Music Generation: Style, Content and Sequence

Shreyas Fadnavis, Siddharth Thiruvengadam, Subramanian Sivaraman

{shfadn, sidthiru, ssivaram}@iu.edu

Indiana University, Bloomington

***Abstract*— The goal of this project is to combine and draw together different elements in music (could be genres of music) and improvise the existing approaches to music generation.**

- **We perform compositional style transfer to synthesize music "content" and "style" independently using the magnitudes of a STFT and Convolutional Neural Networks. Our method utilizes randomly initialized filters with iterative phase reconstruction using the Griffin-Lim algorithm.**
- **We implemented a WaveNet like generative model for music modeling synthesis. This model uses temporal embeddings learned by a deep autoencoder to condition the output of the decoder. The NSynth dataset, a large-scale and high-quality dataset of musical notes that is an order of magnitude larger than comparable public datasets, was used to train this model. Using NSynth, we show that the model learns embeddings that allows morphing between instruments by interpolation in timbre to create new types of sounds that are realistic and expressive.**
- **To understand and explore these models of symbolic sequences of polyphonic music as a general piano-roll representation, we developed a Restricted Boltzmann Machine model based on distribution estimators that is able to discover temporal dependencies in high-dimensional sequences. Our approach further investigates using LSTM-RBMs to serve as a symbolic prior to improve the accuracy of polyphonic music representation.**

***Keywords*— Music Style Transfer, Convolutional Neural Networks, Restricted Boltzmann Machines, Wavenet Autoencoder, NSynth.**

I. Introduction

To understand how to generate 'new' music and how really our brain responds to music, we read through a quite a lot of material on brain cognition and music. In a nutshell, within our own brain is a system that performs the interpretive feat of converting complex acoustic sequences into perceptually discrete elements (such as chords) organized into hierarchical structures that convey rich meanings. Our approaches help exploring both the similarities and the differences between music perception and deepen our understanding of the mechanisms that underlie these interpretations to use them in music generation. Music generation requires the automation of the process of composition by using probabilistic methods (neural networks in our case).

Symbolic music or written music is discrete in nature. In the Equal Temperament system musical pitches are constrained to occupy discrete positions on the continuous frequency scale. Note durations also lie in a discrete space which can be visualized by the Figure 1.

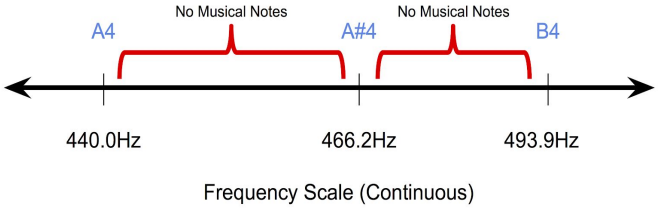


Figure 1

Before getting into the modeling of the sequences, let's look at how do we understand what representing such music really means:

Multi-level and Multi-modal Representation

Score serves as a highly-symbolic and abstract visual representation to efficiently record and communicate music ideas, whereas the sound is a set of continuous and concrete signal representations that encode all the details we can hear. Therefore, we can picture the two representations at different levels, with the score at the top and sound at the bottom. In the middle, people often insert an intermediate representation of performance control. The reasons are twofold. First, *musical semantics* and *expression* rely heavily on performance control that a *funeral hymn can sound really happy by simply tripling the tempo*. Second, the performance control for many instruments can be easily parameterized and therefore are very machine friendly. In order to fully evaluate different aspects of music generation, we investigated these three representations more in-depth.

Score Representation

Score representation is highly symbolic and encodes abstract music features indicated by the composer, including tonality, chord, pitch, timing, dynamics and rich structure information such as phrases and repetitions.

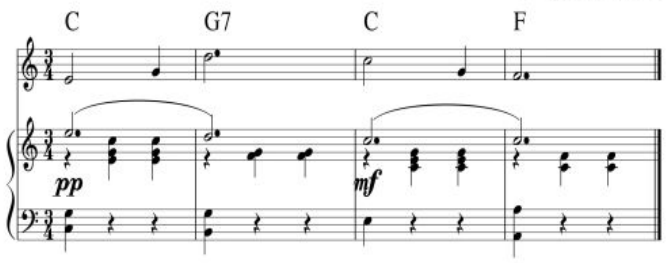


Figure 2

The key character of score representation is that the encoded features are mostly discrete with a mix of measurement scale. Take western music notation (Figure 2) for example. Note onset is a ratio variable and lies on integer multiples of a certain time unit (usually 1/8 beat is short enough). Pitch is an interval variable, whose corresponding frequency always lies in a discrete sequence. (E.g., the frequency of C4 in the equal-tempered tuning is 261.63 Hz, the frequency of its successive pitch C\4 is 277.18 Hz, and there is no other pitch frequencies lie in between. refer Figure 3). *Such characters bring a challenge for generative models since discrete optimization is in general very difficult and a mixed scale makes some numerical operations impossible.*

Performance Control Representation

A performance control encodes an interpretation of the corresponding score, relying on which a performer turns the score into performance motions. A commonly used control representation is MIDI piano roll (used in RBM model for evaluation), where each note is encoded by its pitch, dynamics, onset, and duration. To be specific, pitches are integers in semitones with C4 being 60, dynamics are integers in velocities units (speed with which the keys are hit) ranging from 1 to 127, and timings are floating point numbers in seconds. Compared to score representation, the key character of performance control is the enriched and detailed **timing and dynamics information**, which is more or less determined by the musical expression of a performance. On the other hand, most structural information such as phase, repetition, and chord progression is flattened and become implicit during the translation from the score to performance control. *Note that performance control is largely independent of the actual instrument; it is not yet the final music sound and still considered a middle-level abstraction.*

Representation, Content, and Style

Music can be seen as an acoustic realization of the corresponding performance control via a certain instrument. Two commonly used formats for music representation are waveform and spectrogram. The key character of music representation is purely continuous and rich in acoustic details such as timbre, articulation, and other nuances not available in other levels of representation. At the expense of such acoustic details, all symbolic abstractions together with precise performance control information become no more explicit and get hidden in the audio.

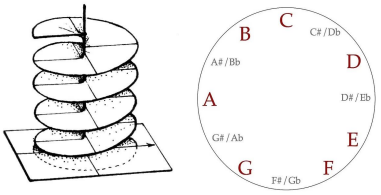


Figure 3

II. Neural Style Transfer

Style transfer is a common technique in he computer vision community but is not clearly explained when it comes to music synthesis. From our research, we discovered that style transfer in music can be broadly classified as in Figure 4.

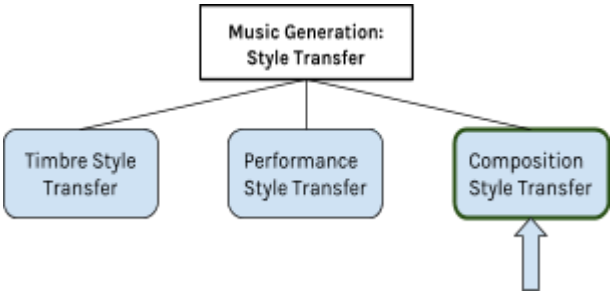


Figure 4

Our method focuses on Composition Style Transfer which preserves the identifiable melody contour and the underlying structural functions of harmony while altering some other score features in a meaningful way. Our composition style transfer allows us the create

variation, improvisation, or re-harmonization of a piece of music. Composition style transfer is basically stylistic automatic composition. It requires a disentanglement of different score features and implies that there is room to create new types/idioms of score features (such as rhythm, texture, and chord progression) through the combination of different ones.

We have made use of random shallow convolutional neural networks (CNNs) that serve as a good model of natural textures. Patches from the same texture are consistently classified as being more similar than patches from different textures. Samples synthesized from the model thus capture spatial correlations on scales much larger than the receptive field size, and sometimes even rival or surpass the perceptual quality of state of the music texture models (but show less variability). Figure 5 shows the workflow of our model to perform style transfer.

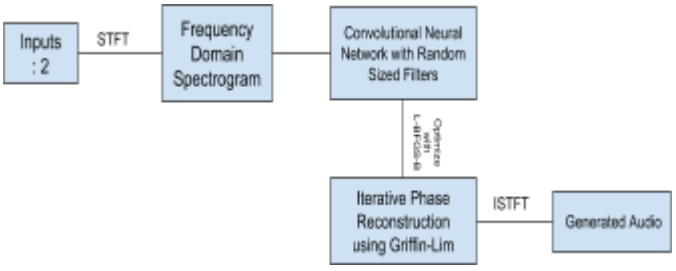


Figure 5

We followed the details on Griffin-Lim and Real-Time phase reconstruction from here for an in-depth explanation of the same. The following is the result of the implementation of our style transfer model. In Figure 6, Content refers to Eminem: Not Afraid and Style refers to the American National Anthem. The result (click here to listen to the generated audio) we would assume to be a well interpolated and properly matched notes of of both songs to get a new track that sounds like both of the components it is made up of.

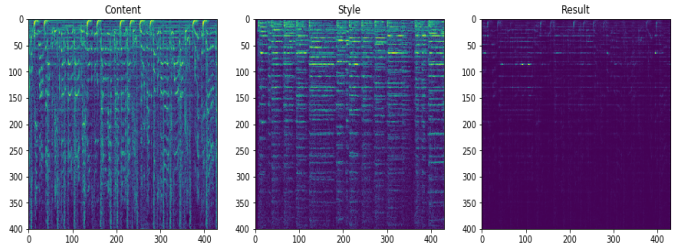


Figure 6

III. Evaluating Music Style Transfer

Let's suppose that we had a way of measuring how different in content two music styles are from one another. Which means we have a function that tends to 0 when its two input music components 'Content' and 'Style' are very close to each other in terms of value, and grows as their value deviates. We call this function the **content loss**:

$$L(content\,(c),\ generated\,(x)) \approx 0$$

Let's also suppose that we define another function that told us how close in style two music compositions are to one another. Again, this function grows as its two input music components 'Content' and 'Style' tend to deviate in style. We call this function the **style loss**:

$$L(style\,(s),\ generated(x)) \approx 0$$

We optimize for both of the above loss functions simultaneously by:

$$x^* = argmin_x(\alpha L_{content}(c,x) + \beta L_{style}(s,x))$$

Where, α and β are simply numbers that allow us to control how much we want to emphasise the content relative to the style and could be seen as weighting factors. Here we notice that the definitions of these content and style losses are based not on the note-by-note differences between 2 types of music, but instead in terms of higher level, more *interpretative differences* between them. All the literature that does this type of music generation relies on subjectivity of the results. Another solution to evaluating this type of generation is to use a classification/ source separation approach where we try to classify the generated audio based on training procedures like the SGD and optimize both the losses simultaneously.

IV. NSynth: Neural Audio Synthesis

NSynth (Neural Synthesizer) is a new approach for music synthesis. It is based on Wavenet, and generates sounds at the level of individual samples. This is different from a traditional synthesizer, which generates audio using components like oscillators and wavetables. Learning directly from data, NSynth provides the ability

to explore new sounds that would be difficult or impossible to produce with a hand-tuned synthesizer.

The model requires raw audio signals for training, and the training is very expensive. The best dataset available for this purpose is the NSynth dataset itself: a collection of musical notes that is orders of magnitude larger than other publicly available music datasets. Training the network takes about 10 days on 32 K40 GPUs to converge at around 200k iterations. In this paper, we use the pre-trained model provided by Magenta (the developers of NSynth), and present ideas for pre and post processing of the audio generated by NSynth.

WaveNet is an architecture that can effectively model short temporal sequences such as speech and music. It an autoregressive network of dilated convolutions, and generates audio one sample at a time. The context, or dependency on prior samples, available while generating a sample is limited to several thousand samples, which amounts only to about a second. And so, capturing long-term structure requires some form of external conditioning.

NSynth removes the need for conditioning on external features. It uses a WaveNet-style autoencoder to learn embeddings, Z = f(x), from raw audio waveform. These embeddings are temporal in nature, and are fed to a Wavenet decoder along with audio samples to generate new samples. The joint probability distribution looks like:

$$p(x) = \prod_{i=1}^{N} p(x_i | x_1 \ldots x_{N-1}, f(x))$$

Let's now take a look at the architecture used in NSynth, depicted in Figure 7. The encoder model is a 30-layer network of dilated convolutions followed by 1x1 convolutions. Each convolution has 128 channels and precedes a ReLU nonlinearity. The output feeds into another 1x1 convolution before average pooling to get the encoding Z.

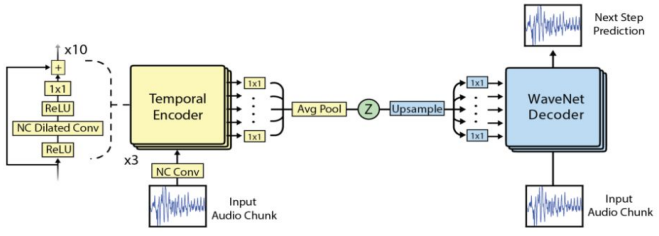
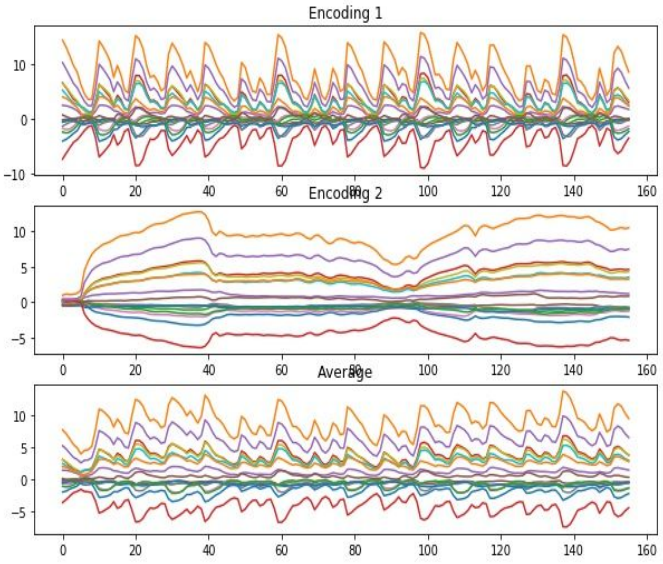


Figure 7

We call it a temporal encoding because the result has separate dimensions for time and channel. It was found that the best configuration is a stride length of 512 for the average pooling, with 16 dimensions per timestep.
Generating new audio by averaging the embeddings from two sources:
We can generate novel sound by averaging the embeddings produced by the NSynth autoencoder. This equates to averaging the representation of their timbres, tonality, change over time, and resulting audio signal, and thus has a different result from that averaging raw audio signals.

We took the encodings of a hip-hop beat and a cello piece, and averaged their embeddings. The averaged embedding is then decoded, to produce new audio. The resulting music is quite interesting. We have provided the audio generated by mixing, and below we show the visual representations of both the individual and averaged embeddings.



V. Modeling Polyphonic Music

Modeling sequences is important since music is inherently sequential. Complex sequences are non-local and are caused by the impact of factors localized in time that can be delayed by an arbitrarily long time-lag. With musical data being so high dimensional at each time step, the conditional distribution turns out to be multi-modal, and therefore need models that predict the

conditional distribution of the next time step given previous time steps. For polyphonic music, occurrence of a particular note at a particular time modifies considerably the probability with which other notes may occur at the same time. This means that notes appear together in correlated patterns, or simultaneities. Traditional RNN models fail to capture all the combinations since the possible number of combinations would increase the computational complexity tremendously. This is why we chose the Restricted Boltzmann Machine energy-based models which allows us to express the negative log-likelihood of a given configuration by an arbitrary energy function.

By separating and recombining music contents and music styles of different pieces, it is possible to generate new music that is both creative and human-like. However, "music style" is a fuzzy term that can literally refer to any aspect of music, ranging from high level compositional features (such as tonality and chord sequence) to low-level acoustic features (such as sound texture and timbre). *This ambiguity is mainly due to the intrinsic multi-level, multi-modal character of music representation — music 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).*

It is convenient to use MIDI files, as this format allows different instruments transcribed on different tracks or channels to be separated easily. You can therefore easily get per-instrument piano rolls with *pretty_midi*. This enables the potential of creating annotations for melody extraction and score-informed source separation. We can thus bring down the complex dimensionality of multiple instruments to make a more general representations of music. In our implementation, we have trained the RBM with MIDI files as inputs in the form of a note state matrix. This way, we are preserving the pitch, upper and lower bounds, notes and velocity. This was then fed as input to the RBM which has 50 hidden units and number of visible units which depends on the note range of the input. We then use the Gibbs Sampler to perform MCMC sampling from the probability distribution of the RBM defined by the weights and biases of the hidden and visible units. The following is the piano roll representation output generated. This has been trained on 28 music instrumentals of different disney theme songs to generate the following piano roll representation (Figure 8 and 9). The length of the generated representation can be easily altered by changing the number of timesteps. We have also performed this on different set of genres to try to get generalized symbolic prior for generating music that is a fusion of Western Pop and Indian Classical: took 100 songs that are a mix of both (Figure 10 & 11).

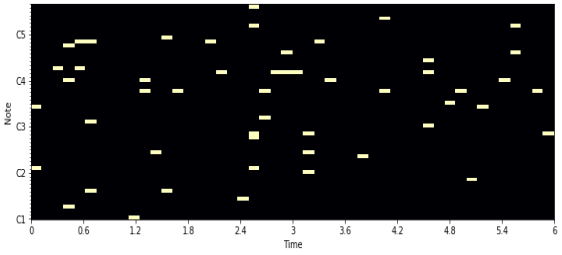


Figure 8

Following is the pitch class histogram which depicts the proportion of notes of the generated music as per pitch class.

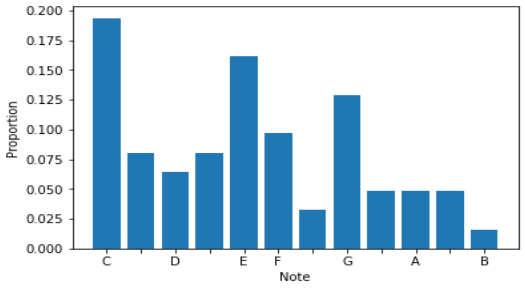


Figure 9

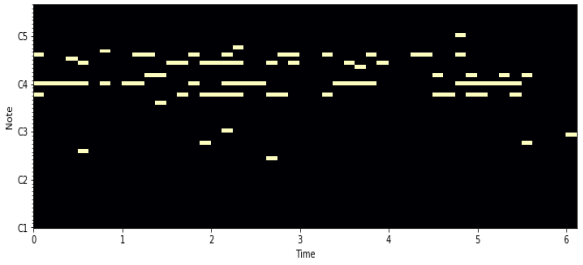


Figure 10

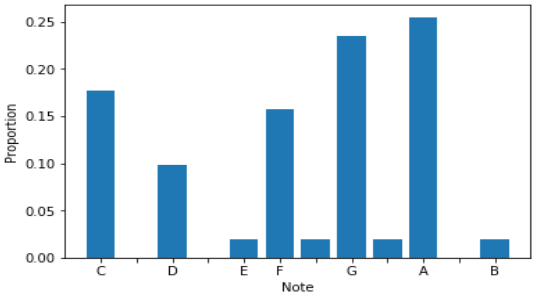


Figure 11

References

Neural Music Style Transfer

[1] Ustyuzhaninov, Ivan et al. "Texture Synthesis Using Shallow Convolutional Networks with Random Filters." *CoRR* abs/1606.00021 (2016).

[2] Verma, P., and Smith, J. O. 2018. Neural style transfer for audio spectrograms. arXiv preprint arXiv:1801.01589.

[3] Boulanger, R. C. 2000. The Csound book: perspectives in software synthesis, sound design, signal processing, and programming. MIT press.

[4] D. Ulyanov and V. Lebedev, "Audio texture synthesis and style transfer," 2016.

[5] Liang, F. 2016. Bachbot: Automatic composition in the style of bach chorales. Masters thesis, University of Cambridge.

[6] D. W. Griffin and J. S. Lim, "Signal Estimation from Modified Short-Time Fourier Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, no. 2, pp. 236–243, 1984.

[7] Z. Pr˚uša and P. Rajmic, "Toward High-Quality Real-Time Signal Reconstruction from STFT Magnitude," IEEE Signal Processing Letters, vol. 24, no. 6, pp. 892–896, 2017.

Wavenet and Neural Audio Synthesis

[8] J. Engel, C. Resnick et al., "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders", Apr 2017

[9] Neural Audio Synthesis, The Magenta blog, https://magenta.tensorflow.org/nsynth

[10] Humphrey, Eric J. Minst, a collection of musical sound datasets, 2016. URL https://github.com/ ejhumphrey/minst-dataset/

[11] Sarroff, Andy M and Casey, Michael A. Musical audio synthesis using autoencoding neural nets. In ICMC, 2014.

[12] Theis, Lucas, Oord, Aaron van den, and Bethge, Matthias, "A note on the evaluation of generative models.", arXiv:1511.01844, 2015.

[13] van den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Si-monyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W., and Kavukcuoglu, Koray, "Wavenet: A generative model for raw audio." CoRR, URL http://arxiv.org/abs/1609.03499

Modeling Polyphonic Music

[14] Boulanger-lewandowski, N., & Vincent, P. (2012). Modeling Temporal Dependencies in High-Dimensional Sequences : Application to Polyphonic Music Generation and Transcription, ICLR 2012.

[15] Eck, D. and Schmidhuber, J. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In NNSP, pp. 747–756, 2002.

[16] Hinton, G.E. Training products of experts by minimizing contrastive divergence. Neural Computation, 14(8): 1771–1800, 2002.

[17] Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. JMLR: W&CP, 15:29–37, 2011.

[18] Paiement, J.F., Bengio, S., and Eck, D. Probabilistic models for melodic prediction. Artificial Intelligence, 173 (14):1266–1274, 2009.

[19] Sutskever, I., Hinton, G., and Taylor, G. The recurrent temporal restricted Boltzmann machine. In NIPS 20, pp. 1601–1608, 2008.

[20] Welling, M., Rosen-Zvi, M., and Hinton, G. Exponential family harmoniums with an application to information retrieval. In NIPS 17, pp. 1481–1488, 2005.

(The code of LSTM-RBM is work in progress! The model is not training properly and we are working on the same. PFA partial implementation)