GENERATING AUDIO MIXTURES USING DEEP CONVOLUTIONAL NEURAL NETWORKS

by

Ariel Herbert-Voss

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

From the

School of Computing

Approved:

_____	_____
Suresh Venkatasubramanian, PhD Thesis Faculty Supervisor	Ross Whitaker, PhD Chair, School of Computing
_____	_____
Erin Parker, PhD Honors Faculty Advisor	Sylvia D. Torti, PhD Dean, Honors College

May 2016

ABSTRACT

Deep neural networks have recently been used in a generative capacity to separate and convolve the content and style of two input images. This is done using a joint cost function during gradient descent that encodes information about style and content to iteratively calculate forward node activations. We extend this methodology to the auditory domain using sound clips converted to spectrograms using the short-time Fourier transform and discuss optimizing signal reconstruction.

TABLE OF CONTENTS

INTRODUCTION

Although computers have come a long way in matching and even beating human performance on a variety of tasks, being able to produce unique creative musical content is an area in which humans still dominate. With the recent advances in machine learning afforded by deep neural networks, the creative gap between humans and computers is now starting to shrink. This project leverages the power of deep neural network models for the task of generating audio mixtures that represent cover songs. We do this by modifying an approach developed for synthesizing images by separating and recombining the content and style of arbitrary input images [1]. We extend this approach to the auditory domain by operating over spectrogram images of input audio samples.

PREVIOUS WORK

Deep neural networks are a class of machine learning models which are used to approximate non-linear functions for large amounts of data. Loosely inspired by biological networks in the brain, they are comprised of nodal units called "neurons" that perform operations in parallel. Illustrated in Figure 1, these networks consist of layers of neurons, the outputs of which are fed forward into subsequent layers according to the strength of an activation signal modulated by the connecting weights for each neuron. The activation function can be mathematically represented by:

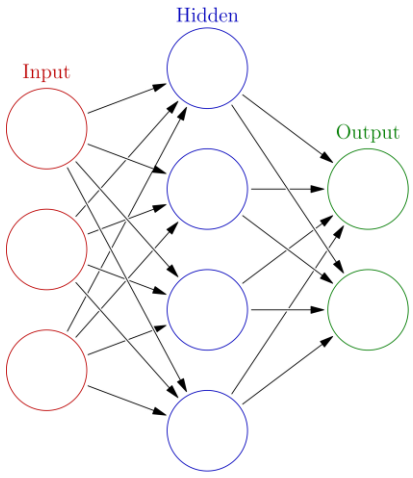


Figure 1: basic structure of a neural network. Source: Wikimedia

$$Y = \sigma(Wx + b)$$

where Y is the neuron output for the next layer, W is the connection weight and b is a bias value, and x is the neuron input. This function is sigmoidal because it has been shown that a neural network with two layers that uses non-linear activation functions is a universal function approximator [2], allowing for the application of neural networks to a broader class of problems.

The term "deep" refers to the network containing many hidden layers—layers that are neither inputs nor outputs. These layers serve to modulate the output according to the function being learned. To obtain the desired output the connection weights are adjusted during a "training" phase using an algorithm such as error backpropogation in conjunction with gradient descent, allowing the network to learn the appropriate internal representations for any arbitrary function of inputs and outputs [3]. The algorithm calculates the gradient of a cost function with respect to all weights and biases in the network, which can be expressed mathematically as

$$\frac{\partial C}{\partial W_{jk}^{l}} \text{ and } \frac{\partial C}{\partial b_{j}^{l}}$$

where C is the cost function, W_{jk}^{l} represents the weight on the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer, and b_{j}^{l} represents the bias on the j^{th} neuron in the l^{th} layer. This is then fed into a gradient descent procedure which updates the weights while simultaneously minimizing the cost function [4].

For this project we are focused on a subclass of deep learning models called "convolutional neural networks" (CNNs). These models are used most commonly for object recognition and other tasks that can benefit from spatial correlations in the input in part because they were designed loosely around the human visual system. Human visual cortex contains neurons specifically designed to detect light in small overlapping subsets

of the visual field called receptive fields [5]. As illustrated in Figure 2, a CNN is organized with multiple hidden layers consisting of small batches of neurons similarly

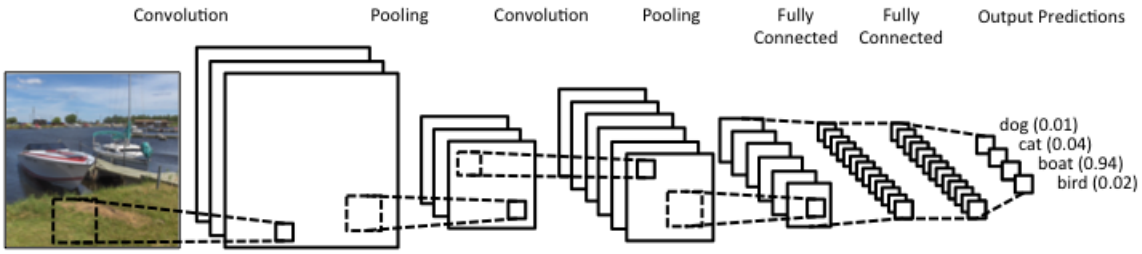


Figure 2: organization of a CNN optimized for object recognition. Source: Clarifai.

called receptive fields which process subsets of the input image. As an example to illustrate how using receptive fields impacts the size of the layers, if the input image is 28×28 neurons (where each neuron is an RGB element) and the receptive field is 5×5 neurons, then the next layer will be 24×24 neurons because we can only move the receptive field 23 neurons across or down before colliding with the edge of the input image[1]. The first convolutional layer consists of running a set of convolutional kernels (the operation of which is described in Figure 3) over the input image to produce a corresponding set of feature maps, which are mappings from the input layer to the next hidden layer. Each feature map is defined by its convolutional kernel and each can detect one type of feature across the entire image.

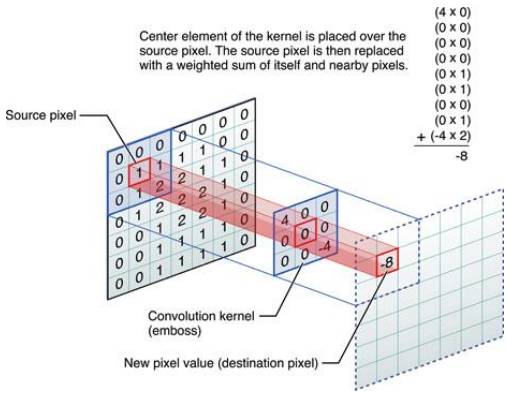


Figure 3: visual description of a convolution kernel moving over an image. Source: Apple

[1] This dimensional discrepancy can be rectified by padding the output to match the size of the input. However, it still means that some information is lost on the boundary of the image.

After each convolutional layer we utilize a pooling layer to reduce the size of the feature map before feeding the results into the next layer. This involves partitioning the feature map output into a set of non-overlapping subregions and downsampling by outputing the maximum value[2] of the subregion. For a 24×24 neuron feature map and a 2×2 subregion size, the output layer would be 12×12 neurons. Pooling also has the added effect of providing translational invariance to the network, which means that the CNN is less sensitive to variations in object positions in the input images. This property increases as we add more convolution and pooling layers because each higher layer sees a larger section of the input image.

Overall, the general structure of a CNN has the lower layers alternating between convolution and pooling and the upper layers corresponding to the fully-connected graph illustrated in Figure 1. The input to the first fully-connected layer is the set of all feature maps computed in the previous layer [6]. For CNNs performing object recognition, the number of neurons in the last fully-connected layer corresponds to the number of output categories.

With regards to auditory tasks, CNNs have found successful application to genre classification and speech recognition. In addition to being effective at extracting features from RGB images, CNNs are particularly adept at modeling complex correlations in speech and music features when operating over spectrogram images. For speech in particular, the special structure of CNNs exhibit some degree of invariance to small shifts in speech features along the frequency axis [7], which is important for handling speaker and environmental variations. For genre classification tasks, CNNs have been used in

[2] This particular pooling procedure is called "max pooling", and may be substituted with an average or other operation.

combination with mel-spectrograms to extract features from music samples and classify the samples according to these features [8]. This approach has also been extended for use in automatic music recommendation systems [9].

METHODS

To generate audio mixtures, we modified a technique previously developed for synthesizing images by mixing content and style from different sources to work on spectrogram images. In this section, we will discuss the original style rendering model and our method of generating spectrograms to allow for audio reconstruction following synthesis with the CNN. We also introduce our experiments to reduce data loss during reconstruction of the audio signal following synthesis.

STYLE RENDERING MODEL

The original technique of rendering one image in the style of another using CNNs involves obtaining the content representation from the image to be rendered and the style representation from the other image lending its style, as these two properties are separable and thus can be manipulated independently [1]. This is achieved by encoding this representation information in the cost function used to perform gradient descent on a white noise image[3] that will iteratively be modified to match the feature responses of the original images. At a high level, we express this cost function as a linear combination of two smaller cost functions:

$$\mathcal{L}_{total}(\overrightarrow{im_1}, \overrightarrow{im_2}, \overrightarrow{out}) = \alpha \mathcal{L}_{content}(\overrightarrow{im_1}, \overrightarrow{out}) + \beta \mathcal{L}_{style}(\overrightarrow{im_2}, \overrightarrow{out})$$

[3] This image is used as an initialization for the final rendering image, and thus begins as an RGB image with the pixels initialized using white noise.

where $\overrightarrow{im_1}$ and $\overrightarrow{im_2}$ are the input images for synthesis and $\overrightarrow{out}$ is the initial white noise image that will be iteratively changed to match the representation responses. The coefficients α and β weight the respective contribution of content and style representation on the reconstruction.

It is important to note that this is performed using a CNN that has been previously trained on an object recognition task because it contains a representation of the image that when propagating through the net makes object information increasingly explicit along the processing hierarchy [10].This means that higher layers in the network increasingly represent the content of the original input image in terms of objects and orientations rather than exact pixel values.

We represent the content information in a given layer *l* as F_{ij}^{l}, which represents the activation at the j^{th} neuron from a given kernel *i*. From this we can compute the cost term for content in the total cost function as follows:

$$\mathcal{L}_{content}\left(\overrightarrow{im_1}, \overrightarrow{out}\right) = \frac{1}{2}\sum_{i,j,l}\left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

This is the squared error loss between the feature representations *F* from one of the original input images $\overrightarrow{im_1}$ and the feature representations *P* from the white noise image $\overrightarrow{out}$.

In addition to the content representation, we construct the style representation by computing a spatial summary statistic on the feature map activations for the second input image [11]. This statistic is the correlations between feature responses in each layer of the network. Using the feature correlations for multiple layers gives us a multiscale representation of the input image and allows us to capture its texture information but not

the global arrangement [10]. We formalize these feature correlations for a given layer *l* as the inner product of the feature map F^l, which represents the activation at the j^{th} neuron from a given kernel *i*. This can be stated mathematically as the Gram matrix for F^l:

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

We compute the cost term for style as follows:

$$\mathcal{L}_{style}(\overrightarrow{im_2}, \overrightarrow{out}) = \sum_l w_l \left(\frac{1}{4N_l^2 M_l^2} \sum_{ij} \left(G_{ij}^l - S_{ij}^l\right)^2 \right)$$

where N_l is a feature map and M_l is the height times the width of the feature map. This is the mean-squared distance between the correlation responses *G* of the original image $\overrightarrow{im_2}$ and the responses *S* of the white noise image $\overrightarrow{out}$ for each layer. We then compute a weighted sum over this distance for each layer where w_l is the weight of the contribution of a given layer.

SPECTROGRAM GENERATION AND SIGNAL RECONSTRUCTION

To modify the style rendering model to work for synthesizing audio, we feed in spectrogram images of the audio samples we aim to mix. As seen in Figure 4, spectrograms are visual representations of the frequency spectrum for an audio signal as it varies over time and can be thought of as a two-dimensional histogram where each pixel value represents binned power; the x-axis represents time and the y-axis represents frequency. To generate spectrograms for the input audio

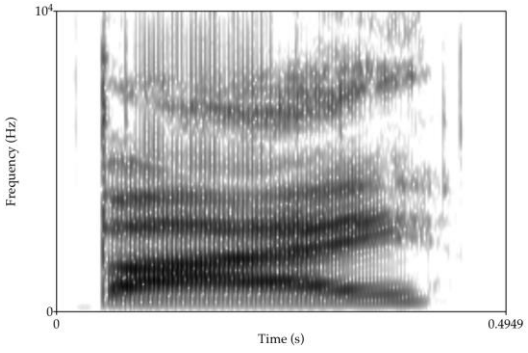


Figure 4: example of a spectrogram for a speech signal. Source: Wikimedia

samples we run a short-time Fourier transform (STFT) over the waveform, which gives us the Fourier coefficients representing the power at each frequency the number of coefficients—and hence frequency resolution—increases with the window size over which the STFT is computed. We can see a graphical representation of this operation in Figure 5. Mathematically, we can define the discrete STFT in terms of the FFT:

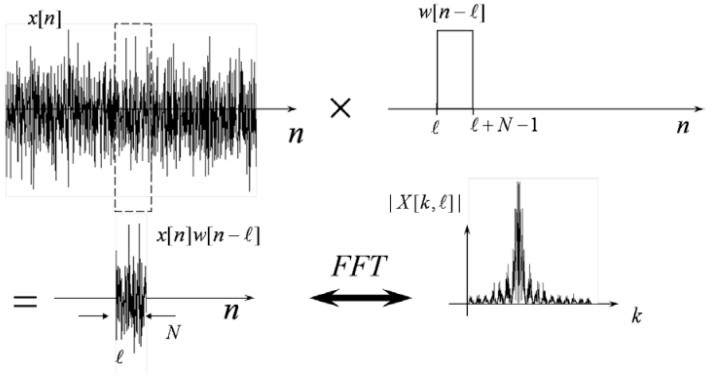


Figure 5: visual explanation of the STFT. Source: Naval Postgraduate School

$$X[k,l] = FFT\{x[l]w[0], \ldots, x[l+N-1]w[N-1]\}$$

where $x[n]$ is the signal, $w[n-l]$ is the window function, N is the window length, k is the frequency index, and l is the time index. To obtain the spectrogram, we then plot the magnitude of the STFT as a changing frequency spectra as a function of time.

In order to check the perceptual quality of the audio mixtures after processing the spectrograms through the CNN synthesis model, we need to obtain the waveform reconstruction from the resultant spectrogram. Because spectrograms contain no information regarding phase, it is impossible to obtain a perfect reconstruction. However, we can obtain an approximation by iteratively applying forward and inverse Fourier transforms to extract the phase from an estimation of the signal and combine it with the magnitude information in the spectrogram [12]. In essence, this finds

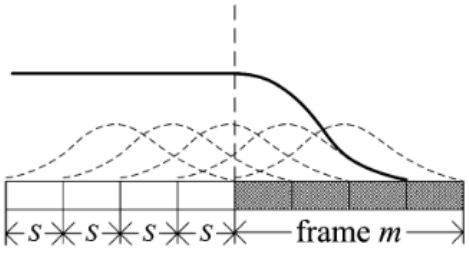


Figure 6: illustration of initial phase estimation using the overlap-added results of previous reconstruction frames. Source: IEEE

the waveform that has an STFT magnitude most like the input spectrogram. An important detail that we implemented in conjunction with the phase estimation was computing the inversion with an informed initial phase estimate based on values in the partially reconstructed frames rather than starting each phase at zero [13]. This process is illustrated in Figure 6. The solid line represents the signal contour over time for the reconstructed signal. In order to reconstruct the signal in frame *m*, the results from the previous frames which overlap with *m* are added. This greatly reduces the number of iterations required to achieve a signal reconstruction by starting the process from a position much closer to a local error minimum, which can then be reached more quickly.

IMAGE REPRESENTATION EXPERIMENTS

Inverting spectrograms to obtain the reconstructed audio waveform is an inherently lossy process and impacts our ability to construct perceptually-sound audio mixtures. However, the data representation of the spectrograms we feed into the network has a direct impact on the quality of reconstruction. Therefore, we can investigate the effects of using different data representations to achieve minimal data loss and preserve as much perceptual integrity in the reconstruction as possible. We will test three different methods of representing the spectrograms in the RGB colorspace and delineate these methods in the following paragraphs.

Replication

The first method we tested was to simply replicate the spectrogram data across all three color channels as 32-bit floating point values rather than integers, which is acceptable because the CNN operates on floats and all spectrogram bin values fall between 0 and 255. This method results in each pixel containing three copies of the

spectrogram bin value, one for each channel. While it represents the simplest method to implement, this method also has some data redundancy as each neuron in a given convolution layer of the CNN maps to a single channel, effectively meaning that three neurons map to each spectrogram value.

Split Integer Scaling

The second method we tested was to improve the resolution over common spectrogram bin values in an effort to improve the resolution of the reconstruction. We can see from the histogram of average numerical spectrogram values in Figure 7 that the average song spectrogram contains a very large number of values less than 10. We compose a nonlinear scaling function that transforms the 32-bit floating point values to a 24-bit integer representation which emphasizes the small float values and compresses the more disparate large values into a smaller part of the 24-bit range. A plot of the conversion function can be seen in Figure 8. We then split the 24-bit integer into three 8-bit integers and use these to populate the RGB channels, as 8-bit integers are valued between 0 and 255 and represent standard RGB colors, as we

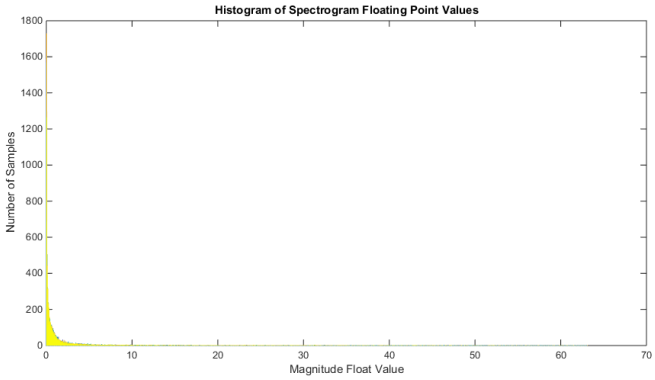


Figure 7: histogram of average numerical spectrogram values for all 14 songs.

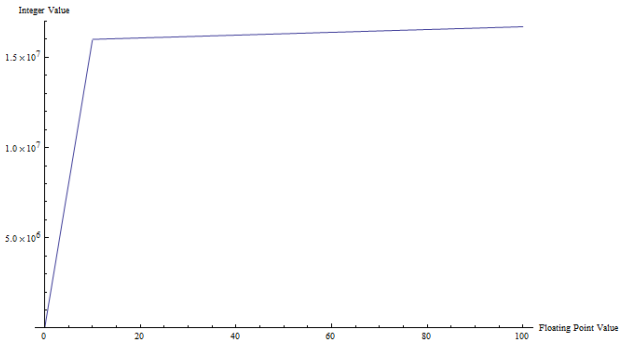


Figure 8: scaling function for conversion between spectrogram floating point and 24 bit integers.

also wanted our representation to mimic the structure of an RGB image without the data redundancy of the first method. As an example, given the floating point value of 0.7329, we use the scaling function to convert this to 1172640 for 24-bit representation. In binary, this is 0b000100011110010010100000, which we cut into three segments: 0b00010001, 0b11100100, and 0b10100000. This translates to the decimal values 160, 228, and 17, which we feed into the R, G, and B channels respectively.

Row Folding

The third method we tested was to preserve as much spatial locality as possible while fitting the memory constraints as closely as possible. We take each pixel row of the spectrogram and feed it successively into each channel of an RGB image. This is performed such that the first row is placed in the red channel, the second row is placed in the green channel, and so forth. This process, illustrated for the first three rows of an input spectrogram in Figure 9, is repeated over the whole image. The resulting RGB image is a third of the height of the original spectrogram, but retains all data of the original without redundancy. Additionally, this method preserves temporal locality and maintains the same memory footprint as the original spectrogram due to the neuron ordering in the CNN. Because each neuron represents a channel of a pixel, the representation of the resulting RGB image is functionally identical to that of the original spectrogram.

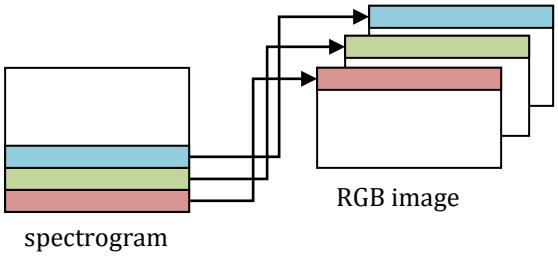


Figure 9: illustration of row folding method. Each row in the spectrogram is fed to a row channel in the RGB image.

IMPLEMENTATION DETAILS

We created seven audio mixtures from the following style and content combinations[4]:

- Pop singer Taylor Swift as metal band Gojira
- "Game of Thrones" TV show theme song as country singer Johnny Cash
- Japanese pop band Kalafina as American rock band Styx
- Punk band Green Day as electronic musician Oneohtrix Point Never
- Baroque composer Georg Friedrich Handel as Scrumpy and Western band The Wurzels
- Zebra finch bird song as classical composer Mozart
- Contemporary avant-garde composer Gyorgy Ligeti as shoegaze alternative rock band My Bloody Valentine

For generating the spectrograms from audio, we selected five-second long clips sampled at 44100 Hz from each song in a given pairing as inputs to the network, which each generate a spectrogram of size 2048x1740 in order to utilize the maximum amount of VRAM on the GPU. The vertical axis represents the number of Fourier coefficients and therefore the spectral resolution. The horizontal axis represents the windows of the STFT and therefore the temporal resolution. The window size is 1024 samples, with a frame step of 128 windows.

We used the VGG-Network architecture trained on an object recognition task [14]. From the model architecture, we used sixteen convolutional layers and five average-pooling layers and omitted the fully connected layers. We obtained the pretrained

[4] Details about the selected clips can be found in the Appendix.

network weights from the original authors and built the model in Python using the Theano and Lasagne neural network packages. We computed the spectrograms in MATLAB and then ported them onto a compute node on a GPU cluster for processing with the model. The utilized node has two K20 NVIDIA GPUs and two 16-core Intel Xeon CPUs, although our implementation only uses one GPU due to constraints within Theano's default settings. Additionally, Theano only utilizes frame buffer memory, which constrains us to 5 GB of VRAM. After obtaining the spectrogram representing the audio mixture, we inverted the spectrogram in MATLAB using the RTISI-LA algorithm [13] to obtain the reconstructed waveform.

To provide a relative error measure, we computed the similarity of each resulting reconstruction with the original content waveform using the cross-correlation. We will look at the average values for each method over all seven audio mixtures in comparison with each other. We expect to see relatively high values for reconstructions with high perceptual quality.

RESULTS

Here we discuss the results from our image representation experiments. Given that our results regarding perceptual integrity are largely qualitative, we will illustrate our findings by looking at a single mixture in detail. For this task we have selected Taylor Swift's "Shake It Off" rendered in the style of Gojira's "Where Dragons Dwell". The original spectrograms are shown in Figure 10. We selected this mixture because it most strongly illustrates the strengths and flaws in each representation, allowing us to get a clear sense of whether our audio synthesis method is effective.

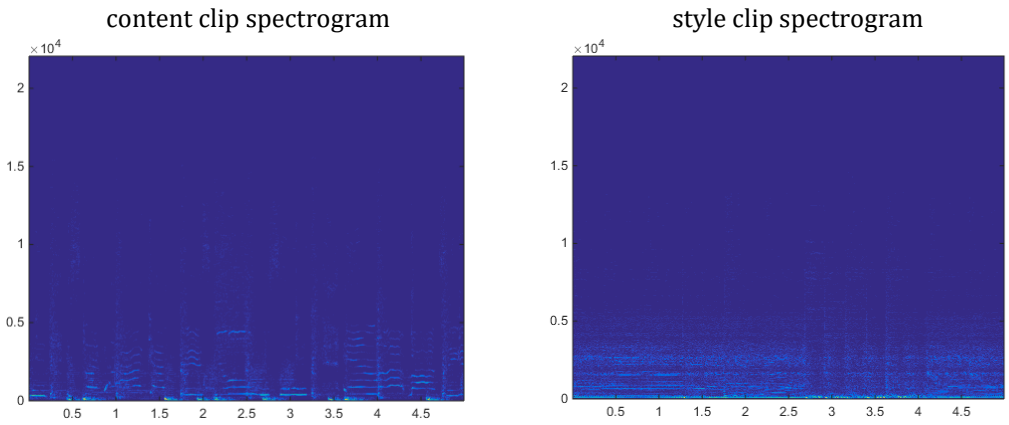


Figure 10: content – "Shake It Off"; style – "Where Dragons Dwell".

We can see in the spectrograms that "Shake It Off" has a very distinct rhythm, corresponding to roughly 160 BPM. We can also see vocal formant features. "Where Dragons Dwell" has a distinct smearing pattern attributed to heavy amplified guitar distortion and a portion of blast drum beats with high BPM starting around the 2.75 second mark. These two spectrograms indicate that the respective songs have very different auditory features, which we perceptually confirm by listening to the waveforms.

REPLICATION

This spectrogram representation method required three times as much memory as we had available, since each image would require 2048×1740×3 values, or roughly a gigabyte of data. The CNN has on average three convolution layers between each pooling layer, causing the GPU to quickly run out of memory during computation. To combat this issue we dropped the clip length from 5 seconds down to 2.5. On the spectrogram in Figure 11 we can see some formants organized in a rhythmic pattern, similar to the original content clip. The reconstructed audio is very noisy, but the original beat can be heard, as well as some faint vocal tonality. The average cross-correlation value for the audio mixtures using this method is 53.39 with a standard deviation of 0.23.

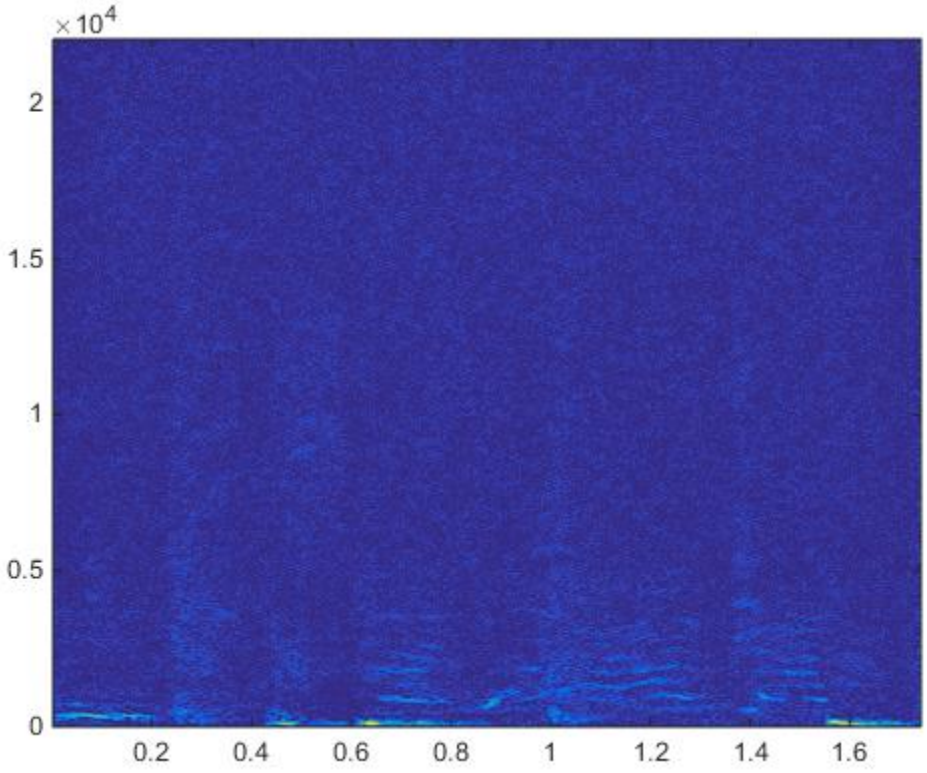


Figure 11: reconstructed spectrogram using replication representation method.

SPLIT INTEGER SCALING

This representation method suffered the same memory issues as the previous method because Theano converts the 8-bit integer color values to 32-bit floats for neural network operations. We thus used a 2.5 second clip size as before. The reconstructed audio sounds like uniformly-distributed white noise. Looking at the spectrogram in Figure 12 confirms that the mixture contains no recognizable features from the original content clip. The average cross-correlation value for the audio mixtures using this method is 44.49 with a standard deviation of 9.83.

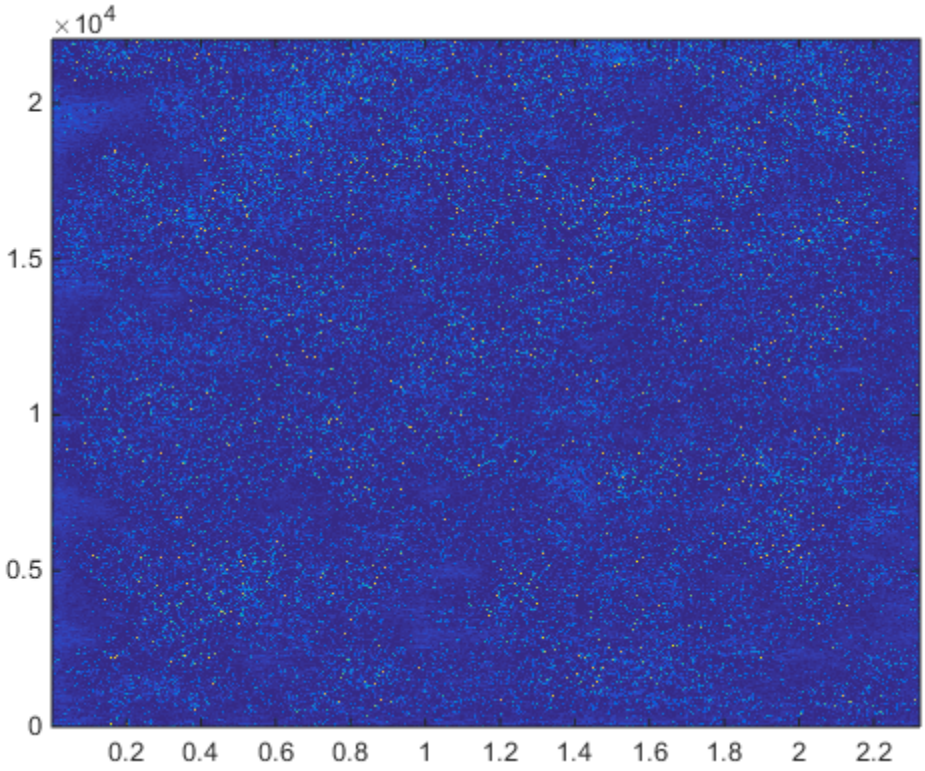


Figure 12: reconstructed spectrogram using split integer scaling representation method.

ROW FOLDING

This method matches the memory footprint of the original spectrogram and thus avoids the pitfalls of the previous methods, resulting in a final image that was the same size as the input. This allows us to examine a higher resolution sample of data. We see in the spectrogram in Figure 13 that the formants are organized in a rhythmic pattern, similar to the original content clip. However, the BPM appears to have increased and the overall spectrogram is noisier. The audio reconstruction contains a fair amount of noise, but we can also hear vocal tonality and enunciation, as well as the beat. The average cross-correlation value for the audio mixtures using this method is 54.97 with a standard deviation of 0.19.

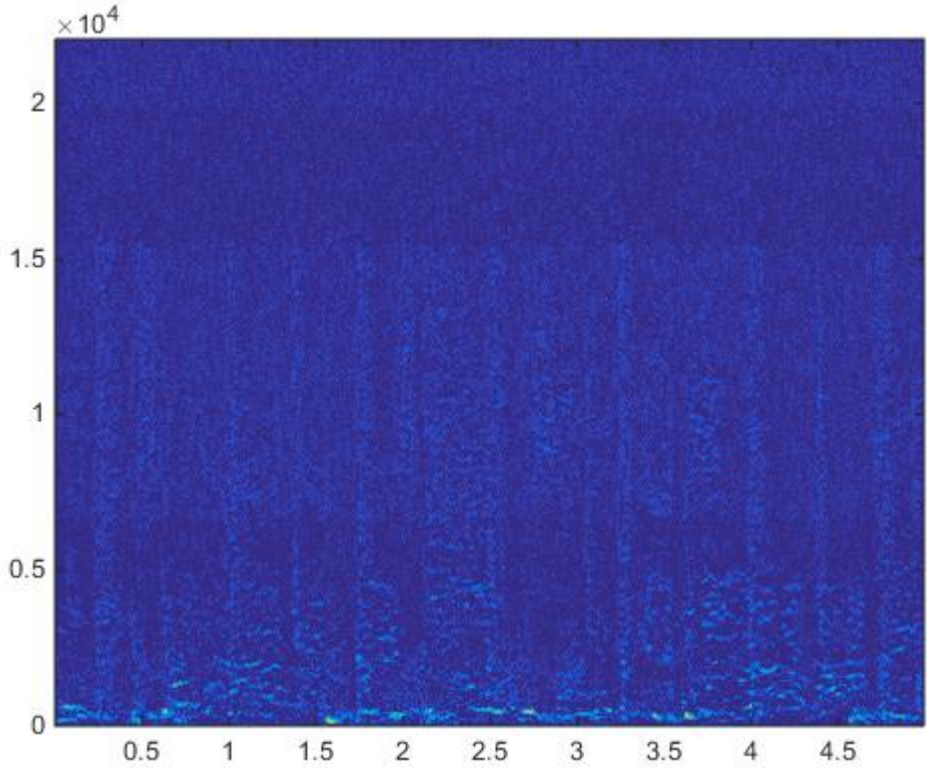


Figure 13: reconstructed spectrogram using row folding representation method.

DISCUSSION

The spectrogram reconstruction of the audio mixture that used the replication representation method produced a result with a relatively high average cross-correlation value. However, it suffered from data overrepresentation by a factor of three. Because our pretrained network architecture has a receptive field size of 3×3 pixels, this means that the receptive field is unable to see more than two different pixel values in each column at a time, and one value will always take up the majority of the field; this is illustrated in Figure 14. This bias makes it more difficult to extract features for content reconstruction.

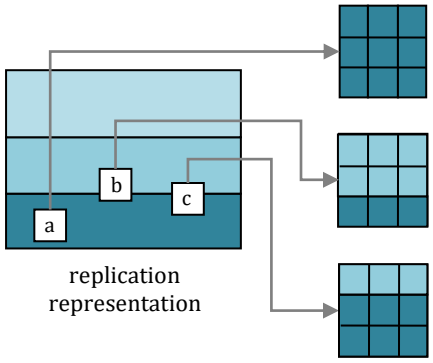


Figure 14: visual explanation of receptive field homogeneity problem. Fields b and c illustrate that one value will always be overrepresented because they are constrained to a size of 3×3.

The reconstruction using the split integer scaling representation performed poorly because the red channel contains the least significant bits of the 24-bit representation, which vary significantly but have less impact on the actual spectrogram value than the green and blue channel bits. The CNN considers all channels equally rather than giving more weight to channels holding more significant bits. This results in fluctuations in the red channel bleeding into the rest of the image, creating an overall noisy image with content and style information smeared out.

The reconstruction using the row folding representation had the highest perceptual quality and performed the best according to our relative error measure, in part due to the higher resolution afforded by the longer audio sample. Given that this method provides us

with the maximum amount of data available for processing, achieving greater perceptual quality is evidently limited fundamentally by our use of the pretrained network.

CONCLUSION

Our results loosely indicate that the methodology for synthesizing photos and paintings can be used for audio synthesis as well, as we can extract recognizable content features from the generated audio samples. However, even on the best reconstruction there was still significant noise which made it difficult to determine how much style was imbued, limiting our claim that this method is successful. Creating a more comprehensive error measure that informs on both content and style might provide us with more insight.

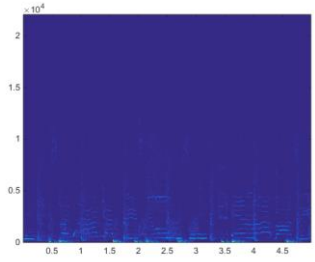
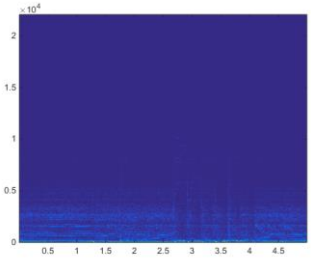
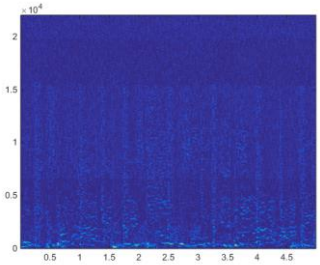
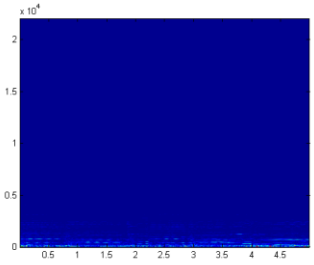
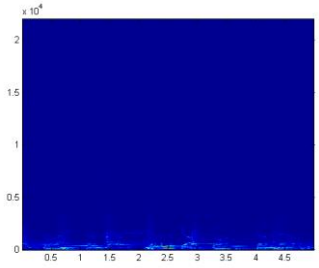
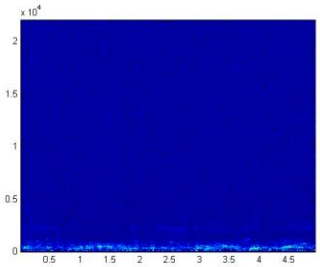
Additionally, the best audio reconstruction is still not fully recognizable by content and finer perceptual tasks such as singer and instrument identification remain impossible. Greater perceptual quality might be obtained with a new CNN architecture trained on spectrogram classification tasks, as it is possible that the relative feature sparsity in spectrograms is moderately incompatible with the representations in VGG. Ultimately, this project shows that to a limited extent, it is possible to extend the image style and content synthesis method to the auditory domain. It also highlights the importance of an invertible image representation of the waveform with respect to data processing considerations, and represents progress towards bridging the creative gap between humans and computers.

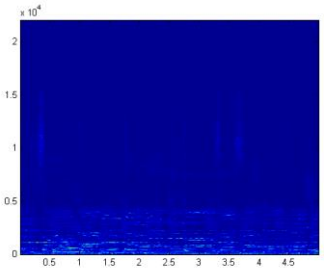
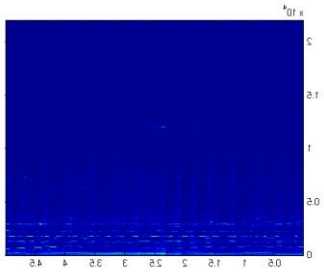
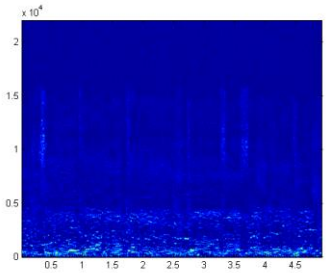
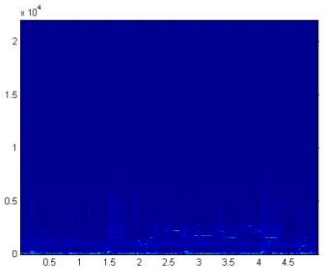
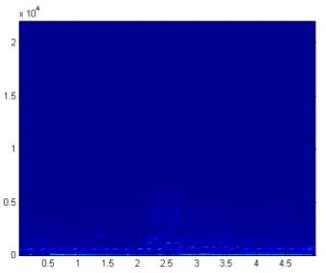
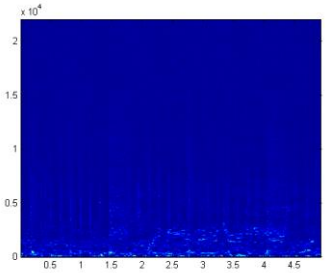
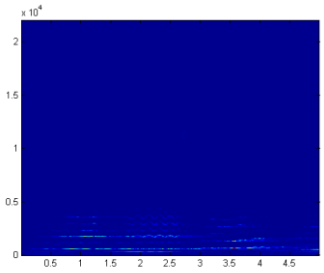
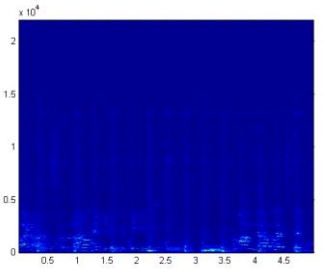
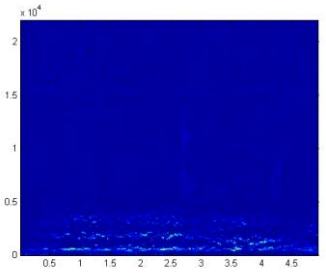
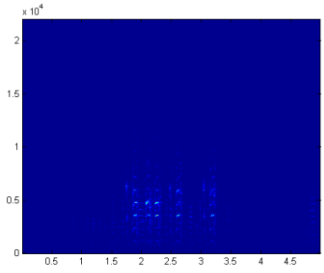
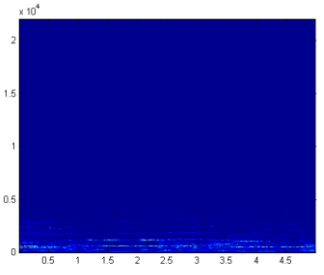
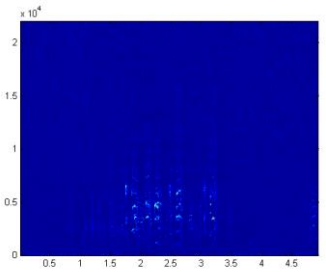
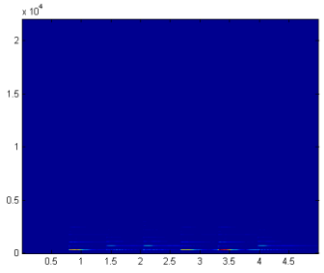
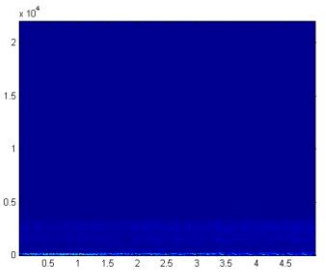
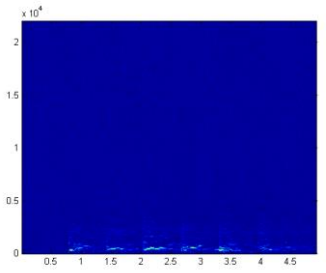
APPENDIX

DATASET DETAILS

Author	Title	Clip	Genre	Author	Title	Clip	Genre
Taylor Swift	Shake it Off	1:42-1:47 /4:01	American Pop	Gojira	Where Dragons Dwell	0:28-0:33 /6:55	Metal
Ramin Djawadi	Game of Thrones Theme	0:10-0:15 /1:41	Soundtrack	Johnny Cash	God's Gonna Cut You Down	1:33-1:38 /2:49	Country
Kalafina	To The Beginning	1:07-1:12 /4:23	Japanese Vocal Group	Styx	Fooling Yourself	0:58-1:03 /5:35	Rock
Green Day	American Idiot	2:11-2:16 /3:02	Punk Rock	Oneohtrix Point Never	I Bite Through It	2:02-2:07 /3:20	Electronic
G.F. Händel	Lascia ch'io pianga	0:47-0:52 /5:29	Baroque Soprano Aria	The Wurzels	Combine Harvester	1:09-1:14 /3:06	Scrumpy and Western
Zebra Finch	Male song	0:09-0:14 /2:45	Birdsong	W.A. Mozart	Requiem in D minor	30:13-30:18 /55:14	Classical
György Ligeti	Musica Ricercata II	0:00-0:05 /4:17	Avant-garde Piano	My Bloody Valentine	Sometimes	1:32-1:37 /5:23	Shoegaze

SPECTROGRAMS

Content	Style	Mixture
 Taylor Swift	 Gojira	
 Game Of Thrones Theme	 Johnny Cash	

		
Kalafina	Styx	
		
Green Day	Oneohtrix Point Never	
		
G.F. Händel	The Wurzels	
		
Zebra Finch	W.A. Mozart	
		
György Ligeti	My Bloody Valentine	

REFERENCES

[1] L. Gatys, A. Ecker and M. Bethge, "A Neural Algorithm of Aritistic Style," 26 August 2015. [Online]. Available: http://arxiv.org/abs/1508.06576. [Accessed 2 September 2015].

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems,* vol. 2, no. 4, pp. 303-314, 1989.

[3] M. Nielson, Neural Networks and Deep Learning, http://www.neuralnetsanddeeplearning.com, 2015.

[4] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[5] D. Yamins, H. Hong, C. Cadieu, E. Soloman, D. Seibert and J. DiCarlo, "Performance optimized hierarchical models predict neural responses in higher visual cortex," *PNAS,* vol. 111, no. 23, pp. 8619-8624, 2014.

[6] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Cambridge: MIT Press, 2016.

[7] O. Abdel-Hamid, A.-r. Mohamad, H. J. Li Deng, G. Penn and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 22, no. 10, 2014.

[8] Q. Kong, X. Feng and Y. Li, "Music genre classification using convolutional neural networks," in *International Society for Music Information Retrieval*, 2014.

[9] A. van den Oord, S. Dieleman and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, 2013.

[10] L. Gatys, A. Ecker and M. Bethge, "Texture synthesis and the controlled generation of natural stimuli using convolutional neural ntworks," 27 May 2015. [Online]. Available: http://arxiv.org/abs/1505.07376. [Accessed 16 February 2016].

[11] J. Portilla and E. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International Journal of Computer Vision,* vol. 40, no. 1, pp. 49-70, 200.

[12] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Transactions on Acoustics, Speech , and Signal Processing,* Vols. ASSP-32, no. 2, pp. 236-243, 1984.

[13] X. Zhu, G. Beauregard and L. Wyse, "Real-time signal estimation from modified short-time Fourier transform magnitude spectra," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 15, no. 5, pp. 1645-1653, 2007.

[14] K. Simonyan and A. Zisserman, 10 April 2015. [Online]. Available: http://arxiv.org/abs/1409.0575. [Accessed 15 September 2015].