SPEECH WAVEFORM RECONSTRUCTION USING CONVOLUTIONAL NEURAL NETWORKS WITH NOISE AND PERIODIC INPUTS

Oliver Watts, Cassia Valentini-Botinhao and Simon King

The Centre for Speech Technology Research, Edinburgh University, UK

ABSTRACT

This paper presents a method for upsampling and transforming a compact representation of acoustics into a corresponding speech waveform. Similar to a conventional vocoder, the proposed system takes a pulse train derived from fundamental frequency and a noise sequence as inputs and shapes them to be consistent with the acoustic features. However, the filters that are used to shape the waveform in the proposed system are learned from data, and take the form of layers in a convolutional neural network. Because the network performs the transformation simultaneously for all waveform samples in a sentence, its synthesis speed is comparable with that of conventional vocoders on CPU, and many times faster on GPU. It is trained directly in a fast and straightforward manner, using a combined time- and frequency-domain objective function. We use publicly available data and provide code to allow our results to be reproduced.

Index Terms— neural vocoder, speech reconstruction, convolutional neural network

1. INTRODUCTION

Until recently, statistical parametric speech synthesis systems worked by predicting a low-dimensional, slowly varying representation of acoustics from text, and then passing that representation to the synthesis part of a vocoder for conversion into a speech waveform [1]. That is, the transformation from vocoder features to waveform is a fixed, carefully-engineered, knowledge-based procedure. In contrast, recent work has successfully replaced this fixed transformation with one learned from data [2, 3]. The impetus for much or all of this work has come from [4], where sequences of waveform samples are generated directly given discrete linguistic features, bypassing conventional acoustic features altogether. We here focus on the acoustic feature to waveform mapping task (sometimes termed *neural vocoding*), because it addresses a more tractable problem and allows the training of TTS systems to be conveniently modularised [3, §2.1], [5, 6]; even in the systems which attempt to map directly from discrete linguistic inputs, it is found necessary to condition also on some continuous acoustic variables [4, §3.2].

The learned reconstruction of acceptable waveforms from acoustic representations is still a challenging task, to the extent that inputs are *underspecified*. This underspecification typically takes the form of missing phase information, and of simplification and compression of the magnitude spectrum [3, 7]. Here, the challenge is for a system to learn to restore acceptable phase and magnitude spectral detail on the basis of some training observations.

A learned waveform reconstructor has several potential advantages over a conventional vocoder synthesis module. Firstly, a neural vocoder might reconstruct more acceptable waveforms from *natural parameters* on a simple copy synthesis task, as it restores the information missing from the input in a data-driven fashion and is thus less limited by the knowledge and assumptions inherent in a vocoder. Secondly, [3, §3.3.1] has shown that neural vocoders can compensate for imperfections in their predicted acoustic inputs if trained with inputs which have been corrupted in a consistent way. Thirdly, a learned function can be trained to map from any reasonable representation of speech to a waveform, even when no knowledge-based procedure for making that transformation exists. Other representations could be desirable due to their interpretability, perceptual relevance, or suitability for acoustic modelling. This is a possibility which to our knowledge has not yet been exploited.

There are therefore good motivations for wanting to train and deploy learned waveform reconstructors. An issue with [4] of major practical importance, however, is the time it takes to generate speech. The autoregression used by the model means that the most naive implementation would involve a forward pass through a deep neural network many thousands of times for each second of speech to be generated. Less naive approaches to generation where computation is reused by storing hidden activations at previous timesteps to a buffer are still computationally impractical [8]. Other approaches look at speeding up the synthesis-time operation in other ways, while remaining autoregressive [9, 10].

A different approach to speeding up generation with these models can be found in [11]. There, the synthesis model is an inverse-autoregressive flow which generates all waveform samples in parallel without incurring the expense of having to make predictions autoregressively. Inputs consist of linguistic features and a noise sequence which is gradually shaped into an appropriate distribution over waveforms during a forward pass through the network. Training is complicated by the fact that the synthesis model is not trained directly. Instead, a wavenet [4] model is first trained and then used as a teacher for the feedforward synthesis model.

This paper presents and evaluates a synthesis model which also works in a purely feedforward fashion, without needing to operate autoregressively at generation time. Our aim is to determine the simplest architectures which can reconstruct reasonable quality waveforms in this way. Our model therefore consists of relatively standard residual convolutional network layers, and is trained directly, without the need for a teacher model. It is trained with a mean squared error criterion to output a point prediction of a speech waveform, rather than a distribution over waveform values.

2. PROPOSED SYSTEM

2.1. Network inputs

As shown in Figure 1, input to the proposed system consists of mel-spectral features upsampled to the desired output waveform sample rate, concatenated with a pulse train encoding the desired fundamen-

Samples & code: `http://homepages.inf.ed.ac.uk/owatts/papers/ICASSP2019`

tal frequency (F0) and a sequence of Gaussian noise. The use of a pulse train at the input is a key feature of the proposed system which sets it apart from other neural vocoders. It disambiguates the positioning of glottal closure and so makes the network's task more tractable. The pulse train we use in the experiments described here can be thought of as normalised position forwards through current glottal cycle (analogously with positional linguistic features in TTS) or equivalently as an initial saw-toothed attempt at a waveform, which is then shaped by a number of learned convolutions into one which better matches the given acoustic features. The glottal closure instances used to generate the pulse train are provided by a pitch-tracker at training time, and synthesised from an F0 track at synthesis time.[1] The Gaussian noise sequence is provided for the network to produce stochastic effects in its output.

2.2. Network structure

The inputs described are then fed through several r residual blocks [12, 13]. Each block computes a transformation of the input data which – when added to that input – better satisfies the optimisation criterion used in training. For each block the following values are configured:

l: number of conv. layers to compute residual representation
c: number of channels input/output by the residual block
w: width and d: dilation of convolutions used in the block
a: activation applied after each convolutional layer in the block.

In the implementation used for the experiments reported here, the last 4 values are constant within each residual block. In cases where the number of channels c for a block is different to that of the input to the block, a time-distributed linear transform (width-1 linear convolution) is applied at the start of the block to impose the required dimensionality. As we used a uniform c value for all blocks in systems built for the experiments reported here, this only applies between the input and the first block. At the end of each residual block, batch normalisation [14] is applied.

Finally, a monaural audio waveform is produced by reducing the final block's output to a 1-dimensional signal with a final trained time-distributed linear transform (width-1 linear convolution). This is the speech waveform predicted by the network. In principle a non-linearity can be applied to the output to enforce the proper range of waveform amplitudes, but in practice this was not found to be necessary.

2.3. Network training

A combination of two losses is used for training the system. The first loss requires mean squared error (MSE) between the output time domain (TD) signal and the reference waveform to be minimised. This will never give good results in general, as minimising squared errors in this way for portions of speech dominated by stochastic effects (e.g. fricatives, such as [s]) will only predict the mean of the training samples, resulting in a silent signal. It is informative to note, however, that reasonably natural and intelligible voiced speech can be generated using only this loss on its own (that is: intelligible, taking into account the fact that many consonants are missing from the prediction).

[1] Note that we therefore supplement the mel-spectrogram features with explicit F0 information. The spectral features implicitly contain information about F0 in the form of harmonics, and so supplementing them with this pulse train adds redundancy: ongoing work seeks to find less redundant representations which are suitable for use as network inputs.

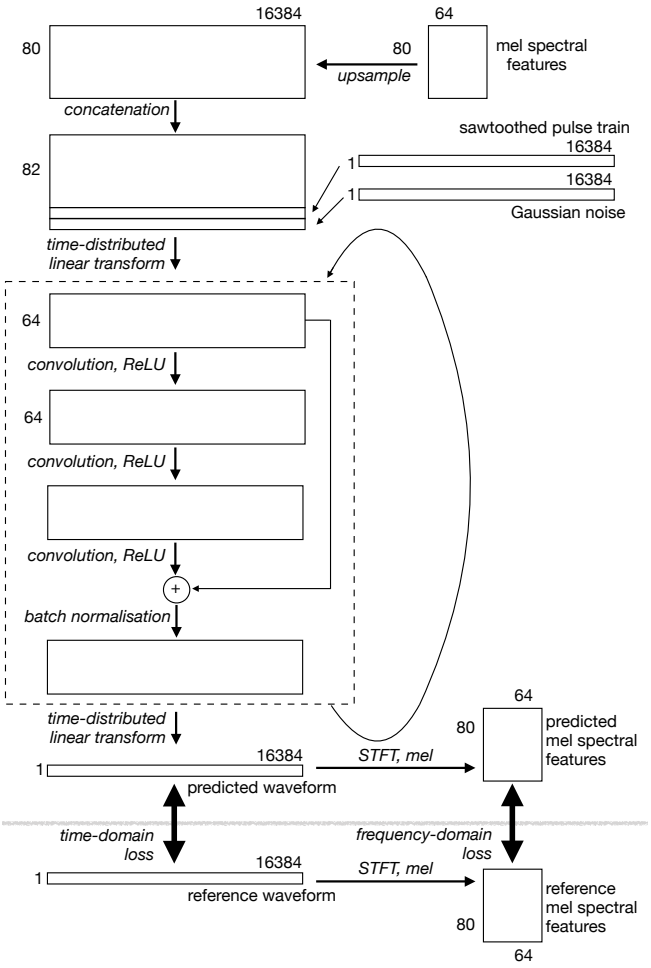

Fig. 1. Proposed system.

In order to force the network to make perceptually acceptable predictions for the stochastic parts of speech, the TD loss is therefore combined with one computed in the frequency domain (FD). The network is extended at training time with extra convolutional layers with fixed weights implementing short-time Fourier transform (STFT), mel warping and compression, so that a forward pass through the network produces two outputs: the TD prediction as already described and the corresponding mel spectrogram of that signal. The TD loss is then combined with a mel-FD loss consisting of the MSE between the spectrogram output of the network and an identically computed spectrogram of the reference signal. This error signal from the FD supervision is backpropagated through the STFT module to tune the weights of the rest of the network in the normal way. Note that the extra layers of the network are needed only in training: these are removed from the network used at runtime which outputs only the TD prediction.

The two parts of the combined loss are weighted by a time loss weight (λ_t) and frequency loss weight (λ_f), where $0 < \lambda_t < 1$ and $\lambda_f = 1 - \lambda_t$. These two weights can be tuned as hyperparameters.

2.4. System refinements

One refinement which generally improved the quality of output speech and was used for the experiments described here was to non-linearly transform the TD prediction and reference before computing the TD part of the loss. The non-linearity used in mulaw quantisation was used for this purpose, although it is important to

Table 1. Experimental conditions.

System	Description
NAT	Natural speech
WO	Speech vocoded with WORLD [16]
MA	Speech vocoded with MagPhase [17]
WA	Speech resynthesised by `wavenet_vocoder`
P0	Proposed system
P1	Proposed system & postprocessing for noise reduction
GL	Speech resynthesised from mel-warped & compressed magnitude spectrogram using Griffin-Lim [18]

note that we did not actually quantise the values transformed in this way. We were seeking only to transform the continuous waveform values in a perceptually-relevant way before computing the loss, analogous to computing the FD part of the loss over a mel-warped spectrogram rather than the underlying linear one. As with the FD loss, this involves extensions to the network which are discarded from the run time system.

The predictions of our models are intelligible and of fairly high quality, but often they are marred by an artifact which gives the impression of additive background hiss-noise, constant across time. We therefore found it advantageous to use the noise suppression framework described in [15] in a postprocessing step. We note that other neural vocoders suffer the same problem, which has been addressed in similar ways [10].

3. EXPERIMENTS

3.1. Database, proposed system and baselines

The conditions summarised in Table 1 were compared side by side in a perceptual experiment. Its purpose is to benchmark the proposed system against other popular methods for waveform reconstruction in a simple copy synthesis task. All systems compared are freely available; they consist of two conventional vocoders (WORLD [16] and MagPhase [17]), a multi-speaker wavenet vocoder, and an implementation of the Griffin-Lim algorithm for phase recovery [18].

Although the end goal of this work is to provide a trainable stand-in for the waveform generation module of an statistical parametric speech synthesis system, we leave the integration of our waveform generator into a complete TTS system for future work.

3.1.1. Data

The speech data of seven speakers from the CMU Arctic 16 kHz speech database were used for the experiment [19]. Use of the same data and same train–test split as the existing wavenet vocoder system allowed proper comparison with that system. Following the data partitioning used for the wavenet vocoder system, the same 350 sentences selected at random from across the pooled sentences of the seven Arctic speakers (awb, bdl, clb, jmk, ksp, rms, slt) were held out for testing. The split used for the wavenet vocoder system makes no effort to set aside the same number of test sentences for each speaker, or to ensure that the same texts are used for the test set data of each speaker. For the experiment reported here, we decided to test systems on the held-out data of one male and one female speaker, and the first speaker alphabetically of each sex was chosen: awb and clb. Thus the training data consisted of the Arctic data with 350 sentences held out, giving 6.47 hours of training data pooled across all speakers, and test sets of 46 and 61 sentences were used for awb and clb respectively.

3.1.2. Proposed system

As mentioned in Section 2.4, we have found it beneficial to apply some denoising to the output of our system [20]. To assess the impact of this postprocessing, we evaluate two versions of the proposed system: P0 where the network output is used directly, and P1 where the denoising is applied to P0. Both systems are therefore based on the same neural network, which has the architecture described in Section 2. Inputs to the network consist of 80-dimensional mel-warped spectra extracted from 1024-point windows with a 256-point frame shift, as well as an excitation sawtoothed pulse train and Gaussian noise at the desired sample rate (16 kHz). The pulse train's maxima correspond to glottal closure instances detected by REAPER [21] in training, and are arbitrarily placed although consistent with a REAPER-extracted F0 track at test time. The evenly-spaced pitchmarks which the tool places in unvoiced speech are ignored when constructing the pulse train, which is set to zero in these regions.

The number of residual blocks r used for the evaluated system was 8. Number of convolutional layers l, number of channels c, width of convolution w, and dilation rate d were 3, 64, 9, and 1, respectively for all blocks, except for the first block, where it was found useful to have a dilated convolution (with rate 20). Rectified linear activations were used after each convolutional layer, and batch normalisation was applied after each residual block.

All audio presented to the system is scaled to the range $[-1, 1]$ and the system's outputs consistently lie in the same range without any further normalisation. Training data is split into fragments of approximately one second each and shuffled into batches of eight for training. Training is done with Adam using suggested default settings [22], for two epochs, optimising a combination of two mean squared error losses: the warped TD loss (λ_t) and the mel FD loss (λ_f) described in Section 2. These losses were weighted 0.2 and 0.8, respectively. The trained system has fewer than one million learned parameters in total.

Most of these settings were chosen as reasonable during initial development on the basis of a different database containing approximately 2 hours of speech from a single male speaker. Only the loss weights and the stopping point (2 epochs) were chosen by informal listening to some held out data from the set used for the current experiment.

The proposed system was implemented using Keras [23] and Tensorflow [24]; the Kapre library [25] was used for mel spectrogram extraction and optimisation.

3.1.3. Benchmark systems

The WORLD vocoder [16] baseline was created using the version of the vocoder distributed with [26]. It was used to extract 60 mel cepstral coefficients and 5 band aperiodicities. We extracted F0 using REAPER [21]. Speech was then reconstructed from these three streams of acoustic features using the WORLD synthesis routines.

The MagPhase vocoder baseline was created using the publicly available implementation described in [17]. It was used to extract 60 magnitude, 45 imaginary and 45 real features. As in the case of WORLD, REAPER [21] was used to extract F0. MagPhase was used to reconstruct speech from these four acoustic feature streams.

The Griffin-Lim baseline was created using the freely available implementation at `https://github.com/bkvogel/griffin_lim`. This code was used to extract mel-warped and compressed spectrograms from the 107 test sentences, and then estimate consistent phase through 300 iterations of the Griffin-Lim procedure, starting with randomly-initialised phase. Feature extraction settings were comparable with those used for the proposed system.

The wavenet vocoder baseline was created using the open source implementation at `https://github.com/r9y9/wavenet_vocoder`, and also the pretrained Arctic multispeaker model linked to from there. Inputs comparable to those of systems P0 and GL were fed to the pretrained wavenet model, which consists of 4

Table 2. Real time factors for synthesis.

System	WO	MA	WA	P0 (*CPU*)	P1	GL
Factor	0.32	0.39	352.05	0.03 (*0.36*)	0.12	2.44

stacks of 6 residual blocks each, each residual block containing approximately 0.5 million weights. This is a mixture density network, outputting at each timestep the parameters for a mixture of 10 logistic distributions.

3.2. Training and synthesis efficiency

The figures for real time factor (RTF) given in Table 2 show synthesis time as a fraction of the duration of the speech being generated. The figures exclude time for loading models and feature extraction, and were computed over the 5.75 minute combined test set for speakers awb and clb. WA, P0 and P1 used the same hardware (GeForce GTX TITAN GPU); all other systems ran on CPUs (and additionally postprocessing for P1 took place on CPU – combined GPU and CPU time was used to compute RTF). Additionally, synthesis run time for P0 on CPU was measured and RTF shown in parentheses. It can be seen that all systems run faster than real time except the wavenet vocoder WA and Griffin-Lim GL. WA took over 12 hours to produce this small test set. GL could probably be run with fewer iterations without much loss in quality, but this has not been tried systematically. The proposed system is considerably faster than conventional CPU-based vocoders when making use of GPU, and comparable with them when running on CPU.

As well as the benefits shown by the synthesis-time figures in Table 2, the proposed system is also fast to train, taking 29 minutes per epoch on a GeForce GTX TITAN GPU. The evaluated system was therefore trained in under an hour. Although we did not train system WA, its training will have taken much longer. It was trained for $740,000$ steps, and we are informed that training a model of the same size and architecture and same database for $10,000$ steps takes 2 hours on an Nvidia P100 GPU, suggesting that it would take over 6 days to retrain system WA.

3.3. Listening experiment design

The experiment was run as a MUSHRA-style test [27] with 21 screens. On each screen, participants could listen to the audio produced for the same sentence by all systems; they were able to listen repeatedly at will. They were asked to rate the quality of each of the samples on a scale from 0 (bad) to 100 (excellent). The first screen was used only for training purposes, and responses collected from it are discarded. The first 10 screens contained sentences from the male speaker (awb), and the last 10 from the female speaker (clb); each screen contains a sentence with a unique text. 40 different sentences were spread across each four participants. As system NAT consists of natural speech, participants have a quality reference and it is possible to check whether participants score it as 100 as instructed, and consequently whether they perform their evaluation with sufficient care and attention. Twenty native English speakers took part in the experiment. Two participants were excluded as they rated NAT less than 100% in more than 20% of screens for both voices. We excluded around 12% of the remaining screens when listeners did not give NAT the highest score.

3.4. Results

Fig. 2 shows a boxplot summarising participants' responses to the stimuli. Solid and dashed lines mark the median and mean rating for each condition. A Mann-Whitney U test was used to determine which differences were significant at $\alpha = 0.05$, Holm–Bonferroni

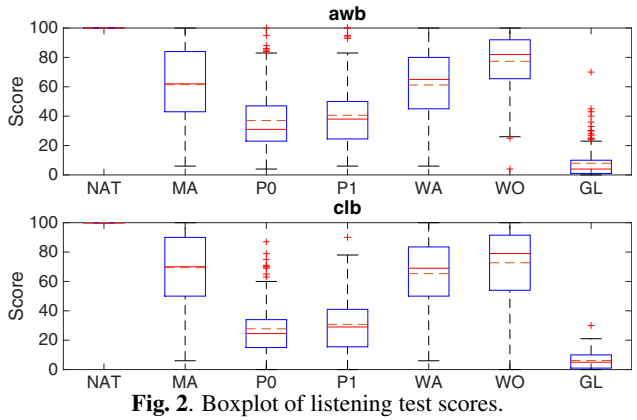

Fig. 2. Boxplot of listening test scores.

corrected for number of pairwise comparisons. There is no significant difference under this test between systems P0 and P1 for both speakers, between MA and WA for speaker awb, or between MA, WA and WO for speaker clb. All other systems were perceived to be significantly different from each other and from these groups.

All participants reliably rated natural speech at 100, and the Griffin-Lim system constitutes a clear bottom-line system, significantly worse than other systems for both speakers. Nowhere does the wavenet vocoder outperform a conventional vocoder; this contrasts with the findings of [5], although both the vocoder and wavenet implementations in that work are different from ours. Results differ by speaker: for the male speaker awb the WORLD vocoder is preferred to the wavenet and MagPhase vocoder. This also contrasts with previous findings, where MagPhase is generally preferred over WORLD [28], although a lower sample rate is used for the current experiment. The experimental systems close a significant part of the gap between the baseline GL and the next best performing system: approximately half of that gap for speaker awb, and a little less for clb. The postprocessing for noise removal improves the mean rating slightly for both speakers, although in neither case significantly.

4. CONCLUSIONS

We have presented a system for waveform reconstruction which can operate faster than all evaluated competitors, and can be trained many times faster than the other trainable system evaluated. Scores for quality do not yet approach those for conventional vocoders, but ongoing work in the form of more thorough hyperparameter tuning is expected to lessen the gap further. Furthermore, the current work only looks at the model in the context of speech analysis-synthesis, where the guide parameters for the waveform to be synthesised are the result of speech analysis. This forms a foundation for ongoing work in which we aim to map from predicted acoustic features to a waveform. Matched training in this case is expected to give both trainable systems evaluated a boost in quality relative to the conventional vocoders, as trained systems can learn to compensate for imperfections in their predicted input.

Aside from basic quality improvements, ongoing work is looking at reconstructing waveforms sampled at higher sampling rates, and considering different input representations.

Acknowledgements: This research was supported by EPSRC Standard Research Grant EP/P011586/1. We thank Zack Hodari for some useful comments in the early stages of this work, and for wavenet training time information.

5. REFERENCES

[1] Heiga Zen, Keiichi Tokuda, and Alan W. Black, "Review: Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, Nov. 2009.

[2] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," in *Proc. ICLR*, 2017.

[3] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *Proc. ICASSP*, 2018.

[4] Aron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016.

[5] Tomoki Hayashi, Akira Tamamori, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda, "An investigation of multi-speaker training for wavenet vocoder," in *Proc. ASRU*, 2017.

[6] Nagaraj Adiga, Vassilis Tsiaras, and Yannis Stylianou, "On the use of WaveNet as a statistical vocoder," in *Proc. ICASSP*, 2018.

[7] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda, "Speaker-dependent WaveNet vocoder," in *Proc. Interspeech*, 2017.

[8] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A Hasegawa-Johnson, and Thomas S Huang, "Fast wavenet generation algorithm," in *Arxiv*, 2016.

[9] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu, "Efficient neural audio synthesis," in *Proc. ICML*, 2018.

[10] Zeyu Jin, Adam Finkelstein, Gautham J. Mysore, and Jingwan Lu, "FFTNet: a real-time speaker-dependent neural vocoder," in *Proc. ICASSP*, 2018.

[11] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proc. ICML*, Jul 2018.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[13] Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," in *ICLR*, 2017.

[14] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[15] Israel Cohen and Baruch Berdugo, "Speech enhancement for non-stationary noise environments," *Signal Processing*, vol. 81, no. 11, 2001.

[16] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa, "WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 7, 2016.

[17] Felipe Espic, Cassia Valentini-Botinhao, and Simon King, "Direct modelling of magnitude and phase spectra for statistical parametric speech synthesis," in *Proc. Interspeech*, 2017.

[18] D. Griffin and Jae Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 32, no. 2, 1984.

[19] John Kominek and Alan W. Black, "The CMU arctic speech databases," in *Proc. SSW*, 2004.

[20] I. Cohen, "Noise spectrum estimation in adverse environments: improved minima controlled recursive averaging," *IEEE Trans. on Speech and Audio Processing*, vol. 11, no. 5, Sept 2003.

[21] "REAPER: Robust Epoch And Pitch EstimatoR," `https://github.com/google/REAPER`, 2017.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *Proc. ICLR*, 2015.

[23] François Chollet et al., "Keras," `https://keras.io`, 2018.

[24] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, 2016.

[25] Keunwoo Choi et al., "Kapre: Keras audio preprocessors," `https://github.com/keunwoochoi/kapre`, 2018.

[26] Zhizheng Wu, Oliver Watts, and Simon King, "Merlin: An open source neural network speech synthesis system," in *Proc. SSW*, 2016.

[27] International Telecommunication Union Radiocommunication Assembly, Geneva, Switzerland, *Method for the subjective assessment of intermediate quality level of coding systems*, March 2003.

[28] Oliver Watts, Cassia Valentini-Botinhao, Felipe Espic, and Simon King, "Exemplar-based speech waveform generation," in *Proc. Interspeech*, 2018.