A BEGINNER'S GUIDE INTO NEURAL SPEECH SYNTHESIS

Sergey Chervontsev
Moscow Institute of Physics and Technology, DIHT
chervontsev.ss@phystech.edu

ABSTRACT

We show a general pipeline for Text-To-Speech translation and explain how recent neural architectures solve disjoint tasks within this pipeline — the extraction of acoustic features from a raw input and the generation of an output signal conditioning on these features. Our explanation is natural and self-sufficient, as we cover only general approaches while omitting numerous possible extensions. In the task of acoustic extraction, we consider ARSG and VoiceLoop as a possible solutions. In the task of voice generation, we compare WaveNet and SampleRNN models.

1 INTRODUCTION

The task of speech synthesis, or Text-To-Speech (TTS), is to map an input text $X = (x_1, \dots x_T)$ into the corresponding acoustic signal $Y = (y_1, \dots, y_N)$[1] containing a spoken natural language utterance. The use of deep learning models has greatly reduced the amount of hand-crafted features in TTS systems and shrank the traditional statistical pipelines. However, there still doesn't exist fully end-to-end solution which exploit only raw text and signal and doesn't rely on external separately trainable components or spectral decompositions.

The current architectures most commonly fell into one of the two categories: frontend readers and backend vocoders. The first transform text into a set of acoustic features $a = (a_1, \dots, a_L)$. These might predict quantities such as fundamental frequency on log scale $(log\ F_0)$, Mel-Cepstral coefficients and so on. The second generate Y conditioning on extracted features a. Note that each step may be replaced with a non- or separate-trainable system. E.g., a backend system might be represented by a pre-trained WORLD vocoder (Morise et al., 2016), and the reader may be trained to produce relevant vocoder features. Alternatively, an acoustic features might be extracted by performing word segmentation, text normalization, part-of-speech (POS) tagging, grapheme-to-phoneme convertion, etc, and vocoder may be trained to efficiently utilize them.

Potentially, one might train both systems simultaneously. However, we are unaware of such solutions — in best case scenario, the mutual training is done to fine-tune the model. A possible reason for this is the complexity of TTS task, as on average, one needs to generate a 6,000 samples per word[2] to get a signal, leading to a large discrepancy between the lengths of input and output sequences.

Note that in this paper we doesn't consider models that predict spectral characteristics, such as Wang et al. (2017). Aside from their complexity, as observed by Arik et al. (2017b), small errors in the spectrogram generation result in unnatural (metallic) noise in the reconstructed speech.

Keeping those facts in mind, we consider multiple reader and vocoder models in the following sections. In sections 2 and 3, we describe possible approaches for acoustic extraction — Attention-based Recurrent Sequence Generator (ARSG) and VoiceLoop readers respectively. In sections 4 and 5, we shift to WaveNet and SampleRNN vocoders. In section 6, we compare the presented architectures.

[1] Usually, the splitting is done to get 24,000 frames per second

[2] According to the average speaking rate for a set of TED talks (https://virtualspeech.com/blog/average-speaking-rate-words-per-minute)

2 ARSG READER

2.1 GRAVES ATTENTION

The use of attention mechanism has greatly improved results in neural machine translation, especially when dealing with long sequences (Bahdanau et al., 2014). If we solve the task of acoustic extraction as a task of translation, it is important to have an alignment that doesn't focus on different parts of an input but rather slides through a sequence as does human when reading aloud.

For this, Graves (2013) proposed to use a mixture of gaussians that are parametrized by a trainable fully-connected neural network N_{Att}[3]:

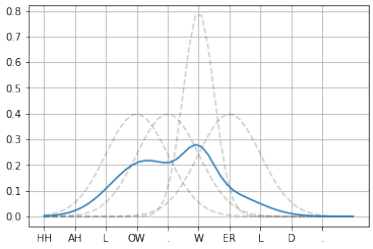

Figure 1: Example of GrAtt alignment for the phonemic sequence of "Hello World" with $K = 4$, $w = (0.4, 0.2, 0.2, 0.2)$, $\mu = (3, 4, 5, 6)$ and $\sigma = (1, 1, 0.5, 1)$.

$$\alpha_t[i] = \sum_{k=1}^{K} w_k \mathcal{N}(i|\ \mu_{t,k}, \sigma_k), \tag{1}$$

$$w_k = \text{Softmax}(\gamma_k), \tag{2}$$

$$\mu_{t,k} = \mu_{t-1,k} + exp(\kappa_k), \tag{3}$$

$$\gamma_k, \kappa_k, \sigma_k = N_{\text{Att}}(f_{t-1}); k = 1, \dots K, . \tag{4}$$

In equations 1 — 4, $\alpha_t[i]$ is the alignment of the i-th element in a sequence at t-th timestamp, K is the number of mixtures, N_{Att} is a trainable FFN network and f_{t-1} is a set of model-specific features that are fed into attention mechanism. We will denote this module as GrAtt(f_{t-1}) in the following sections.

2.2 MODEL DESCRIPTION

An attention-based recurrent sequence generator recalls a recurrent neural network that generates acoustic features $a = (a_1, \dots, a_L)$ conditioned on an input phonemic sequence[4] $X = (x_1, \dots, x_T)$ (Chorowski et al., 2015). At first, X is embeded by an encoder to get the hidden representations $h = (h_1, \dots, h_T)$. The encoder is a standard bidirectional recurrent network, e.g., with GRU nonlinearity cells. Then, at t-th timestamp the ARSG generates new a_t in the following steps:

$$\alpha_t = \text{GrAtt}(s_{t-1}), \tag{5}$$

$$c_t = \sum_{i=1}^{L} \alpha_t[i] h_i, \tag{6}$$

$$a_t = N_{\text{Gen}}([s_{t-1}; c_t]), \tag{7}$$

$$s_t = \text{RNN}([c_t; a_t], s_{t-1}). \tag{8}$$

Differently to common seq2seq architecture, generation is done via another trainable FFN N_{Gen}, and the generated value is fed into RNN cell along with current context c_t to update its hidden state. Usually, both N_{Gen} and N_{Att} have two layers with ReLu activations.

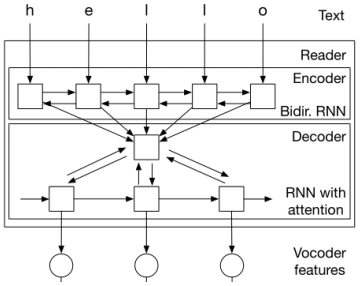

Figure 2: ARSG model. X might be a raw text instead of a phoneme sequence, though it works a little bit worse.

The model is trained to produce WORLD acoustic features with MSE loss. Those can be extracted from the raw audio using Merlin tookit (Wu et al., 2016).

[3]We omit t subscripts where it's possible for brevity

[4]Those may be obtained from the text with the use of pronouncing dictionaries, e.g. CMUDict (Weide, 1998)

3 VOICELOOP READER

The VoiceLoop Reader (Taigman et al., 2017) is a method to extract vocoder features from multiple voices that are sampled in-the-wild. It is inspired by the phonological loop memory model (Baddeley, 1986). The key idea is to replace RNN with an intermediate differentiable buffer, that updates during reading in a FIFO manner.

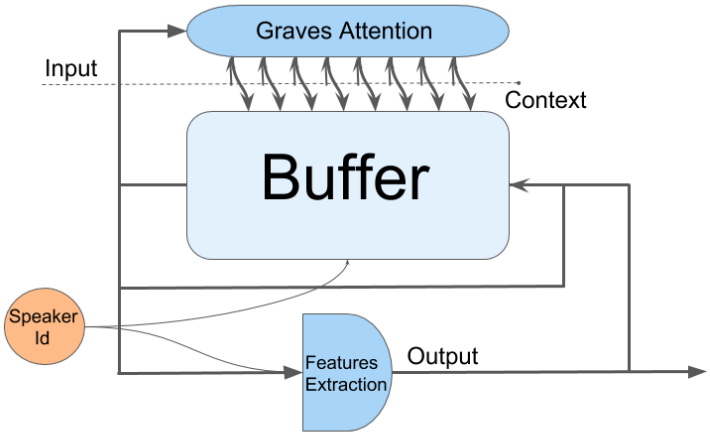

Figure 3: An overview of the VoiceLoop architecture. At t-th timestamp, the current state of the buffer along with the input is used to get a context vector c_t. The generated vocoder features a_{t-1}, as well as context, speaker embedding and current buffer states S_{t-1} are used to get a new buffer state. The updated buffer S_t defines new features a_t.

Technically, the model is defined with the following equations:

$$\alpha_t = \text{GrAtt}(S_{t-1}), \tag{9}$$

$$c_t = \sum_{i=1}^{T} \alpha_t[i] x_i, \tag{10}$$

$$u_t = N_{\text{Upd}}([S_{t-1}; c_t + \tanh(F_u z); a_{t-1}]), \tag{11}$$

$$S_t[0] = u_t; S_t[i+1] = S_{t-1}[i], \tag{12}$$

$$a_t = N_{\text{Out}}(S_t) + F_a z. \tag{13}$$

$$\tag{14}$$

Here, the Graves Attention extract a context vector c_t from the input sequence. Next, a trainable FFN N_{Upd} produces a new encoded state u_t which is pushed into existing queue S_{t-1}, and the latest element in S_{t-1} is discarded. Finally, we use an additional fully-connected network N_{Out} that predicts a new output a_t. N_{Upd} and N_{Out} are conditioned on the speaker embedding z, so all the other parameters are shared between multiple speakers.

Similarly to ARSG, all FFN networks have one hidden layer with ReLu-s, and the model is trained to return a WORLD vocoder features by minimizing MSE loss. This holds a danger of underfitting, because even a human can't replicate its own speech to get exactly the same output. To account for this, the teacher-forcing is used — during training, instead of passing a_{t-1} in the update network we pass

$$\frac{a_{t-1} + a^*_{t-1}}{2} + \eta, \tag{15}$$

where a^*_{t-1} is the correct answer and η is a noise vector. Note that we cannot pass a^*_{t-1} alone, as the model will train to generate only one step ahead.

4 WaveNet Vocoder

WaveNet (van den Oord et al., 2016a) is a non-recurrent generative model based on a dilated causal convolutions. It receives acoustic features $a_1, \ldots, a_L$, produced manually or by using frontend reader, upsamples them to desired length N, yielding $u_1, \ldots, u_N$ and generates wave in an autoregressive way:

$$p(y_t|\, y_1, \ldots, y_{t-1}) = \text{WaveNet}(y_1, \ldots, y_{t-1}; u);\ t = 1, \ldots, N. \tag{16}$$

In the following subsections, we describe the dilated causal convolutions, the residual block and the overall architecture.

4.1 Dilated Causal Convolution

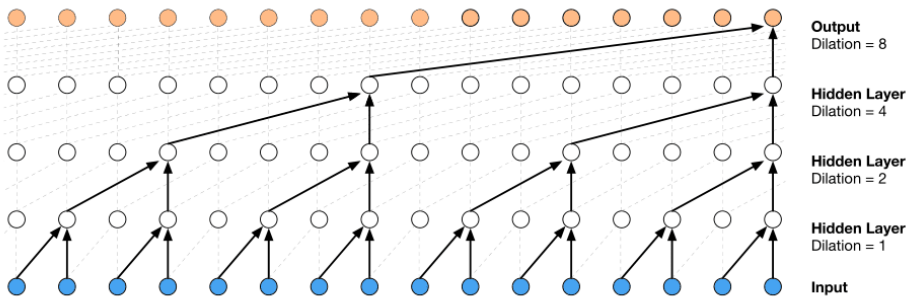

Figure 4: A stack of dilated causal convolutions with kernel size 2. The resulting receptive field after four layers is $2^4 = 16$.

The causal convolution is a convolution that masks the future timestamps in the input data. That means that model doesn't "look ahead" and thus is able to process data sequentially. In the case of 1D indexing this can be efficiently implemented by adding a left zero padding of size $(k-1)$ for a filter size k.

The dilated causal convolution applies its filter over a larger area by skipping a fixed number of elements, as can been seen in figure 4. This trick increases the receptive field exponentially depending on depth, and helps to use a reasonable amount of filters to account for long-term dependencies.

More formally, a dilated causal convolution of kernel k and input f with dilation d may be defined as:

$$(k *_d f)_t = \sum_{s=1}^{t} \sum_{\tau=0}^{K-1} \mathbb{1}(s + d\tau = t) k_{\tau+1} f_s;\ t = 1, \ldots, N, \tag{17}$$

where K is the kernel size and $\mathbb{1}$ is an indicator function.

In addition to speech synthesis, dilated convolutions were successfully applied to image generation (van den Oord et al., 2016b) and image segmentation (Yu & Koltun, 2016). They even outperformed traditional recurrent systems in language modeling (Bai et al., 2018).

4.2 UPSAMPLING & RESIDUAL BLOCK

In the original paper, the upsampling procedure is made with a transposed convolution network. However, Arik et al. (2017a) found it more beneficial to use two bidirectional QRNN layers followed by a simple repetition[5].

To let the model drop irrelevant features as well as transform relevant, WaveNet uses a gating mechanism:

$$z = \tanh\left(k^{(f),y} *_d y + k^{(f),u} *_d u\right) \odot \sigma(k^{(g),y} *_d y + k^{(g),u} *_d u), \tag{18}$$

where y is an output from the previous layer (or the initial input), u is a sequence of upsampled features, and $(k^{(f),y}, k^{(f),u}, k^{(g),y}, k^{(g),u})$ are four different kernels of similar shape.

Even with the use of dilated convolutions, a large depth is required to catch dependencies that exist in audio data. Thus, to improve training, WaveNet utilize both residual and skip connections between convolution layers. The resulting form of the residual block may be seen on figure 5.

4.3 FULL ARCHITECTURE

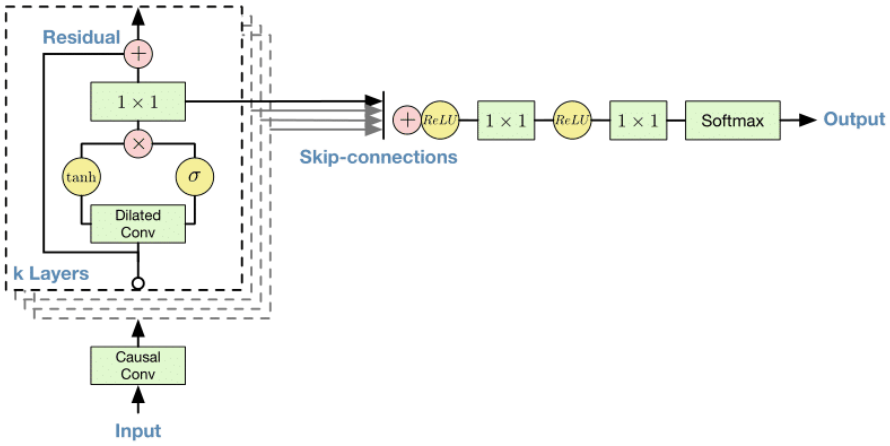

Figure 5: WaveNet architecture. Two different weights are used with $\tanh$ and σ nonlinearities, as well as two different convolutions work with residual and skip connections.

In total, WaveNet consists of a stack of residual blocks along with skip connections, and a simple two-layer FFN that transforms these into final predictions. In the original paper, the dilation is doubled from 1 for every layer up to a limit of 512 and then repeated, which leads to a following sequence of dilations:

$$1, 2, \ldots, 512, 1, 2, \ldots, 512, 1, 2, \ldots, 512.$$

And gives thirty residual blocks in total.

Additionally, instead of predicting a raw wave, we apply a μ-law transformation to it:

$$f_\mu(y_t^*) = \text{sign}(y_t^*)\frac{ln(1+\mu|y_t^*|)}{ln(1+\mu)},$$

where $-1 < y_t^* < 1$ is the true answer, $\mu = 255$, and then quantize $f_\mu(y_t^*)$ to 256 possible values. This gives us a one-hot representation for the output data and hence we can use a standard softmax on top of the model and train it to minimize the negative log-likelihood. van den Oord et al. (2016a) found that this scheme leads to a much better reconstruction than the use of causal MSE loss.

[5]Repetition is done by computing a ratio of frequences and repeating every element in the input with this ratio

5 SAMPLERNN VOCODER

A SampleRNN (Mehri et al., 2016) is a natural extension to the classical recurrent neural network that uses a hierarchy to support RNN with catching long-term dependencies.

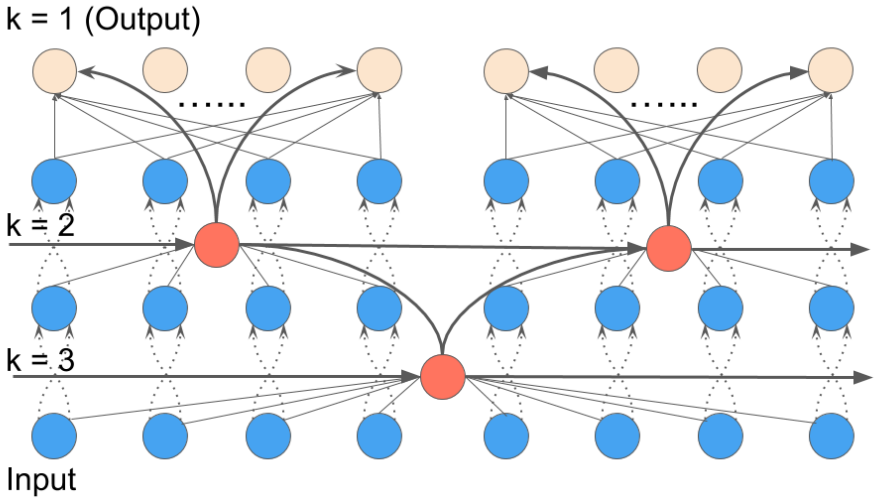

Figure 6: An example of unconditional SampleRNN with 3 tiers operating on frame sizes of 8, 4 and 4 respectively. Dashed lines denote that we use different linear projections of the input sequence at each tier.

The model consists of a multiple RNNs that work on separate depths ("tiers"). Instead of using a single sample, at k-th tier the corresponding RNN operates on non-overlapping concatenations ("frames") of $FS^{(k)}$ samples. To make a hierarchy, these are summed with a linear projection of the hidden state that comes from the higher tier.

That's it, if at k-th tier and t-th timestamp the RNN operates on a frame $f_t^{(k)}$, the input to the recurrent cell is:

$$inp_t^{(k)} = W_t^{(k)} f_t^{(k)} + V_{par,t}^{(k+1)} h_{par}^{(k+1)}, \tag{19}$$

where K is a total number of tiers, $h_{par}^{(k+1)}$ is a hidden state of the "parent" cell (such that $f_{par}^{(k+1)}$ contains f_t^k) and $(W^{(k)}, V^{(k)})$ are a series of tier-specific weights. To account for acoustic features a, the top input is:

$$inp_t^{(K)} = f_t^{(K)} + a_t, \tag{20}$$

And the highest RNN operate on the length L, so the model doesn't require upsampling.

Due to the large dimensionality of the output data, a training of recurrent networks even with the help of hierarchical structure is a challenging task. To make it more tractable, we apply additional tricks.

Firstly, because $FS^{(1)}$ is usually a small value, at the lowest tier we may replace RNN with a single feedforward network which is shared between multiple timesteps. Additionally, Mehri et al. (2016) found it beneficial to embed each sample before concatenating it into $f_t^{(K)}$ with the help of a jointly trainable matrix. This embedding is done only at the highest tier.

Secondly, we use truncated backpropagation through time at every tier, splitting each sequence into short subsequences and propagating gradients only to the beginning of each subsequence.

Similarly to WaveNet, the model is trained to predict quantized values via softmax layer by minimizing negative log-likelihood. At training time, the computations at the first tier may be parallelized due to the shared FFN. However, at test time, the model generates one sample at a time and no acceleration can be made. This "one-step" prediction is crucial for the resulting performance (Mehri et al., 2016).

6 COMPARISON

6.1 READERS COMPARISON

Module	Computation Shapes	# parameters
ARSG Encoder(hid size $= 128$)	$(T, 42) \to (T, 256)$	$2[3(42 + 256)256] \approx 450,000$
ARSG $N_{\text{Att}}(K = 10)$	$256 \to 25 \to 30$	$\approx 7,000$
ARSG N_{Gen}	$512 \to 51 \to 63$	$\approx 30,000$
ARSG Decoder(hid size $= 256$)	$(T, 256) \to (T, 256)$	$3(512 + 63)256 \approx 450,000$
ARSG Total	$(T, 42) \to (T, 63)$	$\approx 940,000$
VoiceLoop Embedding	$(T, 42) \to (T, 256)$	$\approx 10,000$
VoiceLoop N_{Att} $(K = 10, \text{buffer size} = 20)$	$20(63 + 256) \approx 6380 \to 638 \to 30$	$\approx 4,000,000$
VoiceLoop N_{Upd}	$6380 + 63 + 42 \to 648 \to 319$	$\approx 4,500,000$
VoiceLoop N_{Out}	$6380 \to 638 \to 63$	$\approx 4,000,000$
VoiceLoop Total	$(T, 42) \to (T, 63)$	$\approx 12,500,000$

Table 1: An approximate number of parameters for instances of ARSG and VoiceLoop models. Both receive one-hotted phonetic features of length T, use $K = 10$ components in Graves Attention and return 63 vocoder features ($L = T$).

Method	VCTK22	VCTK65
ARSG+SampleRNN(Char2Wav)	2.84 ± 1.20	2.85 ± 1.19
VoiceLoop+WORLD	3.57 ± 1.08	3.40 ± 1.00
Ground Truth	4.61 ± 0.75	4.59 ± 0.72

Table 2: Multi-speaker mean opinion scores (Mean$\pm$SE) on VCTK dataset (Veaux, 2017), as reported by Taigman et al. (2017)

In comparison with ARSG, VoiceLoop usually performs better and get more pleasant samples, e.g. see table 2. The phonological loop model efficiently shares weights between multiple speakers and exploits an intuitive concept of a FIFO buffering, which can be considered as a generalization of an ARSG decoder net and somehow resembles the actual "human" type of listening.

However, as can be seen on table 1, the generalization comes at a price of increased model complexity. Despite the fact that encoder network is replaced with a linear projections, the total amount of parameters is ≈ 10 times higher.

Thus, while training model to extract vocoder features, we propose to begin with a simpler ARSG model with the teacher-forcing technique, similar to 15 in section 3 and shift on to the VoiceLoop model only after the adequate results are reached. To speed up computations, it might also be beneficial to use Quasi-Recurrent cells (Bradbury et al., 2016).

Additionally, it should be mentioned that both models are autoregressive, generating sequences step-by-step. However, for now TTS operates on a small text sequences and that doesn't pose such a problem, at least in comparison with the audio generation.

6.2 VOCODERS COMPARISON

The main advantage of a WaveNet model over SampleRNN is that it can be efficiently parallelized during training, as the computations at t-th timestep doesn't depend on the computations at previous

Model	Blizzard	Onomatopoeia
LSTM	1.434	2.034
WaveNet	1.480	2.285
SampleRNN (2-tier)	1.392	2.026
SampleRNN (3-tier)	1.387	1.990

Table 3: Test NLL on a set of datasets, as reported by Mehri et al. (2016)

timesteps. That leads to a much deeper architecture to account for the long-term dependencies even with the help of dilated convolutions. As a result, WaveNets are far more competitive to train, calling for larger amounts of parallel threads to benefit from their structure.

Moreover, at test time, WaveNet generates audio step-by-step, taking a long time to produce even short samples (172 timesteps/second on a NVIDIA P100 GPU as reported by authors). To fix that, recently it was proposed to use a non-autoregressive version of this model, inspired by the concept of inverse autoregressive flows (Kingma et al., 2016), that produces a new output conditioning on the random input (van den Oord et al., 2017) rather than previous outputs. To optimize this "student" network efficiently a "teacher" pre-trained autoregressive WaveNet is used and the model is learned to mimic "teacher" distribution with the help of KL-divergence loss. However, this trick leads to an even further complexity increase, as we need to use additional non-trivial regularization terms to motivate "student" to generate high quality audio streams. Also, one still needs to have a causal pre-trained "teacher" for this model to train.

On the other hand, SampleRNN has a much lower depth and a simpler structure. When it's trained, it gives a better reconstruction quality in terms of Negative Log Likelihood, as can be seen on table 3. The use of truncated BPTT can sufficiently speed up the training, and due to the structure the model doesn't require an upsampling procedure. Unfortunately, despite these advantages, Mehri et al. (2016) reported that it took them about a week to train SampleRNN on a GeForce GTX TITAN X, which raises serious concerns about the applicability of this method. Potentially, similarly to readers, the use of QRNNs can boost the model even further while achieving the same results, but we are not aware of such experiments.

To sum up, we suppose that for now a WaveNet is a better solution when generating a long (> 5s) output signals, especially when being trained on GPUs, while the SampleRNN might be preferred to deal with short sequences. Overall, we believe that there is a sweep pot between the use of "shallow" recurrent layers and "deep" dilated convolutions. Therefore, we think it is important to account for SampleRNN structure to come up with a "combined" architecture.

REFERENCES

Sercan Ömer Arik, Mike Chrzanowski, Adam Coates, Greg Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, and Mohammad Shoeybi. Deep voice: Real-time neural text-to-speech. *CoRR*, abs/1702.07825, 2017a. URL http://arxiv.org/abs/1702.07825.

Sercan Ömer Arik, Gregory F. Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. *CoRR*, abs/1705.08947, 2017b. URL http://arxiv.org/abs/1705.08947.

A.D. Baddeley. Working memory. *London: Oxford University Press*, 1986.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL http://arxiv.org/abs/1409.0473.

S. Bai, J. Zico Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv e-prints*, March 2018.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *CoRR*, abs/1611.01576, 2016. URL http://arxiv.org/abs/1611.01576.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015. URL http://arxiv.org/abs/1506.07503.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

Diederik P. Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016. URL http://arxiv.org/abs/1606.04934.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *CoRR*, abs/1612.07837, 2016. URL http://arxiv.org/abs/1612.07837.

Masanori Morise, Fumiya YOKOMORI, and Kenji Ozawa. World: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D:1877–1884, 07 2016.

Yaniv Taigman, Lior Wolf, Adam Polyak, and Eliya Nachmani. Voice synthesis for in-the-wild speakers via a phonological loop. *CoRR*, abs/1707.06588, 2017. URL http://arxiv.org/abs/1707.06588.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016a. URL http://arxiv.org/abs/1609.03499.

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016b. URL http://arxiv.org/abs/1606.05328.

Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. 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. *CoRR*, abs/1711.10433, 2017. URL http://arxiv.org/abs/1711.10433.

Junichi; MacDonald Kirsten Veaux, Christophe; Yamagishi. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. 2017. URL http://dx.doi.org/10.7488/ds/1994.

Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: A fully end-to-end text-to-speech synthesis model. *CoRR*, abs/1703.10135, 2017. URL http://arxiv.org/abs/1703.10135.

R. L. Weide. The cmu pronouncing dictionary. 1998. URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Zhizheng Wu, Oliver Watts, and Simon King. *Merlin: An Open Source Neural Network Speech Synthesis System*, pp. 218–223. 9 2016.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.