CTC-Based End-To-End ASR for the Low Resource Sanskrit Language with Spectrogram Augmentation

Anoop C. S.
Dept. of Electrical Engineering
Indian Institute of Science
Bengaluru, India
anoopcs@iisc.ac.in

A. G. Ramakrishnan
Dept. of Electrical Engineering
Indian Institute of Science
Bengaluru, India
agr@iisc.ac.in

***Abstract*—Sanskrit is one of the Indian languages which fares poorly, with regard to the development of language-based tools. In this work, we build a connectionist temporal classification (CTC) based end-to-end large vocabulary continuous speech recognition system for Sanskrit. To our knowledge, this is the first time an end-to-end framework is being used for automatic speech recognition in Sanskrit. A Sanskrit speech corpus with around 5.5 hours of speech data is used for training a neural network with a CTC objective. 80-dimensional mel-spectrogram together with their delta and delta-delta is used as the input features. Spectrogram augmentation techniques are used to effectively increase the amount of training data. The trained CTC acoustic model is assessed in terms of character error rate (CER) on greedy decoding. Weighted finite-state transducer (WFST) decoding is used to obtain the word level transcriptions from the character level probability distributions obtained at the output of the CTC network. The decoder WFST, which maps the CTC output characters to the words in the lexicon, is constructed by composing 3 individual finite-state transducers (FST), namely token, lexicon and grammar. Trigram models trained from a text corpus of 262338 sentences are used for language modeling in grammar FST. The system achieves a word error rate (WER) of 7.64% and a sentence error rate (SER) of 32.44% on the Sanskrit test set of 558 utterances with spectrogram augmentation and WFST decoding. Spectrogram augmentation provides an absolute improvement of 13.86% in WER.**

***Index Terms*—connectionist temporal classification, CTC, ASR, Sanskrit, spectrogram augmentation, WFST decoding**

I. Introduction

Automatic speech recognition (ASR) technologies require a large amount of annotated training data to work reasonably well. Recent approaches to speech recognition employ deep learning techniques and this has made the data scarcity problem more severe, as the deep learning methods are quite data hungry. It is estimated that only 1% of the languages of the world have the minimum amount of data needed to train an ASR [1]. Due to this, speech recognition researchers have been focusing mainly on high resource languages like English and Mandarin till a few years back.

In recent years there has been a stronger focus on developing ASR for low resource languages [2]–[5]. Different approaches like multilingual pre-training [6]–[11] and data augmentation [12]–[15] have been applied to improve the performance of ASR for low resource languages. However such advancements in technology have not been propagated to many of the low resource Indian languages. The eighth schedule of the constitution of India, lists 22 official languages, most of which are low resource in nature. Sanskrit is one among them and is considered as the second oldest language next to Tamil. It is believed to be the mother of many languages in the Indo-European family in the sense that their genesis is tracked to Sanskrit. A huge body of literature in various areas spanning from mathematics, astronomy, science, linguistics, mythology, history and mysticisms are available in this language. Sanskrit assumes enormous importance given the aforementioned considerations although it is not used widely for active communication. There are only very few attempts on building technical tools for Sanskrit [16], [17]. This motivates us to investigate on the application of some of the state-of-the-art technologies and techniques towards building a Sanskrit ASR. We believe that these efforts will aid in enhancing accessibility of the language and thus its contents to a larger population.

Conventional ASR systems consists of three sub-modules namely acoustic models, pronunciation models and language models. The existence of these modules give rise to the following limitations [18].

1) Each of these modules are trained independently with different objectives, which may result in the sub-optimal performance of the resulting system.
2) Preparation of acoustic models requires phonetic alignments, which needs to be obtained from some other system. For example, to train a hybrid ASR architecture with deep neural network (DNN) and hidden Markov models (HMM) [19], we need tied-triphone state alignments which are prepared using a separate ASR architecture consisting of Gaussian mixture models (GMM) and HMMs.
3) Preparation of pronunciation models requires lin-

guistic knowledge and are generally curated by expert linguists. This being a manual process is subjected to human errors.

The above limitations have prompted the ASR community to move away from the conventional ASR systems to end-to-end trained systems which map the input acoustic features directly to graphemes or word sequences. Two most popular approaches to end-to-end speech recognition are: i) attention-based encoder-decoder [20] and ii) connectionist temporal classification (CTC) [21]. Attention-based methods have the advantage that they do not require any conditional independence assumptions. However, the disadvantage is that they do not guarantee the monotonic alignments required in speech recognition problems. On the other hand, CTC allows only monotonic alignments but suffers from conditional independence assumptions, i.e., every output is conditionally independent of other outputs.

To utilise the monotonic alignments offered with CTC, we choose the CTC-based scheme for building a large vocabulary continuous speech recognition (LVCSR) system for Sanskrit. We propose an architecture based on residual convolutional neural networks (CNN) [22] and bidirectional gated recurrent units (GRU). As the multilingual pre-training experiments require paired audio and transcriptions from other languages, we restrict ourselves to the experiments with data augmentation alone. Specifically, we use a feature augmentation technique called SpecAugment [15] to effectively increase the amount of data available for training. We use greedy decoding to assess the performance of the learnt CTC acoustic model. Weighted finite-state transducers (WFST) are used for word level decoding.

The remaining part of the paper is organized as follows: Section II gives an overview of the theoretical background of the CTC-based ASR. Section III describes the Sanskrit data corpus used in this work. Section IV describes the training of the CTC acoustic model followed by the decoding approaches in section V. Performance of the system in terms of the CER and WER are stated in section VI. Conclusions of our experiments are summarised in section VII.

II. An overview of the approach

In speech recognition problem, we have to generate a sequence of output symbols (letters) given a sequence of input features. However, when to output the symbol is unknown, as the alignment of letters in the transcription to the input audio is not available. CTC overcomes this issue by computing the probability of an output sequence given the input sequence as the sum of probabilities of all possible alignments between the two.

Consider a sequence of input acoustic features $\mathbf{X} = \{x_1, x_2, \ldots, x_T\}$ and the corresponding output labels $\mathbf{Y} = \{y_1, y_2, \ldots, y_U\}, U \leq T$. Each of the output labels $y_i \in \mathcal{U}, i = 1, 2, \ldots, U$, where $\mathcal{U}$ is an alphabet of size L. The alignment between $\mathbf{X}$ and $\mathbf{Y}$ is unknown. However, the alignments are monotonic in the speech recognition problem, i.e., the alignment between input and output happens to be in the same order. CTC introduces a special blank character "$-$" to the alphabet $\mathcal{U}$. Let the modified alphabet be $\mathcal{U}' = \mathcal{U} \cup \{-\}$. For a given $x_i, i = 1, \ldots, T$, the CTC network gives the distribution over over all the possible output labels in $\mathcal{U}'$. This distribution is used to evaluate the probability of any given output sequence $\mathbf{Y}$.

Consider a path $\mathbf{Z} = \{z_1, z_2, \ldots, z_T\} \in {\mathcal{U}'}^T$ in the CTC output distribution over time. Define a many-to-one function $\mathcal{F}$, which maps any given path $\mathbf{Z}$ to a label sequence $\mathbf{Y}$, by collapsing repeats and then removing all the blank symbols in the path. For example, the function $\mathcal{F}$ maps the sequence { म, म, −, म, म, −, <SPACE>, −, न, न, −, ा, −, म, − } to { म, म, <SPACE>, न, ा, म }. Symbols separated by the blank symbol are not merged during the process of collapsing repeats. Thus the blank symbol helps to handle the repetition of letters in the orthography.

The probability of a path $\mathbf{Z}$ can be computed as

$$P(\mathbf{Z}|\mathbf{X}) = \prod_{t=1}^{T} p(z_t|z_1, z_2, \ldots, z_{t-1}, \mathbf{X}) \quad (1)$$

$$\approx \prod_{t=1}^{T} p(z_t|\mathbf{X}) \quad (2)$$

Each term inside the product in (2) can be obtained from the output probability distributions computed by the CTC. Now the probability of the output sequence $\mathbf{Y}$ can be computed by summing over all the possible paths $\mathbf{Z}$ that gets mapped to $\mathbf{Y}$ by the function $\mathcal{F}$.

$$P(\mathbf{Y}|\mathbf{X}) = \sum_{\{\mathbf{Z}|\mathcal{F}(\mathbf{Z})=\mathbf{Y}\}} P(\mathbf{Z}|\mathbf{X}) \quad (3)$$

The summation over all the paths is achieved through dynamic programming. During the training, blank symbols are inserted between each of the labels in the output and also at the beginning and end of the output sequence. Training proceeds with the objective to maximise the probability of the correct label sequence. During the decoding, we infer the most likely label sequence $\mathbf{Y}$ given $\mathbf{X}$.

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} P(\mathbf{Y}|\mathbf{X}) \quad (4)$$

The approximation in (2) means that the output at any time instant is conditionally independent of the other outputs given the input. Though this assumption is rather strong, considering the benefits of the monotonic alignment property offered by CTC, we chose to build our architecture based on CTC.

III. Dataset preparation

We have collected around 6.5 hours of Sanskrit speech data consisting of 3395 utterances from 23 speakers. All the data were collected online, mainly from 4 sources: (a) news recordings from All India Radio (AIR) website [23],

(b) video lectures from Indian Heritage Group (IHG), C-DAC, (c) video lectures from Central Sanskrit University [24], and (d) short stories read by Samskrita Bharati volunteers [25]. The data is mainly read speech and lectures and are encoded in mp3 format. The collected data was converted to single channel raw wav file format with 16 kHz sampling frequency and 16 bits per sample. There are 23 speakers: 17 male and 6 female. Each audio file contains recordings from a single speaker. The corpus contains around 12250 words.

The data was randomly divided into 2 sets - train and test, with approximately 5.5 hours in the train set and 1 hour in the test set, the details of which are shown in Table I. The word-level transcriptions of these utterances are saved in Unicode text format. There were 1712 new words in the test set not belonging to the training set. Due to the limited size of the dataset, we have speaker overlap between the train and test sets.

Table I
DETAILS OF THE SANSKRIT DATASET

	Train	**Test**
Files	2837	558
Words	10541	2911
Speakers	23	22
Male	17	16
Female	6	6
Duration	5:22:12	1:05:56
Male	4:09:19	0:50:32
Female	1:12:52	0:15:23

We have randomly selected 6% of the training data for the validation set. Thus, 2667 utterances are used for training and 170 utterances are used for validation.

The text corpus for building the language models makes use of the *wiki* Sanskrit data dump [26]. We have extracted text data from several Sanskrit websites as well. The extracted text is cleaned to remove unwanted characters and pre-processed to restrict the graphemes to the Devanagari Unicode symbols. There are a total of 262338 sentences in the text corpus with around 436800 unique words. We use 3-gram language models.

IV. ACOUSTIC MODEL TRAINING

A. Feature extraction

We use mel-spectrogram as the input features. Input *wav* files have a sampling rate of 16000 Hz. Window length of 25 ms and hop length of 10 ms are used for framing. Hanning window is applied to each of the frames and 1024 point short-time Fourier transform (STFT) is computed. Power spectrum is computed using the magnitude STFT and fed to a mel- filterbank consisting of 80 filters. Delta features are computed from the filterbank output using a window of 5 frames. Delta-delta features were computed from delta features in a similar manner. Filterbank features, delta and delta-delta features are stacked as 3 channels of an image and used as the input to the neural network.

B. Feature augmentation

Data augmentation schemes help us to increase the effective amount of data available for training the neural networks. We use data augmentation at the feature-level using SpecAugment [15]. SpecAugment has been shown to achieve state-of-the-art performance in LibriSpeech 960h and Swichboard datasets. We apply frequency masking and time masking in the mel-spectrogram domain with parameters 10 and 70 respectively, before the computation of delta and delta-delta features. In frequency masking, f consecutive mel frequency channels $[f_0, f_0+f)$ are masked. First f is chosen from a uniform distribution from 0 to the frequency mask parameter F, and then f_0 is chosen from $[0, \nu-f)$, where ν is the number of mel frequency channels. In time masking, t consecutive time steps $[t_0, t_0 + t)$ are masked, where t is chosen from a uniform distribution from 0 to the time mask parameter T, and then t_0 is chosen from $[0, \tau-t)$, where τ is the number of time steps. Figure 1 shows an example of the frequency and time masking scheme in SpecAugment.

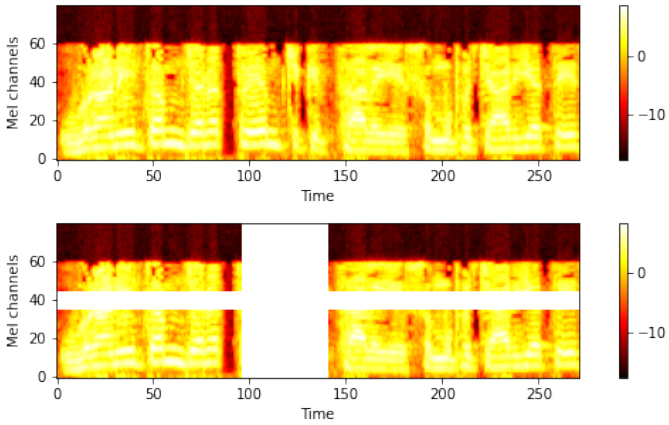

Figure 1. Time and frequency masking using SpecAugment. Top image is the original spectrogram and the bottom image is the augmented spectrogram.

C. Network architecture

For acoustic model training we use an architecture employing CNNs and bidirectional GRUs (BiGRU) as in [27], [28]. The details of the architecture are given in Figure 2, for a mel-spectrogram input with 788 timesteps. The architecture uses 3 residual CNN blocks and 5 Bi-GRU blocks. Each convolutional layer, including those in the residual CNN, uses kernels of size 3x3, stride of 1 and padding of 1, except the first convolutional layer which uses a stride of 2. Each of the BiGRUs has a hidden dimension of 512. Residual CNNs and BiGRUs are preceded by layer normalisation and Gaussian error linear units (GeLU) activation function. A dropout of 0.1 is used in each stage of the network. Logarithm of the

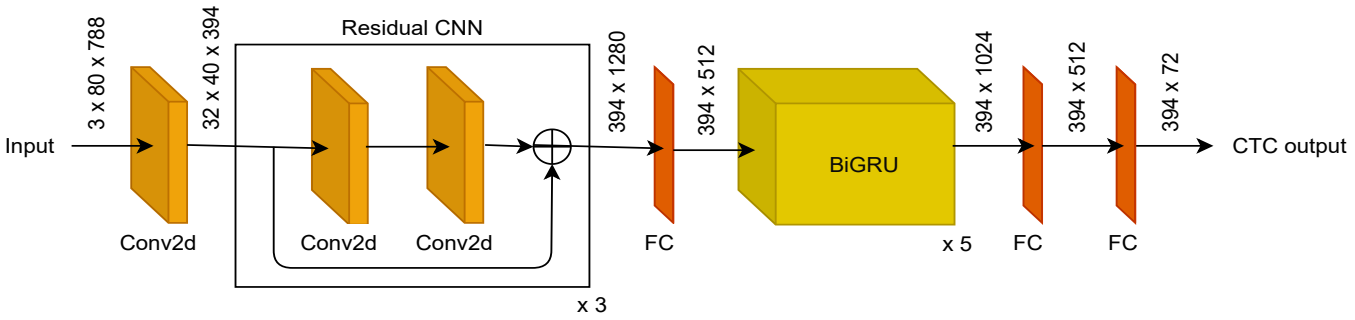

Figure 2. Block diagram of our architecture for learning the CTC acoustic model illustrating an input utterance with timesteps $T = 788$. 80-dimensional mel-spectrogram, their delta and delta-delta are fed as 3 channels of an image. x 3 in the figure indicates that the blocks are repeated 3 times. Conv2d represents the 2D convolutional layers and FC represents the fully connected layers.

softmax function is applied to the CTC output to convert it into log-probabilities. The alphabet $\mathcal{U}$ consists of 13 independent vowel symbols, 12 dependent vowel symbols, 34 consonants, 4 consonants with nukta (ज़, ड़, ढ़, फ़), chandrabindu(ँ), anusvara(ं), visarga(ः), avagraha(ऽ), virama(्), ohm(ॐ), <SPACE> and <UNK>. <UNK> stands for any unknown grapheme symbol occuring in the input labels apart from the above listed ones. The alphabet size, $L = 71$. Including the blank label $(-)$, we have 72 output classes.

D. Training of the CTC acoustic model

AdamW optimiser [29] is used for training. The network is trained with CTC objective [21] for a maximum of 200 epochs. Early stopping, with a patience value of 30 is applied. A batch-size of 10 is employed during training. We adopt the one cycle learning rate policy [30] with a maximum learning rate of 0.001. The network is implemented in PyTorch.

V. Decoding

We use greedy decoding [21], [31] to assess the classification performance of the trained CTC network. Greedy decoding does not use any linguistic information. To assess the word level performance of the ASR, WFST decoding [32]–[34] is used.

A. Greedy decoding

In greedy decoding, the best path $\mathbf{Z}^* \in \mathcal{U}'^T$ is computed in a greedy fashion by picking up the most probable letter from the CTC output distribution at each time step.

$$z_t^* = \arg\max_{l \in \mathcal{U}'} P(z_t = l|\mathbf{X}), t = 1, 2, \ldots, T. \quad (5)$$

To obtain the decoded output, the repeats are collapsed (i.e., letters appearing successively, but not separated by the blank symbols are merged together) and thereafter blank symbols are removed.

$$\mathbf{Y}^* = \mathcal{F}(\mathbf{Z}^*) \quad (6)$$

Output is post-processed to remove some of the invalid combinations of graphemes. This post-processing step includes:

1) removal of multiple occurrences of viramas (्),
2) removal of viramas before and after the dependent/independent vowel symbols, and
3) replacing multiple occurances of vowels with the last vowel appearing in the sequence.

The output of the post-processing step is used to compute the character and word error rates.

B. WFST Decoding

A weighted finite-state transducer (WFST) [32] is a finite automata in which each state transition is associated with an input/output label pair and a weight. A path in the WFST accepts a sequence of input symbols and emits the corresponding sequence of output symbols and the weights associated with that path. WFST constructed for CTC decoding should accept a sequence of letters $z_t \in \mathcal{U}'$ as input and should output a sequence of words from the lexicon. Such a WFST is constructed by fusing 3 individual WFSTs as in [34].

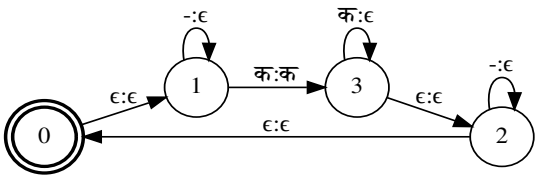

Figure 3. An example of a token FST. Input labels appear before ':' and output labels appear after ':'. Node 0 is the start node and double circled node is the end node. '-' denotes the blank label and 'ϵ' indicates that no inputs are accepted or no outputs are emitted. This FST maps different CTC paths like "−−कक्क−", "कक्कक्कक", "−−कक्कक", etc. to a single unit "क" (collapse-repeats and blank removal operations).

1) Token FST (T): Maps the sequence of frame level CTC labels $l \in \mathcal{U}'$ to a single character in $\mathcal{U}$. This FST effectively performs the collapse-repeat and blank label removal on the input sequence. An example of token FST is shown in Figure 3.

2) Lexicon FST (L): Maps the sequence of characters in $\mathcal{U}$ to words. Each entry in the lexicon is the representation of the word in terms of the constituent characters. An example of the lexicon FST is shown in Figure 4.
3) Grammar FST (G): Encodes the permissible word sequences in Sanskrit. They are generated using the word level 3-gram language models learnt from the text corpus.

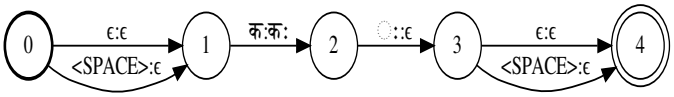

Figure 4. An example of a lexicon FST. Words are allowed to have an optional <SPACE> character at the beginning and at the end.

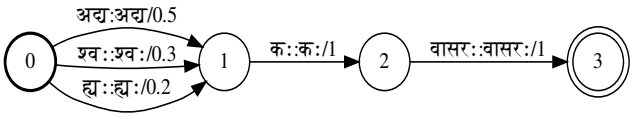

Figure 5. An example of a grammar FST. The weights associated with the arcs denote the probability of emitting the word given the previous word.

These three WFSTs are composed into a search graph which is used to find the most probable word sequence.

$$S = T \; o \; min(det(L \; o \; G)) \tag{7}$$

where o, min and det are the FST operations; composition, minimisation and determinization and S is the search graph.

During the WFST decoding, we normalise the posterior probability of the classes by their priors. The priors for the classes are computed using the label sequences with the blank symbol inserted between each successive character labels and also at the beginning and end of the sentence. For example the label sequence "एषः<SPACE>कः " will be converted to the sequence "–ए–ष–ः–<SPACE>–क–ः– " before the computation of prior. This technique has been shown to improve the performance of WFST decoding in [34].

VI. Results

Performance of the CTC acoustic model is measured using character error rate (CER) on greedy decoding. Recognition performance of the whole ASR sytem is measured using word error rate (WER) and sentence error rate (SER) on WFST decoding. Both CER and WER were computed using Levenshtein distance between the reference sequence and the predicted sequence.

$$WER = \frac{S + D + I}{N} \tag{8}$$

where S, D and I are the number of substitutions, deletions and insertions at the word level and N is the total number of words in the reference. CER is computed in a similar manner, but with errors computed at the character level. SER is calculated as the ratio of the number of correctly decoded sentences to the total number of sentences in the test set.

Table II
Results of greedy decoding on Sanskrit test set

Architecture	CER	WER
Proposed network	23.16	78.74
Proposed network + SpecAugment	20.05	72.04

The results of greedy decoding on the test set are listed in Table II. There are absolute improvements of $\approx 3.1\%$ in CER and $\approx 6.7\%$ in WER when SpecAugment is applied.

Table III
Results of WFST decoding on Sanskrit test set

Architecture	WER	SER
Proposed network	21.50	57.17
Proposed network + SpecAugment	7.64	32.44
Kaldi recipe	5.83	24.01

The results of WFST decoding on the test set are given in Table III. SpecAugment achieves an absolute improvement of 13.86% in WER and 24.73% in SER over the system without spectral augmentation, when WFST decoding is employed. To assess the performance of the system, we also compare the results with a baseline hybrid HMM/DNN system trained with feature space maximum likelihood linear regression (fMLLR) features in Kaldi [33], using the same training-validation split. This is a multilayer perceptron system with 7 hidden layers and 2048 nodes in each layer. The baseline results are listed as the third row in Table III. The kaldi system still fares better, as our end to end system is limited by the amount of training data available. However, SpecAugment reduces the large gap of 15.67% between end to end CTC architecture and hybrid DNN/HMM system to just 1.81%. The limited size of the dataset may also be a factor in attaining a WER closer to the hybrid DNN-HMM system.

VII. Conclusions

In this work, we have built a LVCSR system for Sanskrit using CTC- based end-to-end framework and spectrogram augmentation. The system achieves a WER of 7.64% and a SER of 32.44% on the Sanskrit test set with 558 utterances. The trained CTC acoustic model achieves a CER of 20.05% using greedy decoding. Since a large amount of paired data (audio and transcriptions) were not available for Sanskrit, we experimented with SpecAugment to increase the effective amount of training data. This gives an absolute improvement of 3.11% in CER on greedy decoding and an absolute improvement of 13.86% in WER on WFST decoding. We hope the results can be improved further if more training data is available.

References

[1] G. Adda, S. Stüker, M. Adda-Decker, O. Ambouroue, L. Besacier, D. Blachon, H. Bonneau-Maynard, P. Godard, F. Hamlaoui, D. Idiatov, G.-N. Kouarata, L. Lamel, E.-M. Makasso, A. Rialland, M. Van De Velde, F. Yvon, and S. Zerbian, "Breaking the unwritten language barrier: The BULB project," in *Procedia Computer Science*, 2016, vol. 81, pp. 8–14.

[2] J. Billa, "ISI ASR system for the low resource speech recognition challenge for Indian languages," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, vol. September, pp. 3207–3211.

[3] B. Pulugundla, M.K. Baskar, S. Kesiraju, E. Egorova, M. Karafiát, L. Burget, and J. Cernocký, "BUT system for low resource Indian language ASR," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, vol. September, pp. 3182–3186.

[4] M. Chellapriyadharshini, A. Toffy, K.M. Srinivasa Raghavan, and V. Ramasubramanian, "Semi-supervised and active-learning scenarios: Efficient acoustic model refinement for a low resource Indian language," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, vol. September, pp. 1041–1045.

[5] A. Madhavaraj and A.G. Ramakrishnan, "Design and development of a large vocabulary, continuous speech recognition system for Tamil," in *14th IEEE India Council International Conference (INDICON)*, 2017, pp. 1–5.

[6] S. Toshniwal, T.N. Sainath, R.J. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, vol. April, pp. 4904–4908.

[7] J. Cui et al., "Multilingual representations for low resource speech recognition and keyword search," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 259–266.

[8] E. Chuangsuwanich, *Multilingual techniques for low resource automatic speech recognition*, Ph.D. thesis, Massachusetts Institute of Technology, 2016.

[9] P. Ghahremani, V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Investigation of transfer learning for ASR using LF-MMI trained neural networks," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 279–286.

[10] J. Cho, M. K. Baskar, R. Li, M. Wiesner, S. H. Mallidi, N. Yalta, M. Karafiát, S. Watanabe, and T. Hori, "Multilingual sequence-to-sequence speech recognition: Architecture, transfer learning, and language modeling," in *IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 521–527.

[11] C.-X. Qin, D. Qu, and L.-H. Zhang, "Towards end-to-end speech recognition with transfer learning," *Eurasip Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, 2018.

[12] N. Kanda, R. Takeda, and Y. Obuchi, "Elastic spectral distortion for low resource speech recognition with deep neural networks," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013, pp. 309–314.

[13] A. Ragni, K.M. Knill, S.P. Rath, and M.J.F. Gales, "Data augmentation for low resource languages," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2014, pp. 810–814.

[14] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2015, vol. January, pp. 3586–3589.

[15] D.S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E.D. Cubuk, and Q.V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2019, vol. September, pp. 2613–2617.

[16] C. S. Anoop and A. G. Ramakrishnan, "Automatic speech recognition for Sanskrit," in *2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 2019, vol. 1, pp. 1146–1151.

[17] D. Adiga, R. Kumar, Krishna A., P. Jyothi, G. Ramakrishnan, and P. Goyal, "Automatic speech recognition in sanskrit: A new speech corpus and modelling insights," in *59th Annual Meeting of the Association for Computational Linguistics (ACL Findings)*, 2021.

[18] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[19] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[20] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949.

[21] A. Graves, S. Fernández, and F. Gomez, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *In Proceedings of the International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] All India Radio, "RNU/NSD audio archive search," http://newsonair.com/RNU-NSD-Audio-Archive-Search.aspx.

[24] Central Sanskrit University, "Sanskrit language teaching," http://www.sanskrit.nic.in/sanskrit_language_teaching.php.

[25] Samskrita Bharati, "Balamodini children's stories in sanskrit (1994 to 1999)," https://archive.org/details/bAlamodinI-01.

[26] Wiki Sanskrit data dump, "Wikimedia database dumps," https://dumps.wikimedia.org/sawiki/20200901/sawiki-20200901-pages-articles.xml.bz2.

[27] D. Amodei et al., "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *33rd International Conference on Machine Learning, (ICML)*, 2016, vol. 1, pp. 312–321.

[28] M. Nguyen, "Building an end-to-end speech recognition model in pytorch," https://www.assemblyai.com/blog/end-to-end-speech-recognition-pytorch.

[29] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv e-prints*, 2017.

[30] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," *arXiv e-prints*, Mar. 2018.

[31] T. Zenkel, R. Sanabria, F. Metze, J. Niehues, M. Sperber, S. Stüker, and A. Waibel, "Comparison of decoding strategies for CTC acoustic models," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017, vol. August, pp. 513–517.

[32] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer, Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[33] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Dec. 2011, IEEE Signal Processing Society.

[34] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174.