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

Modeling and training strategies for language recognition systems

Raphaël Duroselle, Md Sahidullah, Denis Jouvet, Irina Illina

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
firstname.name@loria.fr

Abstract

Automatic speech recognition is complementary to language recognition. The language recognition systems exploit this complementarity by using frame-level bottleneck features extracted from neural networks trained with a phone recognition task. Recent methods apply frame-level bottleneck features extracted from an end-to-end sequence-to-sequence speech recognition model. In this work, we study an integrated approach of the training of the speech recognition feature extractor and language recognition modules. We show that for both classical phone recognition and end-to-end sequence-to-sequence features, sequential training of the two modules is not the optimal strategy. The feature extractor can be improved by supervision with the language identification loss, either in a fine-tuning step or in a multi-task training framework. Besides, we notice that end-to-end sequence-to-sequence bottleneck features are on par with classical phone recognition bottleneck features without requiring a forced alignment of the signal with target tokens. However, for sequence-to-sequence, the architecture of the model seems to play an important role; the Conformer architectures leads to much better results than the conventional stacked DNNs approach; and can even be trained directly with the LID module in an end-to-end approach.

Index Terms: language recognition, bottleneck features, end-to-end speech recognition, multi-task training

1. Introduction

Language recognition is the task of determining the language in a spoken speech utterance [1]. Systems that address this task handle the speech utterances in several steps: they first extract features at the frame level, then they process and aggregate these features at the utterance level, and finally, make a decision.

Standard frame-level features for language recognition are spectral features: filterbanks [2], MFCC [3], SDC-MFCC [4] and PLP features [5, 6]. These features can be refined by using a neural network to select relevant information through an auxiliary task, for instance auto reconstruction [7], prosody information prediction [8], phonotactic information encoding [9] or speech attribute prediction [10]. The most successful approach has been the use of a speech recognition task [11].

The complementarity of automatic speech recognition (*ASR*) task to language identification (*LID*) task is most likely due to the fact that they heavily depend on phonetic information [12]. If both tasks use the same information, relevant features could be more easily discovered with a speech recognition loss which uses a sequence of labels, because providing just the label of the language spoken in the utterance is too coarse for supervision [13, 14].

The most commonly used application of this complementarity is frame-level bottleneck features [14, 15]. This kind of bottleneck features are used in state-of-the-art language recognition systems [11, 16]. An *ASR* system is trained and used to perform forced alignment between speech utterances and frame-level tokens: morphemes [17] or tied phone states [11, 18]. Then a deep neural network (multilayer perceptron [17] or time delay neural network [18]) is trained to predict the token for each frame (with some temporal context). Finally, frame-level embeddings are extracted from a hidden layer of the network. Those embeddings are called *bottleneck features* and they are used for the *LID* task. When several neural networks are trained ‘on cascade‘ to predict the phone labels, they have been called *stacked bottleneck features*. Training of a language recognition system that uses bottleneck features [11] requires a two-step strategy. First, the bottleneck feature extractor is trained with an *ASR* task. Then the language recognition system is trained using those features.

However, this two-step training strategy is far from being optimal. The work in [19] has empirically shown that the best models in terms of phone state accuracy are not the most relevant feature extractors for language recognition. It even appears that the best *ASR* performance can be achieved with phone representations independent of the language, which paves the way to language adversarial *ASR* [20]. So far, no method has been investigated to select the best compromise between *LID* and *ASR* tasks during the training of a language recognition model. Our work aims to study the optimal use of the *ASR* task to estimate parameters of a language recognition model.

A straight-forward solution is to include both models into an end-to-end (E2E) architecture. According to [13], E2E training with the language identification loss is not a successful approach. Also, fine-tuning the *ASR* encoder by backpropagating the *LID* loss into the feature extractor does not improve the language recognition performance [13]. They concluded that it is the phonetic information that helps and not the deeper architecture. Another approach is multi-task training of the whole model with both *ASR* and *LID* losses. It has been successfully implemented for one [21, 22] and two [13] languages for the *ASR* task, without proposing a principled method to chose the respective weights of both tasks during training.

Recently, the paradigm of bottleneck features training for language recognition has been evolving. The traditional training of bottleneck feature extractors relies on a frame-level forced alignement between spectral features and phone labels [11]. Consequently, the quality of the bottleneck features depends on the performance of the speech recognition model, which makes inclusion of new languages very expensive. Conversely, several end-to-end automatic speech recognition architectures have been introduced recently [23, 24]. They can be trained with a sequence-to-sequence loss, like connectionist temporal classification (CTC) loss [25] and used to extract frame-level embeddings from a hidden layer of the network. With this approach, state-of-the-art language recognition performance has been achieved without defining a frame alignment of phone labels, using only one language for the *ASR* task [7]. Moreover, for Chinese dialect recognition, the use of the phone forced alignment performed by the acoustic model trained with CTC loss does not improve language recognition performance over the direct use of the sequence-to-sequence bottleneck features [26]. Multi-task training of a joint *ASR* and *LID* model has been successfuly performed for English and Hindi corpora [27].

Other approaches to joint *ASR* and *LID* training have been proposed. The works in [27, 28] include language labels as additional target tokens of the *ASR* model, allowing easy language diarization. Also, [29] introduced a unified system where the *ASR* model uses x-vectors produced by the *LID* module and the *LID* model improves its prediction using *ASR* confidence scores. In this paper, we do not compare to these systems, but rather focus on the problem of optimally using the *ASR* task, as an auxiliary task when training a model to yield frame-level features for language recognition.

In this work, we compare different choices of architectures and of training strategies for a language recognition system constituted of a feature extractor and a language identification module. First, we show that end-to-end sequence-to-sequence multilingual features trained with the CTC loss are on par with state-of-the-art traditional multilingual bottleneck features trained with phone state prediction [11]. Then, we show that both traditional bottleneck features (phone prediction) and sequence-to-sequence bottleneck features can be greatly improved by using the language identification loss during training of the feature extractor.

2. Joint speech and language recognition system

The language recognition system discussed in this work is composed of a frame-level feature extractor and an utterance-level language predictor. We propose to consider it as a whole end-to-end architecture. This point of view encompasses state-of-the-art language recognition systems [16] and joint speech and language recognition architectures like the phonetic temporal neural model of [13]. It allows exploring different system training strategies combining language identification and speech recognition objective.

2.1. Framework of the system

A language recognition system operates over a sequence X of varying length T of frame-level features:

$$X = \{x_1, x_2, \dots x_T\} \tag{1}$$

The task consists in predicting a language recognition label y among a fixed set $\mathcal{Y}$. During training of the system, auxiliary speech recognition information can be provided as a target sequence Z of words, phone states or characters, of length L:

$$Z = \{z_1, z_2, \dots z_L\} \tag{2}$$

A joint speech and language recognition architecture is depicted on Figure 1. It is composed of three modules:

- a frame-level *feature extractor* (FE) that takes as input the sequence X and produces a sequence F of frame-level embeddings of length $N \leq T$:

$$F = \{f_1, f_2, \dots f_N\} \tag{3}$$

- an *automatic speech recognition* (ASR) decoder which takes as input the sequence F and outputs a sequence of speech recognition scores $\hat{Z}$

$$\hat{Z} = \{\hat{z}_1, \hat{z}_2, \dots \hat{z}_L\} \tag{4}$$

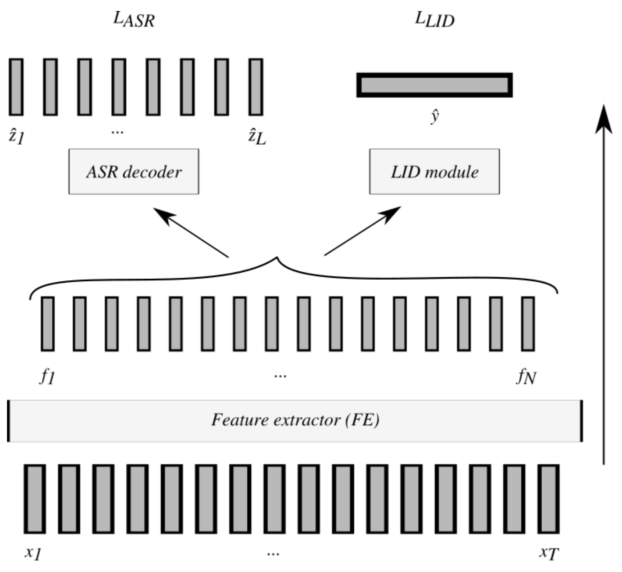

Figure 1: *Schematic diagram of the language recognition system. The input frame-level features x_t are processed by the feature extractor and the language identification module to produce language identification scores $\hat{y}$. The ASR decoder is used for training with the auxiliary speech recognition task.*

- a *language identification* (LID) module which takes as input the sequence F and outputs a vector of language identification scores $\hat{y}$

Parameters of the three modules can be trained with two different loss functions:

- a language identification loss $L_{LID}(\hat{y}, y)$
- an auxiliary automatic speech recognition loss $L_{ASR}(\hat{Z}, Z)$

2.2. Training strategies

Once both the feature extractor and language identification modules have been encompassed into the same model, several training strategies become possible:

- *E2E LID*: the auxiliary speech recognition information is discarded, and FE and LID are trained together to minimize L_{LID}
- *2-step* training: first, FE and ASR are trained to minimize the automatic speech recognition loss L_{ASR}, then parameters of FE are freezed and parameters of LID are trained to minimize the language identification loss L_{LID}
- *2-step then E2E LID*: *2-step* training is performed, then FE and LID are fine-tuned with the language identification loss
- *multi-task* training: all modules are trained together [27]. LID is trained to minimize L_{LID}, and ASR to minimize L_{ASR}. FE is trained with a multi-task training objective (with weight $\lambda \in [0, 1]$):

$$L = \lambda L_{LID} + (1 - \lambda) L_{ASR} \tag{5}$$

2.3. Relation with the classical bottleneck features

Within this framework, the classical bottleneck features approach for language recognition [11] corresponds to the *2-step* training method with the following choices:

- target speech recognition labels Z are phone state labels and a label is provided for each input frame (i.e., $L = T$)
- the *ASR* loss L_{ASR} is a phone state classification loss for each frame.
- the feature extractor applies identical processing with a DNN for each input frame x_t (generally with its context), consequently there is an output frame-level embedding for each input frame (i.e. $N = T$)
- the *ASR* decoder applies identical processing with a DNN for each frame-level embedding f_n

3. Experimental setup

In the following, we compare the different training strategies of a language recognition system, with the same corpora and architectures. We also compare classical bottleneck features [11] with end-to-end sequence-to-sequence bottleneck features.

3.1. Model architectures

We use two architectures for the language identification module:

- the vanilla TDNN architecture described in [16].
- a ResNet architecture with 1D-convolutional layers and a statistical pooling layer identical to the one used in [16]. The model is composed of a serie of four blocks, constituted of respectively 3, 4, 6, and 3 residual layers and having a growing number of channels: 16, 32, 64, and 128.

For the feature extractor and *ASR* decoder, we investigate two architectural choices. Both produce frame-level embeddings of dimension 80:

- stacked DNNs of the mutlilingual bottleneck feature extractor described in [11]. For the *2-step* training strategy, we use the pretrained models (*BUT/Phonexia bottleneck feature extractor*) provided by the authors of [30], which take as inputs mel-filterbank features.
- Conformer [23], a sequence-to-sequence speech recognition model, with one decoder for each target language. We use a model with 2 Conformer blocks with 4 self-attention heads, 80 channels and a kernel of size 17. The initial convolutional layer downsamples the input sequence by a factor 4 (i.e. $N = \frac{T}{4}$). We train this model with mel-filterbank input features. We optimized this architecture for feature extraction during the *Oriental Language Recognition challenge 2020* [31].

3.2. Loss functions

In all experiments, the language identification model is trained with the traditional cross-entropy loss as L_{LID}.

For L_{ASR}:

- the stacked DNNs of [11] have been trained with the cross-entropy as a frame-level classification loss function. The target labels were 3096 phoneme states.
- the Conformer is trained with the CTC loss. The target sequence Z is defined by a sentence piece model [32] trained with 1000 tokens for each target language.

3.3. Corpora

The language recognition performance is evaluated on the corpus NIST LRE 2007 [33]. We perform evaluation for a closed-set identification task with the following 14 languages: Arabic, Bengali, Chinese, English, Hindustani, Spanish, Farsi, German, Japanese, Korean, Russian, Tamil, Thai, Vietnamese. As a training set, we use a subset of the corpora NIST LRE 2003, 2005, 2007 train, 2009 (only recordings from telephone channels), Callfriend, and NIST SRE 2008. This training set is the same as in [2, 7].

We use Babel speech corpus [34] for the *ASR* task. Two different sets of languages are used for experiments with the two feature extractor architectures. The *BUT/Phonexia* pretrained models [11] have been trained with the following 17 languages: Cantonese, Assamese, Bengali, Pashto, Turkish, Tagalog, Vietnamese, Haitian Creole, Lao, Tamil, Zulu, Kurmanji (Kurdish), Tok Pisin, Cebuano, Kazakh, Telugu, Lithuanian. For our experiments with the Conformer architecture we only use 14 of these languages. We do not use Cantonese, Tagalog and Cebuano.

3.4. Training setup

Models are trained with the Adam optimizer [35], with an initial learning rate of 10^{-3}, that is halved every time the validation loss does not improve for 3 epochs. Training is performed for 20 epochs for each phase of training, except for training of the *ASR* model in the *2-step* strategy where we use 100 epochs. Stochastic weight averaging [36] is used: the final model is the average of the 20 models of each epoch. The only data augmentation technique is specAugment [37]. Cepstral mean and variance normalization [38] is applied to the input features.

Language recognition models are trained with balanced minibatches of size 128. Chunks of three seconds are used. For each experiment, at the end of training, two systems are fine-tuned for longer durations with chunks respectively in ranges of 7-13s and 20-40s. These systems are used for evaluation on test sets of respectively 10s and 30s. For each of the three systems (initial 3s, 10s and 30s), calibration parameters are learned on a held-out validation set extracted from the training set and with the same distribution of durations.

4. Results and discussion

4.1. Performance

For each language recognition and feature extractor model combination, we train a system with the three training strategies: *E2E LID*, *2-step*, and *2-step then E2E LID*. Language recognition performance of each method is reported on Table 1 for the three test duration conditions of the NIST LRE2007 corpus. As a baseline system, we train two language recognition models directly with mel-filterbank features, without feature extractor module.

First, we observe that for all architectures, *E2E LID* training gives acceptable results (contrary to what was observed in [13]). The additional processing with the feature extractor helps to improve over the baseline system trained with mel-filterbank features. Then we see that *2-step* training improves over E2E performance for the traditional stacked DNNs but not for the Conformer. It indicates that the sequence-to-sequence *ASR* architecture is more appropriate for *E2E LID* training than the stacked DNNs. For all models, *2-step then E2E LID* improves over *2-step* training, and it always gives a better performance than *E2E LID* training for segments of 3s.

The ResNet-1D architecture is consistently better than the TDNN and our best-performing recipe (Resnet-1D with Conformer feature extractor and *2-step then E2E LID* strategy) achieves state-of-the-art performance in terms of C_{avg} on the

Table 1: *Language recognition performance on the three test sets of the corpus NIST LRE 2007, corresponding to different durations. For two architectures of the language identificaton module and two architectures of the feature extractor, we explore three training strategies presented in Subsection 2.2. Multi-task training is experimented for the best combination of architectures.*

LID model	feature extractor	training strategy	Test - 3s (%)			Test - 10s (%)			Test - 30s (%)		
			EER	minDCF	C_{avg}	EER	minDCF	C_{avg}	EER	minDCF	C_{avg}
TDNN	None		15.25	11.50	13.76	10.03	6.98	9.02	10.06	6.29	9.03
	stacked DNNs	E2E LID: FE+LID	11.43	10.03	11.34	5.58	4.85	5.93	3.03	2.28	2.92
		2-step: FE then LID	7.93	7.31	8.12	**2.43**	1.80	2.40	1.49	0.85	**1.37**
		2-step then E2E LID	**7.21**	**6.53**	**7.38**	2.56	**1.72**	**2.28**	**1.30**	**0.79**	1.61
	Conformer	E2E LID: FE+LID	7.30	6.05	7.21	2.73	2.28	3.17	1.53	0.80	1.27
		2-step: FE then LID	8.01	6.81	8.42	3.08	1.84	2.53	**0.91**	**0.38**	**0.77**
		2-step then E2E LID	**6.00**	**5.48**	**5.91**	**2.38**	**1.63**	**2.28**	0.98	0.52	1.05
ResNet-1D	None		13.37	11.58	13.67	7.34	4.79	6.82	7.99	4.49	5.27
	stacked DNNs	E2E LID: FE+LID	11.92	10.49	11.55	3.73	3.11	3.69	2.13	0.90	1.79
		2-step: FE then LID	7.47	6.32	7.21	**1.69**	**1.17**	**1.69**	**0.88**	**0.35**	**0.62**
		2-step then E2E LID	**6.24**	**5.56**	**6.25**	1.97	1.51	2.16	1.44	0.59	1.55
	Conformer	E2E LID: FE+LID	6.51	5.67	6.56	1.63	**1.02**	**1.50**	**0.70**	0.33	0.59
		2-step: FE then LID	9.13	7.71	8.46	2.03	1.41	1.95	1.16	0.57	1.18
		2-step then E2E LID	**5.76**	**5.12**	**5.93**	**1.48**	1.11	1.58	**0.70**	**0.21**	**0.53**
		multi-task $\lambda = 0.5$	5.99	**4.81**	5.91	2.13	1.16	2.03	1.26	0.67	1.31
		multi-task $\lambda = 0 \to 1$	**5.58**	4.89	**5.78**	1.75	1.12	1.79	1.02	0.50	0.71

three test sets, as compared to the best published results, [7] for 3s and 10s segments and [2, 39] for 30s utterances.

4.2. Multi-task training

We evaluate the *multi-task* training strategy with the ResNet-1D and Conformer architectures. Two systems are trained with different choices for the value of the weight λ of the language identification loss. Both systems are trained for 100 epochs (same number of epochs as for training the feature extractor module in the *2-step* strategy). Results are reported in Table 1.

A first system is trained with a fixed value of $\lambda = 0.5$, which corresponds to an equal weighting of the two loss functions. It achieves similar performance as the *2-step then E2E LID* strategy for segments of 3 seconds. In addition, training is performed in a unique step with the *multi-task* strategy, that makes the training recipe simpler.

In the *2-step then E2E LID* strategy, the feature extractor is first trained with the *ASR* loss and then with the *LID* objective. It is the most successful strategy. We imitate this approach with a *multi-task* training strategy where the weight λ is linearly increased from 0 to 1 during training ($\lambda = 0 \to 1$, in Table 1). This strategy gives the best performance for segments of 3 seconds.

4.3. Discussion

Using a Conformer sequence-to-sequence feature extractor, we have successfully trained ASR-based multilingual bottleneck features without explicltly performing forced phone alignment. A similar behavior was observed for a dialect recognition task [26]. We compare these features with a very strong baseline: classical multilingual bottleneck features [11].

An important part of the performance gain with bottleneck features comes from the additional layers of the feature extractor module. *E2E LID* training of the whole model allows to benefit from the feature extractor and it achieves very good performance with the Conformer model. Pre-training of the encoder with an *ASR* supervision (*2-step* training) allows to achieve a good performance, but fine-tuning of the feature-extractor with the language identification loss improves further. Multi-task training of the feature extractor achieves similar results, with a unique step of training. It means that supervision of the feature extractor training with the language identification loss is necessary to achieve the best language recognition performance.

5. Conclusion

We study the problem of optimally using an automatic speech recognition task to design frame-level features for language recognition. We trained end-to-end sequence-to-sequence multilingual bottleneck features on the Babel corpus and achieved state-of-the-art language recognition performance on the NIST LRE2007 corpus.

We observe that the use of an end-to-end sequence-to-sequence speech recognition model trained with the CTC loss achieves similar performance as classical bottleneck features trained with a frame-level phone recognition task, while being easier to train because it does not need a forced frame alignment of target labels. An important part of this gain comes from the architecture of the feature extractor (Conformer instead of stacked DNNs), that can also be successfully trained in an end-to-end approach with the language identification module and associated loss.

For both classical bottleneck features and end-to-end sequence-to-sequence ones, a substantial improvement can be obtained by fine-tuning the feature extractor with the language identification loss. A similar performance can be achieved with end-to-end training with a multi-task objective. In the future, we plan to study further the optimal scheduling strategy for the relative weights of the two loss functions in the multi-task objective.

6. Acknowledgements

Experiments presented in this paper were carried out using the Grid5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). This work has been partly funded by the French Direction Générale de l'Armement.

7. References

[1] E. Ambikairajah, H. Li, L. Wang, B. Yin, and V. Sethu, "Language identification: a tutorial," *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 82–108, 2011.

[2] W. Cai, J. Chen, J. Zhang, and M. Li, "On-the-fly data loader and utterance-level aggregation for speaker and language recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1038–1051, 2020.

[3] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," in *Proc. Interspeech*, 2011, pp. 861–864.

[4] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller Jr, "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features," in *Proc. Interspeech*, 2002, pp. 89–92.

[5] G. Gelly and J.-L. Gauvain, "Spoken language identification using LSTM-based angular proximity." in *Proc. Interspeech*, 2017, pp. 2566–2570.

[6] M. F. Ben Zeghiba, J.-L. Gauvain, and L. Lamel, "Phonotactic language recognition using MLP features," in *Proc. Interspeech*, 2012, pp. 2041–2044.

[7] S. Ling, J. Salazar, Y. Liu, and K. Kirchhoff, "BERTPHONE: Phonetically-aware encoder representations for utterance-level speaker and language recognition," in *Proc. Odyssey*, 2020, pp. 9–16.

[8] S. Pascual, M. Ravanelli, J. Serr, A. Bonafonte, and Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," in *Proc. Interspeech*, 2019, pp. 161–165.

[9] D. Romero, L. F. D'Haro, and C. Salamea, "Exploring transformer-based language recognition using phonotactic information," in *Proc. IberSPEECH*, 2021, pp. 250–254.

[10] I. Kukanov, T. N. Trong, V. Hautamäki, S. M. Siniscalchi, V. M. Salerno, and K. A. Lee, "Maximal figure-of-merit framework to detect multi-label phonetic features for spoken language recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 682–695, 2020.

[11] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černocký, "Multilingually trained bottleneck features in spoken language recognition," *Computer Speech & Language*, vol. 46, no. Supplement C, pp. 252 – 267, 2017.

[12] H. Li, B. Ma, and K. A. Lee, "Spoken language recognition: from fundamentals to practice," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013.

[13] Z. Tang, D. Wang, Y. Chen, L. Li, and A. Abel, "Phonetic temporal neural model for language identification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 134–144, 2017.

[14] B. Jiang, Y. Song, S. Wei, J.-H. Liu, I. V. McLoughlin, and L.-R. Dai, "Deep bottleneck features for spoken language identification," *PloS One*, vol. 9, no. 7, p. e100795, 2014.

[15] M. McLaren, L. Ferrer, and A. Lawson, "Exploring the role of phonetic bottleneck features for speaker and language recognition," in *Proc. ICASSP*. IEEE, 2016, pp. 5575–5579.

[16] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors." in *Odyssey*, 2018, pp. 105–111.

[17] Z. Ma and H. Yu, "Language identification with deep bottleneck features," *arXiv preprint arXiv:1809.08909*, 2018.

[18] Z. Peng, S. Feng, and T. Lee, "Adversarial multi-task deep features and unsupervised back-end adaptation for language recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 5961–5965.

[19] A. Lozano-Diez, R. Zazo, D. T. Toledano, and J. Gonzalez-Rodriguez, "An analysis of the influence of deep neural network (DNN) topology in bottleneck feature based language recognition," *PloS One*, vol. 12, no. 8, p. e0182580, 2017.

[20] O. Adams, M. Wiesner, S. Watanabe, and D. Yarowsky, "Massively multilingual adversarial speech recognition," in *Proc. ACL*, 2019, pp. 96–108.

[21] M. Zhao, R. Li, S. Yan, Z. Li, H. Lu, S. Xia, Q. Hong, and L. Li, "Phone-aware multi-task learning and length expanding for short-duration language recognition," in *Proc. APSIPA*. IEEE, 2019, pp. 433–437.

[22] L. Li, Z. Li, Y. Liu, and Q. Hong, "Deep joint learning for language recognition," *Neural Networks*, 2021.

[23] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[24] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. ICML*. PMLR, 2016, pp. 173–182.

[25] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*. PMLR, 2014, pp. 1764–1772.

[26] Z. Ren, G. Yang, and S. Xu, "Two-stage training for chinese dialect recognition," in *Proc. Interspeech*, 2019, pp. 4050–4054.

[27] S. Punjabi, H. Arsikere, Z. Raeesy, C. Chandak, N. Bhave, A. Bansal, M. Müller, S. Murillo, A. Rastrow, A. Stolcke *et al.*, "Joint ASR and language identification using RNN-T: An efficient approach to dynamic language switching," in *Proc. ICASSP*. IEEE, 2021.

[28] S. Watanabe, T. Hori, and J. R. Hershey, "Language independent end-to-end architecture for joint language identification and speech recognition," in *Proc. ASRU*. IEEE, 2017, pp. 265–271.

[29] D. Liu, J. Xu, P. Zhang, and Y. Yan, "A unified system for multilingual speech recognition and language identification," *Speech Communication*, vol. 127, pp. 17–28, 2021.

[30] A. Silnova, P. Matějka, O. Glembek, O. Plchot, O. Novotný, F. Grézl, P. Schwarz, L. Burget, and J. Černocký, "BUT/Phonexia Bottleneck Feature Extractor," in *Proc. Odyssey*, 2018, pp. 283–287.

[31] R. Duroselle, M. Sahidullah, D. Jouvet, and I. Illina, "Language recognition on unknown conditions: the LORIA-Inria-MULTISPEECH system for AP20-OLR Challenge," in *Proc. Interspeech*, 2021.

[32] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 66–71.

[33] "The 2007 NIST language recognition evaluation plan," in *website http://www.nist.gov/speech/tests/lre/2007/LRE07EvalPlan-v8b.pdf*, 2007.

[34] "Babel project," in *website https://www.iarpa.gov/index.php/research-programs/babel*

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[36] P. Izmailov, A. Wilson, D. Podoprikhin, D. Vetrov, and T. Garipov, "Averaging weights leads to wider optima and better generalization," in *Proc. Conference on Uncertainty in Artificial Intelligence*, 2018, pp. 876–885.

[37] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[38] S. Molau, F. Hilger, and H. Ney, "Feature space normalization in adverse acoustic conditions," in *Proc. ICASSP*, vol. 1. IEEE, 2003, pp. 656–659.

[39] W. Cai, Z. Cai, W. Liu, X. Wang, and M. Li, "Insights into end-to-end learning scheme for language identification," in *Proc. ICASSP*. IEEE, 2018, pp. 5209–5213.