INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Large-Scale Multilingual Speech Recognition with a Streaming End-to-End Model

Anjuli Kannan, Arindrima Datta*, Tara N. Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, Seungji Lee*

Google, Inc.

{anjuli,arindrimadatta,tsainath,weinstein,bhuv}@google.com

Abstract

Multilingual end-to-end (E2E) models have shown great promise in expansion of automatic speech recognition (ASR) coverage of the world's languages. They have shown improvement over monolingual systems, and have simplified training and serving by eliminating language-specific acoustic, pronunciation, and language models. This work presents an E2E multilingual system which is equipped to operate in low-latency interactive applications, as well as handle a key challenge of real world data: the imbalance in training data across languages. Using nine Indic languages, we compare a variety of techniques, and find that a combination of conditioning on a language vector and training language-specific adapter layers produces the best model. The resulting E2E multilingual model achieves a lower word error rate (WER) than both monolingual E2E models (eight of nine languages) and monolingual conventional systems (all nine languages).

Index Terms: speech recognition, multilingual, RNN-T, residual adapter

1. Introduction

Automatic speech recognition (ASR) systems that can transcribe speech in multiple languages, known as multilingual models, have gained popularity as an effective way to expand ASR coverage of the world's languages. Through shared learning of model elements across languages, they have been shown to outperform monolingual systems, particularly for those languages with less data. Moreover, they significantly simplify infrastructure by supporting n languages with a single speech model rather than n individual models. Successful strategies for building multilingual acoustic models (AMs) include stacked bottleneck features [1–4], shared hidden layers [5, 6], knowledge distillation [7], and multitask learning [8]. Building multilingual language models (LMs) has also been attempted recently (e.g. [9]). However, in most state-of-the-art multilingual systems, only the AM is multilingual; separate language-specific LMs (and often lexicons) are still required.

More recently, end-to-end (E2E) multilingual systems have gained traction as a way to further simplify the training and serving of such models. These models replace the acoustic, pronunciation, and language models of n different languages with a *single* model while continuing to show improved performance over monolingual E2E systems [10–13]. Even as these E2E systems have shown promising results, it has not been conclusively demonstrated that they can be competitive with state-of-the-art conventional models, nor that they can do so while still operating within the real-time constraints of interactive applications such as a speech-enabled assistant.

Our work is motivated by the need for E2E multilingual systems that (1) meet the latency constraints of interactive applications, (2) handle the challenges inherent in large-scale, real-world data, and (3) are competitive with state-of-the-art conventional models while still operating within the same training and serving constraints.

First, we present a streaming E2E multilingual system using the Recurrent Neural Network Transducer (RNN-T) [14]. As [15] has shown, the architecture we employ adheres to the latency constraints required for interactive applications. In constrast, prior E2E multilingual work has been limited to attention-based models that do not admit a straightforward streaming implementation [10–13].

Next, we address the challenges of training such a model with large-scale real world data. Given the dramatic skew in the distribution of speakers across the world's languages, it is typical to have varying amounts of transcribed data available for different languages. As a result, a multilingual model will be more influenced by languages which are over-represented in the training set. Working with a corpus of nine Indian languages comprised of 37K hours of data, we compare several techniques to address this issue: conditioning on a language vector, adjusting language sampling ratios, and language-specific adapter modules. Our experiments demonstrate that the combination of a language vector and adapter modules yields the best multilingual E2E system. While previous works have investigated various aspects of data sampling [16, 17], as well as architectures that include a language vector [11, 18, 19], this is the first study to apply adapter modules [20] to speech recognition.

Lastly, when we combine the above elements, we show that the resulting system surpasses not only monolingual E2E models, but also monolingual conventional systems built with state-of-the-art AMs, lexica, and LMs. The E2E system consistently achieves at least a 10% relative reduction in WER on each of the nine languages, when compared with the monolingual conventional systems. This is demonstrated while still using comparable training and serving resources.

2. Streaming E2E multilingual model

A key requirement for an interactive application is to support streaming ASR. Thus our experiments are conducted on RNN-T [14, 21], a streaming E2E model which has been shown to offer appropriate user latency required for applications such as a speech-enabled assistant [15].

RNN-T consists of an encoder network, a prediction network, and a joint network. The encoder, which is analogous to the AM in a traditional ASR system, is a recurrent network composed of stacked long short-term memory (LSTM) layers. It reads a sequence of d-dimensional feature vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathbb{R}^d$, and produces at

*equal contribution

http://dx.doi.org/10.21437/Interspeech.2019-2858

each timestep a higher-order feature representation, denoted $\mathbf{h}_1^{\text{enc}}, \cdots, \mathbf{h}_T^{\text{enc}}$. Similarly, the prediction network is also an LSTM network, which, like an LM, processes the sequence of non-blank symbols output so far, $y_0, \ldots, y_{u_{i-1}}$ into a dense representation $\mathbf{h}_{u_i}^{\text{dec}}$.

Finally the representations produced by the encoder and prediction networks are combined by the joint network. The joint network then predicts $P(y_i|\mathbf{x}_1, \cdots, \mathbf{x}_{t_i}, y_0, \ldots, y_{u_{i-1}})$, a distribution over the next output symbol.

In this way, RNN-T does not make a conditional independence assumption: the prediction of each symbol is conditioned not only on the acoustics but also on the sequence of labels output so far. However, RNN-T does assume an output symbol is independent of future acoustic frames. This assumption allows us to employ RNN-T in a streaming fashion.

3. Imbalanced multilingual data

This section describes three strategies we investigate for handling data imbalance in the multilingual model. Imbalance is a natural consequence of the varied distribution of speakers across the world's languages. Languages with more speakers tend to produce transcribed data more easily. In conventional ASR systems, only the AM is trained on the transcribed speech data, but in an E2E multilingual model all the components are trained on it. As a result, the latter may be more sensitive to data imbalance. In this section we explore two avenues to address data imbalance: (1) data sampling and (2) extensions to the model architecture.

3.1. Data sampling

Imbalanced data typically leads to having a model perform better on languages with larger data. For instance, suppose a multilingual model is trained on k languages $L_0, \cdots, L_{k-1}$, where L_i has n_i training examples, and $N = \sum_{i=0}^{k} n_i$. At each step of training, we assemble a batch of examples by sampling from the N total examples. Assuming the training examples have been pooled across languages and shuffled, we expect the sampling ratio of language L_i within the batch to be $s(i) = \frac{n_i}{N}$. This means that the model gets updated with $\frac{n_i}{n_j}$ times more gradients generated by language L_i than by language L_j for any i, j where $n_i > n_j$.

One approach to dealing with data imbalance is to upsample data from under-represented languages so that the distribution across different languages is more even. In the extreme case, we can sample each language uniformly, such that $s(i) = \frac{1}{k}$ for all i. More generally, we can let

$$s(i) = \frac{n_i + \alpha * (n^* - n_i)}{\sum_i [n_i + \alpha * (n^* - n_i)]} \tag{1}$$

where n^* is the maximum number of examples for any language and α is a tunable parameter. If $\alpha = 0$ we are sampling at natural frequencies, while if $\alpha = 1$ we are sampling uniformly.

Similar strategies are described in [17, 22], but are applied to the sampling of phonemes rather than languages. Moreover, they are focused on the AM of a conventional model, whereas we investigate an E2E setting. [16] has also considered scalars within the loss function to increase the contribution of under-represented languages, when learning multilingual bottleneck features.

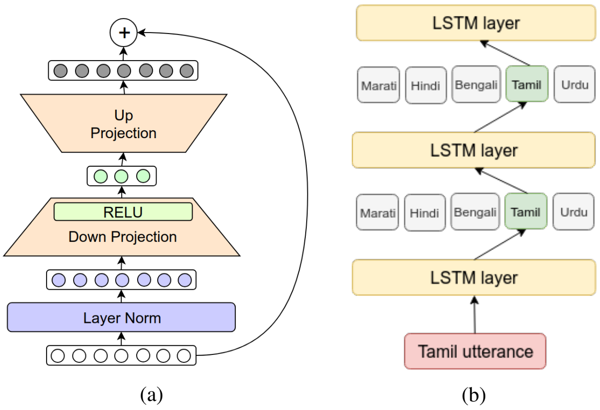

Figure 1: *(a) An adapter module consists of layer normalization, down-projection, non-linearity, and up-projection. (b) Adapter modules in the context of the encoder. For a Tamil utterance, only the Tamil adapters are applied to each activation.*

3.2. Conditioning on language vector

Next we explore the architectural extension of feeding a vector representing the language. The intuition is that a universal model can use a language vector to become an "expert" on each individual language instead of learning a representation tailored to languages with more data. At inference time, we assume the language is either specfied in the user's preferences, or determined automatically from a language identification system.

Various methods of using a language vector have been previously described and directly compared in non-streaming E2E multilingual [11] and multidialect [18] models. The language itself can be represented in several different ways (as a one-hot vector, as an embedding vector, or as a combination of clusters learned through cluster adaptive training (CAT) [23]), but prior work [18, 19] has shown that the simple approach of a one-hot vector performs as well as and sometimes better than the more complex methods. Additionally, given such a language vector, there are many different points in the model where it can be inserted, but our experiments have confirmed earlier observations [11, 18] that simply concatenating it to the input features is sufficient. Thus, for all language vector experiments we represent the language as a one-hot vector, and concatenate it to the input features of the encoder network.

While previous works have investigated the use of the language vector in multilingual modeling, these have done so in the context of either the Listen, Attend and Spell [24] architecture [11, 18] which is a non-streaming E2E model, or conventional ASR systems [19]. Here we explore using a language vector with a streaming RNN-T model and as well as its ability to deal with data imbalance.

3.3. Adapter modules

A second architecture extension that we investigate to handle data imbalance is adapter modules. This modeling technique has seen success for domain and task adaptation in computer vision [20], natural language processing [25] and machine translation [26]. Here we extend it to multilingual speech recognition.

Adapter modules are effectively domain-specific (language-specific in our case) adjustments to the activations coming out of each layer. They aim to capture the quality benefits of fine-tuning a global model on each language while maintaining the parameter efficiency of having a single global

Table 1: *Number of utterances in train and test sets*

Language	Train	Test	Language	Train	Test
Hindi	16M	6.3K	Tamil	1.8M	5.5K
Marathi	4.1M	6.1K	Malayalam	1.5M	9.2K
Bengali	3.9M	3.6K	Kannada	1.2M	1.1K
Telegu	2.4M	2.7K	Urdu	443K	511
Gujarati	2.2M	7.5K	**Total**	33M	43K

model. This is accomplished by a two-stage training process. In the first stage, a global model (the RNN-T described in Section 2) is trained on the union of data from all languages. In a second adaptation stage, we freeze all model parameters, and introduce adapter modules after every layer of the encoder. Importantly, each adapter module contains separate parameters for each language.

More details are shown in Figure 1. Figure 1a shows the exact computation of each module: following [25, 26], each activation is projected down to a smaller dimensionality, passed through a non-linearity, then projected back up to the original size. This result is then added to the original activation before being passed to the next layer. Figure 1b shows what happens at inference time: a Tamil utterance comes in, and thus the Tamil-specific adapter weights are applied after each layer.

The result of this training can be viewed as a single multilingual model with a small number of language-specific parameters (typically less than 10% of the original model size). This allows for efficient parameter sharing across languages, as well as per-language specialization.

A conceptually similar technique called learning hidden unit contributions (LHUC) [27] has been successfuly applied to multilingual ASR by [28]. The main difference is that LHUC modulates the amplitude of the different hidden units, whereas adapter modules are entirely separate layers. [26] has shown adapter modules to be more effective than LHUC for large-scale sequence-to-sequence models, so we limit our study to this method.

4. Experimental details

4.1. Data

Our training data and test data consist of anonymized, human-transcribed utterances, representative of Google's traffic, and spanning nine Indian languages. Train and test set sizes vary due to availability of transcribed data and are shown in Table 1. Training data are further augmented by corrupting clean utterances using a room simulator [29].

Each of the nine languages is written in a different script, except Hindi and Marathi, which both use the Devanagari writing system. In addition, each language's transcriptions have some Latin alphabet mixed in, as transcribers may use Latin alphabet for loaner words and some proper nouns. The frequency of Latin alphabet is similar to what is reported in [30].

Since transcripts can contain a mixture of Latin and native scripts, the metric we report is *transliteration-optimized WER* [30]. Put simply, this means that if the model correctly decodes a word in Latin alphabet but the reference is in the native script (or vice versa), this is not considered an error, because the output can be transliterated to one or the other for rendering.

4.2. Model architecture

All experiments use 80-dimensional log-mel features, computed with a 25ms window and shifted every 10ms. These features are stacked with 7 frames to the left and downsampled to 30ms frame rate.

The encoder network consists of eight 2,048-dimensional LSTM layers, each followed by a 640-dimensional projection layer. The prediction network has two 2,048-dimensional LSTM layers, each of which is also followed by 640-dimensional projection layer. Finally, the joint network also has 640 hidden units. The softmax layer is composed of a unified grapheme set from all languages (988 graphemes in total), that was generated using all unique graphemes in the training data. Adapter modules use a 256-dimensional bottlneck after each of the eight encoder layers.

All RNN-T models are trained in Lingvo [31] on 8×8 Tensor Processing Units [32] slices with a batch size of 4,096.

4.3. Baselines

We report two baselines for comparison. First, RNN-T monolingual baselines are trained on each of the individual languages using the same architecture as the multilingual model described earlier, with the addition of L2 regularization to reduce overfitting. Additionally, monolingual conventional models are trained as follows. The AMs consist of five 768-dimensional LSTM layers and a softmax over over context dependent phone states. They are trained using connectionist temporal classification (CTC) [33], followed by state-level minimum Bayes risk [34, 35]. The AM outputs are then used with standard FST-based beam-search decoders with language-specific lexicons and 5-gram language models for decoding.

5. Results

5.1. Conditioning on language vector

We begin by investigating the impact of conditioning the RNN-T encoder on a language vector. As shown in Table 3 (comparing `A0` and `A1`), providing this information to the model is critical to good performance, particularly for languages with less data such as Urdu, Kannada, and Malayalam, which all see more than 50% relative WER reduction.

This result is somewhat surprising, given previous works have shown a much smaller impact of providing language information [11, 18]. However, our error analysis reveals that this is largely a result of the significant overlap in vocabulary between different languages in our data. Because our data comes from South Asia, we see that the different languages' corpora contain many of the same proper nouns and English loaner words. Moreover, most utterances in our test set are short and contain few glue words which could help with language disambiguation. Queries such as "arundhati movie", "train bangalore", and "doctor rajkumar" could appear in any of the languages' training or test sets.

As a result, the model `A0` often defaults to the script of the dominant language in the training data, Hindi. For instance, about 75% utterances in the Bengali test set consist entirely of proper nouns or English words, and 44% of these are decoded by model `A0` in an incorrect script but are otherwise correct. On the other hand, model `A1` can use the language vector to ensure the correct script is output: its Bengali WER is nearly 50% lower than that of `A0`.

Given the extent to which these languages' vocabularies overlap, it is thus of little surprise that the language vector is such a critical piece of information for this model. What is more notable is how this vector also helps to address the issue of data imbalance, as we demonstrate in the the following section.

Table 2: *WER of the multilingual E2E Model using various techniques to address data imbalance.*

Exp	Model	Hindi	Marathi	Beng.	Telugu	Gujarati	Tamil	Mala.	Kann.	Urdu	Avg
A0	Multilingual RNN-T	18.5	26.2	43.9	49.3	55.3	40.1	69.7	60.8	70.1	48.2
A1	A0 + language vector	16.0	17.6	22.8	23.5	24.3	22.2	46.6	20.5	17.3	22.8
A2	A0 + sampling	22.3	29.8	41.1	45.9	43.9	37.7	64.6	55.4	48.1	43.2
A3	A1 + sampling @ 60K	18.7	18.8	24.0	24.6	24.3	25.0	47.8	21.4	17.7	24.7
A4	A1 + sampling	16.2	17.8	24.1	25.1	24.2	22.9	48.9	24.6	20.4	24.9
A5	A1 + adapters	**15.9**	**17.1**	**21.5**	**23.2**	**24.0**	**21.6**	**45.8**	**18.7**	**16.0**	**22.6**

Table 3: *WER from the best multilingual E2E model and monolingual baselines.*

Exp	Model	Hindi	Marathi	Beng.	Telugu	Gujarati	Tamil	Mala.	Kann.	Urdu	Avg
B0	Monolingual CTC	18.6	19.8	26.8	25.1	29.6	24.5	47.1	30.0	29.5	28.0
B1	Monolingual RNN-T	16.1	21.3	**18.2**	25.5	26.4	27.6	54.4	29.5	27.4	27.4
B2	Multilingual RNN-T	**15.9**	**17.1**	21.5	**23.2**	**24.0**	**21.6**	**45.8**	**18.7**	**16.0**	**22.6**

Languages are listed in descending order of training data amount.

5.2. Data sampling

Next we compare models with various sampling strategies. First, we increase α from 0 to 0.25 on model `A0`, which means we are upsampling smaller languages. Expectedly, we see that the WER on all but the two largest languages (Hindi and Marathi) decreases (compare `A2` to `A0`). We effectively change the model's prior on what language to output, so that when it comes across an ambiguous utterance, it may be less likely to default to the dominant language. However, this does result in a 10-20% relative regression on the two largest languages.

When we repeat this comparison on the model with the language vector (`A1`), we find that all benefit of upsampling has been eliminated (compare `A4` and `A1`). While the upsampled languages do reach their lowest WER earlier in training, it never drops below their WER in `A1`. Moreover, the upsampled languages begin to overfit while larger languages keep improving.

This phenomenon is demonstrated by comparing `A3` and `A4` against `A1`. `A3` shows this model evaluated mid-way through training (approximately 60K steps) when small languages like Kannada and Urdu have hit their lowest WER. At this stage, we see that large languages (Hindi, Marathi) are still underfitting the data compared with `A1`. Yet by the time they have fully fit the data (`A4`) the small languages have heavily overfit. This phenomenon is only exacerbated by increasing α.

We hypothesize that the model is able to use the language vector not only to disambiguate the language (as we discussed in the previous section) but also to learn separate features for separate languages, as needed. Thus, upsampling a small language like Kannada does not help the model to learn a better representation of Kannada speech, as the model is already allotting the necessary capacity.

5.3. Adapters

Lastly, we investigate adapter modules as an architectural modification to address imbalance between languages. Comparing `A5` to `A1` we see that the adapters provide a small additional reduction in WER on all languages, with the largest gains on Kannada (9% relative), Urdu (8% relative), and Bengali (6% relative). We hypothesize that the encoder of the global model is mostly shaped by the dominant languages, Hindi and Marathi, whereas Kannada, Urdu, and Bengali may have distinct acoustic features that can be captured by small per-layer adjustments.

Adding language-specific adapter modules allows the model to specialize in each language in the same way that fine-tuning the whole model would, but in a much more parameter-efficient way: the capacity added for each language (2.5M parameters) is only about two percent of the original model size (120M parameters). We also point out that adapters do not need to be employed when they are not helpful. For example, in practice we might choose to only keep adapters for Kannada, Urdu, and Bengali, where they are most effective, while setting all other adapters to identity operations (i.e., setting the weights to zeroes). This model would only be 6% larger than the original model while showing significant gains on those three languages.

5.4. Comparison with Baseline Models

Combining the above results, we compare our best multilingual RNN-T model against both monolingual RNN-T baselines and monolingual conventional baselines. We observe that the multilingual model has a lower WER than the monolingual RNN-T on eight of the nine languages. Similarly, it achieves a lower WER than the monolingual conventional models on all nine languages, consistently by about 10% relative, with larger relative gains (34% on Kannada and 25% on Urdu) on the lower resource languages. Future work will be needed to understand the role that shared vocabulary and phonetic relatedness plays in this result.

In addition to a lower WER, the multilingual RNN-T model offers the benefit of replacing nine separate recognizers (each of which typically consists of acoustic, pronunciation, and language models) with a single, compact recognizer. To our knowledge, this is the first work to demonstrate that a multilingual E2E model which is suitable for streaming applications can outperform monolingual conventional models.

6. Conclusions

In this work, we extended prior work on E2E multilingual models to address two issues that arise in large-scale practical applications: (1) the need for streaming ASR and (2) the challenge of imbalanced training data. We presented a system that addresses both issues, as well as a comparison of techniques to address the second. Using nine Indian languages, we showed that our best system, built with RNN-T model and adapter modules, significantly outperforms both the monolingual RNN-T models, and the state-of-the-art monolingual conventional recognizers.

7. Acknowledgements

The authors would like to thank Ruoming Pang, Sergey Kishchenko, Pedro Moreno, Mikaela Grace, Bo Li, and Meysam Bastani for helpful discussions.

8. References

[1] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual mlp features for low resource lvcsr systems," in *ICASSP*, 2012.

[2] Z. Tuske, J. Pinto, D. Willett, and R. Schluter, "Investigation on cross and multilingual mlp features under matched and mismatched acoustical conditions," in *ICASSP*, 2013.

[3] J. Cui, B. Kingsbury, B. Ramabhadran, and et al, "Multilingual representations for low resource speech recognition and keyword search," in *ASRU*, 2015.

[4] T. Sercu, G. Saon, J. Cui, B. Ramabhadran, B. Kingsbury, and A. Sethy, "Network architectures for multilingual speech representation learning," in *ICASSP*, 2017.

[5] A. Ghoshal, P. Swietojanski, and S. Renals, "Multilingual training of deep neural networks," in *ICASSP*, 2013.

[6] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *ICASSP*, 2013.

[7] J. Cui, B. Kingsbury, B. Ramabhadran, and et al, "Knowledge distillation across ensembles of multilingual models for low-resource languages," in *ASRU*, 2017.

[8] D. Chen and B. Mak, "Multitask learning of deep neural networks for low-resource speech recognition," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2015.

[9] C. Fugen, S. Stuker, H. Soltau, F. Metze, and T. Schultz, "Efficient handling of multilingual language models," in *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721)*. IEEE, 2003.

[10] S. Watanabe, T. Hori, and J. Hershey, "Language independent end-to-end architecture for joint language identification and speech recognition," in *ASRU*, 2017.

[11] S. Toshniwal, T. Sainath, R. Weiss, B. Li, P. Moreno, E. Weinstein, and K. Rao, "Multilingual speech recognition with a single end-to-end model," in *ICASSP*, 2018.

[12] M. Karafiát, M. Baskar, S. Watanabe, T. Hori, and M. Wiesner, "Analysis of multilingual sequence-to-sequence speech recognition systems," in *arXiv:1811.03451*, 2018.

[13] J. Cho, M. Baskar, R. Li, M. Wiesner, S. Mallidi, N. Yalta, and M. Karafiat, "Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling," in *arXiv:1810.03459*, 2018.

[14] A. Graves, "Sequence transduction with recurrent neural networks," in *ICASSP*, 2012.

[15] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP*, 2019.

[16] T. Alumae, S. Tsakalidis, and R. Schwartz, "Improved multilingual training of stacked neural network acoustic models for low resource languages," in *Interspeech*, 2016.

[17] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep convolutional neural networks for multilingual lvcsr," in *ICASSP*, 2016.

[18] B. Li, T. Sainath, K. Sim, M. Bacchiani, E. Weinstein, P. Nguyen, Z. Chen, Y. Wu, and K. Rao, "Multi-dialect speech recognition with a single sequence-to-sequence model," in *ICASSP*, 2018.

[19] M. Grace, M. Bastani, and E. Weinstein, "Occam's adaptation: A comparison of interpolation of bases adaptation methods for multi-dialect acoustic modeling with lstms." in *SLT*, 2018.

[20] S. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *NIPS*, 2017.

[21] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *ASRU*, 2017.

[22] I. Garcia-Moral, R. Solera-Urena, C. Palaez-Moreno, and F. Diaz-de Maria, "Data balancing for efficient training of hybrid ann/hmm automatic speech recognition systems," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2011.

[23] T. Tan, Y. Qian, and K. Yu, "Cluster adaptive training for deep neural network based acoustic model," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2016.

[24] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *CoRR*, vol. abs/1508.01211, 2015.

[25] N. Houlsby, A. Giurgiu, S. Jastrze¸bski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp." in *ICML*, 2019.

[26] Anonymous authors., "Simple, scalable adaptation for neural machine translation," in *Under review.*, 2019.

[27] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2014.

[28] S. Tong, P. N. Garner, and H. Bourlard, "Multilingual training and cross-lingual adaptation on ctc-based acoustic model," *arXiv preprint arXiv:1711.10025*, 2017.

[29] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, "Generated of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. Interspeech*, 2017.

[30] J. Emond, B. Ramabhadran, B. Roark, P. Moreno, and M. Ma, "Transliteration based approaches to improve code-switched speech recognition performance," in *SLT*, 2018.

[31] J. Shen, P. Nguyen, Y. Wu, Z. Chen *et al.*, "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," 2019.

[32] N. Jouppi, C. Young, and N. Patil, "In-datacenter performance analysis of a tensor processing unit," in *CoRR abs/1704.04760*, 2017.

[33] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML.* ACM, 2006, pp. 369–376.

[34] B. Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proc. ICASSP*. IEEE, 2009, pp. 3761–3764.

[35] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in *Interspeech*, 2014.