CS230

Audio Style Transfer

Kevin Lin
Department of Computer Science
Stanford University
linkevin@stanford.edu

Abstract

Recently, CNNs have been successfully applied to neural style transfer for images. However, not as much research has been done on neural style transfer for audio. In this paper, I experiment with a baseline CNN, as well as a more complex GAN architecture to see how it performs in this field. GANs seem to hold a lot of promise, but takes significantly longer to train.

1 Introduction

Recently, CNNs have been successfully applied to neural style transfer for images. Images are represented by an array of pixels, and the "style" of an image can be defined by its colors and localized structures, which has led to style transfers with high perceptual quality [6]. However, when attempting to apply the same concept with audio (i.e. audio style transfer), previous attempts have faced challenges. Music can be represented in multiple ways - it can be "read, listened to, or performed, and it all depends on whether we are relying on score (the top-level, abstract representation), sound (the bottom-level, concrete representation), or control (the intermediate representation) [...] and no end-to-end system can deal with all levels of music representation together in an elegant manner" [4].

This problem is important because it requires applying deep learning to the field of music and acoustics. A successful model would also produce entertaining and beautiful results. Imagine being able to create a soundtrack like the Duomo Cover of Taylor Swift's *Wildest Dreams* by simply inputting *Wildest Dreams* and a violin soundtrack into a model.

The input to this problem is a content audio spectrogram and style audio spectrogram. For the baseline, we then use a one-layer, one-dimensional CNN with randomized weights to help output an output generated audio spectrogram. For the more advanced model, a GAN architecture is used instead.

2 Related work

In the past, multiple methods have been attempted to solve the problem of audio style transfer. The most rudimentary is using the techniques of image style transfer. [5] and [6] both utilize similar model architectures to and take inspiration from Gatys, Ecker, and Bethge's frequently cited *Neural Art Style Transfer* [7]. These techniques benefit from the extensive research done in the field of CNNs and neural art style transfer. However, they often fail to successfully transfer the complex patterns that define musical style.

A more complex technique involves the use of RNNs. Current research in this area [9] seems to suggest that RNNs do not do a good job discriminating between the content and style of audio.

A more promising method is the use of deep generative models such as VAEs [1, 3, 10] and GANs [1, 8, 11]. Though these models can significantly reduce the need for copious amounts of labelled data, large amounts of data also aren't necessarily needed for the aforementioned techniques, especially given the rise of transfer learning. One potential explanation for the success of deep generative models in audio style transfer is their incredible flexibility in handling audio of wildly different styles [11].

There are also a number of papers that utilize a combination of methods such as [8], which combines the use RNNs and GANs, and [2], which combines the use of AEs and transformers.

3 Dataset and Features

I spent most of my time experimenting with audio files (mp3 and wav) I found interesting. As a Swiftie and classically trained musician, the ones I experimented with the most were *Wildes Dreams* by Taylor Swift and *Partita for Violin No. 1* by Bach.

In order to make the audio data compatible with the models, the Short-time Fourier transform (STFT) was applied to all content and style audio files. An example is shown in Figure 1. Note that power spectrograms are shown for human interpretation.

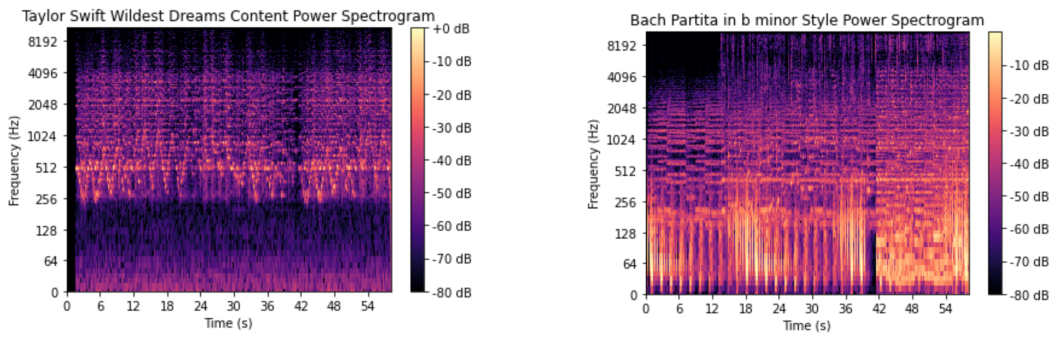

Figure 1: Power spectrograms of a content audio file and style audio file.

For the baseline, we take the natural log of one added to the spectrogram value. For the GAN, the frequencies are converted to the mel scale (a mel spectrogram), data augmentation is applied by taking random crops of the spectrogram, and the spectrograms are divided evenly along the time-axis to a constant size of L in order to allow inputs of variable length.

In order to reconstruct an audio signal from the generated output spectrogram, the Griffin-Lim algorithm is used:

An audio signal was reconstructed from the generated spectrogram by using the following iterative algorithm:

$$x_{n+1} = \text{istft}(S \cdot \exp(i \cdot \text{angle}(\text{stft}(x_n))))$$

where istft stands for the Inverse Short-time Fourier transform.

The data was retrieved by recording audio from YouTube.

4 Methods

For the baseline, I used a CNN architecture from [5]. For the more advanced model, I used a GAN architecture from [11]. If you would like to see an implementation of these methods, please visit the following GitHub repo: `https://github.com/linkevint/230-final-project`.

4.1 Baseline

For the baseline model, I used a one-layer, one-dimensional CNN with random weights implemented in [6]. Due to the fact that the generated image is initialized to the content image and the addition of content loss doesn't seem to significantly improve performance [5, 12], the overall loss was simply the style loss, which can be calculated as follows:

$$L(C,G) = \frac{\sum_{i,j}(G_{\text{gram}_{ij}} - S_{\text{gram}_{ij}})}{4n^2m^2}$$

where C is the content spectrogram, G is the style spectrogram, $G_{\text{gram}_{ij}}$ is the (i,j)th entry of the generated spectrogram's gram matrix, $S_{\text{gram}_{ij}})$ is the (i,j)th entry of the style spectrogram's gram matrix, n is the number of filters, and m is the number of samples (i.e. the width of the input spectrograms).

The model uses an Adam Optimizer, which combines the advantages of both Momentum and RMSProp to improve SGD. More specifially, it uses the average of a gradient's second moments. Intuitively, we would want the learning rate to be smaller when recent gradient calculations have had high variation. It's important to note that, just like with neural art style transfer, the random weights of the CNN are not changed during the optimization process. Only the pixels of the output spectrogram are updated on each step.

4.2 GAN

For the more advanced model, I used a GAN architecture implemented in [11]. The GAN has three networks: a generator (G), discriminator (D), and a siamese network (S). The architecture of these networks is described in-depth in [11], but at a high-level, all are some form of CNN.

For the loss formulas, let A be the original distribution, B be the target distribution, $L_{(G,S),TraVeL}$ be the TraVeL loss described in [11] that tries to preserve low-level information, and $L_{S,margin}$ is a siamese margin-based contrastive loss. Then, the loss functions of the networks are as follows:

$$L_D = -\mathbb{E}_{b\sim B}[min(0, -1 + D(b))] - \mathbb{E}_{a\sim A}[min(0, -1 - D(G(a)))]$$

$$L_G = -\mathbb{E}_{a\sim A}D(G(a)) + \alpha\mathbb{E}_{b^{id}\sim B}[||G(b^{id}) - b^{id}||_2^2] + \beta L_{(G,S),TraVeL}$$

$$L_S = \beta L_{(G,S),TraVeL} + \gamma L_{S,margin}$$

Like with the baseline, the Adam Optimization algorithm was used to train the parameters of the three networks.

5 Experiments, Results, and Discussion

5.1 Baseline

Due to time constraints, I was unable to experiment with different hyperparameters, and thus was only able to run experiments with the hyperparameters proposed in [6] and [11]. However, both papers did thorough hyperparameter searches that at least avoided catastrophic issues like gradient vanishing/exploding.

For the baseline, the initial learning rate of the Adam Optimizer was 0.03, with betas 0.9 and 0.999, a weight decay of 0, and epsilon of 1e-08. The single convolution layer had 4096 filters of size 3, and used a stride of 1, as well as a padding of 1. Recall that the weights were randomized.

An example of an output after 400 epochs is shown in Figure 3, and can be listened to here[1].

[1]https://soundcloud.com/kevinlin900/audio-style-transfer-cnn-400-epochs

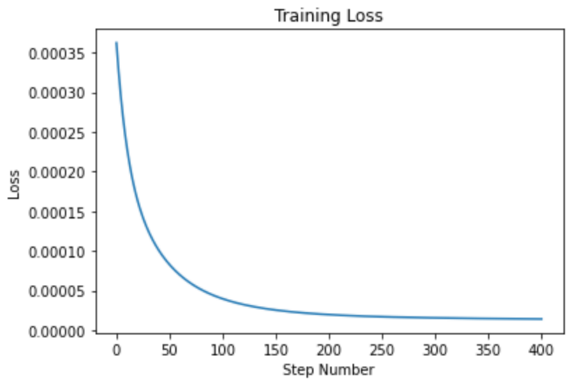

Figure 2: Sample baseline training loss.

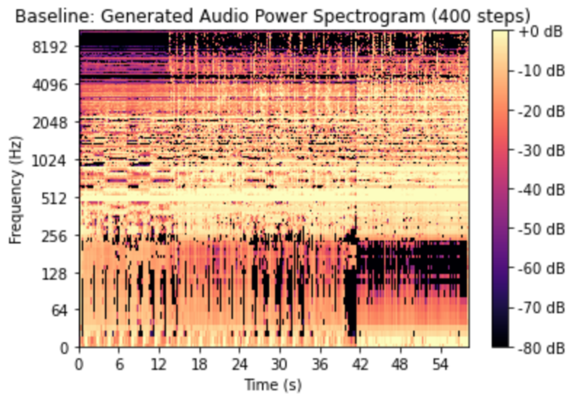

Figure 3: Output sectrogram of Baseline after 400 epochs.

5.2 GAN

For the GAN architecture, the learning rates for the three networks were as follows: $lr_D = 0.0004$, $lr_G = lr_S = 0.0001$. The batch size was 16. The following coefficients were used: $\alpha = 1$, $\beta = 10$, and $\gamma = 10$. L was set to 96. The details of the three network architectures can be found in [11].

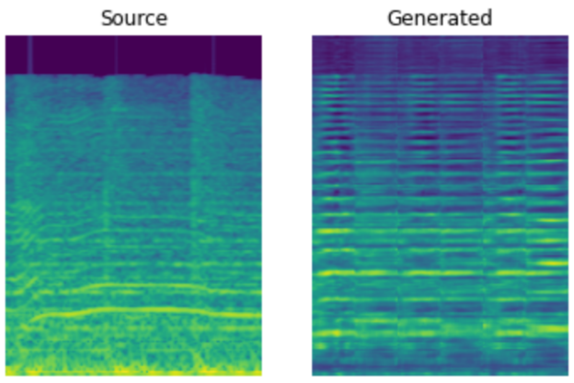

Figure 4: Sample of an input spectrogram and its generator output after 104 epochs.

An example of a result after 400 epochs can be heard here[2].

[2]https://soundcloud.com/kevinlin900/audio-style-transfer-gan-400-epochs-wildest-dreams-in-style-of-bach-partita-in-b-minor

The following file[3] is a record of the discriminator, generator, and identity mapping losses. A brief look will reveal that losses are far from plateauing and that there is potential for far better results. Unfortunately, due to time constraints, I was unable to train for a longer period of time, which I address in the next section.

6 Conclusion and Future Work

Based on the experiments, we can see that GAN architectures hold lots of promise in the field of audio style transfer. Listening to the sample result from the GAN model, we can start to hear the low-level violin sounds from Bach's Partita in b minor being transferred to Taylor Swift's Wildest Dreams, while some of Wildest Dream's content melody remains. However, Taylor Swift's voice was nowhere to be found, long-range melody structures broke down, and overall, the generated audio was not pleasant to listen to.

However, these results can likely be partially explained by the fact that due to time constraints, I wasn't able to train the GAN to completion. Despite training on a p2.xlarge EC2 instance, at the demonstrated rate of training, 500 epochs would have taken over four days. Thus, I also was not able to experiment with different hyperparameters/architectures or audio data besides ones I was personally interested in. Exploring these avenues seem to be promising future directions.

All in all, I learned an incredible amount and found the experience deeply rewarding. This was the first time I was able to apply existing AI models to a problem I was genuinely interested in. It was also the first time I experienced the benefits of cloud computing's scale when it comes to training deep learning models. A huge thank you to the teaching team for a wonderful quarter!

References

[1] Brunner, G., Konrad, A., Wang, Y., & Wattenhofer, R. (2018). MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer. arXiv preprint arXiv:1809.07600.

[2] Choi, K., Hawthorne, C., Simon, I., Dinculescu, M., & Engel, J. (2020, November). Encoding musical style with transformer autoencoders. In International Conference on Machine Learning (pp. 1899-1908). PMLR.

[3] Cífka, O., Şimşekli, U., & Richard, G. (2020). Groove2Groove: One-Shot Music Style Transfer With Supervision From Synthetic Data. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28, 2638-2650.

[4] Dai, S., Zhang, Z., & Xia, G. G. (2018). Music style transfer: A position paper. arXiv preprint arXiv:1803.06841.

[5] Dipani, A. (2018), Intel Blog, https://software.intel.com/content/www/us/en/develop/articles/neural-style-transfer-on-audio-signals.html

[6] Dipani, A. (2018), GitHub repository, https://github.com/alishdipani/Neural-Style-Transfer-Audio

[7] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

[8] Jayakumar, S., Ramesh, R., & Thalasta, P. (2017). ToneNet : A Musical Style Transfer. https://towardsdatascience.com/tonenet-a-musical-style-transfer-c0a18903c910

[9] Malik, I., & Ek, C. H. (2017). Neural translation of musical style. arXiv preprint arXiv:1708.03535.

[10] Mor, N., Wolf, L., Polyak, A., & Taigman, Y. (2018). A universal music translation network. arXiv preprint arXiv:1805.07848.

[11] Pasini, M. (2019). Melgan-vc: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms. arXiv preprint arXiv:1910.03713. https://github.com/marcoppasini/MelGAN-VC

[12] Ulyanov, D., Lebedev, V. (2016) https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/

[3]https://github.com/linkevint/230-final-project/blob/main/logs/GAN_trial2_logs.pdf