Improving Multilingual ASR Robustness to Errors in Language Input

Brady Houston[1]*, Omid Sadjadi*[1]*, Zejiang Hou*[1]*, Srikanth Vishnubhotla*[1]*, Kyu Han*[1]

[1]AWS AI Labs, USA

{hstbrady,sadjadi,zejiangh,srikvish,kyujhan}@amazon.com

Abstract

Explicitly adding language information to multilingual ASR models during training has been shown to improve their performance. However, this also requires using language information during inference. In cascaded systems, this language label may come from external language identification models, which are susceptible to errors. In this work, we characterize the sensitivity to errors in language inputs of several common language-incorporation strategies used in multilingual ASR. We show that some of these strategies are highly sensitive to the correctness of language information being used during inference, and also demonstrate that introducing a small amount of language label noise during training can greatly improve the model's robustness to incorrect language information. As multilingual ASR continues to become more common, this work demonstrates the importance of understanding the sensitivity of these models to language inputs and ensuring models are robust to errors.

Index Terms: speech recognition, multilingual, robustness, language identification

1. Introduction

Creating a single model that can recognize as many as 4,000 languages [1] has emerged as a recent trend in automatic speech recognition (ASR). As modern ASR models typically use grapheme-based targets as outputs, the most basic multilingual ASR (mASR) modeling approach entails pooling all transcripts to train a tokenizer, and then using the tokenized transcripts to train the mASR model. Despite its simplicity, this approach can result in significant improvements over language-specific models [2]. Further improvements have been achieved by incorporating language information into the model; adding one-hot encoded language vectors as inputs to the model during training reduced word error rate (WER) by more than half [3], and adding learned language embeddings disproportionately improved performance on low-resource languages by up to 23% [2]. Creating language-specific sets of parameters within a multilingual model either during training [4, 5] or subsequent fine-tuning resulted in similarly large WER improvements in some cases [3]. Multi-task training, e.g., with language identification (LID), has been used to encourage the model to encode language information from audio, resulting in improved performance [6, 7, 8]. Decoder-based approaches, like prompting using the target language, have also yielded improvements [9].

While these approaches can improve mASR models, explicitly adding language information during training typically necessitates providing the same information during inference. This requires either having *a priori* knowledge about the language(s) seen during inference or the ASR system needs to make a prediction for the language being spoken (either in the ASR model itself, or through a separate LID model). As with any machine learning model, LID models are susceptible to errors, especially when they cover a large number of languages, potentially for highly-similar and/or low-resource languages, or encounter unseen accents [10, 11, 12]. In a cascaded mASR system, these errors are propagated to the mASR model (if it uses them). This is not a well-studied area, but we show in this work that mASR models that were trained to use language information can be very sensitive to the correctness of this LID input and incorrect language inputs can dramatically impact their performance.

There has been little investigation of how sensitive mASR models are to these language inputs, particularly when the language inputs can be predicted by a separate machine learning model as part of a cascaded system. It is difficult to find relevant literature in other fields as well; the most relevant work is [13], where authors characterize the sensitivity of a spoken-language understanding model to label noise, and explore approaches to improve the robustness of the model to such noise. In machine translation, [14] showed that incorporating LID outputs improved a cascaded document translation system, but did not explore how errors in the LID models impact performance. In a multimodal speech and video system, it was shown in [15] that data missing from one modality can cause a drop in performance. The authors proposed several approaches for improving the model's robustness to such phenomena, but did not investigate the impact of incorrect data in one modality. There is ample research on error propagation and improving robustness in cascaded systems that rely on ASR *outputs* (e.g. [16], where authors add noise to ASR outputs to make the downstream translation task more robust to ASR errors), but none that explore robustness in ASR *inputs*.

In this work, we explore several commonly-used approaches for explicitly incorporating language information into an mASR model, validate their effectiveness in improving mASR performance, and characterize the robustness of these different approaches to incorrect LID inputs during inference. We then show that the robustness to errors in LID input can be greatly improved in some language-incorporation approaches by introducing language label noise during model training. This approach of characterization and targeted training to improve robustness are the first we are aware of, and this work highlights the importance of training mASR models to not become overdependent on the correctness of LID information to produce reasonable results.

2. Language-Integration Approaches

We utilize several approaches for injecting language information into mASR models during training. In the first, an 8-

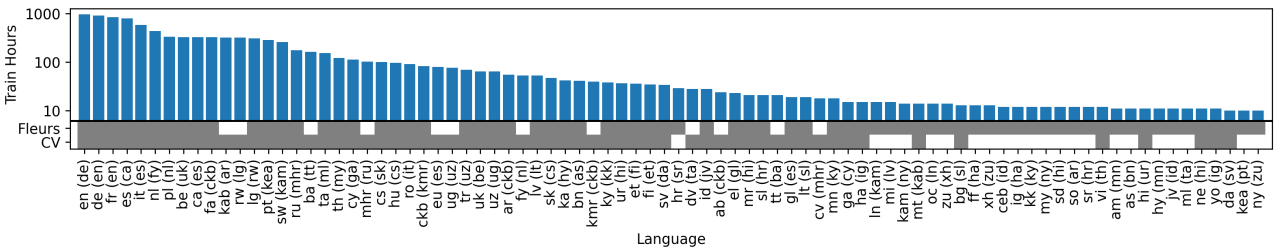

Figure 1: *Training data volume (blue bar) by language code (ISO 639-2) and languages used in evaluation data sets (gray indicates that test set covers the language). The most probable alternate language chosen by the external LID model is indicated in parentheses.*

dimension learned embedding (i.e. learned vector of parameters for each language) is replicated across the time dimension and concatenated with the filter bank inputs [2, 17, 18]. A similar approach that has been explored in previous works is to use one-hot vectors for each language [3]; however this would entail more than doubling the input dimension to the model, so instead we focus on the smaller learned embedding. During inference, a language is provided to the model to select the desired embedding to append to the input features.

The second approach is a multi-task ASR + LID architecture, where the multitask LID posteriors are fed back into the encoder via self-conditioning (SC-LID) [8]. Specifically, the outputs from the fourth encoder layer are average-pooled across the time dimension and inputted to a softmax layer that predicts the segment's language. The softmax posteriors are then transformed back into the model dimension via a dense layer, replicated across the time dimension and added back to the fourth encoder layer's output. During training, a cross-entropy LID loss is added to the ASR task with a weight of 0.1. During inference, this architecture can make an LID prediction from the outputs of the fourth encoder layer and use the predicted posteriors in the remaining encoder layers. Additionally, the predicted LID posteriors can be replaced with a one-hot vector to force the model to use a specified language and ignore the prediction made from the LID module in the remaining encoder layers.

The last approach used is a two-stage training using language adapters [3, 19, 20, 21], where the base encoder is frozen and language-specific adapters are added between all encoder layers and trained for 20 epochs. Each adapter consists of a layer norm, down projection (dim=16), ReLU nonlinearity and up projection back to the encoder dim of 512. The adapter's output is added back to the encoder layer's activations via residual connection before being passed on to the next encoder layer. During training, the adapter parameters for a given language are only updated by training examples from that language. We add about 240K parameters for each language, increasing the total number of model parameters by 20%. During inference, the adapter parameters are selected using a provided language.

3. Experiments

3.1. Data

Paired speech-text data were used from Fleurs [22], Mozilla Commonvoice V14.0 (CV14) [23], Librivox-MLS [24] and VoxPopuli (labelled portion only) [25] (public train, dev and test splits are used). To balance across these, a maximum of 320 hours were randomly chosen from each collection for each language and then only languages that had at least 10 total hours of training data were kept (Chinese, Korean and Japanese were excluded due to their large tokenizer requirements), yielding a 10k hour final training data set comprising 84 languages (see Figure 1, top panel). The goals in creating this dataset were to 1) have as many languages as possible represented, 2) recreate the imbalance in data volumes typically seen by mASR modeling, and 3) be small enough in size for quick experimental iterations but large enough to increase the likelihood that results can be extrapolated to larger datasets.

Transcripts from each language were normalized by removing punctuation and casing, performing NFKC unicode normalization and removing annotator content in between parentheses. For each language, a 192-piece BPE tokenizer (384 pieces for Amharic, due to its larger character set) was trained using the sentencepiece library [26]. Then, the tokenizers were combined and de-duplicated, resulting in a final vocabulary size of 6,144 tokens. A subword 4-gram language model was trained from the pooled transcripts and used during decoding. Although the four different data collections were used for training, we focus only on Fleurs and Commonvoice test sets for evaluation since their language coverage far exceeds that of Librivox-MLS and VoxPopuli. We also note that CV14 test sets are only used for languages that have at least 1.5 hours of CV14 training data, as we find that results are very unreliable on CV14 test sets for languages that do not meet this threshold.

3.2. Modeling

The model architecture used in all experiments is a 100M parameter 16-layer conformer-CTC encoder [27] with 8 self-attention heads and a model dimension of 512. The model uses a 1-layer transformer-based attention decoder during training only for regularization purposes [28, 29]. The model input is 80-dim log-mel filter-bank energies (25ms window, 10ms hop) and a convolutional frontend is used to downsample frames by 4x. Models are trained for 40 epochs (about 160k steps over 40 hours) on eight A100 GPUs with a warmup of 10k steps and a peak learning rate of 1e-3. Parameters are optimized using AdamW with a weight decay value of 0.1. The model parameters from the final 5 epochs are averaged and used for inference. During inference, the model uses only the CTC output layer, and the subword 4-gram LM is incorporated into beamsearch via shallow fusion with an LM weight of 0.6 and beam size of 100. Performance is reported as WER.

We also train an LID model using a 5-layer time-delay neural network (TDNN) architecture that incorporates channel attention as well as multi-layer feature aggregation [30]. Each layer consists of a 1-D Res2Net block with 512 channels and squeeze-and-excitation (SE) components. As input representations for the LID model, we use the outputs from the ultimate layer of the baseline acoustic encoder discussed above. The LID model parameters are optimized using stochastic gradient descent (SGD) with momentum (0.9) and the cross-entropy loss. We use this LID model to choose incorrect languages to use

Model Type	ExpID	Train Perturb		Fleurs (72 Languages)				CommonVoice V14.0 (61 Languages)			
		Wrong LID	Unk. LID	Corr. LID	Alt. LID	Cascade LID	Unk. LID	Corr. LID	Alt. LID	Cascade LID	Unk. LID
Baseline	B0	-	-	27.3	-	-	-	24.3	-	-	-
Language embedding	E0	-	-	23.6	85.6	28.8	-	20.3	84.7	23.9	-
	E1	0.5%	-	23.7	55.2	27.0	-	20.6	48.2	22.7	-
	E2	1.0%	-	23.5	45.5	26.2	-	20.6	38.2	22.3	-
	E3	2.0%	-	23.8	41.4	26.3	-	20.9	34.5	22.3	-
	E4	5.0%	-	24.2	33.4	25.7	-	21.4	29.6	22.5	-
	E5	-	1.0%	23.5	81.5	28.6	30.7	20.3	76.7	23.7	31.0
	E6	1.0%	1.0%	23.8	43.7	26.3	28.9	20.6	37.6	22.2	26.7
SC-LID	S0	-	-	23.5	66.3	27.8	-	20.4	49.3	22.9	-
	S1	1.0%	-	23.5	65.5	27.8	-	20.4	51.2	22.9	-
Adapters	A0	-	-	25.8	29.7	26.4	-	21.2	26.9	22.0	-
	A1	0.5%	-	25.8	29.2	26.3	-	21.3	25.9	22.1	-
	A2	1.0%	-	25.9	28.9	26.3	-	21.4	25.7	22.2	-
	A3	2.0%	-	26.1	28.7	26.5	-	21.6	25.5	22.3	-
	A4	5.0%	-	25.8	29.7	26.4	-	21.8	25.3	22.4	-
	A5	-	1.0%	25.7	29.7	26.4	27.4	21.2	26.4	22.0	24.1
	A6	1.0%	1.0%	25.9	28.9	26.3	27.3	21.5	25.9	22.2	24.0

Table 1: *Average WER (across languages) of models when correct, incorrect LIDs or an "unknown" LID are used during inference. "Alt. LID" shows worst-case WER when the language with the second-highest average posterior as per the external LID model is always used during inference. "Cascade LID" shows WER when the external LID model's predicted posteriors are used select a language during inference (Section 3.2). Training perturbations show what percentage of training examples were replaced with either a wrong LID and/or "unknown" LID.*

during inference. First, a most-probable alternate language is chosen by evaluating the model on all test datasets, averaging the posteriors across all segments, and picking the incorrect language with the highest average posterior probability. This approach typically chooses alternate languages that are within the same language family or are otherwise very similar (see Figure 1 for alternate languages). We also simulate a cascaded ASR system by using these averaged LID posteriors to choose LIDs during inference. For consistency in this simulation, we assign a correct/incorrect LID to each segment using these posteriors and reuse that LID label in all results for the cascaded system. The macro F1 score for this LID model across languages is 94.4%. Out of 84 languages, 62 have F1-scores $> 95\%$, 73 have scores $> 90\%$, and only 3 languages have scores $< 75\%$; Urdu (70%), Serbian (54%) and Hindi (38%).

4. Results and Discussion

4.1. Characterizing LID Error Sensitivity

First, we validate that incorporating language information into the models yields improvements over the baseline model. We note here that we do not include the performance of other mASR models from literature as a reference since the training data set we use is a subset of the full collections (both in languages and data per language), and thus fair comparison is difficult. All three approaches used to incorporate LID information show improvements over the baseline model (B0 in Table 1) when the correct language is provided to the model during inference; the language embedding (E0) and SC-LID (S0) approaches show similar improvements of about 14-16% relative to the baseline model on Fleurs and CV14, while using language adapters (A0) yields a somewhat smaller improvement of 6% on Fleurs and 13% on CV14. The smaller improvements on Fleurs when using adapters can be disproportionately attributed to performance on Urdu; despite using language-specific adapter parameters, the model still mainly predicts Urdu speech using characters from Hindi. In this case, the language adapters for Urdu do not seem to be able to offset the strong bias the base model learns towards Hindi. Removing Urdu from the results increases the WER improvement to 9% on Fleurs.

Next, we characterize model sensitivity to errors in LID during inference in two different scenarios. First, we show the "worst-case" performance; the external LID model's most probable alternate language is always used in order to highlight the sensitivity of different language-integration approaches to incorrect language inputs, even if they are linguistically similar (columns "Alt. LID" in Table 1). Second, we use the LID model's averaged posteriors across all test data as probabilities to randomly assign an LID label to each segment during inference, simulating an actual cascaded ASR system (see section 3.2 for details; results in columns "Cascade LID" in Table 1).

The language embedding approach (E0) shows the highest sensitivity, with WER degrading to around 85% when the most probable alternate language is used, and a relative drop of 18-22% WER in the cascaded system simulation (compared to using correct LID). In worst-case results, language adapters show relative degradation of 15% on Fleurs and 27% on CV14 (A0) and only 2-4% degradation in the cascaded simulation. SC-LID performance (S0) is in between that of the other two approaches when both the most probable alternate and external LID posteriors are used for inference language. The greater inherent robustness of language adapters can likely be attributed to the residual connection in the adapter that can either circumvent or compensate for the incorrect language adapter being used. The relatively better (although still poor) robustness of the SC-LID model is likely due to the errors its internal LID prediction module makes during training, and the model learning to compensate for these errors in the posteriors being fed into the dowstream encoder layers. The macro-average accuracy of the

SC-LID model's internal LID prediction is 89% on Fleurs and 88% on CV14. When using the internal LID prediction mode during inference, the SC-LID model has an average WER of 28.2 on Fleurs and 24.5 on CV14, which is a 1-3% degradation compared to the baseline model's performance on each dataset (not shown in table).

4.2. Improving Robustness to LID Errors

We then re-train the above models but randomly replace the ground truth LID with an incorrect LID (uniformly sampled across all languages) in 0.5% - 5% of training examples and repeat the same characterization. This training strategy has varying effects across the three language-incorporation approaches. First, using wrong LID during training with language embeddings can dramatically improve robustness; providing incorrect LID labels in 5% of training examples (E4) results in WER improvements in Fleurs from 85.6% to 33.4% when using an alternate language, with similar levels of improvement seen on CV14. This comes with a 2-5% WER degradation on Fleurs and CV14 (compared to E0), when using the correct language during inference. Using the simulated cascaded ASR system for generating inference LID labels, we observe best improvements of 11% on Fleurs (E4) and 7% on CV14 (E3) compared to E0. Table 2 shows an example Fleurs segment demonstrating the effects of input LID with and without label noise during training. Even when linguistically similar languages are used for inference (most probable alternate LID German and cascaded system LID Swedish), the ASR output can become incomprehensible (E0). When LID label noise is used to improve robustness, however, the quality of the output is similar when different LID labels are used as inputs (E4).

When using SC-LID, the LID perturbations during training had little effect (S1); since wrong LIDs are provided as a *target* in this approach, they seem to not be strong enough to cause the LID module to learn robustness in the posteriors being predicted and forwarded to the rest of the model. To remedy this, we attempted to explore using higher LID loss weights during training, but this impacted model convergence. Future work will further explore approaches for improving the robustness of this type of model, including possibly forcing LID posteriors to represent wrong LIDs during training (rather than just relying on the LID target being wrong). For language adapters (A1-A5), the effects of using wrong LID labels during training are much smaller than when using language embeddings; on Fleurs, there are slight improvements in the cascaded system simulation, and on CV14, there are slight degradations (compared to A0).

These results highlight the importance of understanding how commonly-used language-incorporation strategies for mASR models may be sensitive to language inputs (in some cases extremely sensitive) or inherently more robust, and how introducing LID label noise during training can improve robustness in some approaches. However, these results raise the question - is simply avoiding modeling approaches that explicitly use language information a better strategy than trying to improve robustness? All language-incorporation strategies show improvements over the baseline when correct language information is provided and degradation over the baseline in the worst case scenario where incorrect language information is always provided (even when robustness approaches are utilized). However, results from the cascaded ASR system simulation (wherein an external LID model is used to provide an inference language) suggest that both language embeddings and language adapters can improve upon a language-free modeling approach even if the system experiences errors in LID inputs to these models, especially if care is taken to ensure robustness to these errors.

Exp ID	Inf LID	Transcript
GT	-	italian is also the everyday language used by most of those who work in the state while latin is often used in religious ceremonies
E0	en	(matches ground truth)
	de	italien ist also der everyday language used bei mustafus hubican state wleitende soft the news and religious ceremonies
	sv	italien är alltså d everyday language usban muss who work a s och lärness after new ze religious ceremonies
E4	en	italian is also that everyday language used by most of those who work in the state while latin is often used in religious ceremonies
	de	italian is also that everyday in language used by most of those who work in a state while latin is often used in religious ceremonies
	sv	italian is also that everyday language used by most of those who work in the state while latin is often used in religious ceremonies

Table 2: *Example from English Fleurs test data showing how models respond to different languages being used during inference. GT in Exp ID refers to the ground truth transcript.*

4.3. Using "Unknown" LID

We also explore replacing the correct LID label with an "unknown"/"blank" language label in 1% of training examples (treated as a new, 85th language). This approach could be used when the confidence of an external LID model does not meet a threshold, and instead of providing a low-confidence LID label, the "unknown" label could be used. Using "unknown" label during training does not typically hurt performance when the correct LID label is used during inference and performance is much better when using "unknown" label for all test segments instead of an incorrect LID label (E5, A5). This effect can be improved further when also using wrong LID label 1% of the time during training (E6, A6). Further work will continue to explore this area, possibly by incorporating the "unknown" and provided LID inputs in a mixture-of-experts approach to see if the model can learn when to use each LID label, as well as using the "unknown" label in a cascaded ASR system simulation where the LID model learns to output this label when it is not confident in its prediction.

5. Conclusion

This novel work shows that three popular approaches for adding language information to multilingual ASR models have widely varying levels of sensitivity to the correctness of these language inputs. It also demonstrates the utility, especially when using learned language embeddings, of occasionally providing incorrect language information during training. While this work explored using language information in encoders, future work will also look at robustness of language information in decoder approaches (e.g. Whisper-style [9]), as well as explore the robustness of language prediction as part of speech translation models.

6. References

[1] V. Pratap, A. Tjandra, B. Shi, P. Tomasello, A. Babu, S. Kundu, A. Elkahky, Z. Ni, A. Vyas, M. Fazel-Zarandi *et al.*, "Scaling speech technology to 1,000+ languages," *arXiv preprint arXiv:2305.13516*, 2023.

[2] 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," Jul. 2020, arXiv:2007.03001 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2007.03001

[3] A. Kannan, A. Datta, T. N. Sainath, E. Weinstein, B. Ramabhadran, Y. Wu, A. Bapna, Z. Chen, and S. Lee, "Large-Scale Multilingual Speech Recognition with a Streaming End-to-End Model," Sep. 2019, arXiv:1909.05330 [cs, eess, stat]. [Online]. Available: http://arxiv.org/abs/1909.05330

[4] N. Gaur, B. Farris, P. Haghani, I. Leal *et al.*, "Mixture of informed experts for multilingual speech recognition," in *Proc. IEEE ICASSP*, 2021, pp. 6234–6238.

[5] B. Houston and K. Kirchhoff, "Exploration of language-specific self-attention parameters for multilingual end-to-end speech recognition," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2023, pp. 755–762.

[6] S. Watanabe, T. Hori, and J. R. Hershey, "Language independent end-to-end architecture for joint language identification and speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 265–271.

[7] C. Zhang, B. Li, T. N. Sainath, T. Strohman *et al.*, "Streaming end-to-end multilingual speech recognition with joint language identification," in *Proc. Interspeech*, 2022, pp. 3223–3227.

[8] W. Chen, B. Yan, J. Shi, Y. Peng, S. Maiti, and S. Watanabe, "Improving massively multilingual asr with auxiliary CTC objectives," in *Proc. IEEE ICASSP*, 2023, pp. 1–5.

[9] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*, 2023, pp. 28 492–28 518.

[10] G. Gelly, J.-L. Gauvain, L. Lamel, A. Laurent, V. B. Le, and A. Messaoudi, "Language recognition for dialects and closely related languages." in *Odyssey*, vol. 2016, 2016, pp. 124–131.

[11] M. Chaitra, A. Mandal, and S. Mukherjee, "Study of ecapa-tdnn models for spoken language identification task," in *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*. IEEE, 2023, pp. 233–237.

[12] K. Kukk and T. Alumäe, "Improving language identification of accented speech," *arXiv preprint arXiv:2203.16972*, 2022.

[13] A. Kumar, P. Sharma, A. Illa, S. Venkatapathy, S. Nandi, P. Varma, A. Dwarakanath, and A. Galstyan, "Learning under label noise for robust spoken language understanding systems," 2022.

[14] A. Babhulgaonkar and S. Sonavane, "Language Identification for Multilingual Machine Translation," in *2020 International Conference on Communication and Signal Processing (ICCSP)*. Chennai, India: IEEE, Jul. 2020, pp. 401–405. [Online]. Available: https://ieeexplore.ieee.org/document/9182184/

[15] S. Parthasarathy and S. Sundaram, "Training Strategies to Handle Missing Modalities for Audio-Visual Expression Recognition," Nov. 2020, arXiv:2010.00734 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2010.00734

[16] X. Li, H. Xue, W. Chen, Y. Liu, Y. Feng, and Q. Liu, "Improving the robustness of speech translation," 2018.

[17] T. Tanaka, R. Masumura, M. Ihori, H. Sato *et al.*, "Leveraging language embeddings for cross-lingual self-supervised speech representation learning," in *Proc. IEEE ICASSP*, 2023, pp. 1–5.

[18] F. Ding, G. Wan, P. Li, J. Pan, and C. Liu, "Improved self-supervised multilingual speech representation learning combined with auxiliary language information," *arXiv preprint arXiv:2212.03476*, 2022.

[19] B. Li, R. Pang, Y. Zhang, T. N. Sainath *et al.*, "Massively multilingual asr: A lifelong learning solution," in *Proc. IEEE ICASSP*, 2022, pp. 6397–6401.

[20] Y. Zhang, W. Han, J. Qin, Y. Wang *et al.*, "Google USM: scaling automatic speech recognition beyond 100 languages," *CoRR*, vol. abs/2303.01037, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.01037

[21] J. Bai, B. Li, Q. Li, T. N. Sainath, and T. Strohman, "Efficient adapter finetuning for tail languages in streaming multilingual asr," *arXiv preprint arXiv:2401.08992*, 2024.

[22] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, "Fleurs: Few-shot learning evaluation of universal representations of speech," in *Proc. IEEE Spoken Language Technology Workshop (SLT)*, 2023, pp. 798–805.

[23] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[24] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "MLS: A large-scale multilingual dataset for speech research," *arXiv preprint arXiv:2012.03411*, 2020.

[25] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," *arXiv preprint arXiv:2101.00390*, 2021.

[26] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[27] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[28] T. Hori, S. Watanabe, and J. R. Hershey, "Joint ctc/attention decoding for end-to-end speech recognition," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 518–529.

[29] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *Proc. IEEE ICASSP*, 2021, pp. 6224–6228.

[30] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Interspeech*, 2020, pp. 3830–3834.