IIT Bombay

CS 753

Automatic Speech Recognition

ASR : Final Project Report

Author:	*Roll No:*
Suraj Kumar	18305R008
Nikhi Saini	183059006
Atul Sahay	18305R003

July 29, 2020

Contents

1 Task definition

To implement and evaluate Automatic Speech Recognition (ASR) system for low resource Indian Languages is quite challenging because, less availability of Indian language's resources deprived the state of the art ASR network in gaining any fruitfull insights from it. We hypothesize to use a Teacher-Student or Parent-Child learning mechanism where insights learn from one can be imparted by other, more profoundly we want to transfer learning from Parent corpus (normally, which has large resource size than child's) to child's corpus to build better models for low resource languages. In addition, we also hypothesize that this learning can be further intensified by making use of the speaker adaption technique.

1.1 Input-Output Behaviour

1.1.1 Input

Recorded human speech in Gujarati language (.wav file format)

1.1.2 Output

Text transcription of speech which is fed into ASR system.

1.2 Evaluation Metric

There are two commonly used metric for evaluation of ASR systems: BLUE(Bilingual Evaluation Understudy) Score and WER(Word Error Rate)

1. BLEU : Output is the utterance of input (audio file) in text format. We have reference utterances for test data. BLUE score is used to evaluate texts.
2. CER : Many papers publish CER. So for using them as baselines, we will also evaluate our models on CER.

1.3 Dataset

We have used 3 datasets:

- Microsoft Training Corpus (Gujarati) - 40 hours
- OpenSLR Male Corpus (Gujarati) - 5 hours
- OpenSLR Female Corpus (Gujarati) - 5 hours

2 Prior work

2.1 Multi-task CTC with language-specific character set (3)

MTL technique has been used for both cascaded and end-to-end ASR systems. Multi-task learning (MTL) improves generalization performance in low-data resources. Training a model on multiple related task serves as inductive bias to improve its performance. In this architecture lower layers are shared among different languages while the output layers are trained specific to languages. The overall loss function for multi-task CTC is a combination of single-task objective functions.

2.2 Multilingual CTC with universal character set (3)

There are many Indian languages which share some common graphemes and phonemes. This architecture uses a universal output label set consisting of the union of a characters from multiple languages. Characters that are common across multiple languages are trained based on all relevant data while language specific characters are trained with data from that language. To further enhance accuracy of model we use language specific gating unit because same grapheme may have different phonetic representation in different languages.

2.3 Multilingual LAS Model(2)

End-to-end LAS model can also be used in multilingual settings which can be trained jointly, conditional or via multi-task methods. In joint model training we do not give any explicit indication that training data set is composed of different languages but it works fairly good. Another variation of joint model is to train the model for different but related tasks like recognize speech and predict it's language. Finally the last variant (i.e. conditional model) utilizes language ID during inference. The conditioning is achieved by feeding in language embedding as an input to first layer of encoder or decoder or both. Here language ID is not used as a part of training.

2.4 Transformer Multilingual Modeling(1)

Multihead Attention (MHA) and position-wise, fully connected layers are used in both decoder and encoder of the transformer. The encoder has N identical layers and each layer has two sub-layers (MHA, position-wise fully connected feed-forward network). The decoder is similar to encoder except it has third sub-layer to perform MHA over the output of encoder stack. For modeling in multilingual settings a multilingual unit is required. Sub-words which are generated by BPE is used as

multilingual unit. There are extensions of this multilingual ASR Transformers which expand the vocabulary to include list of special symbols.

3 Methodology & Implementation

To investigate our hypothesis, we made use of the (Convolutaional Neural Network - Long Short Term Memory) CNN-LSTM encoder decoder network. Various stages involved can be seen from the below described subsection.

3.1 Segmentation

First a data-set taken is clustered into a number of shards on the basis of similar characteristics, for say characteristics can be the time duration of each audio files, now a mixer component on the basis of provided probabilistic distribution [see Figure 1] to it, forms a training batch with a good mix from different shards. In our case we have provided a uniform distribution to the mixer component.

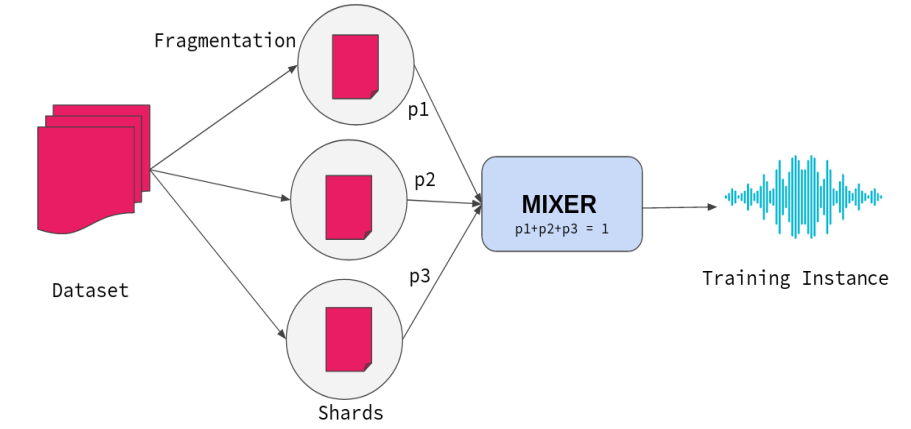

Figure 1: Architectural flow of segmentation

3.2 Feature Extraction

Each of these training instances are then fragmented into a number of time frames, in our case each of such frames are of 6 ms time width. To extract spectral features we first converted audio samples present in time domain to frequency domain using Short Time Fourier Transform (STFT) component. [see Figure 2] Once we got an access to spectrogram of each of those frames, we convolve over them to produce a

dense vector representation of them. These dense vector representation is analogous to word embedding used in NLP domain.

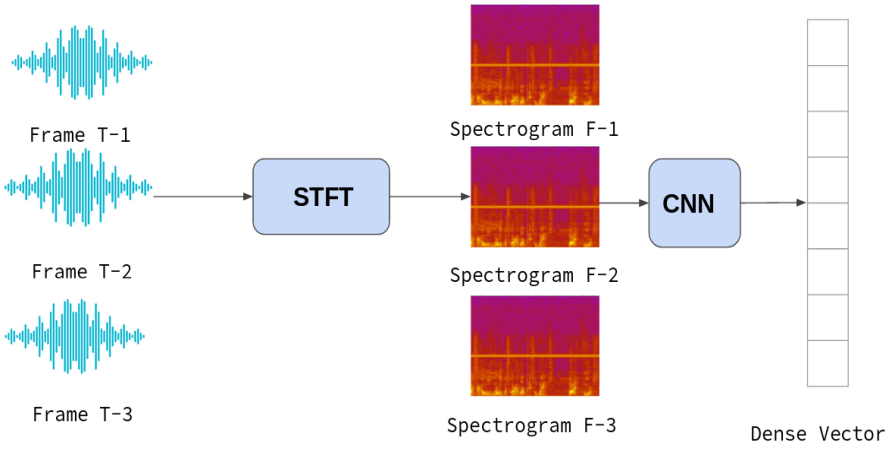

Figure 2: Architectural flow of feature extraction

3.3 Encoding

Feature Tensor build over dense vector representations are then fed to stacked LSTM layers to get the final characteristic representation of the audio frames. [see Figure 3] This encoding stage is very crucial because in this stage only a linguistic as well as contextual characteristics are extracted which further can be used as deciding parameter in later stages. LSTM cells on other hand are efficient to easily handle the contextual preserving task.

3.4 Decoding

Each decoder cell attended over all time steps by making use of the hidden state vectors of the last layer of stacked LSTM encoder.

$$e_i = tanh\,(W_e \times \vec{s_i}) \tag{1}$$

$$\tilde{a_i} = \exp(W_a \times e_i) \tag{2}$$

$$a_i = \frac{\tilde{a_i}}{\sum_{i=1}^{T}(\tilde{a_i})} \tag{3}$$

$$\vec{v_c} = \sum_{i=1}^{C}(a_i \dot{\vec{s_i}}) \tag{4}$$

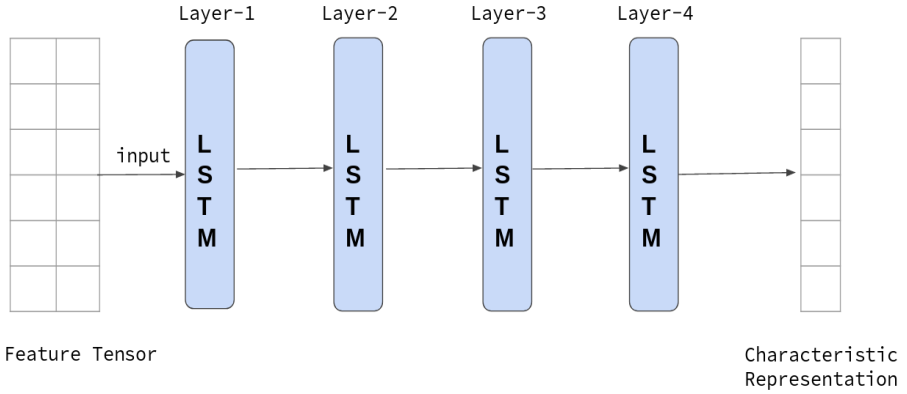

Figure 3: Figure showing encoding of feature tensor using stacked LSTM layers

$$C_i = v_c + O_{i-1} \tag{5}$$

where T is total time steps, s_i is the i^{th} hidden state vector, o_{i-1} output vector produced from decoder at $i-1^{th}$ time step. [See Figure 4]. While figure 5 shows the Unrolling of decoder cell into multiple time-steps with decoding one character at a time,o_i .

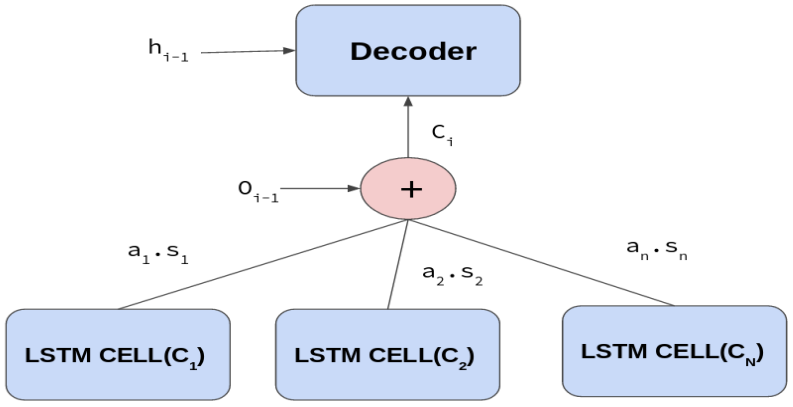

Figure 4: Globally attended context vector formulation by decoder cell

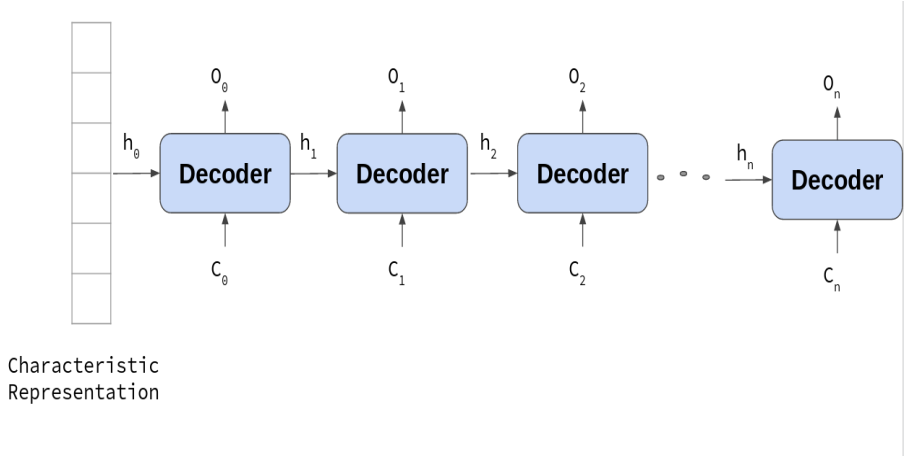

Figure 5: Unrolling of decoder cell into multiple timesteps

3.5 Transfer Learning

1. We first train the above discussed model on a parent language which has large amount of data set [Microsoft Speech Corpus (Gujarati)] and the language is closer to target (child languages).[Open SLR Female (Guj.) and Male (Guj.)] separately. [see the result table 1]
2. Then we fine tuned the pre trained model [Parent language] using one of the child languages. [see the result table 1]
3. We apply transfer learning approach by freezing the encoder part of the model and then simply fine tune only the decoder part of the model. [see the results table 1]

3.6 I-Vectors(5)

I-Vector is a high dimensional vector that captures speaker related characteristics/features which is useful for speaker adaptation. We are using speaker adaptation to further improve accuracy (CER) of low resource ASR system. In our system, we are using 512 dimensions i-vectors. Kaldi toolkit is used to extract i-vectors. Following are the steps used to extract it:

- Data Preparation
- Extract MFCC & VAD
- Train UBM & I-Vector Extractor
- Extract I-Vector

Entire pipeline of Kaldi for i-vector extraction is shown in figure below 6. Once we

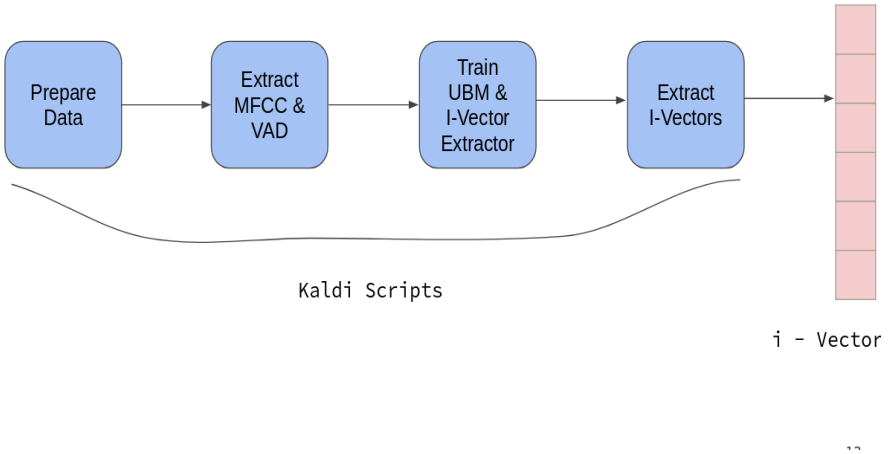

Figure 6: I-Vector Extraction Process

extract i-vectors, we concatenate/sum it with characteristic representation and feed to first timestamp of decoder as shown in figure below 7.

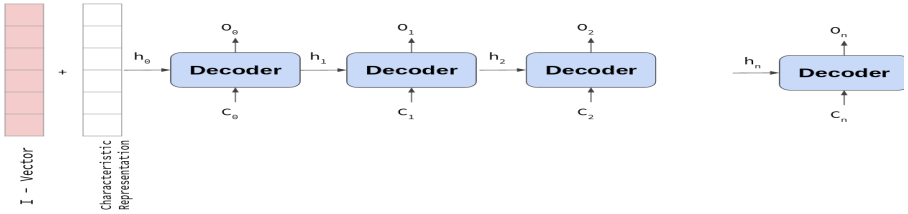

Figure 7: Adding i-Vectors during Decoding

4 Experimental Setup

All the experiments were run on Ubuntu 18.04 with NVIDIA GPU support. Our implementation is being done using pytorch, torchaudio, OpenNMT libraries(4). All these libraries were installed using conda. Firstly data-set is being downloaded, cleaned and prepossessed to feed directly into OpenNMT. Then using this data we train our model. During training we also validate the model using validation set. Finally we generate transcripts of test-set and calculate CER/BLEU score. There are various configurations of training which we do and results for all of them is explained in next section.

5 Experiments and Discussion

Firstly we have trained the model using Microsoft training corpus (in Gujarati), which is of 40 hours. The model used is LSTM Encoder-Decoder with attention. We have 5 hours of test and 2 hours of validation data for the same corpus. The CER for the corpus is 85.1584(%). We have repeated the experiments with two different data set (OpenSLR Male and OpenSLR Female). The size of these data sets are 5 hours each. Since the data set size is very small the CER for the system is very high.

These results were unpromising. Then we have performed an experiment considering training on parent data set (Microsoft Gujarati Corpus) followed by fine tuning of the trained model using OpenSLR Male and OpenSLR Female data set independently. The CER is 85.3722(%) and 87.722(%) for OpenSLR Male and Female corpus respectively. Then we have performed transfer learning in which we freeze the weights of encoder and do fine tuning of trained model on OpenSLR Male and OpenSLR Female corpus independently. There is a slight decrease in CER (85.3722(%) to 79.9101(%) for OpenSLR Male and 87.722(%) to 84.1294(%) for OpenSLR Female).

Finally we use i-vector for speaker adaptation as a technique to further improve the accuracy of the system. We extracted i-vectors using Kaldi toolkit having following pipelines: Data Preparation, Extract MFCC & VAD, UBM & I-Vector Extractor Training, I-Vector Extraction. We combine this i-vector with characteristic representation of encoder using component wise sum or concatenation and feed it to the first timestamp of the decoder. To our surprise CER obtained using this setup were same as previous setup. The reason is components of i-vectors are very small and spans up-to 6 decimal places. The following table summarizes the results:

6 Future work

From our work we identified that alone making use of the i-vector doesn't imparts any extra knowledge to the network as completely different to what we thought would happen, on further investigation of i-vector values, we found that alone these vectors are very small [varies only at 6^{th} significant after decimal]. One the possible future can be seen as to intelligently scale these values via learning a transformation matrix depicted in figure 8 .

Dataset	**Setting**	**Time (hours)**	**BLEU**	**CER**(%)
Microsoft Speech Corpus (Gujarati)	Full training	40	7.28	85.1584
OpenSLR Male (Gujarati)	Full training	5	0.17	-
OpenSLR Female (Gujarati)	Full training	5	0.14	93.8678
OpenSLR Male (Gujarati)	Fine Tuning	5	4.30	85.3722
OpenSLR Female (Gujarati)	Fine Tuning	5	2.77	87.722
OpenSLR Male (Gujarati)	Fine Tuning & Weight Freezing	5	4.37	79.9101
OpenSLR Female (Gujarati)	Fine Tuning & Weight Freezing	5	4.34	84.1294
OpenSLR Male (Gujarati)	Fine Tuning & Weight Freezing & I-Vectors	5	4.37	79.9101
OpenSLR Female (Gujarati)	Fine Tuning & Weight Freezing & I-Vectors	5	4.34	84.1294

Table 1: Results formulated over the various hypothesis

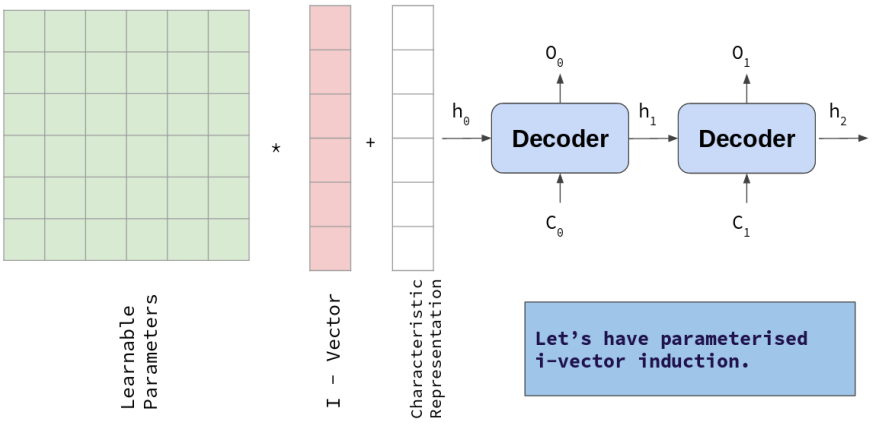

Figure 8: Learning a transformation matrix

References

[1] Shiyu Zhou, Shuang Xu, Bo Xu, Multilingual End-to-End Speech Recognition with A Single Transformer on Low-Resource Languages arxiv.org/abs/1806.05059

[2] Shubham Toshniwal et.al , Multilingual Speech Recognition With A Single End-To-End Model,arxiv.org/abs/1711.01694

[3] Suyoun Kim, Michael L. Seltzer, TOWARDS LANGUAGE-UNIVERSAL END-TO-END SPEECH RECOGNITION, arxiv.org/abs/1711.02207

[4] OpenNMT-py (https://github.com/OpenNMT)

[5] I-vector Extraction (http://jrmeyer.github.io/asr/2017/09/29/challenge.html)

[6] FairSeq (https://github.com/pytorch/fairseq)

[7] Abdelrahman Mohamed, Dmytro Okhonko, Luke Zettlemoyer, Transformers with convolutional context for ASR