Voice Style Transfer

William Corsel		Wouter Stokman

May 27, 2021

1 Introduction

In recent years, voice conversion methods have been researched extensively. The main goal of these methods is to extract a person's voice properties from sound data and applying those to another person's speech, thereby making it seem like the audio is being produced by the first person (see Figure 1). These methods can, for example, be useful for privacy and identity protection, or in the creative industry.

Deep learning methods have been gaining popularity in the voice conversion space, with some of the prominent works using Generative Adverserial Networks (GANs) [1] or Conditional Variational Auto Encoder (CVAE) [2] architectures. However, these methods were found to be imperfect, with GANs being difficult the train and CVAEs often over-smoothing the results [3].

In this work, we investigate AutoVC [4], a voice conversion method based on a vanilla autoencoder structure which promises non-parallel, many-to-many, as well as zero-shot conversion. We aim to improve the conversion quality and speed by retraining the AutoVC model with more data compared to the model supplied by the authors, and replacing the standard WaveNet [5] vocoder by a Multi-band MelGAN [6] counterpart. We experiment with various configurations and develop a graphical voice conversion tool utilising the technology to allow for real-time voice conversion[1].

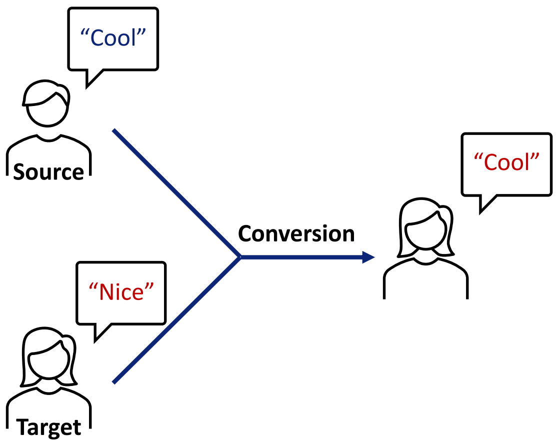

Figure 1: Voice style transfer with a source and a target speaker.

[1]Code, trained models, and audio samples available at https://github.com/Woutah/AutoConvert

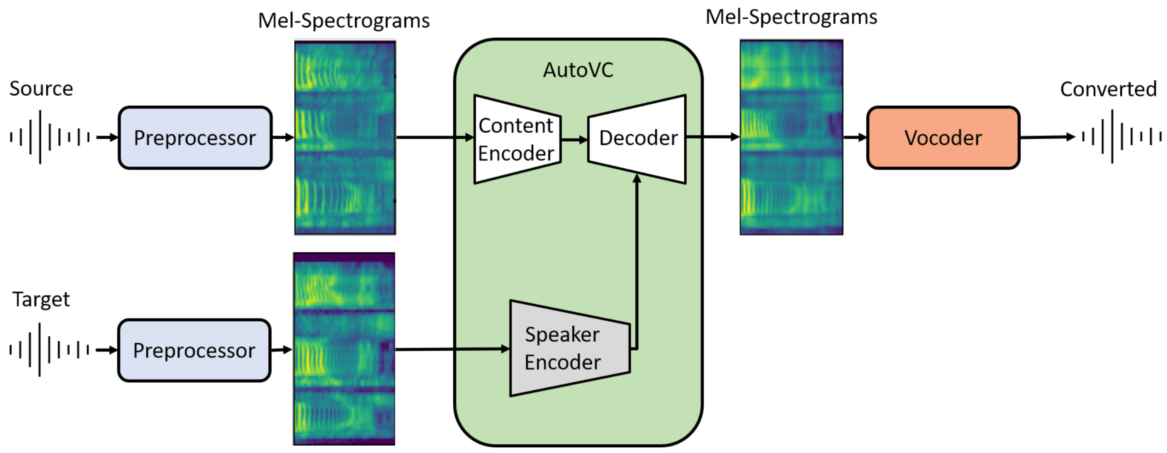

Figure 2: AutoVC framework.

2 AutoVC

AutoVC [4] uses a fairly simple autoencoder framework for voice conversion, as can be seen in Figure 2. It consists of the following components:

- **Preprocessor:** Converts audio data to Mel-Spectrograms using the settings required by the AutoVC framework.
- **Speaker Encoder:** Used to generate an embedding S of the speaker dependent information in the input audio X. When properly trained, this embedding should be the same for any utterance of the same speaker, thus removing any content information. For normal voice conversion, a one-hot encoding of all speakers in the dataset suffices. However, because zero-shot conversion is desirable, AutoVC uses an LSTM-based neural network pretrained on the VoxCeleb1 [7] and Librispeech [8] datasets. This module is not further trained during AutoVC training.
- **Content Encoder:** Used to generate an embedding C of the audio content X, removing any speaker dependent information. An information bottleneck is applied after this module to encourage the network to remove any speaker information from the embedding. This is done by downsampling the output of the content encoder by only passing each 32th value in the vector.
- **Decoder:** Combines the speaker and content embeddings from the previous modules to create a Mel-spectrogram representing the content of the source speaker and the voice style of the target speaker.
- **Vocoder:** Used to recover the audio data from the generated Mel-spectrograms. AutoVC uses a WaveNet vocoder pretrained on the VCTK dataset [9]. This module is not further trained during AutoVC training.

AutoVC trains the content encoder and decoder modules using the self-reconstruction loss and content loss functions by entering the same source speech X through both the speaker and content encoders. The various losses are calculated as follows:

$$L = L_{\text{recon}} + L_{\text{content}} \to \min \tag{1}$$
$$L_{\text{recon}} = \mathbb{E}[||\hat{X} - X||_2^2] \tag{2}$$
$$L_{\text{content}} = \mathbb{E}[||E_c(\hat{X}) - C||_1] \tag{3}$$

with decoder output $\hat{X}$, content embedding C and content encoder E_c. These functions train the content encoder and decoder to reconstruct the input audio. After training, voices can be converted by using separate audio inputs X_1, X_2 for the content and speaker encoders. The difference between the training and conversion procedures can also be seen in Figure 3.

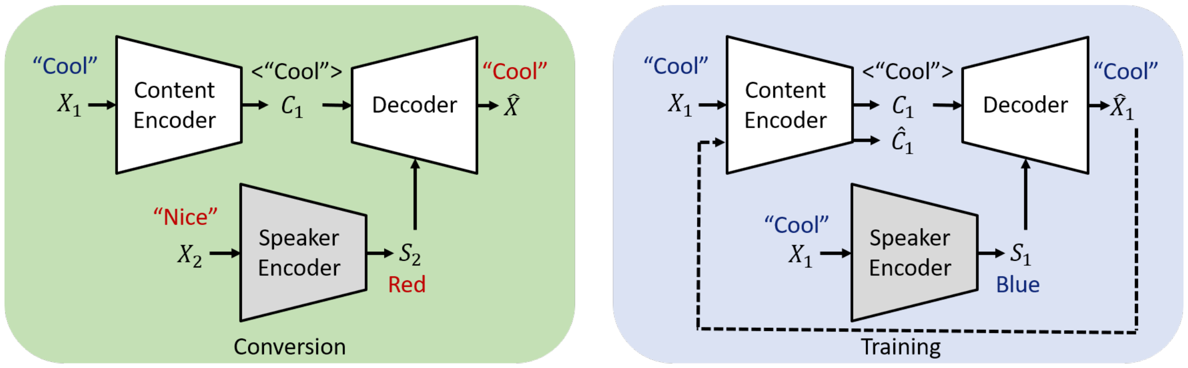

Figure 3: AutoVC conversion and training diagrams.

3 Methods

3.1 AutoVC Configurations

To improve both the quality of generated samples and the conversion speed, we make several alterations to the base AutoVC framework. We define the following models:

1. **AutoVC-WaveNet:** The base AutoVC model as proposed by the authors of [4] using the WaveNet vocoder[2].
2. **AutoVC-Griffin:** The AutoVC framework using a Griffin-Lim vocoder [10]. This algorithm aims to approximate magnitude spectrogram inversion by using alternating forward and inverse Short-Time Fourier Transform (STFT) operations. We set the number of iterations to run to 32.
3. **AutoVC-MelGAN:** The AutoVC framework using a multi-band MelGAN vocoder [6] pretrained on the VCTK corpus[3]. This configuration uses an AutoVC model retrained on the full VCTK dataset. This was done to ensure the AutoVC model Mel-Spectrogram output fit the format required for the MelGAN vocoder. Table 1 shows the difference between the two Mel-Spectrogram configurations.

Table 1: Mel-Spectrogram format of AutoVC-WaveNet and AutoVC-MelGAN models.

Property	**AutoVC-WaveNet**	**AutoVC-MelGAN**
Sample Rate	16000	24000
FFT	1024	2048
Window Length	1024	1200
Hop length	256	300
Channels	80	80

The AutoVC-MelGAN model was retrained on the VCTK corpus [9]. This dataset contains speech data from 110 English speakers with various accents, where each read about 400 sentences. Besides some shared text samples, the dataset uses different samples for each speaker. Before training, the dataset samples were downsampled to fit the sample rate properties stated in Table 1.

[2]AutoVC + WaveNet pretrained models downloadable at https://github.com/auspicious3000/autovc.
[3]MelGAN pretrained models downloadable at https://github.com/kan-bayashi/ParallelWaveGAN.

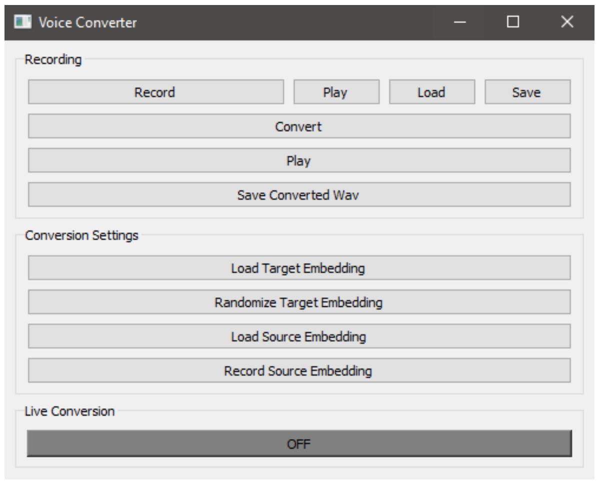

Figure 4: Conversion utility user-interface.

3.2 Conversion Tool

To make our results more accessible, we created a simple-to-use tool with a graphical user interface. This tool allows users to either start the live-conversion process, or to record small samples which can then be converted and saved. The user interface is implemented using PyQt, a python library. An overview of the interface can be seen in Figure 4. It contains the following features:

1. **Recording**
   (a) **Record / Play / Load / Save** - Operations to record/play/load/save a `.wav` file for conversion.
   (b) **Convert** - Start the conversion process using the current `.wav` and settings.
   (c) **Play** - Play the converted `.wav`.
   (d) **Save Converted Wav** - Save the converted audio as a `.wav` file.
2. **Conversion Settings**
   (a) **Load Target/Source** - Load a target/source embedding from a Numpy array of size `[256]`.
   (b) **Randomize Target** - Randomise the target embedding.
   (c) **Record Source Embedding** - Records 8 seconds of audio data which is used to construct a source embedding. This is done by first converting the input to a spectrogram, after which it is converted using the speaker encoder.
3. **Live Conversion**
   (a) **Toggle button** - Switch live conversion on and off. When this is turned on, the microphone will begin to record. After a short delay of several seconds, the continuous live-converted output should start. Real-time conversion has been tested to function correctly using an AMD Ryzen 3800x CPU.

Live conversion is done in steps, as can be seen in Figure 5. First, audio is recorded and put into an audio queue. Next, audio chunks are taken from this queue and converted to spectrograms, which are then used by the AutoVC network to generate a converted sample using the target speaker-encoding. The output spectrograms are put into another queue, which are processed sequentially by the vocoder after which they are played back to the user.

Several steps in this processing sequence require data of a certain time period. For example: the spectrogram conversion by AutoVC normally operates on audio-sequence of several seconds since much of the conversion process is based on context. As such, each queue displayed in Figure 5 buffers the output of the previous step until enough data is collected, at which point the required data is dequeued and processed to be utilised in the next step. Therefore, there is a delay of a couple of seconds before the continuous audio output starts.

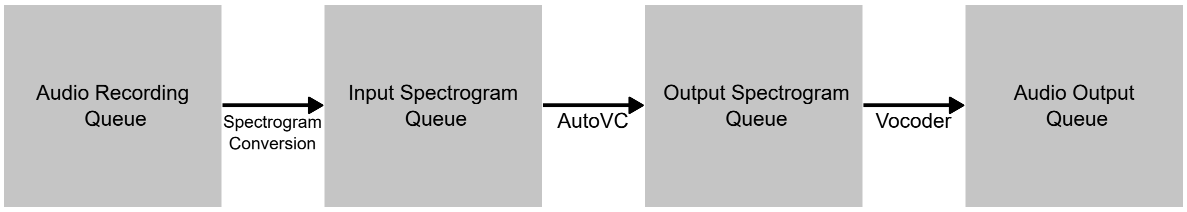

Figure 5: The buffers used for live conversion.

The audio conversion program was written as to allow future work to make use of overlapping conversion-windows, this allows the various queues to reuse previous data in the extraction of context information, which in turn allows for shorter conversion-delays. The used pretrained AutoVC model should be passed as an argument when executing the script. More information can be found in `README.md`, which can be found in our aforementioned GitHub repository.

4 Experiments

To compare the various model configurations described in Section 3.1, we generate output samples to compare their quality. As performing proper comparison tests like the Mean Opinion Score (MOS) test were out of the scope of this project, statements concerning the quality of the generated samples are based solely on the opinion of the authors. We highly recommend the reader to listen to the uploaded samples[4] to judge for themselves. Besides comparing the configurations based on output quality, we also compare the conversion speed. Table 2 contains the conversion times for the different vocoders on a single audio file.

Table 2: Vocoder conversion speed of AutoVC configurations using a 6-seconds-long audio sample.

Configuration	**Conversion time (s)**
AutoVC-WaveNet (original)	1031.1263
AutoVC-Griffin	1.5539
AutoVC-MelGAN	**0.4290**

The conversion quality of AutoVC-WaveNet was found to be high for short samples with seen speakers. However, when this model was tasked with processing longer sequences of data, we noticed the output speech became more and more distorted over time. This behaviour can be explained by the fact the pretrained model uploaded by the authors of [4] was trained on samples with a length of around 2 seconds. This means the model might find it difficult to convert samples of longer length properly. We overcame this limitation by processing the audio data in chunks of around 2 seconds, and stitching them together afterwards. This resulted in higher quality output, even for longer samples. However, as shown in Table 2 the conversion speed of the WaveNet vocoder was slow, even for short audio samples. Furthermore, we noticed that when an unseen target speaker was used, the output became unrecognisable, indicating the network failed to properly separate content from speech.

[4] https://woutah.github.io/AutoConvert/

As seen in Table 2, replacing the WaveNet vocoder by a Griffin-Lim or Multiband MelGAN counterpart, drastically improved conversion speed. We found that AutoVC-MelGAN managed the fastest conversion speed, cutting down processing times of the vocoder by 99.96%. This configuration also vastly improved the audio clarity compared to the Griffin-Lim version.

Our trained model resulted in significantly better output compared to the AutoVC+WaveNet model provided by the original authors, especially when using unseen speakers or samples longer than 2 seconds. However, the voice style transfer quality still varied somewhat in the zero-shot use-case. The performance seemed to be worse especially when using an unseen target speaker. A possible explanation to this phenomenon could be that the content-embedding still contains speaker information, on which the final output is based. To improve the performance, finetuning the bottleneck size could result in a model which generalises better.

5 Conclusion & Future Work

In this work, we investigate the AutoVC framework for voice style transfer. We trained the AutoVC network on the full VCTK dataset using longer training-samples. This resulted in a model which allows the usage of longer conversion samples compared to the model trained on a small subset of the VCTK dataset, which was made available by the original authors. The resulting model performed much better in zero-shot use-cases with an unseen target or source speaker. Furthermore, we trained the AutoVC model using a new preprocessing method, this allowed the model to use a new vocoder, Multiband MelGAN. The newly trained model was able to cut down vocoder processing times by 99.96% compared to the WaveNet vocoder used by the original authors, while performing much better on samples longer than 2 seconds. The resulting model is capable of real-time voice conversion. Lastly, we implemented a simple-to-use GUI which allows users to quickly convert speech recordings and audio files using the selected or randomised target and source embeddings, by utilising the desired conversion models. This tool also allows for real-time continuous voice conversion, using a limited amount of computational resources.

Currently, the AutoVC-MelGAN model uses different Mel-Spectrogram configurations for the speaker and content decoder to be compatible with the pretrained Multiband MelGAN model. In future work, we could eliminate this Mel-Spectrogram mismatch by retraining the speaker encoder or vocoder networks to support a single spectrogram format across the whole framework This might also eliminate the slightly changed voice style issue encountered with the AutoVC-MelGAN model. Furthermore, we could improve our live conversion application to lower latency and reduce audio artifacts. A solution could be to use a sliding-window in each processing step, as this would require less buffering while retaining the context-information needed for the conversion processes. Lastly, the bottleneck of the AutoVC model could be finetuned to allow for even better generalisation and conversion quality. We propose a separate network which will try to classify speakers based on the content embedding, where the loss output to this network will be used to determine whether the bottleneck size should be reduced. A better classification performance would indicate that speaker information is still contained in the content-embedding, which suggests the bottleneck size could be reduced.

References

[1] Ruobai Wang, Yu Ding, Lincheng Li, and Changjie Fan. One-shot voice conversion using star-gan. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7729–7733. IEEE, 2020.

[2] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. Acvae-vc: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder. *arXiv preprint arXiv:1808.05092*, 2018.

[3] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. Stargan-vc: non-parallel

many-to-many voice conversion using star generative adversarial networks. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 266–273, 2018. doi:10.1109/SLT.2018.8639535.

[4] Kaizhi Qian, Yang Zhang, Shiyu Chang, Xuesong Yang, and Mark Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *International Conference on Machine Learning*, pages 5210–5219. PMLR, 2019.

[5] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: Agenerative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[6] Geng Yang, Shan Yang, Kai Liu, Peng Fang, Wei Chen, and Lei Xie. Multi-band melgan: Faster waveform generation for high-quality text-to-speech, 2020. arXiv:2005.05106.

[7] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Utterance-level aggregation for speaker recognition in the wild. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5791–5795. IEEE, 2019.

[8] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[9] Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. 2017.

[10] Nathanaël Perraudin, Peter Balazs, and Peter L Søndergaard. A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE, 2013.