Cycle-Consistent Auto-Encoder For Few-Shot Voice Conversion

Nathan Godey
nathan.godey@eleves.enpc.fr

David Boudin
david.boudin@eleves.enpc.fr

Abstract

Voice Conversion (VC) is the task that consists in using a speech from one person to generate the same speech as said by another person. Among the methods used to perform Voice Conversion, we can distinguish adversarial networks, and auto-encoders. The first are efficient but hard to train, while the latter seem less adequate for the task at first but can be trained quickly.
We propose a method that combines ideas taken from both categories of models, based on the AutoVC [1] auto-encoder model and trained using cycle-consistency loss coming from the CycleGAN [2, 3] adversarial model. This new idea aims at tackling limitations from both types of models, while taking advantage of the efficiency of their approaches.

1. Introduction

An important task in the speech processing field is the one that consists in transforming the style of a speech in terms of the identity of the speaker. For instance, one would like to be able to take a recording of a speech from Donald J. Trump and make Greta Thunberg say it, or vice versa. This specific task is called Voice Conversion (or VC), and is still a great research topic among specialists up to this day.

One of the main question it poses is the one of speech disentanglement, i.e. how can one separate the content information (text, rhythm, emotion, ...) from the style information (pitch, timber, accent, ...) in a speech audio signal. Seeking to achieve speech disentanglement leads to introducing auto-encoders in various forms, as it is done in the works of Kaizhi Qian et al. [1, 4]. Auto-encoders will allow to extract information efficiently in a speech, and then to reconstruct a speech by changing one or several extracted aspects of the speech.

Another approach to this problem is to tackle it as a style-transfer problem and to take inspiration from what is done in Computer Vision, namely to use GANs. For image style-transfer, state-of-the-art methods as CycleGAN [2] or StarGAN [5] have shown impressive results, and have been adapted into Voice Conversion equivalents such as CycleGAN-VC [3] and StarGAN-VC [6].

Both approaches have limitations: GANs are well-known for the complexity of their training and the slow convergence it implies, while auto-encoders seem to be less efficient than GANs at solving the task of Voice Conversion, mostly because they don't use a consistent training for this specific task, but rather specialize in pure speech disentangling.

A many-to-many unparalleled voice converter will provide a conversion for any pair of speakers, even when trained with audio utterances that don't match between different speakers. This means that for such a converter, we don't have a ground truth audio for what our conversion result should sound like. Therefore, training a model on such a voice conversion task takes a trick, whether it is making sure that the disentanglement works perfectly for self-reconstruction in the case of auto-encoders [1], or encouraging a back-and-forth consitency of our conversion model (converting back a converted speech will yield our origin sample) as it is done for GAN-based approaches.

Our approach consists in taking advantage of the efficient training offered by auto-encoders such as AutoVC, and enforcing the cycle-consistency of our model during the training, as it was proven to be efficient in the case of GANs.

2. Problem Definition

We consider a dataset of speech utterances $\mathcal{D} = (\mathcal{X}_{K,i})_{K\in[1,S]}$, where S denotes the number of different speakers. We are in a context of few-shot learning, which means the number of utterances for each speaker is rather low (less than 20). For two speakers $A, B \in [1, S]^2$, we want to define a converter $\mathcal{C}$ such that $\mathcal{C}(\mathcal{X}_{A,i})$ is an utterance conveying the same content as $\mathcal{X}_{A,i}$ but using the style of speaker B. This raises a few questions :

- How to represent an utterance $\mathcal{X}_{K,i}$ efficiently ?
- What do "content information" and "speaker style" mean ?
- What properties can we expect from the converter $\mathcal{C}$?

2.1. Representing Audios

The audio utterances in our dataset $\mathcal{D}$ are initially represented in the time-domain as WAV files. The time is discretized according to a sampling frequency (e.g. 16kHz), and each of the time segments contain bytes representing the signal in time-domain according to a certain byte rate (e.g. 256 bytes/s).

These time-domain representation allow high quality rendering of audio signals, but they don't make it easy for a model to identify specific patterns in an audio file, mostly because the dimensionality of the signal is too large to process.

This is why it is usual to compute Short-Time Fourier Transforms, or spectrograms. Spectrograms are matrices representing Fourier Transforms for several slices of time. The value of coordinate i, j of a spectrogram is the j-th coefficient of the Fourier Transform of the slice of the audio input taken between $i\Delta t$ and $(i+1)\Delta t$.

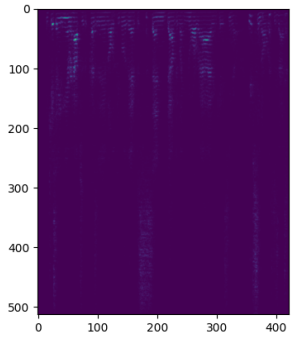


Figure 1: Example of a STFT spectrogram (mel-basis of size 80)

This kind of audio representations present two main flaws : they are mostly sparse, and they require a lots of coefficients of the Fourier Transform to ensure good quality. We also need to take into account that we have lost the phase information in the process.

To solve the first issue, we can introduce a denser and more compact representation : mel-spectrograms. The mel-basis is constructed using a set of triangular filters that help represent human-audible sounds more precisely. One can pick the desired number of filters to use in a mel basis, which allows to control the dimensionality of the mel-spectrogram representation

Applying these triangular filters along with a high-pass filter, thresholding and log-scaling yields our final mel-spectrogram.

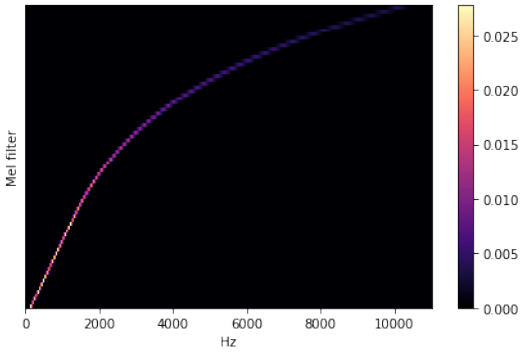


Figure 2: The Mel filter bank

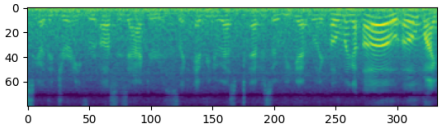


Figure 3: An example of mel-spectrogram

2.2. Speech Disentanglement

A speech signal can be decomposed according to different factors, based on characteristics of the human voice. Among those can be distinguished :

- timbre or style : type of sound produced by the voice. Makes the voice of someone recognizable
- pitch : whether a sound is acute or deep. Can vary along a speech depending on the prosody
- rhythm : can also vary along the speech, or depending on the emotion conveyed
- text content : text information conveyed in the speech, including silences

To disentangle a speech is to be able to separate each of these components inside a natural audio signal. This task is highly related to Voice Conversion because one would like to be able to modify one of these characteristics alone, in order to exchange properties between different speakers. For our study, we only need to separate the style characteristic from the rest of the signal. We will call *content* the infor-

mation represented by the pitch, rhythm and text content of a speech

2.3. Voice Conversion

This section aims at trying to give a definition for an ideal voice converter $\mathcal{C}$. By reducing a speech signal to the four properties presented above, we can suppose that we have a perfect content extractor $\mathcal{E}_c$ such that for any given utterance $\mathcal{X}_{K,i}$ of speaker K:

$$\mathcal{E}_c(\mathcal{X}_{K,i}) = (txtCont(\mathcal{X}_{K,i}), pit(\mathcal{X}_{K,i}), rhy(\mathcal{X}_{K,i}))$$

Let's also consider a perfect style extractor $\mathcal{E}_{sty}$ such that :

$$\mathcal{E}_{sty}(K) = (timb(\mathcal{X}_{K,i}))$$

$\mathcal{E}_{sty}(K)$ does not depend on the considered utterance because the timber of a speaker should be invariant.

Finally, we need to define a perfect recomposition function $\mathcal{R}$ defined as :

$$\mathcal{R}(txtCont(\mathcal{X}_{K,i}), pit(\mathcal{X}_{K,i}), rhy(\mathcal{X}_{K,i}), timb(\mathcal{X}_{K,i})) = \mathcal{X}_{K,i}$$

Given two speakers A and B, the perfect voice converter $\mathcal{C}$ will perform the following task :

$$\mathcal{C}(\mathcal{X}_{A,i}, B) = \mathcal{R}(\mathcal{E}_c(\mathcal{X}_{A,i}), \mathcal{E}_{sty}(B))$$

In words, it will extract the content, pitch and rhythm of an utterance of speaker A, the style of a speaker B, and recombine the four disentangled properties into an artificial utterance.

3. Related Work

3.1. GANs

GANs [7] (generative adversarial networks) are algorithms that were invented in 2014 by Goodfellow et al. GANs are mostly known for the generation of very realistic images, however they can also be used to create sound samples.

A GAN consists of two neural networks; the first called the generator G produces realistic samples whereas the second, called the discriminator, detects if a sample is real or if it was generated by G. The two neural networks are trained using a zero-sum game where the loss of the first network (generator) is the opposite of the loss of the second (discriminator).

3.1.1 CycleGAN-VC

In 2017, Jun-Yan Zhu et al. proposed a method called CycleGAN-VC [2], to train a model for the voice conversion task without using parallel data.

A speech dataset has parallel data when the utterances of the different speakers are made of the same sentences. In other words, $\mathcal{X}_{A,i}$ and $\mathcal{X}_{B,i}$ contain the same sentence read by either speaker A or speaker B. Having parallel data makes it easier to train a model for voice conversion because the expected output of the converter already exists and the loss can simply be:

$$\mathcal{L} = ||\mathcal{C}(\mathcal{X}_{A,i}, B) - \mathcal{X}_{B,i}||$$

The downside of models based on parallel data is that it is difficult to get audio samples of multiple speakers saying the same sentence with the same rhythm and pitch, whereas speech datasets with unparallel data are easy to produce since you can use any utterance from different speakers without having them matched.

During the training of CycleGAN, the loss is composed of the adversarial loss and the cycle-consistency loss. The adversarial loss is the usual loss of GAN. The cycle-consistency loss is:

$$\mathcal{L}_{cycle} = ||\mathcal{C}(\mathcal{C}(\mathcal{X}_{A,i}, B), A) - \mathcal{X}_{A,i}||$$

The adversarial loss ensures that $\mathcal{C}(\mathcal{X}_{A,i}), B)$ is a realistic speech from B. The cycle-consistency loss ensures that $\mathcal{C}(\mathcal{C}(\mathcal{X}_{A,i}, B), A) = \mathcal{X}_{A,i}$, which means using the converter from A to B, and then from B to A, you get the initial speech. This ensures that the content was preserved during the conversion.

CycleGAN only performs one-to-one mapping, i.e. it only learns to perform voice transfer from one speaker to one target. And it needs to train two GANs in parallel to achieve cycle-consistency: one that learns to transfer from A to B and the other from B to A.

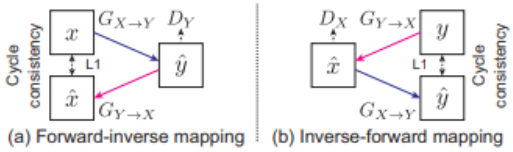


Figure 4: Training procedure of CycleGAN

3.1.2 StarGAN-VC

StarGAN-VC is a method for voice conversion presented by Hirokazu Kameoka et al. in 2018[6], that uses the same structure as CycleGAN, but add a speaker one-hot embedding for the input of the encoder and a speaker classifier that uses the output of the generator and tries to identify to which speaker it belongs to. This classifier add a third loss during training, which allows the model to manage many-to-many voice transfer.

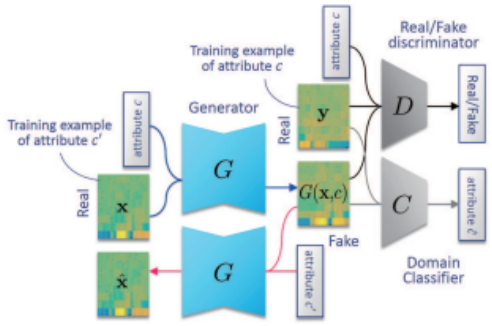


Fig. 2. Concept of StarGAN training.

Figure 5: Training procedure of StarGAN

3.2. GE2E loss

The GE2E loss (generalized end-to-end loss) was published in 2018 by Wan et al.[8]. This loss was created for speaker verification model. Speaker verification is the process of verifying whether an utterance belongs to a specific speaker, based on the speaker utterances. This loss can be used to train a model to produce speaker embeddings.

3.3. AutoVC

In 2019, AutoVC, a method of voice conversion using auto-encoders was published by Kaizhi Qian et al.[1]. AutoVC is trained on unparallel data. The benefit of auto-encoders compared to GANs is that they are easier to train and that the model has mathematical proof of convergence. Moreover, AutoVC supposedly manages to perform few-shot voice conversion and zero-shot voice conversion. A successful few-shot learning model will converge when trained with a few utterances for each speaker, while zero-shot means that the model performs satisfyingly when converting between speakers it has never seen during training.

AutoVC takes a mel-spectrogram as an in input. The idea is that it splits the encoding of the mel-spectrogram in two. One encoder (E_s) handles the style encoding and another one (E_c) handles the encoding of content, rhythm and pitch. The decoder D takes the output of both the encoders as an input and tries to reconstruct a mel-spectrogram from the disentangled properties. Formally, to transfer the voice of B to $\mathcal{X}_{A,i}$, the model computes $\mathcal{C}(\mathcal{X}_{A,i}, B) = D(E_s(B), E_c(\mathcal{X}_{A,i}))$.

The style encoder is trained using the GE2E loss before the training of the rest of the structure, so that we're sure that E_s will encode the style of the speech. Then, E_c and D are trained using two losses, a self-reconstruction loss and a content loss:

$$\mathcal{L}_{recon} = ||\mathcal{C}(\mathcal{X}_{A,i}, A) - \mathcal{X}_{A,i}||$$

$$\mathcal{L}_{cont} = ||E_c(\mathcal{C}(\mathcal{X}_{A,i}, A)) - E_c(\mathcal{X}_{A,i})||$$

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda_{cont}\mathcal{L}_{cont}$$

Where λ_{cont} is a hyperparameter to fix.

At first glance, it seems that AutoVC only learns how to reconstruct $\mathcal{X}_{A,i}$, but not how to generate the converted utterance $\mathcal{C}(\mathcal{X}_{A,i}, B)$. The idea is that if the output dimension of E_c is small enough, $E_c(\mathcal{X}_{A,i})$ will only contain information about the content of $\mathcal{X}_{A,i}$. So learning how to reconstruct $\mathcal{X}_{A,i}$ from $(E_s(A), E_c(\mathcal{X}_{A,i}))$, is the same as learning how to generate $\mathcal{C}(\mathcal{X}_{B,i}, A)$, from $(E_s(A), E_c(\mathcal{X}_{B,i}))$.

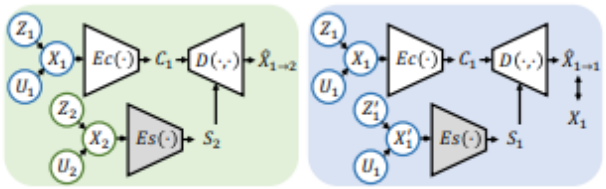


Figure 6: Training structure of AutoVC

4. Methodology

4.1. Datasets

A very commonly used dataset in speech processing is the Voice-Cloning ToolKit from CSTR [9]. It is made of 110 speakers providing around 400 utterances each. This dataset is the one that was used to train AutoVC [1], and is made of very clean audio files, with noise-free recordings spoken out clearly.

Another well-known dataset is VoxCeleb [10], made of over 7000 different speakers and 1 million utterances. The quality of the utterances is lesser than VCTK and resembles more to real data as one could gather using basic recording settings.

We decided to go for small samples of the VCTK dataset, and of VoxCeleb, to check AutoVC and our cycle-consistent model under different conditions. Since the models should be trainable in the context of few-shot learning, we selected around 5 speakers in each dataset, and we added utterances of our targets for voice conversion, namely Greta Thunberg and Donald J. Trump. Each speaker is represented by between 3 and 15 utterances.

We also check the regularity of the model when training on a bigger dataset, by choosing around 50 speakers VoxCeleb and adding once again utterances from Donald Trump and Greta Thunberg.

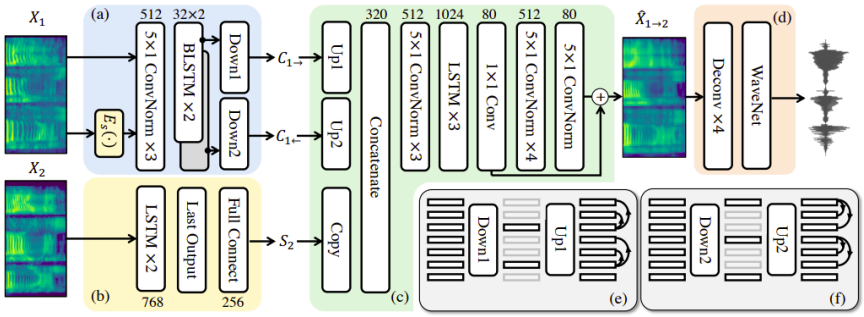


Figure 7: Architecture of AutoVC, the main blocks are the BLSTMs in the content encoder(a), and the LSTMs in the style encoder (b) and the decoder (c). Note that to convert the mel-spectrogram back to an audio file (d), you have to go through another neural network for phase reconstruction which is very complex, and the conversion lasts more than 20 minutes using GPUs.

4.2. Study of AutoVC

As mentioned earlier, a crucial parameter for the AutoVC model is the dimension of the bottleneck. We tried to check the hypothesis made by the authors of [1] :

- if the bottleneck is too narrow, the content information cannot be transmitted through the encoder, and the decoder cannot reconstruct a proper speech
- if the bottleneck is too wide, the content embedding produced by the encoder contains information about the style of the speaker, which leads to poor disentanglement

Therefore, tuning the bottleneck requires one to find a trade-off between poor disentanglement and poor reconstruction.

4.3. Introducing Cycle-Consistency

Looking at vanilla AutoVC, one observation that can be made quickly is that the model is not trained for Voice Conversion. Nevertheless, looking at the results presented by the researchers and at those we were able to reproduce, the architecture of the AutoVC model seems to be able to perform this task, in some specific conditions.

Our idea was to apply the principle of cycle-consistent style-transfer developed with GAN-based models to train the AutoVC auto-encoder.

Considering two speakers A and B, we extract an utterance $\mathcal{X}_{A,i}$ and we compute the cycle-consistency loss $\mathcal{L}_{cycle}$:

$$\mathcal{L}_{cycle}(\mathcal{X}_{A,i}, B) = ||\mathcal{C}(\mathcal{C}(\mathcal{X}_{A,i}, B), A) - \mathcal{X}_{A,i}||_2$$

$\mathcal{L}_{cycle}$ asserts that voice conversion from A to B and from B back to A yields the original utterance. This loss also encourages the model to convey all the information throughout the conversion.

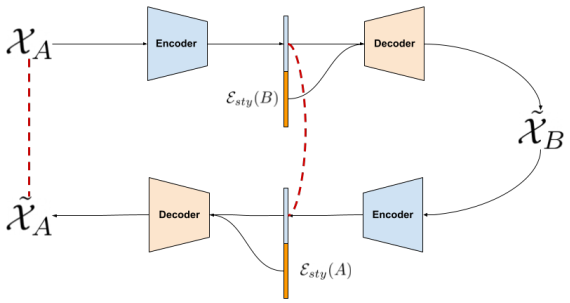


Figure 8: Training AutoVC using cycle-consistency loss

At optimal conversion, we would also like the encoder to provide the same content encoding for two utterances corresponding to the same content. This means that encoding $\mathcal{C}(\mathcal{X}_{A,i}, B)$ and $\mathcal{X}_{A,i}$ should yield the same result. We add the content encoding loss $\mathcal{L}_{encod}$ to encourage such a behaviour:

$$\mathcal{L}_{encod}(\mathcal{X}_{A,i}, B) = ||\mathcal{E}_{cont}(\mathcal{C}(\mathcal{X}_{A,i}, B)) - \mathcal{E}_{cont}(\mathcal{X}_{A,i})||_1$$

Now we are sure that optimizing these losses will encourage full-information conveying throughout the model. However, we don't make sure yet that the result of the converter $\mathcal{C}$ actually has the style of the targeted speaker. To do this, we tried adding a loss $\mathcal{L}_{style}$ defined as :

$$\mathcal{L}_{style}(\mathcal{X}_{A,i}, B) = ||\mathcal{E}_{sty}(\mathcal{C}(\mathcal{X}_{A,i}, B)) - \mathcal{E}_{sty}(B)||_1$$

All in all, our loss function $\mathcal{L}$ can be written:

$$\mathcal{L} = \mathcal{L}_{cycle} + \mathcal{L}_{style} + \lambda_{encod}\mathcal{L}_{encod}$$

4.4. Fine-Tuning

Training AutoVC using our cycle-consistency loss requires much more time than in the vanilla setting, mostly

because one iteration now corresponds to more feed-forward operations through the converter and through the speaker-style encoder. Since the vanilla AutoVC training seemed to provide results close to acceptable, we thought that dividing the training into two phases could help reach an optimum faster:

1. Train AutoVC on the self-reconstruction task only and stop before convergence

2. Fine-tune AutoVC on the conversion task using the cycle-consistent loss

5. Evaluation

We trained our models using Google Colab GPUs and the Pytorch framework. Using GPUs was necessary because training lasted multiple hours, with an iteration time varying from hundreds of milliseconds to 2 seconds. The voice conversion of one utterance lasts 20 minutes overall, because a phase reconstruction algorithm needs to be applied to recover an audio signal from a mel-spectrogram. Using Google Colab led to inconveniences because the duration use of the GPUs is limited and we could only train a model once or twice a day. The tuning of the hyperparameters of our models took us a long time. We trained both models using VCTK and VoxCeleb.

5.1. AutoVC

On the VCTK dataset, we were able to verify the hypotheses presented in the AutoVC article [1], and to find a trade-off for the bottleneck dimension. However, on the VoxCeleb dataset, we observed the expected flaws when the bottleneck was too narrow, and when it was too wide, but the result obtained when seeking for the trade-off was not satisfying.

At optimal bottleneck, the AutoVC model will provide a disentanglement for self-reconstruction that should also work when converting an utterance to a different target. Nevertheless, if the AutoVC model needs a large bottleneck to perform satisfying self-reconstruction ($A \longrightarrow A$), its bottleneck can already be wide enough to allow specialization in self-reconstruction too. This can happen because AutoVC does not *enforce* perfect disentanglement, but thrives on the fact that an optimal bottleneck will just convey the complimentary information needed apart from the speaker-style embedding to reconstruct an utterance, namely the content of the speech.

5.2. CycleAutoVC

CycleAutoVC was trained on 90000 steps for different settings of batch size, neck dimension and on different datasets. From an audio perpective, the best result was achieved with a neck dimension of 64 and a batch size of

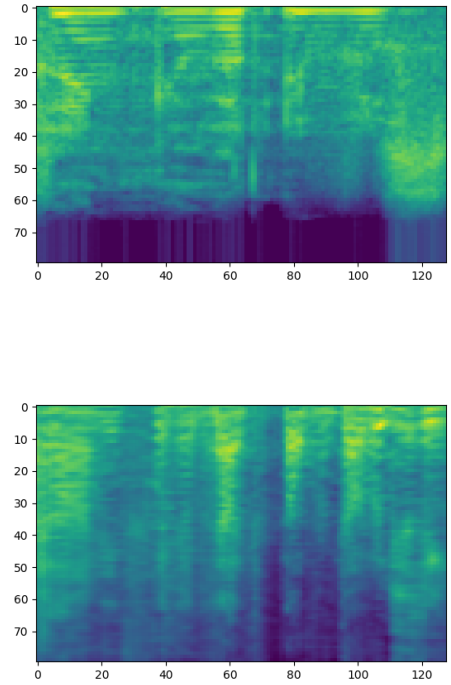


Figure 9: Above: mel-spectrogram of a speech of a person A $\mathcal{X}_{A,i}$. Below: mel-spectrogram of the result of a voice conversion from $\mathcal{X}_{A,i}$ to a speaker B using Autovc. We can see that the rhythm of the speech was kept because vertical blue line are at the same place. The timber is not same because yellow stripe are not distributed in the same way. We can see that the outline of the stripe is clearer on the real speech, this may be why the output of AutoVC is more crackled.

8. We also found that fine-tuning the vanilla AutoVC did not perform as well as starting our training from scratch. The training process that led to the best results was actually quite specific :

1. We trained CycleAutoVC without using the speaker-style loss for 60000 steps, where our model reached convergence. This training was done on a large dataset to avoid any overfitting of the self-conversion task

2. We added the speaker-style loss to the total loss computation and we trained our model for 30000 more steps. This training was done on the reduced dataset to assess the model's performance on a small number of speakers.

The results in mel-spectrogram can be seen on Figure 11.

With the best model, when the voice conversion from Trump to Greta is done, we recognize the voice of Greta

but there are lots of cracklings. The rhythm and the pitch are the same as in the original, although the text content is hardly recognizable. The model struggles to render the text-content properly.

5.2.1 Effect of the style loss $\mathcal{L}_{sty}$

Using the style loss $\mathcal{L}_{sty}$ allows the converted utterances to sound more like human voices. On the other hand, the results have poorer content rendering. Moreover, it seems that we cannot consider that the speaker-style encoder based on the GE2E loss is perfect, which induces a bias when comparing the embedding of a generated audio sample and a real utterance.

5.2.2 Effect of the content-encoder loss $\mathcal{L}_{cont}$ and the coefficient λ_{cont}

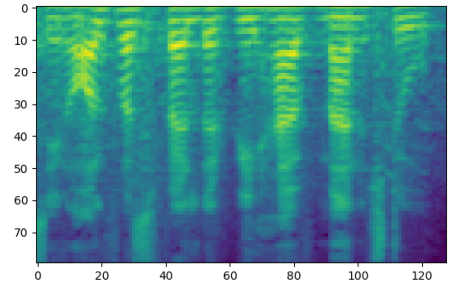
As we tried to train the cycle-consistent AutoVC using hyperparamters tuned for vanilla AutoVC, we saw that using a small bottleneck yielded utterances that focused too much on the content. To solve this issue, we tried to reduce the parameter λ_{cont} to improve the quality of the results. This slightly improved our results for small bottlenecks, but we found out that for optimal CycleAutoVC bottlenecks, this encouraged rendering gibberish as if it were said by the target speaker, so that the speaker-style loss would be low.

5.3. Comparison / Benefits of our approach / Limits of our approach

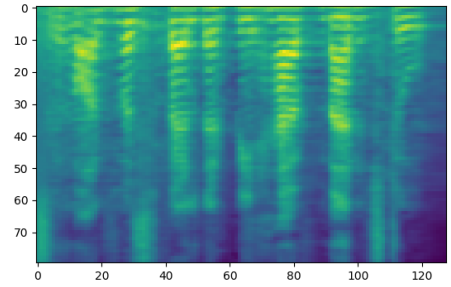
The performances of our method and the one of AutoVC are very similar. Both manage to transfer the style and to reconstruct the rhythm and the pitch. Reconstructing the text content seems to be the main challenge of both methods, which is unfortunate because it is the most important feature to deceive the human ear.

Our method appears to be performing better when the neck dimension of the content encoder is high. If the dataset is voluminous, the encoder neck needs to be large to be able to encode as much content as possible. In this case, it would be better to use our model. This means that our model is also less prone to overfitting a set of utterances, and that training auto-encoders for the cycle-consistency loss might be a good lead to improve voice conversion systems.

The downside of our model is that it takes longer to train because it needs to compute a double conversion which creates a very deep backward loss, and doubles the number of operations needed to complete one iteration (one pass from A to B and another one from B to A). Moreover, our model is much more reliant on the quality of the speaker-style encoding system over one given utterance, and its robustness to poor quality samples as we generate during training. Finally, it is rather unstable in the training as it can easily tend to generate gibberish with a good style match, or to render the content of the speech without caring about the style.



(a) Real utterance $\mathcal{X}_{A,i}$ as a mel-spectogram



(b) Self-converted utterance $\mathcal{C}(\mathcal{X}_{A,i}, A)$

Figure 10: Self-reconstruction performance for AutoVC with a wide bottleneck (dimension 64). We see a very good quality both in the mel-spectrogram and in the produced audio.

6. Conclusion

We explored a new approach to train an auto-encoder for voice conversion, replacing the classical auto-encoder loss with a more adequate loss taking cycle-consistency into account. This method seems to be more efficient in the case of large datasets. The next step would be to perform zero-shot conversion using our CycleAutoVC, yet we think that the results we have for few-shots conversion are not good enough to be able to have decent results on zero-shot setting.

Another way to improve our results would be to adapt the architecture of AutoVC, by removing the bias in its Convolutional layers for instance, so that silence is converted into silence, or by introducing attention mechanisms in its BiLSTM layers.

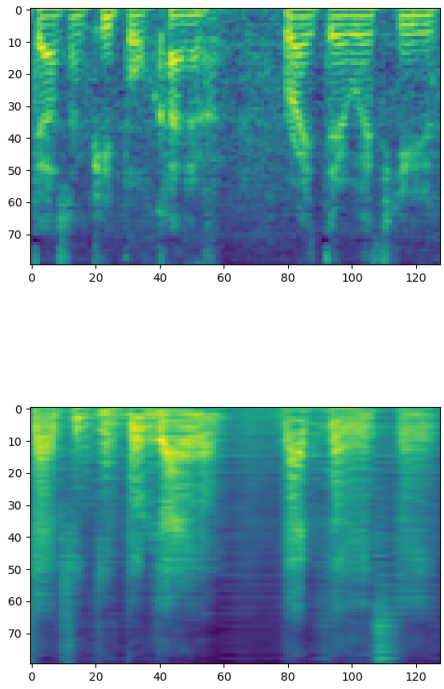


Figure 11: Above: Mel-spectrogram of a speech of a person A $\mathcal{X}_{A,i}$. Below: Below: Mel-spectrogram of the result of a voice conversion from $\mathcal{X}_{A,i}$ to a speaker B using CycleAutoVC.

References

[1] Kaizhi Qian et al. "AUTOVC: Zero-Shot Voice Style Transfer with Only Autoencoder Loss". In: (2019). arXiv: 1905.05879 [eess.AS].

[2] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: (2020). arXiv: 1703.10593 [cs.CV].

[3] Takuhiro Kaneko et al. "CycleGAN-VC3: Examining and Improving CycleGAN-VCs for Mel-spectrogram Conversion". In: (2020). arXiv: 2010.11672 [cs.SD].

[4] Kaizhi Qian et al. "Unsupervised Speech Decomposition via Triple Information Bottleneck". In: (2020). arXiv: 2004.11284 [eess.AS].

[5] Yunjey Choi et al. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". In: (2018). arXiv: 1711.09020 [cs.CV].

[6] Hirokazu Kameoka et al. *StarGAN-VC: Non-parallel many-to-many voice conversion with star generative adversarial networks*. 2018. arXiv: 1806.02169 [cs.SD].

[7] Jun-Yan Zhu et al. "Generative Adversarial Networks". In: (2014). arXiv: 1406.2661.

[8] Quan Wang Li Wan, Alan Papir, and Ignacio Lopez Moreno. "Generalized end-to-end loss for speaker verification". In: (2018). eprint: arXiv:1710.10467.

[9] Junichi Yamagishi, Christophe Veaux, and Kirsten MacDonald. "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92)". In: (2019). DOI: 10.7488/ds/2645.

[10] A. Nagrani, J. S. Chung, and A. Zisserman. "VoxCeleb: a large-scale speaker identification dataset". In: *INTERSPEECH*. 2017.