# Multiple Softmax Architecture for Streaming Multilingual End-to-End ASR Systems

*Vikas Joshi*, Amit Das**, Eric Sun**, Rupesh R. Mehta*, Jinyu Li**, Yifan Gong***

*Microsoft Corporation, India
**Microsoft Corporation, United States of America

vikas.joshi, amit.das, ersun, rupesh.mehta, jinyli, ygong@microsoft.com

## Abstract

Improving multilingual end-to-end (E2E) automatic speech recognition (ASR) systems have manifold advantages. They simplify the training strategy, are easier to scale and exhibit better performance over monolingual models. However, it is still challenging to use a single multilingual model to recognize multiple languages without knowing the input language, as most multilingual models assume the availability of the input language. In this paper, we introduce multi-softmax model to improve the multilingual recurrent neural network transducer (RNN-T) models, by having language specific softmax, joint and embedding layers, while sharing rest of the parameters. We extend the multi-softmax model to work without knowing the input language, by integrating a language identification (LID) model, that estimates the LID on-the-fly and also does the recognition at the same time. The multi-softmax model outperforms monolingual models with an average word error rate relative (WERR) reduction of $4.65\%$ on Indian languages. Finetuning further improves the WERR reduction to $12.2\%$. The multi-softmax model with on-the-fly LID estimation, shows WERR reduction of $13.86\%$ compared to the multilingual baseline.

**Index Terms**: multilingual, RNN-T, end-to-end, language identification, streaming ASR

## 1. Introduction

There are about 6900 languages in the world and about 99% of them are under-resourced for building ASR systems [1]. The problem of low resource language is well addressed by multilingual ASR systems. It is now a common practice to train multilingual acoustic models (AM) by pooling data from multiple languages [2,3]. A single multilingual model often outperforms monolingual models [4]. Moreover, it simplifies the training and maintenance process as the same model can be deployed across multiple languages. Meanwhile, end-to-end (E2E) ASR models are emerging as a popular alternative to conventional hybrid ASR, as they simplify the ASR systems by directly translating the input speech sequence to an output text with a single neural model. Recurrent neural network transducer (RNN-T) [5–10] is a popular streaming E2E model suitable for real-time applications. Hence, there is a significant interest in advancing the performance of multilingual RNN-T models [4, 11–15]. In this work, we aim to advance the multilingual RNN-T model performance with the proposed architecture.

With increasing adoption of voice assistants, newer scenarios arise. Users now expect a truly multilingual experience, where they interact with voice assistants in any language, interchangeably, without explicitly setting the language of the conversation, and also expect the response in real-time. This is challenging with the current multilingual systems (both hybrid and E2E), as most multilingual models need to know the input language before inference, otherwise the recognition accuracy significantly drops [4]. A common approach to provide a truly multilingual experience is by running multiple monolingual models, along with language identification (LID) model in tandem. Such an architecture with so many monolingual models, is not only cumbersome to maintain, but also not suitable for on-device applications, as the model size increases linearly with the addition of languages. This can be alleviated by designing a multilingual ASR model, which does not require the LID input beforehand, and is streaming in nature. We refer to such a model as LID-free streaming multilingual ASR model. In this work, we also propose a novel multilingual RNN-T model which is LID-free, streaming and multilingual.

We propose a streaming multilingual RNN-T model with language specific softmax, joint and embedding layers, while sharing encoder and prediction layers across languages. We refer to this configuration as the multi-softmax model. We further extend the same model to LID-free version, by embedding the LID model into the multi-softmax model. The combined model is able to estimate the LID on-the-fly and also produce text output in real-time.

## 2. Relation to prior work

Several studies have focused on building multilingual or multidialectal AMs using the hybrid modeling pipeline [16–27]. These can be classified into three categories: transfer learning, multi-task learning and ensemble learning. In transfer learning [19, 20], a well trained AM from high-resource language is used to bootstrap the target language AM. In multi-task learning [21–24], multiple languages are pooled together and trained jointly with distinct output layers for each language. In ensemble learning [25–27], output posterior probabilities of multiple pretrained models are combined to generate a single output.

Similarly transfer learning [14, 15] and multilingual modeling is proposed in the context of E2E ASR models as well [4, 11–13, 28]. The authors in [4] proposed a streaming multilingual RNN-T model with language specific adapters and utilizing the language ID to improve the performance. On similar lines, [13] proposed a Transformer-Transducer multilingual model and is adapted based on the language ID. Audio-to-bytes [29] used bytes instead of graphemes in order to scale a multilingual model to support a large number of languages without increasing the size of the softmax. A multilingual sequence-to-sequence (S2S) model using shared encoder and multiple decoders, one decoder per language group, was proposed in [28]. Our proposed method differs from the above in terms of the model architecture. We have language specific output layers to disambiguate output units between languages and thus mitigating inter-language confusion. Moreover none of the above

approaches is LID-free, while our model is.

Recently, LID-free multilingual models are proposed in [11] and [12]. [11] introduces LID tokens as part of the symbol set and adds them to the start and end of the utterance. However, it is in context of S2S models and is not streaming in nature. Authors in [12] use acoustic LID embeddings along with language tokens at the output, instead of using a predefined onehot language ID. In contrast, we propose a new multilingual RNN-T architecture, which we extend to LID-free scenario.

## 3. Multilingual RNN-T model

A RNN-T model [5] has mainly three components namely: encoder, prediction network and joint network. Encoder transforms the acoustic feature vector into higher dimensional representation. A prediction network transforms the previously predicted non-blank symbol into higher order representation. The joint network along with softmax combines the encoder and prediction network representation to produce posterior probabilities over the symbols. A naive multilingual RNN-T model can be trained by simply pooling the training data and output symbols from all languages, referred to as vanilla model. The vanilla model is simple, yet is a LID-free streaming multilingual model that recognizes multiple languages without needing the input LID. Providing the language information to the vanilla model, via language specific onehot vector significantly improve the performance [4] . The resulting model with onehot embedding at input is referred as the onehot model. The onehot model needs to know the spoken language ID apriori and hence is not LID-free. We use vanilla and onehot models as the multilingual baselines to compare our methods with.

## 4. Multi-softmax model architecture

We propose a multi-softmax architecture for multilingual RNN-T model as shown in Fig. 1. The proposed architecture has multiple softmax layers (one per language) and each softmax layer is composed of corresponding language-specific symbols. This contrasts with vanilla and onehot models which use single softmax layer composed of union of symbols from all languages. Multi-softmax model has language-specific joint and embedding layers, with shared encoder and prediction networks.

During training, each mini-batch consists of data randomly sampled from a particular language, and hence only the shared and language specific parameters are updated for that mini-batch. The language is selected randomly from the distribution obtained over the number of hours of training data. During inference, the posterior computation and beam search decoding is done only for the symbols corresponding to the spoken language. This is in contrast with the vanilla and onehot models, where the posteriors are computed over entire symbol set. Multi-softmax model assumes the availability of spoken language ID and hence is not LID-free. The multi-softmax model can be extended to the LID-free version and is discussed in Section 4.2. The performance of the multi-softmax model for a particular language can be further improved by finetuning the shared and language specific parameters to that language (while discarding the remaining parameters).

### 4.1. Multi-softmax with language specific prediction (LSP) networks

The task of predicting the next symbol, is language dependent, and hence, we explored having a separate prediction network

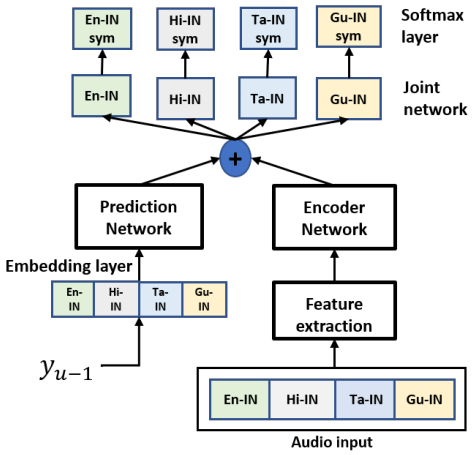


Figure 1: *The multi-softmax model.*

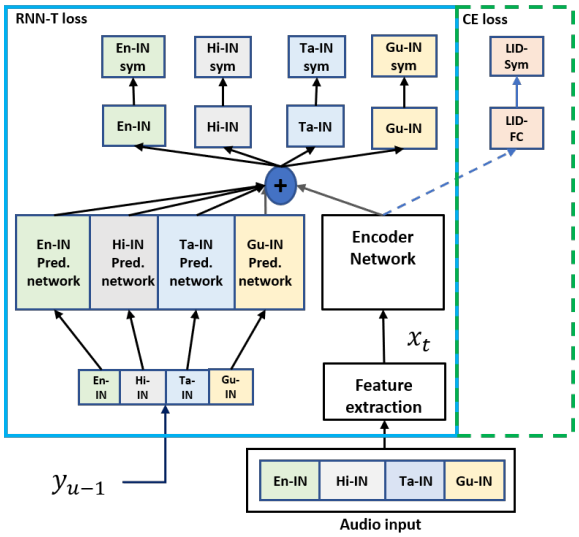


Figure 2: *The blue solid box depicts multi-softmax with language specific prediction network (LSP) and green dotted box shows an integrated LID model.*

per language as shown in the blue solid box in Fig. 2. This model is referred to as multi-softmax LSP model. Training and inference steps are same as done in multi-softmax model.

### 4.2. Multi-softmax with streaming LID

The multi-softmax model is extended to LID-free version by integrating a LID model. This is done by using LID specific output and softmax layer as shown by green dotted box in Fig 2. For the sake of convenience, LID integration is shown with multi-softmax LSP network, however, same can be done with multi-softmax model, referred to as multi-softmax LID model.

**Training and inference:** The output symbols of the LID model is simply the list of all unique symbols representing all the languages. The LID model is trained jointly with the RNN-T model using the same training data, except that the output labels for every frame are replaced with the corresponding language symbol. We use cross entropy (CE) loss, applied at every frame, to train the LID model. The final loss is the addition of RNN-T and CE loss; however, CE loss only affects the encoder and LID specific parameters. During inference, beam search

Table 1: *Training and test data size in hours. Vocabulary size (based on graphemes) per language is also shown.*

| Language | En-IN | Hi-IN | Ta-IN | Gu-IN |
|---|---|---|---|---|
| Train | 10077 | 10591 | 925 | 1205 |
| Test | 46 | 19 | 14.3 | 14.5 |
| Vocab size | 133 | 59 | 97 | 123 |

decoding is run for every language in parallel. The LID model is also run in tandem to produce language posteriors at every frame. As the entire audio is processed in a streaming fashion, beam-search decoded outputs are produced for every language. Meanwhile, LID posteriors averaged over all frames are computed for each language. Finally, the output corresponding to the language with the highest average posterior is chosen as the ASR hypothesis.

The parsimonious multi-softmax LID model requires lower memory and fewer computations compared to using a series of language-dependent monolingual models. First, a new language addition adds only a few parameters in the multi-softmax LID model corresponding to the output and embedding layers whereas a new monolingual model requires the full network of parameters. Second, during decoding, the computational complexity is lower for the multi-softmax model as the forward propagation through the shared encoder is performed only once for all languages. However, in the case of monolingual models, each individual model has to perform a separate forward propagation per language. We further attempt to reduce the computational cost, by switching off the decoders for languages whose confidence scores fall below a certain threshold and is discussed in detail in Section. 6.3.

# 5. Experimental details

We conduct experiments using four Indian languages, namely: Indian English (En-IN), Hindi (Hi-IN), Tamil (Ta-IN) and Gujarati (Gu-IN). The train and test data size per language is shown in Table 1. The output vocabulary consists of unique graphemes from the respective language. Graphemes are distinct between languages. We also include symbols with B_ prefix to be able to convert grapheme sequence into word sequence. The final vocabulary for a language includes the respective graphemes, graphemes with B_ prefix and <blank> symbol. The vocabulary size per language is shown in Table 1.

All experiments use 80-dimensional log mel filter bank features, computed every 10 milliseconds (ms). The encoder is fed with 640-dimensional features, obtained by stacking 80-dimensional features from eight frames in the past. The frames are shifted by 30ms. The encoder and prediction networks are 6 and 2 layer LSTM networks respectively. The LSTM layers are unidirectional with 1024 units. All models are trained using PyTorch [30] toolkit.

# 6. Discussion of results

Tables 2, 3 and Table 4 summarize the experimental results for monolingual and different multilingual models. We next discuss these results in detail.

## 6.1. Results for multi-softmax model

Table 2 compares results for monolingual and different multilingual models that assume the input language to be known. The vanilla model performance is much inferior compared to monolingual models, and conditioning it with onehot vector significantly improves the WER as also reported in [4].

The multi-softmax model performs better than the monolingual models on 3 out of 4 Indian languages, with an average WERR reduction of $4.65\%$. It also shows better performance than vanilla and onehot models on all languages as seen from Table 2. The multi-softmax LSP model with separate prediction network for each language improved the performance marginally on Hi-IN and Ta-IN, while had regressions on the rest of the languages. Since, having a prediction network per language does not help much, in future, we plan to experiment with a separate prediction network for a group of languages as done in [28] for S2S models.

Table 3 compares the WER of the same monolingual models (trained from scratch) with monolingual models seeded with onehot and multi-softmax multilingual models presented earlier in Table 2. We refer to these as multilingual finetuned models. The multi-softmax finetuned model is consistently better than the monolingual and onehot finetuned models, with $12.22\%$ WERR reduction over monolingual models. Therefore, we propose to use multi-softmax models as the preferred seed model to train the respective monolingual models.

Table 2: *WER*$[\%]$ *comparison for monolingual and different multilingual models where the input language is known apriori. The average WER is obtained using word weighted WER.*

| | En-IN | Hi-IN | Ta-IN | Gu-IN | Avg. | WERR |
|---|---|---|---|---|---|---|
| Monolingual | 24.67 | 16.2 | 32.92 | 27.33 | 23.97 | 0 |
| Multilingual Vanilla | 35.21 | 22.94 | 48.56 | 28.83 | 32.45 | -35.35 |
| Multilingual Onehot | 25.88 | 18.88 | 41.92 | 27.9 | 26.7 | -11.37 |
| Multi-softmax | 23.35 | 17.2 | 31.18 | 23.86 | 22.86 | **4.65** |
| Multi-softmax LSP | 23.46 | 16.84 | 30.3 | 24.28 | 22.73 | **5.17** |

Table 3: *WER*$[\%]$ *comparison for monolingual models trained from scratch versus monolingual models trained from multilingual models as seed.*

| | En-IN | Hi-IN | Ta-IN | Gu-IN | Avg. | WERR |
|---|---|---|---|---|---|---|
| Monolingual | 24.67 | 16.2 | 32.92 | 27.33 | 23.97 | 0 |
| Onehot finetuning | 23.65 | 16.64 | 31.36 | 26.01 | 23.24 | 3.05 |
| Multi-softmax finetuning | 21.93 | 15 | 28.45 | 22.66 | 21.04 | **12.22** |

## 6.2. Results for multi-softmax with integrated LID

Table 4 depicts results for LID-free multilingual models, namely vanilla and multi-softmax LID model, that do not need to know the input language. The multi-softmax LID model performs significantly better than the vanilla model, with $13.86\%$ WERR reduction on an average. Both vanilla and multi-softmax LID models are susceptible to errors in identifying the input language. To measure the impact of LID error rate, we conduct an oracle experiment where we choose the output from the decoder corresponding to the known input language as shown by multi-softmax LID Oracle results in Table 4. We expected oracle results to be close to multi-softmax results (upper

Table 4: *WER[%] for different multilingual models where the input language is not known.*

| | En-IN | Hi-IN | Ta-IN | Gu-IN | Avg. | WERR |
|---|---|---|---|---|---|---|
| Multilingual Vanilla | 35.21 | 22.94 | 48.56 | 28.83 | 32.45 | 0 |
| Multi-softmax LID | 28.92 | 20.26 | 35.83 | 31.55 | 27.95 | **13.86** |
| Multi-softmax LID Oracle | 25 | 18.71 | 33 | 25.30 | 24.45 | 24.65 |

Table 5: *LID accuracy on Indian languages for vanilla and multi-softmax model.*

| | En-IN | Hi-IN | Ta-IN | Gu-IN | Avg. |
|---|---|---|---|---|---|
| Multilingual Vanilla | 88.7 | 86.2 | 90.4 | 85.5 | 87.7 |
| Multi-softmax LID | 94 | 90.2 | 95.8 | 83.1 | 90.09 |

bound), however, the difference could be due to the LID loss back-propagating through the encoder network. This could be avoided by having a separate LID model, which would however increase the model size.

We also measure the LID accuracy of vanilla and multi-softmax LID model by computing the percentage of utterances with matching scripts between hypothesis and transcription. Since the script is unique for Indian languages, correctly identifying the script implies correct language identification. The LID accuracy of both the models are shown in Table 5. The LID accuracy of multi-softmax LID model is better compared to vanilla model, owing to an integrated LID model.

#### *6.2.1. Model size and decoding cost*

The number of parameters per model is shown in Table 6. The multi-softmax LID model has comparable model size as vanilla with $1.6\%$ more parameters. Comparing the decoding cost, the vanilla model runs a single beam search decoding on a union of vocabulary, while the multi-softmax LID model runs multiple beam-search decoding on a smaller language specific vocabulary. Since multiple beam-search decoding is run in tandem, multi-softmax LID model is expected to have higher computational cost. Note that encoder forward pass is done only once and all decoders use the same. We attempt to reduce the computational cost by employing *early stopping* as discussed next.

### 6.3. Early stopping for multi-softmax LID decoders

The key idea to early stopping is to switch off the decoders, whose LID scores fall below a particular threshold. We first compute the confidence score, $s_l(t)$ for each language $l$ at time $t$, as follows:

$$s_l(t) = \sum_{t'=0}^{t} \log P_{t'}(l) \tag{1}$$

Table 6: *Number of model parameters (in millions) for different models. Due to space constraints, monolingual model is abbreviated as Mono. and multi-softmax as MS*

| | Mono. | Vanilla | Onehot | MS | MS LSP | MS LID |
|---|---|---|---|---|---|---|
| # of parameters | 67.98 | 68.95 | 68.98 | 68.55 | 118.94 | 70.08 |

where $P_{t'}(l)$ is posterior probability obtained from LID model at time $t'$ for language $l$. Let $s_{l_{max}}(t)$ denote the score corresponding to the language $l_{max}$ which has the maximum score at time $t$. A decoder is turned OFF if it satisfies the following two conditions a) $t > \tau$, where $\tau$ is the minimum wait time (number of frames) where all decoders run b) $s_l(t) < s_{l_{max}}(t) - s_{th}$. The minimum wait time is necessary to have reliable estimations from LID model. $\tau$ and $s_{th}$ are tunable parameters. To measure the decoding cost, we define average decoder time $T_{avg}$ as follows:

$$T_{avg} = \sum_{l}^{L} T_l^{off} \tag{2}$$

where, $L$ is the number of languages, $T_l^{off} = \frac{t_l^{off}}{T}$ represents fraction of total frames the decoder $l$ was active. $t_l^{off}$ is the number of frames after which the decoder $l$ was turned off. $T$ is the total number of frames in an utterance.

Table 7: *Trade-off between WER and $T_{avg}$ for different settings of $\tau$ and $s_{th}$. The WER is computed on subset of Hi-IN test set with 3980 utterances.*

| $\tau$ (ms) | $s_{th}$ | WER | $T_{avg}$ | LID accuracy |
|---|---|---|---|---|
| $\infty$ | $\infty$ | 21.84 | 4 | 92.4 |
| 1000 | 0.2 | 21.97 | 3.96 | 92.1 |
| 100 | 0.2 | 22.59 | 2.92 | 91.3 |
| 30 | 0.2 | 25.73 | 1.52 | 86 |
| 30 | 0.5 | **23.14** | **1.72** | 90 |
| 10 | 0.2 | 25.83 | 1.04 | 85.7 |

Trade-off between $T_{avg}$ and WER with different settings of $\tau$ and $s_{th}$ is shown in Table 7. Setting $\tau$ or $s_{th}$ to $\infty$ corresponds to running all decoders for the entire audio frames and hence is same as conventional multi-softmax LID model. Reducing $\tau$ or $s_{th}$, reduces $T_{avg}$ with a marginal increase in the WER. Let us consider the setting of $\tau = 30$ and $s_{th} = 0.5$. Here all decoders run for atleast 30 frames ($900ms$). Post 30 frames some of the decoders are switched off that satisfy the switching off criterion. Switching off some of decoders early, at-times could result in switching off the correct decoder as reflected by the marginal increase in WER and reduced LID accuracy. However, there is a significant reduction in $T_{avg}$ from 4, that is running all 4 decoders, to 1.72, analogous to running only 1.72 decoders. The above framework can be used to arrive at an appropriate setting with acceptable computational cost and WER trade-off. The correspondence between $T_{avg}$ and CPU inference cost can be obtained by computing CPU utilization metrics, with experiments done under identical CPU and memory profile. We reserve that study for future.

## 7. Conclusions

This work presents the multi-softmax RNN-T model, showing consistent improvements over monolingual and prominent multilingual baselines. When finetuned with monolingual data, we also show that the multi-softmax model serves as a good seed model which results in an average WERR reduction of $12.22\%$. Then we proposed the multi-softmax model with an implicit LID detector which makes it LID-free. This model yields $13.86\%$ WERR reduction over the multilingual vanilla model. Finally, we propose early stopping to reduce the decoding cost, resulting in $57\%$ reduction in the $T_{avg}$ with marginal increase in the WER.

# 8. References


[1] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: A survey," *Speech Communication*, vol. 56, pp. 85–100, 2014.

[2] T. Schultz and A. Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, no. 1, pp. 31–51, 2001.

[3] H. Lin, l. Deng, D. Yu, Y. Gong, A. Acero, and C.-H. Lee, "A study on multilingual acoustic modeling for large vocabulary asr," 04 2009, pp. 4333–4336.

[4] 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," 2019.

[5] A. Graves, "Sequence transduction with recurrent neural networks," arXiv:1211.3711, 2012.

[6] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proceedings of ICASSP*, 2019, pp. 6381–6385.

[7] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proceedings of ASRU*, 2019.

[8] J. Li, , R. Zhao, Z. Meng *et al.*, "Developing RNN-T models surpassing high-performance hybrid models with customization capability," in *Proceedings of Interspeech*, 2020.

[9] G. Saon, Z. Tüske, and K. Audhkhasi, "Alignment-length synchronous decoding for RNN transducer," in *Proceedings of ICASSP*, 2020, pp. 7804–7808.

[10] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A new training pipeline for an improved neural transducer," in *Proceedings of Interspeech*, 2020.

[11] V. M. Shetty, M. Sagaya Mary N J, and S. Umesh, "Improving the performance of transformer based low resource speech recognition for Indian languages," in *Proceedings of ICASSP*, 2020, pp. 8279–8283.

[12] S. Punjabi, H. Arsikere, Z. Raeesy, C. Chandak, N. Bhave, A. Bansal, M. Müller, S. Murillo, A. Rastrow, S. Garimella, R. Maas, M. Hans, A. Mouchtaris, and S. Kunzmann, "Streaming end-to-end bilingual asr systems with joint language identification," 2020.

[13] Y. Zhu, P. Haghani, A. Tripathi, B. Ramabhadran, B. Farris, H. Xu, H. Lu, H. Sak, I. Leal, N. Gaur, P. Moreno, and Q. Zhang, "Multilingual speech recognition with self-attention structured parameterization," in *Proceedings of INTERSPEECH*, 2020.

[14] V. Joshi, R. Zhao, R. R. Mehta, K. Kumar, and J. Li, "Transfer Learning Approaches for Streaming End-to-End Speech Recognition System," in *Proceedings of Interspeech 2020*, 2020, pp. 2152–2156.

[15] M. Giollo, D. Gunceler, Y. Liu, and D. Willett, "Bootstrap an end-to-end asr system by multilingual training, transfer learning, text-to-text mapping and synthetic audio," 2020.

[16] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual MLP features for low-resource LVCSR systems," in *Proceedings of ICASSP*, 2012, pp. 4269–4272.

[17] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in dnn-based lvcsr," in *IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 246–251.

[18] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Proceedings of ICASSP*, 2013, pp. 8619–8623.

[19] J. Kunze, L. Kirsch, I. Kurenkov, A. Krug, J. Johannsmeier, and S. Stober, "Transfer learning for speech recognition on a budget," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: ACL, Aug. 2017, pp. 168–177.

[20] Y. Huang, D. Yu, C. Liu, and Y. Gong, "Multi-accent deep neural network acoustic model with accent-specific top layer using the KLD-regularized model adaptation," in *Proceedings of Interspeech*, 2014.

[21] S. Scanzio, P. Laface, L. Fissore, R. Gemello, and F. Mana, "On the Use of a Multilingual Neural Network Front-End," in *Proceedings of Interspeech*, 2008, pp. 2711–2714.

[22] D. Yu and L. Deng, "Efficient and effective algorithms for training single-hidden-layer neural networks," *Pattern Recognition Letters*, January 2012.

[23] J. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *Proceedings of ICASSP*, 2013, pp. 7304–7308.

[24] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *Proceedings of ICASSP*, 2013, pp. 6965–6969.

[25] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Proceedings of Interspeech*, 2014.

[26] A. Waters, M. Bastani, M. G. Elfeky, P. Moreno, and X. Velez, "Towards acoustic model unification across dialects," in *2016 IEEE Workshop on Spoken Language Technology*, 2016.

[27] A. Das, K. Kumar, and J. Wu, "Multi-Dialect Speech Recognition in English Using Attention on Ensemble," in *Proceedings of ICASSP*, 2021.

[28] 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.

[29] B. Li, Y. Zhang, T. Sainath, Y. Wu, and W. Chan, "Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes," in *Proceedings of ICASSP*, 2019, pp. 5621–5625.

[30] A. Paszke, S. Gross, F. Massa *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.