Non-Parallel Many-to-Many Cross-Lingual Voice Conversion Project Report

Michael Vobejda
Department of Computer Science
Stanford University
549 Lasuen Mall, Stanford, CA 94305
`mvobejda@stanford.edu`

Abstract

Voice conversion (VC) is the task of modifying one speaker's words so that they appears to have been uttered by different speaker. In its final converted form, known as the resulting voice, the speech signal from the first speaker, known as the source speaker, should retain its linguistic content, but it should also be maximally altered in terms of vocal timbre, range, inflection, et cetera in order to match the voice of the second speaker, known as the target speaker. This task has a wide array of applications in synthesizing voices, and it could serve as a key component in producing human-sounding artificial voices for machines. In this project I aim to explore the possibilities of many-to-many voice conversion, particularly cross-language applications in which the source and target speaker speak different languages. This cross-lingual application is motivated by the necessity to reproduce a speaker's voice and content in a different language, ideally sounding as if the original speaker was actually speaking the other language.

This paper contributes to this research space by contributing to a baseline voice conversion model, based on the StarGAN model originally developed by Kameoka et al., and by conducting experiments with the architecture and data. While the experiments on the architecture were minor and resulted in inconclusive quality changes, the application of this system to vocal samples in different languages yielded promising results. The style of the target voice is clearly identifiable in the resulting voice, while the linguistic information of the source voice remains intact. This shows that the application of voice conversion between languages is possible with current techniques.

1 Introduction

Many tasks in natural language processing (NLP) concern creating machines with the ability to complete tasks that are human-like. We see language as both deeply personal and cultural. In this way, the task of a machine speech generator is one of the most impressive techniques to come out of NLP - it is literally a machine speaking like a human does. An additional layer of impressiveness can be added by creating a machine that does not just speak like *a* human but speaks like *a specific* human. This is essentially the task of voice conversion: given a sample of a human voice, use a machine to generate vocal samples of the same human voice saying different things. Specifically, this describes many-to-many voice conversion, wherein the model can learn multiple voices and transfer the vocal style between any combination of them.
I am specifically interested in this problem because I, along with my collaborator Ian McAulay, are in the process of creating a video dubber through which videos can be dubbed into different languages automatically, without the need for a human voice-over artist. An important criteria for this

technology is for the synthetically generated voice to not only sound natural and believable but for the voice to sound similar to the speaker in the original video. From our research, we have determined that this task is possible, and we have identified several areas in which performance can be improved.

There has been a significant amount of work done in this area, mostly in the last year (2018). In fact, almost all of the papers on which we based our research were conducted in the last year, save for some seminal papers such as the original generative adversarial network (GAN) paper. The next section will detail the background research and related work that has contributed to this paper.

2 Related Work

In the past, VC models have often been restricted to only create audio in a target style of voices on which they were trained (one-to-one VC). Recently, models have been created that can perform many-to-many VC, such that the system can create synthetic speech in an arbitrary style using a small sample (in the range of seconds or minutes) of the target voice.

Some non-neural approaches using Gaussian Mixture Models have been effective (Tomoki Toda and Tokuda (2007)), but recently there has been significant improvement using neural methods, which is what I plan to explore. Several variations of Generative Adversarial Networks (GANs) originally used for image tasks have been applied to this problem. One is the CycleGAN, which is a GAN modified with cycle-consistency loss (Fang et al. (2018)). Another is the StarGAN (specifically StarGAN-VC), which is like a CycleGAN that takes additional information about the target style speaker attributes as input (Kameoka et al. (2018)). Both of the papers describing these methods conducted experiments with data from a small number of speakers (fewer than 10), therefore, using data from a wider variety of speakers may improve the generalizability.

Both the CycleGAN and StarGAN take audio as input for the source speaker. In contrast, it's possible to perform TTS using Tacotron (a state-of-the-art TTS system) along with a speaker embedder (Lee et al. (2018)). Normally Tacotron uses predefined voices, but this poses a problem for VC, which should work for arbitrary new voices. Instead, we can create a speaker embedding model which can convert an audio sample of the target speaker into a vector to pass to Tacotron along with the text. This was shown to be effective at imitating voices even with small audio samples that are only 6 seconds long.

There has been some work on back propagation on inputs regarding the loss in some feature activation as well. In particular, this paper performs back propagation by extracting speakers' features with limited data and is able to generate realistic sounding voices (Chorowski et al. (2017)).

3 Approach

3.1 Baseline

For a baseline model on this task, I used Liu Song Xiang's implementation (https://github.com/liusongxiang/StarGAN-Voice-Conversion) and Hu Jin Sen's implement (https://github.com/hujinsen/StarGAN-Voice-Conversion) of the original StarGAN-VC paper by Kameoka et al. (2018). This repository is a PyTorch implementation that uses a somewhat different architecture than the original paper, which I will describe in this section. Firstly, the architecture of the original paper is as follows. There are three main components to the model: the generator, which takes as an input an acoustic feature sequence and outputs another acoustic feature sequence, and the discriminator and domain classifier, which both take in acoustic feature sequences as input and output probabilities. The generator is comprised of an encoder and decoder. The encoder is comprised of five units, themselves comprised of a 2-dimensional convolution layer, a batch norm layer, and a gated recurrent unit (GLU) layer. The decoder is very similar except with only four units with transposed convolution instead of convolution and a final transposed convolution before the output.

The discriminator and domain classifier are generally similar in structure. They are comprised of four units of the same structure as the encoder, followed by a convolution layer, a sigmoid layer, and a product pooling layer.

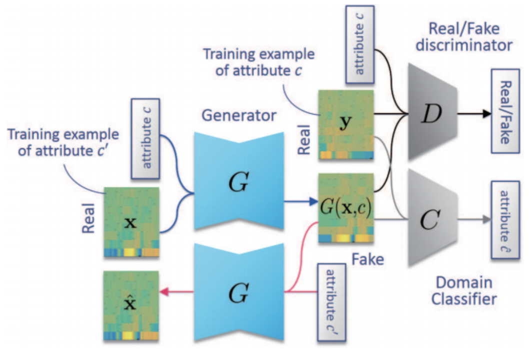


Figure 1: High-level overview of the StarGAN-VC architecture

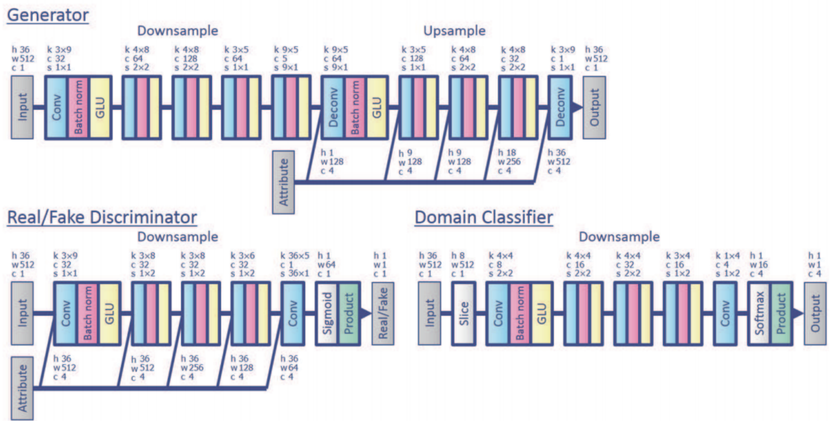


Figure 2: Layer-by-layer breakdown of generator, discriminator, and domain classifier

The role of the discriminator is to take in an acoustic feature sequence and return the probability that the acoustic sample represented by the feature sequence is converted speech rather than real speech. The role of the domain classifier is to return a probability distribution over the set of speakers, producing the probability that the utterance originates from one of the speaker's voices.

The loss functions for the generator, discriminator, and domain classifer as shown below:

$$\mathcal{L}_G(G) = \mathcal{L}^G_{adv}(G) + \lambda_{cls}\mathcal{L}^G_{cls}(G) + \lambda_{cyc}\mathcal{L}_{cyc}(G) + \lambda_{id}\mathcal{L}_{id}(G)$$

$$\mathcal{L}_D(D) = \mathcal{L}^D_{adv}(D)$$

$$\mathcal{L}_C(C) = \mathcal{L}^C_{cls}(C)$$

λ_{cls}, λ_{cyc}, and λ_{id} are hyperparameters used to weight the importance of each respective loss. In this paper, I experiment with increases to λ_{cyc} in order to weight the retention of linguistic content higher in an attempt to create more easily understandable voices. The details and results of this experiment are discussed below.

One of the key departures of Xiang's implementation from the original paper is the use of a ReLU layer instead of a GLU layer in the generator and a leaky ReLU layer instead of a GLU in the discriminator. Additionally, this implementation only uses two units for the encoder and two units for the decoder in the generator as opposed to the recommendation by the original paper of five and four, respectively.

3.2 Architecture Modification

My approach with this first phase of the research was to change the features of Xiang's implementation, that I suspected reduce the performance compared to the results claimed by the original paper, in order to better understand how each substructure in the network affects performance. I conducted several experiments towards this end, which were largely motivated by the original paper. My experiments revolved around adjusting the network it describes one change at a time to isolate the effects of components like the GLU layer, product pooling layer, and number of the conv/batch norm/GLU units on the overall performance. Additionally, I experimented with the hyperparameter values that weight the different losses calculated in the network in an attempt to increase the retention of linguistic information.

My approach with the second phase of research was to conduct experiments with source vocal data from a different language (German) in an attempt to transfer the vocal style and identity of speakers between languages. Because I am interested specifically in applying this technique to artificially created vocal samples, I translated the transcripts of the vocal samples in my dataset (described below) into German, created WaveNet vocal samples of these transcripts, and used the resulting vocal samples as the source utterances to train and test the model.

4 Experiments

4.1 Data

The dataset for all of these experiments CSTR VCTK Corpus, which includes over 10GB of speech data by 109 English speakers of various accents (http://dx.doi.org/10.7488/ds/1994). Each speaker in the dataset speaks around 400 sentences, sourced from newspapers as well as passages specifically designed to identify the speaker's accent. Each vocal snippet lasts a few seconds on average. The samples are preprocessed by finding the mel-frequency cepstrum coefficients (MFCC), which are a cepstral representation of the audio sample.

4.2 Evaluation method

The resulting audio was judged based on three criteria - clarity of audio, perseverance of linguistic content, and quality of vocal style transfer. The last two qualities correlate to the loss functions for the discriminators and the cycle-consistency loss for the generators respectively. I conducted a survey on clarity of audio because I found that this is a key difference between the results of my experiments, generally reflecting how "noisy" the final test output from the network is.

8 colleagues of mine, as well as my collaborator Ian McAulay and I, evaluated the quality of the results from all the experiments and gave feedback on the criteria described above. The averages of these ratings are described in the results section.

4.3 Experimental details

I ran 3 experiments on the architecture of the network. In experiment #1 I changed the ReLU layers in Xiang's implementation back to the GLU layers described in the original paper. In experiment #2 I added three additional conv/batch norm/ReLU components to the generator network. Finally, in experiment #3 I changed the leaky ReLU in the discriminator in Xiang's implementation to a GLU. These experiments were conducted separately, so each change was made in isolation, and the entire network was retrained.

I ran 5 experiments on the data on which the model was given. First, I established a baseline model that was trained by separating the synthetic voices created by WaveNet from the authentic voices from the VCTK dataset. During training and testing, the synthetic voices were always the source vocal samples and the authentic voices were always the target vocal samples. This was done in order to best replicate the conditions under which I hope to employ this technology - using a sample of a human voice to style a synthetic voice in order to engineer a voice that is similar to the original speaker. Once the baseline was established, experiments were then run to improve the retention of

	Audio clarity	Linguistic content	Style transfer	Total average score
TF baseline	3.0	2.1	6.1	3.73
Pytorch baseline	6.4	7.1	6.9	6.8
TF w/ English synth, $\lambda = 10$	1.9	1.8	6.5	3.4
TF w/ English synth, $\lambda = 100$	1.1	1.1	2.3	1.5
TF w/ English synth, $\lambda = 20$	2.4	4.2	4.5	3.7
TF w/ German synth, $\lambda = 20$	2.3	3.2	4.3	3.27
Pytorch w/ English synth, $\lambda = 20$	7.1	7.3	6.8	7.07

linguistic content. This was motivated because the baseline samples, while successfully styled in the identity of the original speaker, were somewhat corrupted and garbled. This suggested that while the adversarial loss was being minimized, the cycle-consistent loss that represents the consistency of the linguistic information of the source speaker was not effectively being minimized.

To combat this issue, I then conducted experiments on the cycle-consistent lambda value to see what effect this would have on the resulting vocal samples with the hope that more of the linguistic content of the sample would be retained, even at the cost of less style transfer. The baseline cycle-consistent lambda value was 10, so I trained a model with $\lambda = 100$ and $\lambda = 20$.

I also trained and evaluated models using German speech. I trained a model on synthetic German data with $\lambda = 20$ once I determined that this was the optimal value.

In the results table, the experiments are labeled with "PyTorch" if they were conducted using the modified PyTorch implementation by Liu Song Xiang (https://github.com/liusongxiang/StarGAN-Voice-Conversion) and "TF" if they were conducted using the modified Tensorflow implementation by Hu Jin Sen (https://github.com/hujinsen/StarGAN-Voice-Conversion).

4.4 Results

The results from the baseline were slightly worse quality than expected given previous samples of voice converted audio I have heard. The main issue with the baseline results is a general fuzziness and incoherence of the voice. This shows that the generator is producing somewhat noisy audio feature sequences. The results of architecture experiment #1 was essentially the same as the baseline in terms of clarity, linguistic content, and style transfer. I believe this is because the function of ReLU and GLU is not very different given their function in this model. The results of architecture experiment #2 was a clear improvement in the audio clarity of the resulting samples. This could simply be because the added layers improved the ability of the network to learn how to generate the finer details in the audio sample and therefore empowered the network to create higher-quality audio in general. The results of architecture experiment #3 was also similar to the results of experiment #1. There was a very slight difference between the results of the baseline and the results of this experiment that could not accurately be described as better or worse in any of the three qualitative categories. As in experiment #2, the function of leaky ReLU is not significantly different than the function of a normal ReLU, so this change was not perceptible in the audio samples.

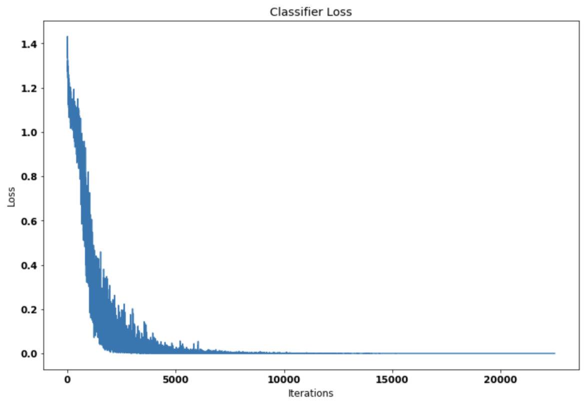
The results of the experiments with data were much more conclusive. Modifying the training and testing of the model, so that there was a separation between the synthetic voices and the authentic voices, created audio samples that were generally less comprehensible and natural sounding. $\lambda = 20$ was the best performing value overall by a significant margin. In general, the Pytorch implementation performed significantly better than the Tensorflow implementation.

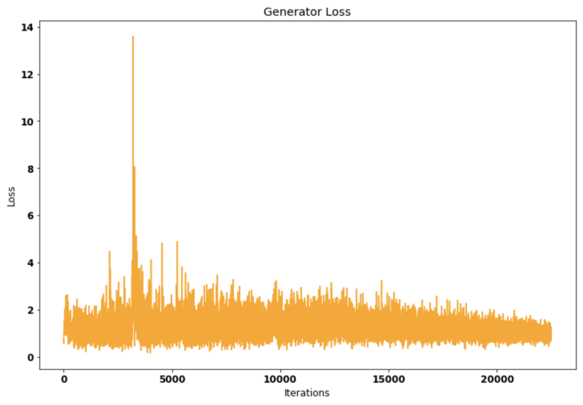
5 Analysis

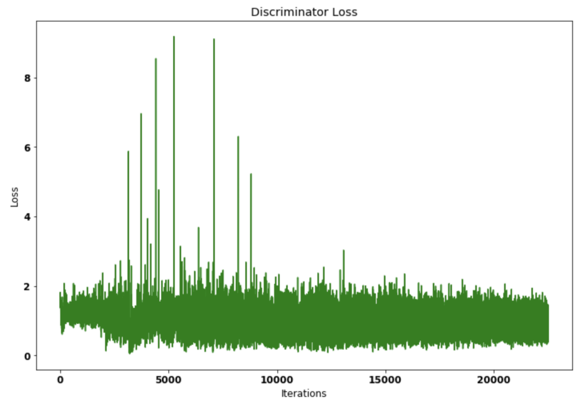
Overall, the results of the experiments were encouraging. The Tensorflow baseline performed fairly poorly as described above, but the Pytorch baseline performed very well without any modification. Although they do have a few architectural differences, I suspect that the key differentiator between the two implementations is the number of speakers on which each is trained. The Pytorch implementation

uses 10 different speakers from the VCTK dataset, while the Tensorflow implementation only uses 4. While this causes the Pytorch implementation to take significantly longer while training, the performance sees a definite increase when the model is given a wider variety of voices from which to learn.
The experiments regarding the cycle-consistent lambda value were a concrete point of learning for this project. I altered the lambda value because the baselines were actually quite impressive when it came to style transfer, but the audio quality was poor and the words sounded garbled. The idea was that by increasing the weight of the cycle-consistent loss, I would motivate the model to retain more of the linguistic information from the source (synthetic) voice at the cost of making the source voice sound less similar to the target voice. This worked, but not as expected. Because the StarGAN model takes over 4 hours to train on a GPU, I started with a large increase to the lambda value such that $\lambda = 100$. The results from this experiment were poor: less style was transfer as intended; however, it also resulted in a significant decrease to the overall quality of the audio clip and made it much harder to understand.
I then set $\lambda = 20$ and retrained and retested the model. This resulted in a much better audio sample, with similar levels of style transfer as the $\lambda = 10$ model but with improved linguistic content retention.
The loss graphs for each unit in the model are shown below. Each experiment showed similar loss graphs. Even though the loss for the generator and discriminator does not seem to decrease over time, the resulting audio samples did significantly improve over the epochs, plateauing around epoch=70. This type of loss graph is not uncommon for GANs, as both the discriminator and generator are constantly improving but staying relatively even with each other such that the loss does not decrease for either. However, you can see that the loss of the classifier decreases dramatically over time, showing that the classifier improves that classifying which voice belongs to which speaker over time. Because this is not in competition with the generator or discriminator, the task of the classifier does not get more difficult over time and its performance in turn sees consistent improvements until around iteration 7000.

The results of the application of this task to German were very encouraging. It was a concern that because German contains different phonemes than English, the model would be unable to transfer the vocal style as effectively. However, this has been shown to not be the case. The synthetic German samples were convincingly styled in the identity of the English samples, even though they were not speaking the same language. The pitch, speaking speed, and overall quality were noticeable in the styled German sample as similar to the target English sample, and the linguistic information from the German sample was retained as effectively as the linguistic information for the English samples were.







6 Conclusion

In this paper, I have demonstrated an improved weighting of the losses for the StarGAN network, as well as the capability of this network to tackle cross-language voice conversion tasks. I have shown that increasing the weighting of the cross-cycle loss can improve the retention of linguistic content in the source sample while still maintaining a comparable level of style transfer as the original weighting. Finally, I have shown that applying this task between different languages is feasible and ripe for future investigation. To continue work on this project, I will further explore cross-language applications, including training on a dataset that contains languages other than English, as well language-specific models, such as training a model to specifically style voices from English to German. I will also explore constructing a deeper network like the one described in the original StarGan paper, along with using training data from more than 10 voices. Finally, this model is only a many-to-many voice converter, meaning that in order to convert to a vocal style, the model must be trained on that voice. I believe that it is possible to create an any-to-any voice converter that uses voice embeddings rather than a one-hot vector to identity the target and source speaker. Using a model such as this, any vocal samples could be provided as the source and tar Mt voices, even if the model hady nt seen them before. This would allow for on-the-fly voice conversion between voices with only a few minutes of audio in the samples. ocollaborator and I are optimistic aboguet the potential for this future work given the results we have demonstrated in this paper.

7 Additional Information

7.1 Mentor

Annie Hu

7.2 External collaborator

Ian McAulay

References

Jan Chorowski, Ron J. Weiss, Rif A. Saurous, and Samy Bengio. 2017. On using backpropagation for speech texture generation and voice conversion.

Fuming Fang, Junichi Yamagishi, Isao Echizen, and Jaime Lorenzo-Trueba. 2018. High-quality nonparallel voice conversion based on cycle-consistent adversarial network.

Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. 2018. Stargan-vc: Non-parallel many-to-many voice conversion with star generative adversarial networks.

Younggun Lee, Taesu Kim, and Soo-Young Lee. 2018. Voice imitating text-to-speech neural networks.

Alan W. Black Tomoki Toda and Keiichi Tokuda. 2007. *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*, 15(8):2222–2235.