The Horizon TTS System for Blizzard Challenge 2019

Yi Zhang, Dameng Hu, Wuwen Yuan, Wei Jiang

Nanjing Institute of Advanced Artificial Intelligence, Nanjing, P.R. China

{yi02.zhang, dameng.hu, wuwen.yuan, wei.jiang}@horizon.ai

Abstract

This paper describes the text-to-speech (TTS) system that we used for the Blizzard Challenge 2019. Unlike the challenge held before, the corpus of this year is composed of Chinese Mandarin from a male Chinese speaker. We build a cascade system for this task. Given input text, the linguistic features are extracted by an untrainable text analyzing system as a 476-dimensional vector, which is transformed to 80-dimensional mel-spectrum by a DCTTS-based acoustic model. The mel-spectrum is then utilized as local conditions by a WaveRNN-based vocoder to generate μ-law encoded 9-bit audio, and finally decoded to 16-bit. Listening evaluation results shows that our system, with indicator Y, performs well in terms of the naturalness and the similarity with the original speaker, and the intelligibility needs to be improved.

Index Terms: speech synthesis, neural network, Blizzard 2019

1. Introduction

The text-to-speech (TTS) system has attracted the attention of many researchers, due to its importance in human-computer interaction. A conventional TTS system is essentially composed of three parts, a text analyzing system that extracts linguistic features from the text input, an acoustic model transforming linguistic features to acoustic features, and a speech synthesis system, or so-called vocoder, that generates speech audio based on acoustic features. In some cases, the text analyzing system and the acoustic model are also called the front-end, and the synthesis system is referred to as the back-end, which will utilize the linguistic and acoustic features collectively [1].

The conventional text analyzing system typically consists of tokenization that normalizes the input text, and grapheme-to-phoneme conversion that produces the well-designed linguistic features, which include grammatical properties and phonological features. Then the acoustic model transforms these verbal-based features to prosody-based features, such as logarithmic fundamental frequency (log F0), phone duration. The correlation between text and speech has been explored for a long time, and the prior knowledge is well integrated into these delicate handcrafted features.

The speech synthesis system has been dominated by two approaches. The first is the unit-selection based concatenative synthesis, which selects the appropriate audio units from a natural speech database according to the linguistic and acoustic features [2]. The second one is the statistical parametric speech synthesis (SPSS), which learns from the speech database and forms a generative model, mostly a hidden Markov model (HMM). Meanwhile the so-called hybrid synthesis combining these two methods, uses a statistical parametric model to guide which units should be selected [3, 4]. We prefer to categorize them as concatenative synthesis and generative synthesis to emphasize whether or not a speech database is required during synthesis. The unit-selection based concatenative synthesis is known for its high naturalness since it reuses the real speech, but suffers from its lack of flexibility. The generative synthesis has the main advantage of being able to generate different kinds of speech (*e.g.*, change the speaker) by transforming its model parameters. However, the generated speech might be slightly inferior to that of the concatenative synthesis in terms of naturalness[5].

With the great improvement achieved by deep neural networks (DNNs) in many areas (*e.g.* computer vision, nature language processing, *etc.*), many attempts have been made to incorporate DNN into speech synthesis. The main idea is to estimate a more accurate DNN-based model to replace the original structure that relies heavily on prior information [6, 7].

For the text analyzing system, Deep Voice 1&2 [8, 9] and Char2Wav [10] are intended to extract linguistic features from text directly using the black box DNN model, instead of using the well-designed classic algorithms.

For the acoustic model, Tacotron 1&2 [11, 12] employ a RNN-based sequence-to-sequence model to generate mel-spectrum, a well-behaved acoustic feature for the following vocoder. The RNN-based acoustic model has been widely accepted as the state-of-the-art. However, this model exploits so many recurrent units, resulting in high computational burden and slow convergence. Since CNNs can get more benefits from GPU's acceleration than RNNs, the fully convolutional sequence-to-sequence model is introduced in Deep Voice 3 [13]. DCTTS [14] adopts a similar CNN-based acoustic model, and reaches the state-of-the-art performance with significantly improvement of the convergence speed than Tacotron.

For the speech synthesis system, according to [15], DNN leads to more improvement on the generative synthesis over the unit-selection based concatenative synthesis. WaveNet [16], a fully convolution neural network, combines linguistic features and acoustic features as local conditions, and produces speech that even more natural than that of the concatenative systems. However, during the synthesis phase, WaveNet has to generate audio point by point, so it runs very slowly thus unsuitable in real-time systems. In addition, the synthesis process is unstable and occasionally produces high-energy broadband noise, as discussed in another study [17]. This instability causes a significant drop in both naturalness and intelligibility as a consequence. Meanwhile, RNN-based systems can be expected to provide a more stable output due to the memory of historical information in their hidden state [18]. WaveRNN [19] can generate real time high-quality audio even on a mobile CPU, with performance that still matches the state-of-the-art. This competitive compact recurrent neural network has been further improved in [18]. And there are also some other networks that show the benefits of DNN-based generative systems, such as Waveglow [20] and ClariNet [21].

We reviewed all the 10 papers in Blizzard Challenge 2018, and found that there are 4 concatenative systems and 6 generative systems, all based on DNNs. USTC's concatenative sys-

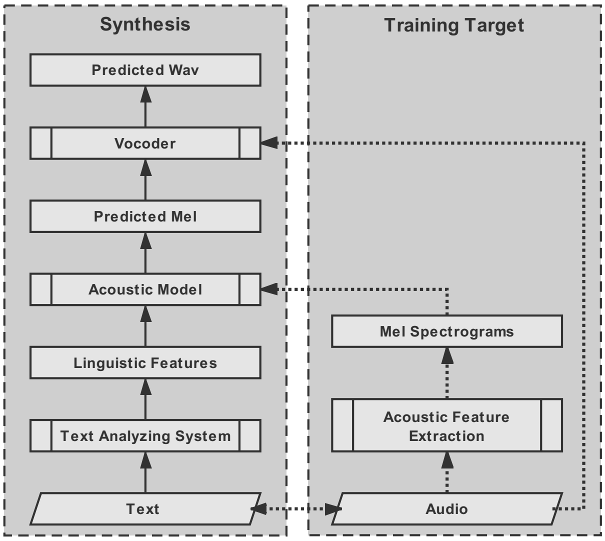

Figure 1: *The flowchart of our proposed TTS system.*

tem [22] got the best performance, demonstrating the superiority of the concatenative system. However, it has the inevitable disadvantage of the reliance on the specific speech database, which limits its application on resource-constrained end devices, *e.g.* mobile phones and smart speakers. Therefore, the study and implementation of the generative synthesis is still very appealing. Thus, for Blizzard Challenge 2019, we used a two-stage system with a DCTTS-based acoustic model and a WaveRNN-based vocoder.

The paper is organized as follows. Section 2 describes our TTS system, followed by the evaluation results in section3. The conclusions are drawn in section4.

2. TTS System

Tacotron2 [12] combines the neural acoustic model and the neural vocoder to an end-to-end system, with mel-spectrum as the intermediate feature representation. The acoustic model and vocoder can be trained separately, which inspired us to try more combinations of the state-of-the-art acoustic models and vocoders.

As shown in Figure 1, our proposed TTS system consists of three sub-modules, the untrainable text analyzing system, followed by the trainable acoustic model and the vocoder. The acoustic model is mainly based on DCTTS[14] and Deep Voice 3[13], and the vocoder is mainly based on WaveRNN[19] with Amazon's modification[18]. The implementation details will be presented in below subsections, respectively.

During the training phase, the <text, audio> pairs were selected from the given dataset, then the linguistic features and the mel-spectrum were extracted and used as the input and the target to train the acoustic model. As addressed in [17], the inconsistency between the mel-spectrum distributions of training and decoding (synthesizing) may causes unstable synthesis. Thus we jointly fine-tuned the acoustic model and vocoder, so that the vocoder can learn the distribution of the mel-spectrum generated by the acoustic model. During synthesis phase, the synthesized speech was generated by passing the given text through the left bottom-up path depicted in Figure 1.

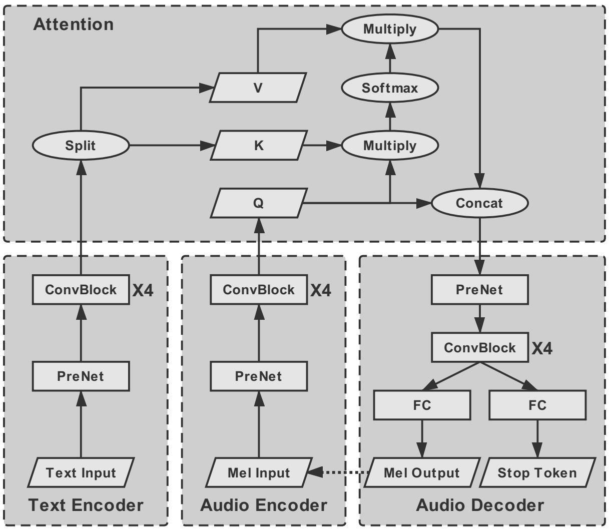

Figure 2: *Acoustic model. The modified DCTTS architecture.*

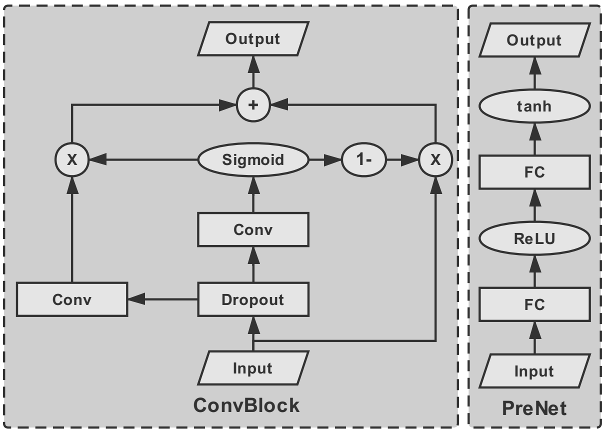

Figure 3: *Components of the acoustic model. The ConvBlock and the PreNet, respectively.*

2.1. Data processing

About 8 hours of speech data from an internet talk show by a well-known Chinese character, Zhenyu Luo, was provided, including audios and the corresponding texts. The given *.mp3* files were first transformed to *.wav* files, and then the 80-dimensional mel-spectra were extracted with 40 ms frame size and 10 ms hop size. The corresponding texts were manually embedded into 476-dimensional vectors using our own text analyzing system. The embedded vectors consisted of one-hot encoded phonemes, tones, part-of-speech, prosodic boundaries and the position information. The prosodic boundaries were composed of phoneme boundaries, syllable boundaries, phrase boundaries, secondary phrase boundaries, *etc.* The position information was composed of the forward and backward positions of the phonemes and the syllables, *etc.*

2.2. Acoustic Model

In our implementation, the DCTTS has been modified with reference to Deep Voice 3. For the sake of brevity, we will still refer to it as DCTTS. As shown in Figure 2, the DCTTS has three sub-modules, a text encoder, an audio encoder and an audio decoder, which are connected by an attention module. The text encoder includes a PreNet followed by four stacked convolution blocks. As shown in Figure 3, the PreNet consists of two stacked fully-connected layers followed by a ReLU and a tanh non-linearity activation, respectively. The convolution block is a highway network [23] with dropout rate of 0.05 before the convolution layer. The output of the text encoder is divided into *value* and *key*, which are then fed into the attention module. The output of the audio encoder are used as *query* to get the corresponding feedback from the attention module. Similar to the encoder, the audio decoder has a PreNet followed by four stacked convolution blocks, then the mel-spectrum and the stop-token are generated by two fully-connected layers respectively.

The training target is to minimize the loss function below:

$$\begin{aligned} L = &\sum |A * W| \\ &+ \sum |y_{mel} - \hat{y}_{mel}| \\ &- [y_{stop}\log\hat{y}_{stop} + (1 - y_{stop})\log(1 - \hat{y}_{stop})]. \end{aligned} \tag{1}$$

The first term is the guided attention loss, where A is the attention matrix from the attention module, and the weight matrix W is calculated as

$$W_{n,t} = 1 - e^{\frac{(n/N-t/T)^2}{2g^2}}, \tag{2}$$

where N and T represent the number of frames of linguistic features and mel-spectra, respectively, and g is the adjustment factor, which is set to 0.2 [14]. By minimizing $|A * W|$, the attention matrix A is “guided” to be “nearly diagonal”, which satisfies the constraint that the system must pronounce the word one by one in the order of the corresponding text sequence. To allow the attention of consecutive mel-spectrum frames on the same linguistic feature, we relax the strong “incremental” constraint by thresholding W as

$$W_{n,t} = \begin{cases} 1 & W_{n,t} \geq \alpha \\ 0 & W_{n,t} < \alpha, \end{cases} \tag{3}$$

where α is the mean value of W.

For the other two terms, y and $\hat{y}$ represent the ground truth and the predicted output, respectively. The subscripts mel and $stop$ stand for the mel-spectrum and the stop-token, respectively. For the mel-spectrum prediction, L1 loss is used to reduce the impact of outliers, and cross-entropy loss is used for the prediction of the stop-token. During the training process, the teacher-forcing scheme was used that the audio encoder module takes ground truth mel-spectrum as input rather than the corresponding prediction. The Adam [24] optimizer with constant learning rate 1e-4 was adopted. During the synthesis process, the output mel-spectrum of the audio decoder was fed back to the audio encoder in an autoregressive form.

2.3. Vocoder

As shown in Figure 4, WaveRNN is composed of a condition module that encodes mel-spectrum input to local conditions, and an autoregressive module that generates audio output according to given conditions. In the condition module, mel-spectrum is passed through 2 branches. One is progressively upsampling with factors 5, 6 and 8, using stretch followed by convolution. The other is 240-fold upsampling directly after 10 stacked residual blocks. The upsampling factor 240 corresponds to the 10 ms hop size at 24 kHz sampling rate. Therefore, the number of upsampled feature frames is equivalent to the number of samples of the final audio. The stretch operation duplicates each frame along the time dimension. The two results are concatenated together as the final output of the condition module. In the autoregressive module, the corresponding frame of the condition module’s output at current time step is concatenated with the embedded audio signal from the previous time step. Then the combined features are fed into a single layer GRU with 896 nodes followed by two fully connected layers. Then a softmax layer is used to predict the categorical distribution.

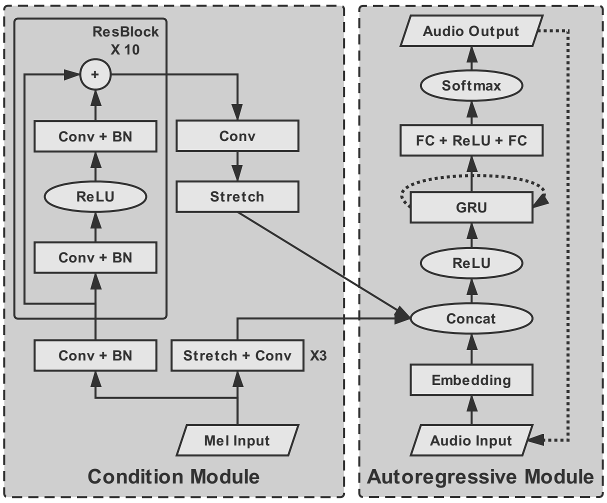

Figure 4: *Vocoder. The modified WaveRNN architecture.*

The original WaveNet predicts a 256-dimensional categorical distribution for each time point to generate μ-law encoded 8-bit audio, which is then decoded to 16-bit. Unlike the traditional Gaussian distribution, the categorical distribution does not introduce any hypothetical prior information, so it can model arbitrary distributions. However, the quantization in the μ-law encoding process will inevitably lead to the loss of information, thus reducing the upper limit of the model. In order to generate 16-bit audio directly, the mixture of logistics (MoL) distribution is introduced in Parallel WaveNet [25]. However, MoL may cause instability in the training process, so as a compromise, we set the training target as μ-law encoded 9-bit audio.

During the training process, the negative log likelihood loss was calculated between the categorical distribution and the ground truth distribution, the corresponding one-hot vector. The autoregressive module was trained in the teacher-forcing scheme. The Adam optimizer with constant learning rate 1e-4 was adopted.

During the synthesis process, the predicted output was sampled from the categorical distribution, and fed back to the autoregressive module as historical audio input. The motivation for using sampling is to reduce quantization error. Given a random variable X, the value with maximum probability is referred as x_{maxP}. The random variable $Y = X - x_{maxP}$ shares the same probability distribution with X, *i.e.* $P\{X = x_i\} = P\{Y = y_i\}$. Therefore, sampling directly from X is equivalent to sampling from Y and then adding to x_{maxP}. This process can be seen as first selecting the probability maximum as 9-bit classification result, and then adding with the categorical

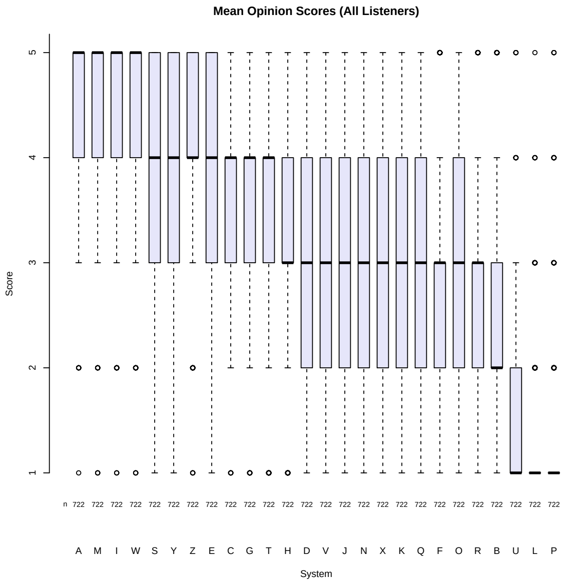

Figure 5: *The evaluated mean opinion scores.*

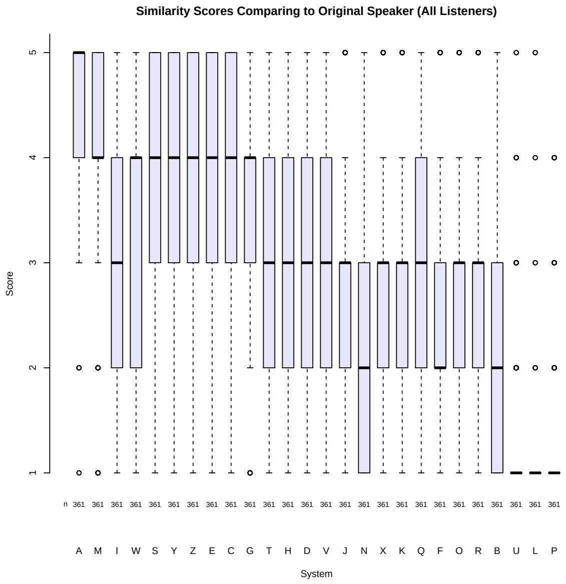

Figure 6: *The evaluated similarity scores comparing to original speaker.*

distribution based noise, also known as dither. Dither is the most common means of reducing the quantization error during bit-depth transforming. The experimental results show that the sampling scheme in the synthesis stage achieves significant improvement over the direct selection of the maximum probability.

3. Results

In this year, there are a total of 26 synthesized results/ground-truth are evaluated, denoted with capital letter *A* to *Z*, of which the natural speech is indicated with *A*, the baseline results proposed by Merlin [26] is indicated with *B*, while our proposed system is indicated with *Y*.

The evaluation is achieved using 3 different metrics: a) Naturalness by the Mean Opinion Score (MOS); b) Similarity comparing to the original speaker by the Similarity Score (SIM); c) Intelligibility by the Pinyin Error Rate with/without Tones (PTER/PER). Below in each subsection, we will report the detailed results.

3.1. Naturalness Test

The mean opinion score is the overall subjective score judged by human listeners focusing on the naturalness of the speech. The human listeners are asked to assign a score from 1 to 5 to the sample they just listened. A score of 1 means the speech is *completely unnatural*, whilst a score of 5 stands for the speech is *completely natural*.

The results are given in Figure 5. Our proposed system obtains an average score of 4.0 with 0.9 standard deviation. For reference, the natural speech has a score of 4.7 with 0.63 standard deviation, and the baseline performance provided by Merlin only gets a score of 2.5 with 1.14 standard deviation.

3.2. Similarity Test

The similarity score measures if the human listeners consider the voice in a synthesized sample is similar to the voice in the two given reference samples. The score is also distributed from 1 to 5, with 1 stands for the voice sounds like *a totally different person*, and 5 stands for the voice sounds like *exactly the same person*.

The results are shown in Figure 6. Our proposed system obtains an average score of 3.8 with 1.18 standard deviation. For reference, the natural speech has a score of 4.5 with 0.79 standard deviation, and the baseline performance provided by Merlin only gets a score of 1.5 with 1.03 standard deviation.

3.3. Intelligibility Test

The intelligibility is evaluated by dictation. Concretely, the human listeners are asked to write down the contents they heard from the given samples. During the calculation of the performance, the contents are converted to the pronunciation of these characters, namely Pinyin. The Pinyin is an intuitive Chinese auditory feature, can eliminate the influence of homophones when calculation the word error rate. Moreover, the results are post-processed to versions with and without tones, which is also an important element of the pronunciation of the Chinese characters. The gap between PTER and PER shows the tone error rate in the correctly recognized text, representing the loss of tone information during synthesis.

The results are shown in Figure 7 for PTER and Figure 8 for PER. Our proposed system Y got 17.3% PTER and 15.7% PER, slightly better than the Merlin baseline with 21.6% PTER and 20.5% PER. And the comparison of PTER and PER is shown in Figure 9, our system Y got 4.8% gap, much larger than the 2.6% gap from Merlin base line, indicates the tone reconstruction error for our system. which should be considered in the future work.

3.4. Discussion

We review several sample speeches synthesized by our proposed system and observe two types of defects that may severely decrease the performance.

Unexpected noise: We notice that meaningless noise occurs

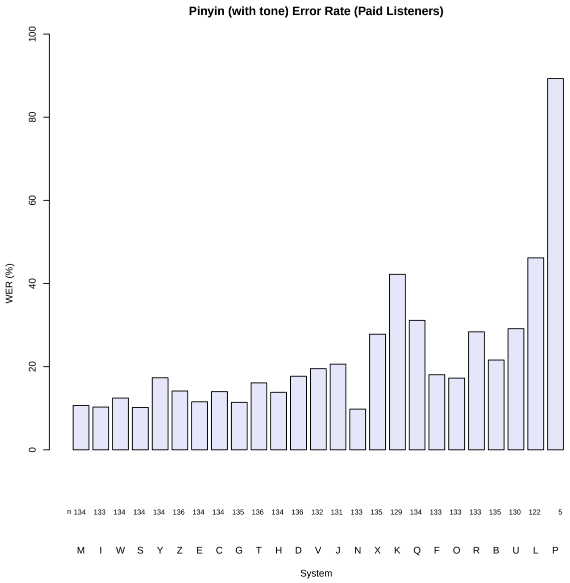

Figure 7: *The evaluated Pinyin error rate with tone.*

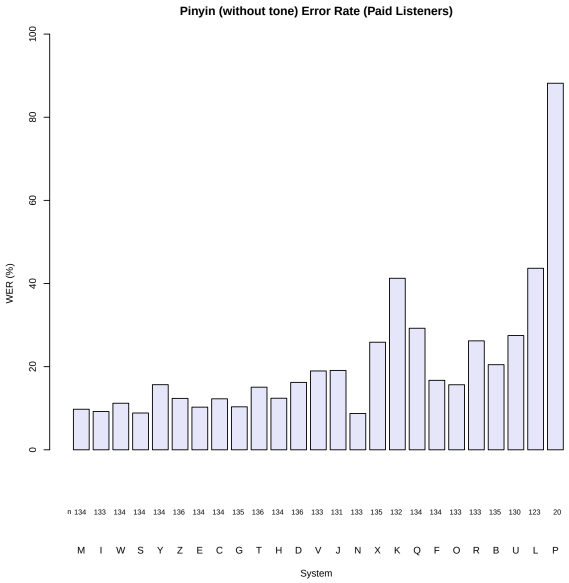

Figure 8: *The evaluated Pinyin error rate without tone.*

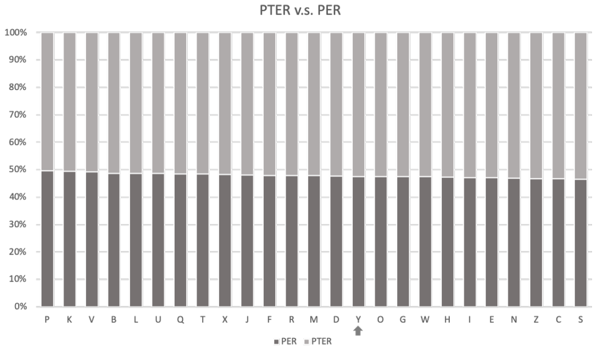

Figure 9: *Differences between PTER and PER.*

occasionally, causes a significant drop in both naturalness and intelligibility as a consequence. One possible reason is the teacher-forcing scheme. During the training process, the historical prediction errors were ignored by the teaching-force scheme. However, during the synthesis process, any historical prediction errors were accumulated, and affect the prediction of the next frame. And after vocoder, these errors may cause the unexpected noise.

The non-pronounced word: We also notice that a small amount of words are skipped during the pronunciation. This happens specially for the repetitive words. We conjecture that the intervals between the repetitive words may be not well maintained through the acoustic model. Hence, the vocoder might view them as one word with longer occasion rather than multiple words. More specific reasons and solutions are left for future studies.

4. Conclusion

We described our proposed TTS system for Blizzard Challenge 2019, which consists of deep neural network based acoustic model and vocoder. Our system performs better than the benchmark system, with 1.5-point higher MOS, 1.5-point higher SIM, 4.8% lower PER, and 4.3% lower PTER. We have noticed several drawbacks of our system by carefully investigating the evaluation results, currently we are focusing on further improvement correspondingly.

5. References

[1] P. Taylor, *Text-to-Speech Synthesis*. Cambridge: Cambridge University Press, 2009.

[2] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1. IEEE, 1996, pp. 373–376.

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

[4] S. King, "An introduction to statistical parametric speech synthesis," *Sadhana*, vol. 36, no. 5, pp. 837–852, 2011.

[5] S. Kayte, M. Mundada, and J. Gujrathi, "Hidden markov model based speech synthesis: A review," *International Journal of Computer Applications*, vol. 130, no. 3, pp. 35–39, 2015.

[6] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *2013 ieee international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 7962–7966.

[7] T. Merritt, R. A. Clark, Z. Wu, J. Yamagishi, and S. King, "Deep neural network-guided unit selection synthesis," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5145–5149.

[8] S. Ö. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman *et al.*, "Deep voice: Real-time neural text-to-speech," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 195–204.

[9] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Advances in neural information processing systems*, 2017, pp. 2962–2970.

[10] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, “Char2wav: End-to-end speech synthesis,” 2017.

[11] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, “Tacotron: Towards end-to-end speech synthesis,” *arXiv preprint arXiv:1703.10135*, 2017.

[12] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, “Natural tts synthesis by conditioning wavenet on mel spectrogram predictions,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 4779–4783.

[13] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, “Deep voice 3: Scaling text-to-speech with convolutional sequence learning,” *arXiv preprint arXiv:1710.07654*, 2017.

[14] H. Tachibana, K. Uenoyama, and S. Aihara, “Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 4784–4788.

[15] V. Wan, Y. Agiomyrgiannakis, H. Silen, and J. Vit, “Google’s next-generation real-time unit-selection synthesizer using sequence-to-sequence lstm-based autoencoders.” in *INTERSPEECH*, 2017, pp. 1143–1147.

[16] A. v. d. 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.

[17] Y.-C. Wu, K. Kobayashi, T. Hayashi, P. L. Tobing, and T. Toda, “Collapsed speech segment detection and suppression for wavenet vocoder,” *arXiv preprint arXiv:1804.11055*, 2018.

[18] J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, and R. Barra-Chicote, “Robust universal neural vocoding,” *arXiv preprint arXiv:1811.06292*, 2018.

[19] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. v. d. Oord, S. Dieleman, and K. Kavukcuoglu, “Efficient neural audio synthesis,” *arXiv preprint arXiv:1802.08435*, 2018.

[20] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 3617–3621.

[21] W. Ping, K. Peng, and J. Chen, “Clarinet: Parallel wave generation in end-to-end text-to-speech,” *arXiv preprint arXiv:1807.07281*, 2018.

[22] J. Yuan, Z. Xiao, D. Chuang, H. Ya-jun, L. Zhen-Hua, and L.-R. Dai, “The ustc system for blizzard challenge 2018,” in *Blizzard Challenge Workshop*, 2018.

[23] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *arXiv preprint arXiv:1505.00387*, 2015.

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

[25] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg *et al.*, “Parallel wavenet: Fast high-fidelity speech synthesis,” *arXiv preprint arXiv:1711.10433*, 2017.

[26] Z. Wu, O. Watts, and S. King, “Merlin: An open source neural network speech synthesis system.” in *SSW*, 2016, pp. 202–207.