The Sequence-to-Sequence Baseline for the Voice Conversion Challenge 2020: Cascading ASR and TTS

Wen-Chin Huang1, Tomoki Hayashi1, Shinji Watanabe2, Tomoki Toda1

1Nagoya University, Japan 2Johns Hopkins University, USA

Abstract

This paper presents the sequence-to-sequence (seq2seq) baseline system for the voice conversion challenge (VCC) 2020. We consider a naive approach to voice conversion (VC), which is to first transcribe the input speech with an automatic speech recognition (ASR) model, followed by the use of transcriptions to generate the voice of the target with a text-to-speech (TTS) model. We revisit this method under a sequence-to-sequence (seq2seq) framework by utilizing ESPnet, an open-source end-to-end speech processing toolkit, and the many well-configured pretrained models provided by the community. Official evaluation results show that our system comes out on top among the participating systems in terms of conversion similarity, demonstrating the promising ability of seq2seq models to convert speaker identity. The implementation is made open source at https://github.com/espnet/espnet/tree/master/egs/vcc20.

Index Terms: voice conversion, voice conversion challenge, espnet, automatic speech recognition, text-to-speech

1. Introduction

Voice conversion (VC) is a technique to transform the para-/non-linguistic characteristics included in a source speech waveform into a different one while preserving linguistic information [1,2]. VC has great potential in the development of various new applications such as speaking aid devices for vocal impairments, expressive speech synthesis, silent speech interfaces, or accent conversion for computer-assisted language learning.

The aim of the voice conversion challenge (VCC) is to better understand different VC techniques built on a freely available common dataset to achieve a common goal and share views about unsolved problems and challenges faced by current VC techniques. The challenges focused on speaker conversion, for which VC models are built to automatically transform the voice identity. In the third version, VCC2020 [3], two new tasks are considered. The first task is semiparallel VC within the same language, in which only a small subset of the training set is parallel with the rest being nonparallel. The second task is cross-lingual VC, in which the training set of the source speaker is different from that uttered by the target speaker in language and content, thus nonparallel in nature. In conversion, the source speaker’s voice in the source language is converted as if it was uttered by the target speaker while keeping linguistic contents unchanged.

It would be worth discussing two important factors when designing a VC system: data and model. First, from the data viewpoint, in either of the VCC2020 tasks, techniques for dealing with nonparallel data should be developed. In the liter-
• The system should be an open-source project made publicly available to benefit potential future researchers.
• The system should serve as a competitive benchmark.

With these requirements in mind, we implemented the system using ESPnet, a well-developed open-source end-to-end (E2E) speech processing toolkit [16, 17], and made as much use of publicly available datasets as possible. Although it is generally believed that a simply cascading system used to perform a certain task is inferior to an end-to-end model, benefitting from recent advances in ASR and TTS models, as well as efforts such as implementation and hyperparameter tuning that are dedicated by the open-source community, we will show that our system is not only easy to use but also serves as a strong competing system in VCC2020.

2. System Overview

A naive approach to VC is a cascade of an ASR model and a TTS model. Although this method is not new, by revisiting this method using seq2seq models, we can model various aspects of prosody such as pitch, duration, and speaking rate, which are usually not well considered in the literature. Conceptually, the ASR model acts similarly to a speaker normalizer that first normalizes the input speech such that attributes of the source speaker are filtered out and only the linguistic content remains. Then, the TTS model functions to add speaker information to the recognition result so that the converted speech sounds similarly to the target speaker.

Our system, as depicted in Figure 1, consists of three modules: a speaker-independent ASR model, a separate speaker-dependent TTS model for each target speaker, and a neural vocoder that synthesizes the final speech waveform.

**ASR model.** ASR models are usually trained with a multi-speaker dataset, thus speaker-independent in nature. For both tasks 1 and 2, the source speech is always in English, so an English transcription is first obtained using an ASR model.

**TTS model.** In the TTS literature, training in a speaker-dependent manner, rather than training speaker-independently is a common practice since the former usually outperform the latter. However, the size of the training set of each target speaker in VCC2020 is too limited for seq2seq TTS learning. In light of this, we employ a pretraining and finetuning scheme that first pretrains on large TTS datasets followed by finetuning on the limited target speaker dataset [18]. This allows us to successfully train on even approximately five minutes of data.

**Neural vocoder.** In recent years, neural waveform generation modules (also known as vocoders) have brought significant improvement to VC. In this work, we used the Parallel WaveGAN (PWG) [19], since it enables high-quality, real-time waveform generation. We adopted an open-source implementation\(^2\) and integrated it with ESPnet.

Our implementation was built on the E2E speech processing toolkit ESPnet [16, 17], which provides various useful utility functions and properly tuned pretrained models.

3. ASR Implementation

3.1. Data

Since the input is always English, we used the Librispeech dataset [20], which contains 960 hours of English speech data from over 2000 speakers.

3.2. Model

The backbone of the ASR model was the Transformer [21–23]. The model was trained in an end-to-end manner using a hybrid CTC/attention loss [24], and a recurrent-neural-network-based language model (RNNLM) was used for decoding. We directly used a pretrained model (including the RNNLM) provided by ESPnet.

4. TTS Implementation

We are faced with a more difficult challenge in implementing the TTS model. In task 2, the input language is different from the languages of the training data. That is, the TTS model should learn the voice of an unseen language. This is sometimes referred to as cross-lingual voice cloning [25]. As there has not been a standard, promising protocol especially when only five minutes of training data is available, we adopt a simple method that constructs x-vector-based [26] bilingual TTS models by pretraining with corpora of English and the target language and finetuning with the target language.

4.1. Data

The target language for task 1 is English. Therefore, for pretraining, we used the multispeaker LibriTTS [30] dataset, which contains around 250 hours of English data from over 2000 speakers. In task 2, the target languages are German, Finnish, and Mandarin. Considering the open-source implementation, we wanted to avoid using commercial or private datasets. Unfortunately, under such constraint, there is not much choice, and the available datasets at the time we developed the system were large but contained only data from a single speaker or a few speakers, as shown in Table 1. Although it has been shown that combining imbalanced multispeaker datasets improves performance [31], this effect remains unknown in the cross-lingual setting. To this end, for the English data, we decided to use not the LibriTTS dataset, which has many speakers yet a small amount of data per speaker, but the M-AILABS dataset [27], which has a large amount of data from only a few speakers. Finally, since the task 2 datasets were of different sampling rates, we downsampled all task 2 data to 16 kHz. As for the x-vector extractor, the Kaldi toolkit was used and the model was pretrained on VoxCeleb [32].

<table>
<thead>
<tr>
<th>Lang.</th>
<th>Dataset</th>
<th>Spkrs</th>
<th>Hours</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eng.</td>
<td>M-AILABS [27]</td>
<td>2</td>
<td>32</td>
<td>phn or char</td>
</tr>
<tr>
<td>Ger.</td>
<td>M-AILABS [27]</td>
<td>5</td>
<td>190</td>
<td>char</td>
</tr>
<tr>
<td>Fin.</td>
<td>CSS10 [28]</td>
<td>1</td>
<td>10</td>
<td>char</td>
</tr>
<tr>
<td>Man.</td>
<td>CSMSC [29]</td>
<td>1</td>
<td>12</td>
<td>pinyin</td>
</tr>
</tbody>
</table>

Figure 2: Illustration of bilingual TTS used in task 2.

\(^2\)https://github.com/kan-bayashi/ParallelWaveGAN

Table 1: TTS training datasets in task 2. "phn" and "char" stand for phoneme and character, respectively.
4.2. Model
We used an x-vector-based [26] multispeaker TTS model [33] with a Transformer backbone [34]. The input was a linguistic representation sequence and the output was the mel filterbank sequence extracted from the (optionally downsampled) waveform. In task 1, since the input is always English, the model simply takes English characters as input.

However, in task 2, it is nontrivial to decide the input representation since it is often language-dependent. For example, there is no overlap in the text representation between Mandarin and English [25]. When we finetune a pretrained model for a Mandarin speaker, since the Mandarin corpus does not contain English words, the model has no clue how the target speaker pronounces English words. This mismatch may cause quality degradation. Below, we describe how we alleviate this issue.

We used a shared input embedding space when training the bilingual TTS model. In neural TTS, the input embedding look-up table is a projection from discrete input symbols to continuous representation and is trained with the rest of the model by backpropagation. It is useful in that the model can implicitly learn how to pronounce each input token, such that different tokens with similar pronunciations can have similar embeddings. The assumption here is that there is an overlap between the input representations of the two languages. For example, if we train a Mandarin/English TTS model, the “ah” phoneme in English and “a” pinyin representation may have similar embeddings. As a result, even if only “a” is seen during training, by learning how the target speaker pronounces such a vowel, the model may still know how to pronounce “ah”.

For the Mandarin/English TTS, we used phonemes and pinyin as the input, whereas for the Finnish/English and German/English TTS, we used characters as the input. In the fine-tuning stage, the parameters are updated using the training utterances of the target speaker, except that the embedding lookup table in Figure 2 is fixed.

5. Neural Vocoder Implementation
The PWG had a non-autoregressive (non-AR) WaveNet-like architecture and was trained by jointly optimizing a multiresolution spectrogram loss and a waveform adversarial loss [19]. The input was a mel filterbank and the output was a raw waveform. For each task, we trained a separate PWG using the training data from all available speakers. That is, data of 8 and 10 speakers were used to train PWGs for tasks 1 and 2, respectively. Notably, in task 2, although mel filterbanks were extracted from a 16kHz waveform as mentioned in Sections 4.1 and 4.2, we still map them to a 24kHz waveform during training, as the quality degradation from such a mismatch has shown to be acceptable [30].

6. Challenge Results
6.1. VCC2020 Dataset
The VCC2020 database had two male and two female English speakers as the source speakers. For task 1, two male and two female English speakers were chosen as target speakers, and one male and one female for each of Finnish, German, and Mandarin in task 2. Each of the source and target speakers has a training set of 70 sentences, which is around 5 minutes of speech data. Note that in task 1, the target and source speakers have 20 parallel sentences, where the remaining 50 sen-
Table 2: Character/word error rates (CER/WER) (%) calculated using a pretrained ASR model. The scores are averaged over all target speakers.

<table>
<thead>
<tr>
<th>Source</th>
<th>Input</th>
<th>Task 1</th>
<th>Task 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CER</td>
<td>WER</td>
<td>CER</td>
</tr>
<tr>
<td>SEM2</td>
<td>2.9</td>
<td>6.5</td>
<td>12.1</td>
</tr>
<tr>
<td>SEM1</td>
<td>0.2</td>
<td>0.9</td>
<td>14.2</td>
</tr>
<tr>
<td>SEM2</td>
<td>2.9</td>
<td>7.5</td>
<td>18.5</td>
</tr>
</tbody>
</table>

Although the official report contained results from Japanese and English listeners, here we only report results of English listeners since the two listener groups share a similar tendency.

7. Analysis on Linguistic Contents

A potential threat of the cascading paradigm is that error in early stages might propagate to downstream models. In our proposed method, the recognition failure in the first ASR stage might harm the linguistic consistency in VC. We examine this phenomenon by measuring the intelligibility with a Transformer-ASR model trained on LibriSpeech, provided in ESPnet.

Table 2 shows the results. First, the error rates of the input source speech were not severe because they are similar to that of the test set of LibriSpeech. However, the scores of the converted speech are much worse, indicating that the imperfect TTS modeling is the main cause of intelligibility degradation. We also observe that the error rates of task 2 are much higher than those of task 1, which is consistent with the results in Section 6.4.

8. Conclusion and Discussion

In this paper, we described the seq2seq baseline system of the VCC2020, including the intuition, system design, training datasets, and results. Built on the E2E, seq2seq framework, our ASR+TTS baseline served as a simple starting point and a benchmark for participants. Subjective evaluation results released by the organizing committee showed that our system is a strong baseline in terms of conversion similarity, confirming the effectiveness of seq2seq modeling. The results also demonstrate the naive yet promising power of combining state-of-the-art ASR and TTS models. Yet, there is still much room for improvement, and below we discuss several possible directions that might be addressed in an advanced version.

Enhance the pretraining data. As stated in Section 4.1, there was not much choice for pretraining data in task 2 under the open-source constraint. Using a multiplicity pretraining dataset as in task 1 might improve the performance. Moreover, using datasets with a higher sampling rate might also improve the quality of the vocoder.

Utilize linguistic knowledge. One principle of E2E learning is to use as less domain-specific knowledge as possible. That is, the system performance is expected to be improved when such knowledge is utilized. For example, as reported in [25], using phoneme inputs can greatly improve multilingual TTS systems, but we were unable to do so in task 2 owing to the unfamiliarity with target languages such as Finnish and German.

Select an advanced multiplexer TTS model. The multi-speaker TTS model [33] we adopted was rather naive, and a more state-of-the-art model similar to that in [35] might improve the performance.

Improve the neural vocoder. We adopted a non-AR neural vocoder for fast generation, but it is generally believed that AR neural vocoders are still superior. Because this is a popular research field, it is expected that real-time neural vocoders maintaining the output quality will soon be developed. Also, fine-tuning the vocoders can further improve the performance, as stated in Section 5.

9. Acknowledgement

This work was supported in part by JST, CREST Grant Number JPMJCR19A3 and JSPS KAKENHI Grant Number 17H06101.

10. References


