Prosody Contour Prediction with Long Short-Term Memory, Bi-Directional, Deep Recurrent Neural Networks

Raul Fernandez[1], *Asaf Rendel*[2], *Bhuvana Ramabhadran*[1], *Ron Hoory*[2]

[1]IBM TJ Watson Research Center, Yorktown Heights, NY – USA
[2]IBM Haifa Research Lab, Haifa – Israel

fernanra@us.ibm.com, asafren@il.ibm.com

Abstract

Deep Neural Networks (DNNs) have been shown to provide state-of-the-art performance over other baseline models in the task of predicting prosodic targets from text in a speech-synthesis system. However, prosody prediction can be affected by an interaction of short- and long-term contextual factors that a static model that depends on a fixed-size context window can fail to properly capture. In this work, we look at a recurrent formulation of neural networks (RNNs) that are deep in time and can store state information from an arbitrarily large input history when making a prediction. We show that RNNs provide improved performance over DNNs of comparable size in terms of various objective metrics for a variety of prosodic streams (notably, a relative reduction of about 6% in F0 mean-square error accompanied by a relative increase of about 14% in F0 variance), as well as in terms of perceptual quality assessed through mean-opinion-score listening tests.

Index Terms: speech synthesis,text-to-speech, prosody prediction, recurrent neural networks, deep learning

1. Introduction

Prosody prediction in text-to-speech (TTS) systems that is perceived as being natural and expressive to the task remains a challenging obstacle. Although unit-selection systems are able to reproduce the inherent natural prosody of the segments, they often sound discontinuous, or produce prosody whose short-term naturalness does not reflect the long-term structure of the input. Parametric models, on the other hand, can produce smoother prosody over a long span of text, but fail to reproduce the range of expressiveness observed in natural speech. There are at least two reasons why the interaction between text and prosody is hard to model in TTS systems. One is that the features available to the model do not typically include the rich semantic, syntactic, and pragmatic information speakers tacitly make use of when producing speech. A second reason is that these features interact in a rather complex, dynamic way with contextual effects from neighboring and distant inputs bearing on the prosodic realization at any given time. In this work, we try to address this second issue by considering Recurrent Neural Networks (RNNs), a class of models equipped to model the dynamic dependency of the outputs on the inputs by accumulating information from a non-fixed "time window" both from the past and the future. We motivate the modeling approach in Section 3 by first reviewing a deep-layer neural architecture that has been previously shown to provide state-of-the-art results for prosody prediction, and then consider an alternative architecture in Section 4 that provides depth both across layers and temporally, thus allowing better modeling of the objective function. We then evaluate the model in Section 5 both in terms of objective metrics and perceptual listening tests.

2. Modeling Approach

We treat prosody prediction as a (weighted) regression problem, where a set of text-derived input features are used to predict a target vector $\mathbf{y}$ consisting of the 7 required statistics needed by the standard TTS parameter-generation algorithm for a Hidden Markov Model (HMM) parametric synthesizer with output Gaussian distributions (that is, state-level mean and standard deviation for the logF0, Δ-logF0 and $\Delta\Delta$-logF0 streams, and phone-level mean for the log duration stream; see [1]):

$$\mathbf{y} = [\mu_{f_0}, \sigma_{f_0}, \mu_{\Delta}, \sigma_{\Delta}, \mu_{\Delta\Delta}, \sigma_{\Delta\Delta}, \mu_{dur}]^T. \quad (1)$$

These targets are defined at the state level for a 3-state HMM, with the phone durations replicated state-wise to allow modeling duration and $F0$ jointly. The front-end module of a TTS engine is used to extract a set of text-based features commonly used for prosody prediction (e.g., categorical labels such as phonetic identity or part of speech, and counts which might reflect surface prosodic structure such as number of phones/syllables/words to a phrase/sentence boundary, etc.). Context is considered at the phone- and word-level by extending the feature-set with categorical features of the previous and next phone/word respectively. All categorical features are encoded using One-of-N codes, and features defined above the state level are propagated down to the constituent states and paired with the targets described above to create the input-output pairs $\{\mathbf{x}_n, \mathbf{y}_n\}$ to train and validate the system. Since some of the values in the target vector (1) may not be naturally "defined" (e.g., $F0$ in voiceless phones for which we still need to predict a duration), we associate with every observation n a binary weight vector $\mathbf{w}$ of the same dimensionality as the target, and use this to control which streams (and for which observations) should contribute to the loss function (and its gradient):

$$\ell = \sum_n (\mathbf{y}_n - f(\mathbf{x}_n;\theta))^T W_n (\mathbf{y}_n - f(\mathbf{x}_n;\theta)), \quad (2)$$

where $f(\mathbf{x};\theta)$ is a parametric regression function with parameter vector θ, and $W_n = \mathrm{diag}(\mathbf{w}_n)$. In the rest of this work, the weighting scheme is as follows: voiceless states (as determined by a pitch tracker) and silences receive a weight of 0 for the $F0$ streams; initial and trailing silences receive a weight of 0 for the duration stream. All other observations (including utterance-medial pauses/silences) receive a weight of 1.

3. Deep Neural Networks

Deep Neural Networks (DNNs) have recently received much attention for the gains they bring to some classical machine-learning tasks, such as image classification, and acoustic modeling for speech recognition. In the synthesis area, we have seen reported gains over traditional baseline systems (typically Decision Trees) by several authors ([2, 3, 4, 5]) making them, to date, a new standard for prediction in TTS systems. In this work, we adopt as our baseline a 3-layer DNN architecture (with 512, 256, and 256 hidden units in each layer respectively) with sigmoid non-linearities in the hidden layers, and a linear unit on the output layer. The model is trained following the common two-pass approach for training deep networks: (i) an unsupervised, layer-wise pre-training pass using Restricted Boltzmann Machines for each layer, followed by (ii) a supervised end-to-end update of the weights using stochastic gradient descent to minimize the loss function in (2) and the back-propagation algorithm to evaluate the network derivatives [2]. We also consider multi-task learning strategies where auxiliary co-targets are used in the target vector during learning to help improve the performance of the main targets of interest, and then discarded at run time. Such techniques can manage to increase the prediction of the targets of interest, at the expense of a very small increase in the number of parameters due to the auxiliary co-targets [6]. After empirically investigating various configurations, we adopt the following two-network strategy: A network DNN_μ is trained with a 4-D μ target vector containing the 3 F0 state-level means plus the phone duration and used at run-time to generate the μ streams. Separately, a network DNN_σ is trained with the full 7-D target, and used at run time to generate the σ streams. For DNN_σ, the μ streams therefore serve as an auxiliary task to help predict the σ streams, and are discarded at run-time. This strategy that was found to yield improved metrics on an independent development set over a single-network strategy that predicts all targets at once (or other configurations involving different partitions of the targets), and is therefore the baseline adopted here.

4. Recurrent Neural Networks

Although DNNs are able to offer improvements over other baseline prosody-prediction models (such as trees), they still fall short of reproducing the naturalness and range observed in natural speech. One possible shortcoming is that they are typically trained using a "localized" window of input patterns, i.e., the current output is predicted from the current input, plus possibly a few fixed-length adjacent input observations to provide context. Unless one is willing to allow for a very large contextual window (an approach that would lead to data sparsity issues unless very large amounts of data are available), this approach would fail to capture long-term dependencies along an utterance (or across utterances, when full paragraphs are available for training). Such non-local dependencies are arguably one of the factors contributing to surface prosody; a prosody-prediction model that managed to exploit such dependencies could in principle lead to better prosodic realizations. A Recurrent Neural Network (RNN) is a class of models that has been proposed as an alternative to address the challenge posed by time series that have complex contextual dependencies that go beyond a fixed time lag. They have been shown to offer state-of-the-art performance in such tasks as handwriting recognition [7] and phoneme recognition [8]. Unlike DNNs, RNNs are able to offer depth across time by making their hidden-unit activations

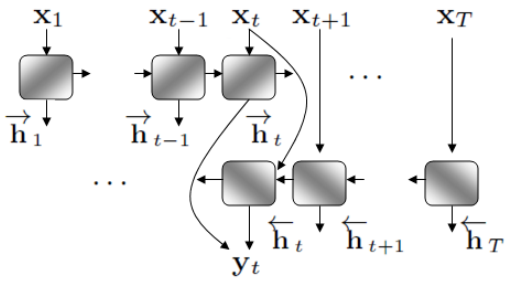

Figure 1: *A bidirectional recurrent neural network unfolded over time. The prediction at time t is a function of the outputs from 2 hidden layers, each recursively accumulating a state from all past and future inputs respectively.*

not just a function of the inputs, but of an internal state that is updated as a function of the entire input history, as shown in Fig. 1 for a structure unrolled over time. A bi-directional RNN, moreover, allows for two independent layers to accumulate contributions from the past and future histories when making a prediction at time t. This type of structure is theoretically well motivated for such a task as prosody prediction, where planning-ahead effects, particularly those more noticeably observed in the case of carefully read speech, can influence the current prosodic realization [9]. Finally, these bi-directional, deep-in-time structures can be stacked to allow for more complex models that are also deep across layers and can, analogously to simple DNNs, progressively extract structure from the successive layer compositionality.

4.1. The Long Short-Term Memory Architecture

Recurrent neural networks are known to be difficult to train. Some of the well-known issues in training a deep neural network, such as gradients vanishing across several layers during back-propagation, are only exacerbated when a dynamic version of back-propagation (across many more time lags) is used. To remedy this, a modified architecture known as a Long Short-Term Memory (LSTM) was introduced in [10] (and later revised in [11] and [12]). The basic building block of an LSTM network is a compound cell that implements continuous versions of memory storage operations. Of the variants described in the literature, we make use of the extended LSTM cell with peephole connections introduced in [12], and shown in Fig 2. This cell implements 3 subtasks via 3 gates known as the *input*, *forget*, and *output* gates, which determine, respectively, (i) when an input is relevant enough to the task to be remembered, (ii) how long it should remember or forget such an input, and (iii) when it should output a value. Each LSTM cell also contains a recurrently self-connected linear unit called the "Constant Error Carousel" (CEC) which recirculates activation and error signals for as long as there is an input history, and is thus able to provide short-term memory storage for extended time periods. Finally, peephole connections provide feedback from the cell to the gates, allowing the gates to carry out their operations as a function of both the incoming inputs and the previous state of the cell. In mathematical terms, the LSTM unit implements the following compound recursive function $\phi_f()$ to obtain the

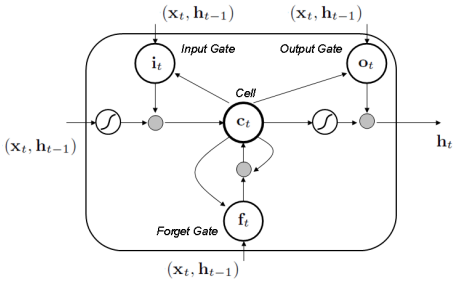

Figure 2: *LSTM gate with peephole connections, showing the internal structure and relation between the cell and the different gates. Solid nodes are multipliers. The mathematical operations carried out by this unit are detailed in Eqs. 3- 7*

activations $\mathbf{h}_t = \phi_f(\mathbf{x}_t, \mathbf{h}_{t-1})$ [8]:

$$\mathbf{i}_t = \sigma(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \quad (3)$$

$$\mathbf{f}_t = \sigma(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \quad (4)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tau(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \quad (5)$$

$$\mathbf{o}_t = \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{co}\mathbf{c}_t + \mathbf{b}_o) \quad (6)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tau(\mathbf{c}_t), \quad (7)$$

where $\mathbf{i}$, $\mathbf{f}$, $\mathbf{c}$ and $\mathbf{o}$ are the input gate, forget gate, cell gate, and output gate activation vectors respectively, all of the same size as the hidden vector $\mathbf{h}$; $W_{x\star}$ are the input-to-gate weights; $W_{h\star}$ are the hidden-to-hidden weights for the various gates; $W_{c\star}$ are the peephole weights; $\mathbf{b}_\star$ are the bias vectors; $\odot$ is the element-wise multiplication operator; $\sigma()$ is a function bound by $(0, 1)$ (sigmoid in this implementation) realizing a differentiable version of a "hard" gating function; and $\tau()$ is a differentiable non-linearity (hyperbolic tangent in this work). These sample equations are defined for a cell on the forward chain; an analogously defined $\phi_b(\mathbf{x}_t, \mathbf{h}_{t+1})$ would operate on the backward chain. Deep recurrent bi-directional structures may be built by stacking several bi-directional layers and feeding the activations of one set of layers in the stack to the next. Letting $\overrightarrow{\mathbf{h}}^n$ and $\overleftarrow{\mathbf{h}}^n$ represent the activations of the forward and backward chains on the n^{th} level of a stack, then

$$\overrightarrow{\mathbf{h}}^n_t = \phi_f(\mathbf{z}^n_t, \overrightarrow{\mathbf{h}}^n_{t-1}) \text{ and } \overleftarrow{\mathbf{h}}^n_t = \phi_b(\mathbf{z}^n_t, \overleftarrow{\mathbf{h}}^n_{t+1}), \quad (8)$$

where $\mathbf{z}^0_t = \mathbf{x}_t$ and $\mathbf{z}^n_t = [\overrightarrow{\mathbf{h}}^{n-1}_t; \overleftarrow{\mathbf{h}}^{n-1}_t]$ for $n \geq 1$, and the final linear-unit output is

$$\mathbf{y}_t = W_{\overrightarrow{h}\,y}\overrightarrow{\mathbf{h}}^N + W_{\overleftarrow{h}\,y}\overleftarrow{\mathbf{h}}^N + \mathbf{b}_y. \quad (9)$$

The task of learning consists of adjusting the weights associated with these gates so as to know when to carry out their respective storage, reset and output operations. Learning algorithms such as steepest descent (SD) or resilient back-propagation (RPROP) can then be used to train the parameters [13]. For the experiments reported here, we have used 3 stacks of bi-directional LSTM layers, with 67, 57, and 46 LSTM units per layer respectively in each hidden layer, and a linear output layer, leading to a network with 478,647 parameters (a size comparable to the DNN structures, which contained 479,236 and 480,007 parameters for the DNN_μ and DNN_σ models respectively). We again experimented with multi-task learning schemes, but found that a single RNN model for all targets yielded better development-set metrics than the various target-dedicated models we explored with DNNs.

5. Evaluation

The approaches just described were evaluated on two different voices, from two different female speakers of North-American English: the first voice (F1) contained about 7 hours of speech read in a business-news style whereas the second voice (F2) consisted of approximately 10 hours of speech from various genres and domains. Each corpus was divided into independent training, development, and test sets with 80%/10%/10% of the data respectively. The corpora were phonetically aligned with 3-state HMMs, and the F0 contours extracted using a 5-msec. frame rate. First- and second-order delta sequences were computed using the operators $\delta_1[y(n)] = 0.5y(n+1) - 0.5y(n-1)$ and $\delta_2[y(n)] = y(n+1) - 2y(n) + y(n-1)$, and state-level mean (and standard deviation) statistics for all 3 F0 sequences aggregated using the state-level alignments. Phone-level durations were extracted from the alignments, and the target vectors formed as previously described.

The DNN models were trained using the two-pass approach already covered. For the reported experiments, the networks are discriminatively trained using online steepest descent with mini-batches of 1000 states, until the best development-set loss fails to improve for 40 continuous iterations.

The RNN models were trained using a two-pass approach. First, an online steepest-descent algorithm (with updates after processing each sequence) runs until the best development-set loss fails to improve for 15 iterations. In the second phase, a batch RPROP algorithm is run on the previously obtained RNN, updating the weights once per epoch, until the best development-set loss does not increase for15 iterations. We have seen this refinement offer a relative improvement on the dev-set loss in the range of 1% to 2% over the first pass. We have used the LSTM implementation available as part of RNNLib Toolkit [14] for the RNN experiments reported here.

Tables 1 and 2 show the mean and standard deviation of 3 different objective metrics (averaged over 5 models trained from random initializations) computed on the test set for each of the voices and models, as well as the relative gains for each metric. The metrics are weighted mean-square error (WMSE), cross-correlation between the predictions and targets (XCORR), and the normalized variance (VARN) indicating the ratio between the predicted variance and the natural variance of the (test-set) targets (as the predictions never exceed the variance of the targets, this number is typically bounded by 1). The relative gains are defined in such a way that a positive sign corresponds to a reduction in WMSE, and to an increase in XCORR and VARN. We can see that the RNN model never under-performs the DNN model in terms of WMSE and XCORR. More importantly, there seems to be a noticeable improvement in the predicted variance of the various $F0$ streams (a property that can have perceptual implications), and this relative increase in predicted variance comes at no cost to the WMSE metric, in contrast to training techniques that explicitly trade-off one for the other [15].

To investigate the perceptual effects of these quantitative gains, we carried out a listening test where subjects were asked to rate on a 5-point scale their overall impression of samples randomly drawn from these systems. Since the DNN model provides a strong baseline for prosody prediction (and in previous evaluations has been shown to provide state-of-the-art performance), there is a question of how much improvement one can obtain over this base system when only prosody is allowed to change. To address this question, we also included in the test samples where the $F0$ and duration contours have been transplanted from natural sentences (the spectral sequences, in all

Table 1:
Objective metrics and relative improvements for the F1 Voice

		WMSE	XCORR	NVAR
F0	DNN	4.009e-2±2.281e-4	0.553±2.653e-3	0.380±9.634e-3
	RNN	3.773e-2±2.600e-4	0.592±4.650e-3	0.437±3.587e-2
	Relative Gain	**5.89%**	**7.05%**	**15.00%**
ΔF0	DNN	3.319e-4±1.278e-6	0.440±3.641e-3	0.420±1.640e-2
	RNN	3.213e-4±5.500e-7	0.469±8.742e-4	0.489±1.959e-2
	Relative Gain	**3.19%**	**6.59%**	**16.43%**
ΔΔF0	DNN	2.397e-4±8.24e-8	0.158±1.765e-3	0.372±2.434e-2
	RNN	2.386e-4±2.70e-7	0.171±2.270e-3	0.431±2.069e-2
	Relative Gain	**0.46%**	**8.23%**	**15.86%**
DUR	DNN	6.277e-2±4.314e-4	0.881±9.362e-4	0.854±1.113e-2
	RNN	6.220e-2±1.840e-3	0.883±1.650e-3	0.872 ±1.529e-2
	Relative Gain	**0.91%**	**0.23%**	**2.11%**

Table 2:
Objective metrics and relative improvements for the F2 voice

		WMSE	XCORR	NVAR
F0	DNN	1.172e-2±4.350e-5	0.758±1.002e-3	0.596±4.561e-3
	RNN	1.099e-2±7.550e-5	0.776±1.527e-3	0.673±5.551e-3
	Relative Gain	**6.23%**	**2.37%**	**12.91%**
ΔF0	DNN	5.295e-5±1.001e-7	0.663±8.106e-4	0.470±8.236e-3
	RNN	4.963e-5±1.992e-7	0.691±1.373e-3	0.555±5.106e-3
	Relative Gain	**6.27%**	**4.22%**	**18.08%**
ΔΔF0	DNN	7.300e-6±2.280e-9	0.425±2.584e-4	0.195±2.602e-3
	RNN	7.169e-6±1.984e-8	0.444±1.737e-3	0.245±5.309e-3
	Relative Gain	**1.79%**	**4.47%**	**25.64%**
DUR	DNN	0.1054±1.750e-4	0.800±4.456e-4	0.656±7.101e-3
	RNN	0.1040±3.527e-4	0.803±7.532e-4	0.668±2.266e-3
	Relative Gain	**1.33%**	**0.37%**	**1.83%**

cases, are synthetic, and identical for all 3 systems). Having the transplanted prosody allows us to establish a measure of an upper threshold, as far as prosody is concerned, and to discover where the predictions fall with respect to this upper threshold.

The two voices were evaluated in two separate tests using a Mechanical Turk set-up [16]. In each test, 40 distinct texts from each corpus's test set were synthesized using DNN and RNN models, and those samples combined with the transplanted-prosody samples for a 3-system comparison. For each voice and model, of the five systems whose average test-set metrics are reported above, the one with the lowest development-set loss was chosen to generate the listening test samples. Fifty participants took part in each test, and an automatic procedure was used to eliminate unreliable listeners. The mean opinion score of the systems for each voice are summarized on Table 3. As we can see, there is less than a quarter of a point between the prosody of the baseline (DNN) system and what we can hope to gain from perfectly natural prosody. To put the performance of the RNN model in this context, the last column shows the percentage of the baseline-to-natural gap covered by the RNN models.

Table 3: *Results of listening test: Mean-Opinion Scores for DNN (baseline), RNN and transplanted-natural-prosody (XPLT) models. The last column shows the percent of the baseline-to-natural prosody gap covered by the RNN model.*

Voice	DNN	RNN	XPLT	Rel.
F1	3.49	3.56	3.69	35.00%
F2	3.76	3.84	3.93	47.06%

6. Conclusions

The prediction of prosodic contours from textual input features is a sample task in which short- and long-term configurations of various input factors contribute to the surface acoustic realization one needs to generate for a parametric speech synthesizer. Although DNNs currently provide state-of-the-art performance for prosody prediction, their formulation inherently fails to address the complex way in which inputs can interact over varying time lags. In this work we have addressed this limitation by investigating the performance of a deep bi-directional recurrent neural architecture with LSTM activation units, and shown that it helps improve various objective metrics over a strong baseline DNN system for different prosodic streams, notably improving the variance of the contours without sacrificing the mean-square fit. We have furthermore verified that such gains bear perceptual improvements by closing 35% and 47% of the gap between the baseline system and the "ceiling" benchmark provided by transplanted natural prosody for two different synthetic voices.

7. Acknowledgments

The authors wish to thank Vincent Pollet and Milan Legat for their assistance with the listening tests and data analysis.

8. References

[1] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *ICASSP*, 2000, pp. 1315–1318.

[2] R. Fernandez, R. Rendel, B. Ramabhadran, and R. Hoory, "F0 contour prediction with a Deep Belief Network-Gaussian Process hybrid model," in *ICASSP*, 2013, pp. 6885–6889.

[3] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using Deep Neural Networks," in *ICASSP*, 2013, pp. 7962–7966.

[4] Z.-H. Ling, L. Deng, and D. Yu, "Modeling spectral envelops using Restricted Boltzmann Machines and Deep Belief Networks for statistical parametric speech synthesis," *IEEE Trans. Audio, Speech, and Lang. Proc.*, vol. 21, no. 10, pp. 2129–2139, 2013.

[5] S. Kang, X. Qian, and H. Meng, "Multi-distribution Deep Belief Networks for speech synthesis," in *ICASSP*, 2013, pp. 8012–8016.

[6] M. L. Seltzer and J. Droppo, "Multi-task learning in Deep Neural Networks for improved phoneme recognition," in *Proc. ICASSP*, 2013, pp. 6965–6969.

[7] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional Recurrent Neural Networks," in *NIPS*, 2009.

[8] A. Graves, M. Abdel-rahman, and G. Hinton, "Speech recognition with Deep Recurrent Neural Networks," in *ICASSP*, 2013, pp. 6885–6889.

[9] S. Shattuck-Hufnagel, "Phrase-level phonology in speech production planning: Evidence for the role of prosodic structure," in *Prosody: Theory and Experiment. Studies Presented to Gösta Bruce*, ser. Text, Speech and Language Technology, G. Bruce and M. Horne, Eds. Netherlands: Springer, 2000, vol. 14, pp. 201–229.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] F. A. Gers, J. Schmidhuber, and F. Cummings, "Learning to forget: Continual prediction with LSTM," *Neural Computaiton*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM Recurrent Networks," *J. of Machine Learning Research*, vol. 3, pp. 115–143, 2002.

[13] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Intnl. Conf. on Neural Networks*, 1993, pp. 586–591.

[14] A. Graves, "RNNLIB: A Recurrent Neural Network library for sequence learning problems," http://sourceforge.net/projects/rnnl/.

[15] T. Toda and K. Tokuda, "Speech parameter generation algorithm considering global variance for HMM-based speech synthesis," in *Interspeech*, 2005, pp. 2801–2804.

[16] F. Ribeiro, D. Florêncio, C. Zhang, and M. Seltzer, "CROWDMOS: An approach for crowdsourcing Mean Opinion Score studies," in *ICASSP*, 2011, pp. 2416–2419.