Multi-output RNN-LSTM for multiple speaker speech synthesis with α-interpolation model

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 interpolation by adding a new output layer (α-layer) on top of the multi-output branches. An identifying code is injected into the layer together with acoustic features of many speakers. Experiments show that the α-layer can effectively learn to interpolate the acoustic features between speakers.

Index Terms: text to speech, acoustic mapping, speaker interpolation, recurrent neural network

1. Introduction

Deep Learning has been applied successfully to different kinds of tasks such as computer vision, natural language processing or speech processing [1], outperforming the existing systems in many cases. In the case of speech synthesis, many works included DNNs and DBNs to perform acoustic mappings and prosody prediction [2, 3, 4], Also, Recurrent Neural Networks (RNNs) and their variants, like the Long Short Term Memory (LSTM) architecture [5], 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, 7, 8, 9].

Previous to deep learning, existing text to speech technologies included the unit selection speech synthesis [10] and the statistical parametric speech synthesis (SPSS) [11]. 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 [11] 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. Also, techniques for interpolating speaker models were proposed in the SPSS framework [12], which exploits the flexibility property of these models.

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 interpolation of these learned representations. 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.

In our previous work [13] we also tested this model architecture for the speaker adaptation problem, attaching a new output branch in parallel to the already trained output branches. It gave good results by just fine-tuning the new output branch, so it seemed reasonable to add another layer to do the interpolation job.

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 3 we describe our proposed model for both the multiple output architecture and the α-interpolation, followed by an explanation of the experimental setup made in section 4. Sections 5 and 6 cover the results and conclusions respectively, where we analyze the response of our model to different questions we make about its properties.

2. Recurrent Neural Network Review

Recurrent Neural Networks (RNN) are the specialization of a neural network into the sequences processing problem, where every neuron (or hidden unit) in the recurrent layer has a memory associated that tracks past decisions, in addition to the feed forward decision. A recurrent layer is characterized by the following equation:

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

where $\boldsymbol{x_t} \in \mathbb{R}^n$ is the input vector at time t, $\boldsymbol{b} \in \mathbb{R}^m$ is the bias vector, $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ is the feed forward weights matrix and $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ is the recurrent weights matrix, where the temporal patterns are learned. After the linear operators an element-wise non-linear function g is applied to get the output of the recurrent layer, which is in $\boldsymbol{h_t} \in \mathbb{R}^m$, the decision given the input at time t and the previous decision $\boldsymbol{h_{t-1}}$. Sometimes this is called the memory state of the recurrent layer.

Note then that for every $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_T}\}$ input sequence we obtain an output sequence $\boldsymbol{H} = \{\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_T}\}$. This dependency on previous decisions make RNNs suitable to process sequences conditioning each output on its memory, also being able to process different sequence lengths with the same

number of static inputs and outputs.
In this work we used Long Short Term Memory (LSTM) units instead of simple RNN units as they cope better with the vanishing gradient problems during training [14], and they also maintain long range dependencies better than conventional RNNs because of their gating mechanisms that control the information flows in and out of the layer without corrupting useful information in the further past [5].

3. 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 to a dense representation about the input data, which is formed by a mixed set of multiple types of features that will be presented in section 4. 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 [15] 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.

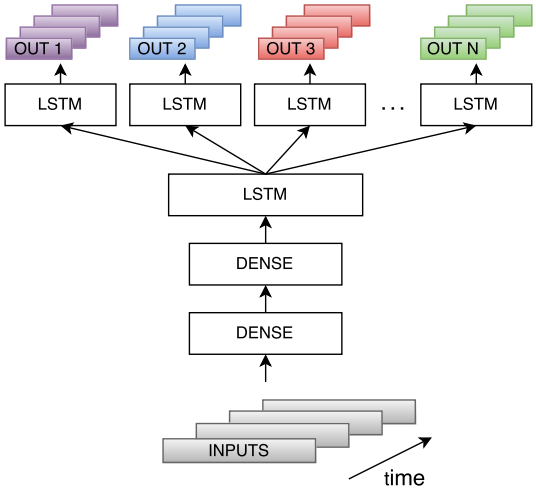

Figure 1: Proposed architecture using regular feed forward (dense) layers and recurrent LSTM layers. There are N outputs belonging to N different speakers.

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 4. 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 [7].
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.

3.1. α-interpolation layer

We also propose a method to interpolate the different speaker acoustic mappings that the network learns. In order to do it, another output recurrent layer (LSTM) is stacked on top of an M set of speakers ($M \leq N$, where N is the total number of output branches available). An example of this is depicted in Figure 2 with $M = 2$.

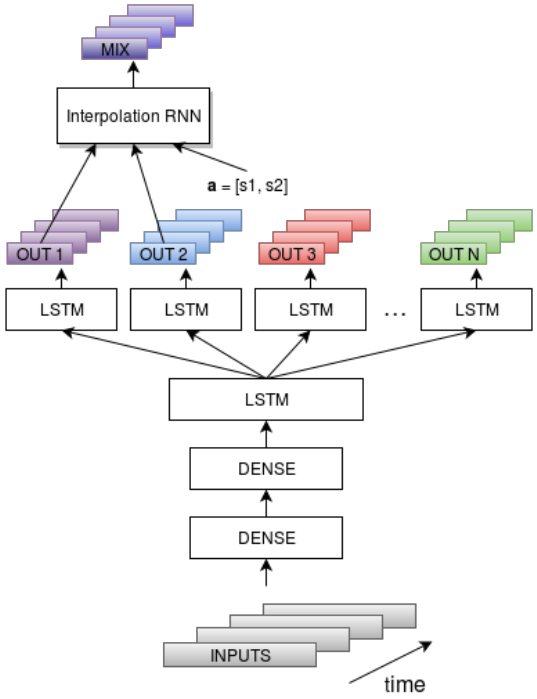

Figure 2: Example of α-interpolated multi-output model with $M = 2$.

The way to control the weight of each speaker is by means of the α-vector (**a** in Figure 2), which, during training time, is a one-hot code vector to indicate the identity of each speaker, as explained in 4.4. During synthesis it is not required to be one-hot anymore, but real values indicating the portion of each speaker we have selected.

4. Experimental Setup

We worked with six voices from the TCSTAR [16] project (M1,M2,M4,F1,F3,F4) where four of them contain expressive speech (M2,M4,F3,F4), and two neutral voices from interface [17](M3,F2). 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). For the multi-output model experiments, four male voices and three female voices were used (seven in total), where five were of the TCSTAR and two were from the interface database.
On the other hand, for the α interpolation experiments up to six voices were used (and another multi-output model with six output branches was pre-trained). All the voices in the α experiments were from TCSTAR.

The models have been implemented with the Keras [18] Deep Learning library.

4.1. 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. 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 [19].
In the training stage, every speaker error is back-propagated once per epoch such that a speaker ID is randomly selected and trained for a full epoch of its own data by means of the RMSprop algorithm [20], and then we move on to the next randomly picked speaker until we have processed all of them. This procedure is repeated for 100 epochs.

4.2. Acoustic Features

We used Ahocoder [21], 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 [22] with a multiplicative increasing factor of $p_f = 1.04$ to overcome the smoothing effect at network outputs, similar to what happened in SPSS [11].

4.3. Input Features

The input set of features fed to the model describe many linguistic properties extracted from the text with the Ogmios [23] front-end. The features are composed of different types of data extracted mainly from the HTS label format [24], so first we have categorical features involving phoneme identity, vowel identity and Part Of Speech tags, all of them encoded in a one-hot fashion.
Besides the categorical and boolean features, there are also numeric features encoding distances between punctuation marks, stressed entities, 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 [7], 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.

4.4. α-layer training

The α-layer is trained by freezing the multi-output model weights. For the α interpolation experiments we pre-trained a 6 speaker multi-output model. As mentioned previously, the layer has M speaker branches as input, thus raising $M \times O$ speaker input units, where O is the acoustic vector dimension. Another input vector is inserted to control the weight that every speaker has in the interpolation, called the α vector, which is a one-hot code of dimension M.
During training, linguistic inputs are injected into the multi-output model and the one-hot α is given, expressing the identity of the current shown speaker at the interpolation layer output. The same training data used for training the multi-output model is used for the M speakers.
This methodology expects the layer to learn not only each extreme case (i.e. each one-hot case shown during training), but it is also expected to infer intermediate values for the acoustic outputs, and as it is seen in section 5.2, it actually learns to interpolate the features.
We designed the experiments to interpolate 2 speakers out of the 6 mentioned previously from the TCSTAR database. For the interpolation, two configurations were trained, $M = (2, 6)$.

5. Results

5.1. 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.

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.

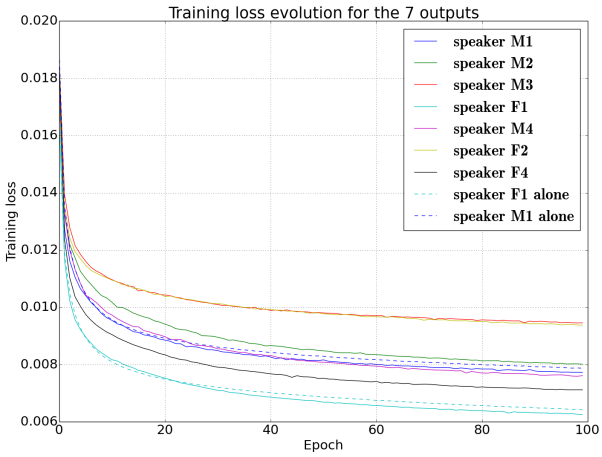

Figure 3: Training loss evolution comparison. Speakers F1 and M1 decrease the learning cost when trained with other speakers altogether.

The Mel Cepstral Distortion [25] (MCD) is known to be correlated to subjective evaluations [26].

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

where T is the number of frames. The MCD is computed without applying post-filtering. We also compute the RMSE of the predicted F0 and the error in UV flag prediction.

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

Table 1: 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]	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

As depicted in Table 1, both speakers improve when trained in the multiple output model almost in all metrics.

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 chose 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.

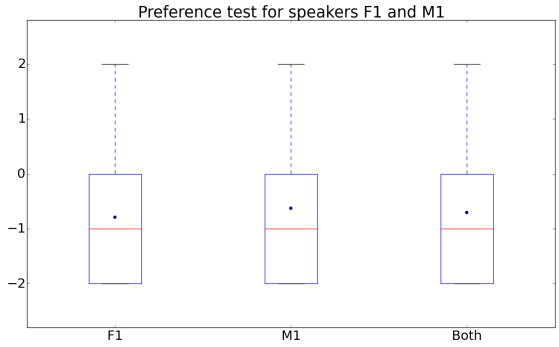

Figure 4: Box plot of preference test scores. Scores range from -2 (multiple output model is preferred) to 2 (single output trained model is preferred). Both is the summary of all the answers, joining both speaker results. Red lines mark the median and blue dots the mean.

5.2. α-interpolation layer

Objective tests have been performed to evaluate the performance of the α interpolation. These consist in analyzing the evolution of the MCD between the interpolation output and each of the M branches, and also the evolution of the F0 RMSE. Figures 5 and 6 show these results, where the α variation is made for speaker F1, so it is α_{F1} , speaker M1 has $\alpha_{M1} = (1-\alpha_{F1})$ and all others are $\alpha_m = 0$. We may refer to $\alpha = 0.5$ for the point at $\alpha_{F1} = \alpha_{M1} = 0.5$.
A preliminary subjective test clearly showed that increasing M improved the naturalness of the output speech although only 2 speakers are interpolated in the evaluation.

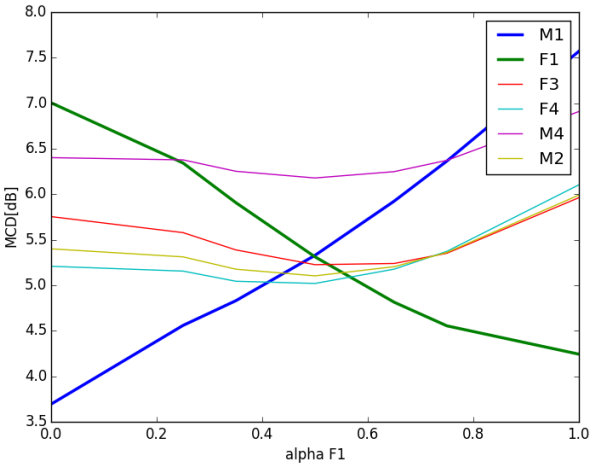

Figure 5: MCD when varying α values. The variation is made for speaker F1 and it is $(1 - \alpha)$ for M1. All others speakers remain 0. $M = 6$.

From the curves we see how, although we only show to the network the extreme values with an orthogonal code, it learns the intermediate representations effectively. The MCD values vary smoothly between the interpolated speakers F1 and M1, whilst other speakers' MCD remain with a short variation. It is interesting the fact that the crossing point is very close to $\alpha = 0.5$.
Note that the values may differ from those in Table 1 because the distances are not computed to natural speech but to the multi-output predictions in this case.

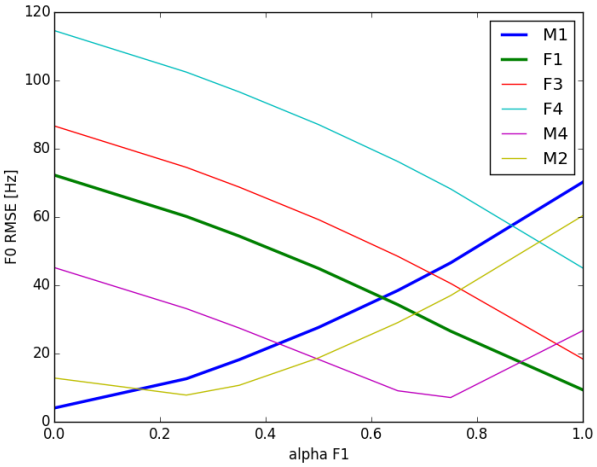

Figure 6: F0 RMSE when varying α values. The variation is made for speaker F1 and it is $(1-\alpha)$ for M1. All others speakers remain 0. $M = 6$.

Regarding the F0 RMSE evolution, there is a biasing of the crossing point, which shows us how the F0 prediction is biased towards the male speaker, as it is the one getting less error for $\alpha = 0.5$.
These interpolation results are coherent with perceptual impression. As previously mentioned, increasing M helped in the naturalness of the 2-speaker interpolation, so an analysis of the F0 distributions is also made for the cases $M = 2$ and $M = 6$. These analysis are shown in Figures 7 and 8 respectively.

First, we can confirm the biasing towards the male speaker when $\alpha = 0.5$ (M1 50% F1 50%) in both cases. Nevertheless an important difference is the fact that training the layer with a higher M increases the distributions variance, which turns out to be a less monotonous sound at the output.

6. 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 with a speaker interpolation approach, where we need to insert another output layer (the α-layer) on top of M pre-trained speaker branches , and fine-tune it without

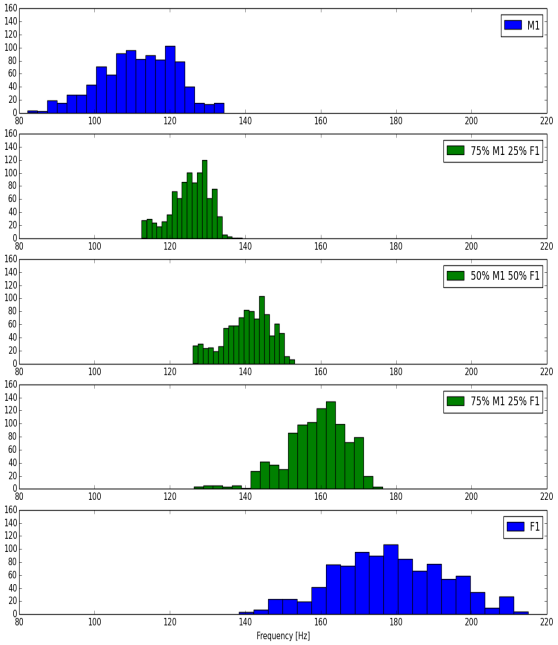

Figure 7: F0 Histograms: original M1 and F1 speakers in blue. $\alpha_{F1} = (0.25, 0.5, 0.75)$ and $\alpha_{M1} = (0.75, 0.5, 0.25)$ interpolations in green. $M = 2$.

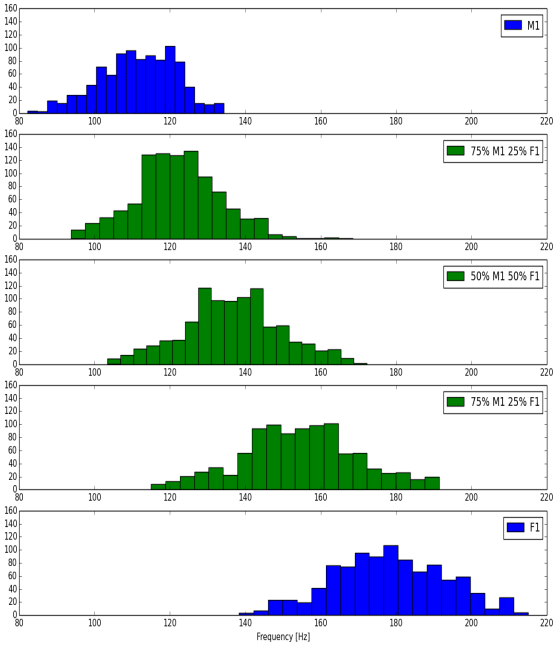

Figure 8: F0 Histograms: original M1 and F1 speakers in blue. $\alpha_{F1} = (0.25, 0.5, 0.75)$ and $\alpha_{M1} = (0.75, 0.5, 0.25)$ interpolations in green. $M = 6$.

modifying the whole structure. We have seen how, when we show the network extreme cases representing the different speakers by means of an orthogonal identity code, it is able to infer intermediate values. Furthermore, training the α-layer with $M = N$ helped the interpolation to increase the variance, thus producing more natural speech.

7. Acknowledgments

This work was supported by the Spanish Ministerio de Economa y Competitividad and European Regional Development Fund, contract TEC2015-69266-P (MINECO/FEDER, UE).

8. References

[1] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 34, pp. 197–387, 2014.

[2] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. of ICASSP*. IEEE, 2013, pp. 7962–7966.

[3] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "F0 contour prediction with a deep belief network-gaussian process hybrid model," in *Proc. of ICASSP*. IEEE, 2013, pp. 6885–6889.

[4] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (dnn) for parametric tts synthesis," in *Proc. of ICASSP*. IEEE, 2014, pp. 3829–3833.

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

[6] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. of INTERSPEECH*, 2014, pp. 1964–1968.

[7] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Proc. of ICASSP*, 2015, pp. 4470–4474.

[8] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "Prosody contour prediction with long short-term memory, bidirectional, deep recurrent neural networks." in *Proc. of INTERSPEECH*, 2014, pp. 2268–2272.

[9] Z. Wu and S. King, "Investigating gated recurrent networks for speech synthesis," in *Proc. of ICASSP*. IEEE, 2016, pp. 5140–5144.

[10] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proc. of ICASSP*, vol. 1. IEEE, 1996, pp. 373–376.

[11] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[12] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Speaker interpolation in hmm-based speech synthesis system," in *Proc. of EUROSPEECH*, 1997, pp. 2523–2526.

[13] S. Pascual and A. Bonafonte, "Multi-output RNN-LSTM for multiple speaker speech synthesis and adaptation," in *Proc. of EUSIPCO*, 2016.

[14] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[16] A. Bonafonte, H. Höge, I. Kiss, A. Moreno, U. Ziegenhain, H. van den Heuvel, H.-U. Hain, X. S. Wang, and M.-N. Garcia, "TC-STAR: Specifications of language resources and evaluation for speech synthesis," in *Proc. of LREC Conf.*, 2006, pp. 311–314.

[17] V. Hozjan, Z. Kacic, A. Moreno, A. Bonafonte, and A. Nogueiras, "Interface databases: Design and collection of a multilingual emotional speech database," in *Proc. of LREC Conf.*, Las Palmas de Gran Canaria, Spain, May 2002.

[18] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[19] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2342–2350.

[20] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, p. 2, 2012.

[21] D. Erro, I. Sainz, E. Navas, and I. Hernáez, "Improved HNM-Based Vocoder for Statistical Synthesizers," in *Proc. of INTERSPEECH*, 2011, pp. 1809–1812.

[22] A. Sorin, S. Shechtman, and V. Pollet, "Uniform speech parameterization for multi-form segment synthesis." in *Proc. of INTERSPEECH*, 2011, pp. 337–340.

[23] A. Bonafonte, P. D. Agüero, J. Adell, J. Pérez, and A. Moreno, "Ogmios: The UPC text-to-speech synthesis system for spoken translation," in *TC-STAR Workshop on Speech-to-Speech Translation*, 2006, pp. 199–204.

[24] K. 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.

[25] M. Mashimo, T. Toda, K. Shikano, and N. Campbell, "Evaluation of cross-language voice conversion based on GMM and STRAIGHT," in *Proc. of EUROSPEECH*, 2001, pp. 361–364.

[26] R. F. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, vol. 1. IEEE, 1993, pp. 125–128.