Multi-output RNN-LSTM for multiple speaker speech synthesis and adaptation

Santiago Pascual, Antonio Bonafonte
Universitat Politècnica de Catalunya
Barcelona, Spain
santiago.pascual@tsc.upc.edu, antonio.bonafonte@upc.edu

***Abstract*—Deep Learning has been applied successfully to speech processing. In this paper we propose an architecture for speech synthesis using multiple speakers. Some hidden layers are shared by all the speakers, while there is a specific output layer for each speaker. Objective and perceptual experiments prove that this scheme produces much better results in comparison with single speaker model. Moreover, we also tackle the problem of speaker adaptation by adding a new output branch to the model and successfully training it without the need of modifying the base optimized model. This fine tuning method achieves better results than training the new speaker from scratch with its own model.**

I. Introduction

Deep Learning has been applied successfully to different kinds of tasks such as computer vision, natural language processing or speech processing [3], outperforming the existing systems in many cases. In the case of speech synthesis, many works included Deep Neural Networks (DNN) and Deep Belief Networks (DBN) to perform acoustic mappings and prosody prediction [21], [7], [17], Also, Recurrent Neural Networks (RNNs) and their variants, like the Long Short Term Memory (LSTM) architecture [10], have leveraged completely the sequences processing and prediction problem, which makes them lead to interesting results in the speech synthesis field, where an acoustic signal of variable length has to be generated out of a set of textual entities. Some example works using this structures can be seen in [6], [22], [8], [20]. Previous to deep learning, existing text to speech technologies included the unit selection speech synthesis [12] and the statistical parametric speech synthesis (SPSS) [23]. Unit selection analyzed the set of phonemes contained in a sentence and their context, and those features were mapped into pieces of recorded natural speech, all being concatenated to produce a continuous stream of voice signal. SPSS introduced the concept of learning a speaker model from data with parametric representations and then throw away the data once speaker characteristics were learned. Some remarkable differences between both was that, although SPSS could not reproduce the same level of naturalness [23] as unit selection did, it had much less footprint in memory, and it also let the user transform any speaker model to adapt the voice to different requirements in speed, pitch, etc. An important feature of SPSS was then the speaker adaptation technique, in which we could add the voice of someone that was not previously in the system, and with few data the model could reproduce the newcomer speech.

In this paper we want to propose an approach to tackle two problems with a single RNN-LSTM model: making multiple speaker models out of the same structure, and make speaker adaptation with new data on top of this model. Therefore, we wanted a system capable of holding many speaker models inside the same shared structure, so that every user shares its characteristics with the others, thus reducing the required number of parameters per user and letting them interact in the lower layers. There was a proposal of a similar approach by [5] with DNNs performing multi-task learning, but in our case we work with RNN architectures, with a different training procedure and also with a different speaker adaptation architecture. The structure of this work is the following; in the next section we make a brief introduction about the RNN-LSTM model. Then in section III we describe our proposed model, followed by an explanation of the experimental setup made in section IV. Sections V and VI cover the results and conclusions respectively, where we analyze the response of our model to different questions we make about its properties.

II. Review of Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a special type of Neural Network topology well suited for processing sequences. When we talk about a unidirectional recurrent layer we can say that it has memory about the past, so at time t they have an input vector $\boldsymbol{x_t} \in \mathbb{R}^n$ and the memory state at time $t-1$ $\boldsymbol{h_{t-1}} \in \mathbb{R}^m$, producing the new memory state $\boldsymbol{h_t}$, also called hidden state, with the following set of operations:

$$\boldsymbol{h_t} = g(\boldsymbol{W} \cdot \boldsymbol{x_t} + \boldsymbol{U} \cdot \boldsymbol{h_{t-1}} + \boldsymbol{b}) \tag{1}$$

where $\boldsymbol{W}$ is the input-to-hidden weights matrix, $\boldsymbol{U}$ is the hidden-to-hidden weights matrix where the feedback is made, $\boldsymbol{b}$ is the bias vector and g is a specified element-wise non-linear transformation, such as the hyperbolic tangent. As we can see, for every input sequence $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_T}\}$ we obtain an output sequence $\boldsymbol{H} = \{\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_T}\}$, where each output from the layer keeps track of dynamic changes in time, and this is what makes the recurrent model a really powerful option for sequences.

In this work we used LSTM layers, as they cope better with the vanishing gradient problems [9] that appeared when training regular RNNs. They also model the long term dependencies in a better way than the simple RNNs do because of their gating mechanisms [10].

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

III. Proposed Architecture

The proposed architecture is depicted in Figure 1. There are two first feed forward layers serving as a bottleneck for the sparse inputs. These intend to get a deeper knowledge about the input data, which is formed by a mixed set of multiple types of features that will be presented in section IV. There is a first LSTM hidden layer, processing every transformed input set of features at each time step, and deriving the results to the output branches. Dropout [19] is performed between the hidden recurrent layer and the output layers to mitigate any over-fitting caused by the low amount of data available. Each output branch belongs to a different speaker, so at prediction time we inject the linguistic parameters to the model to obtain every speaker's speech parameters at the output.

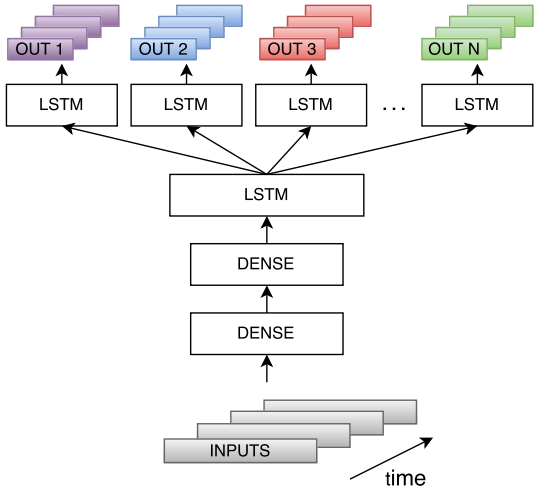

Fig. 1. Proposed architecture using regular feed forward (dense) layers and recurrent LSTM layers. There are N outputs belonging to N different speakers.

Every output branch is independent of each other, so it propagates its own error through the whole shared structure at training time without the need of padding any data for the other outputs. The specifics of the training method will be discussed in section IV. Note that output layers are also recurrent, so that dynamic features are not computed because feedback connections within the layer keep track of the dynamic evolution of outputs [22].

The intuition behind this architecture is that, whilst every output branch is trained, it shares the first linguistic mappings with other branches. This might lead to an improvement in the final acoustic mapping of every speaker in comparison to the speaker model trained in an isolated manner, because we add more information during training time to get to correlated predictions at the different outputs.

IV. Experimental Setup

We worked with six voices from the TCSTAR [2] project, where four of them contain expressive speech, and two neutral voices from the Interface database [11]. We balanced the data per user, such that all of them have approximately the same amount of samples to train, i.e. 20 minutes of speech per speaker. There are four male voices (M1, M2, M3, M4) and four female voices (F1, F2, F3, F4). The F3 data is separated from the other ones because it is used for the speaker adaptation experiment. We will focus the results on the M1, F1 and F3 speakers, all of them having 4 minutes of samples for testing and 4 minutes for validating.

A. Network topology and training method

The first two feed forward layers have 128 hidden units each one with tanh activation functions. The shared hidden LSTM layer contains 256 memory cells, also with tanh activation functions, and the dropout applied is 0.5. The architecture parameters have been selected based on objective seeking procedures performed with a single-output acoustic model (concretely with M1 model). Finally, the output layers contain 43 units to produce the later explained acoustic predictions in a regression fashion after a sigmoid activation function. The LSTM units get the forget gate bias initialized to one for better performance, as specified in [13].

In the training stage, every speaker error is back-propagated independently and sequentially, such that a speaker ID is randomly selected in every round (assigning a turn to the speaker), completing an epoch when all rounds of speakers have been seen. A round then is a sequence of back-propagated mini-batches, having N mini-batches per round as we have N speakers, each speaker having its turn inside the round, and when a speaker mini-batch is back-propagated we move on to the next randomly picked one until all speakers are processed and we can shuffle the IDs again. Figure 2 depicts the training procedure for a round in the epoch, where the turn of every mini-batch is in brackets. Note that with this method we do not require to have the same transcriptions per speaker, as mentioned earlier, and also every mini-batch is related to only one speaker. This training procedure is an important difference regarding the aforementioned work with multi-task DNN [3].

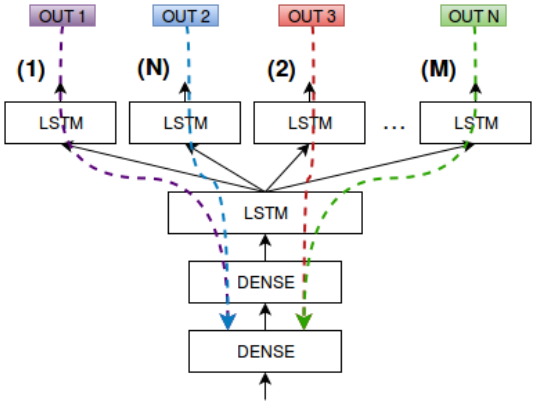

Fig. 2. Exemplified training round for the N mini-batches. Dashed lines represent the corresponding output error back-propagation. The numbers in brackets express the order of that mini-batch inside the round.

B. Acoustic Features

We used Ahocoder [4], a high quality vocoder, for the waveform generation after the acoustic prediction. The network maps an input set of linguistic characteristics to acoustic features, which are then fed into the Ahocoder system to generate the synthetic speech. The predicted set of features include:

- 40 Mel-cepstral coefficients
- max. voiced frequency (fv)
- log-F0 value
- voiced-unvoiced flag (uv)

The acoustic parameters are extracted in frames of 15ms shifted every 5ms. They are normalized to be in the range $[0.01, 0.99]$ during the training to work in the linear region, and they have to be denormalized at prediction time with the dynamic ranges extracted from every speaker's training set.
The maximum voiced frequency output feature is also log-normalized to compress the long tail into a narrower range. Moreover, the log-F0 contours are linearly interpolated in the log domain so that there are continuous values when we have unvoiced frames, but the voiced-unvoiced flag serves to mask those virtual values out at prediction time. During prediction stage, the cepstral parameters are post-filtered based on [18] with a multiplicative increasing factor of $p_f = 1.04$ to overcome the smoothing effect at network outputs, similar to what happened in SPSS [23].

C. Input Features

The input set of features fed to the model describe many linguistic properties extracted from the text with the Ogmios [1] front-end. The features are composed of different types of data defined mainly in the HTS label format [16], involving contextualized prosodic and phonetic features. First we have categorical features involving phoneme identity, vowel identity and Part Of Speech tags, all of them encoded in a one-hot fashion. There are also other boolean types encoded with 1 bit indicating if the current/next syllables are stressed or not, and some boolean features are binary answers (yes/no) to a set of questions regarding the type of phoneme (Affricate, Plosive, Palatal,etc.) and the type of contents in the Part-Of-Speech tags. Besides the categorical and boolean features, there are also numeric ones encoding distances between punctuation marks, number of phonemes in syllable, distance from current phoneme to the end of syllable/word/sentence, etc. We z-normalize them to absorb the possible outliers provoked by long-tailed distributions, such that, for every feature x and its distribution parameters μ, σ:

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{2}$$

One of the inputs is the duration of the current phoneme in order to generate the proper amount of acoustic frames, as well as the relative position of the current frame within the total phoneme duration. This duration would normally be predicted from the linguistic features, similarly to [22], but in this work we focus in the acoustic mapping problem. The duration features then have to be properly normalized to be distributed between $[0, 1]$:

$$\hat{d} = \frac{\ln d - \ln d_{min}}{\ln d_{max} - \ln d_{min}} \quad \hat{r} = \frac{r}{d} \tag{3}$$

where r is the relative position in milliseconds within the frame, and d is the total duration of the phoneme in milliseconds. The input features contain not only the current time-step information, but also the information about the next two following phonemes so that the closest future context is also taken into account without changing the forward-in-time nature of our recurrent model.
The total number of input features for the system are 362, which is the concatenation of all the ones aforementioned.

D. Speaker Adaptation

For the purpose of checking how we can insert a new speaker into the architecture with minimal changes and best results, we have the speaker F3 separated. When the multiple output model is trained, we get what we call the pre-trained multiple speaker weights (the ones in all layers just before the output LSTMs) and attach a new output branch to the model to back-propagate the error for the new speaker through its branch. We make the experiment varying the amount of data available for the new speaker in batches of 25%, 50% and 100%, to check the possible change in the quality of the new voice. Moreover, we mentioned the requirement of a simple adaptation model, where we change the least things possible, so we freeze the shared layers to not update them during the back-propagation of the new speaker, training only the new output branch.

V. Results

A. Multiple output model

We make a first analysis by looking at the training loss evolution of the different speaker outputs, and concretely focusing on two speakers: M1 and F1. To establish a reference, we trained M1 and F1 with a single output architecture and multiple output one. The results can be seen in Figure 3, where the 7 speaker learning curves are shown, depicting that all output converge with a noisy behavior, given by the training methodology where every speaker distorts each other's learning process for mini-batches of data.
There we can see how speakers F1 and M1 get to a lower training loss when they are trained with the multiple output mechanism. This is normally related to a better training procedure where they reach a better point in the optimization. To really see this effect, we first make an objective evaluation by means of specific metrics for each kind of predicted feature. The Mel Cepstral Distortion [15] (MCD) is known to be correlated to subjective evaluations [14]. We also compute the RMSE of the predicted F0 (Hertz scale) and the error in UV flag prediction.

$$MCD = (10\sqrt{2})/(T \ln 10) \sum_{t=0}^{T-1} \sqrt{\sum_{n=0}^{39} (c_{t,n} - \hat{c}_{t,n})^2} \tag{4}$$

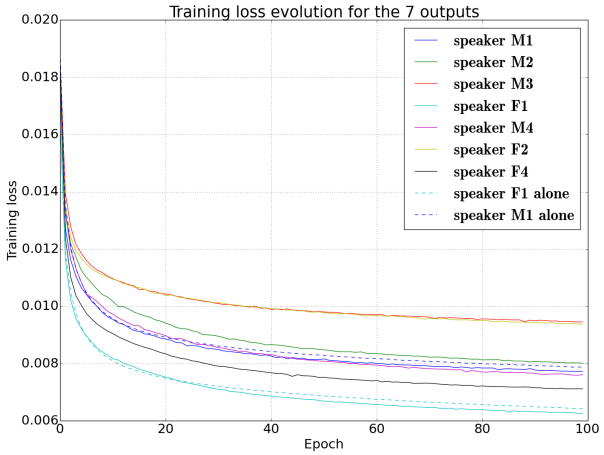

Fig. 3. Training loss evolution comparison. Speakers F1 and M1 decrease the learning cost when trained with other speakers altogether.

where T is the number of frames

$$RMSE\,[Hz] = \sqrt{\sum_{t=0}^{T-1}(f_{0t} - \hat{f}_{0t})^2} \tag{5}$$

As depicted in table I, both speakers improve when trained in the multiple output model almost in all metrics.

TABLE I
OBJECTIVE EVALUATION FOR M1 AND F1 TRAINED ALONE WITH A SINGLE OUTPUT MODEL AND TOGETHER WITH OTHER SPEAKERS (MIXED) IN THE MULTIPLE OUTPUT ARCHITECTURE.

Model	MCD[dB]	RMSE F0[Hz]	UV[%]
M1 alone	7.6	14.4	7.7
M1 mixed	7.2	13.8	5.8
F1 alone	7.0	17.3	4.8
F1 mixed	6.5	17.3	3.8

The subjective evaluation has been carried out with a preference test made by 16 subjects. For both F1 and M1 speakers, 5 sentences are selected and evaluated. The listeners can choose a declining score between two synthesized utterances; one generated by the single output model and another one by the multiple output one. Listeners then find five options available from -2 (multiple output is much preferred) to 2 (single output is much preferred). The results are depicted in Figure 4. It can be seen that the testing subjects have all rather preferred the multiple output model in most of the cases. We also made a Wilcoxon test for the subjective evaluation to find out how statistically meaningful are these results, obtaining the following p-values: $p_{F1} = 5.3 \cdot 10^{-7}$ and $p_{M1} = 2.2 \cdot 10^{-5}$.

B. Speaker adaptation

In Figure 5 we can see how the validation cost for the F3 speaker improves when we add more data, something we could expect, as it learns better with the more data it gets to fine-tune the new output branch. It is interesting to see how freezing the shared layers and training only the new output branch we get to a very similar result, and more smoothly.
Table II summarizes the objective evaluation for this fine-tuning, getting a good result with respect to the single output model of F3 when only the last layer is trained on top of the shared parts of the model. The error values are quite higher in comparison with the previous ones (F1,M1), because this speaker was taken from an expressive subset of data, being it quite different from that of F1 and M1. Also a preliminary listening test suggested that the adaptation sounded close to the original speaker, thus validating this approach.

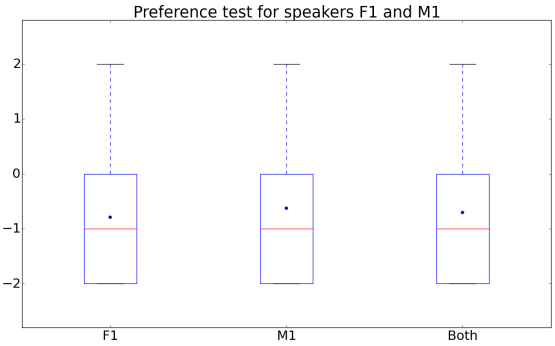

Fig. 4. Box plot of preference test scores. Scores range from -2 (multiple output model is preferred) to 2 (single output trained model is preferred). Both is the summary of all the answers, joining both speaker results. Red lines: medians. Blue dots: means.

TABLE II
OBJECTIVE EVALUATION FOR F3 AS AN ADAPTATION SUBJECT. FULL: ALL LAYERS ARE FINE-TUNED. FROZEN: ONLY NEW OUTPUT BRANCH IS FINE-TUNED.

Model	MCD[dB]	RMSE F0[Hz]	UV[%]
F3 alone	8.11	28.07	9.00
F3 fine-tuned full 100% data	7.96	26.96	7.74
F3 fine-tuned frozen 100% data	7.90	26.44	6.73

VI. CONCLUSIONS

In this work we have implemented an acoustic mapping architecture based on RNN-LSTM layers to handle many speakers simultaneously. We wanted to study the effect of mixing many speakers inside the same model. The results suggest that mixing the first linguistic mappings is useful to capture some patterns that can be included in others' styles, speed, phoneme combinations, etc.
We have also worked in a first speaker adaptation approach, where we just need to insert another output branch on top of the pre-trained system and fine-tune it without modifying the whole structure, thus preserving the multiple output base model and lowering the footprint in memory to get a new

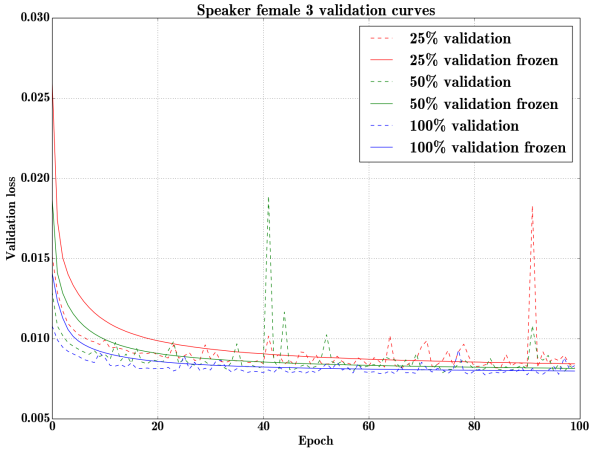

Fig. 5. Validation loss evolution comparison of different batch sizes, with frozen shared layers and fine-tuned shared layers.

speaker model. This approach worked better than training the new speaker in an isolated way.

ACKNOWLEDGMENTS

This work was supported by the Spanish Ministerio de Economía y Competitividad and European Regional Developmend Fund, contract TEC2015-69266-P (MINECO/FEDER, UE). This work has also received funding from Eusipco'11 organization.

REFERENCES

[1] Antonio Bonafonte, Pablo D Agüero, Jordi Adell, Javier Pérez, and Asunción Moreno. Ogmios: The UPC text-to-speech synthesis system for spoken translation. In *TC-STAR Workshop on Speech-to-Speech Translation*, pages 199–204, 2006.

[2] Antonio Bonafonte, Harald Höge, Imre Kiss, Asunción Moreno, Ute Ziegenhain, Henk van den Heuvel, Horst-Udo Hain, Xia S Wang, and Marie-Neige Garcia. Tc-star: Specifications of language resources and evaluation for speech synthesis. In *Proc. of LREC Conf*, pages 311–314, 2006.

[3] Li Deng and Dong Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[4] Daniel Erro, Iñaki Sainz, Eva Navas, and Inma Hernáez. Improved HNM-based vocoder for statistical synthesizers. In *INTERSPEECH*, pages 1809–1812, 2011.

[5] Yuchen Fan, Yao Qian, Frank K Soong, and Lei He. Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4475–4479. IEEE, 2015.

[6] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Interspeech*, pages 1964–1968, 2014.

[7] Raul Fernandez, Asaf Rendel, Bhuvana Ramabhadran, and Ron Hoory. F0 contour prediction with a deep belief network-gaussian process hybrid model. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6885–6889. IEEE, 2013.

[8] Raul Fernandez, Asaf Rendel, Bhuvana Ramabhadran, and Ron Hoory. Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. In *Interspeech*, pages 2268–2272, 2014.

[9] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Vladimir Hozjan, Zdravko Kacic, Asuncion Moreno, Antonio Bonafonte, and Albino Nogueiras. Interface databases: Design and collection of a multilingual emotional speech database. In *LREC*, 2002.

[12] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 373–376. IEEE, 1996.

[13] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

[14] Robert F Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Communications, Computers and Signal Processing, 1993., IEEE Pacific Rim Conference on*, volume 1, pages 125–128. IEEE, 1993.

[15] Mikiko Mashimo, Tomoki Toda, Kiyohiro Shikano, and Nick Campbell. Evaluation of cross-language voice conversion based on gmm and straight. 2001.

[16] Keiichiro Oura. An example of context-dependent label format for HMM-based speech synthesis in English. *HTS-demo CMU-ARCTIC-SLT, from http://hts. sp. nitech. ac. jp*, 2011.

[17] Yao Qian, Yuchen Fan, Wenping Hu, and Frank K Soong. On the training aspects of deep neural network (DNN) for parametric TTS synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3829–3833. IEEE, 2014.

[18] Alexander Sorin, Slava Shechtman, and Vincent Pollet. Uniform speech parameterization for multi-form segment synthesis. In *INTERSPEECH*, pages 337–340, 2011.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[20] Zhizheng Wu and Simon King. Investigating gated recurrent neural networks for speech synthesis. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5140–5144. IEEE, 2016.

[21] Heiga Ze, Alan Senior, and Martin Schuster. Statistical parametric speech synthesis using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7962–7966. IEEE, 2013.

[22] Heiga Zen and Hasim Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4470–4474, 2015.

[23] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, 2009.