Surge 2023

Final Report

Automatic Speech Recognition for Indic Languages

Name: Raghav Karan
Application Number: 2330001
Mentor: Dr. Vipul Arora

Introduction

The title of the project was initially decided upon as Automatic speech recognition (ASR) for Indian English, but early in the duration the focus shifted to ASR for Hindi. The project was about developing an ASR model for Hindi language and utilizing it for a range of audio transcription. The project has been going on for some time with Professor Vipul Arora and Prasar Bharati, India's state-owned public broadcasting. Prasar Bharati have a lot of spoken data as they have been offering their services for a long time now. Their extensive array of network includes All India Radio-Akashvani and Doordarshan among others. They require an ASR model for the extensive transcription of data they have, which is mostly long form audios, which posed another challenge. The details of the project will follow later in the report. After working and completing tasks at hand with my mentor's team on this project, the focus again shifted to a global ASR challenge, MADASR'23, organized by ASRU and IISC Bangaluru. The challenge involved adapting ASR models for low resource Indic Languages-Bhojpuri and Bengali.

My contributions

I worked on specific aspects of the projects as a part of the teams that were already working on them. I was initially tasked to get familiarized with connectionist-temporal-classification which is an important technique to calculate losses and propagate them in speech recognition. The model that was developed from scratch and trained was based on Conformer, which is the state-of-the-art architecture for speech tasks currently.

Generating Inference for long-form audio

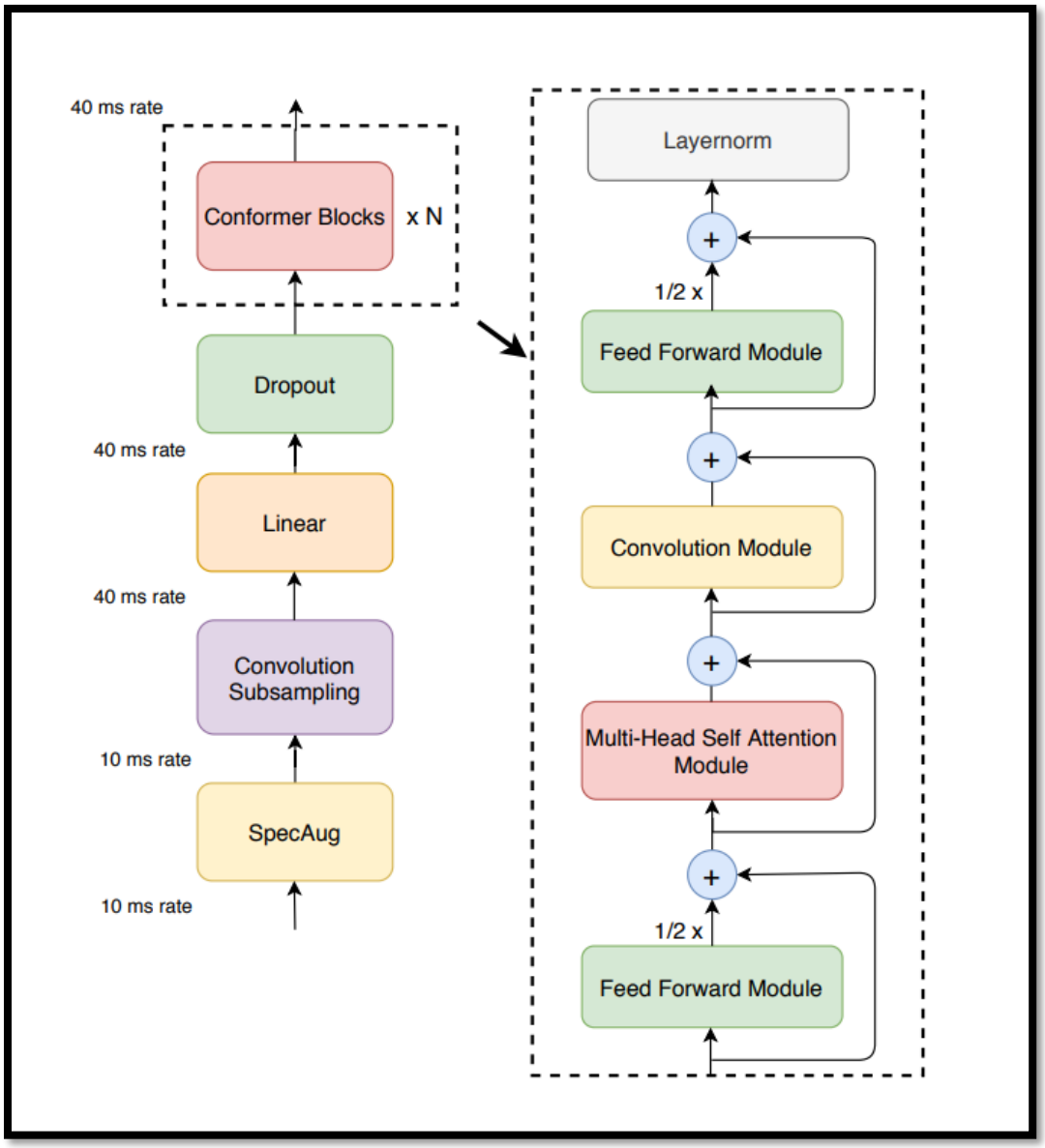
After getting comfortable with pre-requisites, I was provided several research papers about getting inferences for long form audio. The challenge here is that models are trained on audios that are not very long. In our case the model was trained on audios that were 16 seconds of duration each. Also, the size of audios the models can process in a single go is limited by the system configuration, and thus long audios require some extra efforts in this regard.

I worked on the below mentioned methods, details, and results of which are also shown. The results were obtained from few Man ki Baat audios of our respected PM, which are around 30 mins each, and few news audios of 15 mins each. Mann ki Baat audios did not have any background music whereas news audios had a lot of BGM. All the audios were provided by Prasar Bharati. The errors will be reported as **WER** which stands for **WORD ERROR RATE**.

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

where...
S = number of substitutions
D = number of deletions
I = number of insertions
N = number of words in the reference

Metric

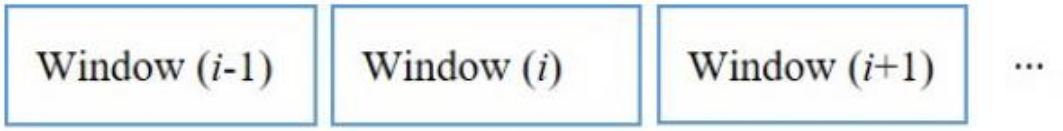


Conformer architecture [1]

Experiments with Long-Form Audio

- **Method 1: Hard Chunking**

In this method, the long audio is broken into smaller segments or windows of fixed length. This length is kept at a suitable value less than the maximum length the model is capable of handling without the GPU going out of memory. Individual transcriptions are simply concatenated together with space in between. [2]



Performance:-

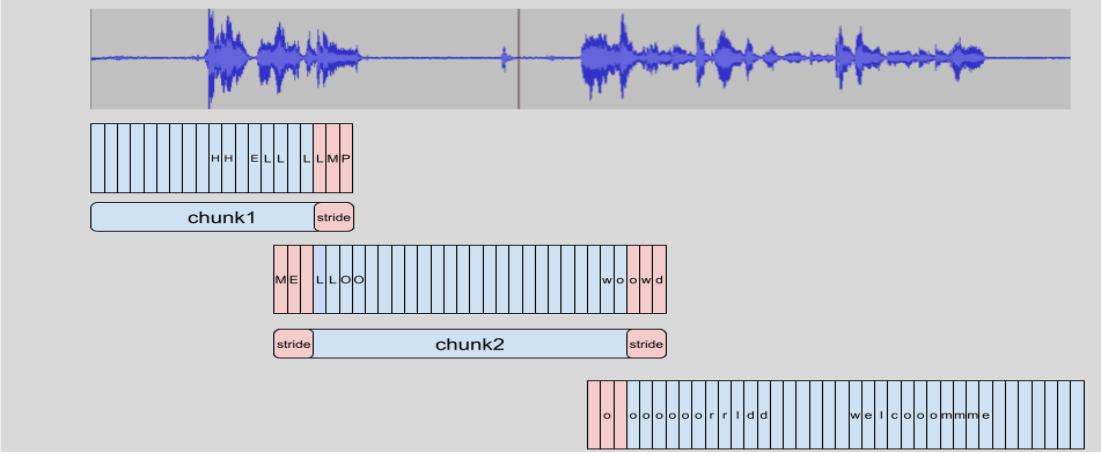
WER without background music : 22.16

WER with background music : 41.53

Time taken: 14 seconds per iteration

- **Method 2: Chunking with strides**

The input audio waveform is broken into smaller chunks with some overlap. Individual audios are fed into the model and frame-by-frame transcriptions are obtained. The transcriptions are sliced from the starting and the ending to delete the overlapping portions. They are then collapsed and stitched into a single transcription.



[3]

Performance:-

WER without background music : 21.35

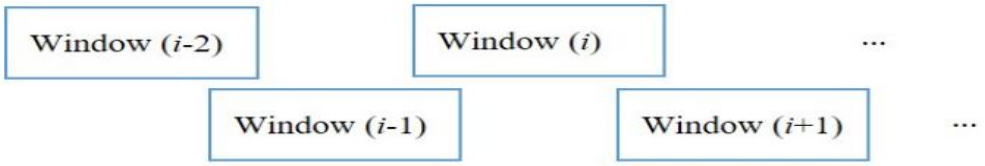
WER with background music : 42.23

- **Method 3: Partially overlapped inference and stitching using edit distance with modified operation costs over individual predictions.**

Input audio is chunked into small waveforms with some overlap. The uncollapsed predictions for each are obtained from the model. The predictions are then stitched together based on an edit-distance based score matrix. The chunk transcriptions are stored in a list. For stitching them together, the list is iterated repeatedly, and in each iteration, we do pairwise alignments of the transcriptions. For example, if the list contains 10 individual chunk transcriptions, in the first iteration, we merge the pairs (1,2),(3,4),(4,5),(5,6)….(9,10) and replace them with single merged transcripts, i.e., (1,2)—are replaced by a single transcription. In the next iteration we repeat the same steps for the novel merged transcripts, and this process continues until the list contains only one transcription. The method is computationally heavy, and the results were unsatisfactory. [2]

- **Method 4: Partially overlapped inference and stitching using edit distance with modified operation costs only over the overlapped window.**

This method is similar to above, the only difference is that the score matrix is generated for the overlapped portions only rather than the entire chunk transcription. The non overlapped portions and overlapped portions are concatenated together. The previous method, in principle, is superior to this as it has more context, but this method reduces the time complexity over the previous, though still higher than some, enabling us to obtain faster inferences. [2]

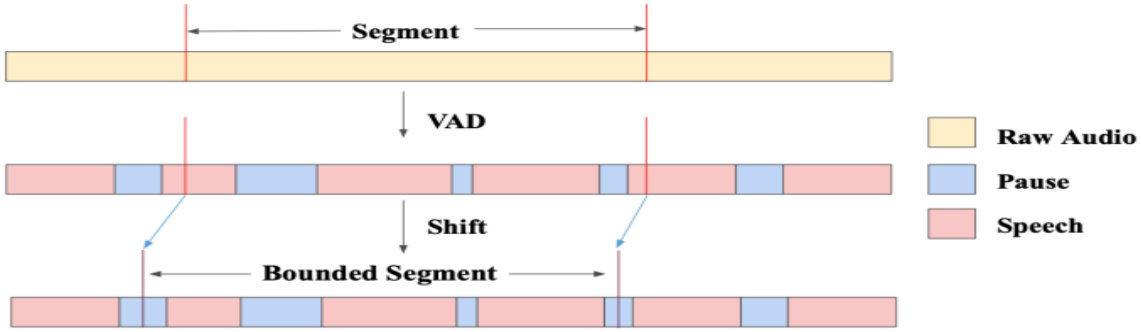


Performance:-

Overlap=10 %	*Overlap=50%*
WER without background music : 22.4	*WER without background music : 20.76*
WER with background music : 46.67	*WER with background music : 39.81*

- **Method 5: Voice Activity Detection (VAD) overlapping inference**

The input audio is passed into a VAD model which outputs the start and end timestamps of non-silent regions. These timestamps are used to generate chunks which begin and end at the middle of silent regions and are of lengths the model is capable of handling. The individual transcriptions are obtained and concatenated together with spaces, to represent silence, in between. This method does not help with audios that contain background music, as the silences are occasional, and the individual chunks even with VAD are mostly longer than the model can manage. The method was found to be extremely slow. [3]



WER without background music : 25.01

Time Taken: 15 minutes per iteration

Method 6: Chunking using CTC decoder outputs.

This method discovers silences with the help of CTC decoder outputs. We begin at the start of the audio, and hard chunk it at a particular duration. This hard chunk is fed into the model and the uncollapsed frame-by-frame predictions are obtained from the CTC decoder. The chunk prediction is then iterated from the end to search for regions of long silence represented by continuous blank tokens. We stop at the first long silence we find. This silence region is then used to modify the current hard chunk by terminating it at the center of found region of silence. The new end point is noted which would serve as the starting point of the next hard chunk in the next iteration. The inferences are obtained for the modified chunk again. If no long silence is obtained for a particular hard chunk, then we continue with its initial prediction. The entire process is repeated until we process the entire audio. Below is snap of CTC decoder output for an audio chunk.

Performance:

WER without background music : 17.68

WER with background music : 33.67

Time taken: 19 seconds per iteration

The results from method 6 were extremely impressive along with reasonably practical time taken per iteration and thus the team decided to proceed with this method.

Method	Word Error Rate	
	Audio without BGM (%)	Audio with BGM (%)
Hard Chunking	22.16	41.53
Chunking with strides	21.35	42.23
Partial Overlapping (10% overlap) (50% overlap)	 22.4 20.76	 46.67 39.81
VAD based silence	25.01	-
CTC based silence	17.68	33.67

Beam Search and Language Model

The method was integrated with beam search and language model which reduced the errors remarkably.

(Without BGM)

WER with Beam search: 11%

WER with Beam search and language model: 7%

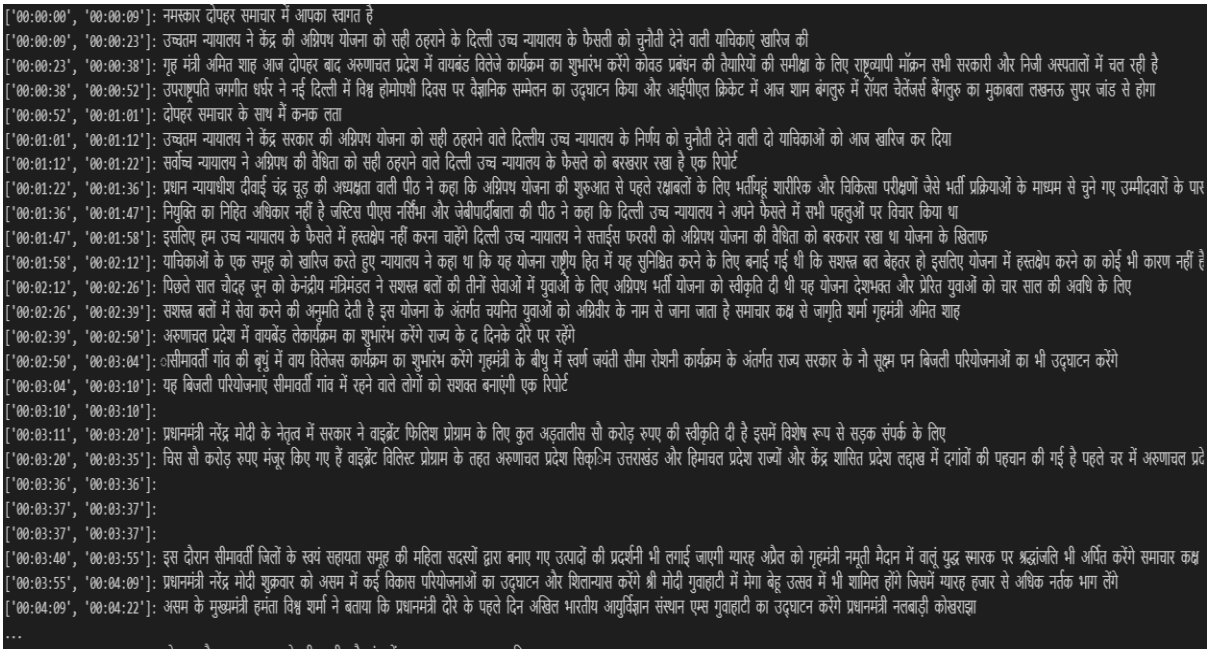
Challenge with beam search:

It requires SoftMax probability matrix as input, as it must go through various probable sequences to find the best one. Therefore, I had to find a way to go back from the transcription to probability matrix. So, I first found silence on uncollapsed indices, then then sliced the particular dimension of the probability matrix accordingly and then found the most confident sequence by providing that matrix to beam search.

Generating timestamps for subtitling

The next task was to generate chunk transcriptions with time stamps. The intricate dependence of frame numbers and the time duration were explored and utilized to generate the time stamps. The raw audio is used by the model as a Mel-spectrogram which has generally 16000 (called the sample rate) samples generated per second, which is subsequently resampled by convolution layers by a factor of 4. The samples are ultimately condensed into frames. The time stamps were obtained with the help of chunk starting and ending points obtained from CTC decoder's outputs. We find the sample number the chunk begins and ends at, which is divided by the sample rate to obtain the time stamp in seconds. Time stamps have many applications, the most prominent being subtitles srt files.

Below are the results obtained:



ASRU MADASR Challenge

After finishing with long audio inference, I started working on an ASR challenge, MADASR'23 (**M**odel **AD**aptation for **ASR** in low-resource Indian languages). The challenge was to adapt existing ASR models for low resource Indic languages: Bhojpuri and Bengali, which have much less data available to train.

Transliteration experiments

The initial approach the team decided was to utilize a pre-trained model and fine-tune it for the new language. Bhojpuri and Hindi both use Devnagari script and thus we can directly use a Hindi pre-trained model and fine- tune it with limited Bhojpuri data.

However, Bengali has an altogether different script. We explored transliteration to English so as to use an English pre-trained model to fine tune with transliterated Bengali data. Different libraries were experimented with, and results are briefly specified below.

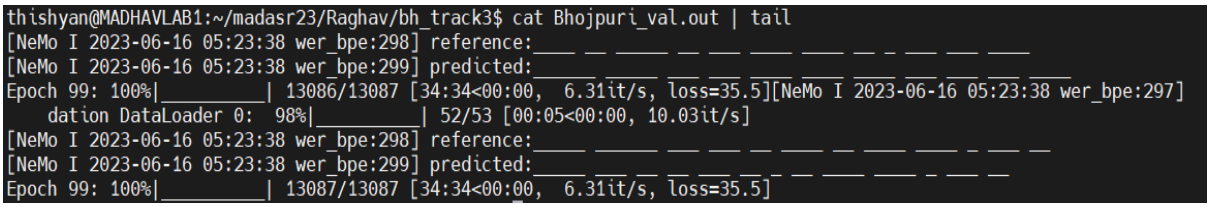
Bengali-English

1. Ai4bharat transliteration tool:	*WER: 33.4 ; CER=11.5 ; Time=1.46s/it*
2. Polyglot:	*Could not work as it required many dependencies in the system and issues with that could not be resolved.*
3. Google transliterate:	*WER: 30.7 ; CER=11.3 ; Time=4s/it*
4. Indicnlp:	*Works only for Indic to Indic, does not support Latin transliteration.*

ASR for Bhojpuri: Fine-tuning pre-trained model

Nemo Pre-trained model

I worked on fine-tuning Hindi pre-trained model for Bhojpuri. I explored the pre-trained models provided by Nvidia's Nemo toolkit and discovered that it contains a Hindi conformer model trained on 128 sub-word piece tokens. The Bhojpuri dataset was prepared as per the requirements of the model. The entire data was made into manifest files (Json), that contained the path to audio, duration, and the reference transcript. The vocabulary is supposed to be updated as per our new dataset, and thus I stuck with 128 token vocabulary as that could use the weights of the model directly. The convergence took 3 days for 100 epochs of training on the data. Below is a snapshot of the terminal when the training was completed.



The team later decided to train a conformer model previously used for Hindi language from scratch and tally the results. The training took a few days, but the results were better than nemo pre-trained model and WER was 25% on validation dataset.

Track	Bhojpuri		Bengali	
	WER	CER	WER	CER
Greedy decoder	27.04	8.98	26.96	7.89
Greedy decoder +Beam Search	26.89	9.72	26.40	9.33
Greedy decoder +Beam Search + LM	20.91	7.78	12.29	5.27

Test-time-adaptation

After the model was trained, the team decided to experiment with test-time adaptation (TTA) to obtain reliable results with the test data.

Source free single utterance TTA (SUTA) : This method aims to adapt a trained model to each new audio in the test set. It generate the labels for the test audio and calculate entropy loss, class confusion, and try to minimize these losses. Entropy minimization sharpens the class distribution and minimum class confusion reduces correlations between classes. Only the layer normalization and encoder weights are updated for better results. [4]

ULCA dataset

ULCA provides an open-source Bhojpuri dataset that contains 60 hours of labelled data. I prepared the metadata files for training Bhojpuri model for the challenge.

Active learning

Generated baseline results for a new method being developed in Lab for active learning in automatic speech recognition. I implemented an existing active learning strategy to obtain results that will be used to compare with the novel method developed in lab.

I experimented with path probabilities. Generated inference for each audio in the dataset. Calculated the confidence scores of the predictions by calculating the total probability of the output sequence.

The method in brief:

$$ppath = \frac{\log\big(\mathrm{P}(y|x)\big)}{len(y)}$$

$$P(y|x) = \prod_{y_i \in y}^{.} P(y_i|x)$$

Y_i 's are the SoftMax probabilities of the char chosen for a particular frame. We multiply them all together to get the probability of the sequence that has been selected for the current audio. The inferences model is less confident about would have lesser total probabilities as the

individual probabilities will be distributed among the classes. We then selected a specific number of audios the model is the least confident about. We get this data annotated and retrain the model. This will continue until satisfactory results are achieved. [5]

Acknowledgment

I would like to express my gratitude to Surge for including me in their program in the first place. I would like to take this opportunity to express my thanks and respect to my mentor, Professor Vipul Arora, for making this opportunity possible for me. My time spent in the laboratory has provided me with an extremely enlightening experience overall. The trip was chock full of educational opportunities. I would also like to express my gratitude towards Rathna Ma'am, Post-Doc Research fellow and Thishyan Raj, MSR student at the Madhav Lab. They have provided me with guidance and mentoring throughout the entirety of the project.

References

[1] Anmol Gulati et al., "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Interspeech 2020*.

[2] Tae Gyoon Kang et al., "PARTIALLY OVERLAPPED INFERENCE FOR LONG-FORM SPEECH RECOGNITION," *ICASSP 2021.*

[3] Jinhan Wang et al., "VADOI: VOICE-ACTIVITY-DETECTION OVERLAPPING INFERENCE FOR END-TO-END," *ICASSP 2022 .*

[4] G. L. S. &. L. H. Lin, "Listen, Adapt, Better WER: Source-free Single-utterance Test-time Adaptation for Automatic Speech Recognition," *ArXiv. /abs/2203.14222.*

[5] Jihwan Bang et al., "BOOSTING ACTIVE LEARNING FOR SPEECH RECOGNITION WITH NOISY," *arXiv:2006.11021v2.*