
Prediction: `but his diespare must have been great as was evident from his attempt to stringled himself inthestation house`

Target: `end quote mr bouck pointed out however that he had no reason to believe that any one federal agency had access to all this information`

Table 1. PSNR and SSIM values for different images

Model	WER
Our Model	0.3099
DS2	0.0906
DS2 (noisy)	0.1707

Prediction: `end quote mr boe pointed out however that he had no reason to believe that any one federal agency had acces to al this imforaion`

Target: `every prison was ordered to keep up an infirmary and the medical supervision was to be strict and continuous`
Prediction: `every prison was ordered to ke up and infirmery and the medical supervision was to be strict and continuous`

7.1. comparision

Original Deep Speech 2 model is implemented to achieve state of the art model, they used batch normalization at every layer and used sorta grad, where data is sorted from less size and less complex voice transcript to more complex data. Since DS2 uses large dataset they done experiments by adding noise to dataset as well. Final performance is tabulated in 1 with similar number of parameters for different models.

8. Conclusion

In this study, we developed a speech recognition model inspired by DeepSpeech2, incorporating convolutional and bidirectional LSTM layers. Our model achieved a Word Error Rate (WER) of 0.3099, which indicates promising performance in speech-to-text conversion. This result demonstrates the model's capability to handle speech data and transcribe it with reasonable accuracy. However, the performance can be further improved by exploring techniques such as data augmentation, hyperparameter tuning, and model optimization. Additionally, incorporating more advanced training strategies and larger datasets could potentially lead to better generalization and reduced WER in future iterations.

References

[1] Aditya Amberkar, Parikshit Awasarmol, Gaurav Deshmukh, and Piyush Dave. Speech recognition using recurrent neural networks. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–4, 2018. 2

[2] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin, 2015. 2

[3] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. 2

[4] Alex Bie, Bharat Venkitesh, Joao Monteiro, Md. Akmal Haidar, and Mehdi Rezagholizadeh. A simplified fully quantized transformer for end-to-end speech recognition, 2020. 2

[5] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, 2016. 2

[6] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition, 2020. 2

[7] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014. 2

[8] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7: 19143–19165, 2019. 2

[9] Jane Oruh, Serestina Viriri, and Adekanmi Adegun. Long short-term memory recurrent neural network for automatic speech recognition. *IEEE Access*, 10:30069–30079, 2022. 2

[10] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986. 1

[11] Vincent Roger, Jérôme Farinas, and Julien Pinquier. Deep neural networks for automatic speech processing: A survey from large corpora to limited data, 2020. 1

[12] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition, 2019. 2

[13] Yaxin Zhang, Michael D. Alder, and Roberto B. Togneri. Using gaussian mixture modeling in speech recognition. *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, i:I/613–I/616 vol.1, 1994. 1