INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Training Hybrid Models on Noisy Transliterated Transcripts for Code-Switched Speech Recognition

Matthew Wiesner[1,2], *Mousmita Sarma*[3], *Ashish Arora*[1], *Desh Raj*[1], *Dongji Gao*[1], *Ruizhe Huang*[1], *Supreet Preet*[3], *Moris Johnson*[3], *Zikra Iqbal*[3], *Nagendra Goel*[3], *Jan Trmal* [1] *Paola García* [1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing, The Johns Hopkins University, USA
[2]Human Language Technology Center of Excellence, The Johns Hopkins University, USA
[3]Go-Vivace Inc., USA

{wiesner,aarora8,draj2,dgao5,ruizhe,yenda,lgarci27,khudanpur}@jhu.edu
{mousmita, supreet, moris, zikra.iqbal, nagendra.goel}@govivace.com

Abstract

In this paper, we describe the JHU-GoVivace submission for subtask 2 (code-switching task) of the Multilingual and Code-switching ASR challenges for low resource Indian languages. We built a hybrid HMM-DNN system with several improvements over the provided baseline in terms of lexical, language, and acoustic modeling. For lexical modeling, we investigate using unified pronunciations and phonesets derived from the baseline lexicon and publicly available Wikipron lexicons in Bengali and Hindi to expand the pronunciation lexicons. We explore several neural network architectures, along with supervised pretraining and multilingual training for acoustic modeling. We also describe how we used large externally crawled web text for language modeling. Since the challenge data contain artefacts such as misalignments, various data cleanup methods are explored, including acoustic-driven pronunciation learning to help discover Indian-accented pronunciations for English words as well as transcribed punctuation. As a result of these efforts, our best systems achieve transliterated WERs of 19.5% and 23.2% on the non-duplicated development sets for Hindi-English and Bengali-English, respectively.

Index Terms: speech recognition, code-switching, transliteration, pronunciation modeling.

1. Introduction

This paper describes the JHU-GoVivace submission for the code-switching portion of the MUlti-lingual and Code-Switching (MUCS) 2021 Challenge. Code-switching has been used to describe a variety of linguistic phenomena. The most relevant definition for this challenge, and the aspect we focus on is the mixing of two languages within a single utterance. This phenomenon has proven challenging for automatic speech recognition (ASR) systems to properly handle [1]. The code-switching part of the MUCS 2021 challenge, focuses specifically on the problem of ASR on two indic-languages, Hindi and Bengali. The difficulty of this task arises from the limited matched domain transcribed speech as well as the lack of phonetic pronunciations for Hindi/Bengali words written in the Latin script, or English words written in native Indic scripts. Compounding these problems is the fact that most data available for training code-switching models is "found" data, whose transcripts are often noisy.

For our submission, we focused on three approaches to address these challenges: (i) we explore 2 techniques aimed at mitigating the detrimental effects of noisy transcripts on ASR training; (ii) we attempt to learn non-standard pronunciations by leveraging transliteration pairs, and acoustically driven pronunciation modeling, making use of transliteration pairs for improved language modeling; (iii) We investigated various acoustic model architectures and training strategies, including supervised pretraining and multilingual training.

The remainder of this paper details the approaches taken to solve each of these problems. We also describe our approach for further curating the code-switched ASR datasets released as a part of this challenge [2].

2. Codeswitching Data and Task Description

The code-switching data from the *Multilingual and code-switching ASR challenges for low resource Indian languages* [2] consists of about 90h of Hindi language, and 46h of Bengali language lectures on technical subjects, which by nature include a large number of English words. The transcripts for these lectures are written using both Devanagari/Bengali and English scripts. For the purposes of this paper, we refer to the alternate use of the scripts to represent the same underlying word as code-switching. Three data partitions were released for each language: a training set, *train*, a development, *test*, and a test set called the *blind-test*. Challenge submissions were evaluated on the basis of the word-error-rate (WER) of hypothesised *blind-test* transcripts with respect to a reference transcript as well as a transliterated reference transcript. Using external data was permitted in training as long as it did not overlap with *blind-test*.

As mentioned in [2], the code-switching data consists of found data. A well-curated and useful speech corpus relies on having training, development, and test sets consisting of disjoint utterances drawn at random from speech instances representative of a particular domain of interest. Unfortunately, found data is often biased which complicates the creation of such well curated sets. In fact, we found significant overlap of speaker and lecture, between the training and test sets, as well as significant transcription and segmentation errors, which prevented fair internal evaluation of competing systems on *test* using the provided data. We therefore took care to create a fair tuning set and evaluation method in order to mitigate these problems.

First, we identified repeated lectures by examining the fraction of utterances in each recording in *test* that were also present in *train*. Recordings for which $> 50\%$ of the segments were repeated were considered duplicate lectures. We therefore, divided the *test* partition into a partition of duplicate lectures

http://dx.doi.org/10.21437/Interspeech.2021-2127

also present in *train* and non-duplicate lectures that were absent from *train*. We refer to these two test sets as **Dup** and **NoDup** respectively. **Whole**, refers to the union of **Dup** and **NoDup**.

We also found significant speaker overlap. Each lecture ends with a sign-off phrase in which the lecturer announces his name, which we used to estimate the identity of the lecturer. We also ran an x-vector based speaker identification system to independently estimate the number of unique speakers in each corpus. The x-vector system and sign-off search showed close to 100% agreement, from which we concluded that the entire Hindi and Bengali data sets likely consist of only about 7 and 10 speakers respectively. Table 1 shows the updated number of estimated unique speakers in each data set. Surprisingly, all speakers in the blind-test set were also seen in training, meaning this is a closed-speaker ASR task. For this reason we did not attempt to remove speaker overlap from our fair tuning set.

Table 1: *Estimated number of unique speakers in each data set.*

	Train	Test	Blind	Total
# Spks Hindi	7	4	5	7
# Spks Bengali	10	7	8	10

To remedy the segmentation errors, we first tried to realign/resegment the *test* data, but this proved challenging due to the numerous transcription errors. Therefore, to circumvent errors in the manual utterance time-marks in the *test* transcripts we scored the automatic transcripts of entire recordings against the concatenation of all the utterances of that recording rather than scoring at an utterance level. We address resegmentation and cleanup of *train* in section 5.

Finally, we replicated the transliterated WER metric used on *blind-test* by mapping all instances of English words written in Indic scripts in *test* to their Latinate forms in both the reference transcript and the ASR output.

3. Lexical Modeling

The code-switching data is characterized by the presence of words with multiple valid spellings. Generally these are English words with a single spelling in the Latin script, and 1 or more spellings in an Indic script. Pronunciation lexicons should therefore share pronunciations for English and Hindi/Bengali forms of the same word. To this end, we use a unified phoneme set based on the International Phonetic Alphabet. Existing pronunciation lexicons scraped from Wiktionary entries are available for both Hindi and Bengali [3].[1]

We use these lexicons to provide pronunciations for all words present in both the training data and the scraped lexicon. Just as the the acoustics of a phoneme may vary according to its phonemic context, the acoustics of corresponding to a phoneme used to describe the pronunciation of an English may be different from when the same phoneme is used to descibe the pronunciation of a Hindi/Bengali word. Therefore, we permit creation of language-specific (senone) models in a data-driven manner, by adding a "language tag" to IPA pronunciations of Hindi/Bengali words, and make them available during decision tree learning. For all English words in training, we simply map the phonemes in the provided pronunciations to XSAMPA phonemes. For words that consist of a mix of English and non-English script, we split the words on the change in script, and assign an English portion the pronunciation using the baseline lexicon provided as part of the baseline challenge systems. Hindi/Bengali word and word-fragment pronunciations are obtained either directly from the Wikipron lexicons or from a G2P trained on these lexicons. All G2P systems are trained using `Phonetisaurus`[4].

To further encourage pronunciation sharing between Latin and Indic scripts, we merge the pronunciations of all transliteration pairs and treat them as a single type in all acoustic and language model training.

3.1. Acoustically Driven Pronunciation modeling

In order to capture model pronunciations of English words in Indian accented speech, we decode the training data using a phoneme-level language model, and match these phoneme sequences to a word-level time alignment of the reference transcripts. Phoneme sequences that frequently align to a word are then added as candidate pronunciations for that word. We can then prune these candidate pronunciations as done in [5] to retain only the most likely alternate pronunciations using a greedy pronunciation selection strategy.

3.2. Transliteration Pair Mining

A key challenge in this particular code-switching ASR task is to avoid redundant representations of words and phonemes that occur due to transliteration pairs: words, mostly technical terms in English, are often rendered using the Devanagari or Bengali scripts. We believe transliteration pair mining or linking is an important component to good code-switched language and pronunciation models.

We use the transliteration pairs in two ways. First, all transliterated pairs are remapped to a single word type, and the union of all pronunciations is used in the pronunciation lexicon. This has the effect of augmenting the pronunciation lexicon with Indian accented pronunciations for English words. It also increases the likelihood of discovering other Indian accented pronunciations for English words during the acoustically driven pronunciation discovery. Second, since the transliterated pairs are remapped to a single type, we can train a language model on a more compact vocabulary that does not spread probability mass over redundant types.

To create the transliteration pairs for the Hindi-English system, we manually combed through a list of Hindi words (all Hindi words in test transcriptions and those which occur at least 10 times in training) and produced English transliterations for the true English words, thus obtaining 968 word pairs. For Bengali, we used a semi-automated strategy where we we first generated a candidate word-pair list from the most confusable word pairs in a first-pass decoding, and then manually pruned this candidate list to keep only the correct pairs. This process resulted in 236 word pairs.

4. Language Modeling

For language modeling, we used both the training transcripts as well as external corpora crawled from the web. These external sources include the Wikipedia dump[2], OpenSubtitles [6], and data crawled from various websites that intuitively matched the topic and where code-switching was found to be common[3]. We

[1] `https://github.com/kylebgorman/wikipron`

[2] `https://meta.wikimedia.org/wiki/Data\_dumps`

[3] e.g. `https://nitin-gupta.com/computer-pdf-notes-free-download-in-hindi-and-english/`

also extracted text from closed caption lectures[4]. To ensure that we only used external data that did *not* overlap with the content of *blind-test*, we excluded all text sources that exhibited a high degree of n-gram overlap with the *test* and *taining* sets.

Duplicated lines and punctuation were removed using heuristic rules, and the all transliteration pairs were mapped to their Latinate forms (cf. Section 3.2). For additional normalization of Hindi and Bengali text we used Indic NLP library[5]. The obtained corpus contained between 790K and 34M words. For both Hindi and Bengali, we constructed expanded vocabularies(282K and 410K words respectively), by including words that appear more than 3 times in the external data and consist of only characters within Hindi, Bengali or English Unicode ranges. Many English OOVs still remained in the Bengali *test* set, so we added all of CMU Dict to our vocabulary to reduce the OOV rate.

For first-pass decoding, we we used a 4-gram maximum entropy (ME) language model on trained on the training transcripts using the SRILM toolkit [7]. For lattice rescoring we used an interpolated LM. We interpolated the ME models with models trained on our external corpora, where we chose the model types, orders, count thresholds and interpolation ratios based on a held-out set while keeping the final model size under 10M n-grams. For Bengali, we found the external corpora did not improve perplexity so we ended up using the in-domain model only. We used the Kaldi toolkit to train recurrent neural network language models (RNNLMs) consisting of an embedding layer and two layers of LSTM with 256 dimensions each, and performed pruned lattice rescoring [8] on the first-pass decoding outputs.

5. Cleanup and Segmentation Strategies

The training and test data provided in this challenge had many misaligned speech transcripts, as well as incomplete words and inconsistent script usage in the reference transcripts. We cleaned the transcripts, following [9], by first training a GMM using the original training data as the seed acoustic model. We then cut the audio into uniform chunks of 30 seconds with 5 seconds overlap, and decoded them using the seed acoustic model and a biased LM trained only on the training transcripts. We then used the Smith-Waterman algorithm [10] to find the best-aligned subsequence between the reference and hypothesis transcript word sequence. New word-level timestamps were obtained from the alignments and used to resegment the reference transcript based on silence. Any new segments exceeding 50% WER between the reference and hypothesis were deemed to be "low-quality" segments and removed from the training data. We refer to this approach as **Toss**.

We also tried an alternative resegmentation approach to minimize the amount of data we throw away. In this approach the graphs used for alignment are slightly modified to allow for non-speech events to more easily align to the `<unk>` and silence symbols. We refer to this approach as **Resegmented**.

6. Acoustic Modeling

6.1. Architectures

The hybrid HMM-DNN model provided as part of the challenge baseline comprised a time-delay neural network (NN) (TDNN) [11] model, for both the Hindi-English and Bengali-English systems. We investigated 2 different NN architectures during the challenge. All the models are trained to predict senones derived from a decision tree built from the bootstrapped GMM alignments. We used 3500 leaves for the decision tree in our experiments.

1. **WRN**: We use a simplified Wide Residual Network [12] where we removed the batch-normalization and drop-out and added an adaptive average pooling layer followed by a linear output layer in order to map variable-length input chunks of speech to the correct number of outputs. Our networks used a depth of 28, and widening factor of 10 in all WRN architectures.

2. **BLSTM**: Following [13], we built a 6-layer bi-directional long short-term memory (BLSTM) [14] network, with a hidden dimension of 1024. This is followed by a 512-dimensional affine layer, and a final classification layer.

6.2. Training details

All our models were trained using the lattice-free MMI [15] objective with cross-entropy regularization. The output layer was subsampled by a factor of 3 or 4 to reduce the time and space complexity for loss computation. The WRN and BLSTM models were implemented in PyTorch [16], using PyChain [17] to obtain the LF-MMI gradients. We used 64-dim log Mel filterbanks without any i-vector-based adaptation. Speed and volume perturbation was applied to the data before the start of training. To further perturb the training data, we used random sampling of chunks for training, with random-sized chunks between 60 and 220 frames, and injected on-the-fly time-frequency masking and Gaussian noise to the batches. We warmed up the learning rate for the first 15k iterations and then decayed it with a factor of 1e-05. We used the Adam optimizer [18] with a weight decay of 1e-07 in all our experiments. All models were trained to convergence unless stated otherwise.

6.3. Pre-training and multilingual training

In addition to training on the cleaned challenge data, we explored supervised pre-training and multilingual training. Both experiments were conducted using the BLSTM model as the base architecture. We pre-trained the BLSTM on the full 960h Librispeech data [19], and then fine-tuned this model on the cleaned challenge data after replacing the final affine layer to account for different senones. For multilingual training, we trained a system jointly using the cleaned Hindi and Bengali training data (with perturbations as described earlier). The network consists of 6 shared BLSTM layers and a shared affine layer (similar configuration as above), followed by separate affine output "heads" for each language. The final affine layer has dimensionality equal to the number of state-tied senones for the corresponding language. Such multilingual systems have been shown to be useful, particularly in low-resource settings [20, 21]. We trained this multilingual network by alternating between batches of Hindi and Bengali chunks, which were sampled as described in Section 6.2.

7. Results

All reported results use our implementation of the the transliterated WER as described in section 2. We first examined the role of the pronunciation lexicon, and access to transliteration pairs in recognizing code-switched speech for the Hindi-English set. We see in table 2 that using a phonetic lexicon, as described in section 3, marginally improves upon the baseline, hybrid pho-

[4]https://spoken-tutorial.org/cdcontent/

[5]https://anoopkunchukuttan.github.io/indic_nlp_library/

netic/graphemic lexicon used in the baseline systems. By unifying transliteration pairs, and using the resulting alignments to learn pronunciations from audio to be used in decoding, we see further improvement of 0.3-1%.

Table 2: *Effect of pronunciation learning and transliteration mapping on ASR Performance (WER) on Hindi-English. Lexicon (1) trains the WRN on alignments obtained using the discovered pronunciations. Lexicon (2) trains on the original alignments, but decodes using the discovered pronunciations.*

System	Split		
	NoDup	Whole	Dup
Baseline HMM-GMM	27.5	20.3	9.8
Phonetic	26.6	22.2	15.7
+ Transliteration Map	26.4	19.2	8.5
+ Learned Lexicon	25.5	18.0	7.2
WRN Learned Lexicon (1)	21.4	15.0	**5.5**
WRN Learned Lexicon (2)	**21.1**	**14.8**	5.6

We then explore the two cleanup and resegmentation strategies, **Toss** and **Resegment**, described in section 5. While the **Resegment** approach recovered 98% of the training data, as opposed to only about 80% for the **Toss** approach, the WER from training using this technique was unchanged or slightly degraded. Presumably this is because the graphs allowed for the speech to freely align, erroneously, with `<unk>` and silence too frequently.

Table 3: *Effect of data cleanup on system performance (WER). We see in table that the most effective approach was to discard transcripts and their corresponding aligned speech that were identified as likely having errors.*

System	Split		
	NoDup	Whole	Dup
WRN	23.7	17.2	7.8
WRN + Resegmented Cleanup	24.5	18.7	8.8
WRN + Toss Cleanup	**21.5**	**15.2**	**5.9**

7.1. Acoustic model training

We explored training Wide Residual networks and BLSTM networks. We had prior experience using these architectures on the 100h subset of Librispeech and since the genre and data amount of the 100h clean subset of librispeech seemed similar to the code-switched data, we used these models as a starting point for further improvements. Since the Bengali data was smaller and training neural networks is fairly expensive, we did all architecture comparisons on this set. Table 4 shows the results of various WRN and BLSTM models on the Bengali test set. We also show the results for multilingual training and pre-training.

We repeated most of these experiments on the Hindi data as well, and saw similar trends (table 5): The BLSTM models pretrained on Librispeech outperformed WRN models; The effect of pretraining or additional multilingual training was unclear. Both techniques improve over training from scratch.

7.2. Language Modeling and System Combination

Using the RNNLM described earlier, we observed consistent WER improvements of about 0.2-0.4% in our models. For instance, on the Bengali-English NoDup set, the performance of the BLSTM pretrained on Librispeech improved from 24.5% to 24.3% when rescored with the RNNLM. For our final submission, we used MBR decoding to combine the BLSTM multilingual and expanded lexicon systems, as well as a WRN system for Hindi.

Table 4: *Effect of acoustic model training on system performance (WER) for the Bengali-English data. We show the effect of network architecture and training strategy, including pretraining on Librispeech (LS), and multilingual training (MLT). The BLSTM outperforms the WRN system.*

System	Split		
	NoDup	Whole	Dup
WRN	28.7	28.0	16.6
BLSTM	26.9	26.2	14.2
+ Cleanup	25.6	24.8	**11.3**
+ Pretrained on LS	25.3	24.5	11.7
+ Expanded lexicon decoding	**24.5**	**23.8**	11.5
+ MLT (expanded lexicon)	25.3	24.6	13.2

Table 5: *Effect of acoustic model training on system performance for Hindi-English data. We show the effect of network architecture and training strategy, including pretraining on Librispeech (LS), and multilingual training (MLT).*

System	Split		
	NoDup	Whole	Dup
Best WRN	20.9	14.9	5.9
BLSTM (Pretrained on LS)	**20.3**	14.5	**5.9**
+ External LM	20.4	**14.5**	6.0
+ MLT	20.8	14.8	6.0

8. Conclusions

We described our systems for the code-switching challenge. We demonstrated the importance of lexical modeling in the code-switching speech in this challenge, through the use of unified pronunciations across script and acoustically-driven lexicon learning. We presented data cleanup strategies and showed that a simple but effective technique **Toss** technique can improve system performance by approximately 10% relative WER. We observed that BLSTM networks pre-trained with supervised data can provide effective results on both the challenge datasets. Multilingual training was also shown to be competitive with training individual language-specific models. Finally, we demonstrated how the positive impact of external crawled text data when obtained in a domain-sensitive manner. Our final systems, including MBR decoding and RNNLM rescoring obtained WERs of 19.5% and 23.2% on the Hindi and Bengali.

9. Acknowledgements

We would like to thank Namit Sharma and Raj Karbar, GoVivace, for their data collection and annotation efforts. We thank Samik Sadhu for helpful discussions on Bengali corpora.

10. References

[1] S. Shah, S. Sitaram, and R. Mehta, "First workshop on speech processing for code-switching in multilingual communities: Shared task on code-switched spoken language identification," *WSTC-SMC 2020*, p. 24, 2020.

[2] A. Diwan, R. Vaideeswaran, S. Shah, A. Singh, S. Raghavan, S. Khare, V. Unni, S. Vyas, A. Rajpuria, C. Yarra, A. Mittal, P. K. Ghosh, P. Jyothi, K. Bali, V. Seshadri, S. Sitaram, S. Bharadwaj, J. Nanavati, R. Nanavati, K. Sankaranarayanan, T. Seeram, and B. Abraham, "Multilingual and code-switching asr challenges for low resource indian languages," *arXiv preprint arXiv:2104.00235*, 2021.

[3] J. L. Lee, L. F. Ashby, M. E. Garza, Y. Lee-Sikka, S. Miller, A. Wong, A. D. McCarthy, and K. Gorman, "Massively multilingual pronunciation modeling with WikiPron," in *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4223–4228. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.521

[4] J. R. Novak, N. Minematsu, and K. Hirose, "Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework," *Natural Language Engineering*, vol. 22, no. 6, p. 907, 2016.

[5] X. Zhang, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic data-driven lexicon learning based on a greedy pronunciation selection framework," in *Proc. Interspeech 2017*, 2017, pp. 2541–2545. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2017-588

[6] P. Lison and J. Tiedemann, "Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles," 2016.

[7] A. Stolcke, "SRILM-an extensible language modeling toolkit," in *Seventh International Conference on Spoken Language Processing*, 2002.

[8] H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur, "Neural network language modeling with letter-based features and importance sampling," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 6109–6113.

[9] V. Manohar, D. Povey, and S. Khudanpur, "Jhu kaldi system for arabic mgb-3 asr challenge using diarization, audio-transcript alignment and transfer learning," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 346–352.

[10] T. F. Smith, M. S. Waterman *et al.*, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[11] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *INTERSPEECH*, 2015.

[12] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.

[13] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention - w/o data augmentation," in *INTERSPEECH*, 2019.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[15] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free MMI," in *INTERSPEECH*, 2016.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[17] Y. Shao, Y. Wang, D. Povey, and S. Khudanpur, "PyChain: A fully parallelized pytorch implementation of lf-mmi for end-to-end asr," in *INTERSPEECH*, 2020.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[20] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8619–8623, 2013.

[21] S. Thomas, K. Audhkhasi, and B. Kingsbury, "Transliteration based data augmentation for training multilingual asr acoustic models in low resource settings," in *INTERSPEECH*, 2020.