

INCORPERATING ARTICULATORY VELOCITY INFORMATION IN ACOUSTIC-TO-ARTICULATORY INVERSION

FANG Qiang

Abstract: Conventional acoustic-to-articulatory inversion methods usually train mappings by using maximum likelihood or least square criterion, which assume that all the articulatory channels are equally important. However, different articulatory channels play different roles in speech production. In this paper, to account for this in acoustic-to-articulatory inversion, the importance of each articulatory channel is modeled as an exponential function of its corresponding velocity profile, and incorporated into the conventional least square loss function. The proposed loss function is applied to optimize a batch normalized Deep Neural Network (DNN) for acoustic-to-articulatory inversion. The result indicates that the DNN trained with the proposed cost function outperforms the DNN trained with traditional cost function for most articulatory channels.

Keywords: batch normalized DNN, acoustic-to-articulatory inversion, critical articulator

1. INTRODUCTION

Acoustic-to-articulatory inversion is the technique that estimates vocal tract shape or articulators' position based on input speech signals. It is not only of theoretical interest but also helpful for many applications, such as automatic speech recognition [10], speech therapy and language training [2, 26], talking head animation and lip-syncing [3, 6, 11], and low bit-rate speech coding [19], etc.

Acoustic-to-articulatory inversion was firstly explored by using codebook-based methods. In the codebook-based method, the codebook was built based on a set of

articulatory-acoustic-parameter pairs, where the articulatory parameters were used to control an articulatory model and the corresponding acoustic parameters were generated by an acoustic model based on the vocal-tract shapes obtained by extensively scanning the articulatory parameter space of the articulatory model. Then, articulation was inferred by looking up the codebook [1, 13] according to the acoustic input. The problem of this approach is that same acoustic features could be generated by different combinations of articulatory parameters, and some of them are never used in human speech production. To deal with the non-uniqueness in acoustic-to-articulatory inversion and remove

the invalid results, some researchers introduced dynamic programming [20] as a post-processing procedure or trained inversion models by using human data.

Thanks to the advent of large-scale corpus of synchronized human articulatory-acoustic data, numerous methods have been proposed to tackle the problem of acoustic-to-articulatory inversion in past few decades. It includes a variety of hidden Markov models [5, 9, 12, 18, 27], Kalman filtering [4], support vector regression [21], Gaussian mixture regression [22], codebook [7], multilayer perceptron [14], mixture density network [15], deep neural network [24, 25], trajectory MDN [16], Bi-directional LSTM [28], etc. In addition, several studies tried to incorporate visual features to conduct an audiovisual-to-articulatory mapping [15, 23].

Most methods mentioned above either applied maximum likelihood or least square error criterion to train inversion models, where all the articulatory channels were equally treated. Nevertheless, different articulator plays different role in speech production. The trajectories of some articulators have consistent patterns in various contexts, such as lower lip for bilabials, tongue tip for alveolars, and tongue dorsum for velars. In phonetics, these articulators are called “critical articulators” for those articulations [20]. Hence, in this study, we want to figure out whether better acoustic-to-articulatory inversion performance can be improved by incorporating information of critical articulators into traditional cost functions.

In this study, we depict the information of critical articulators by their velocity profiles, and formulate the cost function as a weighted least square error. In the proposed cost function, the weighting coefficient of each

articulatory channel is represented by an exponential function of its velocity profile.

2. METHOD

In last ten years, DNN became popular due to its success in many fields. It has also been applied to the task of speech inversion [24, 25]. Wu *et al.* [28] tried general linear model, Gaussian mixture model, artificial neural network, and DNN with sigmoid activation function to estimate articulators’ position from synchronized speech features based on an English articulatory-acoustic corpus MNGU0. Their results demonstrated that the DNN achieved the best performance. In past few years, more advanced networks, for example Bi-directional LSTM, have been devised and obtained the state-of-the-art performance [28].

The purpose of this study is to figure out whether articulatory velocity information is helpful for acoustic-to-articulatory inversion, but not to compare the performance of models with different structures. And even for a simple model, the performance could be improved if the incorporated velocity information is helpful. For this reason, we conduct this work with a simple deep neural network rather than with a complicated model.

Equation 1-2 define the basic functions of a neuron in an artificial neural network.

$$I_{(n),i} = \sum_j w_{(n),ij} o_{(n-1),j} + b_{(n),i} \quad (1)$$

$$o_{(n),i} = f(I_{(n),i}) \quad (2)$$

where $I_{(n),i}$ is the input of the i^{th} neuron in the n^{th} layer, $o_{(n-1),j}$ is the output of the j^{th} neuron in the $(n-1)^{th}$ layer, $w_{(n),ij}$ is the weight that connecting the i^{th} unit in the n^{th} layer and the j^{th} unit in the $(n-1)^{th}$ layer, $b_{(n),i}$ is the bias of the i^{th} neuron in the $(n-1)^{th}$ layer, and $f(x)$ is an

activation function (a sigmoid function usually).

The training of a traditional DNN with sigmoid activation function is slow and the performance tends to be affected by gradient vanish problems [8]. And this will slow down the training by requiring lower learning rates and careful parameter initialization, and makes it hard to train DNN with saturating nonlinearities.

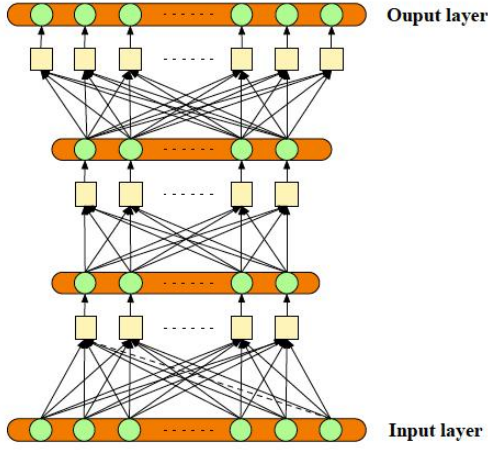


Figure 1: Structure of batch normalized feedforward neural network.

In this study, a batch normalization technique is implemented to perform the normalization for each mini-batch (yellow blocks shown in **Figure 1**). And ‘ReLU’ activation function is used for the neurons in hidden layers. The process of batch normalization is formulated as Eq 3-5:

$$x_{(n),i} = \sum_j w_{(n),ij} o_{(n-1),j} + b_{(n),i} \quad (3)$$

$$\tilde{x}_{(n),i} = \frac{x_{(n),i} - \mu_{(n),i}}{\sqrt{\sigma_{(n),i}^2 + \epsilon}} \quad (4)$$

$$I_{(n),i} = \gamma_{(n),i} \tilde{x}_{(n),i} + \beta_{(n),i} \quad (5)$$

where $\mu_{(n),i}$, $\sigma_{(n),i}^2$ are the mean and variance of $x_{(n),i}$, $\gamma_{(n),i}$ and $\beta_{(n),i}$ are scaling and shifting parameter on normalized value $\tilde{x}_{(n),i}$ so as to keep the representation capability of

the layer. These parameters are optimized with momentum gradient method:

$$\begin{aligned} \mathbf{W}_{n+1} &= \mathbf{W}_n + \Delta \mathbf{W}_{n+1} \\ \Delta \mathbf{W}_{n+1} &= d \cdot \Delta \mathbf{W}_n - \eta \cdot \frac{\partial L}{\partial \mathbf{W}} \end{aligned} \quad (6)$$

$$\begin{aligned} \mathbf{b}_{n+1} &= \mathbf{b}_n + \Delta \mathbf{b}_{n+1} \\ \Delta \mathbf{b}_{n+1} &= d \cdot \Delta \mathbf{b}_n - \eta \cdot \frac{\partial L}{\partial \mathbf{b}} \end{aligned} \quad (7)$$

$$\begin{aligned} \boldsymbol{\gamma}_{n+1} &= \boldsymbol{\gamma}_n + \Delta \boldsymbol{\gamma}_{n+1} \\ \Delta \boldsymbol{\gamma}_{n+1} &= d \cdot \Delta \boldsymbol{\gamma}_n - \eta \cdot \frac{\partial L}{\partial \boldsymbol{\gamma}} \end{aligned} \quad (8)$$

$$\begin{aligned} \boldsymbol{\beta}_{n+1} &= \boldsymbol{\beta}_n + \Delta \boldsymbol{\beta}_{n+1} \\ \Delta \boldsymbol{\beta}_{n+1} &= d \cdot \Delta \boldsymbol{\beta}_n - \eta \cdot \frac{\partial L}{\partial \boldsymbol{\beta}} \end{aligned} \quad (9)$$

where L is the loss over the training set, d is the momentum, and η is the learning rate. The partial derivatives $\frac{\partial L}{\partial \mathbf{W}}$, $\frac{\partial L}{\partial \mathbf{b}}$, $\frac{\partial L}{\partial \boldsymbol{\gamma}}$, $\frac{\partial L}{\partial \boldsymbol{\beta}}$ of each layer can be calculated by using backpropagation algorithm (shown in Equ 10-16).

$$\frac{\partial L}{\partial \mathbf{W}_{(n)}} = \frac{1}{m} \sum_{l=1}^m \mathbf{o}_{(n-1)}^{(l)} \left(\frac{\partial L^{(l)}}{\partial \mathbf{x}_{(n)}^{(l)}} \right)^T \quad (10)$$

$$\frac{\partial L}{\partial \mathbf{b}_{(n)}} = \frac{1}{m} \sum_{l=1}^m \frac{\partial L^{(l)}}{\partial \mathbf{x}_{(n)}^{(l)}} \quad (11)$$

$$\frac{\partial L}{\partial \boldsymbol{\gamma}_{(n),i}} = \frac{1}{m} \sum_{l=1}^m \frac{\partial L^{(l)}}{\partial I_{(n),i}^{(l)}} \tilde{x}_{(n),i}^{(l)} \quad (12)$$

$$\frac{\partial L}{\partial \boldsymbol{\beta}_{(n),i}} = \frac{1}{m} \sum_{l=1}^m \frac{\partial L^{(l)}}{\partial I_{(n),i}^{(l)}} \quad (13)$$

$$\frac{\partial L^{(l)}}{\partial \mathbf{x}_{(n)}^{(l)}} = \frac{\partial I_{(n)}^{(l)}}{\partial \mathbf{x}_{(n)}^{(l)}} \frac{\partial L^{(l)}}{\partial I_{(n)}^{(l)}} \quad (14)$$

$$\frac{\partial L^{(l)}}{\partial I_{(n-1)}^{(l)}} = \frac{\partial \sigma_{(n-1)}^{(l)}}{\partial I_{(n-1)}^{(l)}} \mathbf{W}_{(n)} \frac{\partial I_{(n)}^{(l)}}{\partial \mathbf{x}_{(n)}^{(l)}} \frac{\partial L^{(l)}}{\partial I_{(n)}^{(l)}} \quad (15)$$

$$\frac{\partial I_{(n),i}^{(l)}}{\partial \mathbf{x}_{(n),i}^{(l)}} = \gamma_{(n),i} \frac{\partial \tilde{x}_{(n),i}^{(l)}}{\partial \mathbf{x}_{(n),i}^{(l)}} \quad (16)$$

where $L^{(l)}$ is the loss over the l^{th} example, m is the number training examples in each mini-batch, $\mathbf{x}_{(n-1)}^{(l)}$ is an input of the batch

normalization blocks of the $(n-1)^{th}$ layer, $I_{(n),i}^{(l)}$ is the input to the i^{th} neuron of the n^{th} layer, $\tilde{x}_{(n),i}^{(l)}$ is a batch normalized output of the i^{th} batch normalization block of the n^{th} layer, $\mathbf{o}_{(n)}$ is the output of the n^{th} layer, $\mathbf{W}_{(n)}$ is the connecting weight that link neurons in the $(n-1)^{th}$ layer and the n^{th} layer, and $\mathbf{b}_{(n)}$ is the bias of the n^{th} layer.

3. EXPERIMENT

The MOCHA-TIMIT database, which has synchronized acoustic-articulatory signals, is used in the experiment. The corpus recorded the synchronized acoustic-articulatory data of one male speaker (msak0) and one female speaker (fsew0). Each of them produced 460 TIMIT sentences. In the database, the sampling rates are 16,000Hz for acoustic signal and 500Hz for articulatory signal, respectively.

When collecting the corpus, electromagnetic receiver coils were attached to 9 articulators in the midsagittal plane, as shown in Figure 2. They were velum (V), tongue dorsum (TD), tongue blade (TB), tongue tip (TT), lower jaw (LJ), upper lip (UL), lower lip (LL), and the references (REF) on nose ridge and upper jaw. For each articulator, the horizontal and vertical coordinates were recorded. As a result, 18 channels of articulatory trajectories were recorded in total. Among those coils, the trajectories of V, TD, TB, TT, LJ, LL and UL depict the movement of articulators, and are used in our experiment

The data of the male speaker (msak0) is used in this paper. When preparing the experiment, speech utterances are segmented into frames with a Hanning window. Each frame contains a speech segment of 25ms, and is encoded as a vector of

log-energy+12-order-MFCCs+delta+delta-deltas. The frame shift between two consecutive frames is 10ms.

As for the articulatory data, a Savitzky-Golay filter with the order of 3 and frame size of 21 is applied to remove the high-frequency noise in articulatory trajectories. Then, the smoothed EMA data are down-sampled to 100Hz to match the frame rate of the acoustic features.

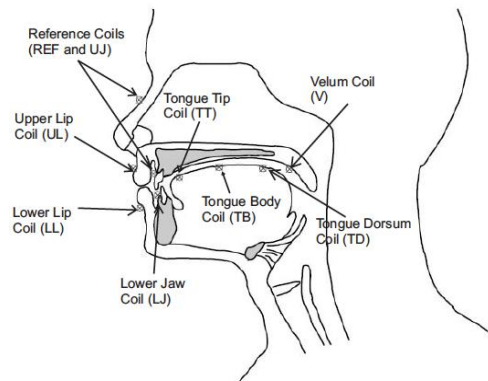


Figure 2: The positions of the EMA coils on the speaker's articulators.

In our experiment, the acoustic features of every 7 consecutive frames are concatenated to represent the input vector of the DNN. As for the output, the EMA data corresponding to the 4th frame in the corresponding contextual acoustic feature vectors is used. Both EMA and acoustic feature vectors are normalized by subtracting their global mean and dividing by their standard deviation.

In the experiment, the database is randomly partitioned into three subsets: a validation set (45 utterances), a testing set (45 utterances), and a training set (370 utterances).

To measure the performance of acoustic-to-articulatory inversion, root mean-squared error (RMSE) and correlation coefficient are adopted. Here, RMSE gives an indication of the average distance between two

trajectories, while correlation coefficient indicates synchrony and similarity between two trajectories. They are defined in Eq17-18:

$$\text{rmse} = \sqrt{\frac{1}{m} \sum_{i=1}^m (\hat{x}_i - x_i)^2} \quad (17)$$

$$c = \frac{\sum_{i=1}^k (x_i - \bar{x})(\hat{x}_i - \bar{\hat{x}})}{\sqrt{\sum_{i=1}^k (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^k (\hat{x}_i - \bar{\hat{x}})^2}} \quad (18)$$

where \hat{x}_i are x_i the estimate and ground truth at instant i , respectively.

Two cost functions are formulated for training the batch normalized DNN, where L_1 is the least square loss and L_2 is a weighted version of L_1 .

$$L_1 = \frac{1}{2} \sum_{i=1}^m (\hat{x}^i - x^i)^T (\hat{x}^i - x^i) \quad (19)$$

$$L_2 = \frac{1}{2} \sum_{i=1}^m (\hat{x}^i - x^i)^T \mathbf{W} (\hat{x}^i - x^i) \quad (20)$$

$$w_{ij} = \begin{cases} \frac{-1}{2\sigma_i^2} v_i^2, & i = j \\ 0, & \text{else} \end{cases} \quad (21)$$

where the weighting coefficient w_{ij} depends on the velocity v_i .

4. RESULT

To determine the number of hidden units in each layer, we conduct an experiment on a neural network with just one hidden layer. In the experiment, the number of hidden units varies from 50 to 1600. The result indicates that the neural network with 400 hidden units achieves the best performance. With reference to this result, we construct a DNN with 6 hidden layers, of which each hidden layer contains 400 neurons. And ‘‘ReLU’’ activate function is applied for neurons in hidden layers.

In the experiment, the momentum d in Equation 6-9 is 0.8, the initial learning rate is 0.0004 and decays with the proportion of 0.9.

Each mini-batch contains 1024 examples. And the maximum number of training epoch is 50.

The RMSE is calculated for the trajectory of each articulatory channel. As shown in Table 1, the first two columns are the RMSEs of the DNN trained with cost functions L_1 and L_2 , respectively. The last two columns are the corresponding correlation coefficients between the ground truth and the estimated trajectories, respectively. The bold italic numbers in the second and fourth column indicate that the RMSE/correlation coefficients between the estimate and the ground truth is reduced/improved.

Table 1: Experiment results.

	<i>rms-L₁</i>	<i>rms-L₂</i>	<i>c-L₁</i>	<i>c-L₂</i>
<i>Vx</i>	0.5186	0.5218	0.9474	0.9466
<i>Vy</i>	0.8845	0.8903	0.8662	0.8629
<i>TDx</i>	1.3495	1.3285	0.8898	0.8957
<i>TDy</i>	1.5108	1.4733	0.8672	0.8758
<i>TBx</i>	1.3870	1.3633	0.9097	0.9128
<i>TBy</i>	1.5342	1.5281	0.8973	0.8986
<i>TTx</i>	1.6221	1.6462	0.9033	0.9048
<i>TTy</i>	1.7540	1.7378	0.9180	0.9200
<i>LJx</i>	0.4249	0.4220	0.8618	0.8609
<i>LJy</i>	0.6762	0.6784	0.9093	0.9102
<i>ULx</i>	0.5405	0.5392	0.7836	0.7826
<i>ULy</i>	0.7679	0.7695	0.8985	0.8975
<i>LLx</i>	0.9085	0.9039	0.8791	0.8798
<i>LLy</i>	1.4086	1.3756	0.9204	0.9240
<i>Avg.</i>	1.092	1.0841	0.8894	0.8909

The performance of 11 of the 14 articulatory channels are improved for either RMSE or correlation. As shown in Table 1, the RMSEs of 9 articulatory channels decrease, and the correlation coefficients increase also for 9 articulatory channels. There are 7 articulatory channels whose performance are improved in the sense of both RMSE and correlation coefficient.

The overall average RMSE and correlation coefficient are 1.092mm and 0.8894 respectively when L_1 cost function is used. The performance is better than the those reported based on a MOCHA database [13, 28]. The overall average RMSE and correlation coefficient are 1.0841mm and 0.8909 respectively when L_2 cost function is used. The results are better on both closeness and shape similarities between estimated trajectory and ground truth.

5. CONCLUSION

In this paper, an articulatory velocity profile weighted cost function is proposed to optimize a DNN for acoustic-to-articulatory inversion. The result indicates that the performance of acoustic-to-articulatory inversion could be improved by using a weighting function based on velocity profile in most cases even a very simple DNN is used. One can expect that the performance can be further improved by incorporating better weighting function into the cost function.

Acknowledgement

This work is supported by the National Natural Science-Foundation of China (No.61977049), Advanced Innovation Center for Language Resource and Intelligence (KYR17005), National Major Social Sciences Foundation of China (15ZDB103), and Innovation Program of Chinese Academy of Social Science.

References

- [1] Atal B. S., Chang J. J., Mathews M. V., and Tukey J. W. 1978. Inversion of articulatory-to-acoustic transformation in the vocal tract by a computer-sorting technique. *J. Acoust. Soc. Am.* vol. 63, no. 5, pp. 1535-1555.
- [2] Badin P., Tarabalka Y., Elisei F., and Bailly G. 2010. Can you ‘read’ tongue movements? Evaluation of the contribution of tongue display to speech understanding. *Speech Communication* vol. 52, no. 6, pp. 493-503.
- [3] Bregler C., Covell M., and Slaney M. 1997. Video rewrite: Driving visual speech with audio. *SIGGRAPH1997*.
- [4] Dusan S. 2000. *Statistical estimation of articulatory trajectories from the speech signal using dynamical and phonological constraints*. Ph.D thesis, Dept. of Electrical and Computer Engineering, University of Waterloo.
- [5] Hiroya S. and Honda M. 2004. Estimation of Articulatory Movements from Speech Acoustics Using an HMM-Based Speech Production Model. *IEEE Transactions on Speech and Audio Processing* vol. 12, no. 2, pp. 175-185.
- [6] Hofer G. and Richmond K. 2010. Comparison of HMM and TMDN methods for lip synchronisation. *InterSpeech2010*.
- [7] Hogden J., Lofqvist A., Gracco V., Zlokarnik I., Rubin P., and Saltzman E. 1996. Accurate recovery of articulator positions from acoustics: New conclusions based on human data. *J. Acoust. Soc. Amer.* vol. 100, pp. 1819-1834.
- [8] Ioffe S. and Szegedy C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML2015*.
- [9] Katsamanis A., Papandreou G., and Maragos P. 2009. Face active appearance modeling and speech acoustic information to recover articulation. *IEEE Trans. Acoust.,*

- Speech, and Language Processing* vol. 17, no. 3, pp. 411–422.
- [10] King S., Frankel J., Livescu K., McDermott E., Richmond K., and Wester M. 2007. Speech production knowledge in automatic speech recognition. *J. Acoust. Soc. Am.* vol. 121, no. 2, pp. 723-742.
- [11] Lewis J. 1991. Automated lip-sync: Background and techniques. *The Journal of Visualization and Computer Animation* vol. 2, no. 4, pp. 118–122.
- [12] Ling Z., Richmond K., and Yamagishi J. 2010. An Analysis of HMM-based prediction of articulatory movements. *Speech Communication* vol. 52, no. 10, pp. 834-846.
- [13] Ouni S. and Laprie Y. 2005. Modeling the articulatory space using a hypercube codebook for acoustic-to-articulatory inversion. *J. Acoust. Soc. Amer.* vol. 118, no. 1, pp. 444–460.
- [14] Papcun G., Hochberg J., Thomas T. R., Laroche F., Zachs J., and Levy S. 1992. Inferring articulation and recognising gestures from acoustics with a neural network trained on X-ray microbeam data. *J. Acoust. Soc. Am.* vol. 92, no. 2, pp. 688–700.
- [15] Richmond K. 2002. *Estimating articulatory parameters from the acoustic speech signal*. Ph.D. thesis, University of Edinburgh.
- [16] Richmond K. 2007. Trajectory mixture density networks with multiple mixtures for acoustic-articulatory inversion. in *Lecture Notes in Computer Science* vol. 4885, M. Chetouani, A. Hussain, B. Gas, M. Milgram, and J. L. Zarader Eds.: Springer-Verlag Berlin, pp. 263–272.
- [17] Richmond K. 2006. A trajectory mixture density network for the acoustic-articulatory inversion mapping. *InterSpeech2006*.
- [18] Roweis S. 1999. *Data driven production models for speech processing*. Ph.D, California Institute of Technology.
- [19] Schroeter J. and Sondhi M. M. 1992. Speech coding based on physiological models of speech production. in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi Eds. New York: Marcel Dekker Inc., pp. 231–268.
- [20] Schroeter J. and Sondhi M. M. 1994. Techniques for estimating vocal-tract shapes from the speech signal. *IEEE Trans. Speech Audio Processing* vol. 2, pp. 133-150.
- [21] Toutios A. and Margaritis K. 2008. Contribution to statistical acoustic-to-EMA mapping. *InterSpeech2008*.
- [22] Toda T., Black A. W., and Tokuda K. 2008. Statistical mapping between articulatory movements and acoustic spectrum using a Gaussian mixture model. *Speech Communication* vol. 50, pp. 215–227.
- [23] Toutios A. and Ouni S. 2011. Predicting tongue positions from acoustics and facial features. *InterSpeech2011*.
- [24] Uría B., Murray I., Renals S., and Richmond K. 2012. Deep Architectures for Articulatory Inversion. *InterSpeech2012*.
- [25] Wu Z., Zhao K., Wu X., Lan X., and Meng H. 2015. Acoustic to articulatory mapping with deep neural network. *Multimed Tools Appl* vol. 74, no. 22, pp. 9889-9907.
- [26] Youssef A. B., Hueber T., Badin P., and Bailly G. 2011. Toward a multi-speaker visual articulatory feedback system. *InterSpeech 2011*.
- [27] Zhang L. and Renals S. 2008. Acoustic-articulatory modeling with the trajectory HMM. *IEEE Signal Process Letter* vol. 15, pp. 245–248.

- [28] Zhu P., Xie L., and Chen Y. 2015. Articulatory Movement Prediction Using Deep Bidirectional Long Short-Term Memory Based Recurrent Neural Networks and Word/Phone Embeddings. *Interspeech2015*.

FANG Qiang PhD He is an associate professor of Institute of Linguistics, Chinese Academy of Social Sciences. His research interests include speech production and modeling.

E-mail: fangqiang@cass.org.cn

[本文原载《中国语音学报》第15辑，2021年]