Instrumental Music Translation

Chitrank Gupta, Satyabrata Naik
170050024, 170050087

Abstract

We adapt the cycleGAN implementation applied on the visual domain to the audio domain by using it to transfer the instrument style of a music sample from one instrument to the other.

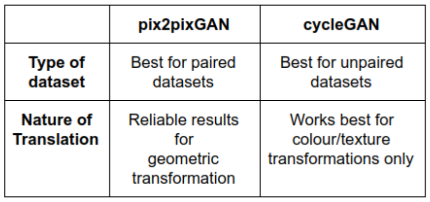
1 Introduction

Our project consists of mainly three parts, preprocessing, Training and Inference.
In preprocessing, we obtain the spectrogram of the music samples via the STFT procedure and use these spectrograms in the training our model.
STFT is widely used for music analysis. Audio signal is divided into shorter time intervals (possibly with overlapping) called frames and Discrete Fourier transform is applied on each frame to get the Fourier spectrum. The Fourier spectrum contains more information as compared to the temporal signals, as it also captures the spectral features of the audio at different instances of time. Similarly, a direct DFT of the signal is useless because it doesn't have any temporal information in it. This makes the STFT as the best procedure for the preprocessing of audio files. The steps of preprocessing can be described as below:
Preprocessing

- *Load data from '.wav' file using* ***librosa*** *library at sampling rate 25KHz*
- *Do STFT on the loaded data with "hann" windowing and number of samples per window equal to 511. This will return 256 frequencies in the spectrogram.*
- *Separate the amplitude and angular components. Only amplitude component will be transformed during training and inference and the angle component will be added as it is.*

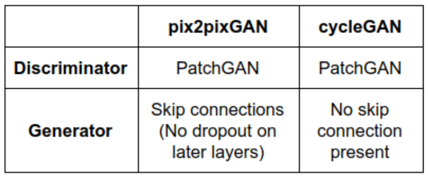
2 Training

Generative Adversarial Networks are widely used for modeling joint probability distribution on the observed data. Henceforth we will be talking about images(”spectrograms” could be used interchangeably).
Conditional GANs are an improvement on vanilla GANs, which can help in translating data in one domain to another domain. Unfortunately, not all type of models of cGAN can do this. Specifically pix2pix GANs [3] are found to perform best on such translation tasks but owing to very specific nature of how they learn, they fail to learn the kinds of datasets where the images from two datasets are not ”paired” that is to say that we possible don’t have any specific target image to which we want to translate our input image.
Zhu et al [1] found an interesting way in which combination of discriminators and generators can be trained without the need of having the paired inputs, at the cost of increase in the training time. The Figure 1,2,3 bring some important differences observed in the architecture and training of the two models-pix2pix GAN and cycleGAN.
The dataset that we have used for our project was created for ”music instrument classification” tasks and hence this dataset obviously lacks the pair-ness in the audio samples. This lead us to use cycleGAN as a natural choice. Mor et al [2] of Facebook AI Research team although, have created a universal music translation network in which the encoder is shared across domains but separate decoders are used for each domain.



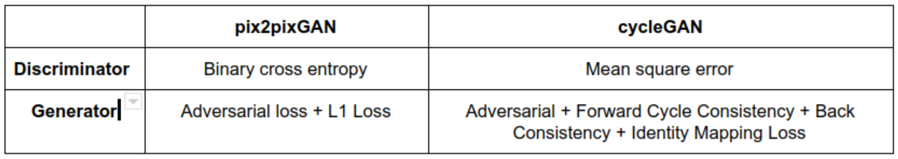
	pix2pixGAN	cycleGAN
Type of dataset	Best for paired datasets	Best for unpaired datasets
Nature of Translation	Reliable results for geometric transformation	Works best for colour/texture transformations only

Figure 1: Difference between pix2pix GAN and cycleGAN



	pix2pixGAN	cycleGAN
Discriminator	PatchGAN	PatchGAN
Generator	Skip connections (No dropout on later layers)	No skip connection present

Figure 2: Difference between pix2pix GAN and cycleGAN in the architecture of Discriminator and Generator



	pix2pixGAN	cycleGAN
Discriminator	Binary cross entropy	Mean square error
Generator	Adversarial loss + L1 Loss	Adversarial + Forward Cycle Consistency + Back Consistency + Identity Mapping Loss

Figure 3: Difference between pix2pix GAN and cycleGAN in the type of Loss function used in its training.

2.1 The model

Our model is adapted from the original cycleGAN paper [1] which made use of two discriminators (called Dx and Dy pertaining to domain x and domain y) and two generators (Gx-y and Gy-x as explained later).

Discriminator

It is a simple patchGAN discriminator (Markovian discriminator) where the output of discriminator is not just a scalar quantity (unlike in ImageGAN). Idea of having an optimal field of reception per output pixel of discriminator has been described in the pix2pix GAN [3], where they argue experimentally that very high value of reception leads to very sharp output (fake) spectrograms but poorer performance in the objective evaluation as compared to if we had limited reception field.
Unlike cGAN, this discriminator is trained "unconditionally". That is to say that without conditioning on the input spectrogram, it learns to classify the real spectrograms as real spectrograms and the fake spectrograms (as output by generator) as fake spectrograms. So for example Dx will learn to classify real spectrograms of class x as real and fake spectrograms of class x (output by Gy-x) as fake. Hence the loss function of Discrimintor (specifically for Dx) looks like (assuming means sqaure error is used for loss calculation)

$$E_x[\|1 - D_X(x)\|_2^2] + E_y[\|D_X(G(y))\|_2^2]$$

where x is the data from domain X, and G is Gy-x

Generator

It is a simple Deep convolutional autoencoder. It makes use of doubly-strided convolutional layers in the encoder for about 6 layers with more feature maps as the depth increases, until the bottleneck layers are reached which keep on passing information to next few layers with skip connection to the next of the next layer (such a unit of 3 layers is also called residual block), after which another 6 layers of doubly-strided transpose convolutional layers help upsampling the latent space until it constructs an spectrogram of original shape. See Figure 4.
As described earlier cGAN suffer from the requirement of having the images from two domains paired. Question is where exactly is cGAN affected? There are two places where the pix2pix GAN architecture gets affected-1)conditioning with the input image in the discriminator and 2)The additional l1 loss term where it tries to lessen the distance between the fake and target image. The cycleGAN alleviates the first problem very easily- by removing the conditioning. The solution to the other problem is quite convoluted in the sense that now generator tries to minimise weighted sum of 4 different components (none of which is that additional L1 norm loss) of the loss.
Specifically the 4 components are

Archi.png

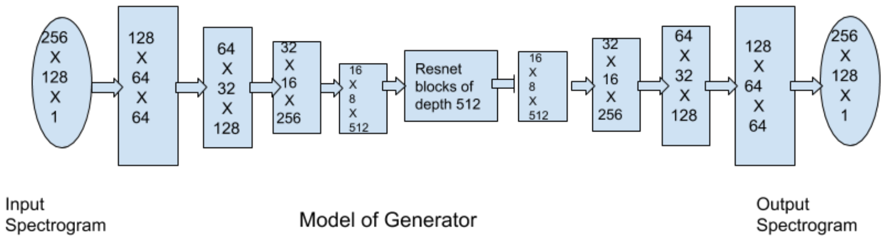


Figure 4: Architecture of Generator

1. The original Adversarial loss objective of Generator. For Gx-y it will be

$$E_x[\|1 - D_Y(G(x))\|_2^2]$$

2. Forward cycle consistency loss. When G1 (or Gx-y) is trained then

$$E_x[\|G2(G1(x)) - x\|_1]$$

   where G1 is Gx-y and G2 is Gy-x

3. Backward Cycle consistency loss. Again when G1 is trained

$$E_y[\|G1(G2(y)) - y\|_1]$$

4. Identity mapping loss. Again G1 is trained

$$E_y[\|G1(y) - y\|_1]$$

The figure 5 and Figure 6 explains the concept in depth.

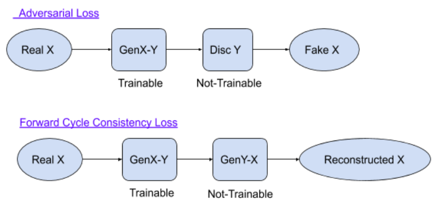


Figure 5: The adversarial loss and the forward cycle consistency Loss

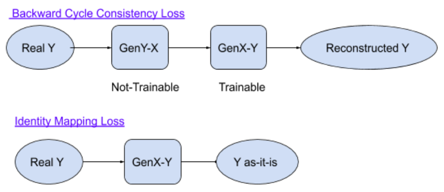


Figure 6: The backward cycle consistency Loss and Identity Mapping Loss

2.2 The training Details

In every iteration of training we trained our model on batch size of 2 training examples. The sampling of training examples was done randomly. Similar was the way of sampling fake images. An interesting paradigm in sampling of fake images was to keep a buffer of fake images produced by the generator in the previous iterations of training and then probabilistically re-create the array of fake images and at the same time update the buffer as well. As reported by Srivastava et al [5]. This tends to reduce oscillations in the training.
First we train both discriminators two times- each time learning the proper classification of their domains (real and fake images) and then trained the two generators ensuring that one generator is trained, the other generator as well as the discriminator of its domain is kept non-trainable. So in our notations, when Gx-y is trained , D-y and Gy-x is kept non-trainable.
Also, because there were so many models each with many layers and many parameters, it seemed that huge amount of training will not lead to overfitting and hence we sub-optimally trained it for about 27 hours where each iteration (training size equal to batch size) took around 5-10 seconds. The training was done on Kaggle (thanks for their "commit" feature) with experimentation on google colab. The tensorflow wrapper of Keras library was used to build our models which greatly simplified the coding.

Hyperparameters for training

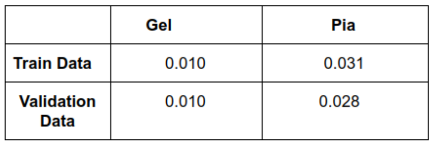
Almost all hyperparameters were adapted from the the paper. The weights coefficients of the first 3 components of the generator loss had the respective values of 1, 10 and 10. The weight coefficient for the identity mapping loss was kept as half as that of the coefficient of forward cycle consistency loss; we changed it to 4. Similarly the discriminators were trained slower than the generators by making their loss function (which has only one component) halved. The learning rate was 2.0e-4. The number of layers in generator were adapted according to the size of the input.

3 Dataset

The dataset used was IRMAS dataset [6] which had around 600-700 audio samples for each kind of instrument. We used the piano and electric guitar as the two main instruments and the two main domains of music styles. 600 audio samples were taken for training while 50 were kept for validation.

4 Experiments and Results

We measured the mean relative forward cycle consistency (FCC) loss as the evaluation metric of our model. Such distance measures do not form good objective measures of performance but nonetheless, the objective metrics used



	Gel	Pia
Train Data	0.010	0.031
Validation Data	0.010	0.028

Figure 7: The mean relative FCC loss after 27 Hrs of Training. here Gel represents electric guitar and Pia means Piano.

in visual domains (for example Inception score and Fréchet Inception Distance could not be used to assess quality of spectrograms which might not even be close to any of the classes of images on which these metrics are trained on. Here is the link to all the audio files generated- Audio Files- Click here The annotation of the files is as follows-
1. rec_gel.wav means the real sample of electric guitar.
2. rec_gel_to_pia.wav means that the output of generator gel to pia.
3. rec_gel_to_pia_to_gel.wav means the output of generator of pal to gel on the fake input of the generatir gle to pia.
Pia Here stands for piano and above 3 points are same for pia and gel reversed. Also without ”2” in the end of the filename means the files are as a result of 18 Hrs of training and ”2” in the end of filename means the files were generated from models trained for 27 Hrs. The Figure 7 shows the mean relative FCC loss on the training and validation data after 27 hrs of training.

5 Conclusions

We showed that cycleGAN can be adapted to the audio domain provided the style transfer between the two domains involve colour or texture changes and not geometrical changes. Change in instruments bring change in the texture of the spectrogram but not in the geometrical structure of it and hence cycleGAN could be a good choice for instrument translation problem. But the main hindrance is that the dataset is often ”corrupted” with other music instruments which disturbs the distribution characteristics of the training dataset, hence leading to failures (and unstable training), as is indicated in the paper.
As an aside note, with more of training time we could definitely achieve better results. One could use better evaluation metrics or adapt the existing ones in audio domain to make evaluations easier. Also the datasets could be tried to make cleaner or tried to be made more ”style-transfer porblem” oriented and not just ”classification problem” oriented.

References

[1] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

[2] A UNIVERSAL MUSIC TRANSLATION NETWORK

[3] Image-to-Image Translation with Conditional Adversarial Networks

[4] Keras Tutorial

[5] Learning from Simulated and Unsupervised Images through Adversarial Training

[6] IRMAS Dataset