The STC text-to-speech system for Blizzard Challenge 2019

Roman Korostik[1], *Artem Chirkovskiy*[1], *Alexey Svischev*[1], *Ilya Kalinovskiy*[1], *Andrey Talanov*[1]

[1]STC-innovations Ltd., St.Petersburg, Russia

{korostik, chirkovskiy, svishchev, kalinovskiy, andre}@speechpro.com

Abstract

The paper presents text-to-speech system developed at STC for the Blizzard Challenge 2019. This year, the task is to build a TTS system for Mandarin Chinese using found data suitable for expressive TTS. Provided corpus contains 8 hours of speech by a native speaker with text annotations.

We describe a neural speech synthesis system for Mandarin Chinese built without any significant prior knowledge about the language. Input text is converted to a sequence of phones using publicly available tools. Then, a sequence of phones is turned into a spectrogram by a Tacotron-based neural network. Finally, the spectrogram is converted into a waveform using a LPCNet-based neural network. Our system is based on learning deep representations and does not explicitly use or predict such features as pitch, duration of every phone, etc.

We also discuss our system's performance in listening tests conducted by organizers of the challenge.

Index Terms: speech synthesis, deep learning, text-to-speech

1. Introduction

The Blizzard Challenge is aimed at comparing and understanding different approaches to building corpus-based text-to-speech systems. The basic task is to build a TTS system in limited time using speech database provided by organizers of the challenge. Participants use their systems to synthesize texts from test set; these audio samples are used to assess systems' perfomance through listening tests and objective metrics.

Traditionally, TTS systems solve their task by dividing process into two parts. First, input text is normalized and transformed into linguisic features of different levels: phone-level, syllable-level, word-level and so on. Then, these linguistic features are transformed into waveform by a heavy engineered module. Usually, extracting linguistic features and building a backend requires a lot of expert knowledge in linguistics. The task may also require time-consuming labeling of speech corpora by hand [1].

Due to recent developments in field of deep learning, a new appealing approach to building TTS backend has emerged. It involves training one or two deep neural networks and requires lesser expert knowledge and expert-labeled data. Resulting systems are able to produce intelligible and natural speech while being trained only on pairs (text, waveform) with no extra labeling.

Usually, these backends consist of two blocks: first block is a deep neural network transforming input sequence of graphemes or phonemes into sequence of acoustic features. Examples include Tacotron [2, 3], Char2Wav [4], DeepVoice [5, 6, 7], VoiceLoop [8], DC-TTS [9]. Second block is a vocoder generating a waveform using features from previous stage. A vocoder may be an iterative algorithm such as Griffin-Lim [10] algorithm, an algorithm built on expert knowledge about speech signal (e.g. STRAIGHT [11], WORLD [12]), or a deep neural network such as WaveNet [13, 3], WaveRNN [14], FFTNet [15], LPCNet [16], ClariNet [17], WaveGlow [18].

We stick to this new appealing approach. First, input text gets normalized and translated into a sequence of phones. Then, a sequence of phones is converted to a mel-filterbank spectrogram by a Tacotron-based deep neural network. Finally, an LPCNet-like neural vocoder converts mel-filterbank spectrogram into a waveform.

Following sections describe data, data preprocessing and each step of the pipeline, as well as discussion of evaluation results.

2. Data and Task

The challenge focuses on found data. The task is to build voice of a single Mandarin Chinese speaker using 8 hours of their speech gathered from an internet talk show. Dataset consists of 1 minute chunks of speech with a corresponding text for each chunk. Provided records have one channel (with a couple of exceptions), 24 kHz sampling rate, 16 bit depth, and are lossily compressed (LAME MP3, 40 kbit/s).

Provided records vary in quality: some of them are rather clean, others have clipping and contain background noise.

While rules of the challenge explicitly allow usage of extra speech not from target speaker, we did not use any.

3. Our approach

The section describes our approach to building a TTS system for the challenge. It must be noted that nobody on the team has any proficiency in Mandarin Chinese.

3.1. Text preprocessing

Chinese characters (also called Hanzi) sometimes can be read in a completely different ways depending on the context. The aim of text preprocessing is to build a pipeline converting input Hanzi text into a sequence which would be more representative of an utterance's pronounciation. The idea behind this approach is that having a closer-to-pronounciation text representation as an input to an end-to-end speech synthesizer should result in easier training of the synthesizer and require less data.

A typical Mandarin Chinese text is a sequence of Hanzi without spaces. As a first step, we split the text into words using jieba [19] Python package, which takes the most probable word segmentation based on word usage statistics. The aim of this step is to account for possible ambiguity of word boundaries.

Pinyin is an alphabetic system for Mandarin Chinese representative of its phonetics. We wish for closer-to-pronounciation representation of text, so every word is transcribed into Pinyin using pypinyin [20] Python package.

Basic unit of Pinyin is syllable, and there are over 1000 possible syllables. With an aim to bring down input characters vocabulary size and to get closer to pronounciation we convert

pinyin sequences into phones using a pre-trained Mandarin Chinese pinyin-to-phones Phonetisaurus model from the Montreal Forced Aligner [21]. This also results in vowels with different tones being considered as completely different input characters.

Texts contain a small number of English words. As learning representations for English phonemes from such data seems unfeasible, we convert English words to phonemes using g2p_en Python package and convert those to Mandarin Chinese phones. Mapping from English phonemes to Mandarin Chinese phones is hand-designed and is based on phoneme-IPA tables from Wikipedia.

In contrast to training set, test set contains a number of non-Hanzi and non-punctuation tokens such as characters denoting longitude and latitude. Test texts also contain tokens signifying time, date and various ranges. Attempts to find tools for Chinese text normalization have not succeeded, so a number of simple normalization rules has been written. The rules are based on various 'learning Mandarin Chinese: saying X correctly' articles from the web and are implemented as regular expressions. We also convert all numbers to Hanzi using a hand-written rule-based algorithm.

To sum up, text preprocessing pipeline is as follows:

1. Text is normalized by hand-written rules to have only Hanzi and punctuation.
2. Normalized text is split into tokens.
3. Each token is converted into pinyin.
4. Pinyin is converted into phones using pre-trained Mandarin Chinese pinyin-to-phones G2P model.
5. English words are converted into phones using a pre-trained G2P model for English and hand-designed mapping from English phonemes to Mandarin Chinese phones.
6. Punctuation from the original text is inserted into corresponding places of the phones sequence.

3.2. Alignment

An essential component of an end-to-end speech synthesizer is an attention module. It learns to align input character sequence with output spectrogram frame sequence. It is necessary to have text exactly correspond to speech signal: teaching the model to align sequences can be much harder if there are no correct alignments for some training examples.

Duration of examples is also important. When training a phones-to-spectrogam model on a NVIDIA GTX 1080Ti GPU, using whole one minute chunks of speech allows us to use only extremely small minibatches.

We address both of these concerns by using Montreal Forced Aligner [21]. It is a forced alignment system build on top of Kaldi speech recognition toolkit. The project offers a pre-trained Mandarin Chinese acoustic model, as well as a pre-trained G2P model for pinyin to phones conversion.

3.2.1. Pronounciation vocabulary

Montreal Forced Aligner requires a pronounciation vocabulary. Both pypinyin and Phonetisaurus allow for retrieving not only the most probable transcription, but also top-n most probable transcriptions. We use this to build the vocabulary. Top-k and top-j predictions from pypinyin and Phonetisaurus are being taken, resulting in at most $k \times j$ pronounciations for a given token. We use $k = j = 4$ resulting in 100 thousand vocabulary entries overall.

3.2.2. Initial experiments

The basic idea is that non-aligning segments should be considered unfitting the task and thrown out. Also, word-level and phone-level alignments can be used to split examples into lesser chunks. Of course, throwing out whole minutes of training data because of a smaller non-alignable segment is wasteful: running Montreal Forced Aligner with default settings results in only 5 of 8 hours being aligned. To counter this, we run Montreal Forced Aligner with a very wide beam ($retry\_beam = 450$) to get a much rougher alignment. Then we split roughly aligned examples by break-inducing punctuation such as commas, colons, semicolons and full stops. Finally, we run forced alignment on split examples with default, resulting in 6 hours of finely aligned speech.

3.2.3. Correcting annotations using automatic speech recognition

While hand-checking non-aligned chunks we noticed a lot of examples with completely non-matching texts or even without corresponding texts. There also were examples with audio and text differing only in interjections and short function words. We used an online ASR system from Wit.ai, Inc. [22] to deal with these samples. By combining original texts and texts received from service we obtained ASR-corrected texts in several versions, based on different suggestions of the ASR system. Montreal Forced Aligner was applied to all versions of combined annotations and for each data sample the version with highest alignment score was selected. A drawback of this approach is that for such homophonic language as Mandarin Chinese, result of automatic recognition can be quite meaningless in semantics.

This procedure allowed us to increase duration of aligned subset by 35 minutes.

3.2.4. Correcting segmentation

Phone sequences for audio samples were obtained from Montreal Forced Aligner segmentations. Hand-checking of segmentations revealed that the aligner denoted a lot of plosives as pairs of pause and a plosive, counting the 'hold' subsegment of the plosive as a pause. Correcting this slip-up resulted in more stable training of the synthesizer.

3.3. Synthesizer

We have chosen Tacotron 2 [3] as a starting point for our system. First, it has been proven to be suitable as a base for making TTS systems for different languages (e.g. English [3], Japanese [23] and even multi-language ones (English, Spanish and Chinese in [24]). Second, it is flexible: there is a lot of research into various add-ons and modifications to this architecture, aimed at improving overall quality and/or making some kind of prosody control feasible [25, 24, 26], improving stability [27], semi-supervised learning, transfer learning and efficient use of data [28, 29], using different architectures for submodules [30], etc.

It is augmented with a GST [25] module for unsupervised prosody modeling. For more stable decoding we employ Forward Attention [27]. In addition to base Tacotron 2 loss we use guided attention loss [9] for faster attention convergence and MS-SSIM [31] loss to make mel-spectrograms less blurry.

3.3.1. Prosody

GST module extracts global prosodic features and does not take into account content of an utterance. However, a person ex-

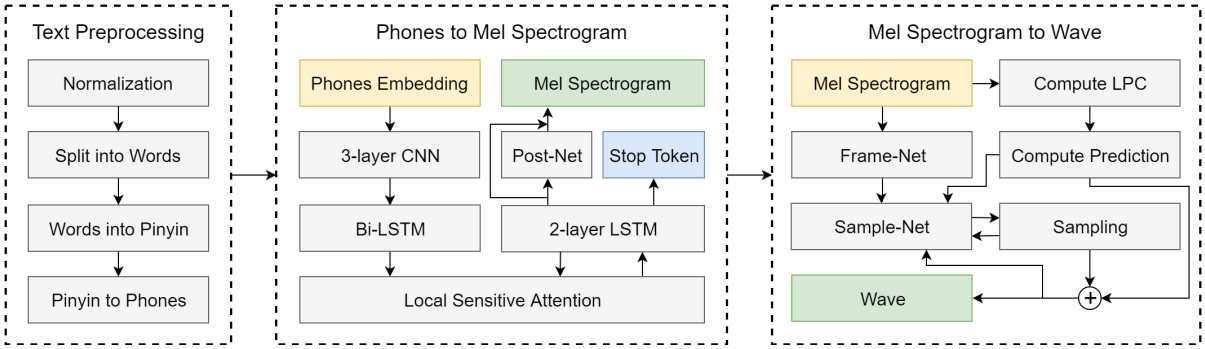

Figure 1: *Overview of our system*

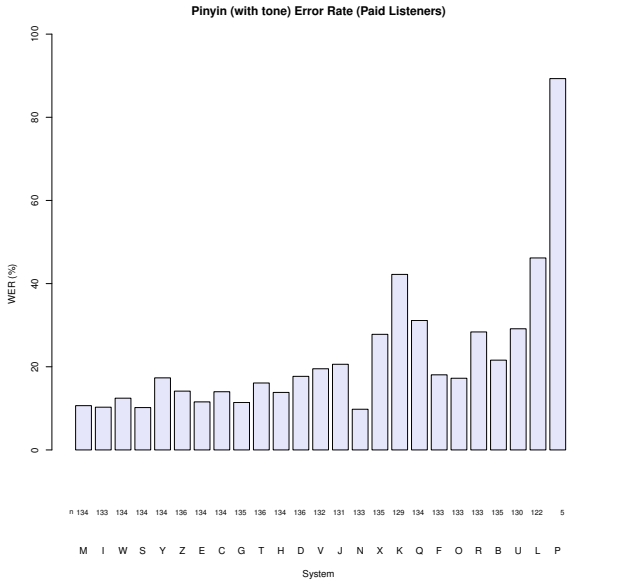

Figure 2: *Pinyin error rates (with tone)*

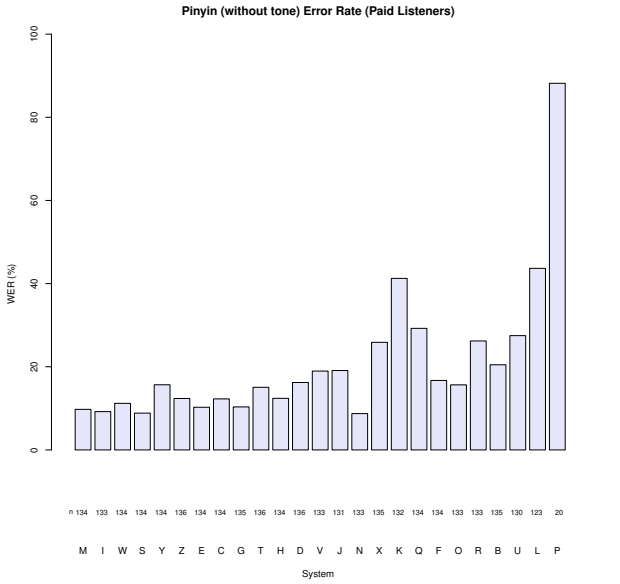

Figure 3: *Pinyin error rates (without tone)*

presses meaning of an utterance, among other things, by dividing their speech into chunks. Speech chunking and other prosody components are crucial for synthesized speech to be perceived as natural.

We deem it unfeasible to explicitly model prosody (e.g. correctly specify prosodic breaks) without labeled corpora and competence in Mandarin Chinese. Thus, we try to model local prosodic features implicitly by using acoustic features and relevant information from pre-trained models.

Punctuation is a predictor of prosodic breaks. A break's depth strongly correlates with pause duration (if present). We model breaks as pauses using auxiliary phones representing different pause durations. Every punctuation token is replaced by a sequence of pause-denoting phones summing to duration of silence behind that punctuation symbol. At inference time, we assume every punctuation token corresponds to a pause of constant duration, with different durations for diffent tokens. The punctuation tokens' durations are medians of corresponding pauses in the train set. This is the only place in whole system where we explicitly use anything related to phone duration.

Such linguistic phenomena as word compounds, lexical categories, etc. may affect prosody in subtle ways. To account for this in an implicit way, we use context-dependent text embeddings, namely output of a pre-trained Chinese BERT [32, 33] aligned with phone sequence. For silence phones we use vectors of corresponding punctuation tokens. After adding these vectors to encoder outputs we observed better pronounciation (as far as non-native speakers can judge) and better overall stability.

The challenge focuses on emotional speech synthesis. Emotions are expressed in speech through prosody, so we aim to model prosody well, hoping that the network will catch speaking style presented in provided recordings, which is quite emotional.

3.4. Vocoder

We use LPCNet [16] modified to take 80-band mel-filterbank spectrograms as input instead of MFCCs. It is a nice companion to Tacotron 2 in both quality and speed, allowing for real-time synthesis. It is also not the first published usage of LPCNet in text-to-speech systems [34].

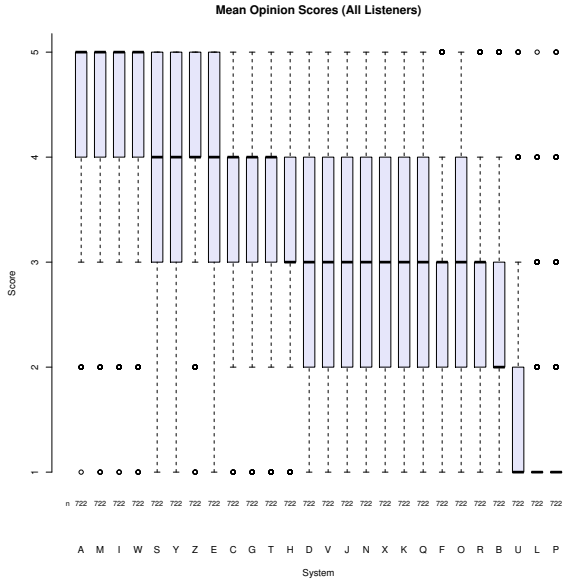

Figure 4: *Mean opininon scoring*

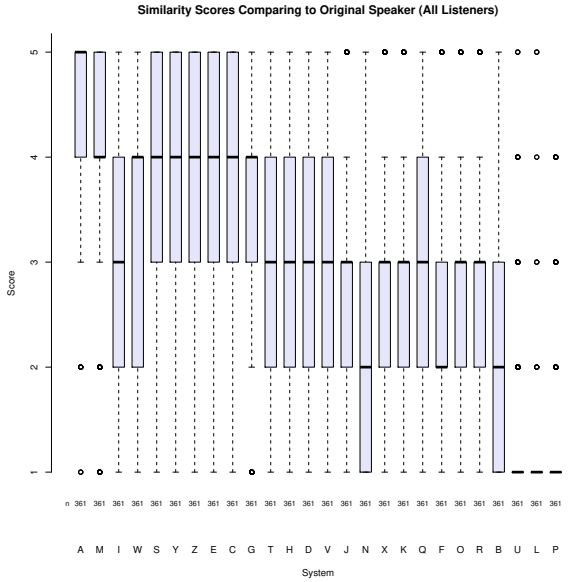

Figure 5: *Similarity scores compared to original speaker*

3.5. Parameters and training

Initial experiments to train LPCNet on original 24 kHz recordings have given unsatisfying results, the corpus has been downsampled to 16 kHz. Waveforms are converted to mel-filterbank spectrograms using librosa [35]. We use 80 mel-filterbanks, 50 ms windows and hops of 20 ms.

Tacotron 2 and GST module parameters and hyperparameters follow original papers [3, 25]. Context-dependent word embeddings from pre-trained BERT are concatented with outputs of Tacotron 2 decoder.

Synthesizer was trained on a machine with a single NVIDIA GTX 1080Ti GPU using batches of 16 samples. Samples' durations were approximately 5 seconds or less.

The vocoder is trained using same setup on whole provided corpora without augmentation used in LPCNet reference implementation. Despite provided database containing recordings of different quality, we have not seen any improvements after removing the most clipped and distorted examples from the provided corpora.

3.6. Overview

Overview of the system is presented in Figure 1. The steps are as follows:

- input text is normalized using hand-written rules;
- normalized text is split into words, every word is converted into pinyin, pinyin is converted into phones;
- normalized text is split into tokens, resulting sequence of tokens is converted into a sequence of context-dependent token embeddings using a pre-trained BERT model;
- sequence of token embeddings is aligned with phones sequence;
- modified Tacotron 2 takes the phone sequence, a reference spectrogram and the sequence of context-dependent token embeddings as input, than outputs a mel-filterbank spectrogram;
- modified LPCNet generates a waveform using the mel-filterbank spectrogram.

For evaluation we chose a clean sounding sample to be used as a reference for the GST module.

4. Results

Our system is denoted by letter K in evaluation results published by organizers of the callenge.

Evaluation results consist of Mean Opinion Scores of all listeners, Pinyin Error Rates with and without tone errors (paid listeners) and Similarity Scores (all listeners). Medians for all participating systems are presented in Figure 4, Figure 2, Figure 3 and Figure 5 respectively. Means and standard deviations for all metrics are presented in Table 1.

5. Discussion

Results show that our system achieves lesser than average perfomance compared to other participants' systems.

We partially attribute low perfomance of our system to huge differences between train set and test set: while train set contains long utterances, test set consists mostly of shorter utterances. Mismatch between semantic content of train and test sets might also have played a role due to use of BERT embeddings. While producing utterances of any length is expected of any decent TTS system, we sadly did not pay enough attention to this aspect, using mostly long texts as development set. For submission, we ended up favoring model pronouncing long and short texts equally well over model excelling at pronouncing long texts but producing much less intelligible utterances for short texts.

Overall, we consider our results to be very positive. This is an example of how a small team can build a real-time text-to-speech system for a completely unfamiliar language in a couple of months using publicly available tools and data-driven approach. Whole pipeline has worked well; polishing rough edges is a matter of effort and time.

Table 1: *Evaluation results*

System	MOS	Similarity	PER	PTER
A	4.7 ± 0.6	4.5 ± 0.8	–	–
B	2.5 ± 1.1	2.0 ± 1.0	0.21 ± 0.2	0.22 ± 0.2
C	3.8 ± 0.9	3.7 ± 1.1	0.12 ± 0.2	0.14 ± 0.2
D	3.2 ± 1.0	2.9 ± 1.2	0.16 ± 0.2	0.18 ± 0.2
E	3.9 ± 1.0	3.8 ± 1.1	0.10 ± 0.1	0.12 ± 0.1
F	2.7 ± 1.1	2.5 ± 1.1	0.17 ± 0.2	0.18 ± 0.2
G	3.7 ± 1.0	3.4 ± 1.2	0.10 ± 0.2	0.11 ± 0.2
H	3.3 ± 1.1	3.1 ± 1.1	0.12 ± 0.1	0.14 ± 0.1
I	4.3 ± 0.8	3.3 ± 1.3	0.09 ± 0.1	0.10 ± 0.1
J	3.0 ± 1.1	2.7 ± 1.2	0.19 ± 0.2	0.21 ± 0.2
K	**2.8 ± 1.0**	**2.6 ± 1.1**	**0.41 ± 0.2**	**0.42 ± 0.2**
L	1.3 ± 0.6	1.1 ± 0.5	0.44 ± 0.2	0.46 ± 0.2
M	4.5 ± 0.7	4.1 ± 1.1	0.10 ± 0.2	0.11 ± 0.2
N	3.0 ± 1.1	2.2 ± 1.1	0.09 ± 0.1	0.10 ± 0.1
O	2.7 ± 1.2	2.7 ± 1.1	0.16 ± 0.2	0.17 ± 0.2
P	1.3 ± 0.7	1.2 ± 0.7	0.88 ± 0.0	0.89 ± 0.0
Q	2.8 ± 1.1	2.9 ± 1.1	0.29 ± 0.2	0.31 ± 0.2
R	2.7 ± 1.0	2.7 ± 1.1	0.26 ± 0.2	0.28 ± 0.2
S	4.0 ± 0.9	3.9 ± 1.0	0.09 ± 0.1	0.10 ± 0.2
T	3.5 ± 1.1	3.1 ± 1.3	0.15 ± 0.2	0.16 ± 0.2
U	1.4 ± 0.7	1.3 ± 0.7	0.28 ± 0.2	0.29 ± 0.2
V	3.2 ± 1.1	3.0 ± 1.1	0.19 ± 0.2	0.20 ± 0.2
W	4.3 ± 0.9	3.3 ± 1.4	0.11 ± 0.1	0.12 ± 0.1
X	3.0 ± 1.1	2.7 ± 1.1	0.26 ± 0.2	0.28 ± 0.2
Y	4.0 ± 0.9	3.8 ± 1.2	0.16 ± 0.2	0.17 ± 0.2
Z	4.0 ± 0.8	3.9 ± 1.1	0.12 ± 0.2	0.14 ± 0.2

6. Conclusion

We have described the Speech Technology Center (STC) text-to-speech system developed for the Blizzard Challenge 2019. Having no proficiency in Mandarin Chinese, we have developed a TTS system built on publicly available Chinese NLP tools and two deep neural networks using 6.5 hours of found speech data.

Subjective evaluation conducted by organizers of the challenge has shown that our system belongs to lower half of all systems participating in the challenge. Possible reasons for such perfomance were discussed.

Future work may include using larger databases for pre-training modules of the system for subsequent fine-tuning on found data, refining text preprocessing pipeline and involvement of native speakers in the development process.

7. References

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

[2] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," 2017. [Online]. Available: https://arxiv.org/abs/1703.10135

[3] 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.

[4] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. C. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017. [Online]. Available: https://openreview.net/forum?id=B1VWyySKx

[5] 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.

[6] 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.

[7] W. Ping, K. Peng, A. Gibiansky, S. Ö. Arik, A. Kannan, S. Narang, J. Raiman, and J. L. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," in *International Conference on Machine Learning*, 2017.

[8] Y. Taigman, L. Wolf, A. Polyak, and E. Nachmani, "Voiceloop: Voice fitting and synthesis via a phonological loop," in *ICLR*, 2017.

[9] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4784–4788, 2017.

[10] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[11] H. Kawahara, "Speech representation and transformation using adaptive interpolation of weighted spectrum: vocoder revisited," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 1997, pp. 1303–1306.

[12] M. Morise, F. Yokomori, and K. Ozawa, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions*, vol. 99-D, pp. 1877–1884, 2016.

[13] 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," in *9th ISCA Speech Synthesis Workshop*, pp. 125–125.

[14] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *International Conference on Machine Learning*, 2018, pp. 2415–2424.

[15] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "Fftnet: A real-time speaker-dependent neural vocoder," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2251–2255.

[16] J.-M. Valin and J. Skoglund, "Lpcnet: Improving neural speech synthesis through linear prediction," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5891–5895.

[17] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," *International Conference on Machine Learning*, 2019.

[18] 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.

[19] "jieba," https://github.com/fxsjy/jieba.

[20] "pypinyin," https://github.com/mozillazg/python-pinyin.

[21] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, 2017, pp. 498–502.

[22] "wit.ai," https://wit.ai/.

[23] Y. Yasuda, X. Wang, S. Takaki, and J. Yamagishi, "Investigation of enhanced tacotron text-to-speech synthesis systems with self-attention for pitch accent language," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6905–6909.

[24] Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Z. Chen, R. Skerry-Ryan, Y. Jia, A. Rosenberg, and B. Ramabhadran, "Learning to speak fluently in a foreign language: Multilingual speech synthesis and cross-language voice cloning," *arXiv preprint arXiv:1907.04448*, 2019.

[25] R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron," in *International Conference on Machine Learning*, 2018, pp. 4700–4709.

[26] W.-N. Hsu, Y. Zhang, R. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, P. Nguyen, and R. Pang, "Hierarchical generative modeling for controllable speech synthesis," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://arxiv.org/pdf/1810.07217

[27] J.-X. Zhang, Z.-H. Ling, and L.-R. Dai, "Forward attention in sequence-to-sequence acoustic modeling for speech synthesis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4789–4793.

[28] Y.-A. Chung, Y. Wang, W.-N. Hsu, Y. Zhang, and R. Skerry-Ryan, "Semi-supervised training for improving data efficiency in end-to-end speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6940–6944.

[29] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in neural information processing systems*, 2018, pp. 4480–4490.

[30] N. Li, S. Liu, Y. Liu, S. Zhao, M. Liu, and M. Zhou, "Neural speech synthesis with transformer network." AAAI, 2019.

[31] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2. IEEE, 2003, pp. 1398–1402.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[33] "pytorch-pretrained-bert," https://github.com/huggingface/pytorch-transformers.

[34] Z. Kons, S. Shechtman, A. Sorin, C. Rabinovitz, and R. Hoory, "High quality, lightweight and adaptable tts using lpcnet," *arXiv preprint arXiv:1905.00590*, 2019.

[35] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.