TTS Synthesis with Bidirectional LSTM based Recurrent Neural Networks

Yuchen Fan [1,2*], *Yao Qian* [2], *Fenglong Xie* [2], *Frank K. Soong* [2]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] Microsoft Research Asia, Beijing, China

fyc0624@sjtu.edu.cn, {yaoqian, v-fxie, frankkps}@microsoft.com

Abstract

Feed-forward, Deep neural networks (DNN)-based text-to-speech (TTS) systems have been recently shown to outperform decision-tree clustered context-dependent HMM TTS systems [1, 4]. However, the long time span contextual effect in a speech utterance is still not easy to accommodate, due to the intrinsic, feed-forward nature in DNN-based modeling. Also, to synthesize a smooth speech trajectory, the dynamic features are commonly used to constrain speech parameter trajectory generation in HMM-based TTS [2]. In this paper, Recurrent Neural Networks (RNNs) with Bidirectional Long Short Term Memory (BLSTM) cells are adopted to capture the correlation or co-occurrence information between any two instants in a speech utterance for parametric TTS synthesis. Experimental results show that a hybrid system of DNN and BLSTM-RNN, i.e., lower hidden layers with a feed-forward structure which is cascaded with upper hidden layers with a bidirectional RNN structure of LSTM, can outperform either the conventional, decision tree-based HMM, or a DNN TTS system, both objectively and subjectively. The speech trajectory generated by the BLSTM-RNN TTS is fairly smooth and no dynamic constraints are needed.

Index Terms: statistical parametric speech synthesis; hidden Markov model; deep neural network; recurrent neural network; Bidirectional LSTM

1. Introduction

Statistical Parametric TTS Synthesis system has many advantages, e.g. the flexibility to change the voice characteristics, small footprint, and robustness, over other systems [1]. The above mentioned advantages have made the parametric, GMM-HMM based TTS synthesis the main stream technology for many, particularly hand-held device, speech synthesis applications [2]. A parametric HMM is effective to model the evolution of speech signals as a stochastic sequence of acoustic feature vectors. Many techniques have been developed for HMM-based speech recognition, e.g. context-dependent modeling, state-tying based on decision tree clustering, and speaker adaptation. They have been applied equally well to HMM-based TTS for speech parameter trajectory generation. The trajectory thus generated with trained HMMs is fairly smooth and very rarely results in concatenation glitches, which occur occasionally in unit-selection synthesis. However, the overly smoothed parameter trajectories, due to statistical average in HMM training, still tend to make synthesized speech sound not as lively as desired.

Recently, motivated by the success of DNN in speech recognition, a few DNN research attempts have been tried in order to improve the performance of vocoder-based speech synthesis. Zen, et al. [3,23] comprehensively listed the limitations of the conventional HMM-based approach, e.g. decision-tree based contextual state clustering, and proposed to use DNN to overcome these limitations for speech synthesis. It shows DNN based approach, which models the relationship between input texts and their corresponding acoustic features, can outperform the HMM-based approach with a similar number of model parameters. Qian, et al. [4] further examined the DNN based TTS synthesis with a moderate size corpus more commonly used for parametric TTS training, on the training aspects, e.g., activation functions and weights initialization in "pre-training". Lu, et al. [5] employed phone, letter and Vector Space Model (VSM) based binary or continuous features as input features, and frame or state for output predictions in DNN based synthesis. Deep Belief Network (DBN) with stacked, restricted Boltzmann machines (RBMs), which can model the structure in the input data as generative "pre-training" and find a region of the weight-space that can reduce over-fitting for the discriminative "fine-tuning" phase in speech recognition [6], is also employed to model joint distribution of linguistic and acoustic features for speech synthesis [7]. In addition, RBM is directly used to represent the distribution of the spectral envelops at each HMM state in [8], where the mode of RBM has been shown to perform better than the mean of GMM and results in better synthesized voice quality.

In DNN-based speech synthesis, DNN simulates human speech production by a layered hierarchical structure to transform linguistic text information into its final speech output. The limitation of decision-tree based contextual state clustering, which is used to tie states of long contexts into generalized ones to predict unseen contexts in testing better than the HMM-based counterpart, is overcome in DNN. DNN also can represent high dimensional and correlated features efficiently and model highly complex mapping function compactly. However, DNN is still suffering from using short unit, e.g., state or frame, as basic modelling unit. To capture co-articulation effect and mimic natural intonation, very rich contexts are used as input features. To synthesize smooth speech speech parameter trajectories, the dynamic model parameters are required to be used together with their static counterparts to generate smooth parameter trajectories [2].

In this paper, we investigate how to use Recurrent Neural Networks (RNN), especially with bidirectional Long Short Term Memory (LSTM) cells [9-13], which in principle can capture information from anywhere in the feature sequence, as a generation model for TTS synthesis. RNN can be unfolded into a wide structure in time scale by taking the previous or following hidden states. Like DNN, RNN can also be stacked together in multiple layers and to have deep structure in space.

[*] Work performed as an intern in the Speech Group, Microsoft Research Asia

2. Deep Bidirectional LSTM (DBLSTM) Recurrent Neural Network

Recurrent Neural Network (RNN) computes hidden state vector sequence $\boldsymbol{h} = (h_1, \cdots, h_T)$ and outputs vector sequence $\boldsymbol{y} = (y_1, \cdots, y_T)$, for a given input vector sequence $\boldsymbol{x} = (x_1, \cdots, x_T)$, iterating the following equations from $t = 1$ to T:

$$h_t = \mathcal{H}(\boldsymbol{W}_{xh}x_t + \boldsymbol{W}_{hh}h_{t-1} + b_h) \qquad (1)$$

$$y_t = \boldsymbol{W}_{hy}h_t + b_y \qquad (2)$$

where $\boldsymbol{W}$ is the weight matrices, e.g. $\boldsymbol{W}_{xh}$ is the weight matrix between input and hidden vectors; b is the bias vectors, e.g. b_h is the bias vector for hidden state vectors; and $\mathcal{H}$ is the nonlinear activation function for hidden nodes.

$\mathcal{H}$ is usually a sigmoid or hyperbolic tangent function in the conventional RNNs, but the gradient vanishing problem caused by these activation function prevents RNN from modeling the long-span relations in sequential features. Long short term memory (LSTM) network [11], shown in Fig. 1, which manually build a memory cell inside, can overcome the problems in conventional RNN and can model signals that have a mixture of low and high frequency components. For LSTM, $\mathcal{H}$ is implemented with the following functions [12]:

$$i_t = \sigma(\boldsymbol{W}_{xi}x_t + \boldsymbol{W}_{hi}h_{t-1} + \boldsymbol{W}_{ci}c_{t-1} + b_i) \qquad (3)$$

$$f_t = \sigma\left(\boldsymbol{W}_{xf}x_t + \boldsymbol{W}_{hf}h_{t-1} + \boldsymbol{W}_{cf}c_{t-1} + b_f\right) \qquad (4)$$

$$\mathrm{c}_t = f_t c_{t-1} + i_t tanh(\boldsymbol{W}_{xc}x_t + \boldsymbol{W}_{\mathrm{hc}}h_{t-1} + b_c) \qquad (5)$$

$$o_t = \sigma(\boldsymbol{W}_{xo}x_t + \boldsymbol{W}_{ho}h_{t-1} + \boldsymbol{W}_{co}c_t + b_o) \qquad (6)$$

$$\mathrm{h}_t = \mathrm{o}_t tanh(\mathrm{c}_t) \qquad (7)$$

where σ is the sigmoid function; i, f, o and c are input gate, forget gate, output gate and cell memory, respectively.

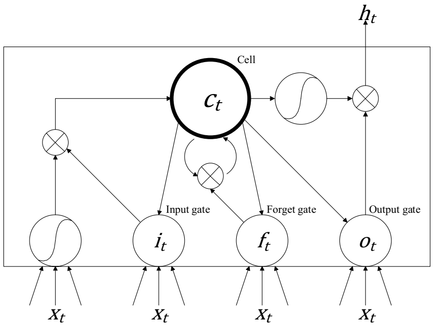

Fig.1. Long Short Term Memory

Bidirectional RNN [13] as shown in Fig. 2 can access both the preceeding and succeeding contexts. It separates the hidden layer into two parts, forward state sequence, $\overrightarrow{h}$, and backward state sequence, $\overleftarrow{h}$. The iterative process is:

$$\overrightarrow{h}_t = \mathcal{H}\left(\boldsymbol{W}_{x\overrightarrow{h}}x_t + \boldsymbol{W}_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right) \qquad (8)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(\boldsymbol{W}_{x\overleftarrow{h}}x_t + \boldsymbol{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right) \qquad (9)$$

$$y_t = \boldsymbol{W}_{\overrightarrow{h}y}\overrightarrow{h}_t + \boldsymbol{W}_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \qquad (10)$$

Deep bidirectional RNN can be established by stacking multiple RNN hidden layers on top of each other. Each hidden state sequence, h^n, is replaced by the forward and backward, $\overrightarrow{h}^n$ and $\overleftarrow{h}^n$, and the iterative process is:

$$\overrightarrow{h}_t^n = \mathcal{H}\left(\boldsymbol{W}_{\overrightarrow{h}^{n-1}\overrightarrow{h}^n}\overrightarrow{h}_t^{n-1} + \boldsymbol{W}_{\overrightarrow{h}^n\overrightarrow{h}^n}\overrightarrow{h}_{t-1}^n + b_{\overrightarrow{h}}^n\right) \qquad (11)$$

$$\overleftarrow{h}_t^n = \mathcal{H}\left(\boldsymbol{W}_{\overleftarrow{h}^{n-1}\overleftarrow{h}^n}\overleftarrow{h}_t^{n-1} + \boldsymbol{W}_{\overleftarrow{h}^n\overleftarrow{h}^n}\overleftarrow{h}_{t+1}^n + b_{\overleftarrow{h}}^n\right) \qquad (12)$$

$$y_t = \boldsymbol{W}_{\overrightarrow{h}^N y}\overrightarrow{h}_t^N + \boldsymbol{W}_{\overleftarrow{h}^N y}\overleftarrow{h}_t^N + b_y \qquad (13)$$

Deep bidirectional LSTM (DBLSTM) is the integration of deep bidirectional RNN and LSTM. By taking the advantages of DNN and LSTM, it can model the deep representation of long-span features.

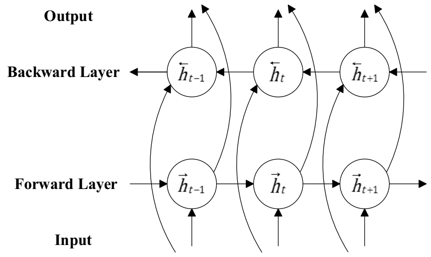
Output
Backward Layer
Forward Layer
Input
$\overleftarrow{h}_{t-1}$
$\overleftarrow{h}_t$
$\overleftarrow{h}_{t+1}$
$\overrightarrow{h}_{t-1}$
$\overrightarrow{h}_t$
$\overrightarrow{h}_{t+1}$

Fig.2. Bidirectional RNN

3. DBLSTM-RNN based TTS Synthesis

Speech production can be seen as a process to select spoken words, formulate their phonetics and then finally articulate output speech with the articulators. So it is a continuous physical dynamic process. DBLSTM-RNN can simulate human speech production by a layered hierarchical and wide in time scale structure to transform linguistic text information into its final speech output. In a TTS synthesis system, where usually a whole sentence is given as input, there is no reason not to access long-range context in both forward and backward directions. We propose to use DBLSTM-RNN for TTS synthesis. The schematic diagram of DBLSTM-RNN based TTS synthesis is shown in Fig. 3.

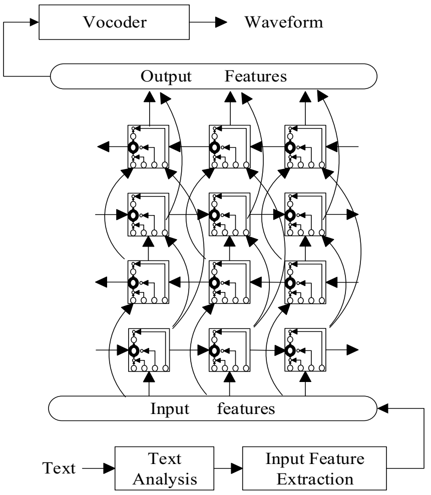

Fig.3. DBLSTM-RNN based TTS synthesis

In DBLSTM-RNN based TTS synthesis, rich contexts are also used as input features, which contain the binary features for categorical contexts, e.g. phone labels, POS labels of the current word, and TOBI labels, and numerical features for the numerical contexts, e.g., the number of words in a phrase or the position of the current frame of the current phone. The output features are acoustic features like spectral envelope and fundamental frequency. Input features and output features can be time-aligned, frame-by-frame by a well-trained HMM model. RNN is also powerful to make it possible to model sequential data where input-output alignment is unknown by Connectionist Temporal Classification [14] and Sequence Transduction [15]. The weights of DNN are trained by using pairs of input and output features extracted from training data to minimize the errors between the mapped output from the given input and the target output.

In RNN training, the training criterion is to minimize the mean square error between the output features and the ground-truth. Back-propagation through time (BPTT) is the most frequently used algorithm for training RNN. BPTT first unfolds the RNN into feed-forward network through time, and then training the unfolded network with back-propagation. For deep bidirectional LSTM, BPTT algorithm is applied to both forward and backward hidden nodes, and back-propagates layer by layer. In DNN training, the weights are trained by back-propagation procedure with a "mini-batch" based stochastic gradient descent algorithm, which select mini-batches of frames randomly from the whole training set. In RNN training, the weight gradients are computed over the entire utterance. In order to parallelize and speed up, tens of utterances are randomly selected for each updating, then used to update the weights of RNN simultaneously.

In synthesis, the input text is converted first into input feature vector through the text analysis, then input feature vectors are mapped to output vectors by a trained DBLSTM-RNN. In HMM-based TTS, the corresponding contextual label is used to access the decision tree to get the contextual HMM state sequence. The corresponding means and covariance of HMM states are fed into the parameter generation module to generate smooth speech parameter trajectories with the dynamic information. In DNN-based TTS, by setting the predicted output features from the DNN as mean vectors and pre-computed (global) variances of output features from all training data, the speech feature generation module can generate smooth trajectories of speech parameter features which satisfy the statistics of static and dynamic features, and voiced/unvoiced flag is determined by an empirical threshold of DNN prediction. Considering the power of RNN in modelling sequential problems, the parameter generation module is implicitly embedded inside our proposed DBLSTM-RNN based TTS synthesis, i.e., the output features of RNN are only the static features: spectral envelop, gain, fundamental frequency and U/V decision and then directly be fed into a vocoder to synthesize final speech waveform.

The effective learning capability of DBLSTM-RNN is expected to benefit TTS speech synthesis. Deep-layered architectures can represent long-span, highly-complex function (transformation) compactly [16]. The functions can be compactly represented with a *k* level deep architecture which can outperform a shallow architecture but with more weights. A decision tree, which is generally used in HMM-based TTS for clustering the similar context-dependent state into tied states, is such a shallow architecture. Both RNN and DNN can also overcome the limitation of conditional independence assumption in HMM, i.e., all observations are dependent only upon the states that generate them. However, DNN only can model the relationship between text and speech on the frame level, take the finite phone or HMM state as input, and generate speech through the transition over the finite states. With the same finite states as input, RNN can take the information from neighboring frames, get different hidden states from the same input and break through the limits from input finite states.

4. Experiments

4.1. Experimental Setup

A corpus of a female, American English, native speaker, both phonetically and prosodically rich, is used in our experiments. The corpus consists of 5,000 training utterances (around 5 hours) and 200 extra utterances are used for testing. Speech signals are sampled at 16 kHz, windowed by a 25-ms window, and shifted every 5-ms. An LPC of 40th order is transformed into static LSPs and their dynamic counterparts. The phonetic and prosodic contexts include quin-phone, the position of a phone, syllable and word in phrase and sentence, the length of word and phrase, stress of syllable, TOBI and POS of word.

In the baseline HMM-based TTS, five-state, left-to-right HMM phone models, where each state is modeled by a single Gaussian, diagonal covariance, output distribution, are adopted. The phonetic and prosodic contexts are used as a question set in growing the decision trees. Minimum description length (MDL) criterion [17] for balancing model complexity and training data size is used as a stopping criterion for state clustering in decision tree growing. HMM parameters are first trained in the Maximum Likelihood (ML) sense and then refined by the minimum generation error (MGE) training. It adjusts HMM parameters trained by the conventional EM algorithm to minimize the generation error between synthesized and original parameter trajectories of the training data [18].

In the DNN-based TTS, the input feature vector contains 355 dimensions, where 319 are binary features for categorical linguistic contexts and the rest are numerical linguistic contexts. The output feature vector contains a voiced/unvoiced flag, log *F0*, LSP, gain, their dynamic counterparts, totally 127 dimensions. Voiced/unvoiced flag is a binary feature that indicates the voicing of the current frame. An exponential decay function [19] is used to interpolate F0 in unvoiced speech regions. 80% of silence frames are removed from the training data to balance the training data and to reduce the computational cost [3]. Removing silence frames in DNN training was found useful for avoiding DNN overlearning silence label in speech recognition task. Both input and output features of training data are normalized to zero mean and unity variance. The weights are trained by back-propagation procedure with a "mini-batch" based stochastic gradient descent algorithm.

In the DBLSTM-RNN-based TTS, the input feature vector is the same as the one in DNN-based TTS, while the output feature vector only contains voiced/unvoiced flag, log *F0*, LSP and gain without the dynamic counterparts, totally 43 dimensions. We can also only use current frame information as input features without the contextual information. Silence frames are not removed to keep the continuity of acoustic features within a sentence for RNN training. CURRENT [21], a machine learning library for RNN, is used in our experiments.

For the testing utterances, both HMM and DNN outputs are firstly fed into a parameter generation module to generate smooth feature parameters with dynamic feature constraints, RNN outputs can skip this procedure. Then formant sharpening based on LSP frequencies [24] is used to reduce the over-smoothing problem of statistic parametric modeling and the resultant “muffled” speech. Finally speech waveforms are synthesized by an LPC synthesizer by using generated speech parameters.

The computational cost of training DBLSTM-RNN is much higher than that of training DNN, despite taking the advantage of GPGPU. Given the same number of hidden layers and same number hidden nodes per layer, the number of model parameters in DBLSTM-RNN is also much larger than that of DNN. In our experiments, we replace one or two top layers of DNN with BLSTM-RNN structure to keep roughly the same number of model parameters as the HMM and DNN models. The preliminary results on a corpus with 1,000 training utterances show that the deeper structures of DBLSTM-RNN don’t always get better performance even with “layer-wise BP” pre-training [20] in the training. We conjecture that the inherent depth both in time and space of DBLSTM-RNN leads to an imprecise gradient decent computation, particularly when the training data is not large enough.

The configurations of model training for HMM, DNN and DBLSTM-RNN based TTS systems are listed as following:

1) HMM: MDL=1 for both LSP and F0 decision tree growing
2) DNN_A: 6 hidden layers with 512 nodes per layer
3) DNN_B: 3 hidden layers with 1024 nodes per layer
4) Hybrid_A: a hybrid of DNN and BLSTM-RNN. 4 hidden layers with 512 nodes per layer, where the bottom 3 hidden layers are feed-forward structure with sigmoid activation functions, while the top hidden layer is Bidirectional RNN structure with LSTM (256 forward nodes and 256 backward nodes)
5) Hybrid_B: the hybrid structure as Hybrid_A, but with 2 lower hidden layers with sigmoid activation functions and 2 upper hidden layers with BLSTM-RNN (256 forward nodes and 256 backward nodes)

4.2. Evaluation Results and Analysis

Objective and subjective measures are used to evaluate the performance of three TTS systems on the test data. Synthesis quality is measured objectively in terms of distortions between natural test utterances of the original speaker and the synthesized speech where oracle state durations (obtained by forced alignment) of natural speech are used. The objective measures are F0 distortion in root mean squared error (RMSE), voiced/unvoiced (V/U) swapping errors and normalized spectrum distance in log spectral distance (LSD). The subjective measure is an AB preference test between speech sentence pairs synthesized by two chosen systems.

The results of objective measures of different configurations in HMM, DNN and Hybrid trainings are listed in Table 1. For HMM training, we set MDL=1 as a stopping criterion for state clustering in both LSP and F0 decision tree growing. According to our previous training experience, an MDL parameter which is larger or smaller results in worse objective measures when HMM parameters are firstly trained in the ML sense and then refined by the MGE training. For DNN training, we found 3 or 6 hidden layers with 512 or 1024 nodes for each layers seem no much changes on all three objective measures [4]. By comparing the results of Hybrid system with those of HMM and DNN, Hybrid system can achieve a best performance on spectra, i.e., the LSD of natural and generated spectra trajectories by Hybrid system is improved by over 0.1 dB. The RMSE of natural and generated F0 trajectories by Hybrid is on par with that by DNN.

Table 1. The results of objective measures of different configurations (the number of system parameters) in HMM, DNN and Hybrid trainings

Measures Model	LSD (dB)	V/U Error rate	F0 RMSE (Hz)
HMM (2.89M)	3.74	5.8%	17.7
DNN_A (1. 55M)	3.73	5.8%	15.8
DNN_B (2.59 M)	3.73	5.9%	15.9
Hybrid_A (2.30M)	3.61	5.7%	16.4
Hybrid_B (3.61M)	3.54	5.6%	15.8

The performance of HMM, DNN and Hybrid systems are further subjectively evaluated by perceptual tests. 50 utterances, which are randomly selected from the testing set and synthesized by the best baseline HMM system (MDL=1), DNN system (DNN_B) and Hybrid systems (Hybrid_A and Hybrid_B), are evaluated in three AB preference tests participated by 60 subjects through Amazon Mechanical Turk [22]. Each subject evaluates 50 pairs by using headsets. There are three choices: 1) the former is better; 2) the latter is better; 3) no preference or neutral (The difference between the paired sentences cannot be perceived or can be perceived but difficult to choose which one is better). The preference scores are shown in Fig. 4. It shows the speech synthesized by the Hybrid system is significantly preferred than the best HMM and DNN systems (at $p < 0.001$ level). Its preference scores (59% and 55%) are higher than the HMM system (22%) and the DNN system (20%). (Some samples of synthesized utterances are given through http://research.microsoft.com/en-us/projects/dnntts/default.aspx)

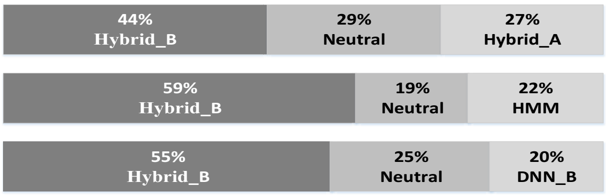

Fig. 4. The preference scores of the HMM, DNN and Hybrid systems

5. Conclusions

In this paper, we investigate the use of BLSTM-RNN to perform train statistical parametric model for TTS speech synthesis. To keep the similar number of model parameters, only the nodes in upper one or two hidden layers of DNN are replaced with bidirectional LSTM RNN. Experimental results show that Hybrid BLSTM-RNN and DNN system performs better than those of HMM and DNN due to its ability to capture deep information in a sentence. In the future, we will try to investigate DBLSTM-RNN with an even deeper structure with a larger corpus.

6. References

[1] H. Zen, K. Tokuda, and W. Black, Alan, "Statistical parametric speech synthesis", Speech Communication, Volume 51, Issue 11, pp. 1039-1064, 2009.

[2] K. Tokuda, T. Kobayashi, T. Masuko, T. Kobayashi, and T. Kitamura,, "Speech parameter generation algorithms for HMM-based speech synthesis", In Proc. ICASSP, pp. 1315-1318, 2000.

[3] H. Zen, A. Senior and M. Senior, "Statistical Parametric Speech Synthesis Using Deep Neural Networks", In Proc. ICASSP, pp. 8012-8016, 2013.

[4] Y. Qian, Y.-C. Fan, W.-P. Hu and F. K. Soong, "On the training aspects of deep neural network (DNN) for parametric TTS synthesis", to be published in Proc. ICASSP, 2014.

[5] H. Lu, S. King, and O. Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis", In 8th ISCA Workshop on Speech Synthesis, pp. 281-285, 2013.

[6] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "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] S. Kang, X. Qian, and H. Meng, "Multi-distribution deep belief network for speech synthesis", In Proc. ICASSP, pp. 7962-7966, 2013.

[8] Z.-H. Ling, L. Deng, and D. Yu, "Modeling spectral envelopes using restricted Boltzmann machines for statistical parametric speech synthesis", In Proc. ICASSP, pp. 7825-7829, 2013.

[9] A. Graves, A.-R Mohamed, and G. Hinton. "Speech recognition with deep recurrent neural networks." In Proc. ICASPP, pp.6645-6649, 2013.

[10] A. Graves, N. Jaitly, and A.-R Mohamed. "Hybrid speech recognition with Deep Bidirectional LSTM." In IEEE ASRU, pp.273-278, 2013.

[11] H. Sepp, S. Jürgen, "Long short-term memory." Neural computation, vol.9, no.8, pp. 1735-1780, 1997.

[12] A. Gers, N. Schraudolph, and S. Jürgen. "Learning precise timing with LSTM recurrent networks." The Journal of Machine Learning Research vol.3, pp. 115-143, 2003.

[13] S. Mike, K. Paliwal. "Bidirectional recurrent neural networks." IEEE Transactions on Signal Processing, vol.45, no.11, pp.2673-2681, 1997.

[14] A. Grave, S. Fernandez., F. Gomez, and J. Schmidhuer, "Connectionist temporal classification: labelling un-segmented sequence data with recurrent neural networks," in ICML, Pittsburgh, USA, 2006.

[15] A. Grave, "Sequence transduction with recurrent neural networks," in ICML Representation Learning Workshop, 2012.

[16] Y. Bengio, "Learning deep architectures for AI," Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009.

[17] K. Shinoda, and T. Watanable, "MDL-based Context-Dependent Sub-word Modeling for Speech Recognition", J. Acoust. Soc. Jpn(E), vol.21, no.2, pp.79-86, 2000.

[18] Y.-J. Wu and R. H. Wang, "Minimum generation error training for HMM-based speech synthesis", In Proc. ICASSP, 2006.

[19] C. J. Chen, R. A. Gopinath, M. D. Monkowski, M. A. Picheny, and K. Shen, "New methods in continuous Mandarin speech recognition.," in EUROSPEECH. 1997, ISCA.

[20] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in IEEE ASRU, 2011.

[21] http://sourceforge.net/projects/currennt/?source=navbar

[22] https://www.mturk.com/mturk/welcome

[23] H. Zen, "Deep Learning in Speech Synthesis," ISCA SSW8, http://research.google.com/pubs/archive/41539.pdf, 2013.

[24] Z.-H. Ling, Y.-J. Wu, Y.-P. Wang, L. Qin, and R.-H. Wang, "USTC System for Blizzard Challenge 2006 an Improved HMM-based Speech Synthesis Method," Proc. Blizzard Challenge 2006 Workshop, 2006.