PARAMETER GENERATION ALGORITHMS FOR TEXT-TO-SPEECH SYNTHESIS WITH RECURRENT NEURAL NETWORKS

Viacheslav Klimkov, Alexis Moinet, Adam Nadolski, Thomas Drugman

Amazon.com

ABSTRACT

Recurrent Neural Networks (RNN) have recently proved to be effective in acoustic modeling for TTS. Various techniques such as the Maximum Likelihood Parameter Generation (MLPG) algorithm have been naturally inherited from the HMM-based speech synthesis framework. This paper investigates in which situations parameter generation and variance restoration approaches help for RNN-based TTS. We explore how their performance is affected by various factors such as the choice of the loss function, the application of regularization methods and the amount of training data. We propose an efficient way to calculate MLPG using a convolutional kernel. Our results show that the use of the L1 loss with proper regularization outperforms any system built with the conventional L2 loss and does not require to apply MLPG (which is necessary otherwise). We did not observe perceptual improvements when embedding MLPG into the acoustic model. Finally, we show that variance restoration approaches are important for cepstral features but only yield minor perceptual gains for the prediction of F0.

Index Terms— Speech Synthesis, TTS, MLPG, RNN

1. INTRODUCTION

Acoustic modeling (AM) is a core component of text-to-speech (TTS) systems, and aims at predicting realistic speech parameters given linguistic information. Acoustic models are not only the center of statistical parametric speech synthesizers [1], but are also used in hybrid unit selection [2] as well as in WaveNet-like synthesis [3] to explicitly specify slowly-varying acoustic parameters (such as F0).

AM in TTS has followed the advances in Machine Learning. AM based on Hidden Markov Models (HMMs) has been extensively studied for more than a decade [4, 5]. With the emergence of Deep Learning, the use of feed-forward Deep Neural Networks (DNN, [6]), Deep Mixture Density Networks (DMDN, [7]) and Long Short-Term Memory (LSTM, [8]) RNN has naturally followed.

In HMM-based TTS, the continuity of frame-level trajectories was ensured by the Maximum Likelihood Parameter Generation (MLPG) algorithm [4]. MLPG is used to take the dynamic feature constraints into account. It generates, at synthesis time, the most probable sequence given the statistics of both static and dynamic features. In DNN-based synthesis, MLPG has been inherited from HMM-based TTS, and is currently widely used on top of feed-forward and RNN models for acoustic parameter prediction [8–10]. However, although the benefit of MLPG is unquestionable with HMMs and was reported in [9] to be helpful with feed-forward neural networks, its advantage is less clear in the case of RNNs. Indeed, with RNNs, the implicit dependency on previous (and next, in case of bidirectional architectures) predictions should help producing smooth, continuous trajectories of acoustic parameters. In the literature, the picture is mixed: while MLPG was found to be beneficial in [10, 11], it was claimed to be unnecessary in [8].

Moreover, recent studies [12–14] have shown that it is beneficial to embed MLPG inside the deep neural network model. In this case, the whole parameter generation is directly incorporated into the training process, and the discrepancies between training and synthesis conditions are alleviated.

Another known problem of conventional AMs is averaging parameter trajectories in similar linguistic contexts. This makes speech less dynamic and is reflected in a decrease of the global variance. As shown in [14], it is possible to restore the global variance of acoustic parameters by performing a sort of multi-task learning, where a loss component that minimizes the difference in variance at the sequence level is added to the usual error minimization term. It is questionable how this method is affected by the parameter generation algorithm.

Yet another approach, that affects continuity of generated trajectories, is including autoregressive output layers into acoutic model of text-to-speech system [10, 11, 15]. In this paper we adress more conventional acoustic modelling approaches and comparison to autoregressive techniques is among future work.

The goal of this paper is to extensively study the impact of parameter generation on RNN-based TTS. We investigate the influence of various factors on the TTS quality: type of loss function, application of regularization techniques, amount of training data, type of generation algorithm and global variance restoration. We establish in which conditions MLPG is beneficial, and most importantly, in which case it is not. The conclusions of our experiments can be seen as practical guidelines when training a RNN-based TTS system. While the main contribution of this paper lies in the comprehensive and methodological analysis, we also propose to use the L1 loss as well as an efficient convolutional implementation of MLPG. Those are novel approaches to the best of our knowledge.

The structure of the paper is as follows. Section 2 describes our RNN-based TTS system. Section 3 presents the techniques for parameter generation studied in this paper. The various factors analyzed in this paper are explained in Section 4. The experimental protocol is detailed in Section 5, and the results of our experiments are discussed in Sections 6 and 7.

2. OUR RNN-BASED TTS SYSTEM

This section describes our RNN-based TTS system which is used throughout our experiments. Categorical linguistic features are expanded to one-hots and concatenated with numerical inputs, resulting in a 453-dimensional input vector per frame. The architecture of our model consists of 2 feed-forward layers, 2 bidirectional recurrent layers and an affine transform for each output stream. This topology follows that initially introduced by [8]. Feed-forward layers have 512 neurons and hyperbolic tangent activation functions. Recurrent layers utilize Gated Recurrent Unit (GRU) memory cells with di-

mension 128 in each direction. Each recurrent layer is followed by dropout with 0.25 rate. Note that we did not apply dropout across time as proposed in [16].

A single network is used to predict the multiple acoustic streams. The output of the last recurrent layer is projected into four streams: band aperiodicities (dimension 3), voiced/unvoiced decision (dimension 1), Mel-Generalized Cepstrum (MGC, dimension 60) and log-F0 (dimension 1). A sigmoid activation function is applied to band aperiodicities and voiced/unvoiced decision, and a linear activation function is used for MGCs and log-F0.

Xavier initialization [17] and L2 regularization with 0.001 scale are applied for both feed-forward and recurrent layers. The total loss consists of a weighted sum of the losses of the various streams. The weight for each stream was set to its dimensionality (e.g. a weight of 60 for the loss of the MGCs). For the log-F0 and band aperiodicities streams, the voicing mask is applied so that only voiced frames account in the loss. For the voiced/unvoiced decision stream, cross-entropy was used. For the rest of the streams, we experimented both with L1 and L2 loss. To the best of our knowledge, the L1 loss has never been used to train an AM for TTS in the literature.

Adam optimization [18] was used during the training until convergence, i.e. until the total loss on the development set stopped decreasing significantly for at least 5 epochs. Noam learning rate scheme (as in [19]) with peak value 0.003 was used. Gradients were clipped by the ratio of the sum of their norms [20]. The training was processed in batches of data, with a batch size adapted to the amount of training data: 2, 12 and 32 sentences respectively for the 1, 7 and 20-hour datasets (see Section 5). Waveform synthesis from the predicted acoustic features was achieved using the WORLD vocoder [21].

3. PARAMETER GENERATION

This section discusses the process of parameter generation. The MLPG algorithm is presented in Section 3.1 and its proposed convolutional formulation is given in Section 3.2.

3.1. Maximum Likelihood Parameter Generation (MLPG)

Given a sequence of speech parameters $C = [c_1^T, c_2^T, ..., c_F^T]^T$, the sequence of so-called super-vectors $O = [o_1^T, o_2^T, ..., o_F^T]^T$ which contain both static and dynamic features, can be expressed as [4]:

$$O = WC \tag{1}$$

where W is a block-diagonal matrix. Assuming that the super vector follows a normal distribution with mean μ and covariance matrix Σ, the probability of the whole sequence is maximal when [4]:

$$\hat{C} = (W^T\Sigma^{-1}W)^{-1}W^T\Sigma^{-1}\mu = M\mu \tag{2}$$

where $\hat{C}$ is the predicted static acoustic feature sequence, which will be used to reconstruct the speech waveform. During synthesis, μ is predicted by the model. Σ matrix is usually diagonal and can be either predicted by the model or composed from the global variances of the parameters and their derivatives.

It has been shown that MLPG can be directly embedded within the deep neural network model [12–14]. There are two possible ways to implement that: *i)* perform matrix multiplication with pre-computed non-trainable matrix M from Eq. (2) and minimize the difference between generated parameters and actual values; *ii)* analytically define formulas for the gradients of RNN outputs bypassing parameter generation. Both approaches require the multiplication with full matrices, making the training unreasonably long. This drawback can be alleviated by the convolutional formulation of MLPG proposed in the next section, where we exploit the sparsity property of M.

3.2. Convolutional Formulation of MLPG

We propose in this section an MLPG implementation which is both efficient and easy to put in place. If the global variance is used to compose covariance matrix Σ in Eq. (2), then the rows of matrix M are identical except they are shifted. Since the embedded parameter generation is applied to normalized sequences, Σ is the identity matrix. The matrix W is block-diagonal of shape $[3 \cdot F; F]$ (where F denotes the number of frames in the sequence) and contains coefficients for static and dynamic features. Without loss of generality for the proposed approach, let us assume typical coefficients of W:

$$\begin{gathered} w_{i;j-1,j,j+1} = (0.0, 1.0, 0.0)\ \text{(static)} \\ w_{i-1;j-1,j,j+1} = (-0.5, 0.0, 0.5)\ (\Delta) \\ w_{i+1;j-1,j,j+1} = (1.0, -2.0, 1.0)\ (\Delta^2) \\ \forall\, i = 3j;\ j = 1...F \end{gathered} \tag{3}$$

To obtain a convolutional kernel, let us compute M as in Eq. (2), and pick up its middle row. Since the elements of the sequence are affected only by neighboring elements, the kernel can be approximated as a finite-length filter whose width can be limited in practice. Using the conventional W defined above, the corresponding kernel is displayed in Figure 1, along with a schematic explanation of the convolutional MLPG. It can be seen that values rapidly decrease from the kernel center and become negligible beyond 15 coefficients on both sides.

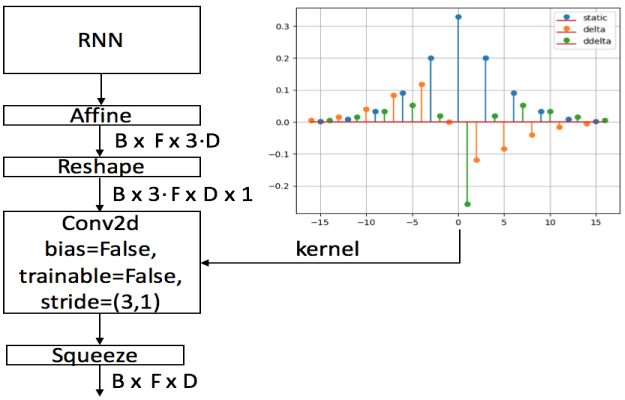

Fig. 1. Workflow of convolutional MLPG, where B stays for batch size, D - for stream dimension.

The drawback of this technique is that it requires a constant variance of the parameters through the whole sequence. Recent studies on trajectory modeling [14] propose to use the likelihood loss function during training, which allows to implicitly involve context-dependent/time-varying non-identity diagonal covariance matrices and use them during parameter generation. We also explored covariance modeling in the scope of our architecture. It is possible to obtain a similar behaviour by applying the convolutional layer with several filters, each calculated for different variance value. The RNN output should be projected to predict the contribution of each filter at a given time step. The softmax activation function should be applied

on top of the predicted filter contributions. A weighted sum of the output of the convolutional layer across the filters axis can be considered as a generated parameter with different local covariances. We made some prior tests that showed that systems trained on a sufficient amount of data ($\geq$ 7 h) produce reasonable filters contribution map: a low covariance is used in regions of slowly-varying acoustic parameters, and a high covariance contributes more in the regions of rapid parameter changes. We plan to explore this path further in a future work.

4. FACTORS STUDIED IN THIS WORK

The goal of our experiments is to establish the impact of various factors on the quality of the predicted trajectories. Those factors include the type of loss function, the application of regularization techniques, the techniques for parameter generation and for variance restoration.

Both the L1 and L2 norms were tried as loss functions. To the best of our knowledge, only the L2 loss was considered in previous studies on acoustic modeling for TTS. With the L2 norm, observations are assumed to follow a Gaussian distribution. With the L1 norm, they are assumed to follow a Laplace distribution. In practice, the Gaussian approximation turns out to best fit the distribution of most of acoustic features. This supports the use of the L2 loss. The L1 norm is however more robust to outliers and it is questionable how it works for RNN-based acoustic modeling.

Under the *regularization* umbrella term, we include: dropout technique applied to the recurrent layers, Xavier initialization, L2 regularization on all layers and gradient clipping (see Section 2). This term therefore covers methods that are designed to take proper care of the convergence of the network. We investigate whether applying those techniques altogether improves the TTS quality.

Five techniques of parameter generation are considered:

- **Baseline:** No parameter generation algorithm. The output of the RNN is directly fed into the vocoder.
- **Smoothing:** Simple smoothing of the generated trajectories using an 11-frame (i.e. 50 ms) wide triangular moving average filter.
- **External MLPG:** Additional streams with dynamic features of first and second orders are modeled for log-F0. The RNN outputs for static and dynamic streams are fed into the MLPG algorithm as vector μ. The diagonal covariance matrix Σ is constructed from the global variances calculated for mean/variance normalization. We used the SPTK implementation [22].
- **Embedded MLPG:** The dimensionality of the log-F0 stream was enlarged by a factor of 3 (to represent additional dynamic features). This stream was connected to the convolutional non-trainable layer with precomputed kernel and stride=3, as shown in Figure 1. The loss was computed for the generated parameters and added to the loss of the other streams. We used the fast convolutional implementation from Section 3.2.
- **Embedded MLPG and pre-training:** Same as above, but the network is first pre-trained before continuing the training with embedded MLPG. This is conform to [13].

Finally, two approaches of variance restoration were applied. The first one is a simple multiplication of the variance of the generated trajectory by a constant factor to match the natural variance, similarly to [23]. The second one is the Sequence Variance Loss (SVL) method proposed in [14]. In SVL, at training time, the difference between the sequence-level variances of predictions and actual values is added to the usual frame-level loss.

5. EXPERIMENTAL PROTOCOL

Our data is from Amazon Alexa US voice, sampled at 24 kHz. We vary the amount of training data to reflect three research/development scenarios: *i)* 1 hour (small dataset, as in [24]), *ii)* 7 hours (medium dataset); *iii)* 20 hours (large dataset, as in [25, 26]). An additional set of 4 hours was held out and equally split for development and test purpose.

We separate our experiments in two parts. In the first part (Section 6), we study the best strategy for RNN-based F0 modeling. Focusing only on F0 is important for hybrid unit selection and WaveNet-like synthesis systems which are fed with a predicted F0 contour. In the second part (Section 7), we analyze the effect of the studied factors on a complete RNN-based TTS where the techniques in question are applied to all acoustic streams.

For our objective tests, we calculate various metrics between the generated and actual trajectories on the 2-hour long test set. The metrics we used for F0 are: the Gross Pitch Errors (GPE, as defined in [27]), the F0 Root-Mean Squared Error (RMSE), the F0 correlation, and the F0 fluctuations (in %) calculated as the relative deviations around the smoothed F0 contour. This latter was obtained using a 15-frame (i.e. 70 ms) wide triangular moving average filter. For the MGC coefficients, we used the standard Mel Cepstral Distortion (MCD, [28]) expressed in decibels.

For our subjective evaluation, we used preference tests between pairs of systems. 100 utterances were randomly selected from the test set and rated by native listeners in the Amazon Mechanical Turk platform. Listeners were asked the question: *Which audio sounds more natural?*. They had to choose among three options: A is better, B is better, or they sound the same. Each utterance was evaluated 5 times, leading to a total of 500 ratings per test. Listeners were not gender-balanced. The statistical significance was inspected using a binomial test and its one-tailed p-value.

6. EXPERIMENTS ON RNN-BASED F0 MODELING

In this section, we study the factors described in Section 4 only on the F0 prediction. We aim at answering the questions left open in the state of the art. Our discussion is based on evidence from both objective and subjective evaluations.

6.1. When is MLPG useful/useless?

We investigate whether external MLPG brings any gain over the baseline (i.e. the raw RNN output). We have performed preference tests in various scenarios of loss function, regularization and amount of training data. We report the Comparative Mean Opinion Score (CMOS) with its 95% confidence interval in Figure 2. To calculate the CMOS, scores of -1 and +1 are given to preferences respectively for the baseline and external MLPG techniques. It can be observed that the external MLPG outperforms significantly the baseline (positive scores) when no regularization techniques are applied. MLPG also brings a slight but statistically significant improvement in the case of L2 loss with regularization. However, when the L1 loss is used with proper regularization, external MLPG turns out *not* to be helpful. Those conclusions are valid irrespective of the amount of training data available.

From a probabilistic perspective, this observation makes sense since the standard MLPG equations assume that the observations are modelled as a Gaussian distribution (and are therefore in line with the L2 assumption). The fact that the L1 norm assumes Laplace distributions creates a theoretical mismatch between training and generation, which could explain why MLPG does not help in the case of L1 with regularization. The probabilistic implications of regularisation terms in the optimisation are less transparent. Relying on empirical results like Figure 2 therefore becomes a necessity.

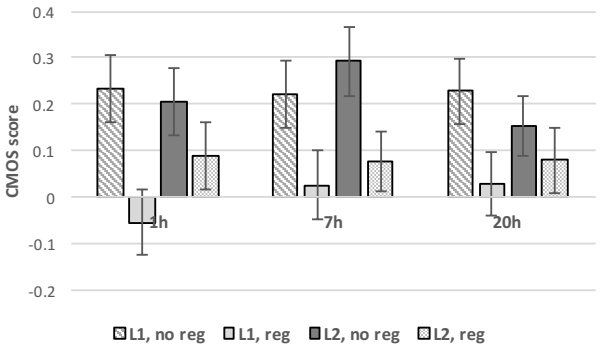

Fig. 2. Subjective gains of external MLPG over the baseline in various conditions.

6.2. What is the best parameter generation approach?

To answer this question, let us have a look at the objective metrics first. Figure 3 shows the F0 RMSE in various configurations of loss function and regularization for a training set of 7 h. It can be seen that all MLPG-based approaches reduce the RMSE compared to the baseline. Smoothing the trajectory, in turn, only helps in case of unregularized training. Among those 5 methods, embedded MLPG with pre-training achieves the best RMSE. Those conclusions were also observed for the GPE and F0 correlation metrics, and using 1 h and 20 h of training data. Those results are not presented in this paper for the sake of conciseness.

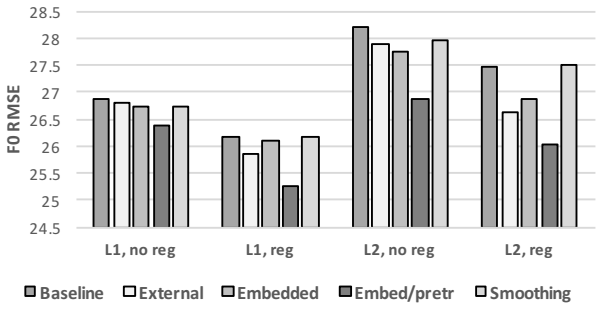

Fig. 3. F0 RMSE for various configurations with the 7 h set.

The results using the F0 fluctuation metric are displayed in Figure 4. This metric is important as fluctuations in the generated F0 contours can have a detrimental perceptual effect characterized by a *trembling* voice. While the external MLPG, the smoothing and, to a lesser extent, the embedded MLPG with pre-training techniques help reducing those fluctuations, it does not seem to be the case with the embedded MLPG (except in the *L2, no regularization* scenario). Again, the conclusions drawn in this section were also found to be valid with the 1 h and 20 h sets.

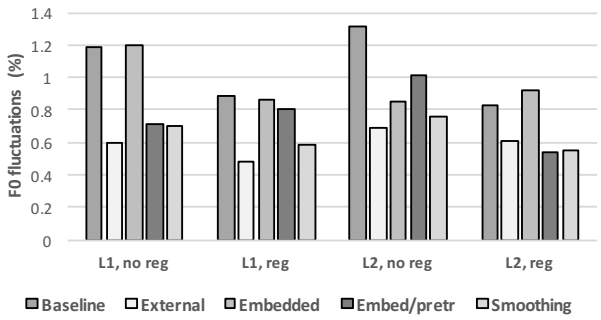

Fig. 4. F0 fluctuations for various configurations with the 7 h set.

To identify which parameter generation technique works the best perceptually, various pair-wise preference tests have been performed using the systems trained on 7 h set with L2 loss without regularization. We use this scenario as it corresponds to a conventional setting in the literature, and as our results from Section 6.1 indicate that MLPG seems the most beneficial in this situation.

Their results are summarized in Figure 5. Across those 6 tests, all preferences were found to be statistically significant except for the comparison of external MLPG and smoothing ($p = 0.153$). The external MLPG turned out to outperform the baseline and the 2 versions of embedded MLPG. A simple technique like smoothing proved to work relatively well as it brought clear gains over the baseline, and provided a similar performance as the external MLPG. Finally, the usage of pre-training was found to be crucial when using embedded MLPG. It is worth noting the correlation between these results and the findings with the F0 fluctuations. Considered in addition to the effectiveness of the smoothing technique, this highlights the issue of *shakiness* in the RNN outputs and their dramatic perceptual impact.

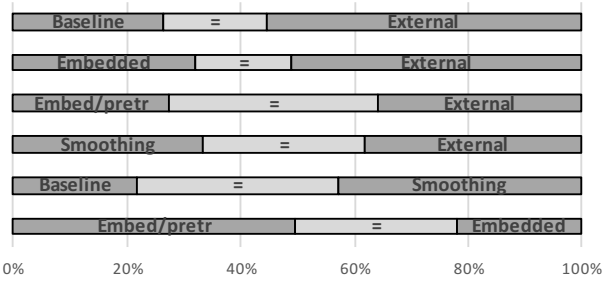

Fig. 5. Preference tests comparing parameter generation techniques.

6.3. Which loss function?

The objective metrics in Figures 3 and 4 are relatively comparable between the L1 and L2 losses. It is questionable which of those two approaches gives the best percetual results. For this purpose, we compare two systems built with the 7-hour dataset. The two systems use regularization and do not apply variance restoration. The first system uses the L1 loss without parameter generation algorithm (since MLPG was shown to be useless in the *L1, reg.* case), while the second one uses the L2 loss with external MLPG (best system using the conventional L2 loss).

Our perceptual test indicated a statistically significant preference for the first system. The *L1, no MLPG* system was indeed preferred in 41.4% of cases, against 27.8% for the *L2, external MLPG* system.

This is an interesting result as it demonstrates the potential in using the L1 loss during training.

6.4. Is variance restoration useful?

The dynamics of the predicted contours is known to be generally lower than that of natural speech. This results in a flatter synthetic speech. This is confirmed in Table 1 which presents the ratio of the Global Variance (GV) compared to natural speech. The GV ratio of the generated F0 is found to vary between 45 and 72%, depending on the configuration. The application of external MLPG reduces the GV ratio by 15% relative on average. Using a L2 instead of a L1 loss also degrades the GV ratio (from 25% at 1 h to 8% at 20 h). As for the regularization methods, they do not seem to have much effect ($< 5\%$ relative). Finally, it should be noticed that the GV ratio slowly gets closer to 1 as the amount of training data increases.

Loss	Regul.	Param.Gen.	1 h	7 h	20 h
L1	No	Baseline	0.65	0.655	0.724
L1	No	External	0.505	0.561	0.626
L1	Yes	Baseline	0.627	0.655	0.72
L1	Yes	External	0.469	0.579	0.571
L2	No	Baseline	0.492	0.666	0.641
L2	No	External	0.455	0.448	0.613
L2	Yes	Baseline	0.467	0.597	0.63
L2	Yes	External	0.457	0.516	0.545

Table 1. Global Variance ratios compared to normal speech.

Applying the multiplication or SVL techniques explained in Section 4 allows to alleviate this issue. With these methods, the GV ratio got much closer to 1 (between 0.92 and 1.01 in our experiments). We conducted preference tests to see if those improvements also translated to perceptual gains. For those tests, we chose the following configuration: 7 h, L1, regularized, external MLPG. Compared to the baseline (without any variance restoration), SVL improved (35.6% vs. 30.6%) but not significantly ($p = 0.132$). The simple multiplication technique outperformed the baseline (37.8% vs. 31.2%), but again not significantly ($p = 0.07$). Finally, multiplication performed significantly better than SVL (39.8% vs. 29.2%, $p = 0.009$). We carried out also tests at 20 h, which went in the exact same direction but with even less strong levels of statistical significance (probably due the fact that the baseline GV ratio is higher at 20 h).

7. EXPERIMENTS ON RNN-BASED TTS

In Section 6, we have analyzed the impact of the studied factors on F0 modeling. Our goal is now to investigate whether those observations translate to a RNN-based TTS where the factors are applied to all streams.

7.1. When is MLPG useful/useless?

We focus here only on the external MLPG based on our conclusions from Section 6.2. In terms of objective measurements, we found out that the external MLPG reduces the MCD by only about 0.1 dB when using the L2 loss, and does not help in the case of the L1 loss. This is not much knowing that the overall MCD is around 5 dB across our systems. As for the smoothing technique, it slightly degraded the MCD by about 0.1 dB compared to the baseline.

Table 2 displays the results of preference tests for various settings of the loss, regularization and parameter generation techniques.

System A	A pref.	Equal	B pref.	System B
L2,no,base	21.4%	35.6%	43.0%	**L2,no,MLPG**
L2,yes,base	22.2%	39.0%	38.8%	**L2,yes,MLPG**
L1,yes,base	36.4%	28.8%	34.8%	L1,yes,MLPG
L2,no,base	32.8%	34.0%	33.2%	L2,no,smooth
L2,no,smooth	18.0%	32.8%	49.2%	**L2,no,MLPG**
L1,yes,base	36.4%	37.6%	26.0%	L2,yes,MLPG

Table 2. Preference tests for various (loss, regularization, parameter generation) settings of RNN-based TTS systems trained on 7 h.

All statistically significant differences are shown in bold. As seen from the first 3 lines, MLPG turns out to be beneficial when using the L2 loss (with and without regularization) but does not bring any gain when using the L1 loss with regularization ($p = 0.377$). The conclusion of Section 6.1 is thus corroborated.

Looking at the 4^{th} and 5^{th} lines of Table 2, it is noticed that the smoothing method gives comparable results to the baseline ($p = 0.482$) and is clearly outperformed by the external MLPG. Smoothing the F0 trajectory proved to help in Section 6.2, but those gains do not translate to other streams. Especially, smoothing the MGC features made the speech more muffled.

7.2. Which loss function?

The last line of Table 2 shows that the conventional system using the L2 loss and MLPG is outperformed by that using the L1 loss without MLPG ($p = 0.011$). This confirms our findings from Section 6.3. While it does not seem reasonable to fit a Laplace distribution to most acoustic features, the robustness of the L1 norm to outliers seems to make it an appropriate loss to train a RNN-based TTS system. It is worth noticing that the L1 loss only improved the MCD by less than 0.1 dB compared to its L2 counterpart.

7.3. Is variance restoration useful?

For this test, we used the same configuration as in Section 7.3: L1, regularized, external MLPG. We tested whether the simple multiplication technique brings an improvement over the baseline (i.e. without variance restoration). Our results show a significant preference towards the multiplication method (44.8% vs 25.8%). While the gains were only minor for F0 in Section 6.4, they are more pronounced when applied to all streams. It seems indeed important to enhance the dynamics of the MGC features. Note that, as expected, the MCD was slightly degraded when using variance restoration (less than 0.1 dB though).

8. CONCLUSION

This paper investigated the effectiveness of parameter generation techniques in the context of RNN-based TTS. We explored the influence of various effects such as the loss function, regularization techniques and the amount of training data on the performance of parameter generation algorithms and variance restoration approaches. It was found that the external MLPG does *not* bring any gain when using the L1 loss and proper regularization, but turned out to be beneficial in other scenarios. In the case of embedded MLPG, pretraining appeared to be crucial. The external MLPG was found to be the best parameter generation strategy. A simple smoothing method improved the generation of F0 trajectories, but made the speech more muffled when applied to cepstral features. Although techniques restoring the variance only brought minor perceptual gains

on F0, they seem to be important for cepstral features in order to make the speech more dynamic. Finally, we show that the use of the L1 loss with regularization outperforms any system built with the conventional L2 loss. Based on our extensive evaluation, the recommended guideline to build a RNN-based TTS system is to use the L1 loss, proper regularization and variance restoration techniques, while the use of MLPG seems optional in that configuration.

9. REFERENCES

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

[2] A. W. Black, C. L. Bennett, B. C. Blanchard, J. Kominek, B. Langner, K. Prahallad, and A. Toth, "Cmu blizzard 2007: A hybrid acoustic unit selection system from statistically predicted parameters," in *Blizzard Challenge Workshop, Bonn, Germany*, 2007.

[3] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[4] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 1315–1318.

[5] K. Tokuda, H. Zen, and A. W. Black, "An hmm-based speech synthesis system applied to english," in *IEEE Speech Synthesis Workshop*, 2002, pp. 227–230.

[6] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7962–7966.

[7] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3844–3848.

[8] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[9] K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "The effect of neural networks in statistical parametric speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4455–4459.

[10] X. Wang, S. Takaki, and J. Yamagishi, "An autoregressive recurrent mixture density network for parametric speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4895–4899.

[11] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4470–4474.

[12] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Sequence generation error (sge) minimization based deep neural networks training for text-to-speech synthesis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[13] Z. Wu and S. King, "Minimum trajectory error training for deep neural networks, combined with stacked bottleneck features." in *INTERSPEECH*, 2015, pp. 309–313.

[14] K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Trajectory training considering global variance for speech synthesis based on neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5600–5604.

[15] M. Shannon, H. Zen, and W. Byrne, "Autoregressive models for statistical parametric speech synthesis," *IEEE transactions on audio, speech, and language processing*, vol. 21, no. 3, pp. 587–597, 2013.

[16] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[20] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[21] M. Morise, F. Yokomori, and K. Ozawa, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[22] S. W. Group *et al.*, "Speech signal processing toolkit (sptk)," *http://sp-tk. sourceforge. net*, 2009.

[23] T. Nose, "Efficient implementation of global variance compensation for parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 10, pp. 1694–1704, 2016.

[24] J. Kominek and A. W. Black, "The cmu arctic speech databases," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[25] K. Ito, "The lj speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[26] L. Technologies, "Lessac technologies, inc. voice release for blizzard 2011," http://http://data.cstr.ed.ac.uk/blizzard2011/lessac, 2011.

[27] P. C. Bagshaw, S. Hiller, and M. A. Jack, "Enhanced pitch tracking and the processing of f0 contours for computer aided intonation teaching," in *Third European Conference on Speech Communication and Technology*, 1993.

[28] J. Yamagishi and T. Kobayashi, "Average-voice-based speech synthesis using hsmm-based speaker adaptation and adaptive training," *IEICE TRANSACTIONS on Information and Systems*, vol. 90, no. 2, pp. 533–543, 2007.