Composition of Classical Music using Deep Learning

Samir Agarwala

The Cathedral and John Connon School, Mumbai, India

Abstract

Music, especially classical music, has long been known to be very hard to learn and perfect by human beings. Each person has his/her own style and identity in the music, which is very hard to replicate otherwise. But can such complex patterns and the sense of "musicality" be learnt by machines, and if so, how and what do they learn? In this paper, we attempt to generate our own music of different kinds and analyze how well our attempts do. Neural Networks have shown great promise in modeling complex patterns. Recurrent Neural Networks specifically are known to be able to understand and model temporal dependencies in data. In this project, we use a Sequence-to-Sequence model to learn musical patterns and then generate classical music. We also try using WaveNet, a start-of-the-art audio generating model, to compare the predictions. We try to learn different classical musicians like Beethoven and Mozart (for the same instrument) and generate music that uses an ensemble of their styles. Additionally, we try to learn only one musician like Beethoven but across several instruments, and see how a machine is able to learn his style across different instruments.

Project Code
`https://github.com/samiragarwala/music_generation`

1. Introduction

Classical music has many fundamental parts like Melody, Motif, Measure, Timbre, Harmony and Rhythm. Each of these parts can be combined in innumerable ways to create music. Pleasing and soothing is considered the art of combining these elements in an "appropriate" manner for the listener.

These listeners have different tastes and would like to listen to the music of their tastes more often. Can we learn what the "art" behind creating music is, and create a model than can learn this "art"and generate personalized music for the listener?

Machine Learning algorithms have shown promise in learning complex patterns. In recent times, Neural Networks have been predominantly used. Convolutional Neural Networks (CNNs) have been known to be excellent image classifiers and feature extractors from raw data, while Recurrent Neural Networks (RNNs) have been known to work extremely well for Speech, Language and data with temporal dependence.

In this paper, we use deep neural networks to model and learn complex patterns and see what these models are able to learn. We also analyze what goes into the "art" of creating music. We first use recently created models that are good at handling raw audio, like WaveNet, and see whether currently published models are able to learn the patterns. Then, we create a type of Recurrent Neural Network model (called a Sequence-to-Sequence model) for the problem, and see how that performs.

2. Training Dataset

We use the University of Washington's publicly available dataset called MusicNet [1]. Musicnet consists of about 330 audio recordings from around 17 composers, and uses a total of 5 instruments. These recordings are raw audio files which have not been preprocessed. Each recording can be as short as 30 seconds or as long as 5 minutes. The sampling frequency is 16000 Hz, which means that each second has 16000 data points to train on. For training our models, we create the datasets in two different ways. The first one is as follows:

1. Take 256 data points, called D_1. This is the first window W_1 of points that we consider. The first 128 points of D_1 is the training input X and the next 128 points of D_1 is the prediction, called Y.
2. Our next window, W_2, is 128 points shifted from the previous window W_1. W_2 also has 256 points D_2, out of which the first 128 points of D_2 is the training input X and the next 128 points of D_2 is the prediction, called Y.

3. Continue shifting the window, collecting points and dividing into training and prediction.

The other kind of dataset we generate is as follows:

1. Take 129 data points, called D_1. This is the first window W_1 of points that we consider. The first 128 points of D_1 is the training input X_1 and the points $2 - 129$ of D_1 is the prediction, called Y_1.
2. Our next window, W_2, is 64 points shifted from the previous window W_1. W_2 also has 129 points D_2, out of which the first 128 points of D_2 is the training input X_2 and the points $2 - 129$ of D_1 is the prediction, called Y_2.
3. Continue shifting the window, collecting points and dividing into training and prediction until end of the raw audio clip.

3. Methodology

3.1. Sequence-to-Sequence Model

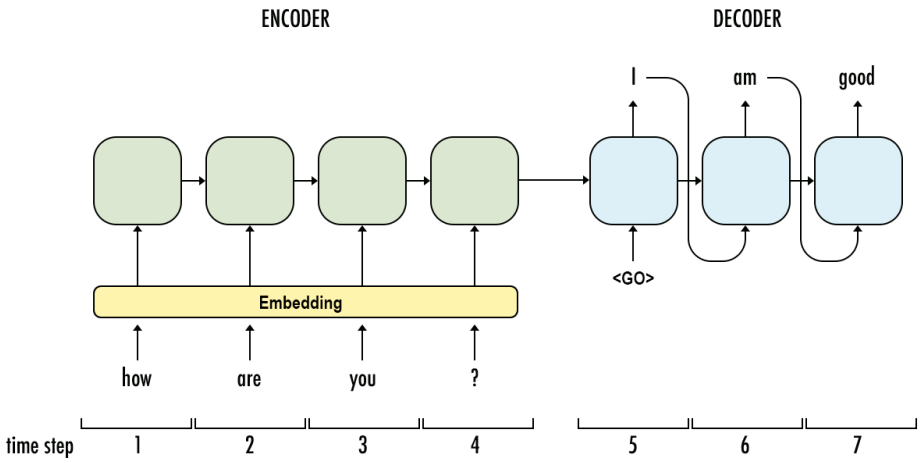

Figure 1: Typical Sequence-to-Sequence Model during inferencing. For music, we use data-samples instead of word-embeddings [2]

A Sequence-to-Sequence model is a type of Recurrent Neural Network, that has an Encoder and a Decoder. The encoder takes an input at every-time step and then encodes all the information over T time-steps, in a sequential

manner, into a single vector $S \in R^n$ for the decoder to use to make predictions for T time-steps. The decoder takes the inputs and the state to make predictions. The state is either from the encoder S if it is the first decoding time-step or the previous time-step of the decoder hidden state.
The encoder takes an inputs X_i, while the decoder has prediction labels for training called Y_i. The inputs to the decoder are $\{x_s, Y_i[1:T-1]\}$, where x_s is a "start" token telling the decoder to start decoding while the remaining are the correct outputs of the previous time-steps. We could feed the output of the previous time-step t into the next time-step $(t+1)$, but if there is an error in at time-step t, then the error will propagate for all time-steps and increase exponentially. So despite getting predictions $\hat{y}_t$ for $1 \leq t \leq T$, we use$\{x_s, Y_i[1:T-1]\}$ as inputs to the decoder.
We use a least-squares loss of the form $\sum_{i=1}^{T}(y_i - \hat{y}_i)^2$ over the time-steps to make predictions. Adam optimizer [3] is used for optimizing the loss function.

3.2. WaveNet

Google DeepMind came up with a generative model for audio samples by learning from raw audio directly called WaveNet [4]. That is, given a list of .wav audio files, the WaveNet model could learn the patterns in the audio files directly and then generate audio of similar nature. The WaveNet model learn to predict the audio data sample values at time-step T by using audio samples $\{T-k, T-k+1, ..., T-1\}$ and $\{T+1, T+2, ..., T+k\}$, where T and $k \in N$. Thus, by using information in past and future data, WaveNet is able to get a good understanding of the nature of the current time-step T. Dilated convolutions are used to extract spatial features across the various time-steps.

4. Programming Implementation

4.1. Pre-processing

The dataset was in 327 .mp3 files, and we had to convert it into *.wav* files using Python. To check that the waveforms were correct after this conversion, we used iPython Notebooks (Jupyter) to visualize the python output and to do the processing.

Then python code was used to read in these *.wav* files and then convert them into time-step based samples needed for the neural network models, as described above.

4.2. Neural Network

The Neural networks were implemented using packages called Keras and Tensorflow, which provide APIs for building neural network models. The Seq2Seq model was built on top of Keras, with functions to create different models, and train them using different methods.

WaveNet was obtained using a Google DeepMind implementation online. We modified relevant parts of the code to make it work with our dataset and pre-processing code and then integrated the files into the GitHub repository.

5. Results

5.1. Quantitative

We see generate arbitrary length of music using our method. We tried generating music for 5 seconds and for 2 minutes. The Seq2Seq model is able to give some non-zero value for most time-steps. There is a good distribution of predictions larger and smaller than the correct data-points, though the model is trying to learn to make these prediction closer to the actual correct values of the points.

The WaveNet model can also generate arbitrary length music pieces. It starts of by producing a 1 second music introduction but then produces blank outputs (the prediction is 0) for the remaining time-steps. But at the end of the music clip, it again produces some music notes for about 1 second.

5.2. Qualitative

Both model implementations have different qualitative aspects. The Seq2Seq model understands that music requires non-zero notes at most time-steps, and hence follows that pattern. But it is unable to understand the pattern, rhythm and musicality. Hence, the values produced are not able to follow the requirements to create music.

The Wavenet model is able to create a single musical beat, lasting about 1 second. For that 1 second, WaveNet creates a musical note by following common classical music practices. But then, it is unable to replicate that pattern for the remaining time-steps. This being said, at the end of the generated musical piece, the model understands that classical music tends to

often end on a high-note. This results in there being a musical note at the end of the generated piece, having similar musicality as the first 1 second. In between, since there are pauses at many time-points in classical music, the model is not able to completely understand the complex patterns behind when the pauses come and why they are inserted. Hence, the model just predicts all zero-values, or blank notes, to be able to best match the little it can understand.

6. Limitations

A lot of limitations of learning music were highlighted as a result of this project. As we knew before, classical music is hard to replicate and even the models found it hard to do so. As a result, we need much more data than the 330 samples we had to be able to teach the model classical music. We could use data augmentation techniques to mimic samples that might have been played by popular classical musicians and then use that as training data in the event that there is not enough data.

We would also need for complex neural network models to be able to learn. From a biological perspective, music is a global process. We require our right brain for the creativity of music when composing new music and the left brain to understand how music should be played and how the notes are stitched together (or memorize some of the common patterns!). If a highly developed machine like our brain requires so many neurons and connections in the music activity, then our current models should at least be able to match that to be generate equally good music. Hence, more deep models with different algorithmic techniques and specialized network layers might be needed.

Predicting multiple notes at the same-time is also much harder than a single note. Most work has been done in simple music generation where the music consists of mostly single notes. But in classical music, there are always multiple notes. To get around this problem, we tried to predict the raw audio value directly without using the actual notes. But there are about 4 billion values that we can predict at a given time-step and to teach the model which value to predict given all the value it has already predicted is a hard task.

Lastly, the precision of values used might be an issue. We tried predicting 32-bit integer values in our models rather than 32-bit floating point values,

so as to make it easier for the model to predict values. But rarely does music come as an absolute integer audio value at any given time-step. The value of the audio wave is bound to be a decimal value, represented as a floating point number, which is much more accurate than an integer. Thus, we might lose precision and as a result, other musical parts since we are not predicting a precise enough value. This limitation could not be accommodated due to lack of computational resources as a lot of memory and compute would be required to predict floating-point values.

7. References

[1] J. Thickstun, Z. Harchaoui, S. Kakade, Learning features of music from scratch, in: International Conference on Learning Representations (ICLR).

[2] M. Chablani, Seq2Seq Model, Medium Corporation, 2017.

[3] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2014).

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, CoRR abs/1609.03499 (2016).