Performing Speech Recognition using DeepSpeech

*DeepSpeech is an Automatic Speech Recognition deep model built by Mozilla which uses a combination of LSTM and CTC.

Mukul Malik
Dept. of CSE,
LNMIIT,
Jaipur, Rajasthan
21ucs133@lnmiit.ac.in

Parag Rachchh
Dept. of CSE,
LNMIIT,
Jaipur, Rajasthan
21ucs143@lnmiit.ac.in

Jeetaksh Gandhi
Dept. of CSE,
LNMIIT,
Jaipur, Rajasthan
21ucs098@lnmiit.ac.in

***Abstract*—Speech recognition technology has witnessed remarkable advancements in recent years, driven by the evolution of deep learning techniques. This project focuses on the training of a speech recognition system which uses Long Short-Term Memory (LSTM) networks. The primary objective is to train the model to accurately recognize spoken words from audio inputs. The project leverages a dataset of labeled audio recordings, employing techniques such as feature extraction, data preprocessing, and deep learning model construction. The model is trained on a split dataset and evaluated to assess its performance in terms of accuracy and loss. The outcome of this project not only contributes to the broader field of natural language processing but also provides a practical demonstration of the application of deep learning in speech recognition. Mozilla's "DeepSpeech" Automatic Speech Recognition model was chosen for this project.**

I. Introduction

In a world increasingly dominated by human-computer interaction, speech recognition plays a pivotal role in facilitating seamless communication between individuals and machines. This project seeks to address the fundamental challenge of accurately recognizing spoken words from audio inputs, catering to a diverse range of applications such as voice-controlled assistants, transcription services, and accessibility features for differently-abled individuals.

The primary problem at the core of this project is to train the DeepSpeech model to recognize speech. In a given audio signal, the challenge lies in deciphering the acoustic patterns associated with various spoken words, enabling the system to transform raw audio data into meaningful textual representations. This task is inherently complex due to variations in accents, pronunciation, and environmental noise. DeepSpeech is already trained on the Common Voice Dataset. This project aims to further improve pre-existing weights by **"fine-tune" training** on a new dataset "LibriSpeech". For our purpose, we chose the Delta segment. Mel-frequency cepstral coefficients (MFCCs) are utilized as feature representations.

By extracting relevant features, such as Mel-frequency cepstral coefficients (MFCCs), and training the DeepSpeech model on a labeled dataset, we aim to create a robust system capable of accurately transcribing spoken words. **A big part of the problem was adapting the LibriSpeech dataset to DeepSpeech format.**

Identify applicable funding agency here. If none, delete this.

II. Literature Review

Understanding the project requires a grasp of fundamental concepts in the fields of signal processing, machine learning, and deep learning.

- *Signal Processing*: The transformation of raw audio data into a format suitable for analysis is a critical aspect of speech recognition. Techniques such as Fourier Transform and Short-Time Fourier Transform lay the groundwork for feature extraction.
- *Machine Learning*: Supervised learning is a cornerstone, where models learn from labeled datasets. Classification algorithms are particularly relevant for mapping audio features to corresponding words.
- *Deep Learning*: Neural networks, especially recurrent neural networks (RNNs) and LSTMs, are essential for capturing temporal dependencies in sequential data. The architecture of these networks plays a crucial role in the success of speech recognition models. **This is the centric part of the project.**
- *Feature Extraction*: Mel-frequency cepstral coefficients (MFCCs) are widely used for representing audio features due to their effectiveness in capturing the spectral characteristics of sound.

Several approaches have been employed to tackle the challenge of speech recognition:

- *Hidden Markov Models (HMMs)*: Traditional models like HMMs have been used for speech recognition. However, they struggle to capture long-term dependencies in sequential data, limiting their performance in handling complex linguistic patterns.
- *Gaussian Mixture Models (GMMs)*: GMMs have been combined with HMMs for modeling acoustic features. While effective to some extent, they lack the capacity to learn intricate patterns in data, hindering their adaptability to diverse linguistic nuances.
- *Deep Neural Networks (DNNs)*: DNNs have shown promise in improving speech recognition accuracy by

automatically learning hierarchical representations of features. However, their performance is often limited by the sequential nature of audio data.

The aforementioned approaches, while valuable, face challenges that LSTM-based solutions aim to overcome:

- **Sequential Dependencies**: HMMs and GMMs struggle with capturing long-term dependencies in sequential data, resulting in suboptimal performance for speech recognition tasks with varying time intervals.
- **Feature Representation**: DNNs provide a notable improvement in feature representation but may fall short in handling the temporal aspects of speech signals, particularly in scenarios where context and timing are crucial.
- **Adaptability**: DeepSpeech, with its ability to capture temporal dependencies and nuanced patterns, seeks to overcome these limitations, offering a more adaptable and robust solution for speech recognition tasks.

III. Related Works

Early Approaches - Hidden Markov Models (HMMs): Initial attempts at speech recognition often relied on Hidden Markov Models (HMMs). Notable works include Rabiner's paper "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition" [1]. HMMs provided a foundation but struggled with capturing long-term dependencies in sequential data.

Transition to Deep Neural Networks (DNNs): As deep learning gained prominence, DNNs were applied to speech recognition. Dahl et al.'s work "Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition" (2012) [2] demonstrated the effectiveness of DNNs in feature learning. DNNs improved feature representation but faced challenges in modeling sequential dependencies.

Introduction of Long Short-Term Memory (LSTM) Networks: The seminal work of Graves and Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures" (2005) [3], introduced the use of Bidirectional LSTMs for phoneme classification, showcasing the potential of LSTMs in sequential tasks. The use of LSTMs became prominent in various natural language processing tasks.

State-of-the-Art: The current state-of-the-art in speech recognition involves end-to-end models, often combining convolutional neural networks (CNNs) and recurrent networks. The Listen, Attend, and Spell (LAS) model by Chan et al. (2016) [4] and Transformer-based models like DeepSpeech 2 by Amodei et al. (2016) [5] represent the forefront of accuracy in large-vocabulary continuous speech recognition (LVCSR).

IV. METHODOLOGY

We're training the DeepSpeech model to perform Speech recognition. DeepSpeech is pre-trained on the Common Voice Dataset. We used those weights as a beginning point and fine-tuned them on a new dataset (LibriSpeech).

Now, we explain the structure of DeepSpeech. DeepSpeech was originally proposed by Awni Hannun et al. in their paper titled "Deep Speech: Scaling up end-to-end speech recognition". However, the engine currently differs in many respects from the engine it was originally motivated by. The core of the engine is a recurrent neural network (RNN) trained to ingest speech spectrograms and generate English text transcriptions.

Let a single utterance x and label y be sampled from a training set

$$S = (x(1), y(1)), (x(2), y(2)), ...$$

Each utterance, $x(i)$ is a time-series of length $T(i)$ where every time-slice is a vector of audio features, $x(i)t$ where $t = 1, \ldots, T(i)$. It uses MFCC's as our features; so $x(i)t$, p denotes the $p - th$ MFCC feature in the audio frame at time t. The goal of the RNN is to convert an input sequence x into a sequence of character probabilities for the transcription y, with $y^t = P(ctx)$, where for English $cta, b, c, ..., z, space, apostrophe, blank.$

The RNN model is composed of 5 layers of hidden units. For an input x, the hidden units at layer l are denoted $h(l)$ with the convention that $h(0)$ is the input. The first three layers are not recurrent. For the first layer, at each time t, the output depends on the MFCC frame xt along with a context of C frames on each side. (We use C=9 for our experiments.) The remaining non-recurrent layers operate on independent data for each time step. Thus, for each time t , the first 3 layers are computed by:

$$h(l)t = g(W(l)h(l1)t + b(l))$$

where $g(z) = minmax0, z, 20$ is a clipped rectified-linear (ReLu) activation function and $W(l)$, $b(l)$ are the weight matrix and bias parameters for layer l . The fourth layer is a recurrent layer. This layer includes a set of hidden units with forward recurrence, $h(f)$:

$$h(f)t = g(W(4)h(3)t + W(f)rh(f)t1 + b(4))$$

Note that $h(f)$ must be computed sequentially from t=1 to $t = T(i)$ for the i-th utterance. The fifth (non-recurrent) layer takes the forward units as inputs

$$h(5) = g(W(5)h(f) + b(5)).$$

The output layer is standard logits that correspond to the predicted character probabilities for each time slice t and character k in the alphabet:

$$h(6)t, k = y^t, k = (W(6)h(5)t)k + b(6)k$$

Here $b(6)k$ denotes the k-th bias and $(W(6)h(5)t)k$ the k-th element of the matrix product. Once we have computed a prediction for y^t,k , we compute the CTC loss $L(y{\cdot}y)$ to measure the error in prediction. (The CTC loss requires the blank above to indicate transitions between characters.) During training, we can evaluate the gradient $L(y{\cdot}y)$ with respect to the network outputs given the ground-truth character sequence y. From this point, computing the gradient with respect to all of the model parameters may be done via back-propagation through the rest of the network. We use the Adam method for training.

The complete RNN model is illustrated in the figure below.

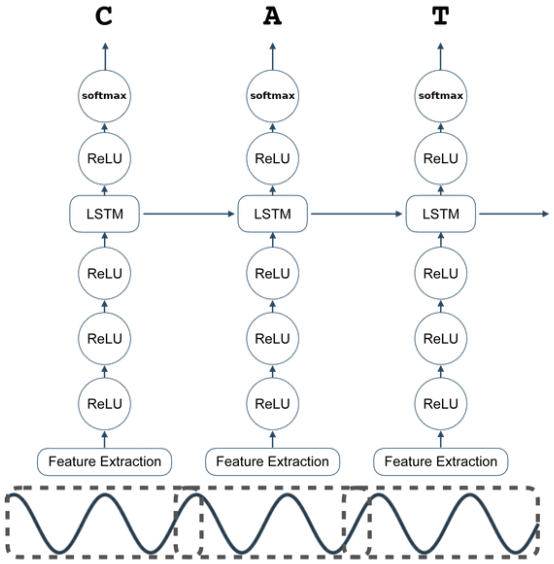

Fig. 1. DeepSpeech Architecture

The model is trained using a connectionist temporal classification (CTC) loss function. CTC allows the alignment of variable-length input sequences to variable-length output sequences, making it suitable for tasks like speech recognition where the duration of spoken words may vary.

To prevent overfitting and enhance generalization, DeepSpeech incorporates regularization techniques. Regularization methods like dropout may be applied during training, randomly disabling some neurons to prevent the network from relying too heavily on specific connections.

To extract features, Mel Spectrograms are computed. Raw audio samples are loaded from an audio file. Optionally, a pre-emphasis filter may be applied to emphasize higher-frequency components. The audio signal is divided into short, overlapping frames using a windowing function (e.g., Hamming window). The log mel spectrogram is computed after that. The logarithm of the mel spectrogram is taken. Optionally, feature normalization may be applied to ensure that the features have a standardized scale. Sequences of features are padded to have a consistent length.

V. Experimental Setup

Before we begin to training the model, we defined the following steps to be done:

- (S1) **Environment Setup**: To train the DeepSpeech model, we first need to setup an environment for its execution.
- (S2) **Dataset Adaptation**: The LibriSpeech dataset needs to be formatted according to DeepSpeech format.
- (S3) **Training**

A. Environment Setup

A docker container created from the official DeepSpeech image was created on an NVIDIA-DGX server to create the training environment.

B. Dataset Adaptation

LibriSpeech is a corpus of approximately 1000 hours (WE chose 100 hours) of 16kHz read English speech. The LibriSpeech dataset consists of multiple folders each of which have audio files in .flac files. Each such folder has a .txt transcript files that contains all transcripts of all audio files. In order to train DeepSpeech, the training data must be in a very different format. Firstly, all audio files must be in a single location for a particular split. So, there should be 3 spits: train, dev and test, each of which contains their audio files. Secondly, the audio files MUST be in .wav format (sampled at 16KHz). The transcriptions must be contained in one .csv file for each split in the format (Path to WAV File, Size of file in bytes, Target Transcript).

To do this processing, we firstly used a C program titled "*waver.c*" to find and move all audio files of a particular split from LibriSpeech's nested folders to one common split directory. After that, we used a Windows batch script (*ffscript.bat*) to use FFMPEG to convert all .flac files to .wav files (.bat). Next up, transcripts were processed. The C program "*texter.c*" and "*processor.c*" created one .txt file with name of the .wav file containing its transcript for each .wav file (by looking into original transcripts). The C program "*csv.c*" then created a .csv file for each split by combining information from all the .txt files. The net size was 12GB in the end with over 100 hours of audio.

This processing was a big part of this project as it involved multiple programs and scripts. The entire 100 hour segment was properly adapted to DeepSpeech format by performing many operations on each file.

C. Training

Training was carried out by using pre-trained weights. The model was trained for 5 epochs.

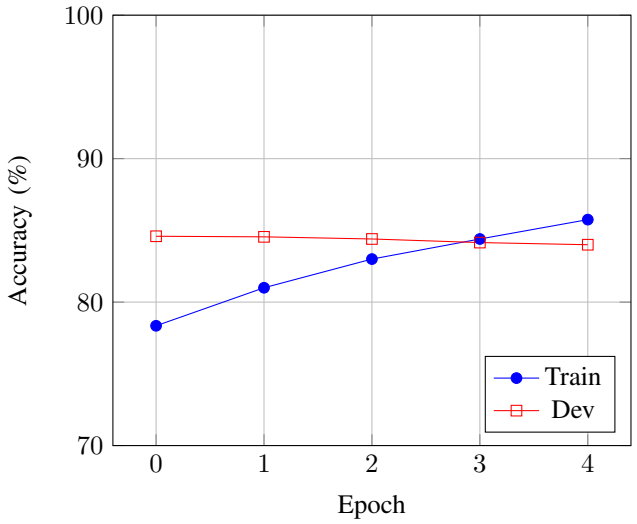

Fig. 2. Train and Dev Set Accuracy Over Epochs

RESULTS

We trained DeepSpeech from previously released official checkpoint for 5 epochs on our LibriSpeech segment of 12GBs. Each epoch took about 1.5 hours. The following table summarizes the results of training:

Epoch	Train Accuracy	Dev Accuracy
0	78.35%	84.59%
1	81%	84.55%
2	83%	84.40%
3	84.4%	84.15%
4	85.75%	84%

TABLE I
RESULTS

Clearly, the model starts to overfit a bit as we progress through the epochs. However, the weights obtained at Epoch 3 seem to be a good tradeoff between train and dev accuracy. For a loss of 0.4% on dev set, we get a whopping increase of 6% on train set! Thus at epoch 3, we have the best fine-tuned weights.

ABLATION STUDY

An ablation study could not be performed unfortunately due to model constraints as explained in README.txt.

CONCLUSION

In conclusion, this paper has delved into the intricacies of training DeepSpeech, a state-of-the-art automatic speech recognition (ASR) system. Our exploration has encompassed various aspects of the training process, shedding light on the nuances that significantly influence the performance of the model.

The effectiveness of DeepSpeech is rooted in its sophisticated architecture, comprising deep neural network layers that capture intricate patterns in the input spectrogram representations. Our experiments have emphasized the critical role of these layers, showcasing the delicate balance between model complexity and generalization.

Training data, a cornerstone of any machine learning model, has been a focal point of our investigation. The ablation study on training data size has illuminated the model's ability to generalize across different speech contexts. Our findings underscore the importance of a judiciously chosen training dataset, striking a balance between richness in diversity and computational feasibility.

In essence, this paper contributes to the collective knowledge surrounding DeepSpeech training, aiming to foster a deeper comprehension of the system's inner workings. Through our systematic analysis and ablation experiments, we pave the way for a more nuanced and informed approach to training deep learning models for automatic speech recognition, ultimately propelling the field towards enhanced performance and broader applicability.

REFERENCES

[1] Rabiner, L. R. (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." Proceedings of the IEEE, 77(2), 257-286.

[2] Dahl, G. E., Yu, D., Deng, L., Acero, A. (2012). "Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition." IEEE Transactions on Audio, Speech, and Language Processing, 20(1), 30-42.

[3] Graves, A., Schmidhuber, J. (2005). "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures." Neural Networks, 18(5-6), 602-610.

[4] Chan, W., Jaitly, N., Le, Q., Vinyals, O. (2016). "Listen, Attend and Spell." arXiv preprint arXiv:1508.01211.

[5] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... Jun, B. (2016). "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin." In Proceedings of the 33rd International Conference on Machine Learning (Vol. 48, pp. 173-182).

CONTRIBUTION OF TEAM MEMBERS

Name	ID	Contribution
Mukul Malik	21ucs133	34%
Jeetaksh Gandhi	21ucs098	33%
Parag Rachchh	21ucs143	33%

TABLE II
CONTRIBUTORS TABLE