INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

ASR for Low Resource and Multilingual Noisy Code-Mixed Speech

Tushar Verma[†], *Atul Shree*[†], *Ashutosh Modi* [‡]

[†]Convin.AI, [‡]Indian Institute of Technology Kanpur (IIT-K)

{tushar,atul}@convin.ai, ashutoshm@cse.iitk.ac.in

Abstract

Developing reliable Automatic Speech Recognition (ASR) system for Indian Languages has been challenging due to the limited availability of large-scale, high-quality speech datasets. This problem is even more pronounced when dealing with noisy code-mixed settings with different grapheme vocabularies. This paper proposes a novel ASR system for low-resource noisy speech code mixed with Indian languages. Our approach involves fine-tuning pre-trained models using text transliterated to Devanagari and mapping similar-sounding characters into one character group. Experiments show the model's effectiveness for low-resource Indian languages, including noisy, code-mixed, and multilingual settings. The approach outperforms several baseline models and demonstrates the potential for adapting state-of-the-art ASR models to new languages with limited resources. The proposed system has been deployed in production, where call centers use it to transcribe customer calls.
Index Terms: speech recognition, low-resource, multilingual, code-mix, noisy speech.

1. Introduction

Indian languages are spoken by a significant portion of the world's population.[1] However, the limited availability of large-scale, high-quality annotated speech datasets hinders the development of reliable Automatic Speech Recognition (ASR) systems for these languages. This challenge becomes even more difficult in the noisy code-mixed setting, where multiple languages are involved, and recordings are noisy. Code-mixing is prevalent in Indian settings, especially with language pairs like Hindi and English (known as Hinglish). *Code-mixing* refers to the seamless mixing of two or more languages within a single utterance [1]. On the technology side, the introduction of transformers and self-supervision mechanisms in ASR models has led to significant improvements in the performance of ASR systems like Whisper [2], Wav2Vec 2.0 [3], AI4Bharat [4], and Vakyansh [5], compared to previous systems like Kaldi [6]. Self-supervision along with the attention mechanism (in transformers [7]) allows the model to learn from a massive amount of unlabelled data that significantly increases the model's performance even if the model is fine-tuned on a relatively small dataset. With these techniques, models like Wav2Vec 2.0 [3] can achieve state-of-the-art performance on ASR tasks. On standard datasets like LibriSpeech [8], it can achieve results comparable to humans.

This paper proposes novel methods for improving code-mixed multilingual Indic language ASR systems using limited data. Our approach involves transliterating text from multiple code-mixed high and low-resource languages into a common writing system based on the Indian script - Devanagari script.[2] This is followed by a grouping of similar-sounding characters. Next, an ASR model is trained on the augmented data. Finally, a disambiguation pipeline is used to recover the native format. Our proposed approach shows promising results and demonstrates the potential for adapting state-of-the-art ASR models to new languages with limited resources. In a nutshell, we make the following contributions:

- We propose a new ASR system for multilingual low-resource noisy code-switched Indian speech data.
- We perform extensive experiments with the proposed techniques in various settings, including noisy practical settings. We show that the proposed model has competitive performance across various settings, thus pointing towards the generalization capabilities of the approach. The proposed model has been deployed in production and is currently being used by many customer-facing call centers.

2. Related Work

Various researchers have recently developed models and techniques for ASR for low-resource and code-mixed languages [9, 10, 11, 12, 13]. For example, [4] have proposed an approach for developing ASR systems for low-resource languages using data augmentation and transfer learning techniques. The evaluation findings demonstrate substantial gains in ASR accuracy and resilience for various languages with limited resources. [14] proposed a novel grapheme-to-phoneme (G2P) model for code-mixed speech synthesis termed *Mixlingual*, that employs a sequence-to-sequence architecture with an attention mechanism to transition between several language models based on the input dynamically. Assessment of Hindi-English code-mixed voice synthesis tasks demonstrates that the proposed Mixlingual technique outperforms previous state-of-the-art models for synthesized speech's naturalness and understandability. [15] showed a single ASR model that can transcribe speech in more than 16,000 hours of audio across 50 distinct languages. The approach also employed SpecAugment, a novel data augmentation technique, to boost the model's resilience to fluctuations in speech patterns. Results indicate that multilingual training of ASR models can increase recognition performance, especially for languages with limited resources. [5] compared the performance of multilingual speech recognition systems to monolingual models with language identification. The authors combined the decoding information from multilingual models for language identification with monolingual models. The results showed that the multilingual models outperform the monolin-

[1] https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

[2] https://en.wikipedia.org/wiki/Devanagari

10.21437/Interspeech.2023-757

gual models with the language identification module. [16] proposed a two-step approach - Reduce and Reconstruct. In the first step (Reduce) acoustically similar graphemes are reduced, and subsequently, in the second step, a finite state transducer maps reduced graphemes to original forms. Similarly, [17] proposed Grapheme-to-Grapheme (G2G) model that can convert graphemes to their corresponding pronunciation sequence. The proposed approach has better generalization than Grapheme-to-Phoneme (G2P) model. The proposed approach in this paper comes close to [16].

3. Data Processing

We experiment with various ASR datasets (§5). We process each ASR dataset (speech-text data) before feeding it into the model for training. Here we describe the common data processing steps.

3.1. Audio Processing

Audio files are first converted to the wav file format and resampled to 16kHz. If the original audio is stereo, it is converted to a mono channel using `pydub`.[3] For audios longer than 15 seconds, `webrtcvad`[4] is utilized to chunk them into smaller segments based on silences. The resulting audio clips have a duration between 4 and 15 seconds. This segmentation approach has two advantages: it enables the processing of smaller audio segments and ensures that the audio transcript does not get split in the middle of a word.

3.2. Text Cleaning and Pre-processing

Any non-spoken characters, symbols, or words are eliminated during the text cleaning and pre-processing step, including all punctuation marks. Symbols, numbers, and special characters are converted to their spoken versions; for instance, "@" is transcribed as "at the rate" or "at," "%" is transcribed as "percent" or "percentage," "9" as "nine," and "90" as "ninety" or "nine zero" depending on how they are spoken in the audio. This conversion is performed manually to guarantee that no contextual information is lost.

3.3. Text Transliteration and Reduction

To have a single ASR model capable of handling all Indic languages and their code-mixed versions, the character set needs to be minimized to reduce confusion while preserving the rich phoneme structure of the language. In order to accomplish this goal, we leverage the fact that in Indic languages, the pronunciation of a word usually corresponds to its spelling. Consequently, we develop a single representation for phonemes that sound similar and then transliterate Indic words character by character into this representation, which we refer to as the *Common Indic Representation*. English dictionary words are also converted to Devanagari script, where "write" would become "राइट." This approach addresses the problem of irregularities in English spelling and pronunciation, including silent letters in words like "knight", "tsunami," and "debt," as well as the variations in the spelling of words with similar letter format, such as "south" and "soup," "crumb," and "crumble." Subsequently, the Devanagari representation of English words is transliterated into the Common Indic Representation. The entire process is done in three steps.

1. *Transliteration for Indic Words.* Initially, a representative letter is designated for all similar-sounding vowels or consonants across all Indic languages. This is feasible since most major Indic languages have a one-to-one correspondence between their spoken and written forms. The representative assignment is done separately for half chars and full chars. For instance:
   - Ka group has letters क, ક, ক, ਕ, ಕ, ക, ka, and ক is marked as their representative.
   - Aa group of full chars has letters आ, આ, আ, ਆ, ಆ, ആ, ଆ, and আ is marked as their representative.
   - Aa group of half chars has letters ા, ા, া, ਾ,ಾ,ാ, ା, ા, and া is marked as their representative.

   The set of representative letters constitutes the character set for Common Indic Representation. Any Indic word in a given language is transliterated to the Common Indic Representation on a letter-by-letter basis. For instance, the word અંકુર in Gujarati and అంకుర in Telugu are transliterated to अंकुर.
2. *English Dictionary Word Transliteration.* To address the irregularity problem, English words are transliterated into Devanagari format based on their pronunciation. For example, the sentence "I have a red rose" would be manually transliterated to "आई हैव अ रेड रोज." An extensive dictionary containing 40,000 English words with their corresponding Devanagari transliterations is maintained and updated regularly to facilitate the transliteration process. The Devanagari representation of the English words is then further converted to the Indic Common Representation.
3. *Reduction.* The Indic Common character set is further reduced into smaller sets by grouping consonants or vowels with similar sounds and designating a single letter as their representative. For instance, श, ष, स are grouped, as are इ and ई, among others.

Reverse Dictionary. While performing the above steps, a reverse dictionary is maintained from a reduced Indic Common Representation of a word to its native form. For example: मिठाइ is mapped to मिठाई and મીઠાઈ. Note that the list of native forms may correspond to different languages.

4. Proposed Architecture

Figure 1 shows the proposed architecture for the ASR system. During training, the text from multiple code-mixed languages is transliterated into a common writing system based on the Devanagari script. As described in §3, the transliterated text is grouped into characters with similar sounds to create a rich phonemic representation of the text. The data in the reduced format is used to fine-tune an ASR system based on Wav2Vec 2.0 model with a 5-gram Ken-LM. The system outputs text in reduced format (in Indic Common Representation). During testing, a disambiguation pipeline is used to convert the output text into its native format. We describe the details of each of the components next.

4.1. Fine-Tuning Details

The fine-tuning phase for all experiments utilizes the Fairseq toolkit [18] and the open-source Indic-Wav2Vec-Large pretrained model provided by AI4Bharat,[5] trained on 17k hours of audio and covering 40 languages. Fine-tuning and inference are combined with a 5-gram Statistical Language Model for better results. Fairseq wav2vec dictionary and lexicon files

[3] `https://github.com/jiaaro/pydub`

[4] `https://github.com/wiseman/py-webrtcvad`

[5] `https://ai4bharat.iitm.ac.in/indicwav2vec`

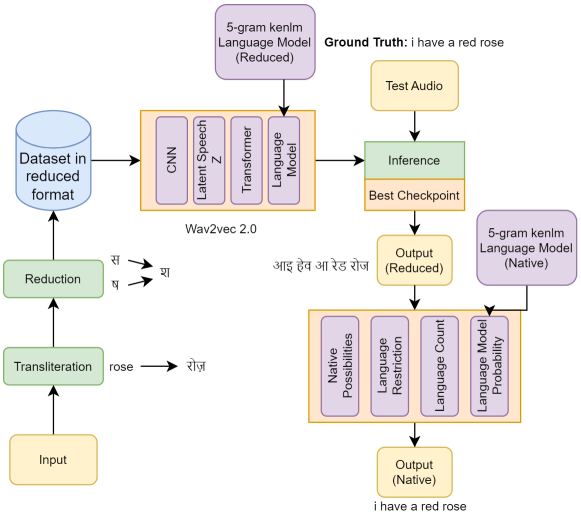

Figure 1: *The proposed architecture*

are built using the train and validation sets and optionally use-case-specific text corpus. The statistical 5-gram language model used for training and benchmarking is built with KenLM,[6] with LM tokens in reduced Indic Common Representation format for fine-tuning and native format for inference. Domain-specific data is trained using the corresponding domain corpus. In contrast, the training data and other sources, such as Google blogs, are used for general training (details in Appendix A[7]).

4.2. Disambiguation and Inference

The ASR system produces the output text in reduced format; the disambiguation pipeline is used to convert all words to their native format using a native LM and reverse dictionary (dict) (§3). For an output sentence $S = [w_1, w_2 ...., w_N]$, an initial score of 1 is assigned to all reduced words. The initial mapping of scores for sentence S becomes [{word:w_1, score:1}....,{word:w_N, score:1}]. The overall word disambiguation is achieved through the following pipeline (described below): Native Possibilities → Language Restriction → Language Count → LM Probabilities → Final Output.

- *Native Possibilities:* For each word (in reduced format) in the output sentence, the corresponding native words are retrieved from the reverse dictionary, and a new score for each word is calculated as follows. For a reverse dictionary, R denoted by $R = [\{w_1\colon [w_{1,1}^{(N)} ...., w_{1,M_1}^{(N)}]\} ....., \{w_R\colon [w_{R,1}^{(N)}, ..., w_{R,M_r}^{(N)}]\}]$, where for a reduced word w_i we have M_i native words $w_{i,1}^{(N)}, ..., w_{i,M_i}^{(N)}$. The score for each native word possibility (corresponding to reduced word w_i) is $1/M_i$. Hence, $w_i = [\{\text{word:}w_{i,1}^{(N)}, \text{score:}1/M_i\} ....,\{\text{word:}w_{i,M_i}^{(N)}, \text{score:}1/M_i\}]$ and so on.
- *Language Restriction:* In a sentence, S, for each reduced word, a non-overlapping window (centered at the word) of 100 is taken, and a set of all native words (coming from the neighboring reduced words in the window) corresponding to the top K languages is selected. If there are n possible native words for a given reduced word, and m belongs to a language L, then the language score for L is computed as m/n. Scores are computed for each language across all native words, and the top K languages are chosen based on the final scores. The native words of these top K languages are selected, and scores for individual possibilities are recalculated using the same method as in the Native Possibilities procedure. For all experiments, the value of K is set to 3.
- *Language Count:* Next, the score of each potential native word is modified based on the languages in its vicinity. A context window (centered at the reduced word in the sentence) of size 17 is selected, and the score for each language is calculated based on potential native words of that language within that window. The score of a language is the sum of scores of all potential native words in that language within the window. The score of each native word is then multiplied by its respective language score for every reduced word. For instance, if a native possibility, $w_{i,1}^{(N)}$ corresponding to reduced word w_1 belongs to Language L with score S_L then its new score would be $Score_{w_{i,1}^{(N)}} * S_L$. Subsequently, the new scores are normalized.
- *Language Model Probability:* After getting the weighted language count scores for various native words across all reduced words, a pre-trained language model is used to assign a score to all possible combinations of native words across a window. For this, a default half window size of 2 with a max size of 5 is chosen. Then potential left and right sentences (within the half-window size boundary) for every native word corresponding to a reduced word are constructed, and the language model is used to assign a probability score, $Score_{LM_{sent}}$ to the overall sentence. Then the overall language model weighted score throughout the window is calculated as $Score_{LM_{sent}} * Score_{left} * Score_{right}$ and the maximum of these scores is considered and multiplied to the score of the native word. These scores are further normalized by the total score across the list of native possibilities for the current reduced word.
- *Final Output* After the computation of this final score, the native words with the highest weighted scores are selected, corresponding to each reduced word.

We also explain the entire disambiguation pipeline with the help of an example in Appendix B.

[6] `https://github.com/kpu/kenlm`

[7] Appendix available at: `https://tinyurl.com/3h9pbvkj`

Table 1: *Dataset Statistics: Numbers of hours of audio.*

Dataset	Languages	Train	Dev	Test	Notation
MUCS	Hindi	92.2	2.85	5.49	MS-H
	Marathi	91.08	2.81	0.667	MS-M
	Tamil	38.8	1.2	4.4	MS-TA
	Telugu	38.8	1.2	4.3	MS-TE
	Gujarati	38.8	1.2	5.00	MS-G
GRAMVAANI	Hindi	100.0	5.00	3.00	GV-H
Call Center (CC)	C-HE	136.04	4.29	2.86	C-HE
	C-HTKE	55.67	1.76	1.17	C-HTKE

5. Experiments

We experimented with three settings: multilingual audio, code-switched/mixed audio, and noisy audio.

5.1. Multilingual Audio

A single multilingual model is more convenient and efficient to maintain and release in production than having multiple models for each language. We experimented with *MUCS (Multilingual and Code-Switching ASR Challenges for Low Resource Indian*

Table 2: *WER different versions of the model on various datasets. The baseline results are on the test set.*

ASR Model	Dataset	Baseline	Validation	Test
M_0	MS-H	32.73	6.42	12.48
	MS-M	29.04	5.22	13.24
	MS-TA	34.09	13.07	19.54
	MS-TE	31.44	14.44	21.33
	MS-G	26.15	13.90	18.15
M_1	GV-H	27.47	16.57	24.35
M_2	C-HE	61.95	21.7	23.9
M_3	C-HTKE	70.87	29.8	31.32

Languages) [19] ASR dataset for this setting. MUCS contains a dataset for six different Indian languages for multilingual ASR tasks: Hindi, Marathi, Odia, Telugu, Tamil, and Gujarati. For this setting, a single lexicon file is built for all languages, and a single language model (based on corpora from all languages) is used.

5.2. Code-Switch and Code-Mixed Audio

Technically speaking, there is a subtle difference between code-switching and code-mixing [1]; however, we group these categories into one for practical purposes. Hindi and other regional languages are frequently mixed with English in India. English words can be inserted in the middle of sentences with the same grammar structure as the regional language, or parts of sentences can be made up of words from multiple languages, each with its grammar structure. We used a propriety dataset from the call-center call recordings for this setting. We refer to this dataset as Call Center (CC) dataset. We did a project on ASR with various companies and curated CC data from them; they (as a part of the standard procedure) recorded telephonic calls from call centers in several commercial settings corresponding to different domains (e.g., Education, Medical, Tourism, etc.). All required prior permissions were taken from the company (and corresponding individuals) during the process. Two Call Center datasets are curated; these closely reflect the challenges posed by noisy, multilingual, and code-mixed speech. The first dataset, called C-HE, consists of a multidomain dataset in Hinglish, a code-mixed language that combines Hindi and English. The second dataset, called C-HTKE, consists of four languages, including Hindi, Tamil, Kannada, and English, and is also noisy and code-mixed. Appendix C provides more details about the CC dataset and annotations.

5.3. Noisy Real-World Audio

The vast majority of Wav2Vec-2.0-based model benchmarks are based on clean datasets. However, how well such models perform on practical noisy audio settings needs to be made clear. Consequently, to find out the generalization capabilities of our model, we experimented with GRAMVAANI [20] dataset. It comprises telephone quality noisy speech data in Hindi; it includes regional/dialectal variations of Hindi. It also contains metadata for the recordings, including location, dialect, emotion, and audio quality. The statistics of datasets are given in Table 1.

6. Results and Discussion

We used the standard evaluation metric of WER (Word Error Rate). Table 2 shows the results for various datasets using different models.

6.1. Results and Analysis

We tried different versions of our proposed architecture. M_0 is our architecture fine-tuned (§4) on MUCS train set. Similarly, M_1, M_2, and M_3 are fine-tuned on GRAMVAANI, C-HE, and C-HTKE data. We compare our system against the test performance of baseline systems proposed for each dataset. For the MUCS dataset, we use Hybrid DNN-HMM model [19] as the baseline, CNN-TDNN ASR based on Kaldi [6] toolkit is used as the baseline for GRAMVAANI. For the CC dataset, the baseline models come from Indian Language ASR: Vakyansh (Wav2Vec based) [5]. We report validation and test data WER. As shown in Table 2, for MUCS, our approach (on the Test set) achieves an average WER of 16.95 across all languages compared to 30.69 WER for the baseline. Our method performs better across all languages. Similarly, our system outperforms baseline models on other datasets by a large margin. Notably, for the CC dataset (noisy and code-mixed speech), our system (23.9, 31.32) outperforms baseline models (61.95, 70.87) by a huge margin. Both MUCS[8] and GRAMVAANI[9] maintain a leaderboard; on both the datasets, at the time of writing this paper, our model performs better than the SOTA. We also compare against Wav2Vec-based models in Appendix E. One of the challenges that we have faced is the need for domain-specific data. In some specialized domains, e.g., medical, banking, and legal, the language used may be very technical and full of jargon. After fine-tuning the acoustic model, we observed that adding data to the language model improves the transcription quality. In certain conditions, out-of-vocabulary words are externally added to the language model and acoustic model vocab to remove the need for additional fine-tuning.

6.2. Limitations

For the CC dataset, we also performed domain (banking, education, medical, and tourism) specific analysis (details in Appendix D). We observed that the proposed ASR system struggles in shouting conditions; shouting distorts speech, making it difficult for the model to accurately transcribe the audio as seen for the BFSI domain compared to the Edtech domain, where the calls are generally less noisy. Another limitation of our approach is that the transliteration step is rule-based. New rules would need to be incorporated to include new languages (e.g., non-Indian languages). Hence, in the future, we would like to develop an automated transliteration module to address this gap.

7. Conclusions

In this paper, we proposed an ASR system for multilingual, code-mixed, and noisy settings. The system has SOTA performance on existing benchmarks, and our experiments show the generalization capability of the system. The proposed system has been successfully deployed in commercial settings with positive responses. In the future, we plan to improve the system further, addressing the mentioned limitations.

[8] https://navana-tech.github.io/MUCS2021

[9] https://sites.google.com/view/gramvaaniasrchallenge

8. References

[1] S. Thara and P. Poornachandran, "Code-mixing: A brief survey," in *2018 International conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 2018, pp. 2382–2388.

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

[3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 12 449–12 460. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf

[4] T. Javed, S. Doddapaneni, A. Raman, K. S. Bhogale, G. Ramesh, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Towards building asr systems for the next billion users," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, pp. 10 813–10 821, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/21327

[5] H. S. Chadha, P. Shah, A. Dhuriya, N. Chhimwal, A. Gupta, and V. Raghavan, "Code switched and code mixed speech recognition for indic languages," 2022. [Online]. Available: https://arxiv.org/abs/2203.16578

[6] 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 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

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

[9] J. Chi and P. Bell, "Improving code-switched asr with linguistic information," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 7171–7176.

[10] B. Yan, C. Zhang, M. Yu, S.-X. Zhang, S. Dalmia, D. Berrebbi, C. Weng, S. Watanabe, and D. Yu, "Joint modeling of code-switched and monolingual asr via conditional factorization," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6412–6416.

[11] I. Hamed, N. T. Vu, and S. Abdennadher, "Arzen: A speech corpus for code-switched egyptian arabic-english," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, pp. 4237–4246.

[12] S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black, "A survey of code-switched speech and language processing," *arXiv preprint arXiv:1904.00784*, 2019.

[13] S. Dalmia, Y. Liu, S. Ronanki, and K. Kirchhoff, "Transformer-transducers for code-switched speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5859–5863.

[14] S. Bansal, A. Mukherjee, S. Satpal, and R. Mehta, "On Improving Code Mixed Speech Synthesis with Mixlingual Grapheme-to-Phoneme Model," in *Proc. Interspeech 2020*, 2020, pp. 2957–2961.

[15] V. Pratap, A. Sriram, P. Tomasello, A. Hannun, V. Liptchinsky, G. Synnaeve, and R. Collobert, "Massively multilingual asr: 50 languages, 1 model, 1 billion parameters," 2020. [Online]. Available: https://arxiv.org/abs/2007.03001

[16] A. Diwan and P. Jyothi, "Reduce and Reconstruct: ASR for Low-Resource Phonetic Languages," in *Proc. Interspeech 2021*, 2021, pp. 3445–3449.

[17] D. Le, T. Koehler, C. Fuegen, and M. L. Seltzer, "G2g: Tts-driven pronunciation learning for graphemic hybrid asr," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6869–6873.

[18] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[19] 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," *Proceedings of Interspeech*, 2021.

[20] A. Bhanushali, G. Bridgman, D. G, P. Ghosh, P. Kumar, S. Kumar, A. Raj Kolladath, N. Ravi, A. Seth, A. Seth, A. Singh, V. Sukhadia, U. S, S. Udupa, and L. V. S. V. D. Prasad, "Gram Vaani ASR Challenge on spontaneous telephone speech recordings in regional variations of Hindi," in *Proc. Interspeech 2022*, 2022, pp. 3548–3552.