President Sings - Voice Conversion Using Generative Adversarial Network

Chang Minwook
2014210066
Department of Computer Science
Korea University
fromme0528@gmail.com

Cho Youngwoo
2014210085
Department of Computer Science
Korea University
cyw314@gmail.com

Min Jungki
2014210093
Department of Computer Science
Korea University
eternalray123@gmail.com

Abstract

Our project aims to synthesize someone's voice singing a song, using Generative Adversarial Network in short, GAN. There is an ongoing research about speech synthesis. The majority of the work have been done in the area of Text-to-Speech(TTS) where it takes text as an input and produces speech as an output. The limitation of TTS we think is that even though it is posssible to reproduce someone's voice, it is hard to capture the features that are the key components in singing such as voice tone, voice pitch, and voice tempo.It is hard to put those voice features in TTS model as an input, as well as make model to learn those features. Another difficulty is that we can't simply earn such feature-labled data. The data we use to learn our model is unpaired data. It is very infeasible to get the data of Trump's singing voice or the data of speaking exactly the same thing that Trump says. Cycle-GAN, current state-of-art GAN, make it possible to convert between unpaired data. Thus we use Cycle-GAN's approach in our voice conversion model.

1 Project Overview

A few months ago, a video titled "Barack Obama Singing Shape of You" had been uploaded on Youtube[1]. Not that Obama actually sang the song, the editor who uploaded the video collected Obama's speech videos and stitched them so that it looks like as if Obama sang the song. This is just one example of funny Internet meme: uploading videos of famous songs sung by celebrities' voice. Although watching those edited videos is fun, the editor has to put a lot of time and effort to make one such video. First, the editor needs plenty of videos that contain voice that she wants to synthesize. Collecting these videos may take a few weeks or worse, a few months. Second, the editor has to edit videos "manually", which means she has to find a time frame of a video that captures the voice needed to build up the song. Then she extracts those time frames, stitchs them, and finally makes one funny piece of music. We think this process, though the result is cool, is definitely painful and inefficient. Here, our project aims to synthesize someone's voice singing a song, using Generative Adversarial Network[1] in short, GAN.

There is an ongoing research about speech synthesis. The majority of the work have been done in the area of Text-to-Speech(TTS)[2] where it takes text as an input and produces speech as an output.

[1] https://www.youtube.com/watch?v=bmhbqKT7ONo

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

The limitation of TTS we think is that even though it is posssible to reproduce someone's voice, it is hard to capture the features that are the key components in singing such as voice tone, voice pitch, and voice tempo. In TTS, the model learns the matching between each phoneme and target voice with a certain feature combination, while in singing a song, same phonemes may have different voice features. Thus, we plan to design our model as Speech-to-Speech(STS), where it takes speech as an input and produces speech as an output. Although this approach is uncommon, we try to build STS model using GAN.

To train our model, large amounts of voice data(perhaps 2~10 hours) have to be collected. These voice data, whether they are target voice or not is not matter because we both have to collect true label data and false label data. We then preprocess the sound data to make a training set because dealing with raw sound data is complicated thing. We used Short-Time Fourier Transformation(STFT)[3] to preprocess the data. Our model, has three internal models. One is Autoencoder, which makes latent vector for the audio. Next one is Generator model, which takes our preprocessed data as an input and produces target voice as an output. Last one is Discriminator model, which takes voice data as an input and outputs true, if the input resembles the target voice, or false, if the input does not sound like the target voice.

Our novel technique can be utilizied in converting voice of celebrities who are alive and also who passed away, so that the fan of those celebrities can enjoy their voice singing new, contemporary songs even after they are dead.

Before beginning our project, we have two major concerns. One is that what if we cannot sufficiently extract the features that consist of singing voice. If we cannot extract the features well enough, the quality of the output will be poor and the output will not resemble the target voice. The other is that our technique can be manipulated by those who want to break the biometric voice authentication system[5]. Biometric voice authentication system uses a client's voice to authenticate whether the given voice is vaild. Although it is not common to use voice recognition technique in traditional authentication area, if that kind of voice authentication applications prevail in the near future, the chance for the intentional attacker to maliciously exploit our technique to break in those applications also goes up.

For our future work, we can think of the way to input the voice singing a song without removing MR. We can consider MR as sort of noise and if we make the model resilient enough to the noise(which is MR in this case). We can develop the noise-resilience model which can directly produce the target voice with MR.

2 Data Sets

We collected audio data on Youtube and SLR[24] data sets. and we preprocess those data to make 10 hours spectrogram. We cut the audio data into three seconds time frame and do STFT that frame. Here the sampling rate is 51200Hz, and the channel is mono. With STFT configuration we followed Praat[21], n_fft to 2048, hop_length to 256, window_length to 1024.

We normalized our spectrogram. To reflect sparseness, we calculate the mean and variance of data larger than 1. Here we appied idea of multiplying 2 to variance[9]

We didn't use dB scale. This is because dB scale is log scale, which is more sensitive. In our case, the noise is a big problem, and the log scale makes this noise problem much more complicated.

3 Problem Definition

Our goal is to transform input voice into target voice we want. This kind of problem is often called "many to one voice conversion".[8] In general, voice that voice conversion deals with is talking voice, but in our case we want to deal with singing voice. This is the reason why we don't use TTS. Current TTS (ex)Tacotron, Deep Voice) can synthesize target voice. But that doesn't mean that we can transform input voice to make target voice sing. Because in singing, voice features such as pitch, tempo are important components. It is hard to put those voice features in TTS model as an input, as well as make model to learn those features. Another difficulty is that we can't simply earn such feature-labled data.

4 Background

Before we propose our model, we illustrate background about voice conversion.

4.1 Spectrogram

Music files in computer represent waveform with respect to amplitude or power. Neither human nor machine learning model can easily recognize this representation and find musical features. Therefore, we transform music wave into spectrogram to help model understand better.

Spectrogram is a graph that represents energy of given signal with respect to time and frequency. Spectrogram is generated using Short-Time Fourier Transformation(STFT). STFT does Fast Fourier Transformation(FFT) given signal with very small window length and hop length, which is used to slide a little bit. The result of STFT is matrix of complex numbers. In general, magnitude of those complex numbers are used in analysis or machine learning.

Our model also uses magnitude of complex numbers. In this case, the phase information is lost, so we can't reconstruct audio signal from magnitude. So we use Griffin-Lim algorithm[20] to iteratively find the lost phase information and reconstruct original audio signal given magnitude of spectrogram.

4.2 Voice Feature

We assume that in spectrogram, we can extract voice features(pitch, tempo, formant, intonation ...)

Especially, the pronounciation of vowel is represented by formant in spectrogram. Formant is a band parallel to time axis in spectrogram. The difference between the first formant in lowest frequency and the second formant in next lowest frequency tells that the given audio frame represents which vowel. In case of consonant, a band parallel to frequency axis in spectrogram represents consonant information.

Pitch is represented by frequency. difference between interval and pitch is that interval is the relative difference while pitch is the absoulte value represented by hertz(Hz). Same pronounciation with different pitch represented by the parallel transference of formant with respect to frequency axis. Intonation is change of pitch.

Tone is determined by spectrum and envelope. Envelope is change of waveform during sound exists. Voice is compound sound mixed with many different frequencies. Spectrum represents the ratio of these frequencies.

5 Model Idea

In voice conversion, the key point is that how to convert intonation and tone while keeping pitch, tempo, and pronounciation. We consider this as a two viewpoint. First, Style Transfer of spectrogram image. Given a spectrogram image, we convert it into spectrogram that has target's style.

Secondly, we think of it as a change of probability distribution. If we see a window of spectrogram, we can see that it is a 2D graph with respect to frequency and amplitude. The graph is a combination of Gaussian probability distributions that has formant as a mean and voice features as a variance. So our model can learn these mean and variance to convert a input probability distribution to target probability distribution.

In general voice conversion models, they use time-dependent model such as RNN. We think we can reflect those two ideas without using RNN, with only using CNN. In phonetics, there are basic sound called IPA. We can classify every single language in the world based on this IPA. This means if we well define convolution filter, we can capture all the features in the voice.

The data we use to learn our model is unpaired data. It is very infeasible to get the data of Trump's singing voice or the data of speaking exactly the same thing that Trump says. Cycle-GAN, current state-of-art GAN, make it possible to convert between unpaired data. Thus we use Cycle-GAN's approach in our voice conversion model.

In Cycle-GAN, it assumes that it can reconstruct the converted data. But in our case, this is a bit awkward because this means that we have to reconstruct random voice from Trump's voice, which is

constructed from again, random voice. In order to handle this awkwardness and consider cycle loss, we introduce autoencoder[6] to make latent vector and use it as an input to our GAN model.

6 Model Structure

MODEL STRUCTURE

Our model has three parts, autoencoder, generator, and discriminator.

6.1 Autoencoder

Current structure of autoencoder is very simple. We have encoder and decoder to make result have same shape with input.

Structure of the encoder(denoted by Enc) is as follows.

Conv(3*3, stride 1, 16ch) – batchnorm – relu – conv(5*5, stride3, 32ch) – batchnorm relu – conv(1*1, stride1, 1ch)

Structure of the decoder(denoted by DecR) is as follows.

Deconv(5*5, stride3, 32ch) – batchnorm – relu or leakyrelu 0.05 – deconv(3*3, stride1, 16ch) – batchnorm – relu or leakyrelu 0.05 - deconv(1*1, stride1, 1ch)

The latent vector between encoder and decoder is not like the normal latent vector, which has information that is hard to recognize, it is more like the activation map in convolution.

Learning of autoencoder utilize pixelwise L1 loss with respect to input spectrogram x and output spectrogram xR,

$$L_{reconstuction}(Enc, DecR, x) =|| \ x - xR \ ||_1 =|| \ x - DecR(Enc(x)) \ ||_1$$

and make this loss minimize.

6.2 Generator

Our generator, DecT is same as decoder in autoencoder with different weight. This is because decoder solves harder problem than generator does. DecR has to reconstruct spectrogram that is same as input spectrogram, while generator only needs to reconstruct spectrogram that represents target voice. This means the range of decoder is larger than the range of generator. Thus if our decoder can reconstruct the given input spectrogram, our generator can generate a spectrogram of target voice.

In the meantime, a spectrogram generated by generator can be an unrecognizable by human. That means the voice features are not in the spectrogram. So we need loss function between latent vector of given input spectrogram and the latent vector of generated spectrogram by generator. This is the reason why we introduce the cycle loss,

$$L_{cycle}(Enc, DecT, x) =|| \ Enc(x) - Enc(DecT(Enc(x))) \ ||_1$$

6.3 Discriminator

In our model, discriminator does two things. First, like the normal GANs, discriminator judges whether the input is real or fake. Second, discriminator judges whether the input spectrogram is target voice or not. We don't have the labled data of target voice so we add this condition to make generator generate target voice. We followed the discriminator model from these.(two papers) Discriminator works as follows.

conv(11*11, stride4, 96ch) – batchnorm – relu – maxpool(2) – conv(5*5, stride1, 256ch) – batchnorm – relu – maxpool(2) – conv(3*3, stride1, 386ch) – batchnorm – relu – maxpool – conv(3*3, stride1, 256ch) – batchnorm – relu – maxpool – fc(4096) – relu – fc(2) – sigmoid

Output is 2D vector, first column means whether the input spectrogram is real or generated from the generator, second column means whether the input spectrogram represents Trump's voice or not. We calculate cross entropy loss as follows.

$$L_{discriminator}(Enc, DecT, Dis, x, y) = H(Dis(DecT(Enc(x))))[0], 0) + H(Dis(x)[0], 1) + H(Dis(x)[1], y)$$

Here, we don't use second column loss from the generated spectrogram because it is not helpful in the early phase of learning and it can generate totally wrong gradient.

7 Result

In out project we want to do something new, we made our model as simple as possible to know whether it is possible approach or not.

Current model can only reconstruct the given input spectrogram. The reason that the voice conversion is not working is that somehow our discriminator cannot discriminate the target voice. We followed exactly the same discriminator structure in the papers[10][7]. We estimate this is because the configuration of our spectrogram and theirs is different.

Also, the noise is a big problem. The reason noise happens is that in original spectrogram, there are zero values that represents empty sound, but in our reconstructed spectrogram, there are no zero values, only near-zero values, which makes noise.

8 Evaluation

In training phase, we proceed with 12288 spectrogram, learning rate of 1e-4, 10 epcoch and batch size of 32. The number of Trump's voice spectrogram is around 2200. We cut front and back part of the audio a little bit to make sure there is no empty sound.

Currently, conversion is not working properly and we can't define ground truth exactly, we couldn't conduct exact experiment. Instead, we focused on loss. Below are the graphs of loss for three cases.

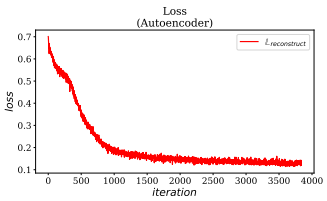

Figure 1: autoencoder loss

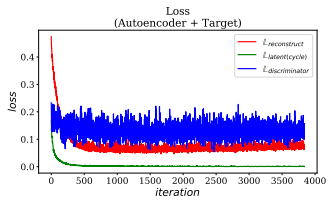

Figure 2: autoencoder + target

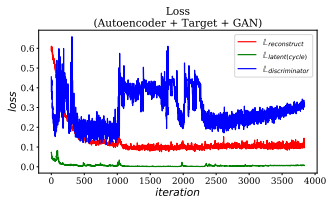

Figure 3: AE + target + GAN

Autoencoder's reconstruction loss and latent vector's cycle loss diminished well, but loss of discriminator heavily fluctuated. This is the reason voice conversion failed.

Also, autoencoder's loss did not converge perfectly, so if we increased epoch, the loss may have decreased more.

For the spectrogram, we chose 3 audio, one is speech without background music, next one is humming with small background music, last one is singing song with loud background music. If we look these spectrograms, we can see that the capability of model to reconstruct and change the target audio, as well as tell apart background music and voice sound. We used Praat to plot the spectrograms and the result of FFT.

8.1 Autoencoder

If you compare figure 4 and 5, it seems that the spectrogram is reconstructed well, but it didn't emphasize the part that has to be, or the other way around.

If you look at the result of FFT(figure 6,7), amplitude suddelny drops at the cropped boundary. This means that the model can't discriminate zero values and non-zero values. Also, there is a problem with normalization cropping and reconstructing.

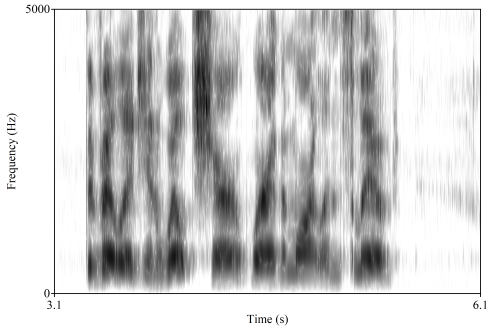

Figure 4: original speech

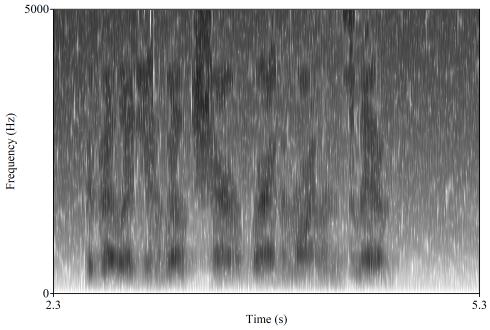

Figure 5: reconstructed speech

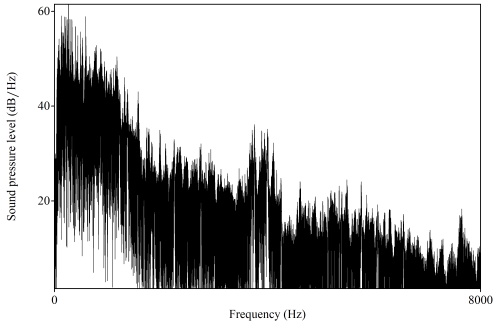

Figure 6: original humming

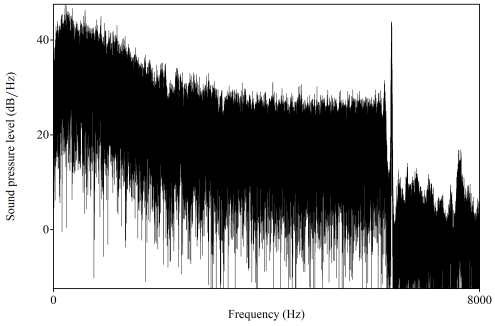

Figure 7: reconstructed humming

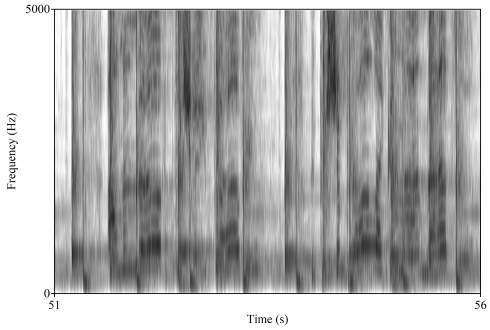

Figure 8: original singing

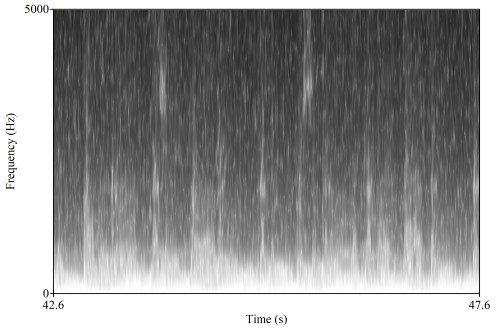

Figure 9: reconstructed singing

8.2 Entire model

Figure 8, 9 shows that the conversion didn't work properly. But compared to 1), reconstruction worked well, but here again, There is no emphasis, only filled with mean value.

Figure 10, 11 shows that the noise ratio is smaller compared to 1).

9 Future Work

First, we need to build more efficient structure. To reduce noise, we have to put the whole spectrogram, not the cropped one. But in this case the resolution of a spectrogram becomes too high to process it fast enough. we need to build efficient model to handle this problem.

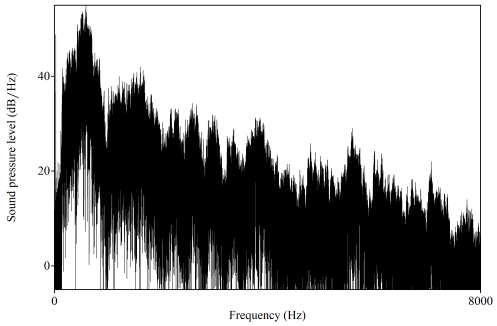

Figure 10: original speech

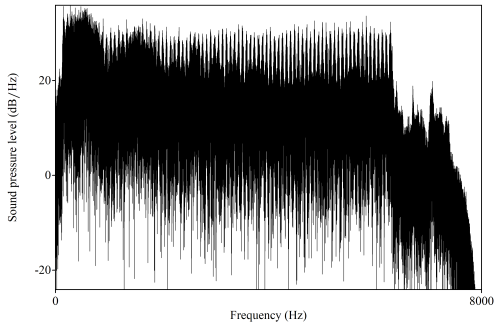

Figure 11: reconstructed speech

Second, current autoencoder makes latent vector similar as a activation map in CNN. This is kind of a problem because the spectrogram now has spatial information, which may hinders the generator from generating Trump's voice. We need to add more fully connected layer to generate proper latent vector.

Third, discriminator is doing multi-task learning. We think we can apply this to autoencoder to have autoencoder extract voice features much more accurately.

Also, we have to find a way to reduce the audio reconstruction time. Current bottleneck is the conversion of spectrogram into audio. This is because Griffin-Lim algorithm is iterative. We can instead use Fast Griffin-Lim algorithm to deal with this problem. Short conversion time is very important in real-time voice conversion application and user experience.

Our model can be developed into many-to-many conversion. If our generator can generate arbitrary target voice, this is possible to convert arbitrary voice to arbitrary voice.

References

[1] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative Adversarial Nets. *NIPS*, 2014.

[2] Allen, Jonathan; Hunnicutt, M. Sharon; Klatt, Dennis. From Text to Speech: The MITalk system. *Cambirdge University Press*, 1987.

[3] Jont B. Allen. Short Time Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1977.

[4] P.Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, 1976.

[5] H Vallabhu, RV satyanarayana. Biometric authentication as a service on cloud: Novel solution. *International Journal of Soft Computing and Enginerring*, 2012.

[6] L.Deng, M.Seltzer, D.Yu, A.Acero, A.mohamed, G.Hinton. Binary Coding of Speech Spectrograms Using a Deep Auto-encoder. *International Symposium on Computer Architecture*, 2010.

[7] Lior Uzan, Lior Wolf. I Know That Voice: Identifying the Voice Actor Behind the Voice. *IEEE International Conference on Biometrics*, 2015.

[8] Lifa Sun, Kun Li, Hao Wang, Shiyin Kang, Helen Meng Phonetic posteriorgrams for many-to-one voice conversion without parallel data training, 2016.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015.

[10] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional Neural Networks, for Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2014.

[11] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, Jiwon Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks, 2017.

[12] Sercan O. Arik, Mike Chranowski, Adam Coates, Gregory Diamos, Andrew Gibiansky. Deep Voice: Real-time Neural Text-to-Speech, 2017.

[13] Siri Team. Deep Learning for Siri's Voice: On-device Deep Mixture Density Networks for Hybrid Unit Selection Synthesis, 2017.

[14] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly. Tacotron: Towards End-to-End Speech Synthesis, 2017.

[15] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals. Wavenet : A Generative Model for Raw Audio, 2016.

[16] Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks *International Conference on Learning Representations*, 2016.

[17] Chris Donahue, Zachary C. Lipton, Julian McAuley. Dance Dance Convolution, 2017.

[18] Elias Sprengel, Martin Jaggi, Yannic Kilcher, and Thomas Hofmann. Audio Based Bird Species Identification using Deep Learning Techniques, 2016.

[19] Lonce Wyse. Audio Spectrogram Representations for Processing with Convolutional Neural Networks, 2016.

[20] Daniel W. Griffin Jae S. LIM. Signal Estimation from Modified Short-Time Fourier Transform *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.

[21] Paul Boersma and David Weenink. Praat: doing phonetics by computer, 2017.

[22] McFee, Brian and Raffel, Colin and Liang, Dawen ane Ellis, Daniel PW and McVicar, Matt and Battenberg, Eric and Nieto, Oriol. librosa: Audio and Music Signal Analysis in Python *Proceedings of the 14th python in science conference 25*, 18-25, 2015.

[23] Eric Jones and Travis Oliphant and Pearu Peterson and others. Scipy: Open source scientific tools for Python *http://www.scipy.org/*, 2001.

[24] Vassil Panayotov, Guoguo Chen, Daniel Povey and Sanjeev Khudanpur. LibriSpeech: an ASR corpus based on public domain audio books *ICASSP*, 2015.