

# Contextual Phonetic Pretraining for End-to-end Utterance-level Language and Speaker Recognition

Shaoshi Ling, Julian Salazar, Katrin Kirchhoff

Amazon AI

{shaosl, julsal, katrinki}@amazon.com

## Abstract

Pretrained contextual word representations in NLP have greatly improved performance on various downstream tasks. For speech, we propose *contextual frame representations* that capture phonetic information at the acoustic frame level and can be used for utterance-level language, speaker, and speech recognition. These representations come from the frame-wise intermediate representations of an end-to-end, self-attentive ASR model (SAN-CTC) on spoken utterances. We first train the model on the Fisher English corpus with context-independent phoneme labels, then use its representations at inference time as features for task-specific models on the NIST LRE07 closed-set language recognition task and a Fisher speaker recognition task, giving significant improvements over the state-of-the-art on both (e.g., language EER of 4.68% on 3sec utterances, 23% relative reduction in speaker EER). Results remain competitive when using a novel dilated convolutional model for language recognition, or when ASR pretraining is done with character labels only.

**Index Terms:** phonetic information, transfer learning, language recognition, speaker recognition, self-attention

## 1. Introduction

Pretrained representations have recently become a key component in many natural language understanding and speech tasks. However, learning high-quality acoustic representations that can be adapted to downstream speech tasks remains challenging. These representations would need to capture both higher-level phonetic and original acoustic information. Two major downstream tasks that have benefited from phonetic information are *language recognition* (LR), which identifies the language spoken, and *speaker recognition* (SR), which identifies the speaker from a known collection [1, 2].

Concurrently, the NLP community continues to build better contextual text representations such as ELMo [3] and BERT [4]. Both as-is usage and finetuning of these representations showed significant improvements in downstream NLP tasks. In the speech community however, only non-contextualized embeddings [5–8] have been proposed thus far, with some success in SR, gender, and emotion recognition. However, the potential gains from contextualization and self-attention [9] for speech tasks has not previously been explored.

In this work, we propose a system that learns phonetically-aware contextual frame representations that can be used for downstream tasks, namely language and speaker recognition, in an utterance-level, end-to-end manner. We do this by leveraging transcribed but unaligned speech data by pretraining a self-attentive CTC model [10] against context-independent labels (phonemes, characters) and then adapting a few BLSTM, CNN, or dilated CNN blocks towards the LR or SR classes during task-specific training. Our contributions are:

- Contextualized frame representations from a pretrained ASR model
- Using end-to-end ASR with unaligned labels (characters, phonemes) as the pretraining objective
- Using phonetic representations from a self-attentive architecture
- Dilated convolution blocks as an alternative to CNN + BLSTM combinations for LR
- Simple task adaptation to LR and SR, outperforming previous state-of-the-art results.

We also include evidence for the use of high-level vs. low-level features in LR vs. SR, and an analysis of how pretraining label sets affect downstream performance.

## 2. Prior work

### 2.1. Acoustic representations

Beyond the frame-wise neural representations implicitly learned by hybrid ASR systems, NLP-inspired phoneme, character, and word embeddings like Phoneme2Vec [8], Char2Wav [5], and Speech2Vec [6] have been proposed in the speech community. Learned utterance embeddings [7, 8] have also been proposed, in order to capture long-term dependencies required for higher-level reasoning. In particular, [7] pooled over the CNN layers of an acoustic model and used them for gender, noise, and speaker recognition.

### 2.2. Language and speaker recognition

Since LR and SR are both classification tasks, early acoustic approaches involved statistical models of frame-level units which are then dimensionally-reduced into *i*-vector space [11, 12]. More recently, deep neural networks (DNNs) have found significant use in both tasks, from feed-forward models [13], along with recurrence encoded by RNNs [14–17] and temporal invariance with CNNs [18–20]. These models work by average pooling posterior predictions to give the final result.

The success of recurrent models suggests there is a lot to gain by viewing frames in an *utterance-level* context. Fully neural embeddings were used in SR with Deep Speaker [21] and VGGVox [22]. To handle variable-length speech utterances, several pooling layers such as temporal average pooling (TAP) [23], self-attentive pooling (SAP) [24–26], learnable dictionary encoding (LDE) layers [27], and GhostVLAD [28] were introduced, with the latter types giving state-of-the-art due to their content-aware aggregation operations.

### 2.3. System combination

Beyond single-task success, the relationship between ASR, LR, and SR has been explored in various combinations. LR and SR

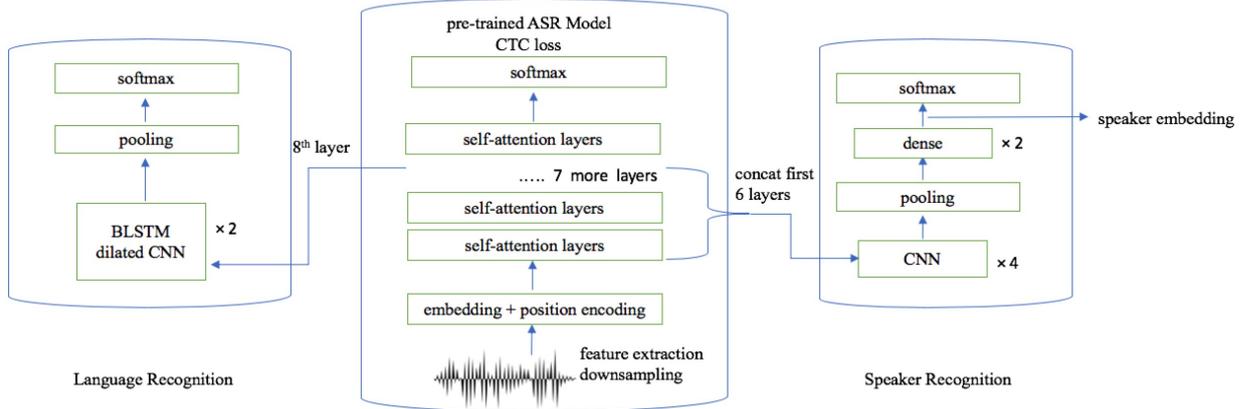


Figure 1: Our proposed contextual representation system

have been unified [29, 30] giving improvements in both tasks. Likewise, using phonetic information from ASR has shown great success in the past, from original phonotactic systems [1] to more recent i-vector and x-vector systems trained on frame-level features extracted from DNN ASR models [2, 29, 31].

### 3. Architecture

In pretrained DNN i-vector systems [2, 29], the DNN gives a distribution over classes of context-dependent phones, which are then aggregated into sufficient statistics for i-vector training. In our system, we incorporate a pretrained end-to-end ASR model (SAN-CTC [10]) as a feature extractor, choosing this model in particular as it closely matches the Transformer encoder used by BERT [4]. We freeze the ASR model’s weights, then train shallower neural networks for either LR or SR on the frame-wise representations from selected layers in a transfer learning manner. The whole system is depicted in Figure 1.

#### 3.1. Contextual phonetic pretraining

Our pretrained model is a deep, fully self-attentional network trained end-to-end with CTC loss [10]. Unlike hybrid HMM-NN models, CTC-based models require less supervision as they perform alignment directly, allowing for context-independent phones and even characters as utterance labels. Unlike encoder-decoder systems, CTC-based models are able to predict tokens in parallel at inference time. It is parallelized by using each position’s representation to attend to all others, giving a contextualized representation for that position. Hence, the full receptive field is immediately available at the cost of  $O(T^2)$  inner products, enabling richer representations in fewer layers.

We then remove the last two layers of the pretrained ASR model. We find that these last layers are task-specific to the CTC loss (e.g., the placement of blank labels for CTC’s label collapsing procedure). The remaining activations from the ASR model can be viewed as contextual phonetically pretrained representations at every time step. Because relevant information (e.g., pitch, accents, rate of speech) might be normalized out in some intermediate ASR layers (as they are not relevant to transcription), we optionally concatenate outputs across these remaining layers [7] to make sure this information is still available in the learned representations.

#### 3.2. Task-specific architecture

For finetuning, we feed these representations (MFCCs for our baselines, SAN-CTC representations for our proposed system) into task-specific networks. We aim to introduce minimal task-specific parameters. The parameters of the new layers are learned to maximize the log-probability of the correct label via cross-entropy loss (classification). The ASR model weights are frozen.

##### 3.2.1. Language recognition

We take outputs of the 8th layer of the self-attentive stack, pass it through 2 BLSTM layers or DiCNN blocks, and then perform self-attentive pooling before a linear + softmax layer.

##### 3.2.2. Speaker recognition

Since lower layers contain more acoustic information relevant to this task, we position-wise concatenate the outputs of the 1st through 6th layers of the self-attentive stack, perform 4 layers of 1D convolutions with kernel sizes (2,2,3,1) (inspired by the front-end in [23]), apply self-attentive pooling, and then use two dense + ReLU layers before a linear + softmax layer.

##### 3.2.3. Dilated convolution (DiCNN) blocks

Given recent work that combining CNNs with BLSTMs before SAP improves LR performance [26], we propose using *dilated convolutions* to capture local extraction and a wide temporal receptive field without resorting to recurrent layers. These are convolutions over an area larger than the number of filter parameters, performed by skipping inputs with a certain step. The WaveNet model for speech synthesis [34] used stacked, causal, dilated convolutions on audio to increase receptive field. We use a non-causal version previously explored for non-recurrent speech recognition; following [35], our dilated convolution (DiCNN) block consists of 6 1D convolutions of increasing dilation (1, 2, 4, 8, 16, 32), along with residual and gating connections. The skip connection is combined with the stack’s output via a 1x1 convolution.

##### 3.2.4. Self-attention pooling (SAP)

Self-attentive pooling [36] allows the content-dependent aggregation of variable-length sequences into a fixed-size vector.

Table 1: NIST LRE07 closed-set, general LR task results for individual models

System	Phn.-aware?	3sec		10sec		30sec		Total	
		$C_{avg}$	EER%	$C_{avg}$	EER%	$C_{avg}$	EER%	$C_{avg}$	EER%
LSTM with attention [15]		–	14.72	–	–	–	–	–	–
CG-LSTM-angular [16]		–	–	–	–	–	–	10.9	–
CNN SAP [30]		8.59	9.89	<b>2.49</b>	4.27	1.09	2.38	–	–
CNN-BLSTM SAP [26]		9.22	9.50	2.54	3.48	<b>0.97</b>	1.77	–	–
CNN-LDE [27]		<b>8.25</b>	<b>7.75</b>	2.61	<b>2.31</b>	1.13	<b>0.96</b>	–	–
i-vector GMM [11, 32]	senone	26.04	–	11.93	–	4.52	–	14.17	–
i-vector DNN [29, 32]	senone	19.67	–	7.84	–	3.31	–	10.27	–
DNN phonotactic [33]	senone	18.59	12.79	6.28	4.21	1.34	0.79	5.99	8.73
DNN PPP features [27]	senone	<b>8.00</b>	<b>6.90</b>	<b>2.20</b>	<b>1.43</b>	<b>0.61</b>	<b>0.32</b>	–	–
BLSTM + SAP		25.26	17.01	19.06	12.33	18.09	10.94	20.80	13.48
DiCNN + SAP		22.37	14.50	14.69	8.43	12.63	7.32	16.56	10.21
pretrain + BLSTM + SAP (no LRE09)	phoneme	7.80	4.68	2.48	1.48	1.11	0.56	3.80	2.32
pretrain + DiCNN + SAP	phoneme	7.56	4.77	2.07	1.30	1.15	<b>0.42</b>	3.59	2.24
pretrain + BLSTM + SAP	phoneme	<b>7.22</b>	<b>4.68</b>	<b>1.95</b>	<b>1.25</b>	<b>1.01</b>	0.51	<b>3.40</b>	<b>2.18</b>

This improves on the older approach of averaging over posteriors or frames, as not all feature frames contribute equally to the utterance-level classification (e.g., silence or noise frames). CNNs, BLSTMs, and DiCNN blocks play a role as local pattern extractors for variable-length inputs, after which self-attentive pooling is used to get a global utterance-level representation.

SAP was first proposed for document classification [36] and then adapted to LR and SR [24, 25]. The importance of each frame is given by the similarity of a hidden state  $\mathbf{h}_t$  (the output of a dense layer with tanh activation) with a learned context vector  $\boldsymbol{\mu}$ . This gives a normalized probability  $\alpha_t$  via softmax:

$$\mathbf{e} = \sum_{t=1}^T \alpha_t \mathbf{h}_t, \quad \alpha_t = \frac{\exp(\mathbf{h}_t^\top \boldsymbol{\mu})}{\sum_{t=1}^T \exp(\mathbf{h}_t^\top \boldsymbol{\mu})}$$

The context vector  $\boldsymbol{\mu}$  can be viewed as a fixed query for informative frames, as later generalized with self-attention layers [9].

## 4. Experimental setup

We use Kaldi [32] for data preparation and MXNet [37] for modeling. Our self-attention code is based on GluonNLP’s implementation.

### 4.1. Feature extraction and pretraining

We use the same audio features for all tasks, taking a window of 25ms, a hop of 10ms, and 20 cepstral mean-normalized MFCCs with temporal first- and second-order differences. This matches past SR work [31], except we forego voice activity detection and concatenate every three frames as in SAN-CTC [10].

We pretrain by doing ASR on the Fisher English telephony corpus [38], using the *train* split of the *fisher\_english/s5* recipe. Our labels are phonemes from CMUdict with lexical stresses<sup>1</sup> (giving 84 non-silence classes). Our model, objective, and learning schedule is largely similar to SAN-CTC [10], with 10 self-attention layers, 552 hidden dimensions, 8 heads, giving 35M parameters. The input to the self-attentive stack is 512 dimensions from the initial dense layer and 40 dimensions from the positional embedding. We take a batch size of 40, and for the inverse square-root decay we take  $\lambda = 400$  and a warmup of 16000 steps for 30 epochs, fix then decay the learning rate by 0.1 for 10 epochs, doing this twice for 50 epochs in total.

<sup>1</sup><http://www.speech.cs.cmu.edu/cgi-bin/cmudict>

### 4.2. Language recognition

We use the closed-set, general LR task of the 2007 NIST LR Evaluation (LRE07), where one identifies an utterance’s language among 14 languages. We use the *train* split of the *lre07/v2* recipe, which includes LRE07’s training data along with CALLFRIEND, LRE96/03/05/09, and SRE08. Each utterance is split into 4s segments with 50% overlap, similar to [16].

Every epoch, 8,000 to 12,000 segments are randomly selected per language and distributed over batches to mitigate class imbalance. We do stochastic gradient descent with batch size 128, momentum 0.9, weight decay  $1e-4$ , and a learning rate of 0.01 for 60 epochs, after which we decay by 0.1 every 20 epochs, doing this thrice. At test time, all duration cases (3sec, 10sec, 30sec) are tested one-by-one on the same model. Segments are split into non-overlapping segments of  $\sim 4$ s before inference, and then the pooling layer occurs across all frames of all segments to give the final language prediction.

### 4.3. Speaker recognition

We use the Fisher speaker recognition task described in [31], using the same training and evaluation sets<sup>2</sup>. This selects 172 hours of data from 5,000 speakers for training, and takes a disjoint set of 1,000 speakers for evaluation. Each person is enrolled using 10 segments (about 30sec in total) and evaluated on three 3sec segments.

We split all utterances into 300 frames, which leaves us 260,294 segments of which we take 1,500 for validation. We perform stochastic gradient descent with batch size 64, momentum 0.9, weight decay  $1e-4$ , and a learning rate of 0.01 which is decayed by 10 when validation loss plateaus. The network is trained to classify the  $N$  speakers in the training data. After training, test data is fed into the model as the whole sequence and embeddings are extracted from the affine component of the dense layer after pooling. Then, we use the same type of PLDA classifier procedure as in existing x-vector systems [23].

## 5. Results

### 5.1. Language recognition

Table 1 shows the performance of our baselines and pretrained models on the LRE07 closed-set task, as compared to end-to-

<sup>2</sup>[https://github.com/mycrazycrazy/Speaker\\_embedding\\_with\\_phonetic\\_information](https://github.com/mycrazycrazy/Speaker_embedding_with_phonetic_information)

end and/or phonetically pretrained systems. The performance is reported in average detection cost  $C_{\text{avg}}$  [39] and equal error rate (EER%). For fairer comparison with [26, 27, 30], we also train a model where LRE09 is omitted to attain a similar utterance count, which affects  $C_{\text{avg}}$  more than the underlying EER.

Our baselines, which are the shallower task systems trained directly on MFCC features, perform poorly, with DiCNN outperforming BLSTM as-is. However, when used with our contextual phonetic pretraining via end-to-end ASR, one gets significant improvements on the state-of-the-art over published non-phonetically aware systems at all durations, with the simpler BLSTM outperforming the DiCNN blocks. These improvements also hold over phonetically-aware systems (the best of which is also pretrained on Fisher English with a senone alphabet of 5,000+) except for the 30sec condition. Our training and evaluation method of using 4sec segments (intended as a regularizer) seemed to bias improvements, giving 4.68% in EER on the challenging 3sec recognition task compared to a previous state-of-the-art of 6.90%. Shortcomings at the 30sec mark suggests room for non-splitting approaches to training/testing, or that using other pooling methods like LDE or VLAD could give further improvements.

## 5.2. Speaker recognition

Table 2 shows our model’s task performance in terms of equal error rate (EER) and minimum detection costs (minDCF08/10) from SRE08/10 [40, 41]. The results of i-vector and x-vector from the original work are first presented. The phonetic systems are pretrained on Switchboard [31] or Fisher (ours). To ablate the merit of the SAP layer, we train a similar x-vector architecture with self-attentive instead of regular statistics pooling to get a 13.2% relative EER reduction. Training on contextual frame representations induced from the ASR model leads to better performance than the x-vector approach and improves performance even over the multitasking approach (where the phonetic extractor is learned jointly between ASR and SR) [31].

Table 2: Fisher speaker recognition task results

System	EER	minDCF08	minDCF10
i-vector [31]	2.10	0.093	0.3347
x-vector + stat. pooling [31]	1.73	0.086	0.3627
phn. vec. + finetune [31]	1.60	0.076	0.3413
+ multi-tasking [31]	1.39	0.073	0.3087
x-vector + SAP	1.50	0.074	0.2973
pretrain + CNN + SAP	<b>1.07</b>	<b>0.052</b>	<b>0.2247</b>

## 5.3. Comparison of downstream tasks

While utterance-level LR and SR appear similar as classification problems on speech, one would expect different features to be discriminative for each task. Inspired by ELMo [3], we learn global, softmax-normalized, scalar weights  $s_\ell$  for each layer:

$$\mathbf{h}_t^{\text{task}} = \sum_{\ell=1}^{10} s_\ell^{\text{task}} \mathbf{h}_{t,\ell}^{\text{ASR}}. \quad (1)$$

Since each  $\mathbf{h}_{t,\ell}^{\text{ASR}}$  is layer-normalized by SAN-CTC, we interpret the weights without rescaling. In Figure 2, we see that the SR model largely assigns its weight to the earlier hidden layers while LR uses representations over all layers. This matches the intuition that SR uses lower-level features (qualities like

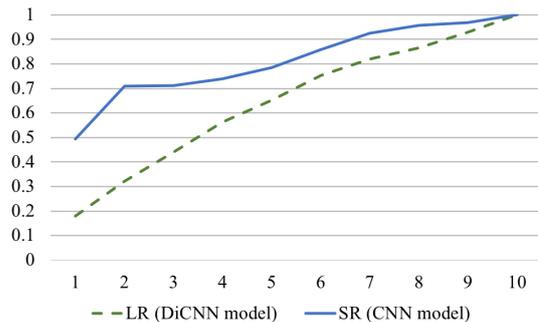


Figure 2: Cumulative weights ( $\sum_{\ell=1}^L s_\ell^{\text{task}}$ ) for LR and SR

pitch and range), while LR additionally uses higher-level features (e.g., a language’s preferred phonetic sequences).

## 5.4. Choice of pretraining alphabet

We evaluate how the choice of label set for ASR affects downstream performance. We train two additional SAN-CTC models with smaller label sets: phonemes without lexical stress (39 non-silence classes), and characters (uncased letters, digits, space, punctuation). We train our best task models on these systems as before; results are in Table 3.

Table 3: Best system results per CTC pretraining alphabet. Token error rate is on the Fisher English test split.

Pretraining alphabet	Tkn. ER	LR EER	SR EER
characters	9.47	2.58	1.28
phonemes (no stress)	9.74	2.13	1.16
phonemes (with stress)	10.07	2.18	1.07

As expected, performance improves as the ASR task becomes more discriminative (more classes) and supervised (lexicon specificity). Removing lexical stress affected SR performance but not LR performance; perhaps relative emphasis is more discriminative for speakers. Even pretraining a lexicon-free character model outperforms previous models without pretraining ( $C_{\text{avg}}$ /EER of 8.57/5.14 on the 3sec set for LR, EER of 1.28 for SR), but this is unsurprising as character CTC is known to still learn phonetic representations internally [42].

## 6. Conclusion

We propose a system that can learn phonetically-aware contextual frame representations via ASR. These can be adapted to downstream tasks, namely language and speaker recognition, in an utterance-level, end-to-end manner. We do this by leveraging transcribed but unaligned speech data to pretrain a self-attentive CTC model, whose intermediate representations can be adapted by recurrent or (dilated) convolutional layers and pooled towards the desired task.

Future work includes building more robust representations and showing effective transfer learning from acoustic representations to further tasks in speech technology (emotion recognition, gender detection, etc.). One could relax the supervised ASR task to semi-supervised or self-supervised learning, as is done with language representation modeling [4]. Success would help low-resource speech tasks in various languages benefit from high-resource speech tasks like English ASR.

## 7. References

- [1] J. Gauvain and L. F. Lamel, "Identification of non-linguistic speech features," in *HLT*, 1993.
- [2] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *ICASSP*, 2014, pp. 1695–1699.
- [3] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, 2018.
- [4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.
- [5] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. C. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," in *ICLR (workshop)*, 2017.
- [6] Y.-A. Chung and J. Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," in *INTERSPEECH*, 2018, pp. 811–815.
- [7] J. Rownicka, P. Bell, and S. Renals, "Analyzing deep cnn-based utterance embeddings for acoustic model adaptation," in *SLT*, 2018, pp. 235–241.
- [8] A. Haque, M. Guo, P. Verma, and L. Fei-Fei, "Audio-linguistic embeddings for spoken sentences," in *ICASSP*, 2019.
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.
- [10] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *ICASSP*, 2019.
- [11] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," in *INTERSPEECH*, 2011, pp. 857–860.
- [12] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *TASLP*, vol. 19, no. 4, pp. 788–798, 2011.
- [13] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *ICASSP*, 2014, pp. 5337–5341.
- [14] J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno, "Automatic language identification using long short-term memory recurrent neural networks," in *INTERSPEECH*, 2014, pp. 2155–2159.
- [15] W. Geng, W. Wang, Y. Zhao, X. Cai, B. Xu, C. Xinyuan *et al.*, "End-to-end language identification using attention-based recurrent neural networks," in *INTERSPEECH*, 2016, pp. 2944–2948.
- [16] G. Gelly and J.-L. Gauvain, "Spoken language identification using lstm-based angular proximity," in *INTERSPEECH*, 2017, pp. 2566–2570.
- [17] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*, 2016, pp. 5115–5119.
- [18] A. Lozano-Diez, R. Zazo-Candil, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez, "An end-to-end approach to language identification in short utterances using convolutional neural networks," in *INTERSPEECH*, 2015, pp. 403–407.
- [19] M. Tkachenko, A. Yamshinin, N. Lyubimov, M. Kotov, and M. Nastasenko, "Language identification using time delay neural network d-vector on short utterances," in *SPECOM*, 2016, pp. 443–449.
- [20] S.-X. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, "End-to-end attention based text-dependent speaker verification," in *SLT*, 2016, pp. 171–178.
- [21] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *CoRR*, vol. abs/1705.02304, 2017.
- [22] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018, pp. 1086–1090.
- [23] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *ICASSP*, 2018, pp. 5329–5333.
- [24] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification," in *INTERSPEECH*, 2018, pp. 3573–3577.
- [25] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," in *INTERSPEECH*, 2018, pp. 2252–2256.
- [26] W. Cai, D. Cai, S. Huang, and M. Li, "Utterance-level end-to-end language identification using attention-based cnn-blstm," in *ICASSP*, 2019.
- [27] W. Cai, Z. Cai, W. Liu, X. Wang, and M. Li, "Insights into end-to-end learning scheme for language identification," *CoRR*, vol. abs/1804.00381, 2018.
- [28] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP*, 2019.
- [29] F. Richardson, D. Reynolds, and N. Dehak, "A unified deep neural network for speaker and language recognition," in *INTERSPEECH*, 2015, pp. 1146–1150.
- [30] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Speaker Odyssey*, 2018.
- [31] Y. Liu, L. He, J. Liu, and M. T. Johnson, "Speaker embedding extraction with phonetic information," in *INTERSPEECH*, 2018, pp. 2247–2251.
- [32] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldı speech recognition toolkit," IEEE Signal Processing Society, Tech. Rep., 2011.
- [33] G. Gelly, J.-L. Gauvain, V. B. Le, and A. Messaoudi, "A divide-and-conquer approach for language identification based on recurrent neural networks," in *INTERSPEECH*, 2016, pp. 3231–3235.
- [34] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *SSW*, 2016.
- [35] N. Kim and K. Park, "Speech-to-text-wavenet." <https://github.com/buriburisuri/>, 2016.
- [36] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *NAACL HLT*, 2016, pp. 1480–1489.
- [37] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.
- [38] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: a resource for the next generations of speech-to-text," in *LREC*, 2004.
- [39] NIST, "The 2007 nist language recognition evaluation plan (Ire07)," <https://catalog.ldc.upenn.edu/docs/LDC2009S04/LRE07EvalPlan-v8b-1.pdf>.
- [40] A. F. Martin and C. S. Greenberg, "NIST 2008 speaker recognition evaluation: performance across telephone and room microphone channels," in *INTERSPEECH*, 2009, pp. 2579–2582.
- [41] —, "The NIST 2010 speaker recognition evaluation," in *INTERSPEECH*, 2010, pp. 2726–2729.
- [42] Y. Belinkov and J. R. Glass, "Analyzing hidden representations in end-to-end automatic speech recognition systems," in *NeurIPS*, 2017, pp. 2438–2448.