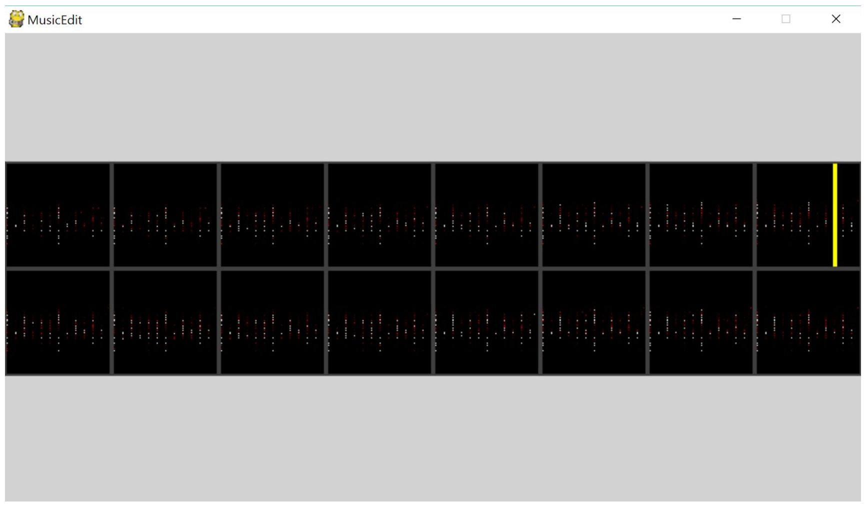
ART AND MACHINE LEARNING
CMU 2019 SPRING
PROJECT 3

Sweet Dreams

Zach Saffran
Josh Moavenzadeh
Chris George
Pancho Cabrera
Angel Hernandez

DESCRIPTION

Sweet Dreams is an interactive machine learning platform used to create lullabies. It is gui-based and requires a Leap Motion, a sensor that measures hand and fingertip movement and feeds that as input to a computer. A photo of the gui is shown below, in which a small white dot represents the midi notes that are to be played and the yellow bar shows the notes currently being played. The platform trains an autoencoder on a dataset of 1000+ midi files of classical music for 400 epochs and saves the model, allowing the user to use it with the Sweet Dreams gui. The input from the Leap Motion scales the top PCA components learned from the autoencoder, producing and playing new music in real time.



Sweet Dreams GUI

CONCEPT

Initially, we sought to create an audiovisual experience which the user control. To create the visuals, we explored using CPPNs which produce abstract results. In accordance with this, we tried to generate abstract music with RNNS which would relate to the visuals using Magenta. However, this prevented us from having an interactive component in our project. Thus, we changed our concept to fit the technical strengths of our autoencoder, and wanted to produce continuous lullaby sounds that could even change depending on the movement of the baby.

By using the autoencoder, we are able to see what the machine learning algorithm views as the most important components of the midi dataset, and by scaling and changing these we are able to customize music in a way that is more understandable to the user than just changing the structure of an RNN model. The autoencoder also allows us to interact with the model in real time, as we can scale the PCA components with the Leap Motion input, then produce new music in real time.

Furthermore, we decided on the concept of a lullaby machine because of the innate rhythm and sound created from interacting with the leap motion. Although not trained on lullabies, the output after user interaction created what sounds like lullabies with at times sudden changes of tone and pitch, alongside other points of fluidity. This concept of changing genres via human interaction was an interesting idea that evolved from our use of and conversations about this project. It would be interesting to further explore the idea of creating some piece of art with some initial set preconditions created either through training or other means, and have users intentionally change those conditions into something entirely new.

TECHNIQUE

As stated in the concept, we used an autoencoder neural network to generate new music. For a single training example, this network would take in a feature representation of a .midi file and forward propagate through two fully connected hidden layers to get an encoded representation of the music. This encoded vector is intended to learn the most predominant features and nuances that makeup a song. Then, the encoded features were passed through two fully connected hidden layers which would decode the encoded feature vector to create a new song. This newly created song is trained to be identical to the input song but because the dimensionality of the encoder network is of a lower order, the model learns to create something similar while being original. Relu activations were used across hidden units, batch normalization was used to speed up training and dropout was used to assist with the model's ability to generalize in creating a new song.

After the model was trained, we applied Principle Component Analysis (PCA) to the encoded feature vector in the neural network. PCA is another form of dimensionality reduction used to create a new set of features (principle components) that are orthogonal to each other, i.e. uncorrelated. Applying PCA to the encoded features allowed us to provide the user with music parameters they could tune in real-time. Using an autoencoder and PCA to create new music enabled us to give the user an experience/say in the creation of final product, which is something not readily practical if we would have used an RNN.

PROCESS

A concept we were exploring for our final project was an immersive experience in which a user controls surrounding visuals and soundscape generated with ML with their physical movements. We explored using CPPNS which allowed the creation of fluidly interpolated visuals (or results can be downloaded here). We planned to tie our visuals into the outputs created from our RNN, and experimented by using the Magenta plugin for Ableton to create machine learning music with human assistance. The way the plugin works is it uses a variational autoencoder and an RNN to support functions called continue, interpolate, and generate. We were able to create a full song using the generate tool first, followed by the continue tool, followed by the interpolation tool. After post-processing, this song was a success in that it sounded like two different ambient soundscapes phasing between each other. Although this was what we were looking for to match the abstraction generated by the CPPN, it would end up not being compatible with the other, more important component of our project. As a result, our project ended up being the creation of music in real time along with an RNN with the help of the Composer by Hackerpoet.

RESULT

Some short example songs produced by Sweet Dreams can be listened to here. These have not been post-processed at all come directly from Sweet Dreams:
Result: https://github.com/saffranzachary/aaml_project3/tree/master/Results
Slides: https://docs.google.com/presentation/d/1yvyHMWIgIifP4IjCOVrH7ABmdwsxryuSFK2FXTAJn8k/edit?usp=sharing

REFLECTION

Overall, the result of the project was pleasing and what we were looking for--a real time machine learning music generator. The ability for each unique song created to be unique, and dependent on the human interaction connects the user to the art in a way that other machine learning-art pieces fail to achieve. Many of these pieces that we have seen have generally lacked the human component that makes an art piece feel personal and sincere. Focusing on the human-computer interaction was a vital aspect of our concept pushing us to create a final result that forced a user to be a part of the art making process.

We did face some issues, however. The first issue is in interactive component of the music generation: while Sweet Dreams does produce music that sounds good, with some combinations of PCA scalings, it produces cacophonous music which sounds more experimental than like a lullaby. This issue could likely be solved by training the model for more epochs, as 400 is slightly too few; this was an issue of time and hardware. Our second issue is in the understanding of the model--while the PCA does lead to some insight into what the model is learning, such as playing more chords versus free notes or more high notes versus low notes, many of the PCA components were not able to be understood. This is a tougher issue to fix, as forcing the model to learn certain components is not feasible.

The result is a good stepping-stone for our group's final project, which will require a lot of interaction with machine learning algorithms in real time. The lullabies are interesting, but the next step is to push it to work with synthesizers and pads in order to make the music more ambient and less structured.

REFERENCE

https://github.com/HackerPoet/Composer

CODE
https://github.com/saffranzachary/aaml_project3/blob/master/README.md
https://github.com/chrismgeorge/CPPN