Parrotron: An End-to-End Speech-to-Speech Conversion Model and its Applications to Hearing-Impaired Speech and Speech Separation

Fadi Biadsy, Ron J. Weiss, Pedro J. Moreno, Dimitri Kanvesky, Ye Jia

Google

{biadsy,ronw,pedro,dkanevsky,jiaye}@google.com

Abstract
We describe Parrotron, an end-to-end-trained speech-to-speech conversion model that maps an input spectrogram directly to another spectrogram, without utilizing any intermediate discrete representation. The network is composed of an encoder, spectrogram and phoneme decoders, followed by a vocoder to synthesize a time-domain waveform. We demonstrate that this model can be trained to normalize speech from any speaker regardless of accent, prosody, and background noise, into the voice of a single canonical target speaker with a fixed accent and consistent articulation and prosody. We further show that this normalization model can be adapted to normalize highly atypical speech from a deaf speaker, resulting in significant improvements in intelligibility and naturalness, measured via a speech recognizer and listening tests. Finally, demonstrating the utility of this model on other speech tasks, we show that the same model architecture can be trained to perform a speech separation task.

1. Introduction
Encoder-decoder models with attention have recently shown considerable success in modeling a variety of complex sequence-to-sequence problems. These models have been successfully adopted to tackle a diverse set of tasks in speech and natural language processing, such as machine translation [1], speech recognition [2], and even combined speech translation [3]. They have also achieved state-of-the-art results in end-to-end Text-To-Speech (TTS) synthesis [4] and Automatic Speech Recognition (ASR) [5], using a single neural network that directly generates the target sequences, given virtually raw inputs.

In this paper, we combine attention-based speech recognition and synthesis models to build a direct end-to-end speech-to-speech sequence transducer. This model generates a speech spectrogram as a function of a different input spectrogram, with no intermediate discrete representation.

We test whether such a unified model is powerful enough to normalize arbitrary speech from multiple accents, imperfections, potentially including background noise, and generate the same content in the voice of a single predefined target speaker. The task is to project away all non-linguistic information, including speaker characteristics, and to retain only what is being said, not who, where or how it is said. This amounts to a text-independent, many-to-one voice conversion task [6]. We evaluate the model on this voice normalization task using ASR and listening studies, verifying that it is able to preserve the underlying speech content and project away other information, as intended.

We demonstrated that the pretrained normalization model can be adapted to perform a more challenging task of converting highly atypical speech from a deaf speaker into fluent speech, significantly improving intelligibility and naturalness. Finally,

we evaluate whether the same network is capable of performing a speech separation task. Readers are encouraged to listen to sound examples on the companion website. 1

A variety of techniques have been proposed for voice conversion, including mapping code books [7], neural networks [8, 9], dynamic frequency warping [10], and Gaussian mixture models [11–13]. Recent work has also addressed accent conversion [14, 15]. In this paper we propose an end-to-end architecture that directly generates the target signal, synthesizing it from scratch. It is most similar to recent work on sequence-to-sequence voice conversion [16–18]. [16] uses a similar end-to-end model, conditioned on speaker identities, to transform word segments from multiple speakers into multiple target voices. Unlike [17], which trained separate models for each source-target speaker pair, we focus on many-to-one conversion. Our model is trained on source-target spectrogram pairs, without augmenting inputs with bottleneck features from a pretrained speech recognizer to more explicitly capture phonemic information in the source speech [17]. However, we do find it helpful to multitask train the model to predict source speech phonemes. Finally, in contrast to [18], we train the model without auxiliary alignment or auto-encoding losses.

Similar voice conversion techniques have also been applied to improving intelligibility for speakers with vocal disabilities [19, 20], and hearing-impaired speakers in particular [21]. We apply more modern machine learning techniques to this problem, and demonstrate that, given sufficient training data, an end-to-end trained one-to-one conversion model can dramatically improve intelligibility and naturalness of a deaf speaker.

2. Model Architecture
We use an end-to-end sequence-to-sequence model architecture that takes an input source speech and generates/synthesizes target speech as output. The only training requirement of such a model is a parallel corpus of paired input-output speech utterances. We refer to this speech-to-speech model as Parrotron.

As shown in Figure 1, the network is composed of an encoder and a decoder with attention, followed by a vocoder to synthesize a time-domain waveform. The encoder converts a sequence of acoustic frames into a hidden feature representation which the decoder consumes to predict a spectrogram. The core architecture is based on recent attention-based end-to-end ASR models [2, 22] and TTS models such as Tacotron [4, 23].

2.1. Spectrogram encoder
The base encoder configuration is similar to the encoder in [24], and some variations are evaluated in Section 3.1. From the input speech signal, sampled at 16 kHz, we extract 80-dimensional log-mel spectrogram features over a range of 125-7600 Hz, cal-
with the same framing as the input features, and a 2048-point
The decoder targets are 1025-dim STFT magnitudes, computed
and not arbitrary audio, jointly training the encoder network to
mate a phase consistent with the predicted magnitude, followed
gram, we primarily use the Griffin-Lim algorithm [28] to esti-
\[\text{LSTM output is passed into a softmax to predict the probability}
\]toward a representation of the same underlying speech content.
We accomplish this by adding an auxiliary ASR decoder to pre-
dict the (grapheme or phoneme) transcript of the output speech,
conditioned on the encoder latent representation. Such a mul-
titask trained encoder can be thought of as learning a latent
representation of the input that maintains information about the
underlying transcript, i.e. one that is closer to the latent represen-
tation learned within a TTS sequence-to-sequence network.

The decoder input is created by concatenating a 64-dim
embedding for the grapheme emitted at the previous step, and
the 512-dim attention context. This is passed into a 256 unit LSTM
activation, to compute the final 512-dim encoder representation.

2.2. Spectrogram decoder

The decoder targets are 1025-dim STFT magnitudes, computed
with the same framing as the input features, and a 2048-point
FFT. We use the decoder network described in [4], consisting of
an autoregressive RNN to predict the output spectrogram from
the encoded input sequence one frame at a time. The prediction
from the previous decoder time step is first passed through
a small pre-net containing 2 fully connected layers of
256 ReLU units, which was found to help to learn attention
[4, 23]. The pre-net output and attention context vector are
concatenated and passed through a stack of 2 unidirectional
LSTM layers with 1024 units. The concatenation of the LSTM
output and the attention context vector is then projected through
a linear transform to produce a prediction of the target spectrogram
frame. Finally, these predictions are passed through 5-layer
convolutional post-net which predicts a residual to add to the
initial prediction. Each post-net layer has 512 filters shaped
5 \times 1 followed by batch normalization and tanh activation.

To synthesize an audio signal from the predicted spectro-
gram, we primarily use the Griffin-Lim algorithm [28] to esti-
rate a phase consistent with the predicted magnitude, followed
by an inverse STFT. However, when conducting human listening
tests we instead use a WaveRNN [29] neural vocoder which has
been shown to significantly improve synthesis fidelity [4, 30].

2.3. Multitask training with an ASR decoder

Since the goal of this work is to generate only speech sounds
and not arbitrary audio, jointly training the encoder network to
simultaneously learn a high level representation of the underly-
ing language serves to bias the spectrogram decoder predictions

<table>
<thead>
<tr>
<th>ASR decoder target</th>
<th>#CLSTM</th>
<th>#LSTM</th>
<th>Attention</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>1</td>
<td>3</td>
<td>Additive</td>
<td>27.1</td>
</tr>
<tr>
<td>Grapheme</td>
<td>1</td>
<td>3</td>
<td>Additive</td>
<td>19.9</td>
</tr>
<tr>
<td>Grapheme</td>
<td>1</td>
<td>3</td>
<td>Location</td>
<td>19.2</td>
</tr>
<tr>
<td>Phonenumber</td>
<td>1</td>
<td>3</td>
<td>Location</td>
<td>18.5</td>
</tr>
<tr>
<td>Phonenumber</td>
<td>0</td>
<td>3</td>
<td>Location</td>
<td>20.9</td>
</tr>
<tr>
<td>Phonenumber</td>
<td>0</td>
<td>5</td>
<td>Location</td>
<td>18.3</td>
</tr>
<tr>
<td>Phonenumber w/slow decay</td>
<td>0</td>
<td>5</td>
<td>Location</td>
<td>17.6</td>
</tr>
</tbody>
</table>

We address the task of normalizing speech from an arbitrary
speaker to the voice of a predefined canonical speaker. As dis-
cussed in Section 2, to make use of Parrottron, we require a
parallel corpus of utterances spanning a variety of speakers and
recording conditions, each mapped to speech from a canonical
speaker. Since it is impractical to have single speaker record
many hours of speech in clean acoustic environment, we use
Google’s Parallel WaveNet-based TTS [31] system to generate
training targets from a large hand-transcribed speech corpus.
Essentially this reduces the task to reproducing any input speech in
the voice of a single-speaker TTS system. Using TTS to generate
this parallel corpus ensures that: (1) the target is always spoken
with a consistent predefined speaker and accent; (2) without any
background noise or disfluencies. (3) Finally, we can synthesize
as much data as necessary to scale to very large corpora.

3. Applications

3.1. Voice normalization

We train the model on a ~30,000 hour training set consisting
of about 24 million English utterances which are anonymized
and manually transcribed, and are representative of Google’s US
English voice search traffic. Using this corpus, we run a TTS
system to generate target utterances in a synthetic female voice.

To evaluate whether Parrottron preserves the linguistic con-
tent of the original input signal after normalization, we report
word error rates (WERs) using a state-of-the-art ASR engine on
the Parrottron output as a measure of speech intelligibility. Note
that the ASR engine is not trained on Griffin-Lim synthesized
speech, a domain mismatch leading to higher WER. Table 1
compares different architecture and loss configurations, evalu-
ated on a hand-transcribed held-out test set of 10K anonymized
utterances sampled from the same distribution as the train set.
The WER on the original speech (matched condition) is 8.3%, which can be viewed as an upper bound. Synthesizing the reference transcripts with a high quality TTS model and transcribing them using our ASR engine obtains a WER of 7.4%.

The top row of Table 1 shows performance using the base model architecture described in Section 2, using a spectrogram decoder employing additive attention [1] without an auxiliary ASR loss. Adding a parallel decoder to predict graphemes leads to a significant improvement, reducing the WER from 27.1% to 19.9%. Extending the additive attention with a location sensitive ASR decoder (obtained from forced alignment to the reference ASR loss. Adding a parallel decoder to predict graphemes leads to a significant improvement, reducing the WER from 27.1% to 19.9%).

### 3.2. Normalization of hearing-impaired speech

Addressing a more challenging accessibility application, we investigate whether the normalization model can be used to to convert atypical speech from a deaf speaker into fluent speech. This could be used to improve the vocal communication of people with such conditions or other speech disorders, or as a front-end to voice-enabled systems.

We focus on one case study of a profoundly deaf subject who was born in Russia to normal-hearing parents, and learned English as a teenager. The subject used Russian phonetic representations of English words and learned to speak them using Russian letters (e.g., cat → k a). Using a live (human in the loop) transcription service and ASR systems for multiple years helped improve their articulation. See [33] for more details.

We experiment with adapting the best model from Section 3.1 using a dataset of 15.4 hours of speech, corresponding to read movie quotes. We use 90% of the data for adaptation (KADPT), and hold out the remainder: 5% (about 45 minutes) for dev and 5% for test (KTEST). This data was challenging; we learned that some prompts were difficult to pronounce by unimpaired but non-native English speakers. The WER using Google’s ASR system on the TTS-synthesized reference transcripts is 14.8%. See the companion website for examples.

### 3.2.1. Experiments

Our first experiment is to test the performance of Google’s state-of-the-art ASR system on KTEST. As shown in Table 4, we find that the ASR system performs very poorly on this speech,
We find that the output of this model was rated as natural as the original speech, but our ASR engine performs even more poorly on the converted speech than the original speech. In other words, Parrotron normalization system trained on standard speech fails completely to normalize this type of speech. We have also manually inspected the output of this Parrotron and found that the model produces speech-like sounds but nonsense words.

Now, we test whether utilizing KADPT would have any impact on Parrotron performance. We first take the fully converged male Parrotron normalization model and conduct multiple fine-tuning experiments using KADPT. With a constant learning rate of 0.1, we (1) adapt all parameters on the fully converged model; (2) adapt all parameters except freezing the spectrogram decoder parameters; (3) freeze both spectrogram decoder and phoneme decoder parameters while fine-tuning only the encoder.

We find that all fine-tuning strategies lead to intelligible and significantly more natural speech. The best fine-tuning strategy was adapting all parameters, which increased the MOS naturalness score by over 1.4 points compared to the original speech, and dramatically reduced the WER from 89.2% to 32.7%. Finetuning strategy (2) obtains 34.1% WER and adapting only encoder parameters (strategy (3)), obtains 38.6% WER.

Note that one advantage of directly converting speech to speech over cascading a finetuned ASR engine with TTS is as follows. Synthesizing the output of an ASR engine may generate speech far from intended, due to unavoidable ASR errors. A speech-to-speech model, however, is likely to produce sounds closer to the original speech. We have seen significant evidence to support this hypothesis, but leave it to future work to quantify.

3.3. Speech separation

Finally, to illustrate that the Parrotron architecture can be used in a variety of speech applications, we evaluate it on a speech separation task of reconstructing the signal from the loudest speaker within a mixture of overlapping speech. We focus on instantaneous mixtures of up to 8 different speakers.

It is important to stress that our intent in this section is not to propose a state of the art separation system, but rather to demonstrate that the proposed architecture may apply to different speech applications. More importantly, in contrast to previous applications which made use of synthetic training targets, we evaluate whether Parrotron is able to generate speech from an open set of speakers, generalizing beyond the training set. Furthermore, unlike state-of-the-art speech separation techniques [34, 35], Parrotron generates the signal from scratch as opposed to using a masking-based filtering approach and is able to rely on an implicit phoneme language model.

We use the same voice-search data described in Section 3.1 to artificially construct instantaneous mixtures of speech signals. For each target utterance in the training data, we randomly select a set of 1 to 7 utterances to mix together as the background noise. The number of background utterances is also randomly selected. Before mixing, we normalize all utterances to have similar gains.

We mix target utterances with the background noise by simply averaging the two signals with a randomly sampled weight $w \in [0.1, 0.5]$ for the background and $1 - w$ for the target utterance. This results in an average SNR across all artificially constructed utterances of 12.15 dB, with a standard deviation of 4.7. 188K utterances from this corpus are held out for testing. While we do not explicitly incorporate reverberation or non-speech noise, the underlying utterances come from a variety of recording environments with their own background noise.

To evaluate whether Parrotron can perform this separation task, we train a model to the best performing architecture as in Section 3.1. We feed as inputs our mixed utterances and train the model to generate corresponding original clean utterances.

We evaluate the impact of this separation model using Google’s ASR system. We compare WERs on three sets of 188K held-out utterances: (1) the original clean speech before adding background speech; (2) the noisy set after mixing background speech; (3) the cleaned output generated by running Parrotron on the noisy set. As shown in Table 5, we observe significant WER reduction after running Parrotron on the noisy set, demonstrating that the model can preserve speech from the target speaker and separate them from other speakers. Parrotron significantly reduces insertions, which correspond to words spoken by background speakers, but suffers from increased deletions, which is likely due to early end-of-utterance prediction.

4. Conclusion

We described Parrotron, an end-to-end speech-to-speech model that converts an input spectrogram directly to another spectrogram, without intermediate symbolic representation. We find that the model can be trained to normalize speech from different speakers into speech of a single target speaker's voice while preserving the linguistic content and projecting away non-linguistic content. We then showed that this model can successfully be adapted to improve WER and naturalness of speech from a deaf speaker. We finally demonstrate that the same model can be trained to successfully identify, separate and reconstruct the loudest speaker in a mixture of overlapping speech, improving ASR performance. The Parrotron system has other potential applications, e.g. improving intelligibility by converting heavily accented or otherwise atypical speech into standard speech. In the future, we plan to test it on other speech disorders, and adopt techniques from [16, 30] to preserve the speaker identity.

5. Acknowledgments

We thank François Beaufays, Michael Brenner, Diamantino Caseiro, Zhi Feng Chen, Mohamed Elfeky, Patrick Nguyen, Bhuvana Ramabhadran, Andrew Rosenberg, Jason Pelecanos, Johan Schalkwyk, Yonghui Wu, and Zelin Wu for useful feedback.

<p>| Table 4: Performance on speech from a deaf speaker. |</p>
<table>
<thead>
<tr>
<th>Model</th>
<th>MOS</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real speech</td>
<td>2.08 ± 0.22</td>
<td>89.2</td>
</tr>
<tr>
<td>Parrotron (male)</td>
<td>2.58 ± 0.20</td>
<td>109.3</td>
</tr>
<tr>
<td>Parrotron (male) fine-tuned</td>
<td>3.52 ± 0.14</td>
<td>32.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
<th>WER</th>
<th>del</th>
<th>ins</th>
<th>sub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original (Clean)</td>
<td>8.8</td>
<td>1.6</td>
<td>1.5</td>
<td>5.8</td>
</tr>
<tr>
<td>Noisy</td>
<td>33.2</td>
<td>3.6</td>
<td>19.1</td>
<td>10.5</td>
</tr>
<tr>
<td>Denoised using Parrotron</td>
<td>17.3</td>
<td>6.7</td>
<td>2.2</td>
<td>8.4</td>
</tr>
</tbody>
</table>
6. References


