# Neural Style Transfer for Audio

15-780 Graduate Artificial Intelligence

Gautam Arakalgud, Poojita Thukral, Prithvishankar Srinivasan
{garakalg, pthukral, prithvis}@andrew.cmu.edu

May 10, 2017

## 1 Abstract

In fine art, humans have, over the years, mastered the skill to create new and unique pieces of art through composing a complex interplay between existing creations. In the domain of music, remixing is a common technique to create new, contemporary music from older ones. Unfortunately an algorithmic basis for this creativity is unknown and machines have been unable to emulate this skill. Gatys et al. [2] have recently demonstrated that deep neural networks are capable of separating the content and style of multiple images and combining them into a single image. In our work, we focus on extending texture synthesis and style transfer techniques, that are used for images/videos, to the field of sound. Our approach is inspired by the methods suggested by Gatys et al. [3] for texture synthesis. For style transfer we followed the work outlined by Gatys et al. [2]. Given an original audio file, and a secondary audio file for style, we try to create the original audio in the style of the second track.

## 2 Introduction

Convolutional neural networks (CNN) are a biologically inspired class of deep neural networks that have been shown to work very well on a variety of practical applications in vision, audio, language processing, etc. In these networks individual networks are trained to respond to a restricted region of space known as their receptive field by performing a convolution operation. Each layer in a CNN can be trained to extract specific information that is further processed in subsequent layers. In this respect each layer behaves as an audio filter to the original data. Hence the activations of these layers behave as feature maps of the original image.

Convolutional neural networks trained on music tagging tasks seem to develop an abstract representation of the audio data that is passed through them. As we move deeper into the network this representation becomes more explicit and the network increasingly cares about the content of the data. It looks like the feature maps from these layers preserve the essential frequencies of the original signal without constraining their amplitudes.

The style or texture of an audio signal can be thought of as spatial correlations between feature maps of the frequency spectrum. To represent style, we do not really care about the specific frequencies, amplitudes and scale that make up the signal, but instead about the correlations between these frequencies. Hence by calculating gram matrices on the filter responses of a layer, we get a representation of style. By including filter responses over multiple layers, we obtain a more long-range, scale-invariant style representation.

Work on style transfer has led to the creation of popular applications like Prisma, Google's deep dream project, etc. This has largely been the motivation for this project in that we wanted to consider the possibility of extending the techniques of style transfer to audio. We also spent a substantial amount of time in class learning different machine learning algorithms, including deep learning, and chose to extend these techniques to the popular and under-examined tasks of audio synthesis and style transfer. In our project we focus on both audio synthesis and style transfer in audio. We try to combine the style and content of two different audios and generate an output that is a complex interplay between the 2 audio sources. Section 3 talks about some related work that this project is based on. Section 4 details the approach we took to solve this problem, and finally, Section 5 showcases some interesting results that we produced.

# 3 Related Work

Gatys et al. [2] first demonstrated that it is possible to separate the content and style of images and then optimize a random image to include the content of one image while mimicking the style of another. The core functionality of that work is reproduced here, while making further optimizations.

Berger and Memisevic [1] introduced the idea of using gram matrices of feature maps to represent texture. They claim that gram matrices work because they capture global statistics of an image due to spatial averaging over features. Averaging over feature maps makes the gram matrices blind to the global arrangement of objects inside the image and captures textures in the image. Further, they go on to say that using gram matrices across multiple layers and averaging across them gives long range structure to this texture.

# 4 Approach

Our approach is outlined in the following steps —

1. Create a 16 layer convolutional neural network and train it on the Magna Tag-A-Tune dataset [5]. Also experiment with a randomly weighted neural network.

2. Perform a Short Term Fourier Transform on the input data to obtain a 2D frequency spectrum of the input audio signal.

3. Extract the content features of the *content track* from an experimentally determined layer in the network.

4. Extract the style features from the *style track* from multiple layers in the network (again, experimentally determined).

5. Define gradients and optimize a randomly initialized frequency spectrum to simultaneously match the content and style features of both audio tracks.

6. Convert the resulting frequency spectrum to an audio track by performing an Inverse Short Term Fourier Transform.

These steps are described in detail in further sections.

## 4.1 Neural Network architecture

Johnson et al. [4] use a pre-trained VGG16 [6] network that performs $3 \times 3$ convolutions, for their style transfer problem. The network they use is shown below —

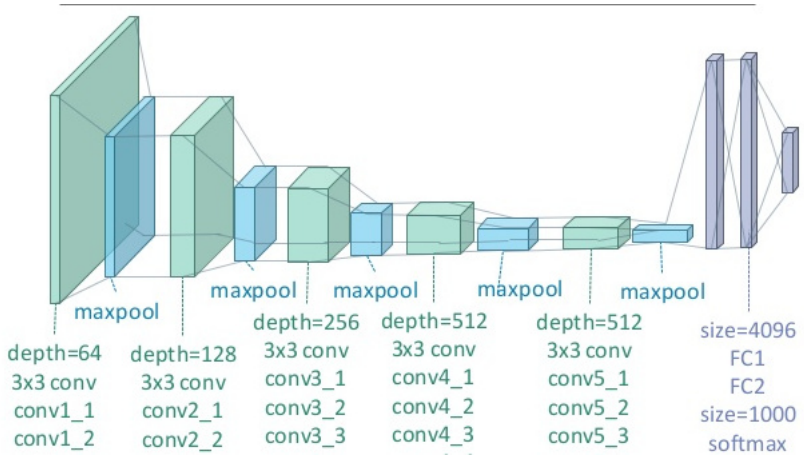


Figure 1: VGG16 network architecture

Since audio signals are 1D in nature, we use a similar network that performs $1 \times 3$ convolutions. Interestingly, we also experiment with randomly weighted neural nets and they seem to give us identical qualitative results.

## 4.2 Spectrogram

It is easier to work with audio data in the frequency domain and hence we perform an STFT over the input data. This works by dividing the raw audio sample into various overlapping chunks of equal length and computing an FFT over these chunks. Easy reproducibility of audio samples from STFTs motivated us to use it instead of other techniques such as wavelet transforms. Spectograms represent audio in a 2D format in terms of how frequency varies for different samples of the input. A sample spectrogram is plotted below —

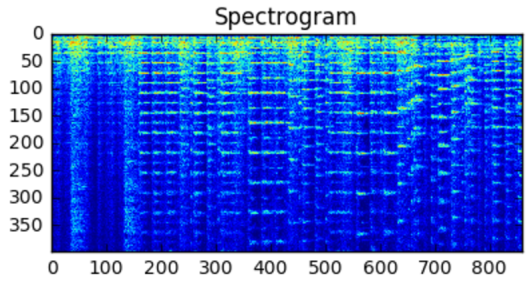


Figure 2: Spectrogram for Imperial.mp3

### 4.3 Encoding Content

Every layer in the CNN defines a non-linear filter-bank. We use the same notations as in the original Gatys paper [2]. An input spectrum $\overrightarrow{x}$ is encoded in each layer of the network by the activations or filter responses at that layer. A layer with $N_l$ different filters at layer $l$ has $N_l$ feature maps of size $(H \times W)$ $M_l$. Hence the filter responses of layer $l$ can be represented as a matrix of size $N_l \times M_l$. To visualize the information captured by layer $l$ in the network, we reconstruct the spectrum from the content representation of that layer. To do this, we perform gradient descent on a randomly initialized spectrum to produce one that matches the feature responses of the original image from layer $l$. Let $\overrightarrow{p}$ and $\overrightarrow{x}$ be the original spectrum and the spectrum that is generated respectively, and let $P_l$ and $F_l$ be the respective feature responses. The squared error loss between the feature responses is defined as

$$\mathcal{L}_{content}(\overrightarrow{p}, \overrightarrow{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

The derivative of the loss with respect to the activations in layer $l$ is given by

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^l} = \begin{cases} F_{ij}^l - P_{ij}^l & \text{if } F_{ij}^l \geq 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases}$$

The gradient with respect to image $\overrightarrow{x}$ can be calculated by standard backpropagation. Over several epochs, we optimize the random spectrum to produce the same filter responses as the *content* spectrum.

### 4.4 Encoding style

Style or texture information is defined as the correlations between filter responses. As stated in [1], filter correlations are given by the gram matrix $G_l$. If the filter responses from layer $l$ are represented by a $N_l \times M_l$ matrix as above, the gram matrix is the inner product of these activations.

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

The style loss is again defined as the squared error between the gram matrices of the style image and the optimized image. Let $\overrightarrow{a}$ and $\overrightarrow{x}$ be the original style spectrum and the

spectrum that is optimized respectively and $A^l$ and $G^l$ be their style representations, the style loss for each layer is given by —

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{ij} (G_{ij}^l - A_{ij}^l)^2$$

We take filter responses over multiple layers to obtain a multi-scale style representation of the spectrum. Hence our final loss function is given by —

$$\mathcal{L}_{style}(\overrightarrow{a}, \overrightarrow{x}) = \sum_{l=0}^{L} w_l E_l$$

The derivative of $E_l$ with respect to the filter responses in that layer is given by —

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} (F_l^T (G^l - A^l))_{ji} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l \leq 0 \end{cases}$$

The gradient with respect to the input spectrum $\overrightarrow{x}$ can be computed by standard back-propagation.

# 5 Results

The results of our project are available at www.tinyurl.com/StyleTransferForAudio

# 6 Conclusion

The crux of this project was to achieve considerable results for audio style transfer which is appealing to the human ear. The quantitative way of analyzing the network is to calculate the style and content loss but it doesn't give a comprehensive idea about the output. We noticed that the choice of songs plays a vital role in the quality of output as shown in Sample 2 in the results folder.
Factors such as dissimilarity of the music, smoothness of the style file have adverse effect on the output generated.

# 7 Future Work

Johnson et al. [4] introduced the clever idea of training a feed forward neural network for image transformation. They used perceptual loss functions as shown by Gatys et al. [2] to train a feed forward neural network to apply style to an input image. This approximation greatly speeds up the process of style transfer enabling applications such as real-time style transfer to videos. This can be extended to our project where we train a feed-forward network to apply style to input audio tracks.

# Bibliography


[1] G. Berger and R. Memisevic. Incorporating long-range consistency in cnn-based texture generation. *CoRR*, abs/1606.01286, 2016. URL http://arxiv.org/abs/1606.01286.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, . URL https://arxiv.org/pdf/1508.06576.pdf.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks, . URL https://arxiv.org/pdf/1505.07376.pdf.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*, pages 694–711. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46475-6. doi: 10.1007/978-3-319-46475-6_43. URL http://dx.doi.org/10.1007/978-3-319-46475-6_43.

[5] Edith Law, Kris West, Michael Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games : The case of music tagging. In *In Proc. wISMIR 2009*.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.