Improved Zero-Shot Voice Conversion Using Explicit Conditioning Signals

Shahan Nercessian

iZotope, Inc.

shahan@izotope.com

Abstract

In this paper, we propose a zero-shot voice conversion algorithm adding a number of conditioning signals to explicitly transfer prosody, linguistic content, and dynamics to conversion results. We show that the proposed approach improves overall conversion quality and generalization to out-of-domain samples relative to a baseline implementation of AutoVC, as the inclusion of conditioning signals can help reduce the burden on the model’s encoder to implicitly learn all of the different aspects involved in speech production. An ablation analysis illustrates the effectiveness of the proposed method.

Index Terms: voice conversion, zero-shot learning, autoencoder, conditioning signals

1. Introduction

Voice conversion (VC) is the rendering of an utterance from one speaker to that of another [1, 2]. Most deep learning approaches to VC rely on a vocoder to synthesize speech based on a set of parameters [3–5]. Accordingly, methods train a conversion model to transform parameters extracted from an utterance of a source speaker to that of a target speaker. Approaches are generally differentiated by their training configurations, the vocoders they use, and the specifics of their conversion models.

A particularly desirable class of VC algorithms are many-to-many, zero-shot, non-parallel approaches [5, 6]. These approaches can render speech from any source speaker, can model several target speakers, generate new voices on-the-fly without needing to perform any model adaptation, and can be trained without requiring source and target utterances to contain the same linguistic content. As such, they can most opportunistically learn the conversion task with minimal data engineering.

The two main classes of vocoders that are used for VC are hand-designed vocoders (such as STRAIGHT [7] or WORLD [8]) and neural vocoders (such as WaveNet [9] or WaveRNN [10]). Neural vocoders have grown increasingly popular, as their data-driven design and high model capacity allow for more expressive synthesis [11, 12]. Neural vocoders treat Mel spectrogram coefficients as their “parameters” [13,14], and act as an inverter of this phaseless, compressed spectral representation.

Generative Adversarial Networks (GANs) [4] and Variational Autoencoders (VAEs) [5,15] are popular choices for conversion models. However, GANs are notoriously difficult to train, and VAEs often suffer from overly smoothed results due to the fact that they impose their latent encodings to follow some prior distribution. Recently, a many-to-many, zero-shot, non-parallel VC algorithm called AutoVC was proposed [6]. The authors assert that vanilla autoencoders are sufficient for performing VC, so long as there is a properly constructed bottleneck in the network. The method only needs to be trained on a self-reconstruction loss, and thus, its training procedure is straightforward and well-behaved. An inductive argument proves the sufficiency of their training objective and the distribution matching property of the network. While the results of AutoVC are compelling, it does not always preserve salient characteristics in its converted speech. Some of these conversion errors (particularly in pronunciation) are prevalent even in the demo materials associated with the paper.

In this paper, we propose a many-to-many, zero-shot, non-parallel VC algorithm which adds a number of conditioning signals to improve VC results. We inject this conditioning information in both the encoder and decoder of the network to more explicitly model prosody, linguistic content, and dynamics. One would expect that their inclusion would improve the transfer of said features in the converted result. We show that the proposed method 1) improves overall VC quality, 2) can generalize better to out-of-domain examples, such as singing voice, and 3) enables user-controlled alterations of conversion results. To the best of our knowledge, this is the first work to tackle zero-shot VC and include all of the conditioning signals that we include.

The remainder of this paper is structured as follows: We review the AutoVC algorithm (which serves as a baseline for this work) and introduce our proposed method in Section 2. We evaluate model performance in Section 3, including a full ablation analysis illustrating the incremental effectiveness of the proposed method. Lastly, we draw conclusions in Section 4.

2. Voice Conversion Algorithms

2.1. AutoVC

AutoVC uses an autoencoder architecture as illustrated in Figure 1 (with exception to the dashed sections exclusive to the proposed method). The approach includes: a speaker embedding network $E_s(\cdot)$ which takes as input a Mel spectrogram and generates a single fixed-dimensional speaker embedding; a content encoder $E_c(\cdot)$ which takes as input a Mel spectrogram and speaker embedding from a source utterance and generates a latent encoding; a decoder network $D(\cdot)$ which constructs a converted Mel spectrogram from a latent encoding and target speaker embedding; and a neural vocoder $V(\cdot)$ which synthesizes audio waveforms from converted Mel spectrograms. We focus our attention to the encoder and decoder networks, as the others are pretrained and frozen when training the converter.

The input to the encoder is an 80-band Mel spectrogram $X_1$ computed from a source utterance $x_1$. This is concatenated with a source speaker embedding $S_1 = E_s(X_1')$ at each time step, where $X_1'$ is the Mel spectrogram of a potentially different utterance $x_1'$ from the same source speaker. The encoder consists of a convolutional prenet, comprised of three 1D (temporal) convolutional layers with 512 output channels and kernel size 5, each followed by batch normalization and ReLU activation. This result is passed through two bidirectional LSTM layers with forward and backward cell dimensions of 32, yielding an encoding of dimension 64. This is temporally downsized by 32, yielding the latent encoding $Z_1$.

The decoder begins by upsampling the latent encoding $Z_1$ to its original temporal resolution. Given the Mel spectrogram
$X'_{2}$ of some utterance $x'_{2}$ from the same target speaker as the target utterance $x_{2}$, the speaker embedding $S_{2} = E_{s}(X'_{2})$ is concatenated with the upsampled encoding. These features pass through a convolutional prenet similar to that in the encoder, followed by three LSTM layers with cell dimension 1024. The outputs of the LSTM layer are linearly projected to dimension 80, serving as an initial estimate $\hat{X}_{1 \rightarrow 2}$. Batch normalization and Tanh are applied to the convolutional postnet consisting of five 1D convolutional layers of spectrogram. This initial estimate is refined by means of a construction loss between the original and initially estimated Mel spectrograms. The second term is a reconstruction loss 

$$L_{1} = E[|X_{1} - \hat{X}_{1 \rightarrow 1}|^{2}] + \mu E[|X_{1} - \hat{X}_{1 \rightarrow 2}|^{2}] + \lambda E[|E(X_{1}, S_{1}) - E(\hat{X}_{1 \rightarrow 1}, S_{1})|]$$

The first term is the reconstruction loss between the original and reconstructed Mel spectrograms. The second term is a reconstruction loss between the original and initially estimated Mel spectrograms, which empirically helps model convergence. The third term is a latent regressor loss [16] penalizing differences in encodings between the original and converted Mel spectrograms. In practice, hyperparameters $\mu$ and $\lambda$ can be set to 1 [6].

2.2. Proposed method

Our method, as shown in Figure 1, leverages the AutoVC architecture, but adds conditioning signals that explicitly capture information about prosody, linguistic content, and dynamics.

2.2.1. Pitch

We estimate (log) fundamental frequency using the CREPE pitch detector [17]. The detector, denoted as $E_{p}()$, operates on time domain signals, and outputs a pitch estimate and its associated confidence value at each time frame. Both of these values are fed to the network, as we hope that the network can learn a suitable voiced/unvoiced detection criteria. The pretrained tiny model provided by the authors with Viterbi smoothing is a sufficient configuration of CREPE for our application.

2.2.2. Phonetic posteriorgrams (PPGs)

We use PPGs [2, 3, 18] extracted from a phoneme classifier to capture the linguistic content of an utterance over time. This classifier, denoted as $E_{p}(\cdot)$, passes 40 Mel frequency cepstral coefficients (MFCCs) per frame through two bidirectional LSTMs with 128 units per direction. A final dense layer with softmax activation yields the classifier output, which is compared to ground truth labels using the categorical cross-entropy (CE) during training. We trained the network on the TIMIT dataset [19], using its prescribed training and test sets. The dataset consists of audio and sample level timestamps of phonetic transitions from one of 61 classes (including a silence class). The output of the phoneme classifier is, therefore, a 61-element vector at each time frame. The classification accuracy on the test set is 65%, which is found to be sufficient to act as a speaker-independent representation of linguistic content.

2.2.3. Loudness

We extract dynamics information by measuring signal loudness ($L$) using the computational steps $E_{L}(\cdot)$ as in [20, 21]. We compute an A-weighted power spectrum, which puts greater emphasis on higher frequencies. The result is aggregated across all frequencies and converted to decibels to produce a loudness value at each time step.

2.2.4. Network architecture

We modify the AutoVC architecture by injecting conditioning signals at both the encoder and decoder of the network. We concatenate $F_{1} = E_{f}(x_{1})$, $P_{1} = E_{p}(x_{1})$, and $L_{1} = E_{l}(x_{1})$ with the source speaker embedding $S_{1}$ to create the source conditioning vector $C_{1}$ at each time step. This conditioning information is concatenated with the source Mel spectrogram $X_{1}$ in the encoder. We concatenate the target vectors $F_{2}$, $P_{2}$, and $L_{2}$ with the target speaker embedding $S_{2}$ to create the target conditioning signal $C_{2}$ at each time step. This is concatenated with $Z_{1}$ in the decoder. During training, we set $C_{1} = C_{2}$, directly transferring conditioning information from the source to the target to train the autoencoder. With the inclusion of $C_{1}$, the output of the encoder acts as a residual for timbre and expressivity in speech that is not explicitly factored by the conditioning signals.
We modify (1) as
\[
\mathcal{L} = E[|X_1 - \hat{X}_{1\to1}|^2] + \\
\rho E[|X_1 - \hat{X}_{1\to1}|^2] + \\
\lambda E[|E(X_1, C_1) - E(\hat{X}_{1\to1}, C_1)|_1] 
\] (2)
This generalized objective function reverts to the one used by the original AutoVC algorithm when \( C_1 = S_t \).

The conditioning signals \( F_0, P_0 \), and \( L_0 \) could technically be set arbitrarily during inference. In conventional conversion scenarios, \( P_1 = P_2, L_1 = L_2, T_1 \) are set to target speaker embedding, \( C_1 \) to target phonemes and loudness, respectively, and finally, the proposed system including all conditioning signals. Training configurations are identical except for the inclusion of different conditioning signals to the network. For audio examples, please visit our demo page (available at https://sites.google.com/izotope.com/interspeech2020-audio-demo).

3. Experimental Results

3.1. Experimental Setup

VC networks were trained on the VCTK corpus, consisting of speech from 109 speakers [22]. As in [6], we retain 90% of the data of each speaker for training, and save the remainder as a test set. We resampled all of the data to 16 kHz, and performed a crude voice activity detection to remove non-speech segments. All conversion networks were trained for 120,000 steps with a batch size of 2, using the ADAM optimizer and a learning rate of \( 10^{-3} \). The speaker embedding network, neural vocoder, CREPE pitch detector, and phoneme classifier are all pretrained and frozen during the training of the encoder and decoder.

Our training of AutoVC and the proposed approach differs from that in [6] in a few ways. As the original authors did not provide their speaker embedding network, we make use of a similar open-source speaker embedding pretrained to minimize the Generalized End-to-End Loss [23] (available at https://github.com/CorentinJ/Real-Time-Voice-Cloning). This speaker embedding network generates a 256-dimensional speaker embedding from a 40-band Mel spectrogram using an LSTM architecture and retaining only the output from the final time step. During training, we use an entire utterance for \( x_1 \), whereas \( z_1 \) is a 2 second cut from the same utterance. The notation \( X' \) reflects this, and also serves to indicate that a different Mel spectrogram is used for the speaker embedding network. We use fast Fourier transforms (FFTs) with 50 ms window size and 12.5 ms hop size for generating 80-band Mel spectrograms for the encoder-decoder, and FFTs with 25 ms window and 10 ms hop size for generating 40-band Mel spectrograms for the speaker embedding network. During inference, WaveRNN is used as a neural vocoder \( W(\cdot) \), as it boasts the ability to achieve faster than real-time performance [20].

3.2. Performance Evaluation

We perform an ablation analysis to illustrate the effectiveness of each conditioning signal to the conversion task. Five different conversion networks were trained: a baseline AutoVC system conditioned on speaker embedding. AutoVC conditioned on speaker embedding and either pitch, phonetic content, or loudness, respectively, and finally, the proposed system including all conditioning signals. Training configurations are identical except for the inclusion of different conditioning signals to the network. For audio examples, please visit our demo page (available at https://sites.google.com/izotope.com/interspeech2020-audio-demo).

3.2.1. VCTK Corpus

We perform qualitative and quantitative evaluations for all networks using our VCTK test set. For our quantitative evaluation, we conducted surveys with 15 participants within our organization who have some critical listening experience, and tabulated mean opinion scores (MOS). In the first of two surveys, participants were asked to rate conversion results on a 1-5 scale based on overall conversion quality and transfer of salient speech characteristics. In a second survey, participants were asked to rate conversions solely on speaker similarity between the target and converted utterances.

The results of our quantitative analysis is shown in Table 1. We can observe that the inclusion of any explicit conditioning signal proposed here improves performance relative to the baseline approach. Furthermore, the best performance is achieved by appending all conditioning signals to the network. For a single source of additional conditioning information, pitch features appear to account for the majority of reconstruction gains, followed by loudness and phonetic information.

The results of our qualitative analysis are shown in Figure 2. We observe that again, the use of explicit conditioning signals perceptually improves conversion results relative to the baseline approach. The highest ranking performer is the network which combines all three conditioning signals, as this network tries to explicitly transfer more salient characteristics into the converted speech. There is also an improvement in speaker similarity, which we simply take to mean that including additional conditioning has not disrupted the bottleneck for providing speaker disentanglement. Unequal variances t-tests (with \( p < 0.05 \)) verify that the baseline model is significantly worse than all other methods. Moreover, the method combining all conditioning signals significantly outperforms all methods in

Table 1: Loss function evaluated on VCTK test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.5471</td>
</tr>
<tr>
<td>Pitch</td>
<td>0.3242</td>
</tr>
<tr>
<td>Phoneme</td>
<td>0.3746</td>
</tr>
<tr>
<td>Loudness</td>
<td>0.3521</td>
</tr>
<tr>
<td>Combined</td>
<td>0.3050</td>
</tr>
</tbody>
</table>

Figure 2: Mean opinion scores for various networks.
terms of quality, and all methods except for the pitch conditioned network in terms of speaker similarity.

3.2.2. Generalization to singing voice

We further illustrate the effectiveness of our proposed approach on internally collected singing voice examples (using the networks trained on speech). In this case, we are particularly sensitive to pitch, and examples contain slower rates of phonetic transitions. We quantitatively compare conditioning signals calculated from the source and converted utterances. Specifically, we measure the average difference in fundamental frequency, the average categorical CE between source and converted PPGs, and the average loudness difference. The average result is shown in Table 2. The various contours from a single example are shown in Figure 3. It is observed that the baseline approach is the worst at preserving any of the features, as it relies solely on its encoding to implicitly learn all of the different aspects of speech production. The network that best transfers any single feature is the one that has only included its corresponding conditioning signal (e.g. prosody is best preserved by the network which is additionally conditioned on pitch, etc.). The combined approach preserves all three features just slightly worse than networks tuned to solely transfer any one feature.

Lastly, we show that it is possible to independently modify the values of the added conditioning signals at the decoder and have them roughly reflected in the voice converted result. We perform pitch shifting by adding an arbitrary offset $F_{\Delta_1\rightarrow 2}$ to the source pitch contour, and loudness shifting by adding $L_{\Delta_1\rightarrow 2}$ to the source loudness contour. A comparison of source and converted contours are illustrated in Figure 4, where in one case, the pitch is raised by an interval of a minor third, and in another, the loudness is increased by 10 dB. Listening to the examples, it is noteworthy that modifying the loudness contour does not just change overall level, but appears to alter the intensity of the performance. This opens the possibility for manipulating conversion results by end users with intuitive controls.

Table 2: Comparison of conditioning features between source and converted performances.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta F_0$ (Hz)</th>
<th>CE_PPGs (nats)</th>
<th>$\Delta L$ (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>44.539</td>
<td>0.0732</td>
<td>6.309</td>
</tr>
<tr>
<td>Pitch</td>
<td>5.653</td>
<td>0.0623</td>
<td>5.160</td>
</tr>
<tr>
<td>Phoneme</td>
<td>36.140</td>
<td>0.0455</td>
<td>3.436</td>
</tr>
<tr>
<td>Loudness</td>
<td>39.026</td>
<td>0.0594</td>
<td>1.830</td>
</tr>
<tr>
<td>Combined</td>
<td>7.0719</td>
<td>0.0456</td>
<td>1.935</td>
</tr>
</tbody>
</table>

4. Conclusions

We proposed a new VC algorithm including a number of explicit conditioning signals to improve VC performance. Improvements were verified by both quantitative and qualitative means, with the best results obtained by the network which combined all conditioning signals utilized here. In future work, we will investigate end-to-end training so that losses can be evaluated in the time domain and learning expressive speech controls to further manipulate conversions.

5. Acknowledgements

The author would like to thank François Germain for his invaluable comments while preparing this paper.
6. References


