# On the Generation of Music

Investigating Popular Modalities for Musical Representation, Classification and Generation using Deep Learning Techniques – a Final Report

Alejandro Cid, Alec Janowski, Bowen Jiang

---

**Abstract**

We introduce Lil Data, a novel approach to music generation relying on visual representations of sound. Our algorithm classifies spectrogram images of music using a convolution neural network. Once trained, we use this convolutional neural network to perform "neural genre transfer" (neural style transfer on spectrograms from different genres). Although we achieved high accuracy (above 95% for 4 genres) with a simple four-layer model, neither neural genre transfer nor LSTMs trained on single genres of music were able to generate pleasing music from spectrogram images. We share the results of our experiment in order to shed light on the advantages, disadvantages, difficulties and potentials of building deep networks on raw audio.

## 1 Introduction

The advent of recorded music marks the beginning of a musical arms race, where new genres and musical styles emerged at an ever-increasing rate alongside the development of new digital composition and production methods. Once the audio industry found a way to capture music in a digital format, producers and musicians gained the ability to manipulate recorded sounds and stitch them together to create increasingly complex compositions. We believe the next wave of music-creation tools will harness machine learning techniques to create intricate, mind-bending compositions. This paper will explore a few cutting edge deep-learning techniques for music encoding, classification and generation using visual as well as sequence models.

Our system consists of two algorithms. The first, Lil Data, is a convolutional neural network which takes a spectrogram image of a 5-second chunk of music as input and outputs a prediction of the given chunk of music's genre. The other is a Neural Style Transfer algorithm which uses Lil Data's weights to blend the spectrograms of two distinct pieces of music, in the hopes of transferring the genre of one song onto another song. After "genre transfer" has been performed on the spectrogram, we convert this spectrogram back to audio so we can listen to the output.

## 2 Related Work

Past methods for music generation relied heavily on MIDI data. Performance RNN[1] used an LSTM-based RNN to generate music, resulting in good local dependencies, but lacking in compositional structure. Magenta's Music Transformer[2] took this idea further, using an attention-based system to learn longer time dependencies than Performance RNN. OpenAI's previous attempt at music generation, MuSeNet[3], composes 'remixes' by riffing off of a few given notes to create a brand-new composition in that style.

Because the methods explored above rely on MIDI data to analyze or generate music, they fail to learn features of sound not represented in MIDI data, such as tone and timbre. This makes it impossible to generate the human voice and certain styles of highly dissonant styles of electronic music, such as drum and bass and dubstep. OpenAI's Jukebox[4], the most successful raw-audio generation algorithm to date, aims to resolve these problems by forsaking MIDI data in favor of learning features from raw-audio through the use of autoencoders.

Our idea for generation, which operates on spectrograms of music, stems from M. Dong's approach[5] to music classification, which relies on a specialized Convolutional Neural Network. We hoped to build a music generator off of a successful music classifier, by attempting Neural Style Transfer.

## 3 Dataset and Features

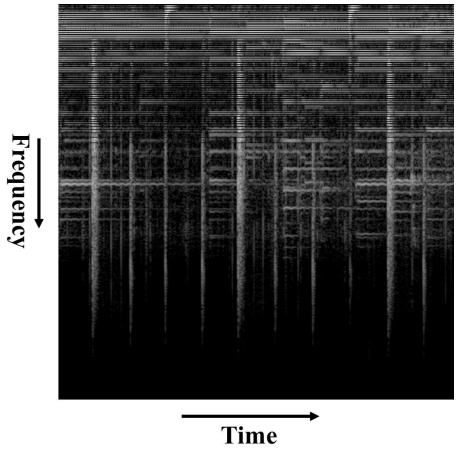


Figure 1: Spectrogram image taken from a 5-second clip of Lo-fi music

Our classifier, Lil Data, was trained on spectrograms taken from 5-second long clips of music from 8 genres—Trap, Future Bass, Lo-fi, Hardstyle, Young Gravy, Liquid Drum and Bass, House, and Noise/Non-Music. We had around 8000 training examples divided among these genres, with 1000 and 750 left over for validation and testing respectively. We extracted the spectrogram images using Librosa with 631 spectrogram features and 631 time steps, yielding inputs of size 631x631x1. We also normalized our input data using the built in Keras function for normalizing with ResNet 50.

## 4 Methods

Given the potential difficulty in working with time-series data, we elected to pursue a convolutional neural network (CNN)-based architecture for our genre classifier. These methods have been attempted before for similar audio classification tasks with success[5]. In our approach, we transform short clips of music into spectrograms, 2D images with dual axes of frequency and time, with pixel intensities corresponding to the amplitude of sound. We previously established that only short portions of audio are necessary for humans to perform genre classification with near-perfect accuracy and also noted the importance of time-based elements of music in this task, which suggest that if spectrograms can present visible differences for music of different genres, they may be excellent candidates for classification.

Our CNN model for classification was fairly simple. It consisted of a (631 x 631) input, 4 (3x3) Conv layers, 4 (4x4) Max Pooling layers, and one (48) Dense layer with an (8) output. This was trained for 20 epochs (after which accuracy was found to peak) on our set of 8 genres.

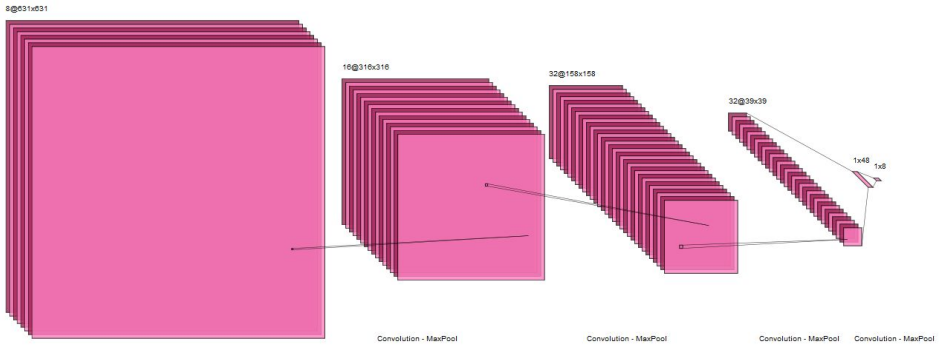


Figure 2: Architecture of Lil Data

After building a CNN music classifier, we then attempted to use it to generate new music via Neural Style Transfer (NST) on spectrogram images. NST works by using the activations of intermediate layers of a trained CNN to define the content and style of an image as well as the cost of a generated image; through comparisons to predefined style and content image activations, the generated image can be passed through the network and its pixels will be updated in gradient descent such that, at convergence, it will minimize style and content costs, preserving elements of both the style and content of the predefined images.

The cost functions for cost and style are detailed as follows:

$$J_{content}(C,G) = \frac{1}{2} |||\, a^{[l][C]} - a^{[l][G]} |||^2$$

$$J_{style}(S,G) = \sum_l \lambda^{[l]} \{ \frac{1}{(2n_h^{[l]} n_w^{[l]} n_c^{[l]})^2} \sum_k \sum_{k'} (G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)})^2 \}$$

$$J(G) = \alpha J_{content}(C,G) + \beta J_{style}(S,G)$$

We first attempted style transfer via transfer learning using a VGG-19 CNN pre-trained on ImageNet followed by Lil Data (supplementary data). For our experiments presented here, lofi and future bass were used for content and style, respectively, because they present very different spectrograms (future bass is much denser). We also experimented with Yung Gravy (a whole genre we included dedicated to the legendary rapper) for our own entertainment. The content image was used to initialize the generated image to maintain audio integrity and reduce the number of iterations needed.

We also attempted to generate music *de novo* using a variety of 50-hidden-unit LSTM networks of depths (number of LSTM layers) ranging from 2-4. 600 spectrograms of lofi were used to train a one-to-many model which takes in a single time step (631-by-1 vector) and return a corresponding 5-second-long spectrogram (631-by-630 matrix); lofi was selected as the genre because it is the sparsest and would hopefully yield the most realistic spectrograms. Training was conducted for 25 and 50 epochs in various experiments, as well as a 1000-epoch model which was trained on two training examples to understand the maximum fidelity that such a model could capture.

## 5 Experiments/Results/Discussion

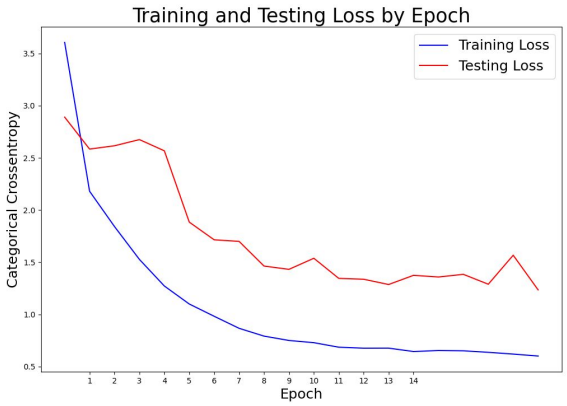


Figure 3: Training and validation loss per epoch for Lil Data

The results of our classifiers were a little surprising. Our simple CNN model was able to classify our 4 genre dataset with 96% accuracy and achieved 81% accuracy on our 8 genre set (in which several genres sound quite similar to the human ear). This is quite good, especially considering that many of the misclassifications we analyzed were between near identical genres such as DnB and Future Bass. We found that adding more layers did not necessarily help our performance either, and we also found that we needed to use a high dropout rate (50%) to prevent overfitting to specific songs within a genre.

Given the success of our relatively simple CNN, we expected solid results from our ResNet implementation. Surprisingly, however, the performance was quite poor. We achieved only 34% accuracy on the 8 genre test set, although the train set accuracy reached 94% fairly consistently. This was clearly extremely overfit, but we were unable to reduce this and improve our test set accuracy without significantly dropping the train accuracy to unacceptable levels.

While interpreting these results, we formulated several hypotheses as to why we observed such high accuracy on the simple network and such poor accuracy with the deep network. The first involves the input of our CNN when compared with ResNet. ResNet requires a 3 channel input (which we provided by duplicating our 1 channel image), however, spectrograms are only 1 channel. We believe that this may have led to many extra useless weights and unnecessary complexity within ResNet. Dealing with these extra weights likely hurt the accuracy and led to the overfitting we observed. We were able to avoid this with our CNN by using 1 channel inputs.

Our second hypothesis is that the relative simplicity and lack of noise within the music we chose allowed a simple model to pick up on features without as much difficulty as traditional music. Our genres were all electronic, computer generated, and as a result, the spectrograms contained no noise (such as that you might observe with traditional instruments in genres like rock). Additionally, the “notes” you might observe in our music were defined by clean and sharp lines, many of which were completely square waveforms. The well defined edges within our “images” were likely easy for a relatively simple network to pick up in a high layer, and we think this partly explains our performance.

### a. Music generation

Our neural genre transfer algorithms from both the VGG19 and the Lil Data architectures were able to demonstrate clear visual changes in style in only a few hundred iterations of gradient descent. The

VGG19 spectrograms became much brighter and lost clear "lines" of sound, becoming replaced instead with small checkered patterns of bright pixels. The Lil Data spectrograms also grew brighter with time, although this seems to be a more uniform brightness increase for all pixels; although the contrast of the resultant image decreased, there were still clear and sharp "lines" of sound, suggesting that the model trained on spectrogram data had better learned to preserve these local structures instead of replacing these sound patterns completely. This difference is also reflected in the audio quality of both resultant spectrograms; while the VGG19 audio is highly distorted, removing almost all harmonic components of the music and shifting the perceptible key of music down by around a quarter step, the Lil Data audio preserves clear percussion and instrumentation at a stable key (A Major) but only introduces a uniform "buzzing" frequency. The LSTM model, meant to generate a remix given a starting point, was also unable to pick up on individual sound components and resulted in distorted sounds without harmony.

The results of neural genre transfer were, however, still largely disappointing as no changes in rhythm, percussion or other instrumentation from the style audio were translated to the generated audio. Although visually, both algorithms seemed to converge toward a brighter and louder spectrogram, the added pixel intensities translated to indiscriminate noise in the resultant audio instead of specific changes. This seems to be because several instruments, when overlapped, could have various frequency components which also overlap, thus making it very difficult for the algorithm to apply "drums" or "vocal" components specifically to a generated image, rather than simply creating a pattern which might reflect the additive sound from several instruments. Additionally, certain patterns are very local to specific regions in the spectrogram (long lines at the top indicating bass; evenly-spaced vertical lines often indicating a beat); larger kernel sizes which cover more of an image or intermediate layer would not be able to capture these patterns as well and could lead to lower-resolution spectrograms, which translates to a "fuzzier" and less clear audio clip after inverting the spectrogram.

## 6 Conclusion/Future Work

Our convolutional neural network successfully classified between 8 genres of music. It was able to learn typical features of different genres from spectrogram data. If we had more time we would focus on reducing overfitting, as our algorithm consistently performed better on training sets.

Despite the relative success of our music genre classifier operating on spectrograms, our music generation was noisy and lacked musicality. We believe that the feature-resharing nature of CNNs led NST to misplace visual components indiscriminately throughout the spectrogram, leading to noise which degraded the quality of audio instead of novel composition. In addition, the visual models force the network to learn more difficult visual patterns of certain instruments which could easily be lost in a denser spectrogram. However, the authors of Jukebox mentioned that NST is still a viable alternative to their encoder/transformer based methods for music generation, which was another motivating factor for our decision to adopt this approach. In the future, we hope to implement more sophisticated methods for music generation, perhaps using transformers and autoencoders for music compression, similar to OpenAI's Jukebox[4], which implicitly parameterize musical instruments and style into tokens which can then be generated and upsampled to produce music without the noise artifacts of raw audio synthesis. Alternatively, we can re-attempt NST by using a more heavily parameterized visualization of music, perhaps by hand-selecting features such as entire instruments or better post-processing raw outputs from the network.

## 7 Contributions

Alec Janowski focused his attention on setting up a remote development environment for the team. He also wrote the convolutional neural network, which was then experimented on and refined by the rest of the team. He also primarily worked to train and tune hyperparameters for the neural networks and softmax classifier.

Alejandro Cid wrote the data-processing utilities used to efficiently convert long-form music mixes into 5-second spectrograms. He also wrote command line programs to train and test Lil Data as well as our benchmark softmax classifier. Furthermore, he produced the final project video.

Bowen Jiang wrote the neural genre transfer framework used to generate music as well as the functions for converting 5-second spectrograms back into audio. He experimented with both VGG-19 and Lil Data. Furthermore, he attempted an LSTM network to generate music, in order to compare with our neural genre transfer's output.

Data retrieval, writing, and research was divided between all members.

## 8 Supplementary Data

| | VGG-19 | Lil Data |
|---|---|---|
| Pre-training task | ImageNet (float32) | Spectrogram |
| Input dimension | 631x631x3 | 631x631x1 |
| Style layers | Block1_conv1, block2_conv1, block3_conv1, block4_conv1 | Conv1, conv2, conv3, conv4 |
| Style weights ($\lambda^{[l]}$) | 0.3, 0.4, 0.2, 0.1 | 0.3, 0.4, 0.2, 0.1 |
| Content layer | block5_conv1 | conv_4 |
| $\alpha, \beta$ values | 100, 1000 | 50, 500 |
| Iterations | 10 (early stop), 100 (full) | 100 (early stop), 500 (full) |

**References**


1. Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2018, August 10). This Time with Feeling: Learning Expressive Musical Performance. Retrieved November 16, 2020, from https://arxiv.org/abs/1808.03715
2. Huang, C., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., . . . Eck, D. (2018, December 12). Music Transformer. Retrieved November 16, 2020, from https://arxiv.org/abs/1809.04281
3. Barina, G., Topirceanu, A., & Udrescu, M. (2014). MuSeNet: Natural patterns in the music artists industry. *2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*. doi:10.1109/saci.2014.6840084
4. Dhariwal, P., Jun, H., Payne, C., Kim, J., Radford, A., & Sutskever, I. (2020, April 30). Jukebox: A Generative Model for Music. Retrieved November 16, 2020, from https://arxiv.org/abs/2005.00341
5. Dong, M. (2018, February 27). Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification. Retrieved November 16, 2020, from https://arxiv.org/abs/1802.09697