Attention, I'm Trying to Speak
CS224n Project: Speech Synthesis

Akash Mahajan
Management Science & Engineering
Stanford University
akashmjn@stanford.edu

Abstract

We implement an end-to-end parametric text-to-speech synthesis model that produces audio from a sequence of input characters, and demonstrate that it is possible to build a convolutional sequence to sequence model with reasonably natural voice and pronunciation from scratch in well under $75. We observe training the attention to be a bottleneck and experiment with 2 modifications. We also note interesting model behavior and insights during our training process. Code for this project is available on: https://github.com/akashmjn/cs224n-gpu-that-talks.

1 Introduction

We have come a long way from the ominous robotic sounding voices used in the Radiohead classic [1]. If we are to build significant voice interfaces, we need to also have good feedback that can communicate with us clearly, that can be setup cheaply for different languages and dialects. There has recently also been an increased interest in generative models for audio [6] that could have applications beyond speech to music and other signal-like data.

As discussed in [13] it is fascinating that is is possible at all for an end-to-end model to convert a highly "compressed" source - text - into a substantially more "decompressed" form - audio. This is a testament to the sheer power and flexibility of deep learning models, and has been an interesting and surprising insight in itself. Recently, results from Tachibana et. al. [11] reportedly produce reasonable-quality speech without requiring as large computational resources as Tacotron [13] and Wavenet [7]. Motivated by the above, we set out to build our own implementation, and run experiments on variations of this model, reporting some interesting observations and insights.

2 Background and Related Work

Most production speech synthesis systems have historically been based on unit-selection as in Siri [2], with more recent attention being gained by end-to-end parametric models such as Google Wavenet, Tacotron and Baidu DeepVoice, that are computationally more expensive.

In 2016 Google WaveNet [7] made waves in the audio community with an end-to-end model working directly with raw audio samples. Follow-up work [8] has shown that is in infamously hugely computationally expensive to train from scratch. Post that, Tacotron [13], Tacotron 2 [10], and DeepVoice 2 [1], took a sequence to sequence approach to generating audio by working in the frequency domain. These models too were quite computationally expensive to run taking about 1-2 weeks to train.

[1]Radiohead - Fitter, Happier (1997)[link]

Module	Layers
TextEnc	$(HC_{1*1}^{2d\leftarrow 2d})^2 \lhd (HC_{3*1}^{2d\leftarrow 2d})^2 \lhd (HC_{3*27}^{2d\leftarrow 2d} \lhd HC_{3*9}^{2d\leftarrow 2d} \lhd HC_{3*3}^{2d\leftarrow 2d} \lhd HC_{3*1}^{2d\leftarrow 2d})^2$ $\lhd C_{1*1}^{2d\leftarrow 2d} \lhd \text{ReLu} \lhd C_{1*1}^{2d\leftarrow e} \lhd \mathbf{L}^e$
AudioEnc	$(HC_{3*3}^{d\leftarrow d})^2 \lhd (HC_{3*27}^{d\leftarrow d} \lhd HC_{3*9}^{d\leftarrow d} \lhd HC_{3*3}^{d\leftarrow d} \lhd HC_{3*1}^{d\leftarrow d})^2 \lhd$ $C_{1*1}^{d\leftarrow d} \lhd \text{ReLU} \lhd C_{1*1}^{d\leftarrow d} \lhd \text{ReLU} \lhd C_{1*1}^{2d\leftarrow F}(\mathbf{S}^F)$
AudioDec	$\sigma \lhd C_{1*1}^{F\leftarrow d} \lhd (\text{ReLU} \lhd C_{1*1}^{d\leftarrow d})^3 \lhd (HC_{3*1}^{d\leftarrow d})^2 \lhd$ $(HC_{3*27}^{d\leftarrow d} \lhd HC_{3*9}^{d\leftarrow d} \lhd HC_{3*3}^{d\leftarrow d} \lhd HC_{3*1}^{d\leftarrow d}) \lhd C_{1*1}^{d\leftarrow 2d}(\hat{\mathbf{R}}^{2d})$
SSRN	$\sigma \lhd C_{1*1}^{F_o\leftarrow F_o} \lhd (\text{ReLU} \lhd C_{1*1}^{F_o\leftarrow F_o})^2 \lhd C_{1*1}^{F_o\leftarrow 2c} \lhd (HC_{3*1}^{2c\leftarrow 2c})^2 \lhd C_{1*1}^{2c\leftarrow c} \lhd$ $(HC_{3*3}^{c\leftarrow c} \lhd HC_{3*1}^{c\leftarrow c} \lhd D_{2*1}^{c\leftarrow c})^2 \lhd (HC_{3*3}^{c\leftarrow c} \lhd HC_{3*1}^{c\leftarrow c}) \lhd C_{1*1}^{c\leftarrow F}(\hat{\mathbf{Y}}^F)$

Table 1: Details about layers for each module

Recently convolutional sequence-to-sequence (seq2seq) [3] models have shown how training can be massively sped up by using only convolutional layers and attention. For our experiments, we implement a simpler model [11] based on this approach that is also quite similar to the approach in Baidu DeepVoice3 [9].

3 Model Architecture

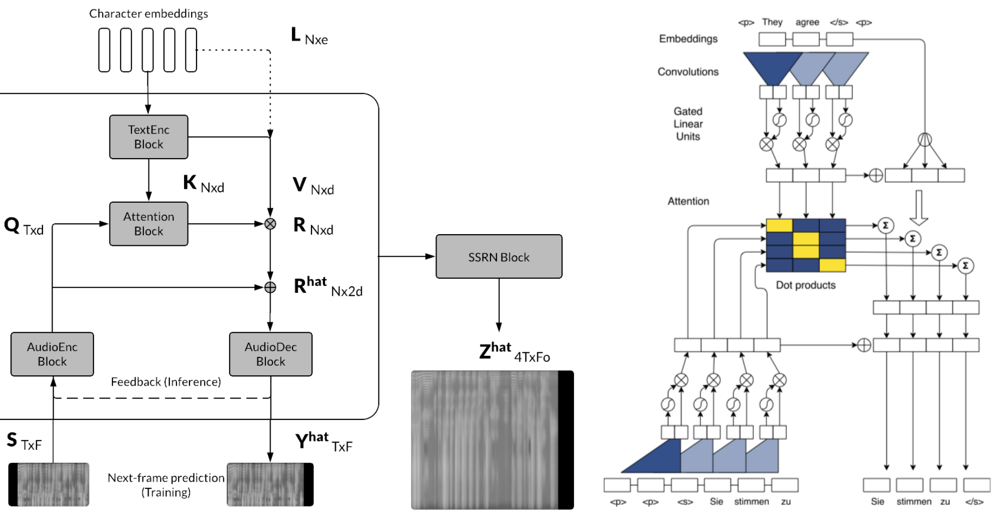

Figure 1: Schematic of the model architecture. (Left) Overview of the two stages Text2Mel and SSRN indicating important modules and dimensions. During training, the feedback input $\mathbf{S}$ consists of shifted target frames $\mathbf{Y}$, while during inference, the next output is appended and fed back and continued for a max number of frames. (Right) Detailed view from the original convolutional seq2seq paper (in our case, inputs on the lower half are a single spectrogram frame). The difference between causal and non-causal convolutions is highlighted, as well as the addition of positional encodings.

In this section, we describe the model architecture we implemented (see Fig. 1 for a detailed schematic). The expected inputs are a sequence of N token embeddings $\mathbf{L}_{N\times e}$ which could represent either characters or phonemes[2] (in our case, we use a simple set of 32 trainable character embeddings for normalized text input). The model produces a sequence of log-magnitude spectrogram frames $\hat{\mathbf{Z}}_{r*T\times F_o}$ which is approximately inverted to audio using the Griffin-Lim [4] algorithm.

[2]https://en.wikipedia.org/wiki/Phoneme

3.1 Pre processing

3.1.1 Text Normalization

We preprocess the text transcriptions converting all inputs to lowercase to a simple character set of `a-z'.?<space>PE` where P is a special token denoting padding from the convolutions, and E is an end token added to the end of every input sentence. For example, an input `Attention I'm trying to speak!` is converted to `attention i'm trying to speak E` where special characters outside our vocabulary are replaced with a space, and numbers and abbreviations are spelled out as in `The exchange of letters dated August 31, 1964,` to `the exchange of letters dated august thirty one nineteen sixty four .` It is easy to extend this to support abbreviations, numbers, dates etc. however we stuck with a simple approach here.

3.1.2 Signal processing basics

This model operates in a more compact representation in the frequency domain converting audio to log-magnitude spectrograms [3] that are essentially freq vs time matrices obtained by taking a windowed FFT over short chunks of an input audio signal, also known as an STFT. The result $Z_{r*T\times F_o}$ is complex-valued and normalized by using the magnitude on a log-scale, clipping to a minimum value in dB (decibels), and normalizing with respect to the maximum value $|Z|/\max(|Z|)$. The resulting normalized $Z \in [0, 1]$.
The Text2Mel model uses an intermediate lower resolution representation: the mel-scale spectrogram $Y_{T\times F}$ that contains a subset of the frequencies in Z that are more perceptible to the human ear. As per Tachibana et. al a smaller number of time frames ($r = 4$) is used to correspond to the full-resolution spectrogram while using the SSRN to predict the higher-resolution Z from Y. As in Tacotron [13], we also use a pre-emphasis factor γ to pre-process the raw audio before the STFT. (details of these parameters are described in the Appendix).

For our case, this results in audio samples 10-20s long, sampled at 22050Hz, transformed into Y of size $80 \times T$ and Z of size $513 \times 4T$ where T depends on the length of the input (it is typically not more than 250 for this dataset).

3.2 Model block details

The architecture consists of the following main components, with the equations below describing the model exactly:

1. **Text2Mel**: A convolutional sequence to sequence model producing lower-resolution mel-scale log magnitude spectrogram frames $\hat{\mathbf{Y}}_{T\times F}$ given an input character sequence $\mathbf{L}_{N\times e}$. This consists of:
   (a) TextEnc: A non-causal block consisting of a deep stack of 1-D convolutions, dilated convolutions and highway activations padded to keep the output length the same. Details are in Table 1 with $C_{3*1}^{F\leftarrow c}$ denoting a 1-D convolution of stride 1, dilation factor of 3 and kernel size of 1. Highway activations $HC_{3*1}^{d\leftarrow d}$ (notation as in original paper) are a variation of gated activations described in Tachibana et. al and in [13].
   (b) AudioEnc: A similar block as above except all convolutions are causal. This module encodes the previously generated audio.
   (c) AudioDec: A causal block similar to AudioEnc that applies a sigmoid at the output to decode to mel frames normalized in $[0, 1]$.
2. **SSRN (Spectrogram super-resolution network)**: A non-causal block that increases the frequency channels from $F = 80$ to $F_o = 513$ via and increase in filter channels, and time samples from T to $4T$ using two deconvolution layers with a stride 2, outlined in Table 1.

[3]https://timsainb.github.io/spectrograms-mfccs-and-inversion-in-python.html

$$\mathbf{K}, \mathbf{V}_{N\times d} = TextEnc(\mathbf{L}_{N\times e}) \tag{1}$$
$$\mathbf{Q}_{T\times d} = AudioEnc(\mathbf{S}_{T\times F}) \tag{2}$$
$$\mathbf{S}_{T\times F} = 0 \oplus \mathbf{Y}_{0:F-1} \qquad \text{(shifted left)} \tag{3}$$
$$\mathbf{R}_{T\times d} = \mathbf{AV} = softmax\left(\frac{\mathbf{QK}^T}{\sqrt{d}}\right)\mathbf{V} \tag{4}$$
$$\hat{\mathbf{Y}}_{T\times F} = AudioDec(\mathbf{R} \oplus \mathbf{Q}) \tag{5}$$
$$\hat{\mathbf{Z}}_{4T\times F_o} = SSRN(\mathbf{Y}) \tag{6}$$
$$\mathbf{J}_{L1} = \mathbb{E}|\mathbf{Y} - \hat{\mathbf{Y}}| \tag{7}$$
$$\mathbf{J}_{CE} = -\mathbb{E}[\mathbf{Y}\log\hat{\mathbf{Y}} + (1-\mathbf{Y})\log(\mathbf{1}-\hat{\mathbf{Y}})] \tag{8}$$

$$\mathbf{S}_{1:t+1,F} = \mathbf{S}_{1:t,F} \oplus \hat{\mathbf{Y}}_{t,F} \qquad \text{(feedback)} \tag{9}$$
$$\hat{\mathbf{Z}}_{4T\times F_o} = SSRN(\hat{\mathbf{Y}}) \tag{10}$$
$$\tag{11}$$

Both models are trained as per Tachibana et. al. with a combination of the L1 loss with target mel-spectrograms and full magnitude spectrograms, along with an additional binary cross entropy loss for each pixel in $\hat{Y}$. This to us seemed like an odd choice of loss function that the authors had justified due to the non-saturating nature of its gradient through the final sigmoid layers in our networks. We try validating this with our model M5.

3.3 Attention

For our base model M1 we implement a standard scaled dot-product attention. We noticed in our experiments that properly learning the attention was the bottleneck for this model, which did not converge for us without adding 2 modifications.

- As in Tachibana et. al. we implement an additional guided attention loss term $\mathbf{J}_{att} = \mathbb{E}(A \circ W), W_{n,t} = (1 - \exp(-n/N) - t/T)^2/2g^2$ with $g = 0.2$. From the equation, we can see that this penalizes terms that are far away from the diagonal, hence is a way of enforcing monotonicity.
- We used positional encodings mentioned in [12] which gave us our best performing model.

3.4 Output Synthesis

Converting from time-domain audio to frequency domain magnitudes is a lossy operation and cannot be inverted back exactly since we lose information about the complex phase when only considering magnitude. Hence we use the Griffin Lim algorithm [4] to invert the prediction from the SSRN network back to an audio signal. This is actually the main source of audible artifacts in the sound samples we generate. Also, a post-emphasis factor (see Appendix) is used to exponentiate the spectrograms before inversion which reportedly improves quality [9].

4 Experiments and Analysis

4.1 Experimental setup

For our experiments, we used the LJ Speech Dataset [5] that is publicly available, and contains $\sim 13k$ pairs of unaligned audio (.wav) & sentence-transcript pairs. These consist of passages from 7 non-fiction English books read by a single American female speaker. Clips vary in length from 1 to 10 seconds and have a total length of approximately 24 hours, and on manual inspection have a slight reverbed sound to them that might be picked up by a model.

We implemented our models in Tensorflow that were trained on single Tesla M60 GPUs with 8GB memory each and a batch size of 16. These were trained with the Adam optimizer with a learning rate of 4×10^{-4} using the Noam learning schedule used in [12]. Since our model architecture is considerably complex and consists of many hyperparameters including signal processing, configurations of different modules and optimizer parameters, these are tabulated in detail in the Appendix.

The original paper [11] reports 3.8 and 6.4 steps/s and totally about $200k$ steps for each model. Probably due to improperly tuned queues in our data pipeline, we get about 1.8-2 steps/s for each model. To ease running experiments, all our models were trained for $60k$ steps (Text2Mel) and $100k$ steps (SSRN) with training times taking about 9 hours and 19 hours respectively. At this point the learning curves flatten out, and the models already begin to produce decent quality outputs.

We focus on the Text2Mel model using the L1 loss and the guided attention loss metrics on the training/validation sets to quantify an estimate of the predicted audio quality and the quality of attention. A final subjective analysis of the outputs is done. An overview containing comparisons is in Table 2.

4.2 Attention variations

Text-to-speech differs from typical sequence to sequence models applied to neural machine translation (NMT) in that we are trying to learn relatively longer alignments (250 time samples x 200 characters) corresponding to unlikely sentence lengths, often ignored in NMT models. We feel that this might be the reason that we observe difficulty in getting the attention to converge (see Figure 2).

To improve this, we exploit the fact that the alignments in this case are also much simpler. Unless we consider intonation that requires understanding context and meaning of sentences with many clauses, such as this sentence itself, the alignment between characters and spoken utterances is more or less monotonic and not as complex as for an NMT task.

To address this, we experiment with two variations. First, using a ”guided attention” loss $\mathbf{J}_{att}(A)$ during training that helps enforce monotonicity from Tachibana et. al [11]. Second, adding ”positional encodings” h_p as discussed in [9], [3], [12].

4.2.1 Guided Attention Loss: Models M1, M2

For M1, the standard attention module fails to converge as seen in Fig. 2, though surprisingly the L1 loss value still continues to be optimized. On inspection, the decoder blocks (AudioEnc and AudioDec) apparently continue to learn step-ahead predictions without any context from the input characters. Attempting to generate samples from this model produces an interesting result - where we get samples containing gibberish utterances that sound a lot like natural speech with common sounds and phonemes [4]. This is an interesting result that resembles an *audio language model* of sorts, that is modeling local context/texture by making step-ahead predictions.

4.2.2 Positional, Local Encodings: Models M3, M4, Loss function edit: M5

As in Fig. 2 and discussed in [3], [9], for M3 we tried a direct ”skip” addition of the input embeddings $\mathbf{L}$ to values $\mathbf{V}$ so that it contains both local context of each character, as well as higher-level context from the convolutions. However as in Table. 2 this doesn’t lead to much improvement. This might be since the gated highway activation layers are able to retain this information.

For M4, we implemented a more established approach using position embeddings, introduced in [12] that is also used in DeepVoice3 [9]. As in Fig. 1 positional encodings h_p are added to key vectors $\mathbf{K}$ and query vectors $\mathbf{Q}$ according to the formula $h_p(i) = sin(\omega_s i/10000^{k/d})$ (for even i) $cos(\omega_s i/10000^{k/d})$, where the *position rate* $\omega_s = 1$ for queries and $\omega_s = 1.48$ for keys (ratio of avg. T/N). This produces the best performing model in Table 2.

In M5, we test whether the cross entropy loss $\mathbf{J}_{CE}$ is actually beneficial to training. We observe that while the model is able to do quite well in performance, the attention alignments are not as smooth, from the guided attention loss. However, when we examine the learned character embeddings by the

[4]Link to audio samples: [link]

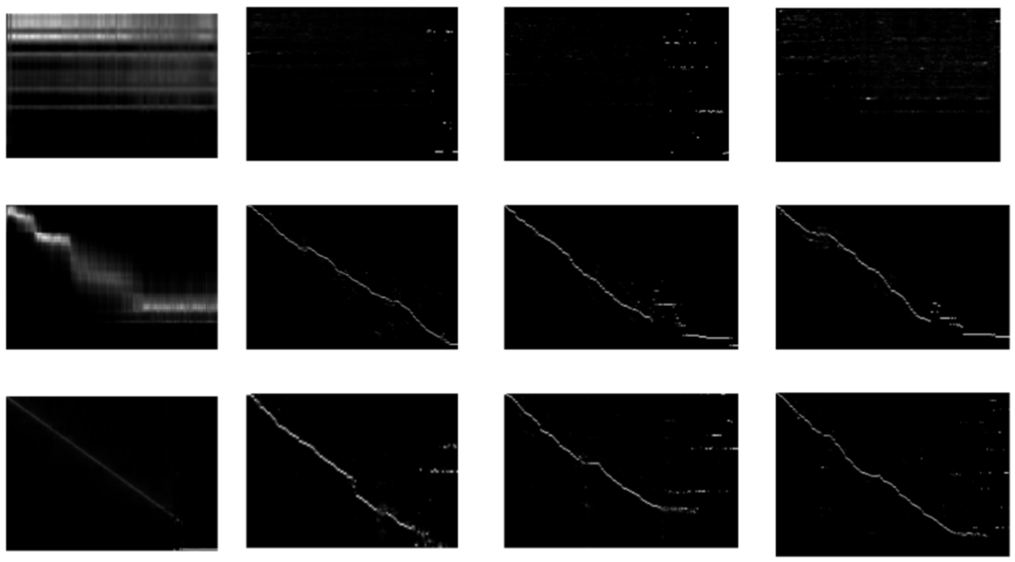

Figure 2: Comparison of learned alignments for models M1 (top) and M2 (middle) M5 (bottom), with steps proceeding from left to right at 5k, 25k, 40k and 60k iterations. The attention for M1 has clearly not yet converged. The leftmost frames for each show M1 starting off very diffuse, guided attention in M2 shaping this diffuse distribution more diagonally, and the positional encodings creating a sharp diagonal initialization. In middle-right and bottom-right, we also see two interesting failure modes. First, the attention has breaks/jumps at certain characters sequences, and series of breaks at the end corresponds to the end/silence.

models (see Fig. 3) , we find that models M4 and M5 have clearer logical clusters, with M5 being marginally better that M4.

4.2.3 Subjective Evaluation & Conclusion

The quantitative comparison of models in Table 2 indicates that all models are overfitting to a small degree on the training data. While models other than M2, M3 were not trained with the guided attention loss, it gives a rough indication of how clean the alignments are. Our final evaluation is to actually listen to a sample of 20 sentences generated from from Harvard Sentences [5] (included in the Appendix) and perform a subjective evaluation. Some insights:

1. M1 clearly is not able to generalize well, given that attention alignments are not converged.
2. While the loss numbers for M2 and M3 might seem okay, there is noticably poorer audio quality and well as clarity of pronunciations in comparison with models M4 and M5. Model M4 sounds distinctly better than the rest with lower artifacts and clear pronunciation on sentences like ”The box was thrown beside the parked truck.” [6]

We acknowledge that this is a pretty informal method of evaluation compared to a typical Mean Opinion Score (MOS) used in literature crowd-sourced via Amazon Mechanical Turk. However with limited resources, we hope that our combination of these metrics with observations made during the training process provides some interesting insights.

4.3 Analysis of embeddings

The last experiment was to try to visualize the learned character embeddings $\mathbf{L}$. The model has to implicitly map characters to phonemes and words that form the units of spoken utterances. Espe-

[5]A standard set used to benchamrk VoIP quality: https://en.wikipedia.org/wiki/Harvard_sentences

[6]Link to audio samples: [link]

Table 2: Comparison of loss scores for model variations

Model	Variation	L1 (Train)	L1 (Validation)	Guided Attention (Validation)
M1	Standard attention, CE loss	0.0288	**0.0611**	$\mathbf{27.5 \times 10^{-4}}$
M2	M1 + guided attention	0.0249	0.0484	3.99×10^{-4}
M3	M2 + local char encodings	0.0245	0.0485	4.19×10^{-4}
M4	M1 + positional encodings	**0.0230**	0.0490	8.42×10^{-4}
M5	M4 without CE loss	0.0235	0.0490	17×10^{-4}

cially in a language like English where the same characters are spoken many different ways this is not trivial. We observe a simple visualization of the learned character embeddings $\mathbf{L}$ in Fig. 3 with some interesting behavior from the model. We could potentially analyze further by examining the output encodings produced by TextEnc, and seeing how these change for words like ”car” and ”cat”.

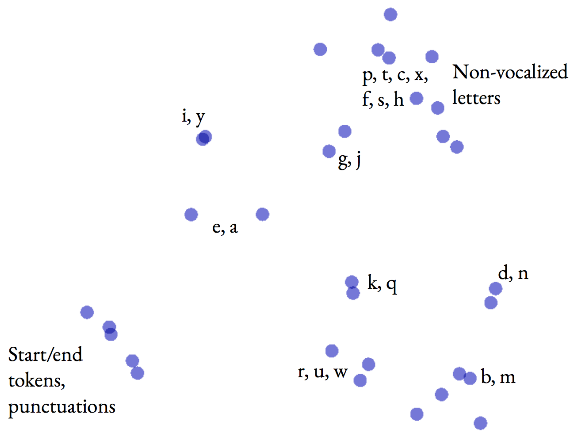

Figure 3: Visualization of implicitly-learned character embeddings. Characters tend to cluster based on their semantics, as well as those with similar sound. We notice an interesting separation of characters with non-vocalized sounds (percussive sounds like ”sh” ”th” etc. that don’t engage our voice)

5 Conclusion and Future Work

In conclusion, an interesting learning from this project has been an appreciation of the sheer power and flexibility of deep learning models to learn alignments, character-phoneme relationships, character embeddings, and an *audio language model* simultaneously. While we used a larger amount of cloud compute credits due to the various experiments we ran - given our training times it is quite possible to build a model with decent prosody, voice and pronunciation in under $75 using our code.

We plan to try to extend this model to learning different languages to get a better estimate of the above. It is possible to improve quality by constraining the attention at inference as in [9]. Also, we would like to experiment with two semi-supervised methods for a better initialization of the character embeddings $\mathbf{L}$ and weights of AudioEnc and AudioDec. This could be done via Word2Vec-like embeddings for $\mathbf{L}$, and separately training the *audio language model* (see Section 4.2.1) on audio without labeled transcripts. This could be useful when labeled transcripts might not be as abundant for other languages. Finally, we are also interested in extending this general idea of ’symbols-to-sound’ to a more general problem of generating audio stylistically from a sequence of token inputs (this could be MIDI inputs for music or similar).

Acknowledgments

We are grateful to Richard Socher for inspiring us to take on the large undertaking of implementing a complex model such as this ourselves, and for timely motivation and advice. We are also grateful to Ziang Xie and Kevin Clark for their ideas at the outset, and near the end of this project respectively - that led to interesting explorations and experiments. Lastly, we would like to acknowledge open-source Github contributors Kyubyong Park and r9y9 whose code was a great source of education, and that we referenced for our data pipeline, pre-processing and signal processing. Having some reference and knowing that this was possible was invaluable to getting our implementation working and debugging the same.

References

[1] S. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou. Deep Voice 2: Multi-Speaker Neural Text-to-Speech. 5 2017.

[2] T. Capes, P. Coles, A. Conkie, L. Golipour, A. Hadjitarkhani, Q. Hu, N. Huddleston, M. Hunt, J. Li, M. Neeracher, K. Prahallad, T. Raitio, R. Rasipuram, G. Townsend, B. Williamson, D. Winarsky, Z. Wu, and H. Zhang. Siri On-Device Deep Learning-Guided Unit Selection Text-to-Speech System. In *Interspeech 2017*, pages 4011–4015, ISCA, 8 2017. ISCA.

[3] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional Sequence to Sequence Learning. 5 2017.

[4] D. W. Griffin, D. W. Griffin, Jae, S. Lim, and S. Member. Signal estimation from modified short-time fourier transform. *IEEE TRANS. ACOUSTICS, SPEECH AND SIG. PROC*, pages 236–243, 1984.

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

[6] 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. 2 2018.

[7] 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. 9 2016.

[8] 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, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. 11 2017.

[9] 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. 10 2017.

[10] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu. NATURAL TTS SYNTHESIS BY CONDITIONING WAVENET ON MEL SPECTROGRAM PREDICTIONS.

[11] H. Tachibana, K. Uenoyama, and S. Aihara. Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention. 10 2017.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. 6 2017.

[13] 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. 3 2017.

Appendix

Table 3: Hyperparameters for model architecture

Parameter	Value
Vocabulary	`a-z'.?<space>PE`
Padding token	`P`
End token	`E`
Sampling Rate	22050Hz
Reduction factor	4
Pre-emphasis factor	0.97
Sharpening factor	1.3
Min dB	-100
Reference dB	30 (used to set max level during generation)
n_fft	1024
hop length	256
Griffin-Lim iterations	150
max_N	200
max_T	250
e, d, F, c, F_o	128, 256, 80, 512, 513
learning rate	4×10^{-4}
Noam warmup steps	4000
beta1, beta2	0.6, 0.95
L1 loss weight	5.0
batch size	16

Test sentences from Harvard Sentences used for subjective evaluation

```
1. The birch canoe slid on the smooth planks.
2. Glue the sheet to the dark blue background.
3. It's easy to tell the depth of a well.
4. These days a chicken leg is a rare dish.
5. Rice is often served in round bowls.
6. The juice of lemons makes fine punch.
7. The box was thrown beside the parked truck.
8. The hogs were fed chopped corn and garbage.
9. Four hours of steady work faced us.
10. Large size in stockings is hard to sell.
11. The boy was there when the sun rose.
12. A rod is used to catch pink salmon.
13. The source of the huge river is the clear spring.
14. Kick the ball straight and follow through.
15. Help the woman get back to her feet.
16. A pot of tea helps to pass the evening.
17. Smoky fires lack flame and heat.
18. The soft cushion broke the man's fall.
19. The salt breeze came across from the sea.
20. The girl at the booth sold fifty bonds.
```