Food for Thought
Is it possible to create computer-generated art (specifically music) that realistically mimics patterns of human expression?

Background
This project serves as a means to begin to answer the fundamental question of whether art and computer science are necessarily at odds with each other, or whether they can converge sometimes. A while back I saw a Youtube video (linked in the Resources section) wherein a Youtuber fed lots of Baroque music, translated as text, into a machine learning algorithm, which then outputted its own version of Baroque music. Unfortunately, the algorithm didn't start to perform well until several hours' worth of music had been fed into it. This past week I found another project by a man named Tyler Doll who worked with a couple other people to try a similar concept: training a machine learning algorithm with MIDI files to produce an output MIDI that closely resembles other songs in a certain genre. Two questions arise out of these and other similar projects:

- What characteristics are shared by songs produced by humans? And
- Can machine learning algorithms replicate these characteristics to the extent that the songs they produce would be nearly indistinguishable from those written by people?

What does human music sound like?
The art and science of music theory aims to describe abstract concepts using specific terminology and analysis. When theorists talk about a genre, period, or era of music, they mean that the vast majority of music produced within a timeframe (or, for overlapping timeframes, by similar people or about similar subjects) shares certain characteristics including form, instrumentation, style, and melodic or harmonic motives and patterns. There are also defined guidelines for writing music, regarding melodic lines, chord progressions, etc. that most composers tend to follow most of the time within their genre or era.

Seems like it would be pretty easy for a computer to make its own music.
Sure, all you would need to do is figure out a way to encode the basic rules of music theory into a program, use pseudorandomness to allow for noise (no pun intended), and have the output of said program be some sort of sound file.

However, there are a few problems with this approach:

- Hard-coding a list of rules into a program does not create a very general solution to a problem. This issue lies at the heart of why machine learning is more effective than traditional programming methods in certain circumstances.
- Regardless of how effective or robust the program would be, music theory is not a static field—new concepts are being discussed and new rules created all the time. So while we might reasonably be able to recreate classical music because no one has changed it in nearly 200 years, we could not do the same with modern minimalist music. Which brings us to the third, and most important problem…
- Human composers are wont to bend, break, or ignore the rules of music theory on a whim because in many of their minds, art should not be put under constraints; when it is, it sounds too formulaic (see this Wikipedia article for information on a dice game that was used by 18th-century composers to generate minuets).

How can machine learning be used to generate music?

Enter Youtuber carykh and avid coder Tyler Doll, who set out to produce computer-generated music using machine learning algorithms trained on songs written by humans. I will be focusing on the work done by Tyler Doll and his team, which was itself inspired by this tutorial by Sigurður Skúli, producing music with a bidirectional long short-term memory recurrent neural network (BLSTM RNN) coded in Python using the Keras library. (Later on, they also did some similar work with generative adversarial networks, or GANs.)

Summary of Tyler Doll's project

Doll and his team gathered a few dozen MIDI files of songs from three different genres: ragtime, rap, and Christmas. Because they wanted the algorithm to produce good results and avoid notes that didn't "belong," all of the songs were stored in the key of C major or A minor. Using Keras and TensorFlow, they created a network with 512 bidirectional long short-term memory nodes per layer. They used BLSTM nodes because that would give them the best chance of producing sequences of notes that sounded like realistic melodies. They encoded each combination of either pitch or rest and duration separately in order to better parse and produce music, then they hard-coded the algorithm to generate only songs of the pattern verse-refrain-verse-refrain-bridge-refrain. Because much of the code to generate different aspects of the pattern is similar, it would be possible to alter the program to create songs of any hard-coded pattern with the method they used.

Specific parameter values and function types:

- Number of nodes per layer: **512**
- BLSTM node dropout rate/layer: **30%**
- Activation function: **softmax** (normalizes input into a vector with a probability distribution whose total is 1; outputs values between 0 and 1)
- Loss function: **categorical cross-entropy** (to work with the multi-class note prediction problem)
- Optimizer: **RMSprop** (recommended by Keras for recurrent neural networks)
- Default number of training epochs: **10** (you can change this as an argument to running the script)

Results

At the end of the Towards Data Science article (linked below) there are three embedded sound files—one for each genre of music that Doll and his team were trying to replicate the style of. If you pick a couple MIDI files from each genre's folder and listen to them (or if you are already pretty familiar with the way each genre sounds) then listen to the embedded sound files, you'll find them to be a fairly accurate representation of the genre at large, assuming the thirty or so songs that were used to train each of the models are themselves fairly accurate representations of the genre.

The README file inside the program folder contains a note to this effect about generative adversarial networks: the results generated by training GANs were not quite as good as those generated using RNNs, but with a bit more tweaking it is possible that both types of model would end up generating decent approximations of human-produced music.

Doll notes too in his article that the scope of this project was very limited in terms of actually reflecting the output of human artists, and that several improvements and additions could be made in order to produce even better output.

My own experimentation

When diving into Tyler Doll's project, the classical genre was noticeably absent from the folders of training music. (I'm being a bad music theorist and lumping baroque, classical, and romantic music into one mega-genre, because that's how most people think of it anyway.) I found an online repository (linked below) containing hundreds of MIDI files of piano music by a couple dozen different composers, downloaded all of them, and placed them in a folder called "classical" inside the songs folder of the Github project that I had downloaded.

There were a couple reasons I wanted a larger training set than the one Doll and his team had used:

- In the world of machine learning, no matter what the problem, project, or method used to find a solution, more data tends to lead to better results more reliably than improved algorithms.
- I knew I would have neither the time, nor the energy, nor the patience to convert all the MIDI files to the same key signature or tempo or isolate only the ones with the same time signature. Because of all this extra "random noise," more songs (and a longer duration of training on those songs) should lead to better results.

Unfortunately, running Python programs on my own machine throughout the duration of this course has never ceased to cause me problems. So even though I created a lesson on the installation and use of the TensorFlow package, I was never able to successfully install or use it on my computer, always getting error messages regarding modules that the Python interpreter is unable to locate, including the native TensorFlow runtime. Since Doll's project is based in Keras, which functions on top of TensorFlow, I couldn't run the training script on his training data or my training data.

My lack of experience with Python also contributes to the aforementioned issues, to which I looked up and tried several possible solutions to no avail. If I had a more extensive knowledge of the language, I might have been able to parse through the code in the span of the last few days and pinpoint the causes of the errors. But as it is, I still learned a great deal about BLSTM RNNs and GANs and their application in the quest to replicate works of art created by human artists.

Resources

- Towards Data Science article that gave me the idea (credit to Tyler Doll): https://towardsdatascience.com/making-music-with-machine-learning-908ff1b57636
    - GitHub code/file repository: https://github.com/tylerdoll/music-generator
- Source for classical piano MIDI files: http://www.piano-midi.de/midicoll.htm
- The Magenta project, open-source art/music ML research project using TensorFlow: https://magenta.tensorflow.org/
- A brief history of music in the world of computing: http://artsites.ucsc.edu/EMS/Music/equipment/computers/history/history.html
- Youtube video by user carykh, "Computer evolves to generate baroque music": https://www.youtube.com/watch?v=SacogDL_4JU