Investigation of Senone-based Long-Short Term Memory RNNs for Spoken Language Recognition

Yao Tian, Liang He, Yi Liu, Jia Liu

National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

tianyao11@mails.tsinghua.edu.cn, heliang@mail.tsinghua.edu.cn
liu-yi15@mails.tsinghua.edu.cn, liuj@mail.tsinghua.edu.cn

Abstract

Recently, the integration of deep neural networks (DNNs) trained to predict senone posteriors with conventional language modeling methods has been proved effective for spoken language recognition. This work extends some of the senone-based DNN frameworks by replacing the DNN with the LSTM RNN. Two of these approaches use the LSTM RNN to generate features. The features are extracted from the recurrent projection layer in the LSTM RNN either as frame-level acoustic features or utterance-level features and are then processed in different ways to produce scores for each target language. In the third approach, the conventional i-vector model is modified to use the LSTM RNN to produce frame alignments for sufficient statistics extraction. Experiments on the NIST LRE 2015 demonstrate the effectiveness of the proposed methods.

Index Terms: Deep neural networks, long short-term memory, recurrent neural networks, spoken language recognition

1. Introduction

Over recent years, many approaches based on Gaussian Mixture Models (GMM) have been proposed for spoken language recognition (SLR) [1], among which i-vector modeling [2, 3] appears to be one of the most effective methods and brings significant performance improvements. In i-vector model, acoustic features (e.g., shifted delta cepstrum (SDC) features) are first converted into high-dimensional statistics. Then they are mapped into a low-dimensional subspace where a speech utterance is represented by a fixed-length vector called i-vector. After the extraction of i-vectors, standard techniques such as Gaussian back-end, logistic regression [4] are applied to the i-vectors of the test utterances to produce scores for each target language.

In the field of speech recognition, the deep neural network (DNN) has become the dominant approach in acoustic modeling as a replacement of GMM, bringing an about 30% relative improvement in word error rate (WER) [5, 6]. In SLR, researchers have also investigated several strategies for using DNNs, and the most successful approaches are those hybrid frameworks where DNNs trained to discriminate between senones (tied triphone states) are combined with conventional language recognition models [7, 8, 9]. In [7], the DNN is used to extract bottleneck features as a replacement of SDC features in i-vector model. This data-driven phonetically-related feature representation has been demonstrated more effective for SLR tasks than the hand-designed feature such as the SDC feature. The method in [9] is also based on i-vector model and they use the DNN to compute frame posterior probabilities during the extraction of sufficient statistics. This work brings impressive relative gains and proves that more accurate content (senones) alignments of frames will benefit SLR tasks. This framework has also been successfully applied in speaker verification [8].

More recently, Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) have been shown to outperform DNNs for acoustic modeling in speech recognition [10]. The recurrent cells and the multiplicative units called gates used to control the flow of information make the model a more powerful tool to model sequence data such as speech signals and their complex long-range correlations. In this paper, we extend some of the senone-based DNN frameworks by using the LSTM RNN as a replacement of the DNN. Three approaches are evaluated in our work. We first explore using the LSTM RNN to extract bottleneck features hoping the inclusion of more temporal information might benefit SLR tasks. Second, we directly use the average outputs of the LSTM recurrent layer as the feature representation of the utterance since it spontaneously stores long-period information. Third, we investigate using the LSTM RNN to provide frame posteriors during the extraction of sufficient statistics for the reason that the LSTM RNN may provide more accurate content frame alignments with sequence information from longer duration. Experiments are carried out on the task defined by NIST LRE 2015. It should be noticed that [11] has also proposed using LSTM RNNs for SLR. However, their work tries to model the language space in a different perspective where the LSTM RNN is trained to predict languages.

The rest of this paper is organized as follows. Section 2 introduces the feature extraction framework with the DNN. Section 3 presents the DNN approach for sufficient statistics extraction. Section 4 gives the LSTM RNN architecture. Experiments and discussions are shown in Section 5. Finally, conclusions are presented in Section 6.

2. DNNs for feature extraction

The DNN based bottleneck features are being widely used in various speech related applications [7, 12, 13]. Bottleneck here means a hidden layer placed in the middle of a DNN which has fewer number of hidden nodes than the other layers. The linear outputs of this layer is referred to the bottleneck feature and it can be regarded as a compact low-dimensional representation of the original inputs. In SLR, the bottleneck feature replaces the traditional acoustic feature (SDC feature) for i-vector modeling. The bottleneck feature contains rich phonetic information since the DNN is trained to discriminate senones. This might benefit the language recognition task where languages are highly content-related. The bottleneck feature based framework is referred to **NN-bottleneck-feature model** in this work.

Another common usage of DNNs as feature extractors is to represent each utterance as the average of outputs derived from one of the DNN's hidden layer [14]. Then decisions can be made between these features in a similar way as in i-vector approach. This framework is referred to **NN-full-feature model**.

3. DNNs for frame alignment

In the traditional i-vector framework, the following sufficient statistics (Baum-Welch statistics) need to be calculated for each utterance during model training and i-vector extraction

$$N_c^{(i)} = \sum_t \gamma_{c.t}^{(i)} \tag{1}$$

$$\mathbf{F}_c^{(i)} = \sum_t \gamma_{c,t}^{(i)} \mathbf{x}_t^{(i)} \tag{2}$$

$$\mathbf{S}_c^{(i)} = \sum_t \gamma_{c,t}^{(i)} \mathbf{x}_t^{(i)} \mathbf{x}_t^{(i)T} \tag{3}$$

Where $\mathbf{x}_t^{(i)}$ is the acoustic feature of speech utterance i at time t. $N_c^{(i)}$, $\mathbf{F}_c^{(i)}$ and $\mathbf{S}_c^{(i)}$ are the zero-order, first-order and second-order statistics with respect to c-th Gaussian component. $\gamma_{c,t}^{(i)}$ is the posterior probability of the c-th Gaussian component and is defined as

$$\gamma_{c,t}^{(i)} = \frac{\omega_c \mathcal{N}(\mathbf{x}_t^{(i)}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}{\sum_{c'=1}^{C} \omega_{c'} \mathcal{N}(\mathbf{x}_t^{(i)}; \boldsymbol{\mu}_{c'}, \boldsymbol{\Sigma}_{c'})} \tag{4}$$

Eq (1) to eq (4) indicate that the extraction of sufficient statistics actually can be seen as a soft alignment of speech frames with respect to each Gaussian component. By treating the senone outputs of the DNN as Gaussian like units in the UBM, Lei [8] proposes to use the senone posterior to do the frame alignment for the i-vector

$$\gamma_{c,t}^{(i)} \leftarrow p(s_c|\mathbf{x}_t^{(i)}) \tag{5}$$

s_c is the c-th units of the DNN's output layer. $p(s_c|\mathbf{x}_t^{(i)})$ is the senone posterior probability given the acoustic feature $\mathbf{x}_t^{(i)}$. Figure 1 presents the processes of the UBM and DNN based sufficient statistics extraction. It can be seen that the features for frame alignments and statistics computation are the same in the UBM framework while these two steps are efficiently decoupled with the introduction of DNNs. This framework is referred to **NN-frame-alignment model**.

4. Long Short-Term Memory RNNs

RNNs have the advantage of modeling longer-range correlations than DNNs. However, the problem of vanishing and exploding gradients in the SGD training makes it difficult to model long-time dependencies using conventional RNNs. The LSTM RNN is an effective solution for this problem. The LSTM RNN architecture realized in this paper is shown in

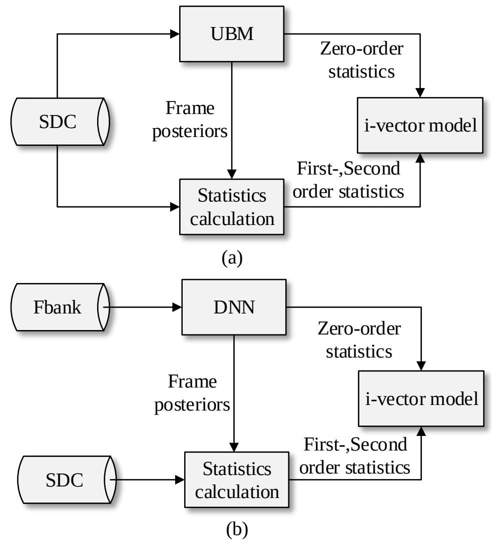

Figure 1: *The flow diagrams of (a) UBM based (b) DNN based sufficient statistics extraction.*

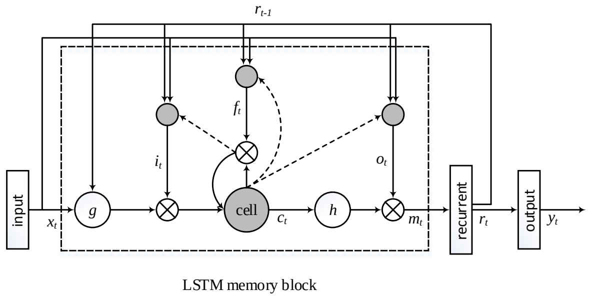

Figure 2: *A single memory block of Long Short-Term Memory recurrent neural network architecture.*

Figure 2. The vector formulas are given below

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ir}\mathbf{r}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{6}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fr}\mathbf{r}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{7}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot g(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{cr}\mathbf{r}_{t-1} + \mathbf{b}_c) \tag{8}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{or}\mathbf{r}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \tag{9}$$

$$\mathbf{m}_t = \mathbf{o}_t \odot h(\mathbf{c}_t) \tag{10}$$

$$\mathbf{r}_t = \mathbf{W}_{rm}\mathbf{m}_t \tag{11}$$

$$\mathbf{y}_t = \phi(\mathbf{W}_{yr}\mathbf{r}_t + \mathbf{b}_y) \tag{12}$$

where $\mathbf{W}_{ix}$, $\mathbf{W}_{fx}$, $\mathbf{W}_{cx}$, $\mathbf{W}_{ox}$ denote weight matrices connected to the inputs, $\mathbf{W}_{ir}$, $\mathbf{W}_{fr}$, $\mathbf{W}_{cr}$, $\mathbf{W}_{or}$ denote weight matrices connected to the LSTM activations. $\mathbf{W}_{ic}$, $\mathbf{W}_{fc}$, $\mathbf{W}_{oc}$ are diagonal peephole connections between cell and the gate signals. $\mathbf{W}_{rm}$ denotes the projection matrix. $\mathbf{W}_{yr}$ denotes the weight matrix of the output layer. The $\mathbf{b}$ terms denote bias vectors. The logistic sigmoid function is used for σ and the tanh function is used for h and g. ϕ refers to softmax function.

The activation of the recurrent projection layer which corresponds to $\mathbf{r}_t$ in eq (11) is regarded as the feature representation when the LSTM RNN is used as a feature extractor.

5. Experiments

5.1. Experimental setup

5.1.1. dataset

Experiments are carried out on the core testing condition of NIST LRE 2015, which is based on the use of only limited and specified training data to develop models for each language. There are twenty target languages and they are further grouped into six language clusters during evaluation process [15]. There are 7020 utterances for target languages modeling. We split them into two sets: 5260 utterances and 1760 utterances. The former set (training dataset) are used for training systems and the latter set (development dataset) are used for self-evaluation, fusion and calibration. To mitigate the degradation caused by the length variation, we further cut each utterance into 3-60 seconds' segments. The training data for senone-driven neural networks is Swichboard-1 provided by NIST LRE 2015 as well.

5.1.2. features

- **Features for language front-end**: 13-dimensional MFCC features with utterance-based mean-covariance normalization are first extracted using a 25ms Hamming window and 10ms frame shift. Then 56-dimensional SDC features are obtained with a 7-1-3-7 parameterization.
- **Features for GMM-HMM model**: 13-dimensional PLP features with speaker-based mean-covariance normalization are extracted first using a 25ms Hamming window and 10ms frame shift. Then their first-, second- and third-order derivatives are concatenated with the basic feature and further reduced to 39 dimensions by HLDA.
- **Features for DNN and RNN models**: 40 log Mel-filterbank (FBank) coefficients with utterance-based mean-covariance normalization are extracted using a 25ms Hamming window and 10ms frame shift. Then delta and delta-delta coefficients are calculated to produce final 120-dimensional feature vectors.

5.1.3. models

- **GMM-HMM**: A GMM-HMM system is firstly trained to generate transcriptions for senones which are used for DNN and RNN training. The GMM-HMM uses 2227 senones tied by a phonetic decision tree.
- **DNN**: The DNN has five-hidden layers and is trained with cross-entropy criterion using the transcriptions generated from the GMM-HMM system. 11 frames (1320 dimension in total) are concatenated as the input of the network. Each hidden layer has 1200 nodes. The fifth layer is the bottleneck feature layer and the full feature layer. The number of nodes of bottleneck layer is 39. The output of the DNN with respect to senones has 2227 nodes.
- **RNN**: The LSTM RNN has two LSTM layers and is trained with cross-entropy criterion using the transcriptions generated from the GMM-HMM system. Each LSTM layer has 800 cells and 512 recurrent projection units. The activation of the recurrent projection layer is regarded as the bottleneck feature layer and the full feature layer. The output of the LSTM RNN with respect to senones has 2227 nodes.
- **Baseline i-vector model**: A language and gender-independent diagonal covariance UBM with 2048 mixtures is trained. The dimensionality of i-vectors is set to 400. Then they are further projected to a 19-dimension subspace by linear discriminant analysis (LDA). Gaussian back-end is adopted to get the final scores.
- **NN-bottleneck-feature model**: SDC features are replaced with bottleneck features. The rest configurations are the same with the baseline i-vector model.
- **NN-full-feature model**: The average of the outputs from the DNN's fifth hidden layer (1200 dimension) and the RNN's recurrent projection layer (512 dimension) is calculated as the representation of the utterance. Then these features are further projected to a 19-dimension subspace by LDA. Gaussian back-end is adopted to get the final scores.
- **NN-frame-alignment model**: The mixture of the UBM is changed to 2227 which is confined by the senone number. The rest configurations are the same with the baseline i-vector model.

The criterion for evaluation is average cost C_{avg} defined by NIST LRE 2015 and can be referred to [15].

5.2. Experimental results

5.2.1. Comparison of DNN and RNN for acoustic modeling

Table 1: *The speech recognition results of DNN and RNN based systems on Hub5'00-SWB. M stands for million.*

system	WER (%)	model size
DNN	18.8	10M
RNN	17.1	7.3M

First, we verify that the LSTM RNN is superior to the DNN for acoustic modeling. The speech recognition results of DNN and RNN based systems on the SWB part of the NIST 2000 Hub5 evaluation set are presented in Table 1. From the results it can be seen that the LSTM RNN model outperforms the DNN approach with a relative improvement of 9.0% in WER.

5.2.2. Results of NN-bottleneck-feature models

Table 2: *Comparison of C_{avg} for i-vector model, DNN-bottleneck-feature model, and RNN-bottleneck-feature models.*

system	Dev	Test
SDC-GMMivector	0.0543	0.288
Fbank-DNN_BN39-GMMivec	0.0350	0.223
FBank-RNN_BN39_L1-GMMivec	0.0431	0.261
FBank-RNN_BN39_L2-GMMivec	0.0419	0.228

The results of i-vector model, DNN-bottleneck-feature model, and RNN-bottleneck-feature model are presented in Table 2. The unit number of bottleneck layer is fixed to 39 for comparison. Bottleneck features extracted from different LSTM recurrent projection layers are evaluated. Both DNN and RNN based models show significant performance improvements over i-vector approach. Bottleneck features extracted from the second projection layer in the LSTM RNN is superior to bottleneck features extracted from the first projection layer. However, there still exists a minor gap between RNN and DNN based models. A reasonable

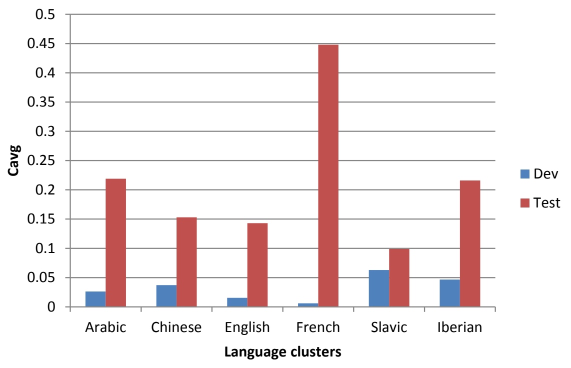

Figure 3: *Comparison of C_{avg} for different language clusters in RNN-bottleneck-feature model.*

explanation might be that the feature extracted from the LSTM recurrent units might not necessarily store information of the current frame. We know that the gates in the LSTM model can help balancing the weights of information flow both from past time (frames) and current time (frame). In some extreme situation the senone predictions might be obtained only based on the information from past time which means that there is no need to pass information from the current frame to the recurrent layer. As a result, using bottleneck features extracted from RNNs might be not accurate enough for frame-level modeling compared with DNNs.

From Table 2 we can see that the results on the development dataset are much better than on the test dataset which means that the development dataset is more similar to the training dataset. To further compare the difference of these two datasets, the performance of each language cluster is shown in Figure 3. The relative trends are almost the same on different models so we use the results obtained from RNN-bottleneck-feature model as an example. From the results we can see that the performance of each language cluster behaves quite different over these two datasets. For example, the French cluster has the best performance in the development dataset but is the worst in the test dataset. The Slavic cluster is exactly the opposite. This data mismatch may result in inaccurate score calibration and other problems and will be analyzed in our later work.

Table 3: *Comparison of C_{avg} for RNN-bottleneck-feature model with different feature processing techniques.*

system	Dev	Test
FBank-RNN_BN39-GMMivec	0.0419	0.228
FBank-RNN_BN39_Whiten-GMMivec	0.0418	0.226
FBank-RNN_BN39_CMVN-GMMivec	0.0417	0.224

Next, we evaluate different feature processing techniques on the RNN based models. The results are presented in Table 3. The results show that feature whitening and CMVN can both improve the performance on the test dataset while the performance on the development dataset remains almost unchanged, which means that these feature processing techniques can make the model generalize better.

The dimension of bottleneck feature is another critical factor that can affect the performance. The results of RNN-bottleneck-feature models with different feature dimension are shown in Table 4. From the results we can see that the performance gets better as the feature dimension increases. CMVN can bring consistent performance improvements as well.

Table 4: *Comparison of C_{avg} for RNN-bottleneck-feature model with different feature dimension.*

system	Dev	Test
Fbank-RNN_BN39-GMMivec	0.0419	0.228
Fbank-RNN_BN60-GMMivec	0.0389	0.224
Fbank-RNN_BN80-GMMivec	0.0377	0.222
Fbank-RNN_BN39_CMVN-GMMivec	0.0417	0.224
Fbank-RNN_BN60_CMVN-GMMivec	0.0387	0.221
Fbank-RNN_BN80_CMVN-GMMivec	0.0378	0.220

5.2.3. *Results of NN-full-feature models*

Table 5: *Comparison of C_{avg} for i-vector model, DNN-full-feature model, and RNN-full-feature models.*

system	Dev	Test
SDC-GMMivec	0.0543	0.288
Fbank-DNN-average	0.0826	0.328
Fbank-RNN_L1-average	0.0808	0.346
Fbank-RNN_L2-average	0.0534	0.283

The results of DNN-full-feature model and RNN-full-feature model are presented in Table 5. The unit number of full feature layer is 1200 in DNN and 512 in RNN. Features extracted from different LSTM layers are evaluated as well. From the results we can see that there's a large performance gap between DNN-full-feature model and i-vector model. However, the RNN-full-feature model that uses features extracted from the second projection layer outperforms i-vector system. This might be attributed to the effectiveness of the LSTM RNN's superiority of capturing long-term temporal dependencies which makes the feature a better representation for the utterance.

5.2.4. *Results of NN-frame-alignment models*

Table 6: *Comparison of C_{avg} for i-vector model, DNN-frame-alignment model, and RNN-frame-alignment model.*

system	Dev	Test
SDC-GMMivec	0.0543	0.288
Fbank-DNNivec	0.0502	0.248
Fbank-RNNivec	0.0739	0.297

The results of DNN-frame-alignment model, and RNN-frame-alignment model are presented in Table 6. Compared with i-vector approach, large performance improvements can be obtained with DNN-frame-alignment model. In addition, the bottleneck feature model is superior to frame alignment model according to Table 2 and Table 6 since the bottleneck feature might be a better choice both for frame alignments and statistics calculation in SLR. Nevertheless, the RNN-frame-alignment performs much worse and is even worse than the baseline i-vector system. Actually, the LSTM RNN can provide more accurate frame alignments than the DNN theoretically according to the results in Table 1 and it can be inferred that the LSTM RNN based system is most likely to outperform the DNN based system. This interesting phenomenon leads us to think that the distribution of senones could be much more complex in some situation and the single Gaussian's hypothesis might not be accurate enough. We'll investigate the distribution of each senone ouput and see whether a mixture model is a better choice for modeling the senone output in our later work.

6. Conclusions

This paper compares several LSTM-RNN based approaches for spoken language recognition. Experiments on the NIST LRE 2015 show that features extracted from LSTM-RNN are effective for SLR. Bottleneck features extracted from LSTM RNNs are competitive to bottleneck features extracted from DNNs. The average of the outputs from the LSTM RNN's recurrent projection layer is an effective feature representation for speech utterances. Nevertheless, the performance of LSTM RNN is much worse than DNN under the frame-alignment framework though the LSTM RNN can provide finer frame alignments theoretically. In the future, we'll further investigate the distribution of each senone output from the LSTM RNN and try different models other than single Gaussian for senone modeling.

7. Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 61273268, No. 61370034 and No. 61403224.

8. References

[1] Haizhou Li, Bin Ma, and Kong Aik Lee, "Spoken language recognition: from fundamentals to practice," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1136–1159, 2013.

[2] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[3] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas A Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction.," in *INTER-SPEECH*, 2011, pp. 857–860.

[4] David Martınez, Oldrich Plchot, Lukás Burget, Ondrej Glembek, and Pavel Matejka, "Language recognition in ivectors space," *Proceedings of Interspeech, Firenze, Italy*, pp. 861–864, 2011.

[5] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[6] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[7] Yan Song, Bing Jiang, YeBo Bao, Si Wei, and L-R Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[8] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phoneetically-aware deep neural network," *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014.

[9] Luciana Ferrer, Yun Lei, Mitchell McLaren, and Nicolas Scheffer, "Study of senone-based deep neural network approaches for spoken language recognition," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 24, no. 1, pp. 105–116, 2016.

[10] Hasim Sak, Andrew W Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling.," in *INTERSPEECH*, 2014, pp. 338–342.

[11] Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno, "Automatic language identification using long short-term memory recurrent neural networks.," in *INTERSPEECH*, 2014, pp. 2155–2159.

[12] Dong Yu and Michael L Seltzer, "Improved bottleneck features using pretrained deep neural networks.," in *Interspeech*, 2011, p. 240.

[13] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3377–3381.

[14] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4052–4056.

[15] "The 2015 nist language recognition evaluation plan," *http://www.nist.gov/itl/iad/mig/lre15.cfm*, 2015.