Music Style Transfer Using Pre-Trained Autoencoders

Beril Besbinar   Jade Maï Cock   Paola Mejia   Baran Özaydin   Ehsan Pajouheshgar

***Abstract*—Convolutional Neural Networks (CNNs) extract features whose representations can encode both the style and the content information of an image. This particular property has made them the perfect candidates to transfer the style of one image onto another one. The quality of the resulting pictures motivates us to adopt the audio equivalent of Image CNNs to perform music style transfer. Most existing methods operate on spectral audio representation instead of raw waveform. Unlike current methods, we propose using auto-encoders trained in a self-supervised manner to transfer the timbre of one instrument to an audio sample played by another instrument. To that effect, we first experiment in the image domain and then apply the methods whose output were promising onto the audio domain. Results show that well-established methods for image style transfer are not very promising for audio style transfer.**

***Index Terms*—Style Transfer, Timbre transfer, JukeBox, CNN, Autoencoder, insert**

I. Introduction and Related Works

The emergence of Deep Convolutional Neural Networks (CNNs) has opened the doors to many artistic applications, one of which is neural style transfer. As the pioneer work in the field, Gatys et al. [4] show how representations encoded in the middle layers of a CNN, such as VGG19 [12], which is trained for image classification task, can be used to define the "content" and the "style" of an image. In that regards, they formulate the problem of neural style transfer as the generation of a new image guided by two input images, one for content and the other one for style. Their method does not require training any neural networks. The image to be generated, i.e., the target image, is the only parameter over which the loss function is optimized. The loss functions, on the other hand, is composed of two terms: the content loss which tries to minimize the $L2$ distance between the embeddings of the content and target images at a certain layer in the pre-trained CNN, and the style loss which penalizes the discrepancy between the statistical properties of embeddings of the style and target images at different layers. They use the Gram matrix[1] to describe the embedding-based statistics. However the optimization process takes a long time and is not suitable for real-time applications.

To overcome this problem, the authors of [10], [11] propose to use CNN decoders to directly decode the intermediate featuremaps to the output image. Then the idea is to perform some transformations on this intermediate representation of the content image to change its stylistic aspects. Li et al. [11] propose Whitening-Coloring Transform (WCT) that matches the Gram matrix of the content image with the Gram matrix of the style image. Huang et al. [10] suggest a simpler method called Adaptive Instance Normalization (AdaIN) to simply swap the channel means and variances of the featuremaps[2]. However, both of these methods train CNN decoders on top of CNNs pretrained for image classifications such as VGG19 [12].

Recently, there has been an increasing interest in adopting neural style transfer to audio signals [1], [7]. As an early attempt, Eric *et.al.* experiment with different approaches to apply neural style transfer to audio signal based on the Short Time Fourier Transform (STFT) spectograms. Their findings show that rather than initializing the target signal with random noise, modifying the content audio by a shallow neural network trained from scratch to match the style with the second input performs the best [7]. Similarly, Huang*et.al.* [8] use Constant Q Transform (CQT) to map the audio signals to a two-dimensional space, where the timbre transfer is achieved with CycleGAN, and the audio is generated by a conditional WaveNet from the resultant CQT representation. On the other hand, few other works [1]

Style Transfer for music is more challenging for several reasons.

- Training CNNs on raw music is significantly more complex[3].
- Spectogram representations are sparser and easier to work with due to the periodicity of the audio signals.
- There is no state of the art network like VGG19 [12] that is trained on raw audio for music recognition.

Because of these reasons, the audio/music style transfer works use spectogram representations such as Short-time Fourier Transform (STFT), Constant Q Transforms (CQT), or Mel Frequency Cepstral Coefficients (MFCC) instead of wave form representations [6], [9], [13]. Grinstein et al. [6] use the STFT representation of an audio signal as an image to perform style transfer. Similarly, [9] uses CQT transform to convert a piece of music to an image. Then they convert the stylized STFT, CQT images back to wave representation to get the Stylized

We would like to thank Paolo Prandoni and Eric Bezzam for the amazing Computers and Music course.

[1]Gram matrix represents the correlations between channels of the featuremap.

[2]The channel variances are basically represented in the diagonal of the Gram matrix.

[3]This is due to the high sampling rate of audio signals and the fact that musics can be arbitrarily long.

audio/music. However, reconstructing the phase from STFT, or CQT is very challenging. Moreover, the proposed timbre transfer method in [9] requires to train a new network for each pair of instruments.

In this project we investigate the applicability of autoencoders for music style transfer[4]. The work of Cífka et al. [2] is the closest to our work, however, they use the MFCC representation of the audio. We want to use the wave representation of audio signals to avoid the phase reconstruction problem. Using autoencoders for audio style transfer has several benefits.

- There is no need to re-train the model for each instrument (zero-shot style transfer).
- There is no need for annotated data to train autoencoders.
- There is an existing autoencoder trained on 1.2 million songs called Jukebox [3].
- There is no need for optimization and we can perform style transfer in real-time.
- There is no need for phase reconstruction if we generate waveform directly [5].

To the best of our knowledge, there is no work exploiting autoencoders trained without supervision for image/music style transfer[5]. We use Jukebox autoencoders, and implement the methods introduced in [10], [11] for music style transfer. We use a few metrics to quantitatively compare the style transfer quality of different methods. We also provide the generated results for qualitative comparison.

In II, we discuss the details of our proposed methods. In the discussion section V, we provide a few reasons explaining why the image style transfer methods do not perform well on music signals, and discuss the challenges. In section IV we provide quantitative and qualitative analysis of the results.

In the supplementary material we provide some of experiments we performed to understand Jukebox. We also provide the image equivalent of our music style transfer experiments in support of the claim that autoencoders are capable of style transfer.

II. Proposed Methods

Our hypothesis is that image style transfer methods such as [4] [10] [11] can be effective for modeling the timbre in music while preserving the harmonic and melodic structure. These methods require rich features of the input signal and we propose using [3] as a feature extractor.

We first introduce JukeBox in II-A. Then we review four different methods we tried for style transfer in sections In section II-BII-C, II-D, II-E.

A. Jukebox

Jukebox is a model that generates music [3]. As shown in figure 1, the model uses a three level VQ-VAE architecture with different temporal resolutions. At each level, the raw input audio is encoded into latent vectors. Then, the vectors are quantized to the closest codebook vector. The decoder then takes the sequence of codebook vectors and reconstructs the audio.

In our work, we take advantage of the pretrained VQ-VAE to perform style transfer. In particular, given source and target audio signals $x, y \in \mathbb{R}^{1\times T}$ where T is temporal length of the signal, our aim is to generate a translation $x \to y$ that preserves the 'content' of x and adopts the 'style' from y.

To that end, we first extract deep features of x and y from a state-of-the-art autoencoder trained on musical signals, JukeBox. We choose a layer inside JukeBox and split it into two parts: $F_{pre} : \mathbb{R}^{1\times T} \to \mathbb{R}^{C\times T'}$ and $F_{post} : \mathbb{R}^{C\times T'} \to \mathbb{R}^{1\times T}$ where C denotes the number of feature channels and T' is the temporal dimension of the deep features. F_{pre} then is used to extract features $h_x, h_y \in \mathbb{R}^{C\times T'}$

$$h_s = F_{pre}(s) \tag{1}$$

where $s \in \{x, y\}$ is the input signal to JukeBox.

Extracted features h_s can be directly decoded to reconstruct the input audio signal $\hat{s} \sim F_{post}(F_{pre}(s))$, and are then used to reconstruct the input audio signal $\hat{s} \sim F_{post}(F_{pre}(s))$

In our work, we propose a transforming the deep features in order to perform style transfer. We denote our transformation function as $F_{trans} : (\mathbb{R}^{C\times T'}, \mathbb{R}^{C\times T'}) \to \mathbb{R}^{C\times T'}$ and the transformed features as $h_{x\to y}$.

$$h_{x\to y} = F_{trans}(h_x, h_y) \tag{2}$$

Finally, we obtain the translation as

$$x \to y = F_{post}(h_{x\to y}) \tag{3}$$

In the next sections, we describe different transformation functions we used.

B. Style Transfer using the VQ-VAE codebook

In this section we describe our simplest method that was used as a baseline model. The encoder first transforms the input audio into latent vectors h_s and then the vectors are quantized to the closest codebook vector e_s. The intuition behind this first method is to quantize the latent vectors from x (content) using only the unique vectors from y (style).

First, we obtain the latent vectors from x and y (h_x and h_y). For y we quantize the vectors using the full codebook to obtain e_y and for x we quantize the vectors using only the unique vectors from e_y. We implement F_{trans} as $F_{codebook}$ in the following manner:

$$F_{codebook}(h_x, h_y) = Codebook_{ey}(h_x) \tag{4}$$

Where $Codebook_{ey}$ is the codebook restricted to the quantized vectors e_y obtained by quantizing h_y.

[4]We use style and timbre interchangeably in this report.

[5]In the supplementary we show that it is possible to perform image style transfer using image autoencoders. This is an encouraging fact since the autoencoder might not have extracted meaningful features for style transfer.

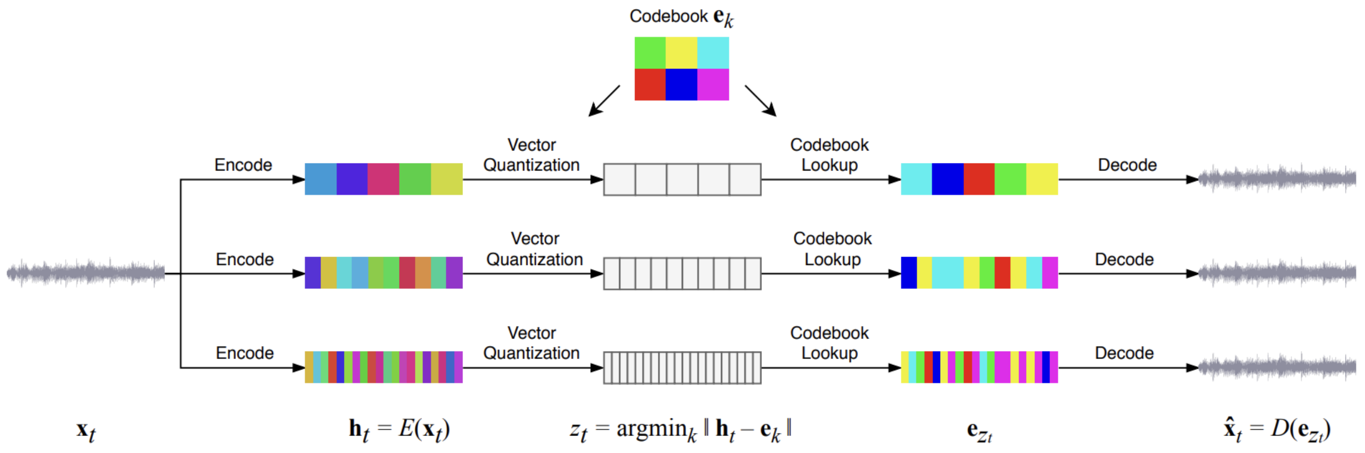

Fig. 1. Jukebox architecture

The translation using the $Codebook_{ey}$ is defined as:

$$x \xrightarrow{codebook} y = F_{post}(F_{codebook}(F_{pre}(x), F_{pre}(y))) \quad (5)$$

$$= F_{post}(F_{codebook}(h_x, h_y)) \quad (6)$$

C. Style Transfer using Optimization

D. Style Transfer using AdaIN

In this section, we describe how we use AdaIN from [10] to implement F_{trans} as F_{AdaIN}.

$$F_{AdaIN}(h_x, h_y) = \sigma_{h_y} \frac{h_x - \mu_{h_x}}{\sigma_{h_x}} + \mu_y \quad (7)$$

where $\mu_{h_s}, \sigma_{h_s} \in \mathbb{R}^{C \times 1}$ temporal average and standard deviation of h_s respectively.

The translation using AdaIN is defined as

$$x \xrightarrow{AdaIN} y = F_{post}(F_{AdaIN}(F_{pre}(x), F_{pre}(y))) \quad (8)$$

$$= F_{post}(F_{AdaIN}(h_x, h_y)) \quad (9)$$

AdaIN preserves the temporal information of the source, 'content', signal but changes the relative importance and mean of the channels according to the target, 'style', statistics. Compared to Gram matrix based approach in [4], AdaIN is computationally more efficient as it does not require per image optimization. On the other hand, AdaIN only acts on the diagonal of the Gram matrix, hence, it might have less control over the output image. We refer the readers to [10] for a more detailed analysis on AdaIN.

E. Style Transfer using WCT

We then use WCT to implement F_{trans} as F_{WCT}. We first extract the normalized features as $\hat{h}_s = h_s - \mu_{h_s}$. Then we apply eigen-decomposition to the normalized features

$$\frac{1}{T'} \hat{h}_s \hat{h}_s^T = E_s A_s E_s^T \quad (10)$$

We perform the whitening on the source features h_x to obtain an uncorrelated feature map.

$$h_x^{white} = E_x A_x^{-1/2} E_x^T \hat{h}_x \quad (11)$$

We then colorize h_x^{white} with $\hat{h}_y$

$$h_x^{colored_y} = E_y A_y^{1/2} E_y^T h_x^{white} \quad (12)$$

Finally, we add the mean of h_y to the colored features

$$h_{x \to y} = h_x^{colored_y} + \mu_y \quad (13)$$

The transformation function for WCT then becomes

$$F_{WCT}(h_x, h_y) = E_y A_y^{1/2} E_y^T (E_x A_x^{-1/2} E_x^T (h_x - \mu_{h_x})) + \mu_{h_y} \quad (14)$$

Notice that similar to AdaIN, we normalized the features along the temporal dimension. However, WCT applies whitening and coloring transforms instead of the standardization used in AdaIN. In that sense, WCT is a stronger transformation compared to AdaIN. Applying stronger transforms is advantageous as it brings the feature statistics of the translation and target signals closer to each other. On the other hand, stronger test time transformations create discrepancy between testing and training, which might result in less realistic outputs. For further analysis on WCT, we refer the readers to [11]

III. Experiments

A. Dataset

In order to evaluate the methods presented in this paper, we select 3 different audio samples interpreted by 2 to 6 different instruments. Specifically, we experiment on 2 *Moonlight* samples (piano, guitar), 6 samples of the note *C4* (flute, piano, sawtooth, sine, trumpet, violin), and 2 non-matching samples of music interpreted by a banjo and an organ.

B. Experiment 1

In the first experiment, the *source* and *style* audio samples both interpret the same note (C4), but are played by different instruments (flute, piano, sawtooth, sine wave, trumpet and violin). Each of those 6 instruments is in turn source and style which results in 30 tests for each of our methods.

C. Experiment 2

We raise the difficulty of Experiment 1's set up by using more complex source and target music pieces using the *Moonlight* dataset, played by a guitar or a piano. This experiment results in 2 tests for each method: from guitar to piano, and from piano to guitar.

D. Experiment 3

The last experiment consists in giving a source and target audio samples which do not interpret the same music piece, and which are played by different instruments. Specifically, we use xxx (banjo) and yyy (organ) to each be target and style file which results in 2 samples per method.

IV. Quantitative and Qualitative Analysis

Objectively quantifying whether a generated sample is closer in timbre to its *source* or *style* sample poses several challenges. Indeed, the three audio pieces may not be aligned, may not be played at the same pace or may not be interpreted in the same way which means that classic losses such as *mse*, *l2 norm* may not work to assess the timbre-distance between the files. Additionally, we do not have any ground truth about what the synthetic audio files should sound like, as similar instruments can sound very different based on its maker, its material and a plethora of other criteria. This is why a relative metric rather than an absolute one is more suitable in our particular case. We analyze each of the generated samples quantitatively in order to analyze whether the timbre-transferred files are closer to the *style* or *source* sample. Because timbre

V. Discussion

References

[1] Ondřej Cífka, Alexey Ozerov, Umut Şimşekli, and Gael Richard. Self-supervised vq-vae for one-shot music style transfer. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 96–100. IEEE, 2021.

[2] Ondrej Cifka, Alexey Ozerov, Umut Simsekli, and Gael Richard. Self-supervised VQ-VAE for one-shot music style transfer. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, jun 2021.

[3] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[5] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[6] Eric Grinstein, Ngoc Q. K. Duong, Alexey Ozerov, and Patrick Pérez. Audio style transfer. *CoRR*, abs/1710.11385, 2017.

[7] Eric Grinstein, Ngoc QK Duong, Alexey Ozerov, and Patrick Pérez. Audio style transfer. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 586–590. IEEE, 2018.

[8] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B Grosse. Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer. *arXiv preprint arXiv:1811.09620*, 2018.

[9] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B. Grosse. Timbretron: A wavenet(cyclegan(cqt(audio))) pipeline for musical timbre transfer. *CoRR*, abs/1811.09620, 2018.

[10] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[11] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. *CoRR*, abs/1705.08086, 2017.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[13] Kıvanç Tatar, Daniel Bisig, and Philippe Pasquier. Latent timbre synthesis. *Neural Computing and Applications*, 33(1):67–84, oct 2020.